Master's Thesis

Title

TCP-based Background Data Transfer using Inline Network Measurement Technique

Supervisor

Prof. Masayuki Murata

Author

Tomoaki Tsugawa

February 15th, 2006

Department of Information Networking

Graduate School of Information Science and Technology

Osaka University

Master's Thesis

TCP-based Background Data Transfer using Inline Network Measurement Technique

Tomoaki Tsugawa

Abstract

As the Internet have been rapidly developed and varied IP-based network services have emerged and currently co-exist on the Internet, some of such Internet services do not necessarily require fair resource allocation with respect to other flows and should be operated in the background through prioritization mechanisms. However, it is impossible for TCP Reno, which is the de facto standard transport-layer protocol in the current Internet, to realize such prioritization mechanisms because of the essential nature of its congestion control algorithm.

In this thesis, we propose ImTCP-bg (ImTCP background mode), a new TCP-based background data transfer mechanism. ImTCP-bg sets the upper limit of the congestion window size of the sender TCP based on the results of ImTCP, an inline network measurement technique which measures the available bandwidth of the network path between the sender and receiver hosts from data/ACK packets transmitted by active TCP connections in an inline fashion. ImTCP-bg can provide background data transfer without affecting competing traffic, whereas previous methods cannot avoid network congestion. ImTCP-bg also employs an enhanced RTT-based mechanism so that ImTCP-bg can detect and resolve network congestion, even when reliable measurement results cannot be obtained.

In this thesis, we first evaluate the effectiveness of ImTCP-bg through simulations in terms of the degree of interference with other traffic and the utilization of link bandwidth. One of evaluation results showed that our proposed mechanism improved the performance up to 60 percent in terms of the utilization of unused bandwidth compared with previous background data transfer mechanisms, while the degree of affecting prioritized traffic is almost the same as those of mechanisms. We also report implementation issues of the proposed mechanism on a FreeBSD system and evaluate them on an actual network. We investigated the performance through the extensive experiments and verified that ImTCP can measure well the available bandwidth of the

network path, independent on the degree of its change, and that ImTCP-bg can utilize the available bandwidth well, without degrading the performance of competing traffic, as the simulation results exhibited. From these experiment results, we confirm the effectiveness of our concept, which is the inline network measurement technique.

Keywords

Background Data Transfer, Inline Network Measurement, Congestion Control, Available Bandwidth, TCP (Transmission Control Protocol)

Contents

1 Introduction **8**

2 TCP-based background data transfer **11**

3 ImTCP-bg: ImTCP background mode **14**

3.1 Outline of ImTCP 14

3.2 ImTCP-bg mechanisms 17

3.2.1 Bandwidth-based congestion control mechanism 17

3.2.2 Enhanced RTT-based congestion control mechanism 17

4 Performance evaluations in simulations **19**

4.1 Parameter settings 20

4.1.1 Smoothing parameter γ 20

4.1.2 RTT threshold δ 20

4.2 Simulation results 22

4.2.1 Case of one connection 22

4.2.2 Case of multiple connections 24

5 Implementation and evaluations on actual networks **30**

5.1 Implementation overview 30

5.1.1 Implementation of ImTCP 30

5.1.2 Implementation of ImTCP-bg 31

5.1.3 Issues in kernel timer resolution 32

5.2 Performance evaluations using an experimental network 33

5.2.1 Evaluations of ImTCP 34

5.2.2 Evaluations of ImTCP-bg 36

5.3 Experiments in the actual Internet environment 39

5.3.1 Experiments of ImTCP 39

5.3.2 Experiments of ImTCP-bg 41

6 Conclusions **43**

Acknowledgements **44**

References **45**

List of Figures

1 Inline network measurement by ImTCP 15
2 Change of congestion window size in TCP Reno and ImTCP-bg 16
3 Network model in simulations 19
4 Simulation results (1): Change of congestion window size and its upper limit . . 21
5 Simulation results (2): Effect of parameter γ 22
6 Simulation results (3): Effect of parameter δ 23
7 Simulation results (4): Results of one connection case 25
8 Simulation results (5): Change of throughput with five connections 26
9 Simulation results (6): Utilization of Available bandwidth with multiple connections 28
10 Outline of ImTCP implementation architecture 31
11 Experimental network environment 33
12 Results in experimental network (1): Change of the available bandwidth and measurement results 35
13 Results in experimental network (2): Change of congestion window size and RTT 37
14 Results in experimental network (3): Change of throughput for a background TCP connection 38
15 Results in experimental network (4): Change of throughput for cross traffic . . . 38
16 Network environment of the actual Internet 39
17 Results in actual network (1): Change of the available bandwidth and the measurement results 40
18 Results in actual network (2): Change of throughput and measurement results . . 42
19 Results in actual network (3): Change of RTT 42

List of Tables

1 Average queue length in the case of five connections 24

2 Kernel compilation time and upper limit of measurement bandwidth 32

3 PC specifications of the experimental network environment 34

4 CPU load in an experimental PC 34

1 Introduction

Due to the rapid development of networking technologies in both access and core computer networks, as well as the sudden increase of the Internet population, new and varied types of service-oriented networks have emerged and currently co-exist on the Internet. Referred to as service overlay networks, these networks include Content Delivery/Distribution Networks (CDNs) [1-3] such as Akamai [4], peer-to-peer networks [5, 6], Grid networks [7, 8], and IP-VPNs [9]. Although these services compete for network bandwidth, Transmission Control Protocol (TCP) [10] currently plays a major and important role for avoiding and solving network congestion collapse by using the congestion control algorithm [11]. Thus, TCP provides effective usage and fair sharing of network resources among competing data transmission flows.

However, some of the Internet services do not necessarily require fair resource allocation with respect to other flows and should be operated in the background through prioritization mechanisms. For example, in CDNs, Web servers transfer various types of data (e.g., backup, caching [12], and prefetching [13, 14]), in addition to the data transferred in response to the document transfer request from Web clients. In this case, the user-requested data should be transferred with the higher priority than the other traffic. Data backup and synchronization in Storage Area Networks (SAN), online updating of operating systems (e.g., Microsoft's Background Intelligent Transfer Service [15]), and data caching in peer-to-peer network [16, 17] are examples of tasks that should be performed without affecting other traffic.

In previous studies, such prioritized behaviors were realized by either IP-based mechanisms or application-based mechanisms. In IP-based mechanisms, such as DiffServ [18], the Internet routers are equipped with prioritization mechanisms and process the incoming packets according to pre-defined prioritization policies. For instance, Assured Forwarding [19] in DiffServ has four classes and three dropping levels at the router buffer to differentiate the incoming flows. However, such mechanisms have well-known shortcomings in scalability, because the prioritization mechanisms should be implemented on all routers between the sender and receiver endhosts.

In application-based mechanisms found in [20, 21], the prioritization mechanisms are provided by upper-layer programs, or sometimes by service administrators. For example, cache synchronization and prefetching in CDNs is performed when the amount of user-requested traffic is small. Data backup is usually scheduled to be performed at midnight in order to avoid de-

grading the throughput of other higher-prioritized flows during the daytime. In such cases, the programs/administrators must monitor the network traffic to determine the time during which the network is underutilized. However, successfully realizing such mechanisms is difficult due to large fluctuations in Internet traffic. Moreover, since each application deploys a prioritization mechanism according to its service requirements, we cannot estimate its performance especially when multiple applications with different prioritization mechanisms co-exist in the network.

Therefore, TCP-based approaches such as TCP Nice [22] and TCP-LP [23], which are herein referred to as *background TCP*, have been introduced in order to handle background (lower-prioritized) data transfer by the transport-layer protocols. These approaches observe Round Trip Times (RTTs) of data packets belonging to a TCP connection and decrease the congestion window size when the RTTs increase, whereas the original TCP Reno continues to increase its congestion window size until packet loss occurs, regardless of increases in RTTs. Generally, on the network congestion in the actual networks, the RTT increases before the packet loss occurs since the incoming packets to the bottleneck router are first stored at the buffer, and when the buffer is fully utilized, the incoming packets are discarded. TCP Nice and TCP-LP utilize this characteristic and detect the network congestion earlier than TCP Reno by observing the change in RTTs.

Although both TCP Nice and TCP-LP can realize data transfer without affecting other higher-prioritized traffic, however, these protocols are unable to utilize the available bandwidth of the network efficiently. This is because the degree to which the congestion window size can decrease when the RTTs increase is fixed to too large value regardless of the network condition, similarly to TCP Reno which halves the window size when packet loss occurs. These TCP variants can easily improve the utilization of link bandwidth by changing the parameters. However, this causes the increase in the degree of interference with other traffic. That is, a trade-off relationship exists between the degree of interference with other traffic and the utilization of link bandwidth. Congestion control mechanisms based on the change in RTTs such as TCP Nice and TCP-LP cannot completely solve the trade-off problem. Therefore, we believe that another kind of network congestion indicator, in addition to packet loss and RTT increase, is required to overcome the trade-off.

In this thesis, we propose a novel background TCP mechanism based on bandwidth measurement, with the goal of achieving both background transfer and network bandwidth utilization. The basic idea of the proposed mechanism is to utilize available bandwidth information as a net-

work congestion indicator. That is, the proposed background TCP variant uses the inline network measurement mechanism proposed in [24, 25], which can measure the available bandwidth of the network path between sender and receiver endhosts. The inline network measurement mechanism, called Inline Measurement TCP (ImTCP), uses the data and ACK packets of a TCP connection for the measurement task, without injecting additional probing traffic, which is an ideal characteristic for background data transfer. The proposed mechanism sets the maximum value of the congestion window size of the sender TCP by using the measurement results of the available bandwidth. In addition, we employ an RTT-based mechanism that dynamically determines the degree to which the congestion window size can decrease according to the observed RTT value, whereas TCP Nice and TCP-LP use a constant degree for the possible decrease.

We first evaluate the effectiveness of our proposed mechanism by simulations to confirm fundamental characteristics. However, simulation evaluations are insufficient to confirm the effectiveness, especially when the effectiveness of network measurement techniques is evaluated. Therefore, we implement ImTCP and ImTCP-bg to FreeBSD [26] kernel system, and evaluate its performance on an experimental network, which is the controlled conditional network. Finally, we confirm the performance of our proposed mechanism in the actual network. When we evaluate in the actual networks, we also investigate the measurement accuracy of ImTCP in addition to the evaluation of our proposed mechanism. From these experiment results, we confirm the effectiveness of our concept, which is the inline network measurement technique, in the actual networks.

The remainder of this thesis is organized as follows. Section 2 describes the purpose of TCP-based background data transfer and presents a discussion on the problems of existing mechanisms. In Section 3, we propose ImTCP-bg, a new TCP-based background data transfer mechanism that uses an inline network measurement technique. Section 4 presents simulation results that are used to evaluate the essential performance of ImTCP-bg. In Section 5, we describe the outline of implementation design in FreeBSD 4.10 kernel system and present the performance of the proposed mechanism on actual networks. Finally, we present conclusions and areas for future study in Section 6.

