Available Bandwidth Measurement via TCP Connection

Cao Le Thanh Man, Go Hasegawa and Masayuki Murata

Graduate School of Information Science and Technology, Osaka University
1-3 Yamadagaoka, Suita, Osaka 560-0871, Japan
{mlt-cao, hasegawa, murata}@ist.osaka-u.ac.jp

Abstract. We introduce a novel mechanism for actively measuring available bandwidth along a network path. Instead of adding probe traffic to the network, the new mechanism exploits data packets transmitted in a TCP connection (*inline measurement*). We call a modified version of TCP that incorporates the proposed mechanism *ImTCP* (Inline measurement TCP). ImTCP is based on Reno TCP, with a modification to the sender program only. The ImTCP sender adjusts the transmission intervals of data packets, then estimates available bandwidth of the network path between sender and receiver utilizing the arrival intervals of ACK packets. Simulations show that the new measurement mechanism does not degrade TCP data transmission performance, has no effect on surrounding traffic and yields acceptable measurement results in intervals as short as 1-4 RTTs (round-trip times).

1 Introduction

Information concerning bandwidth availability in a network path plays an important role in adaptive control of the network. Network transport protocols can use such information to optimize link utilization [1] or improve transmission performance [2]. Service overlay networks in particular need fast and accurate information on available bandwidth for optimal route selection [3]. Available bandwidth information is also used in network topology design and is a key factor in network troubleshooting when isolating fault locations [4].

Available bandwidth can be measured at routers within a network [5]. This approach may require a considerable change to network hardware and is suitable for network administrators only. Some *passive* measurement tools can collect traffic information at some end hosts for performance measurements [6], but this approach requires a relatively long time for data collection and bandwidth estimation. Exchanging probe traffic between two end hosts to find the available bandwidth along a path (an *active* measurement) seems the more realistic approach and has attracted much recent research [7-12].

Sending extra traffic into the network is the common weakness in all active available bandwidth measurement tools. Depending on the algorithm used, the amount of required probe traffic differs. According to one study [12], Pathload [8] generated between 2.5 to 10 MB of probe traffic per measurement. Newer tools

have succeeded in reducing this. The average per-measurement probe traffic generated by IGI [11] is 130 KB and by Spruce [12] is 300 KB. A few KB of probe traffic for a single measurement is a negligible load on the network. But for routing in overlay networks, or adaptive control in transmission protocols, these measurements may be repeated continuously and simultaneously from numerous end hosts. In such cases, the few KB of per-measurement probes will create a large amount of traffic that may damage other data transmission in the network as well as degrade the measurement itself.

We propose an active measurement method that does not add probe traffic to the network, with the idea of "plugging" the new measurement mechanism into an active TCP connection (*inline measurement*). That is, data packets and ACK packets of an TCP connection are utilized for the measurement, instead of probe packets. This method has the advantage of requiring no extra traffic to be sent to the network.

The idea of inline measurement has previously appeared in traditional TCP. To some extent, traditional TCP can be considered a tool for measuring available bandwidth because of its ability to adjust the congestion window size to achieve a transmission rate appropriate to the available bandwidth. One version of TCP, TCP Vegas [13], also measures the packet transmission delay. There are, in addition, other tools that convert the TCP data transmission stack into network measurement tools; Sting [14] (measuring packet loss) and Sprobe [15] (measuring capacity of the bottleneck link) are typical examples.

As for the measurement of available bandwidth in an active TCP connection, there is some related research. Bandwidth estimation in traditional TCP (Reno TCP) is insufficient and inaccurate because it is a measure of *used* bandwidth, not *available* bandwidth. Especially in networks where the packet loss probability is relatively high, TCP tends to fail at estimating available bandwidth. The first measurement algorithm applied in TCP used a passive method in which the sender checks ACK arrival intervals to infer available bandwidth [16]. It is a simple approach based on the Cprobe [7] algorithm, but does not yield good results [17]. A similar technique is used in TCP Westwood [18] where the sender also passively observes ACK packet arrival intervals to estimate bandwidth, but the results are more accurate due to a robust calculation. However, because these methods observe only ACK arrival intervals, changes in available bandwidth cannot be detected quickly.

