ImTCP: TCP with an Inline Measurement Mechanism for Available Bandwidth

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
E-mail: 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 first introduce a new bandwidth measurement algorithm that can perform measurement estimates quickly and continuously and is suitable for inline measurement because of the smaller number of probe packets required and the negligible effect on other network traffic. We then show how the algorithm is applied in RenoTCP through a modification to the TCP sender only. We call the modified version of RenoTCP that incorporates the proposed mechanism *ImTCP* (Inline measurement TCP). 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). We also give examples in which measurement results help improving TCP performance.**

Keywords— **Available Bandwidth, Inline Measurement, TCP**

I. 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, such as:

- P2P (peer-to-peer) networks. When a resource discovery mechanism finds a requested content residing at multiple peers, bandwidth information is used to determine which peer should transmit the content.
- Grid networks. When multiple storage sites contain the same data, bandwidth information is used to determine which storage site is copied or read.
- CDN (content delivery network). When backup data or cached data is transmitted, bandwidth information is used to prevent other network traffic from being deprived of resources.

Available bandwidth is also used in network topology design and is a key factor in network troubleshooting when isolating fault locations [3].

Available bandwidth information can be measured at routers within a network [4, 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 [7–11].

However, sending extra probe traffic into a network is a common weakness in all active available bandwidth measurement tools. This is because the available bandwidth must be filled momentarily by the probe traffic, otherwise the true value of the bandwidth cannot be detected. Depending on the algorithm used, the amount of required probe traffic differs. According to one study [10], 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 [10] 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. For example, in an Internet weather center, probe traffic requires synchronization to reduce conflicts, because accuracy in measurement may decrease by some 50% when such conflicts are present [12].

In this paper, 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*). When a sender TCP sends data packets, it 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 [13]. The sender then utilizes the arrival interval of these packets to calculate the available bandwidth. While the measurement accuracy of this method may be affected by delays at the receiver or by differences in the size of ACK packets and data packets, this method has the advantage of executing at only one host (the sender), making it simple to implement and to use.

We first introduce a measurement algorithm suitable for inline network measurement 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. This is done rather than searching from 0 bps to the upper limit of the physical bandwidth with every measurement as existing algorithms do [8, 9]. By limiting the measurement range, we can avoid sending probe packets at an extremely high rate and keep the number of probe packets small.

We then introduce ImTCP (Inline measurement TCP), a Reno-based TCP that includes the proposed algorithm for inline network measurement described above. When a sender transmits data packets, ImTCP first stores a group up to several packets in a queue and subsequently forwards them at a transmission rate determined by the measurement algorithm. Each group of packets corresponds to a probe stream. Then, considering ACK packets as echoed packets, the ImTCP sender estimates available bandwidth according to the algorithm. To minimize transmission delay caused by the packet store-and-forward process, we introduce an algorithm using the RTO (round trip timeout) calculation in TCP to regulate packet storage time in the queue. 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.

We will also present two examples using control modes of the ImTCP congestion window to show how measurement results can be applied to TCP data transmission. In *background mode*, ImTCP uses the results of bandwidth availability measurements to prevent its own traffic from degrading the throughput of other traffic. This allows a prioritization of other traffic sharing the network bandwidth. In *full-speed mode*, ImTCP uses measurement results to keep its transmission rate close to the measured value necessary for optimum utilization of the available network bandwidth. This mode is expected to be used in wireless and high-speed networks where traditional TCP cannot use the available bandwidth effectively [14, 15].

The remainder of this paper is organized as follows. In Section II, we discuss related works concerning inline measurement. In Section III, we explain the requirements of an inline measurement algorithm and consider problems found with existing network measurement methods. We then introduce our proposed algorithm for inline network measurement. In Section IV, we introduce ImTCP and evaluate its performance. In Section V, we introduce two examples of congestion window control mechanisms for ImTCP. Finally in Section VI, we present concluding remarks and discuss future projects.

II. Inline network measurement

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 [16], also measures the packet transmission delay. There are, in addition, other tools that convert the TCP data transmission stack into network measurement tools; Sting [17] (measuring packet loss) and Sprobe [18] (measuring capacity in a 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. Moreover, the TCP sender window size often does not accurately represent the available bandwidth due to the nature of the TCP congestion control mechanism. The first TCP measurement algorithm to improve accuracy used a passive method in which the sender checks ACK arrival intervals to infer available bandwidth [19]. It is a simple approach based on the Cprobe [13] algorithm, but does not yield good results [20]. A similar technique is used in TCP Westwood [21] 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. Especially when the available bandwidth increases suddenly, the TCP data transmission rate cannot adjust as rapidly and needs time to ramp up because of the self-clocking behavior of TCP. Meanwhile, as transmission proceeds at a rate lower than the available bandwidth, the measurement algorithm yields results lower than the true value.

Our proposed mechanism uses an active approach for inline measurement. That is, the sender TCP does not only observe ACK packet arrival intervals, but also actively adjusts the transmission interval of data packets. The sender thus collects more information for a measurement and improved accuracy can be expected. Moreover, the proposed mechanism requires a modification of the TCP sender only, incurring the same deployment cost as the approaches of [19, 21].

III. Inline Network Measurement Algorithm

In this section we discuss algorithms for TCP measurement of available bandwidth. We first explain why existing algorithms are not used, then summarize our proposed algorithm.

A. Existing network measurement methods

We consider the following factors to be necessary for an inline network measurement algorithm. First, because the TCP window size limits the number of packets available for transmission at any one time, the measurement algorithm should use as small a number of packets as possible for a relatively small TCP window. Second, measurement should not affect the external traffic. Third, if possible, the measurement should provide results continuously and quickly so that up-to-date information on the IP network can be obtained.

We examined the current active measurement methods and found that none satisfies the requirements mentioned above. 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 [13], TOPP [7], PathLoad [8], Pathchirp [9] and netest [22] 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. Cprobe yields a result after only one group of probe packets, but transmits those packets at the highest rate possible by the sender host and impacts other traffic adversely if repeated continuously. Tools based on probe gap calculate available bandwidth from the change in time gaps between successive probe packets. Delphi [23], Spruce [24] are IGI/PTR [11] 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.

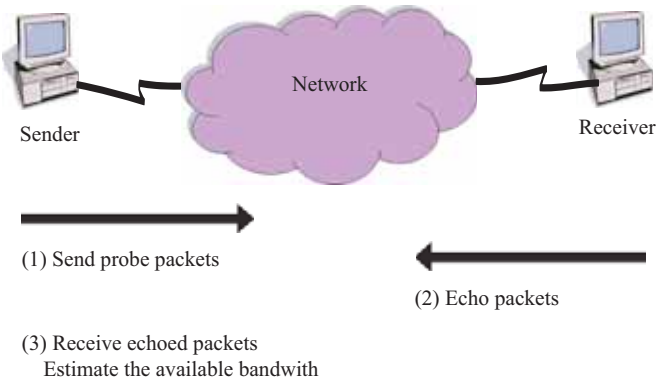


Fig. 1. Outline of proposed measurement algorithm

Our proposed measurement algorithm satisfies the abovementioned requirements using an algorithm based on the probe rate method and a technique to reduce the amount of probe traffic.

B. Proposed measurement algorithm

Figure 1 shows an outline of the proposed measurement algorithm. A sender host transmits measurement packets to a receiver host, which immediately sends received packets back to the sender host. The sender then estimates the available bandwidth of the path using the arrival intervals of the echoed packets.

In every measurement, we use a *search range* to find the value of the available bandwidth. Search range I$=(B_l, B_u)$ is a range of bandwidth which is expected to include the current value of the available bandwidth. The proposed measurement algorithm searches for the available bandwidth only within the given search range. The minimum value of B_l, the lower bound of the search range, is 0, and the maximum value of B_u, the upper bound, is equal to the physical bandwidth of the link directly connected to the sender host. By introducing the search range, we can avoid sending probe packets at an extremely high rate, which seriously affects other traffic. We can also keep the number of probe packets for the measurement quite small. As discussed later herein, even when the value of the available bandwidth does not exist within the search range, we can find the correct value in a few measure-

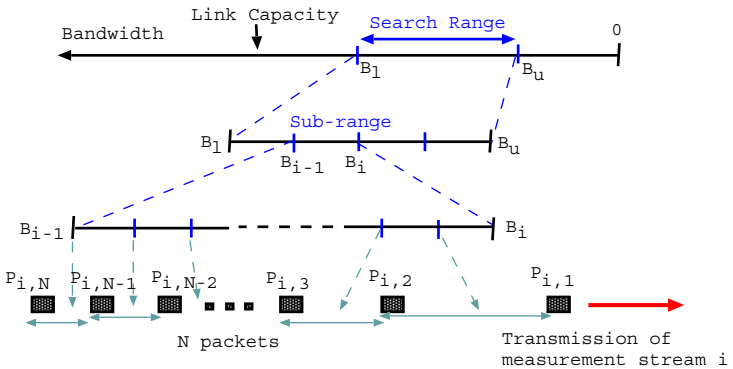


Fig. 2. Relationship of search range, sub-ranges, streams, and probe packets

ments. The following are the steps of the proposed algorithm for one measurement of the available bandwidth A:

1. Set the initial search range.
2. Divide the search range into multiple sub-ranges.
3. Inject a packet stream into the network for each sub-range and check the increasing trend of the packet inter-arrival times of the received stream.
4. Find a sub-range which is expected to include the correct value of the available bandwidth using the increasing trends of sent streams.
5. Calculate the available bandwidth by means of linear regression analysis for the chosen sub-range.
6. Create a new search range and return to Step 2.

A packet stream is a group of packets sent at one time for the measurement. In what follows, we explain in detail the algorithm by which to implement the above steps.