2 TCP-based background data transfer

TCP adjusts the data transmission speed by changing the congestion window size in response to network congestion. The TCP algorithm allows a TCP sender to continue to increase its congestion window size additively until network congestion is detected. TCP decreases the window size multiplicatively when network congestion occurs. As a indicator of the network congestion, TCP Reno uses packet losses in the network (referred to herein as a *loss-based mechanism*). On the other hand, TCP Nice and TCP-LP introduce another congestion indicator, namely the increase of RTTs for data packets (*RTT-based mechanism*). These protocols provide background data transfer without affecting the foreground traffic based on the following assumption. Consider an output link of an Internet router equipped with an output buffer. When the packet incoming rate of the traffic destined for the output link is larger than the output link bandwidth, the excess traffic is stored in the output buffer, which causes some queuing delay, and eventually results in the packet losses when the buffer becomes full. That is, for a TCP connection, the RTTs usually increase before packet losses occur when the network is congested. Therefore, TCP Nice and TCP-LP connections can detect network congestion earlier than TCP Reno connections.

We consider the following two objectives for background data transfer:

(1) no adverse effect on the foreground traffic

(2) full utilization of the network link bandwidth

That is, a perfect background data transfer mechanism can fully utilize the bandwidth that is unused by the foreground traffic, while not degrading the performance of the foreground traffic. However, realizing such a complete mechanism is quite difficult because a trade-off relationship exists between these two objectives. The difficulty in realizing a good background data transfer mechanism lies in balancing this trade-off relationship. For example, TCP Nice and TCP-LP are unable to efficiently utilize the available bandwidth of the network path, especially when the number of background TCP connections is small [22, 23]. This is mainly because these protocols use fixed parameters in detecting network congestion and decreasing the congestion window size. That is, these two background TCP variants set the parameters by which to satisfy objective (1), while sacrificing objective (2). Opposite to this, when these TCP variants change the controlled parameters in order to improve the utilization of link bandwidth, i.e., satisfying the objective

(2), it is obvious that the change causes the increase in the degree of interference with other traffic. In addition, the optimal parameter setting to the trade-off relationship depends on the network condition strongly. Therefore, RTT-based mechanisms such as those of TCP Nice and TCP-LP cannot completely solve this trade-off problem due to their trial-and-error nature. These mechanisms continue to increase the window size until the value of RTT reaches its threshold, that is, until the indication of network congestion appears, and then decrease the window size to some degree in order to avoid the congestion.

In order to satisfy the above two objectives, another congestion indicator is proposed in this thesis, i.e., the available bandwidth of the network path between the sender and receiver hosts (*bandwidth-based mechanism*). An available bandwidth means the residual bandwidth, that is, the bandwidth which no other connections are currently using. The available bandwidth is the most straightforward information by which to describe background data transfer. If the TCP sender obtains the available bandwidth information exactly and quickly, then an ideal background data transfer mechanism, in terms of both the background nature and link utilization can be created.

Many algorithms and tools by which to measure the available bandwidth of network paths have been proposed in the literature [27-31]. However, the existing methods cannot be directly employed for the newly proposed background TCP because these methods utilize numerous test probe packets and require too much time to obtain a single measurement result. For instance, PathLoad [29] sends several 100-packet measurement streams for a measurement. PathChirp [31] is a modification of PathLoad for the purpose of decreasing the number of probe packets. However, the required number of packets to be sent at one time in PathChirp is still large. In order to address this problem, our research group has proposed a novel inline network measurement technique referred to as Inline measurement TCP (ImTCP) in [24, 25]. ImTCP does not inject extra traffic into the network, but rather estimates the available bandwidth of the network path from data and ACK packets transmitted by an active TCP connection in an inline fashion. Since the ImTCP sender obtains bandwidth information every 1–4 RTTs, ImTCP can follow the traffic fluctuation of the underlying IP network. In addition, because the ImTCP mechanism is implemented at the bottom of the TCP layer, various types of TCP congestion control mechanisms can include this measurement mechanism. Therefore, the ImTCP mechanism is integrated into the proposed background TCP in order to obtain the available bandwidth information of the network path.

However, the RTT-based mechanism cannot be discarded even when the bandwidth-based

mechanism is employed, because ImTCP does not always provide accurate measurement results for the available bandwidth. For example, when the congestion window size of the ImTCP sender is small, ImTCP does not measure the available bandwidth. Furthermore, the measurement accuracy of ImTCP depends on the network environment, e.g., the RTT, the physical link bandwidth, and the number of active connections. When the measured available bandwidth value is inaccurate, the background data transfer based on the measured value may affect the foreground traffic. Therefore, the RTT-based mechanism should be used in conjunction with the bandwidth-based mechanism.

In the next section, we describe the mechanism of ImTCP-bg, a new TCP-based background data transfer mechanism based on the available bandwidth measurement technique of ImTCP. The proposed mechanism also employs an enhanced RTT-based mechanism, which dynamically determines the control parameters.

3 ImTCP-bg: ImTCP background mode

Basically, we do not care which traffic should be transferred by background TCP mechanism. That is, we do not introduce a mechanism that automatically selects the transfer mode (normal or background) for traffic from upper-layer applications. Instead, we prepare the interface to select the transfer mode by socket option. Therefore, each application should select the transfer mode when it establishes a new TCP connection.

TCP Nice and TCP-LP use fixed parameters in detecting network congestion and decreasing the congestion window size. This is mainly because these TCP variants use only RTTs as an indicator of the network congestion. RTTs change by various factors including the network congestion, so these protocols should set parameters in consideration of that. On the other hand, since ImTCP-bg also uses the available bandwidth as an indicator, it can detect whether or not the change in RTTs is caused by the network congestion and it can use the information of RTTs more effectively.

ImTCP-bg consists of two major congestion control mechanisms: a bandwidth-based mechanism with inline network measurement and an enhanced RTT-based mechanism for adjusting the congestion window size when the first mechanism does not well, as we described in the previous section. In this section, we first introduce the outline of the inline network measurement technique, ImTCP, and then describe the detail of each proposed mechanism. The ImTCP algorithm is described in detail in [24, 25].

3.1 Outline of ImTCP

ImTCP measures the available bandwidth of the network path between sender and receiver hosts. In TCP data transfer the sender host transfers a data packet and the receiver host replies the data packet with an ACK packet. ImTCP measures the available bandwidth using this mechanism. That is, ImTCP adjusts the interval of data packets according to the measurement algorithm, and then calculates the available bandwidth by observing the change of ACK intervals as shown in Figure 1.

During each measurement, ImTCP uses a search range to find the value of the available bandwidth. The search range is a range of bandwidth that is expected to include the current available bandwidth. ImTCP searches for the available bandwidth only within a given search range. By

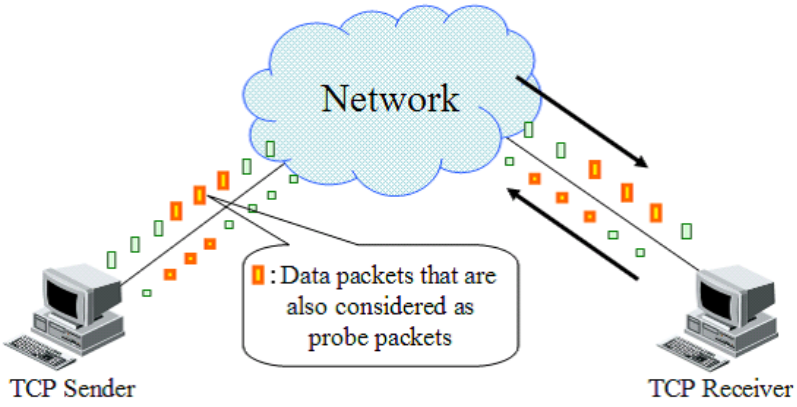


Figure 1: Inline network measurement by ImTCP

introducing the search range, ImTCP can avoid sending probe packets at an extremely high rate, which seriously affects other traffic. ImTCP can also keep the number of probe packets for the measurement quite small. The search range is determined using the previous measurement results. Therefore, when the available bandwidth changes rapidly according to changes in the network condition, the available bandwidth does not exist in the search range. ImTCP can guess the new available bandwidth through a few measurement trials. The following are the steps of the proposed algorithm for one measurement:

(1) Send a packet stream (a group of packets sent simultaneously) according to the Cprobe [27] algorithm to obtain a very rough estimation of the available bandwidth and use the result to set the initial search range.

(2) Divide the search range into multiple sub-ranges of identical width of bandwidth. Send a packet stream for each of sub-range. The transmission rates of the packets vary to cover the sub-range of the bandwidth range.

(3) Find a sub-range that is expected to include the available bandwidth by checking to see if an increasing trend exists in the transmission delay of each stream. Because the increasing trend of the transmission delay in a stream indicates that the transmission rate of the stream is larger than the current available bandwidth of the network path, ImTCP can choose a sub-range that is most likely to include the correct value of the available bandwidth.

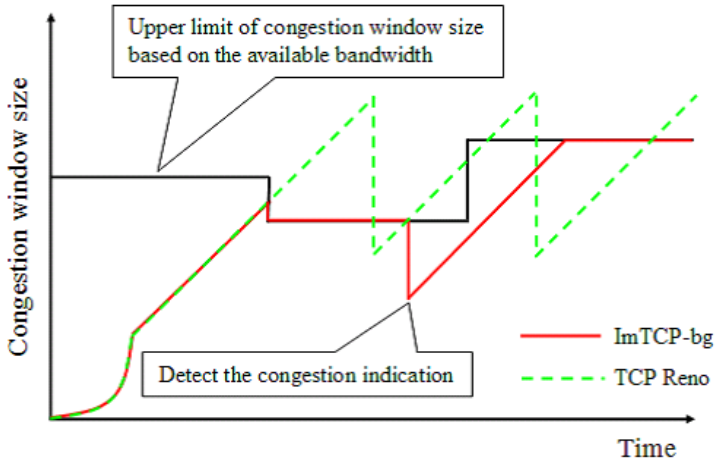


Figure 2: Change of congestion window size in TCP Reno and ImTCP-bg

(4) Determine the available bandwidth from the arrival rates of every two successive packets of the stream corresponding to the sub-range chosen in Step 3. Since arrival intervals being larger than the transmission intervals indicates that the transmission rate of the two packets is larger than the available bandwidth, ImTCP determines the available bandwidth as the largest rate of the packet pairs, for which the arrival interval is the same as the transmission interval.

(5) Create a new search range using the 95% confidence interval of the previous results and use the current available bandwidth as the center of the search range. When the available bandwidth can not be found within the search range, the network status may have changed greatly so that the available bandwidth shifts out of the search range. ImTCP then widens the search range in the possible direction of change of the available bandwidth. Return to Step 2 after determining the new search range.

3.2 ImTCP-bg mechanisms

3.2.1 Bandwidth-based congestion control mechanism

In order to realize the background data transfer, we measure the available bandwidth of the network path using the ImTCP mechanism and utilize the information of the available bandwidth to control the congestion window size of a TCP connection as shown in Figure 2. ImTCP-bg smoothes the measurement results of ImTCP using a simple exponential weighted moving average, as follows:

$$\bar{A} \leftarrow (1-\gamma) \cdot \bar{A} + \gamma \cdot A_{cur} \tag{1}$$

where A_{cur} denotes the current result of the available bandwidth measured by ImTCP mechanism, γ is a smoothing parameter, and $\bar{A}$ is the smoothed available bandwidth. Note that the appropriate setting of γ will be described in Section 4. The ImTCP-bg sender then sets the upper limit of the congestion window size ($maxcwnd$) using the following equation:

$$maxcwnd \leftarrow \bar{A} \cdot RTT_{min} \tag{2}$$

where RTT_{min} is the minimum RTT experienced throughout the lifetime of the connection.

