An Empirical Workload Model for Driving Wide-Area TCP/IP Network Simulations

Peter B. Danzig* Sugih Jamin* Ramón Cáceres† Danny J. Mitzel* Deborah Estrin*

*Computer Science Department, University of Southern California,
Los Angeles, California 90089-0781

†Computer Science Division, University of California,
Berkeley, California 94720

traffic@excalibur.usc.edu

Abstract

We present an artificial workload model of wide-area internetwork traffic. The model can be used to drive simulation experiments of communication protocols and flow and congestion control experiments. The model is based on analysis of wide-area TCP/IP traffic collected from one industrial and two academic networks. The artificial workload model uses both detailed knowledge and measured characteristics of the user application programs responsible for the traffic. Observations drawn from our measurements contradict some commonly held beliefs regarding wide-area TCP/IP network traffic.

The simulation techniques presented here will be useful in studying congestion control, routing algorithms, and other resource management schemes, for existing and future networks.

Keywords: Wide-Area Networks, Artificial Workload Models, Trace Analysis, Simulation, TCP/IP Internetworks

1. Introduction

When simulating new congestion control, flow control, and adaptive routing algorithms one needs to model the overall pattern of traffic flowing through the network, from distribution of packet sizes and interarrival times, to characteristics such as distribution of host reference patterns and direction of traffic flow. This paper presents an artificial workload model of wide-area network traffic based upon application-level analysis of wide-area TCP/IP [1] traces collected on two campus networks, the University of Southern California (USC) and the University of California at Berkeley (UCB), and one industrial research site, Bellcore.

This research was supported by an equipment grant from the Charles Lee Powell Foundation. Ramón Cáceres was supported by the NSF and DARPA under Cooperative Agreement NCR-8919038 with CNRI, by AT&T Bell Laboratories, Hitachi, a University of California MICRO grant, and ICSI. Danny Mitzel was supported by a fellowship from Hughes Aircraft Company.

The artificial workload model consists of a set of stub-network-specific arrival processes for new conversations between application programs, and stub-network-independent artificial workload model of each of these applications. The model is based on detailed knowledge and certain measured characteristics of those applications responsible for a significant percentage of wide-area traffic on the Internet. Since TCP packets constituted roughly 80% of the wide-area network traffic that we collected,[1] we restrict our model to TCP-based applications. Nevertheless, the model is not TCP-specific. We eliminated dependencies on the transport protocol so that the model could be used to study new transport protocols.

We selected to work on Internet traffic because it is the most successful example of a large and rapidly growing *internetwork*. To appreciate the size of the Internet, consider that more than 500,000 hosts are registered in the Internet domain name system [2] and that the NSFnet backbone connects more than 2,300 university, industry, and government networks [3], To this vast internetwork, large number of subnetworks are being attached almost on a 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 [4, 22].

Future broadband wide-area networks will probably transfer larger amounts of data and carry a mix of traffic currently not found on the Internet. We believe this does not trivialize our present study, for several reasons. First, it will be several years before the current traffic mix changes appreciably. Second, as it changes, it will not obviate the existence of traditional traffic. Third, the workload model and many of the observations that we made while constructing it will be useful in evaluating new algorithms for internetworks in general; and the modeling approach we used can be applied to other types of internetwork. We encourage fellow researchers to construct and publish artificial workload models of their networks.

[1]For the UCB data, UDP packets make up 16% of all network traffic, while ICMP packets account for only 1% of all traffic. Of all UDP packets, 63.63% belongs to DNS, 15.82% to ROUTE, and 10.51% to NTP.

Previous traffic studies of TCP/IP have examined the statistics of the aggregated packet arrival process on local area networks [9, 10, 11], at border routers [12], and inside a wide-area backbone [13]. These studies have shown that packet interarrival times are not Poisson, but rather follow a *packet-train* model. The *packet-train* model has proven valuable in the study of packet routers design [14, 15].

The study presented in this paper is different from all the studies mentioned above. Instead of confining ourselves to the network and transport layers, we studied the characteristics of key applications responsible for 96% of wide area TCP traffic. We believe these applications are representative of applications currently running on wide-area networks other than the Internet. Two related studies, one at University College London and the other at Lawrence Berkeley Laboratory, identified a similar set of applications as being responsible for most of their wide–area TCP traffic [16, 17].

The decision to characterize application traffic was supported by the following observations. Measured interarrival times alone are not adequate to characterize conversations for the purpose of driving flow and congestion control algorithm simulations; interarrival times are themselves a function of existing flow control mechanisms. Interarrival times may be useful in characterizing interactive traffic due to its low volume. However, bulk traffic must be characterized by the amount of data transferred; its duration mostly reflects network link speed and flow control algorithms. Furthermore, although interactive conversations are bidirectional, they send much more data in one direction than in the other; an accurate model must take this into account. Finally, some applications converse with more networks than do others (see Figure 8).

From these observations, we concluded that researchers would benefit from more realistic traffic models, particularly in studying switching and control mechanisms through simulation. When simulating wide-area networks, current practice ignores the distribution of number of bytes transmitted, the bidirectionality of bulk traffic sources, and the duration of interactive connections., instead idealized FTP [5] and TELNET [6] sources are used; the idealized FTP sources send huge quantities of data in one direction, while the idealized TELNET sources send a

Poisson stream of small packets in one or both directions [7, 8]. This paper makes the first step towards an internetwork source model. It outlines the necessary steps to describe and simulate a new conversation between two networks. However, it does not seriously investigate the question of when to establish a conversation between two networks; we are currently addressing this problem.

2. Measurement and Analysis Methodology

Below we describe the data collection methods, their associated loss rates, and our definition of a conversation.

2.1. Data Collection Sites

Wide-area traffic data was collected at two university campuses (UCB and USC) and one industrial research laboratory (Bellcore). The data collected at UCB traced all traffic between the campus and the Bay Area Regional Research Network (BARRNET); data collected at USC traced all traffic between the campus and Los Nettos; and data collected at Bellcore traced all traffic between their Morristown laboratory and the John von Neumann Center Network (JVNCNET).

2.2 Trace Contents

We analyzed a sample of 5,891,622 TCP packets from UCB, 5,221,036 from USC, and 1,703,269 from Bellcore. These packets correspond to one day of continuous trace collection at UCB and USC, and three days of continuous tracing at Bellcore. The collection started at 10:20 on Tuesday, October 31, 1989 at UCB, 14:24 on Tuesday, January 22, 1991 at USC, and 14:37 on Tuesday, October 10, 1989 at Bellcore.

Each trace record consists of a time stamp and the first 56 bytes of raw network data. The time stamp records the arrival time of the packet at the tracing apparatus. The 56 bytes of data

hold the packet headers from the datalink layer (Ethernet), the network layer (e.g. IP), and the transport layer (e.g. TCP and UDP).[2]

2.3. Tracing Instrumentation and Packet Loss Rate

The UCB data was collected with a Sun 3 workstation equipped with a microsecond timer [18]. The resulting time stamp resolution was 10 microseconds. The workstation ran a modified Unix kernel with a circular buffer big enough to hold 128 full-size Ethernet packets. A dedicated user program transferred trace records from this buffer to tape. No packet losses due to buffer overflows were detected during the UCB measurements. The packet loss rate induced by separate stress testing was less than 5% in the worst case.

The USC data was collected using the NNStat program suite [19] on a Sun SparcServer 4/490. The NNStat program uses the Sun *gettimeofday()* system call which has a 20-millisecond resolution. During similar measurements, we estimated the loss rate by sending a Poisson stream of *ping* packets. We observed that 0.6% of these packets were missing from the tape.

The Bellcore data was collected using a Sun 3 workstation augmented with a microsecond interval timer and a single-board computer dedicated to collecting and time-stamping trace packets. The time-stamps have a 10 microsecond resolution. A hierarchical system of double buffering carried the trace records from the single-board computer to tape. No packet loss was detected anywhere in the monitoring system during the Bellcore measurements [11].

2.4. Are the Traces Representative?

Both USC and UCB campuses use mostly UNIX and IBM computing systems. Bellcore uses mostly UNIX systems. We believe that the systems traced are representative of sites currently attached to the Internet, and that our analysis also applies to other sites. However, we recognize

[2]We did not encounter any packets with IP protocol options. We do take into account the length of any TCP option present when calculating the size of the data part of a packet. In the UCB trace, we found 0.02% of the IP packets carrying TCP data to be IP fragments. For USC, the number was 0.05%, and for Bellcore, the number was 0.02%. We ignored packets that have their *more fragment* flag set.

that traces collected at other sites might show a different application breakdown than the ones reported here. Future studies will be conducted to further validate our results.

