Enhanced Congestion Control Algorithm for High-Speed TCP

Young-Soo Choi, Sung-Hyup Lee, and You-Ze Cho

School of Electrical Engineering and Computer Science
Kyungpook National University, Korea
{yschoi,tenetshlee,yzcho} @ee.knu.ac.kr

Abstract. Current TCP congestion control can be inefficient and unstable in high-speed wide area networks due to its slow response with a large congestion window. Several congestion control proposals have already been suggested to solve these problems. In this paper, we propose a new variant of TCP for a high-speed network, which combines delay-based congestion control with loss-based congestion control. Our simulation results show that the proposed scheme performs better than the existing high-speed TCP protocols in terms of fairness, stability, and scalability, while providing TCP friendliness at the same time.

1 Introduction

The demand for high-speed applications such as bulk-data transfer, storage area network, and grid networking has increased. It, however, has been reported that as the bandwidth-delay product continues to grow, TCP underutilizes the bandwidth and it will eventually become a performance bottleneck itself [1]. Recently, various schemes have been designed. Such schemes include HSTCP [1], STCP [2], and BIC [3] and two properties have been considered: TCP friendliness and scalability. This is to ensure that a protocol does not take away too much bandwidth from TCP, while utilizing a bandwidth of high speed networks efficiently.

In this paper, we propose a new variant of HSTCP, called eHSTCP (enhanced HSTCP), which is a hybrid scheme of loss-based congestion control and delay-based congestion control. First, we develop mechanism which avoids the effect of backward path congestion. Second, eHSTCP refines the Additive Increase Multiplicative Decrease (AIMD) mechanism of HSTCP to enhance scalability, TCP friendliness, stability, and fairness.

The remainder of this paper is organized as follows: Section II introduces the eHSTCP protocol. Section III presents the simulation results, and finally conclusions are given in Section IV.

2 eHSTCP Protocol

If network congestion occurs in the backward path, delay-based congestion control protocols may overestimate RTT and unnecessarily decrease the congestion window. By using the TCP timestamp option, our mechanism obtains samples of queueing delay on the forward and backward paths separately. Note that the sender and receiver clocks do not have to be synchronized since we are only interested in the relative time difference. We define the effective RTT (eRTT) as

$$eRTT = RTT - d_{b,q} \tag{1}$$

$$d_{b,q} = d_b - min(d_b) \tag{2}$$

where RTT is a newly measured round-trip time, $d_{b,q}$ is the backward queueing delay, d_b is a measured backward delay and $min(d_b)$ is the minimum of all measured backward delays. Consequently, the $eRTT$ indicates a round trip time when there is no backward path congestion.

Because previous research has shown that the HSTCP provides acceptable performances, we adopt the HSTCP as our baseline congestion control algorithm throughout this paper. In order to improve scalability, fairness, and stability, the proposed scheme uses the RTT which has up-to-date information about congestion levels and the HSTCP's AIMD mechanism is modified as follows.

Since random noise in RTT measurements cannot be avoided in practice, we use the RTT as a binary feedback signal in additive increase mechanism. To prevent throughput degradation from the reverse cross-traffic, we define N' as follows:

$$N' = (Expected - \frac{cwnd}{eRTT}) \times RTT_{min} = cwnd \times d_{f,q}/eRTT \tag{3}$$

where $Expected$ is the current congestion window size divided by RTT_{min} (the minimum of all measured RTTs) and $d_{f,q}$ is the forward queue delay. Consequently, according to the Little's Law, N' indicates the measured backlog when there is no backward queueing delay.

If the measured backlog (N') is lower than threshold (N^*), we assume that the network is underutilized and the HSTCP's congestion control algorithm is used. When the network is fully utilized, eHSTCP behaves like TCP Reno. eHSTCP stays at the fully utilized region longer, because eHSTCP does not increase its window size as quickly as HSTCP does when the critical region is reached. This mechanism not only reduces packet loss, but also improves stability by avoiding unnecessary decrease of the congestion window. Additionally, this mechanism leaves a buffer space for other traffic and thus makes eHSTCP TCP friendly. From the equation (3), since each source behaves like TCP Reno at the same throughput for a given queueing delay, the proposed mechanism can significantly correct the RTT fairness problem as compared with other protocols.

Setting (1-β) as RTT_{min} divided by RTT_{max} ensures that the buffer is empty while preventing buffer underflow (for more detail, see [5]). To prevent link underutilization and exclude the effect of backward pach congestion, eHSTCP uses:

$$1 - \beta_{eHSTCP} = \frac{RTT_{min}}{eRTT} \tag{4}$$

Note that we use $eRTT$ instead of RTT_{max}. By inspecting the raw data from our simulation results, we found that the measured RTTs are frequently smaller

Table 1. Comparison of utilization, fairness, packet loss ratio, and stability under 2.5Gbps bottleneck link

	HSTCP	STCP	BIC	eHSTCP
Link utilization	0.92	0.99	0.95	0.99
Packet loss ratio(%)	0.0197	0.1281	0.0206	0.0065
Normalized standard deviation	0.148	0.149	0.107	0.047

than the maximum RTT when a packet loss occurs. The main reason behind this phenomenon is TCP burstiness [4].

