Stochastic Models for Generating Synthetic HTTP Source Traffic

Jin Cao[1], William S. Cleveland[1], Yuan Gao[1], Kevin Jeffay[2], F. Donelson Smith[2], Michele Weigle[2]
[1]Mathematical Sciences Center, Bell Labs, Murray Hill, NJ
[2]Computer Science, University of North Carolina, Chapel Hill

***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 [27] 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. Section IV describes the source-level variables that are modeled and that are necessary for a connection-based generator of synthetic web traffic, how these variables are measured, the process of building models for the source variables, and the process of validation. Section V describes the stochastic models for the source variables. Section VI describes the validation of the models and the architecture that encompasses them. The Appendix provides a description of how this design has been implemented in the *ns* network simulator and how to obtain the *ns* code for the traffic generator.

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 [15]. 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 [18], and Crovella, *et al.*, [3], [2], [13], [14] 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.*, [20]. These models have also been used to generate web-like traffic in network testbeds [4], [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 [19]. 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.*, [26]. 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 [17] 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 V that there are significant dependencies between variables associated with different connections. 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

We now describe the architecture of our approach — what is modeled and how it interacts with the TCP transport layer. 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. 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).

At a very high level, the web traffic in a 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. (We use the term "cloud" to refer to a set of clients or servers whose traffic is aggregated at some link). A simple example of modeling web traffic where the traffic from one set of clients and one set of servers is aggregated at a single link is shown in Figure 1. This example would be used to model a single access link connecting an enterprise or campus network to the Internet. All the web clients are on the enterprise or campus network and all the servers are someplace on the Internet. By using multiple client and server "clouds", arbitrary configurations and loadings of the links in a testbed or simulation can be achieved.

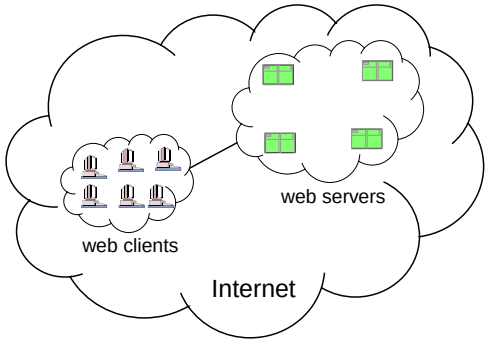

Fig. 1. Client cloud, server cloud, and link carrying web traffic.

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 queueing 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 process 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 [9], 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 of synthetic traffic generated using the model.

In the source-level model building, the packet traces are 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. Section V 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 [5].

The packet-level validation consists of a set of simulation experiments using the *n*s implementation (described in the Appendix) 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 *n*s) 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 VI.

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 [6]. For HTTP, all clients are on one side of the link and all servers are on the other side (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 [26]. There are HTTP clients on both sides of this link. We only use traces for the outbound traffic (from UNC clients to Internet servers), since the monitored link is close to the the clients, similar to the Bell Labs link. The UNC database consists of 42 traces collected during six one-hour sampling periods over 7 consecutive days in late September 2000.

The time-stamps of the BELL trace data are accurate to a few μs. 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 connection variables are nonstationary [12], [6] and their statistical properties change with ρ, the number of new TCP connections per second or, simply, the connection rate. To study the dependence of connection variables on ρ, we break the measurements into 5 minute time blocks, and obtain a sample of blocks with the log of connection rate spaced as uniformly as possible from the minimum to the maximum log of 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 ρ ranging from 2.41 c/s to 230 c/s.

C. Source Variables and Their Measures

The source-level variables describe information about the data transfers, such as the size of the response data from the server, as well as information about the Internet environment at the time of request such as the round-trip time between the server and the client at the time of the request. In the following, we define the source variables, give mathematical notation, and in parentheses, describe how we measured the variables from the time-stamps of the packet arrivals on the links and the corresponding TCP/IP headers.

