

# Stochastic Models for Generating Synthetic HTTP Source Traffic

Jin Cao, William S. Cleveland, Yuan Gao, Kevin Jeffay, F. Donelson Smith, Michele Weigle

**Abstract**—New source-level models for aggregated HTTP traffic and a design for their integration with the TCP transport layer are built and validated using two large-scale collections of TCP/IP packet header traces. An implementation of the models and the design in the *ns* network simulator can be used to generate web traffic in network simulations.

**Index Terms**—Simulations, Statistics, Network measurements

## I. INTRODUCTION

Networking research has long relied on simulation as the primary vehicle for demonstrating the effectiveness of proposed algorithms and mechanisms. Typically one constructs either a network testbed and conducts experiments with actual network hardware and software, or one simulates network hardware and software in software and conducts experiments via simulation of the network. In either case, experimentation proceeds by simulating the use of the real or simulated network by a population of users with applications such as file transfer, web browsing, or peer-to-peer file sharing. Source traffic generators are used to inject synthetic traffic into the network according to a model of how the corresponding applications or users behave.

For nearly the last 10 years (and for the foreseeable future) traffic generators for synthetic web traffic have been an essential component of virtually every simulation of the Internet. While newer applications such as peer-to-peer file sharing are consuming a significant share of network resources, by any measured quantity — bytes, packets, flows — the web remains the dominant application on most wide-area links. The reason is that the web has evolved from a simple hypertext document delivery system to a sophisticated client-server system for delivering a vast array of static and dynamic media. The HTTP protocol is now routinely used to deliver content once carried on more specialized application-level protocols. For example, the web is now often the *de facto* end-user interface for remote data processing systems, commercial transactions, and sending and receiving email, news, and instant messages. A recent (April 2003) measurement study by Sprint [1] on 19 of the OC-48 links in their network found that on 16 of them web traffic was the largest application class and ranged from 31% to 59% of the total bytes transmitted. On the other 3 links,

web traffic was the second largest (behind peer-to-peer file sharing) with 16% to 31% of the total bytes.

Thus when performing network experiments and simulations involving web usage, it is essential that one consider both the effects of web traffic on the mechanism/protocol under study and the effects it has on the performance (*e.g.*, response times) of web applications. Good characterizations of how web traffic “looks” in the network are, therefore, essential for networking experiments.

This paper presents a new model of HTTP 1.0 and 1.1 traffic as it appears in aggregate on backbone or high-speed access links. The model derives from a large-scale empirical study of web traffic on the two access links that connect Bell Labs and the University of North Carolina at Chapel Hill to the Internet. The model is novel in that it expresses web traffic as a collection of independent TCP connections, each characterized by values of source variables: arrival time of the connection, round-trip time for the client, round-trip time for the server, number of request/response exchanges, time gaps between exchanges, sizes of individual requests, sizes of individual responses, and server delays. As explained below, this approach differs from the dominant present-day approach of constructing “page-based” models of web traffic. We argue that our “connection-based” modeling method is more appropriate for network traffic simulation, as opposed to server workload simulation, because it both captures relationships and dependencies not present in existing page-based models and it is an approach that is more likely to scale to modeling the traffic generated by other application classes such as peer-to-peer file sharing traffic.

Our model of aggregate HTTP traffic has been implemented in the *ns* network simulator and is available for use in generating realistic synthetic web traffic in network simulations. The model and its implementation in *ns* have been validated through both empirical and analytical analyses that are presented here.

The remainder of the paper is organized as follows. Section II reviews the state-of-the-art in synthetic web traffic generation and discusses the important issues in developing application-level models of network traffic. Section III presents the architecture: what is modeled and how it interacts with the TCP transport layer; the description is a special case of our general architecture, but one that must address all of source traffic modeling issues. Section IV describes the source-level variables that are necessary for a connection-based generator of synthetic web traffic, how these variables are measured, the process of building and validating models for these source variables. Section V describes the validation of the source models and of the architecture that encompasses them. Section VI summarizes the previous sections, describes the

J. Cao (cao@bell-labs.com) and Y. Gao (yuangao@research.bell-labs.com) are in the Statistics and Data Mining Research Department of Bell Labs, Murray Hill, NJ. W. Cleveland (wsc@stat.purdue.edu) was in the same department at Bell Labs when the work of this article was completed and is now in the Department of Statistics and the Department of Computer Science, Purdue University, W. Lafayette, IN. K. Jeffay (jeffay@cs.unc.edu), F. D. Smith (smithfd@cs.unc.edu), and M. Weigle (mcweigle@cs.unc.edu) are in the Department of Computer Science, University of North Carolina, Chapel Hill, NC. This research was supported, in part, by DARPA under federal contract F30602-02-C-0093.

general architecture of our source traffic modeling, provides information on how to obtain the *ns* code for the synthetic source traffic generator, and describes how a user specifies the synthetic traffic that is generated; a reader interested in an overview of the work can proceed directly to this section. Appendix A provides details of the stochastic models for the source variables. Appendix B provides details of the implementation of our synthetic traffic generator in the *ns* network simulator.

## II. BACKGROUND AND MOTIVATION

### A. Source-Level Generation of Synthetic Traffic

Our vision of network simulation follows the philosophy of using source-level descriptions of network traffic advocated by Floyd and Paxson [2]. In this paradigm one simulates *the use* of the network by either an application (or set of applications) or a collection of users. Traffic generators inject synthetic traffic into the network according to a model of how some application or class of users behave. This is in contrast to a network simulation using packet-level descriptions of traffic wherein one simply simulates the arrival process of packets at a particular network element according to some mathematical process. For applications using TCP, packet-level traffic generators cannot be used to model traffic because TCP's end-to-end congestion control (perhaps influenced by router-based mechanisms such as RED packet drops or ECN markings) shapes the low-level packet-by-packet arrivals in a feedback control loop which is not modeled with an "open loop" packet-level generator. For this reason Floyd and Paxson stress the importance of using source-level traffic generators layered over real or simulated TCP implementations. Therefore, a critical problem in network simulations is generating application-dependent but network-independent synthetic traffic that corresponds to a valid, contemporary model of application or user behavior.

### B. Empirically-Derived Web Models

Web-traffic generators in use today are usually based on data from the two pioneering measurement projects that focused on capturing web-browsing behaviors: the Mah [3], and Crovella, *et al.*, [4], [5], [6], [7] studies. Traffic generators based on both of these sources have been built into *ns*, which has been used in a number of studies related to web-like traffic, *e.g.*, [8]. These models have also been used to generate web-like traffic in network testbeds [9], [10]. For both of these web measurement studies, the populations of users were quite distinctive and the sizes of the traces gathered were relatively small. Mah captured data reflecting a user population of graduate students in the Computer Science Department at UC Berkeley. His results were based on analysis of approximately 1.7 million TCP segments carrying HTTP protocols. The more extensive measurement programs by Crovella and colleagues reflected a user population consisting primarily of undergraduate students in the Computer Science Department at Boston University and in aggregate represented around 1 million references to web objects. Both sets of data are now relatively old. The Mah data were collected in 1995 and the Crovella, *et al.*, data in 1995 and 1998.

It is especially important to note that these studies were conducted before significant deployment of HTTP version 1.1 protocol implementations that introduced the concepts of persistent connections and pipelining [11]. Persistent connections are provided to enable the reuse of a single TCP connection for multiple object references at the same IP address (typically embedded components of a web page). Pipelining allows the client to make a series of requests on a persistent connection without waiting for a response between each request (the server must, however, return responses in the same order as the requests are sent). Persistent connections are widely implemented in both web servers and browsers but pipelining is largely supported only in server implementations. A more contemporary measurement study of web traffic that produced models suitable for page-based traffic generation was reported by Smith, *et al.*, [12]. They found that approximately 40% of all data bytes transmitted by web servers were carried on persistent connections used for two or more requests.

### C. Page vs. Connection Models

These empirical studies influence (and are influenced by) the way traffic generators have been designed for web traffic. The studies of web traffic cited above have been used to design traffic generators that we characterize as "page based". Traffic generators designed around these models explicitly use web-page structure, the location of page components on servers, and the human actions of thinking and page selection to indirectly control the creation of new TCP connections and the dynamic request/response data transfers within a connection. For example, the web traffic generator used in [13] consisted of a program to emulate client-side user actions and a server-side program to respond to client generated requests.

For networking studies this "paged-based" design for modeling TCP connection arrivals and their internal data-transfer dynamics makes the traffic generation programs somewhat complicated. Further each random variable in such models is considered to be independent and is sampled from an independent distribution. Potentially important correlations between different variables are not explicitly modeled. We show in Section IV and Appendix A that there are significant dependencies in the collection of variables that describe the HTTP connection request-response process. To generate web traffic carried by network links, routers, and protocol stacks, we claim that it is better to model TCP connections in terms of connection establishment rates and the sizes and timing of exchanges of request and response data within the connections (including persistent connections). We call such a model "connection-based" in contrast to the page-based approach described above.

