Comparative performance evaluation of TCP Hybla and TCP Cubic for satellite communication under low error conditions

Conference Paper · December 2010

DOI: 10.1109/IMSAA.2010.5729424 · Source: IEEE Xplore

CITATIONS
6

READS
302

4 authors, including:

Siddharth Trivedi
Defence Research and Development Organisati…
1 PUBLICATION **6** CITATIONS
SEE PROFILE

Shrisha Rao
International Institute of Information Technolo…
93 PUBLICATIONS **464** CITATIONS
SEE PROFILE

Proceedings of the 4th IEEE International Conference on Internet Multimedia Systems Architecture and Application (IMSAA-2010), Bangalore, December 15-17, 2010.

Comparative Performance Evaluation of TCP Hybla and TCP Cubic for Satellite Communication Under Low Error Conditions

Siddharth Trivedi, Sanjay Jaiswal, Rituraj Kumar
Centre for Artificial Intelligence and Robotics
Bangalore, India
{siddharthtrivedi,sanjaykumarjaswal,rituraj}@cair.drdo.in

Shrisha Rao
International Institute of Information Technology - Bangalore
Bangalore, India
shrao@ieee.org

***Abstract*—Satellite communication has become important in the current global data communication due to its large coverage area. Satellite links are affected by two major concerns: long propagation delays, and relatively high error rates. Standard TCP congestion control variants were not designed to address these challenges. They generally depend on round trip times (RTTs), and hence suffer heavily if the RTTs are large. Various TCP variants like Hybla have been proposed in order to optimize the channel throughput in heterogeneous networks incorporating satellite links. The error rate of satellite channels can be substantially mitigated using forward error correction (FEC) techniques. In such a situation, according to the results obtained by our comparative study, the performance of Cubic, which is designed for high-speed networks, is better than Hybla, in conditions of high-latency and low error rates.**

***Index Terms*—congestion control, satellite communication, TCP variants, Hybla, Cubic**

I. Introduction

Experimental evidence has established that TCP variants suffer from performance limitations due to communication delays and probability of bit errors [1]. These limitations are becoming increasingly important for satellite communication (sat-com) systems, due to their large-scale deployment for supporting globally dispersed work centers. The detailed analysis of file transfer as reported in literature reveals that the *throughput* of a TCP channel depends on the congestion window size and on the process used to manage this window size. Slow start and congestion control are the two key steps of this process. The standard versions of the TCP protocol (Tahoe, Reno, New Reno) include two phases, namely Slow Start (SS) and Congestion Avoidance (CA), where CA is a part of congestion control followed by fast re-transmit with fast recovery. In the slow-start phase, a sender opens the congestion window (*cwnd*) exponentially, doubling *cwnd* every Round-Trip Time (RTT) until it reaches the slow-start threshold (*ssthresh*) or encounters packet loss. The connection then switches to CA, where *cwnd* grows conservatively, by only 1 packet every RTT (or linearly). The initial *ssthresh* is set to an arbitrary default value, ranging from 4 K to 64 K bytes, depending on the operating system implementation. By setting the initial *ssthresh* to an arbitrary value, TCP performance may suffer from two potential problems: (a) if *ssthresh* is set too high relative to the network bandwidth delay product (BDP), the exponential increase of *cwnd* generates too many packets too fast, causing multiple losses with significant reduction of the connection throughput; (b) if the initial *ssthresh* is set too low relative to the BDP, the connection switches to linear *cwnd* growth prematurely, resulting in low utilization of the channel. In the event of significant packet losses or large delays, TCP's recovery mechanism performs quite poorly, as it erroneously deduces the reason as congestion, causing totally inappropriate activation of the congestion avoidance mechanisms [2], [3]. This is due to the TCP window-based transmission algorithm which, being triggered by acknowledgment (ACK) arrivals, depends on network delays. Long RTTs reduce the congestion window growth rate and result in significant throughput degradation. In this case, the performance degradation is a direct consequence of the intrinsic behavior of the TCP protocol's window management and its dependency on RTTs.

