Variability in TCP Round-trip Times

Jay Aikat Jasleen Kaur F. Donelson Smith Kevin Jeffay

Department of Computer Science
University of North Carolina at Chapel Hill
Chapel Hill, NC 27599-3175
{aikat, jasleen, smithfd, jeffay}@cs.unc.edu

ABSTRACT

We measured and analyzed the variability in round trip times (RTTs) within TCP connections using passive measurement techniques. We collected eight hours of bidirectional traces containing over 22 million TCP connections between end-points at a large university campus and almost 1 million remote locations. Of these, we used over 1 million TCP connections that yield 10 or more valid RTT samples, to examine RTT variability within a TCP connection. Our results indicate that contrary to observations in several previous studies, RTT values within a connection vary widely. Our results have implications for designing better simulation models, and understanding how round trip times affect the dynamic behavior and throughput of TCP connections.

Categories and Subject Descriptors

C.4 [**Computer Systems Organization**]: Performance of Systems – *measurement techniques, performance attributes*

General Terms

Measurement

Keywords

TCP Round-trip Times

1. INTRODUCTION

Understanding the end-to-end delay performance of Internet flows is invaluable for a variety of purposes, including evaluation of current transport protocols, provisioning and traffic engineering, construction of realistic simulation models, and design of distributed wide-area services. A number of studies have measured the round-trip delay of Internet paths, but none analyze, on a large-scale, the variability in segment round-trip times (RTTs) within TCP connections. In this paper, our objective is to answer the question: *how does round-trip delay vary within TCP connections in the Internet?*

We analyzed packet traces collected at the ISP link of the University of North Carolina at Chapel Hill (UNC). This link carries a large volume of traffic—averaging over 350 Mbps of outbound traffic and 150 Mbps of inbound traffic. We exploited our proximity to a large set of traffic end-points within UNC, and the diversity of destinations to which data is sent, to compute and analyze the RTTs of over 22 million TCP connections, covering close to 1 million distinct Internet destinations. We collected traces for a total duration of 8 hours spread out over a week, and used behavioral knowledge of the TCP protocol to compute and analyze the per-segment RTTs. Our most important finding is that variability of RTTs within a single TCP connection can be much greater than reported in previous studies. Even robust measures of variability such as the inter-quartile range of RTTs within individual connections can have values running into several seconds.

The rest of this paper is organized as follows. In Section 2, we outline our methodology for measuring round-trip delays. Section 3 presents our observations and analysis. We summarize related work in Section 4 and our conclusions in Section 5.

IMC'03, October 27–29, 2003, Miami Beach, Florida, USA.

2. MEASUREMENT METHODOLOGY

The *round-trip time* of a TCP segment is defined as the time it takes for the segment to reach the receiver and for a segment carrying the generated acknowledgment to return to the sender. We compute RTTs by passively analyzing packet traces of TCP connections—below, we describe our source for traces and our methodology for extracting valid RTT samples from traces.

2.1 The Data Source

Our trace collection was obtained by placing a network monitor on the full-duplex Gigabit link connecting the UNC campus network to the Internet via our ISP. All units of the university—including administration, academic departments, research organizations, and a major regional medical complex—use this single link for Internet connectivity. The user population is large (over 35,000) and diverse in their use of Internet applications, such as student "surfing" (and music downloading), access to research publications and data, business-to-consumer shopping, and business-to-business applications used by the university. UNC also operates a very popular set of web servers known collectively as www.ibiblio.org, www.metalab.org, and www.sunsite.unc.edu. These servers average around 350 Mbps of out-bound traffic, a significant amount of which is related to their role as a major distribution point for Linux software. In-bound traffic entering UNC averages around 150 Mbps. Note that one endpoint of each TCP connection is located at UNC, but we expect the collection of remote endpoints to cover a large sampling of Internet paths with significant geographic diversity (caused both by our users accessing remote sites and by remote users accessing the web servers mentioned above).