We deploy an active method for inline measurement. When the sender TCP sends data packets, it also adjusts the packet transmission intervals, just as active measurement tools do with probe packets. When the corresponding ACK packets return, they are considered to be the echoed packets of probe traffic, such as the ICMP packets of Cprobe [7]. The sender then utilizes the arrival interval of these packets to calculate the available bandwidth. The sender thus collects more information for a measurement and improved accuracy can be expected.

We previously introduced a measurement algorithm suitable for inline network measurement [19] that generates periodic measurement results at short intervals, on the order of several RTTs. The key idea in measuring rapidly is

to limit the bandwidth measurement range using statistical information from previous measurement results.

In this paper, we introduce ImTCP (Inline measurement TCP), a Reno-based TCP that includes the proposed algorithm for inline network measurement. We introduce a packet store-and-forward system which allows ImTCP sender to regulate the packet transmission time according to the measurement algorithm. We then explain how to set important parameter to minimize the transmission delay caused by the packet store-and-forward process. We evaluate the inline measurement system using simulation experiments. The results show that the proposed algorithm works with the window-based congestion control algorithm of TCP without degrading transmission throughput.

2 Algorithm for Inline Measurement

We examined the current active measurement methods and found that none can be used for TCP measurement of available bandwidth. These current tools can be classified as *probe rate* type or *probe gap* type according to how they convert information from the probe packet into a value of interest. Tools based on probe rate, such as Cprobe, PathLoad, pathChirp [9] and netest [10] utilize changes in the transmission rate of a group of packets to infer an available bandwidth value. Such tools, except for Cprobe, provide good measurement results but require many packets for one measurement. Because the TCP window size limits the number of packets available for transmission at any one time, we need a measurement algorithm using a smaller number of packets. Cprobe can yield a result after only one group of probe packets, but transmits probe packets at the highest rate possible by the sender host so it impacts other traffic in the network if repeated continuously. Tools based on probe gap calculate available bandwidth from the change in time gaps between successive probe packets. IGI/PTR and Spruce are examples. This type of tool requires a smaller amount of probe traffic because the calculation depends strongly on the time gaps of only some probe packets. Such tools are weak if deployed in a TCP connection because the arrival intervals of some ACK packets may not reflect the available bandwidth very well, due to delays at the receiver or a difference in the size of data and ACK packets. The calculation may then be based on inaccurate data leading to extremely poor results.

Here, we summarize the measurement algorithm proposed for use in ImTCP. The proposed algorithm is based on the probe rate method and a technique by which to reduce probe traffic. The algorithm is described in detail in [19]. Measurement packets are sent by a sender host to a receiver host, which immediately transmits the packets back to the sender host. The sender host, using the arrival intervals of the echoed packets, estimates the bandwidth available in the path. When this algorithm is plugged into a TCP connection, the packets sent from a receiver to a sender are replaced by ACK packets. During every measurement, the proposed measurement algorithm searches for the available bandwidth only within a given *search range*. The search range is a range of bandwidth that is

expected to include the current available bandwidth. By introducing the search range, we can avoid sending probe packets at an extremely high rate. We can also keep the number of probe packets for the measurement quite small. The following are the steps of the proposed algorithm for one measurement:

1. Set the initial search range.
   We first send a packet stream (a group of packets sent simultaneously) according to the Cprobe algorithm to obtain a very rough estimation of the available bandwidth and use the result to set the initial search range.
2. Actively probe the search range.
   We divide the search range into multiple sub-ranges of identical width of bandwidth. A packet stream is sent for each of the sub-ranges. The transmission rates of the packets vary to cover the sub-range of the bandwidth range. Simulation results in [19] show that in order to probe a search range, only two to four packet streams are required, depending on the width of the search range compared to the value of the last measurement result.
3. Find a sub-range which is expected to include the available bandwidth.
   We then check to see if an increasing trend exists in the transmission delay of each stream when the echoed packets arrives at the sender host. Because the increasing trend of the transmission delay in a stream indicates that the transmission rate of the stream is larger than the current available bandwidth of the network path, we can choose a sub-range which is most likely to include the correct value of the available bandwidths. That is, the sub-range corresponding to the stream indicates the middle of streams which have increasing trends and those which do not.
4. Determine the available bandwidth.
   We examine the arrival rates of every two successive packets of the stream corresponding to the sub-range chosen in Step 3. Since the arrival rate being larger than the transmission rate indicates that the transmission rate of the two packets is larger than the available bandwidth, we determine the available bandwidth as the largest rate of the pairs, for which the arrival rate is the same as the transmission rate.
5. Create a new search range and return to Step 2.
   We use the 95% confidence interval of previous measurement results as the width for the next search range, and the current available bandwidth is used as the center of the search range. When the available bandwidth can not be found within the search range (i.e., there is no appropriate sub-range in Step 3), the network status may have changed greatly so that the available bandwidth shift out of the search range. We then widen the search range in the possible direction of change of the available bandwidth.

