Implementation and evaluation of an inline network measurement algorithm and its application to TCP-based service

Tomoaki Tsugawa, Go Hasegawa, and Masayuki Murata
Graduate School of Information Science and Technology, Osaka University
1–5 Yamadaoka, Suita, Osaka, 560–0871, Japan

Abstract— **In our previous studies, we proposed ImTCP, an inline network measurement technique that can obtain available bandwidth information of the network path between sender and receiver hosts continuously and in an inline fashion. We also introduced ImTCP-bg, a background TCP data transfer mechanism based on the measurement results of ImTCP. In the present paper, we report implementation issues of the proposed mechanisms on a FreeBSD system and evaluate them on an experiment network and in the actual Internet. We investigate the performance through the extensive experiments and verify that ImTCP can measure well the available bandwidth of the network path, independent of the degree of its change, and that ImTCP-bg can utilize the available bandwidth well, without degrading the performance of competing traffic. From these experiment results, we confirm the effectiveness of our concept, which is the inline network measurement technique, in an actual network.**

I. 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 coexist in the Internet. Referred to as service overlay networks, these networks include Content Delivery/Distribution Networks (CDNs) [1, 2], P2P network [3, 4], Grid network [5, 6], and IP-VPN [7]. Service overlay networks are upper-layer networks that provide special-purpose services built onto the lower-layer IP network. Under these circumstances, we believe that information concerning bandwidth availability in a network path is important in adaptive control of the network. For example, in P2P networks, when a resource discovery mechanism finds multiple peers having the same requested contents, the bandwidth information is used to determine which peer should transmit the content. Transmission Control Protocol (TCP) [8], which is a major network transport protocol, can use such information to optimize link utilization [9] or improve transmission performance [10].

The term "available bandwidth" in this paper means the bandwidth information that is the unused portion of the network path between sender and receiver hosts. Although a large number of studies have examined methods for measuring the available bandwidth [11–13], these algorithms have fundamental disadvantages, when we intend to use them for service overlay networks. It includes the disadvantage that many probe packets are sent at a high transmission rate. For instance, PathLoad [12] sends several 100-packet measurement streams for a measurement. PathChirp [13] 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. 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 measurement algorithms also require a long time to obtain one measurement result. Long-term measurement can provide an accurate result but cannot follow the dynamic traffic in the IP network. Against the above problems, our research group has proposed a novel inline network measurement technique, ImTCP [14], which can continuously obtain the available bandwidth information of the network path between sender and receiver hosts. 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 well.

We have also proposed the background TCP data transfer, which is referred to as ImTCP-bg [15]. This application technique is based on the measurement results of ImTCP. When background data transfer is realized, the quality of several network services can be improved. For example, in CDNs, Web servers transfer various types of data (e.g., backup, caching [16], and prefetching [17, 18]), in addition to the data transferred in response to the document transfer request from Web clients. In this case, the user-requested data can be transferred quickly, while the other data are transferred in the background. Generally, background data transfer should satisfy the following two objectives:

1. no adverse effect on other traffic
2. full utilization of the network link bandwidth

In previous studies, several transport-layer approaches have been introduced in order to handle background data transfer [19, 20]. These approaches observe the Round Trip Times (RTTs) of the data packets of 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. Although these approaches can realize data transfer without affecting other traffic, they are unable to utilize the available bandwidth of the network efficiently because the degree to which the congestion window size is decreased when the RTTs increase is fixed and is too large, regardless of the network condition. On the other hand, ImTCP-bg can achieve both of the above two objectives by controlling the congestion window based on the measurement result of the available bandwidth of the network path.