- a_i: arrival time of ith connection (time stamp of client SYN arriving on the link)
- t_i: inter-arrival time ($t_i = a_{i+1} - a_i$)
- R_i: server-side nominal round-trip time (time between the client SYN and the server SYN+ACK)
- r_i: client-side nominal round-trip time (time between the server SYN+ACK and the client ACK that completes the 3-way handshake)
- n_i: number of request/response exchanges using the connection (each exchange is a transmission from the client of packets containing request data followed by a transmission from the server of packets containing response data)
- $f_{i,1}, f_{i,2}, \ldots, f_{i,n_i}$: client request size for each request (computed from sequence numbers)
- $F_{i,1}, F_{i,2}, \ldots, F_{i,n_i}$: server response size for each response (computed from sequence numbers)
- $D_{i,1}, D_{i,2}, \ldots, D_{i,n_i}$: server delay (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)
- $g_{i,1}$, ..., g_{i,n_i-1}: inter-exchange time gap between the end of a server response and the start of the next client request (time between arrival of last data packet from server and the next data packet from client).

There are certain weaknesses in using these measures in simulations. For example, variables such as $g_{i,j}$ are defined and used as if they are observed at the client but in fact were measured at the link monitor. Analysis during the validation shows that the discrepancies do not produce more than minor problems [5]. However, the problems for the a_i would be more than minor were it not the case that for each of the links we traced, the clients are very close to the monitoring point.

V. Source Models

A. Per-Connection Models and Per-Request Models

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. TCP connections used for more than one HTTP request-response exchange are called persistent connections.

Per-request, or request, variables are those that can take on more than one value per TCP connection because it may be a persistent connection. They are the request sizes $f_{i,1}, f_{i,2}, \ldots, f_{i,n_i}$, the response sizes $F_{i,1}, F_{i,2}, \ldots, F_{i,n_i}$, the server delays $D_{i,1}, D_{i,2}, \ldots, D_{i,n_i}$, and the time gaps $g_{i,1}$, ...,g_{i,n_i-1}. All other variables together with the first values of the request variables (e.g., $f_{i,1}$) are per-connection, or connection, variables. There is one value per connection of these variables.

The connection models are stochastic models for the connection variables. Each model treats the sequence of values of its connection variable (e.g., $R_1, R_2, R_3, \ldots$) as a time series. Driving the statistical behavior of each connection variable is the sequence of client-server pairs in i. The temporal locality of these sequences induces persistent long-range dependence in the resulting packet-level traffic: 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 and therefore the long-range dependence dissipates as the new connection rate ρ increases because the intermingling of connections from different clients breaks up the locality [12], [6]. We account for this in the modeling. Furthermore, the different connection variables are taken to be independent of one another.

The 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 $f_{i,2}, \ldots, f_{i,n_i}$ given $f_{i,1}$ are independent of $f_{j,k}$ for all $j \neq i$ and all k.

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 [12], [6], [8]. 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 variable that has a normal marginal distribution with mean 0 and variance 1. z_i is assumed to be a Gaussian time series with two parameters d and θ and with the following form: $z_i = \sqrt{1-\theta}\ s_i + \sqrt{\theta}\ n_i$, where $0 \leq \theta \leq 1$. n_i is a Gaussian white noise time series with mean 0 and variance 1. s_i is a long-range dependent time series with mean 0 and variance 1, is independent of n_i, and has the form $(I - B)^d s_i = \epsilon_i + \epsilon_{i-1}$, where ϵ_i is normal white noise with mean 0 and variance $\sigma_\epsilon^2 = (1-d)\Gamma^2(1-d)(2\Gamma(1-2d))^{-1}$, B is the backward shift operator $Bs_i = s_{i-1}$, and d is the fractional-difference exponent. (d + 0.5 is the Hurst parameter.)

z_i is made up of two components, the long-range dependent component $\sqrt{1-\theta}\ s_i$ and the white noise component $\sqrt{\theta}\ n_i$. Thus θ is the variance of the white noise component. When θ is close to 1, z_i is nearly independent, and when not, z_i has significant long-range dependence. For all of our connection variables modeled by an FSD, we will see that $\theta \to 1$ as $\rho \to \infty$, so the long-range dependence dissipates and the series tend to white noise.

C. Per-Connection Models

1) Connection Inter-Arrivals t_i: The connection inter-arrivals t_i are modeled by an FSD time series. The marginal distribution is Weibull with shape parameter $0 \le \lambda(\rho) \le 1$ and scale parameter $\alpha(\rho)$ that change with the new connection rate ρ. 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, $\alpha(\rho) = \left[\rho\Gamma(1+\lambda^{-1}(\rho))\right]^{-1}$. 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 ρ and its estimate is 0.33.

