Inline Path Characteristic Estimation to Improve TCP Performance in High Bandwidth-Delay Networks

Cesar Marcondes, Anders Persson, M.Y. Sanadidi, Mario Gerla
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90095, USA
cesar,anders,medy,gerla@cs.ucla.edu

Hideyuki Shimonishi, Takayuki Hama, Tutomu Murase
System Platforms Research Laboratories
NEC Corporation
Kanagawa, Japan
h-shimonishi,t-hama@cd.jp.nec.com
t-murase@ap.jp.nec.com

Abstract – *We present a measurement study of TCPW BBE (Bandwidth and Buffer Estimations) [5] and associated path characteristics estimation techniques over large pipes. Our study addresses the accuracy of our inline estimation methods used in TCPW BBE. Buffer size, capacity, and achieved rate estimation are used in TCPW BBE. We shed some light on the accuracy and speed of these estimates, and on the efficiency and friendliness resulting from using them in TCPW BBE. The results show that TCPW BBE alone is more efficient than Reno in that it provides much higher utilization of link bandwidth, while TCPW BBE is also shown to be friendly when co-existing with Reno.*

Keywords: Inline Estimations, TCP variant implementation, High-speed long distance paths, Measurements experiments

1 Introduction

There has been increased interest recently in designing transport protocols that can overcome the inefficiencies of the standard protocols over large leaky pipes; that is, over high-speed long distance paths. Most of these proposals, for example, FAST [1], BIC [2], and Westwood (TCPW) [3] [4] improve the performance in large bandwidth-delay product (BDP) networks by monitoring and estimating path characteristics such as delay, narrow link capacities, buffer size. The estimates are then used to better adjust the effective sending rate. Using such an approach, the importance of reliable path characteristics estimators cannot be stressed enough.

In this paper, we present a measurement study of TCPW BBE [5] and related path characteristic estimators [6] [7] over an actual transpacific high BDP path between Los Angeles (UCLA) and Tokyo.

2 Measurement Network Configuration

We start by describing the measurement configuration encompassing resources at Los Angeles and Tokyo, and the path between the two end points. Figure 1 presents an overview of the communications path.

For the path shown in Figure 1 we conjecture that the narrow link is on a local 100Mbps router interface to an FTTH access link that connects the receiver in NEC Tokyo to its ISP. The sender, located at UCLA, is directly connected to a core switch of the campus backbone, and thus is connected to Internet2 at 1 Gigabit/s. All other routers in the Cenic/Verio and Level3 subnets are core routers. Examining the network schematics from the respective NOCs, we observe that the path includes fast OC48/192 links within these ISP networks. In spite of the good capacity along the path, the bottleneck point provided an additional limitation, quite common inside buildings: a switching device with small buffers [8]. Such conclusion was inferred when we got *traceroute* measurements along with heavy cross-traffic, and it showed an increase in RTT only at a single hop near the Tokyo site. The increase was about 15 msec, and we roughly estimate that to be 128KB of buffer space based on the assumed link capacity. The minimal RTT observed in this topology across all experiments was 133 msec.

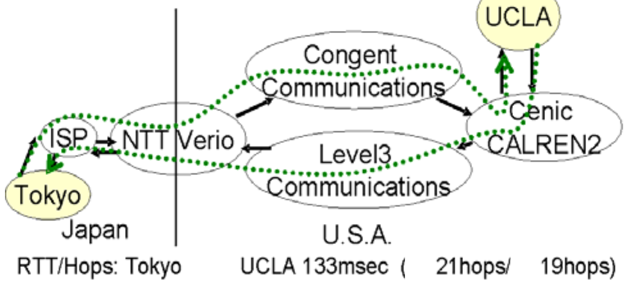

Figure 1 - Network Topology

2.1 Bottleneck Capacity

Based on an assumed narrow link capacity of 100Mbps, we launched, as part of a baseline path characterization, a number of capacity estimation experiments. We used Pathrate [6], Capprobe [7] and the "inline" measurement provided by TCPProbe [9]. Below, we provide a brief summary of each of these tools.