In [14, 15], the effectiveness of the proposed mechanisms

(ImTCP and ImTCP-bg) are evaluated through the simulation experiments. Ns-2 [21] is used for the performance evaluation. In [14], ImTCP is confirmed to be able to measure well the available bandwidth, independent of the degree of change in available bandwidth. Furthermore, ImTCP can also measure the available bandwidth continuously using only a small number of packets. We also confirmed that ImTCP preserves the characteristics of the original TCP and maintains TCP compatibility. In [15], background TCP data transfer based on the measurement results is confirmed to be able to effectively utilize the bandwidth that is unused by the other traffic, while not degrading the performance of the other traffic. However, simply evaluating the effectiveness of the inline network measurement algorithm and its application technique is insufficient. 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 mechanisms.

In the present paper, we implement the inline network measurement algorithm, ImTCP, proposed in [14] and the background TCP data transfer mechanism based on the measurement results, ImTCP-bg, proposed in [15]. We evaluate the effectiveness of these mechanisms on actual networks. We implement ImTCP and ImTCP-bg in a FreeBSD 4.10 [22] kernel system, and evaluate the measurement accuracy of ImTCP and the performance of ImTCP-bg in terms of the utilization of the available bandwidth and the degree of interference with other traffic in the experimental network. Through these experiments, we confirm the effectiveness of the concept, *inline network measurement*, on actual networks.

The remainder of this paper is organized as follows. In Section II, we briefly introduce the algorithms of ImTCP and ImTCP-bg. Section III describes the outline of implementation design in the FreeBSD 4.10 kernel system. In Section IV, we present the results of the implementation experiments in order to evaluate the performance of the ImTCP and ImTCP-bg mechanisms. Section V describes the performance of the proposed mechanism in an actual Internet environment. Finally, we present conclusions and areas for future study in Section VI.

II. Algorithms for ImTCP and ImTCP-bg

In this section, we introduce the algorithms of ImTCP and its application technique, ImTCP-bg. The ImTCP algorithm is described in detail in [14], and the ImTCP-bg algorithm is described in detail in [15].

A. ImTCP algorithm

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.

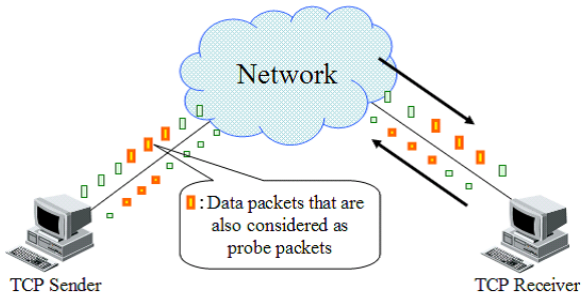

Fig. 1. Inline network measurement by ImTCP

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 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 [23] 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.
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

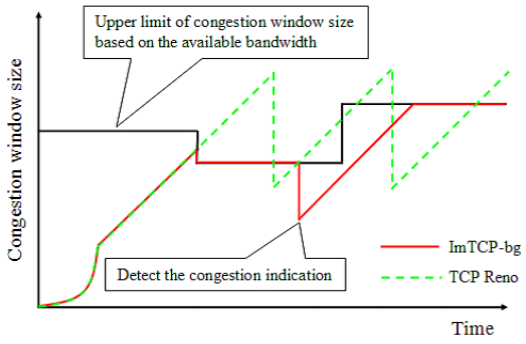

Fig. 2. Change of congestion window size in TCP Reno and ImTCP-bg

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.

B. ImTCP-bg algorithm

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 the ImTCP mechanism, γ is a smoothing parameter, and $\bar{A}$ is the smoothed available bandwidth. 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.

The effectiveness of the above-described mechanism depends largely on the accuracy of the measurement by ImTCP of the available bandwidth. 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 to this, when the 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.

If the value of RTT tends to increase despite controlling the congestion window by Equation (2), then the measurement result is inaccurate and ImTCP-bg cannot perform background data transfer. Therefore, ImTCP-bg observes the change of RTTs and decreases the congestion window size. ImTCP-bg detects network congestion using the current and minimum values of RTT. Note that we do not use the maximum RTT value. When an increase in RTT is detected, the ImTCP-bg sender decreases its congestion window size immediately in order to resolve the congestion. 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 by which to judge whether network congestion occurs. The ImTCP-bg sender detects network congestion when the following condition is satisfied:

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

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

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

where $cwnd$ 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. As such, ImTCP-bg avoids unnecessary the underutilization of the link bandwidth while maintaining the background-based data transfer.

