Multiplexing TCP Traffic over Wide-Area ATM Networks

Ramón Cáceres

Matsushita Information Technology Laboratory
2 Research Way, Princeton, NJ 08540
ramon@mitl.research.panasonic.com

ABSTRACT

We explore the effects of network-level multiplexing policies on application-level performance in the context of wide-area networks that carry small fixed-size cells. We simulate an ATM network with traffic sources, traffic sinks, routers, switches, and links. Sources obey new empirical models of network applications and use the TCP transport-level protocol with the slow-start congestion control algorithm. We report on the delay and throughput achieved when the network separates traffic by application-level conversation, when it separates traffic by application type, and when it treats all traffic the same way. A multiplexing policy that separates traffic by conversation and services queues on a cell-by-cell round-robin basis is superior to policies that separate traffic to a lesser degree and use first-in first-out queuing. This policy offers low delay and low variability of delay to interactive applications, and provides a fair share of the network bandwidth to each bulk transfer applications, even as network load rises to saturation. This policy is also simple to implement in real networks.

1. Introduction

Networks based on the Asynchronous Transfer Mode (ATM) standard[19] are rapidly gaining popularity. These networks carry all data in small fixed-size cells. Network applications that use the Transmission Control Protocol (TCP)[26] are the dominant traffic sources in current wide-area networks and will remain important clients of these networks for some time. Examples of these applications are file transfers (FTP), network news (NNTP), electronic mail (SMTP), and remote terminal sessions (TELNET), They fall into two broad categories, interactive applications (TELNET) and bulk transfer applications (FTP, NNTP, and SMTP). In general, interactive applications demand low delay, while bulk transfer applications are more concerned with high throughput. We need to multiplex data traffic over wide-area cell networks in ways that offer good performance and fair treatment to all traffic sources.

In this paper, we explore how well cell networks using different multiplexing policies achieve these performance, fairness, and efficiency objectives. We simulate a wide-area ATM network composed of traffic sources, traffic sinks, routers, switches, and links. We drive our simulations with new empirical models of FTP, NNTP, SMTP, and TELNET traffic[5]. Sources and sinks communicate using TCP with the slow-start congestion avoidance and control algorithm[14]. Simulated routers, switches, and links form the wide-area network fabric that carries data from sources to sinks.

We compare the delay and throughput characteristics of three multiplexing policies: a policy that separates traffic by application-level conversation, a policy that separates traffic by type of application, and a policy that treats all traffic the same way. 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. The network separates two traffic streams by assigning each stream its own cell queue at every multiplexing point and servicing queues in a cell-by-cell round-robin fashion. The network treats two streams equally

by assigning them the same first-in first-out queue at every shared multiplexing point.

We find that wide-area cell networks should separate traffic by application-level conversation. This multiplexing policy provides interactive applications with delays very near to one network round-trip time, the minimum delay possible. It also provides them with low variability of delay. Furthermore, this policy provides bulk transfer applications with throughput equal to a fair share of the available bandwidth. This policy is simple to implement in the important and real case of connection-oriented ATM networks.

The remainder of this section further motivates our work and surveys the literature in the area. Section 2 describes the simulation environment used in this study. Sections 3 through 5 present our simulation results in detail. Sections 6 and 7 explore the sensitivity of these results to variations in two important simulation parameters. Finally, Section 8 discusses the feasibility of implementing the recommended multiplexing policy in a real network.

1.1. Motivation

Our work is motivated by the failure of past networks to meet the stated multiplexing objectives, and by trends in future networking. As an example of a current network, the Internet exhibits unacceptable interactive delays in the face of congestion. Cross-continental response times longer than 100 to 200 milliseconds are common. Furthermore, these response times are highly variable, ranging from under 80 to over 200 milliseconds. † Studies have found that humans perceive interactive response to be ‘‘bad’’ when it takes longer than 100 to 200 milliseconds, and that predictable response times are preferable to variable ones[27]. It is often difficult for interactive users to work productively across the Internet. With regard to bulk transfer applications, networks like the Internet do not always offer a fair share of their bandwidth to each traffic source. TCP traffic sources can encounter phenomena such as *synchronization* and *ACK-compression* that reduce the throughput obtained by some sources to a fraction of their fair share. These phenomena have been discovered through simulation[9,30] and also observed in real networks[20,29].

These shortcomings are due in a large part to the multiplexing policies of networks like the Internet. These networks indiscriminately mix application-level traffic streams, use a first-in first-out queueing discipline to schedule data for transmission, and multiplex data with the granularity of variable-sized packets. Under heavy loads, this combination of policies can interact badly with the transmission window mechanism in transport-level protocols, resulting in high interactive delay and unfairness in bulk transfer throughput.

Trends in wide-area networks suggest that the problems just described will intensify. The number of hosts connected to the Internet is increasing exponentially[18]. The amount of traffic, carried by the wide-area portion of Internet is also increasing exponentially[1]. Even if the growth in the number of hosts drops to sub-exponential, there is evidence that the amount of wide-area traffic will continue to grow exponentially due to increases in wide-area network usage by individual users[24]. In contrast, the installed capacity of the long-haul portion of the Internet is increasing at a much slower pace. Wide-area networks will continue to be heavily loaded due to the aggregation of traffic from many sources.

We aim to find multiplexing policies for use in cell networks that coexist well with modern transport protocols in spite of high loads. Cell networks using round-robin queueing disciplines hold promise for achieving our objectives without undue complexity because of the fine multiplexing granularity provided by cells and the inherent fairness of round-robin.

---

† Measured on a typical weekday afternoon with the Unix *ping* utility between the University of California at Berkeley and Matsushita Information Technology Laboratory in Princeton, New Jersey.

