2012 IEEE 20th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems

Towards Traffic Benchmarks for Empirical Networking Research: The Role of Connection Structure in Traffic Workload Modeling

Jay Aikat[+], Shaddi Hasan[*], Kevin Jeffay[+], F. Donelson Smith[+]
University of North Carolina at Chapel Hill
Chapel Hill, NC, USA
[+]{aikat, jeffay, smithfd}@cs.unc.edu ; [*]shaddi@berkeley.edu

Abstract – Networking research would be well served by the adoption of a set of traffic benchmarks to model network applications for empirical evaluations; such benchmarks are common in many other areas of computing. While it has long been known that certain aspects of modeling traffic, such as round trip time, can dramatically affect application and network performance, there is still no agreement as to how such components should be controlled within an experiment.

In this paper we advance the discussion of standards for empirical networking research by demonstrating how certain components of network traffic, such as the structure of application data exchanges within a TCP connection, can have a larger impact on the results obtained through experimentation than other dimensions of traffic such as round-trip time. Such findings point to the pressing need for traffic benchmarks in networking research.

Through testbed experiments performed with synthetically generated network traffic from two very different traffic sources, and using several models of TCP connection structure, we demonstrate the strong effects of connection structure in traffic workload modeling on performance measures such as queue length at routers, number of active connections in the network, user response times, and connection durations.

KEYWORDS

Measurement, Workload modeling, Traffic generation, Experimental evaluations, Performance evaluations.

1. INTRODUCTION AND MOTIVATION

In their seminal paper, Floyd and Paxson [6] articulated the challenges to performing realistic and meaningful experiments. The research community responded and over time, ever more sophisticated means of performing evaluations of proposed network improvements using simulation, emulation, and live experimentation have been developed. Software simulators (*e.g.*, ns-2, ns-3, and GTNets), laboratory testbeds (e.g., Emulab), and wide-area network testbeds, e.g. GENI (Global Environment for Network Innovation) have evolved to provide capabilities to more faithfully reproduce protocol and network dynamics that occur on production networks.

Synthetic traffic is vital to performing reproducible, controlled experiments. Traffic generation itself contains several important components. It is widely accepted that emulating round trip time (RTT) is a key factor in faithfully representing traffic conditions on production networks. However, there is little attention given to emulating the application-level data exchanges, we call connection structure, within the TCP connections which dominate Internet traffic. State of the art traffic generation tools, such as Harpoon [14], Tmix [8], and Swing [16], all provide a means of turning measurements of production network links into "realistic" synthetic traffic. The traffic is realistic because it directly reproduces measurements of real traffic such as total number of flows present, total number of bytes transmitted, and emulates some model of measured RTT. However, even as these tools exist, the adoption of such realistic traffic emulation has been slow.

Leading network performance evaluation studies reveal that there is no accepted best practice for connection structure emulation in network experiments. Previous studies [8] have shown that using *per-connection RTTs* for traffic generation faithfully represents the realistic conditions captured on the production network. But how does connection structure used to generate traffic in an experiment affect the outcome of that experiment?

In this paper we make the case that structure of the synthetic traffic used in experimentation matters. By "structure" we mean the extent to which, for example, a synthetic TCP connection mirrors the pattern of application-level data exchanges of a TCP connection on some production network. While on its face, this can hardly be a surprising result (after all, how could traffic structure *not* matter?!), what we believe will be surprising for most readers is the extent to which modeling such structure *does* matter for performance studies.

Here we provide empirical data that structure matters even more than RTT. Specifically we show that for several important performance metrics, the structural properties of the TCP connections in traffic generation have a larger impact on performance than do RTTs. These data suggest that if one wants to perform experiments from which conclusions can be drawn that can be applied on real networks, the structure of the traffic used in experiments are at least as important, if not more important, than RTT. We focus on a comparison against the effects of RTT on performance simply because we feel that among the many measures of traffic that affect performance, RTT is likely the most understood and appreciated parameter.

1526-7539 2012

DOI 10.1109/MASCOTS.2012.19

In this paper, we present the results of extensive experiments comparing the effects of several synthetic traffic generation paradigms on familiar measures of network performance. We compare different connection structures to generate the same traffic input from two different sources, and compare the results for four performance metrics: queue length at routers, number of active connections in the network, user response times, and connection durations. These performance metrics are not specific to our experiments. They represent application performance at the ends, through measuring connection duration and response times, and network performance through measuring the queuing dynamics for a FIFO (first-in-first-out) queue and the number of active connections in the network.

Our results show the following. First, we confirm that RTT matters. For any given paradigm of synthetic traffic generation, and for any of our performance measures, different results are obtained when one varies the manner in which RTTs are assigned to connections. However, for a given RTT assignment scheme, the variation in performance seen by varying the connection structure exceeds that observed when varying RTTs. This demonstrates that all things being equal, connection structure can have an effect on measures of network performance at least as great as, and usually greater than RTT. Specifically, adding endpoint latencies (think times) to the connection structure significantly shifts the results for any experiment. Therefore, unless one pays attention to connection structure, it will be difficult to reach fundamental conclusions about the results of experiments.