- *Pathrate* – based on the dispersion of packet pairs and packet trains. It uses a relatively large number of packet pair samples to reveal capacity modes. And an extra statistical Asymptotic Dispersion Rate estimate to reject wrong modes. Pathrate include statistic processing to identify modes, and select the mode corresponding to the narrow link capacity.

- Capprobe – also based on the dispersion of packet pairs, although the method uses a simple observation: a packet pair that produces over-estimation or under-estimation of the capacity must have suffered queuing delay at some link, therefore Capprobe filters out biased observations, relying only on packet pairs that have the minimal delay sum to infer accurately and fast the narrow link capacity.

- TCPProbe – based on an ongoing TCP flow, it uses innovative passive measurement by flipping pairs of packets at the top of the transmission queue. This flipping mechanism is strictly required since modern TCP receivers usually reply using delayed ACKs, destroying the possibility of obtaining an ACK pair reflecting the forward probing packet dispersion. Upon the reception of both flipped ACKs, it applies the CapProbe algorithm to estimate the capacity accurately. This method is non-intrusive because normal data packets are used as probes, and only a few additional ACKs will be generated.

The results from the three tools substantially matched each other, estimating an average capacity slightly less than 80Mbps. The tools varied, however, in the length each of them took to converge to its estimate. The widths of the rectangles in Figure 2 represent the time to converge to an estimate. Pathrate, since it relies on more significant statistical processing of gathered data, took the longest: roughly 70 sec. Capprobe/TCPProbe were faster since they rely on the minimum delay sum identification and no post processing of statistical data. CapProbe/TCPProbe converged in approximately 6 and 12 seconds, respectively, on average.

In the case of TCPProbe, we observed that whenever the link was highly congested, there was a higher deviation from the mean of about 20 Mbps. In addition, TCPProbe takes longer to converge than CapProbe since it must wait for packets accumulation in the TCP queue so that packet flipping becomes feasible, and the packet pair is sent back to back. CapProbe, on the other hand freely sends its packet pair probes, and converges faster as a result. We are investigating methods to speed up TCPProbe since it is the least intrusive of the methods analyzed above.

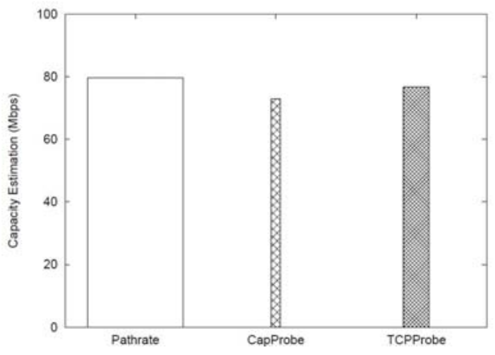

Figure 2 - Comparing Capacity Estimations

All capacity measurement tools pointed to an 80 Mbps narrow link capacity, which is less than the 100Mbps interface capacity. To further investigate the difference we conducted an additional experiment using a group of 10 TCP connections to fill up the bottleneck. In this case, we observed an aggregate throughput saturated at 64.9 Mbps, which is a reasonable utilization of the estimated 80 Mbps narrow link capacity. Thus, we conclude that the capacity estimation tools are fairly accurate and there should be a bottleneck of this capacity within one hop from the Tokyo receiver.

3 In-line Loss Discrimination and Buffer Estimation

In this section, we present an evaluation of loss discrimination and buffer size estimation. The loss discrimination is used for improved friendliness to NewReno by behaving closer to NewReno when a loss is recognized as a congestion loss.

3.1 Loss discrimination through buffer estimation

The key idea in buffer estimation is to correlate the loss of a packet with the RTT observed just before the loss. Such reasoning has been explored in different contexts independently of our research (e.g., [10]) but the alignment with TCP data stream is a novelty introduced in BBE.