1.2. Previous Work

There has been considerable research exploring the behavior of transport-level protocols like TCP over various types of networks. Recent examples are work by Floyd and Jacobson[9], Zhang et al[30], and Wilder et al[29]. We refer to much of this work throughout this paper. We limit the following survey to work that, like ours, treats the benefits and drawbacks of separating traffic streams to various degrees and using different queueing disciplines. In a virtual circuit setting, Hahne[13] and Morgan et al[10,21,22] 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[16], Nagle[23], and Demers et al[8], who reached similar conclusions for datagram networks.

Our work adds to previous efforts in two respects. First, we drive our simulations with new traffic models derived from extensive measurements of the Internet. As shown in the next section, these models contradict many previous beliefs regarding wide-area network traffic. Second, we specifically investigate the interaction between modern TCP and networks based on ATM. TCP with slow-start is the dominant transport protocol in current wide-area networks, while ATM is an an increasingly popular multiplexing technique. Previous efforts have used less realistic traffic models, protocols other than TCP with slow-start, or networks not based on small fixed-size cells.

2. Network Simulation

We built a discrete-event simulator, CELLSIM, to study the problem of multiplexing data traffic over wide-area cell networks. As shown in Figure 1, CELLSIM simulates five network components: sources, sinks, routers, switches, and links. Sources generate packets destined for the sinks and send them along a link to the nearest router. Routers at the edge of the wide-area network fragment packets into cells, and send these cells to an adjacent switch. Switches forward the cells along a path to the router closest to the destination sink. This router reassembles the original packet from the component cells and forwards it to the intended sink. Sinks send response and acknowledgment packets along the reverse path. Links connect two nodes of any type (source, sink, router, or switch) and introduce transmission and propagation delay.

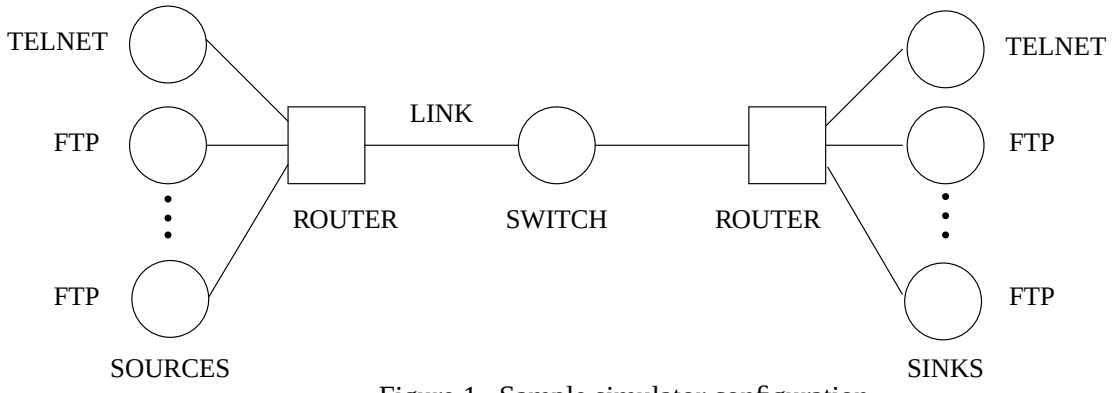

Figure 1. Sample simulator configuration

CELLSIM is derived from NETSIM[15], an earlier simulator of wide-area ATM networks. Functionality added to NETSIM includes new empirical models of traffic sources, multiple mappings of application-level conversations to network-level channels, and multiple queueing disciplines.

Inside the simulated network, cells are routed along *channels* that flow between two routers. The channels in CELLSIM consist of only two things: a fixed path through the network for the lifetime of a channel, and a separate cell queue for each channel at every multiplexing point. We do not assume any reservation of resources, either memory or bandwidth; all data is statistically multiplexed. We also do not

assume any flow control within channels or any error recovery beyond checks for the integrity of protocol headers.

CELLSIM implements three multiplexing policies: *channel-per-conversation*, *channel-per-type*, and *channel-per-destination*. Channel-per-conversation multiplexing assigns a network-level channel to each application-level conversation flowing between two routers. Each conversation gets its own channel, and thus its own cell queue at each multiplexing point. Channel-per-type multiplexing assigns a channel to each type of traffic flowing between two routers. Conversations of the same type (FTP, NNTP, SMTP, or TELNET) flowing between two routers share the same channel and thus the same queues. Channel-per-destination multiplexing assigns only one channel for all traffic flowing between two routers. All conversations flowing between the same router pair share the same channel and thus the same queues.

2.1. Sources and Sinks

A source generates traffic for one application-level conversation at a time. In previous work, we extracted from wide-area Internet traces important characteristics of application-level conversations for each major type of wide-area network traffic (FTP, NNTP, SMTP, and TELNET)[3]. These per-conversation characteristics remained stable across three administratively and geographically separate sites at which traces were gathered, across more than one year of tracing activity, across different days of the week, and across different times of the day.

Our findings contradicted many common beliefs previously incorporated into models of wide-area data traffic. We found that bulk transfer conversations transfer less data than previously assumed; that they contain pauses in transmission due to control handshakes and thus do not transmit a continuous stream of data packets; that they are bidirectional rather than unidirectional; and that they transfer small packets in addition to large packets. We found that interactive conversations account for a higher percentage of traffic than previously believed; that they can be strongly unidirectional rather than symmetrically bidirectional; and that they transfer large packets in addition to small packets.

In related work, we derived new empirical workload models for driving wide-area internetwork simulations[5]. Our models are realistic, in that they accurately reproduces characteristics of real traffic; they are efficient, in that they is suitable for simulations of high-speed wide-area networks while consuming negligible amounts of computing resources; and they are network-independent, in that they are applicable to network conditions other than the ones prevalent when the traces were gathered. These models are embodied in *tcplib*[6], a library of software routines that we link with CELLSIM.

The simulator uses TCP[26] with the slow-start congestion avoidance and control algorithm[14] to manage the transmission of data generated by the models just described. CELLSIM implements features of modern TCP such as slow growth of windows up to a dynamic threshold, faster growth after the threshold is reached, abrupt window shutdown upon detection of a dropped packet, exponential backoffs in retransmission upon consecutive packet drops, and fast retransmits when duplicate acknowledgements arrive. CELLSIM also enforces a static limit on the transmission window size based on the window size advertised by the receiving end of a TCP connection.