In: Proceedings of the 2003 ACM SIGCOM Internet Measurement Conference, Miami Beach, FL, October 2003, pages 279-284.

Category	# Destinations	# Connections	# Packets	# Bytes	# RTT samples
All Connections	$962,494$	$22,690,942$	$511\ M$	$628\ GB$	$252\ M$
Connections with ≥ 10 samples	$258,429$	$1,068,555$	$464\ M$	$581\ GB$	$236\ M$

Table 1: Trace Statistics

We placed a monitor on the Gigabit Ethernet link by passively inserting a fiber splitter to divert some light to the receive port of a Gigabit Ethernet network interface card (NIC) set in "promiscuous" mode. Because the link was full-duplex, the monitor required two NICs to monitor each direction of transmission (to and from the ISP). The NICs were installed in a 1.4 GHz IBM server-class machine (Intel architecture) with 2 GB memory, multiple PCI buses of 64bits/133MHz, and 15,000 RPM disks running FreeBSD version 4.7. An instance of the *tcpdump* program was run on each of the interfaces to collect a trace of TCP/IP packet headers. The tcpdump program reports statistics on the number of packets dropped by bpf. We found that no packets were ever dropped by bpf in all traces. Since our monitor machine is configured with sufficient CPU and bus capacity for handling a Gigabit link and the bpf buffers never overflowed, we believe that it is highly unlikely that some packets were not traced because of loss on the NICs.

We collected eight two-way traces, each for a one-hour interval. The aggregate statistics are given in Table 1. For each one-hour interval, we combined the two one-way traces using a merge sort on three keys in the following order: source IP address and port, destination IP address and port, and time-stamp (the "source" and "destination" portions of the 4-tuple were swapped in one of the traces before sorting). This produced a time ordered trace of the TCP segments within each TCP connection. We detected instances of port reuse and eliminated all connections after the first that reuse the same 4-tuple within each one-hour trace. We also detected and handled cases of sequence number wrap-around.

2.2 Extracting Valid RTTs from Traces

The basic idea for extracting RTTs from packet traces collected near TCP sources is fairly simple: *measure the time difference between the observed transmission of a data segment from the source and the observed receipt of an ACK containing an acknowledgment number that exactly corresponds to (is one greater than) the highest sequence number contained in an observed data segment.*[1] This simple notion, however, is complicated by several factors. In choosing how to deal with these, our guiding principle was to be conservative and include in our data only those RTT values where there is an unambiguous correspondence between an acknowledgment and the data segment that triggered its generation. We start with Karn's algorithm [6] and add additional tests to ensure such valid RTT estimates. While our conservative approach does eliminate some potentially valid RTT samples, we prefer to have strong confidence that all the included samples are valid.

The most serious complications arise from lost and reordered segments. If a SYN or data segment is retransmitted and an ACK matching it is received, it is ambiguous whether the RTT should be calculated from the transmission time of the initial segment or that of the retransmitted segment. Further, in a flight of data segments, the last segment may have a matching ACK but it may be generated only after the retransmission and receipt of a lost segment earlier in the flight. To eliminate the possibility of invalid (and large) RTT measures in such cases, we ignored all RTT estimates yielded by retransmitted data segments and by those transmitted between an original segment and its retransmitted copy. Another subtle complication arises because segments may occasionally be lost in the network between the sender and the tracing monitor. In this case, the retransmission of the segment will be detected as an out-of-order transmission of a sequence number, not as a duplicate transmission. We tackled such cases by ignoring all RTT estimates for data segments that were in-flight (not yet acknowledged) when an out-of-order segment was seen. Table 1 lists the total number of valid RTT samples yielded by our traces.

Testing of our processing tools was done by manually verifying their output for a number of traces. We selected three categories of TCP connections for testing. First, we selected about 50 connections that included instances of retransmissions, non-duplicate but out-of-order segments, port reuse, sequence number wrap, etc. Second, we examined *all* connections with suspicious RTT samples (*e.g.*, $> 120s$ or twice a typical *maximum segment lifetime* value).[2] Third, we examined several connections with high median RTT values (tens of seconds). Our results are surprising but valid—an example of a connection with a large median RTT can be found in [1].