To begin, we first ran TCP NewReno flows, and gathered RTT values determined at the sender preceding a packet loss. We plot in Figure 3 the relative frequencies of RTT values for one of 10 competing flows. The data was gathered over a fairly long time (900 sec). In this figure we plot two distributions, one for all RTT values obtained, and the other for RTT values immediately preceding a packet loss. The distributions show a minimum RTT of 133 msec. On the other hand, packet losses occur mostly when RTT is close to 148 msec. Therefore, this figure indicates that, when the bottleneck link is congested, packet loss is predominantly due to buffer overflows, and we can estimate the buffer delay at congestion loss to be the difference between the two peaks in the distributions. Thus we infer a maximum buffer delay of 15 msec. And at a capacity of 80 Mbps, this translates to a buffer size of 150 Kbytes.

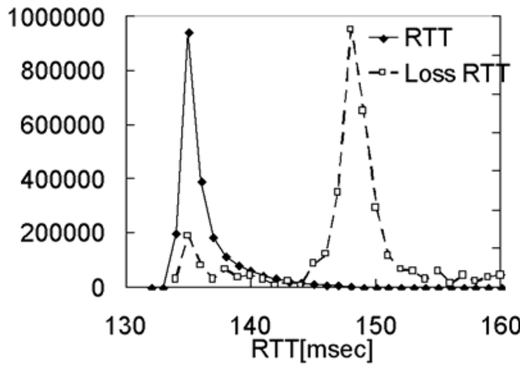

Figure 3 - RTT Discrimination (Histogram Multiple flows)

In this experiment, a small fraction of the packet losses (Fig. 3) have an RTT smaller than 147msec. These losses could have happened due to non-congestion events. Based on the effective rate of each connection, we estimate the non-congestion packet loss rate to be possibly 0.005%. This packet loss rate has a small impact on the throughput of these TCP flows, since they obtain in total 64.9Mbps, roughly 86% of the measured bottleneck link capacity. However, as we will see, the behavior is much different when the narrow link is not congested, as is the case of a single NewReno flow over the LA-Tokyo path.

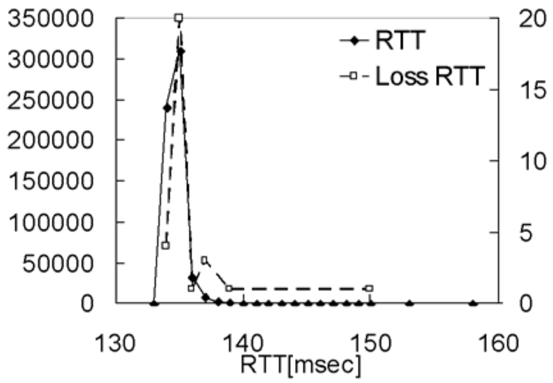

Figure 4 – RTT from A Single Flow Experiment

In Figure 4, we show the relative frequency of RTT when only one TCP-Reno flow is applied to the path. At the time of this experiment, we also ran *pathload* (an accurate residual capacity measurement tool [11]). We estimated the residual capacity at the time of the experiment to range between 57-80 Mbps. The throughput of the single NewReno flow was just 15.4Mbps. Note that in this case, there are no peaks in the distribution of RTT before loss, and such distribution overlaps with the overall RTT distribution. Assuming that congestion losses happen when RTT is around 147msec from the previous experiments, most of the losses appear to be random losses with loss rate estimated to be 0.005 %. This packet loss rate is almost the same as the previous experiment. We performed additional single CBR UDP flow experiments varying the sending rate, from 1Mbps-10Mbps, as well as single Poisson UDP flow with rate varying from 1Mbps-10Mbps. We measured a loss rate of approximately 0.001% of errors at low loads in this path.

The explanation of such non-congestion losses is the subject of further investigation. Potential and preliminary explanations for such errors include CRC/TCP checksum failures [12], duplex mismatch failures [13], and possibly others. In addition, it is well known that network paths can reorder segments resulting in TCP degraded performance [14]. A simulation validation was conducted using a uniformly distributed random error model at the level measured and it revealed similar throughput and RTT behaviors.

3.2 In-line Bottleneck Link Capacity Measurement

Originally, TCP Westwood estimations came from the ACK feedback loop. From the ACK flow one could infer two main estimates, (1) the flow recently achieved rate, roughly equal to cwin/rtt, called Rate Estimation (RE), and obtained from "packet train" measurements, (2) an admittedly noisy estimate of the bottleneck capacity called Bandwidth Estimation (BE), obtained from packet pairs measurement [6]. The next figure shows these estimates in a time series format. We also include the TCP Probe estimate for comparison.

