Multiplexing Traffic at the Entrance to Wide-Area Networks

Ramón Cáceres

Ph.D. Dissertation

Report No. UCB/CSD 92/717

Computer Science Division

University of California

Berkeley CA 94720

December 1992

ramon@cs.berkeley.edu

Multiplexing Traffic at the Entrance to Wide-Area Networks

A mis padres, Mon y Mirtha.

Abstract

Many application-level traffic streams, or conversations, are multiplexed at the points where local-area networks meet the wide-area portion of an internetwork. Multiplexing policies and mechanisms acting at these points should provide good performance to each conversation, allocate network resources fairly among conversations, and make efficient use of network resources.

In order to characterize wide-area network traffic, we have analyzed traces from four Internet sites. We identify characteristics common to all conversations of each major type of traffic, and find that these characteristics are stable across time and geographic site. Our results contradict many prevalent beliefs. For example, previous simulation models of wide-area traffic have assumed bulk transfers ranging from 80 Kilobytes to 2 Megabytes of data. In contrast, we find that up to 90% of all bulk transfers involve 10 Kilobytes or less. This and other findings may affect results of previous studies and should be taken into account in future models of wide-area traffic.

We derive from our traces a new workload model for driving simulations of wide-area internetworks. It generates traffic for individual conversations of each major type of traffic. The model accurately and efficiently reproduces behavior specific to each traffic type by sampling measured probability distributions through the inverse transform method. Our model is valid for network conditions other than those prevalent during the measurements because it samples only network-independent traffic characteristics. We also describe a new wide-area internetwork simulator that includes both our workload model and realistic models of network components.

We then present a simulation study of policies for multiplexing datagrams over virtual circuits at the entrance to wide-area networks. We compare schemes for mapping conversations to virtual circuits and queueing disciplines for scheduling datagrams onto virtual circuits. We find that networks should establish one virtual circuit per type of traffic flowing between two network points of presence, and provide round-robin service to transmission resources shared by virtual circuits. This multiplexing policy exhibits good performance and consumes moderate amounts of resources at the expense of some fairness among traffic sources of the same type. In particular, it maintains interactive delay nearly constant and close to the possible minimum, and maintains bulk transfer throughput near the possible maximum, even as network load increases beyond saturation. Furthermore, it results in bottleneck buffer consumption that rises slowly with offered load. Other multiplexing policies exhibit interactive delay that increases with offered load, and buffer consumption that rises quickly with offered load.

Again using our traffic characterization, we evaluate mechanisms for multiplexing variable-sized datagrams onto small fixed-size cells. Cells offer performance and implementation advantages to networks that service many types of traffic, but they incur bandwidth inefficiencies due to protocol headers and cell fragmentation. We find that cell-based networks using standard protocols are inefficient in carrying wide-area data traffic. For example, ATM-based networks using SMDS and IEEE 802.6 protocols lose more than 40% of their bandwidth to overhead at the network level and below. Furthermore, we find that viable compression techniques can significantly improve efficiency. For example, a combination of three compression techniques can regain more than 20% of the bandwidth previously lost to overhead.

Acknowledgements

I owe much to my teachers, colleagues, and friends, many of whom fit into more than one of these categories. I would like to specifically thank:

Domenico Ferrari, my research advisor, for giving me the freedom to define my own interests and the guidance necessary to pursue them to a successful conclusion. I am also very grateful for his timely review of this dissertation despite adverse conditions. John Ousterhout and Terry Speed, for serving on my qualifying exam and dissertation committees, and Mike Stonebraker, for chairing my qualifying exam committee. Their comments have substantially improved this work.

Sandy Fraser at Bell Labs, for supervising much of my work there and serving on my dissertation committee. His insight and experience helped guide my research in many ways. Sam Morgan, also at Bell Labs, for his consistent interest in and advice on my work. His thoroughness and enthusiasm have been an inspiration.

Peter Danzig, Deborah Estrin, Sugih Jamin, and Danny Mitzel at USC, for their collaboration on traffic measurement, analysis, and modeling, and their permission to include our joint work in parts of this dissertation. The many people who helped to gather traffic traces: Dave Presotto and Dennis Ritchie at Bell Labs, Dan V. Wilson at Bellcore, Cliff Frost, Balakrishnan Prabhakaran, and Brian Shiratsuki at Berkeley, and Mark Brown, John Conti, Dick Kaplan, and Jim Pepin at USC.

Mark Sullivan, for his support and criticism. Luis Felipe Cabrera, for his advice on technical and career plans. The members of the Tenet, Xunet, Progres, and Dash projects: Riccardo Gusella, Chuck Kalmanek, Srinivasan Keshav, Joann Ordille, Gary Murakami, Colin Parris, Vern Paxson, Stuart Sechrest, Shin-Yuan Tzou, Dinesh Verma, and others. They have been a sounding board for many ideas.

Rebecca Wright, for her companionship and encouragement. The Hillegass Boys: Will Evans, John Hartman, Steve Lucco, and Ken Shirriff, for making our house a home. Nina Amenta, Diane Hernek, Sandy Irani, Dan Jurafsky, Lu Pan, Dana Randall, Ronitt Rubinfeld, and again Will Evans and Mark Sullivan, for Three Winnebagos in the Desert, late-night violin recitals on Golden Gate Bridge, dim sum outings, and many other good things along the way. Marti Hirsch, for helping me through prelims. Randi Weinstein, for making available the notebook computer that made writing this dissertation a lot less painful than it otherwise would have been.

The staff of the CS Division, especially Kathryn Crabtree, for holding my hand and walking me through the bureaucracy. The staff at ICSI, especially Bryan Costales, for providing a productive place to work and a pool of superbly managed workstations on which to run parallel simulations.

Finally, I gratefully acknowledge financial assistance from AT&T Bell Laboratories, BP America, the Corporation for National Research Initiatives, Hitachi, Ltd., the International Computer Science Institute, and the State of California.

Table of Contents

1. Introduction

Lo, soul, seest thou not God’s purpose from the first?
The earth to be spann’d, connected by network,
The races, neighbors, to marry and be given in marriage,
The oceans to be cross’d, the distant brought near,
The lands to be welded together.

Walt Whitman, *Leaves of Grass*, 1892

Communication networks have changed the way people interact. The telegraph first allowed fast, reliable communication over long distances. The telephone network now links a significant portion of the world’s population. Data networks hold the promise of further improving human communication as we increasingly rely on access to widely dispersed information. However, many challenges remain before data networks can serve as many people as today’s telephone network.

This dissertation addresses problems that arise from the aggregation of traffic in large data networks. It characterizes sources of traffic and evaluates ways to mix, or *multiplex*, traffic from many different sources at the entrance to wide-area networks. It presents multiplexing policies and mechanisms that provide good performance and fair treatment to each source of traffic while making efficient use of network resources.

1.1. Context

Throughout this work we are concerned with wide-area internetworks of the type shown in Figure 1.1. *Host* computers communicate through the internetwork with other, geographically distant, hosts. Hosts are directly connected to a local-area network (LAN). Local-area networks are in turn connected to the wide-area portion of the network (WAN) through *router* systems. Routers communicate with other routers through the wide-area network. The wide-area network is itself composed of a collection of switching nodes and communication links (not shown in the figure). In general, data flows from a source host, over a LAN, through an input router, over the WAN, through an output router, over another LAN, and finally to a destination host.

Application programs running on the hosts are the ultimate sources and sinks of network traffic. Examples of *traditional* wide-area network applications are file transfers, electronic mail, network news, and remote terminal sessions. These traditional applications fall into two broad categories: bulk transfer (file transfers, electronic mail, and network news) and interactive (remote terminal sessions). Examples of *non-traditional* applications are voice, high-fidelity audio, and video.

We use the term *conversation* to mean the stream of traffic flowing between an application program on one host and an application program on another host. Conversations are of different *types*, corresponding to the different applications using the network. For example, we speak of file transfer conversations and electronic mail conversations. Each type may offer a different

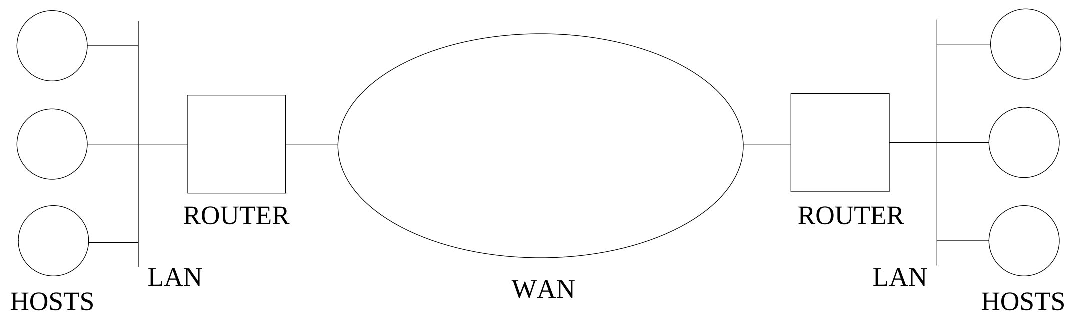

Figure 1.1. A Wide-Area Internetwork

workload to the network and demand a different level of performance from the network.

1.2. Traffic Multiplexing Objectives

Many conversations of different types come together at the entrance to wide-area networks. Our work aims to define policies and mechanisms for multiplexing traffic at a router that satisfy the following criteria: *performance*, *fairness,* and *efficiency*. We motivate these three objectives below.

First, each conversation should obtain the performance desired by the application that drives it. For example, bulk transfer applications transmit batch data such as files. They are concerned mostly with throughput, or how fast data is carried by the network. In contrast, interactive applications transmit interactive data such as keystrokes. They are concerned with delay, or the response time of the network. Within the limits of available resources, network control policies should provide high throughput to bulk transfer applications and low delay to interactive applications.

Second, the network should allocate its resources fairly among all conversations. In particular, it should provide low delay to interactive applications in spite of high offered loads from bulk transfer applications. It should not let high volumes of bulk data monopolize the network at the expense of low volumes of interactive data. In addition, all conversations of the same type should obtain equal performance.

Third, the network should make efficient use of its resources. In particular, it should not waste bandwidth or buffer space. High-speed wide-area networks include costly hardware components such as routers and switches that are made more expensive by their memory requirements. These networks also use expensive long-haul communication lines. Network control schemes with low buffer requirements are preferable over schemes with high buffer requirements. Similarly, protocols and multiplexing techniques that conserve bandwidth are preferable over those that waste bandwidth.

Our discussion is in the context of networks whose users are willing to pay for the level of performance desired by their applications, or networks that charge all users equally (including networks that do not charge). In networks where these conditions do not hold, the above three multiplexing objectives take on a slightly different form. First, a conversation should obtain the grade of service that has been mutually agreed upon by the network user and provider. A conversation may obtain a lower grade of service than that desired by the associated application. Second, the network should allocate its resources according to grade of service contracts. It may allocate more resources to conversations that have contracted for a higher grade of service than to those who have contracted for a lower grade, even if the underlying application is the same. Third, although efficiency remains an important consideration, an overriding goal is to minimize the investment required to provide the services contracted for. Inefficiencies may occur if they are unavoidable to insure a grade of service that users are willing to pay for.

1.3. Motivation

Our work is motivated by the failure of past networks to meet the stated multiplexing objectives and by trends in future networking.

We use two examples to illustrate the shortcomings of past networks. First, consider NSFnet [82]. NSFnet is a 1.5 Megabit/second, U.S. cross-country network that forms the backbone for the current Internet. Although its communication links have been gradually upgraded to 45 Megabit/second over the last two years, it continues to suffer from congestion due to the growth of the Internet. More to the point, in the face of this congestion, NSFnet gives poor performance to interactive applications – cross-country response times longer than 100-200 milliseconds are common, and much longer delays are not rare.† Studies have found that humans perceive interactive response to be ‘‘bad’’ when it takes longer than 100-200 milliseconds [90].

This behavior is a result of the networks's traffic multiplexing policies. NSFnet does not separate different conversations or types of traffic. It treats all traffic the same way in a first-in first-out basis. Short interactive packets compete for resources with much larger bulk transfer packets and thus incur high delays. It is often difficult for interactive users to work productively across NSFnet.

Second, consider Xunet 1 [96]. Xunet 1 was an experimental 1.5 Megabit/second, U.S. cross-country network that was dismantled in early 1991. It was based on Datakit [33] switches, which separated the traffic from different conversations and gave priority to short packets. This policy avoided interactive delay problems – cross-country response times remained well below 100 milliseconds even under congestion. However, Xunet 1 did not make efficient use of its bandwidth resources – the aggregate throughput of bulk transfer applications never reached the full bandwidth of the network.‡

This behavior was due to a combination of the network's multiplexing policies and its lack of buffer memory. The Datakit design dictates that, at each switch, each conversation be allocated an amount of buffer space equal to the network's *round-trip window*. The round-trip window is the network's inherent storage capacity, that is, the product of its bandwidth and round-trip propagation delay. Since congestion at a link can cause an entire window of data to temporarily accumulate in the buffer in front of the congested link, this much buffer space insures that the network does not drop data and senders can successfully utilize the full bandwidth of the network. However, even allowing for rapidly declining memory prices, the costs of implementing this design in high-speed wide-area networks can be prohibitive.

Table 1.1 shows the size of one round-trip window for a hypothetical continental network with different speeds. It assumes a 45-millisecond round-trip delay that approximates the speed-of-light propagation delay across 6,000 miles of glass fiber. This much memory is

† Measured between the University of California in Berkeley and AT&T Bell Laboratories in Murray Hill, New Jersey, on a typical weekday afternoon.

‡ Measured between the University of California at Berkeley and AT&T Bell Laboratories in Murray Hill, New Jersey, under various load conditions.

Link Speed (Megabit/second)	Window Size (Kilobytes)
1.5	8
45	253
150	844
600	3,375
1,700	9,562

Table 1.1. Round-trip windows for a network with a 45-millisecond round-trip time

necessary for each conversation flowing through each Datakit switch. Xunet 1 switches did not have enough buffer memory to support high throughput across long distances.

Five trends involving wide-area internetworks suggest that the problems just described will intensify. First, long-haul communication speeds are increasing. Second, local-area network speeds are increasing. Third, host speeds are increasing. Fourth, the number of traffic types is increasing. Fifth, the number of local-area networks and hosts connected to wide-area internetworks is increasing. For wide-area networks, these trends portend the following three conditions. First, there will be increasing demands for resources, both bandwidth and memory. Second, there will be a need for more grades of service, in particular delay and throughput. Third, there will continue to be congestion due to the aggregation of traffic.

We see a need for better traffic multiplexing policies and mechanisms. We aim to find techniques that avoid the shortcomings of past networks and accommodate the demands of future networks. In particular, we would like to prevent a stream of traffic from interfering with the performance of another stream, and to make efficient use of memory and bandwidth.

1.4. Scope of the Investigation

All wide-area networks multiplex data at their entrance. There are many types of networks carrying many different workloads. It would be impractical to explore every aspect of the multiplexing problem as described so far. Instead, we concentrate on an important class of network and workload, namely, *cell-based virtual circuit networks that carry traditional datagram traffic*. We define the scope of our investigation below.

1.4.1. Datagrams over Virtual Circuits

Datagrams and virtual circuits are widely used data communication styles. The two terms have been defined many ways. A useful analogy compares datagrams to the common postal service, and virtual circuits to ordinary telephone service [98]. Like letters flowing through the postal system, datagrams are individually addressed and delivered items of information. A datagram service delivers information with no guarantees regarding ordering or reliability. In contrast, like voice conversations in the telephone system, virtual circuits set up and tear down a path through the network before and after their use. A virtual circuit service delivers information in the same order it was sent, although not necessarily free of errors.

Throughout this work, we use the following restricted definition of virtual circuits. Virtual circuits imply only three things: a connection establishment and teardown procedure, a fixed path through the network for the lifetime of a virtual circuit, and a separate queue for each virtual circuit at each queueing point. We do not assume many other features that are sometimes associated with virtual circuits. For example, we do not assume any reservation of resources, either memory or bandwidth; all data is statistically multiplexed. We also do not assume any error recovery beyond checks for the integrity of protocol headers.

There is a long history of wide-area networks that transport datagrams over virtual circuits, and such networks continue to exist. In these networks, hosts outside the wide-area portion of the network exchange datagrams, while nodes inside and at the edge of the wide-area portion of the network communicate through virtual circuits. The Advanced Research Projects Agency Network (ARPANET) [82], built in the late 1960's and used until the late 1980's, used virtual circuits for communication between its internal nodes, or Interface Message Processors (IMPs). However, ARPANET hosts exchanged datagrams from the Internet Protocol (IP) [15]. More recent networks built in the 1980's carried IP datagrams over X.25 virtual circuits [65], for example the Defense Data Network (DDN) [24] and parts of the Computer Science Network (CSNET) [14].

There will also be future virtual-circuit-based networks that carry datagrams. Although there are competing opinions as to how networks will evolve, there is widespread agreement that future networks will be integrated. An integrated network is one that carries many different types of traffic – traditional data traffic as well as non-traditional traffic. A prominent design calls for Broadband Integrated Services Digital Networks (B-ISDN) based on Asynchronous Transfer Mode (ATM) virtual circuits [41] [72]. New traffic types are appearing because increasing communication speeds now allow networks to support applications with stringent bandwidth and delay requirements, for example audio and video. It is not yet clear what protocols these new applications will use. However, traditional traffic will continue to use well-established datagram protocols such as IP over the new ATM virtual circuits.

The reasons for this mix of styles are many. Datagrams are attractive for end-to-end internetwork applications, while virtual circuits offer advantages to providers of integrated-services wide-area networks. Some network applications like Remote Procedure Call (RPC) [8] are natural datagram applications. They exchange data in largely independent pieces that do not fit well in the virtual circuit mold. On the other hand, a large network based on virtual circuits is easier to manage than one based on datagrams. A virtual circuit is a convenient unit on which to carry out flow control and congestion control policies, allocate bandwidth and delay, and perform accounting. For example, it is easier for a virtual circuit network than for a datagram network to apply back-pressure along a path back to a misbehaving source, because each virtual circuit uniquely identifies a complete path from source to destination, while datagrams between the same source and destination may follow different paths.

1.4.2. Cell Networks

There will also be future cell-based networks, again due to the push for service integration. *Cells* are small, fixed-length units of data. For example, ATM networks use cells that are 53 bytes long. Before transporting application data units such as variable-sized datagrams, the network fragments them into cells. Cells are the basic multiplexing unit throughout the network. They allow fine control over bandwidth and delay allocation. Small cells provide lower delay than large datagrams because a short multiplexing unit quickly frees up resources for units from other conversations. No multiplexing unit is forced to wait for a long unit to complete transmission. The fixed-size nature of cells also allows efficient hardware implementations since the network always deals with homogeneous units of data. Thus, cells offer performance and implementation advantages to integrated networks.

1.4.3. Traditional Data Traffic

Our choice of workload is traditional data traffic, by which we mean traffic from existing wide-area network applications such as file transfers, electronic mail, network news, and remote terminal sessions. We chose this workload because we can accurately characterize it through direct measurements of current network traffic. Future networks will begin to carry audio and video traffic with characteristics different from current traffic. However, representative quantities of this traffic are not available for measurement. Furthermore, traditional forms of traffic will prevail for several years and will continue to be used for much longer. As a result, future networks, including B-ISDN networks based on ATM, must efficiently support traditional traffic.

1.5. Applicability of Results

Although we have narrowed the scope of our investigation, our results will be applicable to a wide range of networks. For example, our results could contribute to the design of future datagram-based networks as well as future reservation-based networks. We will return to the applicability issue at the end of the dissertation, after we have presented our work in detail.

1.6. Underlying Approach

We take an empirical approach to our work that is seldom used in wide-area network research. There is considerable research activity addressing the needs of future types of traffic, in particular real-time traffic like audio and video [44] [45]. These studies make educated guesses as to the characteristics of future traffic and use these best-guess models to make network design decisions. No other approach is possible with traffic that does not yet exist.

However, we can do much better with mature traffic types. File transfers and remote terminal sessions have been important applications of wide-area networks for close to 20 years [61]. We expect that many characteristics of these traffic types remain stable or can be parametrized to track changes in technology. Nevertheless, no realistic per-conversation models of wide-area traffic exist. We should be able to accurately characterize this traffic from observations of real networks, without relying on our intuition. We can use what we learn to tune existing networks and design new ones.

This dissertation exemplifies this empirical approach to network research. We measure real networks and identify the characteristics of their traffic. We find that many important characteristics are indeed stable. We then derive traffic models directly from the measurements. Finally, we use these models to study traffic multiplexing at the entrance to wide-area networks. Through simulation, we evaluate different multiplexing policies in terms of their performance, fairness, and efficient use of buffer memory. We also evaluate different network protocols and multiplexing techniques in terms of their efficient use of bandwidth.

1.7. Outline of the Dissertation

Chapters 2 through 5 constitute the main body of this dissertation. They present our work in the areas of traffic characterization, network simulation, multiplexing policies, and transmission efficiency, respectively. Each of these chapters also surveys previous work in the area it addresses. We outline these chapters below.

Chapter 2 characterizes the traffic flowing between the local-area and wide-area portions of a hierarchical internetwork. We have traced all traffic flowing in and out of two universities and two industrial research laboratories at the point their local networks connect to the rest of the Internet. The traces represent four different and geographically separate organizations, capture several days of continuous network activity at each site, and span one year and a half.

For each application responsible for a significant percentage of wide-area traffic, we identify characteristics common to all conversations of that type. We found that file transfers, network news, electronic mail, and remote terminal sessions account for the majority of packets, bytes, and conversations in current wide-area traffic. We present probability distributions for the number of bytes and packets per conversation, the duration of conversations, the time between packet arrivals, and the size of packets, among other statistics. These traffic characteristics contradict many common beliefs and shed light on many network research problems, including traffic multiplexing at the entrance to wide-area networks.

Chapter 3 describes VCSIM (Virtual Circuit Simulator), a new simulator of wide-area internetworks. VCSIM has two main aspects: its workload model and its network model. From the traffic characteristics of Chapter 2 we derive a new workload model for driving wide-area network simulations. It takes the form of a generative traffic model for individual application-level conversations of each major type of traffic. The model incorporates both detailed knowledge of application program behavior and the measured probability distributions just described, the latter sampled through the inverse transform method. Our workload model is network-independent, realistic, and efficient.

VCSIM also includes realistic models of wide-area internetwork components, including hosts, routers, switching elements, and communication lines. Hosts are sources and sinks of datagram traffic driven by the workload model just described. Of particular importance to our study, routers simulate a variety of traffic multiplexing policies. Routers fragment datagrams into cells and send the cells on virtual circuits through the wide-area portion of the network. Routers also reassemble packets from the component cells and pass the packets on to hosts. Switches transfer cells along virtual circuit paths through the network, and links carry packets and cells between the other components. VCSIM facilitates the study of multiplexing datagram traffic at the entrance to cell-based virtual circuit networks.

Chapter 4 turns to the problem of multiplexing datagrams over virtual circuits. The coexistence of datagrams and virtual circuits creates problems for multiplexing policies at the entrance to a wide-area network. Datagrams arrive at a router and must find a virtual circuit to go on. Routers must assign datagrams to virtual circuits and choose the order in which to transmit datagrams onto the wide-area portion of the network.

Through simulation, we assess the benefits and drawbacks of different multiplexing policies acting at a router, subject to our performance, fairness, and efficiency criteria. We evaluate three schemes for mapping a set of application-level conversations onto a possibly smaller set of virtual circuits: establishing one virtual circuit per conversation, establishing one virtual circuit per traffic type, and establishing one virtual circuit per destination router. We also evaluate two queueing disciplines for scheduling datagrams onto these virtual circuits: first-in first-out and round-robin. We find that the multiplexing objectives are best served when the network separates traffic types by giving each type its own virtual circuit. In addition, networks should provide round-robin service to transmission resources shared among virtual circuits.

Chapter 5 addresses another aspect of multiplexing wide-area data traffic over cell-based networks, namely transmission efficiency, or the ratio of useful bytes to total bytes carried by a network. Small fixed-size cells offer many advantages over variable-size datagrams in integrated-services networks like those based on ATM. However, cells incur bandwidth inefficiencies due to protocol headers and fragmentation. Again using our traffic characterization, we compare the efficiency of three ATM-related protocol combinations. We also calculate the efficiency effects of three non-standard compression techniques, and discuss how to implement these techniques cheaply.

We find that networks using standard ATM procedures are inefficient in carrying wide-area data traffic – depending on the protocols used, ATM-related overhead consumes 23 to 41% of the network bandwidth. Moreover, due to interaction between datagram lengths and cell padding, efficiency responds abruptly to changes in certain protocol parameters – for example, a 4-byte increase in cell size can yield a 10% increase in efficiency. Using one compression technique in isolation can improve efficiency by 12%, and simultaneously using three techniques can improve it by 31%. These issues should be considered when designing future networks.

Chapter 6 concludes this dissertation by summarizing our contributions, pointing out the applicability of our results, suggesting areas for future work, and commenting on the benefits of our methodology.

Appendices A through C present work that would detract from the readability of the dissertation's main body. Appendix A is a comparison of traffic characteristics from three sites at which traffic was measured. It shows that many per-conversation characteristics are common across all sites, allowing us to use one site as representative. Appendix B is a comparison of transmission efficiency results obtained with data from four sites at which traffic was measured. Due to the uniformity of traffic characteristics already noted, efficiency is also uniform across sites, allowing us to use one set of results as representative. Finally, Appendix C is a glossary of acronyms and abbreviations used throughout the dissertation.

2. Traffic Characterization

2.1. Introduction

Human intuition is often wrong when relied upon to characterize the behavior of complex systems. Large computer networks are some of the most complex systems ever built, with millions of independent components operating at minuscule time scales. A study of network performance needs an accurate workload model, that is, a model of the traffic injected into the network. Measurements of network traffic are valuable because they provide a snapshot of real workloads. This chapter presents a set of detailed traffic measurements of the Internet taken between 1989 and 1991. As we shall see, the measured characteristics contradict many common beliefs.

In order to drive our network simulations with realistic models of existing traffic sources, we gathered traces of real network traffic. We aim to investigate wide-area network issues, more specifically traffic multiplexing issues at the junction between local-area and wide-area networks. The traces used in this study were taken exclusively from such junction points in the current Internet. They therefore exhibit packet length, packet interarrival time, and other characteristics that accurately represent what current Internet routers are exposed to. These traces are more appropriate for our research than traces gathered in local-area networks or in other points of a wide-area network. In the next chapter, we derive from these traces generative models for individual sources of different types of traffic.

We chose to measure the Internet because it is the most successful example of a large internetwork and it is growing rapidly. To appreciate the size of the Internet, consider that more than 500,000 hosts throughout the world are registered in the Internet name domain system [66]. In addition, the Internet backbone, NSFnet, links over 2,300 university, industry, and government networks [71]. To this vast internetwork, large numbers of subnetworks are being attached on an almost daily basis: more than 720 new subnetworks joined the NSFnet backbone in the eighteen months between January, 1990, and June, 1991, and 38 subnetworks joined in the second week of June, 1991, alone [70].

When analyzing wide-area network traffic, we are interested in such questions as:

- How does traffic break down into interactive and bulk traffic?
- How ‘‘bulky’’ is the data transmitted by bulk transfer applications?
- What are the characteristics of conversations in terms of packets transferred, bytes transferred, duration, and packet interarrival time?
- Is traffic flow unidirectional or bidirectional?

The following summarizes our most important observations regarding wide-area data traffic.

- Interactive applications are responsible for 40% of wide-area packets, but less than 10% of wide-area bytes.
- Depending on the application, 60 to 90% of the conversations categorized as bulk transfers send less than 10 Kilobytes of data. Bulk transfer is request-response in nature, with responses larger than 1 Megabyte responsible for only 15 to 20% of all bytes transferred.
- Over 80% of interactive conversations send fewer than 1,000 packets and 10 Kilobytes, and 50% of interactive conversations last less than 200 seconds.
- A large portion of bulk transfer applications, which are responsible for more than 50% of observed network traffic, show bidirectional traffic flow (even after excluding transport-level acknowledgements and other control information).
- Interactive applications can generate 10 times more data in one direction than the other, and generate packet sizes ranging from the minimum to the maximum supported by the network.

The next section references previous work in the area of network traffic measurement, while the rest of this chapter discusses our own measurements in more detail. Section 2.3 presents the methodology used in our measurements. It describes the sites where we gathered our traces, what the traces contain, the instruments used to gather them, and their applicability beyond the time and place they were gathered. Section 2.4 outlines how we refined the raw traces to obtain meaningful traffic statistics. Section 2.5 presents characteristics of wide-area TCP conversations in terms of measured probability distributions of many important statistics. Section 2.6 concludes this chapter by pointing out the implications of our findings.

2.2. Previous Work

Researchers have long used traffic measurements to gain insight on network behavior. There have been many local-area and wide-area network measurement studies, for example those listed in the bibliographies by Mogul [73] and Pawlita [80]. Below, we survey those studies most relevant to our research. Our study was the first to separate wide-area traffic into user-level conversations and extract per-conversation statistics using the complete stream of packets belonging to a conversation.

In the late 1960's, Jackson and Stubbs [46] and Fuchs and Jackson [36] measured application-level characteristics of low-speed, half-duplex terminal traffic in a local-area environment. In the mid-1980's, Marshall and Morgan [69] measured similar characteristics in a Datakit local-area network that supported hosts accessed by terminals and diskless workstations over 9.6 Kilobit/second links. They identified distributions of interactive burst lengths and inter-burst intervals, and distributions of file transfer sizes and interarrival times. Later in this chapter, we draw interesting parallels between these early findings and our more recent measured characteristics of wide-area interactive conversations.

Gusella [38] analyzed diskless workstation traffic on a 10 Megabit/second Ethernet local-area network. He used a measurement methodology similar to ours, and went on to characterize different types of traffic sources, as we are doing. However, his study was limited to local-area networks. His traffic mix was thus dominated by network file system and window system traffic, unlike the wide-area traffic mix we present later in this chapter. In another example of measurement-based analysis, Jain and Routhier [49] derived a new model of traffic, the packet train, based on local-area network observations. Neither of these studies separated application-level conversations.

In a more recent effort, Leland and Wilson [63] reported on high time-resolution measurement and analysis of Ethernet traffic. Again, this was only a local-area network study that did not separate conversations, but the network monitor they constructed was also used to collect traces of wide-area Internet traffic. We used these wide-area traces in our research, as described in the next section.

The next section also describes three other traces of wide-area Internet traffic used in our research. In a preliminary study, we extracted from one of these traces packet length characteristics broken down by the type of application responsible for the traffic [11]. In a further study involving three traces, we separated packet streams of user-level conversations and reported per-conversation statistics [12].

Since these earlier measurements, there has been considerable activity in the measurement and analysis of wide-area network traffic. Heimlich [43] analyzed aggregate traffic on the NSFnet national backbone and found evidence to support the packet train model. Crowcroft and Wakeman [18] inspected all Internet traffic flowing between the United Kingdom and the United States. Paxson [81] measured traffic entering and leaving Lawrence Berkeley Laboratory in Berkeley, California. Schmidt and Campbell [86] traced traffic at three different points: a local-area network at the University of Illinois at Urbana-Champaign, where the local networks at that university meet the corresponding NSFnet regional network, and the nearest NSFnet backbone node.

The latter three studies identified beginnings and ends of user-level conversations, but did not record the intervening packets. Thus, they report overall characteristics of conversations such as duration and number of bytes transferred, and their findings corroborate ours. However, they do not extract many of the detailed characteristics we present later in this chapter.

2.3. Measurement Methodology

2.3.1. Measurement Sites

We would like to characterize the traffic flowing through the junction points between local-area and wide-area portions of the current Internet. For this purpose, we obtained traffic traces from two universities and two industrial research laboratories: the University of California in Berkeley UCB), the University of Southern California (USC) in Los Angeles, AT&T Bell Laboratories (Bell Labs, BL) in Murray Hill, New Jersey, and Bell Communications Research (Bellcore, BCR) in Morristown, New Jersey. We gathered packet headers from the gateway Ethernet carrying all traffic between the local networks at each site and the corresponding regional access network: the Bay Area Regional Research Network (BARRnet) for Berkeley, Los Nettos for USC, and the John Von Neumann Center Network (JVNCnet) for Bell Labs and Bellcore. We filtered out local and transit traffic but kept all IP datagrams flowing between each site and the rest of the world.

At Bell Labs, we inspected one full week day and one full weekend day of traffic during July, 1989. We collected a full week of traces for both Berkeley and Bellcore during the October-November, 1989, time frame. We repeated the exercise more than a year later in January, 1991, and captured several days of activity at USC. The Berkeley and USC data should be representative of wide-area network traffic at a major university campus, while the Bell Labs and

Bellcore data should play a similar role for large corporate sites.

Here we present the analysis of 24 hours of traces from each site. The analysis is for measurements that began at 10:30 on Tuesday, October 31, 1989, at Berkeley; 14:24 on Tuesday, January 22, 1991, at USC; 7:59 on Thursday, July 13, 1989, at Bell Labs; and 14:37 on Tuesday, October 10, 1989, at Bellcore.

2.3.2. Trace Format

The traces from Berkeley, USC, and Bellcore were saved on 8-millimeter helical scan magnetic tapes. Every network packet generated a trace record consisting of a time stamp and the first 56 bytes of raw Ethernet data. The time stamp records the arrival time of the packet at the tracing apparatus. The 56 bytes of data are enough to hold the Ethernet header, the IP header, and either a TCP header or a UDP header.† Figure 2.1 is a diagram of the protocol headers most commonly found in the Internet.

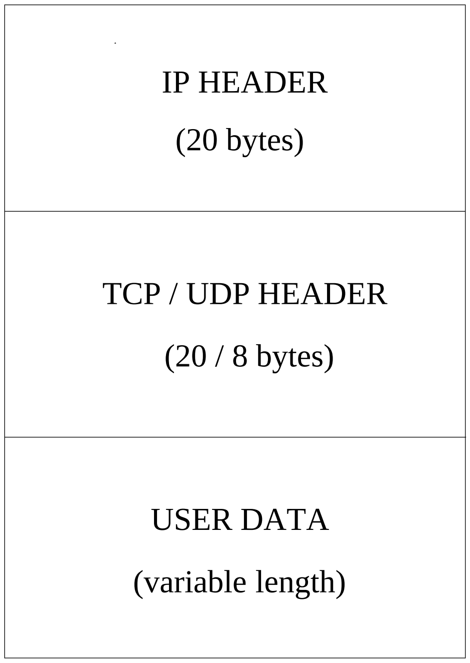

Figure 2.1. Internet Protocol Headers

Due to hardware limitations, the Bell Labs traces were not saved on tape. There was no tape drive available on the Bell Labs tracing machine. In addition, the hardware clock used at Bell Labs did not have a fine enough resolution to discern meaningful packet interarrival time statistics. For these reasons, packet counts were immediately broken down during tracing by application type and length. These statistics were written to disk and the raw trace data was discarded. We use the limited Bell Labs statistics to help validate the measurements from the other sites.

† We did not encounter any packets carrying IP or TCP options, and ignored packets containing IP fragments. IP fragments accounted for only 0.02% of all IP packets at Berkeley, 0.05% at USC, and 0.02% at Bellcore.