2) Client-Side Round-Trip r_i; Server-Side Round-Trip 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 $0 < b_1 < b_2 \ldots < b_r < \infty$ 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 ψ_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 θ and d can be modeled in the same way. As with all variables, θ changes with ρ 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) First Request Size $f_{i,1}$, First Response Size $F_{i,1}$: The marginal distributions of the $f_{i,1}$ and the $F_{i,1}$ do not change with ρ 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 [21], [22], [1], [23], [13]. 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 ϕ_i be an indicator variable, which is 1 if the response contains a content object and 0 otherwise. The time sequence ϕ_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 α and shape λ and rounding up to the nearest integer. If the shape parameter is 1, then the run length distribution is geometric; when this occurs, the ϕ_i is a Bernoulli series, that is, independent.

The shape parameter λ depends on ρ, 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 α, because the probability of "not modified" message derived from the fitted piecewise Pareto distribution of $F_{i,1}$ is $q = 27.5\%$ and because ϕ_i approaches to a Bernoulli series as ρ 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.

4) First Server Delay $D_{i,1}$: 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 $\lambda = 305$.

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 [18], [3], [26], 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.

Since the measured time gap includes the client round trip time, and there is a significant population of home users with modem access in the BELL data, (which produces large client-side round trip times), we use only the UNC data to set the threshold and to model the gaps $g_{i,1}, \ldots, g_{i,n_i-1}$ and the number of requests n_i.

We first fit the distribution of 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) Number of Requests, n_i: Let p_i be the number of top-level pages that have requests using the ith connection, and let $m_{ij}, 1 \le j \le p_i$ be the number of requests for these top-level pages. Then $n_i = \sum_{j=1}^{p_i} m_{ij}$. Each set of variables — n_i, p_i, and m_{ij} — and are independent and identically distributed.

We model the n_i by a sequence of models. We begin by modeling the probability of a non-persistent connection, i.e., $n_i = 1$. This is a special case in which $p_i = m_{i1} = 1$. The estimate of this probability is 0.91 for both BELL and UNC. Next we model n_i for persistent connections. We first model the probability that $p_i = 1$ given $n_i > 1$. The estimate of the probability here using UNC data is 0.82. Next we model p_i given $p_i > 1$ by 1 plus a discrete Weibull with shape parameter λ and scale parameter α. (The discrete Weibull is introduced in Section V-C.3). The estimate of λ is 0.89 and the estimate of α is 0.37. Finally, we model m_{ij} first by considering the probability of $m_{ij} = 1$; this is estimated to be 0.69. Then conditional on $m_{ij} > 1$, m_{ij} is modeled by 1 plus a discrete Weibull; the estimate of λ is 0.74 and the estimate of α is 2.12.

2) Time Gaps, $g_{i,1}, \ldots, g_{i,n_i-1}$: We find the variability of time gaps either within or between page requests is much smaller when compared to the overall variability across all 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 [11], and we use the method of moments based on the estimates of mean and variance within each persistent connection to obtain estimates of parameters.

The g_{ij}, j = 1 to n_i for different i are independent. We now specify the model for g_{ij} for fixed i. Let μ_i be independent normal random variables with mean μ and variance $\sigma^2(\mu)$. This random variable will serve as a random location effect across i. Let γ_i^2 be independent Gamma random variables with shape λ and scale $1/\lambda$. Note that E(γ_i^2) = 1. The model is $\log_2 g_{ij} = \mu_i + \gamma_i \epsilon_{ij}$, where ϵ_{ij} is a normal random variable with mean 0 and variance $\sigma^2(\epsilon)$. For time gaps within top-level pages, the parameter estimates from UNC data are $\mu = -5.26$, $\sigma^2(\mu)$ = 2.14, λ = 2.18, and $\sigma^2(\epsilon)$ = 2.40. For time gaps between top-level pages, the fitted parameters are μ = 1.73, $\sigma^2(\mu)$ = 0.19, λ = 1.77, and $\sigma^2(\epsilon)$ = 0.85.

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

4) Persistent Connection Response Sizes, $F_{i,2}, \ldots, F_{i,n_i}$: 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 $F_{i,2}, \ldots, F_{i,n_i}$ 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) [26]. 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 μ_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, γ_i^2 is a Gamma random variable with shape 3.22 and scale 3.22^{-1}, and ϵ_{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 SectionV-C.3, 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 $F_{i,1}, \ldots, F_{i,n_i}$ (including the first) independently from the mixed effects model.

