Implementation Experiments of the TCP Proxy Mechanism

Kana YAMANEGI†, Takayuki HAMA††, Go HASEGAWA†,
Masayuki MURATA†, Hideyuki SHIMONISHI††, and Tutomu MURASE††

†Osaka University, 1-5 Yamadaoka, Suita, Osaka 565-0871, Japan
††NEC Corporation,1753 Shimonumabe, Nakahara-ku, Kawasaki, Kanagawa, 211-8666, Japan

E-mail: †{k-yamanegi, hasegawa, murata}@ist.osaka-u.ac.jp,
††{t-hama@cb, h-shimonishi@cd, t-murase@ap}.jp.nec.com

Abstract

Interest in the TCP overlay network, which controls the data transmission quality at the transport layer, has grown as the user demand for sophisticated and diversified services in the Internet has increased. In the TCP overlay network, TCP proxy is a fundamental mechanism that transparently splits a TCP connection between sender and receiver hosts into multiple TCP connection at some nodes in the network and relays data packets from the sender host to the receiver host via the split TCP connections. In the present paper, we investigate the performance of the TCP proxy mechanism through experiments using the actual public network. The TCP proxy mechanism is shown to enhance the data transfer throughput without any change in the TCP/IP protocol stack of the endhost. In addition, we evaluate the performance of gentle High-Speed TCP, as proposed in a previous study, on the TCP connection between the TCP proxy nodes.

1 Introduction

The remarkable degree to which the Internet has grown is due in part to access/backbone network technologies such as xDSL and optical fiber. In addition, user demand for diversified services has increased due to the rapid growth of the Internet population. Some of these applications require high-quality transport services in terms of, for example, end-to-end throughput, packet loss ratio, and delay. However, data transmission quality across the current Internet cannot be assured, essentially because of the best-effort basis of the Internet.

IntServ [1] and DiffServ [2] are possible solutions to this problem that add control mechanisms at the network layer. For example, the DiffServ architecture is based on a simple model in which traffic entering a network is classified, and possibly conditioned, at the boundaries of the network and is then assigned to different behavior aggregates. However, implementation of DiffServ architecture would require additional mechanisms to be deployed to all routers through which traffic-flows traverse in order to achieve sufficient benefit from the introduction of IntServ/DiffServ into the network. Therefore, due to factors such as scalability and cost, we believe that these schemes have almost no chance of being deployed on large-scale networks.

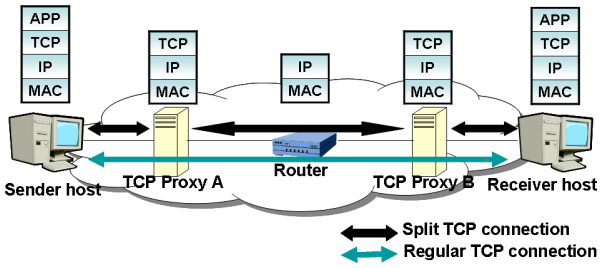

Figure 1: TCP overlay network

Proxy cache servers in Contents Delivery Networks (CDNs) [3] and media streaming in P2P (Peer-to-Peer) networks are typical examples of the overlay networking approach. One disadvantage of such methods is the need for complicated control mechanisms specific to each application. In addition, parameter setting is very sensitive to various network factors.

We are now investigating TCP overlay network architecture [4], which controls data transmission quality at the transport layer, meaning that the IP layer remains providing only minimum fundamental functions, such as routing and packet forwarding. One of the important mechanisms of TCP overlay networks is to divide the end-to-end TCP connection into multiple split TCP connections and relay data packets from the sender host to the receiver host via the split TCP connections (Figure 1). In the present paper, we refer to this splitting mechanism as TCP Proxy. TCP Proxy is expected to enhance the end-to-end data transfer throughput, mainly because the feedback-loop of the TCP connection becomes short, meaning that the round trip time and packet loss ratio of each split TCP connection is reduced.