As shown above, the proposed bandwidth-based mechanism is quite simple: the upper-limit of the congestion window size is simply set as the product of the measured available bandwidth and the minimum RTT. In Section 4 and Section 5, this simple mechanism is demonstrated to be effective for background data transfer.

3.2.2 Enhanced RTT-based congestion control mechanism

The effectiveness of the above-described bandwidth-based mechanism depends largely on the accuracy of the measurement by ImTCP of the available bandwidth. In [25] the authors demonstrated that ImTCP can give the reasonably accurate measurement results every 1–4 RTTs. However, ImTCP does not always provide reliable measurement results, and may result in the congestion of the bottleneck link. For example, when the current window size if smaller than the number of packets required for a measurement, ImTCP does not measure the available bandwidth. In addition, when other traffic send data packets in a bursty manner, the intervals of ImTCP data packets are disturbed by the bursty traffic, making the measurement result inaccurate. Therefore, the RTT-based mechanism is employed to quickly detect and resolve the undesirable network congestion.

ImTCP-bg detects network congestion using only the current and minimum values of RTT, whereas TCP Nice and TCP-LP also use the maximum RTT, which is difficult to observe in the actual network. When an increase in RTT is detected, the ImTCP-bg sender decreases its congestion window size immediately in order to resolve the congestion. Next, denote $\overline{RTT}$ as the smoothed RTT value that is calculated by the traditional TCP mechanism and RTT_{min} as the minimum RTT. Here, δ (> 1.0) is the threshold parameter to judge whether network congestion occurs. The ImTCP-bg sender detects the network congestion when the following condition is satisfied:

$$\frac{\overline{RTT}}{RTT_{min}} > \delta \tag{3}$$

When Equation (3) is satisfied, it means that the queuing delay occurs at the bottleneck router. Since we treat the increase of queuing delay as the indication of network congestion, ImTCP-bg decreases its congestion window size according to the following equation in order not to affect the foreground traffic.

$$cwnd \leftarrow cwnd \cdot \frac{RTT_{min}}{\overline{RTT}} \tag{4}$$

Here, $cwnd$ is the current congestion window size, $\overline{RTT}$ is the smoothed RTT value and RTT_{min} is the minimum RTT. Equation (4) implies that ImTCP-bg determines the degree of decrease of the congestion window size based on the ratio of the current value of RTT and its minimum value. Thereby, ImTCP-bg avoids unnecessary the underutilization of the link bandwidth while maintaining the background-based data transfer. Note that this modification of the RTT-based mechanism is effective because the bandwidth-based mechanism is used concurrently.

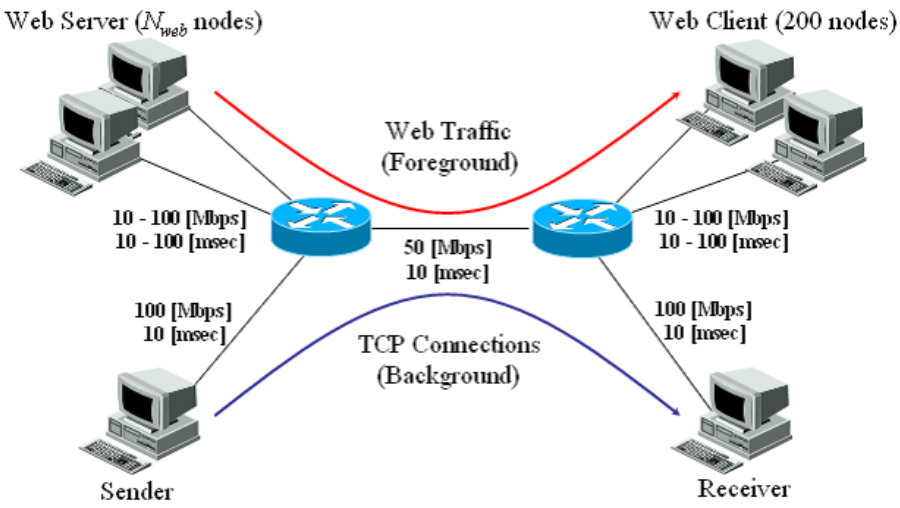


Figure 3: Network model in simulations

4 Performance evaluations in simulations

In this section, simulation results are used to evaluate the performance of ImTCP-bg, as proposed in Section 3, and ns-2 [32] is used for the simulations. Traditional TCP Reno, TCP Nice and TCP-LP were chosen for performance comparison. We first describe the parameter setting issues of ImTCP-bg in Subsection 4.1, and then compare the performance of ImTCP-bg to that of TCP Nice and TCP-LP in Subsection 4.2.

The network model used in the simulation is depicted in Figure 3. This model consists of sender/receiver hosts, two routers, and links between the hosts and routers. The bandwidth of the bottleneck link is set to 50 Mbps, and the propagation delay is 10 msec. A DropTail discipline is deployed at the router buffer, and the buffer size is set to 1000 packets. The packet size is 1500 Bytes. Web traffic is assumed to be foreground traffic. N_{web} Web servers transfer Web documents to 200 Web clients. The bandwidth of the access link of each Web node is set randomly between 10 and 100 Mbps, and the propagation delay is also a random value between 10 and 100 msec. The amount of foreground Web traffic is adjusted by changing N_{web}. In addition, one or more TCP connections are established in order to perform background data transfer. The performance of the background TCP variants are compared with respect to the following: the transfer time of the foreground Web traffic, the queue length of the bottleneck link buffer, the

throughput of the background data transfer, and utilization of the available bandwidth. The control parameters for TCP Nice and TCP-LP are configured according to [22, 23].

4.1 Parameter settings

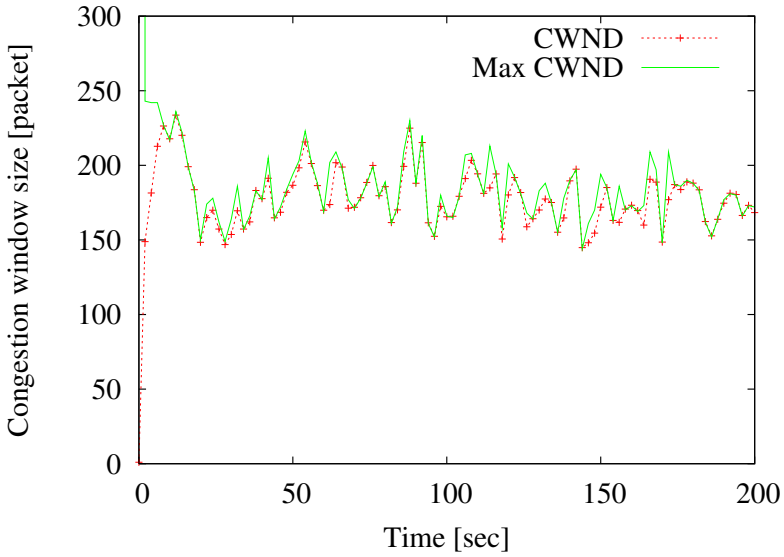
4.1.1 Smoothing parameter γ

First, we consider the effect of the smoothing parameter of the measured available bandwidth of Equation (1), which is denoted as γ ($0 < \gamma < 1$). If γ is set to a larger value, then measurement errors have a greater influence on ImTCP-bg, and changes in the network environment can be detected more rapidly. On the other hand, if γ is set to a smaller value, the ability to detect changes in the available bandwidth may be degraded, but the instantaneous measurement error can be filtered out. Therefore, γ should be set in a way such that these two performance aspects are balanced.

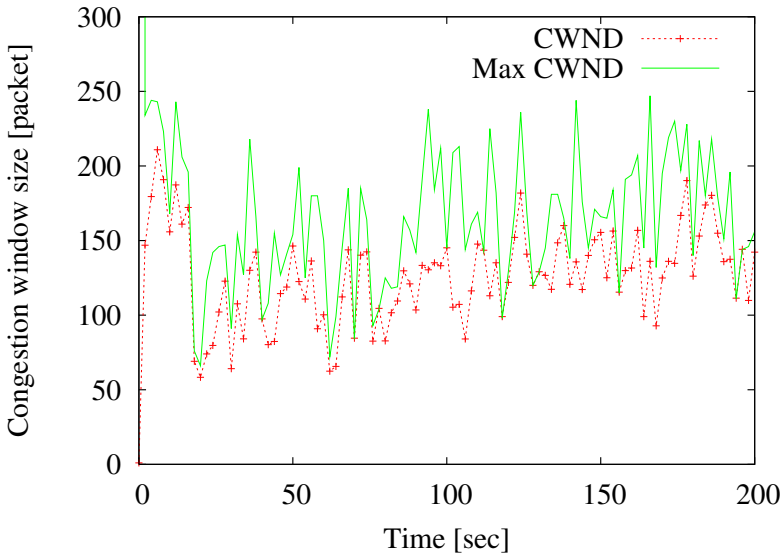
Figure 4 shows the changes in the congestion window size and its upper limit, tuned by ImTCP-bg, as a function of time, when γ is set to 0.1 (Figure 4(a)) and 0.9 (Figure 4(b)). Figure 5 shows the average utilization of the available bandwidth as a function of γ. Here, N_{web} is set to 20 and δ is set to 1.2. These figures indicate that when the value of γ is large, the congestion window size tends to be small. ImTCP-bg increases its congestion window size additively until the congestion size reaches its upper limit. Opposite to this, when the congestion window size exceeds its upper limit, ImTCP-bg decreases the congestion window size to the upper limit immediately. Therefore ImTCP-bg cannot utilize the available bandwidth effectively when the value of γ is large. Based on the above considerations, we set γ to $\frac{1}{8}$ in the following simulations. This value is small enough to utilize the available bandwidth effectively and is typically used for calculating the smoothing RTT for TCP.

4.1.2 RTT threshold δ

Next, we consider the RTT threshold δ ($\delta \geq 1.0$) of Equation (3). In order to determine the appropriate value, we change δ to various values and conduct a number of simulations. Figure 6 shows the changes in the available bandwidth utilization and average queue length at the bottleneck link buffer as functions of δ, where γ is set to $\frac{1}{8}$ according to the results in Subsection 4.1.1, N_{web} is set to 20 and the number of ImTCP-bg connections is set to 1, 2, 5, and 10.