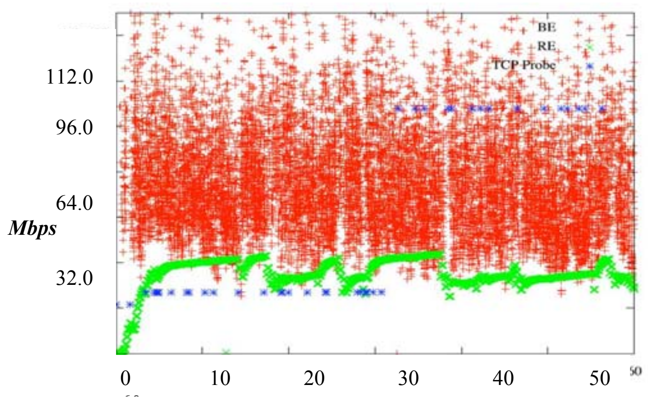

Figure 5 - Rate, Bandwidth and Capacity Estimation

In the figure above, we note that BE estimates fluctuate excessively. This can be fixed by adjusting the exponential averaging parameters. In this case, we also ran *pathload* prior to this execution, and we obtained a residual capacity estimate range of 30-40 Mbps. The rate achieved by the TCP BBE connection was around 32 Mbps. In comparison to the BE and RE estimates, TCPProbe (in blue), vary little over time, except for the jump to the actual capacity of the path after 30 seconds.

As stated previously, this is due to TCPProbe determination of the path capacity by looking for a minimum delay sum among packet pairs. In this experiment, the final min delay sum took approximately 30 seconds to be identified [9].

The current implementation of TCPProbe requires two packets back to back in the transmit queue for use as a packet pair probe, which is why the frequency of probing is so much lower than BE, which obtains an estimation sample every packet. The result is TCPProbe slower convergence. We are investigating methods to speed up the convergence of TCPProbe.

4 TCPW/BBE Performance Measurements

Above, we studied some aspects of accuracy and speed of various estimation techniques used in TCPW BBE. In this section we show how the combination of the methods above performs in the context of BBE. In BBE, a sender calculates its eligible rate (the rate which the TCP connection should be eligible to send) using a linear combination of BE and RE. The linear combination results in an Eligible Rate Estimate (ERE) that is close to BE after non-congestion losses, but then quickly fall back to RE as the congestion level increases.

This combination has been successfully used on latest versions of TCP Westwood [3] [4] to improve efficiency; however, it's the buffer inference that improves friendliness of TCP Westwood towards TCP NewReno on arbitrary buffers scenarios.

The RTT as observed by a BBE flow is presented in Figure 6 below. We can see from the figure that there is a clear separation between "normal" RTT and what is observed prior to a packet loss, thus allowing BBE to differentiate between congestion and non-congestion losses. Thus, we define RTT_{cong}, the value that RTT is expected to take when a packet is lost due to congestion.

Given that packet losses are mainly due to congestion, RTT_{cong} is estimated using RTT values measured right before the losses. Namely, RTT_{cong} is updated upon each packet loss event as the following equation (also the estimation is further shielded against errors using an adaptive moving average filter):

$$RTT_{cong}^{j} = (1-a)RTT_{cong}^{j-1} + aRTT^{j}$$

where index j express j-th packet loss event and a is an exponential smoothing factor.