In some previous studies [4, 5], we confirmed the effect of the TCP proxy mechanism from simulation and mathematical analysis results. We found that, with a TCP proxy mechanism, the end-to-end throughput of data transmission is increased and the file transfer delay is shortened. In the present paper, we investigate the performance of the TCP proxy mechanism by con-

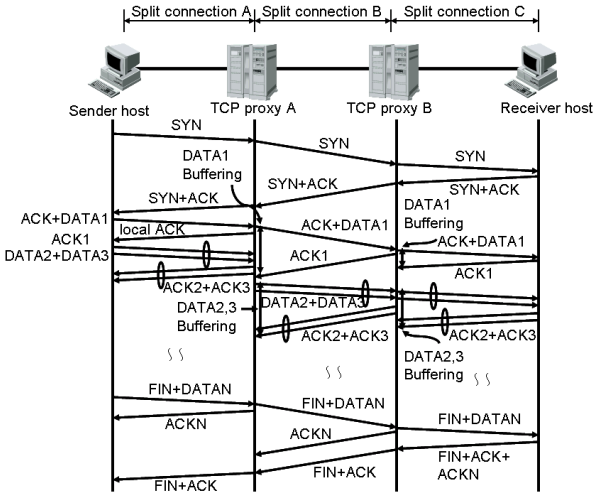

Figure 2: TCP proxy mechanism

ducting experiments using the public network. Furthermore, we also evaluate the performance of TCP variants for high-speed networks [6, 7] when used on a TCP connection between TCP proxy nodes.

The remainder of the present paper is organized as follows. In Section 2, we explain the TCP proxy mechanism and High-Speed TCP. In Section 3, we explain the environment and settings used in the experimental evaluation and present experimental results for the characteristics of the actual public network used in the present study. We evaluate the effect of TCP proxy and High-Speed TCP in Section 4. Section 5 summarizes the conclusions of the present study and discusses areas for future consideration.

2 Research Background

2.1 TCP proxy mechanism

TCP proxy is a fundamental mechanism that splits a TCP connection between the sender and receiver hosts into multiple TCP connection at some nodes in the network. TCP proxy modes relay data packets from the sender host to the receiver host via the split TCP connections. TCP proxy also use a local ACK packet; a TCP proxy node sends back a pseudo ACK packet to the upward sender/proxy when it receives a data packet, without waiting to receive an ACK packet from the downward receiver/proxy. TCP proxy is expected to improve the data transfer throughput of connections by shortening the RTT. Furthermore, a TCP proxy has send/receive socket buffers for storing data packets, just like a regular TCP host. When a data packet is lost between the TCP proxy and the receiver host, the dropped packets can be retransmitted from the TCP proxy instead of the sender host. TCP proxy is also expected to improve data transfer performance, compared to regular TCP connections. Figure 2 depicts the mechanisms used in processing and forwarding TCP packets via split TCP connections, where there are two proxy nodes between the sender and receiver hosts and three split TCP connections are used. One advantage of the TCP proxy mechanism is its transparent behavior. TCP connections traversing TCP proxy nodes are split automatically, and there is no need to modify the protocol stack of sender/receiver hosts.

2.2 High-Speed TCP

In high-speed networks having bandwidths greater than 100 Mbps, obtaining sufficient throughput by TCP based on TCP Reno is difficult, as pointed out in [6]. Therefore, a number of TCP modifications related to High-Speed TCP that are capable of achieving high throughput by modifying the congestion control algorithm have been proposed in [6–9]. In the present paper, we evaluate the performance of High-Speed TCP (HSTCP) [6] and its improvement variant, gentle High-Speed TCP (gHSTCP) [7], on the TCP connection between TCP proxy nodes. The expected benefit of using HSTCP/gHSTCP between TCP proxy nodes is that the advantage of HSTCP/gHSTCP can be obtained, while retaining the TCP/IP stack of the sender/receiver endhosts.

2.2.1 High-Speed TCP (HSTCP)

To overcome the problems inherent in TCP, HSTCP was proposed in [6]. The HSTCP algorithm employs the principle of Additive Increase Multiplicative Decrease (AIMD), as in standard TCP, but HSTCP is more aggressive in its increases and more conservative in its decreases. HSTCP addresses this by altering the AIMD algorithm for the congestion window adjustment, making it a function of the congestion window size rather than a constant, as is the case in standard TCP. That is, the “ increase ”parameter becomes larger, and the“ decrease ”parameter becomes smaller, as the congestion widow size increases (Figure 3). In this way, HSTCP can sustain a large congestion window and fully utilize the high-speed long-delay network. HSTCP is described in detail in [6].

However, a number of problems regarding HSTCP have been reported in [7]. For example, the relative fairness between standard TCP and HSTCP worsens as the link bandwidth increases. When HSTCP and TCP Reno compete for bandwidth on a bottleneck link, we do not attempt to provide the same throughput that they are capable of achieving. However, in this case, high throughput by HSTCP should not occur by excessively sacrificing TCP Reno throughput, i.e., HSTCP should not pillage too many resources at the expense of TCP Reno.