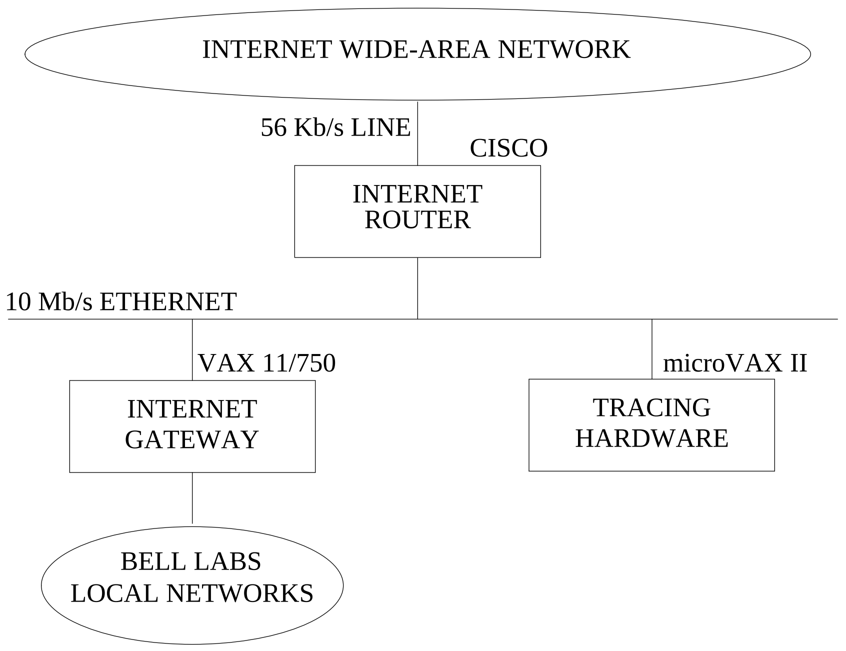

Figure 2.2. Sample Network Tracing Configuration

2.3.3. Tracing Instruments

The common approach to the four tracing efforts was to place a dedicated workstation-class computer on the Ethernet cable that carried all traffic flowing through the wide-area network routers at each site. For example, the tracing configuration used at Bell Labs is shown in Figure 2.2. The Ethernet controller for the tracing computer was placed in promiscuous mode, allowing it to passively inspect all the traffic on that Ethernet without disturbing the network, routers, or hosts.

There were two main goals when building our network tracing instruments:

- The traces should be complete, that is, they should contain every packet of interest. We wanted to insure that the bandwidth provided by the measurement system was sufficient to record all the incoming Ethernet packets, without dropping any packets. We found it effective to discard any unnecessary data from the Ethernet packet as soon as possible. Thus, we retained protocol headers but threw away application-level data. This technique lowered the required bandwidth to one that the 8-millimeter tape drives could maintain. Incidentally, discarding application-level data helped to safeguard its privacy, a concern at every measurement site.
- The traces should contain accurate and meaningful timestamps, that is, the time when packets arrive at the tracing instrument should be recorded promptly and with high resolution. We found it useful to read and record timestamps as soon as possible after interrupts signaled the arrival of an Ethernet packet at the tracing instrument. In addition, we added non-standard microsecond-resolution timers to some of the measurement systems in order to improve upon the millisecond-resolution timers originally provided with those systems.

Our tracing instruments met these goals with varying degrees of success, as described below.

The Berkeley traces were gathered using a Sun 3/50 workstation configured with a LANCE Ethernet controller, a SCSI disk drive, and a SCSI 8-millimeter tape drive. It was also equipped with a microsecond-resolution timer board designed at Berkeley and connected through the unused DES data encryption port [19]. The Ethernet driver in the SunOS kernel was modified to manage a circular buffer big enough to hold 3,000 trace records. Immediately after an interrupt was received signaling the arrival of an Ethernet packet, the system read the current value of the timer, truncated the Ethernet packet to 56 bytes, and appended the timestamp and truncated packet to the buffer. The resulting timestamp resolution was 10 microseconds. A dedicated user-level program inspected the buffer and wrote new trace records to tape. No packet losses due to buffer overflows occurred during the Berkeley measurements.

The USC data was collected using the NNStat program suite [9] on a Sun SparcServer 4/490. The NNStat package uses the standard SunOS timer, accessed through the *gettimeofday()* system call [16], which on that system has a 20-millisecond resolution. During tracing similar to that reported here, the loss rate was estimated by injecting a Poisson stream of short packets. Only 0.6% of these packets were missing from the tape.

Bell Labs traffic was inspected with a DEC microVAX II configured with a DEQNA Ethernet controller and the standard microVAX II disk. A new streams module was inserted in the Ninth Edition Unix kernel between the standard Ethernet driver and user space. Using the standard system clock, it prepended a 10-millisecond-resolution timestamp to each incoming Ethernet packet and forwarded the packet to user space. A user-level program read the time stamps and Ethernet packet data, extracted the statistics of interest, and wrote them to a set of disk files. No packet losses due to buffer overflows occurred during the Bell Labs measurements [11].

The Bellcore traces were collected with a Sun 3 workstation augmented with the following VME bus peripherals: a single-board computer with a Motorola MC68030 processor and an on-board LANCE Ethernet controller, an interval timer board with a 4-microsecond resolution, several hard disk drives, and two 8-millimeter tape drives. A system of hierarchical buffers stretching between the single-board computer and the tape drives carried the trace records to tape. No packet losses occurred during the Bellcore measurements [63].

The hardware used to gather trace data was different at each site because the three measurement sites were geographically and administratively disjoint. In addition to resulting in multiple software development efforts to produce the tracing software, this heterogeneity made our analysis more difficult. For example, the timestamps for Berkeley and Bellcore have significantly different formats, as well as different resolutions. On the other hand, many important similarities found among the various sets of data have served to validate the different measurements.

2.3.4. Applicability of the Traces

The success of the Internet and its family of protocols suggests they will continue to be a significant portion of the traditional data traffic carried by wide-area networks. Therefore, it is important to characterize the traffic generated by applications that use IP and related Internet protocols. We observed that more than 94% of wide-area Internet traffic is due to TCP and UDP. Future networks will carry audio and video traffic with different characteristics from current traffic. However, traditional forms of traffic will prevail for several years and will continue to be used for much longer.

Given that traditional wide-area data traffic will be an important application of future networks, we believe our traces are representative of such traffic. A look at aggregate statistics for the NSFnet backbone [70] lends weight to this claim. The breakdown of traffic in our traces shows strong similarities to the breakdown of NSFnet backbone traffic during October, 1989, when our Berkeley and Bellcore traces were gathered, and January, 1991, when our USC traces were gathered.† Most importantly, the seven applications we identified as transmitting the most packets also appear as such on the NSFnet backbone (these applications are TELNET, RLOGIN, FTP, SMTP, NNTP, DOMAIN, and VMNET, as shown later in this chapter).

NSFnet statistics for the total packet activity generated by organizational subnetworks also suggest that our traces are representative. Of 1,174 subnetworks for which backbone activity was detected during October, 1989, Berkeley was the 3rd busiest and Bellcore was the 130th busiest. Of 2,345 subnetworks measured during January 1991, USC was the 31st busiest. Thus, our traces capture activity generated by a range of subnetworks, from a very active one like Berkeley, to a moderately active one like USC, to a less active but non-trivial one like Bellcore.

The uniformity of our trace data further justifies its use as representative of wide-area Internet traffic. Although our data comes from four different organizations and spans a period of one year and a half, traffic from all four sites shares many characteristics. In particular, the distributions of number of bytes transferred, conversation durations, total packets per conversation, and packet sizes are indistinguishable. Furthermore, these characteristics are shared by two different days of Berkeley traces, and by a one-day trace and a three-day trace of Bellcore traffic.

For legibility, we present mainly data derived from the Berkeley traces in the body of this dissertation. The Berkeley traces are arguably the best-quality traces in our set. They represent the largest subnetwork studied, suffer from negligible packet losses, and contain very high-resolution timestamps. For completeness, Appendix A contains representative traffic characteristics comparing data from Berkeley, USC, and Bellcore. It shows that application-level conversation characteristics are uniform across all our trace data.

2.4. Analysis Methodology

2.4.1. Data Reduction

We need to refine the raw trace data in order to isolate the behavior of individual traffic endpoints. Types of endpoints include transport-level protocols, network applications, and application-level conversations. We would like to maintain separate statistics for these different endpoints, and not aggregate statistics for the all the traffic in the traces, in order to produce traffic models with sufficient detail to drive meaningful simulations.

The volume of traffic flowing through a local-area to wide-area network junction in the Internet is large. More specifically, the Bell Labs router sees approximately half a million packets a day, the Bellcore router sees approximately three quarters of a million packets a day, the USC routers see approximately five million packets a day, and the Berkeley routers see more than eight million packets a day. Capturing this traffic produces Gigabytes of data on a daily basis, even when only packet headers and time stamps are retained. Managing this much data presents a considerable challenge.

† Bell Labs network activity is not listed in the NSFnet statistics database.

We have used a mixture of software tools in the Unix programming environment [56] to meet this challenge. We wrote a collection of programs in various languages: the programming language *C*; file and data manipulation languages such as the *Bourne shell*, *awk*, and *make*; and the graph typesetting language *grap* [1] [2] [6][55]. These programs work together to reduce the raw trace data into meaningful and manageable sets of traffic statistics.

2.4.2. Transport-Level Protocols

The first step in refining the data was to separate the different transport-level protocols that use IP (Internet Protocol). We did this by inspecting the protocol type field in the IP header. Based on this field, we gathered separate statistics for TCP (Transmission Control Protocol) and UDP (User Datagram Protocol). Tables 2.1 and 2.2 show the number of packets and bytes attributed to different transport-level protocols at each site.†

Protocol	Site							
	UCB		USC		BL		BCR	
	#	%	#	%	#	%	#	%
TCP	5,891,621	83.3	5,094,504	97.5	393,305	78.7	1,466,297	88.5
UDP	1,046,027	14.8	52,068	1.0	104,799	20.9	103,355	6.3
other	135,406	1.9	77,696	1.5	1,896	0.4	86,587	5.2

Table 2.1. Packet counts from each site
(These figures reflect all packets captured in the traces, including packets carrying only acknowledgements and other protocol control information.)

Protocol	Site							
	UCB		USC		BL		BCR	
	#	%	#	%	#	%	#	%
TCP	504,969,623	88.9	662,737,646	99.3	50,070,497	93.7	154,093,992	96.5
UDP	63,033,070	11.1	4,351,976	0.7	3,362,217	6.3	5,532,723	3.5

Table 2.2. Byte counts from each site
(These figures reflect only application-level data bytes. They do not include protocol header bytes.)

2.4.3. Network Applications

The next step was to identify the different network applications responsible for wide-area Internet traffic. The application defines the type of traffic, and is thus particularly important for our study of multiplexing different traffic types at the entrance to a wide-area network. Network applications were identified by a heuristic that inspects the source and destination port numbers

† Byte counts for protocols other than TCP and UDP are not available.

found in TCP and UDP headers and records the smaller of the two. This heuristic relies on the convention that major Internet applications rendezvous with their clients by means of well-known port numbers, which are typically the smaller of the two port numbers used in a conversation. Major TCP network applications include FTP (File Transfer Protocol), NNTP (Network News Transfer Protocol), SMTP (Simple Mail Transfer Protocol), TELNET and RLOGIN (both remote terminal protocols). Major UDP network applications include DOMAIN (name service) and ROUTE (routing service).

Table 2.3 is a breakdown by application of the TCP traffic at Berkeley, USC, and Bellcore.† It shows the percentages of bytes, packets, and conversations attributed to each application, except when percentages are zero or near zero, in which case they are represented by a dash. The acronyms for these applications are expanded in Appendix C. Two related studies, one at University College London [18] and the other at Lawrence Berkeley Laboratory [81], as well as statistics from the NSFnet national backbone [70], identify a similar application breakdown in their wide-area TCP traffic.

Application	% Packets			% Bytes			% Conversations		
	UCB	USC	BCR	UCB	USC	BCR	UCB	USC	BCR
TELNET	28.0	16.6	36.3	5.5	2.3	6.5	4.1	8.7	8.7
RLOGIN	15.5	5.8	18.5	2.8	0.7	3.1	2.1	2.7	4.3
FTP	12.0	5.0	18.7	36.2	10.6	54.9	2.8	3.2	4.8
NNTP	11.6	36.3	9.2	15.8	44.5	15.6	0.4	1.6	0.8
SMTP	11.6	3.1	12.6	11.0	1.9	10.6	69.6	52.3	68.0
FINGER	1.1	0.4	0.5	0.6	0.2	0.2	18.3	17.8	7.6
RCP	0.2	3.6	1.4	0.4	12.5	4.3	0.2	0.2	0.7
UUCP	0.2	0.1	0.8	0.4	0.1	1.3	0.3	1.1	2.2
X11	0.2	5.0	0.4	0.2	2.5	0.1	—	0.5	0.4
VMNET	10.0	9.1	—	25.4	20.7	—	0.1	3.3	—
DOMAIN	0.1	0.1	—	—	0.2	—	0.2	3.3	0.1
IRCD	4.6	—	—	1.3	—	—	0.6	0.2	—
DC 10	—	3.5	—	—	0.8	—	—	1.5	—
other	4.9	11.3	1.6	0.4	3.1	3.1	1.3	3.6	2.4

Table 2.3. Breakdown of wide-area TCP traffic by application
(See Appendix C for a glossary of acronyms and abbreviations.)

2.4.4. Application-Level Conversations

The final data reduction step was to sort the overall packet stream into application-level conversations. This step is more involved than the ones discussed so far because it cannot be performed by independently inspecting each packet. We must detect the beginning of an application-level conversation, record all subsequent packets for that conversation in a separate set of statistics, and detect the end of a the conversation.

† An equivalent traffic breakdown by application is not available for Bell Labs.

2.4.4.1. Separating Conversations

We wrote a traffic pattern analyzer to produce histograms of per-conversation characteristics. We define a conversation to be a stream of packets traveling between the end points of an association, delimited by silences of at least twenty minutes. An association is in turn defined as a tuple of the form { *application, source host, source port, destination host, destination port* }. We explain these tuple components below.

The analyzer separates conversations in three steps. First, it identifies the application responsible for the conversation via the heuristic described earlier. Second, it identifies hosts by inspecting the source and destination addresses in the IP headers. Each host on the Internet is uniquely identified by a 32-bit address. The network uses these host addresses to route IP packets from their source to their destination. Third, the analyzer looks at the source and destination port numbers found in TCP and UDP headers. These port numbers are the same numbers used above for differentiating applications, but in this case we kept both numbers instead of distilling the well-known one into the name of an application. The combination of source and destination host addresses together with source and destination port numbers uniquely identify a pair of processes communicating over the Internet. A hashing scheme based on these tuples served to index the statistics for each conversation.

2.4.4.2. Using the Packet Train Model

Our use of silence periods to delimit conversations follows the common packet-train model of network traffic [43] [49]. This model has recently been used in place of earlier Markov models of data traffic [38] [42]. In the packet-train model, a stream of packets is broken up into trains. Two consecutive trains are delimited by a maximum allowable inter-car gap (MAIG). The MAIG is usually chosen to encompass 90 percent of all packet interarrival gaps. Different researchers have used different MAIGs, ranging from 500 milliseconds to 50 seconds, depending on the network measured. In our case, conversations are analogous to trains delimited by twenty-minute gaps. We chose this period because it is longer than FTP's idle-connection timeout value of fifteen minutes. Early on we experimented with a five-minute silence rule. The difference in results was minimal.

For several reasons, we decided to define our conversations in terms of packet trains and not in terms of the more obvious transport-level protocol connections. Network applications use transport-level protocols such as TCP to carry data over the network. The endpoints of these protocols exchange explicit control messages to establish and tear down transport-level connections. We could have detected these messages and used them to determine the beginning and end of a conversation. However, when modeling traffic sources for the purpose of network design, it is the active periods that matter. For example, the fact that a transport-level protocol connection for a remote terminal session is open for 8 hours is irrelevant if the session is inactive and produces no network traffic for 7 of those 8 hours. We are most interested in what happens when traffic is actually flowing between application-level endpoints. We are also interested in the lengths of idle periods, but it is not necessary to consider those periods as part of a conversation.

2.4.4.3. Grouping Multiple Transport-Level Connections

Another reason for ignoring transport-level connections is that in many cases an application uses multiple connections within one logical session. We would like our conversation statistics to capture the behavior of these logical sessions regardless of how many transport-level connections are used. Below we outline those cases in which our traffic analyzer groups multiple transport-level connections into one application-level conversation.

FTP conversations can subsume multiple TCP connections. Each FTP session initiates one FTP-control and zero or more FTP-data connections, corresponding to the zero or more files transferred during the session. We clumped these TCP connections into one conversation. We also clumped back-to-back and concurrent FTP sessions between the same host pair. Similarly, for NNTP, we clumped back-to-back and concurrent TCP connections between the same host pair.

However, while FTP always uses a control connection to send control packets and a separate data connection for each file transferred, NNTP sometimes uses the same connection for both handshaking and transferring multiple news articles. NNTP sets up a new connection for each article sent to the initiator of the conversation, but does not do so for each article sent in the other direction. Since we wanted the distribution of the number of articles sent per NNTP conversation and also the distribution of article sizes, we needed to distinguish articles from handshake traffic and also articles from each other. From the NNTP documentation and by sampling articles posted to the Internet, we observed that articles have a minimum size of 250 bytes. Since NNTP tries to send articles using the largest allowable packet size, we used the 250-byte minimum size to detect the start of an article transfer. We further noticed that for a given burst of large-sized packets, the last one of the burst usually has its TCP *push flag* set, while those in the middle of the burst do not.† Thus, we used the first packet with its push flag set to mark the end of an article. Packets smaller than 250 bytes between the end of an article and the beginning of the next article we considered handshake packets.

Our decision to group multiple transport-level connections into one conversation affects some of the traffic characteristics that we present later in this chapter, but it does not invalidate our results. In particular, our measured distributions of the number of packets and bytes per bulk transfer conversation are skewed towards bigger conversations, because some of our conversations include the packets and bytes of several transport-level connections. In spite of this bias towards larger conversations, we report conversation sizes that are much smaller than what has been traditionally assumed. Therefore, our claims regarding conversation sizes are conservative rather than excessive.

2.4.4.4. Ignoring Transport-Level Protocol Details

Since we want to model the characteristics of application-level traffic in general, independent of transport-level protocols like TCP, we further decided to drop all TCP-specific traffic. We dropped TCP connection establishment packets and all zero-byte packets, assuming that these were acknowledgement packets. We also filtered out all retransmitted packets. Retransmitted packets were detected by matching their sequence numbers against those of the last 128 packets from the same conversation. Most retransmitted packets match one recently transmitted within the previous 64 packets. The oldest retransmitted packet detected in the traces was 104 packet-sequence numbers in the past. Since we threw away retransmissions, we also threw away most of the TCP *keep-alive packets*, which share a single sequence number.‡

† The push flag is a control bit in TCP headers often used to delimit application-level data units.

‡ Keep-alive packets are used by TCP implementations to signal that a connection, although idle, should remain open.

This also meant that we would occasionally interpret a single keep-alive packet as a full-fledged conversation, albeit one that transfers only one packet. We filtered out all such false conversations in our analysis.

For the Bellcore trace, we further noticed that 50% of all NNTP conversations between Bellcore and Rutgers University consisted of a single 6-data-byte packet. After closer examination, we attributed those conversations to an implementation fault at either Bellcore or Rutgers. Our traffic pattern analyzer filtered out all such conversations.

In the next section we present the measured probability distributions produced by our traffic pattern analyzer. For all the reasons just described, we believe these per-conversation statistics capture those characteristics relevant to the study of traffic multiplexing at the entrance to a wide-area network.

2.5. Conversation Characteristics

We now detail properties of network conversations from the five applications responsible for 78% of wide-area TCP packets, 71% of wide-area TCP bytes, and 79% of wide-area TCP conversations: FTP, NNTP, SMTP, TELNET, and RLOGIN. The observations below are presented under five general categories: ratio of bulk to interactive traffic, characteristics of traffic from bulk transfer applications, characteristics of traffic from interactive applications, and direction of traffic flow.

2.5.1. Ratio of Bulk to Interactive Traffic

For lack of a more accurate workload model, previous wide-area network studies that simulate flow control, congestion control, multiple access protocols, and traffic dynamics in general have been forced to assume a rather simple traffic model [25] [32] [99] [100] [101] [102]. These studies use either exclusively bulk transfers or an arbitrary mix of bulk and interactive traffic. The percentage of packets attributed to interactive applications typically range from 0 to 20%.

We find that TCP traffic consists of bulk and interactive traffic, as commonly assumed. In particular, the top five applications cleanly break down into the two categories. FTP, NNTP, and SMTP are bulk transfer applications. They send a number of data items as quickly as possible. For FTP these items are files, for NNTP they are news articles, and for SMTP they are mail messages. The data has been prepared previously and people are not directly involved in the data transfer. In contrast, TELNET and RLOGIN are interactive applications. A person directly generates data for at least one side of a conversation, typically keystrokes that form commands for a remote computer.

However, our measurements do not agree with common assumptions regarding the ratio of bulk to interactive traffic. Although bulk applications send more data than interactive ones, Table 2.3 shows that interactive conversations send 25 to 45% of wide-area Internet packets and 5 to 10% of wide-area Internet bytes. We note that an earlier study [69] also found that interactive traffic accounted for roughly 40% of the bytes in their local-area network.

We think it is important to realize that interactive applications are responsible for 25 to 45% of all network packets. Simulations that model internetwork traffic as mostly large bulk transfers may overestimate the benefit of mechanisms proposed to improve bulk transfer performance.

2.5.2. Bulk Transfer Applications

2.5.2.1. Abundance of Small Transfers

Many simulation studies commonly overestimate the amount of data sent by bulk transfer applications such as FTP. Simulated transfer sizes usually range from 80 Kilobytes to 2 Megabytes, or the transfer simply lasts until the end of the simulation run [25] [32] [99] [101] [102]. In contrast, Figure 2.3 shows that 60 to 90% of bulk transfer conversations involve less than 10 Kilobytes. Our data also shows that the few conversations that transfer more than one Megabyte of data are responsible for 40 to 50% of all bytes transferred. However, as explained previously, bulk transfer conversations often transfer more than one data item per conversation. Individual items larger than one Megabyte make up only 15 to 20% of all bytes transferred.

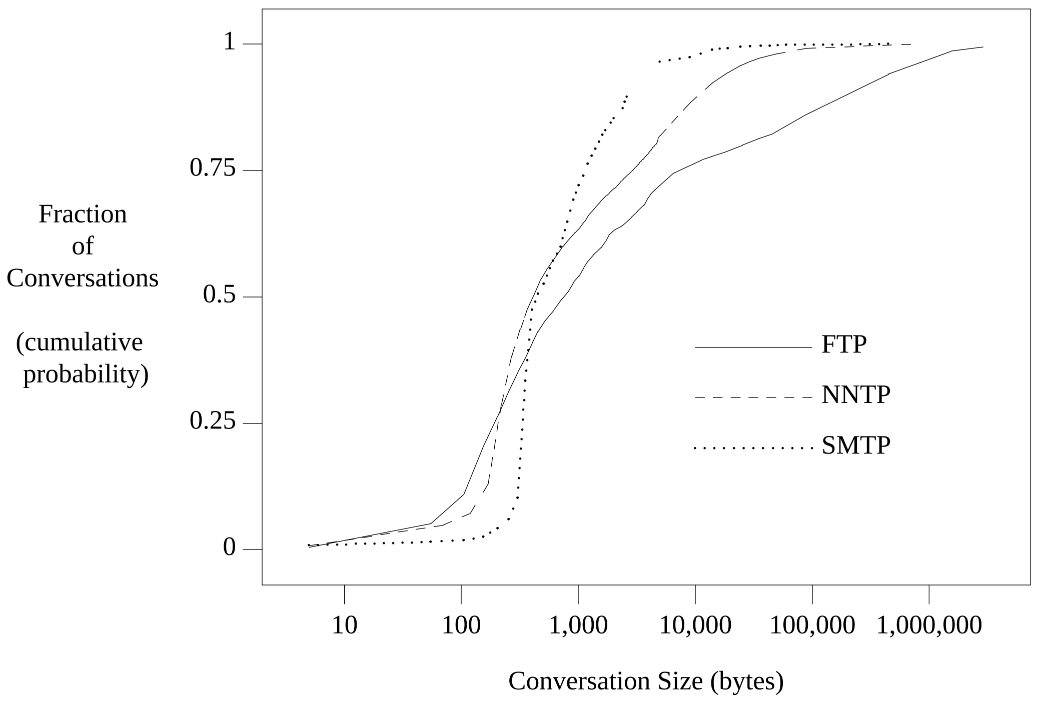

Figure 2.3. Bytes per conversation of bulk transfer applications

Our findings again agree with an earlier local-area network measurement study [69] that found a mean file transfer size of only 20 Kilobytes. These statistics also agree with observations made in file system studies [5] [78] that most files are small, but that a few large files account for a disproportionate amount of file space.

If small transfers are representative of Internet traffic, they should be taken into account in wide-area network simulations. To the extent that simulated algorithms employ feedback mechanisms, such as many used for congestion and flow control [47] [59] [83], it is important to know that in many conversations the data transfer may complete before any such feedback is received. The ineffectiveness of feedback will become increasingly acute as networks become faster.

2.5.2.2. Pauses Between Small Transfers

Bulk transfer applications wait at least one network round-trip time between the transfer of two data items. The pause is due to an exchange of application-level control information, or handshake, between the endpoints of a conversation. Bulk applications like NNTP, which appear to exchange large amounts of data, thus make hundreds of small exchanges separated by at least one round-trip time. Wide-area network simulations should take into account these pauses in data transmission during bulk transfers.

2.5.2.3. Dependence on Network Conditions

We have noted the abundance of small 'bulk' transfers due to the small sizes of files, news articles, and mail messages handled by the respective applications. The sizes of these transfers thus reflect true application-level behavior that should be included in traffic models.

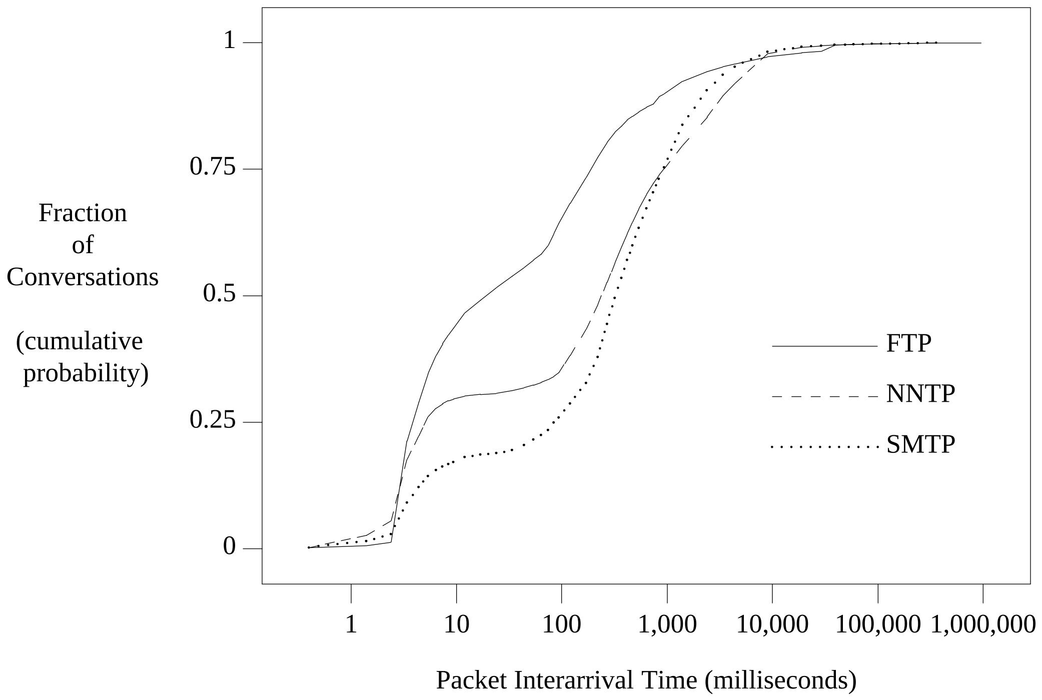

Figure 2.4. Packet interarrival times of bulk transfer applications

However, many other characteristics of bulk transfer conversations do not reflect application-level behavior, but instead depend on prevailing network conditions. For example, the packet interarrival times of bulk transfer conversations are to a great extent determined by the line speed and degree of congestion of a network. This dependence is apparent in Figure 2.4, where the minimum interarrival time for FTP (approximately 3 milliseconds) is equal to the quotient of the dominant packet size for FTP (512 bytes in Figure 2.5) and the line speed of the Internet backbone during our measurements (1.5 Megabit/second). The abundance of interarrivals times near the minimum corresponds to the arrival of back-to-back packets and exemplifies classic packet train behavior. This behavior follows from the attempt by bulk transfer applications to send data items as fast as the internetwork allows.

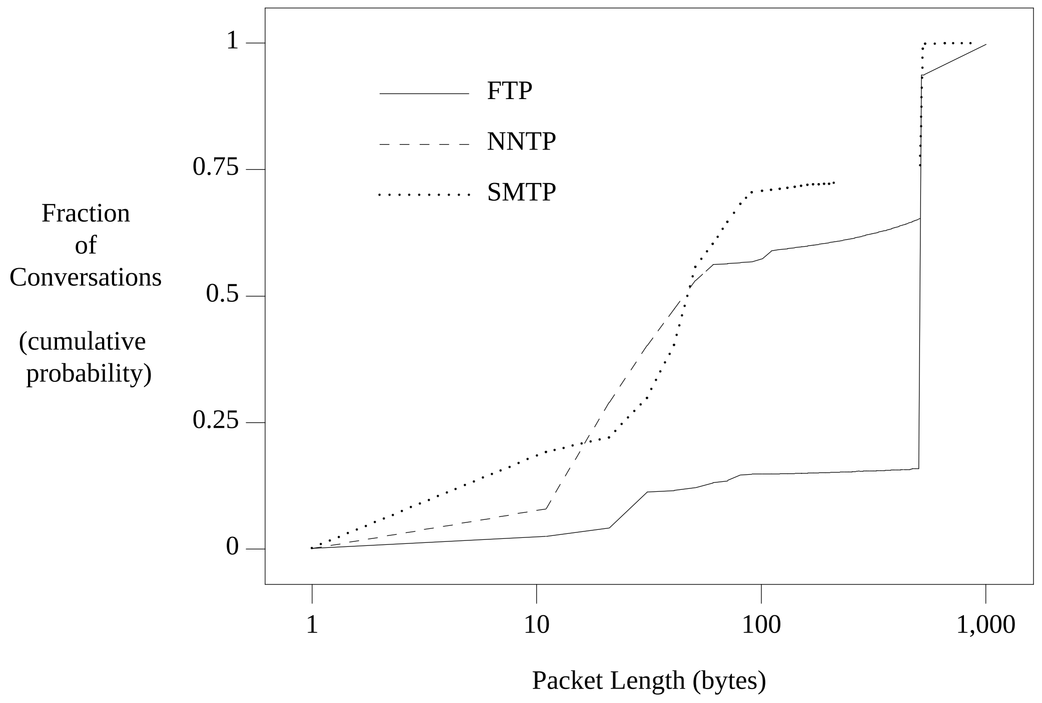

Figure 2.5. Packet lengths of bulk transfer applications

In addition, packet size characteristics themselves are a function of the maximum packet size allowed by a network, or its Maximum Transfer Unit (MTU). This dependence is apparent in Figure 2.5, where the dominant FTP packet size (512 bytes) is equal to the historical MTU of the Internet (also 512 bytes).

Therefore, many of the statistics presented here should not be relied on to model bulk transfer applications. Workload models should be network-independent in order to be widely useful. We address this issue further when we describe our simulation workload model in Chapter 3.

2.5.3. Interactive Applications

2.5.3.1. Independence from Network Conditions

Interactive applications are much less affected by network conditions than bulk transfer applications. For example, the packet sizes and interarrival times of interactive conversations are not tied to the line speed. Figures 2.6 and 2.7 show that more than 80% of TELNET and RLOGIN packets carry less than 10 bytes of data and have interarrival times greater than 100 milliseconds, respectively. These numbers are not limited by either the line speed or the MTU of the Internet at the time of our measurements. These measured characteristics thus reflect the true nature of those applications. We are free to use them to form a network-independent model of interactive conversations. We derive such a model for simulation purposes in Chapter 3.

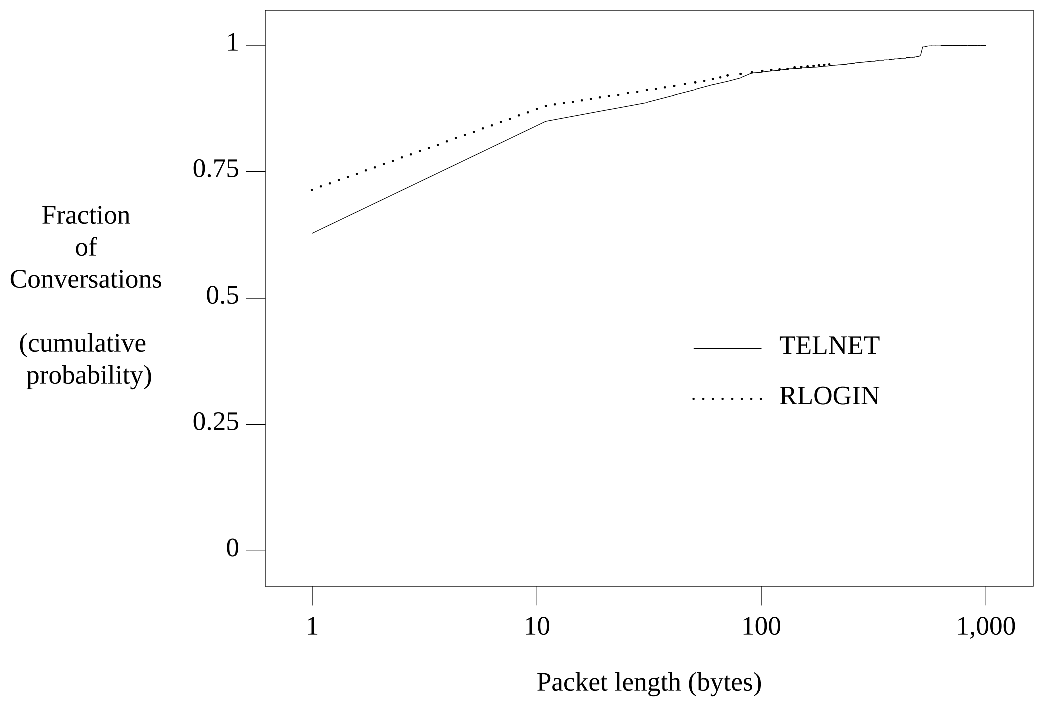

Figure 2.6. Packet lengths of interactive applications

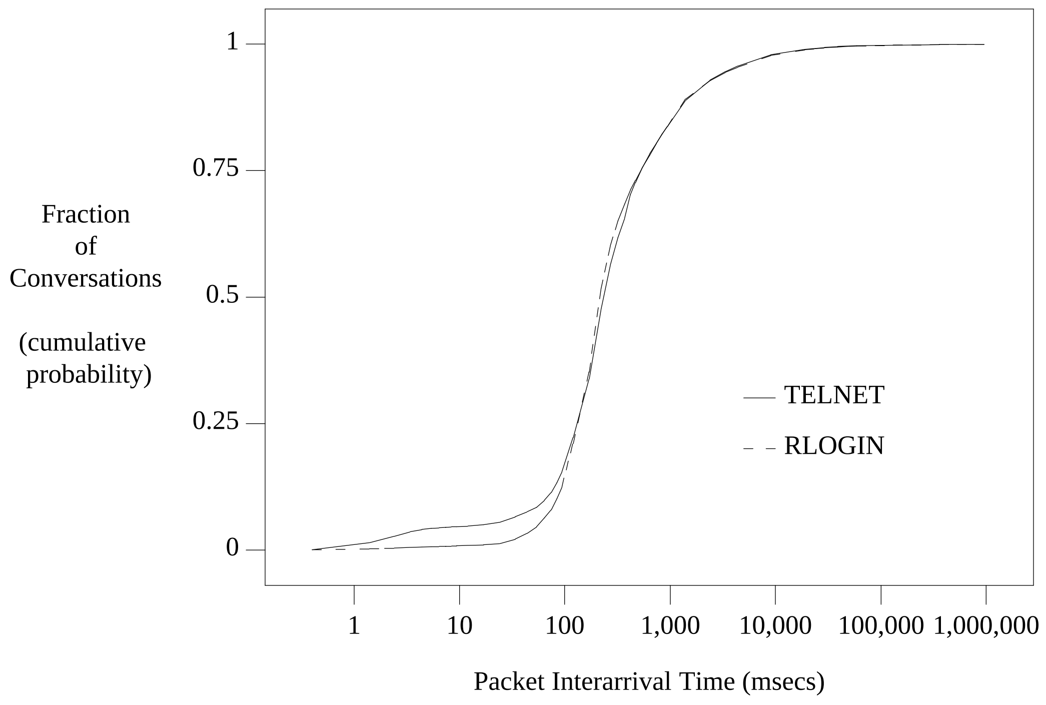

Figure 2.7. Packet interarrival times of interactive applications

2.5.3.2. Abundance of Short Conversations

In most traffic models used by previous simulations, interactive conversations are assumed to last 500 seconds, 600 seconds, or simply for the remainder of the simulation [25] [32] [67] [99] [102]. In reality, interactive conversations are much shorter. Figure 2.8 shows that 50% of TELNET and RLOGIN conversations last 200 seconds or less. These short conversations correspond to short periods of activity in what may be much longer but often inactive idle sessions. Granted, our definition of a conversation (active periods delimited by long silences) biases our observed conversation lifetimes. However, we feel our definition captures the most important aspects of traffic behavior (active periods as opposed to idle periods). Models of interactive traffic should reproduce this abundance of short conversations.