1. Set initial search range
   We first send a packet stream according to the Cprobe algorithm [13] to find a very rough estimation of the available bandwidth. We set the search range to $(A_{cprobe}/2, A_{cprobe})$, where A_{cprobe} is the result of the Cprobe test.
2. Divide the search range
   We divide the search range into k sub-ranges $\mathrm{I}_i = (B_{i+1}, B_i)$ $(i = 1, 2..k)$. All sub-ranges have the identical width of the bandwidth. That is,

$$B_i = B_u - \frac{B_u - B_l}{k}(i-1) \quad (i = 1, ..., k+1)$$

   As k increases, the results of Steps 4 and 6 become more accurate, because the width of each sub-range becomes smaller. However, a larger number of packet streams is required, which results in an increase in the number of used packets and the measurement time.
3. Send packet streams and check increasing trend

For each of k sub-ranges, a packet stream i $(i = 1...k)$ is sent. The transmission rates of the stream's packets vary to cover the bandwidth range of the sub-range. We denote the j-th packet of the packet stream i as $\mathrm{P}_{i,j}$ $(1 \leq j \leq N$, where N is the number of packets in a stream) and the time at which $\mathrm{P}_{i,j}$ is sent from the sender host as $S_{i,j}$, where $S_{i,1} = 0$. Then $S_{i,j}$ $(j = 2..N)$ is set so that the following equation is satisfied:

$$\frac{M}{S_{i,j} - S_{i,j-1}} = B_{i+1} + \frac{B_i - B_{i+1}}{N-1}(j-1)$$

where M is the size of the probe packet. Figure 2 shows the relationship between the search range, the sub-ranges and the packet streams. In the proposed algorithm, packets in a stream are transmitted with different intervals, for this reason the measurement result may not be as accurate as the Pathload algorithm [8], in which all packets in a stream are sent with identical intervals. However, the proposed algorithm can check a wide range of bandwidth with one stream, whereas the Pathload checks only one value of the bandwidth with one stream. This reduces the number of probe packets and the time required for measurement. By this mechanism, the measurement speed is improved at the expense of measurement accuracy.

We then observe $R_{i,j}$, the time the packet $\mathrm{P}_{i,j}$ arrives at the sender host, where $R_{i,1} = 0$. We calculate the transmission delay $D_{i,j}$ of $P_{i,j}$ using the function $D_{i,j} = R_{i,j} - S_{i,j}$. We then check if an increasing trend exists in the transmission delay $(D_{i,j} - D_{i,j-1})$ $(2 \leq j \leq N)$ according to the algorithm used in [8]. As explained in [8], the increasing trend of transmission delay in a stream indicates that the transmission rate of the stream is larger than the current available bandwidth of the network path.

Let T_i be the increasing trend of stream i as follows:

$$T_i = \begin{cases} 1 & \text{increasing trend in stream } i \\ -1 & \text{no increasing trend in stream } i \\ 0 & \text{unable to determine} \end{cases}$$

As i increases, the rate of stream i decreases. Therefore, T_i is expected to be 1 when i is sufficiently small. On the other hand, when i becomes large, T_i is expected to become -1. Therefore, when neither of the successive streams m or $m+1$ have an increasing trend ($T_m = T_{m+1} = -1$), the remaining streams are expected not to have increasing trends ($T_i = -1$ for $m+2 \leq i \leq k$). Therefore, we stop sending the remaining streams in order to speed up the measurement.

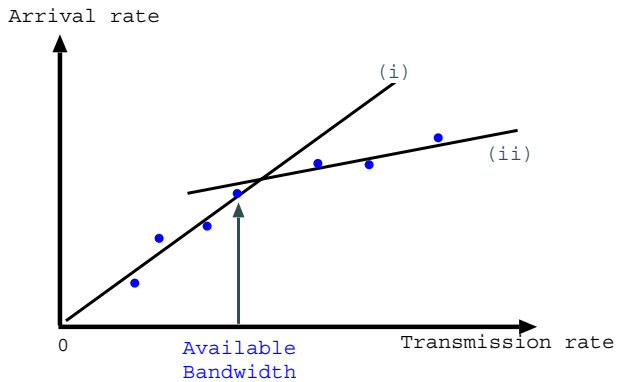


Fig. 3. Finding the available bandwidth within a sub-range

4. Choose a sub-range
   Based on the increasing trends of all streams, we choose a sub-range which is most likely to include the correct value of the available bandwidth. First, we find the value of a $(0 \leq a \leq k+1)$, which maximizes $(\sum_{j=0}^{a} T_j - \sum_{j=a+1}^{k} T_j)$. If $1 \leq a \leq k$, we determine the sub-range I_a is the most likely candidate of the sub-range which includes the available bandwidth value. That is, as a result of the above calculation, I_a indicates the middle of streams which have increasing trends and those which do not. If $a = 0$ or $a = k+1$, on the other hand, the algorithm decides that the available bandwidth does not exist in the search range (B_l, B_u). We determine that the available bandwidth is larger than the upper bound of the search range when $a = 0$, and that when $a = k+1$ the available bandwidth is smaller than the lower bound of the search range.
   In this way, we find the sub-range which is expected to include the available bandwidth according to the increasing trends of the packet streams.
5. Calculate the available bandwidth
   We then derive the available bandwidth A from the sub-range I_a chosen by Step 4. We first determine the transmission rate and the arrival rate of the packet $\mathrm{P}_{a,j}$ $(j = 2...N)$ as $\frac{M}{S_{a,j}-S_{a,j-1}}$, $\frac{M}{R_{a,j}-R_{a,j-1}}$, respectively. We then approximate the relationship between the transmission rate and the arrival rate as two straight lines using the linear regression method, as shown in Figure 3. Since we determine that the sub-range I_a includes the available bandwidth, the slope of line (i) which consists of small transmission rates is nearly 1 (the transmission rate and the arrival rate are almost equal), and the slope of line (ii) which consists of larger transmission rates is smaller than 1 (the arrival rate is smaller than the transmission rate). Therefore, we determine that the highest transmission rate in line (i) is the value of the available bandwidth.
   On the other hand, when we have determined that the available bandwidth value does not exist in the search range (B_l, B_u) in Step 4, we temporarily set the value of available bandwidth as follows:

$$A = \begin{cases} B_l & a = 0 \\ B_u & a = k+1 \end{cases}$$

6. Create a new search range
   When we have found the value of the available bandwidth from a sub-range I_a in Step 5, we accumulate the value as the latest statistical data of the available bandwidth. The next search range (B'_l, B'_u) is calculated as follows:

$$B'_l = A - \max\left(1.96\frac{S}{\sqrt{q}}, \frac{B_m}{2}\right)$$

$$B'_u = A + \max\left(1.96\frac{S}{\sqrt{q}}, \frac{B_m}{2}\right)$$

   where S is the variance of stored values of the available bandwidth and q is the number of stored values. Thus, we use the 95% confidential interval of the stored data as the width of the next search range, and the current available bandwidth is used as the center of the search range. B_m is the lower bound of the width of the search range, which is used to prevent the range from being too small. When no accumulated data exists (when the measurement has just started or just after the accumulated data is discarded), we use the same search range as that of the previous measurement.
   On the other hand, when we can not find the available bandwidth within the search range, it is possible to consider that the network status has changed greatly. Therefore, we discard the accumulated data because this data becomes unreliable as statistical data. In this case, the next search range (B'_l, B'_u) is set as follows:

$$B'_l = \begin{cases} B_l & a = 0 \\ B_l - \frac{B_u - B_l}{2} & a = k+1 \end{cases}$$

$$B'_u = \begin{cases} B_u + \frac{B_u - B_l}{2} & a = 0 \\ B_u & a = k+1 \end{cases}$$

   This modification of the search range is performed in an attempt to widen the search range in the possible direction of the change of the available bandwidth.

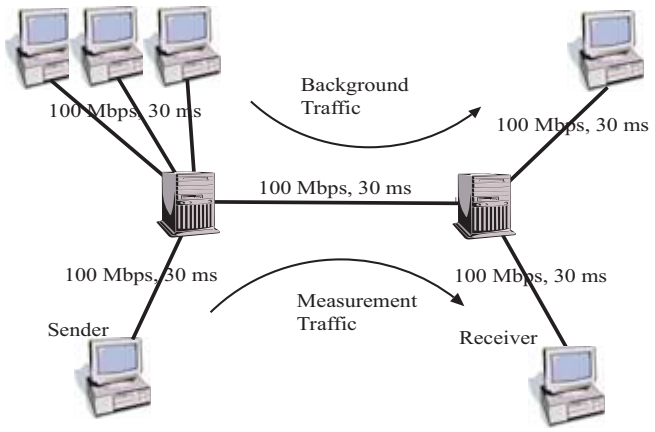


Fig. 4. Network model for evaluation of the proposed measurement algorithm

By this statistical mechanism, we expect the measurement algorithm to behave as follows: when the available bandwidth does not change greatly over a period of time, the search range becomes smaller and more accurate measurement results can be obtained. On the other hand, when the available bandwidth varies greatly, the search range becomes large and the measurement can be restarted from the rough estimation. That is, the proposed algorithm can give a very accurate estimation of the available bandwidth when the network is stable, and a rough but rapid estimate can be obtained when the network status changes.

C. Simulation results

This Subsection shows some simulation results in ns [25] and validates the measurement algorithm proposed in Subsection III-B. Figure 4 shows the network model used in the simulation. A sender host connects to a receiver host through a bottleneck link. The capacity of the bottleneck link is 100 Mbps and the propagation delay is 30 msec. All of the links from the endhosts to the routers have a 100-Mbps bandwidth and a 30-msec propagation delay.

