A Proposal of TCP Congestion Control Scheme Suited for Bandwidth Reservation Network

Article · June 2006

DOI: 10.1109/ICC.2006.254847

5 authors, including:

Nobuyoshi Komuro
Chiba University
84 PUBLICATIONS **353** CITATIONS

SEE PROFILE

A Proposal of TCP Congestion Control Scheme Suited for Bandwidth Reservation Network

Nobuyoshi Komuro[1], Kenta Osaki[1], Yukihiro Shinmura[2], Hiromi Ueda[1], and Toshinori Tsuboi[1]

[1]School of Computer Science, Tokyo University of Technology
1404-1 Katakura, Hachioji, Tokyo, 192-0982 JAPAN
Tel.: +81-426-37-2111
E-mail : nkmr@cs.teu.ac.jp

[2]Hitachi Communication Technologies, Ltd.
216 Totsuka-cho, Totsuka-ku, Yokohama, 244-8567, JAPAN

Abstract: The congestion control provided by Transmission Control Protocol (TCP) plays a key role in ensuring network stability and increased Quality of Service (QoS). However, in the face of the increasing number of high-speed networks such as Multi-protocol Label Switching with bandwidth reservation, conventional congestion control algorithms limit the network resource utilization efficiency since they overreact to packet loss. This paper proposes a new TCP congestion control scheme that improves the goodput by using reserved bandwidth to set the slow start threshold and the congestion window after a timeout or three duplicate acknowledgments. The advantage of this algorithm is that the TCP sender recovers faster after packet loss, especially when the connections have large round trip times; it avoids Reno's overreaction to loss.

1. Introduction

Since Multi-protocol Label Switching (MPLS) can make packet transmission (label switching) run faster, interest in it has been increasing of late [1]. Moreover, combining MPLS with functions such as traffic engineering, Virtual Private Network (VPN), and priority service is expected to yield high-performance and sophisticated switching networks. One of the motivations behind MPLS is to increase the Quality of Service (Qos). This is possible by adding the bandwidth reservation technique to MPLS. This technique allows multiple senders to transmit to multiple groups of receivers, permits individual receivers to switch channels freely, and optimizes bandwidth utilization rates. MPLS with bandwidth reservation decides a physical path based on the information about the bandwidth needed and the paths available. MPLS is able to ensure the minimum level of utilizable bandwidth.

Several congestion control schemes have been proposed but when too many packets are sent, performance degrades. When the number of packets dumped into the subnet by the hosts does not exceed the subnet's capacity, they are all delivered. However, as traffic increases, the routers are no longer able to cope and they begin dropping packets. At very high traffic loads, performance collapses completely and almost no packets are delivered. Thus, congestion control is a very important issue. Transmission Control Protocol (TCP) Tahoe, a type of congestion control scheme, was proposed by Van Jacobson at the end of the 1980s; TCP Reno is a variant of TCP Tahoe [2]. While congestion control provides stability to the Internet, current congestion control schemes do not support bandwidth reservation since they do not allow the bandwidth to be used appropriately.

This paper proposes a congestion control scheme that suits bandwidth reservation. It sets the congestion window size according to the reserved bandwidth.

2. Congestion control for TCP

When too many packets are present in the network, performance degrades. This situation is called congestion. When congestion occurs, there are two important issues: (a) to realize that two potential problems exist — network capacity and receiver's buffer size — and, (b) to deal with each of them separately. To do so, each sender maintains two windows:
the window the receiver has been granted and a second window, the congestion window. Each reflects the number of bytes the sender may transmit. The number of bytes that may be sent is the minimum of the two windows. Thus the effective window is the minimum of what the sender thinks is right and what the receiver thinks is right.