(a) $\gamma = 0.1$



(b) $\gamma = 0.9$

Figure 4: Simulation results (1): Change of congestion window size and its upper limit

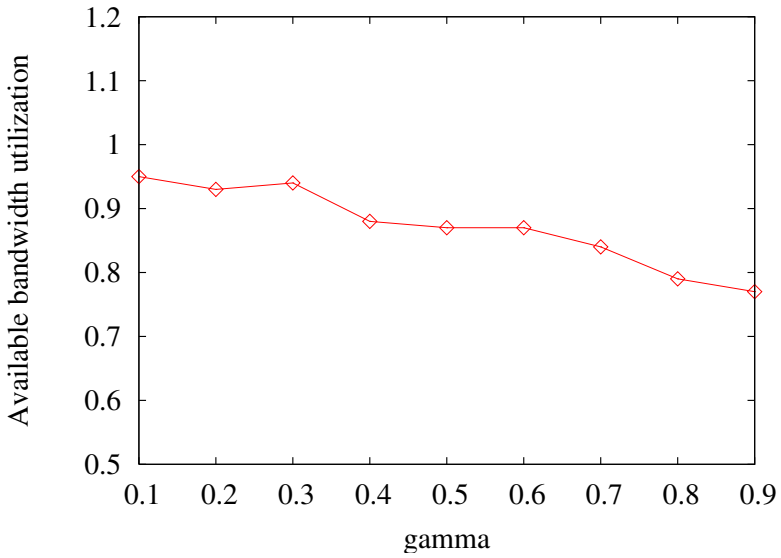


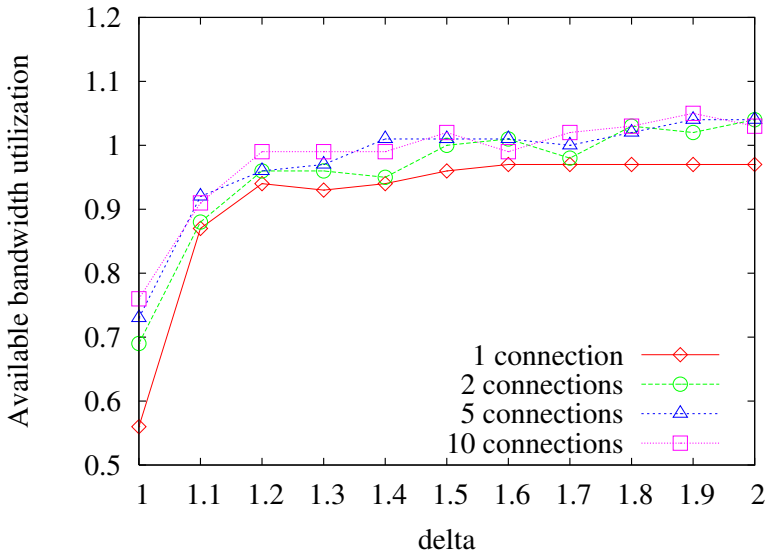
Figure 5: Simulation results (2): Effect of parameter γ

Figure 6 shows that if δ is set to a very small value, then ImTCP-bg cannot utilize the available bandwidth effectively, even though the queue length is small. This is because when ImTCP-bg uses small δ, the instantaneous measurement error is sensed and the congestion window size is frequently decreased. Note that the larger the average queue length, the larger the affect on the foreground traffic. Therefore, we should set δ so as to balance the degree of interference with the foreground traffic and the utilization of available bandwidth. From the extensive simulation results, including those shown in Figure 6, $\delta = 1.2$ is determined to be a good selection in order to balance these two requirements.

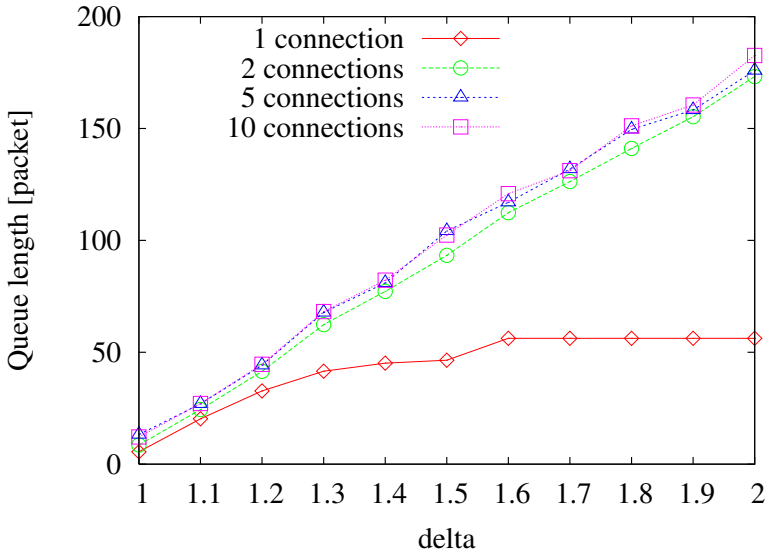
4.2 Simulation results

4.2.1 Case of one connection

First, we present the simulation results for the case in which one background data transfer connection is established, and evaluate the degree of interference with the foreground traffic and the utilization of network bandwidth. The number of Web servers, N_{web}, is changed from 10 to 50. Figure 7 shows the change in the average download time of foreground Web documents and the average throughput of the background TCP connection. The results labeled as "available band-



(a) Average utilization of available bandwidth



(b) Average queue length

Figure 6: Simulation results (3): Effect of parameter δ

Table 1: Average queue length in the case of five connections

	TCP Reno	TCP Nice	TCP-LP	ImTCP-bg
Average queue length [packet]	583.34	8.44	64.63	44.11

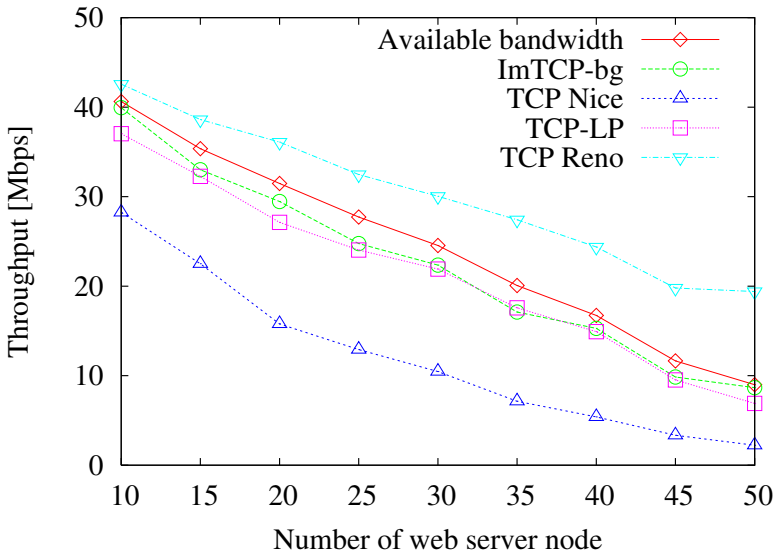
width" in Figure 7(a) and "no background traffic" in Figure 7(b) show the results for the case in which no background data transfer exists.

Figure 7(a) shows that although the TCP Reno connection achieves the highest throughput, the throughput exceeds the available bandwidth. Furthermore, Figure 7(b) shows that the average download time of the foreground Web documents is larger than for the case in which no background traffic exits. That is, TCP Reno cannot be used for background data transfer. On the other hand, for TCP Nice, TCP-LP and ImTCP-bg, the average download time in Figure 7(b) is almost identical to the case of no background traffic. This means that these protocols do not affect the foreground Web traffic, satisfying one of the objectives of the background data transfer. Furthermore, Figure 7(a) shows that the average throughput of the ImTCP-bg connection is the closest to the available bandwidth. Therefore, ImTCP-bg is determined to have the most ideal characteristics for background data transfer, which satisfies objectives (1) and (2) in Section 2.

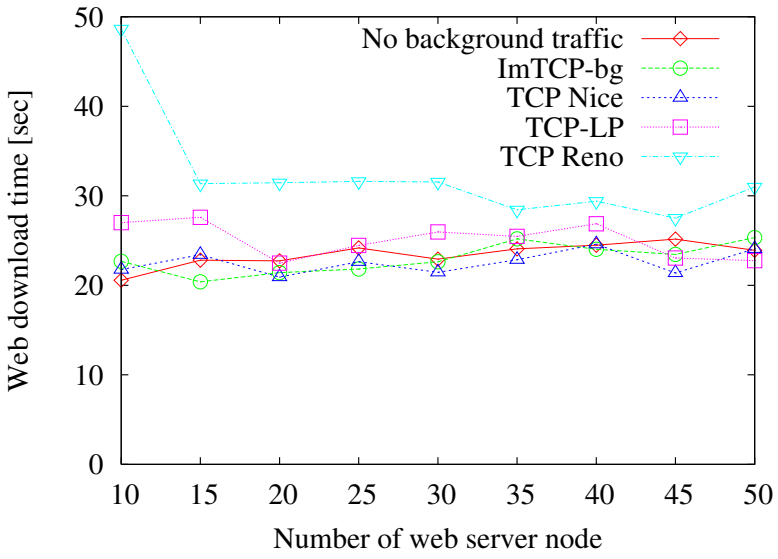
4.2.2 Case of multiple connections

Next, we show the results for the case in which two or more background data transfer connections are established, and evaluate the effect of multiple background TCP connections. In the first simulation, five background TCP connections join the network at 0, 50, 100, 150, and 200 seconds, and end data transmissions at 500, 450, 400, 350, and 300 seconds. This means that the number of active background TCP connections in the network is as follows: 1, 2, 3, 4, 5, 4, 3, 2, and 1. Table 1 shows the average queue length at the output link buffer of the bottleneck router and Figure 8 shows the throughput of background data transfer connection as functions of time. Here, N_{web} is set to 20.

TCP Reno shows the worst behavior for the background data transfer, in terms of large queue length at the bottleneck link (Table 1) and over-utilization of the available bandwidth of the network (Figure 8(a)). Table 1 also shows that the average of queue length of TCP Nice is the smallest among the four variants, meaning that TCP Nice is the best choice for satisfying objective (1), de-

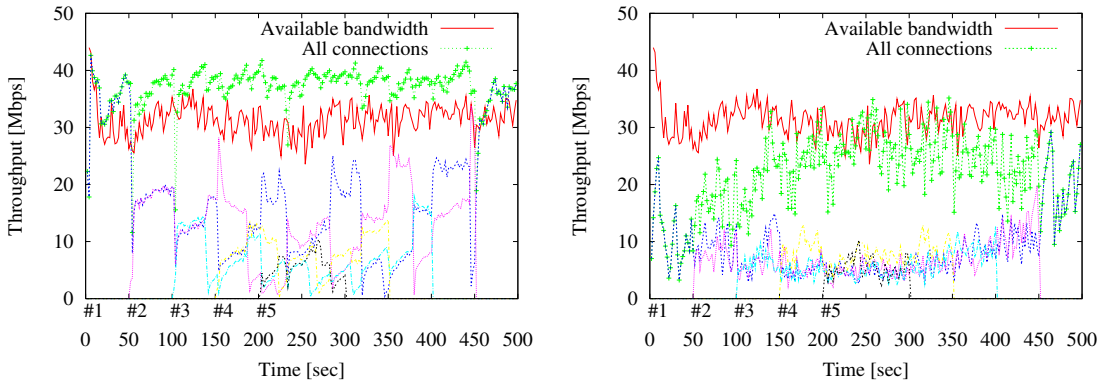


(a) Average of throughput



(b) Average of download time

Figure 7: Simulation results (4): Results of one connection case



(a) TCP Reno

(b) TCP Nice

(c) TCP-LP

(d) ImTCP-bg

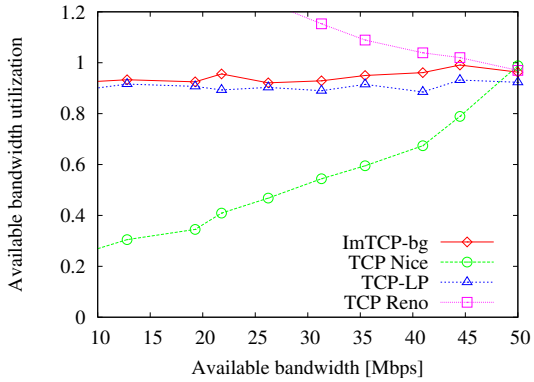
Figure 8: Simulation results (5): Change of throughput with five connections

scribed in Section 2. However, Figure 8(b) shows that the throughput of the background data transfer is the lowest among the four variants, especially when the number of connection is small.

