

IMPROVING THE CONVERGENCE TIME OF HIGHSPEED TCP

Masayoshi Nabeshima, Kouji Yata
NTT Cyber Solutions Laboratories, NTT Corporation
Yokosuka-shi, 239-0847 Japan
Email: {nabeshima.masayoshi, yata.kouji}@lab.ntt.co.jp

Abstract - It is reported that TCP does not perform well in high-speed and long-distance networks. This problem led to the proposal of HighSpeed TCP (HS-TCP). HS-TCP is designed to achieve a steady-state throughput of 10 Gbps with a packet loss rate of 10^{-7} when the packet size is 1500 bytes and the round trip time is 100 ms. However, HS-TCP has a problem in terms of convergence times. That is, it takes a long time for a new HS-TCP flow to achieve fair bandwidth allocation if the existing HS-TCP flows have large congestion windows. This paper proposes a new mechanism to improve the convergence time of HS-TCP. The basic idea of our mechanism is that if a flow has larger window than is fair its window is decreased more aggressively than usual. Simulations show that our mechanism significantly improves the convergence time of HS-TCP.

1. INTRODUCTION

The transmission control protocol (TCP) is the most commonly used reliable transport protocol. The current stability of the Internet depends on the end-to-end congestion control of TCP. The demand for high bandwidth applications such as high speed bulk data transfer, telemedicine, and computational grids is now expanding. TCP is expected to provide an effective mechanism for such applications. However, it is reported that TCP does not perform well in high-speed wide area networks. To achieve a steady-state throughput of 7.2 Gbps with 1500 byte packets and a 100 ms round trip time (RTT), for example, the packet loss rate must be less than 4.17×10^{-10} . This is beyond the limits of achievable fiber error rates. In addition, TCP requires 40,000 RTTs, or almost 70 minutes, to recover from a single packet loss. This means that TCP cannot fully utilize the available bandwidth in high-speed and long-distance networks.

This problem led to the proposal of HighSpeed TCP (HS-TCP) [1]. HS-TCP modifies the TCP response function in environments with large congestion windows. Compared to standard TCP, HS-TCP increases the congestion window more aggressively upon receiving ACK packets. It decreases the congestion window more gently upon detecting packet losses. The author in [2] investigated the performance of HS-TCP in high bandwidth delay product networks by computer simulations. The results show that HS-TCP can efficiently utilize the available bandwidth without requiring unrealistically low packet loss rates. However, HS-TCP has a problem in terms of convergence times. That is, when a new HS-TCP flow starts up in an environment where the

congestion windows of existing HS-TCP flows are large, it takes a long time for the new HS-TCP flow to achieve fair bandwidth allocation. The main reason is that the existing flows decrease their congestion windows slowly even though their windows are larger than is fair.

This paper proposes a new mechanism to improve the convergence time of HS-TCP. In our mechanism, the congestion window size just before a loss event is checked. If the size continuously declines, the window is assessed to be on a downward trend. Once the difference between the maximum and minimum values of the checked size during the current downward trend exceeds a threshold, the congestion window decrease parameter is set larger than usual. The logic of our mechanism is that flows that fulfill the condition described above are likely to have a larger window size than fair. Thus, it is reasonable to decrease their window more aggressively than usual to improve the convergence times.

The remainder of this paper is as follows. Section 2 overviews HS-TCP. Section 3 elaborates our proposed mechanism. Section 4 uses computer simulations to evaluate the performance of HS-TCP using our mechanism. Finally, conclusions are provided in Section 5.

2. HIGHSPEED TCP

Before we describe HS-TCP, we present the congestion avoidance algorithm of standard TCP and describe its main problem.

The congestion avoidance algorithm of standard TCP has the form of additive increase and multiplicative decrease (AIMD). Senders use the following algorithm to update their congestion windows ($cwnd$) in response to the acknowledgement of received packets and the detection of packet losses.

$$\text{ACK} : cwnd \leftarrow cwnd + \frac{a}{cwnd}$$

$$\text{LOSS} : cwnd \leftarrow cwnd - b \times cwnd$$

$cwnd$, a , and b are all defined in units of packets. The congestion window increase parameter a is 1, and the congestion window decrease parameter b is 0.5. That is, standard TCP increases $cwnd$ by roughly one packet per RTT and halves $cwnd$ when packet loss is detected.