Sinks simulate the destination ends of application-level conversations. 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 measured characteristics of the receiving end of real application-level conversations. Sinks also manage the destination end of the TCP sliding window.

2.2. Routers and Switches

Routers assign arriving packets to a channel based on the multiplexing policy in use and the conversation and type identifiers carried by the packet. A queue of cells is associated with each channel terminating in a router. When a packet arrives at a router from a host, it is fragmented into one or more cells, and these cells are placed as a group on the appropriate channel queue. Thus, data from conversations

that share a channel is interleaved in a packet-by-packet first-in first-out order. The set of channel queues is serviced by a round-robin server. Thus, data from conversations that do not share a channel is interleaved in a cell-by-cell round-robin order. The server dequeues a cell whenever the wide-area link is free for another transmission and there are any cells to transmit. Finally, the cells are transmitted over the link to the adjacent switch.

When cells arrive at a router from a switch, router operation is symmetric to the above. Packets are reassembled as the component cells arrive. When a packet is complete, it is demultiplexed to the appropriate sink based on its conversation identifier.

Switches accept cells from an incoming link and transfer them to an outgoing link based on a channel identifier carried by the cell. A queue of cells is associated with each channel using an outgoing link. The set of queues for an outgoing link is serviced by a round-robin server whenever the link is free for transmission and there are any cells to transmit. That is, data from different channels is interleaved in a cell-by-cell round-robin fashion. When data arrives to a router or switch and there is not enough buffer memory to queue it, the data is dropped.

2.3. Simulation Methodology

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

We further insured that our simulation results reflect more than just initial transients in network behavior. By simulating very long runs and plotting important output metrics over time, we determined that initial transients ceased before 1,000 round-trip times of simulation time. All simulation experiments reported in this paper ran for 4,000 round-trip times.

We also evaluated confidence intervals to bound the variations intrinsic to stochastic simulation. For each multiplexing scenario of interest, we ran a number of simulations with identical inputs except for the seeds to CELLSIM's random number generator. We then calculated the mean and standard deviation for the set of results to obtain 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.

2.4. Simulation Scenario

This section describes the simulation scenario we used for the bulk of our study. CELLSIM can be configured many ways through a rich set of input parameters, but we maintained many parameters constant to keep the number of simulations tractable. We later explored the sensitivity of our results to variations in important parameters, and verified that our results remain valid. We discuss this sensitivity analysis towards the end of this paper.

In the experiments described in the following sections, we configure CELLSIM to simulate a network like the one in Figure 1. This network has a number of traffic sources of various types, corresponding sinks, two routers, a switch, and the necessary links. Between simulation runs, we vary the multiplexing policy as well as the number and types of sources and sinks. Sources and sinks can be any of FTP, NNTP, SMTP, and TELNET. The multiplexing policy can be channel-per-conversation, channel-per-type, or channel-per-destination. Other relevant parameter choices are as follows.

All links have zero propagation delay, with one exception. The link between the switch and the router attached to the sources has a 22.5 millisecond delay, which simulates a U.S. cross-continental network with a 45-millisecond round-trip delay. In addition, all links are 1.5 Megabits per second. We match all link speeds so as not to limit how much a single source can load the network. We scale our simulation experiments to a relatively low but realistic speed to allow a small number of concurrent sources

to saturate the network, thus keeping our computation requirements down. We show that our results are insensitive to increases in link speed later in the paper.

We use 8 Kilobytes as the receiver window size. The receiver window size is an upper limit on the transport-level windows in the sources. Our choice matches the network round-trip window in our simulated network. This choice insures that sources are not window-limited, that is, that sources can transmit packets as fast as the network allows without being unnecessarily throttled by transport-level flow control.

Sources begin a new conversation as soon as the previous one ends. There are no idle periods between conversations, only idle periods within conversations. Interactive conversations contain long idle periods because they include think times and other human pauses in activity. Bulk transfer conversations contain idle periods in the order of the network round-trip time during which control handshakes take place. This choice of conversation interarrival pattern allows a small number of concurrent sources to saturate the network, again keeping computation requirements down.

Finally, we configure routers and switches with enough buffer memory to hold one network round-trip window of data for each conversation flowing through them. The round-trip window is the network's inherent storage capacity, or the product of its bandwidth and round-trip propagation delay. This choice ensures that a single source can make full use of the network bandwidth.

The network configuration just described stresses the multiplexing problem at the router attached to the sources. This router and the link to the its adjacent switch 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 network-level multiplexing policies on application-level performance. Simple scenarios such as this have been successfully used in previous simulation studies to uncover important phenomena[21,30].

3. Network Load

Figure 2 shows the load imposed on the network by the four traffic types modeled by CELLSIM. 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 channels, 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 application behavior reproduced by our traffic models. 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 each differently.

We have so far presented simulation results for a network that carries traffic of only one type. We now present results using a mix of bulk transfer and interactive traffic. In the simulations discussed below we use FTP as an example of a bulk transfer application. We maintain a single TELNET conversation while varying the number of simultaneous FTP conversations.

