

Transport Fairness Characterization and Evaluation – comparison of variants of TCP and UDP based transport with respect to transient and steady state traffic on real networks

Ruchi Gupta, Saad Ansari, R. Les Cottrell (all of SLAC) & Richard Hughes-Jones (Manchester)
SLAC, 2575 Sand Hill Road, Menlo Park, CA 94025
University of Manchester, Oxford Road, Manchester, UK M13 9PL
{Cottrell, ruchig} (at) slac.stanford.edu, R.Hughes-Jones (at) manchester.ac.uk

Introduction:

The existing TCP protocol (Reno TCP) does not perform well in different type of scenarios, due to its AIMD congestion control algorithm. Through this paper we intend to characterize and evaluate the Fairness of different TCP stacks (Scalable, HSTCP, HTCP, Fast TCP, Reno, BICTCP, HSTCP-LP and LTCP) and a different transport protocol namely UDTv2 (UDP based application level transport protocol) on both production and testbed networks. Analysis is done with respect to both the transient traffic (entry and exit of different streams) and the steady state traffic on production Academic and Research (A&R) networks, using different hardware setup. We will also report on measurements made on 10Gbits/sec testbeds set up for SC2004.

Experimental Setup:

The experimental setup on the sender side for the production A&R networks is done with machines (Intel Xeon 3.06GHz) running GNU/Linux 2.4.22 through GNU/Linux 2.4.25 kernels, patched with advanced TCP stacks. On the receiver's side a standard Linux kernel without any patch is used. On the sender side we have 2 machines, one running ping and the other machine runs iperf with the advanced TCP stack. We run iperf with a report interval of 1 second. With iperf we specify the maximum window size the congestion window can reach as 16384k. For the receiver side we have chosen 3 different nodes depending on the Round Trip Time (RTT) seen from SLAC, small (Caltech – RTT 10ms), medium (UFL – RTT 80ms) and large (CERN –RTT 180 ms).

For SC2004 [10], we will have two dedicated 10Gbits/s circuits from Pittsburgh to Sunnyvale. In addition we will have 10 Sun V20Z and one V40Z AMD Opteron 64 bit based hosts all running Solaris 10 or Linux 2.6. Six of these hosts will be at Pittsburgh and five at Sunnyvale. The above hosts will have a mix of 10Gbits/s Network Interface Cards (NICs) from Chelsio (with a TCP Offload Engine), S2IO and Intel. We also hope to make measurements with Intel Xeon based hosts.

Short descriptions of the various advanced TCP stacks can be found in [9]. More details can be found in the papers describing the stacks and referenced in [9]. UDT [8] is a UDP based transport protocol, developed to overcome the efficiency and fairness of the existing TCP stacks. It is implemented in user space so it requires no kernel modifications.

Result and Analysis:

We run four TCP flows, one after the other each separated by an interval of 2 minutes, and the complete test ran for approximately 17 minutes. The flows leave in a LIFO (Last In First Out order). As shown below in Figure 1, we divide the experiment into seven regions (regions 1,2,3,5,6 & 7 for 2 minutes and region 4 for 5 minutes) and statistics are collected for each of the

seven regions as well as per individual flows. Aggregate throughput values are also collected for each of the regions as well as for the overall test.

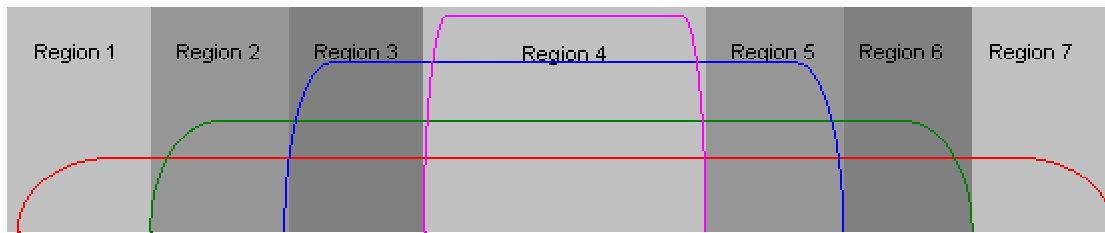


Figure 1: Seven Flow regions in the Experiment

Table 1, shows aggregate (for all seven flows) values for average throughput, standard deviation, stability [9], minimum and maximum fairness index, sender % CPU utilization and MHz/Mbps for SLAC to CERN test. Results for Caltech and UFL will be presented in the final paper. Detailed Stability and fairness indices for each of the 7 regions are shown in Table 2.