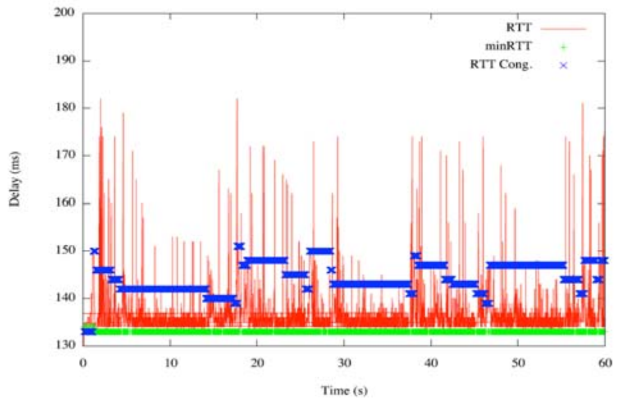

Figure 6 - BBE RTT upon loss Estimations

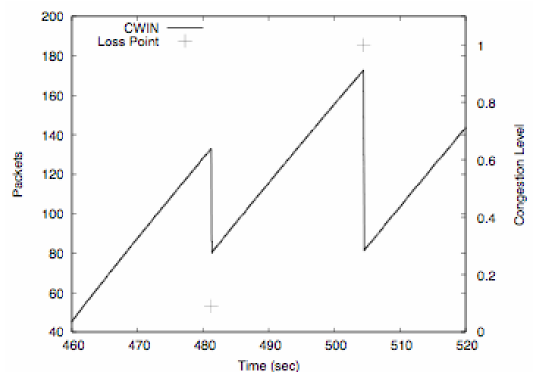

Figure 7 - Congestion Window Behavior on BBE

When current RTT is close to RTT_{cong}, the network is recognized as congested and a packet loss occurring under such a condition is viewed as a congestion loss. On the other hand, when RTT is close to its minimum value, the network is recognized as underutilized. Thus, we define congestion level *c* $(0 \le c \le 1)$ as follows:

$$c = \min\left(\frac{RTT - RTT_{\min}}{RTT_{cong} - RTT_{\min}},\ 1 \right)$$

In Figure 7, we can see the behavior of BBE as it is reacting to losses. After 480 sec a loss is detected, but the congestion level is low, resulting in a minor cut in the congestion window due. This can be compared to the loss occurring after 510 sec, which has a high congestion level, and thus the window reaction is similar to NewReno, i.e., halving the congestion window.

4.1 Link Utilization

In order to evaluate how efficiently BBE and NewReno utilize the link we ran each protocol by itself for 100 seconds. The available bandwidth measured at the time of the experiments by Pathload was 61.32 - 74.60 Mbps, indicating low load. The experiment was repeated multiple times on similar conditions and the average throughput and standard deviation was calculated. The results shown in

Figure 8, demonstrate that BBE has a clear advantage in this environment, since it does not suffer as badly as Reno when losses are not related to congestion.

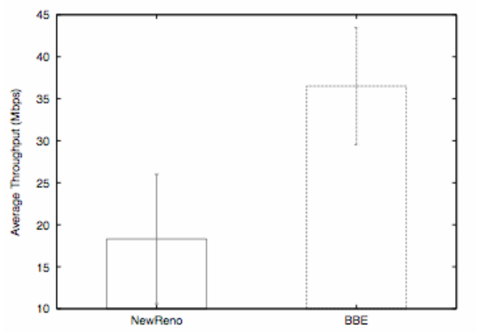

Figure 8 - Average Link Utilization of BBE and New Reno

4.2 Friendliness of TCPW/BBE

In spite of the better utilization in a single flow case, the main goal of BBE is to increase efficiency, while still remaining friendly to NewReno. To present this feature, we tested the protocol during a period of the day when the link was heavily loaded (Pathload estimated 30Mbps available). Therefore, in this scenario much more cross-traffic was in place than the previous experiment leading to more congestion errors. Excellent friendliness can be seen from Figure 9, when running BBE and NewReno over a very lengthy run of 20 minutes. In order to make the throughput measurement clear in this figure, we plot the cumulative throughput throughout the experiment.