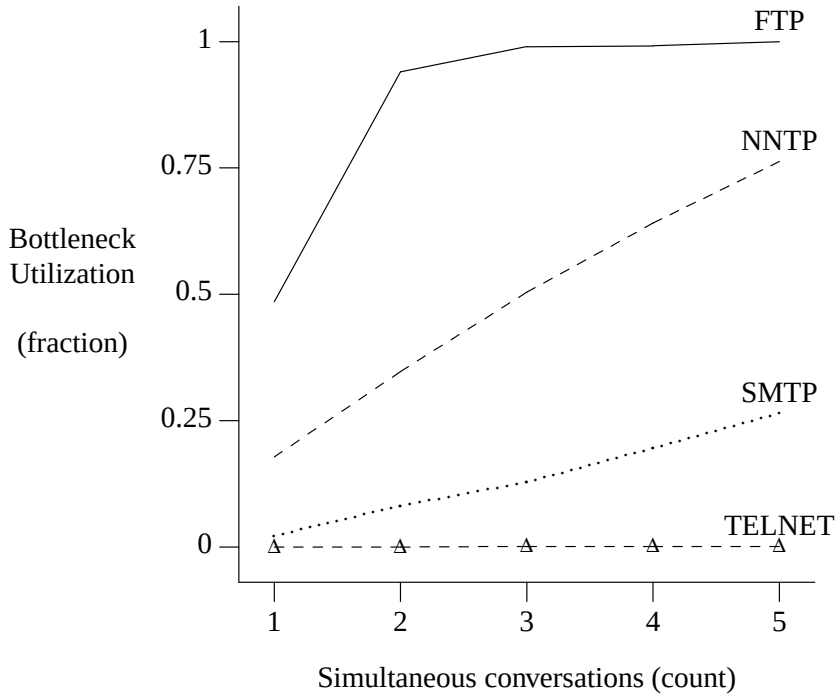

Figure 2. Load imposed on the network by each traffic type acting alone

4. Interactive Delay

Figure 3 shows the 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 concurrent FTP conversations. Figure 4 shows the standard deviation of delay observed by interactive applications under the same conditions.

As shown, per-type channels and per-conversation channels exhibit good performance under load while per-destination channels 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 channels. 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 channels. In wide-area networks with substantial round-trip times, for example a cross-continental network like the Internet backbone, this performance degradation is readily noticeable by human users and is not acceptable. Similarly, for all three policies, the standard deviation of delay begins very near zero. This low level of variability remains constant with increased load under per-type and per-conversation channels. However, variability increases rapidly with load under per-destination channels.

The reasons for the difference in performance are as follows. Interactive sources mostly send widely-spaced small packets, while bulk transfer sources attempt to send relatively large amounts of data at once. The multiplexing policies using per-type and per-conversation channels 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 and regular service regardless 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 channels indiscriminately mixes the two

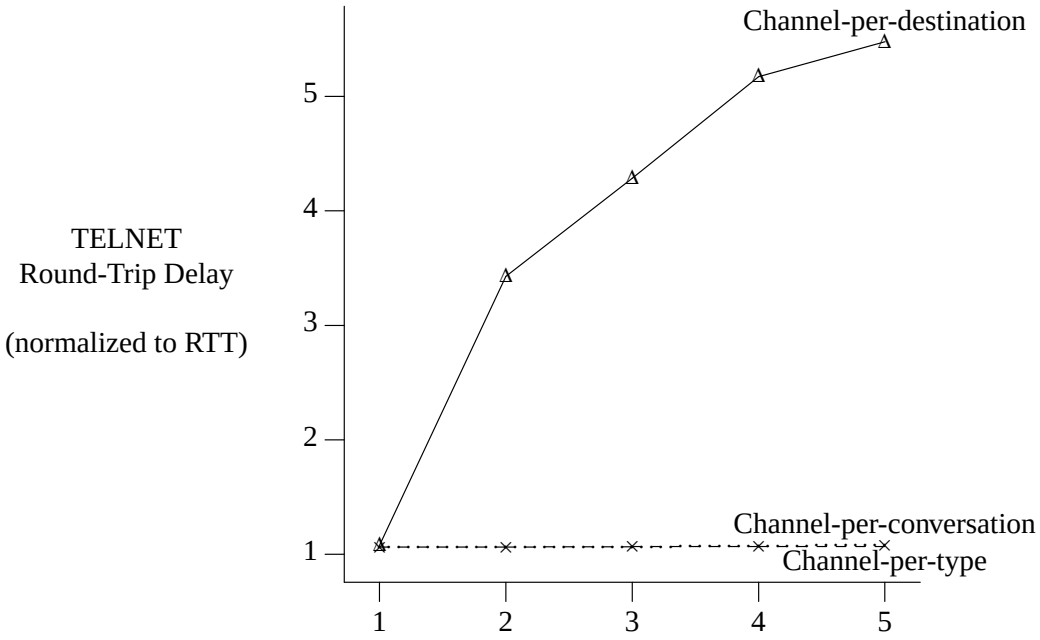

Figure 3. Interactive delay under different multiplexing policies

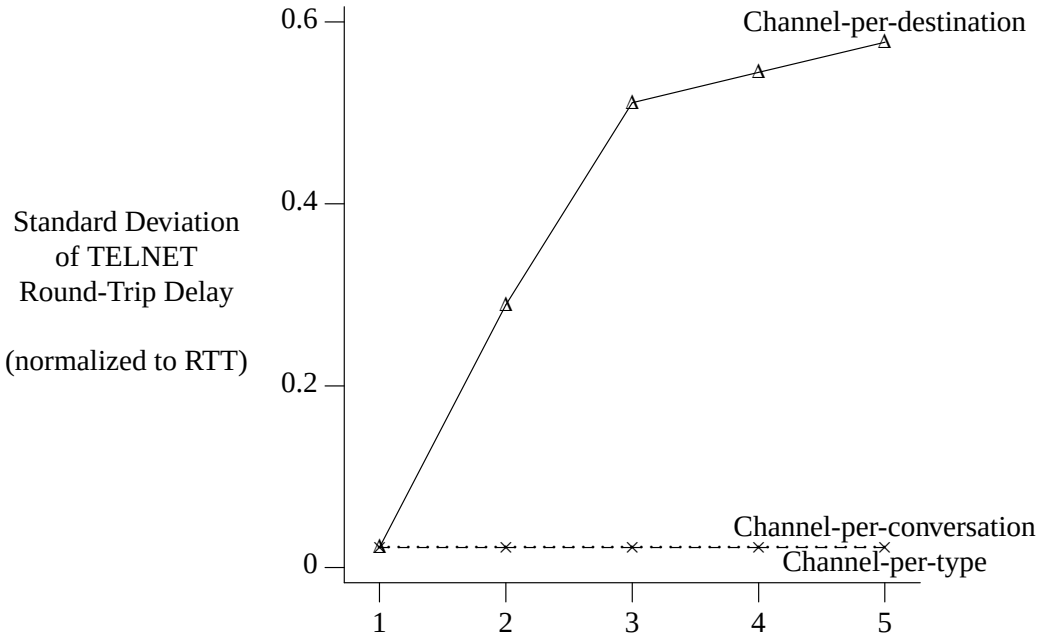

Figure 4. Variability of interactive delay under different multiplexing policies

types of traffic in a shared first-in first-out queue. Under load, interactive traffic is queued behind varying amounts of bulk transfer traffic, thus incurring longer and more variable delays.