Although tcpdump timestamps have a precision of $1\mu s$, they may not accurately represent the time at which the packet arrived on the link. In particular, interrupt scheduling and driver executions may introduce variable time-stamping delays. We reduce the precision of RTT values by rounding them to the nearest millisecond (RTTs $< 1ms$ are set to $1ms$).

Another issue to consider in analyzing RTT values obtained with our methodology is that a TCP endpoint may delay sending the ACK for an incoming segment for up to $500ms$ in order to piggyback the ACK on the next outgoing data segment (common implementations delay the ACK only up to $200ms$). This means that some RTT values may have additional time added because the ACK is delayed. However, TCP implementations may not have more than one delayed ACK pending at any time—for a receiver that has multiple incoming segments this leads to behavior commonly called "ACK every other segment". In this case, half of the data segments will yield valid, non-delayed RTTs and the others will not yield any RTT sample at all. Our plans for follow-on work include an assessment of the effects of delayed-ACKs on the overall population of TCP RTTs.

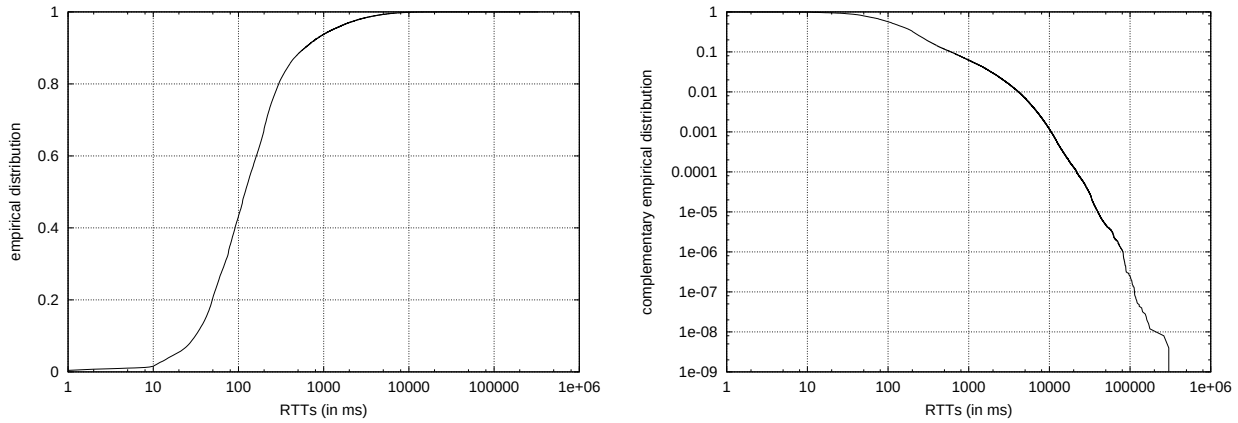
3. ANALYSIS OF ROUND-TRIP TIMES

3.1 Per-segment RTTs

Figures 1(a) and 1(b) plot the cumulative distribution (CDF) and complementary cumulative distribution (CCDF), respectively, of all the RTT samples collected from all traces. We observe the following:

[1] We also obtain an RTT sample value for the connection by measuring the elapsed time between an out-bound SYN segment and the corresponding SYN+ACK segment.

[2] Specifically, we found 9 connections that yielded such large valid RTTs; in all of these cases, we found that the ACK that returned after such a large time prompted the source host to immediately send back a connection *reset* message.



(a) CDF (b) CCDF

Figure 1: Per-segment RTT Distribution

1. The observed RTTs range from $1ms$ to more than $200s$—the minimum and maximum observed RTTs differ by more than 5 orders of magnitude.