Several mechanisms have been proposed in order to compensate the erroneous ascription of congestion, and to provide better congestion window management as well as independence from RTT [4]. The proposed TCP variants like Hybla [5] address long RTTs, and Westwood [6] addresses high error rates in low-latency networks. TCP Variants like Cubic [7], BIC [8], FAST [9], HSTCP [10], STCP [11], and HTCP [12] are major improvements for the small-RTT case. The performance of Hybla has been reported to be better than Westwood in case of high-latency and high error channels [13].

The improvements in TCP congestion control using Hybla for high-latency networks with high error rates, like sat-com links, have already been established [14]. Similarly, the advantages of Cubic in high-speed networks have been reported [7]. With the usage of robust FEC techniques deployed in most static distributed work centers, the error rates incurred due to bit corruption at the transport layer are steadily reducing [15]. This fact motivates us to consider sat-com links with low error rates, for this paper.

This paper presents a comparative analysis of Hybla and Cubic for high-latency networks with low error rates. The NS-2 network simulation tool using the TCP-Linux patch [16] is

978-1-4244-7932-0/10/$26.00

used for the simulation of Hybla and Cubic over heterogeneous sat-com networks (satellite + wired links).

The paper is organized as follows. In Section II, mechanisms for congestion window management of Cubic and Hybla are presented. In Section III, the benchmark network topology and the simulation set-up are specified along with performance evaluations of Cubic and Hybla, and their interpretations. In Section IV, conclusions are derived based on simulation results.

II. TCP Variants

A. Cubic

Cubic [7] is an implementation of TCP with an optimized congestion control algorithm for high-speed networks. This is an improved version of BIC-TCP with better window control and friendliness in high-speed environments. It greatly simplifies the window adjustment algorithm of BIC-TCP by replacing the concave and convex window growth portions of BIC-TCP by a cubic function (which contains both concave and convex portions). In fact, any odd-order polynomial function has this shape—the choice of a cubic function is incidental and for convenience. A key feature of Cubic is that its window growth depends only on the time between two consecutive congestion events.

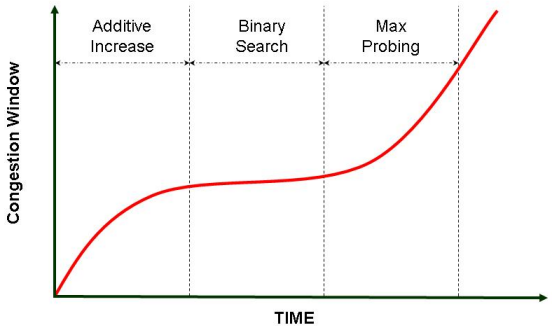

Fig. 1. Window Growth Function of BIC

Fig. 1 shows the window growth function of BIC TCP which achieves good scalability and stability in a high-speed network while in a low-speed network, the window growth function is still too aggressive. Several phases (additive increase, binary search increase and max probing) of window control add complexity in implementing the protocol and analyzing its performance. Cubic, while retaining the strength of BIC (specially its scalability and stability), simplifies the window control and maximizes its throughput.

More specifically, the congestion window of Cubic is determined by the following function:

$$W_{cubic} = C(t-K)^3 + W_{max} \tag{1}$$

where C is a scaling factor, t is the elapsed time from the last window reduction, W_{max} is the window size just before the last window reduction, and $K = \sqrt[3]{W_{max}\beta/C}$, where β is a constant multiplication decrease factor applied for window reduction at the time of loss event (i.e., the window reduces to βW_{max} at the time of the last reduction).

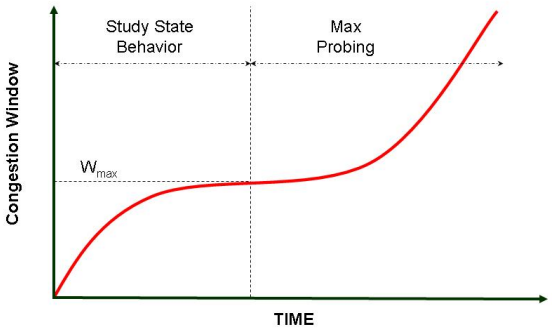

Fig. 2. Window Growth Function of Cubic

Fig. 2 shows the growth function of Cubic with the origin at W_{max}. The window grows very fast initially, but as it gets closer to W_{max}, it slows down its growth. Around W_{max}, the window increment becomes almost zero. Above that, Cubic starts probing for more bandwidth in which the window grows slowly initially, accelerating its growth as it moves away from W_{max}. This slow growth around W_{max} enhances the stability of the protocol, and increases the utilization of the network while the fast growth away from W_{max} ensures the scalability of the protocol.