2.2.2 Gentle High Speed TCP (gHSTCP)

Based on HSTCP, gHSTCP, as proposed in [7], can achieve better fairness with competing traditional TCP flows, while ex-

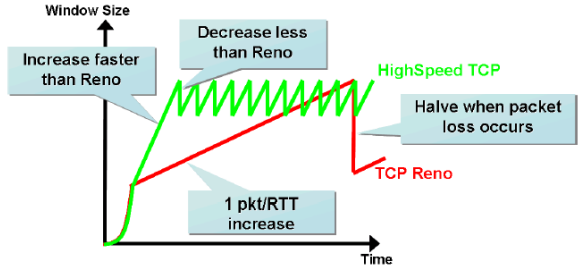

Figure 3: Change of cwnd using HSTCP

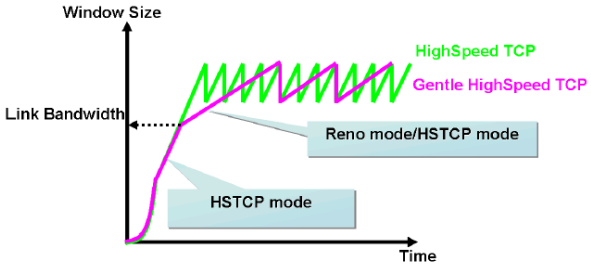

Figure 4: Change of cwnd using gHSTCP

tending the advantage of high throughput provided by HSTCP. The original HSTCP increases the congestion window size based solely on the current congestion window size. This may lead to bursty packet losses because the window size continues to increase rapidly even when packets begin to be queued at the router buffer. In addition, differences in speed gains among the different TCP variants result in unfairness. To alleviate this problem, gHSTCP changes the behavior of HSTCP for speed increases so as to account for full or partial utilization of bottleneck links. In addition, gHSTCP regulates the congestion avoidance phase in two modes and switches between these modes based on the trend of changing RTT. When an increasing trend in the observed RTT values occurs, gHSTCP adopts the congestion control algorithm of TCP Reno (Figure 4). This is expected to reduce the rate of packet loss in the buffer of routers and improve fairness with TCP Reno. gHSTCP is described in detail in [7].

3 Experimental Environment

3.1 Experimental Network and Settings

We prepared the public Internet environment between Tokyo and Osaka, as depicted in Figure 5.

There are two TCP proxies in Tokyo network and Osaka network, and the TCP connection between the sender and receiver hosts is split into three TCP connections when the TCP proxy mechanism is activated. We compare the data transfer throughput using a single TCP connection between the sender and receiver hosts (Case 1), and that using the split TCP connections by TCP proxy 1 and TCP proxy 2 (Case 2). In case 1, data is transferred between Osaka B and Tokyo B. In Case 2, the TCP connection between Osaka A and Tokyo A is split in three connections (Osaka A - TCP proxy 1, TCP proxy 1 - TCP proxy 2, TCP proxy 2 - Tokyo A), and data is relayed via the split TCP connections with TCP proxy mechanism. Furthermore, we used a network emulator at Tokyo network to emulate the long-delay network between the sender and receiver hosts. We tested the Osaka-Tokyo case with no delay emulated at the network emulator, and the Okinawa-Tokyo case with a 25-msec delay.

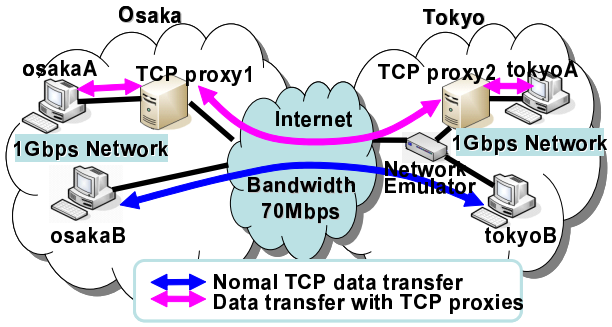

Figure 5: Experimental environment

In the experiment, we inject TCP traffic into the network using the measurement tool iperf [10] and measure the average throughput at the receiver host. Note that we define the throughput as the amount of data arriving at the receiver host per unit time. In addition, we measure the round-trip times (RTTs) and congestion window size (cwnd) using the log of `/proc/net/tcp` in Linux system in order to analyze the detailed TCP behavior.