The breakdown of traffic varies greatly from site to site (see Table 1) and should be accounted for when simulating sequences of conversations. However, the *characteristics* of conversations, shown later, are essentially identical between the three sites, even though the USC trace was collected one year and three months after the others. Furthermore, these characteristics are shared by two different days of UCB traces, and by a one-day trace and a three-day trace of Bellcore traffic. That is, the distributions of number of bytes transferred, conversation durations, total packets per conversation, and packet sizes are indistinguishable. For legibility, we present only UCB data in the body of the paper. Appendix 1 contains representative figures comparing data from the three sites.

Traffic Type	% Packets			% Bytes			% Conversations		
	UCB	USC	Bell	UCB	USC	Bell	UCB	USC	Bell
ftp (ctrl+data)	**12.0**	**5.0**	**18.7**	**36.2**	**10.6**	**54.9**	**2.8**	**3.2**	**4.8**
shell (rcp)	0.2	3.6	1.4	0.4	12.5	4.3	0.2	0.2	0.7
smtp	**11.6**	**3.1**	**12.6**	**11.0**	**1.9**	**10.6**	**69.6**	**52.3**	**68.0**
dc_10	—	3.5	—	—	0.8	—	—	1.5	—
vmnet (bitnet)	10.0	9.1	—	25.4	20.7	—	0.1	3.3	—
uucp	0.2	0.1	0.8	0.4	0.1	1.3	0.3	1.1	2.2
nntp	**11.6**	**36.3**	**9.2**	**15.8**	**44.5**	**15.6**	**0.4**	**1.6**	**0.8**
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**
x11	0.2	5.0	0.4	0.2	2.5	0.1	—	0.5	0.4
ircd	4.6	—	—	1.3	—	—	0.6	0.2	—
finger	1.1	0.4	0.5	0.6	0.2	0.2	18.3	17.8	7.6
domain	0.1	0.1	—	—	0.2	—	0.2	3.3	0.1
other	4.9	11.3	1.6	0.4	3.1	3.1	1.3	3.6	2.4

Table 1: Breakdown of unidirectional TCP traffic

NSFnet statistics for the total packet activity generated by stub-network also suggest that our traces are representative [4, 22]. The six applications identified in Table 1 as transmitting the largest number of packets (TELNET, RLOGIN, FTP, SMTP [20], NNTP [21], and VMNET) are also the TCP applications most often encountered on the NSFnet backbone. (Table 2 provides a short glossary of Internet protocols and applications.) Of the 1,174 stub-networks for which backbone activity was detected during October 1989, UCB was the 3rd busiest, and Bellcore was the 130th

busiest. Of the 2,345 active stub-networks measured during January 1991, USC was the 31st busiest. These numbers indicate that our traces capture activity generated by a range of stub-networks, from a very active one like UCB, to a moderately active one like USC, to a less active but non-trivial one like Bellcore.

DC_10	Cadre Teamwork Mailbox 10
DNS	Domain Name Service, host name resolution protocol
DOMAIN	Domain Name Service
FINGER	User information query application
FTP	File Transfer Protocol
ICMP	Internet Control Message Protocol
IP	Internet Protocol, a network layer datagram protocol
IRCD	Internet Relay Chat Program Server, a tele-conferencing application
NTP	Network Time Protocol
NNTP	Network News Transfer Protocol
RLOGIN	Remote login application
ROUTE	Routing Information Exchange Protocol
SHELL	Remote shell application, often used for remote copy (rcp) operations
SMTP	Simple Mail Transfer Protocol
TCP	Transmission Control Protocol, a reliable transport layer protocol
TELNET	Remote terminal application
UDP	User Datagram Protocol, an unreliable transport layer protocol
UUCP	Unix to Unix Copy Program, used for mail, news, and file transfer
VMNET	A method of running the RSCS protocol (usually from IBM mainframes running VM) on top of TCP; it is used to handle a majority of the BITNET backbone traffic
X11	X window system

Table 2: Internet Protocols and Applications

2.5. Traffic Pattern Analyzer

We wrote a traffic pattern analyzer to reduce the raw packet trace data and produce the probability distributions employed by the application-specific workload models. One of the first decisions we had to make was how to break up the trace into meaningful units. Should we adopt the *packet-train* model or should we maintain a state machine per TCP connection? We look at these alternatives below.

The *packet-train* model has largely replaced earlier Markov models of network traffic [9, 10, 13]. 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 interarrival gaps. Different researchers have used different MAIGs, ranging from 500 milliseconds to 50 seconds, depending on the network measured.

In contrast, we divided up the traffic into application-level *conversations*. We define a *conversation* to be a stream of packets travelling between the end points of an association, delimited by a twenty-minute silence; an association is in turn defined as a <protocol, source address, source socket, destination address, destination socket> tuple. A twenty-minute silence 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. Alternatively, we could have detected the TCP connection establishment handshakes between a source and destination pair and used them to determine the beginning and end of a conversation. This required maintaining a state machine and associated timers for every live connection. Lack of memory space prevented us from doing so.

In the case of FTP, conversations can subsume multiple TCP connections. We clumped several TCP connections into one conversation because each FTP session initiates one FTP-control and zero or more FTP-data connections. We also clumped back-to-back and concurrent FTP sessions between the same source-destination IP-address pair into one conversation. Similarly for NNTP, a new TCP connection is set up for each article sent to the initiator of the conversation—though not vice versa. We clumped back-to-back and concurrent TCP connections used by NNTP.

While FTP uses a control connection to send control packets and a separate data connection for each file transferred, NNTP can use one connection for both handshaking and sending of multiple news articles. Since we wanted the distribution of the number of articles sent per NNTP conversation, and the distribution of article sizes, we needed a scheme for distinguishing individual articles from packets used for handshaking. We had to recognize the first and last packets of an article. From the NNTP documentation and by sampling several 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 possible, 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 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 transfer. Notice that if the last packet of an article arrived out of sequence an error would be introduced.

Since we want to model the characteristics of transport layer traffic in general, independent of TCP itself, 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.[3] 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 analysis of the traces was at position 104 into the buffer. Since we threw away retransmissions, we also threw away most of the keep-alive packets, which share a single sequence number. This also meant that every now and then we saw a lonesome keep-alive as a conversation transferring a single 1-data-byte 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 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.

3. Characterization of Application Conversations

We now detail several interesting properties about the applications responsible for most wide area TCP/IP traffic. In Section 4 we describe the application-specific artificial workload models. The observations below are presented under five general categories: traffic breakdown, bulk data

[3]Retransmitted packets accounted for between 0.3% to a little below 3% of all packets belonging to an application.

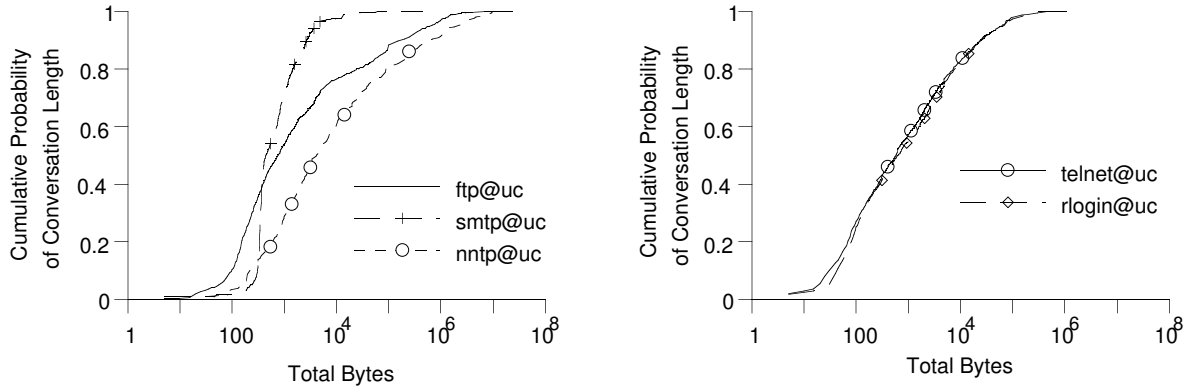
transfer applications, interactive applications, traffic flow, and wide-area network locality. We are interested in such questions as:

- How does TCP traffic break down into interactive and bulk traffic?
- How "bulky" is the data transferred by bulk applications?
- What are the characteristics of interactive applications in terms of bytes transferred, burstiness, duration, and interarrival time?
- Is traffic flow unidirectional or bidirectional?
- Is there network-pair locality on wide-area networks and how many concurrent conversations are there between such network pairs?