When a TCP connection is established, the sender initializes the congestion window to the size of the maximum segment. It then sends one maximum segment. As each of the segments is acknowledged, the congestion window is increased by an amount equal to one maximum segment. When the congestion window is n segments, if all n are acknowledged on time, the congestion window is increased by the byte count corresponding to the n segments. In effect, each burst acknowledged doubles the congestion window. The congestion window keeps growing exponentially until either a timeout occurs or a duplicate acknowledgments is received (Fig. 1 (a)). When the congestion window exceeds a certain threshold, the congestion window grows linearly (Fig. 1 (b)).

When a timeout occurs, the threshold is set to half of the current congestion window, and the congestion window is reset to one minimum segment.

In "Fast recovery mode", when duplicate acknowledgments are received, the threshold and the congestion window are set to half of the current congestion window (Fig. 1 (c)).

In the Reno algorithm, when packet loss is detected by duplicate acknowledgments, TCP Reno overreacts by halving its window size. If bandwidth reservation networks are truly efficient, the congestion window size should vary only slightly as shown in Fig. 2.

1-4244-0355-3/06/$20.00 (c) 2006 IEEE

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2006 proceedings.

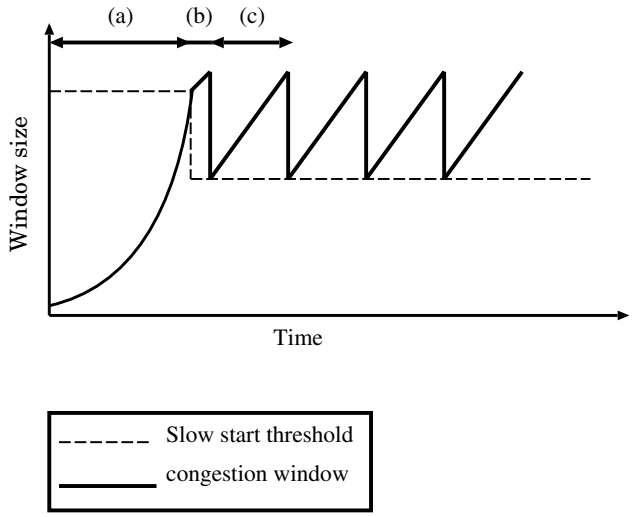

Figure 1. Window size for TCP Reno

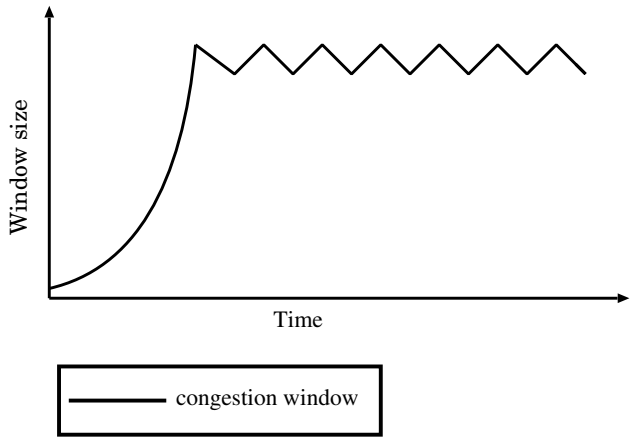

Figure 2. Window size that suits bandwidth reservation networks

3. A new TCP congestion control algorithm

We propose the idea of setting the congestion window and the slow start threshold by using the reserved bandwidth once congestion occurs. One goal of the proposed system is to minimize variation in the congestion window. We assume that r [bit/s] are reserved and, for round trip time RTT [s], the sender transmits $ssthresh$ packets; in this case the reserved bandwidth r is given by

$$r = \frac{ssthresh \times (size \times 8)}{RTT},$$

where $size$ is packet size [byte]. Thus the slow start threshold, $ssthresh$, is

$$ssthresh = \frac{r}{size \times 8} \times RTT.$$

The algorithm of the proposed scheme proceeds as follows;

1. When a TCP connection is established, the sender initializes the congestion window to the size of the maximum segment.
2. When packet loss is detected, the congestion window is set to the slow start threshold, and the algorithm moves into the congestion avoidance mode (the congestion window size is increased by one packet).
3. When the congestion window size exceeds the slow start threshold, the algorithm also moves into the congestion avoidance mode (the congestion window size is increased by one packet). If a timeout occurs, the congestion window is reset to one minimum segment.
4. The slow start threshold is set to a window size that suits bandwidth reservation, and how to decide the slow start threshold is outlined below.