III. Implementation of ImTCP and ImTCP-bg

In this section, we describe the outline of the implementation of ImTCP and ImTCP-bg in the FreeBSD 4.10 kernel system. In addition, we discuss the resolution of the kernel timer, which is an important issue when implementing packet-pair/train based measurement mechanisms.

A. 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 3. 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.

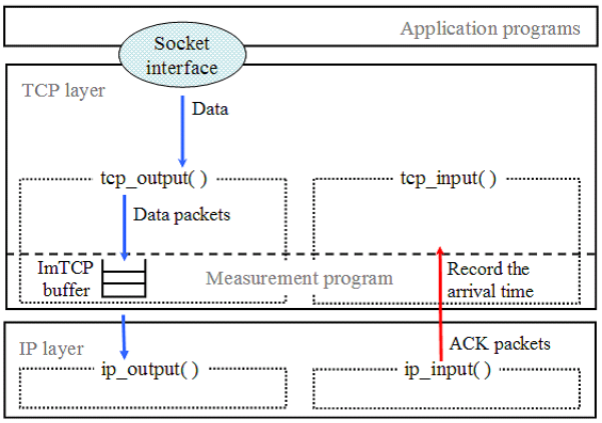

Fig. 3. Outline of ImTCP architecture

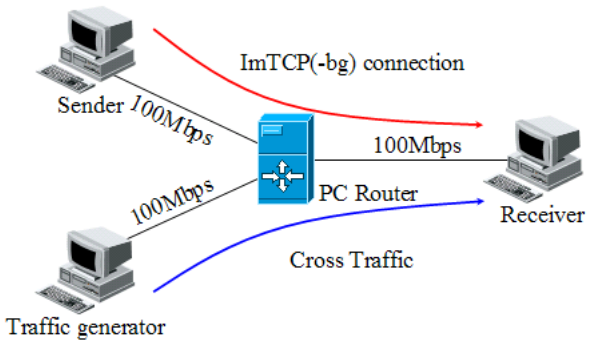

Fig. 4. Experimental network environment

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 Section II-A.

B. 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 value of $snd\_maxcwnd$ to snd_cwnd.

C. 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 III-A. 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 [24]. 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 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 I 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.

TABLE I

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

TABLE II

PC SPECIFICATIONS OF THE EXPERIMENTAL NETWORK ENVIRONMENT

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

IV. PERFORMANCE EVALUATIONS USING AN EXPERIMENTAL NETWORK

In this section, we evaluate ImTCP and ImTCP-bg on an experimental network. Figure 4 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 II 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 of an ImTCP(-bg) connection between the Sender and the Receiver becomes 30 msec. We first evaluate the measurement accuracy of ImTCP in Subsection IV-A, and then evaluate the performance of ImTCP-bg in Subsection IV-B. The performance of ImTCP-bg is compared to those of TCP Reno and TCP-LP [20], which is the previously proposed background data transfer mechanism. The source code of TCP-LP can be obtained from the TCP-LP Web page [25].

A. 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.

A.1 Case of UDP cross traffic

We first evaluate the measurement accuracy of ImTCP for the case when UDP traffic exists as cross traffic. Figure 5 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 5 shows that ImTCP can measure well the available bandwidth in the experimental network. Moreover, the measurement accuracy is as high as the evaluation of simulation experiments in [14]. Therefore, we can conclude that the measurement algorithm described in [14] is also effective in actual network environments. In addiiton, we checked the CPU load during the experiments. Table III 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 [14] can be realized without a heavy load on the CPU.