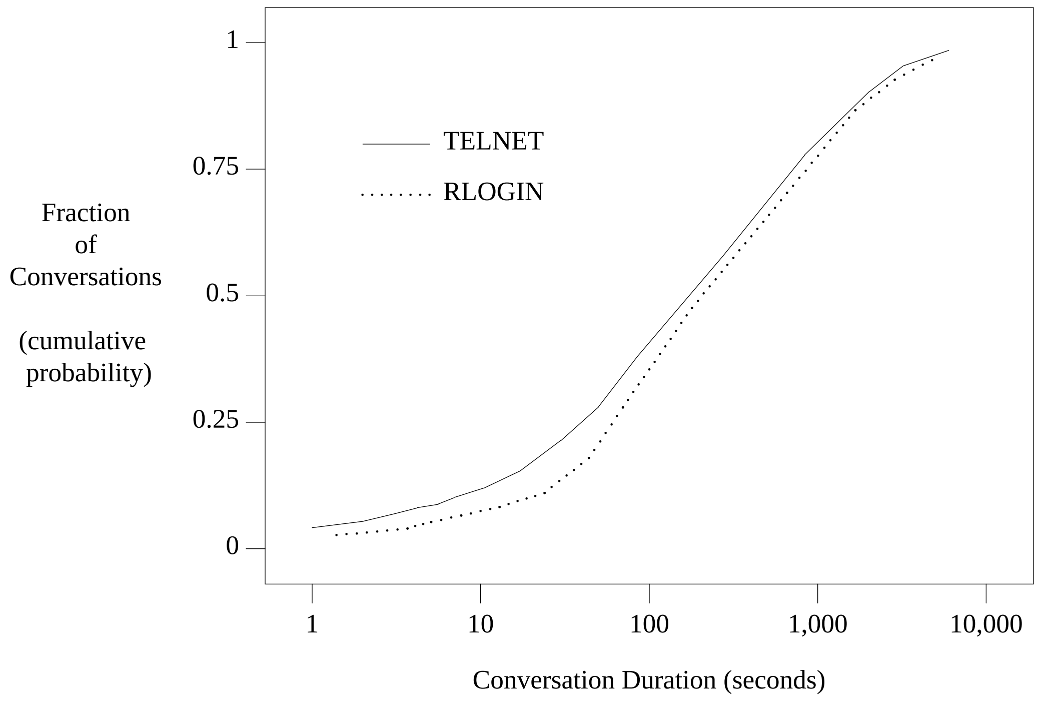

Figure 2.8. Seconds per conversation of interactive applications

Regarding the amount of data transferred during these short conversations, Figures 2.9 and 2.10 show that more than 80% of TELNET and RLOGIN conversations involve less than 1,000 packets and 10 Kilobytes, respectively. However, we should not assume that interactive conversations are responsible for less network data than bulk transfer conversations – a comparison of Figures 2.3 and 2.9 shows that 80% of all interactive conversations send as much data as the average bulk transfer conversation. Rather, we should keep in mind that bulk transfer applications send less data than is often assumed.

2.5.3.3. Absence of Packet Trains

Our data also shows that while interarrival times for bulk transfers exhibit packet-train behavior, including many back-to-back packets, interarrival times for interactive applications do not. They should be modeled by the distribution shown in Figure 2.10.

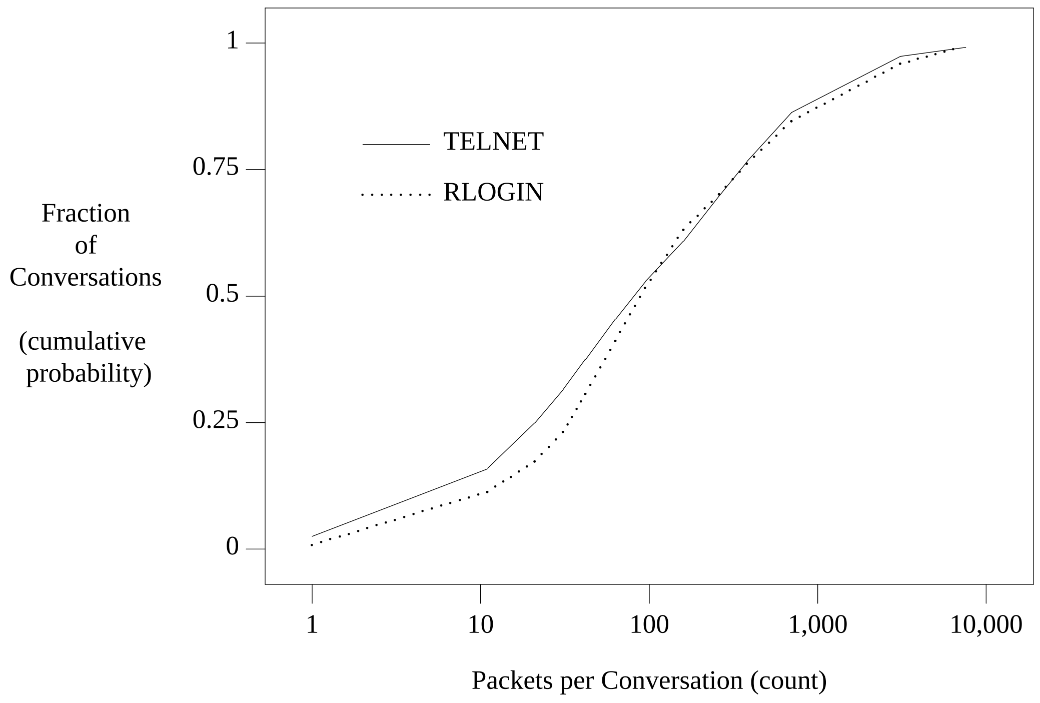

Figure 2.9. Packets per conversation of interactive applications

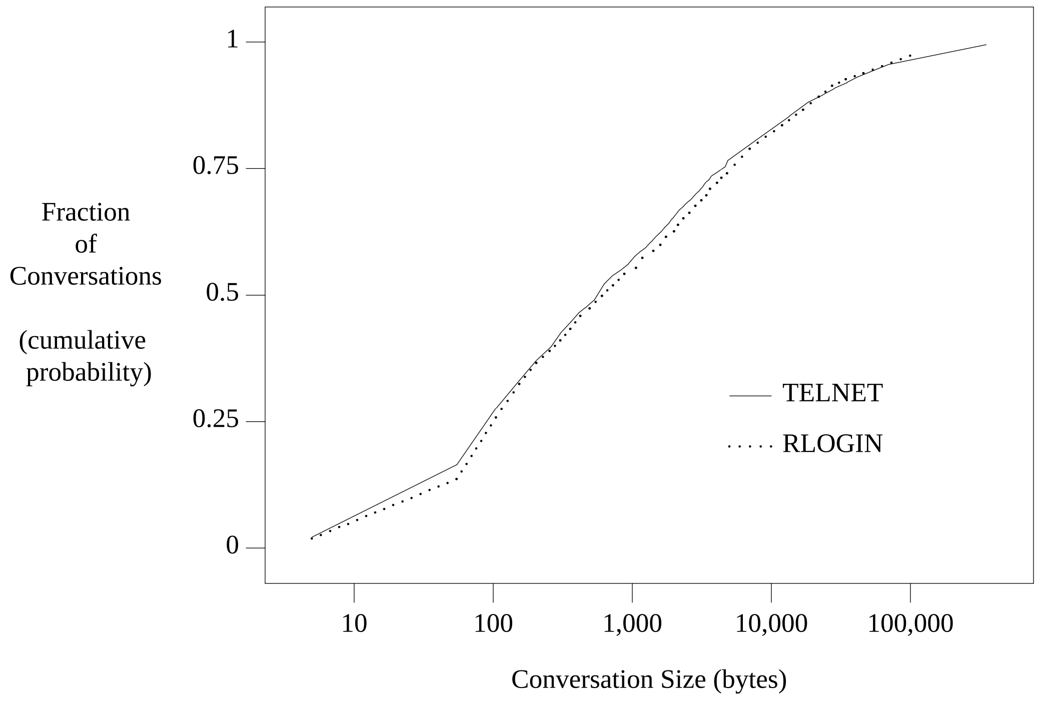

Figure 2.10. Bytes per conversation of interactive applications

2.5.4. Direction of Traffic Flow

Many wide-area network simulations have assumed unidirectional data flow within bulk transfer conversations (bulk transfer data flowing in one direction with no data flowing back) and symmetrically bidirectional data flow within interactive conversations (interactive keystrokes flowing in one direction with matching echoes flowing back) [25] [32] [101]. In contrast, we find that many bulk transfer conversations are bidirectional and many interactive conversations are asymmetric. Simulated traffic sources should reproduce these characteristics.

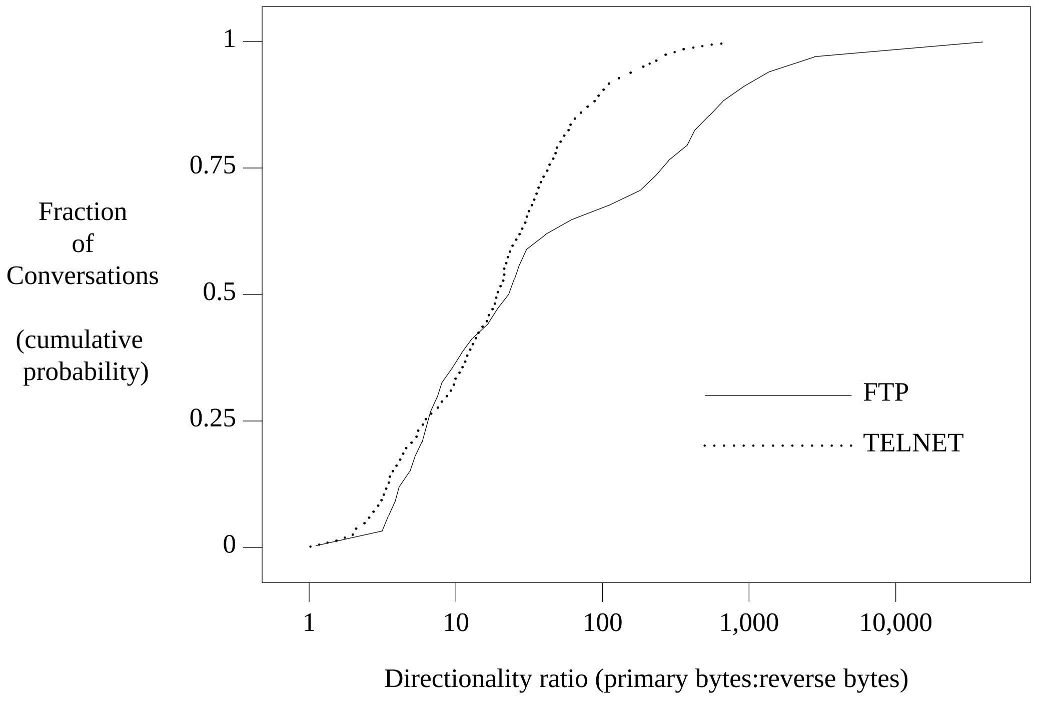

Figure 2.11. Direction of traffic flow

We define a conversation's *directionality ratio* as the ratio of the number of application-level bytes flowing in the conversation's primary direction to the number of application-level bytes flowing in the reverse direction. The primary direction is simply that in which the most bytes are transmitted. Figure 2.11 shows the distribution of directionality ratios for FTP and TELNET conversations.

We find that 36% of FTP conversations exhibit less than a 10:1 directionality ratio. The percentage of NNTP and SMTP conversations for which this ratio holds is even higher. Thus, bulk transfer conversations are not symmetric, but there is a significant amount of reverse-flowing data. On the other hand, we find that 67% of TELNET conversations exhibit more than a 10:1 directionality ratio. The percentage of RLOGIN conversations for which this ratio holds is approximately the same. Thus, interactive conversations are not as asymmetric as bulk transfer conversations, but there is significantly more data flowing in one direction than in the other.

In support of the relatively low directionality ratios in bulk transfer conversations, our traces show that these conversations contain a request-response phase during which no data flows in either direction. In turn, this phase causes classic packet train behavior: a small handshake followed by a big burst. For example, FTP performs a control handshake before every file transfer. NNTP sends a query, waits for a response, and then transfers a news article, and similarly for SMTP and electronic mail messages. Thus, even when data bytes flow in only one direction, application-level control bytes flow in both directions. We again draw a parallel with an earlier local-area network study [69] that found file transfer conversations to have a directionality ratio between 2:1 and 5:1. This request-response behavior may influence congestion and flow control algorithms and should be included in simulation studies of these algorithms.

The relatively high directionality ratio in interactive conversations is explained by the contrast between the person and the computer at either end of a conversation. During an interactive session, a person typically generates single keystrokes and short commands at one end of the conversation. However, a computer at the other end often generates long replies to these commands, for example the contents of a file to be displayed on a screen. The result is a much larger number of bytes flowing from the computer to the person than from the person to the computer.

Small packets, short conversations, and bidirectional flow all contribute to the traffic dynamics of an internetwork. These characteristics of current traffic could affect previous studies of traffic segregation and oscillation [32] [58] [99] [102].

2.6. Conclusions

We have analyzed wide-area traffic traces from four different organizational subnetworks on the Internet. We have isolated the stream of traffic flowing between two application processes and termed this a conversation. Conversations fall into two categories, bulk transfer and interactive, according to which network application generates data for transmission. Bulk transfers are those generated by FTP (file transfers), SMTP (electronic mail), and NNTP (network news). Interactive conversations are those generated by TELNET and RLOGIN (both remote terminal applications).

We have identified conversation characteristics that contradict the following commonly held beliefs regarding wide-area traffic:

- Bulk sources transfer large amounts of data per conversation.
- Bulk sources transmit a uniform stream of large packets in only one direction.
- Bulk transfers account for 80 to 100% of the traffic. Interactive applications account for 20% or less.
- Interactive sources send small packets in one direction, and receive echoes of comparable size in the opposite direction.

In contrast, we have found that:

- Bulk transfer sources such as FTP transfer less than 10 Kilobytes during 75% of their conversations. Other bulk traffic sources, such as SMTP and NNTP, transfer even smaller amounts of data per conversation (see Figure 2.3).
- Bulk transfers follow a cycle composed of a control handshake followed by a data transfer, and they generate bidirectional traffic (see Figure 2.11).
- SMTP and NNTP send as many small packets as large packets (see Figure 2.5).
- Interactive applications account for 40% of wide-area traffic (see Table 2.3).

- Interactive applications routinely generate 10 times more data in one direction than the other, using packet sizes ranging from the minimum to the maximum packet size permitted on the network (see Figures 2.6 and 2.11).

These are important contradictions that may affect results of previous studies of network performance. These studies should be redone to verify that their results are insensitive to the inaccuracies in the original models. Certainly, the new findings should be incorporated in future traffic models to insure realistic results.

Another important observation is that the network behavior of a mature application is stable. The five applications responsible for most wide-area data traffic, FTP, NNTP, SMTP, TELNET, and RLOGIN, have remained largely unchanged for many years. Our analysis shows that the traffic generated by these applications can be described by certain application-specific characteristics that did not vary over the geographic sites sampled and did not change over eighteen months of measurements (see Appendix A). These characteristics are also independent of other factors such as day of the week and time of day. Such stability allows us to characterize individual conversations for each major application and confidently apply the results to a wide range of network conditions.

The next chapter describes a new wide-area internetwork simulator that directly samples our measured conversation characteristics to generate application traffic.

3. Network Simulation

3.1. Introduction

Computer simulation is a valuable tool for evaluating the performance of communication networks. It has several advantages over the alternatives, namely analysis and experimentation. Mathematical, statistical, and queueing analysis are useful and elegant approaches to many problems. However, they often force oversimplification in order to keep solutions tractable. The other alternative, experimentation with a real network, yields the most realistic results of all three techniques. However, high-speed wide-area networks are very expensive to build and operate. In addition, once built, networks are not always flexible enough to allow experimentation with the design parameters of interest.

A simulator has some of the advantages of each approach while avoiding most of their disadvantages. It can incorporate compact analytical models of network components. Simulation results using these models can be validated against closed-form analytical solutions. However, where an analytical model loses accuracy due to oversimplification, a simulator can substitute software models with arbitrary levels of detail. Thus, a simulator can accurately mimic real hardware. However, where a real network is built with expensive and inflexible hardware components, a simulator is a piece of easily modified software.

One disadvantage of simulations is that they are difficult to validate. A concise analysis can be carefully checked by a number of people, especially if it is published. In contrast, a simulator typically contains thousands of lines of code that cannot be rigorously checked or effectively published. As described in Chapter 4, we address this problem by comparing the results of simulations to known results and by continually inspecting debugging output from the simulator.

Another disadvantage of simulations is that they are compute-intensive. During a simulation study of high-speed wide-area networks, performing meaningful experiments on available hardware in reasonable amounts of time is a substantial challenge. A related problem is the large number of simulation runs necessary to explore the parameter space. As described in Chapter 4, we overcome these challenges by choosing a small number of relevant simulation scenarios, by scaling the problem description, and by running simulations in parallel on a network of workstations.

This chapter presents VCSIM (Virtual Circuit Simulator), a new wide-area internetwork simulator. We developed VCSIM to study traffic multiplexing at the entrance to a wide-area network. As shown in Figure 3.1, VCSIM simulates four network components: hosts, routers, switches, and links. It reads parameter settings from an input file that describes the network to be simulated. Source hosts send packets destined for the sink hosts along local-area links to the nearest router. Input routers at the edge of the wide-area network forward these packets to a wide-area virtual circuit. They fragment packets into the cells used by the wide-area portion of the network. Switches forward the cells along the path to the output router. The output router reassembles the original packet from the component cells and forwards it to the sink host. The sink hosts send response and acknowledgment packets along the reverse path. At the end of a simulation run, VCSIM reports performance metrics of interest.

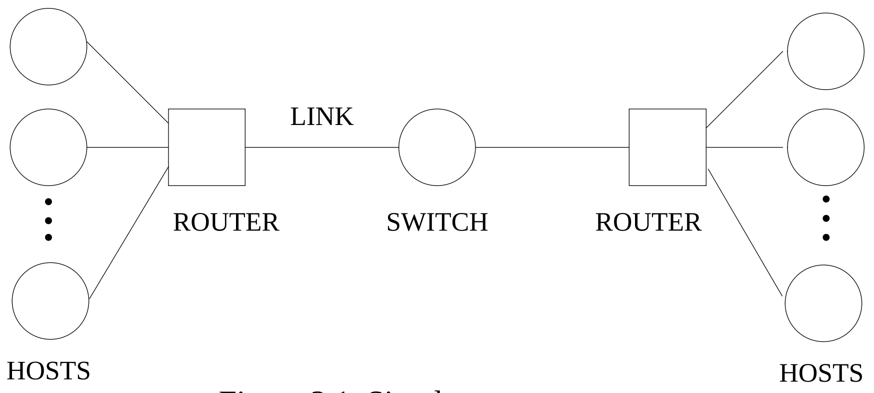

Figure 3.1. Simulator components

VCSIM is a discrete event simulator, that is, one that changes state only at discrete points in simulated time. It is derived from NETSIM [51], a simulator for wide-area ATM networks. In addition to functionality borrowed from NETSIM, VCSIM allows dynamic virtual circuit establishment and teardown, many-to-one mapping of application conversations to network virtual circuits, multiple queueing disciplines, and variable-sized cells. It is written in C++ [91], an object-oriented programming language evolved from C [55]. VCSIM is based on the C++ *task library* [92]. Tasks are co-routines, or concurrent subprograms with their own locus of control and private data. The library is geared for use in discrete-event simulation. It includes support for creating, scheduling, and deleting tasks, a general queue mechanism for synchronization and message-passing, and routines for managing a simulated clock. Tasks can communicate through message passing and shared memory.

The rest of this chapter describes VCSIM in more detail. Section 3.2 references previous work in wide-area network simulation. Section 3.3 describes the workload model we use to drive VCSIM. Our workload model is derived from the traffic characterization presented in the previous chapter. Section 3.4 presents the network model used by our simulator, that is, how VCSIM simulates network components such as hosts, routers, switches, and communication lines. Section 3.5 explains the input parameter choices available when using VCSIM, and Section 3.6 describes the output of VCSIM.

3.2. Previous Work

Researchers have developed many general-purpose simulation packages. An example is RESQ [85], by Sauer, MacNair, and Kurose, a simulator based largely on queueing theoretic models. It includes models and solutions for standard queueing systems, such as *M/M/1*, that can be combined to model more complex systems. Another example is CSIM [89], by Schwetman, a library of C routines. It includes support for co-routines, generators of standard probability distributions, and routines for managing a simulated clock. These packages can be used to study wide-area network behavior, as Verma did with CSIM [97], but they are also applicable to many other performance evaluation problems.

Other researchers have developed more specialized simulators. NEST, by Bacon, Dupuy, Schwartz, and Yemini [27], is tailored to communication networks. REAL, by Keshav [57], is built upon NEST and simulates TCP-IP wide-area networks. It emphasizes simulation of congestion control algorithms and has been used for that purpose in several studies [25] [58]. Similar network simulators have been used in studies by Ramakrishnan and Jain [83] and Zhang [101]. As mentioned earlier, we derived our simulator from NETSIM, by Kalmanek and Morgan [51]. NETSIM is an ATM wide-area network simulator that has also been used to study congestion control [40].

We chose to build a largely new simulator instead of using an existing one for three main reasons. First, to our knowledge, no existing simulator was well-suited to the problem of multiplexing datagrams over virtual circuits. VCSIM provides functionality that is key to our study and could not be found in other simulators. Second, we wanted to understand every aspect of a tool that was central to our study. It is difficult to achieve the same degree of familiarity with software written by someone else, perhaps software for which only binaries are available. Third, we wanted to keep the simulator small. Small often translates to fast in this type of system, and simulation speed was a concern from the beginning. We wanted to explore the implications of small cells on network performance, and simulating a cell-based network entails processing many more events than simulating a packet-based network. By including in VCSIM only functionality that was relevant to our study, we created a small and efficient simulator.

3.3. Workload Model

3.3.1. Motivation and Goals

An accurate workload model is an important component of any simulation study. A network's workload is its traffic. Thus, network simulations need a good model of traffic because network performance is critically dependent on the input traffic mix. We used the traffic characteristics presented in the previous chapter to derive a model of application-level conversations for each of the major types of wide-area traffic, namely FTP, NNTP, SMTP, TELNET, and RLOGIN. In a previous study, we laid the foundations for the workload model described here [21].

There were three main goals when building our model:

- The model should be realistic. It should accurately reproduce the measured traffic characteristics, not rely on intuitive ideas of how traffic behaves.
- The model should be efficient. It should be fast and compact so that it can drive simulations of large high-speed networks without consuming inordinate amounts of computing resources.
- The model should be network-independent. In other words, it should be applicable to a large variety of network simulation scenarios and not be limited to the measured network conditions.

3.3.2. Achieving Realism and Efficiency

Here we address the first two modeling goals: realism and efficiency. We need to faithfully reproduce the characteristics of real traffic in a way that is suitable for detailed computer simulations of high-speed networks.

We reproduce our measured probability distributions with the *inverse transform method* [50]. In general, the inverse transform method maps uniformly distributed 0-1 random variables from the ordinate of a cumulative distribution onto its abscissa. If a distribution obeys a known analytical expression such as *y = f(x)*, the inverse transform method simply inverts the function *f*, where *y* ranges between 0 and 1 and *x* ranges through the domain of the random variable of interest.

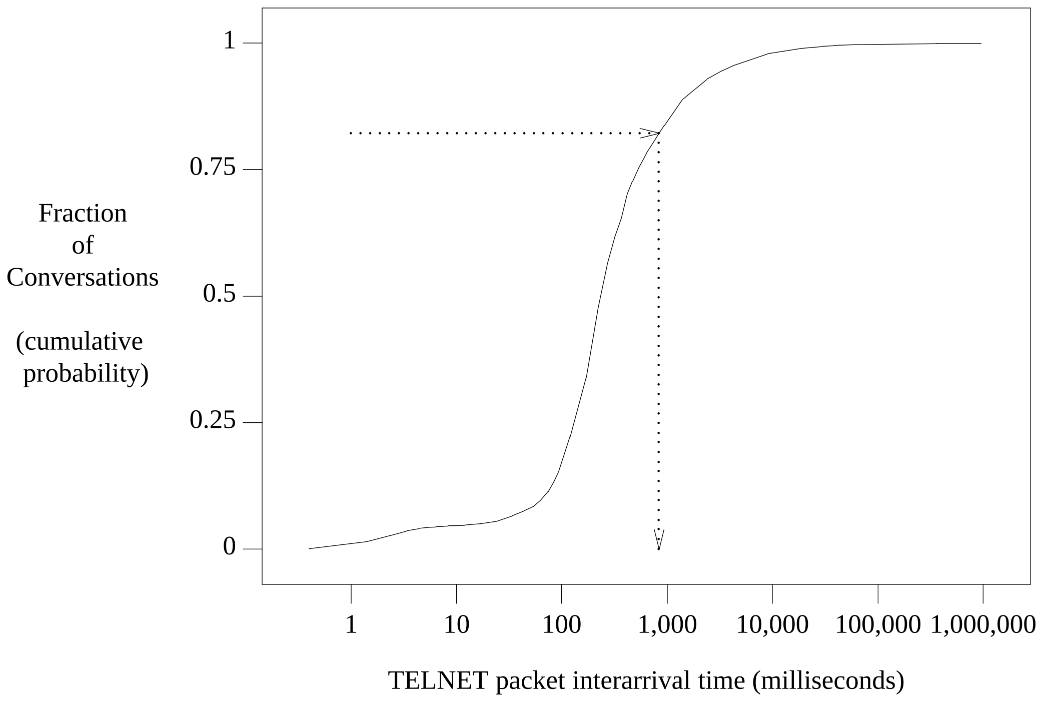

Figure 3.2. The inverse transform method of generating traffic characteristics

Figure 3.2 shows an example of how this process generates measured traffic characteristics. It graphs the cumulative distribution of packet interarrival times for TELNET conversations from our Berkeley trace (see Chapter 2). First, we sample the ordinate with a random number between 0 and 1 (0.82 in Figure 3.2). Then, we use the corresponding point on the abscissa (840 milliseconds in Figure 3.2) as the next packet interarrival time to use for simulation. A simulation run takes many such samples, generating random numbers throughout the 0-1 range and thus producing application data lengths with the desired probability distribution.

Since we do not have analytical expressions but measured data points, we invert a piecewise linear representation of the distributions. We build a histogram of the measured points, sum the histogram bin heights, and divide each by the total number of measured points to create a cumulative histogram with values between 0 and 1 (we saw many of these cumulative histograms in Chapter 2). We store each measured cumulative histogram in a two-dimensional software array. The first dimension, *x*, holds measured values. The second dimension, *y*, holds the corresponding probability values between 0 and 1. To generate a random variable, we first generate a 0-1 uniform random number *r*. We then perform a binary search on the second dimension of the array until we find the two *y* values between which *r* falls. Finally, we interpolate between the corresponding two *x* values to determine our random variate.

Rather than using a raw histogram, another way to implement the inverse transform method would be to sort the measured data points and keep only every 100th or 1,000th point. These measured values could be placed in a one-dimensional array indexed by 100 or 1,000 values between 0 and 1. To generate a random variate, the array could be indexed directly by a 0-1 random number *r*, thus eliminating the binary search of the previous discussion. This alternative approach is compact and efficient, but it loses information. We chose the raw histogram approach because it automatically adapts to the amount of information in the measured data. The number of bins in one of our raw histograms is proportional to the number of distinct values measured for a given traffic characteristic. The same is not true of a static scheme with 100 or 1,000 array elements.

Furthermore, we found the binary search on the raw histograms to have good performance. Measurements of running times show that VCSIM spends less than 0.02% of its time generating traffic characteristics. For example, on a SPARCstation 2 computer, a call to the routines that implement our traffic models takes an average of 90 microseconds. On the same computer, a sample simulation run that makes 6,049 calls to these routines takes 49 minutes of wallclock time. Thus, the latency introduced into the simulation process by our implementation of the inverse transform method is negligible.

There remain some concerns regarding the accuracy of the inverse transform method. In particular, the method ignores correlations in the measured data by sampling the data with a sequence of statistically independent random numbers. For example, if we were to use the procedure described above to directly reproduce our measured packet interarrival time distribution for bulk transfer applications, we would not capture the packet train behavior of bulk transfer traffic noted in Chapter 2. We in fact incorporate important correlations in our workload model by using detailed knowledge of application program behavior and using only a subset of our measured probability distributions. For example, our model reproduces the tendency of bulk transfer packets to arrive back-to-back by ignoring the packet interarrival time distribution for bulk transfer applications. Instead, our model sends packets as fast as the simulated internetwork allows. The following section describes other instances of this technique. The only additional inaccuracies introduced by our use of the inverse transform method stem from quantization errors incurred while building histograms and interpolating between histogram bins. As mentioned above, we have used enough histogram bins to make these errors negligible.

In short, the inverse transform method faithfully reproduces the measured traffic characteristics. Furthermore, it results in compact array representations of these characteristics, and it can quickly generate these characteristics through binary search. This approach thus meets our realism and efficiency goals.

3.3.3. Achieving Network-Independence

Our third modeling goal is network-independence. An empirical traffic model should not be restricted to the network conditions prevalent during the measurements it is based on, for example, the state of the Internet at noon on October 31, 1989. It should be applicable to the simulation of faster and slower networks, more and less heavily-loaded networks, larger and smaller networks, and so on.

We are armed with a wide array of measured traffic characteristics, as shown in Chapter 2, and a method to faithfully and efficiently reproduce them for simulation, as described in the previous section. However, we are not free to use all these characteristics in our models because many of them are network-dependent. For example, as we discussed in Chapter 2, packet interarrival times for bulk transfer applications depend on network speed, network congestion, and other factors. Similarly, packet sizes for bulk transfer applications reflect the maximum transmission unit of the network carrying the traffic. We would like our models to capture application-specific behavior that applies to all networks carrying that type of application traffic.

Our approach is to discard network-dependent traffic characteristics and incorporate in our models only network-independent characteristics. The next two sections explain how we sample a subset of our measured probability distributions to model individual conversations of two general categories of wide-area traffic: bulk transfers and interactive sessions. The relevant traffic characteristics are reproduced by a library of software routines that implement the inverse transform method and can be linked with any network simulator [20].

In contrast to our success in modeling characteristics of individual conversations, we were unable to derive a network-independent model of conversation arrivals. Our measured traffic data shows that conversation interarrival times depend on geographical site, day of the week, time of day, and other factors. Thus, we were forced to leave the choice of a conversation interarrival distribution to the user of VCSIM. For this purpose, we provide generators of standard probability distributions. The available distributions include *constant* and *exponential* (more can easily be added). Fortunately, the lack of an empirical model for this aspect of the workload does not hinder our study of multiplexing traffic at the entrance to a wide-area network. We can learn much about the performance and resource consumption of a network with a simple model of conversation arrivals, as described in Chapter 4.

3.3.4. Bulk Transfer Model

We model conversations for the three dominant bulk transfer applications of wide-area networks: FTP (file transfers), NNTP (network news), and SMTP (electronic mail). These applications are concerned mostly with throughput, not delay, and they share many important behavioral characteristics. More specifically, they have a set of data items to send across the network as quickly as possible. For FTP these data items are files, for NNTP they are news articles, and for SMTP they are mail messages.

Figure 3.3 contains pseudo-code for our model of bulk transfer conversations. The model has two parts, source and destination. The source model represents the side of the conversation that originates the bulk transfer conversation and sends data items. The destination model represents the other side; it receives the bulk data. In our pseudo-code, to *generate* means to sample the corresponding probability distribution for the appropriate application type, using the inverse transform method as described above.

For example, in the case of an FTP conversation, the source model first uses the inverse transform method to sample the cumulative distribution of the number of files sent during FTP conversations. After thus selecting the number of files to be sent by the current simulated FTP conversation, the model goes into a loop that iterates over each of those files. For each file, it samples the distribution of the size of FTP handshake packets, sends a packet of that size, and waits for a response from the destination. This handshake mimics the exchange of control information carried out before the transfer of every data item by FTP and the other bulk transfer applications, as noted in Chapter 2.

Once the handshake is complete, the model generates the number of bytes in the current file. These bytes are partitioned into packets all having the maximum size allowed by the simulated internetwork (except for the last packet, which may be smaller). The data packets are then sent to the destination as fast as the simulated internetwork allows. For example, if the network uses 1.5 Megabit/second links, that is the maximum rate at which data leaves a host. Similarly, if the hosts use TCP as its transport protocol, then TCP’s window flow control mechanism

```
Generate the number of data items to be transferred

For each item
    Generate the size of the handshake packet
    Send the handshake packet and wait for a response
    Generate the number of bytes for the item
    Fragment the item into maximum-sized packets
    Send the packets as fast as the simulated internetwork allows
```

A. Source

```
Forever
    Wait for a handshake packet from the source
    Generate the size of the response packet
    Send the response packet
    Sink data packets from the source
```

B. Destination

Figure 3.3. Model of a bulk transfer conversation

dictates how many packets can be outstanding between source and destination. Actions below the application level, such as packet acknowledgements and retransmissions, are not the responsibility of the workload model and are also handled according to the protocols being simulated.

The model for the destination end is simpler. The destination waits for a handshake packet from the source and responds without delay. It first generates the size of the response packet by using the inverse transform method, as usual. It then sends the response packet, completing the handshake and allowing the source to send the packets making up a data item. Finally, it sinks all data packets and goes back to waiting for new handshake packets.