5) Persistent Connection Server Delay, $D_{i,2}$, ..., 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 $D_{i,2}, \ldots, D_{i,n_i}$ by taking them equal to $D_{i,1}$.

E. Specifying the Models for Generation

There are two categories of source-model variables that need to be specified to utilize the *ns* implementation to generate packet-level traffic — those that are application dependent and those that are dependent on the network properties of the clouds. The most salient application variable is the new connection rate ρ, and for this reason it must be specified. For other application variables, the defaults will frequently suffice. For network-dependent variables, the marginal distributions of the client round-trip time and the server round-trip time are salient network properties and need to be considered as part of the cloud network properties. Their marginal is specified by the break points and the interval probabilities. The default server delay marginal distribution will frequently suffice.

VI. 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 ρ. The reason is that studies show that the statistical properties of the variables change with the magnitude of the multiplexing [6], [7].

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. In these simulations the connection bottleneck link speeds for each cloud is chosen independently from the interval 1 Mbps to 10 Mbps. The connection packet drop rates for each cloud are also chosen independently from a marginal distribution that with probability 0.95 is uniformly distributed on the interval 0% to 1% and with probability 0.05 is 30%; this reflects a phenomenon of most connections having low loss but a small number experiencing serious congestion. 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 ρ.

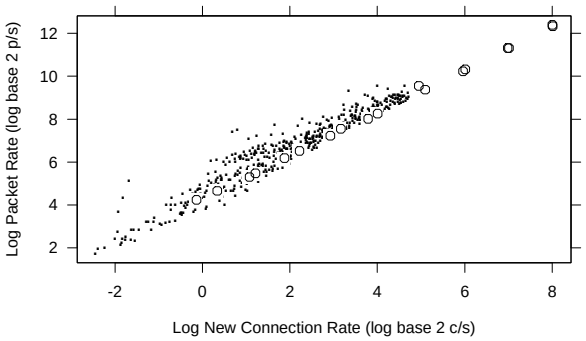

Fig. 2. Log packets/sec is graphed against log new connection rate. Dots: live traces. Circles: synthetic traces.

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

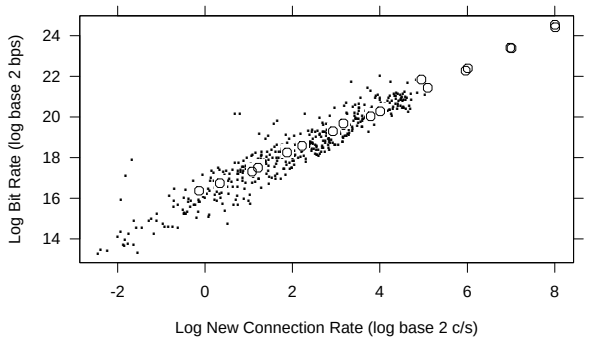

Fig. 3. Log bits/sec is graphed against log new connection rate. Dots: live traces. Circles: synthetic traces.

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.

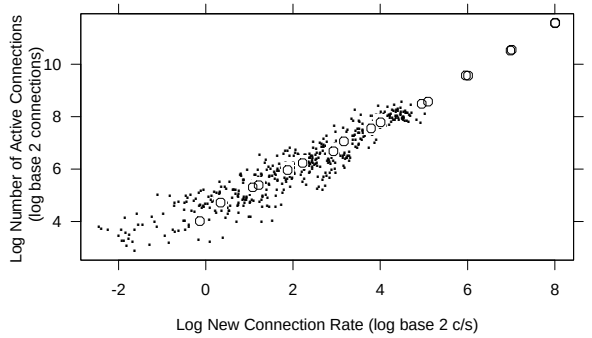

Fig. 4. Log average number of active connections is graphed against log new connection rate. Dots: live traces. Circles: synthetic traces.

c is ρ 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 ρ, 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 an interesting topic.

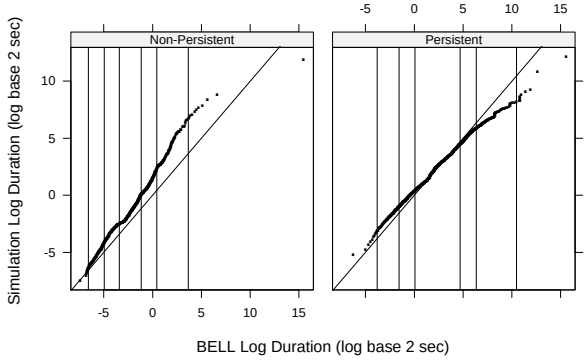

Fig. 5. Quantile-quantile plot of log connection durations. Left panel: non-persistent connections. Right panel: persistent connections.