5. Bulk Transfer Throughput

5.1. Average Throughput to Bulk Transfer Conversations

Figure 5 shows the average throughput obtained by FTP conversations under different multiplexing policies 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.

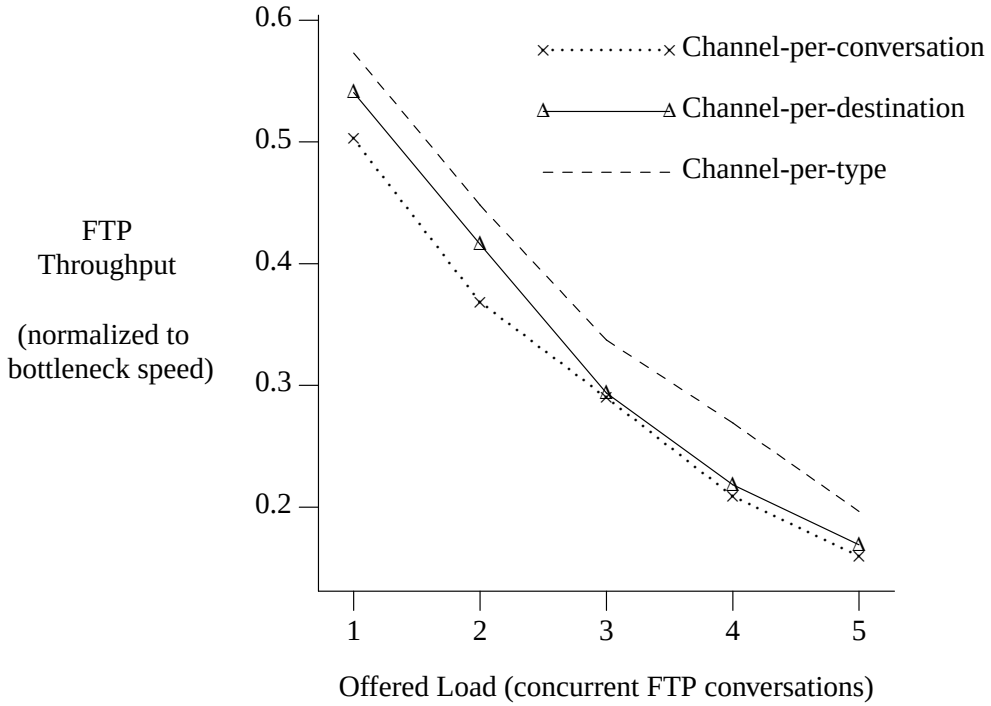

Figure 5. Bulk transfer throughput under different multiplexing policies

We see that all three multiplexing policies exhibit similar 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 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 2. 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, the bottleneck link saturates, queues build up, and queuing delays rise, regardless of the multiplexing policy. The transport-level protocol in the sources only sends new data when acknowledgements return. With higher delays, acknowledgements take longer to return to sources. The transport-level protocol sends new data at a slower rate, and average throughput decreases for all sources 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.

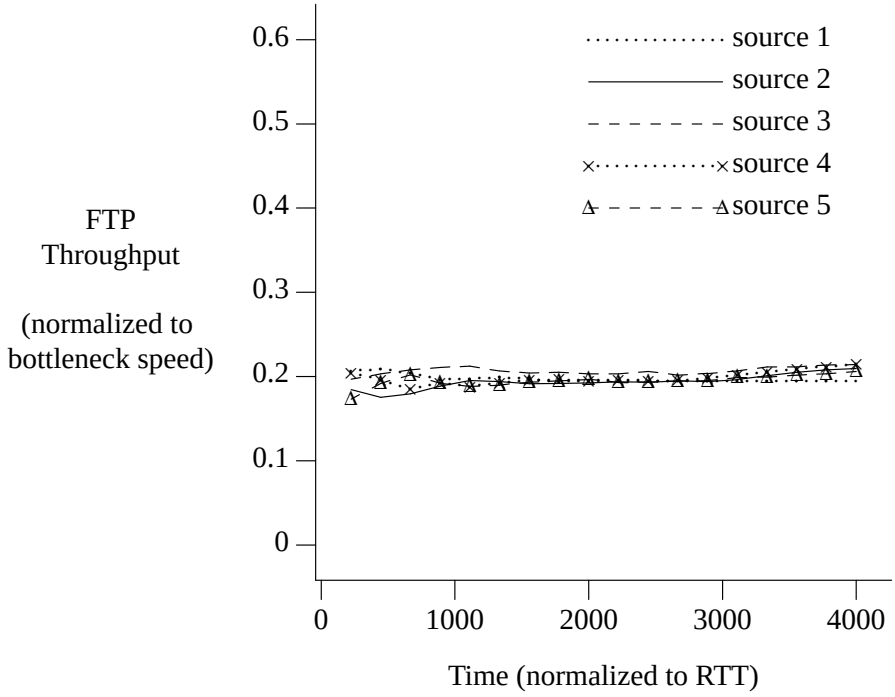

Figure 6. Fairness in bulk transfer throughput under channel-per-conversation multiplexing