There is background traffic generated by endhosts connecting to the routers. The background traffic is made up of UDP packet flows, in which various packet sizes are used according to the monitored results in the Internet reported in [26]. The correct value of the available bandwidth of the bottleneck link is calculated as the bottleneck link capacity minus the total rate of UDP traffic. We make the available bandwidth on the bottleneck link fluctuate by changing background traffic rates.

The sender host sends probe packets to the receiver host and the receiver host echoes the packets back to the sender. The sender, using the algorithm proposed in Subsection III-B , measures the available bandwidth of the path between the two hosts. In this situation, the result corresponds to the available bandwidth of the bottleneck link between the routers.

The number of sub-range k, which a search range is divided into, is decided according to the width of the search range and the latest result of the measured available bandwidth, A_{prev};

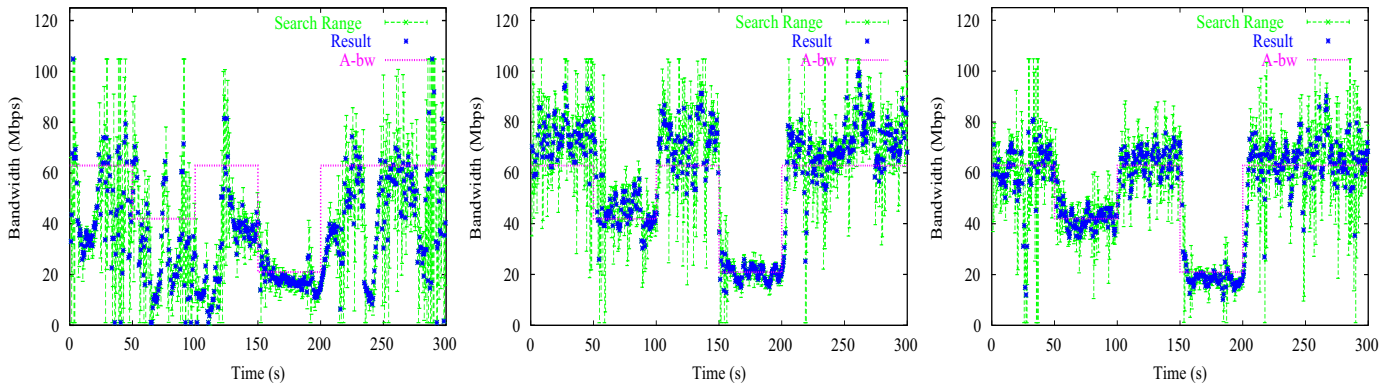
$$k = \begin{cases} 2 & (0 \leq \frac{B_u - B_l}{A_{prev}} < 0.15) \\ 3 & (0.15 \leq \frac{B_u - B_l}{A_{prev}} < 0.2) \\ 4 & (0.2 \leq \frac{B_u - B_l}{A_{prev}}) \end{cases}$$

B_m, the lower bound of the width of search ranges, is set to 10% of A_{prev}. The probe packet size is 1500 Bytes.

Figure 5 shows the measurement results of the available bandwidth and the search ranges for a simulation time of 300 sec. During the simulation, the background traffic is changed so that the available bandwidth of the bottleneck link 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. We also plot the correct values of the available bandwidth in all figures. Figures 5(a)-5(c) show the results when the number of the probe packets in a stream (N) is 3, 5 and 8, respectively. These figures indicate that when N is 3, the measurement results are far from the correct values. When N becomes larger than 5, on the other hand, the estimation result accuracy increases. The proposed measurement algorithm can determine the available bandwidth rapidly, even when the available bandwidth changes suddenly. When N is very small, we can not determine the increasing trend of the streams correctly in Step 3 in the proposed algorithm, which leads to the incorrect choice of sub-range in Step 4. Although the accuracy of measurement results increases as N is increased from 5 to 8, N=5 is judged to be the better setting since we place a higher priority on measurement speed than on measurement accuracy.

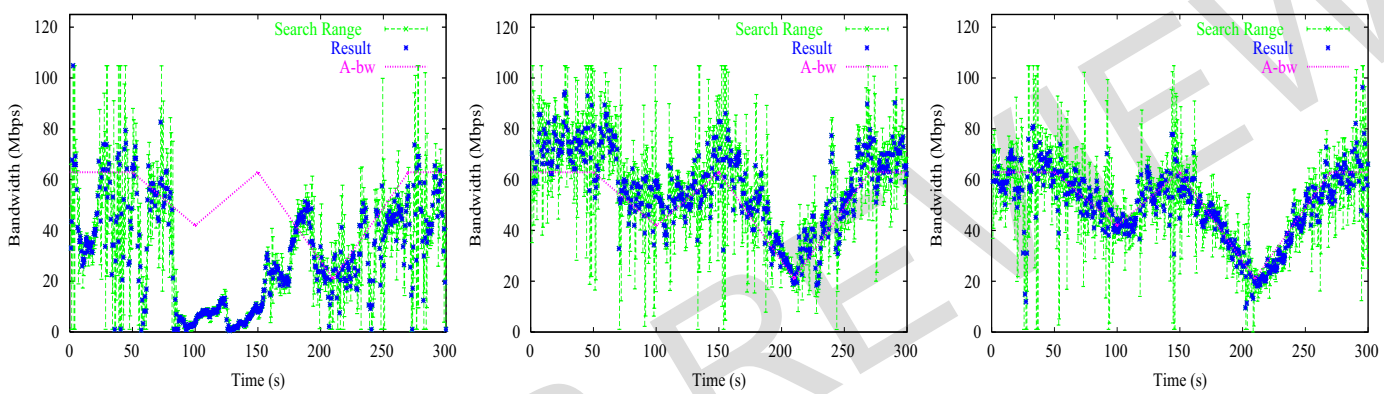
We next show another result in which we change the available bandwidth slowly as follows: from 0 sec to 50 sec, the available bandwidth is 60 Mbps; from 50 sec to 100 sec, decreases to 40 Mbps; from 100 sec to 150 sec, increases to 60 Mbps; from 150 sec to 210 sec, decreases to 20 Mbps; from 120 sec to 270 sec, increases to 60 Mbps; and from 270 sec to 300 sec the available bandwidth is 60 Mbps. The simulation results are shown in Figure 6. When N = 3, the estimation results are not accurate, but when N is 5 or 8, the results becomes acceptable.

From these simulation results, we can conclude that the proposed algorithm can measure well the available band-



(a) N=3 (b) N=5 (c) N=8

Fig. 5. Results of the proposed measurement algorithm (1)



(a) N=3 (b) N=5 (c) N=8

Fig. 6. Results of the proposed measurement algorithm (2)

width, independent of the degree of change in available bandwidth.

IV. ImTCP: TCP with Inline Network Measurement

A. Overview

In this Section we show how to apply the measurement algorithm mentioned in Section III to TCP by considering data packets as probe packets and ACK packets as echoed packets.

Because the program for inline network measurement must know the current size of the TCP congestion window, it should be implemented at the bottom of TCP layer as shown in Figure 7. When a new TCP data packet is generated at the TCP layer and is ready to be transmitted, it is stored in an intermediate FIFO buffer (hereafter called the *ImTCP buffer*) before being passed to the IP layer. On the other hand, when an ACK packet arrives at the sender host, the measurement program records its arrival time and passes it to the TCP layer for TCP protocol processing. When ImTCP performs a measurement, the program creates packet streams; it waits until a sufficient number of packets are in the ImTCP buffer then sends them at the transmission rate determined by the measurement algorithm. This is repeated until all streams required for a measurement have been transmitted. When ImTCP is not performing a measurement, it passes all TCP data packets immediately to the IP layer.

The program dynamically adapts to changes in the TCP window size. It stores no data packets when the current window size is smaller than the number of packets required for a measurement stream. This is because the TCP sender cannot transmit a number of data packets larger than the window size. On the other hand, when the window size is sufficiently large, the program creates all streams required

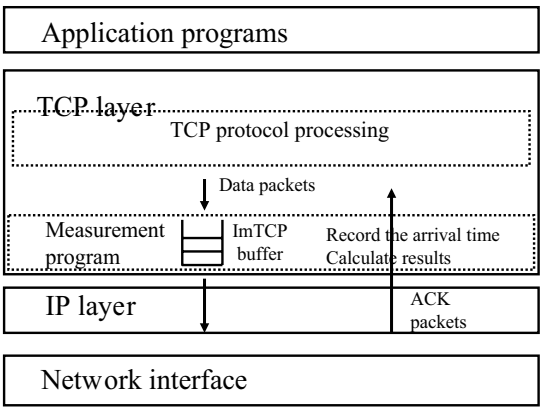


Fig. 7. Placement of measurement program at TCP sender

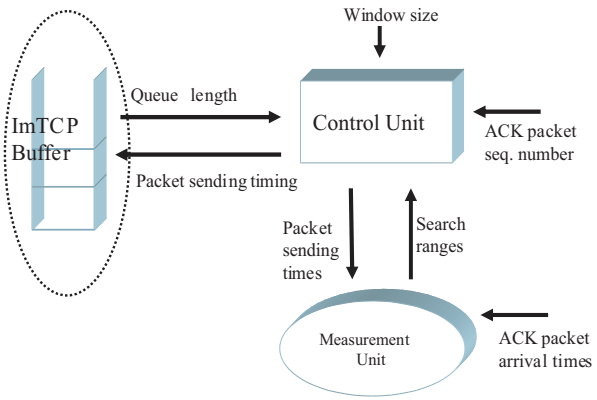


Fig. 8. Structure of the measurement program

for a measurement in every RTT.

B. Packet storing mechanism

Figure 8 shows the structure of the measurement program. It 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*. It informs the Control unit when a new TCP packet arrives. The Control unit determines when to send the packets stored in the buffer. It receives search ranges from the *Measurement unit* and creates the measurement streams. The Measurement unit checks the arrival times of ACK packets and calculates measurement results using corresponding sent packet departure times passed from the Control unit.