Perhaps the most important reason to prefer the approach of modeling TCP connection usage instead of application-specific details like page structure is that it scales better to large mixes of applications. The HTTP model presented here is a first step toward a more general framework of TCP connection modeling. If we can model the inter-arrival process of TCP connections and the size and timings of exchanges of abstract "data units" within the connection, we can model any arbitrary mix of TCP applications without having to explicitly know or represent any other application-specific information. Given

the large and ever-changing mix of application-level protocols used in the Internet today, this approach clearly scales better than an approach that attempts to include more application details such as page structure. The TCP connection model for the web presented here is a first step in this direction.

### III. A TCP CONNECTION-BASED ARCHITECTURE (SPECIAL CASE)

We now describe the architecture of our approach — what is modeled and how it interacts with the TCP transport layer. Our description in this section is a simple special case, but one that must address all of source traffic modeling issues; Section VI describes the general framework. We also describe a high-level view of using this model to generate web traffic in network experiments. As stated earlier, our approach models TCP connections when used to carry HTTP protocols rather than explicitly modeling web page structure or user browsing actions in detail. The design described here assumes that web traffic uses a mix of HTTP/1.0 and HTTP/1.1 protocols at the application level. More specifically, it assumes that some TCP connections will be used for only one HTTP request/response exchange (a non-persistent connection) and some TCP connections will be used for two or more HTTP request/response exchanges (a persistent connection).

Figure 1 illustrates our architecture for a simple case. There is a modeled network, in this case a single link. By a “modeled network” we mean a testbed of actual network hardware and software, or a software simulation of actual network hardware and software. At a very high level of our architecture, the web traffic in the network is modeled as flowing between a set of clients (browsers) whose traffic is aggregated at some link in the network, and a set of servers whose traffic is also aggregated at some link in the network. By using multiple client and server “clouds”, arbitrary configurations and loadings of the links in a testbed or simulation can be achieved. For our simple case in Figure 1, the aggregation link for the client cloud and for the server cloud is the single link of the network. But the simple case is a realistic one because it can be used to model a single access link connecting an enterprise or campus network to the Internet. All the web clients could be on the enterprise or campus network, and all the servers could be spread out across the Internet. As we will see, the data of our study take this form.

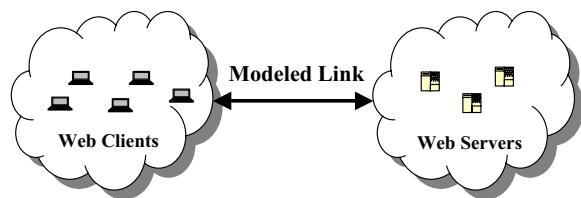


Fig. 1. Client cloud, server cloud, and a modeled network (a single link) carrying web traffic.

A fundamental parameter for the model is the rate at which new TCP connections are initiated by the cloud of web clients. Many other variables necessary to model TCP connection usage for HTTP protocols have statistical properties that depend on the connection rate. Given a value

for the rate parameter, the inter-arrival times between TCP connections in a cloud is determined by generation from the stochastic model for the inter-arrival process. Once a TCP connection is established between a client and server pair, the number of request/response transactions is generated as well as delays between these transactions if there is more than one. The number of bytes in the first client request sent to the server is stochastically generated and transmitted over the connection. After the server receives the request, it waits for a stochastically generated server-delay interval. A size for the response is generated and transmits that number of bytes back to the client. This continues until the number of requests is satisfied, when the TCP connection is then terminated. This constitutes the application-dependent aspects of our model of TCP connections used for HTTP protocols.

To provide more complete modeling support for testbeds and simulations, we also model certain aspects that are clearly network-dependent but may be useful for certain types of experiments. One important factor in TCP connection throughput is the round-trip time experienced by the connection in the cloud of clients or the cloud of servers due to propagation and queuing delay. For each connection, one time is generated for the client cloud and one for the server cloud; all packets of the connection are delayed by these amounts within each cloud. To model drops for packets entering or exiting each cloud, a cloud drop probability is generated for the connection, and each packet of the connection is dropped with that probability. Similarly, a cloud bottleneck link speed is generated for the connection and each packet of the connection is delayed according to the packet size and the link speed. In other words, we assume the hosts within a cloud are widely distributed and use simulated delay and loss processes to influence the packets along the path from an individual client or server to or from the aggregation link.

### IV. SOURCE VARIABLES AND THEIR MODELING

#### A. The Modeling Process: Building and Validation

The HTTP models were built and validated based on packet traces — time-stamps and TCP/IP protocol headers for packets in TCP connections used for HTTP protocols — from two Internet links. The measured links are well modeled by an architecture like that in Figure 1; that is, for the connections that we studied, clients are on one side of the link, servers are on the other side, and all packets in both directions use the link. The packet trace database is described in more detail later in this section.

We analyzed the data using the S-Net system for the analysis and visualization of packet traces [14], employing a large number of tools to build and validate the model. There were two stages of the analysis — source-level model building and packet-level validation.

In the source-level model building, the packet traces were used to construct measurements of the source variables that are modeled. The measurements allowed us to identify initial models for the variables, then to check the models, then to alter them to improve the fits, then to check the new models, and so forth in an iterative fashion. It was vital in this process to have the two links, with certain quite different measured characteristics, so we could determine which modeled variables

changed based on the characteristics. Appendix I describes the source-level models, but, in the interest of space, not the model building process. A detailed account of the model building is given elsewhere [15].

The packet-level validation consists of a set of simulation experiments using the *ns* implementation (described in Appendix II) to model the traffic carried on one of the links for which we have packet traces. First we estimate parameters of the HTTP source models and derive the network-dependent properties needed; this is done so that the simulation matches the characteristics of the client and server clouds aggregated at the measured link. Then we use the models to create and utilize TCP connections (using the TCP protocol model provided in *ns*) and produce synthetic packet traffic on the simulated link between clients and servers. We record both directions of traffic on the simulated link, just as we did for the live traces. We then compare synthetic and live values of a number of traffic variables. This validation process is described in Section V.

### B. Packet Traces

We collected packet traces on two links. The first is a 100 Mbps Ethernet link at Bell Labs that connects a network of 3000 hosts to the rest of the Internet [16]. For HTTP, all clients are on one side of the link and all servers are on the other side, so incoming packets on the link are from servers and outgoing are from clients. The time period of the traces used for analysis is from 1/1/00 through 2/16/00. The second link is a 1 Gbps Ethernet link connecting the Chapel Hill campus of the University of North Carolina to an OC48 fiber ring that carries UNC traffic from over 35,000 users to other local campuses and to the rest of the Internet [12]. There are HTTP clients on both sides of this link, but we used just the packets for connections from UNC clients to Internet servers. The UNC database consists of 42 traces collected during six one-hour sampling periods over 7 consecutive days in late September 2000. For both links, the clients are very close to the link, and there is little congestion on the client network. This allows us, for modeling purposes, to use the time stamps of the client SYN packets as the client connection start times. For both links, the servers are spread out across the Internet.

The time-stamps of the BELL trace data are accurate to a few  $\mu$ s. At 100 Mbps, time-duration variables as small as packet inter-arrival times are sufficiently accurate to study their statistical properties. This means the BELL data can be used for the source-level model building and for the packet-level validation. The UNC trace data have time-stamps accurate enough to model source-level variables — the connection inter-arrivals, client time gaps, and round-trip times involve differences of time-stamps large enough to be supported by the time-stamp accuracy. But the accuracy is not sufficient for study of the inter-arrival times of successive packets at 1 Gbps, so we do not use the UNC data for packet-level validation.

Recent work has shown that the per-connection variables are nonstationary [17], [16] and their statistical properties change with  $\rho$ , the number of new TCP connections per second or, simply, the connection rate. To study the dependence of per-connection variables on  $\rho$ , we break the measurements into 5 minute time blocks, and obtain a sample of blocks

with the log connection rate spaced as uniformly as possible from the minimum to the maximum log rate. For the BELL database, there are 500 such blocks with the connection rate ranging from 0.18 connections/sec (*c/s*) to 34 *c/s*. For the UNC database, there are 318 such blocks with the  $\rho$  ranging from 2.41 *c/s* to 230 *c/s*.

### C. Source Variables

The stochastic source-level variables describe information about the HTTP transfers, such as the size of the response file from the server, as well as information about the Internet environment at the time of the request, such as the round-trip time between the server and the point of ingress/egress of the modeled network.

Each TCP connection carries one or more request-response exchanges between an HTTP client and server pair. Each exchange consists of request data sent from client to server and response data sent from server to client. An exchange falls into one of two categories depending on the nature of the response file. The response can be the top-level page file, the first one that is sent, or an embedded file. As we will see, certain source variables use the distinction between the top-level file and the embedded file.