In this algorithm, the slow start threshold lies in the neighborhood of the threshold that suits bandwidth reservation. The slow start threshold in this algorithm is set as follows;

1. After receipt of 3 duplicate acknowledgments:
   Set slow start threshold, $ssthresh$, as follows;

$$ssthresh = \frac{r}{size \times 8} \times RTT \quad (1)$$

   (Instead of $ssthresh = cwnd/2$ in Reno)
   We set $cwnd = ssthresh$.
2. After timeout:
   Set ssthresh as follows;

$$ssthresh = \frac{r}{size \times 8} \times RTT \quad (2)$$

   (Instead of $ssthresh = cwnd/2$ in Reno)
   We set $cwnd = 1$.

$ssthresh$ is the slow start threshold, $cwnd$ is congestion window, r is reserved bandwidth [bit/sec], $size$ is packet size [bit], and RTT [sec] is estimated round trip time.

When 3 duplicate acknowledgments are received, the slow start threshold is set equal to the available pipe size, and the congestion window is set equal to 1 packet (we should not increase it sharply during timeout to maintain the state). We call the proposed scheme Bandwidth Reserved TCP (BR-TCP).

4. Simulation

4.1 Simulation model

We modeled a bandwidth reservation network and established ftp communication via BR-TCP and New Reno. Figure 3 shows the model. The bandwidth between each router and each personal computer (PC) was 100 Mbps. The delay time between each router and each PC was 2 ms, and the time between routers was changed from 10 ms to 25 ms. PC1 sent packets to PC3 by using TCP, but PC2 sent packets to PC4 by using User Datagram Protocol (UDP). Each router set the bandwidth available to the PC client by using Class Based Queuing (CBQ) [4]. All simulation results presented here were gained from the NS2 network simulator [5].

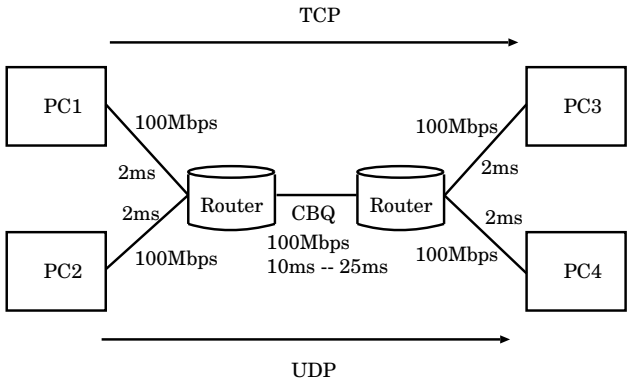

Figure 3. Simulation model

4.2 Simulation results

Figure 4 and 5 show the congestion window size and the slow start threshold of TCP New Reno and BR-TCP, respectively. In these figures, the reserved bandwidth is 40 Mbps, router buffer size is 100 packets, RTT is 58 ms, and the delay time between routers is 25 ms. The vertical axis represents the window size, and the horizontal axis represents elapsed time. The congestion window size depends on the slow start threshold. In BR-TCP, the slow start threshold is set to a window size that suits the bandwidth reservation network, so it is considered that the congestion window size of BR-TCP is more suited for such network than that of TCP New Reno.