Details of the Measurement unit were introduced in Subsection III-B. 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 9. The Control unit is initially in the STORE PACKET state. In what follows, we describe the detailed behaviors 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 certain length of time T. The timer value T is discussed in Subsection IV-C.
  - Go to the SEND STREAM state if the number of stored packets equals to m. The value of m is discussed in Subsection IV-C.
  - Go to the EMPTY BUFFER state if the current TCP window size becomes smaller than N or 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 of a measurement, go to 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.

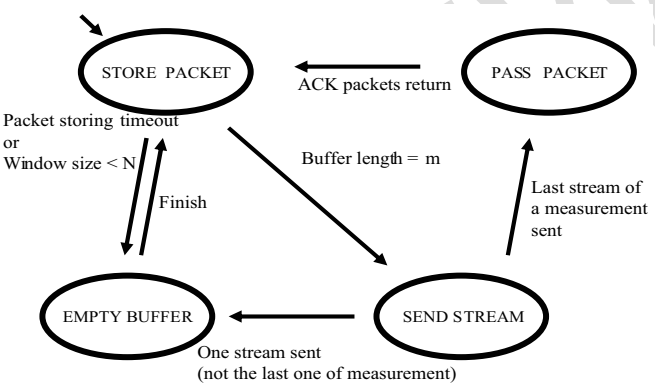


Fig. 9. State transition in the Control unit

C. Parameter settings

C.1 Number of packets required to start a measurement stream (m)

The timing for sending packets in a measurement stream is determined by the measurement algorithm. If N packets were stored prior to the beginning of transmission, the long storage time would slow the TCP transmission speed. Instead, transmission begins when only a partial number of packets (m out of N packets) have arrived in the ImTCP buffer. The timing is such that the former part of the stream is being transmitted as the latter part of the stream is still arriving at the buffer, and the latter packets are expected to arrive in time for transmission. Thus, we reduce the effect of the packet storing mechanism on TCP transmission.

If we set m to a very small value (for example, 1 packet), the latter part of the stream will not be available when the former part of the stream has already been transmitted, in which case the stream transmission fails. Therefore, m must be large enough to ensure successful transmission of the measurement stream, but 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 of m is 2, and the maximum of m is N.
- If F successive measurements are completed successfully, and m is greater than its minimum of 2, then decrement m by 1. We set F to 2.
- If a stream creation fails, and m is less than its maximum of N, then decrement m by 1 and create the stream again.

C.2 Number of packets in a measurement stream (N)

When network traffic changes rapidly, the fluctuating bandwidth may reduce the transmission rate (R_{send}) of measurement streams less than that the arrival rate of packets at the ImTCP buffer ($R_{arrival}$). In this case, packets congest the buffer during stream transmission. We can calculate the number of packets present in the buffer at completion of a measurement stream transmission as:

$$m + N(\frac{R_{arrival}}{R_{send}} - 1) \tag{1}$$

After transmitting a measurement stream, the Control unit transitions to the EMPTY BUFFER state and quickly sends all remaining packets in the buffer. We want to keep the number of transmitted packets as small as possible, however, to avoid degrading the TCP transmission rate. Considering that value m in Equation (1) is determined by algorithm C.1 of Section IV, we must choose a small value for N to reduce the number of packets in the ImTCP buffer. With a small N, measurements are performed quickly but with less accuracy. However, giving TCP performance priority over measurement accuracy, a small value for N is suitable.

According to simulation experiments in III-C, the proposed measurement algorithm yields successful measurement results if, and only if, $5 \leq N$. Therefore, we choose $N = 5$.

C.3 Packet storing timer (T)

We avoid degrading the TCP transmission speed, caused by storing data packets before they are passed to the IP layer, by appropriately setting a timer to stop the creation of a stream. Obviously, there is a trade-off between measurement frequency and TCP transmission speed when choosing the timer value. That is, for large timer values, the program can create measurement streams frequently so measurement frequency increases. In this case, however, because TCP data packets may be stored in the intermediate buffer for a relatively long period of time, TCP transmission speed may deteriorate. Moreover, long packet delays may lead to TCP timeout events. On the other hand, for small timer values, the program may frequently fail to create packet streams, leading a low frequency of measurement success. In the following discussion, we derive the appropriate value for the packet storing timer by applying an algorithm similar to the RTO calculation in TCP [27].

If we assume a normal distribution of packet RTTs with average A_{RTT} and variance D_{RTT}, A_{RTT} and D_{RTT} can be inferred from the TCP timeout function [27]. We use the following notation;

- X: RTT of a TCP data packet
- Y: The time since the first of N successive data packets is sent until the ACK of the last packet arrives at the sender
- Z: The time necessary for N successive ACK packets to arrive at the sender

We illustrate X, Y and Z in Figure 10. We need to know the distribution of Z to determine the appropriate value for the packet storing timer. From Figure 11, we can see that:

$$Z = Y - X \tag{2}$$

From the assumption mentioned above, X has a normal distribution $N(A_{RTT}, D_{RTT})$. Note that Y is the period of time from sending the first packet until the last packet is sent (we denote the length of this period as K) plus the RTT of the last packet. That is, we can conclude that the distribution of Y is $N(A_{RTT} + K, D_{RTT})$. From Equation (2) we then obtain the distribution of Z, as $N(K, 2 \cdot D_{RTT})$.

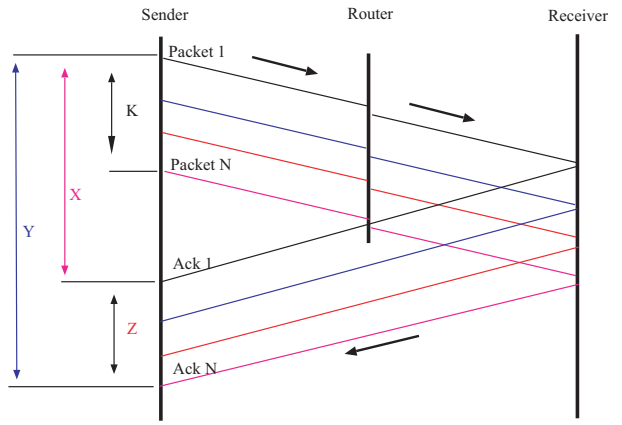


Fig. 10. Packets transmission times in TCP

Here, we provide a simple estimate of K. In a TCP flow, due to the self-clocking phenomenon, the TCP packet transmission rate is a rough estimate of the available bandwidth of the network link. The average time needed to send N successive TCP data packets is

$$K = \frac{M}{A}(N-1) \tag{3}$$

where M is the packet size and A is the value of available bandwidth which can obtain from the measurement results. From the distribution of Z and Equation (3), we determine the waiting time for N ACK packets as below:

$$T = \frac{M}{A}(N-1) + 4 \cdot D_{RTT}$$

Using this value for the timer, the probability of successfully collecting N packets reaches approximately 98% due to the characteristics of the normal distribution. Thus, we are using a relatively short timer length that reduces additional processing delays caused by the measurement program but provides a high probability of collecting a sufficient number of packets for creating measurement streams.

C.4 Measurement frequency

As explained in Subsection III-B, the measurement algorithm uses previous measurement results to determine a search range for the next measurement. Therefore, it is natural that only one measurement operation should be performed for one RTT. If the TCP window size is sufficiently large, we can perform multiple measurements for one RTT by introducing a quite complex mechanism. However, many difficulties must be overcome to accomplish this, including interaction of measurement tasks, delays caused by multiple streams. We therefore decided that ImTCP should perform at most one measurement operation per RTT. One RTT is long enough for ImTCP to recover the transmission rate after a measurement.

Thus, a measurement result is yielded in from one to four RTTs. Four RTTs per measurement occurs when the number of sub-range in a measurement reaches its maximum value (four) and the window size is so small that the measurment program can create only one measurement stream per RTT.

D. Other issues

D.1 Effect of Delayed ACK option

When a TCP receiver uses the delayed ACK option, it sends only one ACK packet for every two data packets. In this case, the proposed algorithm does not work properly since it assumes the receiver host will send back a probe packet for each received packet. To solve this problem, Step 3 in Subsection III-B of the proposed algorithm should be changed so that intervals of three packets are used rather than intervals of two packets. That is, we calculate the transmission delay $(D_{i,2j'+2} - D_{i,2j'})$ $(1 \leq j' \leq \lfloor N/2 \rfloor)$ for the probing packets in stream i in order to check its increasing trend. This modification has almost the same effect as halving the number of packets in one stream, resulting in a degradation in measurement accuracy. Therefore, the number of packets in a stream should be increased appropriately.

D.2 Effect of packet fragmentation

In the case where TCP packets are transmitted through a queue or node for which the MTU (Maximum Transmission Unit) is smaller than the packet size, the packets will be fragmented into several pieces in the network. The problem here becomes a question of whether measurement result will still be accurate if the packets in measurement streams become fragmented somewhere on the way to the receiver. We argue that fragmentation has little effect on the measurement results. The measurement algorithm is based on the increasing trend of the packet stream in order to estimate available bandwidth. Even with fragmentation, the stream still shows an increasing trend when and only when the transmission rate is larger than the available bandwidth. However, fragmentation does increase the packet processing overhead, which may in turn raise the increasing trend of packet streams if it occurs at a bottleneck link. This may lead to a slight underestimation in the measurement results.