Next, we define the source variables that we use to drive the HTTP flow generation, and give mathematical notation for the variables. Connections are ordered by the start time, and  $i$  denotes the  $i$ -th connection:

- 1)  $t_i$ : Inter-arrival time between the start of connection  $i$  on the client and the start of connection  $i + 1$ .
- 2)  $R_i$ : Round-trip time between the server and the point of ingress/egress of the modeled network; includes transmission, propagation, and congestion, and is constant for the duration of each connection; does not include bottleneck link speed.
- 3)  $r_i$ : Same as  $R_i$ , but for the client.
- 4)  $B_i$ : Bottleneck link speed for server.
- 5)  $b_i$ : Same as  $B_i$ , but for the client.
- 6)  $L_i$ : Packet loss probability for server; each packet is dropped with probability  $L_i$ .
- 7)  $\ell_i$ : Same as  $L_i$ , but for the client.
- 8)  $p_i$ : Number of pages requested.
- 9)  $m_{i,j}, j = 1, \dots, p_i$ : Number of request-response exchanges for page  $j$  of connection  $i$ ; the total number of request-response exchanges is  $n_i = m_{i,1} + \dots + m_{i,p_i}$ .
- 10)  $F_{i,\ell}, \ell = 1, \dots, n_i$ : Sizes of server request files
- 11)  $f_{i,\ell}, \ell = 1, \dots, n_i$ : Same as  $F_{i,\ell}$ , but for the client.
- 12)  $g_{i,j}^{(p)}, j = 1, \dots, p_i - 1$ : For  $p_i > 1$ , inter-page time gaps; the time between the arrival at the client of the last packet of the last response file for page  $j$ , and the emergence at the client of the first packet of the first request file for page  $j + 1$ .
- 13)  $g_{i,j,k}^{(e)}, k = 1, \dots, m_{i,j} - 1$ : For  $m_{i,j} > 1$ , inter-exchange time gaps; the time between the arrival at the client of the last packet for response file  $k$ , and the emergence at the client of the first packet of the request file  $k + 1$ .
- 14)  $D_{i,\ell}, \ell = 1, \dots, n_i$ : Server delay; time between the arrival at the server of the last packet of the request file and the emergence at the server of the first packet of the response file.

#### D. Per-Connection Models and Per-Request Models

Per-request variables are those that can take on more than one value per TCP connection when it is a persistent connection. They are variables 9 to 14 in the above list. All other variables, together with the first values of the per-request variables (e.g.,  $f_{i,1}$ ), are per-connection variables. There is one value per connection of these variables.

A per-connection model is a time series model for the sequence of values of a per-connection variable. Driving the statistical behavior of the variable is the sequence of client-server pairs in  $i$ . The temporal locality of these sequences can induce persistent long-range dependence in the variable — positive autocorrelations that fall off slowly. Empirical study, both for our modeling purposes here, and in other work in modeling, shows that the magnitude of the locality dissipates as the new connection rate  $\rho$  increases [17], [16]. The cause is the increased intermingling of connections from different clients and servers. We account for this in the modeling. Furthermore, the different per-connection variables are taken to be independent of one another.

The per-request models are stochastic models for all values of the request variables beyond the first, conditional on the first. Empirical study shows that, conditionally, the subsequent values are independent of the values of request variables of all other connections. For example, the  $f_{i,2}, \dots, f_{i,n_i}$  given  $f_{i,1}$  are independent of  $f_{j,k}$  for all  $j \neq i$  and all  $k$ .

#### V. PACKET-LEVEL VALIDATION

We carried out a packet-level validation by comparing the measured BELL packet traffic with synthetic packet traffic generated from an *ns* simulation of HTTP traffic on the Bell Labs link. The traffic variables studied are the packet inter-arrival time series, the packet size time series, connection duration, bit rate, packet rate, and the number of simultaneous active connections. An important aspect of these variables is that they are not directly specified by the source traffic models, but rather are derived from the packet-level process. Except for the connection duration, the variables are studied as a function of the new connection rate  $\rho$ . The reason is that studies show that the statistical properties of the variables change with the magnitude of the multiplexing [16], [18].

We ran 18 simulations to produce 18 synthetic traces: two at each of 9 connection rates ranging from 1 c/s to 256 c/s in multiplicative steps of 2. We ran each simulation to produce about 1 million packets if the duration is at least 600 sec; otherwise we kept running to achieve a duration of 600 sec. This provides a large sample size. The BELL traces are 300 sec. The different durations does not interfere with the validation process but it does mean that when we compare a BELL parameter for the 500 traces and the corresponding synthetic traffic parameter for the 18 traces we can expect to see more variability in the BELL results. Because the BELL traces encounter very little congestion on the measured link, we kept the utilization of the aggregate link low.

##### A. Rates: Packet and Bit

Figure 2 graphs the log packet rate against the log new connection rate,  $\log_2(\rho)$  for the 500 BELL traces and the 18 synthetic traces. Figure 3 graphs the log bit rate against

$\log_2(\rho)$ . Clearly the synthetic traffic is in close agreement with the live traffic. In addition, as one would expect, the patterns on the plots are linear, so the bit rate and the packet rate are each proportional to  $\rho$ .

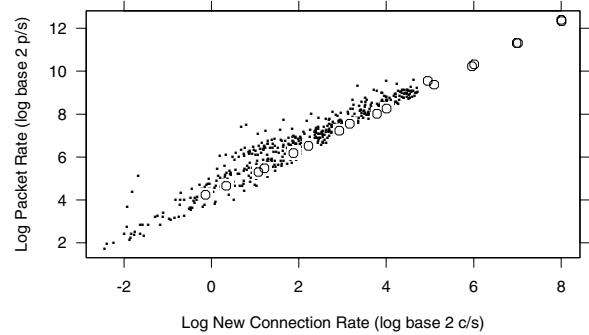


Fig. 2. Log packets/sec is graphed against log new connection rate. Dots: live traces. Circles: synthetic traces.

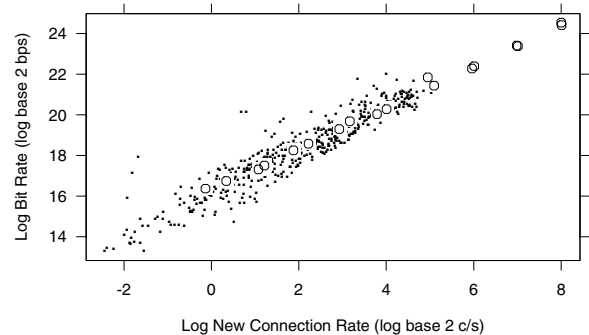


Fig. 3. Log bits/sec is graphed against log new connection rate. Dots: live traces. Circles: synthetic traces.

##### B. Connections: Number Active and Duration

At any given moment, there is a number of simultaneous active HTTP connections. For each trace, live and synthetic, we computed the average number of active connections,  $c$ , across the trace. Figure 4 graphs  $\log_2(c)$  against  $\log_2(\rho)$ . Both patterns, live and synthetic are linear and in agreement.

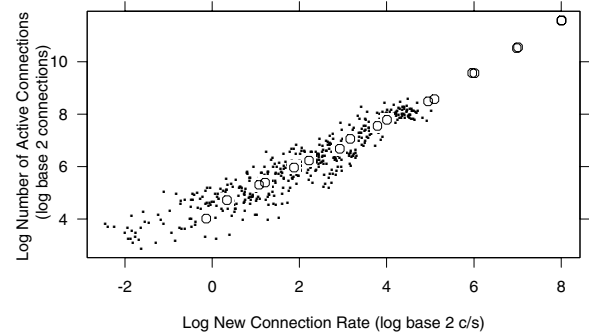


Fig. 4. Log average number of active connections is graphed against log new connection rate. Dots: live traces. Circles: synthetic traces.

$c$  is  $\rho$  times the average duration of a connection. We found that the duration distribution of the live traces and

of the synthetic traces did not change with  $\rho$ , as one would expect, because there is little congestion. Figure 5 is a quantile-quantile (q-q) plot that compares the distribution of the log synthetic durations on the vertical scale with that of the log live durations on the horizontal scale. On the plot, corresponding quantiles of each distribution of values are graphed against one another; for example, the median is graphed against the median, the upper quartile is graphed against the upper quartile, and so forth. The vertical lines on the plot show, left to right, certain quantiles: 1%, 10%, 25%, 75%, 90%, and 99%. If the points follow the line with intercept 0 and slope 1, drawn on the plot, then the distributions are identical. The live and synthetic distributions are close, but there are discrepancies in the tails; for the non-persistent connections the synthetic values are greater and for the persistent connections the synthetic values are smaller. Of course, the source modeling could be the cause, but our validation for the source variables showed good agreement. The likely source of the discrepancy is the packet loss distribution since the durations in the tails are sensitive to it. We did not pursue this further since the discrepancy is minor, but a study of the dependence of the durations on loss would be an interesting topic.

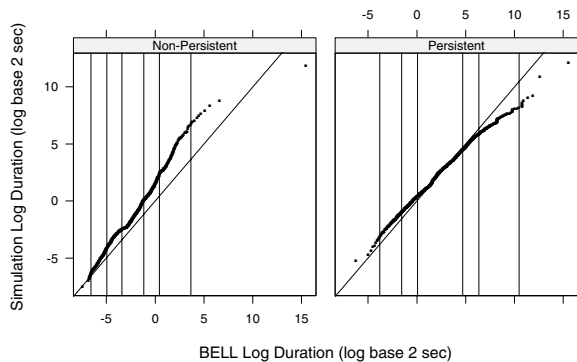


Fig. 5. Quantile-quantile plot of log connection durations. Left panel: non-persistent connections. Right panel: persistent connections.