2. TRAFFIC GENERATION

In this section we define the key workload and network characteristics that are varied in our experiments. These include the connection structure of a TCP connection, the RTT, and the receiver maximum window size.

2.1 Connection Structure

We represent the structure in modeling application data exchanges by the connection structure for the underlying TCP connection. This is modeled in two dimensions – size and time. The size dimension defines the counts of bytes transferred by the connection in both directions. The time dimension models the internal dynamics of a connection consisting of any synchronization and latencies introduced by exchanges of application-level protocol data units, typically in a request-response pattern as in a client-service model of communication. The time dimension includes all the "endpoint latencies" related to synchronization between requests and responses, the elapsed time between a request and its response ("server" latency) or between subsequent requests ("client" latency).

We represent connection structures in this study by starting with a simple model, based on Harpoon [14] for structures defined only in the size dimension. Consider an exchange between two TCP endpoints that transfers a total of ***X*** bytes in one direction and ***Y*** bytes in the opposite direction over the duration of the connection. Harpoon would use two separate connections for each original connection with a unidirectional transfer in each connection of all the bytes in a given direction in a single block. We modified this concept to use a single TCP connection for each original connection, but with two different methods of synchronizing the bidirectional data transfers. In both methods, all the bytes flowing in one direction are sent as one large block without internal gaps or latencies. In one method, the two blocks are sent concurrently in both directions while in the other method the two blocks are sent sequentially as a single request-response exchange. We call the first method the block-concurrent (blk-conc) model and the second method the block-sequential (blk-seq) model.

The two ways of representing connection structure described so far are based solely on the size component of connections. To introduce the time dimension, we turn to the representations exemplified by the Swing and Tmix traffic generators. Because we have chosen to use the Tmix traffic generation system in our research, we will adopt their terminology to explain the connection structures with both size and time dimensions. In Tmix each connection found in a trace of TCP/IP headers from a production network link is analyzed to produce a "*connection vecto*r" representation. The connection vector includes the connection's start time relative to the beginning of the trace and a descriptor of each request-response exchange found by the analysis tool. A request-response exchange (called an "*epoch*") is described by a 4-tuple consisting of the request size (called the *a* unit size), the response size (called the *b* unit size) and two latency values (called the *t* values) for the time between a request and its response and the time between successive epochs. Unidirectional transfers have only an *a* or *b* value depending on the direction of transfer. A high level summary of the Tmix analysis and generation framework is given in Figure 1.

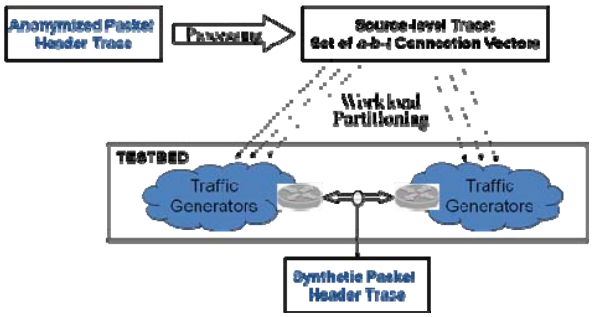

Figure 1: Traffic Generation

Using this framework, we can describe several variations for representing connection structures. First, we could retain the set of epochs representing the request-response exchanges along with the *a* and *b* values for each epoch but without any of the *t* values. This representation that we call the "*ab model*" includes the time dimension

only for the implied synchronization between a request and its response. The full representation (the "*a-t-b-t* model") adds the latency between a request and its response (server processing time) and between successive requests (client processing and user "think" times), if they exist in the original traffic.

The Tmix traffic generation system allows us to easily construct all the above described connection structures to represent the same set of TCP connections found in a link trace. This is done by using the Tmix analysis tools to generate the connection vectors containing the full representation (a-t-b-t) and then applying processing scripts to generate new connection vectors representing different structures but retaining the desired properties from the original TCP connection.

Thus, we have four distinctive cases (*blk-conc*, *blk-seq*, *ab*, and *a-t-b-t*). The *blk-conc* structure includes the total number of bytes transferred in each direction but none of the time dimensions for request-response synchronization or latencies within or between exchanges. The *blk-seq* structure is similar but models a minimal structure by sending the bytes in the two directions sequentially, like one request-response exchange. The *ab* structure explicitly represents the bytes transferred for each request-response exchange (epoch) along with the inherent synchronization between requests and responses, but no latencies within or between exchanges. The *a-t-b-t* structure is a full representation of all sizes, synchronization, and timings in the connections.

2.2 Round Trip Time Emulation

We experimented with seven different methods of emulating RTTs in our experiments. All of these have been used in previously published work. For one extreme we first tried emulating no RTT latency beyond that inherent in the laboratory network used in the experiments which is typically 1 millisecond or less (reasonable for studying local networks but obviously wrong for wide-area emulation). At the other extreme, we used the Tmix capability to emulate the specific minimum RTT found for each connection by analyzing the TCP/IP header traces.