In the present paper, we show the experimental result for the case of using the hosts at Osaka (Osaka A, Osaka B) as the sender hosts and using the hosts at Tokyo (Tokyo A, Tokyo B) as the receiver hosts, because the connection from Osaka to Tokyo could obtain higher throughput than the connection in the opposite connection, indicating less background traffic.

3.2 Investigation of Experimental Environment

First, we show a number of experimental results using regular TCP traffic in order to investigate the characteristics of the actual public network used in the present study. In this experiment, we used multiple TCP connections simultaneously and checked the average throughput of two minutes data transmissions every

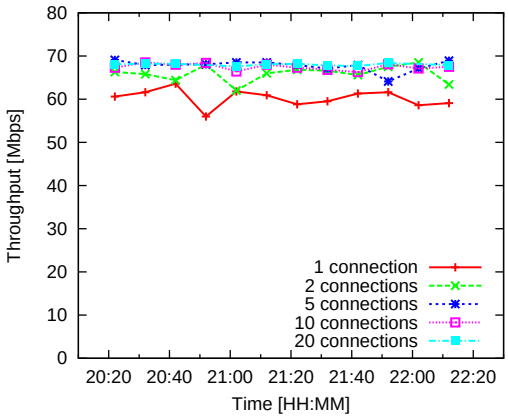

Figure 6: Average throughput with normal TCP

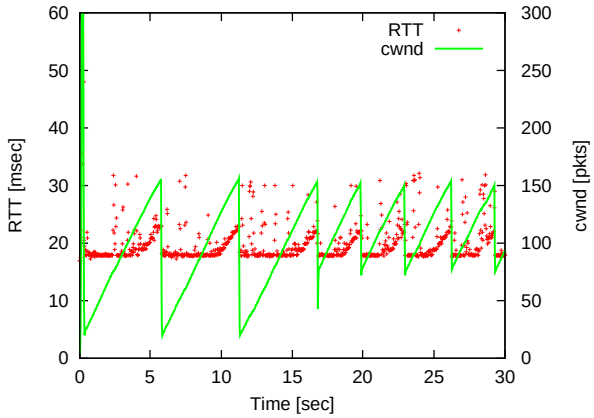

Figure 7: Typical change of the RTT and the congestion window size

five minutes in Jan. 21, 2005. Figure 6 shows the change in the average throughput when the number of simultaneous TCP connections is set to 1, 2, 5, 10, and 20. The figure shows that the upper-limit of the bottleneck link bandwidth in the experimental network can be estimated to be approximately 70 Mbps because the throughput increases only slightly when the number of connections is more than 5.

Figure 7 shows the changes of the round-trip times (RTTs) and the congestion window size (cwnd) of the TCP connection when only one TCP connection is utilized. From this figure, the following observations can be made regarding the experimental network:

- The change of the cwnd is synchronized with that of the RTT, meaning that the TCP connection itself causes the network congestion.
- The minimum RTT is approximately 18 msec.
- The buffer size at the bottleneck router is approximately 12 msec (equivalent to approximately 100 KBytes when the bandwidth is 70 Mbps), because the maximum RTT is approximately 30 msec.
- The buffer at the bottleneck router is equipped with the drop tail discipline because the RTT is stable when the packet loss occurs.
- In most cases of packet loss, only one packet is dropped because the cwnd is halved when packet loss occurs.

Note that these characteristics are equivalent to the typical results of simulation using ns-2 [11]. This means that we can obtain results for the performance of TCP proxy in the experimental network similar to those reported in the simulation studies in [4, 5, 7].

4 Evaluation Results

In this section, we evaluate the effectiveness of TCP proxy mechanism and High-Speed TCP based on extensive data transmission experiments.

4.1 Effect of TCP proxy

Figures 8 and 9 show the average throughput of data transfer with and without TCP proxies when the size of the TCP socket buffer at the sender and receiver hosts and the propagation delay between the hosts are varied. The average throughput is calculated from five two-minute data transmission experiments. Case 1 is that for data transfer without TCP proxies: the TCP connection between sender and receiver hosts is set. Case 2 is that for data transfer with TCP proxies: three split TCP connections with two TCP proxies are used, as shown in Figure 5. Note that the socket buffer size at the TCP proxies is set to be sufficiently large.