### C. Marginal Distributions: Inter-Arrivals and Sizes

Studies have shown that the marginal distribution of packet inter-arrivals are well approximated by a Weibull distribution with a heavier upper tail than the exponential [19], [16]. Starting with a quite low new connection rate  $\rho$ , for example, 1 c/s, the approximation is reasonable but with clear departures, but improves with the rate and is quite good about 16 c/s.

The synthetic inter-arrivals show the same characteristics with increasing  $\rho$ , an improving approximation. Figure 6 shows Weibull quantile plots for two traces, one live Bell trace and one synthetic, both with  $\rho$  close to 32 c/s. Each panel graphs quantiles of  $\log_2(t_j)$  against quantiles of the  $\log_2$  of an exponential distribution with mean 1. If the pattern of the points form a line, the distribution of the data is well approximated by a Weibull distribution with a shape parameter the inverse of the slope of the line. The vertical lines indicate the 1%, 10%, 25%, 75%, 90%, and 99% quantiles of the distribution. The oblique line is drawn through the quantiles. Overall, the Weibull approximation is excellent. The only deviation, a minor one, is a truncation at the bottom end of the

distribution of the data for both the synthetic and the live data, so we still have a very close match of live and synthetic. The truncation occurs because there is a minimum inter-arrival time, the smallest packet size divided by the link speed.

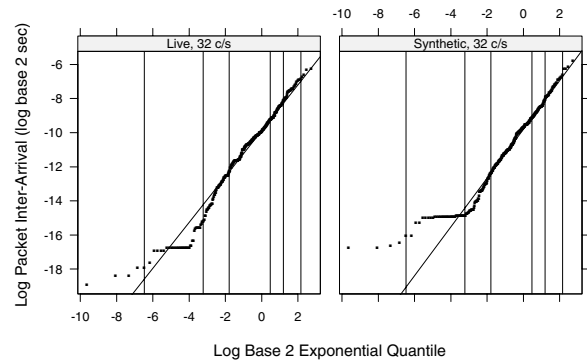


Fig. 6. Weibull quantile plot of inter-arrivals. Left panel: live trace. Right panel: synthetic trace.

Earlier we cited results that as  $\rho$  increases, the packet arrivals tend to Poisson. This means the Weibull shape parameter,  $\lambda$ , less than 1, increases toward 1, the shape parameter of the exponential distribution. In fact, for live traces,  $\log_2(\lambda/(1-\lambda)) = \text{logit}_2(\lambda)$  increases linearly with  $\log_2(\rho)$  [16]. (Note that as  $\lambda \rightarrow 1$ ,  $\text{logit}_2(\lambda) \rightarrow \infty$  and conversely.) Figure 7 graphs  $\text{logit}_2(\lambda)$  against  $\log_2(\rho)$  for the live and synthetic traces. For both live and synthetic there is marked trend of  $\lambda$  toward 1. The pattern on the plot is linear for the live traces but slightly curved for the synthetic traces, a small but consistent departure.

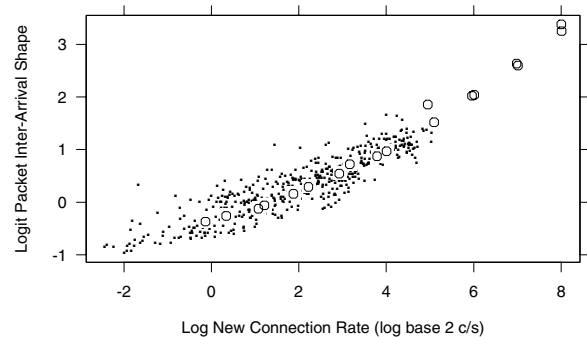


Fig. 7. The logit function of the Weibull shape is graphed against log new connection rate. Dots: live traces. Circles: synthetic traces.

The packet size distribution for live traces is a discrete-continuous distribution that does not change appreciably with  $\rho$  on a link, but does change from link to link [16]. The HTTP packet sizes in a single direction on the link depend on the mix of clients and servers in the transmitting cloud; servers send a higher proportion of 1500 byte packets (data) than clients and clients send a higher proportion of 40 byte packets (ACKs). Figure 8 shows quantile plots of packets sizes for a random sample of live BELL connections and for all connections of the synthetic traces. The distributions are similar.

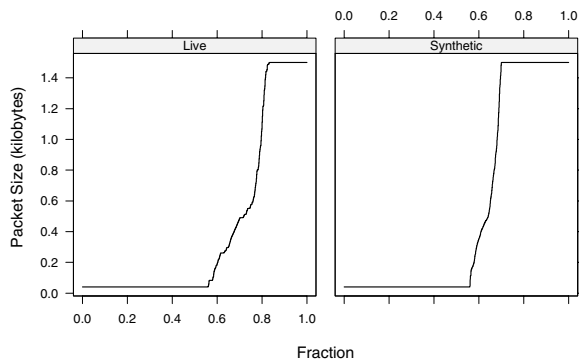


Fig. 8. Packet size quantile plot. Left panel: live trace. Right panel: synthetic trace.

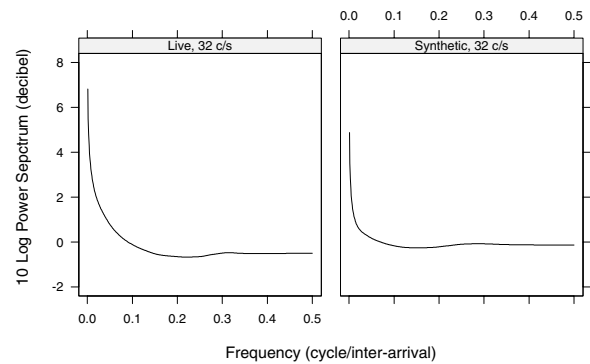


Fig. 10. Packet inter-arrival spectrum is graphed against frequency. Left panel: live trace. Right panel: synthetic trace.

#### D. Time Dependence: Inter-Arrivals and Sizes

The packet inter-arrivals and size processes of live traces are long-range dependent [20], [21], [16]. This is evident in the live BELL packet traces. It is also evident in the synthetic traces. We used the power spectrum of the live and synthetic inter-arrivals and sizes to study their time dependence. First, though, using their marginal distributions, we transformed these series to have Gaussian marginals with mean 0 and variance 1 so that the power spectrum would come closer to fully characterizing their dependence. We found that the synthetic spectra followed closely the live spectra, including the form of the dissipation of the long-range dependence in the series. Figure 9 shows packet size spectrum estimates for two traces, one live Bell trace and one synthetic, both with  $\rho$  close to 16 c/s. The rapid ascent at the origin is the result of the long-range dependence; in both cases the spectra decline nearly monotonically as the frequency increases. Similar results hold for the packet inter-arrival spectra, graphed in Figure 10.

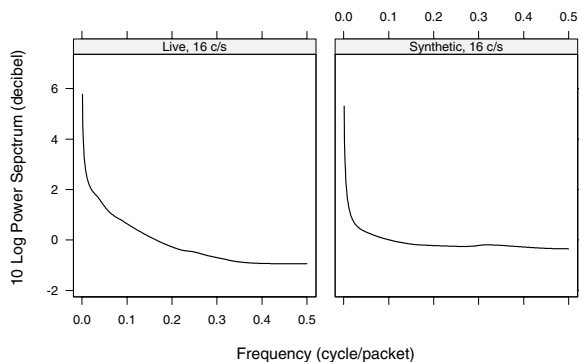


Fig. 9. Packet size spectrum is graphed against frequency. Left panel: live trace. Right panel: synthetic trace.

As  $\rho$  increases, the live and synthetic packet size spectra tend toward a constant, a white noise spectrum; this is the dissipation of the long-range dependence cited earlier. As  $\rho$  increases, the fraction of power at low-frequencies decreases and the fraction of power at high frequencies increases. The one-step entropy  $\tau_1$  reflects this change; it is the variance of the error of linear prediction one step ahead from the infinite past. Because our packet sizes are transformed and rescaled to have mean 0 and variance 1,  $0 < \tau_1 \leq 1$ . If

$\tau_1 = 1$ , the series is independent (uncorrelated); if  $\tau_1$  is close to 0, the series is highly dependent in the sense that it can be predicted reliably from the past. As  $\rho$  increases,  $\tau_1 \rightarrow 1$ . For a wide range of live traces, the logit one-step entropy,  $\log_2(\tau_1/(1 - \tau_1)) = \text{logit}_2(\tau_1)$ , is linear in  $\rho$  [16]. Figure 11 graphs  $\text{logit}_2(\tau_1)$  against  $\log_2(\rho)$ . The patterns of the live and synthetic traces agree. Similar results hold for the entropy of the packet inter-arrival process, which is displayed in Figure 12.