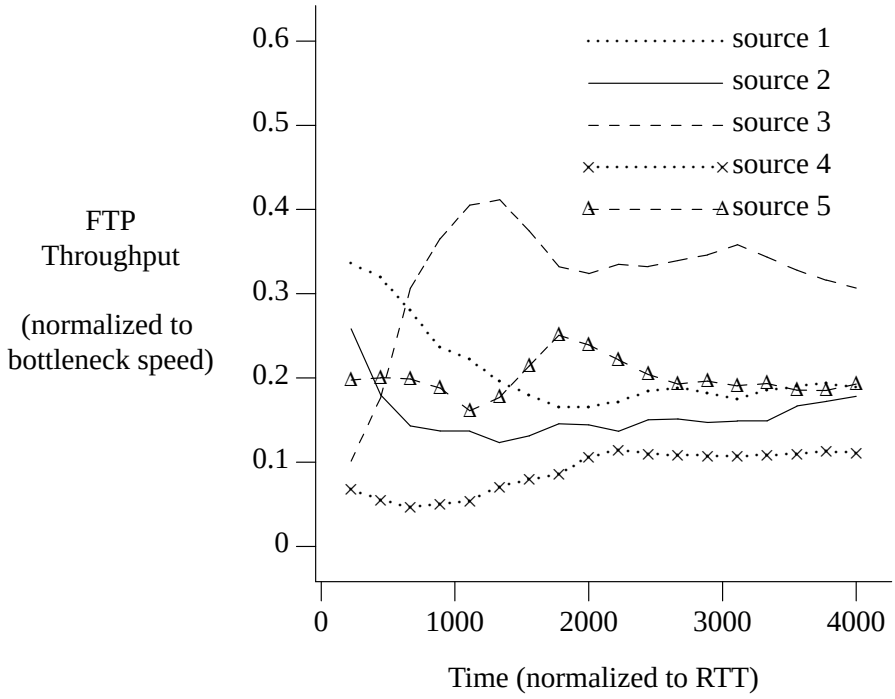

Figure 7. Fairness in bulk transfer throughput under channel-per-destination multiplexing

5.2. Fairness in Throughput to Bulk Transfer Conversations

Figures 6 and 7 show the throughput obtained over time by each of five simultaneous FTP conversations, under per-type channels and under per-type channels, respectively. The results for per-type channels are very similar to those for per-destination channels, and we leave it out of the discussion below. As shown, per-conversation channels divide network resources fairly and provide each conversation with the same level of throughput. In contrast, per-type channels provide some conversations with better throughput than others.

The differences in throughput are caused by the interaction of multiplexing policies in the bottleneck router and the transport-level protocol in the sources. Per-conversation channels give each conversation its own queue and service the set of queues on a round-robin basis. In this case, all sources experience similar delays, they receive acknowledgments at similar rates, and they transmit new data at similar rates. The result is nearly identical throughput for all sources.

On the other hand, per-type channels 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, and they continue to send new data. The losing conversations see higher delays, their acknowledgements take longer to return, and they send new data at a slower rate. The result is higher throughput for the winners than for the losers.

The losing conversations may also suffer packet losses due to lack of buffer memory inside the network. Since their data arrives last at queueing points, their data is more likely to be dropped. Packet losses cause sources to stop sending new packets, retransmit the lost packet, and wait for it to be acknowledged. Because of the randomness built into retransmission strategies and natural pauses in transmission within a conversation, losers can become winners, causing previous winners to become losers. These phase dynamics are apparent in Figure 7, where different sources obtain preferred treatment at different times. Segregation of traffic sources into winners and losers, as well as the oscillations just noted, have been noted in previous studies of TCP behavior[9,30].

5.3. Behavior of Transport-Level Windows

For another look at the dynamics behind the observed differences in bulk transfer throughput, we turn to the behavior of transport-level windows. Figures 8 and 9 show the TCP window size over time for a sample FTP conversation, under per-conversation channels and per-destination channels, respectively. The window behavior for per-type channels is very similar to that for per-destination channels and we leave it out of the discussion below.

As shown, under per-conversation channels, window sizes increase steadily until they reach their maximum value and remain there. These sources are able to make full use of their fair share of the bandwidth. In contrast, under per-type channels, window sizes often shrink to their minimum value before growing again. When their windows shrinks, these sources cannot make full use of the available bandwidth until their windows again reach their maximum value. To make matters worse, the slow-start algorithm grows windows very slowly after a retransmission. This window behavior under per-destination and per-type channels contributes to the segregation phenomenon described above.

The following summarizes our results regarding the three multiplexing policies. Channel-per-conversation multiplexing offers low delay and low variability of delay to interactive applications, while providing a fair share of the bandwidth to bulk transfer applications. Channel-per-type multiplexing exhibits good interactive delay characteristics, but suffers from unfairness in the throughput it provides bulk transfer applications. Channel-per-destination multiplexing suffers from bad performance in both delay and throughput metrics. Therefore, channel-per-conversation is the recommended multiplexing policy.