We also experimented with emulating a single value for all connections, either the mean or median of the RTTs found by analyzing the TCP/IP header traces. Another form of RTT emulation was done by assigning a small set of values to the paths between pairs of traffic generator machines in the lab network. The network described in section 5 has a maximum of 30 paths between pairs of generator machines. In one case, we assigned a unique emulated RTT to the path between sets of three pairs (a total of 10 path RTTs). The values chosen for this case were the values recommended for the TMRG common TCP evaluation suite [2]. In a second variation, we assigned a unique RTT value for each of the 30 paths between pairs of generator machines. In this case, we used a discrete approximation to the RTT distribution [1] found from analysis of the traces (see Figure 3).

While we ran experiments with the full cross product of connection structures and RTT emulations, we report results here only for the most distinctive cases of RTT emulation: mean RTT for the traffic data emulated for every connection (labeled "meanRTT"), 10 path RTT values ("10pathRTT") and the emulation of the specific minimum RTT found for each connection by analyzing the TCP/IP header traces (called "usernet" after the name of the Tmix component that emulates per-connection RTTs).

2.3 Receiver window sizes

For all the experiments discussed in this paper, we assigned each side of every connection the maximum receiver window size exactly as was determined from analysis of the original trace. We also ran experiments where the maximum receiver window sizes were fixed for all connections as 8KB, 16KB, or 64KB. Results for the latter set are not reported here.

It is important to emphasize that all the results presented here are based on the same raw measurements (two packet header traces). The only differences are in the ***choices*** *of how we modeled the application data exchanges in the traffic and how we emulated RTT for traffic generation.*

3. TRAFFIC CHARACTERISTICS

We use two very different network traces collected at two diverse locations on the Internet in all our experiments. Both traces are one hour long. The first one, we refer to as "UNC" was taken on the campus border link of a large US university campus, connecting the campus to the Internet. The second trace was taken at an aggregation switch for four internal networks at one of IBM Corporation's large development sites. The trace includes all traffic between these networks and an external Internet service provider.

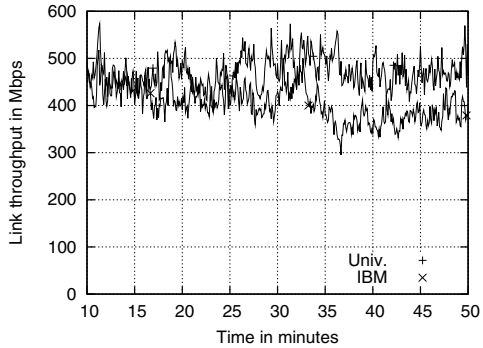

Figure 2: Link throughput / offered load

The campus trace was captured on Jan 10, 2008 from 2 to 3 PM on a weekday, which represents a very busy typical day at the university. The IBM trace was captured on Oct 10, 2006, for an hour, starting at 2:20 PM, which is also representative of typical peak workday traffic on this corporate network. We chose these two traces to demonstrate our results because of their diverse characteristics. The UNC trace has about 4.7 million

connections with an offered load of 471 Mbps in one direction and 202 Mbps in the other. The IBM trace has about 2.8 million connections with an offered load of 404 Mbps in one direction and 366 Mbps in the other. Figure 2 shows the link throughput for both traces in the high throughput direction only.

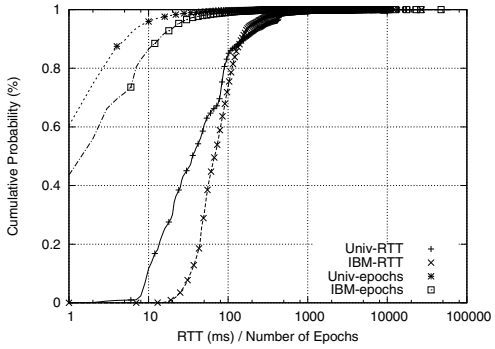

Figure 3: RTT and epochs - CDF

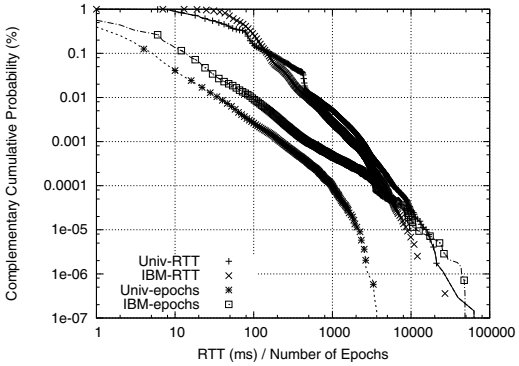

Figure 4: RTT and epochs - CCDF

The mean RTT for connections in the UNC trace was 80 ms while in the IBM trace, it was 92 ms. On average, the connections in the UNC trace used 4 epochs to transfer bytes with a standard deviation of 22. The IBM trace had a mean of 9 epochs with a standard deviation of 123.

Traffic source	Median/Mean server processing times	Median/Mean user think times
UNC	47 ms / 3.1 s	173 ms / 5.6 s
IBM	53 ms / 4.7 s	55 ms / 5.9 s

Table 1: Server Processing and User Think Times

The cumulative distribution of round trip times and number of epochs for both traces are shown in Figure 3, with their complementary cumulative distributions shown in Figure 4. The connection sizes in total bytes transferred differed significantly between the two traces as well. The UNC connections transmitted roughly 50KB on average while the mean for IBM connections was 129KB. The standard deviation for the bytes transferred was 1.3MB and 5.5MB respectively.