From the results for the Osaka-Tokyo connection, shown in Figure 8, when the TCP proxies are not used, sufficient throughput cannot be obtained with a small socket buffer at the sender/receiver hosts. Therefore, the socket buffer must be set to a large value in order to effectively utilize the bottleneck link bandwidth because the socket buffer can not utilize the link bandwidth with a 64-Kbyte socket buffer for the 150-Kbyte

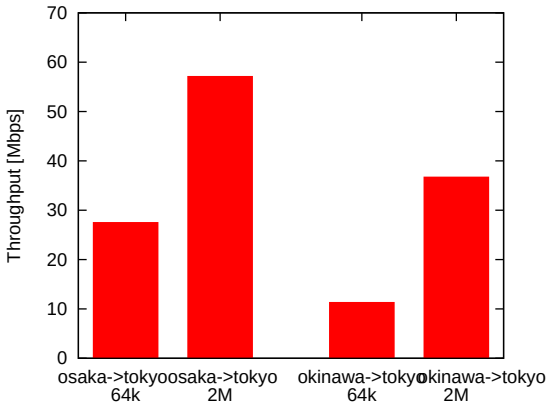

Figure 8: Average throughput without a TCP proxy

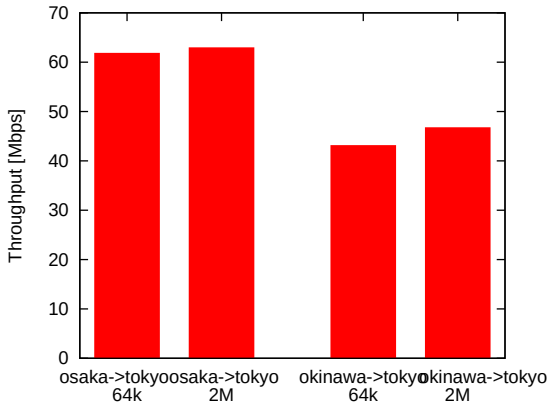

Figure 9: Average throughput with a TCP proxy

bandwidth-delay product of the experimental network. In addition, the results for the Okinawa-Tokyo case reveal that the link bandwidth cannot be used, even with the larger buffer, because TCP Reno cannot effectively utilize the large bandwidth-delay product [6]. A TCP Reno connection increases its window size only one packet per RTT, and reduces its window size by half when packets are lost.

On the other hand, for the Osaka-Tokyo case, when the TCP proxies are used (Figure 9), high throughput could be obtained, even with a small socket buffer. This is the case because the data transmission throughput of the split TCP connection between TCP proxies is sufficiently large since the TCP proxies utilize the large socket buffer and since high data transmission throughput between the sender/receiver host and the TCP proxy can be obtained with a small socket buffer as a result of the small bandwidth-delay products. This is one of the advantages of the TCP proxy mechanism, that is, it is not necessary to modify the settings of the sender and receiver terminals in order to obtain higher throughput.

On the other hand, based on the results obtained for the Okinawa-Tokyo case, similar results can be obtained with the small and large socket buffers. However, compared to the throughput for the Tokyo-Osaka case, the throughput for the Okinawa-Tokyo case is lower, even with the TCP proxies. This is the case because the TCP proxies are located near the sender/receiver and the propagation delay between the TCP proxies is approximately equal to that between the endhosts. The effectiveness of a TCP proxy depends on its location. In [5], the TCP proxy was reported to be most effective when located such that the delay between the endhosts is divided equally.

4.2 Effect of High-Speed TCP

To solve the above-mentioned problem, the use of TCP variants for high-speed and long-delay networks is one possible solution. Figure 10 shows the results of data transfer experiments for the Okinawa-Tokyo case using HSTCP/gHSTCP for the connection between TCP proxies. Based on this result, the throughput was found to be increased by using HSTCP. In addition, we found that a higher throughput could be obtained with gHSTCP, which uses the refined HSTCP algorithm proposed in [7]. This is because it takes less time to retransmit packets using gHSTCP because the number of dropped packets is decreased. However, we cannot obtain the maximum possible value of throughput (approximately 70 Mbps), even with gHSTCP because timeout often occurs with packet loss, since the propagation delay between the sender and receiver hosts is large.

Finally, we evaluate the fairness property for the case in which both connections are used. Figure 11 shows the results for the three protocols for TCP proxies for the Okinawa-Tokyo case. The figure shows that, higher throughput can be obtained by using HSTCP between TCP proxies, by decreasing the throughput of the normal TCP data transmission. This is one of the shortcomings of HSTCP, that is, although HSTCP can obtain high data transmission throughput in high-speed networks, it ignores the effect on the performance of co-existing connections.