Our bulk transfer model achieves network-independence by letting the simulated network dictate much of the behavior of bulk transfer conversations. The handshake process takes at least one round-trip time in the network of interest, as it should. For instance, it takes very little time over a state-wide network and considerably longer over a country-wide network. Similarly, packet sizes depend on the network being simulated. In addition, the rate of packet transmission depends on link speed, degree of congestion, flow and congestion control algorithms, and other characteristics of the simulated network. As we noted in Chapter 2, a good workload model should account for the relationship between the characteristics of bulk transfer traffic and the characteristics of the network carrying the traffic.

3.3.5. Interactive Session Model

We model conversations for the two dominant interactive applications of wide-area networks, TELNET and RLOGIN (both remote terminal applications). Interactive traffic is not as affected by network conditions as bulk transfer traffic, as we saw in Chapter 2. Interactive packet lengths and interarrival times reflect application behavior, not factors such as a network's maximum transmission unit or speed. Therefore, when building our interactive session model, we are free to use some of the measured probability distributions that we did not use for the bulk transfer model.

Figure 3.4 contains pseudo-code for our model of interactive conversations. The model has again two parts, source and destination. The source is the side of the interactive conversation where a person is directly producing data, for example by entering keystrokes that make up commands for a remote computer. The destination is the side where a computer receives and responds to data from the source, for example by executing commands and sending back the results. Again, the word *generate* means to sample a probability distribution using the inverse transform method.

```
Generate the duration of the conversation

While the conversation's duration has not expired
    Generate a packet interarrival time
    Wait for packet interarrival interval
    Send a minimum-sized packet
```

A. Source

```
Forever
    Wait for a packet from the source
    Generate the size of the reply
    Send the reply
```

B. Destination

Figure 3.4. Model of interactive conversations

The source model first generates the duration of the new conversation in milliseconds. Then, while simulated time has not reached the conversation expiration time, the model sends and receives packets. For each source packet, it generates the next packet interarrival time, waits for the interarrival interval to pass, and sends the packet. These packets have the minimum length supported by the simulated network, since they predominantly carry single keystrokes or short command strings.

The model for the destination end always waits for a packet from the source and replies without delay. It first generates the size of the reply. This response could be a simple echo of the source data, or a larger reply from the execution of a command. Finally, it sends the reply in the same way our bulk transfer model sends a data item. That is, it fragments the reply into maximum-sized packets and sends these packets as fast as the simulated internetwork allows. For instance, if the reply is a simple keystroke echo, a single small packet flows back to the source. On the other hand, if the reply contains a large file to be displayed on the remote user's terminal, it will be sent as a stream of packets subject to link speed, congestion, and other simulated network factors.

As we saw in Chapter 2, TELNET and RLOGIN have similar enough characteristics that we can use a single set of measured probability distributions to represent both applications. We chose to use the TELNET characteristics to model a composite TELNET/RLOGIN traffic type.

3.3.6. Miscellaneous Conversation Model

VCSIM models one additional category of network traffic, miscellaneous (MISC). A miscellaneous source generates traffic according to standard probability distributions. The user specifies the distributions for conversation length in packets, packet interarrival time in seconds, and packet length in bytes. The available distributions are the same as for conversation interarrivals, namely constant and exponential (more can easily be added).

The MISC traffic model can simulate traffic types not represented by FTP, NNTP, SMTP, or TELNET/RLOGIN. An example is a malicious traffic source that floods the network with back-to-back maximum-size packets without obeying any flow control or congestion control scheme. Such traffic types are useful in studying the robustness of network control algorithms under atypical operating conditions.

Thus, VCSIM can use any mix of FTP, NNTP, SMTP, TELNET/RLOGIN, and MISC models to drive a simulation. The following section describes how VCSIM simulates the network components that carry the traffic generated by our workload model.

3.4. Network Model

3.4.1. Overview

VCSIM is made up of functional modules that represent wide-area internetwork components such as hosts, routers, switches, and links. Each module is implemented as a set of C++ tasks that work together to simulate the behavior of a network component. VCSIM simulates network activity at the level of individual packets and cells. Packets flow between hosts and routers in the local-area part of the simulated internetwork, while cells flow between routers and switches, and between switches, in the wide-area part of the network. Communication between modules, as well as between tasks within a module, takes place through queues of packets and cells. The simulator keeps track of every packet and cell on the network, and advances simulated time whenever a packet or cell incurs transmission delay or speed-of-light propagation delay while going over a communication line.

Figure 3.5 shows a sample internetwork built using these simulation modules. The rest of this section describes these modules in more detail.

3.4.2. Hosts

Hosts are sources and sinks of simulated packets. A source host generates traffic for one application-level conversation at a time, according to the workload model described earlier in this chapter. Thus, there are five types of sources: FTP, NNTP, SMTP, TELNET/RLOGIN, and MISC. The control actions specified by the Transmission Control Protocol (TCP) [15] are superimposed on the traffic generated by the conversation models.

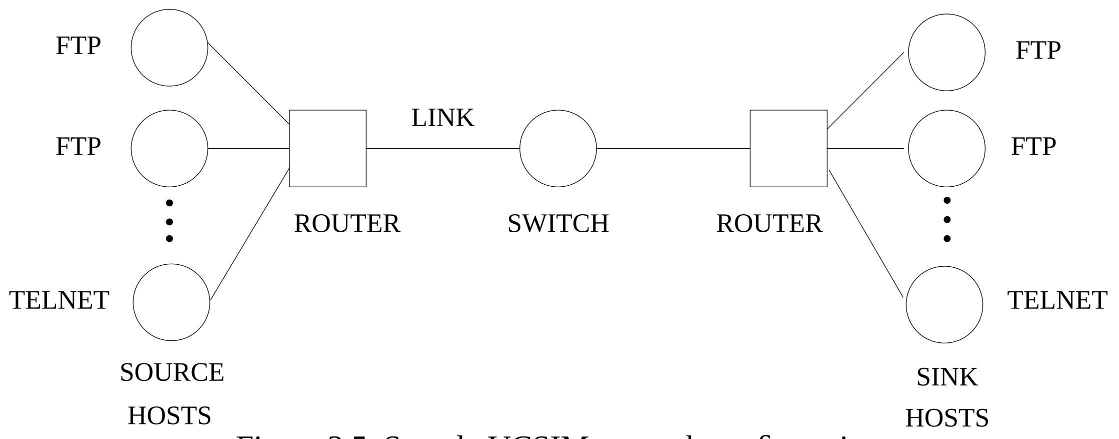

Figure 3.5. Sample VCSIM network configuration

In particular, VCSIM sources reproduce TCP's ordered, reliable data delivery and its sliding window flow control mechanism. Sequence numbers are assigned to packets in the order the packets are generated. These numbers serve to guarantee that packets are delivered in order at the destination, and allow the destination to individually acknowledge the receipt of each packet. Any packet sent but not yet acknowledged is considered outstanding. If a packet is outstanding longer than a certain interval proportional to a dynamically estimated round-trip time, the source considers the packet lost and retransmits it.

The window mechanism insures that at most one window size worth of data is outstanding between source and the destination. It can be used for flow control to prevent the source from sending more packets than the destination can handle. It can also be used for congestion control to prevent overloading the network as a whole. VCSIM implements the dominant variant of this mechanism, namely the slow-start congestion control scheme [47]. It can also simulate a simple fixed-window version [15].

Sink hosts simulate the destination ends of application-level conversations according to the workload model described earlier in this chapter. They accept packets from the network and reply with acknowledgments. Replies larger than a minimum-length packet sometimes accompany these acknowledgements, as dictated by the traffic destination models described earlier. Sinks also manage the destination end of the TCP sliding window. The window mechanism keeps track of the sequence numbers in incoming packets and acknowledges those packets that arrive in sequence. The source uses these acknowledgments to adjust its window and to retransmit packets when necessary.

3.4.3. Routers

Routers are a key network component in our study of multiplexing traffic at the entrance to a wide-area network. They are responsible for interleaving multiple application-level traffic streams from the local-area portion of the network and scheduling them onto the shared link to the wide-area portion of the network. Input routers accept packets from source hosts, fragment them into cells, and send the cells on a virtual circuit to an output router. Output routers take cells from a virtual circuit, reassemble them into the original packet, and deliver it to a sink host.

Routers offer four other major services: they establish virtual circuits, they tear down virtual circuits, they map application-level datagram streams to wide-area virtual circuits, and they service queues of packets and cells. Each of these services is controlled by a simulation parameter that can be configured to a range of values. We describe the available values and their effect below.

Virtual circuits can be permanent or dynamic. Permanent virtual circuits are pre-established to all destinations at the beginning of the simulation. They are never torn down. Dynamic virtual circuits are established when data arrives for a destination to which no appropriate virtual circuit yet exists. Dynamic virtual circuits are torn down if they have been idle longer than a timeout interval. Timeout intervals can be set to any value with the granularity of the underlying simulation clock.

The mapping scheme can be one of three: one virtual circuit per application-level conversation, one virtual circuit per destination router, and one virtual circuit per type of traffic flowing to each destination router. The queueing disciplines available are first-in first-out (FIFO) and round robin (RR), both among conversations sharing a virtual circuits and among virtual circuits sharing a communication link. Other priority-based disciplines like shortest-packet-first were considered but discarded because, in their simplest form, they can alter the order of conversation-level data. Wide-area networks should avoid delivering out-of-order data because this can trigger costly reordering procedures in transport-level protocols. The reordering problem can be avoided by treating data for the same conversation as a special case, but only at the cost of the simplicity provided by FIFO and RR. Simplicity is important to allow efficient hardware implementations of network control algorithms.

Figure 3.6 is an internal representation of a VCSIM router. Packets arrive on a local-area link from a host on the left of the figure, and cells leave on a wide-area link to a switch on the right. Packets carry an application-level conversation identifier and a traffic type identifier (one of FTP, NNTP, SMTP, TELNET, or MISC). Packets arriving on the local-area link are placed on a packet queue associated with each conversation. They are also assigned a virtual circuit (VC) according to the conversation-to-VC mapping scheme in use. Associated with this virtual circuit is a cell queue. Packet queues are serviced in the order dictated by the packet queueing discipline and fragmented into cells. These cells are then placed on the appropriate cell queue. Finally, cell queues are serviced in the order dictated by the cell queueing discipline.

We now examine the effects on router behavior of different conversation-to-VC mapping schemes. Assume the network topology shown in Figure 3.5, where there is only one input router and one output router. Further assume that all necessary virtual circuits are already established. To see what happens in one extreme, when the mapping calls for one VC per conversation, consider the lower half of Figure 3.6. Packets from a TELNET conversation arrive and are placed in the packet queue for that conversation. That conversation is assigned its own VC, and therefore its own cell queue. To see what happens in the other extreme, when the mapping calls for one VC per destination router, consider the upper half of Figure 3.6. Packets from two FTP conversations arrive and are placed in the packet queue for each conversation. In this case, however, both conversations are assigned the same virtual circuit since they flow through the same destination router. These conversations thus share a single cell queue.

To see what happens in the intermediate case, when the mapping calls for one VC per traffic type per destination router, consider all of Figure 3.6 as a whole. Two FTP conversations and one TELNET conversation are flowing through the router. As always, each conversation has its own packet queue. In this case, however, the two FTP conversations share one VC and thus one cell queue, while the TELNET conversation is assigned its own VC and cell queue.

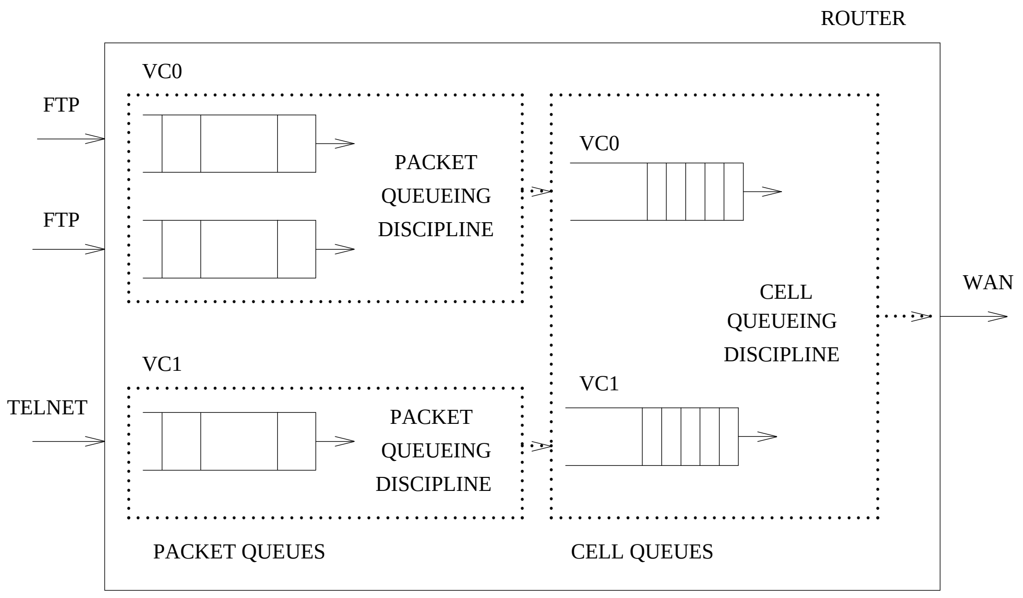

Figure 3.6. Simulation model of a router

Queue handling also has important effects on the behavior of the router. Recall that packet and cell queueing disciplines can be either FIFO or RR. A set of queues serviced in FIFO order effectively collapse into a single queue. For example, if the packet queueing discipline for the router in Figure 3.6 is set to FIFO, then packets from the two FTP queues are removed in the order they arrive and the two FTP packet queues behave as one. In contrast, if RR is used, then packets from the two FTP queues are interleaved. First a packet from the first queue is removed, then a packet from the second queue, and so on. Naturally, the round-robin is performed only on non-empty queues.

Another aspect of queue handling is the time at which queues are serviced. In VCSIM, the set of cell queues is always serviced whenever the wide-area link is free for another transmission and there are any cells to transmit. The timing for servicing packet queues is more complicated. VCSIM can be configured to perform window-based flow control on the virtual circuits established between routers. This network-level flow control mechanism operates in the routers in conjunction with the transport-level flow control operating in the hosts. When network-level flow control is in use, VCSIM routers will service a packet queue only when the corresponding virtual circuit has an open transmission window. That is, it will dequeue a packet, fragment it into cells, and queue the cells only when transmitting those cells will not violate flow control for the corresponding virtual circuit. On the other hand, when network-level flow control is not used, routers service packet queues as soon as data appears in them. In this simpler case, packet queues are always empty and cell queue handling dominates router behavior.

3.4.4. Switches

Switches perform a routing function internal to the wide-area portion of the internetwork. They accept cells from an incoming link and transfer them to the appropriate outgoing link based on the virtual circuit identifier carried by the cell. Associated with each virtual circuit using an outgoing link is a queue of cells. Switches perform cell-by-cell round-robin among the virtual circuit queues. That is, the set of cell queues for an outgoing link is serviced by a round-robin discipline whenever the link is free for transmission and there are any cells to transmit. We recall that cells are the smallest unit of data handled by the network, and the only unit of data handled by the switches. When these conditions hold, round-robin has been proven to be fair under high load, while first-in first-out disciplines are not [39] [76]. For this reason, round-robin is the only queueing discipline provided by VCSIM switches. Switches can have any number of ingoing and outgoing links, allowing the wide-area portion of the simulated network to take an arbitrary graph topology.

3.4.5. Links

Links simulate communication lines. They connect two nodes of any type (host, router, or switch) and introduce transmission and propagation delay. Transmission delay is the quotient of the length of the packet or cell being transmitted and the speed of the link. Propagation delay is the quotient of the length of the link and the speed of light in the medium. The passage of a packet or cell through a link thus causes simulated time to advance. The only other direct causes of delay in VCSIM are the waits in traffic sources for the next conversation or packet interarrival interval to complete. Delays indirectly incurred by packets and cells, such as waiting for processing in a queue while other packets or cells are transmitted, do not explicitly advance simulated time.

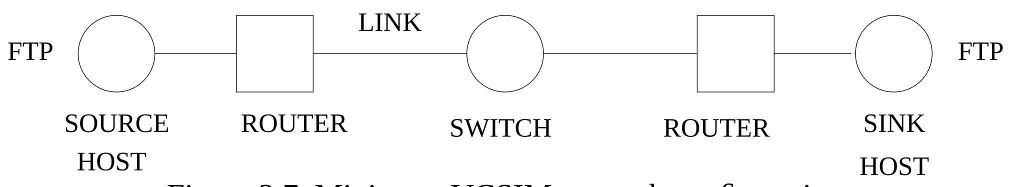

Figure 3.7. Minimum VCSIM network configuration

3.5. Input to VCSIM

The behavior of VCSIM is controlled by a specification file containing parameter settings in the form of *name-value* pairs [7]. Figure 3.8 is a sample specification file. It describes the network in Figure 3.7, with two hosts, two routers, one switch, and four links. Related parameters are grouped into blocks, for instance the block delimited by `router` and `router_end`. Parameters have default values, so that a partial specification file constitutes legal input. Similarly, the value of a parameter carries through from one block to the next block of the same type, so that we need only reset parameters that we want to change. Characters on a line after `//` are considered comments and ignored by the simulator.

3.5.1. Global Parameters

Global parameters apply to the overall simulation. `ticks_per_sec` scales the resolution of the underlying simulation clock. It specifies how many clock ticks constitute one second of simulated time. `bytes_per_cell` sets the amount of data carried by each fixed-size cell in the wide-area portion of the network. `max_pkt_size` defines the MTU of the network. The `seed` block contains seeds for the random number generator that controls all stochastic processes in VCSIM. VCSIM uses a composite congruential-Tausworth generator [68] that requires two odd integer seeds.

The `timer` block specifies a set of global timers in units of simulated seconds. `log` controls when statistics gathering begins. If can be set to a value greater than zero to postpone

```
// MINIMUM CONFIGURATION

// global parameters

ticks_per_sec 100000

bytes_per_cell 48

max_pkt_size 512

seed
    3091 4917    // odds
seed_end

timer    // seconds
    log 0.0
    checkpoint 0.0
    regeneration 175.0
    end 180.0
timer_end

// network components

router
    router_id 1
    vc_map per_conversation
    packet_queue fifo
    cell_queue round_robin
    buffer_policy infinite
    dynamic_vc 0      // boolean
    idle_time 0.090   // seconds
    meter_time 0.100  // seconds
router_end

router
    router_id 2
    report 0    // boolean
router_end

switch
    switch_id 3
switch_end
```

```
link
    link_id 1
    connects 1 3    // node ids
    speed 1.5e6     // bits/second
    delay 0.0225    // seconds
link_end

link
    link_id 2
    connects 3 2
    delay 0.0
link_end

host
    host_id 10
    function source
    traffic_type ftp
    flow_control slow_start
    window_limit 128   // packets
    retransmit 1       // boolean
host_end

host
    host_id 110
    function receiver
host_end

link
    link_id 10
    connects 10 1
link_end

link
    link_id 110
    connects 2 110
link_end

// conversation routes

route
    10 1 3 2 110    // node ids
route_end
```

Figure 3.8. VCSIM specification file for the network in Figure 3.7

statistics gathering until initial transients have passed. checkpoint specifies how often statistics are reported. It allows measurements of network behavior over time. regeneration specifies when to begin looking for a regeneration point, that is, a point at which the simulation is quiescent. The simulation terminates after end simulated seconds whether or not a regeneration point has been reached.

3.5.2. Component Parameters

The bulk of the specification file selects parameters for each components of the simulated network. Components can be nodes or links, where nodes are either hosts, routers, or switches. Many parameters for these components are closely related to the workload and network models described earlier in this chapter. An identifier parameter, for example router_id, is common to all components. We ignore identifier parameters in the discussion below.

3.5.2.1. Router Parameters

The first three router parameters, vc_map, packet_queue, and cell_queue, specify the conversation to virtual circuit mapping scheme, the packet queueing discipline, and the cell queueing discipline, respectively. Three other important router parameters follow. First, buffer_policy controls how buffers are allocated to active conversations and virtual circuits. This parameter can be set to either infinite, when packet and cell queues are allowed to grow without bound, or statistical, when a finite buffer pool is statistically multiplexed among the active conversations and virtual circuits. Second, dynamic_vc selects whether virtual circuits are dynamic or permanent. Third, in the case of dynamic virtual circuits, idle_time decides how long before an idle circuit is torn down.

The remaining two router parameters control statistics gathering and reporting. meter_time controls how often to sample certain fine-grained router behavior such as queue lengths. It is a separate timer to insure the samples are not inadvertently synchronized with other aspects of the simulation such as protocol timers, which would create sampling problems. report controls the verbosity of a router. When set to 0, it inhibits statistics reporting.

3.5.2.2. Host Parameters

Hosts have one of two functions, source or sink, and one of five traffic types, ftp, nntp, smtp, telnet, and misc. Sources run one of two variants of transport-level window flow control, slow_start or fixed_window. The next two parameters control aspects of this flow control. window_limit sets an upper limit to the size of the transmission window, and retransmit specifies whether the source will retransmit packets it deems lost.

3.5.2.3. Switch and Link Parameters

Switch and link parameter blocks are simpler. Switches operate inside the wide-area network, while we are mainly concerned with multiplexing traffic at the entrance to a wide-area network. We use switches for their basic function of routing cell traffic from link to link. For this reason, we have no need to change switch configuration between simulations. Link behavior is completely specified by three parameters: which two nodes the link connects, how fast the link is in bits per second, and how many seconds of propagation delay it introduces.

```
Starting simulation

Timeout reached
Simulated 180.000 seconds
Measured last 180.000 seconds


ROUTER 1 - PER CONVERSATION VCs - FIFO PACKETS - RR CELLS

Sum of queue lengths (bytes)
  1799 samples   Min: 0.000    Max: 432.000
  Mean: 102.857    Std: 141.412    95%ile: 419.191
Packet delays through router (seconds)
  34709 samples   Min: 0.000    Max: 0.003
  Mean: 0.003    Std: 0.000    95%ile: 0.003
Packet throughput (packets/second) 192.828
Cell throughput (cells/second) 2059.194
Bit throughput (bits/second) 790730.667
Packets dropped due to buffer overflow (number) 0
Packets dropped due to virtual circuit timeout (number) 0
Utilization of switch link (fraction) 0.485


HOST 10 - FTP

Round trip packet delays (seconds)
  34692 samples   Min: 0.047    Max: 0.054
  Mean: 0.054    Std: 0.001    95%ile: 0.056
Packet throughput (packets/second) 174.970
Bit throughput (bits/second) 716657.292
Packets retransmitted (number) 17


Ending simulation
```

Figure 3.9. VCSIM output for the simulation specified in Figure 3.8

3.5.2.4. Other Parameters

The final parameter block specifies a static route through the network in the form of node identifiers. The route must contain a source host, an input router, at least one switch, an output router, and a sink host, in that order. All data from the specified source host follows this route to the specified destination host.

A number of simulation parameters are absent from Figure 3.8. If they are relevant, VCSIM will use their default settings. Alternatively, we could explicitly choose their values by adding name-value pairs to the specification file. Examples of absent host parameters are `conv_arr_dist`, `conv_arr_mean`, and `conv_arr_std`. They specify one of several conversation interarrival time probability distributions, the mean of the distribution, and its standard deviation, respectively. Choices of probability distributions include `constant` and `geometric`, that is, discrete exponential.

Examples of absent router parameters are `flow_control`, `default_window`, and `maximum_window`. They configure the network-level flow control mechanism that regulates the transmission of cells in the virtual circuits between routers. The default is not to use network-level flow control.

3.6. Output of VCSIM

Figure 3.9 is the output of the simulation specified by the input file in Figure 3.8. It reports the behavior of the simulated network as seen by the single input router and the single source host. Some statistics, such as packet delays, are sampled throughout the simulation in the form of histograms. They are reported in terms of the mean, standard deviation, and other important values of the histogram. Other statistics, such as link utilization, are gathered only at each checkpoint or at the end of the simulation. They are reported as mean or total values up to that point.

For example, the FTP source achieved a throughput of 716,657 bits per second and observed a mean round-trip packet delay of 54 milliseconds. Similarly, on the average, the traffic consumed 102 bytes of queue space in the router, and the link between the input router and switch was busy 48.5% of the time.

We can extract statistics from VCSIM output and display them graphically, as we will see in Chapter 4.

3.7. Conclusions

We have described VCSIM, a discrete-event simulator of wide-area internetworks. VCSIM includes detailed models of network components such as hosts, routers, switches, and links. Of particular importance to our study of traffic multiplexing are hosts and routers. Hosts are sources and sinks of traffic. Routers mix traffic from different sources at the entrance to a wide-area network. VCSIM reports network performance as seen by the hosts and resource consumption as seen by the router.

The major research contribution of VCSIM is its workload model. VCSIM includes traffic models for individual conversations of each major type of traffic found in the wide-area portion of the Internet: FTP, NNTP, SMTP, TELNET, and RLOGIN. These models reproduce the traffic characteristics presented in Chapter 2 by using the inverse transform method to generate measured probability distributions. Our new workload model is network-independent, realistic, and efficient, and is applicable to a wide range of research problems.

We applied VCSIM to the problem of multiplexing application datagram streams at the entrance to virtual circuit networks. The next chapter describes our simulation study and presents our results.

4. Multiplexing Policies

4.1. Introduction

We now return to the problem of multiplexing datagram traffic over virtual circuit networks, previously introduced in Chapter 1. Datagrams arrive at the entrance to a wide-area network and must find a virtual circuit to go on. Routers assign datagrams to virtual circuits, and choose the order in which to transmit datagrams over the wide-area portion of the network. We aim to answer two questions:

- How should a router map a set of application-level datagram streams, or conversations, to a possibly smaller set of wide-area virtual circuits?
- What queueing discipline should the router use for multiplexing datagrams onto these virtual circuits?

The criteria for deciding how to answer these questions, previously defined in Chapter 1, are threefold:

- The network should give good performance to each conversation.
- The network should be fair to each conversation.
- The network should make efficient use of its resources.

The type of application responsible for each datagram stream is central to the multiplexing problem. Conversations are of different types, corresponding to different applications such as remote terminal sessions and remote file transfers. Each type offers a different workload to the network and demands different performance from the network. For example, traffic from terminal sessions is composed mostly of small packets and is more sensitive to delay than to throughput. In contrast, file transfer traffic is composed mostly of large packets and is more sensitive to throughput. If these two traffic types are indiscriminately mixed at the entrance to a wide-area network, a small remote terminal packet may be queued behind a number of large file transfer packets and may thus unfairly incur long delays. These problems suggest that networks should separate traffic streams.

In this chapter, we use simulation to investigate the benefits and drawbacks of separating traffic streams to varying degrees. In one extreme, each conversation maps to its own VC and the virtual circuits are served in a round-robin basis. This scheme cleanly separates traffic types as well as conversations. Similar schemes have been used in virtual circuit-based networks like Datakit [33]. In the other extreme, all conversations flowing between a source and destination router share a common VC, and all traffic is served in a first-in first-out basis. This scheme does not differentiate between traffic types or conversations. It has been used in datagram-based networks like NSFnet [82]. An intermediate approach is to assign all conversations of the same type flowing between a pair of routers to one virtual circuit for that type. This scheme separates traffic types but not conversations of the same type. It is motivated by shortcomings of the other two schemes and by trends in future networks, as explained in Chapter 1. Together with first-in first-out and round-robin queueing disciplines, these are the multiplexing choices provided by VCSIM, as described in Chapter 3.

We evaluate these schemes using two measures. One is performance as seen by network applications, in terms of throughput and delay. Throughput and delay also show how fairly the network allocates resources among conversations. The second measure is resource consumption as seen by the network provider, in terms of bandwidth and buffer space. We expect that establishing more VCs may offer better performance than sharing fewer VCs. On the other hand, sharing fewer VCs may use less memory than establishing more VCs. We use VCSIM to test this hypothesis and quantify the tradeoffs between performance, fairness, and resource consumption.

We find that networks should separate traffic types but not individual conversations of the same type. They should establish one virtual circuit per type of traffic flowing between two network points of presence, namely, between two routers. They should not establish one virtual circuit per conversation or one virtual circuit per pair of routers. In addition, networks should use round-robin scheduling among the virtual circuits sharing a communication link, not first-in first-out scheduling.

In a congested network, this multiplexing policy exhibits good performance and consumes moderate amounts of resources at the expense of some fairness among traffic sources of the same type. It maintains interactive delay near the possible minimum (a constant equal to one network round-trip time) and bulk transfer throughput near the possible maximum (a fair share of the network bandwidth) even as network load increases beyond saturation. Furthermore, it results in bottleneck buffer consumption that rises by only one network round-trip window with each bulk transfer conversation added to the offered load. Other multiplexing policies exhibit interactive delay that increases with offered load and buffer consumption that rises as a multiple of the offered load.

The following section surveys previous work in this area. The rest of the chapter presents our simulation study in more detail. Section 4.3 describes our simulation methodology while Section 4.4 presents our simulation results.

4.2. Previous Work

All networks multiplex traffic at various points throughout their fabric. Multiplexing policies and mechanisms acting at these points affect all aspects of network behavior. As a result, they have received attention since the outset of computer networking. Seminal work on many aspects of traffic multiplexing is covered in textbooks by Kleinrock [60] [62], Schwartz [88], Tanenbaum [93], and others.

There is also a wealth of more recent multiplexing studies. Examples are those by Feldmeier [29] and Tennenhouse [94], which discuss where in the protocol stack it is best to multiplex traffic. They conclude that multiplexing of traffic streams from different applications should be restricted to the network level. Our multiplexing schemes act in the network level at the entrance to wide-area networks, and therefore follow their guidelines.

In the interests of relevance, we limit the rest of our survey to work that, like ours, treats the benefits and drawbacks of separating traffic streams to various degrees. In a virtual circuit setting, Fraser, Morgan, and Lo [34] [76] [75], and Hahne [39], investigated the performance of various queueing disciplines when exposed to a mixture of traffic types They found that round-robin disciplines are fair under high load, while first-in first-out disciplines are not. Their results agree with those of Katevenis [53], Nagle [77], and Demers, Keshav, and Shenker [25], who reached similar conclusions for datagram networks. We based on these results our hypothesis that providing round-robin service among traffic types would achieve fairness across traffic types

All these studies were based on either analysis, simulation, or a combination of the two. In a more experimental vein involving the transmission of IP datagrams over X.25 virtual circuits, we found that it was necessary to open more than one virtual circuit to the same destination in order to improve both throughput and delay [10]. The throughput provided by a single virtual circuit was much lower than the available bandwidth due to the inadequate window sizes provided by public X.25 wide-area networks. Comer and Korb [14] noted the same throughput limitation. In addition, we observed high delays with a single virtual circuit even in situations were the window size was not a limiting factor. This experience helped motivate our proposal for establishing more than one virtual circuit to each destination, namely one virtual circuit per traffic type.

Our work adds to the existing body of traffic multiplexing results in two main ways. First, our methodology combines experimental and simulation techniques. We drive our simulations with traffic models derived directly from extensive measurements of real networks. Second, we present a new aspect of multiplexing policies, namely their effect on consumption of buffer memory. We show how establishing one virtual circuit per traffic type can meet performance goals with significantly less memory than establishing one virtual circuit per conversation.

4.3. Simulation Methodology

There are four methodology issues to address before we can attack the multiplexing problem with simulation:

- What simulation configurations are appropriate?
- Do the simulations provide accurate results?
- Can we run meaningful simulations on available computing resources?
- Can we reproduce previous simulation conditions when necessary?

This section describes how we addressed these issues.

4.3.1. Network Configuration

We configured VCSIM to simulate a network like the one in Figure 4.1. This network has a number of source hosts of various types, corresponding sink hosts, an input router, an output router, a switch, and the necessary links.

This standard configuration stresses the multiplexing problem at the input router. As a result of the parameter choices described below, the input router and its wide-area link constitute the only bottlenecks in the network. Any overload, and thus any queueing, will occur at these points. This configuration allows us to isolate the problem and study the effects of various input parameters on the output metrics of interest.

4.3.1.1. Input Parameters

We vary the following input parameters between simulation runs: the number and types of hosts sending and receiving traffic, the conversation-to-virtual circuit mapping scheme in the routers, and the packet and cell queueing disciplines in the routers. Host types can be any of FTP, NNTP, SMTP, and TELNET. Mapping schemes can be one virtual circuit per

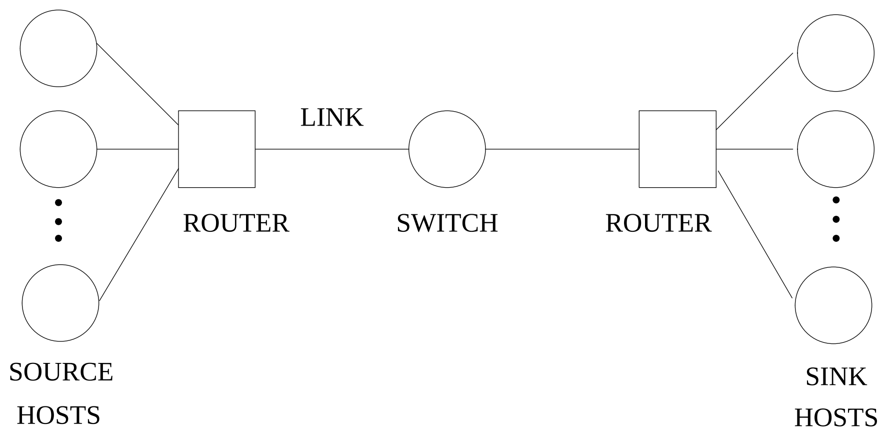

Figure 4.1. Standard VCSIM multiplexing configuration

conversation, one virtual circuit per traffic type, and one virtual circuit per router pair. The latter always translates to a single virtual circuit in our standard configuration. Queueing disciplines can be either FIFO or RR.

As described in Chapter 3, these are not the only parameters accepted by VCSIM, but we maintain most other parameters constant to keep the simulation study tractable. In general, our input parameter values follow the example shown in Figure 3.8. Later in this chapter, we explore the sensitivity of our results to variations in some of these parameters, in particular link speed and the limit on transport-level window sizes. We describe our specific parameter choices below.

All links are 1.5 Megabit/second, a relatively low but realistic speed. We match all link speeds so as not to limit how much a single host can load the network. Rather than using futuristic link speeds of 600 Megabit/second or higher, we scale our simulation experiments to 1.5 Megabit/second to keep computation requirements down. In addition, all links have zero delay, with one exception. The link between the input router and the switch has a 22.5 millisecond delay, which simulates a U.S. cross-country network with a 45-millisecond round-trip delay.

In the hosts, we use a limit of 128 outstanding packets on transport-level flow control windows. This limit is higher than the number of maximum-length packets that fit in a network round-trip window. This choice insures that hosts are not window-limited, that is, that hosts can transmit packets as fast as the internetwork allows without being unnecessarily throttled by transport-level flow control.

Again in the hosts, we use the default conversation interarrival distribution, namely a constant distribution of value zero. As a result, a source begins a new conversation as soon as the previous one ends. In combination with our choice of permanent virtual circuits, this interarrival distribution allows us to concentrate on network response to activity within conversations. There are no idle periods between conversations, no virtual circuit teardowns due to timeouts, and no virtual circuit establishment delays.

We also use default values for the parameters controlling network-level flow control in the routers. These parameters specify that there be no flow control acting on the virtual circuits between routers. Under these conditions, routers transfer data from their packet queues to their cell queues as soon as data appears. As a result, packet queues are always empty, which renders moot the choice of packet queueing discipline.