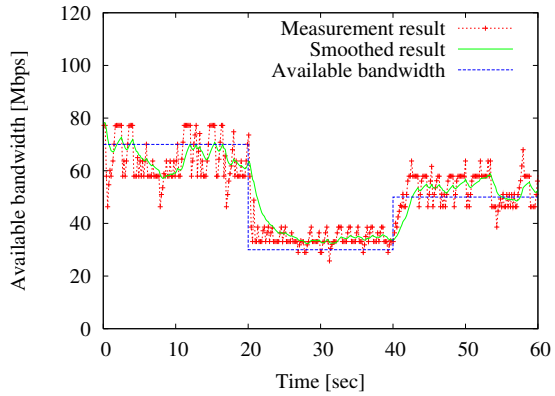

Fig. 5. Change of the available bandwidth and the measurement result (UDP cross traffic case)

A.2 Case of TCP cross traffic

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

TABLE III
CPU LOAD

	ImTCP	TCP Reno
Average CPU load [%]	19.12	18.62

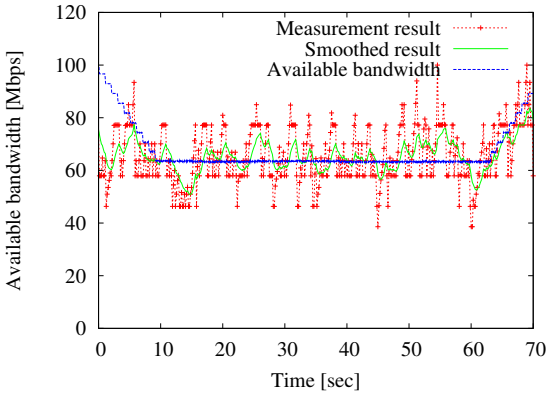

Fig. 6. Change of the available bandwidth and the measurement result (TCP cross traffic case)

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 6 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 5, which means that ImTCP can also measure the available bandwidth of the network path even when the TCP cross traffic exists in the network.

B. 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 Subsubsection IV-A.2, 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 advertised window size at the receive host. Figure 7 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 figure 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 8 shows the changes in throughput of the background TCP connection, and Figure 9 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

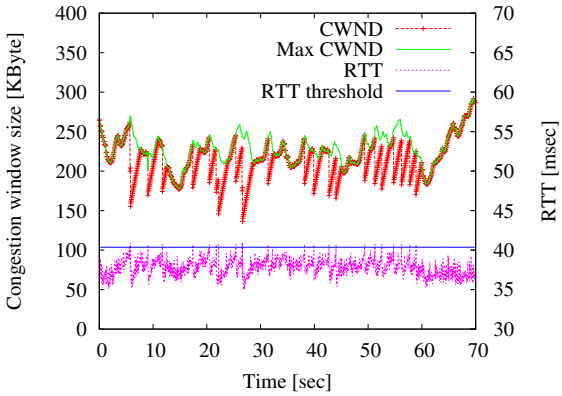

Fig. 7. Changes of congestion window size and RTT

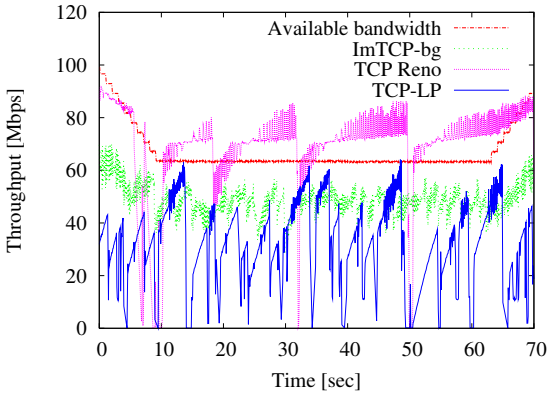

Fig. 8. Change of throughput for a background TCP connection

background data transfer for comparison with the performance of ImTCP-bg. Figure 8 shows that although the TCP Reno connection achieves the highest throughput, the throughput exceeds the available bandwidth. Furthermore, Figure 9 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 8 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.