In Figure 2, the graph of achievable throughput per flow for Reno TCP is shown; graphs for other protocols will be presented in the final paper with their respective analysis. The flows are colored according to the Figure 1 color scheme and they enter and leave the experiment at an interval of 2 minutes each. We see that whenever a flow enters or leaves the network the average bandwidth changes due to bandwidth distribution among different flows. For a stable and fair protocol we would want that whenever a flow leaves the network, the remaining flows get back to almost their original bandwidth. Similarly whenever a new flow enters the network the bandwidth should be fairly distributed among all the flows (Fairness of 1 is the desired value).

As see from Figure 2, the aggregate throughput decreases dramatically when the second flow (Green) enters the network. Similarly as the other flows enter the network the aggregate throughput keeps decreasing and we also observe that the bandwidth is not fairly divided among different flows. When a flow leaves the network, the ideal behavior would be that the available bandwidth is distributed fairly among the remaining flows, but this is not what we observe from our tests. We observe that the aggregate throughput is not able to recover back to near its initial value even after the flow 2, 3, 4 have departed the network.

The results appear to confirm the theoretical and simulation results seen by the Hamilton Institute team [11] where packets being sent in bursts lead to lockout, gross unfairness and relatively long convergence times following the bursts (in our case sometimes caused by the slow start of a new flow). The final paper will compare and contrast the effectiveness of the various TCP stacks and UDT with respect to such lockout and unfairness behavior.

$$F = \frac{\sum_{i=1}^n \bar{x}_i)^2}{n \sum_{i=1}^n \bar{x}_i^2}$$

Average = μ , Standard Deviation = SD, Stability = SD / μ , Fairness (F) =

TCP Stack	Average (μ)	SD	Stability (SD/ μ)	Fairness Min-Max	CPU % Utilization	MHz/Mbps
Reno	248.40	163.25	0.66	0.59-1.0	0.02	0.63
HSTCP	255.06	187.99	0.74	0.79-1.0	0.03	0.90
HTCP	402.08	113.10	0.28	0.99-1.0	0.03	0.65
Scalable	423.90	115.88	0.27	0.82-1.0	0.03	0.64
Fast-TCP	335.29	110.30	0.33	0.58-1.0	0.03	0.67
LTCP	350.52	120.47	0.34	0.56-1.0	0.02	0.67
HSTCP-LP	228.87	114.62	0.50	0.64-1.0	0.02	0.65
BICTCP	412.34	117.50	0.28	0.98-1.0	0.03	0.71
UDTv2	391.83	136.35	0.35	0.95-1.0	0.73	16.2

Table 1: Aggregate Values for all the 7 Regions

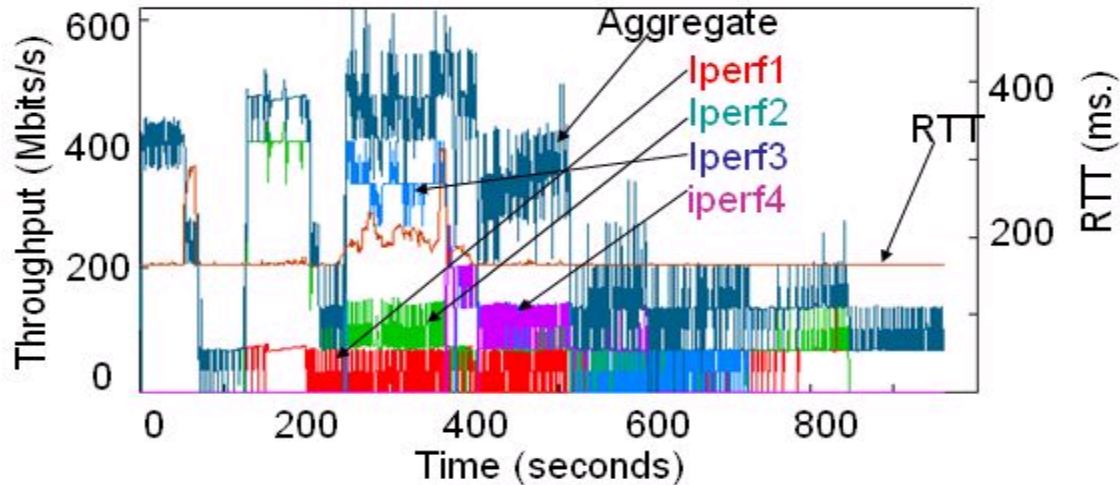


Figure 2: Bandwidth and RTT vs. time for Reno TCP (SLAC-CERN)