In steady-state, the average congestion window w is given by:

$$w_{std} = \frac{\sqrt{1.5}}{\sqrt{p}}$$

,where p is the packet loss ratio [3].

Assume that the packet size is 1500 bytes and the round trip time is 100 ms. For a standard TCP flow to maintain a sending rate of 7.2 Gbps, standard TCP would require

$$\frac{1500 \times 8}{0.1} \times \frac{\sqrt{1.5}}{\sqrt{p}} = 7.2 \times 10^9,$$

or $p \approx 4.17 \times 10^{-10}$. This packet loss ratio is beyond the limits of achievable fiber error rates.

The congestion avoidance algorithm of HS-TCP is specified using three parameters: W_{low} , W_{high} , and P_{high} . If the current congestion window is less than W_{low} , HS-TCP uses the same response function as standard TCP. Otherwise, it uses a new response function. W_{high} and P_{high} are used to specify the upper bound of the HS-TCP response function. The default values of W_{low} , W_{high} , and P_{high} are 38, 83000, and 10^{-7} , respectively.

The HSTCP congestion control can be expressed using the following equations.

$$\text{ACK: } cwnd \leftarrow cwnd + \frac{a(cwnd)}{cwnd}$$

$$\text{LOSS: } cwnd \leftarrow cwnd - b(cwnd) \times cwnd$$

, where

$$a(cwnd) = \frac{2 \times cwnd^2 \times b(cwnd) \times p(cwnd)}{2 - b(cwnd)}$$

$$b(cwnd) = \frac{\log(cwnd) - \log(W_{low})}{\log(W_{high}) - \log(W_{low})} (b_{high} - 0.5) + 0.5$$

$$p(cwnd) = \exp \left[\frac{\log(cwnd) - \log(W_{low})}{\log(W_{high}) - \log(W_{low})} \times \{\log(P_{high}) - \log(P_{low})\} + \log(P_{low}) \right]$$

When $cwnd$ is more than W_{low} , the value of a is more than 1 and the value of b is less than 0.5. This means that HS-TCP increases $cwnd$ more aggressively and decreases $cwnd$ more gently than standard TCP when $cwnd$ is more than W_{low} .

It is desirable for high-speed transport protocols to have the following characteristics.

- **Scalability:** high-speed protocols can scale up their throughput without requiring unrealistically low packet loss rates.
- **TCP friendliness:** at high loss rates where standard TCP performs well, high-speed protocols must offer TCP-compatible performance.
- **Responsiveness:** high speed protocols respond quickly to changes in the available bandwidth.

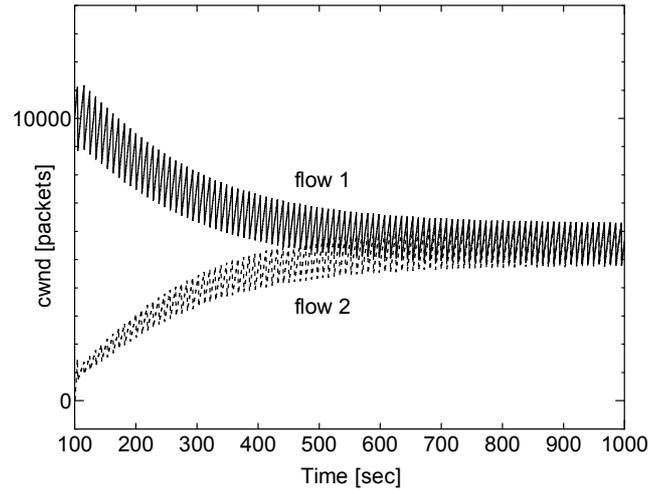


Figure 1. Congestion window size versus time for the original HS-TCP.

