Master's Thesis

Title

A Study on Inline Network Measurement Mechanism for Service Overlay Networks

Supervisor

Prof. Masayuki Murata

Author

Cao Le Thanh Man

February 13th, 2004

Department of Information Networking

Graduate School of Information Science and Technology

Osaka University

Master's Thesis

A Study on Inline Network Measurement Mechanism
for Service Overlay Networks

Cao Le Thanh Man

Abstract

The performance of service overlay networks depends primarily on how well they take advantage of the characteristics and resources of the underlying network. To improve performance, therefore, service overlay networks need fast and accurate information about the IP network concerning resource availability, network congestion, routing and so on. In this thesis we propose ImTCP (Inline measurement TCP), a real-time inline measurement mechanism with very small overhead for measuring end-to-end bandwidth availability in service overlay networks. Utilizing the transmitted packets of an active TCP connection to perform measurements, the inline measurement mechanism of ImTCP has the advantage of yielding results quickly and accurately without using additional probe packets.

In this thesis, 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 TCP through a modification to the TCP sender only. We also present a system we designed for storing measurement results from which an application can retrieve results data for any time scale. In addition, we present examples in which TCP data transmission performance is improved by using the measurement results.

We show through extensive simulation results that the inline measurement mechanism successfully measures bandwidth availability in the end-to-end path at intervals of several RTTs while

exhibiting no degradation in TCP transmission speed. We also show how measurement results can be used to prevent TCP data transmission from affecting other network traffic and how available bandwidth can be utilized more successfully than with conventional TCP.

Keywords

TCP (Transmission Control Protocol), Available bandwidth, Inline measurement, Sender-based measurement, Service overlay network

Contents

List of Figures

1 Introduction

Network measurement techniques have received a great deal attention, and numerous measurement tools have been developed [1–10]. These tools observe and/or monitor network characteristics such as the physical link bandwidth [5–8], available bandwidth [1–3], delay [7], loss [9] and topology [10] of the network. The observed results are often used for network trouble-shooting, isolation of fault locations, and network provisioning [11]. Measurement techniques can be categorized into *active* and *passive* approaches. Active approaches [1–4, 7–10] inject test packets into the network, and utilize the feedback information to derive measurement results. Passive approaches [5, 6] do not use test packets but rather monitor the packets already traversing a router interface or a link.

As the Internet increasingly diversifies and the user population grows rapidly, new and varied types of service-oriented networks are emerging. Called *service overlay networks*, they include Peer-to-Peer (P2P) networks [12, 13], Grid networks [14–17] and Content Delivery/Distribution Networks (CDNs) [18–20] and IP-VPNs [21]. Service overlay networks are upper-layer networks providing special-purpose services built onto the lower-layer IP network. Their performance depends primarily on how well they take advantage of the characteristics and resources of the underlying network. To improve performance, therefore, service overlay networks need fast and accurate information concerning resource availability in the IP network to realize adaptive control mechanisms. Some examples of this are:

- P2P networks. When a resource discovery mechanism finds multiple peers having the same requested contents, this information is used to determine which peer should transmit the contents.

- Grid networks. When multiple sites contain the same data, this information is used to determine from which site data will be copied or read.

- CDNs. When backup data or cached data is transmitted, this information can be used to prevent other network traffic from being deprived of resources during the transmission.

Our study focuses on measuring *available bandwidth* in the network path of a service overlay network. Many approaches for measuring bandwidth availability, including active and passive, have been presented in the previous literature ([3, 5, 22] and references therein). However, existing measurement algorithms cannot be used directly to measure the network characteristics of a service overlay network. One reason is that active measurement approaches, utilizing probe packets with continuous measurements, degrade the performance of the other network traffic. In addition, existing measurement techniques, especially the passive approaches, require a relatively long time to collect each measurement result. We must collect as quickly as possible the latest, and hence most accurate, information on network characteristics when the volume of IP network traffic fluctuates greatly.

Based on these considerations, we propose a technique for measuring end-to-end network bandwidth availability that satisfies requirements for service overlay networks. Our method is based on existing active measurement approaches of PathLoad [3] and PathChirp [22], but does not require the sending of additional packets. Instead, information on network characteristics is collected from transmitted packets of an existing TCP service connection. We call this approach *Inline Network Measurement*. The rational behind this is to change TCP from a data-transfer-only tool to one that can also be used for measurement. The method proposed here is a sender-based approach; it does not require changes to the TCP receiver.

Traditional TCP can be considered to some extent as a tool for measuring available bandwidth because of its ability to adjust the congestion window size until the transmission rate is relevant to the available bandwidth. However, the resulting estimation is insufficient and inaccurate because it is a measure of *used* bandwidth, not *available* bandwidth. Especially, in networks where the probability of packet loss is relatively high, TCP tends to fail in estimating the available bandwidth.

Moreover, the TCP sender window size often does not accurately represent the available bandwidth due to nature of the TCP congestion control mechanism. A study by TCP Westwood [23] presents an improved TCP bandwidth measurement mechanism in which the TCP sender uses the arrival intervals of ACK packets to estimate the available bandwidth and control the transmission rate accordingly. It is a good measurement algorithm in terms of simplicity. However, it tends toward underestimation when the available bandwidth of the network path transitions from a low to a high value. This is due to the fact that when available bandwidth suddenly increases, the TCP data transmission rate, due to the self-clocking phenomenon of TCP, does not change as quickly and needs time to ramp up. Meanwhile, as the transmission rate continues at a rate lower than the available bandwidth, the measurement algorithm yields results lower than the real values. This shortcoming arises from the fact that Westwood TCP, as with traditional TCP, measures the used bandwidth, not the available bandwidth.

In this paper, we first introduce a measurement algorithm suitable for the inline network measurement. This algorithm periodically yields measurement results at short intervals such as 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 out the physical bandwidth [3, 22], from 0 bps to the upper limit of the range, in every measurement as existing algorithms do. 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 in which the proposed algorithm for inline network measurement described above is included. When a sender transmits data packets, ImTCP first stores a group of five to 10 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

similar to the RTO (round trip timeout) calculation in TCP to regulate packet storage time in the queue. We evaluate the inline measurement system with simulation experiments. The results show the proposed algorithm works with the window-based congestion control algorithm of TCP without degrading transmission throughput.

We also propose some modifications to the socket interface of traditional TCP so that upper-layer applications can control ImTCP behavior and access ImTCP measurement results. We propose a storage mechanism and new API allowing retrieval of measurement results for any given time interval by applications. We also present two examples using ImTCP congestion window control modes to show how measurement results can be applied to TCP data transmission. In *background transfer mode*, ImTCP uses bandwidth availability measurement results to prevent its own traffic from degrading the throughput of other traffic. This allows priority to be given to the other traffic sharing the network bandwidth. In *full-speed transfer mode*, ImTCP uses measurement results to keep its transmission rate close to the measured value necessary for optimum utilization of available network bandwidth. This mode is expected to be used in wireless and high speed networks where traditional TCP cannot utilize the available bandwidth effectively [24, 25].

The remainder of this paper is organized as follows. In Section 2, we discuss requirements for network measurement in service overlay networks and the problems found with existing network measurement methods. We introduce the proposed algorithm for inline network measurement and evaluate it. In Section 3, we introduce ImTCP (Inline measurement TCP) and evaluate its performance. In Section 4 we discuss some of the implementation issues of ImTCP. In Section 5, we introduce two examples of congestion window control mechanisms for ImTCP. And finally in Section 6, we present concluding remarks and discuss future projects.

2 Inline Network Measurement in IP-based Service Overlay Networks

2.1 Requirements

In accordance with the discussion in Section 1, we consider the following factors to be the requirements of the measurement algorithm of inline network measurement:

- Small number of packets used

  Because our method uses TCP packets for the measurement, there is a limitation on the number of packets available for transmission at any one time because of the TCP window size. Since the TCP window size is relatively small and changes dynamically, the measurement algorithm should use as small a number of packets as possible.

- Little effect on other traffic on the network

  Since the goal of measurement is to improve the quality of services of the service overlay network, the measurement should not affect either the traffic of the supported services itself or the external traffic. The measurement may adversely affect the network in two ways: by sending numerous probe packets and by sending probe packets at a high rate.

- Providing results continuously

  Since the characteristics of the IP network changes constantly and dynamically, measurement should provide periodic estimation results. Furthermore, the interval should be as small as possible in order to provide an accurate depiction of the rapid network change.

- Providing results quickly

  The measurement should be performed quickly in order to obtain up-to-date information of the IP network. In the proposed method, we therefore assign a higher priority to measurement speed than to measurement accuracy.

2.2 Existing network measurement methods

As mentioned in Section 1, the existing measurement methods can be divided into two groups: passive measurement methods and active measurement methods. The passive methods, represented by SPAND [5] and Nettimer [6], observe passing traffic at some certain points in the network and use the monitored information to obtain the measurement results. Although these approaches are good in terms of not affecting other traffic, they require quite a long time to gather information for accurate measurement results and many measurements are necessary in order to estimate the characteristics of the end-to-end path. Furthermore, passive approaches cannot provide high-accuracy measurement results because the available information is very limited.