B. Hybla

In heterogeneous networks, TCP connections characterized by large RTTs (i.e., including a path through a satellite) achieve a poor performance as compared to wired connections with lower RTT values. TCP Hybla [5] is mainly focused on preventing connections having a high-latency getting "punished" by this high latency. The basic idea of the TCP Hybla is to obtain the same transmission rate of packets of a comparatively fast reference TCP connection (e.g. wired) for long RTT connections (e.g., sat-com and wireless). The congestion window update rules for the standard TCP implementation (eg. Tahoe, Reno, New Reno) can be expressed as equation (2) when we have an index i, which denotes the reception of the i^{th} ACK.

$$W_{i+1} = \begin{cases} W_i + 1 & SS \\ W_i + \frac{1}{W_i} & CA \end{cases} \tag{2}$$

Rewriting this formula for the time domain gives more insight into the performance. For the SS phase, this results in a discrete exponential increase with RTT, as the congestion window is doubled at every RTT. For the CA phase, the growth is one segment per RTT and therefore amounts to a linear increase with time. Denoting by t_γ the time at which the *ssthresh*, γ, is reached, this gives

$$W(t) = \begin{cases} 2^{\frac{t}{RTT}} & 0 \le t < t_\gamma \quad SS \\ \frac{t-t_\gamma}{RTT} + \gamma & \text{t} \ge t_\gamma \quad CA \end{cases} \tag{3}$$

where $t_\gamma = RTT \log_2 \gamma$.

For performance comparison, it is useful to calculate the amount of data transmitted, which can be the segment transmission rate, defined as

$$B(t) = \frac{W(t)}{RTT} \tag{4}$$

Now the key motivation behind TCP Hybla is to make the data transfer rate independent of the RTT. This is achieved by using a normalized RTT (ρ), which is defined as follows:

$$\rho = \frac{RTT}{RTT_0} \tag{5}$$

The RTT_0 is the round trip time of a connection (the reference link) with a lower RTT than the current connection (the TCP Hybla link). It is desired in TCP Hybla to be able to compete with this low RTT connection, *goodput* wise.

To equalize the window growth with the reference link, the RTT variant part of the equation (3) is multiplied by the ρ. This results in an equation where the congestion window growth is determined by the RTT_0, giving the connection the ability to grow its window W^H just as fast as the reference link. Next, the resultant equation is multiplied by ρ to compensate for the fact that the window can only update each RTT and not RTT_0. Applying these rules we get:

$$W^H(t) = \begin{cases} \rho 2^{\rho \frac{t}{RTT}} & 0 \le t < t_\gamma \quad SS \\ \rho \left[\rho \frac{t - t_\gamma}{RTT} + \gamma \right] & \mathrm{t} \ge t_\gamma \quad CA \end{cases} \tag{6}$$

Here, superscript H indicates window growth rate for Hybla.

Now from equation (4) and equation (6), we have:

$$B^H(t) = \begin{cases} \frac{2^{\frac{t}{RTT_0}}}{RTT_0} & 0 \le t < t_\gamma \quad SS \\ \frac{1}{RTT_0} \left[\frac{t - t_\gamma}{RTT_0} + \gamma \right] & \mathrm{t} \ge t_\gamma \quad CA \end{cases} \tag{7}$$

As can be observed, $B^H(t)$ is achieved without any dependence on the RTT. When rewriting this formula to congestion window update rules, the "normal" TCP congestion window update rules of equation (2) are replaced by:

$$W^H_{i+1} = \begin{cases} W^H_i + 2^\rho - 1 & \mathrm{S}S \\ W^H_i + \frac{\rho^2}{W^H_i} & \mathrm{C}A \end{cases} \tag{8}$$

So the final objective of Hybla to make the data transmission rate independent of RTT in order to get maximum throughput that can be achieved by implementing the above algorithm.