As shown in Table 1, there was also a significant difference in the measured server processing times and user think times between the two traces. This latency contributes directly to response times for request-response exchanges, as well as connection durations. The connection durations in turn affect the number of active connections seen in the network at any given moment.

4. EXPERIMENTAL METHODOLOGY

We conducted experiments using a laboratory network configured to emulate the real-world environments where the traces we use in the evaluations were captured – at a link connecting a large university or enterprise campus to its Internet service provider. The laboratory network used to emulate this environment is shown in Figure 5. At each edge of the network are 30 workload generator systems which are Intel-architecture machines with speeds ranging from 0.5 to 3.0 GHz running FreeBSD 6.0. The 30 machines on the left side in the figure handle generation tasks to emulate the workload generated by application processes on the university or enterprise campus while the 30 machines on the right side emulate the workloads from application processes located anywhere in the Internet.

We used the Tmix traffic generation system (that can be obtained from the contact given in [15]) on all 60 machines to create workloads based on different connection structures, and to emulate per-connection RTTs with different methods (described in section 3).

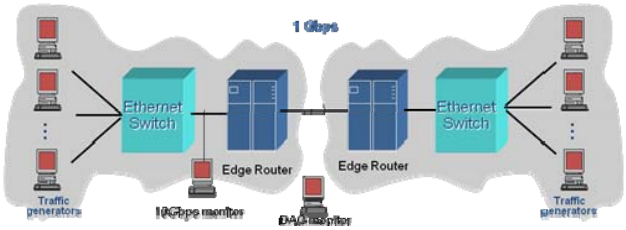

Figure 5: Experimental Network Setup

The core of this network consists of two Ethernet switches that aggregate the packet traffic from the generator machines on each edge. The two switches each have a fiber uplink to a 10 Gbps NIC in a 3.6 GHz machine that runs FreeBSD 6.0 and acts as a software router between the two sides of the network. The two router machines are connected by a 1 Gbps link. For calibrations and for experiments with an uncongested network ("*unconstrained mode*") the 1 Gbps link is used directly since its capacity is significantly greater than the load generated from either trace. For experiments with a link operating under a load that is near saturation (95% of link capacity), the dummynet bandwidth emulation function was used to constrain the link bandwidth ("*constrained mode*") to the target capacity of the router-to-router link set to 496Mbps for the university trace, and 424Mbps for the IBM trace. In all cases the router queues were set to a very large size (> 10,000 packets) which was determined to be

sufficient to avoid any packet drops at the queue loss rates were not a factor in any of the results, even in constrained mode. Measures of unbounded queue lengths have proven useful in the past for assessing issues of router buffer sizing and for understanding packet arrival processes. The Tmix capability to emulate RTTs on a per-connection basis ("*usernet*") means that the simple dumbbell topology of the lab network can have a similar effect on the dynamic time-dependent behavior of TCP connections as would be encountered in a real wide-area network.

To monitor traffic in the lab network, we used two monitoring machines with passive optical taps. One was placed on the 10 Gbps uplink between the switch and the router machine on the left side of the network. This monitor used an Intel 10 Gbps NIC with a locally modified version of tcpdump that counted packets and bytes sent from the switch to the router in one millisecond intervals. The second monitor was placed on the 1 Gbps link between the routers. Two Intel 1 Gbps NICs (one for each direction) are used to count packets and bytes in 1 millisecond intervals and to take a tcpdump of only SYN, FIN, or RST packets (to count active connections). A monitoring program run on the routers creates a log of the queue size (number of packets in the queue) sampled every 10 milliseconds. The traffic generator programs measured and recorded all the metrics related to application or user perceived performance at the end systems (connection durations, response times, etc.)

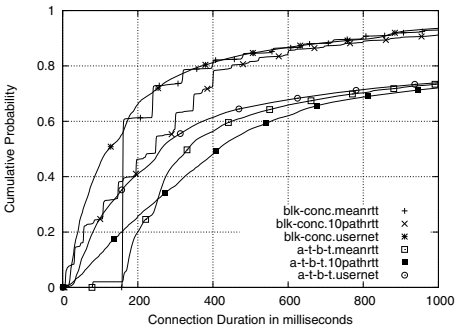

Figure 6a: Connection Duration – UNC

Although the results for only a subset of experiments are presented in the next few sections, we ran a large number of additional experiments. First, we calibrated the network. There are some critical elements of our experimental procedures that had to be validated before performing experiments. A very important step after the initial network setup is to ensure that no specific part of the network itself, other than the intentionally constrained router-to-router path, will present a resource limit for the experiments that are run. Hence, a series of experiments were run with target loads of bytes and packets similar to the final set of experiments and the CPU, memory, and NIC utilizations from all the machines were recorded to ensure that none represented a resource constraint. The experiments were then run with each experiment having exactly the same set of connections from the original trace but with their internal connection structure and RTT emulated in different ways as described in section 3. Each experiment was run for 60 minutes but data used in the results was collected only during the middle 40 minutes to eliminate startup and termination effects.