On the other hand, the active measurement methods inject probe packets into the network and collect the feedback information from monitored results including transmission delay, packet arrival-interval time, packet loss ratio and so on. Therefore, we can expect a higher accuracy of measurement results in an end-to-end fashion than what is possible by passive methods. Cprobe [1], Topp [2], and Pathload [3], are representative tools to measure the available bandwidth of the network path between two endhosts. These algorithms work on endhosts and require no change inside the network, so they seem suitable for application to measurement in service overlay networks. However, these algorithms also have fundamental disadvantages. One is that many probe packets are sent at a high transmission rate. For instance, Topp sends 5000 packets to obtain only one measurement, and Cprobe injects 100-200 probe packets at the physical bandwidth speed of the link connected to the sender host. PathLoad sends several 100-packet measurement streams for a measurement. PathChirp [22] is a modification of PathLoad on the purpose of decreasing the number of probe packets. But the required number of packets to be sent at one time in PathChirp is still large. The probe traffic can affect other traffic along the path, for example by degrading traffic throughput and increasing the packet loss ratio and packet transmission delay. Existing active measurement algorithms also require a long time to obtain one measurement result (for

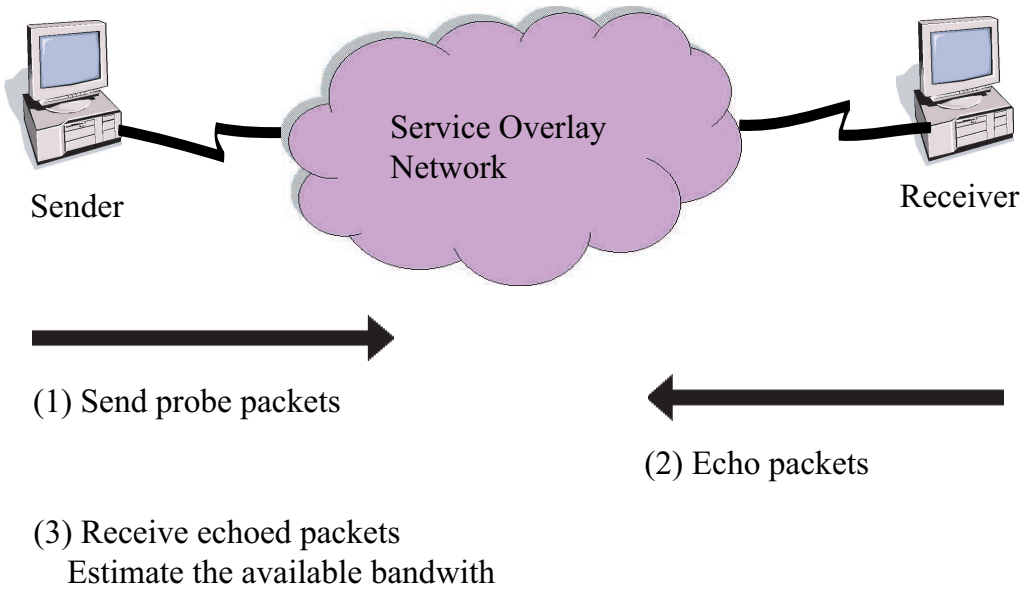


Figure 1: Outline of proposed measurement algorithm

example, 50-100 RTTs are necessary to obtain one estimation value in Topp and Pathload). Long-term measurement can provide an accurate result but cannot follow the dynamic changes on the IP network.

Thus, the existing active measurement algorithms do not satisfy the requirements mentioned in Subsection 2.1. In the next subsection, we introduce a measurement algorithm which satisfies the requirements. Note that we do not attempt to replace the existing active measurement approaches by the proposed measurement algorithm. Rather, the proposed measurement algorithm is useful in the inline network measurement.

2.3 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 measurements. 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

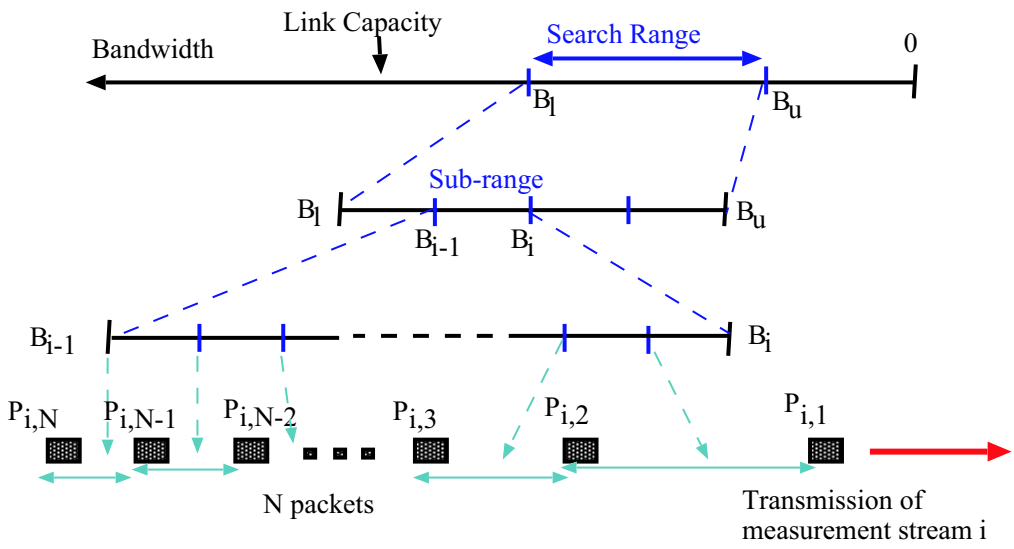


Figure 2: Relationship of search range, sub-ranges, streams, and probe packets

We first send a packet stream according to the Cprobe algorithm [1] 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,

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

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 $P_{i,j}$ $(1 \leq j \leq N$, where N is the number of packets in a stream) and the time at which $P_{i,j}$ 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 [3], 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, as described in Subsection 2.1.

We then observe $R_{i,j}$, the time the packet $P_{i,j}$ 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 [3]. As explained in [3], 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:

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

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.

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 $P_{a,j}$ ($j = 2 \ldots N$) 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

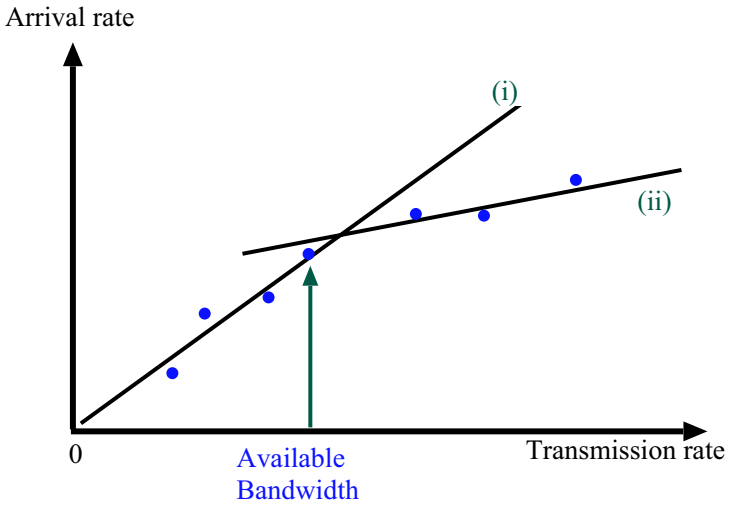


Figure 3: Finding the available bandwidth within a sub-range

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:

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

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.

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

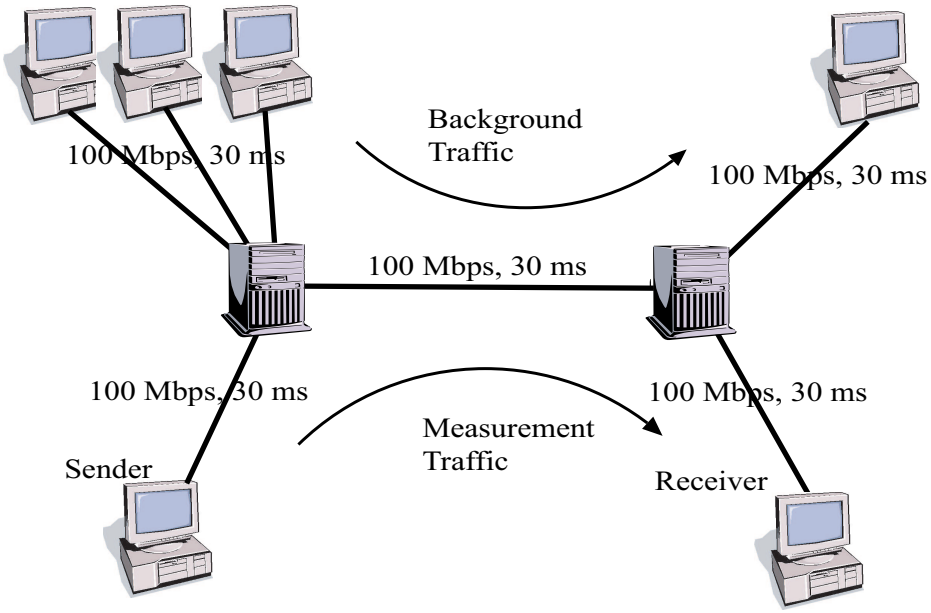


Figure 4: Network model for evaluation of the proposed measurement algorithm

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.

2.4 Simulation results

This Subsection shows some simulation results in ns [26] and validates the measurement algorithm proposed in Subsection 2.3. 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 [27]. The correct value of the available bandwidth of

the bottleneck link is calculated as:

$$Bottleneck\ link\ capacity - Total\ rate\ of\ background\ traffic$$

We make the available bandwidth on the bottleneck link fluctuate by changing background traffic rates.

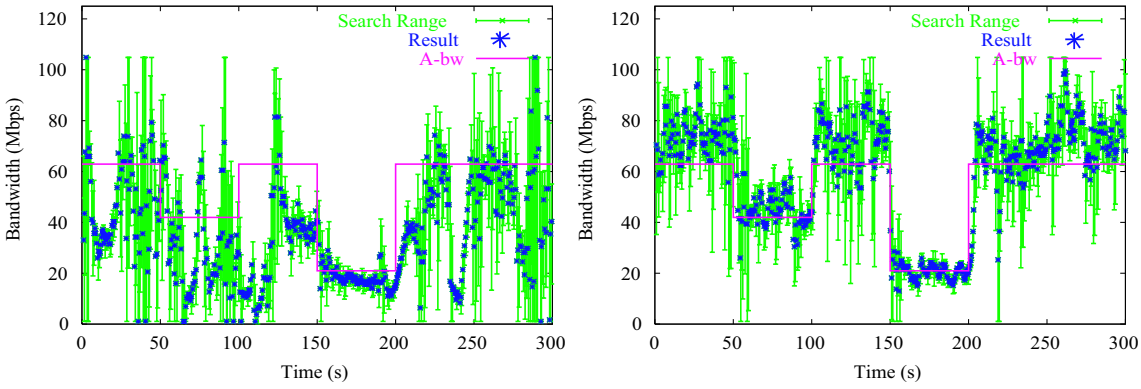
The sender host sends probe packets to the receiver host and the receiver host echoes the packets back to the sender. The sender, using the algorithm proposed in Subsection 2.3 , measures the available bandwidth of the path between the two hosts. In this situation, the result corresponds to the available 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};

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

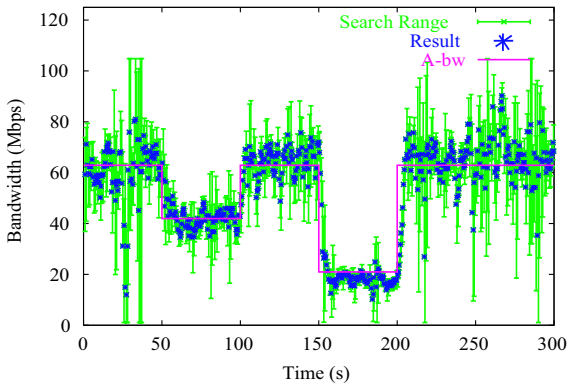
B_m, the lower bound of the width of search ranges, is set to 10% of A_{prev}. The probe packet size is 1500 Bytes.