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 [23], [6], [8]. Starting with a quite low new connection rate ρ, 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 ρ, an improving approximation. Figure 6 shows Weibull quantile plots for two traces, one live Bell trace and one synthetic, both with ρ 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 quartiles. 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 the there is a minimum inter-arrival time, the smallest packet size divided by the link speed.

Earlier we cited results that as ρ increases, the packet arrivals tend to Poisson. This means the Weibull shape parameter, λ, 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 $\log_2(\rho)$ [8]. (Note that as $\lambda \to 1$, $\text{logit}_2(\lambda) \to \infty$ and conversely.) Figure 7 graphs $\text{logit}_2(\lambda)$ against $\log_2(\rho)$ for the live and synthetic traces. For both live

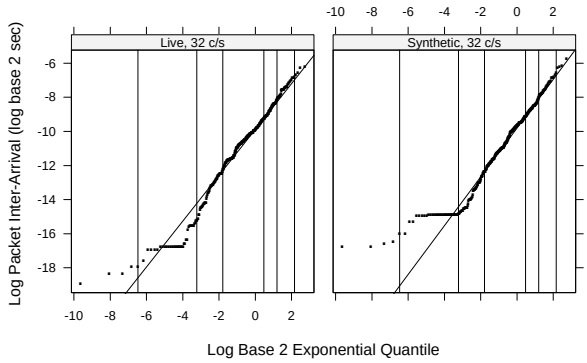

Fig. 6. Weibull quantile plot of inter-arrivals. Left panel: live trace. Right panel: synthetic trace.

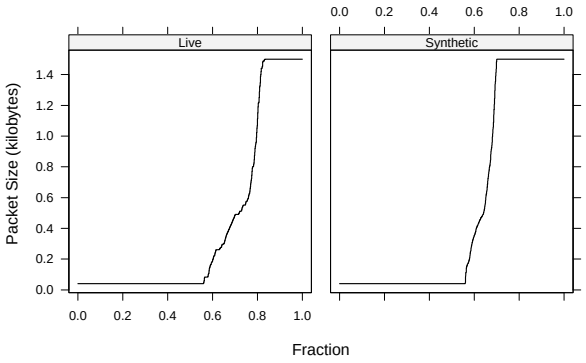

Fig. 8. Packet size quantile plot. Left panel: live trace. Right panel: synthetic trace.

and synthetic there is marked trend of λ 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.

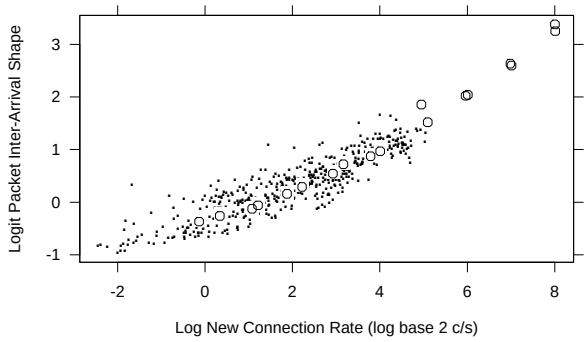

Fig. 7. The logit function of the Weibull shape is graphed against log new connection rate. Dots: live traces. Circles: synthetic traces.

The packet size distribution for live traces is a discrete-continuous distribution that does not change appreciably with ρ on a link, but does change from link to link [8]. 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.

D. Time Dependence: Inter-Arrivals and Sizes

The packet inter-arrivals and size processes of live traces are long-range dependent [24], [16], [6]. 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 ρ 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.

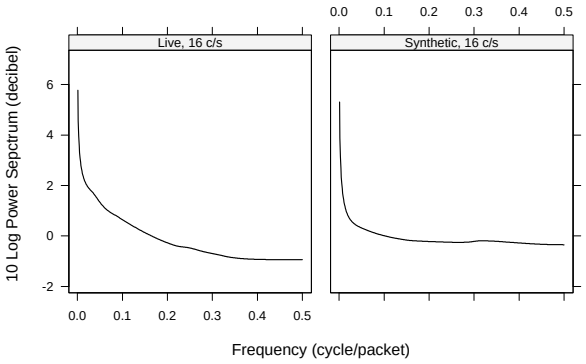

Fig. 9. Packet size spectrum is graphed against frequency. Left panel: live trace. Right panel: synthetic trace.