For scalability, HS-TCP is designed to achieve the steady-state throughput of 10 Gbps with the packet loss rate of 10^{-7} when the packet size is 1500 bytes and the RTT is 100 ms. For TCP friendliness, HS-TCP is designed to use the same response function as standard TCP when the packet loss ratio exceeds 10^{-3} . In terms of responsiveness, however, HS-TCP has poor performance. We show a simulation result to confirm the poor responsiveness of HS-TCP. The details of the simulation configurations and parameters used are described in Section 4. We assumed that there were only two HS-TCP flows. The start time of flows 1 and 2 were 0 and 100 secs, respectively. Fig. 1 shows the congestion window size of the two flows. We can see that it takes a long time for flow 2 to achieve fair bandwidth allocation. To the best of our knowledge, a mechanism for improving the convergence time of HS-TCP has not been addressed in the literature.

3. PROPOSED MECHANISM

From Fig. 1, the main reason why HS-TCP has the poor responsiveness is that flow 1, which has a larger window size than fair, decreases its congestion window slowly. Note that the window size of flow 1 shows a downward trend. To improve the convergence time of HS-TCP, therefore, the window size should be decreased more aggressively than usual when a continuous fall in window size is noted as we take this as evidence of one or more new flows.

We enhance the algorithm for determining the value of the congestion window decrease parameter in HS-TCP. We use three new variables: W_{max} , W_{prev} , and $numDec$. W_{max} represents the maximum window size during the current downward trend. W_{prev} represents the window size just before the previous loss event. $numDec$ represents the number of times that the window has been decreased during the current downward trend (Fig. 2).

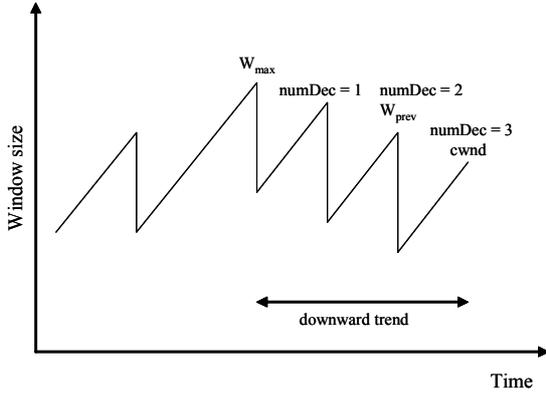


Figure 2. Illustration of W_{max} , W_{prev} , and $numDec$.

The details of the algorithm are as follows. When packet loss is detected, if the current congestion window size $cwnd$ is less than W_{low} , the congestion window decrease parameter b is set to 0.5. Otherwise, we check whether W_{prev} is greater than $cwnd$ or not.

If W_{prev} is less than or equal to $cwnd$, we can say that the window is not on a downward trend. Thus, b is determined using the equation defined in Section 2. W_{max} and W_{prev} are set to $cwnd$. $numDec$ is set to 0.

If W_{prev} is greater than $cwnd$, we can say that the window is on a downward trend. Thus, $numDec$ is increased by 1. Next, we check whether the following two conditions are fulfilled.

1. $numDec$ is more than or equal to $N1$
2. $(W_{max} - cwnd)$ is more than or equal to S packets

If these are fulfilled, we determine that the window should be decreased more aggressively than usual. Thus, b is set to 0.5. That is, the window is halved as in standard TCP. W_{max} , W_{prev} , and $numDec$ are set to 0. If these conditions are not fulfilled, b is determined using the equation defined in Section 2. W_{prev} is set to $cwnd$. In addition, if $numDec$ equals $N2$, we judge that the value of W_{max} is too old. Thus, W_{max} and $numDec$ are updated to $cwnd$ and 0, respectively.

For completeness, we give the pseudo-code of the proposed mechanism in Fig. 3.

4. SIMULATED PERFORMANCE

We simulated the performance of our mechanism to confirm that it can improve the convergence time of HS-TCP. All simulations were performed using ns-2 [4]. We incorporated our mechanism into the code of HS-TCP taken from the ns-2 distribution.