3 ImTCP: TCP with Inline Network Measurement

3.1 Overview

We next insert a measurement program into the sender program of TCP Reno to create ImTCP sender. The measurement program is located at the bottom of

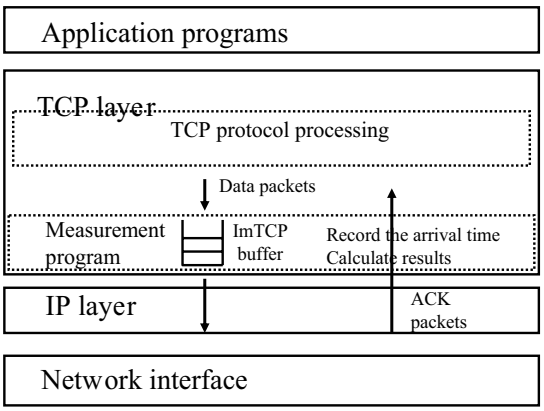


Fig. 1. Placement of measurement program at TCP sender

the TCP layer as shown in Figure 1. When a new data packet is generated at the TCP layer and is ready to be transmitted, the packet is stored in an intermediate FIFO buffer (hereafter referred to as the *ImTCP buffer*). The measurement program decides the time at which to send the packets in the buffer. On the other hand, when an ACK packet arrives at the sender host, the measurement program records its arrival time and passes the packet to the TCP layer for TCP protocol processing.

When the current window size is smaller than the number of packets required for a packet stream, ImTCP does not perform measurement, which means that the program passes all data packets immediately to the IP layer. When the window size is sufficiently large, ImTCP performs measurements. The measurement program waits until the number of packets in the ImTCP buffer is sufficient to form a packet stream, and then sends the stream at the transmission rate determined by the measurement algorithm. The program sends all streams required for a measurement (from two to four streams) and then calculates the measurement result from the time intervals of the corresponding ACK packets. While waiting for the ACK packets, the program passes all data packets immediately to the IP layer. The measurement requires the longest time when the window size is so small that only one measurement stream can be created in one RTT. In this case, it requires up to four RTTs. Thus, a measurement result is yielded in one to four RTTs.

The measurement program does not require any special changes in the TCP receiver program, except that a probe packet must be sent back for each received packet. Therefore, delayed ACKs must be disabled at the TCP receiver, otherwise ImTCP will not perform measurement properly. For example, a CDN company may disable delayed ACKs at all CDN servers so as to utilize ImTCP in its own private network. When delayed ACKs can not be disabled at the TCP receiver, in order to avoid degradation of the measurement accuracy, the mea-

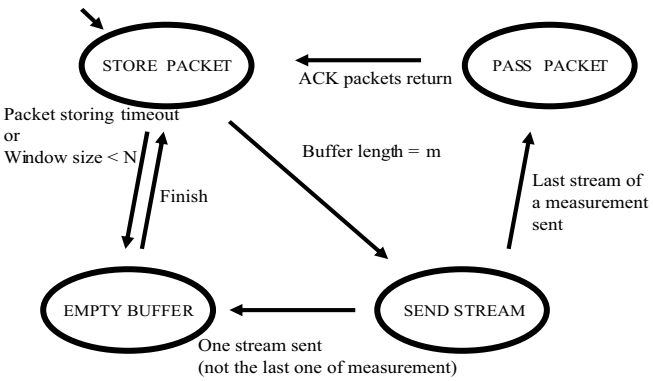


Fig. 2. State transition in the Control unit

surement program should double the number of packets per stream and Step 3 of the proposed measurement algorithm should be changed so that three-packet intervals are used rather than two-packet intervals.