Table 3 summarizes our most important observations about TCP traffic.

Depending on the application, 60-90% of the conversations categorized as bulk transfer send less than 10 kilobytes of data. Bulk transfer is request-response in nature, with responses larger than 1 megabyte responsible for only 15-20% of all bytes transferred.
A large portion of bulk transfer applications, which are responsible for more than 50% of observed network traffic, show bidirectional traffic flow.
Over 90% of interactive conversations send fewer than 1,000 packets and 50% of interactive conversations last less than a minute and a half. Packets belonging to interactive applications are mostly smaller than 512 bytes.
Interactive applications can generate 10 times more data in one direction than the other, using packet sizes ranging from 1 byte to 512 bytes.
A uniform plus exponential distribution best models interarrival times of packets belonging to interactive applications.

Table 3: Selected Observations.



(a) Bulk transfer (b) Interactive applications

Fig. 1: Total bytes transferred per unidirectional conversation.

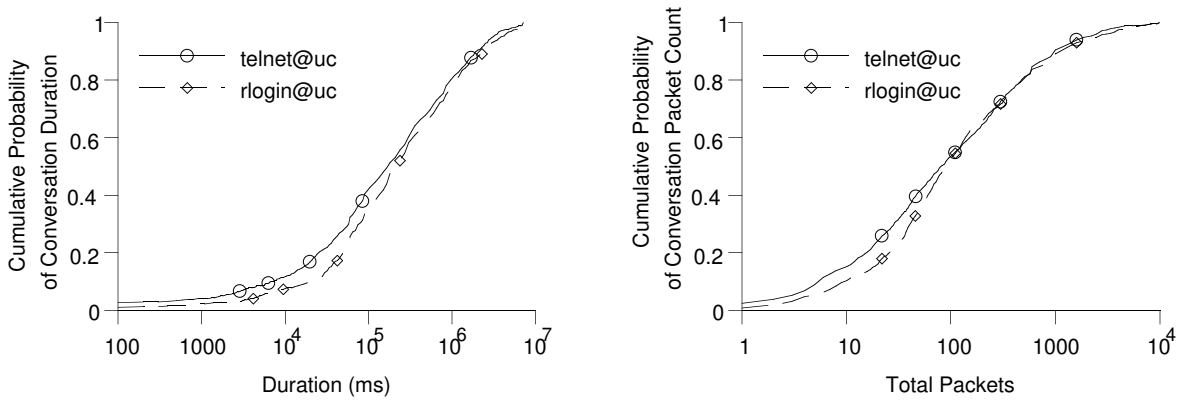
3.1. Traffic Breakdown

For lack of a more accurate workload model, previous 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 [7, 23, 22, 25, 26, 27]. These studies either used a continuous bulk transfer or an arbitrary mix of bulk and interactive traffic.

TCP traffic consists of bulk and interactive traffic as commonly assumed. Table 1[4] shows the distribution of number of bytes, packets, and conversations attributed to each application. Even though bulk applications send more data than interactive ones, interactive conversations still send 5-10% of network bytes and 25-45% of network packets.

We think it important to realize that interactive applications are responsible for 25-45% of all Internet packets. Simulations that model internetwork traffic as mostly large bulk transfers may overestimate the benefit of mechanisms proposed to improve bulk transfer performance. Most existing studies evaluate the robustness of designs and algorithms under worst case loads, but fail to contrast their performance to that of equally robust designs or algorithms when running under average loads.

[4]The applications which appear in boldface are the ones we concentrate our study on.



(a) Conversation duration (b) Packets transferred

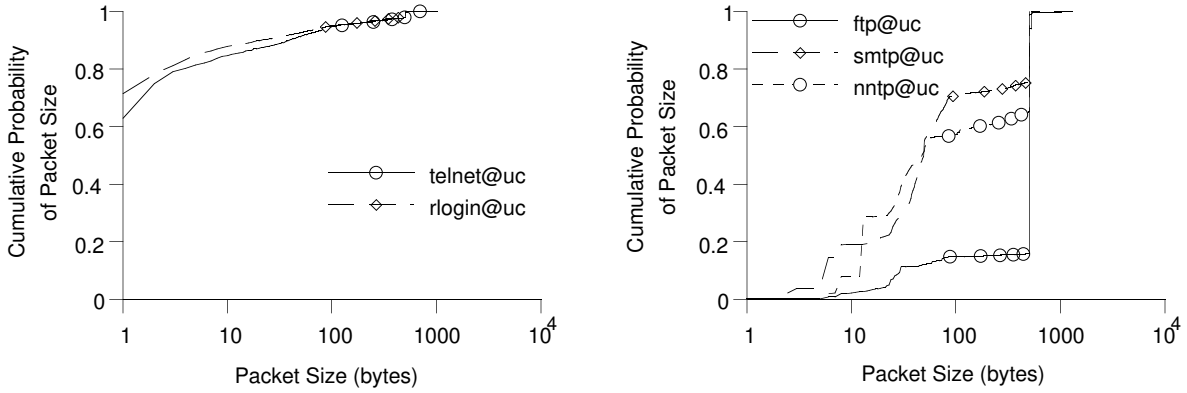
Fig. 2: Duration and packets transferred per conversation for interactive applications.

3.2. Bulk Data Transfer

Many simulation studies commonly overestimate the amount of data sent by bulk data transfer applications such as FTP. Transfer sizes usually range from 80K to 2M bytes, or simply continue to the end of the simulation run [7, 23, 24, 26, 27]. Figure 1a shows that about 60-90% of bulk transfer conversations transfer less than 10K bytes. Incidentally, our data shows that conversations that transfer more than one megabyte of data are responsible for 40-50% of all bytes transferred. However, as explained in section 3.4, bulk transfer conversations usually transfer more than one item per conversation—an *item* is a news article for NNTP or a file for FTP, with items larger than one megabyte making up only 15-20% of all bytes transferred. This observation correlates with the observation made in Reference 29 that most files are small.

If this is true of Internet source traffic in general, then it should be taken into account in future internetwork simulations. To the extent that simulated algorithms employ feedback mechanisms (such as congestion or flow control) [8], it is important to know that in most sessions data transfer will complete before any such feedback is received. Also, bulk transfer applications typically wait at least one network round-trip time between exchanging items. Therefore, as currently implemented, bulk applications, like NNTP, that appear to make exchanges of large amount of data actually make hundreds of exchanges of small data pieces separated by at least one network round trip time. We believe these observations are important

and long lasting because.for delay insensitive applications such as NNTP, this application design may be desirable in that it regulates the rate at which the applications send data. The emergence of voluminous real-time traffic will not make existing delay-insensitive traffic disappear.



(a) Interactive applications (b) Bulk transfer

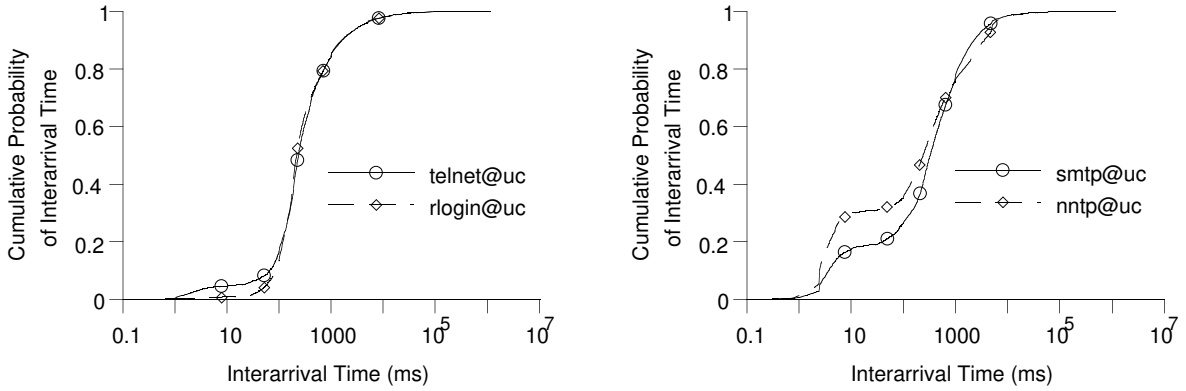
Fig. 3: Distribution of packet size by application. Packet sizes reflect only user data without protocol headers.