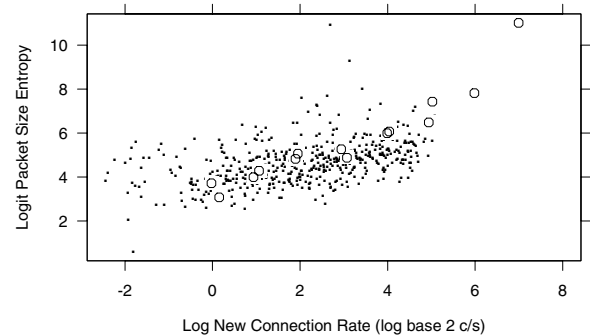


Fig. 11. Logit one-step entropy is graphed against log new connection rate for the packet sizes. Dots: live traces. Circles: synthetic traces.

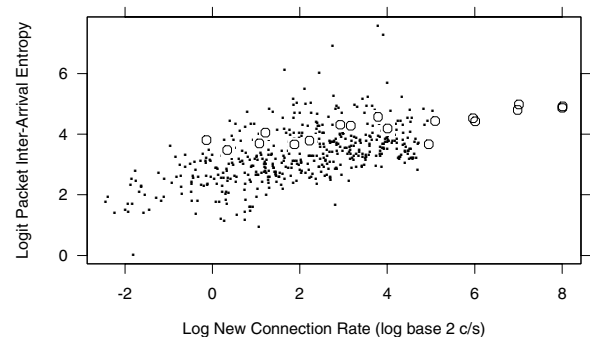


Fig. 12. Logit one-step entropy is graphed against log new connection rate for the packet inter-arrivals. Dots: live traces. Circles: synthetic traces.

## VI. SUMMARY, ARCHITECTURE, AND GENERATION

### A. Models for Generation

This paper presents a new model of HTTP 1.0 and 1.1 source traffic as it appears in aggregate on an access link. The model is used to generate synthetic source traffic for a network modeled either by a network simulator such as *ns* or a hardware testbed.

The model derives from a large-scale empirical study of web traffic on the two access links that provide Internet connections for Bell Labs and for the University of North Carolina at Chapel Hill. The model is novel in that it expresses web traffic as a sequence of TCP connections in which aspects of each connection are described by values of stochastic source-traffic variables. The model consists of a collection of statistical models that determine the stochastic properties of the variables.

### B. General Architecture

The architecture is illustrated in Figure 13. First, there is the modeled network: routers, links, protocol stacks, and so forth. Next, there are host clouds. Each cloud is a collection of servers or a collection of clients that is connected to the modeled network at a node consisting of an ingress, where the packets leave the cloud and enter the network, and an egress, where the packets leave the network and enter the cloud. The modeled network runs the TCP connections, but certain stochastic source variables of our model generate values that interact with each connection as described below.

Each TCP connection carries one or more request-response exchanges between an HTTP client and server. Each exchange consists of request data sent from client to server and response data sent from server to client. TCP connections used for more than one HTTP request-response exchange are called persistent connections.

Suppose there are  $n_c$  client clouds and  $n_s$  server clouds. For each of the  $n_c n_s$  client-server pairs, the model is used to generate the source traffic for the pair that traverses the modeled network. The resulting  $n_c n_s$  source traffic streams are mutually independent.

### C. Source Traffic Variables

Connections are ordered by the start time, and  $i$  denotes the  $i$ -th connection. The source-traffic variables are the following:

- 1)  $t_i$ : Inter-arrival time between the start of connection  $i$  on the client and the start of connection  $i + 1$ .
- 2)  $R_i$ : Round-trip time between the server and the point of ingress-egress of the modeled network; includes transmission, propagation, and congestion, and is constant for the duration of each connection; does not include bottleneck link speed [67 ms].
- 3)  $r_i$ : Same as  $R_i$ , but for the client.
- 4)  $B_i$ : Bottleneck link speed for server.
- 5)  $b_i$ : Same as  $B_i$ , but for the client.
- 6)  $L_i$ : Packet loss probability for server; each packet is dropped with probability  $L_i$ .
- 7)  $\ell_i$ : Same as  $L_i$ , but for the client.
- 8)  $p_i$ : Number of pages requested.

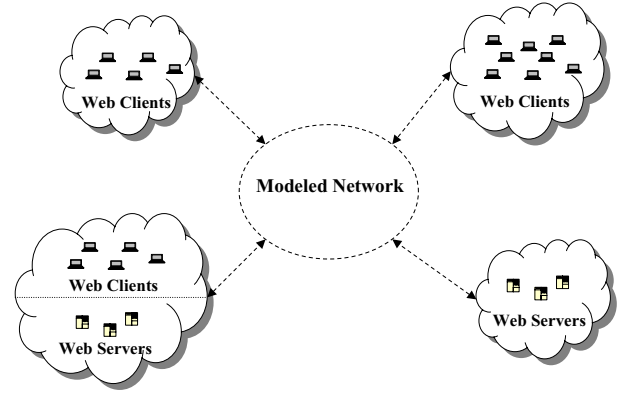


Fig. 13. Architecture of Server Clouds, Client Clouds, and Modeled Network. The example shows two client clouds, one server cloud and one cloud (bottom left) that has both a set of servers and a set of clients sharing the same ingress-egress link to the modeled network. The dashed lines represent the ingress-egress links. The modeled network (together with the ingress-egress links) represent how the clouds interact with the other.

- 9)  $m_{i,j}, j = 1, \dots, p_i$ : Number of request-response exchanges for page  $j$  of connection  $i$ ; the total number of request-response exchanges is  $n_i = m_{i,1} + \dots + m_{i,p_i}$ .
- 10)  $F_{i,\ell}, \ell = 1, \dots, n_i$ : Sizes of server request files.
- 11)  $f_{i,\ell}, \ell = 1, \dots, n_i$ : Same as  $F_{i,\ell}$ , but for the client.
- 12)  $g_{i,j}^{(p)}, j = 1, \dots, p_i - 1$ : For  $p_i > 1$ , inter-page time gaps; the time between the arrival at the client of the last packet of the last response file for page  $j$ , and the emergence at the client of the first packet of the first request file for page  $j + 1$ .
- 13)  $g_{i,j,k}^{(e)}, k = 1, \dots, m_{i,j} - 1$ : For  $m_{i,j} > 1$ , inter-exchange time gaps; the time between the arrival at the client of the last packet for response file  $k$ , and the emergence at the client of the first packet of the request file  $k + 1$ .
- 14)  $D_{i,\ell}, \ell = 1, \dots, n_i$ : Server delay; time between the arrival at the server of the last packet of the request file and the emergence at the server of the first packet of the response file.

Variables 9 to 14 in this list are per-request variables, those that can take on more than one value per TCP connection when it is a persistent connection. All other variables together with the first values of the per-request variables are per-connection variables; there is one value per connection of these variables.

For each connection, the round-trip time between the ingress-egress links in the network and a host in the server cloud is given by a value of source variable 2 in the above list, and the round-trip time between the ingress-egress link in the network and a host in the client cloud is given by a value of variable 3. To implement this in a network simulator or in a hardware test bed, TCP needs to interact with these round-trip times in that a packet traveling between a host and the modeled network is delayed by one-half of the round-trip time. Similar statements hold for the bottleneck link speeds and the packet losses.

### D. Differences with Previous Approaches

Our approach differs from the dominant present-day approach of constructing “page-based” models of web traffic. We



argue that our “connection-based” modeling method is more appropriate for network traffic simulation, as opposed to server workload simulation, because it both captures relationships and dependencies not present in existing page-based models and it is an approach that is more likely to scale to modeling the traffic generated by other application classes such as peer-to-peer file sharing traffic.

#### E. NS Implementation

Our model of aggregate HTTP traffic has been implemented in the *ns* network simulator and is available for use in generating realistic synthetic web traffic in network simulations. The model and this implementation, named PackMime-HTTP, have been validated through both empirical and analytical analyses that are presented here.

There are three levels of specification of the properties of the source traffic. For Levels 1 and 2, specification means giving values to one or more specification parameters. The parameters, except for one, have defaults. Level 3 specification provides a mechanism to generate values of any one of the source traffic variables using any generation facility of *ns*.

The one parameter without a default is  $\rho$ , the number of new connections per second. It measures the source traffic load. Also,  $1/\rho$  is the mean of the inter-arrival variable  $t_i$ . There is no default because the load is so critical to network traffic characteristics and must be commensurate with the modeled network. In Level 1 specification, only  $\rho$  is specified.

The value of  $\rho$  also has a dramatic effect on many of the statistical properties of the source traffic variables. If we multiply  $\rho$  by 10, then, of course, the mean of the inter-arrival variable  $t_i$  is decreased by a factor of 10, but the statistical properties change in a number of other ways. There is a major change in the general form of the marginal distribution of  $t_i$ ; it tends toward the exponential as  $\rho$  increases. There is a major change in the time dependence of  $t_i$ ,  $r_i$ ,  $R_i$ ,  $f_i$ ,  $F_i$ , and  $D_i$  as  $\rho$  increases; long-range dependence in these variables dissipates, and the variables tend toward independence. Our models change with  $\rho$  to reflect these changing properties.