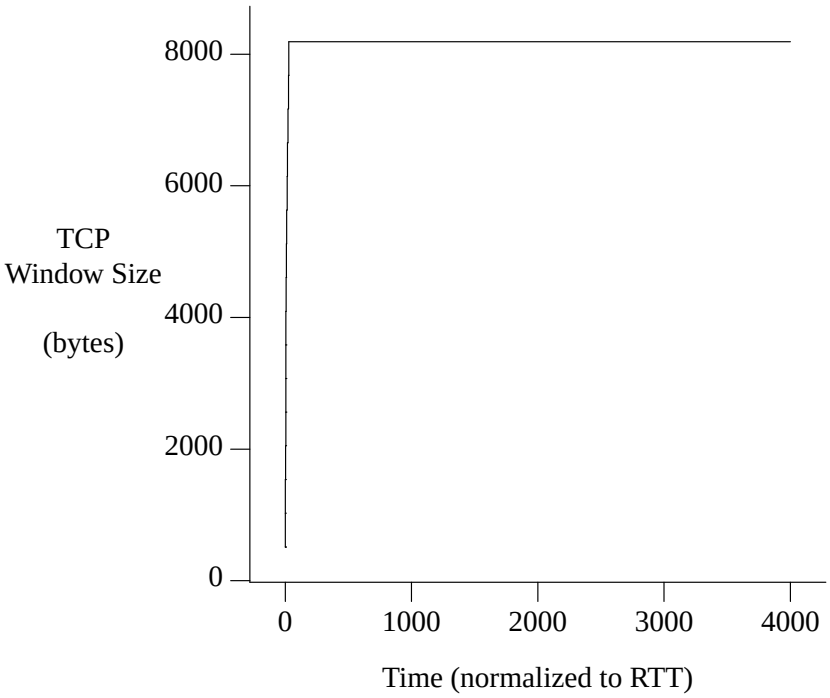

Figure 8. Behavior of TCP windows in FTP sources under channel-per-conversation multiplexing

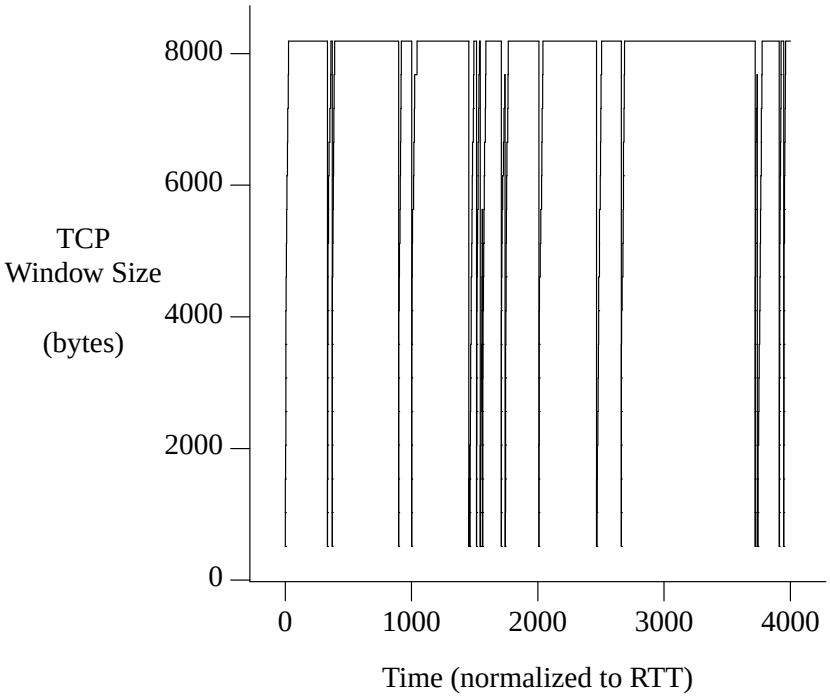

Figure 9. Behavior of TCP windows in FTP sources under channel-per-destination multiplexing

6. Sensitivity to Network Speed

Our results are insensitive to increases in network speed. The simulations described above used a relatively low link speed (1.5 Megabits per 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 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 Megabits per 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). All of the multiplexing phenomena reported earlier remained intact under the new conditions.

7. Sensitivity to Bulk Transfer Size

The conclusion that networks should use channel-per-conversation multiplexing proved insensitive to increases in bulk transfer size. We modified *tcplib* to remove an anomaly involving 2,315 FTP connections that each transferred only 74 bytes, all but three of which were between the same two hosts, and 95% of which came between 30 and 45 seconds apart[25]. The anomaly had biased the FTP conversation model in *tcplib* towards conversations that transferred less data than FTP conversations in real networks. After removing the anomaly, we found that the unfairness in bulk transfer throughput under channel-per-type and channel-per-destination multiplexing worsened because the larger bulk transfer size increased queueing delays for conversations that shared channels. At the same time, channel-per-conversation multiplexing retained its low delay, low variability of delay, and fair throughput characteristics.

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 would intensify the fairness shortcomings of multiplexing policies that do not separate application-level conversations, increasing the desirability of channel-per-conversation multiplexing.

8. Ease of Implementation

The channel-per-conversation multiplexing policy is simple to implement in a connection-oriented ATM network. Such networks route cells along virtual circuits much like the channels that CELLSIM simulates. It is simple to allocate a queue to each virtual circuit at every multiplexing point during virtual circuit establishment. Since ATM cells carry a virtual circuit identifier, it is also simple to place cells in the appropriate queue when they arrive at a multiplexing point. Virtual circuit networks such as Datakit[11] have supported similar schemes for some time.

A round-robin server for queues of fixed-size cells is also simple to implement. One particularly efficient technique maintains a separate queue of tokens that identify non-empty cell queues. When the token queue is empty, there are no cells to transmit. When a cell is placed on a previously empty cell queue, an identifier for that queue is added to the back of the token queue. To determine which cell to transmit next, the server dequeues the token at the front of the token queue, dequeues a cell from the identified cell queue, and transmits the cell. If the cell queue is left empty by the previous operation, the server discards the token and revisits the front of the token queue to obtain a new token. Otherwise, the server places the old token in the back of the token queue and then revisits the front. The server never scans the set of cell queues looking for a cell to transmit, a significant optimization in networks that support thousands of virtual circuits, such as current and proposed wide-area ATM networks. Switches in the

Xunet 2 wide-area ATM network contain hardware that can carry out the above queue management scheme at 45 Megabits per second[12].

Allocating a network-level channel to every application-level conversation requires that the ATM network identify individual conversations. Our traffic analysis software accomplished this task by inspecting each TCP/IP packet. It reached into the stack of protocol headers for IP addresses that identify hosts and TCP port numbers that identify conversations within hosts. An ATM network network can do the same when it receives a TCP/IP packet from a host. This procedure violates strict protocol layering, but it is necessary in an internetwork where transport-level protocols are not closely integrated with network-level protocols. The problem is common to other multiplexing schemes that act on individual application-level conversations, for example Fair Queueing[8]. The problem is somewhat simplified if the ATM network reaches to the host. In that case, both TCP/IP and ATM-related protocols reside on the host. This closer coupling of transport-level and network-level protocols facilitates their exchange of information regarding individual conversations. ATM is rapidly gaining popularity for local-area network use[2,4,7,17,28], so that future wide-area ATM networks will indeed reach to individual hosts.