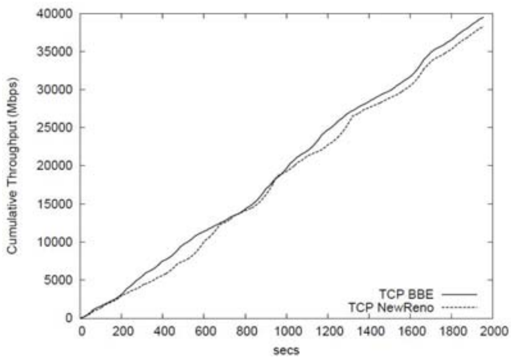

Figure 9 - Cumulative Throughput of BBE and NewReno

5 Conclusions

In this paper we presented TCPW BBE measurements results, and studied some accuracy and speed aspects of its estimation mechanisms, such as buffer capacity estimation schemes. BBE was shown to improve the efficiency in a large BDP path between LA and Japan, while still remaining friendly to NewReno. We showed that TCPW BBE by itself improved the throughput by nearly 70% in a real trans-pacific network, and that it shared the bandwidth fairly when coexisting with NewReno. The improvement was due to correct differentiation of packet losses in the path, indicating that TCPW BBE is indeed a robust option. Regarding buffer estimation, we found that buffer estimation is most accurate when the narrow link is congested, which is when accuracy matters. Finally, we believe that a combination of BE and TCPProbe estimates can be used to provide accurate as well as fast capacity estimation, a subject that we will explore in the near future.

References

[1] Cheng Jin, David X. Wei and Steven H. Low, "FAST TCP: motivation, architecture, algorithms, performance", IEEE Infocom, Hong Kong, March 2004.

[2] Lisong Xu, Khaled Harfoush, and Injong Rhee, "Binary Increase Congestion Control for Fast Long-Distance Networks", INFOCOM 2004.

[3] M. Gerla, B. K. F. Ng, M. Y. Sanadidi, M. Valla, and R. Wang, "TCP westwood with adaptive bandwidth estimation to improve efficiency/friendliness tradeoffs", Computer Communications, vol. 27, no. 1, pp. 41–58, 2004.

[4] Ren Wang, Kenshin Yamada, M. Yahya Sanadidi, and Mario Gerla " TCP with sender-side intelligence to handle dynamic, large, leaky pipes ", IEEE Journal on Selected Areas in Communications, 23(2):235-248, 2005.

[5] Hideyuki Shimonishi, Medy Sanadidi, Mario Gerla. "Improving Efficiency-Friendliness Tradeoffs of TCP: Robustness to Router Buffer Capacity Variations". ICC 2003.

[6] Constantinos Dovrolis, Parameswaran Ramanathan and David Moore. "What Do Packet Dispersion Techniques Measure?", INFOCOM 2001.

[7] R. Kapoor, L.-J. Chen, L. Lao, M. Gerla, and M. Y. Sanadidi, "Capprobe: a simple and accurate capacity estimation technique," SIGCOMM Comput. Commun. Rev., vol. 34, no. 4, pp. 67–78, 2004.

[8] David X. Wei, Sanjay Hedge and Steven H. Low. "A Burstiness Control for High Speed Long Distance TCP". PFLDNet 2005 – Third International Workshop on Protocols for Fast Long Distance Network. Lyon, France, Feb. 2005.

[9] C.A.C. Marcondes, A. Persson, Ling-Jyh Chen, M. Y. Sanadidi, and M. Gerla. "TCP Probe: A TCP with built-in Path Capacity Estimation". In: The 8th IEEE Global Internet Symposium (in conjunction with IEEE Infocom.05), Miami, USA, 2005.

[10] Mark Claypool, Robert Kinicki, Mingzhe Li, James Nichols, and Huahui Wu. "Inferring Queue Sizes in Access Networks by Active Measurement". Technical Report WPI-CS-TR-04-04, CS Department, Worcester Polytechnic Institute, February 2004.

[11] M. Jain and C. Dovrolis. "Pathload: A measurement tool for end-to-end available bandwidth". In Proceedings of Passive and Active Measurements (PAM) Workshop, Mar. 2002.

[12] Jonathan Stone and Craig Partridge. "When the CRC and TCP checksum disagree". SIGCOMM '00: Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication. 2000.

[13] Stanislav Shalunov and Richard Carlson. "Detecting Duplex Mismatch on Ethernet". Passive and Active Network Measurement, 6th International Workshop, PAM 2005, Boston, MA, USA, March 31 - April 1, 2005.

[14] Sumitha Bhandarkar, A. L. Narasimha Reddy, Mark Allman and Ethan Blanton. "Improving the Robustness of TCP to Non-Congestion Events". IETF Internet-Draft Expires May 2006.