2. A little over 40% of RTT samples are $100ms$ or less, but 50% of the samples have values between $100ms$ and $1s$. Less than 3% of RTT samples have values smaller than $10ms$ while 5% of RTT samples have values between $1s$ and $10s$.

These observations indicate that the range of RTTs experienced by TCP segments is extremely large. Some amount of diversity should be expected as segments belong to different TCP connections, each of which may traverse different network paths and end-hosts. In particular, each end-to-end path is characterized by its own minimum possible RTT, given by the sum of link propagation latencies and transmission and processing delays on all nodes in the forward and reverse direction. This quantity represents the *fixed* component of RTT experienced by each segment of a connection. Segment RTTs may have additional *variable* components due to queuing and processing delays at overloaded routers and end-hosts. It is not clear from Figure 1 if the diversity in RTT samples comes from the fixed delays associated with the $962,494$ different destinations, or from the variable delays experienced by individual connections. To better understand this issue, we next examined the variability in RTTs on a per-connection basis. Connections that see a larger number of samples are likely to yield better estimates of variability—in what follows, therefore, we consider only connections with at least 10 valid RTT samples. There are over 1 million such connections in our traces (see Table 1).

3.2 Per-connection RTTs

Figures 2(a) and 2(b) plot the distributions of the minimum, maximum, mean, median and $90th$ percentile RTTs observed for each connection. We observe the following:

1. Our traces include connections with *minimum* RTTs that vary from $1ms$ to more than $10s$ (around 10 connections have a minimum greater than $12s$). The connections exhibit great diversity in their fixed end-to-end delays. About 40% of the connections yield minimum RTTs larger than $100ms$.

2. Around 40% of connections yield a median RTT between $100ms$ and $1s$. Nearly 15% of connections yield a median RTT larger than $1s$. The distribution of median and mean RTTs is similar indicating they are roughly comparable measures for our study.

3. Around 17% of connections have a $90th$ percent RTT larger than $1s$; around 22% of connections have a max RTT larger than $1s$.

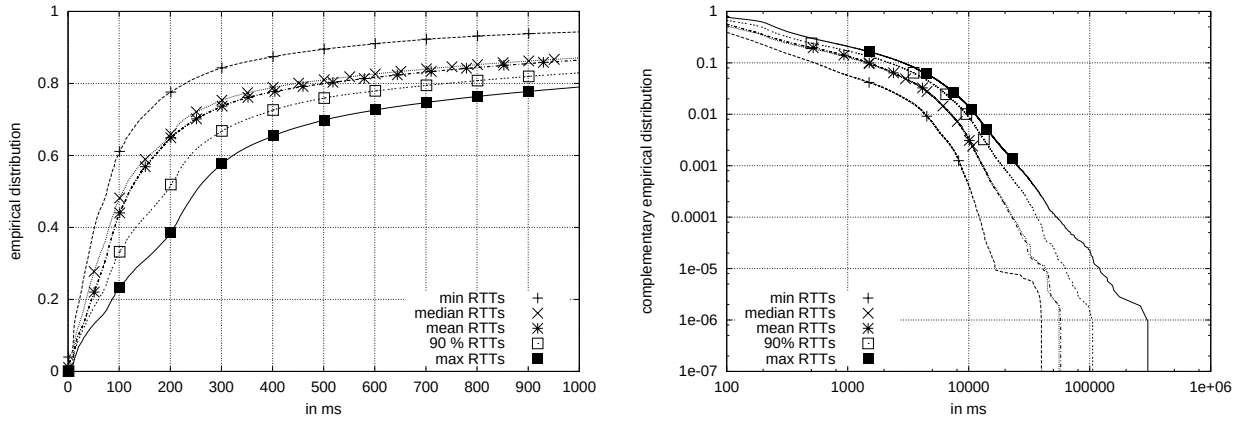
Observe that the minimum observed RTT is our best approximation of the fixed delay associated with a path (the actual delay may be less than the minimum observed RTT). In what follows, we use several measures to analyze the variable components of per-segment RTTs.