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



(a) N=3

(b) N=5

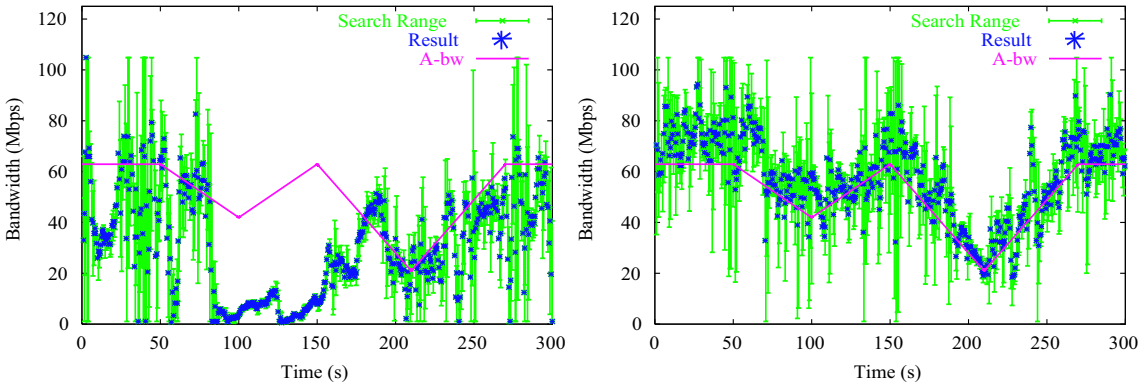


(c) N=8

Figure 5: Results of the proposed measurement algorithm (1)

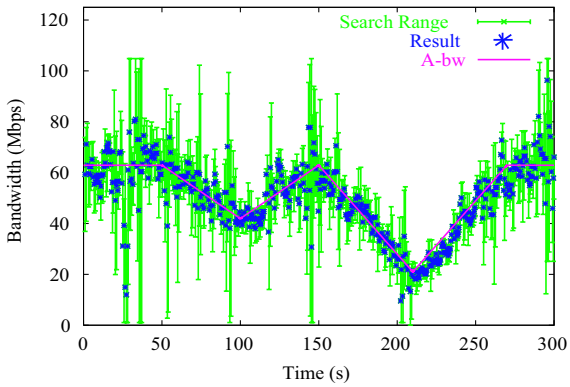
determine the increasing trend of the streams correctly in Step 3 in the proposed algorithm, which leads to the incorrect choice of sub-range in Step 4. Although the accuracy of measurement results increases as N is increased from 5 to 8, N=5 is judged to be the better setting since we place a higher priority on measurement speed than on measurement accuracy, as described in Subsection 2.2.

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



(a) N=3

(b) N=5



(c) N=8

Figure 6: Results of the proposed measurement algorithm (2)

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

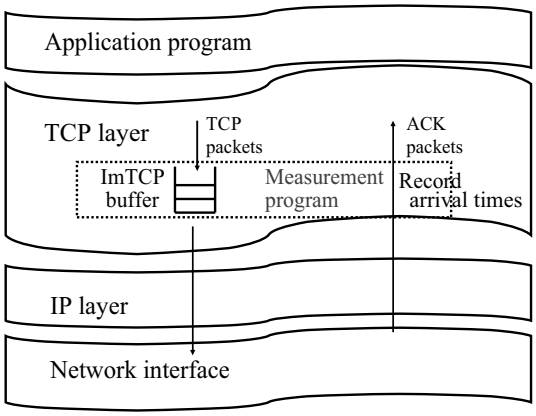


Figure 7: Placement of measurement program at TCP sender

3 ImTCP: TCP with Inline Network Measurement

3.1 Overview

In traditional TCP, a sender transmits data packets to a receiver and the receiver sends corresponding ACK packets back to the sender. Based on this characteristic, we apply the measurement algorithm proposed in Section 2 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

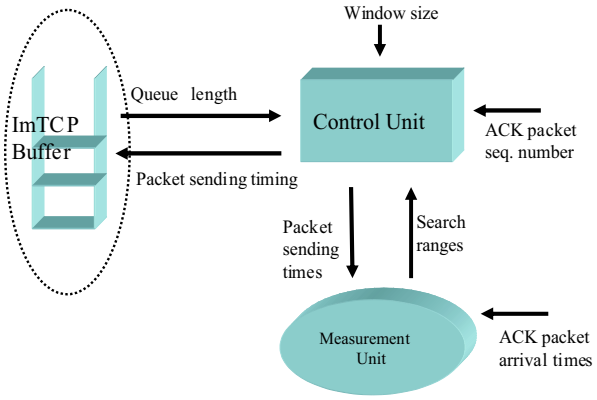


Figure 8: Structure of the measurement program

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 for a measurement with every RTT. That is, one measurement result per RTT can be obtained if the window size is large enough. When the window size is small, ImTCP may create only one stream per RTT with the result that several RTTs are required for one measurement.

3.2 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

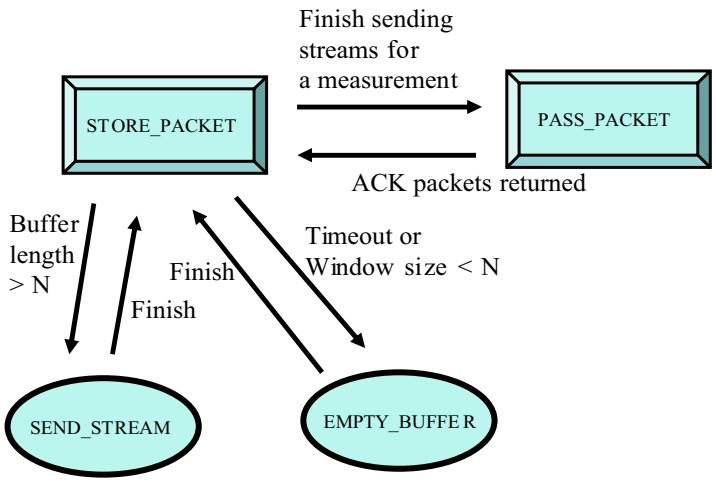


Figure 9: State transition in the Control Unit

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 2.3. 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. The timer value is discussed in Subsection 3.3.
  - Go to the EMPTY BUFFER state if the current TCP window size becomes smaller than N or the packet storing timer expires. N, as mentioned in Subsection 2.3, is the number of packets needed to create a measurement stream.

  - Go to the SEND STREAM state if the number of stored packets becomes larger than N.
  - Go to the PASS PACKET state if all packet streams for the current measurement have been sent.

- **EMPTY BUFFER** state
  - Clear the timer if it is set.
  - Pass currently stored packets to the IP layer until the buffer becomes empty.
  - Return to the STORE PACKET state.

- **SEND STREAM** state
  - Clear the timeout if it is set.
  - 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.
  - Go to the STORE PACKET 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.

Figure 10 shows an example of how ImTCP sends packets for a measurement operation. The change in queue length (the number of packets stored in the ImTCP buffer) is also shown in

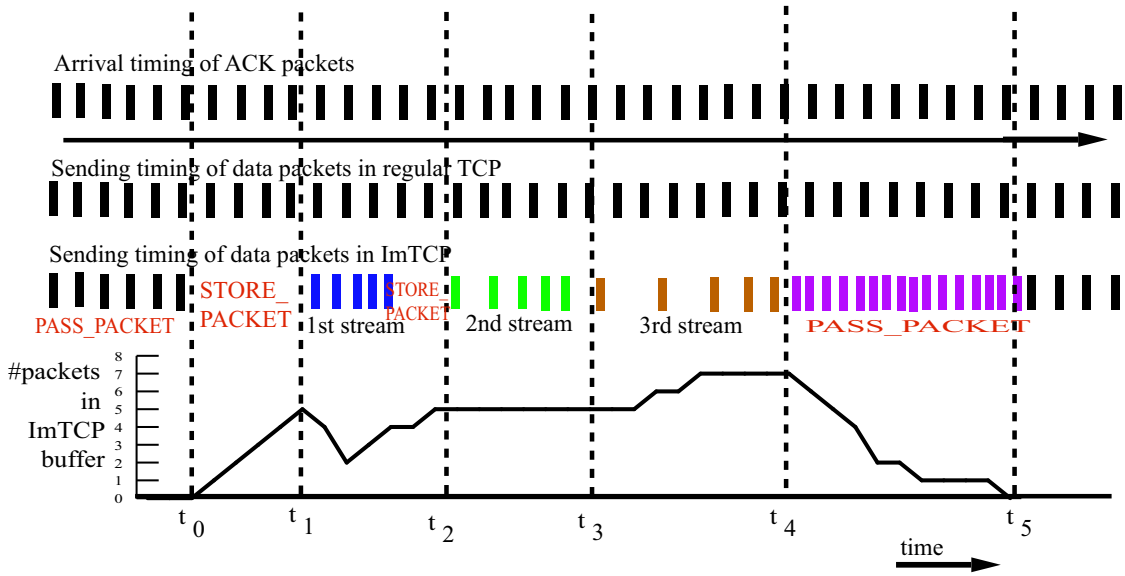


Figure 10: ImTCP packet transmission

the figure. We assume that TCP sends a new data packets immediately after receiving an ACK packet. Before t_0, ImTCP transmits packets in the normal TCP fashion since ImTCP is in the PASS PACKET state. From t_0, ImTCP moves to the STORE PACKET state. In this state, ImTCP data packets are stored and the queue length increases. At t_1, the queue length reaches N, set to 5 in this example, and ImTCP moves to the SEND STREAM state. The first packet stream is then sent from t_1. After sending the first packet stream, ImTCP reenters the STORE PACKET state. At t_2, the queue length again reaches N and ImTCP sends the second stream until t_3. Note that the average transmission rate of the second stream assumes the same ACK arrival rate, so the queue length remains constant from t_2 to t_3. Therefore, the third stream starts sending immediately after the second one, at t_3. At t_4, when the third stream is completely sent, ImTCP returns to the PASS PACKET state (here, we have assumed that the measurement requires only three packet streams). At t_4 some packets still remain in the buffer, so ImTCP may transmit those packets at a high rate for some time in the period from t_4 to t_5 until the buffer becomes empty. From t_5, ImTCP returns to sending packets in normal manner.