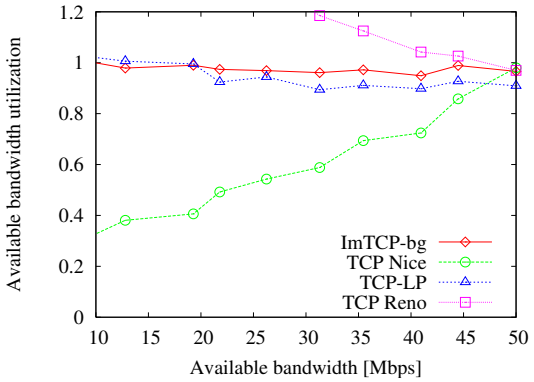
Figure 8(c) shows that when TCP-LP is used for background data transfer, packet losses occur immediately after a new connection is established. This is because TCP-LP needs the maximum RTT to control its congestion window size. TCP Nice and TCP-LP detect network congestion by using the minimum and the maximum RTT (or one-way packet delay). However, essentially, monitoring the maximum RTT by background TCP is difficult because these TCP variants decrease the congestion window size at an early stage of network congestion. Therefore TCP-LP intentionally continues to increase its congestion window size until packet losses occur at the initial slow start phase to determine the maximum RTT value. Consequently, the TCP-LP connection cannot avoid occurring packet losses in the beginning of the data transfer. Furthermore, in Figure 8(c) we can see that the throughput of TCP-LP connections are shown to be quite low for some time after the packet loss. This is because the fast retransmission and fast recovery mechanism of TCP Reno is activated.

Figure 8(d) shows that ImTCP-bg connections can utilize the available bandwidth of network path effectively even when only one connection exists. This is because ImTCP-bg controls its congestion window size appropriately using the results of inline network measurement. Furthermore, when multiple connections exist in the network, ImTCP-bg connections can maintain high utilization of the available bandwidth and the change in the throughput of each ImTCP-bg connection is stable compared with other background TCP variants. That is because ImTCP-bg dynamically changes the degree of decrease congestion window size according to the change in the RTT. From these simulation results, the bandwidth-based algorithm with inline measurement and the RTT-based algorithm are determined to co-exist well in ImTCP-bg to realize background data transfer.

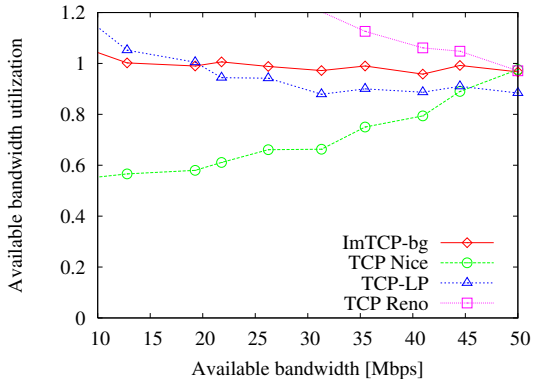
We then check the utilization of the available bandwidth in more detail. In the next simulation, K background TCP connections ($C_1, C_2, ..., C_K$) exist in the network. C_1 joins the network at 0 second and ends data transmission at 500 seconds. C_i $(i = 2, 3, ..., K)$ starts at $\frac{200}{i-1}$ seconds and ends at $500 - \frac{200}{i-1}$ seconds. This means that the number of active background TCP connections in the network changes as time passes similarly to the previous simulation (Figure 8). Figure 9 shows the change of the utilization of the available bandwidth as a function of the available bandwidth when the number of background TCP connections is 1 (Figure 9(a)), 2 (Figure 9(b)), 5 (Figure 9(c)), and 10 (Figure 9(d)). Here, available bandwidth is changed by changing the number of



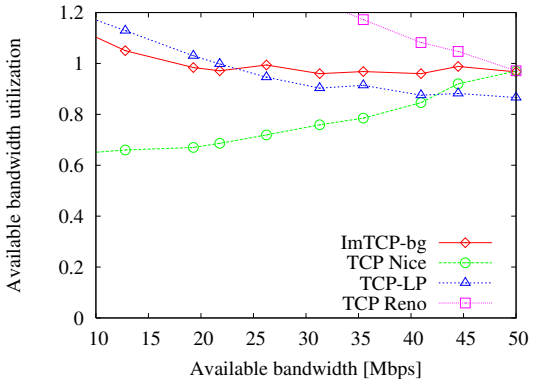
(a) Number of connections: 1



(b) Number of connections: 2



(c) Number of connections: 5



(d) Number of connections: 10

Figure 9: Simulation results (6): Utilization of Available bandwidth with multiple connections

servers, N_{web}.

Figure 9 shows that the bandwidth utilization of TCP Reno connections greatly exceeds the available bandwidth, meaning that it makes severe effect on the foreground traffic. When using TCP Nice, the less the available bandwidth is, the less the utilization of available bandwidth becomes. This is because when the available bandwidth is small, TCP Nice frequently detects the network congestion and decreases the congestion window size in order not to affect the foreground traffic. For TCP-LP, the utilization of the available bandwidth exceeds 1.0 when the available bandwidth is small. This means that it does not satisfy the purpose of background transfer, that is, TCP-LP affects the foreground traffic. Furthermore, when the available bandwidth is large and many background TCP connections exist, the utilization of the available bandwidth becomes small. This is because packet losses occur after the new TCP-LP connection is established, as Figure 8(c) shows. On the other hand, ImTCP-bg shows the best behavior regardless of the available bandwidth and the number of background TCP connections. Therefore, we conclude that ImTCP-bg can utilize effectively the available bandwidth in any cases.

5 Implementation and evaluations on actual networks

Simulation plays a vital role in attempting to characterize a protocol, whereas the simulation condition is relatively ideal compared to the actual network. Because the heterogeneity of the actual network ranges from individual links and network equipments to protocols that inter-operate over the links and a ”mix” of different applications in the Internet, the protocol behavior in the simulation may be quite different from that on an actual network. Therefore, the measurement-related mechanisms must be tested on actual networks in order to evaluate the effectiveness of the proposed mechanism. In this section, we first describe the outline of the implementation of ImTCP and ImTCP-bg in the FreeBSD 4.10 kernel system, and then evaluate the performance of our proposed mechanism on actual networks.

5.1 Implementation overview

5.1.1 Implementation of ImTCP

When new data is generated at the application, the data is passed to the TCP layer through the socket interface. The data (packet) is passed to the IP layer after TCP protocol processing by the $tcp\_output()$ function and is injected into the network. Because the program for inline network measurement must know the current size of the congestion window of TCP, it should be implemented at the bottom of the TCP layer as shown in Figure 10. Therefore, the measurement program is implemented in the $tcp\_output()$ function. When a new TCP data packet is received from the application and is ready to be transmitted, it is stored in an intermediate FIFO buffer (hereafter referred to as the ImTCP buffer) before being passed to the IP layer. The stored packets are passed to the $ip\_output()$ function in the intervals based on the measurement algorithm. The measurement program records the transmission time of the data packet when it departs the ImTCP buffer.

On the other hand, the measurement program should also be implemented in the $tcp\_input()$ function. An ACK packet that arrives at the IP layer of the sender host is passed to the $tcp\_input()$ function for TCP protocol processing. The measurement program records the time when the ACK packet arrives at the $tcp\_input()$ function. The measurement program also guesses the current available bandwidth based on the sending time of data packets and the receiving time of ACK packets according to the algorithm explained in Subsection 3.1.

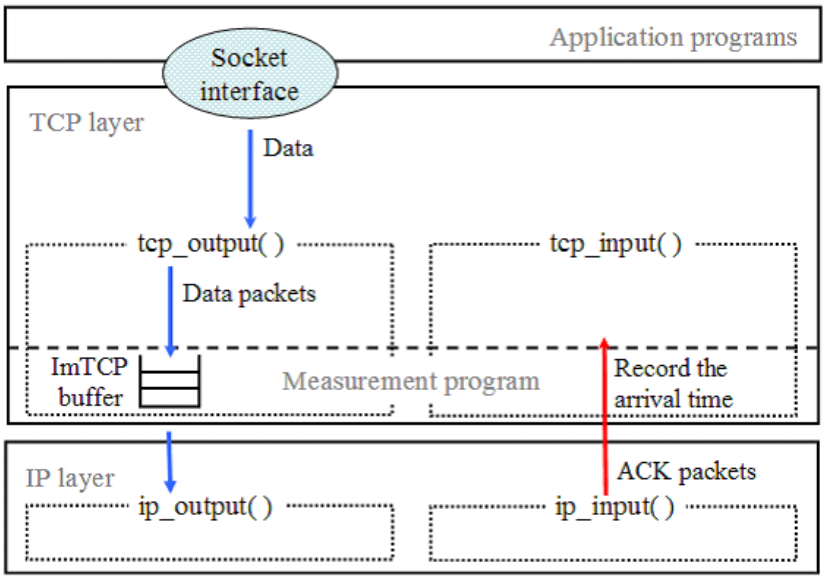


Figure 10: Outline of ImTCP implementation architecture

5.1.2 Implementation of ImTCP-bg

The congestion window size of a TCP connection is updated when an ACK packet is passed to the $tcp\_input()$ function for TCP protocol processing. Therefore, the congestion window control program for ImTCP-bg should be implemented in the $tcp\_input()$ function. That is, whenever the congestion window size is updated by the $tcp\_input()$ function, the congestion window control program determines the congestion window size and its upper limit according to the ImTCP-bg algorithm. ImTCP-bg uses some of the information stored by the TCP connection to control the congestion window. This is referred to as called the *tcpcb* structure. ImTCP-bg also adds the new variable $snd\_maxcwnd$ to the *tcpcb* structure in order to record the upper limit of the congestion window size.