3.2 Packet storing mechanism

The measurement program consists of three units. The ImTCP Buffer unit stores TCP data packets and passes each packet to the IP layer under control of the *Control unit*. ImTCP Buffer unit informs the Control unit when a new TCP packet arrives. The Control unit determines when to send the packets stored in the buffer. The Control unit receives search ranges from the *Measurement unit* and creates the measurement streams. The Measurement unit checks the arrival times of ACK packets and calculates the measurement results.

The Measurement unit is described in detail in Section 2. Here, we explain the operation of the Control unit. The Control unit has four functional states, STORE PACKET, PASS PACKET, SEND STREAM and EMPTY BUFFER, as shown in Figure 2. The Control unit is initially in the STORE PACKET state. In the following, we describe the detailed behavior of the Control unit in each state.

– *STORE PACKET* state
  - Start storing packets for the creation of measurement streams. Set the packet storing timer to end packet storing after a certain length of time T. The timer value T is discussed in Subsection 3.3.
  - Go to the SEND STREAM state if the number of stored packets is m. The value of m is discussed in Subsection 3.3.
  - Go to the EMPTY BUFFER state if the current TCP window size becomes smaller than N or if the packet storing timer expires. N is the number of packets needed to create a measurement stream.

– *EMPTY BUFFER* state
  - Pass currently stored packets to the IP layer until the buffer becomes empty.
  - Return to the STORE PACKET state.
– *SEND STREAM* state
  - Send a measurement stream. The transmission rate of the stream is determined according to the measurement algorithm. During stream transmission, packets arriving at the buffer are stored in the ImTCP buffer.
  - After the transmission of the stream, if the stream is the last for a measurement, go to the PASS PACKET state, if not, go to the EMPTY BUFFER state.
– *PASS PACKET* state
  - Pass every packet in the buffer immediately to the IP layer.
  - Go to the STORE PACKET state when all ACK packets of the transmitted measurement streams have arrived at the sender.

3.3 Parameter settings

– Number of packets in a measurement stream (N)
  We must choose a small value for N in order to reduce the number of packets stored in the ImTCP buffer for the measurement because storing a large number of packets may slow data transmission. According to simulation experiments in [19], the proposed measurement algorithm yields successful measurement results with acceptable accuracy if $5 \leq N$. Therefore, we choose $N = 5$.
– Number of packets required to start a measurement stream (m)
  If we set m to be large, then the packets will be stored in the ImTCP buffer for a long time, this slows the TCP transmission speed. If we set m to be very small (for example, 1 packet), then the latter part of the stream will not be available when the former part of the stream has already been transmitted, in which case stream transmission fails. Therefore, m must be just large enough to ensure successful transmission of the measurement stream, and no larger. The algorithm for determining m is given below. In the algorithm, m is adjusted according to whether or not transmission of the previous measurement streams was successful.
  - Set $m = N$ initially. The minimum value of m is 2, and the maximum value of m is N.
  - If F successive measurements are completed successfully, and m is greater than its minimum value of 2, then decrement m by 1. We set F to 2.
  - If a stream creation fails, and m is less than its maximum value of N, then decrement m by 1 and create the stream again.
– Packet storing timer (T)
  We determined that the time period required for N packets to arrive at the ImTCP buffer is a good bound for the time waiting packets for a measurement stream (T). In fact, if the timer is too short, the probability for m