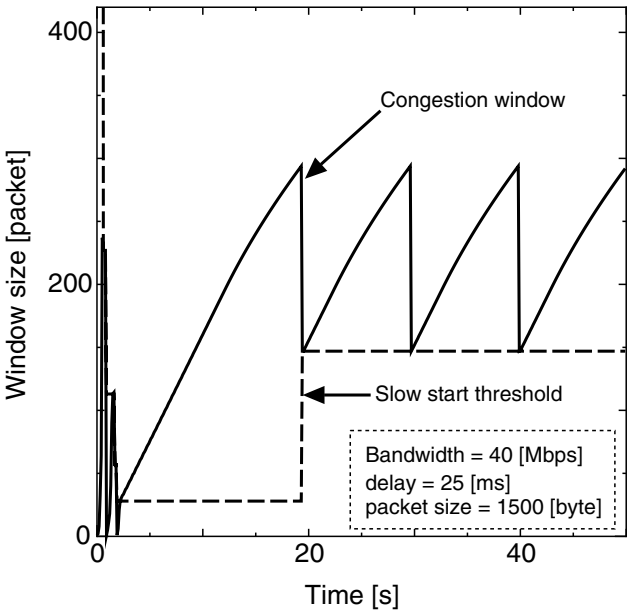

Figure 4. Congestion window size and slow start threshold of TCP New Reno; the reserved bandwidth is 40 Mbps, buffer size of routers is 100 packets, RTT is 58ms, and the delay time is 25 ms.

Figure 6 shows the congestion window size of BR-TCP and TCP New Reno. In this figure, the reserved bandwidth is 40 Mbps, router buffer size is 100 packets, RTT is 58 ms, and the delay time between routers is 25 ms. The vertical axis represents the congestion window size, and the horizontal axis represents elapsed time. We have found that the change in congestion window size is less with BR-TCP than with TCP New Reno. That is, the congestion window size of BR-TCP is more stable than that of TCP New Reno.

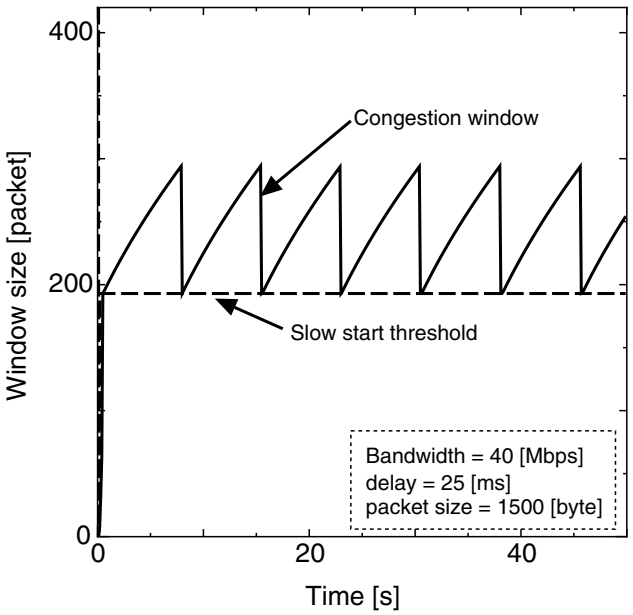

Figure 5. Congestion window size and slow start threshold of BR-TCP; the reserved bandwidth is 40 Mbps, buffer size of routers is 100 packets, RTT is 58 ms, and the delay time is 25 ms.

Figure 7 shows the RTT of BR-TCP. In this figure, the reserved bandwidth is 40 Mbps, router buffer size is 100 packets, and the delay time between routers is 25 ms. The vertical axis represents RTT, and the horizontal axis represents elapsed time. In BR-TCP, the slow start threshold depends on RTT. In this simulation, minimum RTT was adopted in Eqs. (1) and (2).

Figure 8 shows the throughput of BR-TCP and TCP New Reno. In this figure, the reserved bandwidth is 40 Mbps, router buffer size is 100 packets, RTT is 58 ms, and the delay time between routers is 25 ms. The vertical axis represents the throughput, and the horizontal axis represents elapsed time. We find that the throughput of BR-TCP is more stable than that of TCP New Reno.