A major motivation behind ATM is its ability to support traffic from non-traditional sources, such as audio and video, as well as traffic from the traditional TCP sources discussed in this paper. Non-traditional traffic sources have strict performance requirements that require that each conversation be isolated from others. Isolating TCP conversations by following a channel-per-conversation multiplexing policy would thus fit well with the overall operation of an ATM network.

9. Conclusions

We have shown that a wide-area cell network that carries data traffic should separate application-level conversations. It should assign each conversation its own queue at every multiplexing point and serve those queues in a cell-by-cell round-robin basis. It should not separate conversations to a lesser degree and use first-in first-out queueing disciplines. The channel-per-conversation multiplexing policy provides low delay and low variability of delay to interactive conversations, and a fair share of the available bandwidth to bulk transfer conversations. The policy is robust in the face of high loads, increasing network speeds, and larger bulk transfers. Furthermore, it is simple to implement in a connection-oriented cell network.

Our results are of particular relevance to ATM networks that carry TCP traffic. ATM is an international standard with broad support from telecommunications providers and computer manufacturers alike. At the same time, TCP with slow-start will continue to be used widely due to its proven functionality and immense installed base. Future ATM networks should incorporate the recommendations made in this paper in order to efficiently provide good performance and fair treatment to TCP conversations.

Acknowledgements

Fred Douglis, Domenico Ferrari, Sandy Fraser, Brian Marsh, and John Ousterhout commented on previous versions of this material. Peter Danzig and Sugih Jamin provided *tcplib*. Chuck Kalmanek and Sam Morgan provided NETSIM.

References

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

2. 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.

3. 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.

4. E. Cooper, O. Menzilcioglu, R. Sansom, and F. Bitz, ''Host Interface Design for ATM LANs,'' *16th Annual Conference on Local Computer Networks*, 1991.
5. 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*, vol. 3, no. 1-26, 1992.
6. P. B. Danzig and S. Jamin, ''tcplib: A Library of TCP Internetwork Traffic Characteristics,'' *Technical Report USC-CS-91-495*, University of Southern California, Los Angeles, California, September, 1991.
7. B. S. Davie, ''A Host-Network Interface Architecture for ATM,'' *Proc. of ACM SIGCOMM '91*, Zurich, Switzerland, September, 1991.
8. A. Demers, S. Keshav, and S. Shenker, ''Analysis and Simulation of a Fair Queueing Algorithm,'' *Proc. of ACM SIGCOMM '89*, Austin, Texas, September, 1989.
9. S. Floyd and V. Jacobson, ''Traffic Phase Effects in Packet-Switched Gateways,'' *Computer Communication Review*, vol. 21, no. 2, April, 1991.
10. A. G. Fraser and S. P. Morgan, ''Queueing and Framing Disciplines for a Mixture of Data Traffic Types,'' *AT&T Bell Laboratories Technical Journal*, vol. 63, no. 6, July-August, 1984.
11. A. G. Fraser, ''Towards a Universal Data Transport System,'' *IEEE Journal on Selected Areas in Communication*, vol. 1, no. 5, November, 1983.
12. 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.
13. E. L. Hahne, ''Round Robin Scheduling for Max-Min Fairness in Data Networks,'' *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 7, September, 1991.
14. V. Jacobson, ''Congestion Avoidance and Control,'' *Proc. of ACM SIGCOMM '88*, August, 1988.
15. 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)
16. M. G. H. Katevenis, ''Fast Packet Switching and Fair Control of Congested Flow in Broadband Networks,'' *IEEE Journal on Selected Areas in Communication*, vol. 5, no. 8, October, 1987.
17. I. Lesley and D. R. McAuley, ''Fairisle: An ATM Network for the Local Area,'' *Proc. of ACM SIGCOMM '91*, Zurich, Switzerland, September, 1991.
18. M. Lottor, ''Internet Growth (1981-1991),'' *RFC 1296*, Network Information Center, SRI International, Menlo Park, California, 1992.
19. S. E. Minzer, ''Broadband-ISDN and Asynchronous Transfer Mode (ATM),'' *IEEE Communications*, vol. 27, no. 9, September, 1989.
20. J. C. Mogul, ''Observing TCP Dynamics in Real Networks,'' *Proc. of ACM SIGCOMM '92*, Baltimore, Maryland, August, 1992.
21. S. P. Morgan, ''Queueing Disciplines and Passive Congestion Control in Byte-Stream Networks,'' *IEEE Transactions on Communications*, vol. 39, no. 7, July, 1991.
22. S. P. Morgan and C. Y. Lo, ''Mean Message Delays for Two Packet-FIFO Queueing Disciplines,'' *IEEE Transactions on Communications*, vol. 38, no. 6, June 1990.
23. J. Nagle, ''On Packet Switches with Infinite Storage,'' *IEEE Trans. on Communications*, vol. COM-35, 1987.

24. V. Paxson, ‘‘Growth Trends in Wide-Area TCP Connections,’’ *submitted to ACM SIGCOMM ’93*, February, 1993.

25. V. Paxson, ‘‘Empirically-Derived Analytic Models of Wide Area TCP Connections,’’ *unpublished technical report*, Lawrence Berkeley Laboratory, Berkeley, California, November, 1992.

26. J. Postel, ‘‘Transmission Control Protocol,’’ *RFC 793*, Network Information Center, SRI International, Menlo Park, California, 1981.

27. B. Shneiderman, *Designing the User Interface*, Addison-Wesley, 1987.

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

29. 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*, vol. 21, no. 2, April, 1991.

30. 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.