When the ACK packet is passed to the $tcp\_input()$ function and the original TCP updates the congestion window size, ImTCP-bg first checks the variable t_srtt, the smoothed RTT value, and $t\_rttbest$, the minimum RTT. When t_srtt reaches the RTT threshold, which is calculated as $\delta \cdot t\_rttbest$, ImTCP-bg decreases the snd_cwnd, i.e., the congestion window size based on the $t\_rttbest$ and t_srtt. At the same time, ImTCP-bg sets the variable $snd\_maxcwnd$ based on the measurement result of ImTCP. When snd_cwnd is larger than $snd\_maxcwnd$, ImTCP-bg set the

Table 2: Kernel compilation time and upper limit of measurement bandwidth

HZ	Kernel compilation time [sec]	Upper limit of measurement bandwidth [Mbps]
100	168.20	1.2
1,000	170.09	12
10,000	183.38	120
20,000	199.78	240
50,000	277.84	600
100,000	734.10	1,200

value of $snd\_maxcwnd$ to snd_cwnd.

5.1.3 Issues in kernel timer resolution

The measurement algorithm for ImTCP adjusts the transmission intervals of data packets and guesses the current available bandwidth by observing ACK intervals corresponding to the data packets. Data packets are stored in the ImTCP buffer before being passed to the IP layer, and are passed to the $ip\_output()$ function in intervals based on the measurement algorithm, as explained in Subsection 5.1.1. This mechanism is realized by using the task scheduling function offered by the kernel system. When the measurement program utilizes this function, the resolution of the task scheduling timer becomes an issue. The resolution of the kernel system timer is generally coarser than that of the application timer [33]. For example, the default value of the resolution of the kernel system timer is 10 msec, while that of the application timer is 1 μsec in FreeBSD. Therefore, this coarse timer resolution may reduce the accuracy of measurement results.

The resolution of the timer is determined by the parameter HZ, which is defaulted to 100 in FreeBSD kernel system. When HZ is chosen to be 100, the timer resolution becomes 10 msec. Under this setting, ImTCP can only measure the available bandwidth up to 1.2 Mbps with 1500-Byte data packets. Moreover, the bandwidth resolution becomes coarse as the measurement result approaches the upper limit. Therefore, if ImTCP measures the available bandwidth in the broadband networks, HZ should be set larger. For example, if HZ is set to 100,000, then the resolution of the timer becomes 10 μsec and ImTCP can measure the available bandwidth up to 1.2 Gbps. However, when HZ is set to such large value, the timer interrupts by the kernel system

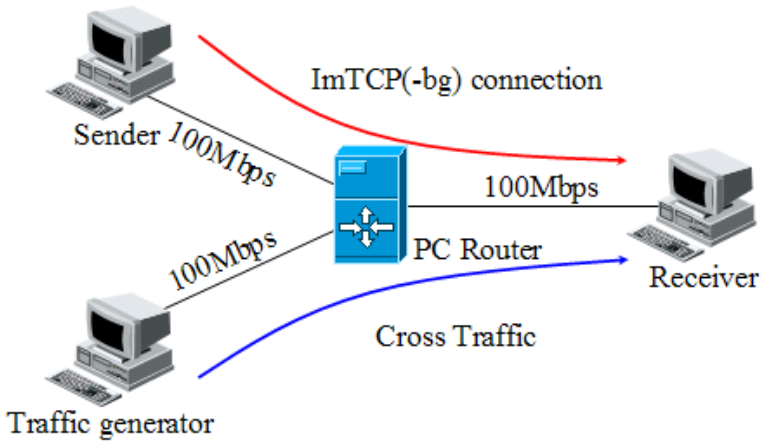


Figure 11: Experimental network environment

occur frequently and the overhead for processing interrupts affects the performance of the system. Our concept of inline network measurement means that a TCP data transfer and a bandwidth measurement task are conducted simultaneously on a single endhost, so too much large overhead for measurement should be avoided. For example, Table 2 summarizes the required time for the compilation of the kernel source code in the FreeBSD system in a PC having a 3.0-GHz CPU (Intel) and a memory of 1,024 MBytes and the upper limit of the bandwidth measurement. The results show that the processing time becomes large rapidly, meaning that the performance of the system is degraded, as HZ becomes large. Therefore, when determining the value of HZ, we should consider the trade-off relationship between the timer resolution and the performance.

5.2 Performance evaluations using an experimental network

In this section, we evaluate ImTCP and ImTCP-bg on an experimental network. Figure 11 shows the experimental network environment. This network environment consists of a PC router in which DUMMYNET is installed, an endhost that generates cross traffic (Traffic generator), an endhost that measures the available bandwidth and performs background data transfer based on the measurement result (Sender), and an endhost that receives packets from each endhost (Receiver). All endhosts and the PC router are connected by a 100-Mbps Ethernet connection. Table 3 shows the specifications of the PCs of the experimental network environment. The value of HZ at the sender host (Sender) is set to 20,000. We configured the DUMMYNET setting so that the minimum RTT

Table 3: PC specifications of the experimental network environment

	Sender	Receiver
CPU	Intel Pentium 4 3.0 GHz	Intel Pentium 4 3.4 GHz
Memory	1,024 MB	1,024 MB
OS	FreeBSD 4.10	FedoraCore 4
Network	100 Base-TX Ethernet	100 Base-TX Ethernet
	PC Router	Traffic generator
CPU	Intel Pentium 4 3.0 GHz	Intel Pentium 4 3.4 GHz
Memory	1,024 MB	1,024 MB
OS	FreeBSD 4.10	FedoraCore 4
Network	100 Base-TX Ethernet ($\times$ 3)	100 Base-TX Ethernet

Table 4: CPU load in an experimental PC

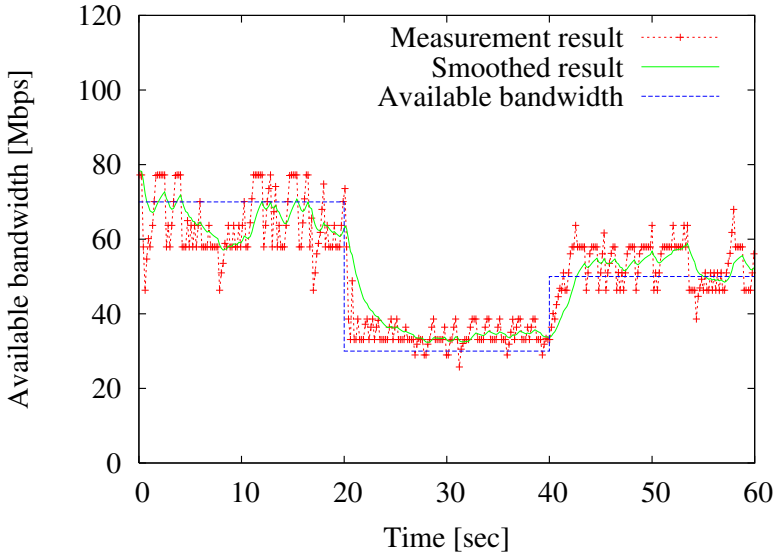
	ImTCP	TCP Reno
Average CPU load [%]	19.12	18.62

of an ImTCP(-bg) connection between the Sender and the Receiver becomes 30 msec. We first confirm the measurement accuracy of ImTCP in Subsection 5.2.1, and then evaluate the performance of ImTCP-bg in Subsection 5.2.2. The performance of ImTCP-bg is compared to those of TCP Reno and TCP-LP [23]. The source code of TCP-LP can be obtained from the TCP-LP Web page [34].

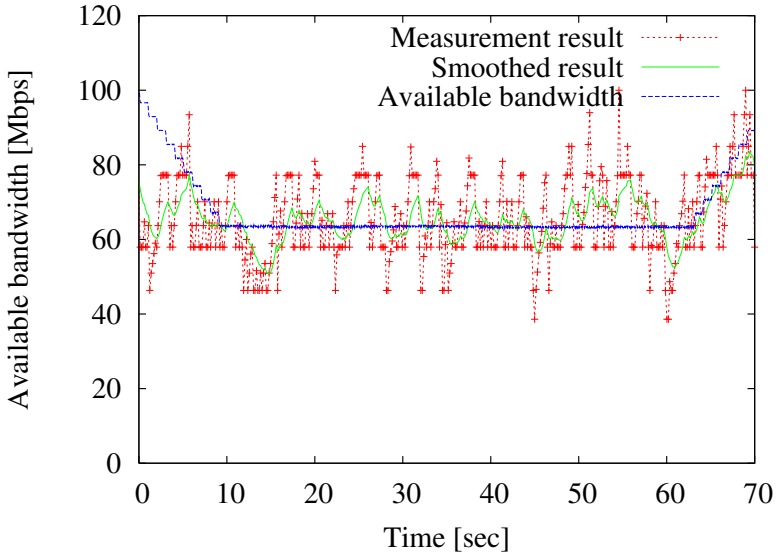
5.2.1 Evaluations of ImTCP

We conducted two types of experiments in which we utilized UDP and TCP traffic for the cross traffic between the Traffic generator and the Receiver. We then observed the measurement accuracy in each case in order to check the performance of ImTCP competing TCP traffic, which has a bursty nature. Note that the experiments with UDP traffic were conducted to check the fundamental behavior of ImTCP.

We first evaluate the measurement accuracy of ImTCP for the case when UDP traffic exists



(a) UDP cross traffic



(b) TCP cross traffic

Figure 12: Results in experimental network (1): Change of the available bandwidth and measurement results

as cross traffic. Figure 12(a) shows the measurement results of the available bandwidth for an experiment time of 60 sec. During the experiment, the rate of the cross traffic is changed so that the available bandwidth of the bottleneck link is 70 Mbps from 0 sec to 20 sec, 30 Mbps from 20 sec to 40 sec and 50 Mbps from 40 sec to 60 sec. We also plot the correct values of the available bandwidth. Figure 12(a) shows that ImTCP can measure well the available bandwidth in the experimental network. Moreover, the measurement accuracy is as high as the evaluation of simulations in [25].

We next evaluate the measurement accuracy of ImTCP for the case in which TCP traffic exists as the cross traffic. In the experiment, 10 TCP connections ($C_0, C_1, ..., C_9$) exist for generating cross traffic. C_i ($i = 0, 1, 2..., 9$) joins the network at i sec. Each connection performs data transfer for 60 seconds. We limit the maximum data transmission rate of each TCP connection to 4 Mbps by setting the receive socket buffer size at the receive host. Figure 12(b) shows the measurement results of the available bandwidth and the correct values of the available bandwidth. This figure shows that the measurement accuracy of ImTCP is as high as the result in Figure 12(a), which means that ImTCP can also measure the available bandwidth of the network path even when the TCP cross traffic exists in the network. Therefore, we can conclude that the measurement algorithm described in [25] is also effective in actual network environments.

In addition, we checked the CPU load during the experiments. Table 4 shows the average CPU loads when we use ImTCP and the original TCP Reno for the data transfer. These results show that the measurement algorithm proposed in [25] can be realized without a heavy load on the CPU.