5. Implementing BR-TCP in Linux

5.1 Testbed

We built a testbed of the bandwidth reservation network and established ftp communication using BR-TCP and New Reno. Figure 9 shows the MPLS testbed which is composed of Cisco 7200 Series routers. The client and the server were connected to the routers via 100 Mbps Ethernet links. Each router set the available bandwidth by using Class Based Weighted Fair Queuing (CBWFQ) [4]. In these trials, the packet size was 1500 bytes.

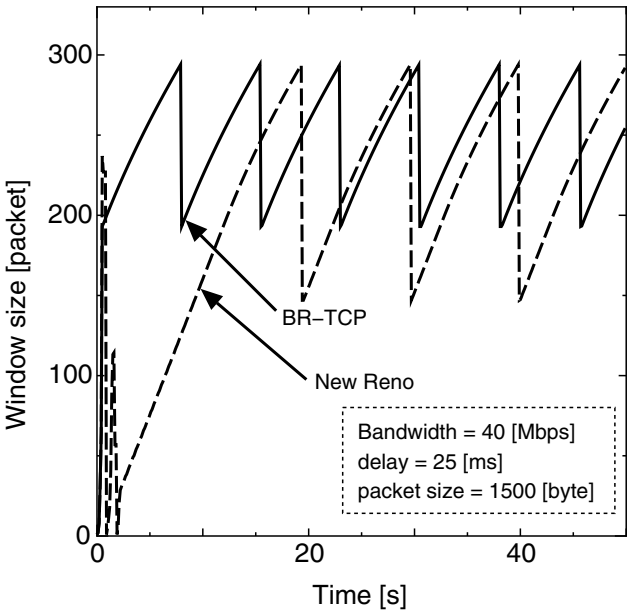

Figure 6. Congestion window versus elapsed time of BR-TCP and TCP New Reno; reserved bandwidth is 40 Mbps, buffer size of routers is 100 packets, delay time is 25 ms and RTT is 58 ms.

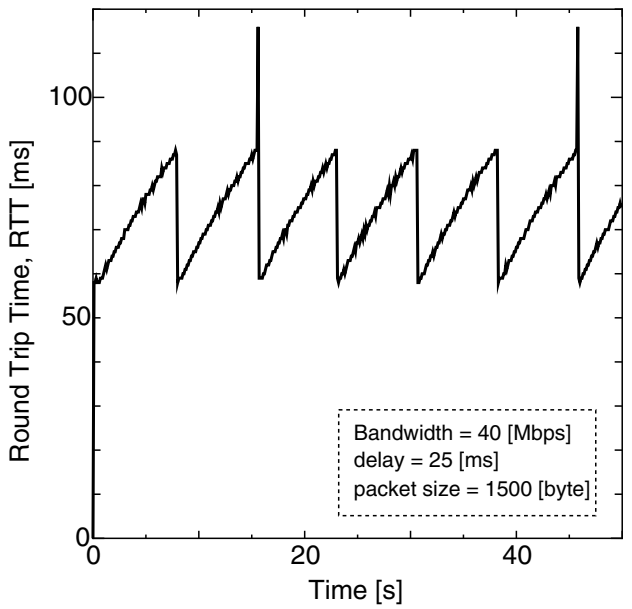

Figure 7. RTT of BR-TCP; reserved bandwidth is 40 Mbps, buffer size of routers is 100 packets and delay time is 25 ms.