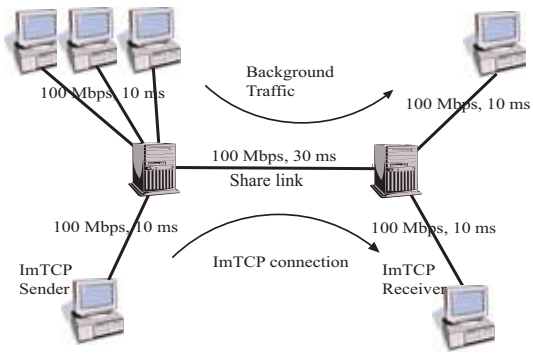


Fig. 3. Network model for evaluation of the measurement results for ImTCP

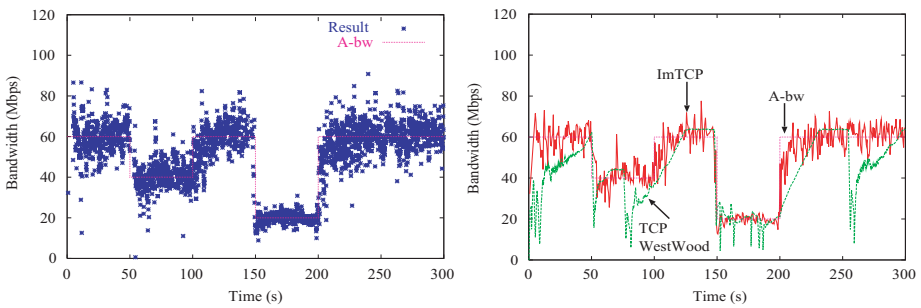
packets to gather at the ImTCP buffer is low, so the program may frequently fail to create packet streams. If the timer is too large, TCP data packets may be stored for a long time, and consequently TCP transmission speed may deteriorate.

The time for N packets to arrive at the ImTCP buffer can be considered as the time for N successive ACK packets to arrive at the sender (Z), if we suppose that one data packet is sent immediately when an ACK packet arrives. We also assume a normal distribution of data packet RTT (X) with average A_{RTT} and variance D_{RTT}, both of which can be inferred from the TCP timeout function [20]. Note that the time from the sending of the first of N successive data packets until the ACK of the last packet arrives at the sender (Y) is K plus the RTT of the last packet, where K is the time from the sending of the first packet until the last packet is sent. Therefore, the distribution of Y is $N(A_{RTT} + K, D_{RTT})$. Since $Z = Y - X$, we can obtain the distribution of Z, as $N(K, 2 \cdot D_{RTT})$. Considering that the TCP packet transmission rate is a rough estimate of the available bandwidth, we approximate K by $\frac{M}{A}(N-1)$, where M is the packet size and A is the value of available bandwidth which can be obtained from the measurement results. From the distribution of Z we determine: $T = \frac{M}{A}(N-1) + 4 \cdot D_{RTT}$. Using this value for the timer, the probability of N packets arriving at the ImTCP buffer reaches approximately 98% due to the characteristics of the normal distribution.

4 Simulation results

4.1 Measurement accuracy

Our first simulation in ns-2 uses the network model described in Figure 3. The ImTCP sender sends data to the receiver and, at the same time, measures the available bandwidth of the share link. The background traffic is made up of



(a) Measurement results for ImTCP

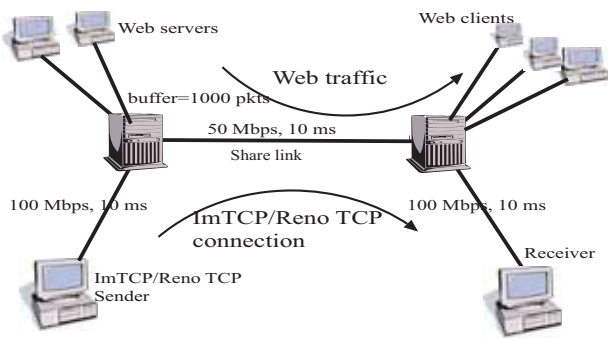
(b) Average measurement results for ImTCP and TCP Westwood

Fig. 4. Measurement results

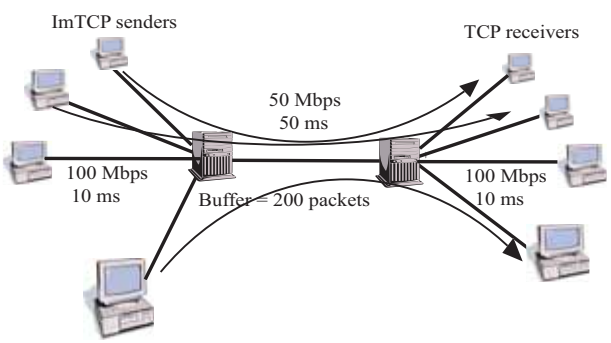
UDP packet flows. The correct value of the available bandwidth of the share link is calculated as the share link capacity minus the total rate of background traffic. During the simulation, the background traffic is changed so that the available bandwidth is 60 Mbps from 0 sec to 50 sec, 40 Mbps from 50 sec to 100 sec, 60 Mbps from 100 sec to 150 sec, 20 Mbps from 150 sec to 200 sec and 60 Mbps from 200 sec to 300 sec. Figure 4(a) shows the measurement results and correct values of the available bandwidth (A-bw). From the result, we can conclude that the proposed algorithm can measure well the available bandwidth, independent of the degree of change in available bandwidth. Figure 4(b) shows the average measurement results for every 0.5 sec. For comparison, we also show the measurement results of TCP Westwood (using the latest available version [18]) under the same network conditions. The figure indicates that when the available bandwidth increases suddenly, TCP Westwood yields results that are lower than the true value because the data transmission rate cannot adjust as rapidly and needs time to ramp up because of the self-clocking behavior of TCP. In contrast, the measurement results of ImTCP can reflect well the changes in the available bandwidth.