3.3. Interactive Applications

Network flow control and the Maximum Transferrable Unit (MTU)[5] determine, to a great extent, the measured statistics of bulk internetwork traffic. In contrast, Figure 1b and 2a show that about 90% of TELNET and RLOGIN conversations send less than 10K bytes, over a duration of 1.5 to 50 minutes. Figure 3a shows that about 90% of TELNET and RLOGIN packets carry less than 10 bytes of user data, which is much smaller than the MTU. Thus interactive applications are more or less unaffected by flow control and MTU size.

If interactive applications are not affected by network flow control and MTU, then the observed characteristics reflect the true nature of such applications. However, we should not assume that interactive traffic carries less data—Figure 1b shows that 80% of all interactive conversations send as much data as the average bulk transfer conversation—rather, it means that bulk transfer applications send a smaller amount of data than is often assumed.

[5]For historical reasons, wide-area TCP connections still use an MTU of 512 data bytes despite the fact that the NSFNET backbone supports 1500-byte packet.



(a) Interactive applications (b) Bulk transfer

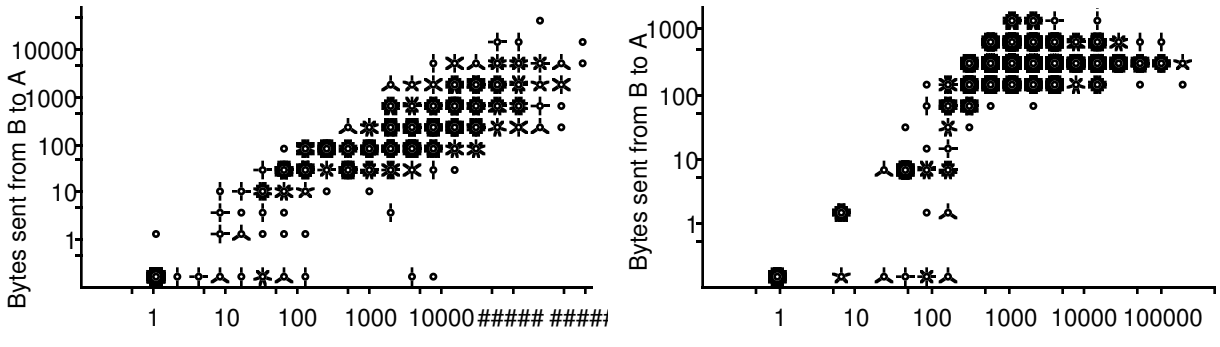
Fig. 4: Distribution of packet interarrival by application.

In most traffic models used in existing simulations or testbed studies, conversations are assumed to last anywhere from 500 seconds, 600 seconds, to "keep on forever" [7, 23, 24, 27, 29]. Figure 2a shows that the duration of interactive conversations is highly variable. This fact, along with the small number of packets per conversation (see Figure 2b), might influence steady-state feedback assumptions, as well as per-packet processing time with respect to gateway algorithms.

Finally, our data shows that while interarrival times for bulk data transfers exhibit the packet-train phenomenon, interarrival times for interactive applications should be modeled by a uniform plus exponential random time (see Figure 4a). Section 4 describes this phenomenon in more detail.

3.4. Directionality of Traffic Flow

Most simulations of gateway queueing [7, 23, 26] have assumed unidirectional data flow. Figure 5 shows that a large percentage of traffic, both interactive and bulk, is bidirectional. In other words, simulations should generate traffic in both directions.



(a) Telnet (b) SMTP

Fig. 5: Bidirectionality of traffic flow.

Figure 3b and 4b together affirm that many bulk transfer applications contain a request-response phase, which causes a synchronization point where no data is flowing in either direction. In turn, this synchronization point causes classic packet train behavior: a handshake followed by a big burst. For example, NNTP sends a query, waits for a response, and then does a bulk transfer, while FTP sets up a separate TCP connection for each item transferred. Since we tracked conversations in a unidirectional fashion, we found that 60% of all FTP and NNTP

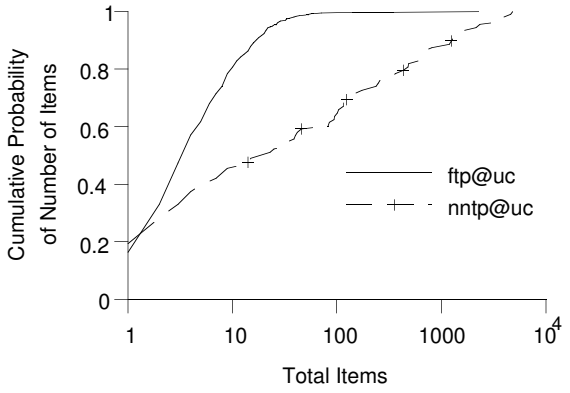


Fig. 6: Number of items per conversation.

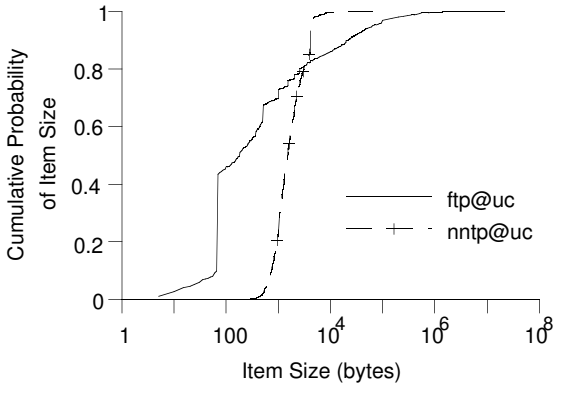


Fig. 7: Distribution of item sizes.

conversations do not send any items. However, when paired with their corresponding conversation going in the other direction, we found that bytes flow bidirectionally due to protocol handshaking—in the case of NNTP—or user control—in the case of FTP. Ignoring conversations that send zero items, Figure 6 shows distribution of the number of items sent per conversation for FTP and NNTP, and Figure 7 shows the distribution of the item sizes. This

request-response behavior may influence congestion and transport mechanisms and should be included in simulation studies of these mechanisms.

Small packets, short conversations, and bidirectional flow all contribute to the traffic dynamics of the internetwork. These characteristics of current internetwork traffic could affect traffic segregation and oscillation studies [23, 24, 27].

3.5. Wide-Area Network Locality

Mogul reports strong locality of reference between pairs of hosts on a local area network [31]. This locality of reference means that certain hosts communicate more with one another than with other hosts. Does such locality of reference exist between host pairs or network pairs in wide-area internetworks? Figure 8 shows that it indeed occurs. For example, half of UCB TELNET conversations are directed to just 10 stub-networks.

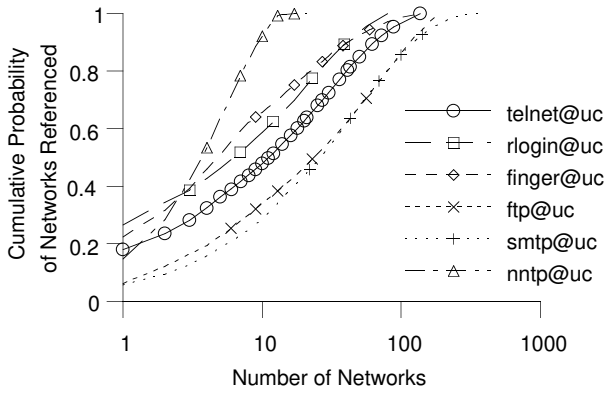
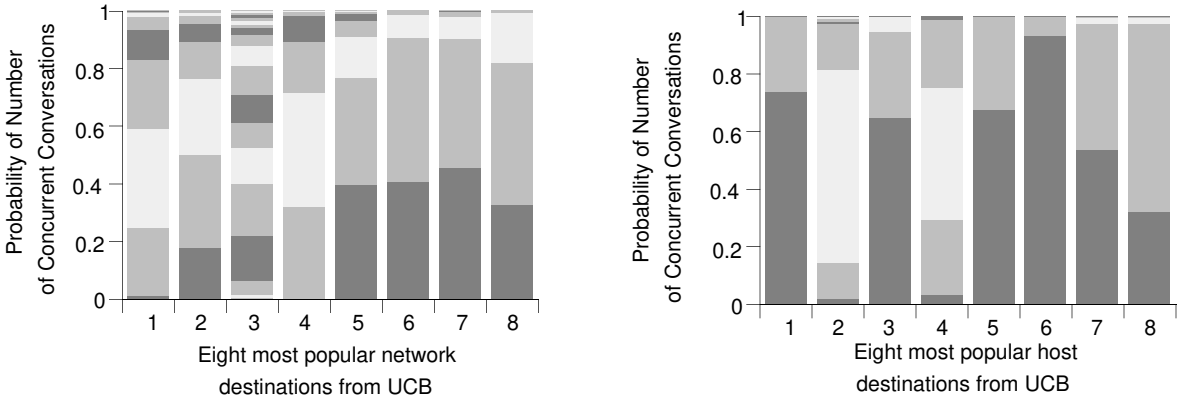


Fig. 8: Number of networks referenced by UCB.