5.2.2 Evaluations of ImTCP-bg

We performed the background data transfer using ImTCP-bg in the same network environment as in the previous subsection, and observed the utilization of the available bandwidth and the degree of interference with co-existing TCP cross traffic. As in Subsection 5.2.1, 10 TCP connections ($C_0, C_1, ..., C_9$) exist for generating cross traffic. C_i ($i = 0, 1, 2..., 9$) joins the network at i sec. Each connection performs data transfer for 60 seconds. We limited the maximum data transmission rate of each TCP connection to 4 Mbps by setting the receive socket buffer size at the receive host. Figure 13 shows the change of the congestion window size and those of the RTTs. The line "Max CWND" indicates the upper limit of the congestion window size set by the ImTCP-bg mechanism and the line "RTT threshold" is calculated as $\delta \cdot RTT_{min}$. We can see from this fig-

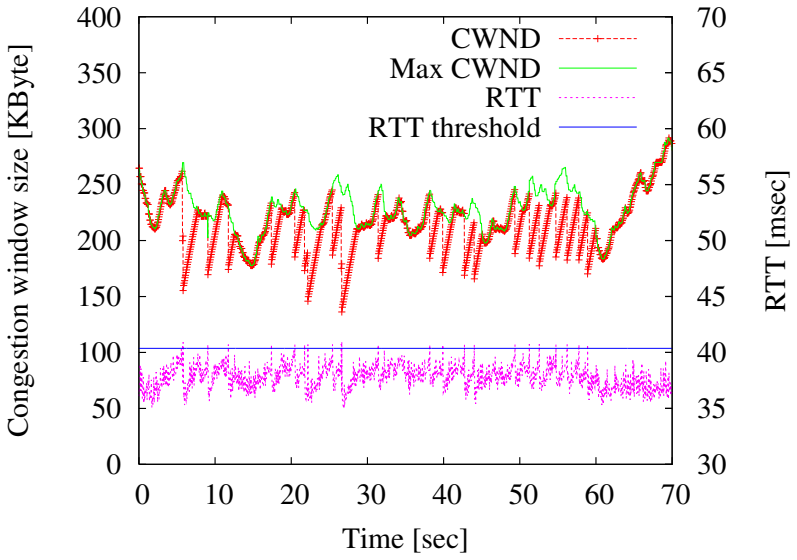


Figure 13: Results in experimental network (2): Change of congestion window size and RTT

ure that ImTCP-bg can limit the congestion window size to ”Max CWND”, which is determined by the measurement result. Moreover, when the value of RTT reaches the threshold, ImTCP-bg decrease the congestion window size based on Equation (4).

Figure 14 shows the changes in throughput of the background TCP connection, and Figure 15 shows the changes in the throughput of co-existing cross traffic. These figures also show the results for the case in which TCP Reno and TCP-LP perform background data transfer for comparison with the performance of ImTCP-bg. Figure 14 shows that although the TCP Reno connection achieves the highest throughput, the throughput exceeds the available bandwidth. Furthermore, Figure 15 shows that the throughput of cross traffic is much lower than for the case in which no background traffic exists. That is, TCP Reno cannot be used for background data transfer. On the other hand, TCP-LP and ImTCP-bg do not decrease the throughput of the cross traffic. This means that these protocols do not affect the co-existing foreground traffic. Furthermore, Figure 14 shows that the throughput of the ImTCP-bg connection is the closest to the available bandwidth. Therefore, ImTCP-bg can utilize the available bandwidth better than TCP-LP. These results clearly show that the proposed background data transfer mechanism, which utilizes the available bandwidth information obtained by the inline network measurement, performs well in the experimental network environment.

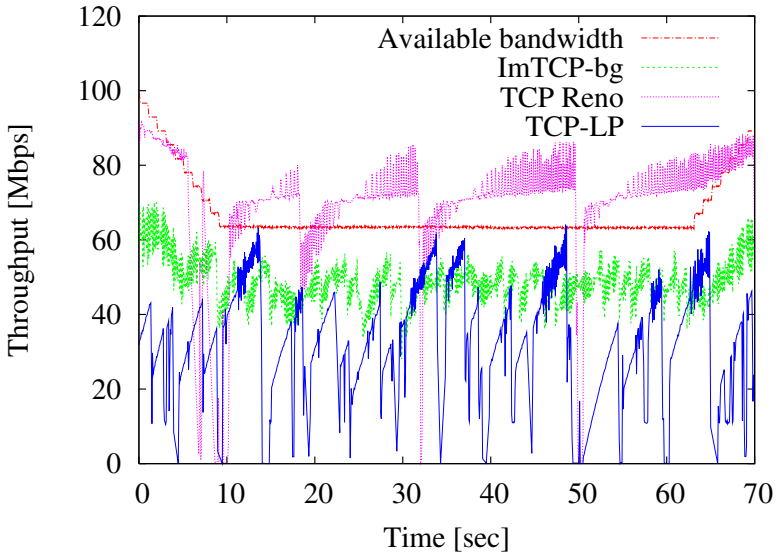


Figure 14: Results in experimental network (3): Change of throughput for a background TCP connection

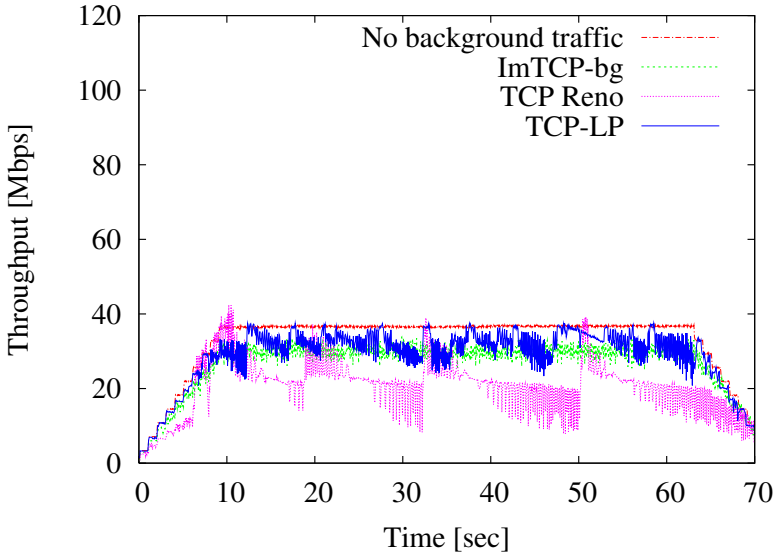


Figure 15: Results in experimental network (4): Change of throughput for cross traffic

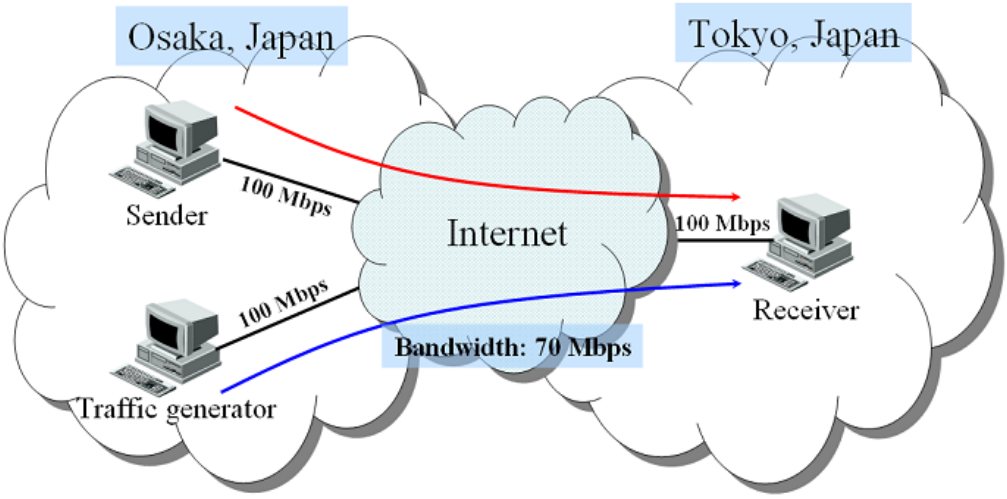


Figure 16: Network environment of the actual Internet

5.3 Experiments in the actual Internet environment

We finally confirm the performance of ImTCP and ImTCP-bg in the actual Internet. Figure 16 shows the network environment, which consists of two endhosts in Osaka, Japan and an endhost in Tokyo, Japan. We perform the data transfer and measure the available bandwidth in the network path from Osaka to Tokyo. The value of HZ at the sender host (Sender) is set to 20,000, as in Subsection 5.2. Through preliminary investigations, we confirmed the following with regard to the network between Osaka and Tokyo:

- Sixteen hops exist in the network path from Osaka to Tokyo.
- The minimum value of RTTs is 17 msec.
- The upper limit of the bandwidth between Osaka and Tokyo is 70 Mbps.

5.3.1 Experiments of ImTCP

We first check whether ImTCP can also measure well the available bandwidth in the actual Internet, as well as in the experiment network. In this experiment, we injected the UDP traffic as cross traffic. We also performed the data transfer using one ImTCP connection and measured the

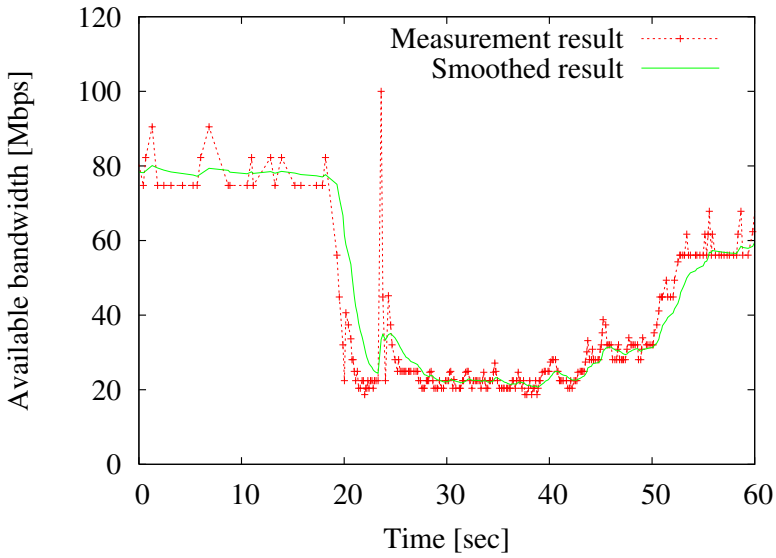


Figure 17: Results in actual network (1): Change of the available bandwidth and the measurement results

available bandwidth. Figure 17 shows the measurement results for the available bandwidth for an experimental time of 60 sec. During the experiment, we changed the rate of the cross traffic as follows: 0 bps from 0 sec to 20 sec, 50 Mbps from 20 sec to 40 sec, and 30 Mbps from 40 sec to 60 sec. We have no way to obtain the exact information about the available bandwidth of the network path. However, since we know that there the traffic in the network is relatively light, we expected that the available bandwidth would be approximately 70 Mbps from 0 sec to 20 sec, 20 Mbps from 20 sec to 40 sec, and 40 Mbps from 40 sec to 60 sec. From this figure we observe that ImTCP can measure well the available bandwidth in the actual Internet. We can observe the spike of the measurement results at around 24 sec. This may be caused by the short burst-traffic injected onto the Internet at that time. Sudden changes in the network bandwidth often occur in the Internet. When we use the measurement results obtained by ImTCP, the results are smoothed, as indicated by the line labeled ”Smoothed measurement” in the figure. This is one of the advantages of ImTCP, that is, we can smooth the sudden spikes in the observed results because ImTCP obtains the available bandwidth information continuously.