Finally, we configure the routers to have infinite buffer space. Packet and cell queues are thus allowed to grow without bound and consequently routers never drop data. Similarly, by default, VCSIM switches never drop cells. Wide-area networks avoid dropping data because the losses trigger retransmissions in the transport-level protocols running in the hosts. Retransmissions in turn lead these protocols to slow the rate at which they send packets, which results in lower throughput for the hosts. As this process acts on all traffic sources experiencing data loss, network utilization may decrease even as offered load increases. Networks that avoid these resource inefficiencies are considered *stable*. Naturally, we cannot build real networks with infinite buffers. However, simulating this scenario lets us measure how much memory would be needed in order not to drop data. Knowledge of this upper bound on memory sizes is very useful for wide-area network design.

4.3.1.2. Output Metrics

We keep track of the following output metrics: the throughput and delay obtained by the hosts, the buffer consumption inside the input router, and the utilization of the link connecting the input router to the switch. Host throughput and delay measure the performance obtained by network applications. A comparison of the performance obtained by individual conversations also measures fairness. Buffer consumption at the bottleneck router is an important measure of resource consumption in the network. Finally, bottleneck link utilization measures the overall load on the network. In high-load conditions, bottleneck link utilization also measures how efficiently the network is using the available bandwidth.

4.3.2. Accuracy

It is important to verify that our simulation results are accurate. We discuss two aspects of this process: validation, lengths of simulation runs, and confidence intervals.

4.3.2.1. Validation

We validated VCSIM by comparing simulation output to expected results. We first did this for simple deterministic cases where the expected output could be calculated from the input through arithmetic. We also ran restricted stochastic examples where the output could be compared to results predicted by statistical analysis. VCSIM passed all these tests. Finally, throughout our study, we continually checked detailed debugging output from VCSIM to confirm that it behaved as intended.

4.3.2.2. Lengths of Simulation Runs

In general, simulation results should reflect a system's steady-state behavior, not initial transients. We need to verify that our simulation runs are long enough to allow the network to reach a steady state. We arrived at an appropriate run length by plotting output metrics over time for very long runs of representative simulation experiments, and determining when initial transients ceased. We learned that our simulation scenario reaches steady state after approximately 1,000 round-trip times. All simulation experiments reported in this dissertation ran for 4,000 round-trip times.

A related issue is that of regeneration points. A simulation that has become temporarily quiescent is said to have reached a regeneration point. In our case, a regeneration point is reached when all queues in our simulated network are empty. In essence, a simulation begins anew after a regeneration point, and therefore is again subject to initial transients. We instructed VCSIM to stop simulating if a regeneration point was reached to avoid incorporating initial transients in our results. All simulation runs reported here ran without reaching a regeneration point.

4.3.2.3. Confidence Intervals

We also need to bound the variations intrinsic to stochastic simulation. A simulator like VCSIM generates many events based on a random number generator. Consequently, simulation results are in general not deterministic. Rather, they fall into a statistical distribution. A single simulation run produces only one sample from this distribution. The sample may come from the tail of the distribution and thus be unrepresentative of network behavior. We must gain confidence that we have statistically meaningful samples.

For this purpose we evaluated *confidence intervals* [30]. Assuming a set of results obeys a normal distribution, the central limit theorem states that 95% of values fall within one standard deviation from the mean. Similarly, 99% of values fall within two standard deviations of the mean. These two ranges are known as the 95% and 99% confidence intervals, respectively. For each multiplexing scenario of interest, we ran a number of simulations with identical inputs except for the seeds to VCSIM’s random number generator. We then calculated the mean and standard deviation for the set of results to obtain the desired confidence intervals. Typically, we simulated each scenario with 10 or 20 different seeds and verified that the results we report lie within the 95% confidence intervals.

All simulation results reported in this dissertation were verified through this procedure. For readability, we present only the means of the samples. However, in all cases we evaluated confidence intervals to insure our results lie near the true mean of their distribution. Incidentally, this process increased by an order of magnitude the number of simulation runs and therefore the amount of computing resources needed to produce our results.

4.3.3. Parallel Simulations

Computer simulation of cell-based wide-area networks entails large amounts of computation, even after scaling our link speeds to the relatively slow 1.5 Megabit/second. For example, simulating 180 seconds of network activity (approximately 4,000 round-trip times) in a multiplexing scenario with one source and one sink on a Sun Microsystems SPARCstation 2 computer takes approximately 2,000 seconds of wallclock time. Computation time increases with more complicated configurations. For example, simulating 180 seconds in a multiplexing scenario with five sources and five sinks on the same computer takes approximately 3,500 seconds of wallclock time. We ran many hundreds of such simulations.

We met the demand for processing power by running simulations in parallel on a network of workstations. Given a uniform processor architecture and file system name space, the Unix *pmake* utility executes jobs on idle workstations.† When told how many workstations it may use, pmake launches that many jobs in parallel, one per workstation. When one job terminates, it reuses the free workstation for another job as long as jobs remain.

† The pmake utility was originally developed for the Sprite operating system [79].

Our simulations are well-suited to this environment. An important consequence of our effort to keep VCSIM small is that its binary and run-time image fit easily into the main memory of our workstations. There was no need to page portions of VCSIM to secondary storage. In addition, each simulation run is self-contained: the VCSIM binary loads, it reads a small input file, computes until the simulation terminates, and finally writes a small output file. There is no communication between simulation runs.

Because of the low input and ouput activity, especially to and from any shared storage servers on the workstation network, we achieved high parallelism. We typically ran late at night on 25 to 30 SPARCstations. For example, we ran a suite of 150 multiplexing experiments in just over 6 hours of wallclock time. If the experiments had been executed serially, the same suite would have consumed more than 100 hours of wallclock time.

4.3.4. Provenances

Simulation experiments should be reproducible in case we need to revisit an earlier experiment to explore some aspect in more detail. With VCSIM, the combination of an output file and the corresponding input file uniquely identify a simulation run. In our study, we saved both files to provide our results with the necessary *provenances* [7]. Incidentally, this ability to exactly reproduce previous conditions is another advantage of simulation over experiments with a real network.

4.4. Simulation Results

We are now ready to evaluate different traffic multiplexing policies through simulation. As described in the previous section, we use VCSIM in our standard multiplexing configuration. We vary the number and types of hosts along with the multiplexing policies at the input router, and track the performance seen by applications along with the resources consumed by the network.

4.4.1. Three Multiplexing Policies of Interest

We began by investigating a total of twelve multiplexing policies. These twelve policies represent all combinations of our three conversation-to-virtual circuit mapping schemes, our two packet queueing disciplines, and our two cell queueing disciplines. As expected, however, they fell into three groups according to performance, fairness, and resource consumption. For readability, we present results for three representative policies:

- *Per-conversation VCs* – One virtual circuit per conversation with first-in first-out packet queueing and round-robin cell queueing.
- *Per-type VCs* – One virtual circuit per traffic type with first-in first-out packet queueing and round-robin cell queueing.
- *Per-destination VCs* – One virtual circuit per destination router with first-in first-out packet queueing and first-in first-out cell queueing.

Our choice of network configuration is responsible for the collapse of our twelve parameter combinations into three groups. Certain conversation-to-VC mapping schemes negate the effect of certain queueing disciplines, and vice versa. On the one hand, a round-robin discipline behaves like first-in first-out if it is only servicing one queue. This situation occurs, for example, with per-destination VCs; there is only one VC and thus only one cell queue. On the other hand, the performance characteristics of multiple queues are identical to those of a single queue if they are serviced by a first-in first-out discipline. In addition to these two factors, we recall that the absence of network-level flow control renders moot the choice of packet queueing discipline. The three policies outlined above represent the parameter space of interest.

4.4.2. Network Load

As apparent in the traffic characterization of Chapter 2 and the traffic models of Chapter 3, conversations of different types offer different workloads. We can now quantify this behavior through simulation.

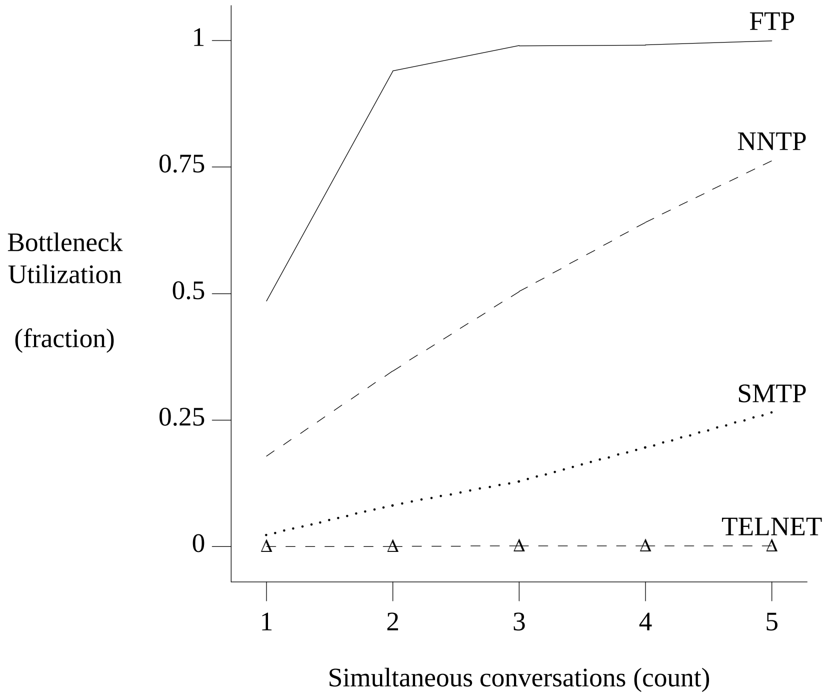

Figure 4.2. Load imposed on the network by each traffic type

Figure 4.2 shows the load imposed on the network by the four traditional traffic types modeled in the previous chapter. It graphs the average bottleneck link utilization as the number of conversations of each type of traffic is increased. The curves are for the multiplexing policy with per-conversation VCs, but all three multiplexing policies yield similar results under this metric. As shown, file transfers (FTP) saturate the network with only three or four simultaneous conversations. At the other extreme, remote terminal sessions (TELNET) leave the network mostly idle. Electronic mail (SMTP) and network news (NNTP) load the network to varying degrees.

The differences in offered load follow from the findings in Chapter 2 and the models of Chapter 3. A bulk transfer source sends the packets forming a data item as fast as flow control allows, except when it is engaged in a control handshake between data items. In the case of FTP, these data items are files composed mostly of multiple maximum-sized packets. The resulting back-to-back arrivals of large packets enables a small number of FTP sources to saturate the network. In contrast, NNTP and SMTP send news articles and mail messages, respectively, that are in many cases smaller than the maximum packet size. They thus send smaller packets than FTP with more frequent pauses for control handshakes. As a result, a higher number of NNTP and SMTP conversations are needed to saturate the network. Finally, interactive sources send mostly minimum-size packets separated by pauses in human time scales. The long pauses and small packets account for the fact that a great many TELNET sources are necessary to perceptibly load the network. Each traffic type behaves differently, suggesting that the network should treat them differently.

Incidentally, the network's ability to reach 100% utilization confirms that statistical multiplexing makes efficient use of bandwidth. A single conversation of any type cannot bring about full utilization due to the pauses in transmission described above. For the same reason, a single conversation cannot achieve throughput equal to the full available bandwidth. However, enough simultaneous conversations will reach 100% on both utilization and throughput counts in our simulated network. In a real network, protocol overhead can significantly curtail the throughput obtained by applications, as we shall see in Chapter 5.

We have so far presented simulation results for a network that carries traffic of a single type. In the following sections, we present results using a mix of bulk transfer and interactive traffic. In all the simulations discussed there, we maintain a single TELNET conversation while varying the number of simultaneous FTP conversations. For brevity, we use FTP as an example of a bulk transfer application, but our results are similar for NNTP and SMTP. As shown in Figure 4.2, NNTP and SMTP conversations impose a lighter load than FTP conversations. However, their overall behavior is the same: they send bulk data as fast as they can. Therefore, an appropriately higher number of simultaneous NNTP and SMTP conversations has similar effects on performance and resource consumption as the number of FTP conversations involved in our discussion.

4.4.3. Performance

In addition to network load, VCSIM allows us to quantify the delay and throughput obtained by each type of traffic involved in a simulation. We have discussed in Chapters 2 and 3 how different traffic types demand different performance. Within its resource limits, the network should provide high throughput to bulk transfer applications. In addition, the network should provide low delay to interactive applications. It should also be fair to individual conversations of each type of traffic.

4.4.3.1. Interactive Delay

Figure 4.3 shows the average delay obtained by interactive applications under different multiplexing policies. It graphs the average round-trip delay observed by a TELNET conversation as network load increases. Delay is expressed in units of network round-trip times (RTT), in our case 45 milliseconds. Normalizing metrics in this way helps yield a network-independent representation of our results. Load is expressed in units of simultaneous FTP conversations.

As shown, per-type VCs and per-conversation VCs exhibit good performance under load while per-destination VCs exhibit poor performance. For all three policies, delay begins very near 1 RTT, the minimum round-trip delay provided by the network. Delay increases almost imperceptibly with load under per-type and per-conversation VCs. These two multiplexing policies provide optimal performance from the point of view of interactive traffic sources. However, delay increases rapidly with load under per-destination VCs. In wide-area networks, with their substantial round-trip times, this performance degradation is readily noticeable by human users and is not acceptable.

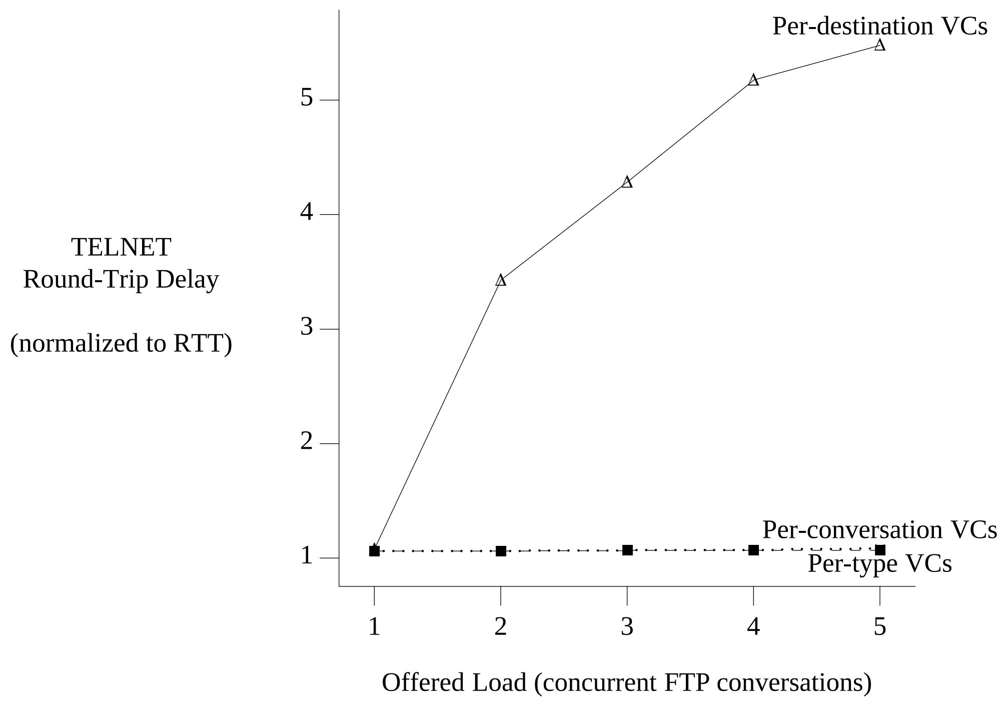

Figure 4.3. Interactive delay under different multiplexing policies

The reasons for the difference in observed delay are as follows. Interactive sources send mostly widely-spaced small packets, while bulk transfer sources send predominantly bursts of large packets. The multiplexing policies using per-type and per-conversation VCs separate these two types of traffic, giving each its own queue and servicing the set of queues in a round-robin fashion. The interactive traffic queues are assured of timely service regarless of how much bulk transfer traffic is queued, since the round-robin server visits the interactive queues as often as it visits the bulk transfer queues. In contrast, the multiplexing traffic using per-destination VCs indiscriminately mixes the two types of traffic in one first-in first-out queue. Under load, interactive traffic is often queued behind large amounts of bulk transfer traffic and thus incurs long delays.

4.4.3.2. Bulk Transfer Throughput

Figure 4.4 shows the average throughput obtained by bulk transfer applications under different multiplexing policies. It graphs the average throughput obtained by FTP conversations as network load increases. Throughput is normalized to units of bottleneck link speed, in our case 1.5 Megabits per second. Load is expressed in units of simultaneous FTP conversations. A background TELNET conversation is also present in these simulations to maintain a basis for comparison with the previous graph.

We see that all three multiplexing policies have the same average throughput characteristics. For all three policies, throughput begins at the maximum obtainable by a single FTP conversation. This maximum corresponds to the network utilization under one FTP conversation shown in Figure 4.2, and is not equal to the full bandwidth of the network for the reasons explained in Section 4.4.2. Throughput then degrades with load, but this is inevitable given that three or more simultaneous FTP conversations saturate the network, as shown also in Figure 4.2.

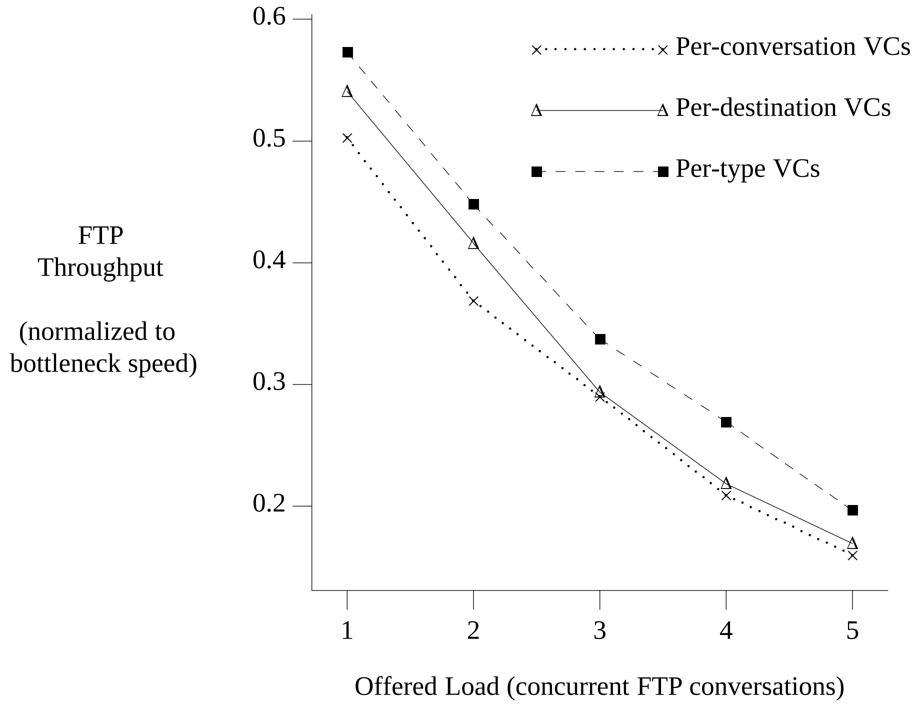

Figure 4.4. Bulk transfer throughput under different multiplexing policies

The key observation is that average throughput degrades equally with all three multiplexing policies.

The reasons for the comparable throughput behavior are as follows. When the network is lightly loaded, there is little queueing. Under these conditions, the choice of multiplexing policy has little effect on throughput. As the network becomes congested, regardless of the multiplexing policy, the bottleneck link saturates, queues build up, and delays rise. The transport-level protocol in the hosts only sends additional data when acknowledgements return. With higher delays, acknowledgements take longer to return to sending hosts, which in turn decreases average throughput for all hosts under all three multiplexing policies.

Although the average throughput characteristics are similar under all three multiplexing policies, individual bulk transfer conversations observe unequal throughput under certain multiplexing policies. We address this fairness issue below.

4.4.3.3. Fairness Among Bulk Transfer Conversations

Figures 4.5 and 4.6 show the throughput obtained over time by each of five simultaneous FTP conversations, under per-conversation VCs and under per-type VCs, respectively. The results for per-destination VCs are very similar to those for per-type VCs and we leave them out of the remaining discussion. We first note that the network achieves 100% utilization under both multiplexing policies (each of the 5 sources obtains approximately 20% of the available bandwidth). Both policies thus meet our bandwidth efficiency goals. However, per-conversation VCs meet our fairness goals more successfully than per-type VCs. As shown, per-conversation VCs divide network resources fairly and provide each conversation with the same level of throughput. In contrast, per-type VCs provide some conversations with better

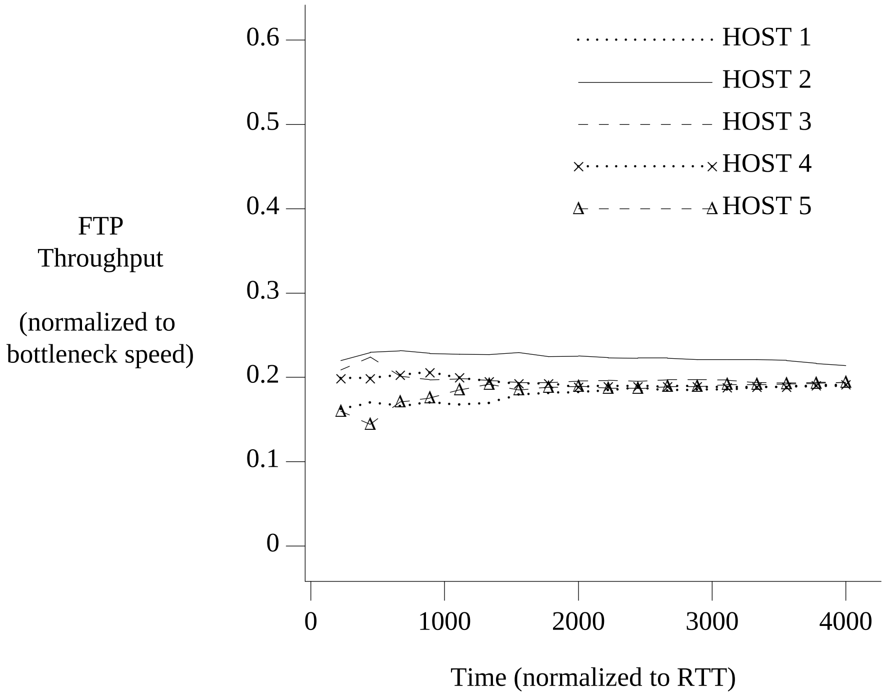

Figure 4.5. Fairness provided to bulk transfer conversations under per-conversation VCs

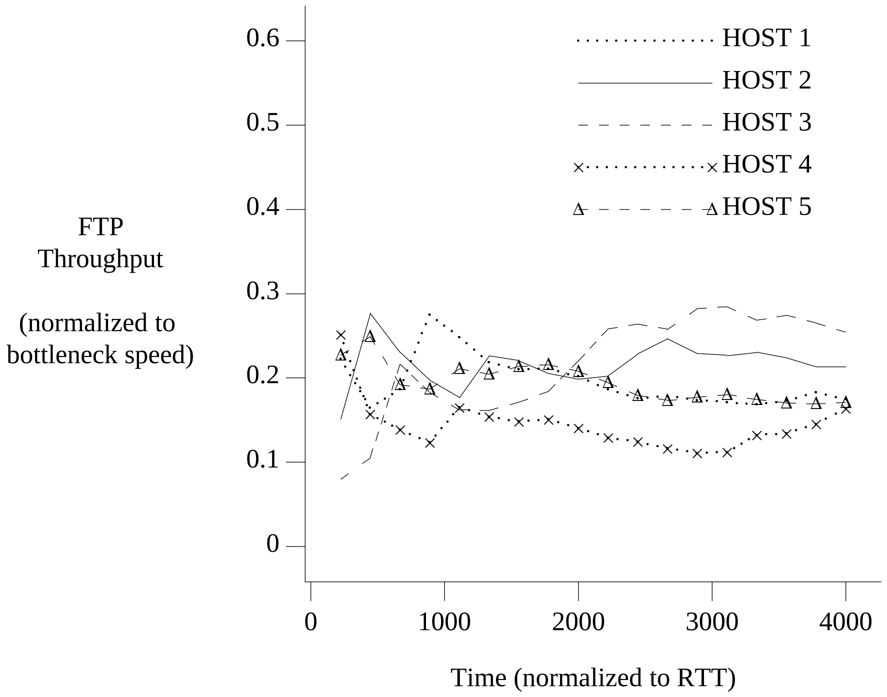

Figure 4.6. Fairness provided to bulk transfer conversations under per-type VCs

throughput than others.

The differences in throughput are caused by the interaction of queueing delays in the network and window flow control in the hosts. Per-conversation VCs give each conversation its own queue and service the set of queues on a round-robin basis. In this case, all hosts see nearly the same delay and their flow control mechanisms act in lockstep, that is, their windows remain in phase. The result is nearly identical throughput for all hosts.

On the other hand, per-type VCs mix conversations of the same type in a single first-in first-out queue. In this case, a subset of the conversations win the race for network resources. Their data arrives first at the front of the bottleneck queues and other conversations necessarily fall behind. The winning conversations see low delays in the form of timely acknowledgements, their windows advance, and they continue to send data. The losing conversations, however, see high delays, their windows do not advance, and they slow their transmission. The result is higher throughput for the winners than for the losers. In a related effect, hosts interpret delays higher than a certain threshold as dropped packets and proceed to retransmit those packets. Because of the randomness built into retransmission strategies, losers can become winners, causing previous winners to become losers. These phase dynamics are apparent in Figure 4.6, where different hosts obtain preferred treatment at different times.

Such *segregation* of traffic sources into winners and losers has been noted in previous studies of flow and congestion control [32] [58] [99] [102]. However, we find the resulting loss of fairness to be less severe than previously reported due to our use of smaller bulk transfers in our simulations. Previous studies have used large bulk transfers that do not reflect those found in real networks, as noted in Chapter 2. In contrast, our workload model faithfully reproduces real bulk transfers, as described in Chapter 3. Small bulk transfers hold resources for a relatively short time before a control handshake or the end of the conversation occurs. These pauses between data items allow other conversations access to resources.

We see in the discussion below how the per-type VCs multiplexing policy trades this loss in throughput fairness for substantial savings in memory costs.

4.4.4. Resource Consumption

In addition to performance, we are also concerned with resource consumption. Memory and bandwidth are costly resources in wide-area networks. Since the highest degree of queueing occurs at a network's bottleneck points, the amount of buffer memory consumed at these points is a useful measure of total memory consumption. We explore this measure below and separately address the efficient use of bandwidth in the next chapter.

4.4.4.1. Bottleneck Buffer Consumption

Figure 4.7 shows the bottleneck buffer consumption under different multiplexing policies. It graphs the instantaneous sum of queue lengths at the input router as network load increases, in particular the average of many such sums sampled throughout the simulation. Buffer consumption is normalized to units of network round-trip windows (RTW), in our case 8,438 bytes. Load is expressed in units of simultaneous FTP conversations. A background TELNET conversation is also present in these simulations.

In this case, per-type VCs and per-destination VCs consume moderate amounts of resources, while per-conversation VCs consume excessive amounts of resources. For all three policies, buffer consumption begins near zero. When the network is lightly loaded, there is no

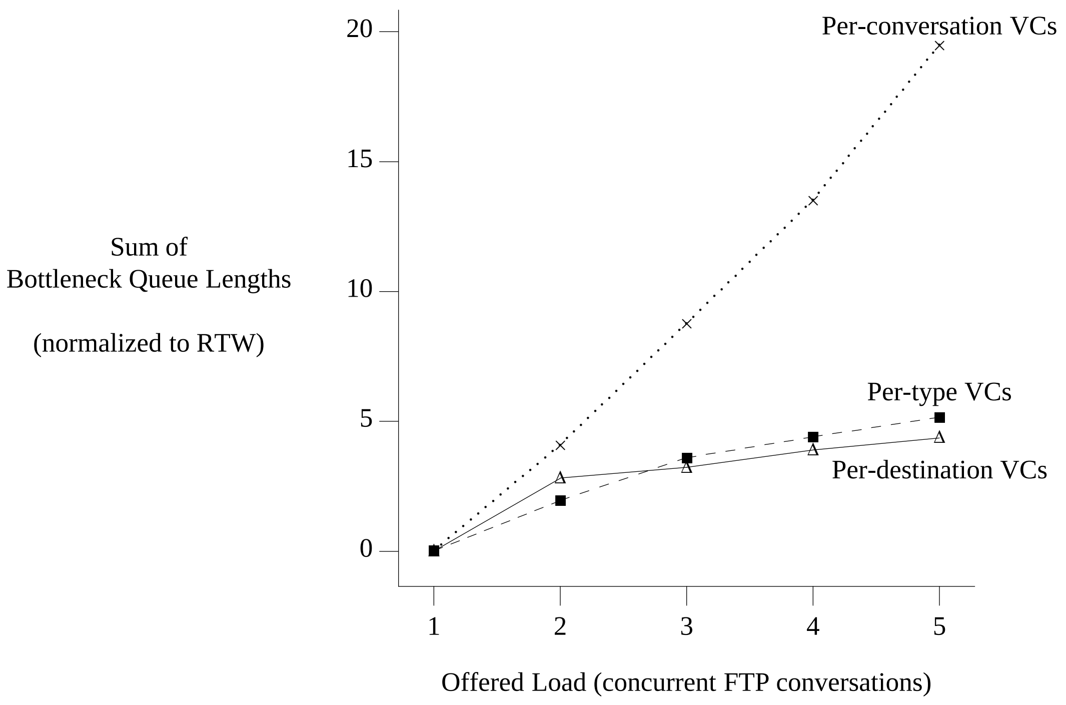

Figure 4.7. Bottleneck buffer consumption under different multiplexing policies

queueing and thus no buffers are consumed. As the network becomes more heavily loaded, buffer consumption under per-type VCs and per-destination VCs increases by one network round-trip window with each bulk transfer conversation added to the offered load. However, buffer consumption under per-conversation VCs rises as a multiple of the offered load.

This buffer consumption result may seem non-intuitive in a network without resource reservations. With all other factors being equal, there may appear to be no reason why any number of statistically multiplexed virtual circuits should consume more resources than another number. For an explanation of this apparent discrepancy, we turn to the behavior of the transport-level windows that regulate transmission on the hosts.

4.4.4.2. Behavior of Transport-Level Windows

We recall that transport-level protocols like modern TCP dynamically change their window size to adjust to perceived network conditions. The predominant algorithm for this purpose is the *slow-start* congestion control scheme [47].

The slow-start algorithm begins with a one-packet window and increases the window size up to a preset limit as it receives timely acknowledgements. It first increases the window size rapidly, doubling it with each acknowledgement, until a threshold is reached. It then increases the window size slowly, by one packet with each acknowledgment. The protocol also sets timers to measure how long acknowledgements take to return. It uses these measurements to maintain estimates of the mean and deviation of round-trip delays. If the time before a packet is acknowledged exceeds the current estimate of the mean by more than some multiple of the current estimate of the deviation, the protocol assumes the packet is lost and retransmits it. After every retransmission, the window size is reduced to one packet and the procedure repeats. All

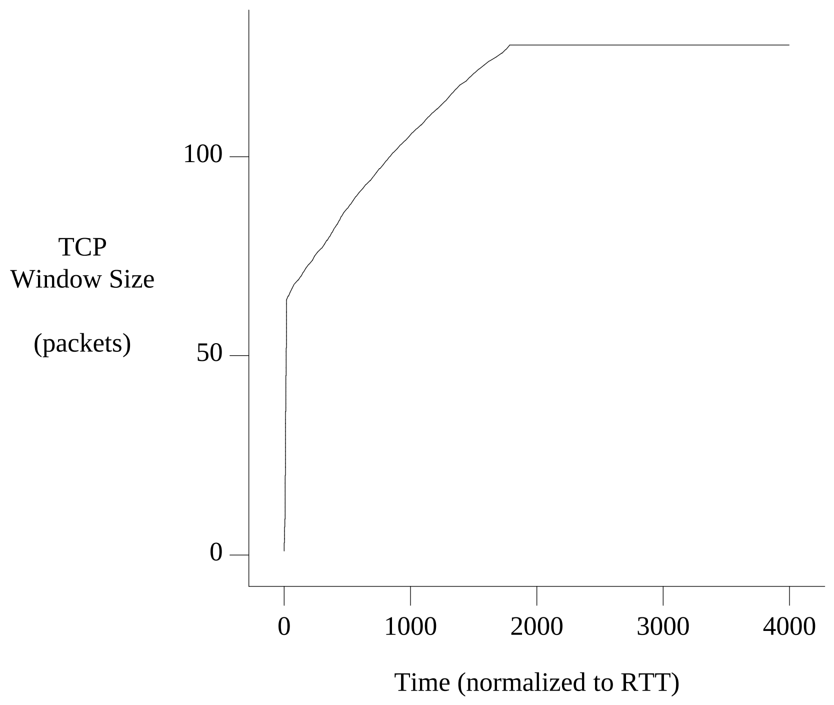

Figure 4.8. Behavior of flow control windows in FTP hosts under per-conversation VCs

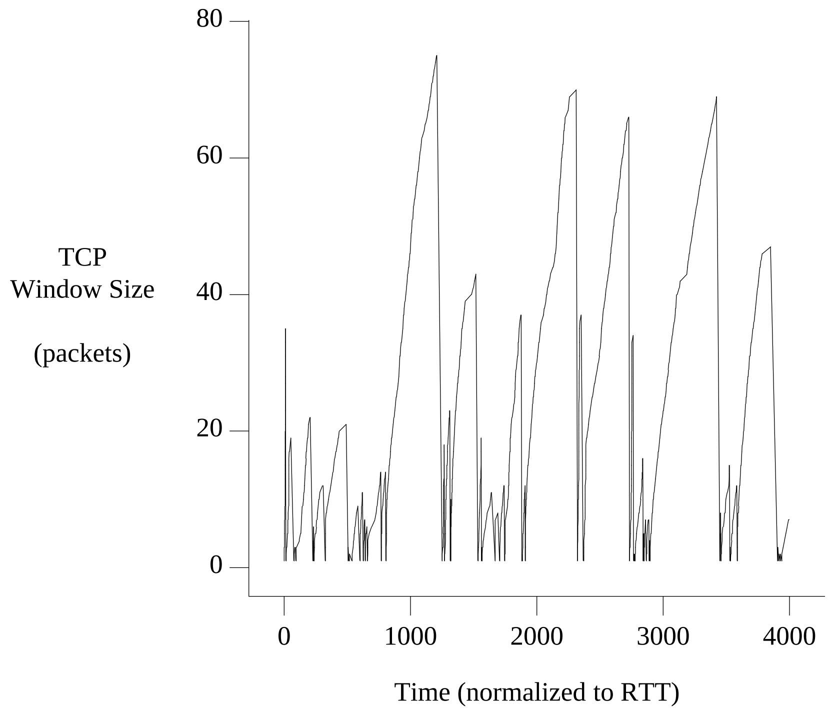

Figure 4.9. Behavior of flow control windows in FTP hosts under per-type VCs

the simulations reported here use the slow-start algorithm in the hosts.

Figures 4.8 and 4.9 show the TCP window size over time for FTP conversations, under per-conversation VCs and per-type VCs, respectively. The window behavior for per-destination VCs is very similar to that for per-type VCs and we leave it out of the discussion below. As shown, under per-conversation VCs, window sizes increase steadily until they reach their maximum value and remain there. Under per-type VCs, window sizes never reach the maximum before they shrink to one packet and resume growing.