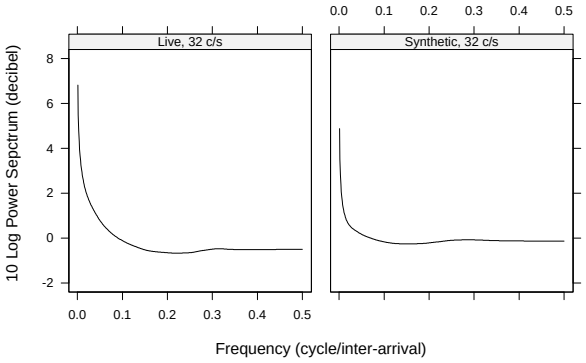

Fig. 10. Packet inter-arrival spectrum is graphed against frequency. Left panel: live trace. Right panel: synthetic trace.

As ρ 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 ρ increases, the fraction of power at low-frequencies decreases and the fraction of power at high frequencies increases. The one-step entropy τ_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 τ_1 is close to 0, the series is highly dependent in the sense that it can be predicted reliably from the past. As ρ increases, $\tau_1 \to 1$. 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 ρ [8]. 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.

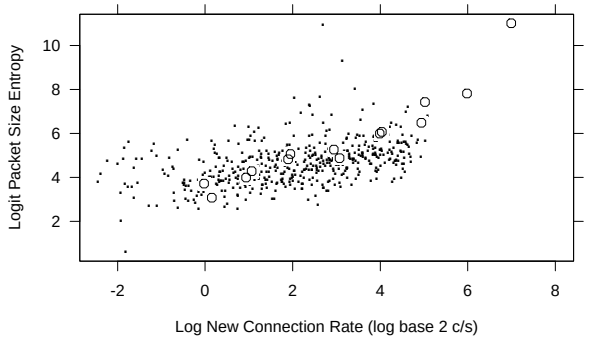

Fig. 11. Logit one-step entropy is graphed against log new connection rate for the packet sizes. Dots: live traces. Circles: synthetic traces.

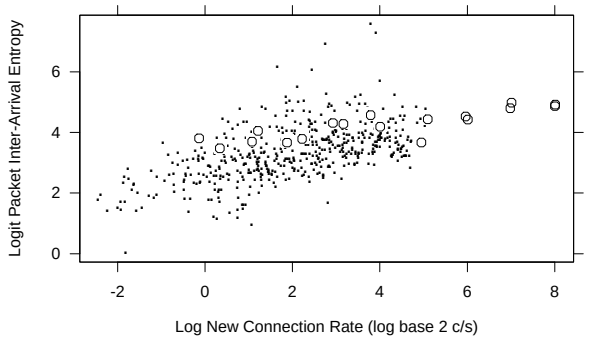

Fig. 12. Logit one-step entropy is graphed against log new connection rate for the packet inter-arrivals. Dots: live traces. Circles: synthetic traces.

APPENDIX

PackMime-NS 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-NS creates a new web client and a new web server, sets up a TCP connection between the client and server, has the client sends 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-NS 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-NS 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 [25], 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-NS 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 13. One

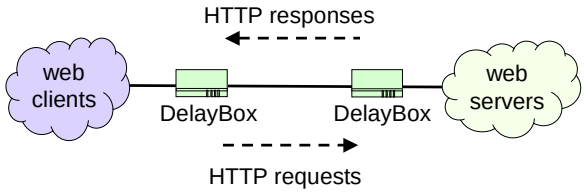

Fig. 13. PackMime-NS network with DelayBox

node is placed 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. More information on DelayBox is given at the above web site.

REFERENCES

[1] M. F. Arlitt and C. L. Williamson. Web Server Workload Characterization: The search for Invariants. *ACM SIGMETRICS*, pages 126–137, 1996.

[2] 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*, 2:15–28, 1999.

[3] P. Barford and M. Crovella. Generating Representative Web Workloads for Network and Server Performance Evaluation. *ACM SIGMETRICS*, pages 151–160, 1998.