3.3 Parameter settings

3.3.1 Packet storing timer

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 [28].

If we assume a normal distribution of packet RTTs with average A_{RTT} and variance D_{RTT} , then A_{RTT} and D_{RTT} can be inferred from the TCP timeout function [28]. 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 11. 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:

$$Z = Y - X \tag{1}$$

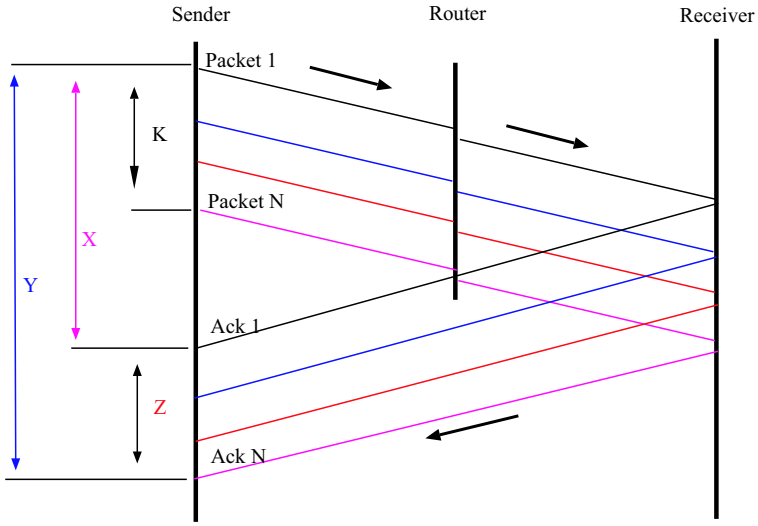


Figure 11: Packets transmission times in TCP

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 (1) we then obtain the distribution of Z, as $N(K, 2 \cdot D_{RTT})$.

Obviously, the ImTCP sender can record the departure times of data packets to calculate the exact value of K. However, this will introduce additional overhead and a waste of memory. 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

$$K = \frac{M}{A}(N - 1) \tag{2}$$

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 (2), we determine the waiting time

for N ACK packets as below:

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

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.

3.3.2 Number of packets in a measurement stream

In general, some packets will remain in the ImTCP buffer after sending streams for a measurement, as shown in Figure 10. If the number is large, ImTCP transmission delays may occur. We investigate the relationship between the number of remaining packets and N, then determine an appropriate value of N to minimize leftover packets. ImTCP begins sending a new packet stream when the queue length (the number of packets in ImTCP buffer) becomes larger than N. If the transmission rate of the packet stream is smaller than the arrival rate of ACK packets, the queue length increases during stream transmission. This fact can be seen for the third stream of Figure 10.

If we examine the case when the upper bound of the search range is smaller than the ACK arrival rate we can see that all measurement streams are transmitted at a rate lower than ACK arrival rate. Therefore, the queue length is always increasing during the measurement. The effect of N on the number of packets left in the ImTCP buffer after a measurement is most obvious in this case.

Suppose that $R_a > B_u$ where R_a is the arrival rate of the ACK packets and (B_l, B_u) is the search range mentioned in Subsection 2.3. Using the notation of Subsection 2.3, T_i, the time

needed for packet stream i to be transmitted, is

$$\begin{aligned} T_i &= \sum_{j=2}^{N} S_{i,j} - S_{i,j-1} \\ &= \sum_{j=2}^{N} \frac{M}{B_{i+1} + \frac{B_i - B_{i+1}}{N-1}(j-1)} \end{aligned}$$

During the period T_i, the number of ACK packets arriving at the sender is $R_a \cdot T_i$. Since ImTCP sends $N-1$ packets of the packet stream (excluding the first one at the beginning of the period), the packets entered into the ImTCP buffer during period T_i becomes $R_a \cdot T_i - (N-1)$. When all streams have been transmitted, the number of packets remaining in the ImTCP buffer ($Q\,L$) is

$$Q\,L = \sum_{i=1}^{k} \left(\sum_{j=2}^{N} \frac{M \cdot R_a}{B_{i+1} + \frac{B_i - B_{i+1}}{N-1}(j-1)} - (N-1) \right) + N$$

where k is the number of packet streams for one measurement operation. Since ImTCP starts to send the first stream when the buffer length reaches N, N is added at the right side of the equation. Note that

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

Therefore,

$$\begin{aligned} Q\,L &= \sum_{i=1}^{k} \left(\sum_{j=2}^{N} \frac{M \cdot R_a}{B_u - \frac{B_u - B_l}{k} \cdot i + \frac{B_u - B_l}{k} \cdot \frac{j-1}{N-1}} - (N-1) \right) + N \\ &= \sum_{i=1}^{k} \left(\sum_{j=2}^{N} \left(\frac{M \cdot R_a}{B_u - \frac{B_u - B_l}{k} \cdot i + \frac{B_u - B_l}{k} \cdot \frac{j-1}{N-1}} - 1 \right) \right) + N \end{aligned}$$

From the equation, we can see that when N becomes large, the number of packets left in the ImTCP buffer also becomes large. Therefore, we conclude that N should be set to as small value as possible.

According to the results of Subsection 2.4 is the smallest number of N that the measurement algorithm can accept. Therefore, unless explicitly stated otherwise, we will use N = 5 in the experimental simulations that follow.

3.3.3 Measurement frequency

As explained in Section 2, the measurement algorithm uses the previous measurement results to determine a search range for the next measurement. Therefore, it seems 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 in one RTT as described in Subsection 3.3.2, and so on. We therefore decided that ImTCP should perform at most one measurement operation per RTT. One RTT is considered long enough for ImTCP to recover the transmission rate after a measurement.

3.4 Other issues

3.4.1 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 2.3 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 arrival intervals $(S_{i,2j'+2} - S_{i,2j'})$ $(1 \leq j' \leq \lfloor N/2 \rfloor)$ for the probe 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.

3.4.2 Effect of packet fragmentation

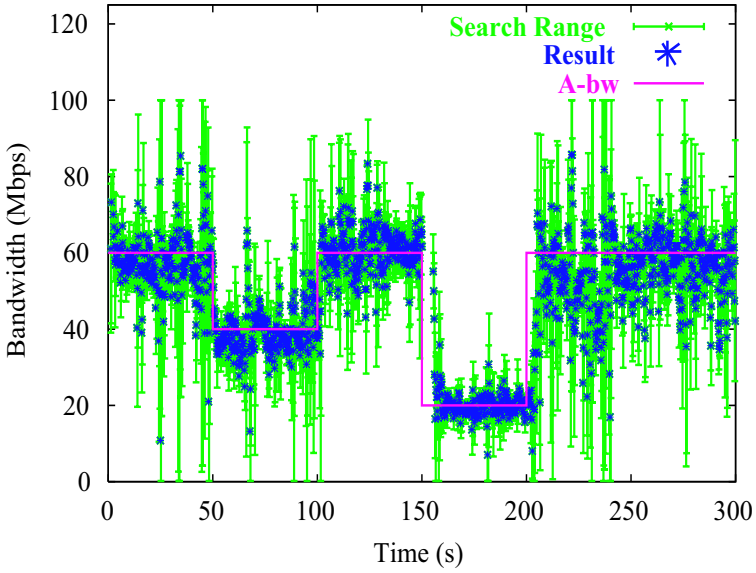
In the case where TCP packets are transmitted through a queue or node for which the MTU (Maximum Transmission Unit) is smaller than the packet size, the packets will be fragmented into several pieces in the network. The problem here becomes a question of whether measurement result will still be accurate 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.

3.5 Simulation results

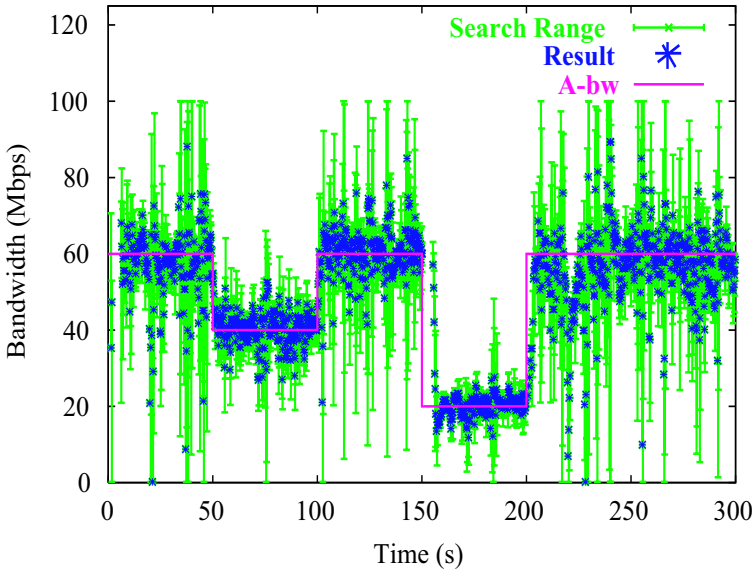
3.5.1 Measurement accuracy

Our first simulation uses the same topology as described in Subsection 2.4 except that the UDP sender and receiver are replaced by an ImTCP sender and an ImTCP receiver, respectively. Figures 12(a) and 12(b) show the measurement results of the proposed method when the number of packets (N) in a measurement stream is five and eight, respectively. Comparing Figure 12(a) with Figure 5(b) and Figure 12(b) with Figure 5(c), we observe that our measurement method can be successfully applied to TCP with no degradation in measurement accuracy. Also, the effect of N on the accuracy of measurement results is the same as in the case of Figure 5. That is, N=5 is again found to be a good setting.

Figure 13 shows the change in throughput of ImTCP in this simulation. For comparison, we also show the case of RenoTCP under the same network conditions. From the figure, we can see



(a) N=5



(b) N=8

Figure 12: Measurement results of ImTCP

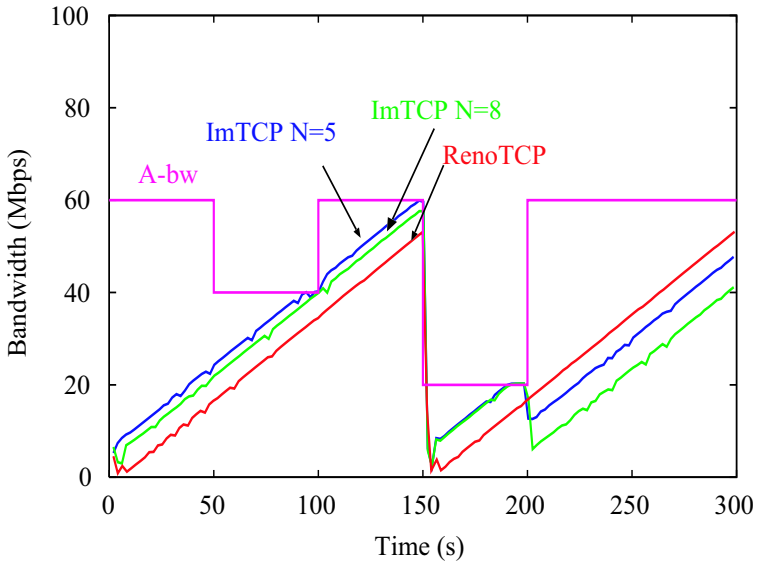


Figure 13: Throughput of ImTCP and RenoTCP

that ImTCP performs the measurement with a throughput almost the same as that of RenoTCP. An important point we can take 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 50 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.