4.1 Simulation environment

Unless stated otherwise, the following parameter values were used as default. We used the single congested link topology shown in Fig. 4, where S_i was sending packets to D_i . All traffic passed through the bottleneck link. The bottleneck link

```

if (cwnd > W_low) {
    if (W_prev > cwnd) {
        numDec++;
        if ((numDec ≥ N1) && (W_max - cwnd ≥ S)) {
            b = 0.5;
            W_max = W_prev = numDec = 0;
        } else {
            b = HS_b;
            W_prev = cwnd;
            if (numDec == N2) {
                W_max = cwnd;
                numDec = 0;
            }
        }
    } else { /* W_prev ≤ cwnd */
        b = HS_b;
        W_max = W_prev = cwnd;
        numDec = 0;
    }
} else { /* cwnd ≤ W_low */
    b = 0.5;
}

```

HS_b: congestion window decrease parameter of the original HS-TCP

Figure 3. Pseudo-code of our mechanism.

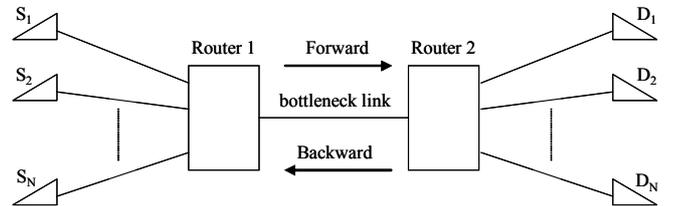


Figure 4. Single congested link topology.

bandwidth was 1 Gbps, the link delay was 50 ms. Each router implemented a drop-tail buffer whose size was 50% of the bandwidth delay product of the bottleneck link.

Each flow implemented the SACK option [5], the timestamp option [6], and the limited slow start algorithm [7]. Their maximum window size was large enough so as not to impose any limitations. The packet size was 1500 bytes. We paced TCP packets so that no more than three packets were sent in burst. For HS-TCP, we used the default values of the simulator. For our mechanism, $N1$, $N2$, and S were 2, 10, and $W_{max}/32$, respectively. The maximum and minimum values of S were limited to 200 and 50, respectively.

As background traffic, a set of web traffic flows and a set of standard TCP flows whose maximum window size was limited to 8 packets were generated in both directions.

When the total number of flows was N , we assumed that flows 1 to $(N-1)$ started sending packets randomly. Specifically, their start time was randomized in the range 0 to 10 secs in each simulation run. Flow N was assumed to start sending packets at 100 secs. We measured the throughput of flow N every 5 secs. We then calculated the time from 100 secs until the measured throughput of flow N exceeded the fair one. The time was used as the convergence time of flow N .

We repeated each simulation 15 times and the average values of these simulations are reported below.

4.2 Performance without background traffic

In this section, we evaluated the convergence times in an environment with no background traffic. Fig. 5 shows the congestion window size for two flows. The start times of flows 1 and 2 were 0 and 100 secs, respectively. We can see that the window size of flow 1 is decreased more aggressively than usual at 134, 172, and 225 secs. From Figs. 1 and 5, we can say that our mechanism significantly improves the convergence time of HS-TCP.

Fig. 6 shows the convergence time of flow N when the total number of flows was varied from 2 to 10. We can see that the convergence time of HS-TCP using our mechanism is shorter than that of the original HS-TCP regardless of the total number of flows. When the original HS-TCP is used, decreasing the total number of flows increases the convergence time. The reason is that the fair share rate increases as the total number of flows is decreased. Thus, it takes a longer time for the measured throughput of flow N to exceed the fair share rate as the total number of flows is decreased. With our mechanism, on the other hand, the convergence time is the shortest when the total number of flows is 2. The reason is that when the total number is 2 the two conditions described in Section 3 are fulfilled with higher probability than when the total number is more than 2.

Fig. 7 shows the convergence time of flow N when the total number of flows was 2, and the bottleneck link bandwidth was varied from 100 Mbps to 1 Gbps. We can see that the convergence time of HS-TCP using our mechanism is shorter than that of the original HS-TCP regardless of the bandwidth.