III. Performance Evaluation of Cubic and Hybla

A. Simulation Setup and Topology

The simulation has been done on NS-2 with TCP-Linux patch [16], to support various TCP congestion control algorithms. The implementation loosely follows the Linux TCP implementation, and has been shown to produce results comparable to Linux experimental results. The network topology considered in simulation runs is shown in Fig. 3 and has incorporated a heterogeneous sat-com network comprised of wired and satellite segments with appropriate bandwidth, propagation delay, and error rate.

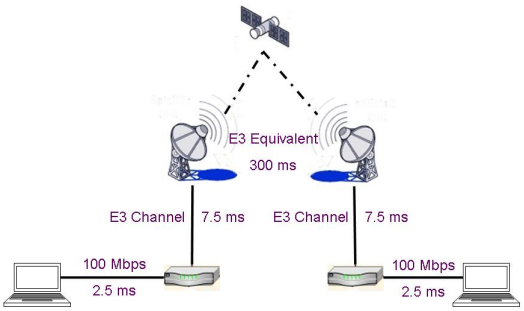

Fig. 3. Network Topology

All the connections share a bottleneck link, whose limited bandwidth has been deliberately chosen to study the effects of congestion. For wired connections in local area networks, the bandwidth is chosen as 100 *Mbps*, while the outer channel assisted with satellite link is presumed to be an E3 channel or equivalent with 34.368 *Mbps*. The propagation delay for a wired connection varies from 2.5 *ms* to 7.5 *ms* while the two-way propagation delay of the satellite link varies in such a way that the RTT of the connection ranges from 25 *ms* (considered only for comparison purposes with the wired connections) to 600 *ms* (corresponding to the case of a GEO Satellite link). The wired links are assumed to be error free, while a uniformly distributed error model with a packet error rate of 0.01% is adopted for both the up-link and the down-link of the satellite channel. The choice of channel bandwidths as well as PER have been modeled on reference field-deployed systems, which have strong FEC mechanisms implemented at lower layers of the stack.

B. Measures of Performance

Using the simulation setup described, a comparative study of the two algorithms is carried out on the basis of the following metrics:

1) **Number of acknowledgments received:** The number of acknowledgments received is an indicator of the successful transmission of packets. The higher the number of acknowledgment packets, greater is the data transmitted in a given time. So the number of acknowledgments received can very well be used as a performance metric in order to compare the two TCP variants.
2) **Goodput:** *Goodput* is defined as the amount of useful information that is delivered per unit time excluding protocol overhead and retransmitted data packets. Goodput is taken as a performance metric instead of throughput, because throughput is not a well-defined parameter when dealing with protocol overhead. Throughput is typically measured at a reference point below the network layer

and above the physical layer. Throughput often is thus an unreliable measurement of perceived bandwidth.

C. Impact of Congestion Window Growth Rate

The congestion window growth rate is the major differentiator for all TCP variants. Channel utilization can be effectively maximized by optimizing *cwnd*. Growth of *cwnd* with respect to the simulation time is as shown in Fig. 4. In the case of Hybla, the algorithm starts very aggressively, and begins to face packet losses very soon. This is observed as the initial spike. While it remains stable at a reasonably high value for some time, it eventually faces large packet losses, and drops to an average that is lower than that of Cubic. Another key observation is that Hybla displays consistent and large oscillations, when compared to Cubic that is far more stable. Due to lesser degree of oscillations, Cubic is expected to result in greater stability of the network, and produce greater predictability in network performance, specially when the network would include routers.

The abrupt drop in the *cwnd* in the case of Hybla can be explained by the packet loss due to the aggressive start. Hybla sets its congestion window to the defined maximum size at the start, while Cubic begins with a congestion window of minimum defined size. Hybla blasts packets into the channel, while cubic starts sending packets quite slowly to reach *ssthresh*. According to our simulation study, the window size in case of Hybla goes down compared to Cubic with high oscillation. The oscillations can be explained by the fact that the increase in RTT due to packet loss or congestion continues to accelerate the growth of ρ, which in turn leads to even larger *cwnd*. A larger *cwnd* leads to more packets reaching the network. Hence a positive feedback loop gets established, which along with the inherent delay in the system response, leads to oscillations. These oscillations were observed to persist even in very long simulation runs, due to the absence of damping coefficients in the system.