V. EXPERIMENTS IN THE ACTUAL INTERNET

We finally confirm the performance of ImTCP and ImTCP-bg in the actual Internet. Figure 10 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.

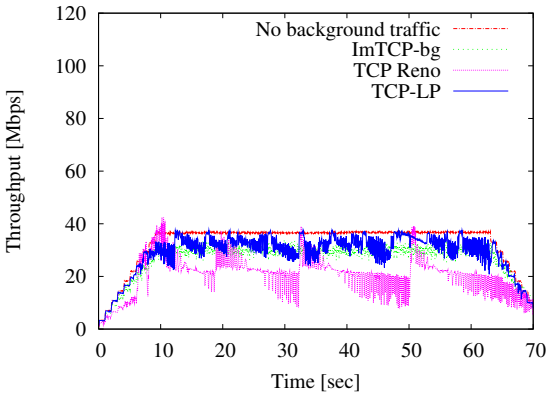

Fig. 9. Change of throughput for cross traffic

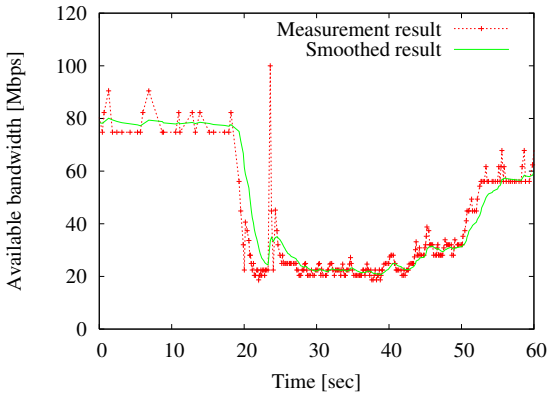

Fig. 11. Change of the available bandwidth and the measurement results

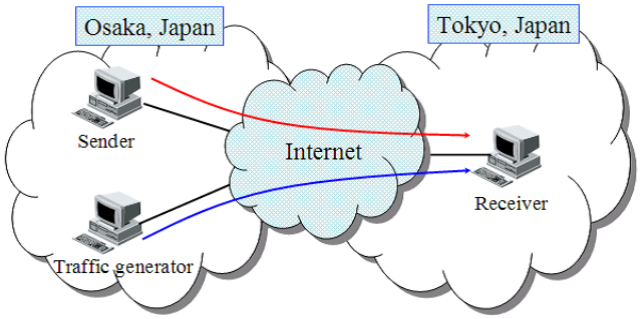

Fig. 10. Network environment of the actual Internet

The value of HZ at the sender host (Sender) is set to 20,000, as in Section IV. 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.

A. 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 available bandwidth. Figure 11 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.

B. 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 to this, 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 12 shows the total throughput of the cross traffic, the throughput of ImTCP-bg and the measurement results of ImTCP, and Figure 13 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