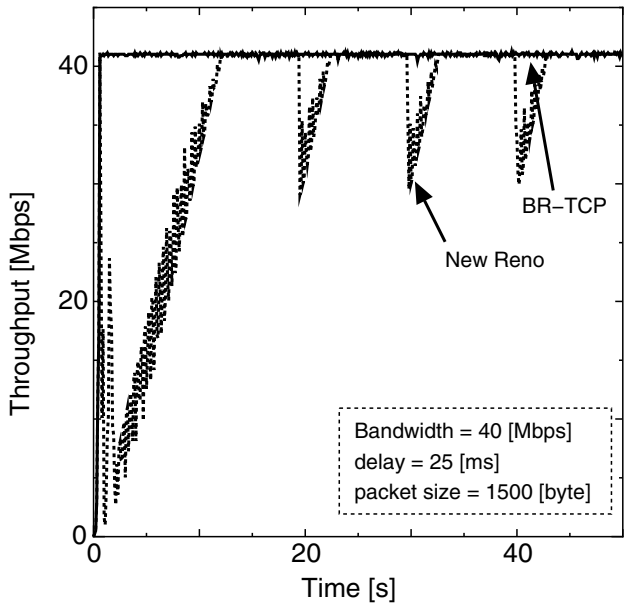

Figure 8. Throughput of BR-TCP and TCP New Reno; reserved bandwidth is 40 Mbps, buffer size of routers is 100 packets and RTT is 58 ms.

We implement BR-TCP on Fedora Core 2 Linux (Kernel 2.6.5-1). This kernel offers TCP New Reno, so we compared BR-TCP to TCP New Reno.

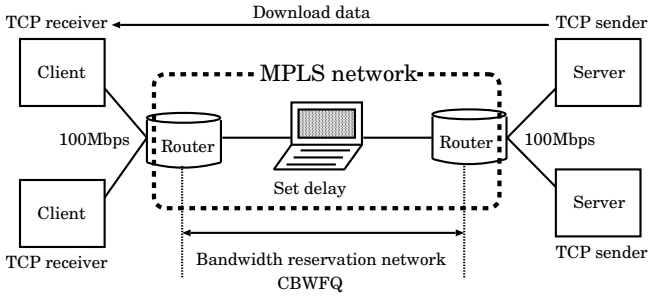

Figure 9. Model of experiment

5.2 Results

Figure 10 shows the transitions in the congestion windows of BR-TCP and TCP New Reno. In this figure, the reserved bandwidth is 40 Mbps, delay time is 25 ms, and Round Trip Time, RTT, is 58 ms. The vertical axis represents the congestion window size, and the horizontal axis represents elapsed time. With BR-TCP, changes in the congestion window size are small, so the window size is relatively stable.

Figure 11 shows the throughput of TCP New Reno; in this figure, the reserved bandwidth is 40 Mbps, delay time is 25 ms, RTT is 58 ms, and the packet size is 1500 bytes. Figure 12 shows the throughput of BR-TCP; in this figure, the reserved bandwidth is 40 Mbps, delay time is 25 ms, RTT is 58 ms, and the packet size is 1500 bytes. The maximum and average throughputs of TCP New Reno are about 4.3M byte/s, and

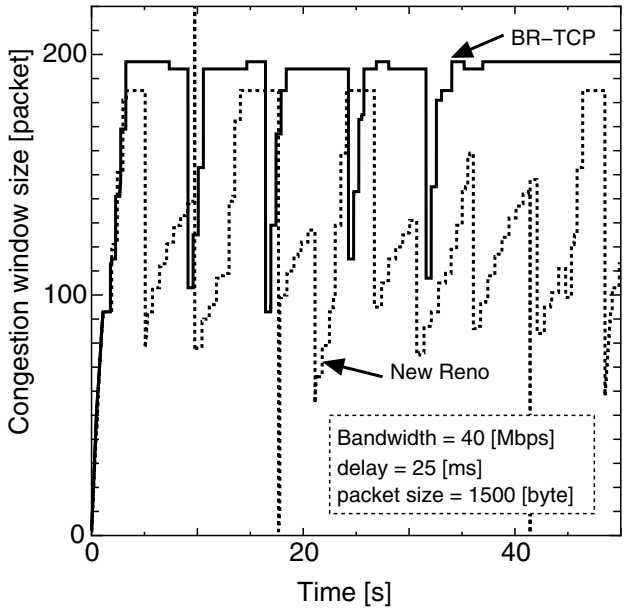

Figure 10. Comparison of the congestion window between BR-TCP and TCP New Reno; the reserved bandwidth is 40 Mbps and delay time is 25 ms.

about 2.5M byte/s, respectively. The corresponding BR-TCP values are about 4.5 M byte/s and 3.4 byte/s. BR-TCP offers higher throughput than TCP New Reno.