On the other hand, when gHSTCP is used between the TCP proxies, we can obtain higher throughput and the throughput of TCP Reno is not decreased because gHSTCP controls its win-

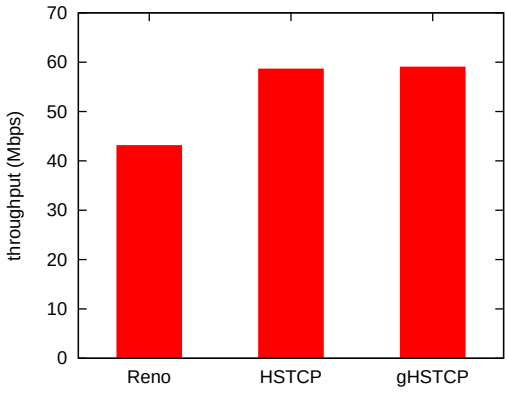

Figure 10: Effect of High-Speed TCP

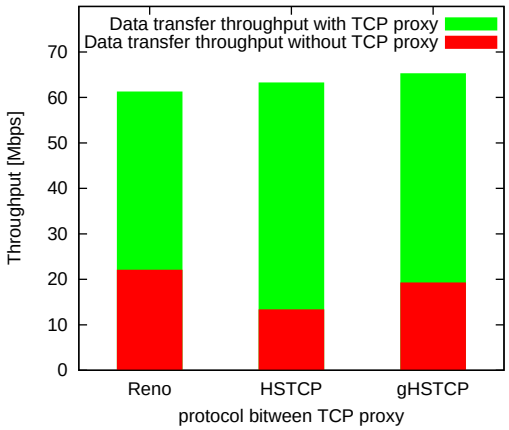

Figure 11: Evaluation of fairness property

dow size depending on the condition of congestion in the network. That is, gHSTCP achieves high throughput while not affecting the performance of co-existing connections.

5 Conclusions

In the present paper, we investigate the performance of the TCP proxy mechanism and HSTCP/gHSTCP through experiments using the actual public network between Tokyo and Osaka. The results of these experiments revealed that higher throughput can be obtained without changing the TCP protocol or the size of the socket buffer of the endhost. When HSTCP/gHSTCP is used on the TCP connection between the TCP proxy nodes, we showed that we can obtain high throughput in the environment where TCP Reno cannot obtain enough throughput such as the long-delay network. However HSTCP make co-existing connection's throughput decrease, and gHSTCP solves this problem.

In future studies, we would like to evaluate the TCP proxy mechanism in a larger experimental network using more than three networks and to set the parameters of gHSTCP. In addition, we would like to evaluate the performance of the TCP proxy mechanism in a high-speed network.

Acknowledgement

This work was partly supported by Strategic Information and Communications R&D Promotion Programme (SCOPE) from Ministry of Internal Affairs and Communications.

References

[1] J. Wroclawski, "The use of RSVP with IETF integrated services," *RFC 2210*, Sept. 1997.

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

[3] S.Williams, M. Abrams, C. R. Standridge, G. Abdulla, and E. A. Fox, "Removal policies in network caches for World-Wide Web documents," in *Proceedings of ACM SIGCOMM'96*, Aug. 1996.

[4] I. Maki, G. Hasegawa, M. Murata, and T. Murase, "Performance analysis and improvement of TCP proxy mechanism in TCP overlay networks," in *Proceedings of IEEE International Conference on Communications, Wireless Networking (ICC 2005)*, May 2005.

[5] I. Maki, G. Hasegawa, M. Murata, and T. Murase, "Throughput analysis of TCP proxy mechanism," in *Proceedings of ATNAC 2004*, Dec. 2004.

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

[7] Z. Zhang, G. Hasegawa, and M. Murata, "Analysis and improvement of HighSpeed TCP with TailDrop/RED routers," in *Proceedings of MASCOTS 2004*, Oct. 2004.

[8] T. Kelly, "Scalable TCP: Improving performance in highspeed wide area networks," in *Proceedings of PFLDnet'03: workshop for the purposes of discussion*, Feb. 2003.

[9] T. Iguchi, G. Hasegawa, and M. Murata, "A new congestion control mechanism of TCP with inline network measurement," in *Proceedings of ICOIN 2005*, Jan. 2005.

[10] NARANR, "NARANR/DAST: Iperf 1.7.0 - The TCP/UDP bandwidth measurement tool," available from `http://dast.nlanr.net/Projects/Iperf/`.

[11] T. V. Project, "UCB/LBNL/VINT network simulator - ns (version 2)," available from `http://www.isi.edu/nsnam/ns/`.