To provide TCP friendliness which is comparable to that of HSTCP, we employ the following compensation algorithm. After a packet loss, if β_{eHSTCP} is smaller than β_{HSTCP}, eHSTCP reduces its congestion window using β_{eHSTCP} and it enters a safety check phase. At the same time, w_{desg} is calculated using β_{HSTCP}. During this safety check phase, eHSTCP does not increase its congestion window but monitor the backlog. If N' exceeds N^* in the safety check phase, eHSTCP assumes that β_{eHSTCP} is too aggressive and reduces its congestion window to w_{desg}. Therefore, it takes one RTT time for eHSTCP to decrease its window size to the size of HSTCP. Otherwise, after the safety check phase, eHSTCP enters the additive increase phase.

3 Simulation Results and Discussion

We use ns simulator and the topology used for the simulation is dumbbell network. For background traffic, web traffic, 25 small TCP flows with a limited congestion window size under 64, and 4 long lived TCP flows are created in both directions. We use N^*=10 and the safety check phase $=5\times RTT_{max}$.

In order to evaluate bandwidth scalability, we measure utilization and the average packet loss rate of the bottleneck link. So as to evaluate stability for high-speed TCP, we use the sample standard deviation normalized by the average throughput of high-speed flows. Table 1 shows that link utilization of eHSTCP is relatively comparable to that of STCP. Also, eHSTCP shows a good performance among all protocols under the packet loss rate evaluation criterion. Additionally, we found that eHSTCP showed the best stability.

In this experiment, two high speed flows with a different RTT are used. The RTT of flow 1 is 40ms, while we vary the RTT of flow 2 between 120ms and 240ms. Table 2 depicts that eHSTCP outperforms other high-speed protocols in terms of RTT fairness.

Fig. 1 shows the percentage of the bandwidth shared by each flow type with different bottleneck bandwidth. For 20Mbps, all high-speed TCP protocols show similar TCP friendliness. As the bandwidth increases, the share of the bandwidth taken by the background traffic is substantially reduced due to the TCP scalability problem. Note that, under 2.5Gbps, eHSTCP flows achieve slightly higher throughput than HSTCP flows. The increase in eHSTCP bandwidth shares is

Table 2. The throughput ratio of two high-speed flows over various RTT ratios under 1Gbps bottleneck link

	HSTCP	STCP	BIC	eHSTCP
RTT ratio = 3	42.46	111.45	12.03	3.88
RTT ratio = 6	197.80	341.65	84.65	4.77

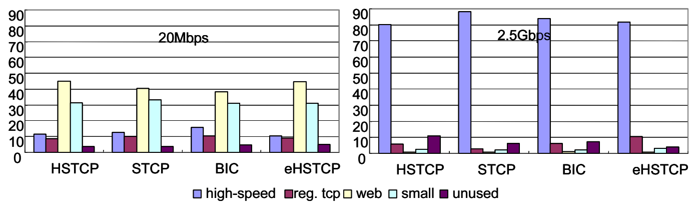

Fig. 1. A comparison of TCP friendliness for various bandwidth networks.

due to the better utilization of the available bandwidth. In most cases, eHSTCP shows good TCP friendliness but we did not present simulation results with various bandwidth here due to lack of space.

4 Conclusion

In this paper, we propose a new variant of TCP for a high-speed network which combines delay-based congestion control with loss-based congestion control. We define the effective RTT and adopt it to refine the HSTCP's AIMD mechanism. We have shown that the proposed scheme outperforms other high-speed protocols about fairness, scalability, and stability, while offering TCP friendliness.

Acknowledgement

This work was supported in part by the ITRC of the Ministry of Information and Communication (MIC), Korea.

References

1. S. Floyd, "HighSpeed TCP for Large Congestion Windows," *RFC3649*, 2003.
2. T. Kelly, "Scalable TCP: Improving Performance in Highspeed Wide Area Networks", *ACM SIGCOMM Computer Communication Review*, vol.33, pp. 83-91, 2003.
3. L. Xu, K. Harfoush, and I. Rhee, "Binary Increase Congestion Control for Fast, Long Distance Networks," In *Proceedings of IEEE Infocom*, 2004.
4. Y. Choi, K. Lee, and Y. Cho, "Performance Evaluation of High-Speed TCP Protocols with Pacing," *Lecture Notes in Computer Science*, 2004.
5. R. Shorten and D. Leith, "H-TCP: TCP for high-speed and long-distance networks," In *Proceedings of the PFLDnet*, 2004.