5. EFFECT OF RTT ON CONNECTION DURATION AND RESPONSE TIME

We begin with the effects of RTT on user visible measures of TCP performance, namely connection duration and response time in an unconstrained network. Connection duration is simply the time between the transmission of the first data byte of a connection and the receipt of the last data byte. Response time is the time between the transmission of the first data byte of a request and the receipt of the last data byte of its response. For block sequential, connection duration and response time are the same (since block sequential connections consist of a single "request-response" exchange). For connection structures such as a-b and a-t-b-t, connection duration is the sum of the response times of all the request-response exchanges within the connection, including any endpoint latencies (if present).

Figure 6a shows the effects of various paradigms of RTT emulation on connection duration for generation of the UNC workloads using the block concurrent and a-t-b-t

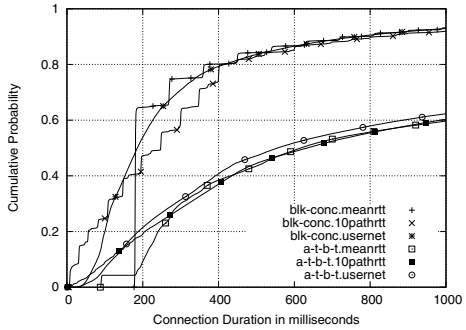

Figure 6b: Connection Duration – IBM

models for TCP connections. Focusing on the performance of the block-concurrent connections for the moment, we see that in these experiments, approximately 80% of the connections complete within 500 *ms*. For these flows, not surprisingly, RTT has a pronounced effect. For example, 50% of the connections complete in approximately 250 *ms* or less under the 10-path model of RTT emulation whereas 50% of the connections complete in approximately 125 *ms* or less under the more realistic usernet model of RTT emulation. Thus, the median connection duration is shifted by a factor of two by the choice of RTT emulation model. Figure 6a clearly shows that how one deals with RTT can have a very significant effect on a performance measure such as connection duration. An experimenter can realize a wide range of distributions of connection durations simply

by their choice of RTT emulation method. Our results (plots not included here) also show that the same is true for response times.

These experiments confirm the well-known fact that RTT has a significant impact on TCP performance and underscore the need to pay attention to proper RTT emulation when performing experiments. But how does connection structure affect performance? Looking more broadly at Figures 6a and 6b, it is clear that while the paradigm of RTT emulation used has a significant effect on the distributions of connection durations, the effects of connection structure are greater. In particular, for the last 40% of the distribution, the structure of the connections has a far greater impact on connection duration than RTT. Whereas for a given connection structure, RTT affects connection duration by a factor of two, for a given RTT emulation method, connection structure affects connection duration by up to a factor of 8 (see also Figures 7a and 7b). This effect is even more pronounced in the generation of the IBM traffic as shown in Figure 6b. Here, the durations of 80% of the connections are more influenced by connection structure than RTT.

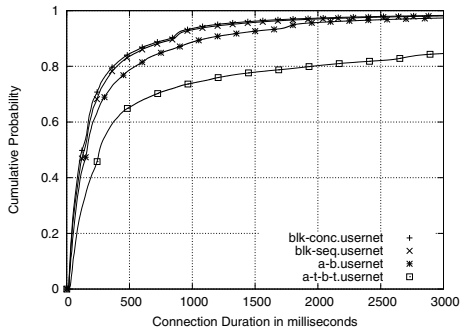

Figure 7a: Connection Duration – UNC

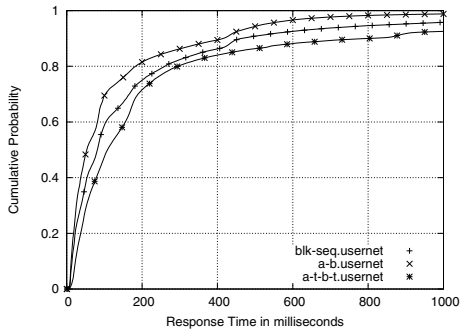

Figure 8a: Response Time – UNC

The following sections show that this result holds for other important measures of application and network performance. Henceforth, we only show results for one model of RTT emulation, namely the usernet model [8] (wherein each connection is assigned the minimum RTT that was measured for that connection). We do this because this is the most realistic method of RTT emulation – however, we have verified through extensive experimentation that the conclusions below hold for all the methods of RTT emulation we have considered.

6. EFFECT OF CONNECTION STRUCTURE ON CONNECTION DURATION AND RESPONSE TIME

Figures 7a and 7b show the full effect that connection structure has on connection duration in an unconstrained network using UNC and IBM traffic respectively. These results confirm our intuition that block concurrent connections would have the "lightest" distribution of durations (*i.e.*, that connections would have the shortest durations when generated using the block concurrent model since both endpoints of the connection transmit all their bytes at once and at the same time), and that the a-t-b-t connections would have the "heaviest" distribution of durations (*i.e.*, that connections would have the longest durations when generated using the a-t-b-t model since endpoints alternate the transmission of ADUs and incur any endpoint latencies before transmitting).