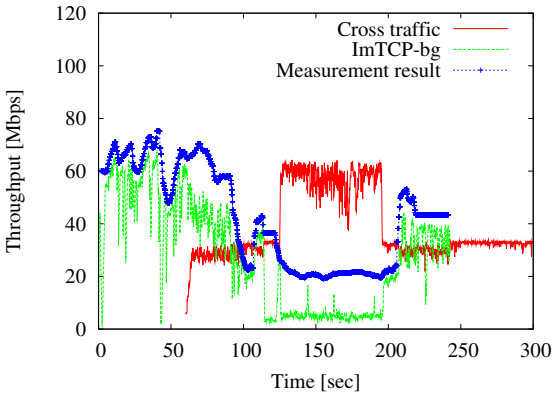

Fig. 12. Change of throughput and measurement results

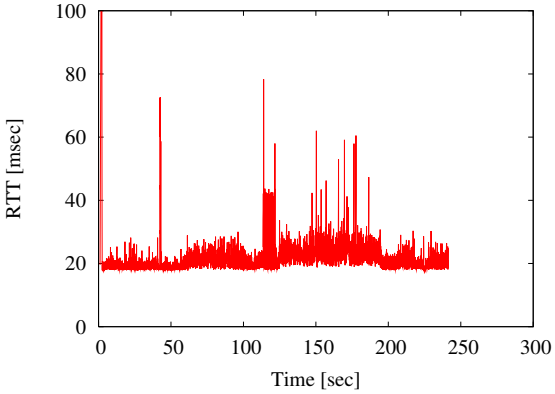

Fig. 13. Change of RTT

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.

VI. Conclusions

In the present paper, we implemented an inline network measurement algorithm, ImTCP, and its application technique, ImTCP-bg. Through evaluation in an experimental network, we confirmed that ImTCP can measure well the available bandwidth, independent of the degree of change in available bandwidth. We also confirmed that ImTCP-bg can utilize the available bandwidth well, while not degrading the performance of other traffic. Moreover, we confirmed the basic performance of these mechanisms in the actual Internet. The source codes of ImTCP and ImTCP-bg can be found at our web site: `http://www.anarg.jp/imtcp/`.

In future studies, 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

The authors wish to thank Man Le Thanh Cao, for his helpful suggestions and comments, and Kazunari Mori for helping conduct the experiments in the actual Internet.

References

[1] 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.
[2] Akamai Home Page. available at `http://www.akamai.com/`.
[3] A. Rao, K. Lakshminarayanan, S. Surana, and I. Stoica, "Load balancing in structured P2P systems," in *Proceedings of IPTPS 2003*, Feb. 2003.
[4] 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.
[5] 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.
[6] 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.
[7] 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.
[8] J. B. Postel, "Transmission control protocol," *RFC 793*, Sept. 1981.
[9] R. Wang, G. Pau, K. Yamada, M. Sanadidi, and M. Gerla, "TCP startup performance in large bandwidth delay networks," in *Proceedings of INFOCOM 2004*, Mar. 2004.
[10] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," in *Proceedings of ACM SIGCOMM 2002*, Aug. 2002.
[11] 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.
[12] 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.
[13] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cottrell, "pathChirp: Efficient available bandwidth estimation for network paths," in *Proceedings of NLANR PAM 2003*, Apr. 2003.
[14] M. L. T. Cao, G. Hasegawa, and M. Murata, "Available bandwidth measurement via TCP connection," in *Proceedings of IFIP/IEEE MMNS 2004 E2EMON Workshop*, Oct. 2004.
[15] T. Tsugawa, G. Hasegawa, and M. Murata, "Background TCP data transfer with inline network measurement," in *Proceedings of APCC 2005*, Oct. 2005.
[16] 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.
[17] M. Crovella and P. Barford, "The network effects of prefetching," in *Proceedings of the IEEE INFOCOM 1998*, Mar. 1998.
[18] 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.
[19] A. Venkataramani, R. Kokku, and M. Dahlin, "TCP Nice: A mechanism for background transfers," in *Proceedings of OSDI 2002*, Dec. 2002.
[20] A. Kuzmanovic and E. W. Knihtly, "TCP-LP: A distributed algorithm for low priority data transfer," in *Proceedings of IEEE INFOCOM 2003*, Apr. 2003.
[21] The VINT Project, "UCB/LBNL/VINT network simulator - ns (version 2)." available at `http://www.isi.edu/nsnam/ns/`.
[22] FreeBSD Home Page. available at `http://www.freebsd.org/`.
[23] 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.
[24] M. K. McKusick and G. V. Neville-Neil, *The Design And Implementation Of The FreeBSD Operating System*. Addison-Wesley, 2004.
[25] TCP-LP Home Page. available at `http://www-ece.rice.edu/networks/TCP-LP/`.