Flow 1 is represented by RED, Flow 2 by GREEN, Flow 3 by BLUE & Flow 4 by PINK

TCP stack	Region1	Region 2	Region 3	Region 4	Region 5	Region 6	Region 7
Reno	0.70/1.0	0.46/0.67	0.24/0.59	0.49/0.85	0.46/0.98	0.23/0.99	0.40/1.0
HSTCP	0.13/1.0	0.30/0.95	0.24/0.97	0.55/0.79	1.02/0.99	0.63/1.0	0.40/1.0
HTCP	0.33/1.0	0.26/1.0	0.30/0.99	0.24/0.99	0.20/1.0	0.12/1.0	0.40/1.0
Scalable	0.13/1.0	0.36/0.99	0.35/0.82	0.17/0.99	0.21/1.0	0.36/0.99	0.18/1.0
Fast-TCP	0.61/1.0	0.36/0.62	0.28/0.74	0.16/0.80	0.30/0.73	0.23/0.58	0.38/1.0
LTCP	0.67/1.0	0.18/0.99	0.33/0.82	0.31/0.76	0.25/0.56	0.19/0.97	0.15/1.0
HSTCP-LP	0.12/1.0	0.28/0.64	0.23/0.70	0.26/0.75	0.21/0.71	0.69/0.99	0.35/1.0
BICTCP	0.35/1.0	0.30/0.98	0.14/0.98	0.17/0.98	0.12/0.99	0.19/0.98	0.34/1.0
UDTv2	0.39/1.0	0.25/1.0	0.50/0.95	0.25/1.0	0.24/1.0	0.26/0.99	0.43/1.0

Table 2: Stability / Fairness for all the seven regions for (SLAC-CERN) experiment.

Future Work:

In the final paper results from tests made on 10Gbits/s testbeds at SC2004 will be included. These will include results from using various NICs, different operating systems, TCP stacks and host hardware

The next set of experiments will entail the analysis of cross-traffic measurement. In this setup we will have a third machine on the sender side, which will run the advanced TCP stack for cross-traffic. We will do some detail analysis on the effect of cross-traffic on bandwidth and RTT in particular. We will also make in-depth analysis of UDTv2.0, which supports user defined congestion control algorithms and compare its performance with respect to the advanced TCP stacks. Results from the other two nodes namely Caltech and UFL will also be presented in the final paper.

References:

- [1] S. Floyd. HighSpeed TCP for large congestion windows. IETF Internet Draft, draft-oydhighspeed-02.txt, February 2003.
- [2] R. Shorten, D. Leith, J. Foy, and R. Kildu. Analysis and Design of Congestion Control in Synchronized Communication Networks, 2003.
- [3] T. Kelly. Scalable TCP. Improving Performance in Highspeed Wide Area Networks, 2002.
- [4] C. Jin, D. Wei, S.H. Low, G. Bushmaster, J. Bunn, D.H. Choe, R. L. Cottrell, J.C. Doyle, W. Feng, O. Martin, H. Newman, F. Paganini, S. Ravot, and S. Singh. Fast TCP – from Theory to Experiments. PFLDnet 2003, Geneva, February 2003.
- [5] S. Bhandarkar, S. Jain and A.L.N. Reddy. LTCP: A Layering Technique for Improving the Performance on TCP in Highspeed Networks.
- [6] A. Kuzmanovic and E.W. Knightly. TCP-LP: A Distributed Algorithm for Low Priority Data Transfer. In IEEE INFOCOM, San Francisco, April 2003.
- [7] L. Xu, K. Harfoush, and I. Rhee. Binary Increase Congestion Control (BIC) for Fast, Long-Distance Networks, INFOCOM 2004.
- [8] Y. Gu and R.L. Grossman. UDT: A UDP based Transport Protocol.
- [9] H. Bullo, R. L. Cottrell, and R. Hughes-Jones. Evaluation of Advanced TCP Stacks on Fast Long-Distance Production Networks, PFLDnet 2004.
- [10] High Speed Terabyte Data Transfers for Physics - SC2004 Bandwidth Challenge Proposal <http://www-iepm.slac.stanford.edu/monitoring/bulk/sc2004/hiperf.html>
- [11] D. Leith, R. Shorten. H-TCP Protocol for High-Speed Long Distance Networks, PFLDnet 2004.