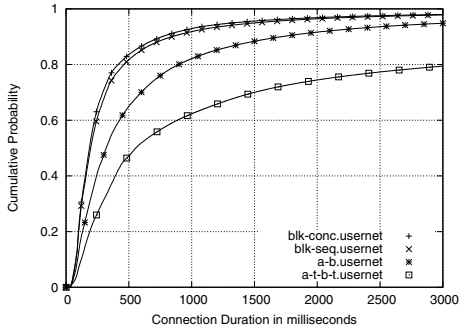

Figure 7b: Connection Duration – IBM

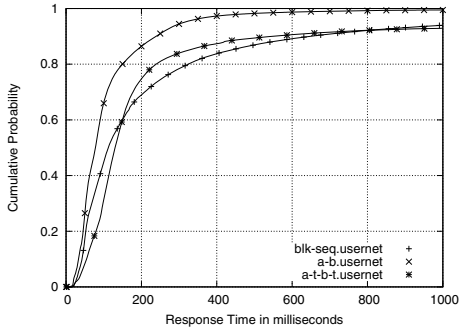

Figure 8b: Response Time – IBM

Figures 8a and 8b show the effect that connection structure has on the response time of request-response exchanges (recall that response time is only defined for the blk-seq, a-b, and a-t-b-t models of connection structure; the blk-conc model has no notion of request-response exchanges). Our intuition is generally confirmed with these results, namely that the more structure that is present in the model, the longer the response time, although the IBM data

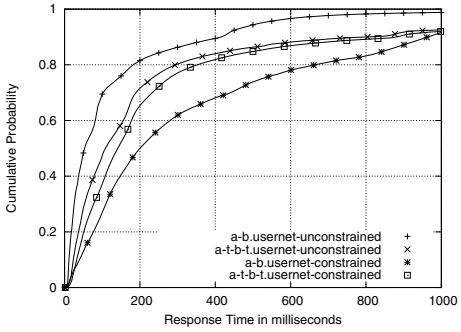

Figure 9a: Response Time – UNC

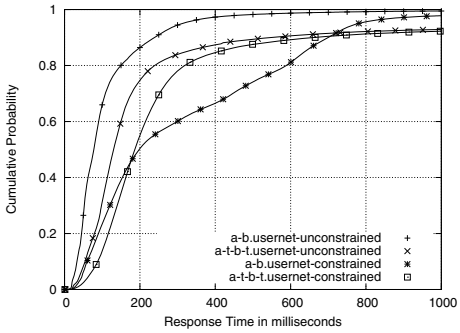

Figure 9b: Response Time – IBM

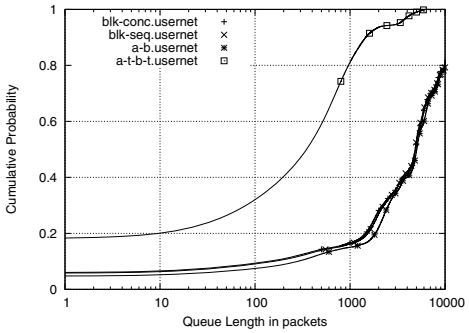

Figure 10a: Queue length – UNC

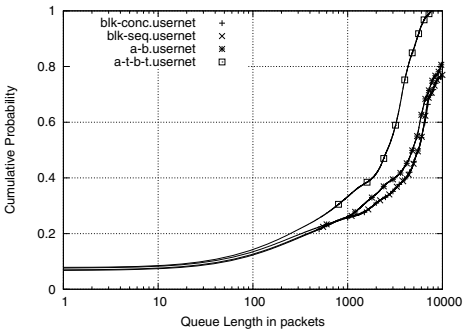

Figure 10b: Queue length – IBM

do not strictly follow this pattern. Figure 8a further emphasizes the role of structure in the UNC connections because the blk-seq connection epochs have faster response times than the a-t-b-t epochs due to the lack of server processing times in the blk-seq connections. Conversely, Figure 8b shows that for 30% of the request-response exchanges, the a-t-b-t model of connection structure generates shorter response times than the blk-seq model. That is, the presence of endpoint latencies in the a-t-b-t model, can, for a subset of connections, result in *shorter* response times for request-response exchanges. This shows that subtle interactions between flows that are a function of data sizes and latencies can exist and lead to counter intuitive results. The existence of these interactions further argues for a careful consideration of the appropriate connection structure to use when generating TCP connections.

7. Effect of Connection Structure on Network Level Performance Measures

These subtle interactions between structural elements are also apparent when considering the effects of congestion on flows. Figures 9a and 9b show the response time of a-b and a-t-b-t flows in both a constrained, bandwidth limited environment, and an unconstrained environment. On the one hand, the results for the experiments using the a-b connection structure seem obvious: response times degrade significantly (become longer) in the constrained environment. The median response time is approximately four times greater in the constrained environment than in the unconstrained environment. Surely this is the effect of congestion induced by the bandwidth constraint. But is the bandwidth constraint inherently causing the gross shift in response time or is connection structure playing a role? Figures 9a and 9b show that in fact connection structure is impacting response times more so than any bandwidth constraint. Figures 9a and 9b show that while the response times of a-t-b-t connections are also negatively affected by the bandwidth constraint, the effect is quite modest. Thus if one were to perform experiments using the a-b flows, they would possibly erroneously conclude that the bandwidth constraint would have a very large effect on the original UNC or IBM traffic when the reality is that this is not quite the case.