4.2 Effect of ImTCP on other traffic

We activate ImTCP and Reno TCP in turn on a network involving a large number of active Web document accesses, as shown in Figure 5(a). We use a large buffer (1000 packets) in the router at the shared link to help ImTCP/Reno TCP connections achieve high throughput because, here, the effect of ImTCP/Reno TCP connections on Web traffic is the focus of the simulation. We run the simulation for 500 sec and find that the average throughput of ImTCP is 25.2 Mbps while that of Reno TCP is 23.1 Mbps. This indicates that the data transmission speed of ImTCP is almost the same as that of Reno TCP (but not exactly

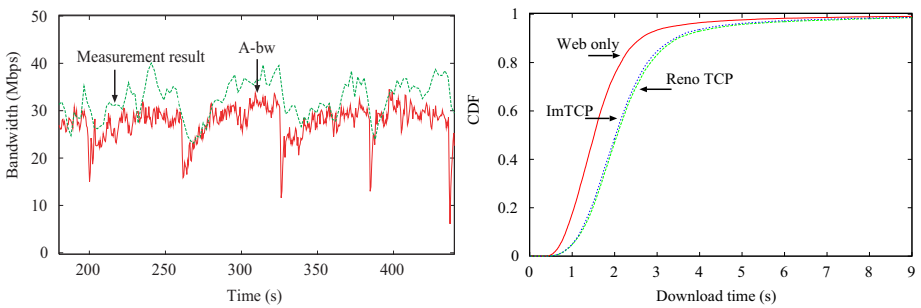


(a) Network model for evaluation of the effect of ImTCP on Web traffic



(b) Network model for investigating bandwidth utilization and fair share

Fig. 5. Network models



(a) Measurement results for ImTCP in a Web traffic environment

(b) Comparison of Web page download times

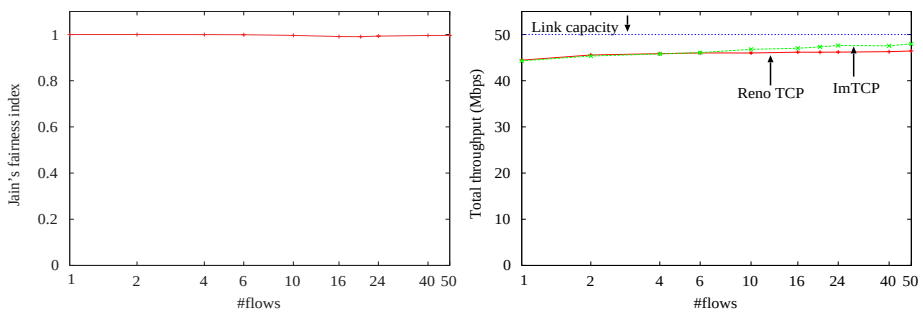
Fig. 6. Performance of ImTCP in a Web traffic environment

the same, due to the difference in the behaviour of TCP senders). Figure 6(a) confirms that the ImTCP measurement result reflects well the change in the available bandwidth.

We compare the effect of ImTCP and Reno TCP on Web page download time (for Web clients) in Figure 6(b). This figure shows the cumulative density functions (CDFs) of Web page download time for Web clients. ImTCP and Reno TCP have almost the same effect on Web page download time. This indicates that inline measurement does not affect other traffic sharing the link with ImTCP.

4.3 Bandwidth utilization and fair share

We use the network topology shown in Figure 5(b) with several ImTCP connections sharing a bottleneck link. By using a small buffer (200 packets) in the router at the bottleneck link to force conflict among connections, we vary the number of ImTCP connections while observing the total throughput and fairness among the connections. The line in Figure 7(a) denotes the Jain's fairness index [21] for the ImTCP connections. This index takes a value from 0 to 1; the fairness of a share is considered to increase as its Jain's fairness index approaches 1. We can see that the ImTCP connections share the bandwidth link fairly. Figure 7(b) shows the link utilization of ImTCP as we vary the number of connections. Also shown are the results when ImTCP is replaced by Reno TCP. ImTCP and Reno TCP have almost the same link utilization regardless of the number of connections.