4.3 Performance with background traffic

We also evaluated the convergence times in an environment with background traffic. Fig. 8 shows the convergence time of flow N when the total number of flows was varied from 2 to 10. We can see that the convergence time of HS-TCP using our mechanism is shorter than that of the original HS-TCP regardless of the total number of flows. From Figs. 6 and 8, we can see that the convergence time is shorter with background traffic than without it. This is because the

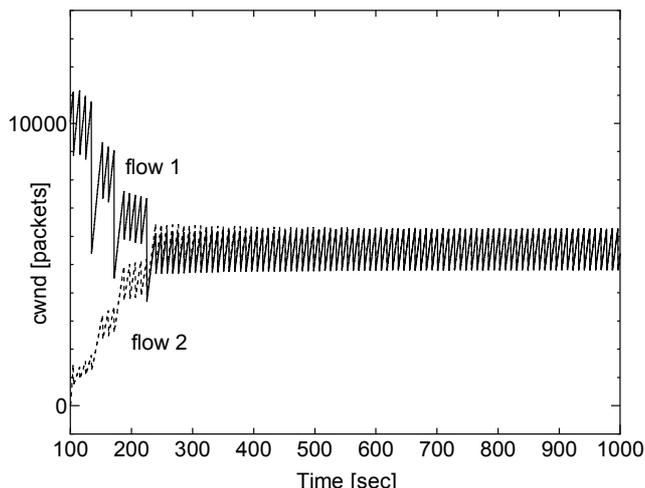


Figure 5. Congestion window size versus time for the HS-TCP with our mechanism.

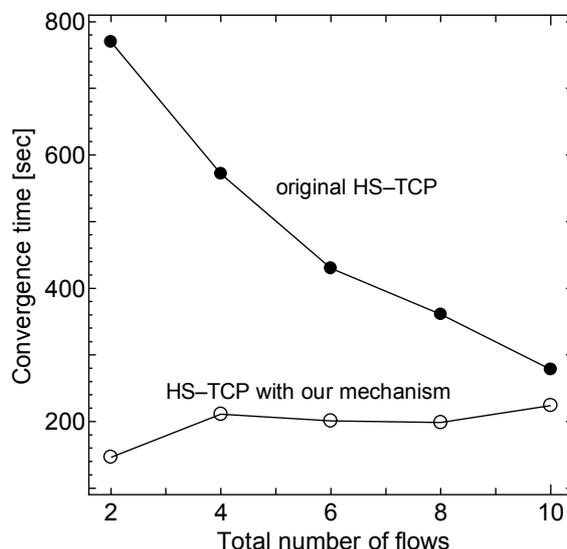


Figure 6. Convergence time versus total number of flows without background traffic.

background traffic reduces global synchronization. With global synchronization, flows 1 to N experience packet losses simultaneously. Thus, flow N requires a longer time to achieve the fair rate. When global synchronization is reduced, on the other hand, flows 1 to N do not always experience packet losses simultaneously. In addition, the probability that at least one packet is lost from a single window increases exponentially as the window size increases. Thus, flow N requires a shorter time to achieve the fair rate.

Fig. 9 shows the convergence time of flow N when the total number of flows was 2, and the bottleneck link bandwidth was varied from 100 Mbps to 1 Gbps. We can see that the

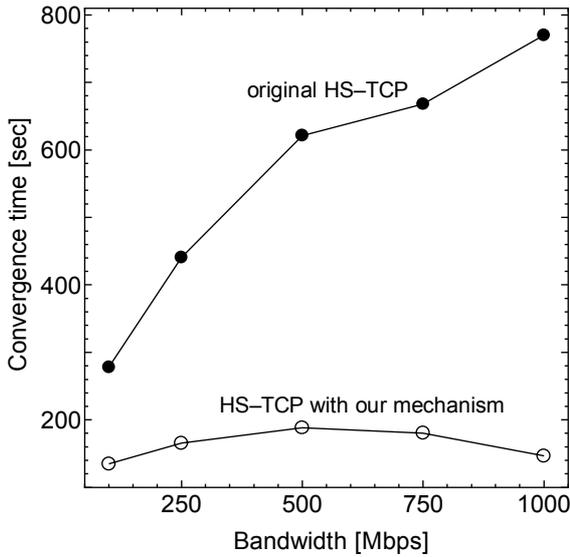


Figure 7. Convergence time versus bandwidth without background traffic.