3.3 Standard Deviation of Per-connection RTTs

Our first measure of variability was the standard deviation in RTTs measured for each connection. Figure 3(a) plots the CDF of this measure. Note that approximately 50% connections have a standard deviation greater than $50ms$ and about 25% are greater than $100ms$. To focus the analysis more clearly on the variable components of RTT (estimated by the standard deviation), we next considered how variability is related to the fixed components (estimated by the minimum). We did this by first binning all connections by their minimum RTT and computing the average of all the RTT standard deviations of the individual connections in each bin. We also found the $90th$ percentile of the RTT standard deviations in each bin. These results are plotted in Figure 3(b). The interesting observation here is that RTT variability increases only slightly as the fixed (minimum) delay increases up to about $100ms$. From 100 to $1000ms$, RTT variability increases linearly with increasing minimum RTT but then increases only slightly with minimums from $1sec$ to $10sec$. Finding a physical explanation for this observation is a topic for future work.

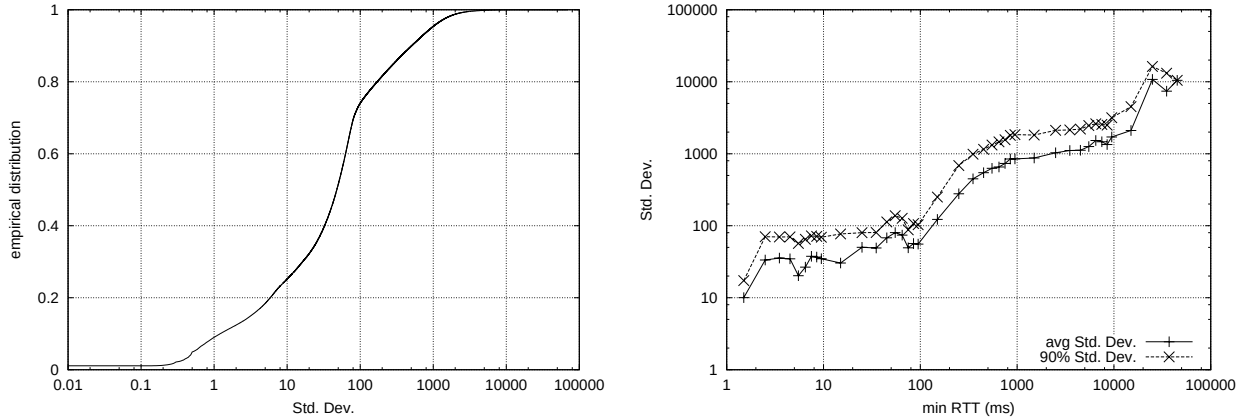
3.4 IQR of Per-connection RTTs

We next computed the *inter-quartile range* (*IQR*) of each connection's RTT distribution. The *IQR* of a distribution is defined as the difference between the 75^{th} and 25^{th} percentile values. It represents the dispersion of values in the center of a distribution (*i.e.*, after removing the smallest 25% of values and the largest 25% of values). Figure 4(a) plots the CDF of the per-connection *IQR*. We