Level 2 provides for increases or decreases in the marginal distribution of source variables 2 through 14. For example, the source variable  $R_i$  has a default marginal distribution. The user can specify an alternative distribution that is a positive constant times the default variable; for example, if the constant is 2, the round-trip times double. The details of Level 2 specification are given in our documentation for PackMime-HTTP.

### APPENDIX I SOURCE MODELS

#### A. Measures of Source Variables

In the following, we describe how we measured the source variables from the time-stamps of the packet arrivals on the links and the corresponding TCP/IP headers, with the exception of  $L_i$ ,  $l_i$ ,  $B_i$  and  $b_i$ .

- $t_i$ : time between the previous client SYN and current client SYN at the monitoring link
- $R_i$ : time between the client SYN and the server SYN+ACK
- $r_i$ : time between the server SYN+ACK and the client ACK that completes the 3-way handshake

- $p_i$ : determined by thresholding the inter-exchange time gaps as discussed in Section I-D. An request/response exchange is defined as a transmission from the client of packets containing request data followed by a transmission from the server of packets containing response data, and an inter-exchange time gap is the maximum of (1) zero and (2) time between the arrival of last data packet from the server of previous exchange and the next data packet from client of current exchange, minus the client-side round trip time)
- $m_{i,j}$ ,  $j = 1, \dots, p_i$ : determined by thresholding the inter-exchange time gaps as discussed in Section I-D
- $f_{i,1}, f_{i,2}, \dots, f_{i,n_i}$ : computed from sequence numbers
- $F_{i,1}, F_{i,2}, \dots, F_{i,n_i}$ : computed from sequence numbers
- $g_{i,j}^{(p)}$ ,  $j = 1, \dots, p_i - 1$ : maximum of (1) zero and (2) time between arrival of last data packet from server of previous exchange and the next data packet from client of current exchange from a different top-level page, minus the client-side round trip time  $r_i$
- $g_{i,j,k}^{(e)}$ ,  $k = 1, \dots, m_{i,j} - 1$ : maximum of (1) zero and (2) time between arrival of last data packet from server of previous exchange and the next data packet from client of current exchange from the same top-level page, minus the client-side round trip time  $r_i$
- $D_{i,1}, D_{i,2}, \dots, D_{i,n_i}$ : maximum of (1) zero and (2) time between the last client data packet and the first server data packet, minus the server-side round trip time  $R_i$

There are certain weaknesses in using the measurement data to build models used in the simulation. For example, we model the  $t_i$  measured at the monitoring link but use the derived model to generate connection arrivals at the client in the simulation. To minimize this problem, for each of the links we traced, the clients are close to the monitoring point.

#### B. Fractional Sum-Difference (FSD) Time Series Models

Most of the connection variables are well modeled by fractional sum-difference, or FSD, time series models [17], [16]. Let  $y_i$  be an FSD time series. The marginal cumulative distribution function of  $y_i$ , which is general, is used to transform the variable to  $z_i$ , a time series with a normal marginal with mean 0 and variance 1.  $z_i$  is assumed to be a Gaussian time series with parameters  $d$  and  $\theta$ :  $z_i = \sqrt{1-\theta} s_i + \sqrt{\theta} n_i$ , where  $0 \leq \theta \leq 1$ ,  $n_i$  is white noise, and  $s_i$  is a simple fractional ARIMA with fractional difference exponent  $d$ .

#### C. Per-Connection Models

1) *Connection Inter-Arrivals,  $t_i$* : The  $t_i$  are modeled by an FSD time series. The marginal distribution is Weibull with shape parameter  $0 \leq \lambda(\rho) \leq 1$  and scale parameter  $\alpha(\rho)$  that change with the new connection rate  $\rho$ . Let  $\text{logit}_2(x) = \log_2(x/(1-x))$  be the logistic transformation where  $\log_2$  is the log base 2. The model for  $\lambda(\rho)$  is a logistic that is linear in  $\log_2(\rho)$ :  $\text{logit}_2(\lambda(\rho)) = 0.352 + 0.388 \log_2(\rho)$ , where the numeric values are estimates from the combined UNC and BELL traces. (Separate estimates were found to be close.) From the properties of the Weibull,  $\alpha(\rho) = [\rho \Gamma(1 + \lambda^{-1}(\rho))]^{-1}$ . The time series parameter  $\theta(\rho)$  is also modeled by a logistic linear in  $\log_2(\rho)$ :  $\text{logit}_2(\theta(\rho)) = 0.333 + 0.414 \log_2(\rho)$ . Finally, the

fractional exponent  $d$  is constant with  $\rho$  and its estimate is 0.33.

2)  $r_i$ ;  $R_i$ : The  $r_i$  and  $R_i$  are modeled by FSD time series. For BELL and UNC,  $r_i$  and  $R_i$  have marginal distributions that do not change with the rate. However, there is a large difference in the marginals between the two links. The  $R_i$  of UNC tends to be somewhat bigger than that of BELL: the median is 89msec for UNC and 67msec for BELL. Most of  $r_i$  in the two datasets are close to zero except that of connections generated from client remote access (typically dial-up modem connections). There are a little over 20% such connections in BELL and around 6% in UNC. Because the marginals of round trip times are network dependent, they need to be specified for generation. The marginal distributions of the  $r_i$  and  $R_i$  are each modeled by piecewise Weibulls. The shape parameter is  $1/3$  for the  $r_i$  and  $1/5$  for the  $R_i$ . A piecewise Weibull is specified in the following way. Let  $0 < b_1 < b_2 \dots < b_r < \infty$  be  $r$  specified break points which divide the positive real line into  $r + 1$  intervals where the last interval is  $b_r$  to infinity. Probabilities  $\psi_k$  for  $k = 0$  to  $r$  are assigned to the interval whose left endpoint is  $b_k$ . Interestingly, although BELL and UNC have vastly different marginals, the time dependence of the FSD models is consistent in that  $\theta$  and  $d$  can be modeled in the same way. As with all variables,  $\theta$  changes with  $\rho$  but  $d$  is constant. For  $r_i$ ,  $d = 0.31$  and  $\theta(\rho)$  is logistic linear in  $\log_2(\rho)$ :  $\text{logit}_2(\theta(\rho)) = -0.445 + 0.554 \log_2(\rho)$ . Similarly, for  $R_i$ ,  $d = 0.32$  and  $\text{logit}_2(\theta(\rho)) = -0.053 + 0.396 \log_2(\rho)$ .

3)  $B_i$ ;  $b_i$ ;  $L_i$ ;  $l_i$ : We do not model these source variables using the packet header databases directly, but rather used the data indirectly by finding models that resulted in expected properties of the packet-level traffic. Based on the BELL data, we sample  $B_i$  and  $b_i$  independently from a uniform distribution from the interval 1 Mbps to 10 Mbps for each client and server cloud, and sample the loss probability independently from the mixture of a uniform distribution on the interval 0% to 1% with probability 0.95 and a point mass at 30% with probability 0.05. The model for the  $L_i$  and  $l_i$  reflects a phenomenon of most connections having low loss but a small number experiencing serious congestion.

4)  $f_{i,1}$ ;  $F_{i,1}$ : The marginal distributions of the  $f_{i,1}$  and the  $F_{i,1}$  do not change with  $\rho$  and were fitted using an approach similar to that for the  $r_i$  and the  $R_i$  except that the marginal is piecewise Pareto, including the upper tail above the final breakpoint  $b_r$  which is consistent with previous work on response size distributions [22], [23], [24], [19], [6]. Interestingly, the marginal distributions of the request and response sizes for both BELL and UNC data are very similar.

The time series  $f_{i,1}$  is modeled by an FSD model. The estimated value of  $d$  is 0.31 and the logistic model for  $\theta(\rho)$  is  $\text{logit}_2(\theta(\rho)) = 0.123 + 0.494 \log_2(\rho)$ .

For the  $F_{i,1}$ , we discovered that a specialized model fitted the data better than the FSD because of the special nature of the  $F_{i,1}$  which generally contain two types of responses, “not modified” messages and content objects (HTTP “entity-body”). We found that response sizes less than 275 bytes were largely “not modified” messages, but there are few such messages above 275 bytes, so we took 275 bytes to be a cut-off to distinguish the two.

To model the dependence of the  $F_{i,1}$ , let  $\phi_i$  be an indicator

variable, which is 1 if the response contains a content object and 0 otherwise. The time sequence  $\phi_i$  consists of alternating runs of 0’s and 1’s. The lengths of the sequences are taken to be independent. For the first 0 of each run, we generate the response size using the portion of the distribution below 275 bytes; and for a 1, we do the same, but using the portion of the distribution above 275 bytes.