E. Simulation results

E.1 Measurement accuracy

Our first simulation uses the same topology as described in Subsection III-C except that the UDP sender and receiver are replaced by an ImTCP sender and an ImTCP

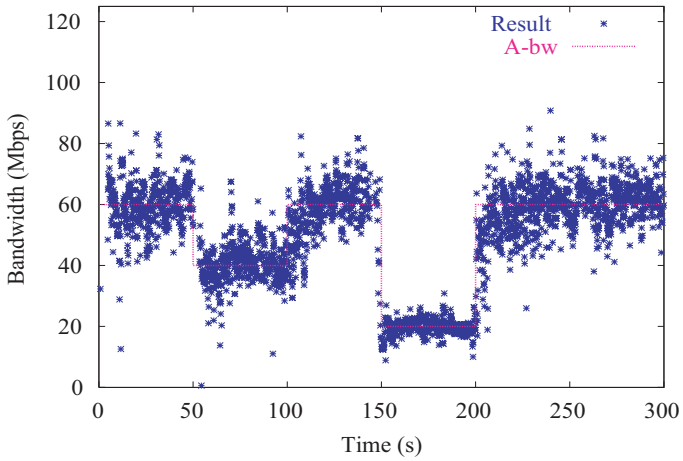


Fig. 11. Measurement results of ImTCP

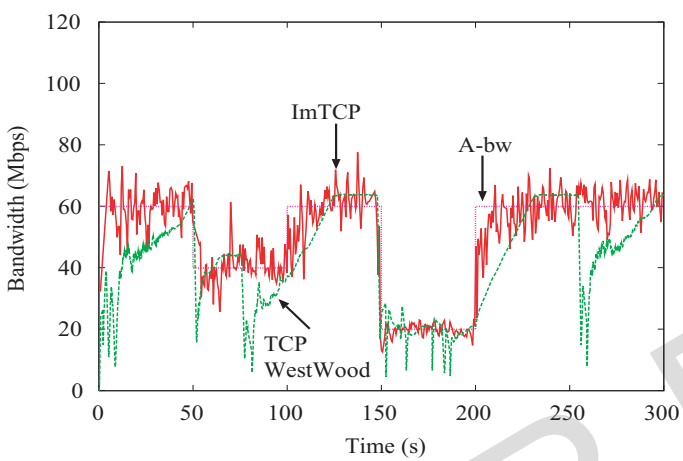


Fig. 12. Average measurement results of ImTCP and TCP Westwood

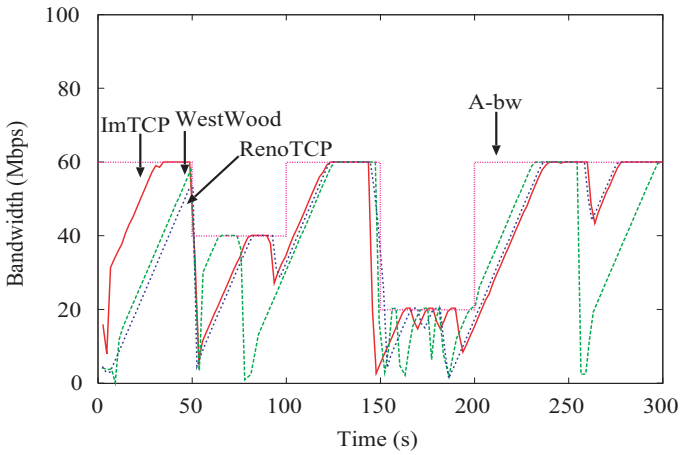


Fig. 13. Throughput of ImTCP and Reno TCP

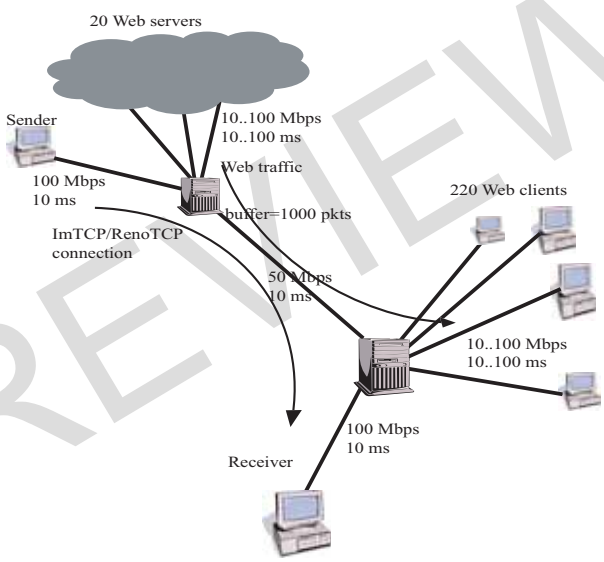


Fig. 14. Network model for evaluation of ImTCP's effect on Web traffic

receiver, respectively. Figure 11 show the measurement results for ImTCP in this case. Comparing Figure 11 with the measurement results of the algorithm where the number of packets (N) in a measurement stream is also five (Figure 5(b)), we can see that our measurement method can be successfully applied to TCP with no degradation in measurement accuracy.

Figure 12 shows the average measurement results of every 0.5 sec of ImTCP in this simulation. For comparision, we also show the measurement results of TCP Westwood (we use the latest version of TCP Westwood we have found [21]) in the same network conditions. Figure 13 shows the change in throughput of ImTCP in this simulation. We also show the case of Reno TCP and TCP Westwood. From the figures, we can see that ImTCP performs the measurement with a throughput almost the same as that of Reno TCP. An important point we can see from Figure 12 and 13 is that ImTCP yields accurate results even when the current throughput is lower than the available bandwidth. For example, from 0 sec to 30 sec in the simulation, although the throughput of ImTCP is less than 60 Mbps, the available bandwidth value is still realized, as shown in Figure 12. In contrast, TCP Westwood can not detect the real value of available bandwidth if it is much larger than the transmission throughput. This is the main difference between the measurement results of TCP Westwood and ImTCP.

E.2 Effect of ImTCP on other traffic

To investigate the effect of inline measurement on other traffic sharing the network, we compare the case of ImTCP to that of Reno TCP using the network model depicted in Figure 14. We activate ImTCP and Reno TCP in turn in the network involving a large number of active Web doc-

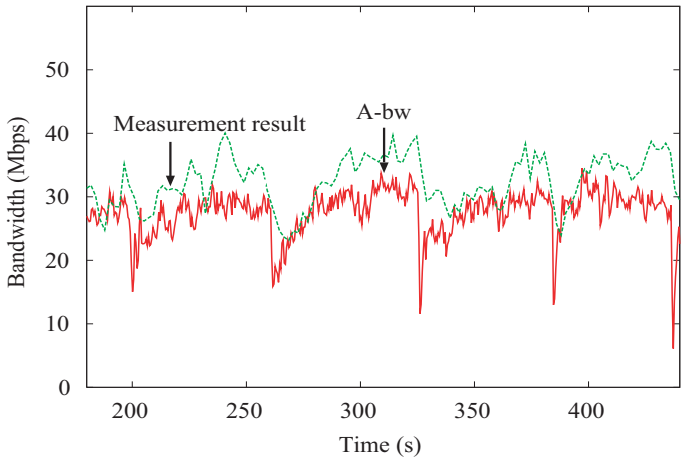


Fig. 15. Measurement result of ImTCP in Web traffic environment

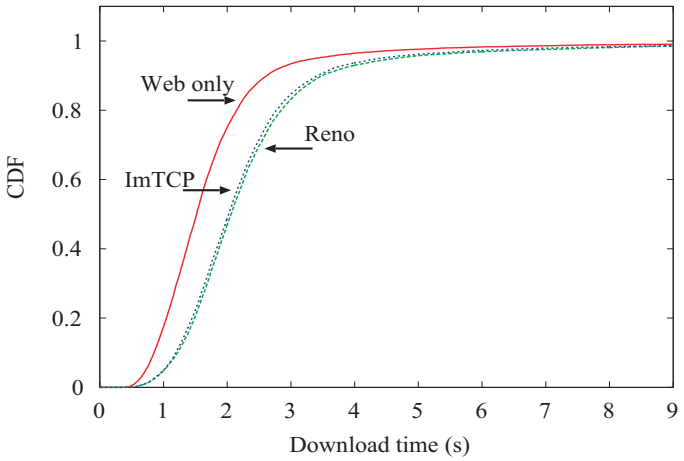


Fig. 16. Comparison of Web page download times

ument accesses. There are 220 Web clients downloading Web pages from 20 Web servers through a 50 Mbps shared link. We use a Pareto distribution for the Web object size distribution. According to previous studies in [28], we use 1.2 as the Pareto shape parameter with 12 KBytes as the average object size. The number of objects in a Web page is eight. The TCP sender and TCP receiver connect to a shared link through 100 Mbps links. 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. The results therefore show that data transmission speed of ImTCP is almost the same as that of Reno TCP. Figure 15 confirms that the ImTCP measurement result reflects the change in available bandwidth well.

We compare the effect of ImTCP and Reno TCP on Web page download time in Figure 16. This figure shows cumulative density functions (CDFs) of the Web page download time of Web clients. We can see that ImTCP and Reno TCP have almost the same effect on the download time of a Web page. This indicates that inline measurement does not affect other traffic sharing the link with ImTCP.

E.3 Bandwidth utilization and fair share

Two important characteristics of the Internet transport protocol are full utilization of link bandwidth and fair sharing of bandwidth among connections. We use the following simulation to show that ImTCP has these two characteristics.