Given network-pair locality on wide-area networks, we want to know how many concurrent conversations run between popular network-pairs. Figure 9a shows the concurrent conversations to UCB's eight most popular destination networks. In Figure 9, each band represents a number of concurrently running conversations. The band at the bottom of a bar represents the probability of finding zero on-going conversations. The next band up represents the probability of finding one on-going conversation. The third represents the probability of finding two simultaneous conversations, and so on. The third bar in Figure 9a shows that it is very probable to find more

than two concurrently running conversations between the two networks the bar represents. However, this particular bar represents the traffic between UCB and Lawrence Berkeley Laboratory which are located several hundred yards from each other.

Given that we frequently find concurrent conversations between popular network pairs, how often do we find concurrent conversations between host pairs on wide area networks? Figure 9b shows that it is unlikely with the present Internet traffic, but this may change in the future. The second host-pair in Figure 9b frequently exhibits two or three concurrent conversations. This host pair connects an UCB host to an Andrew host at CMU; we suspect that we captured traces of an experiment with the Andrew File System. From this measurement of current traffic, we can say that there are not many concurrent conversations between host pairs over wide-area network.



(a) Network pair (b) Host pair

Fig. 9: Number of concurrent conversations for the eight most popular network and host pairs coming out of UCB (see section 3.5).

4. An Artificial Workload Model

Our internetwork traffic model has two parts: a stub-network-dependent part and a stub-network-independent part. The first part models conversation arrivals for each type of network application. It determines when a new conversation arrives and what its destination is. We found these characteristics varied widely among stub networks, hence they are stub-dependent. The second part models packet arrivals within a conversation. It determines when a packet

arrives and what size it has. We found these characteristics to be constant across all conversations for the same application, hence they are stub-independent.

4.1 Stub-Dependent Models

The stub-network-dependent part of our model consists of a of stochastic process for each application responsible for outgoing traffic from a stub-network. These stochastic processes select the starting time and destination network of new conversations. We call the collection of stochastic processes from all stub networks the internetwork *traffic matrix*. Specifying the traffic matrix is hard because certain applications reference more stub-networks than do others (see Figure 8). For example, we see that half of UCB TELNET conversations are directed to just 11 stub-networks, with the other half referencing over 100 stub-networks. NNTP references just 11 stub-networks for the whole trace. Half of SMTP conversations reference over 50 stub-networks, and the other half reference 300 other stub-networks. Overall, half of UCB's conversations are directed to just 17 stub-networks. The traffic matrix must capture that some stub-networks send more traffic than others, and that some stub-networks use one application more heavily than another.

Analysis of conversation arrival times from all three traces indicates that we can approximate the application arrival processes as time-varying, independent Poisson processes with rate proportional to the stub-network's traffic breakdown (Table 1). We choose the application type of a stub-network's next conversation from the stub-network's traffic breakdown. Independence is only an assumption, because conversations depend on one another. For example, one is more likely to send mail to a stub-network shortly after fingering it than if one had never referenced it before. However this effect is not particularly pronounced in the data. We found that the types of successive conversations are independent, although we did not investigate correlations on the sequence of conversation types between a specific network pair or host pair. Hence, we model arrivals of new conversations as time-varying Poisson processes with stub-network and time-of-day dependent rates. For example, Figure 10 plots measured arrival rates of UCB conversations

for several applications. By making the rate depend on time of day, it is possible to model stub-network-specific configurations. For example, at UCB, VMNET runs just four times per day at specified times, while at USC VMNET runs on demand.

A stub's conversation arrival processes must also specify destination stub-networks for each conversation. More study is needed to characterize conversation destination networks However, present technology limits existing network simulators to well less than fifty stub-networks, and typical studies simulate four to sixteen nodes. Larger size of future simulations will make the issue of representative traffic matrix more critical.

4.2 Stub-Independent Models

The stub-network-independent part of our artificial workload model consists of *source* models for an individual conversation of each popular application type. We constructed source models for five of the six applications responsible for more than 96% of wide area network bytes transmitted. The other twenty-nine applications identifiable in the traces account for the remaining 4%. Currently we model FTP, SMTP, NNTP, TELNET, and RLOGIN. The source model is implemented as a library of functions, described in Section 4.3, that can be called from a network simulator.

Modeling a source consists of two steps. First, we select the application-specific characteristics of the new conversation. If the application is bulk we select the amount of data exchanged. If it is interactive, we select the duration of the conversation. Second, we determine the sequence of packets that the conversation will send.

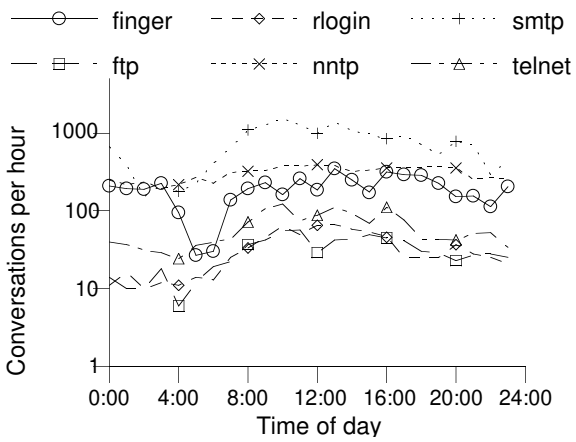


Fig. 10: Conversation arrival rate.

The first step depends on whether the conversation is bulk or interactive. If it is bulk, we choose the number of bytes transmitted in each direction from the joint distribution of bidirectional bytes transmitted. We illustrate one such distribution for SMTP in Figure 5b. This figure plots the larger side of a conversation on the x-axis and the smaller side on the y-axis. Bigger, darker marks indicate higher likelihood. If the conversation is interactive, we choose its duration from the distribution of duration. We illustrate two such distributions for TELNET and RLOGIN in Figure 2a. We show a distribution of duration for bulk protocols in Figure 2b, but do not employ it in the models because the duration of a bulk transfer depends on network bandwidth and flow control, rather than the traffic sources.

For bulk transfer such as FTP and NNTP, we also model the number of items transferred, such as the number of news articles exchanged during an NNTP conversation. Given the distribution of the number of items transferred (see Figure 6) and the distribution of the number of bytes in an item (see Figure 7), we can model the synchronous interactive phase inherent in most bulk applications. During this phase, file names, commands, and article numbers are exchanged. These interactive phases act as synchronization points; at their start, no outstanding packets exist between end points. Hence, there is at least one round trip time between bulk exchanges.

The rule for specifying packet arrival times and sizes depends on the application. For bulk applications, packet sizes and interarrival times depend on physical characteristics of the network, the bidirectional distribution of bytes transferred, and the distribution of items

transferred. While their packet interarrival times depend on the network, their packet sizes depend on the application. During bulk transfers, packet sizes are a network MTU followed, if necessary, by a final smaller fragment. During control exchanges, packet sizes are smaller, corresponding to file names and commands; it is necessary to draw their packet sizes from the measured distributions (see Figure 3b).