Of particular note is the fact that in the experiments in the constrained environment, request-response exchanges for a-t-b-t flows, exchanges that *include* endpoint latencies, have *shorter* response times than the request-response exchanges of the a-b flows (exchanges that *do not* include any endpoint latencies). That is, intuitively we might expect that request-response exchanges modeled in a-b flows should always have shorter response times than the same exchanges in a-t-b-t flows. Figures 9a and 9b show that this is not the case.

To understand this interaction, it is useful to consider the impact of connection structure on network level measures of performance such as the lengths of queues at routers. For the experiments illustrated in Figures 9a and

9b, Figures 10a and 10b show the distribution of the number of packets queued at the bottleneck router over time. As Figures 10a and 10b show, connection structure has a very significant impact on router queue length. In particular, Figures 10a and 10b show that a-t-b-t flows produce dramatically shorter queues in the bandwidth constrained experiments. Even though a-b flows and a-t-b-t flows transfer the same number of bytes according to the same pattern of request-response exchanges, because a-b flows attempt to transmit data "faster" (since the end systems in the a-b traffic generation do not pause between transmissions as end systems in the measured networks did), the a-b flows in a constrained environment actually end up transmitting data slower because they induce longer queues at the router. For this reason, the response times of a-b request response exchanges which have no endpoint latencies (other than connection RTTs), take significantly longer than a-t-b-t request response exchanges which do have them. The endpoint latencies present in the a-t-b-t model have a significant smoothing effect on the traffic arriving at the bottleneck router, allowing the router queue to drain more often.

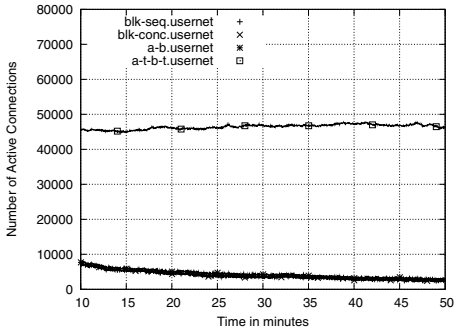

Figure 11: Active Connections – UNC

Thus, if one were to perform buffer sizing experiments for a given router, the structure of the traffic used to test any sizing policy would clearly greatly impact the results. A related measure of network performance is the number of connections active at any one time (the number of connections for which a SYN has been sent but no FIN or RST has yet been generated). Number of active connections is a measure that influences router buffer sizing as well as queuing disciplines that attempt to provide differing levels of service to flows.

Figure 11 shows a time series of the number of active connections for experiments using our four models of connection structure. Once again, the structure of connections has a dramatic effect on performance. For the UNC traffic, experiments using the a-t-b-t model of connection structure generate nearly a factor of 10 times greater number of active connections than experiments using the other models. We observed a factor of 14 times greater number of active connections for experiments with the a-t-b-t model, compared with other models, when using IBM traffic. That result is not shown here due to space limitations. We note, however, that Figures 10a and 11 combine to show that while the a-t-b-t experiments result in significantly more connections being active at any given time, these flows nonetheless generate significantly shorter queues in the constrained environment (and according to Figure 9a, result in shorter response times).

8. RELATED WORK

Most of the early work in workload generation focused on one or a limited set of application protocols such as FTP, Telnet, and SMTP [13], HTTP [4] [5] [10] [11], RealAudio [12] and other forms of multimedia [7] [9]. The obvious limitation of these approaches is that real links carry a continuously evolving mix of hundreds or thousands of different "applications" so that modeling each application or application class is clearly an approach with many difficulties and does not scale well.

Harpoon was a landmark contribution because it addressed the issue of representing a complete set of applications using both TCP and UDP transport protocols without specific knowledge of application protocols or port usage. Swing and Tmix follow this approach but depart from the Harpoon approach by using the additional information available in a packet header trace to represent the internal dynamic structure of connections that reflects application-level operations. Swing also includes characterizations of the "user" and "session" interarrivals which implicitly determine connection start times (Tmix uses the measured relative start times of connections). Harpoon, Swing and Tmix all generate traffic by read/write operations on sockets using real TCP/IP protocol stacks.

Both Harpoon and Swing use distribution-based models parameterized from analysis of empirical data that are then used with random sampling methods to generate statistically representative workloads in laboratory networks. Tmix, however, emphasizes faithful replays in the laboratory using derived details about each connection to create a replay trace that is used to initiate operations at the socket level to generate workloads. In addition to the details of request-response exchanges, Tmix can reproduce the relative start time, RTT, receiver maximum window size, and loss rate for each connection found in the original tcpdump from a production link. We chose the Tmix generation system [8] for conducting our experiments because of its capabilities for implementing all the different connection structures and RTT emulations that we use. While we chose to use Tmix, we expect similar results could be obtained with other tools, especially Swing.

