HSTCP-LP: A Protocol for Low-Priority Bulk Data Transfer in High-Speed High-RTT Networks

Aleksandar Kuzmanovic*, Edward W. Knightly*, R. Les Cottrell†
*Department of Electrical and Computer Engineering, Rice University
†SLAC/SCS-Network Monitoring, Stanford University

This work presents HSTCP-LP (High-Speed TCP Low Priority), a high-speed TCP stack whose goal is to utilize only the excess network bitrate (bandwidth) as compared to the "fair-share" of bitrate as targeted by other TCP variants. By giving a strict priority to all non-HSTCP-LP cross-traffic flows, HSTCP-LP enables a simple two-class prioritization without any support from the network. It enables large file backups to proceed without impeding ongoing traffic, a functionality that would otherwise require a multi-priority or separate network.

One class of applications for HSTCP-LP is low-priority background file transfer over high-speed networks. Examples are bulk data transfers of huge scientific data across the Internet, database replication, or Internet content distribution. A second class of applications is available bitrate optimization (e.g., to select a mirror server with the highest available bitrate). Current techniques first estimate the available bitrate and then download data via a transport protocol. HSTCP-LP, since it only uses excess/available bitrate, is able to estimate available bitrate while doing a useful data transfer.

We develop HSTCP-LP by merging two existing protocols: the first is High Speed TCP [1]; and the second is TCP-LP [2]. The goal is for HSTCP-LP to inherit the desired functionality of both, TCP-LP's ability to give strict priority to the cross-traffic, and HSTCP's efficiency in utilizing the excess network bitrate. Moreover, since HSTCP maintains strict fairness with current (non-high-speed) TCP implementations on low-speed links [1], it consequently enables HSTCP-LP to achieve a strict low-priority service in a broad span of networking environments: vs. current TCP implementations (e.g., TCP Reno) on low-speed links (in heavy or moderate packet drop ranges), and vs. high-speed TCP implementations (e.g., HSTCP [1], Scalable TCP [3], FAST TCP [4], BI-TCP [5], and H-TCP[6]) in high-rate networks.

On one hand, HSTCP-LP inherits two low-priority mechanisms from TCP-LP. First, in order to provide non-intrusive low-priority service, HSTCP-LP flows must detect oncoming congestion prior to cross-traffic flows. Consequently, HSTCP-LP uses inferences of one-way packet delays as early indications of network congestion rather than packet losses as used by the TCP-Reno-like cross-traffic flows. Second, HSTCP-LP inherits TCP-LP's congestion avoidance policy with two objectives: (1) quickly back off in the presence of congestion from the background flows and (2) achieve fairness among HSTCP-LP flows. On the other hand, HSTCP-LP inherits HSTCP's increase/decrease policy for large window sizes that enables it to quickly utilize and retain the available excess bitrate in the absence of sufficient cross-traffic. In summary, HSTCP-LP is a TCP-LP version with HSTCP-like agile properties, or alternatively, a HSTCP stack with built-in TCP-LP-like low-priority mechanisms.

However, HSTCP-LP is far from being a trivial fusion of the two ancestor TCP stacks. The key challenge in designing the protocol is overcoming a magnified tradeoff (when compared to lower-rate links) between the ability to successfully utilize the excess bitrate on one hand and to quickly backoff in moments of congestion on the other. For example, the original TCP-LP backoff policy that radically reduces window size when detecting persistent congestion (see reference [2] for details) is not entirely applicable to a high-speed environment since it can significantly degrade HSTCP-LP's performance. Consequently, HSTCP-LP applies a hybrid congestion avoidance scheme that utilizes TCP-LP-like mechanisms only in low excess-bitrate ranges, and then converges toward the less-backoff-responsive HSTCP policy as the window size increases.

Our implementation of HSTCP-LP is derived by modifying the Linux-2.4-22-web100 kernel, which by default uses the HSTCP stack. The HSTCP-LP source code is available at *http://www.ece.rice.edu/networks/TCP-LP/*. We perform an extensive set of Internet experiments on fast-production networks. In the majority of the experiments, we launch flows from SLAC (Stanford, CA) to UFL (Gainesville, FL), as well as from SLAC to UMICH (Ann Arbor, MI), with the maximum achievable bitrate on both paths being around 450 Mb/s. We perform experiments with and without a light periodic UDP cross traffic (the average is 10% of the maximum bitrate) to evaluate HSTCP-LP's ability to utilize the excess bitrate. Also, we multiplex a HSTCP-LP flow with the other TCP stacks to explore their mutual behavior.

Our results show that HSTCP-LP is able to utilize significant amounts of the excess bitrate when there is no cross-traffic in the network or when it multiplexes with a light periodic UDP traffic. On average, HSTCP-LP's performance is similar to the performance of other advanced TCP stacks, while the actual throughput varies in the 80% - 127% range (when compared to other high-speed TCP stacks) depending on various parameters such as the UDP cross-traffic period, the maximum window size or the sending-interface transmission queue length (*txqlen* in Linux). Next, our experiments show that HSTCP-LP is largely non-intrusive to other high-speed TCP stacks. HSTCP-LP consistently utilizes less bitrate than the other stacks when it multiplexes with them, and the level of prioritization dominantly depends on the bottleneck-queue length: it ranges from *strict* low prioritization for larger bottleneck queue lengths (when the maximum queuing delay[1] is approximately $\geq 50\,ms$) to somewhat lighter levels of prioritization for smaller queue lengths. Finally, we multiplex an HSTCP-LP flow with an aggregate of TCP Reno flows in a *high-speed* environment (on the SLAC-UMICH path). HSTCP-LP applies a more agile (than TCP Reno) window increase policy, yet uses one-way packet delays for early congestion indication. Our goal is to evaluate which of the above mechanisms is more prevalent. The experiment shows that HSTCP-LP utilizes only 4.5% of the bitrate in this scenario, thus confirming its low-priority nature. We are currently in the process of making more measurements that we will report in the paper.

REFERENCES

[1] S. Floyd, "Highspeed TCP for large congestion windows," Aug. 2003, Internet draft draft-ietf-tsvwg-highspeed-01.txt, work in progress.

[2] A. Kuzmanovic and E. Knightly, "TCP-LP: A distributed algorithm for low priority data transfer," in *Proceedings of IEEE INFOCOM '03*, San Francisco, CA, Apr. 2003.

[3] T. Kelly, "Scalable TCP: Improving performance in highspeed wide area networks," Dec. 2002, submitted for publication.

[4] C. Jin, D. Wei, S. H. Low, G. Buhrmaster, J. Bunn, D. H. Choe, R. L. A. Cottrell, J. C. Doyle, W. Feng, O. Martin, H. Newman, F. Paganini, S. Ravot, and S. Singh, "FAST TCP: From theory to experiments," Apr. 2003, submitted for publication.

[5] L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control for fast long-distance networks," July 2003, submitted for publication.

[6] R. N. Shorten, D. J. Leith, J. Foy, and R. Kilduff, "Analysis and design of congestion control in synchronized communication networks," June 2003, submitted for publication.

[1]We infer the maximum queuing delay by performing a parallel *ping* measurement.