In contrast to bulk traffic, packet interarrival times of interactive traffic depend on the user. Users' keystrokes generate “byte-sized” packets with a uniform plus exponential interarrival time distribution. The destination process sends a response for every packet that it receives; occasionally it returns a large response (see Figure 3a). A close inspection of the interarrival times of TELNET and RLOGIN packets presented in Figure 4a reveals that 10% of the time, interarrival times are less than 100 milliseconds. These short interarrival times occur for two reasons. First, when the destination sends a response greater than a network MTU, its packets arrive in rapid succession. These back-to-back MTUs account for roughly a quarter of the interarrival times less than 100 milliseconds. Second, network queueing and operating system unresponsiveness can deliver single key strokes to the destination in rapid succession. Back-to-back single data-byte packets constitute roughly three quarters of these short interarrival times.

Application Type	Routine Name
Interactive	float telnet_duration() float telnet_interarrival() int telnet_pktsize()
Bulk Transfer	int ftp_nitems() int nntp_nitems() int ftp_itemsize() int nntp_itemsize() int smtp_itemsize() int ftp_ctlsize()

Table 4: Distribution functions included in the traffic library.

4.3 Generating Conversation Characteristics

The application-specific source models are implemented as a library to be linked with a network simulator. Table 4 presents a summary of our library of routines, *tcplib*[6]. Figures 11 and 12 illustrate how these routines can be used to generate simulated TELNET and FTP traffic. Because curve fitting loses information, and since it makes no difference to the simulator whether there exists an analytical representation of the distributions, *tcplib* generates random numbers by the inverse transform method [31]. Each routine inverts a piecewise linear representation of the measured distribution. Below, we briefly describe the inverse transform method.

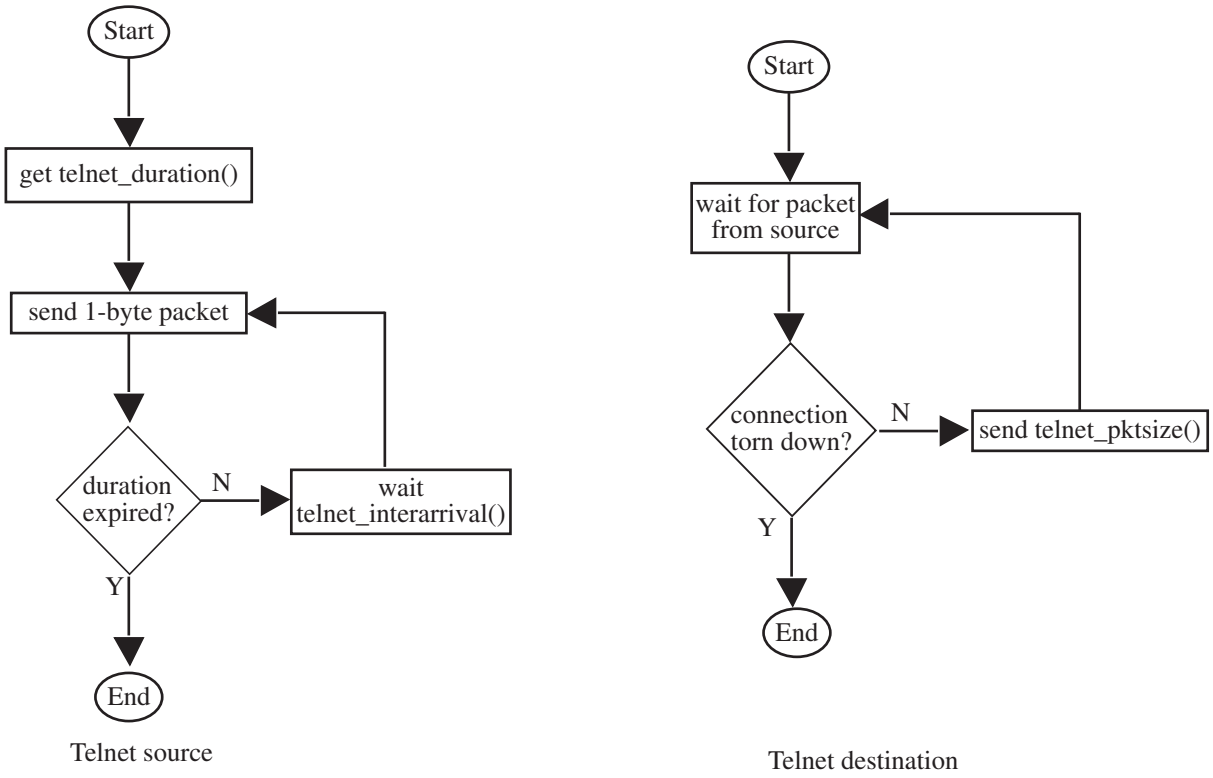


Fig. 11: Flow chart for controlling TELNET and RLOGIN conversations.

[6]tcplib is available from anonymous ftp from jerico.usc.edu in directory pub/jamin/tcplib

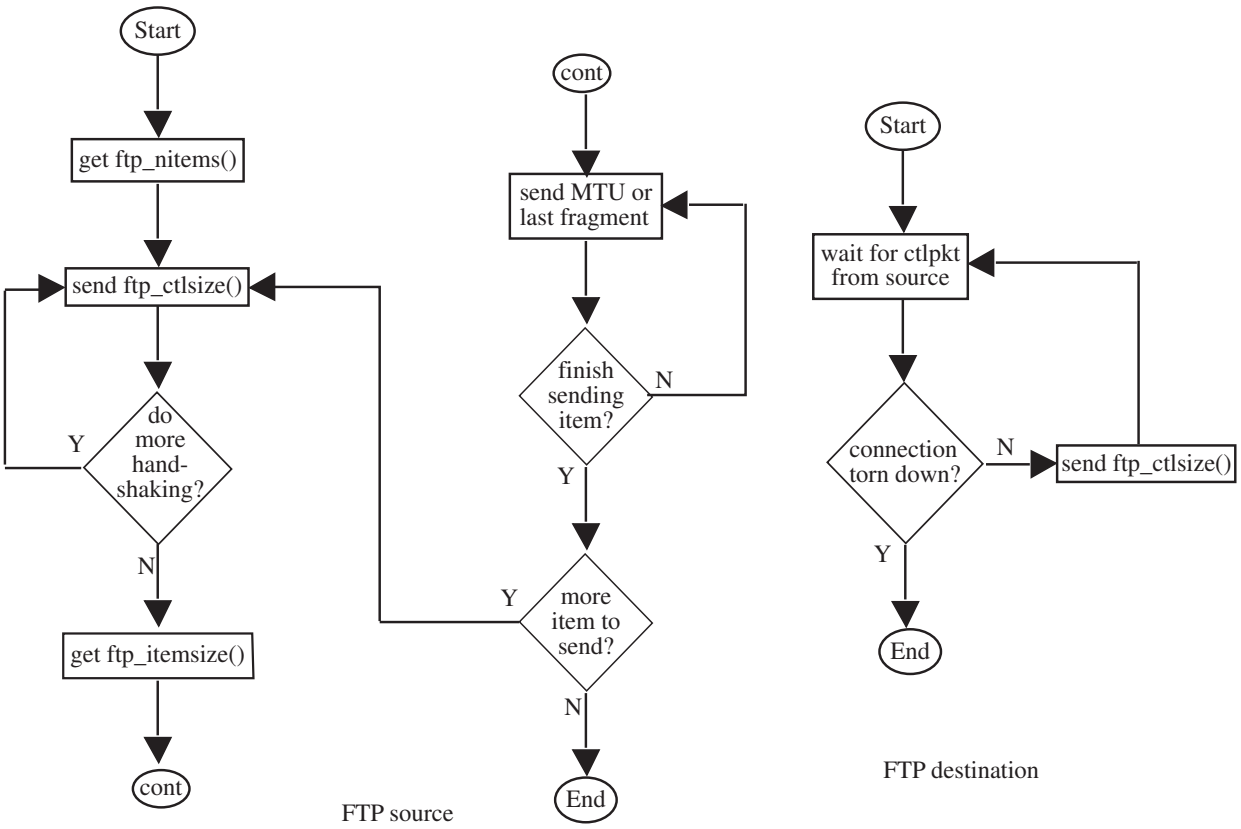


Fig.12: Flow chart for controlling FTP conversations.