3.5.2 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 RenoTCP using the network model depicted in Figure 14. We activate ImTCP and RenoTCP in turn in the network involving a large number of active Web document 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 [29], 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

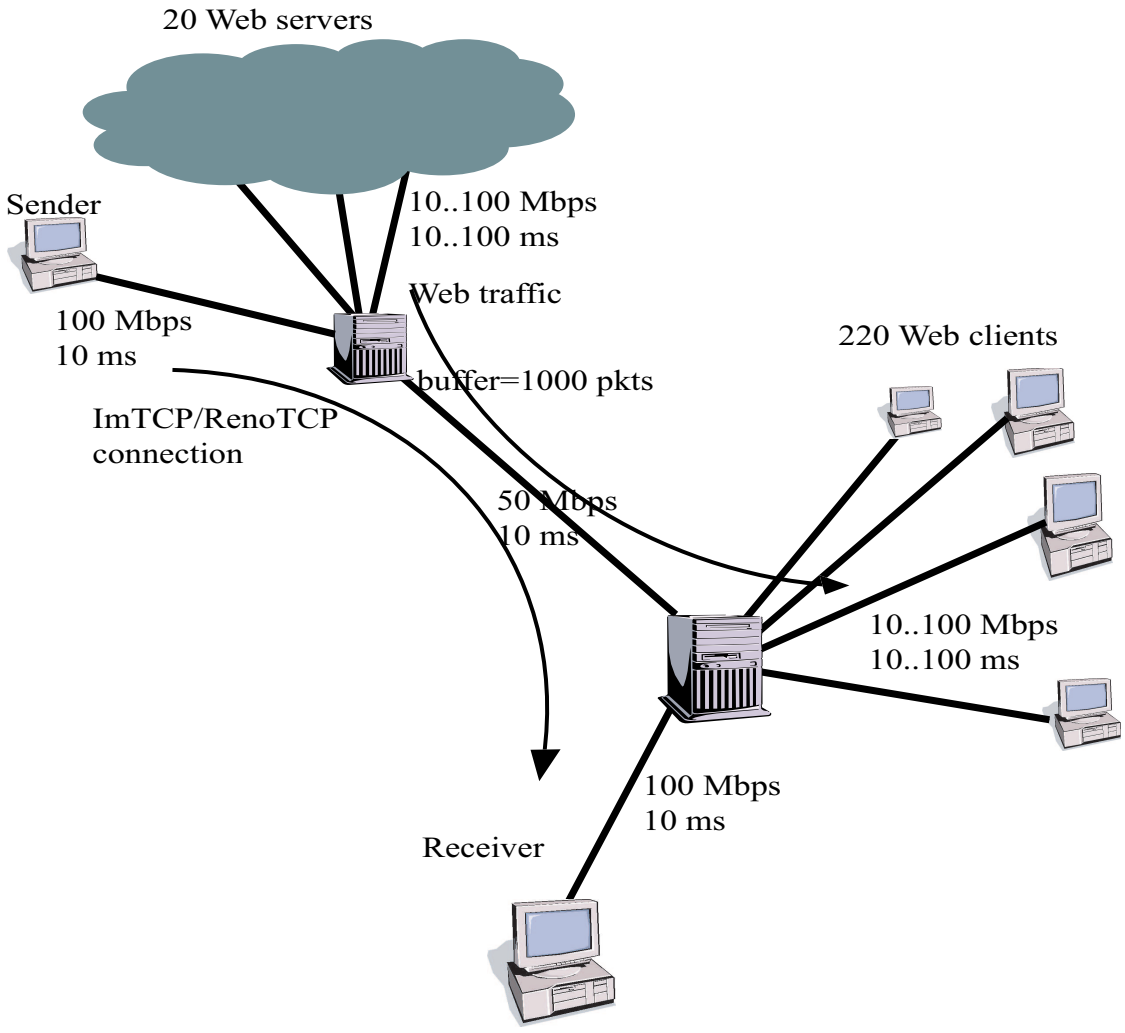


Figure 14: Network model for evaluation of ImTCP's effect on Web traffic

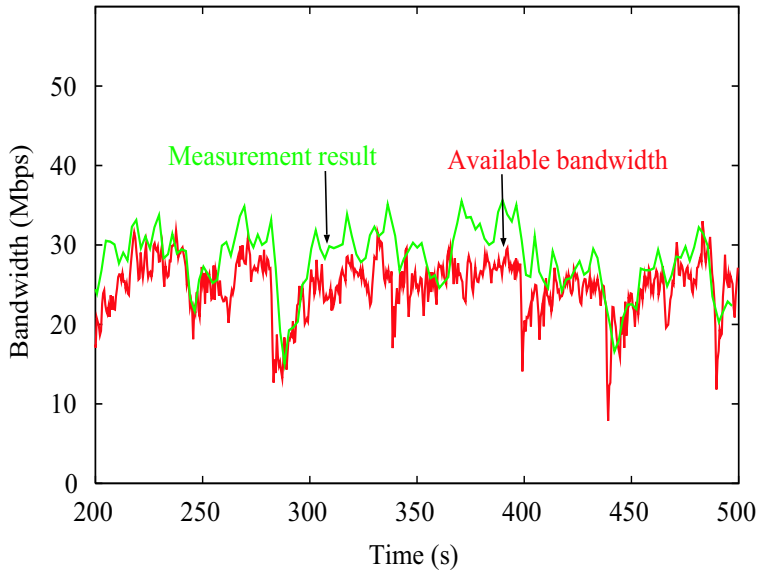


Figure 15: Measurement result of ImTCP in Web traffic environment

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/RenoTCP connections achieve high throughput because, here, the effect of ImTCP/RenoTCP 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 22.3 Mbps while that of RenoTCP 23.1 Mbps. The results therefore show that data transmission speed of ImTCP is almost the same as that of RenoTCP. Measurement results of ImTCP are also shown in Figure 15. In the figure, we round the measurement results using this function:

$$Value = 0.6 \cdot average\,of\,previous\,values + 0.4 \cdot current\,value$$

Figure 15 confirms that the ImTCP measurement result reflects the change in available bandwidth very well.

We compare the effect of ImTCP and RenoTCP 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 RenoTCP 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

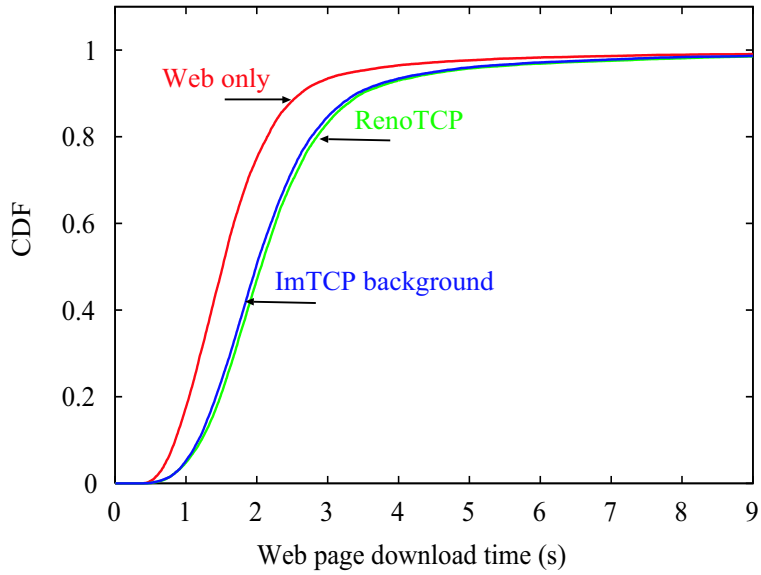


Figure 16: Comparison of Web page download times

with ImTCP.

3.5.3 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 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.

Figure 18(a) shows the Jain's fairness index [30] for the 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 RenoTCP. We can

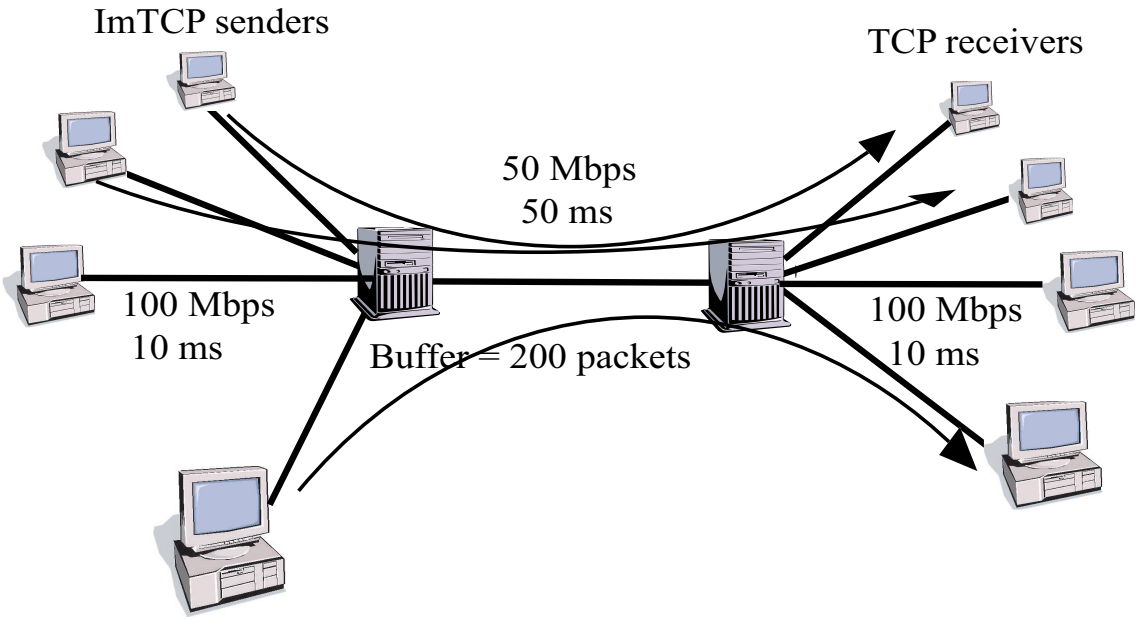
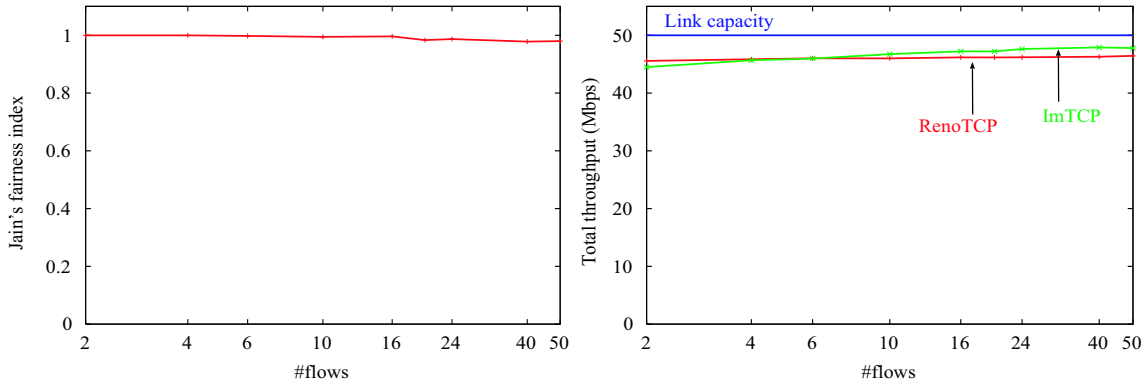