Like the unfairness phenomena already described, this behavior is due to the different delay characteristics of the different multiplexing policies. Per-conversation VCs provide delays that vary relatively slowly for each conversation. This policy gives each conversation its own virtual circuit and queue, and services the set of queues on a round-robin basis. The results of slow-varying delay are as follows: the round-trip delay estimators in the hosts are able to track network behavior, acknowledgements return within the expected time interval, hosts do not time out or retransmit packets, and windows grow steadily to their maximum value and remain there. Predictable delays have already been suggested by Figure 4.3, where per-conversation VCs were shown to provide consistently low delays to a source with its own virtual circuit.

In contrast, per-type VCs provide delay that varies quickly for individual bulk transfer sources. This policy mixes traffic from different bursty sources in the same virtual circuit and queue, and serves these sources on a first-in first-out basis. The results of quickly-varying delay are as follows: the round-trip delay estimators in the hosts are not able to track network behavior, acknowledgments take longer than expected, hosts time out, and windows shut down. We substantiate our reasoning with additional simulation results below.

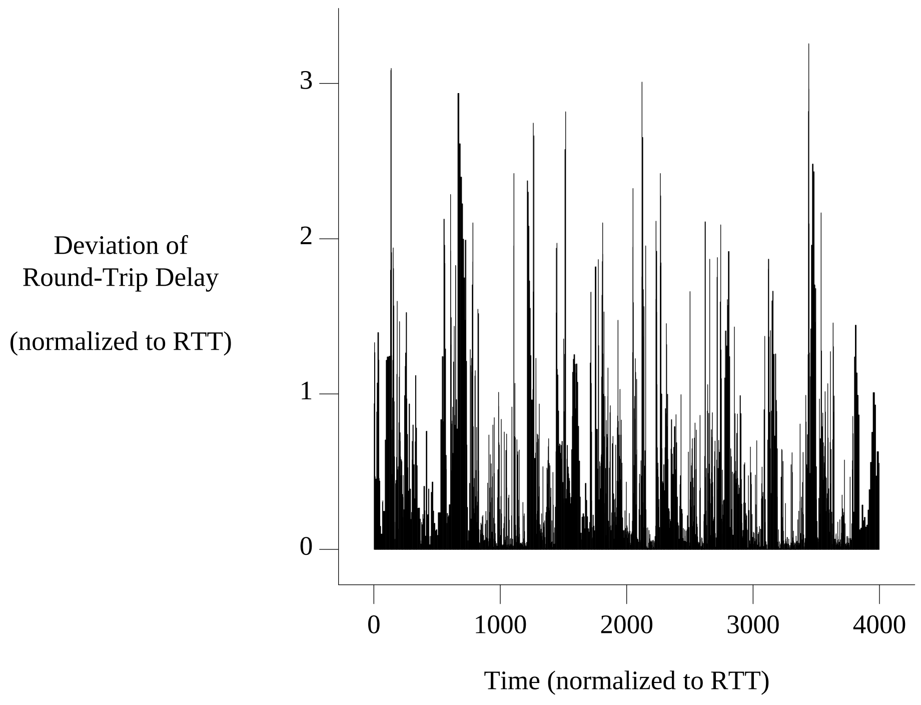

Figure 4.10. Deviation in delay under per-type VCs

Figure 4.10 shows, for the case of per-type VCs, the difference over time between the latest delay measured by a host and the current value of the host's delay estimator. We recall that our simulated network never drops packets, and therefore these variations in delay can only be due to variations in queueing delay. Figure 4.11 shows the sum of queue lengths in the router over time, again for per-type VCs. We note a high correlation between the peaks in these queue lengths, the peaks in delay deviation shown in Figure 4.10, and the drops in window size shown in Figure 4.9. Queue buildup causes unexpectedly long delays, which in turn cause windows to shut down.

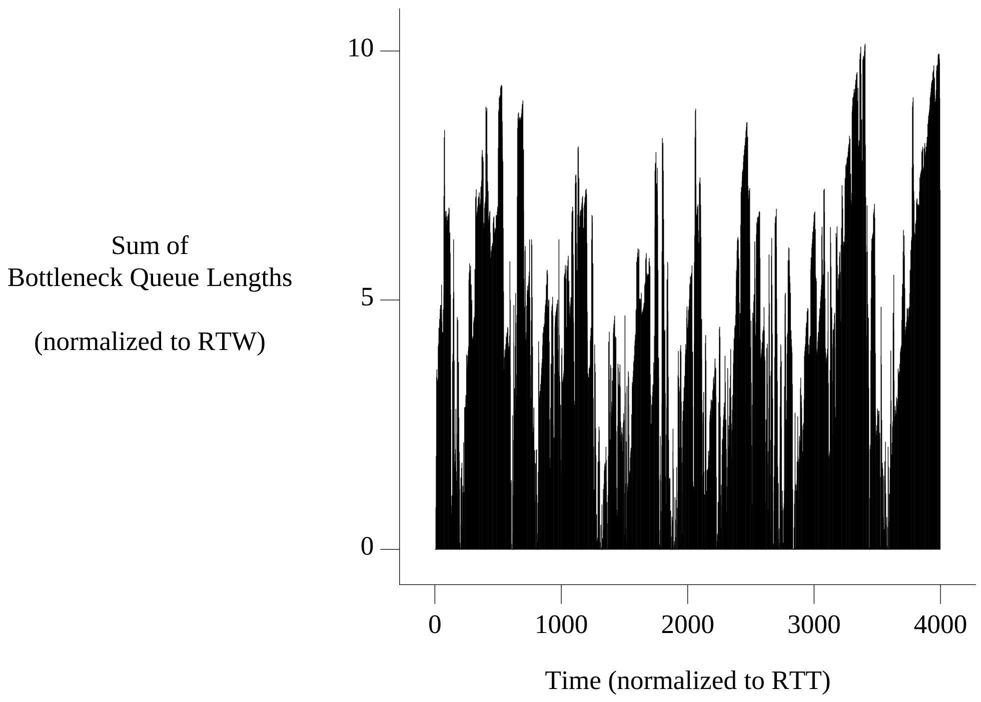

Figure 4.11. Sum of bottleneck queue lengths under per-type VCs

Returning to our unexplained buffer consumption results, we now see that they are caused by the interaction between queueing delays in the network and transport-level windows in the hosts. Under per-conversation VCs, bulk transfer sources maintain large windows and continue to send data in the face of congestion. Consequently, bottleneck queues continue to grow and buffer consumption reaches high levels. In the case of per-type VCs, bulk transfer sources shut down their windows and stop sending data in the onset of congestion. Consequently, bottleneck queues drain and buffer consumption remains at moderate levels.

Our results apply to real networks even though our simulated network has infinite buffer capacity. Real networks do not have an infinite supply of buffers. After congestion reaches a certain point, a real network would drop packets and retransmissions would occur, especially with per-conversation VCs because of the excessive queue buildup noted above. However, our results show that a network with per-type VCs can more easily manage congestion since it can avoid packet losses using moderate amounts of memory.

Although a network with per-type VCs and the requisite amount of memory is not completely free of retransmitted packets due to variations in delay, it is more stable than one with real packet losses. In support of this claim, we note that average throughput does not suffer under any multiplexing policy and that the network reaches 100% utilization under all multiplexing policies (see Figures 4.5 and 4.6). They key observation is that hosts perceive only isolated packet drops under per-type VCs. They do not perceive consecutive packet drops, which would cause exponential back-offs before the slow-start algorithm resumes transmission. Thus, although per-type VCs cause hosts to occasionally retransmit packets and temporarily shut down their windows, these hosts quickly resume growing their windows. In short, per-type VCs and the slow-start algorithm interact to allow bulk transfer hosts to obtain close to their fair share of throughput and to efficiently utilize the available bandwidth, while consuming moderate amounts of buffer memory.

4.4.5. Sensitivity to Transport-Level Window Size Limits

The simulations described above used a consistent upper limit for the size of transport-level windows on the hosts. However, our results apply to other values, with some qualification. Transport protocols like TCP with slow-start set an upper limit to the size of their flow control windows. We have seen these protocols dynamically grow their windows to make full use of the available bandwidth. That is, they grow their windows to avoid being window-limited. However, beyond a certain point the network and receiving host become saturated and an increase in window size does not improve throughput. As we have also seen, protocols can stop growing their windows to match these bandwidth-limited conditions. Nevertheless, setting an upper limit is a conservative measure that helps to maintain stability.

There is no direct relationship between window size limits on hosts and the intrinsic round-trip window of a network. Sending hosts set a limit that has as much to do with the expected capacity of the receiving host as with the capacity of the intervening network. The choice is made more difficult by the many different types of hosts (workstations, servers, supercomputers) and the many different types of networks (LANs, MANs, WANs) that comprise a large internetwork. In practice, hosts set oversize limits to avoid ever being window-limited, relying on the dynamic window mechanism to adjust to prevailing conditions. We ran our simulation experiments using a high limit for the size of the flow control window in the hosts (128 outstanding packets). Our oversize limit follows common practice, but raises concerns that our results do not extend to other scenarios.

To explore the sensitivity of our results to window size limits, we repeated our experiments with a smaller limit, one that more closely matches the intrinsic round-trip window of the network (16 outstanding packets). The effects of the multiplexing policies reported above are still apparent under these conditions. In particular, the interactive delay and bulk transfer throughput results carry over to the new conditions without change. Buffer consumption again rises faster with per-conversation VCs than with per-type and per destination VCs, but the ratio in slopes narrows from 5 with oversize windows to 2 with matched windows. In general, buffer consumption rises more slowly with smaller windows because hosts send data in smaller bursts. Less bursty arrivals at a queueing point result in smaller average queue lengths [60] [62].

Thus, the savings in buffer memory attributed to per-type VCs are not as dramatic with matched windows as with oversize windows. However, we recall that matching window sizes, although simple in the context of simulation, is difficult in the context of a large internetwork. Underestimating the true round-trip window that is in effect during a conversation will reduce buffer memory consumption regardless of multiplexing policy, but may window-limit the conversation and reduce its throughput.

4.4.6. Sensitivity to Link Speed

The simulations described above used a relatively low link speed (1.5 Megabit/second), but the exact choice is unimportant. We were careful to insure that our workload model is independent of link speed. Scaling the simulation to a low speed substantially reduced our computation requirements. For instance, we were able to bring about the multiplexing problems of interest by mixing an interactive conversation with only a small number of bulk transfer conversations (the network saturates with 3 to 4 simultaneous FTP conversations; we simulated up to 5). However, these multiplexing problems are not a function of the link speed per se but of the bottleneck link utilization, that is, of the load on the network. If we scale the network speed up or down and scale the offered load accordingly, our multiplexing results should remain the same.

To verify this hypothesis, we repeated our experiments with faster links (3 Megabit/second). As expected, we needed a correspondingly higher number of conversations to bring out the multiplexing problem (the faster network saturates with 7 to 8 simultaneous FTP conversations; we simulated up to 10). Under these conditions, all the multiplexing phenomena reported earlier repeat with the same magnitude under the new conditions. Therefore, our results scale with link speed.

4.5. Conclusions

We have presented a simulation study of traffic multiplexing issues at the entrance to wide-area networks. We evaluated schemes for mapping datagram streams to virtual circuits and queueing disciplines for scheduling datagrams onto virtual circuits. We pursued a methodology that included choosing appropriate configuration parameters, validating our simulator, and obtaining confidence intervals. Our simulation results quantify the tradeoffs between performance, fairness, and resource consumption that should govern multiplexing decisions.

We found that the best multiplexing policy is to establish one virtual circuit per traffic type and to provide round-robin service among virtual circuits. Per-conversation virtual circuits with round-robin service provide low interactive delay but consume excessive amounts of memory. Per-destination virtual circuits with first-in first-out service consume moderate amounts of memory but provide high interactive delay. Per-type virtual circuits with round-robin service combine the advantages of the other two policies while avoiding their disadvantages, with the exception of a fairness degradation in bulk transfer throughput. There are also less tangible benefits to opening only one virtual circuit per traffic type. In addition to the substantial memory savings described in this chapter, there are minor savings in routing and switching table space. For all these reasons, per-type VCs are superior to per-conversation and per-destination VCs.

These results suggest an attractively simple scheme for managing a virtual circuit network that carries datagrams: The network should establish a set of permanent virtual circuits, one for each type of traffic flowing between two network points of presence. The set of traffic types and thus the set of virtual circuits that must be supported is well-defined and changes slowly. These permanent virtual circuits avoid dynamic establishment delays and behave like the virtual circuits in our simulations.

We have compared throughput and delay to buffer consumption in the context of multiplexing datagrams over cell-based virtual circuits. In the same context, the next chapter addresses transmission efficiency, an issue that affects both the throughput obtained by applications and the efficient use of network bandwidth.

5. Transmission Efficiency

5.1. Introduction

Bandwidth is the most expensive resource in a long-haul network. The efficient use of bandwidth is thus an important consideration when multiplexing traffic at the entrance to wide-area networks. Network designers avoid transmission overhead to give users better access to available bandwidth. Transmission efficiency, or the ratio of useful bytes to total bytes carried by a network, measures how well networks achieve this goal. Protocols and techniques that improve efficiency should be considered when designing a network.

This chapter presents the transmission efficiency achieved by cell-based networks when transporting wide-area data traffic. Variable-length datagrams arrive at the entrance to a cell network and are fragmented into fixed-sized cells. Factors such as the size of incoming datagrams, the size of cells, and the size of protocol headers all affect transmission efficiency. This treatment of bandwidth consumption adds to our investigation of buffer consumption and other traffic multiplexing issues in the previous chapter.

We use Asynchronous Transfer Mode (ATM) networks as the prevalent example of cell-based networks. ATM is the multiplexing technique recommended for future Broadband Integrated Services Digital Networks (B-ISDN) [41] [72] [84], that is, high-speed networks that will mix traditional data traffic with voice, video, and other forms of non-traditional traffic. Many ATM networks are being designed and built [35], and many more will come into operation in the near future.

By wide-area data traffic we mean traffic carried by the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP) of the Internet family of protocols. The success of the Internet family of protocols suggests they will continue to be a significant portion of the traditional data traffic carried by wide-area networks. As discussed in Chapter 2, the Internet is the largest internetwork and is growing rapidly. We observed that more than 94% of wide-area Internet traffic is due to TCP and UDP. Future networks will carry audio and video traffic with different characteristics from current traffic. However, traditional forms of traffic will prevail for several years and will continue to be used for much longer. As a result, future B-ISDN networks based on ATM must efficiently support TCP and UDP traffic.

Transmission efficiency will remain an important issue as networks evolve. Granted, the performance effects of protocol overhead decrease as communication lines become faster. However, there remains a significant economic cost to wasted bandwidth. Historically, high-speed communication outside the local area has carried a high price. This situation may continue due to the high cost of upgrading land lines. Even if long-haul prices fall due to sharing of lines among many users, not everyone will benefit from the economies of scale. Consider an integrated services network that reaches to the home. Installing a high-speed access line between a home and the nearest network point of presence will continue to be costly. This cost will not be shared among many users. In some cases the old, slower line will not be upgraded because of economic considerations. Whether an access line is upgraded to high speeds or is kept at slower speeds, both the user and provider of the line will remain conscious of the quality of service provided for a certain price. It will always be necessary for networks to operate efficiently.

There are three major causes of transmission inefficiency. First, as application-level data flows through the network, protocols add header and trailer overhead. Internet protocols like TCP, UDP, and IP add headers to form a *datagram*, and network-level protocols add additional headers and trailers to from a *frame*. Second, some protocols add overhead to satisfy byte-alignment requirements for their headers and trailers. Finally, in cell-based networks, another source of inefficiency is the fragmentation of variable-length packets into fixed-length cells. When a packet is presented to an ATM network, it is separated into a sequence of cells. In general, the last cell in the sequence will be only partially filled with packet data. Any unused space in the last cell is padded with dummy bytes that add to the overall inefficiency. For a given stack of protocols, the extent of these three problems is highly dependent on the workload presented to the network, particularly on the size distribution of application data units.

We calculate three measures of transmission efficiency to isolate the effects of overhead contributed by three different levels in the protocol hierarchy. We define transmission efficiency as the ratio of useful bytes to total bytes carried by a network's communication lines, expressed as a percentage. Useful bytes can refer to bytes originally sent by an application program, or also include header and trailer bytes added by lower level protocols. First, for *application efficiency*, useful bytes refer only to application-level bytes; it is efficiency as viewed by an application program. Second, for *datagram efficiency*, useful bytes include application-level bytes and TCP, UDP, and IP header bytes; it is efficiency as viewed by a host transmitting IP datagrams. Third, for *frame efficiency*, useful bytes include datagram-level bytes and framing protocol bytes; it is efficiency as viewed by the framing protocol. Bytes inside ATM cell headers, adaptation headers, adaptation trailers, and cell padding are not considered useful in any of these measures.

Figure 5.1 illustrates some of the efficiency concerns addressed in this chapter. It shows datagram efficiency as a function of payload size, that is, the portion of an ATM cell used to carry higher-level data. The input to the calculation was a histogram of wide-area TCP-IP and UDP-IP datagram lengths extracted from the wide-area traffic traces described in Chapter 2. The ATM-related protocol overhead used was 28 bytes per datagram and 9 bytes per cell, corresponding to standard protocols described later in this chapter. The solid line represents the efficiency obtained by following standard ATM procedures, including padding partially-filled cells to the standard fixed size. The cross-marked line represents the efficiency obtained when this padding is not transmitted. The difference between the two lines is the efficiency lost to cell padding.

We observe the following:

- Efficiency responds abruptly to changes in payload size. The sharp drop in efficiency near 48 bytes of payload is due to the abundance of wide-area TCP-IP datagram lengths between 40 and 50 bytes. After ATM-related overhead, these short datagrams generate one full ATM cell and a second cell that is mostly padding. Moving from the 48-byte CCITT standard payload to a 54-byte payload increases efficiency by almost 10%. The previously proposed payload sizes of 32 bytes and 64 bytes both perform better than the size chosen for the standard.

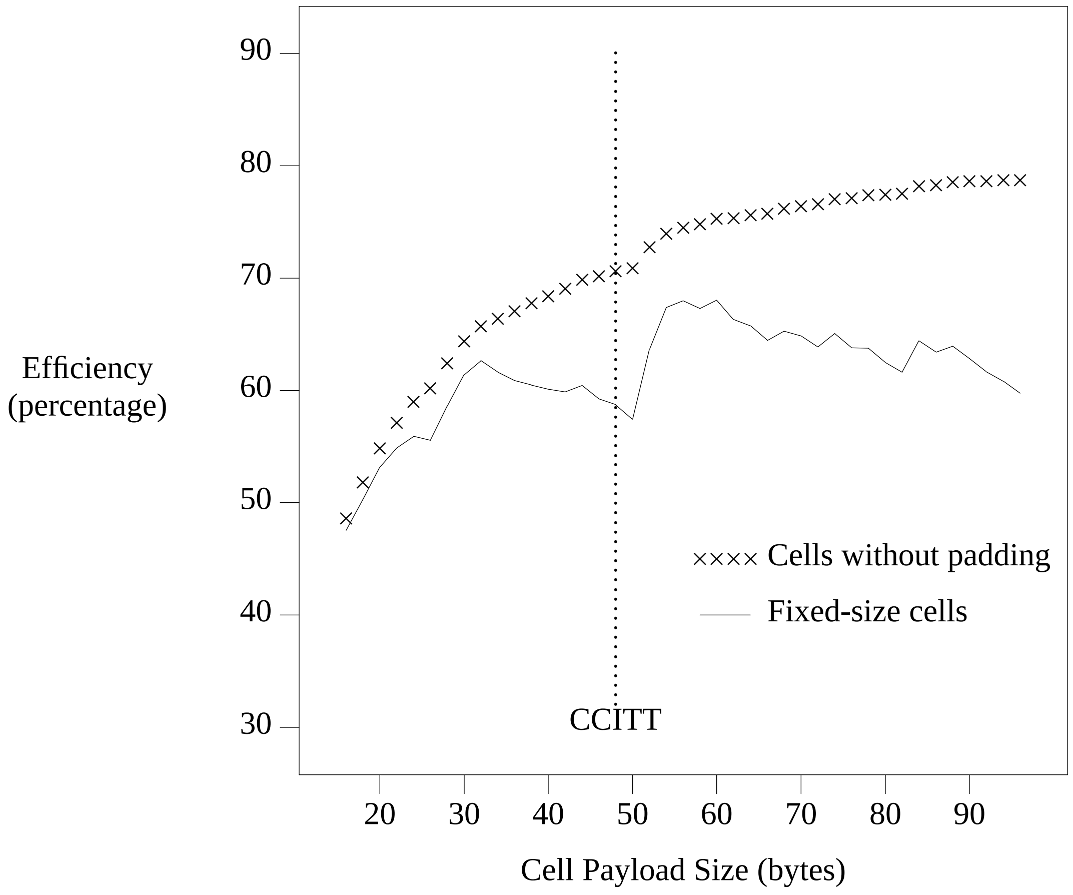

Figure 5.1. Efficiency of ATM Networks in Transporting Wide-Area Traffic

- Discarding cell padding not only reduces overhead but also smooths the efficiency curve. At the standard payload size, not padding partially-filled cells increases efficiency by 12%. As evident by the smoothness of the upper curve, the jaggedness in the lower curve is almost entirely due to cell padding.
- With padding, transmission efficiency as seen by a host is only 62.9% at the standard operating point and never rises above 72%. Under the same conditions, a similar calculation shows that efficiency as seen by an application program is considerably worse – it never rises above 49%.

These results suggest that ATM networks are inefficient in transporting wide-area data traffic. In this chapter, we investigate whether the phenomena evident in Figure 5.1 extend to other ATM-related protocol choices. We also study the effects on efficiency of non-standard compression techniques such as pad deletion. To insure an accurate traffic characterization, we drive our efficiency calculations with the measured traffic statistics described in Chapter 2.

The rest of this chapter presents our study in more detail. Section 5.2 surveys previous work in this area. Section 5.3 presents the transmission overhead introduced by Internet-related and ATM-related protocols. Section 5.4 presents a characterization of TCP and UDP datagram sizes derived from the traffic measurements described in Chapter 2. Finally, Section 5.5 discusses our efficiency calculations.

5.2. Previous Work

Kleinrock et. al. [61] studied line overhead in the ARPANET during the mid-1970's. They noted the critical dependence of network efficiency on traffic mix – they found that average transmission efficiency could be as low as 1% for single-character traffic and as high as 79% for file transfer traffic. They suggested that designers of future network protocols take more account of the effect of line overhead on network performance. We note that ATM networks also exhibit severe efficiency problems for an important class of traffic, and make a similar suggestion to designers of protocols for ATM networks.

DeSimone [23] calculated ATM transmission efficiency using a TCP and UDP traffic model based partly on network measurements. He derives an application mix and the corresponding packet length distribution from a combination of previous network measurement studies [11] [37] and his expectations of how networks will evolve. In contrast, we drive our calculations directly with TCP and UDP packet lengths measured in the wide-area Internet. Important differences between his and our workload models include our greater ratio of interactive to bulk transfer traffic and our smaller average packet size. He calculates the same three measures of efficiency that we call application, datagram, and frame efficiency. He presents results for one of the three protocol combinations and two of the three compression techniques we discuss here. Because of differences in assumed workload, he finds the effects of discarding ATM cell padding less significant than we do – he saw an improvement in efficiency of only 2 to 3%; we see 4 to 12%. Similarly, after compressing TCP-IP headers he saw improvements of 0 to 7%; we see 5 to 12%. As a result, we draw different conclusions regarding the merits of these two compression techniques.

Cidon et. al. [13] critique ATM from a data communications perspective. They conclude that for traditional datagram applications, networks that carry variable-sized packets are better than those that carry small fixed-size cells because of several considerations, including transmission efficiency. We note that future networks must support real-time traffic as well as traditional traffic, and that cells can serve the needs of these varied types more easily than packets. Given that many cell networks will exist, we evaluate their transmission efficiency in detail and explore ways to improve that efficiency.

5.3. Protocol Overhead

In an ATM network transporting TCP and UDP traffic, protocols introduce transmission overhead at all levels in the hierarchy. This overhead takes three forms: header and trailer bytes, alignment bytes, and padding bytes. Header and trailer overhead are fixed per data unit, while alignment and padding overhead are variable. In our calculations, we consider only overhead added below the application level. We ignore overhead added to user data by applications such as TELNET and FTP.

After an application data unit is prepared for transmission, Internet protocols add their share of overhead while forming a datagram, as shown in Figure 5.2. In the absence of rarely-used protocol options, TCP adds 20 bytes of header at the transport level and UDP adds 8. IP adds another 20 bytes at the internetwork level.

The network then forms a frame by wrapping each datagram with an optional frame header, trailer, or both, as shown in Figure 5.2. Examples of proposed protocols at the frame level are AT&T's Frame Relay Service for Xunet 2 (XFRS) [52] [35] and Bellcore's Switched Multi-Megabit Data Service (SMDS) [103]. These protocols add to the services provided by the

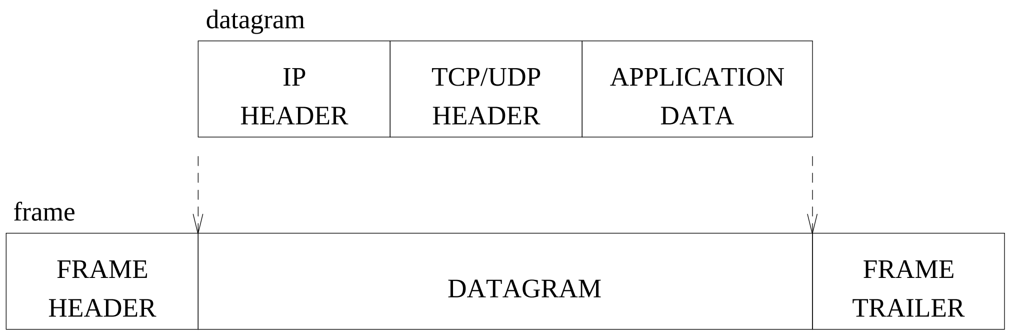

Figure 5.2. Frame Format

lower-level protocols. All three include in the frame the length of the enclosed datagram so that the datagram can be reconstructed from a stream of fixed-length cells. In addition, XFRS provides sequencing to detect the loss of full frames and detects bit errors in both the datagram and the frame trailer. This combination of techniques allows XFRS to handle both the corruption of single bits and the loss of full ATM cells. SMDS detects similar types of errors and supports a larger source and destination address space than IP does. XFRS adds 12 bytes of trailer and SMDS adds 24 bytes of header and 4 bytes of trailer per frame. An XFRS trailer must be also 8-byte aligned and reside entirely in one ATM cell, which may introduce additional overhead in the form of alignment bytes.

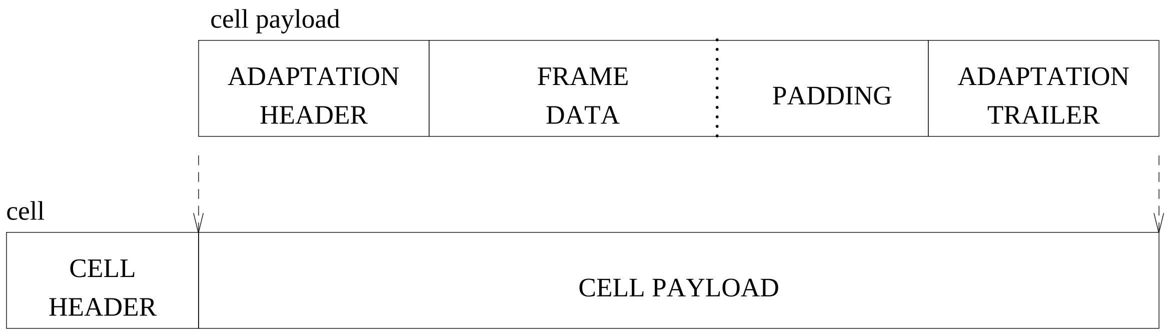

Figure 5.3. Cell Format

An ATM network will then break up a frame into fixed-length *cells* whose format is shown in Figure 5.3. The CCITT ATM standard calls for 53-byte cells with 5 bytes of header and 48 bytes of payload. The ATM header provides virtual circuit identifiers and single-bit error detection on the contents of the cell header alone. An optional adaptation layer inserts in each payload a header, a trailer, or both, leaving the remaining space to carry frame data. Examples of proposed adaptation protocols are Bolt, Beranek and Newman's Segmentation and Reassembly (SAR) protocol [28] and IEEE's 802.6 segmentation and reassembly layer [26]. These protocols add services not provided by the bare ATM standard, in particular error detection and correction. SAR provides sequencing to detect the loss of full cells, and also detects bit errors in the payload; 802.6 provides similar error-detection. In order to account for partially-filled cells, 802.6 also specifies the number of higher-level bytes actually carried in the payload; SAR leaves this responsibility to higher levels. SAR adds 3 bytes of trailer and 802.6 adds 2 bytes of header and 2 bytes of trailer.

Table 5.1 summarizes the fixed overhead contributed by the services and protocols just discussed.

Level	Data Unit Affected	Protocol	Bytes of Overhead
Transport	Datagram	TCP UDP	20 8
Internetwork	Datagram	IP	20
Framing	Frame	SMDS XFRS	28 12
Adaptation	Cell	802.6 SAR	4 3
Data Link	Cell	ATM	5

Table 5.1. Fixed Protocol Overhead

In addition to header, trailer, and alignment overhead, ATM networks introduce fragmentation overhead in the form of padding bytes. When the length of a frame is not an integral number of usable payload lengths, the last cell for that frame is padded to the required fixed length. The amount of overhead contributed by fragmentation is equal to the number of bytes left unused in the last cell for the frame.

Finally, the physical layer also introduces overhead. For example, the DS3 transmission standard offers a raw 44.7 Megabits/second. The IEEE 802.6 standard calls for 12 ATM cells per 125-microsecond DS3 frame, which yields 40.7 Megabits/second, or 91% transmission efficiency [23]. The effects of the physical layer are ignored in our efficiency calculations.

5.4. Datagram Length Characterization

Since transmission efficiency is strictly a function of the number of bytes carried by a network, the efficiency of ATM networks in transporting wide-area Internet traffic is critically dependent on the length distributions of IP datagrams. Figure 5.4 is a histogram of TCP application data lengths extracted from the network traffic traces described in Chapter 2, and Figure 5.5 is a similar histogram for UDP. These histograms depict data from the University of California at Berkeley, but trace data from other sites produce very similar histograms (see Appendix A).

TCP is the dominant protocol on the Internet – it is responsible for more than 80% of all packets, as we saw in Chapter 2. Consequently, the distribution shown in Figure 5.4 dominates our efficiency calculations. It is bimodal, with very large peaks near 0 bytes and 512 bytes. The 0-byte peak represents almost 40% of all TCP-IP datagrams and is due to acknowledgements and other control information traveling without accompanying data.† The next largest peak represents 30% of all TCP-IP datagrams; it is due to data units between 0 and 10 bytes exchanged by interactive network applications like TELNET and RLOGIN. The 512-byte and

† A majority of 40-byte TCP-IP datagrams contain only acknowledgements. The fact that TCP is seldom able to piggy-back acknowledgements on reverse-flowing data suggests that TCP connections between wide-area applications are predominantly one-way. If there was more reverse-flowing data, piggy-backed acknowledgements would improve efficiency.

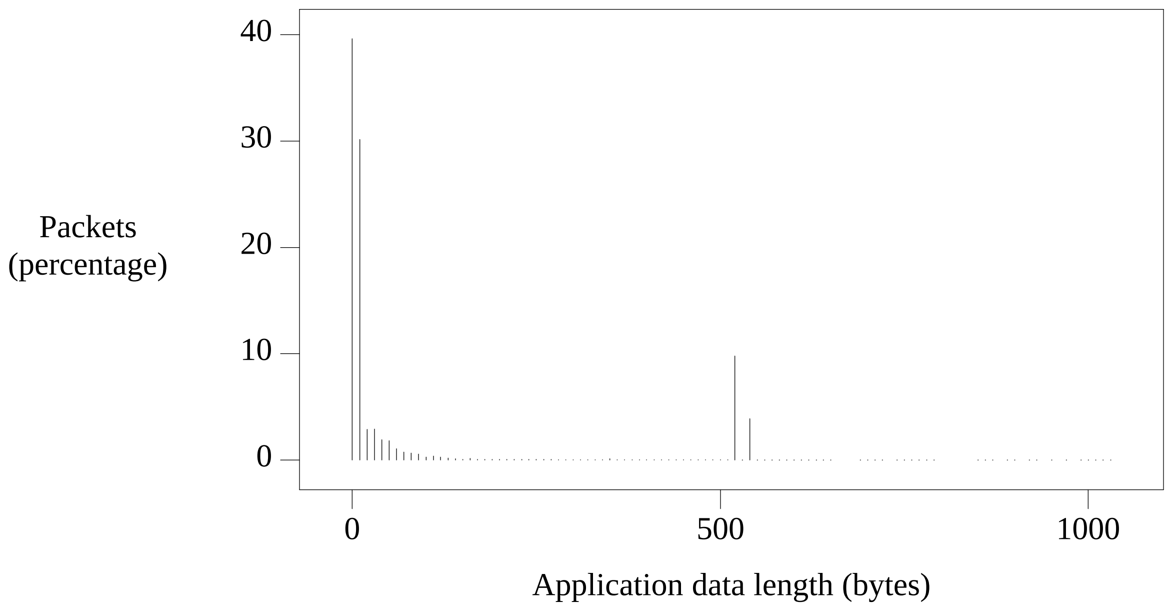

Figure 5.4. Histogram of TCP application data lengths

536-byte peaks are due to the Maximum Segment Size (MSS) for wide-area networks configured in most Internet hosts.‡ Figure 5.5 shows similarly well-defined peaks in UDP data lengths.

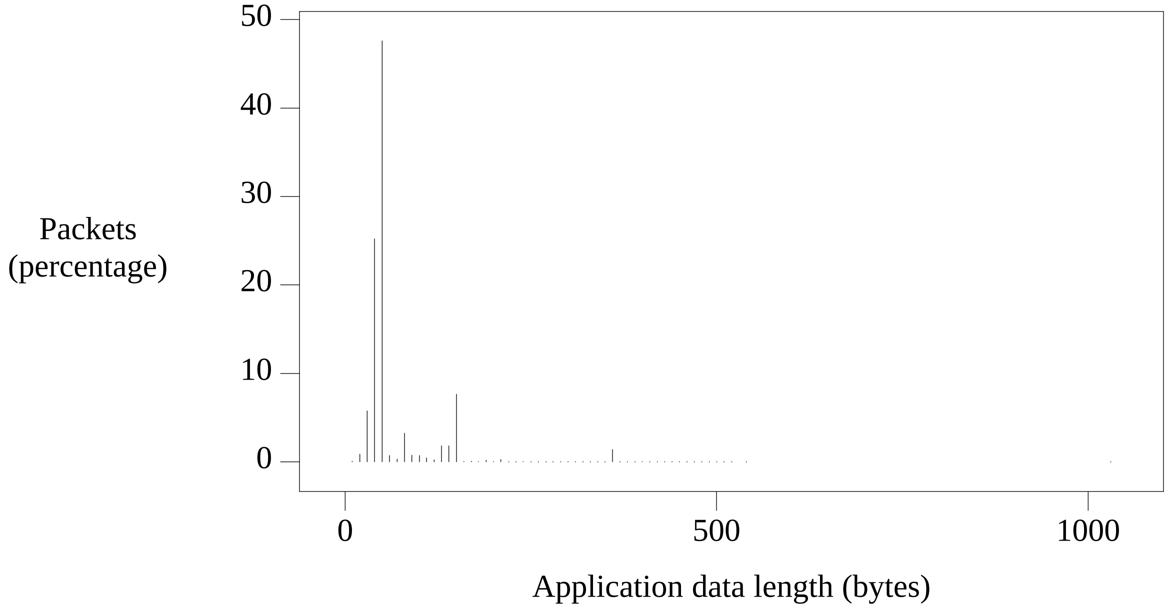

Figure 5.5. Histogram of UDP application data lengths

‡ To avoid IP fragmentation [54], the TCP MSS is set to roughly 40 bytes less than 576, the commonly-used IP Maximum Transmission Unit (MTU) for wide-area networks. The choice of 576 is historical and should be reconsidered since the actual MTU of the NSFnet backbone is at least 1500 bytes. Methods for setting a more accurate IP MTU are discussed in Reference [74]. A higher MTU would result in higher efficiency.