We use the network topology shown in Figure 17 with

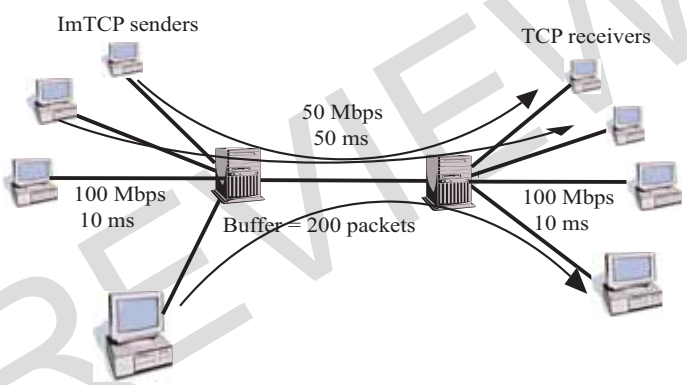
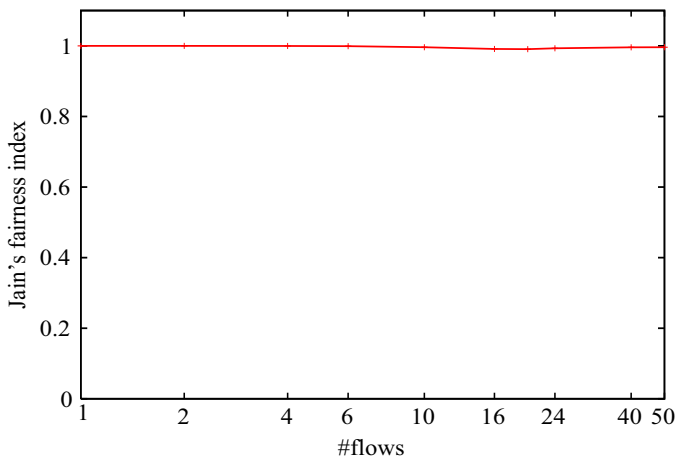


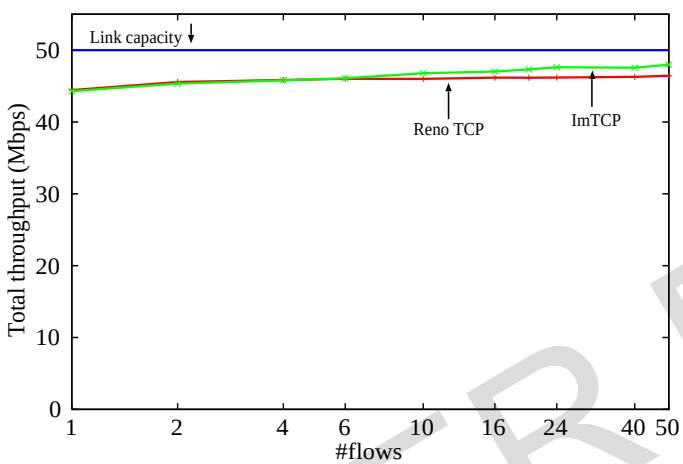
Fig. 17. Network model for investigating bandwidth utilization and fair share

many ImTCP connections sharing a bottleneck link. 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 total throughput and fairness among the connections.

The line in Figure 18(a) shows the Jain's fairness index [29] for the ImTCP connections. This index takes a value from 0 to 1; a share is considered fair as its index is near 1. We can see that the ImTCP connections share the bandwidth link fairly. Figure 18(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. Due to the small buffer size of the bottleneck link, when the number of connections are small the total throughput is not very high. When the number of connections is large, total throughput increases. We can see that ImTCP and Reno TCP have almost the same link utilization regardless of the number of connections.



(a) Fairness between ImTCP conections



(b) Link utilization of ImTCP

Fig. 18. Fairness and link utilization of ImTCP

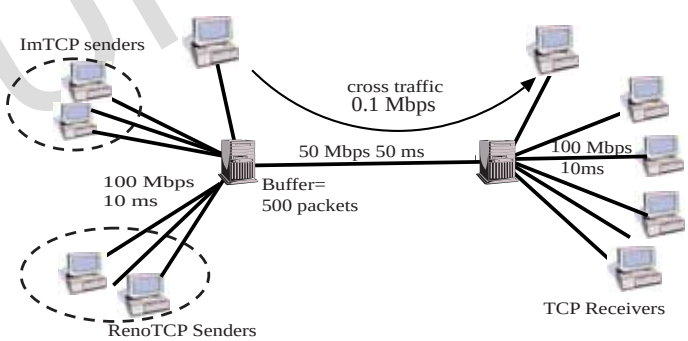


Fig. 19. Network model for investigating Reno TCP compatibility

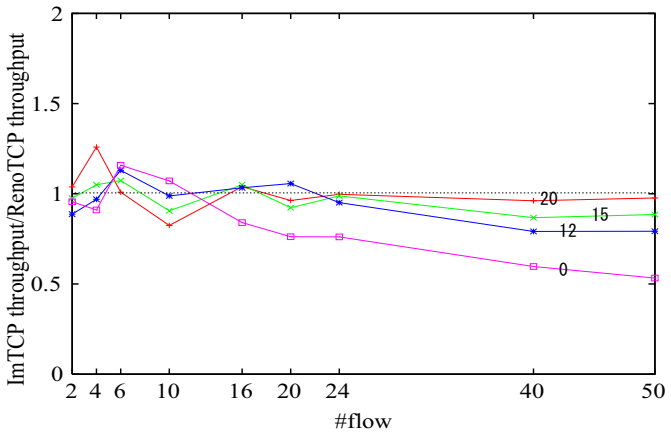


Fig. 20. Comparision of ImTCP and Reno TCP throughput. The number on the lines are the number of RTT between two measurements of ImTCP.

E.4 TCP-friendliness and TCP-compatibility

ImTCP is *TCP-friendly*; it achieves the same throughput as Reno TCP under the same condition, as shown in Figure 18(b). Although ImTCP buffers packet stream at the sender host, the buffered packets is quickly transmitted after each transmission of a packet stream (in the EMPTY BUFFER state). Therefore, there is almost no degradation in transmission speed of data packets.

A network protocol is called *TCP-compatible* if the connections using this protocol fairly share the bandwidth in an bottleneck link with Reno TCP [30]. We examine the TCP-compatibility of ImTCP by observing the throughput of ImTCP connections when they coexist with Reno TCP connections and non-TCP traffic. The non-TCP traffic is indicated by a 0.1 Mbps UDP flows with randomly varied packet size (300-600 bytes). All TCP and non-TCP traffic conflict at the 50 Mbps bottleneck link (see Figure 19). We use the same number of ImTCP and Reno TCP connections.

The ratio of the total throughput of ImTCP connections to that of Reno TCP connections is shown in Figure 20. When the ratio is around 1, ImTCP is TCP-compatible. The horizontal axis shows the total number of the TCP connections. In the current version of ImTCP, there is no time interval between 2 measurements. The result of this version is shown by the line numbered 0. We can see that ImTCP receives lower throughput than Reno TCP. The reason is as follows. Some of packets of ImTCP may not be transmitted in burst due to the affect of packets buffering at the sender. On the other hand, traditional TCP connections in competing environment have the trend to transmit packets in a bursty fashion. When the packets of ImTCP collide with the bursts of packets of Reno

TCP, they have higher probability to be dropt. Therefore, ImTCP with high measurement frequency may lost more packets when conflicting with Reno TCP, leading to a lower throughput.

The simple and effective way to overcome this problem is increasing the measurement interval of ImTCP. We next consider the cases when the measurement intervals are 12, 15 and 20 RTTs, and show the results by the line numbered 12, 15 and 20, respectively, in Figure 20. Note that the RTT in this case is 0.14 seconds and each measurement takes at most 4 RTTs. Therefore, 12, 15 and 20 RTT interval means ImTCP releases measurement results in 2.24(s), 2.66(s) and 3.36(s), respectively. When the measurement interval is relatively small, ImTCP achieves lower throughput than Reno TCP. On the other hand, when the measurement interval is equal to or larger than 20 RTTs, ImTCP is compatible to Reno TCP. In other words, when the measurement frequency is smaller than a certain value (in this simulation, that is $1/3.36$ times per second) there is a trade-off relationship between the TCP compatibility and the measurement frequency.

In such a heavy congested network that there is no available bandwidth even when ImTCP does not exist, ImTCP must be TCP-compatible in order to gain the equal throughput to other connections. Moreover, in this environment, the measurement results themselves usually do not bring so much valuable information so they will be not required updated frequently. Therefore, in this case, ImTCP must take a low measurement frequency. When the network is vacant, ImTCP will not conflict with other connections so much. In this case, TCP-compatibility does not strictly required, because ImTCP is TCP-friendly so that ImTCP will perform exactly like traditional TCP. Besides, the information about the vacancy in the network will be of interest. In this case, ImTCP should increase its measurement frequency. Thus, there should be a dynamic adjustment for the measurement frequency according to the network status. We will consider the problem in our future works.

V. Transmission Modes of ImTCP

Here we introduce two examples in which ImTCP controls the transmission using measurement results to obtain performance that is not realizable with traditional TCP.

A. Background transmission

The transmission for backup data or cached data (background traffic) should not degrade throughput of other traffic (foreground traffic), which may be more important. We introduce an example showing that ImTCP successfully uses the results of bandwidth availability measurements to prevent its own traffic from degrading the throughput of other traffic. We call this type of ImTCP data transmission *background mode*.