Figure 17: Network model for investigating bandwidth utilization and fair share

see that ImTCP and RenoTCP have almost the same link utilization regardless of the number of connections. Due to the small buffer size of the bottleneck link, when the number of connections are small the total throughput is not very high. However, when the number of connections is large, total throughput increases.

We next replace half of the ImTCP connections with RenoTCP connections in the above simulation and calculate the fairness index of each connection. Figure 19 shows the results. We also show the results when N is set to eight. The results show that ImTCP may not share fairly with RenoTCP. The reason for this is that ImTCP connections, due to its packet storing-and-forward mechanism, may lose bandwidth when conflicting with RenoTCP connections. The unfairness between ImTCP and RenoTCP may limit the scope where ImTCP can be used. We argue that the inline measurement mechanism may limit the application of TCP in some cases, but can enhance TCP performance in other environments where traditional TCP is not effective. In Section 5 we present two examples where TCP performance is improved with the inline measurement



(a) Fairness between ImTCP conections

(b) Link utilization of ImTCP

Figure 18: Fairness and link utilization of ImTCP

mechanism.

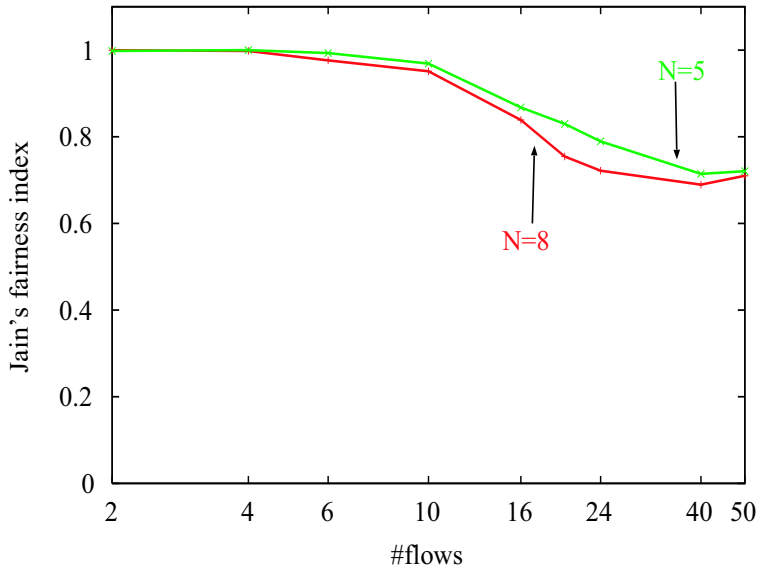


Figure 19: Fairness between RenoTCP and ImTCP

4 Implementation Issues

4.1 Storage system for measurement results

4.1.1 Time scale problem

Available bandwidth is a dynamic value, that is, the value depends strongly on its time scale. Depending on the characteristics of applications in the overlay network, the required time scale for measurements results varies. To be generally useful, ImTCP must provide measurement results for any time scale that applications require. As explained in Section 3, ImTCP can yield a measurement result for several RTTs. In other words, several RTTs is the smallest measurement time scale ImTCP can provide. The problem, however, is that it is impossible to store every measurement result when building values for larger time scales due to limitations in memory space and data processing time. In the following discussion, we first define the requests for measurement results that an application may make to ImTCP. We then introduce a data storage system that can solve the abovementioned problems.

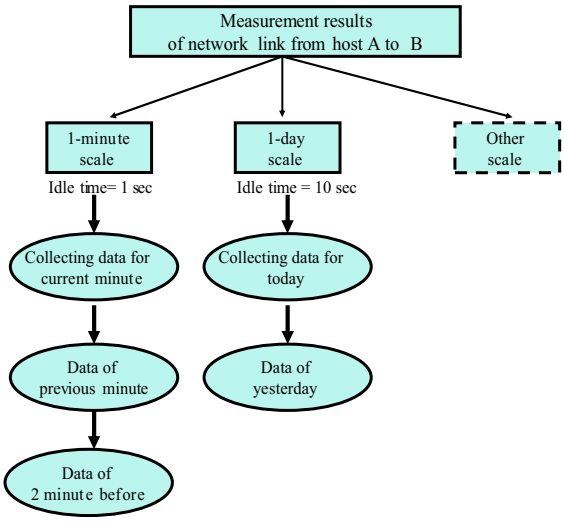


Figure 20: ImTCP storage database for measurement results

4.1.2 Application request

We assume an application will provide the following information with every request for measurement results.

- Time scale: the time interval between two measurement results.
- Number of measurement results.

For example, a replica selection service in a data Grid network may request for measurement results in the past 1 hours, in every 1 minute (time scale=1 minutes, number of results=60), or an ISP traffic investigation center may request measurement results of every days in the past 5 days (time scale=1 day, number of results=5).

4.1.3 Storage system

Because the most recent requests have a high probability of being repeated, the system stores measurement results in the time scale appropriate to the latest requests. Thus, the system can successfully reply to many requests without requiring a large amount of memory space. The

database shown in Figure 20 stores measurement results for an ImTCP connection between host A and host B. The primary functions of the system are:

- For every request in a time scale not currently available in the database, a queue is created in the database. This queue is used to store data in the time scales requested with the number of data in each queue the same as the number of required measurement results.
- For a request in a time scale not currently available in the database, the system first returns the most nearly suitable result from the currently available data (for example, an average of results for three weeks is returned when the result for one month is requested but unavailable) then starts collecting data appropriate to the request. The system will return increasingly accurate values for subsequent requests of the same type.
- When ImTCP produces a new measurement result, all entries in all queues are updated with the result.
- Each queue records its idle time (the time elapsed from the last request for data in the queue to the current time). When the storage space limit is reached, the queue with the longest idle time is discarded.

4.2 API for ImTCP

So that upper-layer applications can use ImTCP easily, we added new functions to the TCP socket interface with the following objective:

- Application can request measurement results from ImTCP.
- Applications can use some ImTCP special transmission modes.

Note that measurement results, in addition to being passed to applications, can also be used to control ImTCP transmission. In Section 5 we introduce two examples of transmission modes. The

socket interface allows an application to select the ImTCP transmission mode. Details of the new functions are given below.

New functions

*void measure_control (tcp_Socket *s,int ctl)*

s: Pointer to socket data structure

$ctl = 1$: ImTCP executes the measurement

$ctl = 0$ or other values: ImTCP does not execute the measurement

*void measure (tcp_Socket *s,int f)*

s: Pointer to socket data structure

$f(= 1; 2...)$: ImTCP transmission modes

double measure_result (tcp_Socket *s,double sc,int req)*

s: Pointer to socket data structure

sc: time scale

req: number of measurement results

returned value: req measurement results

5 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.

5.1 Background transmission

In a CDN service such as Akamai [18] or Exodus [19], originating Web servers and Web proxy (caching) servers are deployed in the network in a distributed manner [31, 32]. When a proxy server receives a document transfer request from a Web client, it checks the document cache and, if the requested document is not present, redirects the request to the originating Web server. In addition, some proxy servers perform Web prefetching, that is, transferring in advance those documents that Web clients are expected to request in the near future [33, 34]. These two types of transmission are referred to as *foreground transfer* and *background transfer*, respectively. Since foreground transfers should be completed as soon as possible, background transfers should not affect foreground transfer performance. Such prioritization mechanisms may be realized by an additional mechanism in the underlying IP network such as Diffserv [35]. However, this appears to violate the "end-to-end principle" and considerably limits the service deployment. Rather, service differentiation (between foreground and background transfers in the current context) should be performed at end hosts (corresponding to original Web servers and Web proxy servers in this case) by assuming that no such IP network mechanisms exist.