These length characteristics have a strong negative impact on efficiency. As we observed, TCP-IP datagrams are often the smallest possible, and both TCP and UDP datagrams are predominantly small. Almost 70% of TCP-IP application data units are smaller than 10 bytes, and more than 84% are smaller than 256 bytes. Almost 80% of UDP application data units are smaller than 48 bytes, and more than 97% are smaller than 150 bytes. In general, the efficiency of a network carrying small data units is more adversely affected by protocol overhead than the efficiency of a network carrying large ones.

The abundance of TCP application data unit lengths between 0 and 10 bytes has a particularly strong effect on ATM efficiency. In ATM networks, the loss of efficiency due to cell padding depends on how application data lengths map into cell payload lengths. TCP-IP datagrams with 0-10 bytes of application data are 40-50 bytes long. In comparison, the standard ATM cell payload is 48 bytes long. After framing and adaptation overhead are added, these short TCP-IP datagrams often fill one complete 48-byte payload and very little of another. Cell padding overhead becomes significant when for every two cells transmitted one consists almost entirely of padding. This phenomenon was responsible for the large efficiency loss near 48 bytes of payload evident in Figure 5.1.

5.5. Transmission Efficiency

5.5.1. Efficiency of Standard Procedures

Table 5.2 shows the transmission efficiency of four protocol combinations. The SMDS-802.6 combination is a proposed standard, and XFRS is intended for use in Xunet 2 [35]. XFRS does not need an adaptation layer, thus our use of the XFRS-none combination, which incidentally shows the effects of doing without an adaptation layer. SAR is a recent proposal that can be used alone or with SMDS or some other framing protocol. We include the none-SAR combination to show the effects of doing without a framing layer. The overhead introduced by these protocols was discussed in Section 5.3. All calculations were driven directly by the measured application data lengths discussed in Section 5.4. Throughout, we used a 53-byte cell with a 5-byte header and a 48-byte payload.

Figure 5.6 displays the same information in graphical form. Each horizontal bar represents one protocol combination. From left to right, the three subdivisions in each bar represent application, datagram, and frame efficiency, respectively. Since the none-SAR protocol combination introduces no framing overhead, its datagram and frame efficiency results are identical.

As expected, protocols with less overhead are more efficient than protocols with more overhead. However, due to the interaction of datagram lengths and cell padding, efficiency is insensitive to large variations in certain protocol design dimensions, and sensitive to small variations in others. Table 5.2 shows that moving from SMDS-802.6, with 28 bytes of per-frame overhead and 4 bytes of per-cell overhead, to XFRS-none, with only 8 bytes of per-frame overhead, improves application efficiency by only 4.6%. In contrast, Figure 5.1 shows how a change in payload size from 50 to just 54 bytes can increase datagram efficiency by 10%.

Datagram header overhead, not ATM network overhead, is responsible for the difference between application and datagram efficiency. For the protocol configurations shown in Table 5.2, the overhead intrinsic to TCP, UDP, and IP is responsible for 18.6 to 24.3% of the 47.7 to 59.9% overall loss in application efficiency. ATM-related protocols and procedures are not to blame for this portion of the efficiency loss. However, ATM networks could make use of non-standard compression techniques to improve this aspect of network performance without

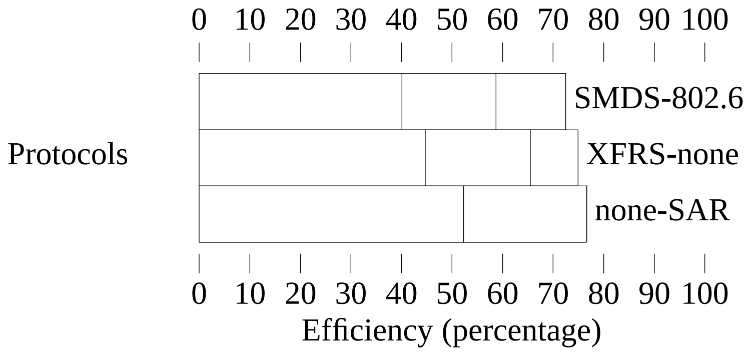

Figure 5.6. Efficiency of Different Protocols Using Standard ATM Procedures (The three non-zero vertical lines for each protocol combination represent, from left to right, the application, datagram, and frame efficiency for that combination.)

Protocol		% Efficiency		
Framing	Adaptation	Application	Datagram	Frame
SMDS	802.6	40.1	58.7	72.5
XFRS	none	44.7	65.5	74.9
none	SAR	52.3	76.6	76.6

Table 5.2. Efficiency of Different Protocols Using Standard ATM Procedures

modifying the Internet protocols running on the end hosts. The following discussion addresses this and other compression issues.

5.5.2. Effects of Non-Standard Compression Techniques

Figures 5.7 through 5.9 and Tables 5.3 through 5.5 show the efficiency obtained when an ATM network uses three non-standard compression techniques in isolation and in tandem. These techniques make use of several types of redundancy in the protocol hierarchy. Cell padding carries no useful information and is by definition redundant. There is also redundancy between TCP-IP headers in consecutive datagrams for the same TCP connection, and between cell headers in consecutive cells for the same frame. Finally, there is redundancy between a reliable transport protocol like TCP and proposed framing and adaptation protocols.

The first technique, *pad deletion*, deletes padding bytes from partially filled cells and transmits the resulting variable-length cells. The second technique, *cell header suppression*, inhibits the headers for all but the first cell in a frame. It transmits all cell payloads for a frame in a burst accompanied by only one cell header. The third technique, *datagram header encoding*, compresses TCP-IP headers through differential encoding. It takes advantage of the predictable changes between successive headers for the same TCP connection. This encoding scheme has been previously demonstrated to achieve a 10-to-1 compression ratio of TCP-IP headers [48]. In our calculations, we assume the same compression ratio for 40-byte TCP-IP headers but leave 28-byte UDP-IP headers intact.

Although the transmission efficiency advantages of these compression techniques are evident in Figures 5.7 through 5.9 and Tables 5.3 through 5.5, two concerns remain: First, can the techniques be used without violating the ATM standard? Second, can they be implemented

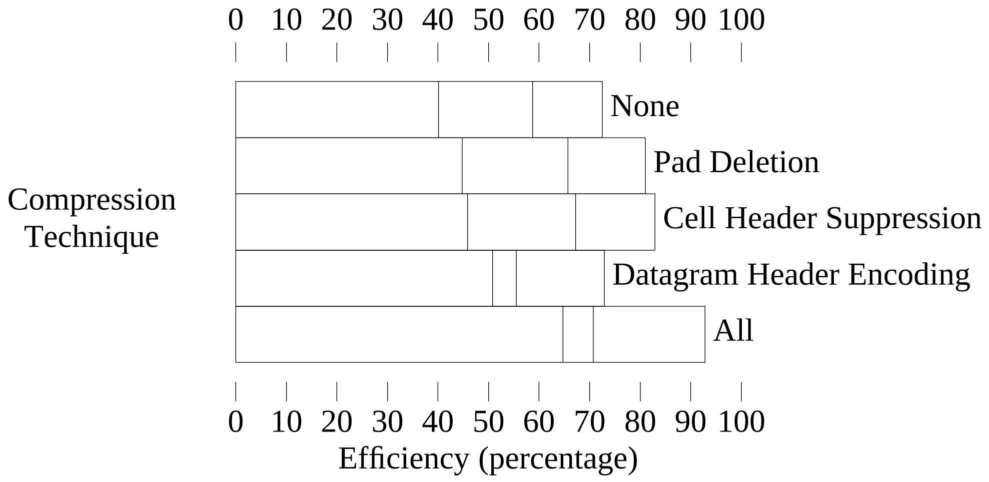

Figure 5.7. Efficiency of the SMDS-802.6 Protocols Using Compression Techniques (The three non-zero vertical lines for each compression technique represent, from left to right, the application, datagram, and frame efficiency for that technique.)

Compression Technique	% Efficiency		
	Application	Datagram	Frame
None	40.1	58.7	72.5
Pad Deletion	44.8	65.7	81.0
Cell Header Suppression	45.8	67.2	82.9
Datagram Header Encoding	50.8	55.5	72.9
All	64.7	70.7	92.8

Table 5.3. Efficiency of the SMDS-802.6 Protocols Using Compression Techniques

cheaply? We address these two issues by discussing how existing networks can implement these techniques. First, we note that all three techniques are realizable without violating the ATM standard in the interface presented to network clients. ATM is merely a service definition – it defines an interface, not an implementation technique. The ATM network can transform data to the compressed format once the data is inside the network, and convert it back to the standard format before it leaves the network. Second, we show how these compression techniques can be implemented with low hardware, software, and processing overhead.

Pad deletion and cell header suppression have been successfully implemented in 45 Megabit/second hardware for Xunet 2, an experimental wide-area ATM network that spans the continental United States [35]. Pad deletion and cell header suppression are available as an optional framing format in which the data for each frame is carried as a consecutive train of compressed cells. This non-standard format is used only in the 45 Megabit/second trunks used for long-haul transmission, since they are a scarce and expensive resource. The ATM standard is maintained in other parts of Xunet 2 where transmission efficiency is not a concern, for example in the 500 Megabit/second switch backplane, which is over-engineered to avoid congestion.

In the trunk interfaces of Xunet 2 switches, cell header suppression can be implemented with a 2-byte register to hold the last ATM Virtual Circuit Identifier (VCI) seen, and a comparator to match new VCIs with the last VCI seen. When the first uncompressed cell for a frame

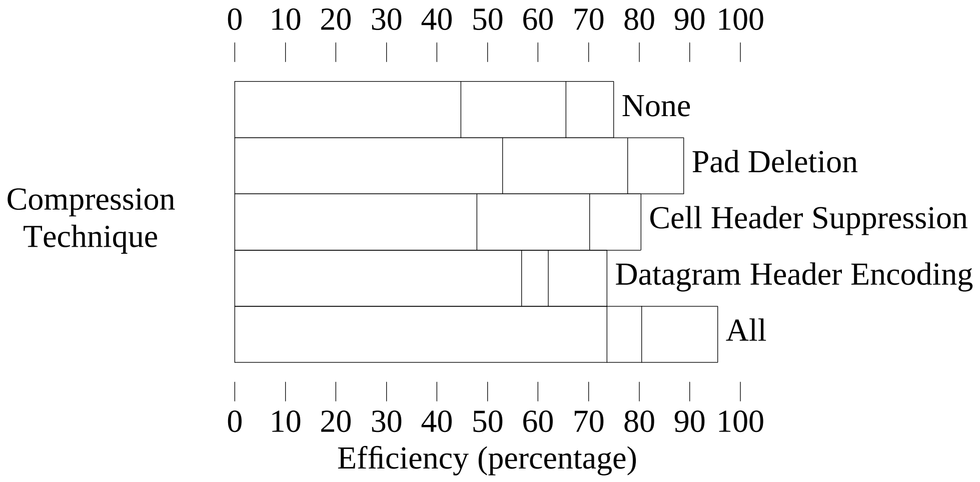

Figure 5.8. Efficiency of the XFRS-none Protocols Using Compression Techniques
(The three non-zero vertical lines for each compression technique represent, from left to right, the application, datagram, and frame efficiency for that technique.)

Compression Technique	% Efficiency		
	Application	Datagram	Frame
None	44.7	65.5	74.9
Pad Deletion	53.0	77.7	88.8
Cell Header Suppression	47.9	70.2	80.3
Datagram Header Encoding	56.7	62.0	73.6
All	73.6	80.5	95.5

Table 5.4. Efficiency of the XFRS-none Protocols Using Compression Techniques

arrives for transmission, the VCI is saved and the full cell is prepared for transmission. If subsequent uncompressed cells with matching VCIs arrive, the cell header is suppressed and only the payload is transmitted. To expand the compressed frame at the receiving trunk interface, the VCI in the first cell header is again saved. As subsequent 48-byte payloads are assembled, a new cell header is dynamically manufactured using the saved VCI and then prepended to the payloads. The frame length information provided by XFRS helps to identify the last payload for a frame.

A drawback of cell header suppression is that once a cell has been added to a compressed train, the cell cannot be independently transmitted. This grouping can preclude some of the multiplexing policies explored in Chapter 4, for example cell-by-cell round robin among virtual circuits. However, we note that compressed trains are formed immediately before cells are transmitted over a trunk and are disassembled immediately after they are received. Thus, multiplexing techniques that act on independent cells can still function in other parts of the network.

Pad deletion can be pipelined with header suppression in Xunet 2 trunk interfaces. It requires a 48-byte first-in first-out (FIFO) buffer to prepare arriving payloads for transmission, and a counter to keep track of how many bytes of padding are in a payload. The counter is initialized to zero before each uncompressed payload is processed for transmission. As the bytes of a payload arrive, they are buffered in the FIFO. The counter is incremented whenever a zero byte arrives, and reset to zero whenever a non-zero byte arrives. When the full payload is

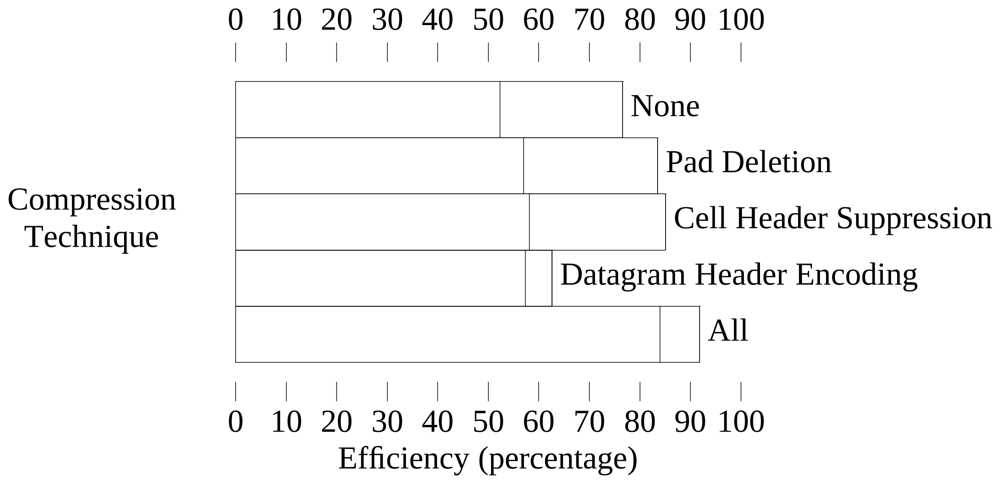

Figure 5.9. Efficiency of the none-SAR Protocols Using Compression Techniques (The left and right non-zero vertical lines for each compression technique represent the application and datagram/frame efficiency, respectively, for that technique.)

Compression Technique	% Efficiency		
	Application	Datagram	Frame
None	52.3	76.6	76.6
Pad Deletion	57.0	83.5	83.5
Cell Header Suppression	58.1	85.1	85.1
Datagram Header Encoding	57.3	62.6	62.6
All	84.0	91.8	91.8

Table 5.5. Efficiency of the none-SAR Protocols Using Compression Techniques

processed, the counter holds the number of consecutive zero bytes at the end of the payload. In the case of the last payload for a frame, these trailing bytes are considered padding and are not transmitted, resulting in a variable-length payload. To regenerate fixed-length payloads at the receiving trunk interface, data is fragmented into 48-byte payloads until the last, possibly incomplete, fragment is reached. This last fragment is then padded with zeros to a length of 48 bytes, if necessary. Again, the frame length information provided by XFRS helps to identify the last payload for a frame.

Datagram header encoding is currently used in the Internet by hosts connected to low-speed serial lines. Predictive differential encoding of TCP-IP headers has been implemented with minor modifications to the Unix kernel operating system kernel. The nature of these modifications are documented in Reference [48]. They are also included in the Serial Line IP (SLIP) software distribution that has been used for several years throughout the Internet. The same techniques are applicable within an ATM network if each TCP connection maps to a separate virtual circuit in a connection-oriented ATM network. However, they may not be feasible if multiple TCP connections are multiplexed onto a single virtual circuit, as explored in Chapter 4.

5.5.3. Summary of Efficiency Results

In addition to the results shown in Figures 5.6 through 5.9 and Tables 5.2 through 5.5, we calculated efficiency for all combinations of two or more compression techniques. For XFRS, we also compared the efficiency of the proposed XFRS with trailer alignment to that of a modified XFRS without alignment. Finally, we produced curves of efficiency versus cell payload size similar to those in Figure 5.1 for our range of protocol combinations and efficiency measures. From all these results, we reach these conclusions:

- Transmission efficiency is a function of cell payload size, protocol overhead, and traffic mix. The impact of cell fragmentation is sensitive to small changes in certain design dimensions and insensitive to large changes in others.
- Pad deletion can improve application efficiency by 4.7 to 8.1%, datagram efficiency by 6.9 to 11.9%, and frame efficiency by 6.9 to 13.9%.
- Cell header suppression can improve application efficiency by 3.2 to 5.9%, datagram efficiency by 4.7 to 8.8%, and frame efficiency by 5.4 to 10.4%.
- Datagram header encoding can improve application efficiency by 5.0 to 12.7%. In this case, datagram and frame efficiency may drop because compressing datagram headers subtracts from the numerator in those efficiency calculations.
- Simultaneously applying all three compression techniques improves application efficiency by 24.6 to 34.0%. Datagram efficiency improves by 11.9 to 21.1% and frame efficiency by 15.2 to 24.4%, but these figures also include the negative effects of datagram header compression on these calculations.
- For XFRS-none, aligning the frame trailer so that it is on an 8-byte boundary and fits entirely in the last cell causes less than 1% loss in application and datagram efficiency. Frame efficiency with alignment is better by almost 3% because the larger aligned frames add to the numerator in that efficiency calculation.
- For XFRS-none, the combination of pad deletion and cell header suppression can raise application efficiency to 57.5%, datagram efficiency to 84.4%, and frame efficiency to 96.5%, for improvements of 12.8%, 18.9%, and 21.6%, respectively. This combination is used by Xunet 2's optional compressed framing format, and can translate to substantial bandwidth savings on that network's 45 Megabit/second long-haul trunks.

5.6. Conclusions

We have compared the transmission efficiency obtained by different ATM-related network protocols when transporting wide-area data traffic, using both standard procedures and a range of non-standard compression techniques. Throughout, we found that ATM efficiency is alarmingly low for many proposed protocols and procedures, and that efficiency has variable sensitivity to changes in protocol overhead due to the effects of cell padding. We conclude that the standard payload size of 48 bytes is particularly bad for an ATM network that carries traditional types of traffic. The choices and compromises made by the standards bodies were in this case unfortunate.

Given that the ATM standard is fixed, we are not free to change its basic service. However, we are free to select many other network parameters. For example, the bare ATM service does not detect corrupted data. Some framing and adaptation protocols address single-bit errors, some address the loss of an integral number of cells, and some address both types of corruption.

The benefits of providing a rich framing or adaptation service should be compared to the efficiency lost to larger headers and trailers.

Our calculations show that reducing the overhead of related adaptation and framing protocols improves efficiency, sometimes dramatically due to the variable sensitivity already mentioned. We have also shown that compression techniques can moderately improve efficiency when used in isolation, and significantly improve efficiency when used in combination. Moreover, these techniques can be implemented without adding undue complexity to the network. These results should be taken into account when designing future networks, and transmission efficiency should be among the principal tradeoffs considered when engineering a cell-based network.

6. Conclusions

In this dissertation, we have presented work in the areas of traffic characterization, network simulation, multiplexing policies, and transmission efficiency. This chapter concludes the dissertation. It summarizes our contributions, points out the applicability of our results, suggests areas for future work, and comments on the benefits of our methodology.

6.1. Summary of Contributions

We have produced results in five main areas: wide-area traffic traces, characteristics of application-level conversations, workload models for wide-area internetwork simulation, policies for multiplexing traffic at the entrance to wide-area networks, and transmission efficiency of cell-based networks. We summarize these contributions below.

First, we gathered traces of several days of continuous wide-area network activity at a number of sites in the Internet. These traces contain a record of every packet flowing through the local-area to wide-area network junction at their respective sites. Each record includes the raw transport-level and network-level protocol headers from the packet, and a high-resolution timestamp of the packet's arrival at the tracing instrument. We have upwards of 75 million such records saved on magnetic tape. No previous measurements of wide-area Internet traffic combine raw data for every packet and high-resolution timestamps. Our traces thus contain a wealth of detailed information that is not readily available from other sources.

Second, we extracted from the traces important characteristics of application-level conversations for each major type of wide-area network traffic. We found that FTP, NNTP, SMTP, TELNET and RLOGIN account for the majority of packets, bytes, and conversations in current wide-area traffic. We separated these traffic types into two broad categories, bulk transfer and interactive. Examples of the former are FTP, NNTP, and SMTP, and examples of the latter are TELNET and RLOGIN. We collected per-conversation histograms of the number of bytes transferred, the number of packets transferred, packet size, packet interarrival time, duration, and other statistics. Regarding bulk transfer conversations, we found that they are smaller than previously assumed, are bidirectional rather than unidirectional, and include small packets in addition to large packets. Regarding interactive conversations, we found that they account for a higher percentage of traffic than previously believed, are asymmetrical rather than symmetric, and include large packets in addition to small packets. These observations may affect results from previous studies of network performance.

Third, we derived from our traces a new empirical workload model for driving wide-area internetwork simulations. We produced a generalized model for two major classes of network application, bulk transfer and interactive. The model reproduces behavior specific to each application by sampling measured probability distributions through the inverse transform method. Our model is realistic, in that it accurately reproduces characteristics of real traffic, and efficient, in that it is suitable for simulations of high-speed wide-area netwoks without consuming inordinate amounts of computing resources. Most importantly, unlike other trace-based traffic models, our model is network-independent. It is applicable to network conditions other than the ones prevalent when the measurements were taken.

Fourth, we evaluated policies for multiplexing datagrams over virtual circuits. We showed that networks should establish one virtual circuit per type of traffic flowing between two network points of presence, not one virtual circuit per conversation or one per pair of such points. In addition, networks should provide round-robin service to the virtual circuits sharing a communication link, not first-in first-out service. This policy provides minimum delay to interactive applications and maximum throughput to bulk transfer applications. This policy also consumes modest amounts of buffer memory at the bottleneck point. These buffer consumption benefits come at a moderate loss of fairness in the throughput provided to the bulk transfer conversations sharing a virtual circuit. Our results suggest a simple and effective scheme for managing wide-area networks, namely establishing and servicing permanent virtual circuits according to the policy outlined above.

Fifth, we calculated the transmission efficiency of cell-based networks in transporting wide-area data traffic. We find that networks using standard protocols are inefficient. For example, ATM-based networks using SMDS and IEEE 802.6 protocols lose more than 40% of their bandwidth to overhead at the network level and below. Due to interaction between TCP-IP datagram lengths and ATM cell padding, efficiency responds abruptly to changes in certain protocol parameters − for example, a 4-byte increase in cell payload size can yield a 10% increase in efficiency. Furthermore, we find that viable compression techniques can significantly improve efficiency. For example, a combination of three compression techniques can regain more than 20% of the bandwidth previously lost to overhead.

6.2. Applicability of Results

In order to keep the multiplexing problem tractable, we narrowed the scope of our study to cell-based virtual circuit networks that carry traditional datagram traffic. Our results naturally apply to this important class of network and workload. However, the contributions outlined above also apply to many research areas not directly addressed by this dissertation. We describe five such applications below.

First, the uses for our traffic traces have been only partially exploited. For example, the information in the traces could be used to characterize geographic patterns of network usage [81] and thus plan where in the network available resources should be allocated. Similarly, these usage patterns can be used to study routing issues. We have already made these traces available to other network researchers at Bolt, Beranek, and Newman in Cambridge, Massachusetts, Lawrence Berkeley Laboratory in Berkeley, California, and the Swedish Institute of Computer Science in Stockholm.

Second, our characterization of wide-area data traffic and the resulting workload model can be used to drive a wide range of network performance studies. Our traffic model is independent of any specific network or any specific transport protocol. Therefore, it can be used whenever traditional wide-area data traffic is part of the workload of interest. For example, flow control and congestion control strategies for wide-area networks are active areas of current research that can benefit from our model. Our traffic model could also be used to study how new types of wide-area networks would react to a traditional workload. Future networks will most likely carry a mix of traffic not found in our traces. However, these networks will need to support traditional data traffic. It will be several years before the current traffic mix changes appreciably, and considerably longer before traditional traffic disappears altogether.

Third, our multiplexing results apply to datagram-based networks. We recall that our definition of a virtual circuit implies only a connection establishment and teardown procedure, a fixed path through the network, and a separate queue at every multiplexing point. Given that our study used permanent virtual circuits, our results do not depend on the dynamics of virtual circuit establishment and teardown. In addition, since routing tables seldom change in practice, our fixed path assumption holds even in the case of datagram networks. Datagram networks traditionally have not separated traffic either by type or by conversation. However, there is no fundamental reason why they cannot. Therefore, we suggest that datagram networks should separate traffic types and that they should provide round-robin service to shared communication links. A datagram network can inspect each arriving datagram in the same way we inspected our traffic traces. It can then separate traffic by type and provide each type a separate queue.

Fourth, our results apply to reservation-oriented networks. Some proposed networks reserve parts of their memory and bandwidth in order to provide performance guarantees [31] [102]. These networks will carry two broad types of traffic: real-time traffic, which has stringent delay and bandwidth requirements, and best-effort traffic, which has less rigorous requirements. The network gives priority to the real-time traffic component, and then allocates any remaining resources to the best-effort traffic. We simulated reservationless networks carrying best-effort traffic. However, in the context of a reservation-oriented network, our results show how to give good performance to best-effort traffic while making efficient use of the resources left over by real-time traffic.

Fifth, our results may find application in local-area ATM networks. ATM is rapidly gaining popularity in local-area environments [3] [17] [22] [64] [95]. This popularity is due both to the service integration advantages of cell-based networks over packet-based networks and to the benefits of point-to-point networks over broadcast networks [87]. Many of the traffic multiplexing issues we have addressed in the context of wide-area networks also apply to the limited resource conditions found in ATM host-network interfaces. In that environment, many design decisions are guided by the costs of connecting each host to the network, including buffer memory costs.

At the very least, the results of our work are directly applicable to wide-area networks based on ATM. There will exist many such networks. ATM is an international standard with broad support from telecommunication providers and computer manufacturers alike [4]. One early wide-area ATM network is Xunet 2 [35], alluded to in Chapter 5. Xunet 2 is an experimental 45 Megabit/second network that spans the continental United States. It began operation in 1992 after replacing the slower Xunet 1, which we used as an example in Chapter 1. Xunet 2 connects the University of California at Berkeley, the University of Illinois at Urbana-Champaign, the University of Wisconsin at Madison, AT&T Bell Laboratories in Murray Hill, New Jersey, and Lawrence Livermore and Sandia National Laboratories in Livermore, California. Xunet 2 will eventually carry scientific visualization and other experimental traffic. However, it will also carry traffic from traditional applications that use the IP datagram protocol.

6.3. Areas for Future Work

We have succeeded in finding multiplexing policies and mechanisms that meet our stated objectives of performance, fairness, and efficiency. However, our work could be extended in several areas. In this section, we outline six such areas for future work.

First, we should evaluate the impact of our new workload model on the results of previous studies. As discussed in Chapters 2 and 3, our measured traffic characteristics and consequently our workload model contradict previous models of wide-area data traffic. Many simulation studies of wide-area networks have been based on the older models. We should repeat these studies, preserving all previous conditions but substituting our new workload model. We believe our model is more realistic and will therefore yield more valid results. If the new results match the old ones, then the results are not sensitive to the differences in the underlying traffic models and will help to validate each other. If the results don't match, however, the conclusions of previous studies should be re-evaluated in light of the discrepancies.

Second, we should characterize the conversation arrival pattern at the entrance to wide-area networks. As noted in Chapter 3, we were unable to form a realistic and network-independent model of conversation arrivals. Our traces showed that these arrivals depend on geographic site, day of the week, time of day, and possibly other factors. Further analysis of traces from more than the four sites used in our study may yield insight into these conversation arrival processes.

Third, we should find effective policies for caching virtual circuits in networks without permanent virtual circuits. Due to the lack of a conversation interarrival model noted above, our study concentrated on periods of intra-conversation activity and used permanent virtual circuits. It did not address the issue of how long dynamic virtual circuits should remain open once the conversations using them have ceased. Idle virtual circuits waste network resources, especially in a network with resource reservations. On the other hand, when there is no appropriate virtual circuit to carry newly arrived data, the data must wait until one is established. In a wide-area network this process can entail considerable delays due to the round-trip time through the network. We need a policy for managing these tradeoffs. It is likely that an adaptive scheme will be better than any static policy due to the variability of conversation arrival processes. A more thorough investigation into this matter would be valuable.

Fourth, we should explore the effects of larger bulk transfers on our multiplexing results. One of the expected effects of increasing communication speeds is an increase in the size of bulk transfers. Our one year and three months of measurement activity did not show this trend, but we should not ignore the possibility that bulk transfers will get larger. For example, consider the rising popularity of document facsimile (FAX) and other digitized images, which can involve large amounts of data per item. Larger bulk transfers may increase the severity of the unfairness problem reported in Chapter 4. A possible solution woud be to perform round-robin queueing among the packet queues for conversations sharing a virtual circuit, not only among the cell queues for virtual circuits sharing a link. This policy may improve fairness in the throughput given to bulk transfer conversations. However, this policy may increase memory consumption for the same reasons given in Chapter 4 for why per-conversation VCs consume excessive amounts of memory. Evaluating the exact tradeoffs is a subject for future work.

Fifth, we should extend our simulation study to larger networks. In order to keep the simulations tractable, we simulated a simple network with only two routers and one switch. This configuration isolated the multiplexing problem to one of the routers at the entrance to the wide-area portion of the network, but voided the functionality of the switch internal to the wide-area portion of the network. It would be worthwhile to simulate networks where switches had multiple input and output lines in order to study the effect of the resulting cross-traffic on the multiplexing performance of the network.

Sixth, we should confirm our simulation results through experiment. Throughout our work, we were careful to insure realistic results. For this purpose, we based our study on measurements of real network traffic, we developed empirical models of this traffic, and we validated our simulator in various ways. Nevertheless, a judicious choice of experiments involving a real network like Xunet 2 would serve to further validate our simulation results. As already mentioned, the Xunet 2 network embodies the traffic multiplexing issues addressed in this dissertation. Unfortunately, as of this writing, the full Xunet 2 configuration is not yet operational. Although the network's switches and links already provide basic ATM connectivity between the member sites, hardware and software for its routers are still under development. When these routers come on line, traffic multiplexing experiments on Xunet 2 would be a valuable addition to our work.

6.4. Benefits of Our Methodology

We would like to stress the benefits of our research methods to the study of networks in general. Our empirical approach to constructing models of traditional data traffic should be applied to other types of traffic. We should measure, analyze, and empirically model traffic types as they mature. We should not rely on intuitive ideas of their behavior as we have with traditional wide-area network traffic. As we saw in Chapter 2, our intuition is often wrong.

We have described the first steps in building and maintaining a collection of compact generative models of existing traffic types. We should extend our collection to include the increasingly common document facsimile (FAX) traffic. FAX is a bulk transfer application with many qualitative similarities to file transfers, network news, and electronic mail. We should measure its exact characteristics and extract those histograms used by our generalized bulk transfer model. We should also form new empirical models for real-time traffic as soon as video and audio applications become widespread, and add them to the current collection.

This process should be continuous, not limited to isolated research activities. From their inception, networks should be instrumented to monitor usage patterns as part of their normal operation. When these patterns change due either to the introduction of new traffic types or to other changes in user behavior, past design decisions should be re-evaluated and network parameters tuned if necessary. The latest traffic information should also guide the design of future networks.

6.5. Final Remarks

Communication networks are growing rapidly as people and their data become better connected across large distances. The resulting aggregation of traffic creates many problems for the design of data networks. We hope our measurement, analysis, and modeling efforts have brought about a better understanding of the sources of wide-area network traffic. We also hope our work will lead to better ways of mixing traffic from many different sources at the entrance to wide-area networks.

References

1. *UNIX Research System, Tenth Edition*, AT&T Bell Laboratories, Murray Hill, New Jersey, 1990.
2. A. V. Aho, B. W. Kernighan and P. J. Weinberger, *The AWK Programming Language*, Addison-Wesley, Reading, Massachusetts, 1987.
3. T. E. Anderson, S. S. Owicki, J. B. Saxe and C. P. Thacker, High Speed Switch Scheduling for Local Area Networks, *Proc. of ACM ASPLOS-V*, Boston, Massachusets, September, 1992.
4. *Network Compatible ATM for Local Network Applications, Phase I, Version 1.0*, Apple Computer, Inc., Bell Communications Research, Sun Microsystems, Xerox Palo Alto Research Center, April, 1990.
5. M. G. Baker, J. H. Hartman, M. D. Kupfer, K. W. Shirriff and J. K. Ousterhout, Measurements of a Distributed File System, *Proc. of the 13th ACM Symposium on Operating System Principles*, Pacific Grove, California, .
6. J. L. Bentley and B. W. Kernighan, GRAP - A Language for Typesetting Graphs, *Communications of the ACM 29*, 8 (February, 1986).
7. J. L. Bentley, *More Programming Pearls*, Addison-Wesley, Reading, Massachusetts, 1988.
8. A. Birrell and B. Nelson, Implementing Remote Procedure Calls, *ACM Transactions on Computer Systems 2*, 1 (February, 1984).
9. B. Braden and A. L. DeSchon, NNStat: Internet Statistics Collection Package - Introduction and User Guide, *University of Southern California/Information Sciences Institute*, December, 1989.
10. R. Caceres, The Pyramid IP to X.25 Protocol Interface: Merging DDN and PDN Approaches, *Proceedings of the Uniforum 1987 Conference*, Washington, DC, January, 1987.
11. R. Caceres, Measurements of Wide-Area Internet Traffic, *Technical Report UCB/CSD 89/550*, Berkeley, California, December, 1989.
12. R. Caceres, P. B. Danzig, S. Jamin and D. J. Mitzel, Characteristics of Wide-Area TCP/IP Conversations, *Proc. of ACM SIGCOMM '91*, September, 1991.
13. I. Cidon, J. Derby, I. Gopal and B. Kadaba, A Critique of ATM from a Data Communications Perspective, *Proc. of ICCC '90*, November, 1990.
14. D. Comer and J. Korb, CSNET Protocol Software: The IP-to-X.25 Interface, *Proc. of ACM SIGCOMM '83*, March 1983.

15. D. Comer, *Internetworking with TCP/IP*, Prentice Hall, Englewood Cliffs, New Jersey, 1988.

16. *UNIX Programmer's Manual, 4.2 Berkeley Software Distribution*, Computer Systems Research Group, Computer Science Division, University of California, Berkeley, California, 1984.