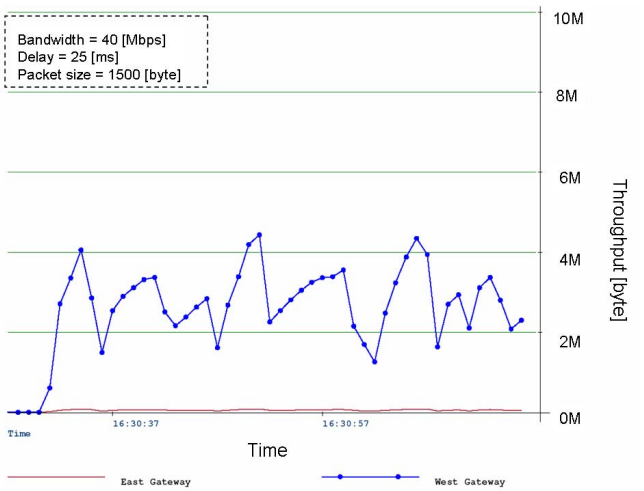

Figure 11. Throughput of TCP New Reno; the reserved bandwidth is 40 Mbps, delay time is 25 ms and RTT is 58 ms.

Figure 13 plots the throughputs of BR-TCP and New Reno versus RTT. In this figure, the reserved bandwidth is 40 Mbps and the packet size is 1500 bytes. We find that the throughput superiority of BR-TCP increases with RTT.

Figure 14 shows the throughput improvement of BR-TCP over TCP New Reno versus RTT at four reserved bandwidths: 10 Mbps, 20 Mbps, 30 Mbps, and 40 Mbps. We find that the superiority of BR-TCP increases with both RTT and reserved

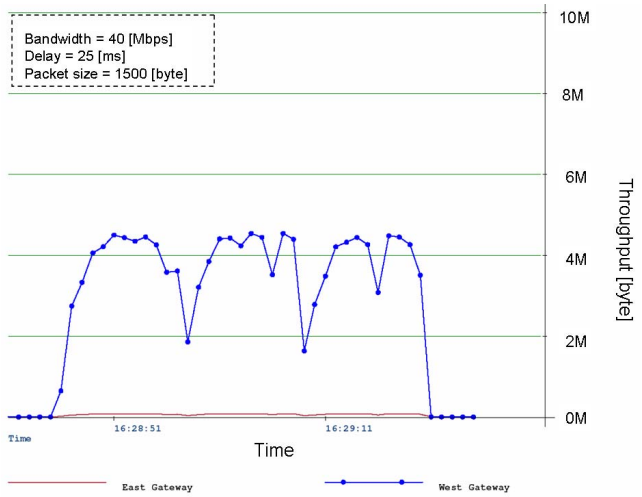

Figure 12. Throughput of BR-TCP; the reserved bandwidth is 40 Mbps, delay time is 25 ms and RTT is 58 ms.