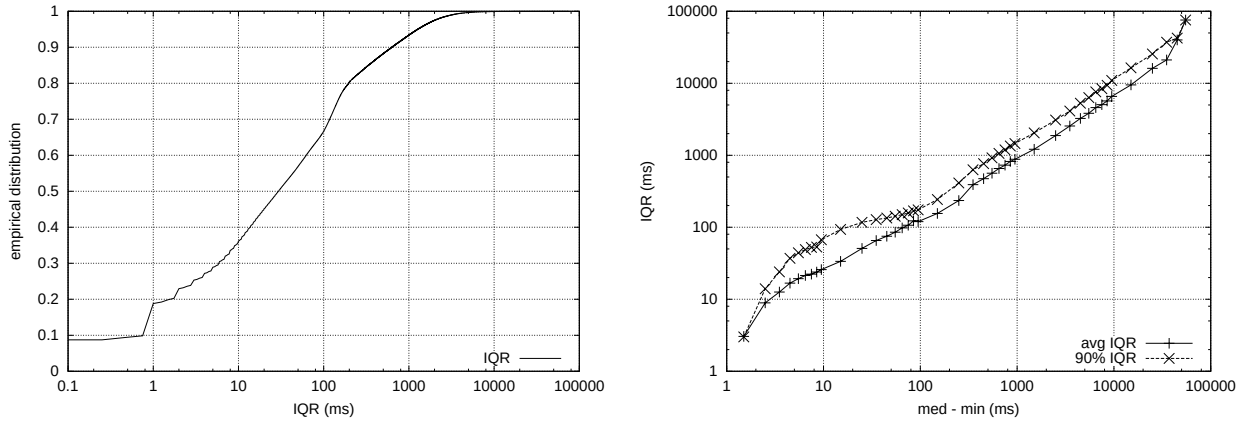
(a) CDF (b) CCDF

Figure 2: Min, Median, Mean, 90%, and Max Per-connection RTTs



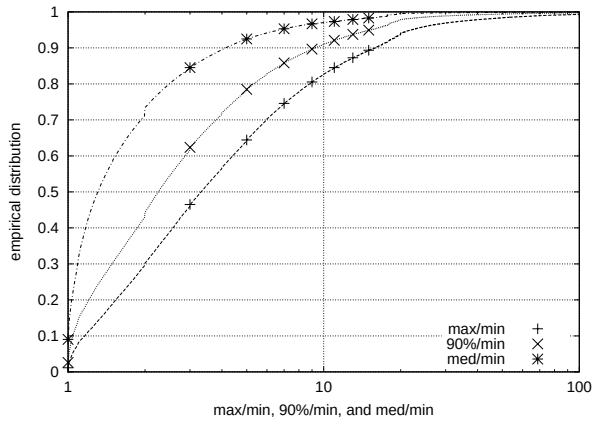
(a) CDF (b) Std. Dev. vs. min RTT

Figure 3: Per-flow Standard Deviation

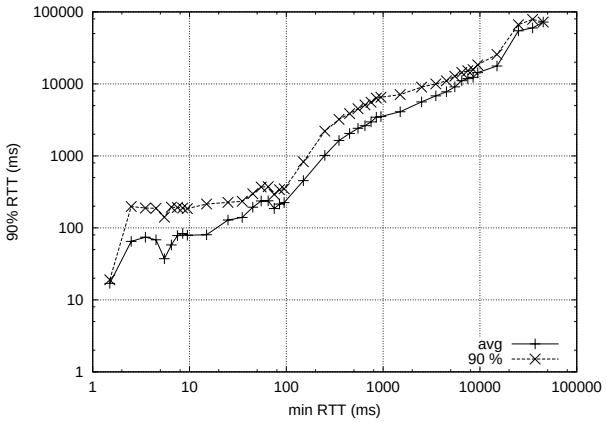


(a) *IQR* (b) *IQR* vs. (median - min)

Figure 4: Inter-Quartile Range



(a) {max,90%,med}/min



(b) 90% vs min RTT

Figure 5: Comparison of Per-connection Max, 90%, and Med RTT to Min RTT

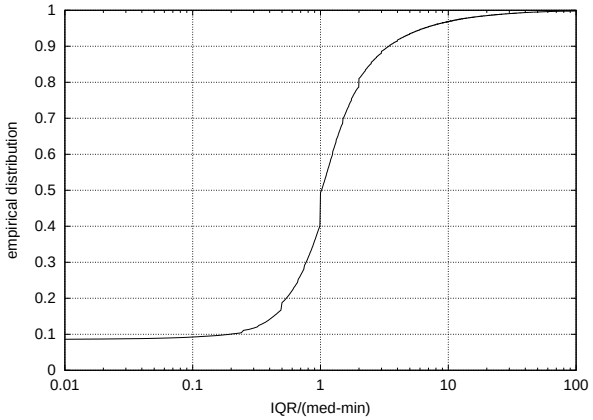


Figure 6: Distribution of IQR/(med - min)

found that the *IQR* of RTTs within a TCP connection can be as large as several seconds. A large fraction of connections, 35%, have an *IQR* of more than $100ms$.