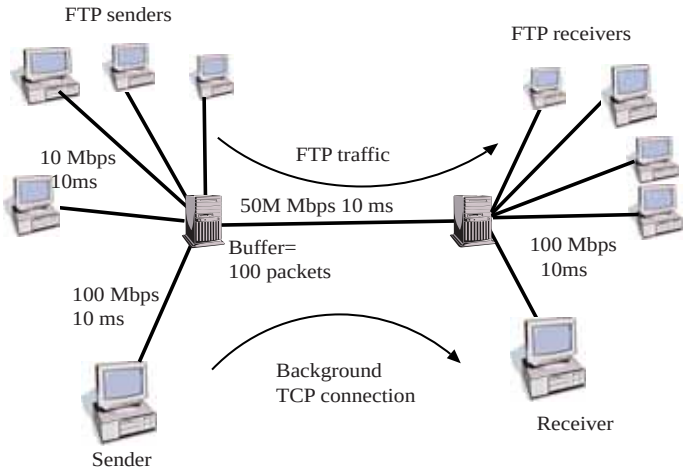


Fig. 21. Network model for evaluation of ImTCP background mode

The main idea is to set an upper bound on the congestion window size according to estimated values so that the transmission rate does not exceed the available bandwidth. This reduces the effect ImTCP has on other traffic in the same network links. We use the following control mechanism. When

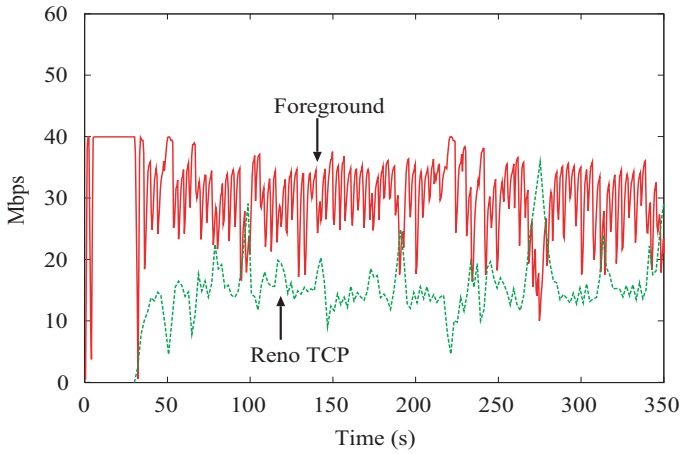
$$g \cdot RTT \cdot A > m \cdot N$$

we set

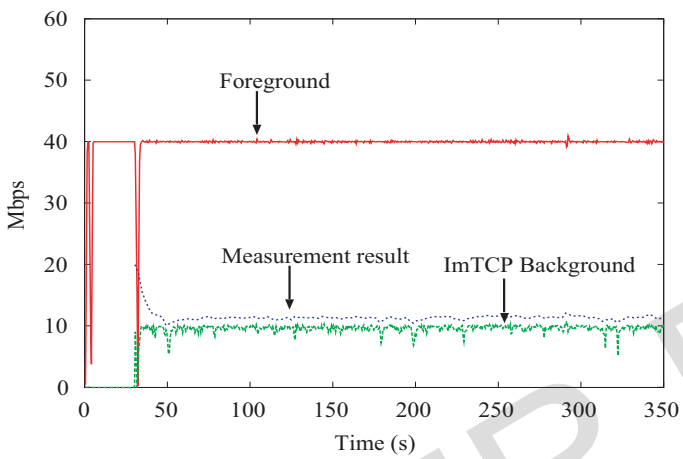
$$MaxCwnd = g \cdot RTT \cdot A$$

where A is the estimated value of available bandwidth, $MaxCwnd$ is the upper bound of the congestion window size and N is the number of packets for a measurement stream. The parameter g can range from 0 to 1. When g is small, ImTCP uses less bandwidth and interferes only very slightly with foreground traffic. When g is near 1, ImTCP uses more bandwidth and its effect on foreground traffic grows. We set the upper bound of the congestion window size ($MaxCwnd$) to $g \cdot RTT \cdot A$ only when the value is large enough for ImTCP to continue performing measurements well. We do not use the value to restrain the window size when it becomes smaller than $5 \cdot N$ (Note that up to four packet streams are needed for a measurement. Therefore, when the congestion window size is smaller than $5 \cdot N$, the measurement task becomes difficult).

We first examine the behavior of the background mode of ImTCP when the foreground traffic is persistent TCP connections. We generate foreground traffic from four TCP connections, each of which passes through a 10 Mbps bottleneck link before passing the 50 Mbps share link, as shown in Figure 21. Therefore, in case when there is no



(a) Reno TCP



(b) ImTCP background

Fig. 22. Comparison of ImTCP background and Reno TCP performance

background traffic, the total transmission rate in the foreground is approximately 40 Mbps, as can be seen for 0-30 sec in Figure 22. At 30 sec in the simulation, we activate the background flow through the shared link. The background traffic does not pass through any link with smaller bandwidth than the shared link. In the simulation, g is set to 0.9.

Figure 22 shows the change in throughput of foreground transfer and background transfer as a function of simulation time. In the case where we use Reno TCP for the background transfer, we find that the background traffic greatly affects the foreground, as can be seen in Figure 22(a), where the average throughput of foreground traffic becomes 30 Mbps and that of the background is 16 Mbps. When we use ImTCP background mode for background transfer, we find that ImTCP can exactly estimate the available bandwidth in the shared link (10 Mbps) and successfully prevent its transmission rate from exceeding the estimated values. The average throughput of the foreground traffic is 39 Mbps total and that of the background traffic about 9 Mbps (Figure 22(b)), which matches well with the setting of the value of g (=0.9).

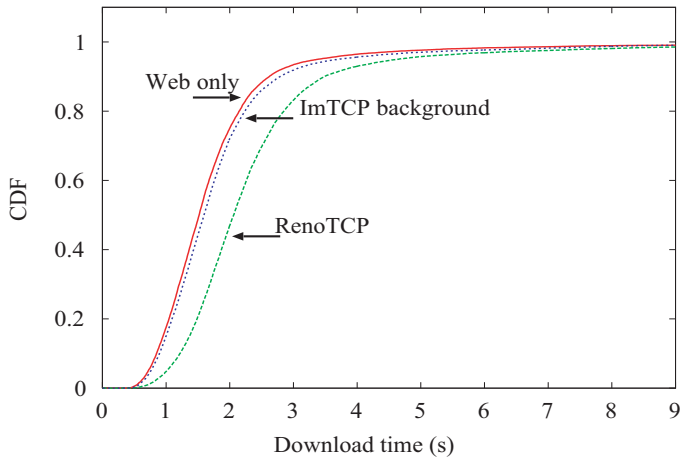


Fig. 23. Average of Web page download time

We also examine the behavior of ImTCP in background mode when foreground traffic is originated with Web document transfers. We replace the ImTCP connection in the simulation in Figure 14 with a background mode ImTCP connection. Figures 23 compare the download time for Web pages under ImTCP and Reno TCP. We find that ImTCP has only a very small effect on the download time of the foreground Web traffic. The average throughput of ImTCP in this case is about 72% that of Reno TCP. Figure 24 shows the measurement value and throughput of ImTCP connection as a function of simulation time in this case. Note that the throughput of ImTCP does not approach the actual value of available bandwidth. This indicates that ImTCP background mode is successfully avoiding interference with Web traffic. And we can also say that the measurement results in this case confirm that ImTCP in background mode performs the measurement very well.

B. Full-speed transmission

We introduce another example of a modified congestion control mechanism to show that ImTCP can enhance link utilization using its measurement results.

TCP considers packet loss events as indicators of bandwidth starvation and consequently decreases the transmission rate whenever packet losses occur. Therefore, in wireless networks where packets may be lost due to channel noise, TCP tends to use the available bandwidth ineffectively [31]. Similarly, in satellite networks and high-speed

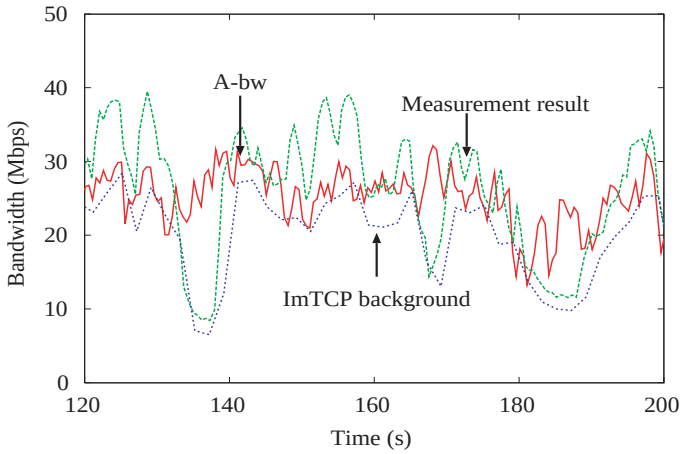


Fig. 24. Throughput and measurement result of ImTCP background mode

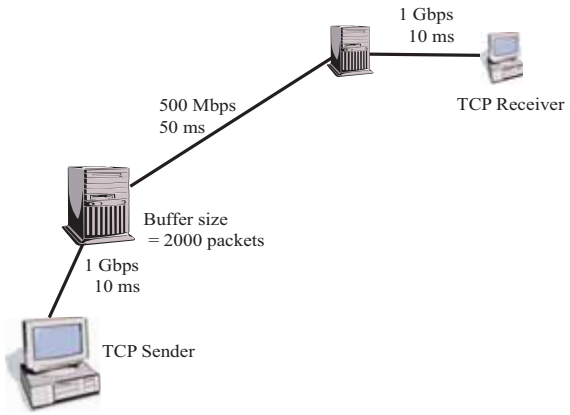


Fig. 25. High speed network topology

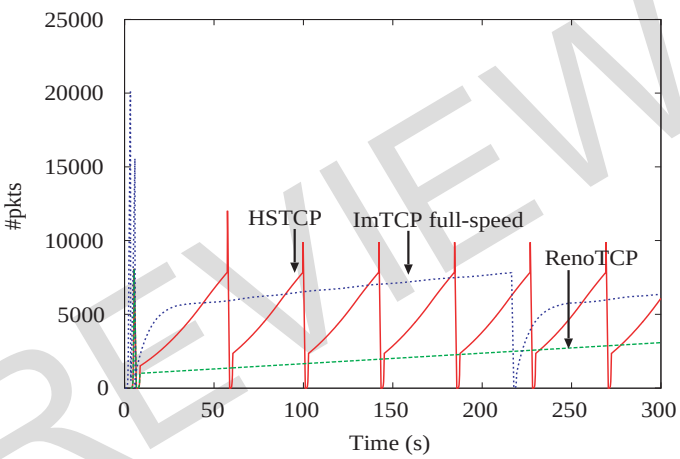


Fig. 26. Comparision of TCP window sizes

networks, where the bandwidth-delay product of an end-to-end path is extremely large, TCP with traditional congestion avoidance requires a long time and an extraordinarily low packet loss probability to fully utilize the link bandwidth [15].

