

# An extended evaluation of a collection of TCP Congestion Control Algorithms

Harhalakis Stefanos

Department of Informatics, TEI of Thessaloniki, Greece

v13@priest.com

Samaras Nikolaos, Fragiadaki Eleni

Department of Applied Informatics, University of Macedonia, Greece

samaras@uom.gr, eleni.fra@gmail.com

## Abstract

TCP is the most used Transport Layer protocol in the Internet. Started as a reliable transmission method it has evolved through time to a mix of some quite complex protocols. Its ability to provide flow control is a quite complex research area and a well studied part with a multitude of algorithm proposals. Linux is a widespread open-source operating system that provides a solid TCP/IP network stack and implements 11 flow control algorithms. The study is an experimental comparison and evaluation of Reno, Reno, HighSpeed, BIC, CUBIC, Westwood, H-TCP, Hybla, Vegas, Scalable and Low-Priority algorithms. All 11 algorithms were benchmarked by varying link delay, packet loss, and the number of parallel connections which resulted to more than 9.000 different runs.

During the last 5 years the overall available bandwidth of Internet was greatly increased. Typical end-user connection speed increased 50-100 times, corporate Internet connections jumped to speeds no less than 2Mbit, Gigabit Ethernet was introduced and deployed as the major backbone and server connection technology. Apart from that wireless networks are a commonplace for home, corporate and Metropolitan Area Networks (MANs) networking. While research on wired Local Area Networks (LANs) and Wide Area Networks (WANs) in conjunction with fiber optics almost eliminated packet loss, wireless networks still result in very high packet loss rate. Since existing networks experienced such a dramatic improvement, protocols needed to evolve. TCP[1] being the most used Transport Layer protocol for IP based networks became the center of great research activities. The flow control ability of TCP is implemented by Congestion Control Algorithms that dictate how various aspects of dynamic TCP parameters, like window size[2], are changed.

Recent research includes an experimental comparison of Scalable, HighSpeed, BIC, Fast-TCP and H-TCP [3] and an initial evaluation of BIC, CUBIC, Fast-TCP, HighSpeed, H-TCP and STCP[4]. Since Internet is an extremely complex network it is impossible to emulate its behavior. Factors that affect TCP efficiency, overall bandwidth and throughput consist of the variance of the behavior of deployed congestion control algorithms, the different network technologies that a packet traverses, the multitude of queuing algorithms, the unpredictable rate of continuous or momentary packet loss, the number of active data transfers through each network link and many more, all of them interacting with each other.

TCP algorithms share some basic goals like overall bandwidth utilization and bandwidth fairness. In contrast with other technologies, TCP algorithms must not compete. Instead they have to cooperate with their selves and with each other to provide increased overall throughput instead of high transfer rates for individuals. Since the

majority of Internet connections are bursty and short-lived, TCP algorithms need also to be able to provide small Round Trip Times (RTTs) apart from high throughput.

In this paper we present the experimental results of an extended evaluation of TCP Congestion Control Algorithms. To accomplish that we took advantage of the Linux operating system which includes a solid and well tested network TCP/IP stack. Along with BSD derivatives they are being used as the basis of almost all on-going networking research efforts. As of kernel version 2.6.20, Linux includes an implementation of 11 Congestion Control Algorithms: Vegas, newReno, Westwood, Veno, Scalable, Highspeed, Low-Priority, BIC, H-TCP, Hybla and CUBIC, listed in year of publication order.

We benchmarked all these algorithms using one-way, symmetrical transfers, emulating an homogeneous, high-speed, not-congested with background traffic, network. We ran tests using 11 packet loss rates from 0% to 30%, 1-10 parallel connections and 11 propagation delays from 0ms to 512ms. Each test ran for 15 seconds, providing 7 intermediate samples and one cumulative result per connection. This resulted in more than 450.000 data entries, more than 60.000 cumulative results and more than 9.600 benchmark runs. Since each run lasted 15 seconds, the running time was 40hours. The actual benchmark time was about 4 full days because of overhead. We provide results for individual connection throughput, cumulative throughput and fairness.

We created a plethora of tests by varying propagation delay (0-512 using powers of 2), packet loss (0% - 30% using 2% steps), number of parallel connections (1,3,5,7 or 10) and Congestion Control Algorithm. During each test we recorded data once every 2 seconds and a cumulative one at the end. Recorded data also include detailed information about each individual TCP connection. Results include Throughput and Fairness for all combinations of the above parameters.

For our study we used one IBM Compatible PC with two 1Gbit Network Interface Cards (NICs), each one operating at 100Mbit. The PC was connected to itself via a 1m FTP ethernet cable and was running Linux kernel 2.6.20 using 250Hz as the scheduler timer frequency. For the needs of this experiment we wrote a Linux based benchmarking tool in C/C++ customized for our needs which provided us with a wealth of data, a lot more than we would have got by using one of the existing opensource benchmarking tools used by other researchers. We also used PostgreSQL database for data storage and a set of custom shell scripts for automating the runs. The results were studied using a custom python program that rendered the graphs.

The analysis of the experimental results lead us to quite interesting conclusions, not shown by any of the related papers we have examined. For example we found out that two of these algorithms are ineffective for normal use since they are very unfair. Even though one of them is the Low-Priority algorithm which is supposed to consume only the available bandwidth, it cannot be trusted for use in the Internet. Conclusions also include the suggested algorithm for today's dial-up connections and DSLs, knowing that they are characterized by their packet loss and propagation delays. Since the research provided us a great number of collected data we are able to present the best of hundreds of graphs showing various aspects of TCP behavior under different circumstances.

## REFERENCES

- [1] J. Postel, "Transmission Control Protocol," RFC 793 (Standard), Sep. 1981, updated by RFC 3168. [Online]. Available: <http://www.ietf.org/rfc/rfc793.txt>
- [2] V. Jacobson, R. Braden, and D. Borman, "TCP Extensions for High Performance," RFC 1323 (Proposed Standard), May 1992. [Online]. Available: <http://www.ietf.org/rfc/rfc1323.txt>
- [3] Y.-T. Li, D. Leith, and R. N. Shorten, "Experimental evaluation of tcp protocols for high-speed networks," Hamilton Institute, NUI Maynooth, Tech. Rep., 2005.
- [4] S. Ha, Y. Kim, L. Le, I. Rhee, and L. Xu, "A step toward realistic performance evaluation of high-speed tcp variants," Department of Computer Science, North Carolina State UniversityDepartment of Computer Science, University of Nebraska, Tech. Rep., 2006.