Recently, there has been increased awareness and consensus among networking researchers for the need to create a common TCP evaluation suite. In [3], the authors create a case for a common evaluation standard for TCP evaluations. Their paper does not present results of experimentation, but acted as a catalyst for discussions on this topic. This led to an ongoing effort by the Transport Modeling Research Group (TMRG) in the IRTF to come up with a consensus for a baseline standard for TCP

evaluation. In [2], the authors describe the requirements for a TCP evaluation suite. They propose a benchmark consisting of a set of network configurations (that is, topologies, routing matrix, etc.), a set of workloads (that is, traffic generation rules), and a set of evaluation metrics. The benchmark would have two modes: NS simulation mode, and hardware experiment mode. The fact that Tmix is used in the TMRG TCP evaluation suite also influenced our decision to use it for this study. The TMRG effort, however, is concerned with defining the benchmark and not with research on issues about how details of benchmarks may influence the outcomes in experimental network research.

The researchers that developed the Harpoon, Swing, and Tmix workload generators reported extensive validations to show that the resulting synthetic packet-level traffic on an emulated network link was a realistic or faithful reproduction of the traffic seen on a real-world network link. To the best of our knowledge, however, ours is the first research that explores in detail the effects of using different models of application workloads and path characteristics on various metrics of network performance.

9. CONCLUSIONS

In this paper, we show the results of testbed experiments generating TCP traffic from measurement data from two very different traffic sources. Using several models of TCP application workload structure, we demonstrate the strong effects of application and connection structure on performance measures such as queue length at routers, active connections in the network, connection response times, and connection durations. We provide empirical data showing that while RTT matters for these important performance metrics, the structural properties of TCP application workloads generated have an even larger impact on performance than do RTTs. These results show that detailed measurements of network traffic are required if one is to generate credible synthetic versions of the measured traffic in experiments.

Having demonstrated that "structure matters," this work obviously begs the question of which structure, or which paradigm of synthetic traffic generation of applications is better than another and which paradigm is "best"? This is not a question we address in this work. We do not shy away from this important question but rather we believe that there is no right answer. In any simulation or emulation, one is always faced with this fundamental question: how "real" does an experiment have to be in order to be realistic? Ultimately this is a question for a research community to address through discussion and development of best practices and standards. However, such a discussion needs to be informed by data. In this paper we provide some preliminary data to further the discussion of best practices towards developing benchmarks for empirical networking research. Without a set of benchmarks for traffic workload modeling, we cannot realistically compare experimental results from different testbeds or research groups, or validate a set of published results.

REFERENCES

[1] J. Aikat, S. Hasan, K. Jeffay, F.D. Smith, *Discrete-Approximation of Measured Round Trip Time Distributions: A Model for Network Emulation*, GENI Research and Education Experiment Workshop 2012 (GREE12), March 2012.

[2] L. Andrew, S. Floyd, and G. Wang, *Common TCP Evaluation Suite*, Internet draft, July 2009.

[3] L. Andrew, C. Marcondes, S. Floyd, L. Dunn, R. Guillier, W. Gang, L. Eggert, S. Ha, and I. Rhee, *Towards a common TCP evaluation suite*, Proceedings of PFLDnet, March 2008.

[4] P. Barford and M. E. Crovella, *Generating representative web workloads for network and server performance evaluation,* Proceedings of ACM SIGMETRICS, pages 151–160, 1998.

[5] J. Cao, W.S. Cleveland, Y. Gao, K. Jeffay, F.D. Smith, and M.C. Weigle, "Stochastic Models for Generating Synthetic HTTP Source Traffic," *Proceedings of INFOCOM 2004*, pp. 1546-1557

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

[7] M. Garrett, and W. Willinger, Analysis, Modeling, and Generation of Self-Similar VBR Video Traffic, *Proc. ACM SIGCOMM '94*.

[8] F. Hernandez-Campos, K. Jeffay, and F. D. Smith, *Modeling and Generation of TCP Application Workloads*, Proceedings of the Fourth IEEE International Conference on Broadband Communications Review, September 2007.

[9] D. Heyman, and T.V. Lakshman, Source Models for VBR Broadcast Video Traffic, In *IEEE/ACM ToN*, vol. 4, no 1, pp. 37–46, Feb. 1996

[10] L. Le, J. Aikat, K. Jeffay, and F. D. Smith, *The effects of active queue management and explicit congestion notification on web performance*, IEEE/ACM Transactions on Networking, 15(6):1217–1230, December 2007.

[11] B. Mah, An Empirical Model of HTTP Network Traffic, *Proc. IEEE INFOCOM '97*

[12] Mena, and J. Heidemann, An Empirical Study of Real Audio Traffic, *Proc. IEEE INFOCOM 2000*

[13] V. Paxson. Empirically Derived Analytic Models of Wide-Area TCP Connections, *IEEE/ACM ToN*, 2 (4) 316-36, August 1994

[14] J. Summers and P. Barford, *Self-configuring network traffic generation*, Proceedings of Internet Measurement Conference, 2004.

[15] Tmix, http://netlab.cs.unc.edu/Tmix

[16] K. Vishwanath and A. Vahdat, *Swing: Realistic and responsive network traffic generation*, IEEE/ACM Transactions on Networking, August 2009.