One possible way to deploy such mechanisms is to use network measurement techniques to estimate available bandwidth and regulate the sending rate of background transfers to avoid degrading the performance of foreground traffic. Here, we introduce an example showing that ImTCP successfully performs the background transfer role using its measurement results. We call this type of ImTCP data transmission *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 $m \cdot N$, where m is set to 5. (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 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 RenoTCP for the background transfer, we

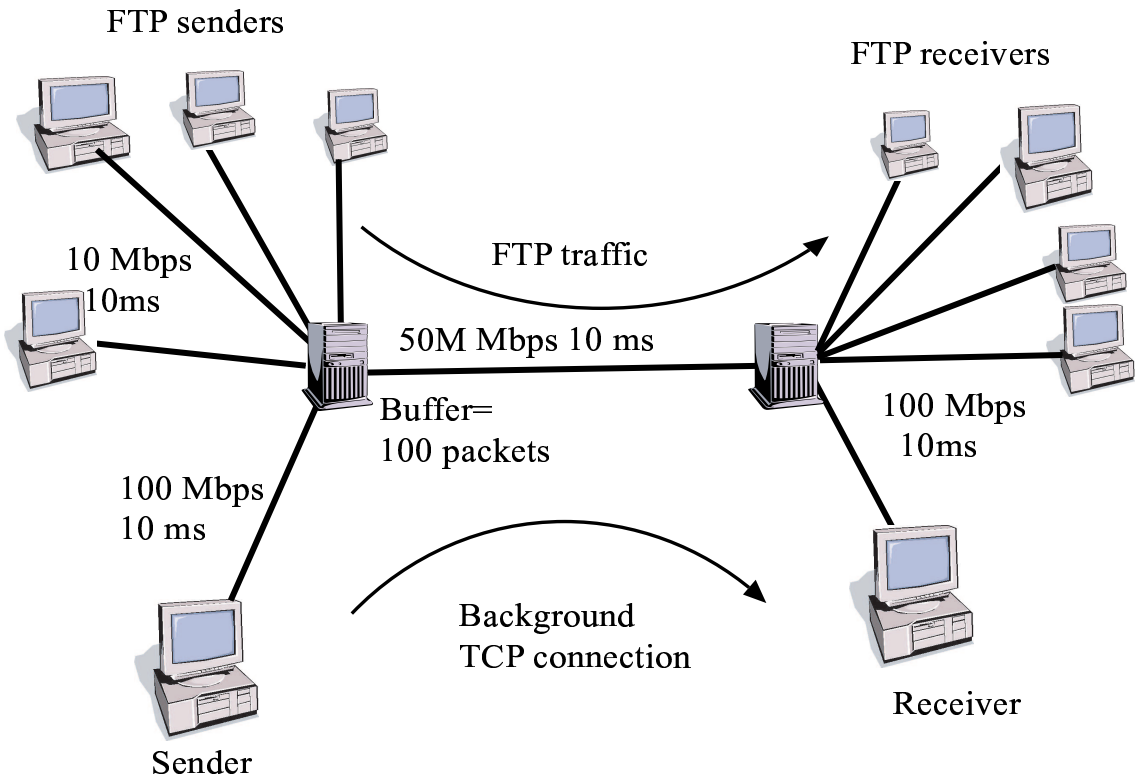
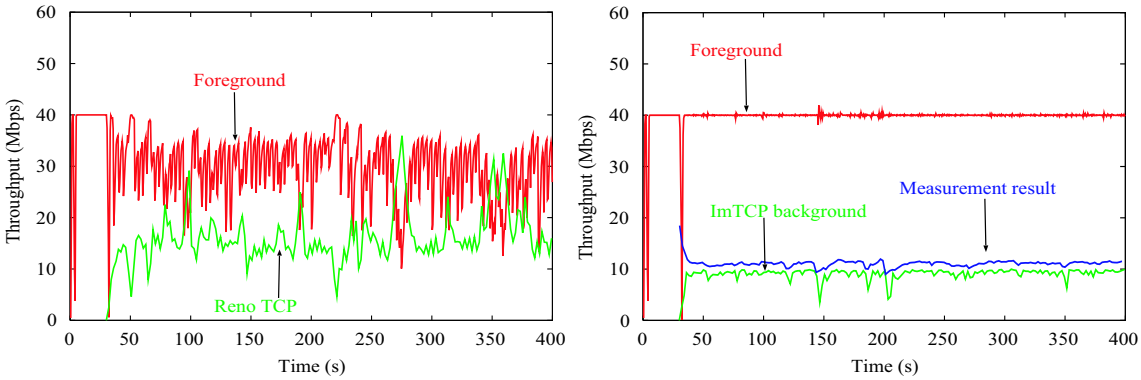


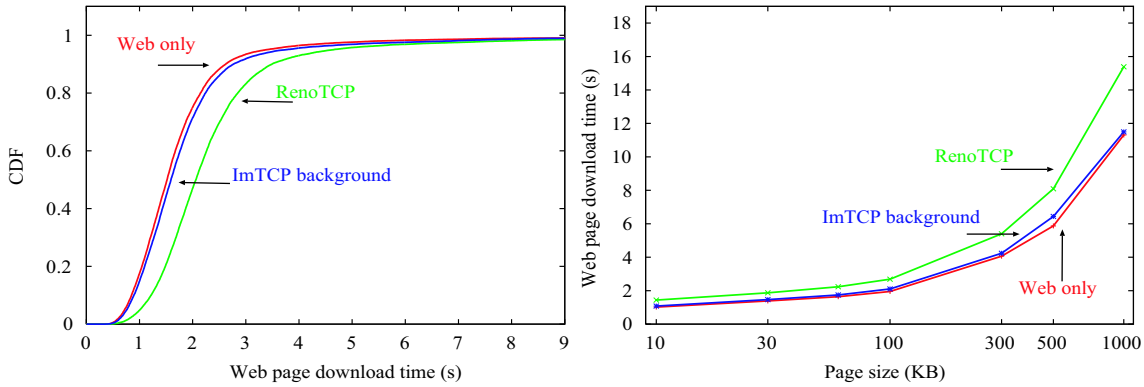
Figure 21: Network model for evaluation of ImTCP background mode



(a) RenoTCP

(b) ImTCP background

Figure 22: Comparison of ImTCP background and RenoTCP performance



(a) CDF (b) Average

Figure 23: CDF and average of Web page download time

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).

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(a) and 23(b) compare the download time for Web pages under ImTCP and RenoTCP. 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 70% that of RenoTCP. 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

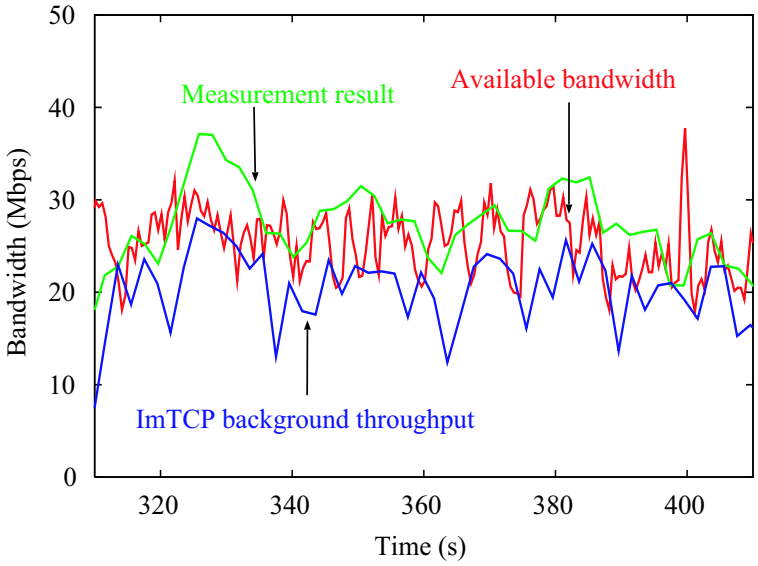


Figure 24: Throughput and measurement result of ImTCP background mode

results in this case confirm that ImTCP in background mode performs the measurement very well.

5.2 Full-speed transmission

Finally, 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 [36]. Similarly, in satellite networks and high-speed 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 [25].

To improve TCP throughput in wireless networks, aggregation of multiple TCP connections [37] can be used. Our approach is similar to the approach introduced in [23] in which the band-

width measurement results are used to improve the TCP performance. For high bandwidth-delay product networks, one study based on real network experiments [25] proposes a more aggressive mechanism for increasing window size to achieve higher utilization of the network link bandwidth. Our approach introduces a more available-bandwidth-aware window size adjustment.

We modify only the congestion avoidance phase of the TCP congestion control mechanism because the slow-start phase is fast enough to obtain the link bandwidth. 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 adjustment given in the following equations.

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

$$V = A \cdot RTT$$

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 RenoTCP.

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) [25] and RenoTCP

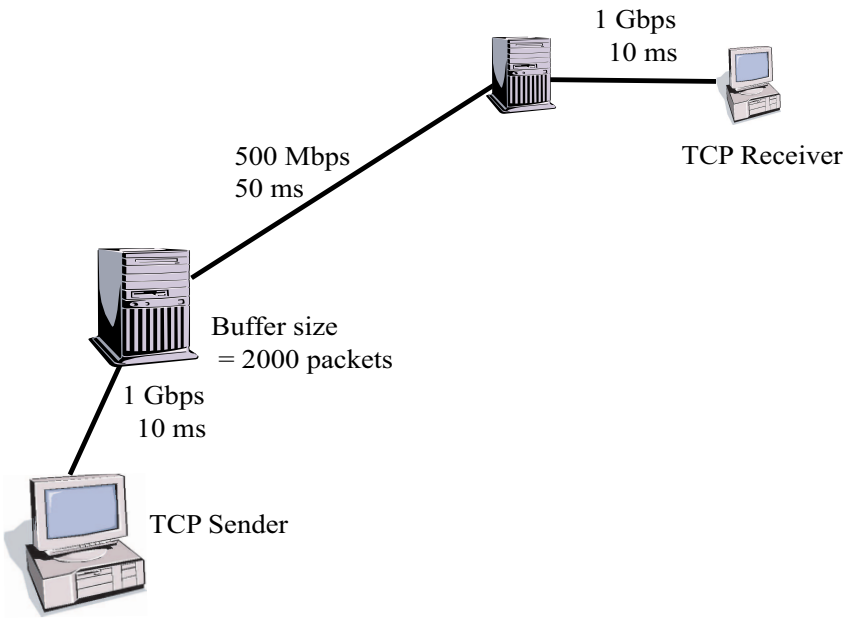
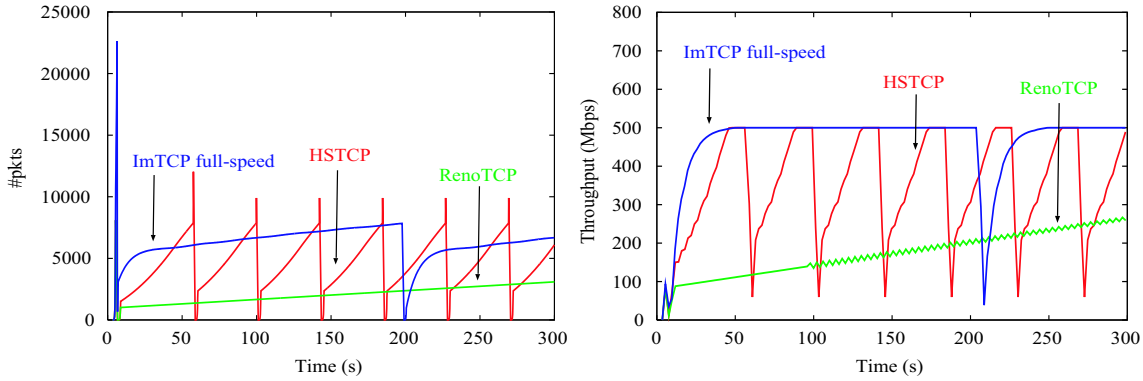


Figure 25: High speed network topology

for data transmission in the network.