To improve TCP throughput in abovementioned networks, we introduce an available-bandwidth-aware window size adjustment. The idea is to use the measurement result to adjust the increasing speed of the congestion window size. When the available bandwidth is large, the window size increases quickly to make full use of available bandwidth, and when the available bandwidth is small due to the existence of other traffic, the window size increases slowly. We call this type of ImTCP data transmission *full-speed mode*.

In the congestion avoidance phase, we do not increase the congestion window size ($Cwnd$) by one in every RTT. Instead, we use the following adjustment.

$$Cwnd \leftarrow Cwnd + max\left(1, h \cdot (1 - \frac{Cwnd}{V})\right)$$

$$V = A \cdot RTT$$

In the equation, h ($h \geq 1$) is a parameter that determines how fast the window size increases. If h is large, ImTCP can successfully utilize the bandwidth link, but it may encroach bandwidth share of other connections and cause unfairness in the network. When h is small or equal to 1, ImTCP behaves the same as Reno TCP.

We use the topology in Figure 25 to investigate the performance of ImTCP in full-speed mode. The ImTCP sender and ImTCP receiver is connected by two routers with Gigabit links. Therefore, the 500 Mbps link between the two routers becomes the bottleneck link in the path. We assume the buffer of the TCP receiver is large so the TCP throughput can achieve 500 Mbps. The buffer size of the router at the bottle neck link is also large (2000 packets). We compare the throughput obtained when using ImTCP in full-speed mode, High-Speed TCP (HSTCP) [15] and Reno TCP for data transmission in the network.

Figure 26 shows the changes in the window size of TCP connections. Reno TCP requires a long time to reach a large window size. HSTCP increases the window size quickly to fully use the free bandwidth, however, the increasing speed is non-sensitive to the available bandwidth such that packet loss events occur frequently. Therefore, overall, the throughput of HSTCP is not as large as expected. ImTCP increases the window size quickly when the window size is small and decreases the speed when its transmission rate reaches the available bandwidth to avoid packet losses. Therefore, the throughput of ImTCP is better than the others.

Finally, we compare the throughput of ImTCP in full-

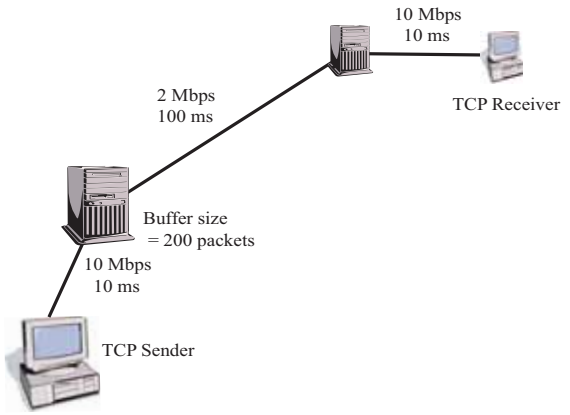


Fig. 27. Wireless network topology

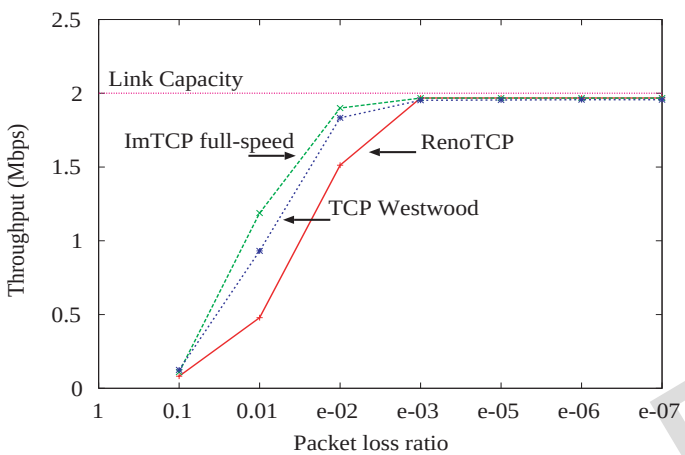


Fig. 28. TCP throughput in wireless network

speed mode with Reno TCP in a wireless network. We insert a 2 Mbps network link in the path between a TCP sender and TCP receiver to simulate a wireless link, as shown in Figure 27. We vary the packet loss rate of the network links and find that ImTCP can achieve a larger throughput than TCP Westwood and Reno TCP when the loss rate is high, as shown in Figure 28.

Parameter h is set to 100 in this case. When the packet loss rate is high, a higher value for parameter h can help ImTCP obtain higher available bandwidth. When the packet loss rate is low, the value of h should be low so that ImTCP will share bandwidth fairly with other traffic. We will investigate an appropriate adaptive control mechanism for h in future works.

VI. Conclusions

In this paper, we introduced a method for measuring the available bandwidth in a network path between two end hosts using an active TCP connection. We first constructed a new measurement algorithm that uses a relatively small number of probe packets yet provides periodic measurement results quickly. We then applied the proposed algorithm to an active TCP connection and introduced ImTCP, a version of TCP that can measure the available bandwidth. We evaluated ImTCP through simulation experiments and found that the proposed measurement algorithm works well with no degradation of TCP data transmission speed. We also introduced examples of ImTCP special transmission modes.

In future projects, we will develop new transmission modes for ImTCP as well as evaluate the performance of the modes introduced in this paper. We have implemented a tool based on the measurement algorithm and found that it yields results equal in accuracy to that of the simulations. We will also consider 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*, 2002.

[3] "The Internet Bandwidth Tester (TPTEST)," available at `http://tptest.sourceforge.net/about.php`.

[4] T. Oetiker and D. Rand, "Multi router traffic grapher," available at `http://people.ee.ethz.ch/~oetiker/webtools/mrtg/`.

[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] Srinivasan Seshan, Mark Stemm, and Randy H. Katabi, "SPAND: Shared passive network performance discovery," in *Proceedings of the 1st Usenix Symposium on Internet Technologies and Systems (USITS '97)*, Dec. 1997, pp. 135–146.

[7] Bob Melander, Mats Bjorkman, and Per Gunningberg, "A new end-to-end probing and analysis method for estimating bandwidth bottlenecks," in *Proceedings of IEEE GLOBECOM 2000*, Nov. 2000.

[8] Manish Jain and Constantinos 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] J.Strauss, D.Katabi and F.Kaashoek, "A measurement study of available bandwidth estimation tools," in *Proceedings of the Internet Measurement Conference*, 2003.

[11] N.Hu and P.Steenkiste, "Evaluation and characterization of available bandwidth probing techniques," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 6, Aug. 2003.

[12] "B. Gaidioz and R. Wolski and B. Tourancheau, "Synchronizing network probes to avoid measurement intrusiveness with the network weather service," in *Proceedings of the 9th IEEE Symposium on High Performance Distributed Computing*, 2000.

[13] Robert L. Carter and Mark E. Crovella, "Measuring bottleneck link speed in packet-switched networks," Tech. Rep. TR-96-006, Boston University Computer Science Department, Mar. 1996.

[14] H. Balakrishnan, N. Padmanabhan, S. Seshan and H. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 756–769, 1997.

[15] S. Floyd, "Highspeed TCP for large congestion windows," *RFC 3649*, Dec. 2003.

[16] 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*, Aug. 1994, pp. 24–35.

[17] Stefan Savage, "Sting: A TCP-based network measurement tool," in *Proceedings of USITS '99*, Oct. 1999.

[18] "Sprobe," available at `http://sprobe.cs.washington.edu`.

[19] 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*. 1996, vol. 26,4, pp. 270–280, ACM Press.

[20] Mark Allman and Vern Paxson, "On estimating end-to-end network path properties," in *Proceedings of SIGCOMM '99*, 1999, pp. 263–274.

[21] 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*.

[22] "Network Test," available at `http://www-didc.lbl.gov/NCS/netest/`.

[23] V. Ribeiro, M. Coates, R. Riedi, S. Sarvotham, B. Hendricks and R. Baraniuk, "Multifractal cross-traffic estimation," in *Proceedings of ITC Specialist Seminar on IP Traffic Measurement, Modelling, Management*, Sept. 2000.

[24] J.Strauss, D.Katabi and F.Kaashoek, "A measurement study of available bandwidth estimation tools," in *Proceedings of Internet Measurement Conference 2003*.

[25] NS Home Page, "`http://www.isi.edu/nsnam/ns/`," .

[26] "NLANR web site," available at `http://moat.nlanr.net/Datacube/`.

[27] Richard Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*, Addison-Wesley, 1994.

[28] A. Feldmann, C. Gilbert, P. Huang and W. Willinger, "Dynamics of IP traffic: A study of the role of variability and the impact of control," in *Proceedings of SIGCOMM '99*, 1999, pp. 301–313.

[29] R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*, Wiley-Interscience, 1991.

[30] S. Jin, L. Guo, I. Matta and A. Bestavros, "TCP-friendly SIMD congestion control and its convergence behavior," in *Proceedings of ICNP'01: The 9th IEEE International Conference on Network Protocols*, Nov. 2001.

[31] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang and M. Gerla, "The impact of multihop wireless channel on TCP throughput and loss," in *Proceedings of IEEE INFOCOM 2003*, Mar. 2003.