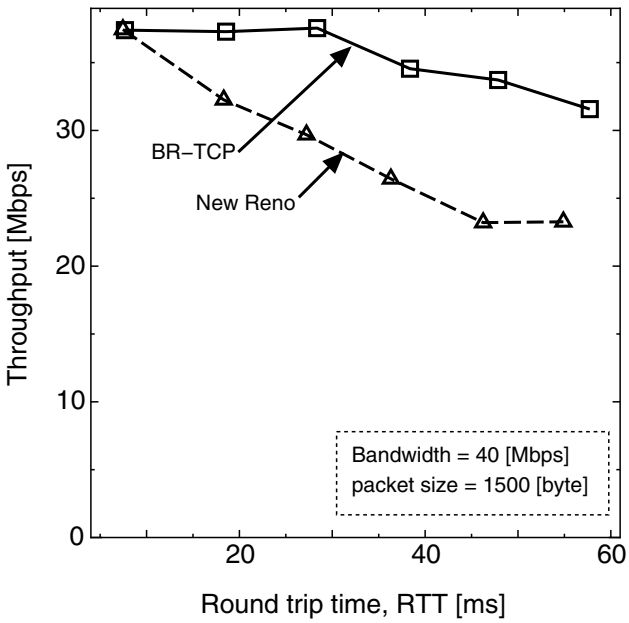

Figure 13. Throughput of BR-TCP and TCP New Reno versus RTT; reserved bandwidth is 40 Mbps and the packet size is 1500 bytes.

bandwidth. In New Reno, it takes a long time to settle the congestion window if the RTT or the reserved bandwidth is large, which degrades the throughput of TCP New Reno. BR-TCP, on the other hand, minimized the changes in congestion window size which maintains the throughput of BR-TCP. The testbed and simulation results show that BR-TCP is expected to yield more efficient communication. The presence of flow control is the reason for the drop in superiority of BR-TCP when the reserved bandwidth is 40 Mbps and $RTT > 45$. Flow control limits the window size of both schemes to the same size which decreases the difference between BR-TCP and TCP New Reno.

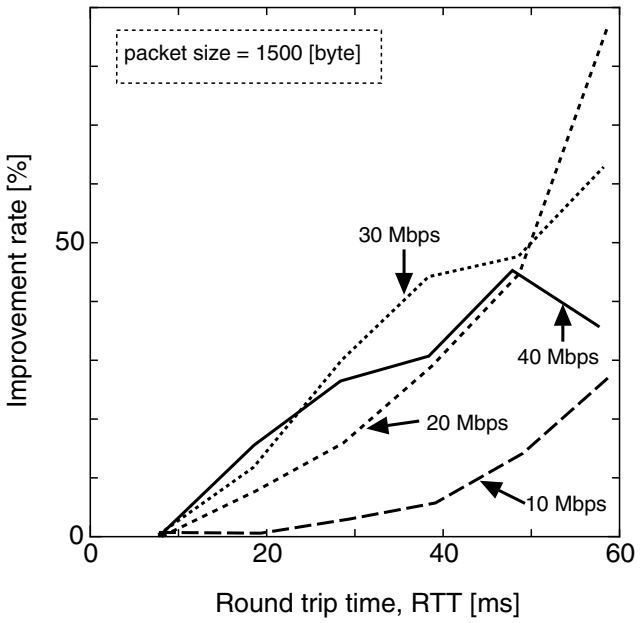

Figure 14. Improvement of the throughput versus RTT; reserved bandwidth is 10 Mbps, 20 Mbps, 30 Mbps, and 40 Mbps.

6. Conclusion

We proposed a new congestion control scheme to improve the performance of bandwidth reservation networks. The scheme sets the slow start threshold to run in the neighborhood of the threshold by using the reserved bandwidth. The proposed scheme was evaluated through simulations and testbed experiments. The following results were gained;

- The proposed scheme stabilizes the congestion window size.
- The proposed scheme has greater throughput than TCP New Reno.

It is considered that the proposed scheme well suits TCP networks that use the bandwidth reservation technique.

The slow start threshold depends on RTT, so we need to discuss what RTT values are recommended.

References

[1] Vivek Alwayn, "Advanced MPLS Design and Implementation", Cisco Press, 2002.

[2] J. Jacobson, "Congestion Avoidance and Control", Proc. ACM SIGCOMM 88, August 1988.

[3] C. Casetti, M. Gerla, S. S. Lee, S. Mascolo and Sanadidi, "TCP with Faster Recoverry", Proc. of MILCOM 2000, October 2000.

[4] S. Vegesna, "IP Quality of Service", Cisco Press, 2001.

[5] K. Fall and K. Varadhan, "The ns Manual (formerly ns Notes and Documentation)", URL: *http://wwwisi.edu/nsnam/ns/doc/index.html*