Figure 26(a) shows the changes in the congestion window size of TCP connections. RenoTCP 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, as can be seen in Figure 26(b).

Finally, we compare the throughput of ImTCP in full-speed mode with RenoTCP 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 RenoTCP when the loss rate is high, as



(a) Window sizes

(b) Throughputs

Figure 26: Comparision of TCP window sizes and throughputs

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.

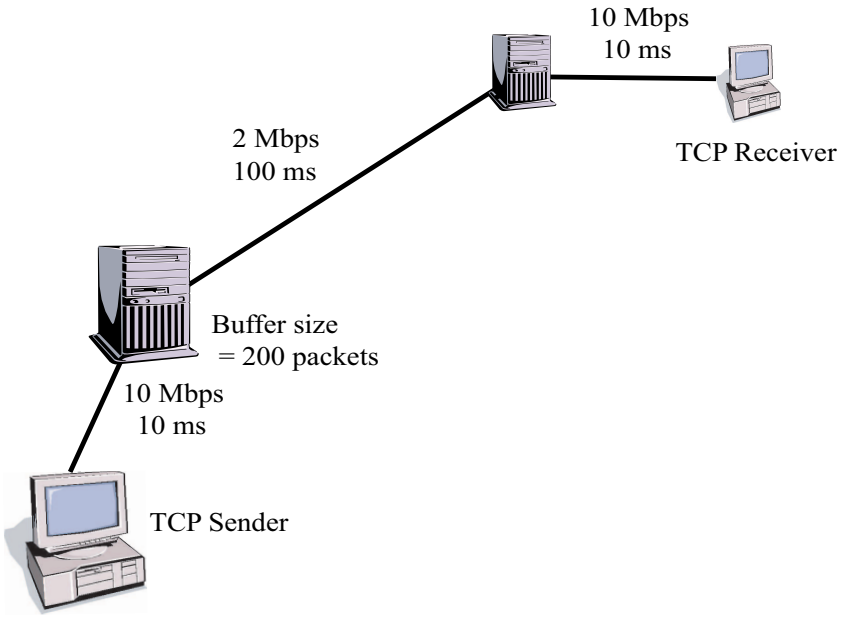


Figure 27: Wireless network topology

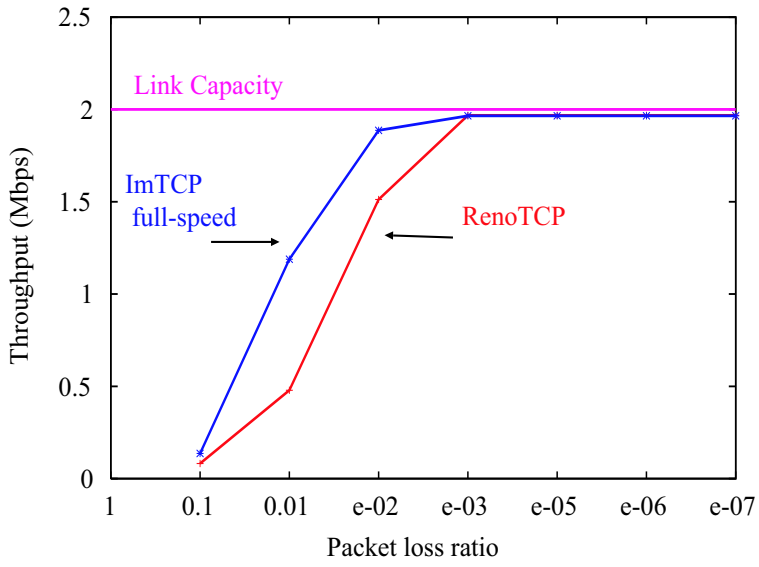


Figure 28: TCP throughput in wireless network

6 Conclusion

In this thesis, we introduced a method for measuring the available bandwidth in a path between two endhosts using an active TCP connection. We first constructed a new measurement algorithm that uses a relatively small number of probe packets and yet provides periodic measurement results quickly. We presented a number of simulation results in order to validate the effectiveness of the proposed algorithm. We then applied the proposed algorithm to an active TCP connection and introduced ImTCP, a version of TCP that can perform measurement of the available bandwidth. We evaluated ImTCP through simulation experiments and found that the proposed measurement algorithm works well in TCP with no degradation in TCP data transmission speed. We also discussed some implementation issues of ImTCP and introduced examples of ImTCP special transmission modes. The implementation of ImTCP is currently in progress and has achieved some of the expected results in the initial steps [38].

In future projects, we will develop new transmission modes for ImTCP as well as evaluate the performance of those introduced in this thesis. We will also consider a bandwidth measurement algorithm that can be deployed at the TCP receiver. Moreover, we intend to develop inline network measurement tools that can be applied not only to available bandwidth but to other characteristics of network links as well, such as bottleneck link capacity and packet loss rate.

Acknowledgements

I would like to express my sincere appreciation to Prof. Masayuki Murata of Osaka University. His guidance and inspiration have provided an invaluable experience that has helped me to fulfill this research.

I would like to express my deepest gratitude to Associate Professor Go Hasegawa of Osaka University. He has been a constant source of encouragement and advice throughout my studies and in the preparation of this manuscript.

I am also indebted to Associate Professors Hiroyuki Ohsaki, Naoki Wakamiya and Research associates Shinichi Arakawa and Ichinoshin Maki of Osaka University who gave me helpful comments and feedback.

Finally, I want to thank my many friends and colleagues in the Department of Information Networking of the Graduate School of Information Science and Technology of Osaka University for their support.

References

[1] 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. 1999.

[2] B. Melander, M. Bjorkman, and P. Gunningberg, "A new end-to-end probing and analysis method for estimating bandwidth bottlenecks," in *Proceedings of IEEE GLOBECOM 2000*, Nov. 2000.

[3] 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.

[4] M. Allman, "Measuring end-to-end bulk transfer capacity," in *Proceedings of ACM SIGCOMM Internet Measurement Workshop 2001*, ACM SIGCOMM, Nov. 2001.

[5] S. Seshan, M. Stemm, and R. H. Katz, "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.

[6] K. Lai and M. Baker, "Nettimer: A tool for measuring bottleneck link bandwidth," in *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, Mar. 2001.

[7] V. Jacobson, "Pathchar-A tool to infer characteristics of Internet paths," `http://www.caida.org/tools/utilities/others/pathchar/`, 1997.

[8] A. B. Downey, "Using Pathchar to estimate internet link characteristics," in *Proceedings of ACM SIGCOMM '99*, pp. 241–250, 1999.

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

[10] B. Hu, A Daniel and P. David, "Topology discovery by active probing," in *Proceedings of the Symposium on Applications and the Internet (SAINT 2002)*, Jan. 2002.

[11] K. Matoba, S. Ata, and M. Murata, "Capacity dimensioning based on traffic measurement in the Internet," in *Proceedings of IEEE GLOBECOM 2001*, pp. 2532–2536, Nov. 2001.

[12] A. Rao, K. Lakshminarayanan, S.Surana, R.Karp and I.Stoica, "Load balancing in structured P2P systems," in *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, Feb. 2003.

[13] F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz and I. Stoica, "Towards a common API for structured peer-to-peer overlays," in *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, Feb. 2003.

[14] Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman, "Grid information services for distributed resource sharing," in *Proceedings of the tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10), IEEE Press*, Aug. 2001.

[15] Y. Zhao and Y. Hu, "GRESS - a Grid replica selection service," in *Proceedings of the 16 International Conference on Parallel and Distributed Computing Systems (PDCS-2003)*, Aug. 2003.

[16] ZB. Allcock, J. Bester, J. Bresnahan, A. L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnal, S. Tuecke, "Data management and transfer in high performance computational Grid environments," *Parallel Computing Journal*, vol. 28, May 2002.

[17] S. Vazhkudai and S. Tuecke and I. Foster, "Replica selection in the globus data Grid," in *Proceedings of International Workshop on Data Models and Databases on Clusters and the Grid (DataGrid 2001)*, May 2001.

[18] Akamai Home Page, `http://www.akamai.com/`.

[19] Exodus Home Page, `http://www.exodus.com/`.

[20] G. Pierre and M. van Steen, "Design and implementation of a user-centered content delivery network," in *Proceedings of the third IEEE Workshop on Internet Applications*, June 2003.

[21] J. Jha and A. Sood, "An architectural framework for management of IP-VPNs," in *Proceedings of the 3rd Asia-Pacific Network Operations and Management Symposium*, Sept. 1999.

[22] 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* , 2003.

[23] M. Gerla, M. Y. Sanadidi, R. Wang, A. Zanella, C. Casetti and S. Mascolo, "TCP Westwood: Congestion window control using bandwidth estimation," in *Proceedings of IEEE Globecom 2001*, pp. 1698–1702, Nov. 2001.

[24] 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.

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

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

[27] NLANR web site, `http://moat.nlanr.net/Datacube/`.

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

[29] 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*, pp. 301–313, 1999.

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

[31] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *Proceedings of IEEE INFOCOM '99*, Mar. 1999.

[32] A. Feldmann, R. Caceres, F. Douglis, G. Glass, and M. Rabinovich, "Performance of Web proxy caching in heterogeneous bandwidth environments," in *Proceedings of IEEE INFOCOM '99*, Mar. 1999.

[33] M. Crovella and P. Barford, "The network effects of prefetching," in *Proceedings of INFOCOM '98*, Apr. 1998.

[34] L. Fan, P. Cao, and Q. Jacobson, "Web prefetching between low-bandwidth clients and proxies: Potential and performance," in *Proceedings of ACM SIGMETRICS '99*, May 2000.

[35] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss, "An architecture for Differentiated Services," *IETF Request for Comments 2475*, Dec. 1998.

[36] 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.

[37] R. Chakravorty, S. Katti, I. Pratt and J. Crowcroft, "Flow aggregation for enhanced TCP over wide area wireless," in *Proceedings of IEEE INFOCOM 2003*, Mar. 2003.

[38] I. Leuth, "Implementation and evaluation of an inline network measurement mechanism for service overlay networks," *Bachelor thesis. Osaka University. Japan*, Feb. 2004.