The *IQR* represents dispersion of values in a range of percentiles equally spaced about the median ($50th$ percentile). We next examined how variability in RTT is related to the median RTT for a connection. We first linearly translated the median RTT for a connection by subtracting the minimum RTT to remove the fixed delay component. We binned all connections by their (median - min) value and computed the average of all the *IQR*s of the individual connections in each bin. We also computed the $90th$ percentile of the *IQR*s in each bin. These results are plotted in Figure 4(b). We found that the *IQR* shows a linearly increasing trend as the translated median RTT increases—connections with higher median RTTs also exhibit a larger disparity in the distribution of RTTs. For connections where the translated median is $100ms$ or greater, the average *IQR* is approximately equal to it. Further, the $90th$ percentile of the *IQR* is substantially higher than the average when the connection minimum RTT is $100ms$ or less. This suggests that connections with smaller minimum RTT see a greater variability in RTTs.

The average IQR seems to be positioned roughly symmetrically around the line $y = x$ with unit slope. We adopt this axis as our operational definition of high variability of RTTs within a connection. If $IQR > (med - min)$, we conclude the connection is experiencing "high" variability of RTTs. Figure 6 plots the CDF of the ratio $\frac{IQR}{med-min}$. We observe the following:

1. Over 50% of all connections have an *IQR* value greater than their translated median (*i.e.*, 50% of all connections experience high variability of RTTs).

2. Approximately 20% of connections have an *IQR* to translated median ratio greater than 2 (experience "extreme" RTT variability).

3. Approximately 18% of connections have a ratio less than 0.5 (experience "little" RTT variability). The remainder experience "moderate" variability.

3.5 {Median, 90th, Max} vs. Min RTT

To further assess the extent of variable delays in RTT samples within a connection we normalized the median, $90th$ percentile, and maximum RTTs observed for each connection by its minimum RTT. Figure 5(a) plots the CDF of these ratios for all connections. We find that:

1. Around 25% of connections see a median RTT that is 2–10 times the minimum RTT. Around 7% of connections see a median RTT that is more than 5 times the minimum.

2. Around 35% of connections see a maximum RTT, and 20% see a $90th$ percentile RTT, that is more than 5 times the minimum RTT.

To put these observations in a different perspective, in Figure 5(b) we treat the $90th$ percentile in the same way as we did the standard deviation in Figure 3(b)—binned connections by min RTT and computed the average and $90th$ percentile of the $90th$ percentiles. The resulting observations are similar to those for standard deviations. There are three distinct slopes of the plots for the regions below $100ms$, between $100ms$ and $1sec$ and between $1sec$ and $10sec$. The figure also indicates that connections with smaller min RTTs see a greater variability in RTTs.

The main conclusion of our study is: *the presence of significant variability in the per-segment RTTs of TCP connections*. Many previous studies and analyses have, however, used a single value to approximate per-packet RTT within a TCP connection. Often, the RTT sample yielded by the SYN/SYN+ACK pair (SYN RTT) is used as the "typical" RTT for the connection. In Figure 7, we plot the ratio of the minimum and the median RTTs to the SYN RTT of connections. We find that:

1. For almost 72% of connections, the min RTT is equal to the SYN RTT. This suggests that the SYN RTT may be used as a reasonable approximation of the min RTT. However, for 14% of the connections, the SYN RTT exceeds the min RTT by more than 10%.

2. Around 50% of connections sample a SYN RTT that differs from the median RTT by more than 10%. This indicates that the SYN RTT may not be used to approximate the "typical" RTT experienced by a connection.