17. E. Cooper, O. Menzilcioglu, R. Sansom and F. Bitz, Host Interface Design for ATM LANs, *16th Annual Conference on Local Computer Networks*, 1991.

18. J. Crowcroft and I. Wakeman, *Traffic Analysis of Some UK-US Academic Network Data*, University College London, London, England, June, 1991.

19. P. B. Danzig and S. Melvin, High Resolution Timing with Low Resolution Clocks and a Microsecond Timer for Sun Workstations, *ACM Operating Systems Review 24*, 1 (January, 1990).

20. P. B. Danzig and S. Jamin, tcplib: A Library of TCP Internetwork Traffic Characteristics, *Technical Report USC-CS-91-495*, Los Angeles, California, September, 1991.

21. P. B. Danzig, S. Jamin, R. Caceres, D. J. Mitzel and D. Estrin, An Empirical Workload Model for Driving Wide-Area TCP/IP Network Simulations, *Journal of Internetworking: Research and Experience 3*, 1-26 (1992).

22. B. S. Davie, A Host-Network Interface Architecture for ATM, *Proc. of ACM SIGCOMM '91*, Zurich, Switzerland, September, 1991.

23. A. DeSimone, Efficiency of Asynchronous Transfer Mode (ATM) Networks in Carrying TCP/IP Traffic, *AT&T Bell Laboratories*, December, 1989. (Unpublished technical memorandum).

24. *Defense Data Network X.25 Host Interface Specification*, Defense Communications Agency, Washington, D.C., December, 1983.

25. A. Demers, S. Keshav and S. Shenker, Analysis and Simulation of a Fair Queueing Algorithm, *Proc. of ACM SIGCOMM '89*, Austin, Texas, September, 1989.

26. S. Dravida, H. Ruben and J. S. Swenson, Segmentation and Reassembly Layer for IEEE 802.6/B-ISDN, *Proposed Standard: DQDB Metropolitan Area Network, IEEE 802.6*, August, 1989.

27. A. Dupuy, J. Schwartz, Y. Yemini and D. Bacon, NEST: A Network Simulation and Prototyping Testbed, *Communications of the ACM 33*, 10 (October, 1990).

28. J. Escobar and C. Partridge, A Proposed Segmentation and Reassembly (SAR) Protocol for Use with Asynchronous Transfer Mode (ATM), *High Performance Network Research Report*, October, 1990.

29. D. C. Feldmeier, Multiplexing Issues in Communication System Design, *Proc. of ACM SIGCOMM '90*, Philadelphia, Pennsylvania, September, 1990.

30. D. Ferrari, *Computer Systems Performance Evaluation*, Prentice-Hall, Englewood Cliffs, New Jersey, 1978.

31. D. Ferrari and D. Verma, A Scheme for Real-Time Channel Establishment in Wide-Area Networks, *IEEE Journal on Selected Areas in Communications 8*, 3 (April, 1990).

32. S. Floyd and V. Jacobson, Traffic Phase Effects in Packet-Switched Gateways, *Computer Communication Review 21*, 2 (April, 1991).

33. A. G. Fraser, Towards a Universal Data Transport System, *IEEE Journal on Selected Areas in Communication 1*, 5 (November, 1983).

34. A. G. Fraser and S. P. Morgan, Queueing and Framing Disciplines for a Mixture of Data Traffic Types, *AT&T Bell Laboratories Technical Journal 63*, 6 (July-August, 1984).

35. A. G. Fraser, C. R. Kalmanek, A. E. Kaplan, W. T. Marshall and R. C. Restrick, Xunet 2: A Nationwide Testbed in High-Speed Networking, *Proc. of IEEE INFOCOM '92*, May, 1992.

36. E. Fuchs and P. E. Jackson, Estimates of Distributions of Random Variables for Certain Computer Communications Traffic Models, *Communications of the ACM 13*, 12 (December, 1970).

37. R. Gusella, A Measurement Study of Diskless Workstation Traffic on an Ethernet, *IEEE Transactions on Communications 38*, 9 (September, 1990).

38. R. Gusella, A Characterization of the Variability of Packet Arrival Processes in Workstation Networks, *Technical Report UCB/CSD 90/612*, Berkeley, California, December 1990. (Ph.D. Dissertation).

39. E. L. Hahne, Round Robin Scheduling for Max-Min Fairness in Data Networks , *IEEE Journal on Selected Areas in Communications 9*, 7 (September, 1991).

40. E. L. Hahne, C. R. Kalmanek and S. P. Morgan, Dynamic Window Flow Control on a High-Speed Wide-Area Network, *to appear in Computer Networks and ISDN Systems Journal*, 1992.

41. R. Handel, Evolution of ISDN Towards Broadband ISDN, *IEEE Network*, January, 1989.

42. H. Heffes and D. M. Lucantoni, A Markov Modulated Characterization of Packetized Voice and Data Traffic and Related Statistical Multiplexer Performance, *IEEE Journal on Selected Areas in Communications 4*, 6 (September, 1986).

43. S. A. Heimlich, Traffic Characterization of the NSFNET National Backbone, *Proc. of the 1990 Winter USENIX Conference*, January, 1990.

44. R. G. Herrtwich, ed., *Network and Operating System Support for Digital Audio and Video*, Springer, Heidelberg, Germany, 1992.

45. First International Workshop on Network and Operating System Support for Digital Audio and Video, *Technical Report Tech. Rpt.-90-062*, Berkeley, California, November, 1990.

46. P. E. Jackson and C. D. Stubbs, A Study of Multiaccess Computer Communications, *Proc. of AFIPS 1969 SJCC 34* (1969), AFIPS Press.

47. V. Jacobson, Congestion Avoidance and Control, *Proc. of ACM SIGCOMM '88* , August, 1988.

48. V. Jacobson, Compressing TCP/IP Headers for Low-Speed Serial Links, *Network Working Group Request for Comments 1144*, February, 1990.

49. R. Jain and S. Routhier, Packet Trains: Measurements and a New Model for Computer Network Traffic, *IEEE Journal on Selected Areas in Communication 4*, 6 (September, 1986).

50. R. Jain, *The Art of Computer Systems Performance Analysis*, Wiley, New York, New York, 1991.

51. C. R. Kalmanek and S. P. Morgan, NETSIM: A Simulator for a Wide-Area ATM Data Network, *AT&T Bell Laboratories*, Murray Hill, New Jersey, November, 1990. (Unpublished technical memorandum).

52. C. R. Kalmanek, *Frame Relay Service for Xunet 2*, AT&T Bell Laboratories, June, 1991. (Private communication).

53. M. G. H. Katevenis, Fast Packet Switching and Fair Control of Congested Flow in Broadband Networks, *IEEE Journal on Selected Areas in Communication 5*, 8 (October, 1987).

54. C. Kent and J. C. Mogul, Fragmentation Considered Harmful, *Proc. of SIGCOMM '87 Workshop on Frontiers in Computer Communications Technology*, Stowe, Vermont, August, 1987.

55. B. W. Kernighan and D. M. Ritchie, *The C Programming Language*, Prentice-Hall, Englewood Cliffs, New Jersey, 1978.

56. B. W. Kernighan and R. Pike, *The UNIX Programming Environment*, Prentice-Hall, Englewood Cliffs, New Jersey, 1984.

57. S. Keshav, REAL: A Network Simulator, *Technical Report UCB/CSD 88/472*, Berkeley, California, December, 1988.

58. S. Keshav, Congestion Control in Computer Networks, *Technical Report UCB/CSD 91/649*, Berkeley, California, September, 1991. (Ph.D. Dissertation).

59. S. Keshav, The Packet Pair Flow Control Protocol, *Technical Report 91-028*, Berkeley, California, May, 1991.

60. L. Kleinrock, *Queueing Systems, Volume I*, Wiley, New York, New York, 1975.

61. L. Kleinrock, W. E. Naylor and H. Opderbeck, A Study of Line Overhead in the ARPANET, *Communications of the ACM 19*, 1 (January, 1976).

62. L. Kleinrock, *Queueing Systems, Volume II*, Wiley, New York, New York, 1976.

63. W. E. Leland and D. V. Wilson, High Time-Resolution Measurement and Analysis of LAN Traffic: Implications for LAN Interconnection, *Proc. of IEEE INFOCOM '91*, 1991.

64. I. Lesley and D. R. McAuley, Fairisle: An ATM Network for the Local Area, *Proc. of ACM SIGCOMM '91*, Zurich, Switzerland, September, 1991.

65. *The X.25 Protocol and Seven Other Key CCITT Recommendations*, Lifetime Learning Publications, Belmont, California, 1981.

66. M. Lottor, Internet Domain System, *Communications of the ACM 34*, 11 (November, 1991). (Letter to ACM Forum).

67. A. Mankin, Random Drop Congestion Control, *Proc. of ACM SIGCOMM '90*, Philadelphia, Pennsylvania, September, 1990.

68. G. Marsaglia, K. Ananthanarayanan and N. Paul, *Random Number Generator Package – 'Super Duper'*, School of Computer Science, McGill University, Montreal, Canada, 1973.

69. W. T. Marshall and S. P. Morgan, Statistics of Mixed Data Traffic on a Local Area Network, *Computer Networks and ISDN Systems 10*, 3-4 (October-November, 1985).

70. *NSFnet Backbone Statistics*, Merit/NSFnet Information Services, June, 1991. (Obtained by anonymous FTP from nis.nsf.net).

71. *The Merit/NSFnet Backbone Link Letter 3*, 6 (January/February, 1991), Merit/NSFnet Information Services.

72. S. E. Minzer, Broadband-ISDN and Asynchronous Transfer Mode (ATM), *IEEE Communications 27*, 9 (September, 1989).

73. J. C. Mogul, The Experimental Literature of the Internet: An Annotated Bibliography, *WRL Research Report 88/3*, Palo Alto, California, August, 1988.

74. J. C. Mogul and S. Deering, Path MTU Discovery, *Network Working Group Request for Comments 1191*, November, 1990. (Obsoletes RFC 1063).

75. S. P. Morgan and C. Y. Lo, Mean Message Delays for Two Packet-FIFO Queueing Disciplines, *to appear in IEEE Transactions on Communications 38*, 6 (June 1990).

76. S. P. Morgan, Queueing Disciplines and Passive Congestion Control in Byte-Stream Networks, *IEEE Transactions on Communications 39*, 7 (July, 1991).

77. J. Nagle, On Packet Switches with Infinite Storage, *IEEE Trans. on Communications COM-35* (1987).

78. J. K. Ousterhout, H. DaCosta, D. Harrison, J. A. Kunze, M. D. Kupfer and J. G. Thompson, A Trace-Driven Analysis of the UNIX 4.2 BSD File System, *Proc. of the 10th ACM Symposium on Operating System Principles*, December, 1985.

79. J. Ousterhout, A. Cherenson, F. Douglis, M. Nelson and B. Welch, The Sprite Network Operating System, *IEEE Computer 21*, 2 (February, 1988).

80. P. F. Pawlita, Two Decades of Data Traffic Measurements: A Survey of Published Results, Experiences and Applicability, *Teletraffic Science for New Cost-Effective Systems, Networks, and Services, ITC-12*, 1989.

81. V. Paxson, Measurements and Models of Wide-Area TCP Conversations, *Technical Report LBL-30840*, Berkeley, California, June, 1991.

82. J. Quarterman and J. Hoskins, Notable Computer Networks, *Communications of the ACM 29*, 10 (October 1986).

83. K. K. Ramakrishnan and R. Jain, A Binary Feedback Scheme for Congestion Avoidance in Computer Networks, *ACM Transactions on Computer Systems 8*, 2 (May, 1990).

84. M. J. Rider, Protocols for ATM Access Networks, *IEEE Network*, January, 1989.

85. C. H. Sauer, E. A. MacNair and J. F. Kurose, *The Research Queueing Package Version 2*, IBM T. J. Watson Research Center, Yorktown Heights, New York, 1982.

86. A. Schmidt and R. Campbell, Internet Protocol Analysis with Applications for ATM Switch Design, *Technical Report TR92-1735*, Urbana-Champaign, Illinois, March, 1992.

87. M. Schroeder, A. Birrell, M. Burrows, H. Murray, R. Needham, T. Rodeheffer, E. Satterthwaite and C. Thacker, Autonet: A High-Speed Self-Configuring Local Area Network Using Point-to-Point Links, *IEEE Journal on Selected Areas in Communications 9*, 8 (October, 1991).

88. M. Schwartz, *Telecommunications Networks: Protocols, Modeling, and Analysis*, Addison-Wesley, Reading, MA, 1987.

89. H. Schwetman, CSIM Reference Manual Version 12, *Technical Report ACA-ST-252-87*, Austin, Texas, November, 1987.

90. B. Shneiderman, *Designing the User Interface*, Addision-Wesley, 1987.

91. B. Stroustrup, *The C++ Programming Language*, Addison-Wesley, Reading, Massachusetts, 1986.

92. B. Stroustrup and J. E. Shopiro, A Set of C++ Classes for Co-Routine Style Programming, *AT&T C++ Language System Release 2.0 Library Manual*, Murray Hill, New Jersey, June, 1989.

93. A. Tanenbaum, *Computer Networks*, Prentice-Hall, Englewood Cliffs, New Jersey, 1981.

94. D. L. Tennenhouse, Layered Multiplexing Considered Harmful, *Proc. of IFIP Workshop on Protocols for High-Speed Networks*, 1989.

95. C. B. Traw and J. M. Smith, A High Performance Host Interface for ATM Networks, *Proc. of ACM SIGCOMM '91*, Zurich, Switzerland, September, 1991.

96. T. VandeWater, Delay and Throughput Measurements of the XUNET Datakit Network, *Technical Report UCB/CSD 88/474*, Berkeley, California, November, 1988.

97. D. C. Verma, Guaranteed Performance Communication in High Speed Networks, *Technical Report UCB/CSD 91/663*, Berkeley, California, December, 1991. (Ph.D. Dissertation).

98. S. Wecker, Computer Network Architectures, *IEEE Computer 23*, 12 (December 1980).

99. R. Wilder, K. K. Ramakrishnan and A. Mankin, Dynamics of Congestion Control and Avoidance of Two-Way Traffic in an OSI Testbed, *Computer Communication Review 21*, 2 (April, 1991).

100. C. L. Williamson and D. R. Cheriton, Loss-Load Curves: Support for Rate-Based Congestion Control in High-Speed Datagram Networks, *Proc. of ACM SIGCOMM '91*, Zurich, Switzerland, September, 1991.

101. L. Zhang, Virtual Clock: A New Traffic Control Algorithm for Packet Switching Networks, *Proc. of ACM SIGCOMM '90*, Philadelphia, Pennsylvania, September, 1990.

102. L. Zhang, S. Shenker and D. D. Clark, Observations on the Dynamics of a Congestion Control Algorithm: The Effects of Two-Way Traffic, *Proc. of ACM SIGCOMM '91*, Zurich, Switzerland, September, 1991.

103. Generic System Requirements in Support of Switched Multi-Megabit Data Service, *Bellcore Technical Advisory TA-TSY-000772, Issue 2*, March, 1989.

A. Comparison of Traffic Characteristics

Figures A.1 through A.8 present a comparison of application-level conversation characteristics found in data from three sites at which traffic was measured: the University of California at Berkeley (UCB), the University of Southern California (USC) in Los Angeles, and Bell Communications Research (BCR) in Morristown, New Jersey. The characteristics are very similar. The only significant discrepancies are in the packet interarrival times for bulk transfer conversations (see Figure A.6), which can be explained by the different clock resolutions represented in the three traces. In the body of the dissertation, we treat the UCB characteristics as representative of the three.

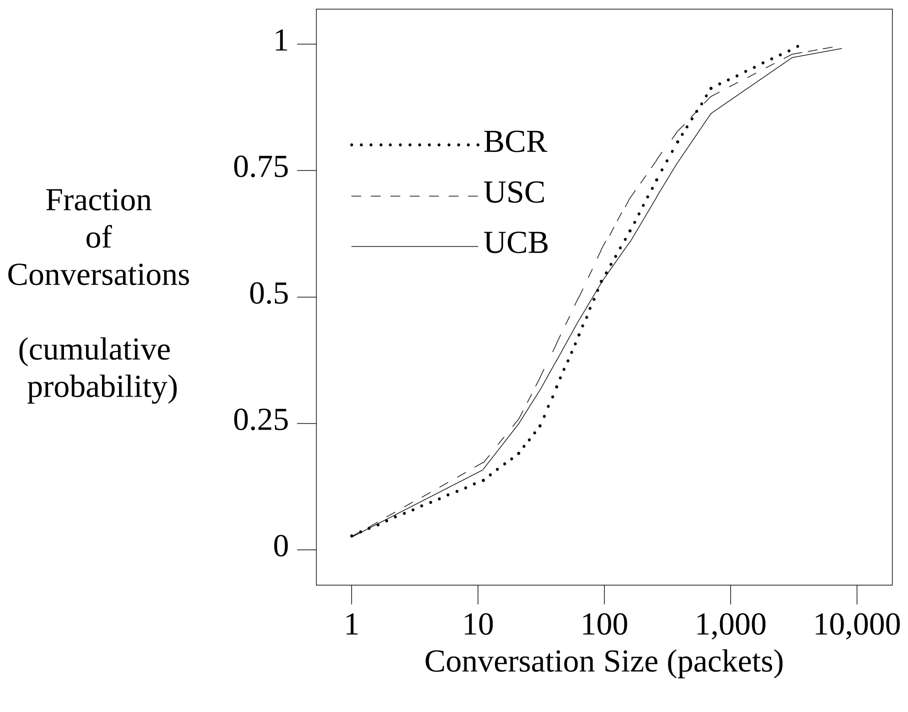

Figure A.1. Number of packets transferred per TELNET conversation

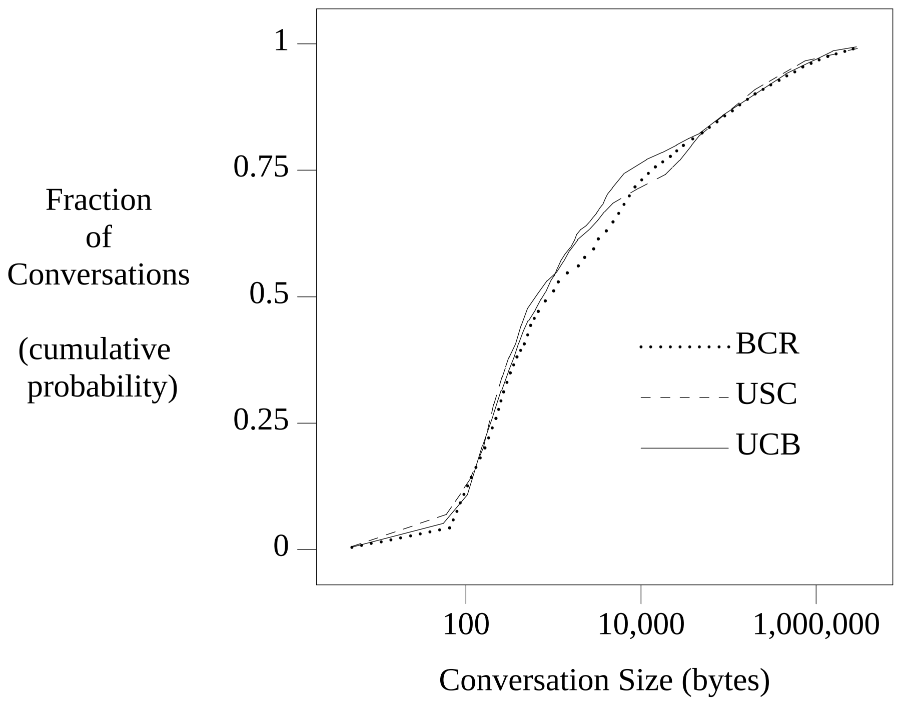

Figure A.2. Number of bytes transferred per FTP conversation

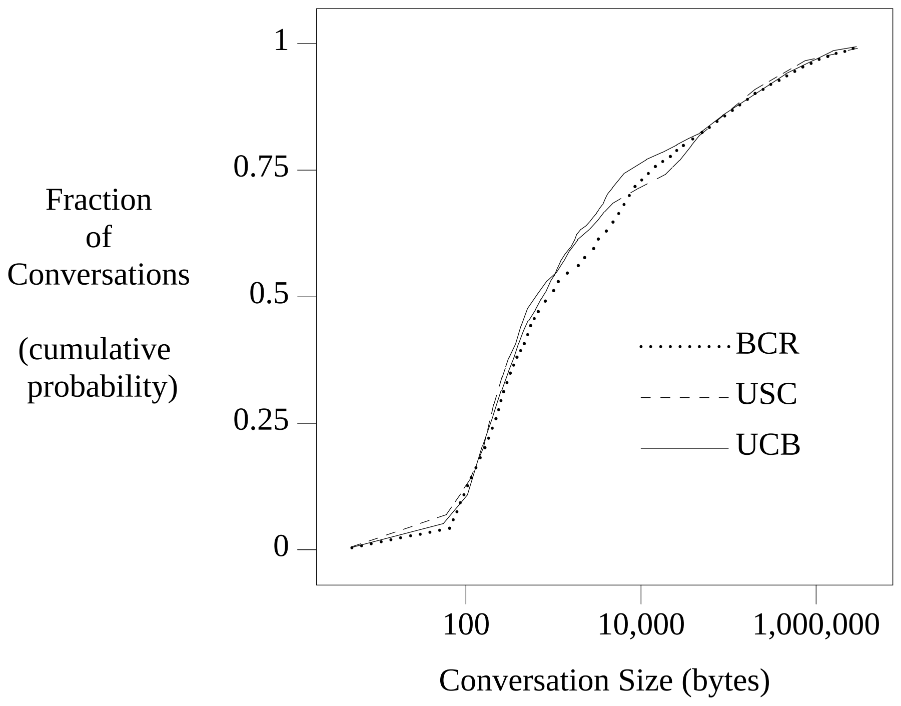

Figure A.3. Number of bytes transferred per TELNET conversation

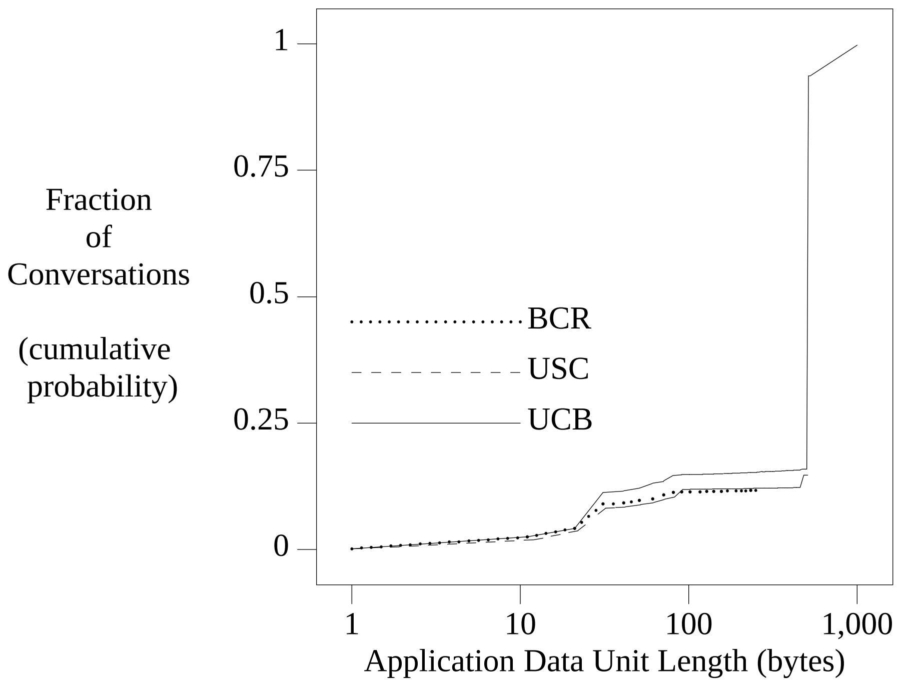

Figure A.4. Lengths of application data units in FTP conversations

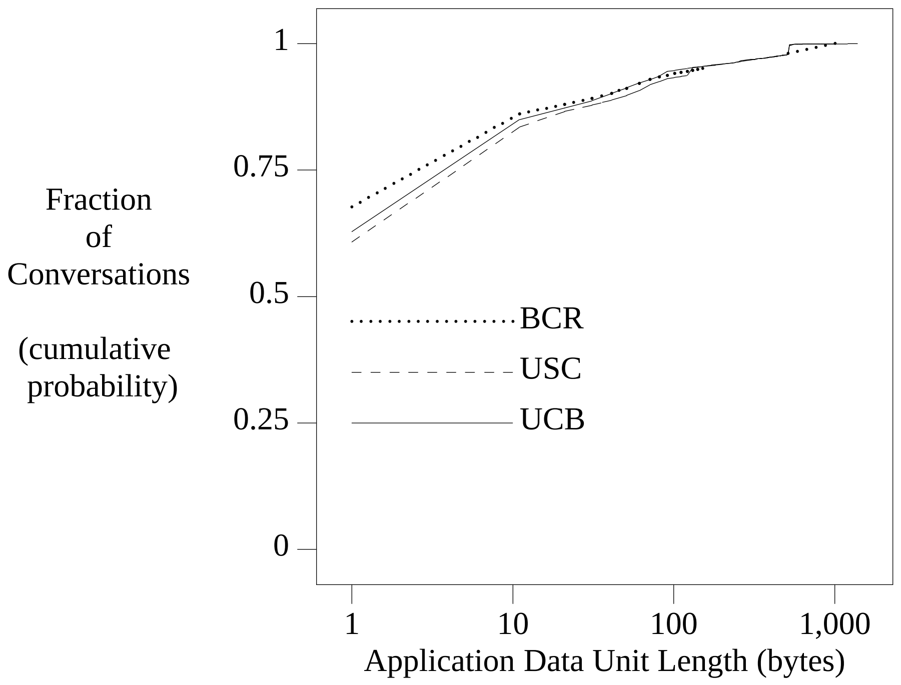

Figure A.5. Lengths of application data units in TELNET conversations

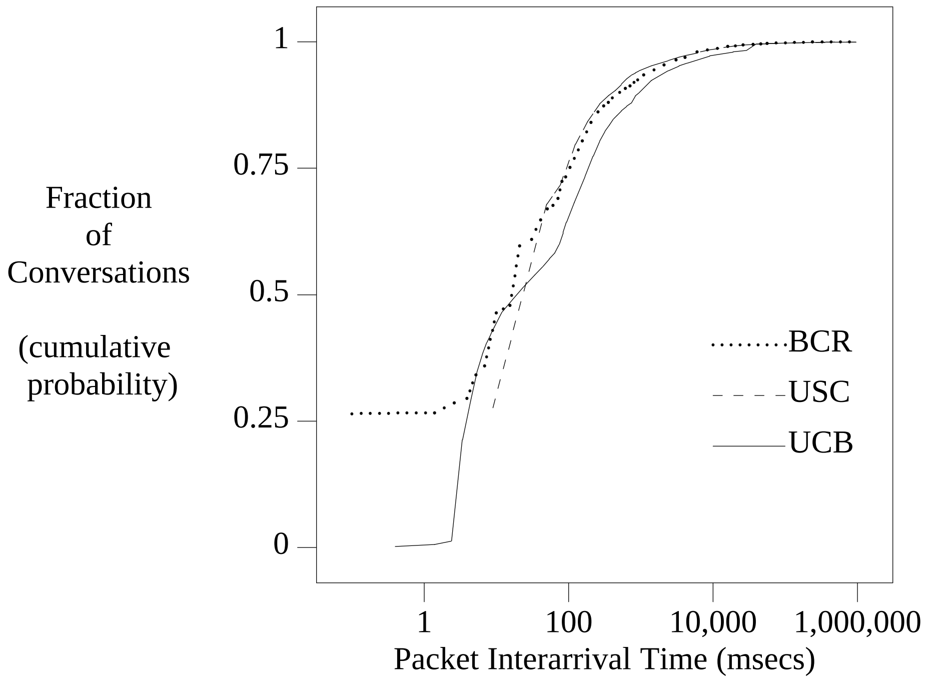

Figure A.6. Packet interarrival times in FTP conversations

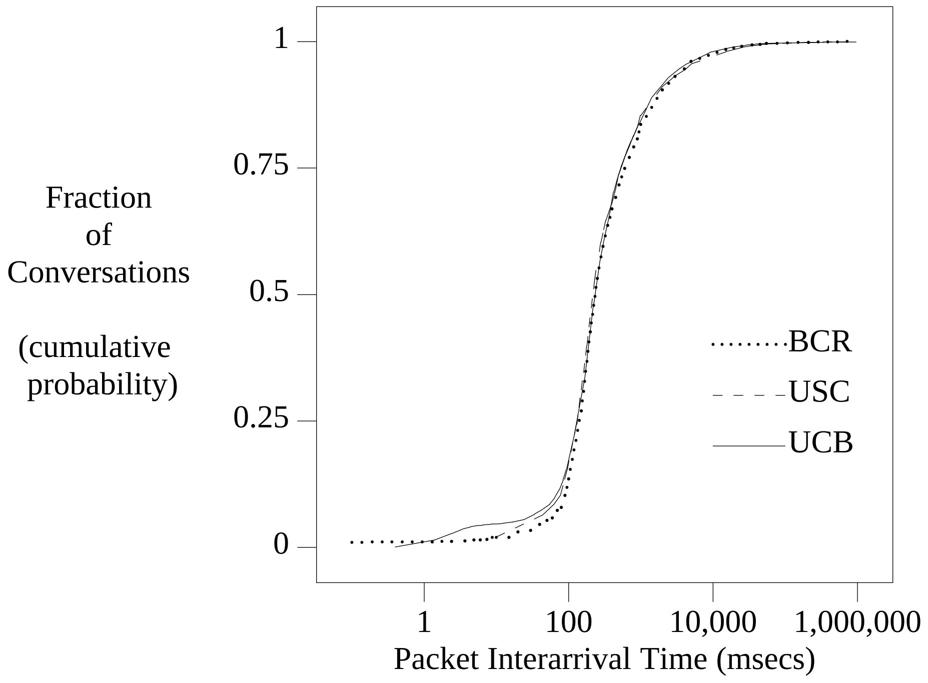

Figure A.7. Packet interarrival times in TELNET conversations

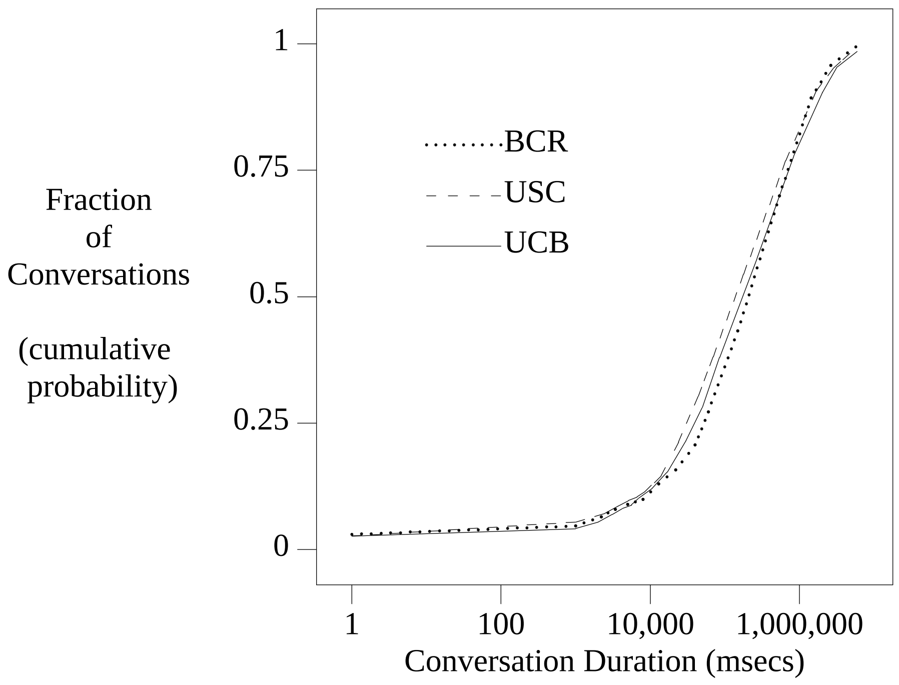

Figure A.8. Number of milliseconds per TELNET conversation

B. Comparison of Efficiency Results

Tables B.1 through B.3 present a comparison of transmission efficiency results obtained with data from four sites at which traffic was measured: the University of California at Berkeley (UCB), the University of Southern California (USC) in Los Angeles, AT&T Bell Laboratories (BL) in Murray Hill, New Jersey, and Bell Communications Research (BCR) in Morristown, New Jersey. The results are very similar. In the body of the dissertation, we treat the UCB results as representative of the four.

Protocol		% Application Efficiency					
Framing	Adaptation	UCB	USC	BL	BCR	Mean	Std. Dev.
SMDS	802.6	40.1	49.0	46.1	44.2	44.9	3.23
XFRS	none	44.7	54.8	51.3	49.0	49.9	3.67
none	SAR	52.3	59.2	55.4	55.9	55.7	2.45

Table B.1. Application Efficiency Using Standard ATM Procedures

Protocol		% Datagram Efficiency					
Framing	Adaptation	UCB	USC	BL	BCR	Mean	Std. Dev.
SMDS	802.6	58.7	64.1	62.2	61.2	61.6	1.95
XFRS	none	65.5	71.6	69.2	67.8	68.5	2.21
none	SAR	76.6	77.4	74.8	77.5	76.6	1.08

Table B.2. Datagram Efficiency Using Standard ATM Procedures

Protocol		% Frame Efficiency					
Framing	Adaptation	UCB	USC	BL	BCR	Mean	Std. Dev.
SMDS	802.6	72.4	74.7	74.3	73.4	73.7	0.89
XFRS	none	74.9	78.8	77.0	76.2	76.7	1.41
none	SAR	76.6	77.4	74.8	77.5	76.6	1.08

Table B.3. Frame Efficiency Using Standard ATM Procedures

C. Acronyms and Abbreviations

This appendix contains a glossary of acronyms and abbreviations used throughout the dissertation.

802.6	IEEE segmentation and reassembly protocol for ATM networks
ATM	Asynchronous Transfer Mode
BCR	Bell Communications Research or Bellcore
B-ISDN	Broadband Integrated Services Digital Network
BL	AT&T Bell Laboratories
CCITT	International Telegraph and Telephone Consultative Committee
DC 10	Cadre Teambox Mailbox 10
DOMAIN	Domain name service protocol
DS3	45 Megabit/second long-haul transmission standard
FINGER	User information query protocol
FTP	File Transfer Protocol
IEEE	Institute of Electrical and Electronic Engineers
ICMP	Internet Control Message Protocol
IP	Internet Protocol
IRCD	Internet Relay Chat Daemon
MSP	Message Stream Protocol
MTU	Maximum Transmission Unit
NNTP	Network News Transfer Protocol
NTP	Network Time Protocol
RCP	Remote copy protocol
RLOGIN	Remote login protocol
ROUTE	Routing information exchange protocol
SAR	Bolt, Beranek and Newman Segmentation And Reassembly protocol
SMTP	Simple Mail Transfer Protocol
SMDS	Switched Multi-Megabit Data Service
TCP	Transmission Control Protocol
TELNET	Remote terminal protocol
UCB	University of California at Berkeley
UDP	User Datagram Protocol
USC	University of Southern California
UUCP	Unix to Unix Copy Program
VC	Virtual Circuit
VCSIM	Virtual Circuit Simulator
VMNET	Virtual machine job transfer protocol
X11	X11 window system
XFRS	Xunet 2 Frame Relay Service