Evaluation of Advanced TCP Stacks on on Fast Long-Distance Production Networks*

Hadrien Bullot
School of Computer & Communication Sciences
Swiss Federal Institute of Technology, Lausanne
hadrien.bullot@epfl.ch

R. Les Cottrell
Stanford Linear Accelerator Center
Stanford University, Menlo Park
cottrell@slac.stanford.edu

With the huge amounts of data gathered in fields such as High Energy and Nuclear Physics (HENP), Astronomy, Bioinformatics, Earth Sciences, and Fusion, scientists are facing unprecedented challenges in managing, processing, analyzing and transferring the data between major sites like SLAC in Menlo Park (California, USA) and CERN in Geneva (Switzerland) that are separated by long distance. Fortunately, the rapid evolution of high-speed networks is enabling the development of data-grids and super-computing that, in turn, enable sharing vast amounts of data and computing power. Tools built on TCP, such as bbcp, bbftp and GridFTP are increasingly being used by applications that need to move large amounts of data.

The AQM (Active Queue Management) of standard TCP (Transmission Control Protocol) has performed remarkably well and is generally known for having prevented severe congestion as the Internet scaled up. It is well-known that the current version of TCP - which relies on the Reno congestion avoidance algorithm to measure the capacity of a network - is not appropriate for high speed long-distance networks.

Today the major approach to improve the performance of TCP is that of adjusting the TCP window size to the product of the bandwidth and the RTT (Round Trip Time) of the network, and using parallel TCP streams.

In this paper, we will analyze the performance and the fairness of various new TCP stacks. We ran tests in 3 network configurations: short distance (10 ms), middle distance (70 ms) and long distance (170 ms). With these different network conditions, our goal is to find a protocol that is easy to configure, that provides optimum throughput, that is network friendly to other users, and that is responsive to changes in available bitrates. We tested 7 different TCP stacks: Parallel TCP Reno (P-TCP), S-TCP [4], Fast TCP [3], HS-TCP [2], HSTCP-LP [5], H-TCP [6] and Bic-TCP [7]. The main aim of this paper is to compare and validate how well the various TCPs work in real high-speed production networks.

We provide here a summary of the results below. More details and analysis of the results will be provided in the final paper.

RTT We found that all the evaluated advanced TCP stacks are "fair" with respect to the RTT (i.e. do not dramatically increase RTT) except for P-TCP Reno. On the short distance, the RTT of parallel TCP Reno increases from 10 ms to 200 ms. On long distances, the variation is less noticeable but still more apparent than for the other stacks.

*This work was supported in part by the Director, Office of Science, Office of Advanced Scientific Computing Research, Mathematical, Information, and Computational Sciences Division under the U.S. Department of Energy. The SLAC work is under Contract No. DE-AC03-76SF00515.

txqueuelen Scalable TCP by default uses a txqueuelen of 2000 but all the others use 100. Thus, we tested the various protocols with a txqueuelen sizes of 100, 2000 and 10000 in order to see how this parameter could change the throughput. In general, the advanced TCPs performs better with a txqueuelen of 100 except for S-TCP which performs better with 2000. With the largest txqueuelen, we observe more instability in the throughput.

Throughput Depending on the networks, we could achieve throughputs varying from 300 to 500 Mbps. In general the protocols have a similar throughput. P-TCP performs better than the others. Then comes, by order of throughput achievable, S-TCP, Bic-TCP, Fast-TCP, HSTCP-LP, HS-TCP, H-TCP and single stream TCP Reno. We notice for some protocols that a window size greater than the optimum - calculated using the bandwidth delay product - can hinder the performance.

Sinusoidal UDP The throughput of a protocol is not sufficient to describe its performance. Thus, we analyzed how the protocol behaves when competing with a UDP stream varying in a sinusoidal manner. Our results show that in general, all protocols respond quickly to changes in the available bandwidth and maintain a roughly constant aggregate throughput - especially for P-TCP and Bic-TCP.

Stability We defined stability as the coefficient of variation of the throughput. The results show that P-TCP, Bic-TCP and S-TCP are far more stable than the other protocols. Also these protocols remain stable when the window size is greater than optimal. With the sinusoidal UDP traffic, P-TCP and Bic-TCP remain stable and are more stable than the other advanced TCP stacks. We noticed that S-TCP, like H-TCP, is more stable with the largest window size than with the optimal window size contrary to the other stacks which become more unstable with a window size above optimal.

Intra-protocol fairness The intra-protocol fairness, defined by Jain and al. in [1], is the fairness between two flows of the same protocol. Each flow is sent from a different server. In general, all the protocols have a good intra-fairness except HS-TCP which have the poorest intra-protocol fairness with window sizes larger than the optimal.

Inter-protocol fairness For the inter-protocol fairness we sent two different flows on the link from two different machines. The aim of this experience was to see how each protocol behaves with a competing protocol. We desired that the protocol would neither be too aggressive nor too gentle (non-aggressive) towards the other protocols. We quantified the inter-protocol fairness by calculating the asymmetry in the throughput of the competing stacks. Our results show that Bic-TCP has the best performance (asymmetry closest to zero), HSTCP-LP (by design) is too gentle and P-TCP is too aggressive.

Reverse-traffic Reverse-traffic causes queuing on the reverse path. This in turn can result in the ACKs being lost or coming back in bursts (compressed ACKs). So, we tested the protocols by sending from SLAC to UFL an advanced TCP and from UFL to SLAC a P-TCP with 16 streams. Our tests show that Fast TCP - which is based on TCP Vegas that measures RTT - is more heavily affected by reverse-traffic that affects (usually increases) the reverse path delays and hence the RTTs.

References

[1] D. Chiu and R. Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. In *Computer Networks and ISDN Systems*, pages 1–14, June 1989.

[2] S. Floyd. HighSpeed TCP for large congestion windows. IETF Internet Draft, draft-floyd-highspeed-02.txt, February 2003.

[3] C. Jin, D. X. Wei, and S. H. Low. FAST TCP: Motivation, architecture, algorithms, performance. In *Submitted to IEEE INFOCOM 2004*, Hong Kong, 2004.

[4] T. Kelly. Scalable TCP: Improving performance in highspeed wide area networks. *Submitted for publication*, December 2002.

[5] A. Kuzmanovic and E. W. Knightly. TCP-LP: A distributed algorithm for low priority data transfer. In *IEEE INFOCOM*, San Francisco, April 2003.

[6] R. Shorten, D. Leith, J. Foy, and R. Kilduff. Analysis and design of congestion control in synchronised communication networks, 2003.

[7] L. Xu, K. Harfoush, and I. Rhee. Binary increase congestion control for fast, long distance networks. *Submitted for publication*, 2003.