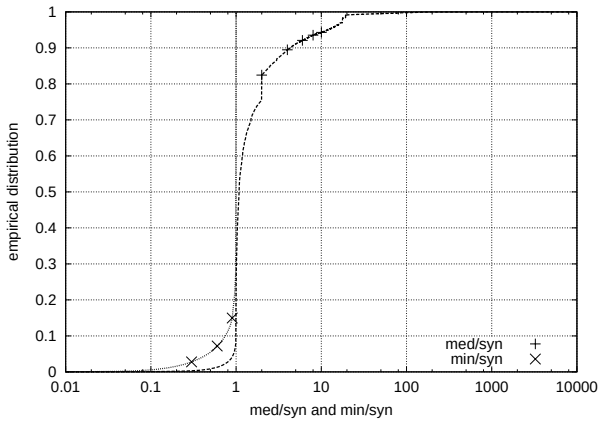


Figure 7: SYN/SYN+ACK RTTs

4. RELATED WORK

There are a few studies that report round-trip delays actually experienced by TCP connections carrying application data. One such study was conducted about 3 years ago [2]. RTTs were derived using Karn's algorithm from a long-running tcpdump trace on the network interface to a web server at NASA's Glenn Research Center. It considered about 500,000 TCP connections to around 50,000 unique endpoints. The distribution of mean RTT reported in [2] is similar to ours. However, only one variability measure was computed and analyzed.

There are a few other studies that measure and report TCP RTTs. In [5], a single per-connection RTT is computed using two techniques, one of which is close to the SYN/SYN+ACK method. In [4] RTTs of TCP connections were estimated from bidirectional traces from Sprint backbone links. These were used primarily as input to a procedure for classifying out-of-order packets. Only a CDF of average RTTs per connection are reported; the authors also demonstrate the inaccuracy of the triple handshake method in estimating the typical RTT of a connection. Other studies that estimate TCP RTTs include [3, 8] but they do not report statistical characterizations of RTT variability. The study in [7] attempts to decompose the delays between *all* data packets and their ACKs into server delays, round-trip propagation latencies, and congestion delays.

5. SUMMARY

In this paper, we study the degree of variability in TCP round-trip times by passively analyzing traces of over 1 million TCP connections between sources at a large campus and more than $250,000$ destinations. We find the presence of significant variability in the per-segment RTTs of TCP connections.

6. ACKNOWLEDGMENTS

We thank Darryl Veitch and the anonymous reviewers for their invaluable comments.

This work was supported in parts by the National Science Foundation (grants ANI-0323648, CCR-0208924, and EIA-0303590), Cisco Systems Inc., and the IBM Corporation.

7. REFERENCES

[1] http://www.cs.unc.edu/Research/dirt/RTT/trace.html/.

[2] M. Allman. A Web Server's View of the Transport Layer. *ACM Computer Communication Review*, 30(5), October 2000.

[3] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and C. Diot. Packet-Level Traffic Measurements from the Sprint IP Backbone. In *IEEE Network*, 2003.

[4] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley. Measurement and Classification of Out-of-Sequence Packets in a Tier-1 IP Backbone. In *Proceedings of IEEE INFOCOM*, April 2003.

[5] H. Jiang and C. Dovrolis. Passive Estimation of TCP Round Trip Times. *ACM Computer Communication Review*, 32(3):75–88, August 2002.

[6] P. Karn and C. Patridge. Improving Round-Trip Time Estimates in Reliable Transport Protocols. *ACM SIGCOMM Computer Communication Review*, 17(5):2–7, Oct-Nov 1987.

[7] H.S. Martin, A. McGregor, and J.G. Cleary. Analysis of Internet Delay Times. In *Proceedings of Passive and Active Measurement Workshop (PAM)*, April 2000.

[8] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker. On the Constancy of Internet Path Properties. In *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, November 2001.