D. Performance Summary

The identified performance metrics (acknowledged packets as well as goodput) are emergent properties of the congestion window. The observed behavior of *cwnd* leads these performance metrics to show predictable behavior. A superior way to manage congestion windows results in better performance. As depicted in Fig. 5 and Fig. 6, both metrics have marginal advantage for Hybla initially, but as the simulation proceeds, they start falling behind Cubic.

It is however important to note that as the error rates become higher on high-latency links, the performance of Cubic degrades perceptibly below Hybla with respect to the identified metrics [14]. This happens as Cubic tends to get increasingly conservative with reference to the *cwnd* as packet drops get higher with high PER. On the other hand, Hybla continues to remain aggressive.

IV. Conclusions

The results presented in this paper establish that in high-latency and low error rate channels, like sat-com links with

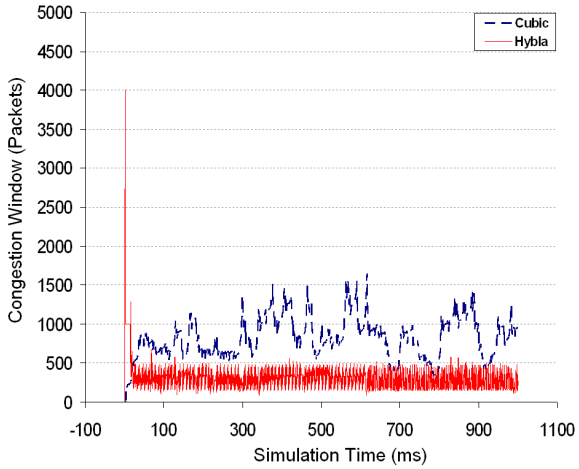

Fig. 4. Congestion Window/Time

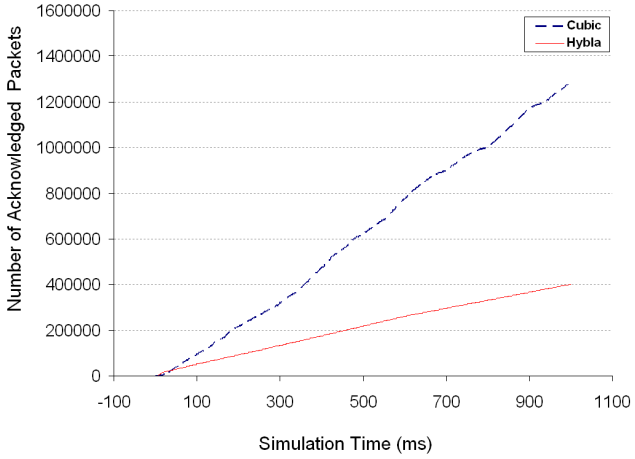

Fig. 5. Acknowledgment Packet/Time

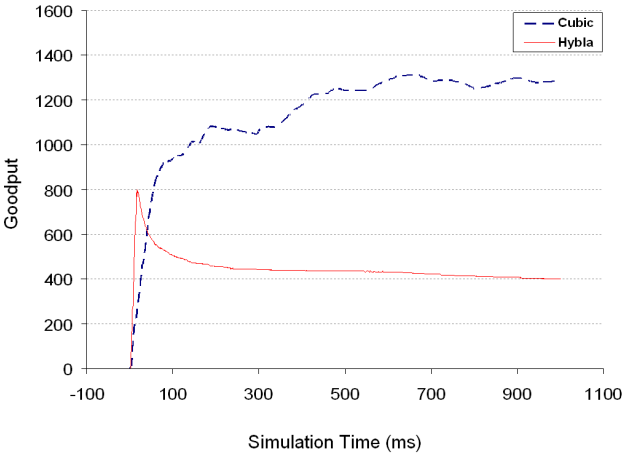

Fig. 6. Goodput/Time

link-level FEC, Cubic performs better than Hybla. The aggressive *cwnd* policy adopted by Hybla results in an overall degraded performance in the low PER conditions. Further, the oscillatory behavior of the *cwnd* in the case of Hybla would make the network performance less predictable, and unacceptable for many applications.