The inverse transform method maps uniformly distributed 0–1 random variates through the "y-axis" of the cumulative probability distribution onto the "x-axis." With distributions fitted to analytical expressions, the inverse transform method involves inverting an equation. Consider, for example, generating an exponential random variate. If μ is a 0–1 uniform random variate and l is the parameter of the exponential, then $x=-log(1-\mu)/l$ is an exponentially distributed random variate. In our case, we built a histogram of the individual data points, and then summed the histogram bin heights to create our distribution function. Hence, our distributions are represented by arrays rather than expressions. An array index i corresponds to a particular value of the distribution. The contents of the array element at index i, $x[i]$, is the value of the cumulative distribution. Hence, to generate a random variate, we first generate a 0–1 uniform random variate μ. We then perform a binary search on the array elements until we find the element $x[k]$ into which μ falls. Finally, we linearly interpolate between $x(k)$ and $x(k+1)$ to determine our random variate x.

Rather than forming a histogram, another approach is to keep every single data point and sort the set. Then sample every 100th or 1000th element and place them in subsequent array locations that correspond to 0.01-quantile and 0.001-quantile increments. The array can then be directly indexed by μ, eliminating the search for the bin in which μ fell. We didn't adopt this approach because it takes more memory to implement than the scheme described in the previous paragraph.

5. Applying the Traffic Model

Since we are not suggesting that algorithm robustness testing should use our workload model in place of worst-case scenarios, just what good is a tool for generating realistic internetwork traffic? This section describes one problem that needs a realistic internetwork traffic model.

The problem of multiplexing application datagram traffic over wide-area virtual circuits reappears with the advent of high-speed Asynchronous Transfer Mode (ATM) networks. Assuming the existence of a reservation scheme for handling the requirements of multimedia traffic [32], we still have to accommodate the dynamics and requirements of traditional datagram traffic. When a datagram arrives at an ATM gateway, it needs to be routed onto an appropriate virtual circuit. If such a circuit doesn't exist, data transmission must wait until one is established. On the other hand, idle virtual circuits consume resources inside the ATM network. We want to find ways to multiplex TCP conversations over ATM virtual circuits that provide adequate performance while making efficient use of network resources.

We need to trade the performance costs of establishing new virtual circuits with the resource utilization advantages of closing idle circuits. Evaluating this tradeoff requires a good, average case internetwork traffic source model. With such a model we could decide how to map a set of TCP conversations onto a possibly smaller set of ATM virtual circuits, choose the queueing discipline for multiplexing datagrams onto these virtual circuits, and arrive at a timeout algorithm for reclaiming idle virtual circuits.

No previous model of wide-area traffic is appropriate for this study. To evaluate the performance of different mapping schemes, we need a realistic internetwork traffic matrix. Without accurate knowledge of application mix and behavior, we cannot predict the effect of multiplexing several different TCP conversations through a single ATM virtual circuit. To evaluate timeout schemes, we need the distribution of conversation durations and conversation interarrival times.

There are other cases where a detailed characterization of applications as presented in this paper will be required. Even for studies that aim to prove only the robustness of new designs or algorithms, using our model can show how new designs or algorithms perform on the common case.

6. Implications and Conclusions

Analysis of traffic traces collected from three different stub-networks show that the sequence of packets that application programs generate can be characterized by certain application-specific characteristics which are independent of the stub-network. We constructed an artificial workload model of a TCP/IP internetwork composed of a stub-network independent set of application source models and a stub-network dependent set of application arrival processes.

We also identified application characteristics that contradict the following commonly held beliefs regarding current wide-area traffic:

- Bulk sources transfer large amounts of data per conversation.
- Bulk sources send large packets in only one direction.
- Interactive sources send small packets in one direction, and receive echoes of comparable size in the opposite direction.
- Internetwork traffic can be modeled by either a Poisson interarrival process or a packet-train model alone.

Addressing these myths in order, we have shown that:

- Eighty percent of the time, classic bulk transfer application such as FTP transfer less than 10 kilobytes per conversation. Other applications commonly categorized as bulk traffic sources, such as SMTP, transfer even smaller amounts of data (see Figure 1a).
- Traffic generated by FTP, SMTP, NNTP, and VMNET is strongly bidirectional. Furthermore, SMTP and NNTP send as many small packets as large packets (see Figures 5b and 3b).
- Interactive applications routinely generate 10 times more data in one direction than the other, using packet sizes ranging from 1 byte to 512 bytes (see Figures 5a and 3a).
- Interactive packet interarrivals closely match a uniform plus exponential distribution (see Figure 4a).

We are continuing work on tools to create wide-area network traffic based upon our characterizations. We will also study various algorithms' responses to average case data, especially flow control and congestion control algorithms whose robustness, but not average case behavior, was evaluated in previous studies. There is more work to be done in understanding traffic reference patterns, and a better understanding of these should impact the design of future networks.

Acknowledgements

We would like to thank D. Ferrari, S. Floyd, S. Keshav, S. Morgan, J. Mogul, and M. Sullivan for their useful suggestions. M. Cooper and P. Lapsley imparted to us the inner workings of NNTP. For the data collection at Berkeley, B. Prabhakar was instrumental in collecting the traces, and C. Frost provided access to the gateway Ethernet. At USC, J. Pepin, M. Brown, and R. Kaplan provided access to the campus network, and J. Conti assisted in collecting the traces. At Bellcore, D. V. Wilson gathered the traces with his Ethernet monitor and made them available to us.

References

[1] Postel, J.B., "DoD Standard Transmission Control Protocol," RFC 761, January '80.

[2] Mark Lottor, Internet Domain System, in ACM Forum, CACM, Vol. 34, No. 11, November, 1991.

[3] Merit/NSFNET Information Services, *Link Letter*, Vol. 3, No. 6, January/February, 1991.

[4] NSFnet Backbone Statistics, Merit/NSFnet Information Services, June 1991 (obtained by anonymous FTP from nis.nsf.net).

[5] Postel, J.B., Reynolds, J.K., "File Transfer Protocol," RFC 959, October '85.

[6] Postel, J.B., Reynolds, J.K., "Telnet Protocol Specification," RFC 854, May '83.

[7] Demers, A., Keshav, S., and Shenker, S., "Analysis and Simulation of a Fair Queueing Algorithm," *ACM SIGCOMM '89*, pp. 2-12.

[8] Ramakrishnan, K.K. and Jain, R., "A Binary Feedback Scheme for Congestion Avoidance in Computer Networks," *ACM TOCS*, 8:2, May 1990, pp. 158-181.

[9] Jain, R. and Routhier, S., "Packet Trains—Measurements and a New Model for Computer Network Traffic," *IEEE JSAC*, Sep. '86.

[10] Gusella, R., "A Measurement Study of Diskless Workstation Traffic on an Ethernet," *IEEE Transactions on Communications*, Sep. 1990.

[11] Leland, W.E. and Wilson, D.V., "High Time-Resolution Measurement and Analysis of LAN Traffic: Implications for LAN Interconnection," *Proc. of INFOCOM '91*.

[12] Cáceres, R.. *Measurements of Wide-Area Internet Traffic*. UCB/CSD 89/550, University of California, Berkeley, Dec. '89.

[13] Heimlich, H., "Traffic Characterization of the NSFNET National Backbone," *USENIX Conf. Proc.*, Winter '89.

[14] Feldmeier, D., "Improving Gateway Performance with a Routing-Table Cache," *Proc. IEEE INFOCOM '88*, March '88.

[15] Jain, R., *Characteristics of Destination Address Locality in Computer Networks: A Comparison of Caching Schemes*, DEC-TR-592, Feb. '89.

[16] Crowcroft, J., *Traffic Analysis of Some UK-US Academic Network Data,* University College London TR RN/90/UK.

[17] Paxson, V., *Measurements and Models of Wide-Area TCP Conversations,* Lawrence Berkeley Lab TR LBL-30840, 1991.

[18] Danzig, P.B. and Melvin, S., "High Resolution Timing with Low Resolution Clocks and a Microsecond Timer for Sun Workstations," *ACM OS Review*, 24:1, Jan '90, pp. 23-26.

[19] Braden, B. and DeSchon, A.L., *NNStat: Internet Statistics Collection Package—Introduction and User Guide*, USC-ISI, Dec. '89.

[20] Postel, J.B., "Simple Mail Transfer Protocol," RFC 821, August '82.

[21] Kantor, B., Lapsley, P., "Network News Transfer Protocol," RFC 977, February '86.

[22] Merit/NSFNET Information Services, "New Statistics Available in Database," *Link Letter,* December 1991.