We use a discrete Weibull distribution, a probability distribution on the positive integers, to model the run length distributions. The distribution is formed by taking a Weibull with scale  $\alpha$  and shape  $\lambda$  and rounding up to the nearest integer. If the shape parameter is 1, then the run length distribution is geometric; when this occurs, the  $\phi_i$  is a Bernoulli series, that is, independent.

The shape parameter  $\lambda$  depends on  $\rho$ , and can be modeled by a logistic that is linear in  $\log_2(\rho)$ . For the “not modified” messages,  $\text{logit}_2(\lambda(\rho)) = 0.718 + 0.357 \log_2(\rho)$ . For the content objects,  $\text{logit}_2(\lambda(\rho)) = 1.293 + 0.316 \log_2(\rho)$ . For the scale parameter  $\alpha$ , because the probability of “not modified” message derived from the fitted piecewise Pareto distribution of  $F_{i,1}$  is  $q = 27.5\%$  and because  $\phi_i$  approaches to a Bernoulli series as  $\rho$  goes to infinity, the scale parameters of the run lengths tend to  $-1/\log(q) = 0.775$  for “not modified” message and  $-1/\log(1-q) = 3.11$  for content. For simplicity, we take the scale parameters to be constant with these values.

5)  $D_{i,1}$ : Our measurements of  $D_{i,1}$  can determine their marginal distributions, but because data are routinely missing through time, it is not possible to determine their time series characteristics. Thus we take  $D_{i,1}$  as an independent series. The marginal distribution of  $D_{i,1}$  for BELL and UNC are quite similar and are well modeled by an inverse Weibull. The parameter estimate of the inverse Weibull distribution based on the BELL data is shape  $\alpha = 0.63$  and scale  $\lambda = 305$ . To avoid generating very large delays, we limit the maximum delay to 10 sec.

#### D. Per-Request Models

Each persistent connection may contain HTTP requests resulting from different references to top-level web pages (typically a file with HTML content). Empirical analysis reveals that connection variables tend to be similar within top-level pages, so explicit clustering of the requests according to top-level pages adds to the verisimilitude of the models. Similar to the approach developed in [3], [4], [12], we use time gaps between consecutive requests to do the clustering, because the time gap tends to be small if they are related to the same top-level page. We set a time threshold, and gaps above this threshold are taken to be the start of a new top-level page.

In the BELL data, there is a significant population of home users with modem access which produces large client-side round trip times. Since the client round trip time varies within a connection, with possibly greater variability for larger round trip times, the measured time gaps in the BELL data has a higher percentage of zeroes. For this reason, we use only the UNC data to set the threshold and to model the number of pages, requests of the pages, as well as the time gaps within a connection.

We first fit the distribution of a time gap  $g$  by a two component mixture model of log-normal distributions, where we

use the log-normal with the smaller mean for the conditional distribution of  $g$  between requests within the same top-level page, and the one with the larger mean for the conditional distribution of  $g$  between requests from different top-level pages. For UNC data, since the two log-normal components cross at 0.5 log base 2 sec, we use a threshold of  $2^{0.5}$  sec = 1.4 sec for the clustering.

1)  $p_i; m_{i,1}, \dots, m_{i,p_i}$ : We assume that the set of variables  $\{p_i, m_{i,j}, j = 1, \dots, p_i\}$  for each connection are independent. From our exploratory analysis, it is also reasonable to assume that conditional on  $p_i > 1$ ,  $m_{i,j}, j = 1, \dots, p_i$  are independent.

We model the  $p_i$  by a mixture distribution. First we estimate the probability that  $p_i = 1$ . Note that  $P(p_i = 1) = P(n_i = 1) + P(p_i = 1, n_i > 1)$ , and the estimate of  $P(n_i = 1)$  is approximately 0.91 for both BELL and UNC data, and the estimate of  $P(p_i = 1 | n_i > 1)$  is 0.82 using UNC data, our estimate of  $P(p_i = 1)$  is 0.984. Next we model  $p_i$  given  $p_i > 1$  by 1 plus a discrete Weibull with shape parameter  $\lambda$  and scale parameter  $\alpha$ . (The discrete Weibull is introduced in Section I-C.4). The estimate of  $\lambda$  is 1.00 and the estimate of  $\alpha$  is 0.37.

To model  $m_{i,1}$ , we first consider the probability of  $m_{i,1} = 1$  given that  $p_i = 1$ , i.e.,  $P(m_{i,1} = 1 | p_i = 1)$ . This is the same as  $P(n_i = 1)/P(p_i = 1)$ , whose estimate is 0.91/0.984 = 0.925. Next we model the probability of  $m_{i,j} = 1$  given that  $p_i > 1$ , which estimated to be 0.69. Finally, we model  $m_{i,j}$  conditional on  $m_{i,j} > 1$  independent of  $p_i$ , using 1 plus a discrete Weibull; the estimate of scale parameter  $\lambda$  is 1.00 and the estimate of shape parameter  $\alpha$  is 2.12.

2)  $g_{i,j}^{(p)}, j = 1, \dots, p_i - 1; g_{i,j,k}^{(e)}, k = 1, \dots, m_{i,j} - 1$ : We find the variability of these time gaps within a connection is much smaller than the variability between connections. In addition, this reduced variability varies from connection to connection. To reflect these observations, we developed a mixed-effects model with random location and scale effects. The random location-scale model has been investigated in [25], and we use the method of moments based on the estimates of mean and variance within each persistent connection to obtain estimates of parameters.

We assume that the all time gaps for different connections are independent. We now specify the model of time gaps for fixed  $i$ . Let  $\mu_i$  be independent normal random variables with mean  $\mu$  and variance  $\sigma^2(\mu)$ . This random variable will serve as a random location effect across  $i$ . Let  $\gamma_i^2$  be independent Gamma random variables with shape  $\lambda$  and scale  $1/\lambda$ . Note that  $E(\gamma_i^2) = 1$ . Let  $g_{ij}$  either  $g_{i,j}^{(p)}$  or  $g_{i,j,k}^{(e)}$  inter-exchange gap. The model for  $g_{ij}$  is  $\log_2(g_{ij}) = \mu_i + \gamma_i \epsilon_{ij}$ , where  $\epsilon_{ij}$  is a normal random variable with mean 0 and variance  $\sigma^2(\epsilon)$ . For inter-exchange gaps, the parameter estimates from UNC data are  $\mu = -5.26$ ,  $\sigma^2(\mu) = 2.14$ ,  $\lambda = 2.18$ , and  $\sigma^2(\epsilon) = 2.40$ . For inter-page time gaps, the fitted parameters are  $\mu = 1.73$ ,  $\sigma^2(\mu) = 0.19$ ,  $\lambda = 1.77$ , and  $\sigma^2(\epsilon) = 0.85$ .

3)  $f_{i,2}, \dots, f_{i,n_i}$ : We found the variability of the request sizes relative to the mean within a persistent connection is small. Thus we model the  $f_{i,2}, \dots, f_{i,n_i}$  by taking them equal to  $f_{i,1}$ .

4)  $F_{i,2}, \dots, F_{i,n_i}$ : We found that for the responses, it is almost always the case that all are “not modified” messages

or all are content objects. So the model, for simplicity, makes all responses within a connection of the same type.

For the “not modified” cases, the variability of the sizes relative to the mean is small, so we model the  $F_{i,2}, \dots, F_{i,n_i}$  by taking them equal to  $F_{i,1}$  when  $F_{i,1}$  is less than 275 bytes.

For the content objects, it has been observed that the embedded objects tend to be smaller than the top-level object (typically HTML content) [12]. This suggests that the first response size in a top-level page tends to be larger than the remaining sizes in the page (embedded objects). We did see evidence of this in BELL and UNC traces, with the average ratio of the two types of response sizes at about 90%. This is a rather small difference, and for simplicity, we ignore this in the modeling.

We found that the response sizes are less variable within persistent connections relative to the mean than across all connections. We model this using the mixed-effects model with random location and scale effects that we use for the time gaps. The estimates we obtained from BELL and UNC traces are very similar, and we use the average of the two in the model. The model is  $\log_2(F_{ij}) = \mu_i + \gamma_i \epsilon_{ij}$ , where  $\mu_i$  has a piecewise Pareto distribution same as that of the content objects ( $F_{i,1} | F_{i,1} > 275$ ) but rescaled to have a variance of 0.94,  $\gamma_i^2$  is a Gamma random variable with shape 3.22 and scale  $3.22^{-1}$ , and  $\epsilon_{ij}$  is a normal random variable with mean 0 and variance 2.43.

Although the marginal distribution of sizes of content objects derived from the above mixed effects model is different from that derived from  $F_{i,1}$  in Section I-C.4, it is a very close approximation: the body of the distribution is very similar but the tails are somewhat heavier. Also note that  $F_{i,1}$  for the top-level page is independent across connections, we can simply generate all sizes  $F_{i,1}, \dots, F_{i,n_i}$  (including the first) independently from the mixed effects model.

5)  $D_{i,2}, \dots, D_{i,n_i}$ : We found the variability of the server delays relative to the mean within a persistent connection is small. Thus we model the  $D_{i,2}, \dots, D_{i,n_i}$  by taking them equal to  $D_{i,1}$ .