From a system engineering perspective, the results present two design options. The first is to use raw satellite channels without FEC, which would have high-latency and high PER, and leverage Hybla for TCP congestion control. An alternative approach would be to use robust FEC on the satellite channel, and leverage the comparatively simpler and well evaluated Cubic algorithm for TCP congestion control.

REFERENCES

[1] H. Bullot, R. L. Cottrell, and R. Hughes-Jones, "Evaluation of advanced TCP stacks on fast long-distance production networks," *Journal of Grid Computing*, vol. 1, no. 4, pp. 345–359, Dec. 2003, doi:10.1023/B:GRID.0000037555.53402.4f.

[2] Y. Hu and V. Li, "Satellite-based internet: a tutorial." *IEEE Communication Magazine*, vol. 39, no. 3, pp. 154–162, Mar. 2001.

[3] G. Xylomenos, G. C. Polyzos, P. Mahonen, and M. Saaranen, "TCP performance issues over wireless links," *IEEE Communication Magazine*, vol. 39, no. 4, pp. 52–58, Apr. 2001.

[4] A. Afanasyev, N. Tilley, P. Reiher, and L. Kleinrock, "Host-to-Host Congestion Control for TCP," *IEEE Communications Surveys Tutorials*, vol. 12, no. 3, pp. 304–342, 2010.

[5] C. Caini and R. Firrincieli, "TCP Hybla: a TCP enhancement for heterogeneous networks," *International Journal of Satellite Communication and Networking*, vol. 22, no. 5, pp. 547–566, Sep. 2004.

[6] C. Casetti, M. Gerla, S. Mascolo, M. Y. Sanadidi, and R. Wang, "TCP Simulation Time (seconds) Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links," in *ACM Mobicom 2001*, Rome, Italy, Jul. 16-21 2001, pp. 287–297.

[7] S. Ha, I. Rhee, and L. Xu, "CUBIC : A new TCP-friendly high-speed TCP variant," *ACM SIGOPS Operating System Review*, vol. 42, no. 5, pp. 64–74, Jul. 2008.

[8] L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control (BIC) for fast long-distance networks," in *IEEE INFOCOM 2004*, 2004.

[9] C. Jin, D. X. Wei, S. H. Low, and S. Hegde, "FAST TCP: Motivation, architecture, algorithms, performance," *IEEE/ACM Transactions on Networking*, vol. 14, no. 6, pp. 1246–1259, Dec. 2006, doi:10.1109/TNET.2006.886335.

[10] S. Floyd, "HighSpeed TCP for large congestion windows," Internet Engineering Task Force, RFC 3649, Dec. 2003.

[11] T. Kelly, "Scalable TCP: Improving performance in high-speed wide area networks," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 2, pp. 83–91, Apr. 2003.

[12] R. Shorten and D. Leith, "H-TCP: TCP for high-speed and long-distance networks," in *Second International Workshop on Protocols for Fast Long-Distance Networks*, Argonne, Illinois USA, Feb. 16-17, 2004.

[13] C. Caini, R. Firrincieli, D. Lacamera, T. de Cola, M. Marchese, C. Marcondes, M. Y. Sanadidi, and M. Gerla, "TCPlive experiments on Real GEO satellite testbed," in *IEEE ISCC*, Aveiro, Portugal, Jul. 2007, pp. 523–529.

[14] C. Caini, R. Firrincieli, and D. Lacamera, "Comparative Performance Evaluation of TCP Variants on Satellite Environments," in *IEEE International Conference on Communications*, Dresden, Germany, 2009, pp. 5161–5165.

[15] H. Lundqvist and G. Karlsson, "TCP with end-to-end FEC," in *2004 International Zurich Seminar on Communications*, 2004, pp. 152–155.

[16] D. X. Wei and P. Cao, "NS-2 TCP-Linux: an NS-2 TCP implementation with congestion control algorithms from Linux," in *WNS2 '06: Proceedings from the 2006 Workshop on NS-2: the IP network simulator*. New York, NY, USA: ACM Press, 2006, p. 9.