[4] P. Barford and M. E. Crovella. A Performance Evaluation of HyperText Transfer Protocols. In *Proceedings of ACM SIGMETRICS '99*, pages 188–197, 1999.

[5] J. Cao, W. S. Cleveland, and Y. Gao. Internet Traffic: Statistical Models for Aggregate HTTP Source Variables. Technical report, Bell Labs, stat.bell-labs.com, 2003.

[6] J. Cao, W. S. Cleveland, D. Lin, and D. X. Sun. On the Nonstationarity of Internet Traffic. *ACM SIGMETRICS*, 29(1):102–112, 2001.

[7] J. Cao, W. S. Cleveland, D. Lin, and D. X. Sun. Internet Traffic Tends *Toward* Poisson and Independent as the Load Increases. In C. Holmes, D. Denison, M. Hansen, B. Yu, and B. Mallick, editors, *Nonlinear Estimation and Classification*. Springer, New York, 2002.

[8] J. Cao, W. S. Cleveland, and D. X. Sun. Fractional Sum-Difference Models for Open-Loop Generation of Internet Packet Traffic. Technical report, Bell Labs, stat.bell-labs.com, 2003.

[9] J. Cao, W. S. Cleveland, and D. X. Sun. The S-Net System for Internet Packet Streams: Strategies for Stream Analysis and System Architecture. Technical report, Bell Labs, stat.bell-labs.com, 2003.

[10] M. Christiansen, K. Jeffay, D. Ott, and F. D. Smith. Tuning RED for Web Traffic. In *Proceedings of ACM SIGCOMM 2000*, pages 139–150, 2000.

[11] W. S. Cleveland, L. Denby, and C. Liu. Random Scale Effects. Technical report, Bell Labs, stat.bell-labs.com, 2002.

[12] 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*, pages 166–177, 2000.

[13] M. E. Crovella and A. Bestavros. Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes. *ACM SIGMETRICS*, 24(1):160–169, 1996.

[14] C. R. Cunha, A. Bestavros, and M. E. Crovella. Characteristics of WWW Client-based Traces. Technical Report TR-95-010, Boston University Computer Science Department, 1995.

[15] S. Floyd and V. Paxson. Difficulties in Simulating the Internet. *IEEE/ACM Transactions on Networking*, 9(4):392–403, August 2001.

[16] J. Gao and I. Rubin. Multiplicative Multifractal Modeling of Long-Range-Dependent Network Traffic. *International Journal of Communications Systems*, 14:783–201, 2001.

[17] L. Le, J. Aikat, K. Jeffay, and F. Smith. The Effects of Active Queue Management on Web Performance. In *Proceedings of SIGCOMM 2003*, 2003. (to appear).

[18] B. Mah. An Empirical Model of HTTP Network Traffic. In *Proceedings INFOCOMM*, 1997.

[19] H. Nielsen, J. Gettys, A. Baird-Smith, E. Prud'hommeaux, H. Lie, and C. Lilley. Network Performance Effects of HTTP/1.1, CSS1, and PNG. In *Proceedings of ACM SIGCOMM '97*, pages 155–166, 1997.

[20] T. Ott, T. Lakshman, and L. Wong. SRED: Stabilized RED. In *Proceedings IEEE INFOCOM '99*, pages 1346–1355, 1999.

[21] 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.

[22] V. Paxson. Empirically-Derived Analytic Models of Wide-Area TCP Connections. *IEEE/ACM Transactions on Networking*, 2:316–336, 1994.

[23] V. Paxson and S. Floyd. Wide-Area Traffic: The Failure of Poisson Modeling. *IEEE/ACM Transactions on Networking*, 3:226–244, 1995.

[24] V. J. Ribeiro, R. H. Riedi, M. S. Crouse, and R. G. Baraniuk. Simulation of NonGaussian Long-Range-Dependent Traffic Using Wavelets. *ACM SIGMETRICS*, pages 1–12, 1999.

[25] L. Rizzo. Dummynet: A simple approach to the evaluation of network protocols. *ACM Computer Communication Review*, January 1997.

[26] F. D. Smith, F. H. Campos, K. Jeffay, and D. Ott. What TCP/IP Procotol Headers Can Tell Us About the Web. In *Proceedings of ACM SIGMETRICS*, pages 245–256, 2001.

[27] Sprint-Labs. ipmon.sprintlabs.com/packstat/packet.php?030407. Technical report, Packet Trace Analysis, 2003.