## APPENDIX II NETWORK SIMULATION IN NS

PackMime-HTTP is the *ns* object that drives the generation of HTTP traffic using the models and integration described in this paper. It is available at <http://www.isi.edu/nsnam/ns/ns-contributed.html>. The object consists of one client cloud and one server cloud. Each cloud is represented by a single *ns* node that can produce and consume multiple HTTP connections at a time. For each HTTP connection, PackMime-HTTP creates a new web client and a new web server, sets up a TCP connection between the client and server, has the client send an HTTP request, and sets a timer to expire when the next new connection should begin. The time between new connections is governed by the connection rate parameter supplied by the user. New connections are started according to the connection arrival times without regard to the completion of previous requests, but a new request between the same client and server pair begins only after the previous request-response pair has been completed.

Each web client controls the HTTP request sizes that are transferred. The client is started when a new TCP connection

is started. PackMime-HTTP samples the number of requests for this connection from the number-of-requests distribution, and then samples the inter-request times and the HTTP request sizes from the appropriate distributions. Then the client sends the first HTTP request to the server, and listens for the HTTP response. When the entire HTTP response has been received the client sets a timer to expire when the next request should be made. When the timer expires, the next HTTP request is sent, and the above process is repeated until the requests are exhausted.

Each web server controls the response sizes that are transferred. The server is started by when a new TCP connection is started. The server listens for an HTTP request from its associated client. When the request arrives, the server samples the server delay time from the server delay distribution and sets a timer to expire when the server delay has passed. When that timer expires, the server samples the HTTP response sizes from the HTTP response size distribution. This process is repeated until the requests are exhausted. (The server is told how many requests will be sent in the connection.) Then the server sends a FIN.

PackMime-HTTP uses *ns* to model the TCP-level interaction between web clients and servers on the simulated link. To simulate network-level effects of HTTP transfer through the clouds, we implemented a new *ns* module called DelayBox. DelayBox is an *ns* analog to dummynet [26], often used in network testbeds to delay and drop packets. The transit times model cloud propagation and queueing delay. Since all HTTP connections in PackMime-HTTP take place between only two *ns* nodes, there had to be an *ns* object to delay packets in each flow, rather than just having a static delay on the link between the two nodes. DelayBox also models bottleneck links and packet loss on an individual connection basis. Two DelayBox nodes are used as shown in Figure 14. One node is placed

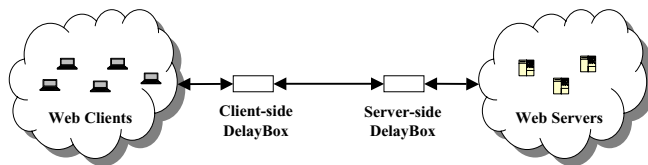


Fig. 14. PackMime-HTTP network with DelayBox

in front of the web client cloud *ns* node to handle client-side delays, loss, and bottleneck links. The other DelayBox node is placed in front of the web server cloud *ns* node to handle the server-side delays, loss, and bottleneck links.

DelayBox manages the per-flow delays and packet drops by maintaining a rule table and a flow table. The rule table is specified by the user and describes how flows from a specific source node to a specific destination node should be treated. The fields in the rule table include the source node, the destination node, the delay distribution, the loss rate distribution, and the bottleneck link speed distribution. The loss rate and bottleneck link speed distributions are optional. Packets in a flow are guaranteed to be transmitted in the same order they arrived at the DelayBox node.

## REFERENCES

- [1] S. Bhattacharyya, C. Diot, R. Gass, E. Kress, S. Moon, A. Nucci, D. Papagiannaki, and T. Ye, "Packet Trace Analysis," Sprint Labs, Tech. Rep., 2003, ipmon.sprintlabs.com.
- [2] S. Floyd and V. Paxson, "Difficulties in Simulating the Internet," *IEEE/ACM Transactions on Networking*, vol. 9, no. 4, pp. 392–403, 2001.
- [3] B. Mah, "An Empirical Model of HTTP Network Traffic," in *IEEE INFOCOMM*, 1997.
- [4] P. Barford and M. Crovella, "Generating Representative Web Workloads for Network and Server Performance Evaluation," *ACM SIGMETRICS*, vol. 26, pp. 151–160, 1998.
- [5] P. Barford, A. Bestavros, A. Bradley, and M. E. Crovella, "Changes in Web Client Access Patterns: Characteristics and Caching Implications," *World Wide Web, Special Issue on Characterization and Performance Evaluation*, vol. 2, pp. 15–28, 1999.
- [6] M. E. Crovella and A. Bestavros, "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes," *ACM SIGMETRICS*, vol. 24, no. 1, pp. 160–169, 1996.
- [7] C. R. Cunha, A. Bestavros, and M. E. Crovella, "Characteristics of WWW Client-based Traces," 1995, technical Report TR-95-010, Boston University Computer Science Department.
- [8] T. Ott, T. Lakshman, and L. Wong, "SRED: Stabilized RED," in *IEEE INFOCOM*, 1999, pp. 1346–1355.
- [9] P. Barford and M. E. Crovella, "A Performance Evaluation of HyperText Transfer Protocols," in *ACM SIGMETRICS*, 1999, pp. 188–197.
- [10] M. Christiansen, K. Jeffay, D. Ott, and F. D. Smith, "Tuning RED for Web Traffic," in *ACM SIGCOMM*, 2000, pp. 139–150.
- [11] H. Nielsen, J. Gettys, A. Baird-Smith, E. Prud'hommeaux, H. Lie, and C. Lilliey, "Network Performance Effects of HTTP/1.1, CSS1, and PNG," in *ACM SIGCOMM*, 1997, pp. 155–166.
- [12] F. D. Smith, F. H. Campos, K. Jeffay, and D. Ott, "What TCP/IP Protocol Headers Can Tell Us About the Web," in *ACM SIGMETRICS*, 2001, pp. 245–256.
- [13] L. Le, J. Aikat, K. Jeffay, and F. Smith, "The Effects of Active Queue Management on Web Performance," in *ACM SIGCOMM*, 2003, (to appear).
- [14] J. Cao, W. S. Cleveland, and D. X. Sun, "The S-Net System for Internet Packet Streams: Strategies for Stream Analysis and System Architecture," *Journal of Computational and Statistical Graphics: Special Issue on Streaming Data*, p. to appear, 2003.
- [15] J. Cao, W. S. Cleveland, and Y. Gao, "Statistical Models for HTTP Aggregate Source Traffic," Bell Labs, stat.bell-labs.com, Tech. Rep., 2004.
- [16] J. Cao, W. S. Cleveland, D. Lin, and D. X. Sun, "On the Nonstationarity of Internet Traffic," *ACM SIGMETRICS*, vol. 29, no. 1, pp. 102–112, 2001.
- [17] W. S. Cleveland, D. Lin, and D. X. Sun, "IP Packet Generation: Statistical Models for TCP Start Times Based on Connection-Rate Superposition," *ACM SIGMETRICS*, pp. 166–177, 2000.
- [18] J. Cao, W. S. Cleveland, D. Lin, and D. X. Sun, "Internet Traffic Tends Toward Poisson and Independent as the Load Increases," in *Nonlinear Estimation and Classification*, C. Holmes, D. Denison, M. Hansen, B. Yu, and B. Mallick, Eds. New York: Springer, 2002, pp. 83–109.
- [19] V. Paxson and S. Floyd, "Wide-Area Traffic: The Failure of Poisson Modeling," *IEEE/ACM Transactions on Networking*, vol. 3, pp. 226–244, 1995.
- [20] V. J. Ribeiro, R. H. Riedi, M. S. Crouse, and R. G. Baraniuk, "Simulation of NonGaussian Long-Range-Dependent Traffic Using Wavelets," *ACM SIGMETRICS*, vol. 27, pp. 1–12, 1999.
- [21] J. Gao and I. Rubin, "Multiplicative Multifractal Modeling of Long-Range-Dependent Network Traffic," *International Journal of Communications Systems*, vol. 14, pp. 783–201, 2001.
- [22] K. Park, G. Kim, and M. Crovella, "On the Relationship Between File Sizes, Transport Protocols, and Self-Similar Network Traffic," in *Proceedings of the IEEE International Conference on Network Protocols*, 1996.
- [23] V. Paxson, "Empirically-Derived Analytic Models of Wide-Area TCP Connections," *IEEE/ACM Transactions on Networking*, vol. 2, pp. 316–336, 1994.
- [24] M. F. Arlitt and C. L. Williamson, "Web Server Workload Characterization: The search for Invariants," *ACM SIGMETRICS*, pp. 126–137, 1996.
- [25] W. S. Cleveland, L. Denby, and C. Liu, "Random Scale Effects," Bell Labs, stat.bell-labs.com, Tech. Rep., 2003.
- [26] L. Rizzo, "Dummynet: A simple approach to the evaluation of network protocols," *ACM Computer Communication Review*, January 1997.