5.3.2 Experiments of ImTCP-bg

Next, we confirm the behavior of ImTCP-bg in the actual Internet environment. In this experiment, five TCP connections join the network at 60 sec and perform data transfer for 240 seconds. In addition, another five TCP connections join at 120 sec and send data for 60 sec. We limit the maximum data transmission rate of each TCP connection to 7 Mbps by setting the receive socket buffer size at the receive host. Therefore, the total throughput of the cross traffic is 0 bps from 0 sec to 60 sec, 35 Mbps from 60 sec to 120 sec, 70 Mbps from 120 sec to 180 sec, and 35 Mbps from 180 sec to 300 sec. In the network environment, we send data using ImTCP-bg for 240 seconds. We expect that the throughput of the ImTCP-bg connection is 70 Mbps from 0 sec to 60 sec, 35 Mbps from 60 sec to 120 sec, 0 bps from 120 sec to 180 sec, and 35 Mbps from 180 sec to 240 sec. We also check whether ImTCP-bg affects the cross traffic. Figure 18 shows the total throughput of the cross traffic, the throughput of ImTCP-bg and the measurement results of ImTCP, and Figure 19 shows the change of RTTs in the network path. In this figure, we can see that when the available bandwidth exists, ImTCP-bg can limit the throughput within the measurement results. When the rate of the cross traffic becomes 70 Mbps from 120 sec to 180 sec, the available bandwidth of the network path is limited. In this case, we can see from the figure that the measurement results of ImTCP are inaccurate. However, even when the measurement results are inaccurate, ImTCP-bg can decrease the amount of data transmission by using RTT-based mechanism and does not affect cross traffic. We can also see that the total throughput of the cross traffic does not decrease and the change of the RTTs are smooth. Therefore, we can confirm that the degree of interference with the cross traffic is not so large. Through these results, we conclude that the proposed background data transfer mechanism can also perform well in the actual Internet.

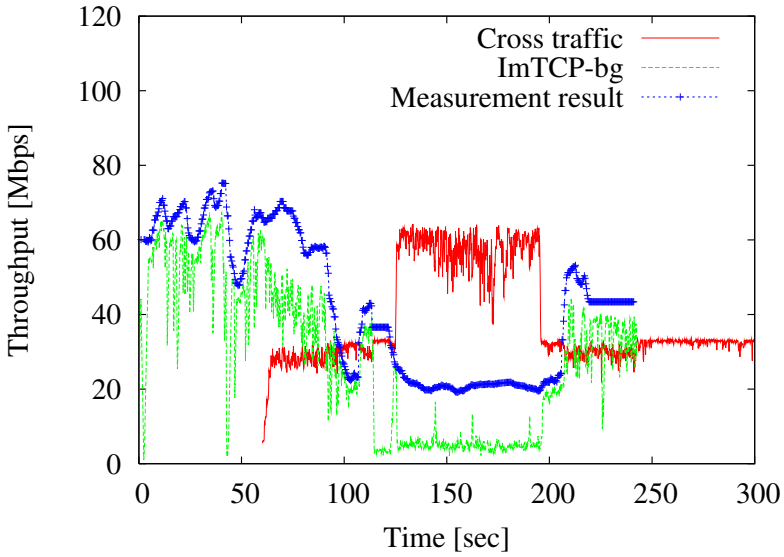


Figure 18: Results in actual network (2): Change of throughput and measurement results

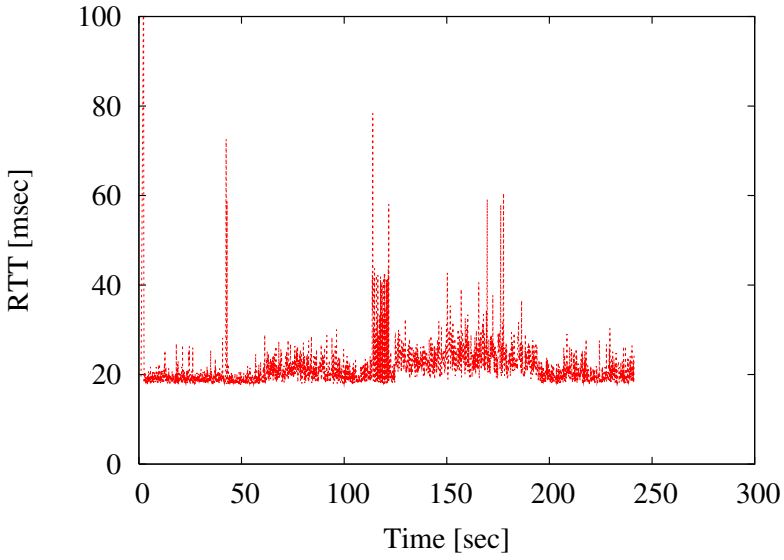


Figure 19: Results in actual network (3): Change of RTT

6 Conclusions

In this thesis, we proposed ImTCP-bg, a new TCP-based background data transfer mechanism that uses an inline network measurement technique. ImTCP-bg provides a background data transfer without interfering with other traffic by setting the upper limit of its congestion window size based on the results of the inline network measurement. ImTCP-bg also employs an enhanced RTT-based mechanism, which dynamically determines the control parameters. According to the RTT-based mechanism, ImTCP-bg can detect and resolve network congestion even when reliable measurement results cannot obtained. Through simulation evaluations, we confirmed the effectiveness of ImTCP-bg in terms of the degree of interference with foreground traffic and utilization of the available bandwidth. Moreover, we confirmed that ImTCP-bg also can utilize the available bandwidth well, while not degrading the performance of foreground traffic in an experimental network and in the actual Internet. From these experiments, we indicated the effectiveness of our concept in actual networks. The source codes of ImTCP and ImTCP-bg can be found at our web site: `http://www.anarg.jp/imtcp/`.

In future projects, we will evaluate the performance of these mechanisms in other actual network environments. In addition, we will propose other useful mechanisms based on the measurement results, and will implement and evaluate these mechanisms in actual networks.

Acknowledgements

I would like to express my gratitude to my supervisor Professor Masayuki Murata of Osaka University, for his expensive help and continuous support through my studies of this thesis.

I also wish to express my sincere appreciation to Associate Professor Go Hasegawa of Osaka University for his appropriate guidance and invaluable direct advice. All works of this thesis would not been possible without his support.

I am most grateful to Professors Koso Murakami, Makoto Imase, Teruo Higashino, Hirotaka Nakano, and Tetsuji Satoh of Osaka University, for their appropriate guidance and invaluable firsthand advice.

I am indebted to Associate Professors Hiroyuki Ohsaki, Naoki Wakamiya and Research Assistants Shin'ichi Arakawa and Masahiro Sasabe of Osaka University, who gave me helpful comments and feedback.

I also wish to thank Cao Le Thanh Man and Kazunari Mori of Osaka University, who gave me useful comments and helped conduct the experiments in actual networks. Their helps greatly contributed to my thesis.

Finally, I want to say thanks to 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. Our conversations and work together have greatly influenced this thesis.

References

[1] B. Krishnamurthy, C. Wills, and Y. Zhang, "On the use and performance of content distribution networks," in *Proceedings of ACM SIGCOMM 2001 Internet Measurement Workshop*, Nov. 2001.

[2] S. Saroiu, K. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy, "An analysis of Internet content delivery systems," in *Proceedings of OSDI 2002*, Dec. 2002.

[3] G. Pierre and M. van Steen, "Design and implementation of usercentered content delivery network," in *Proceedings of the 3rd IEEE Workshop on Internet Applications*, June 2003.

[4] Akamai Home Page. available at `http://www.akamai.com/`.

[5] A. Rao, K. Lakshminarayanan, S. Surana, and I. Stoica, "Load balancing in structured P2P systems," in *Proceedings of IPTPS 2003*, Feb. 2003.

[6] F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica, "Towards a common API for structured peer-to-peer overlays," in *Proceedings of IPTPS 2003*, Feb. 2003.

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

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

[9] 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.

[10] J. B. Postel, "Transmission control protocol," *RFC 793*, Sept. 1981.

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

[12] 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 1999*, Mar. 1999.

[13] M. Crovella and P. Barford, "The network effects of prefetching," in *Proceedings of the IEEE INFOCOM 1998*, Mar. 1998.

[14] A. Venkataramani, P. Yalagandula, R. Kokku, S. Sharif, and M. Dahlin, "The potential costs and benefits of long term prefetching for content distribution," *Computer Communication Journal*, vol. 25, pp. 367–375, Mar. 2002.

[15] Microsoft Corporation, *Background Intelligent Transfer Service in Windows Server 2003*, Sept. 2002. available at `http://www.microsoft.com/windowsserver2003/techinfo/overview/bits.mspx`.

[16] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Wide-area cooperative storage with CFS," in *Proceedings of SOSP 2001*, Oct. 2001.

[17] A. Rowstron and P. Druschel, "Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility," in *Proceedings of SOSP 2001*, Oct. 2001.

[18] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," *RFC 2475*, Dec. 1998.

[19] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured forwarding PHB group," *RFC 2597*, June 1999.

[20] R. Kokku, P. Yalagandula, A. Venkataramani, and M. Dahlin, "NPS: A non-interfering deployable web prefetching system," in *Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems*, Mar. 2003.

[21] P. Key, L. Massoulie, and B. Wang, "Emulating low-priority transport at the application layer: A background transfer service," *ACM SIGMETRICS Performance Evaluation Review*, vol. 32, pp. 118–129, June 2004.

[22] A. Venkataramani, R. Kokku, and M. Dahlin, "TCP Nice: A mechanism for background transfers," in *Proceedings of OSDI 2002*, Dec. 2002.

[23] A. Kuzmanovic and E. W. Knihtly, "TCP-LP: A distributed algorithm for low priority data transfer," in *Proceedings of IEEE INFOCOM 2003*, Apr. 2003.

[24] C. L. T. Man, G. Hasegawa, and M. Murata, “A new available bandwidth measurement technique for service overlay networks,” in *Proceedings of IFIP/IEEE MMNS 2003 E2EMON Workshop*, Sept. 2003.

[25] C. L. T. Man, G. Hasegawa, and M. Murata, “Available bandwidth measurement via TCP connection,” in *Proceedings of IFIP/IEEE MMNS 2004 E2EMON Workshop*, Oct. 2004.

[26] FreeBSD Home Page. available at `http://www.freebsd.org/`.

[27] R. L. Carter and M. E. Crovella, “Measuring bottleneck link speed in packet-switched networks,” *International Journal on Performance Evaluation*, vol. 27–28, pp. 297–318, Oct. 1996.

[28] 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.

[29] 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.

[30] C. Dovrolis and D. Moore, “What do packet dispersion techniques measure ?,” in *Proceedings of IEEE INFOCOM 2001*, Apr. 2001.

[31] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cottrell, “pathChirp: Efficient available bandwidth estimation for network paths,” in *Proceedings of PAM 2003 Workshop*, Apr. 2003.

[32] The VINT Project, “UCB/LBNL/VINT network simulator - ns (version 2).” available at `http://www.isi.edu/nsnam/ns/`.

[33] M. K. McKusick and G. V. Neville-Neil, *The Design And Implementation Of The FreeBSD Operating System*. Addison-Wesley, 2004.

[34] TCP-LP Home Page. available at `http://www-ece.rice.edu/networks/TCP-LP/`.