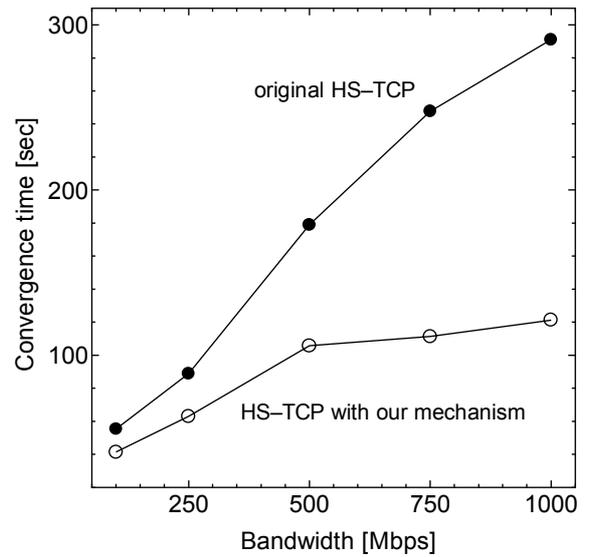


Figure 9. Convergence time versus bandwidth with background traffic.

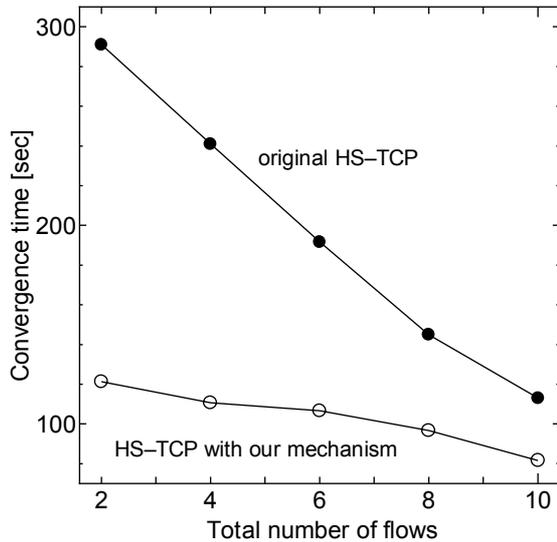


Figure 8. Convergence time versus total number of flows with background traffic.

convergence time of HS-TCP using our mechanism is shorter than that of the original HS-TCP regardless of the bandwidth.

5. CONCLUSIONS

This paper has proposed a mechanism for improving the convergence time of HS-TCP. We used three new variables: W_{max} , W_{prev} , and $numDec$. W_{max} is the maximum window size during the current downward trend. W_{prev} represents the window size just before the previous loss event. $numDec$ is the number of times the window has been decreased during

the current downward trend. If ($W_{prev} > cwnd$) we increase $numDec$ by 1. If ($numDec \geq N1$) and ($W_{max} \geq S$) then the window is decreased more aggressively than usual. Flows are likely to have larger window than is fair if the condition described above is fulfilled. Thus, decreasing their window more aggressively results in improving the convergence time of HS-TCP.

Computer simulations were used to evaluate the performance of HS-TCP with our mechanism. The results clarified that our mechanism improves the convergence time of HS-TCP with and without background traffic.

As a future research topic, we plan to estimate the optimum values of $N1$, $N2$, and S .

REFERENCES

- [1] S. Floyd, "HighSpeed TCP for large congestion windows," *RFC 3649*, Dec. 2003.
- [2] E. Souza, "A simulation-based study of HighSpeed TCP and its deployment," *LBNL technical report LBNL-52549*, April 2003.
- [3] M. Hassan and R. Jain, *High performance TCP/IP networking: Concepts, issues, and solutions*, New Jersey, Pearson, 2004.
- [4] S. McCanne and S. Floyd, NS simulator, available from <http://www.isi.edu/nsnam/ns>.
- [5] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP selective acknowledgement options," *RFC 2018*, Oct. 1996.
- [6] V. Jacobson, R. Braden, and B. Borman, "TCP extensions for high performance," *RFC 1185*, May 1992.
- [7] S. Floyd, "Limited slow-start for TCP with large congestion windows," *RFC 3742*, March 2004.