(a) Fairness between ImTCP connections

(b) Link utilization of ImTCP

Fig. 7. Fairness and link utilization of ImTCP

5 Conclusions and Future Works

In the present paper, we introduced ImTCP, a version of TCP that can measure the available bandwidth. We evaluated ImTCP through simulation experiments and verified that the proposed measurement algorithm works well with no degradation of TCP data transmission speed. We are now implementing ImTCP in a real environment. In the future, we will also investigate a bandwidth measurement algorithm that can be deployed at the TCP receiver.

References

1. R.Wang, G.Pau, K.Yamada, M.Sanadidi and M.Gerla, "TCP startup performance in large bandwidth delay networks," in *Proceedings of INFOCOM '04*, 2004.
2. D.Katabi, M.Handley and C.Rohrs, "Congestion control for high bandwidth-delay product networks," in *Proceedings of ACM SIGCOMM*, 2002.
3. D. Andersen, H. Balakrishnan, M. Kaashoek and R. Morris, "Resilent overlay networks," in *Proceedings of 18th ACM SOSP*, Oct. 2001.
4. "The Internet Bandwidth Tester (TPTEST)," available at `http://tptest.sourceforge.net/about.php`.
5. R.Anjali, C.Scoglio, L.Chen, I.Akyildiz and G.Uhl, "ABEst: An available bandwidth estimator within an autonomous system," in *Proceedings of IEEE GLOBECOM 2002*, Nov. 2002.
6. S. Seshan, M. Stemm, and R. H. Katabi, "SPAND: Shared passive network performance discovery," in *Proceedings of the 1st Usenix Symposium on Internet Technologies and Systems (USITS '97)*, pp. 135–146, Dec. 1997.
7. R. L. Carter and M. E. Crovella, "Measuring bottleneck link speed in packet-switched networks," Tech. Rep. TR-96-006, Boston University Computer Science Department, Mar. 1996.

8. M. Jain and C. Dovrolis, "End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput," in *Proceedings of ACM SIGCOMM 2002*, Aug. 2002.
9. V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil and L. Cottrell, "PathChirp: Efficient available bandwidth estimation for network paths," in *Proceedings of Passive and Active Measurement Workshop*, 2003.
10. "Network Test," available at `http://www-didc.lbl.gov/NCS/netest/` .
11. N.Hu and P.Steenkiste, "Evaluation and characterization of available bandwidth probing techniques," *IEEE Journal on Selected Areas in Communications*, vol. 21, Aug. 2003.
12. J.Strauss, D.Katabi and F.Kaashoek, "A measurement study of available bandwidth estimation tools," in *Proceedings of Internet Measurement Conference 2003*.
13. M. Gerla, Y. Sanadidi, R. Wang, A. Zanella, C. Casetti and S. Mascolo, "TCP Vegas: New techniques for congestion detection and avoidance," in *Proceedings of the SIGCOMM '94 Symposium*, pp. 24–35, Aug. 1994.
14. S. Savage, "Sting: A TCP-based network measurement tool," in *Proceedings of USITS '99*, Oct. 1999.
15. "Sprobe," available at `http://sprobe.cs.washington.edu`.
16. J. C. Hoe, "Improving the start-up behavior of a congestion control sheme for TCP," in *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, vol. 26,4, pp. 270–280, ACM Press, 1996.
17. Mark Allman and Vern Paxson, "On estimating end-to-end network path properties," in *Proceedings of SIGCOMM '99*, pp. 263–274, 1999.
18. M.Gerla, B.Ng, M.Sanadidi, M.Valla, R.Wang, "TCP Westwood with adaptive bandwidth estimation to improve efficiency/friendliness tradeoffs," *To appear in Computer Communication Journal*.
19. Cao Man, Go Hasegawa and Masayuki Murata, "A new available bandwidth measurement technique for service overlay networks," in *Proceeding of 6th IFIP/IEEE International Conference on Management of Multimedia Networks and Services Conference, MMNS2003*, pp. 436–448, Sept. 2003.
20. R. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*. Addison-Wesley, 1994.
21. R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley-Interscience, 1991.