[23] Floyd, S. and Jacobson, V., "Traffic Phase Effects in Packet-Switched Gateways," *Comp. Communication Review*, April 1991, pp. 26-42.

[24] Wilder, R., Ramakrishnan, K.K., and Mankin, A., "Dynamics of Congestion Control and Avoidance of Two-Way Traffic in an OSI Testbed," *Comp. Comm. Review*, 21:2, April 1991, pp. 43-58.

[25] Williamson, C.L. and Cheriton, D.R., "Load Curves: Support for Rate-Based Congestion Control in High-Speed Datagram Networks," *Proc. ACM SIGCOMM '91*.

[26] Zhang, L., "Virtual Clock: A New Traffic Control Algorithm for Packet Switching Networks," *Proc. of SIGCOMM '90*, pp. 19-29.

[27] Zhang, L., Shenker, S., and Clark, D.D., "Observations on the Dynamics of a Congestion Control Algorithm: The Effects of Two-Way Traffic," *Proc. ACM SIGCOMM '91*.

[28] Ousterhout, J.K. et al., "A Trace-Driven Analysis of the UNIX 4.2 BSD File System," *Proc. of the 10th ACM SOSP*, Dec 1-4, 1985, pp. 15-24.

[29] Mankin, A., "Random Drop Congestion Control," *Proc. ACM SIGCOMM '90*, pp. 1-7.

[30 Mogul, J., "Network Locality at the Scale of Processes," *Proc. ACM SIGCOMM '91*.

[31] Jain, R., *The Art of Computer Systems Performance Analysis*, Wiley, 1991.

[32] Ferrari, D. and Verma, D.C., "A Scheme for Real-Time Channel Establishment in Wide-Area Networks," *IEEE JSAC*, 8:3, April '90.

Appendix 1

Comparative Data from the Three Sites

In the following figures, curves labelled *uc* represent UCB data, ones labelled *bc* represent Bellcore data, and ones labelled *sc* represent USC data.

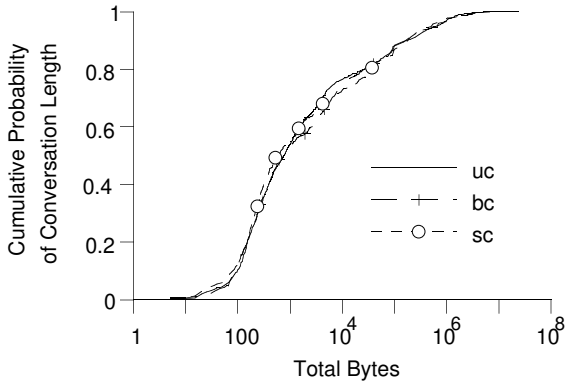


Fig. A: Total bytes transferred per unidirectional FTP conversation.

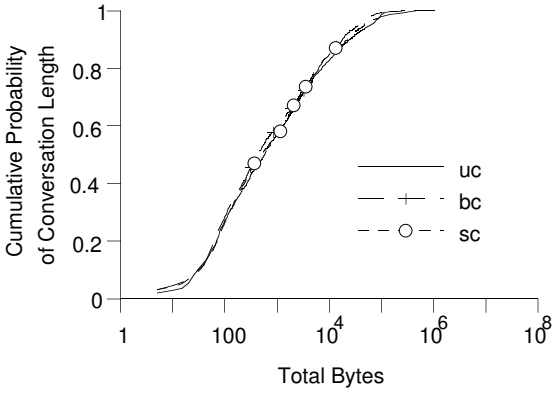


Fig. B: Total bytes transferred per unidirectional TELNET conversation.

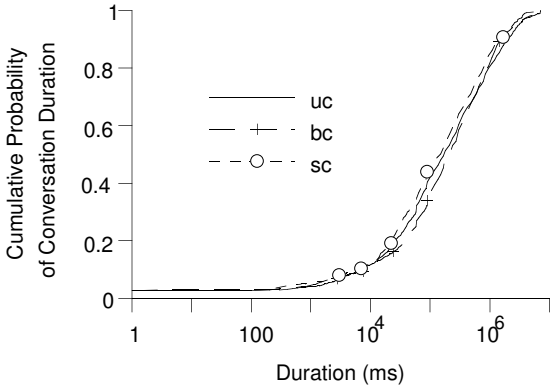


Fig. C: Duration of TELNET conversations.

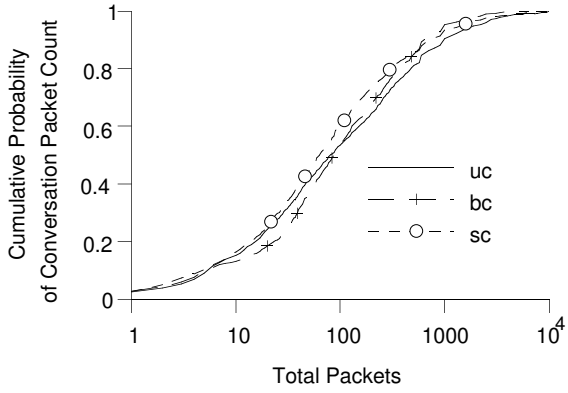


Fig. D: Packets transferred per TELNET conversation.

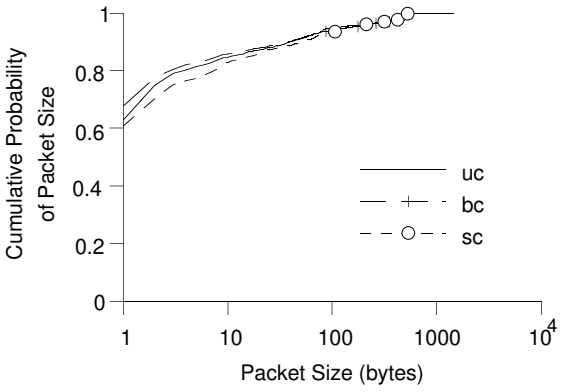


Fig. E: Distribution of TELNET packet sizes.

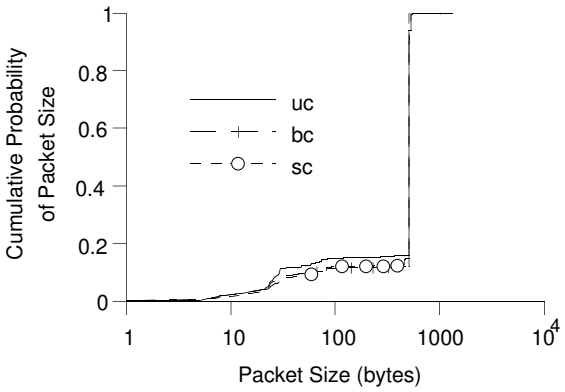


Fig. F: Distribution of FTP packet sizes.

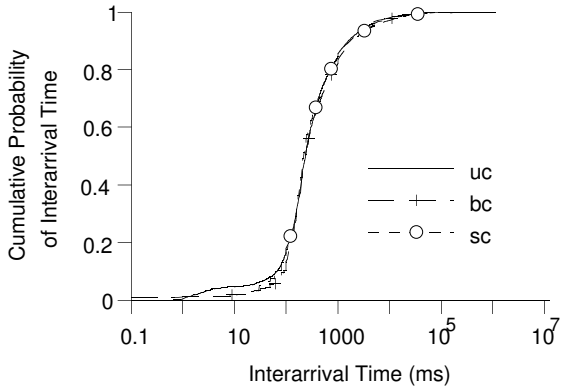


Fig. G: Distribution of TELNET packet interarrivals.

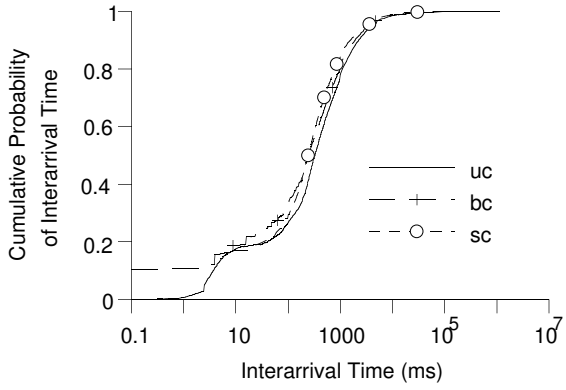


Fig. H: Distribution of FTP packet interarrivals.