A Survey On Congestion Control Protocols For High Speed Networks

K. Satyanarayan Reddy[†] and Lokanatha C. Reddy[††]

Research Scholar, Dept. of CS,
School of Science & Technology,
Dravidian University, Kuppam-517425, A.P., India.

Professor, Dept. of CS,
School of Science & Technology,
Dravidian University, Kuppam-517425, A.P., India

Summary: Conventional TCP suffers from poor performance on high bandwidth delay product links meant for supporting transmission rates of multi Gigabits per seconds (Gbps). This is largely due to TCP's congestion control algorithm, which can be slow in taking advantage of large amounts of available bandwidth. A number of high-speed variants have been proposed recently, the major ones being BIC TCP, CUBIC, FAST, High-Speed TCP, Layered TCP, Scalable TCP and XCP. In this paper an effort has been made to comparatively analyze the aforementioned protocols based on various parameters viz., Throughput, Fairness, Stability, Performance, Bandwidth Utilization and Responsiveness and study the limitations of these protocols meant for the High Speed Networks.

Keywords: Congestion Control, High-speed Networks, TCP variants.

1. Introduction

Congestion control is an important component of a transport protocol in a packet-switched shared network and most congestion control algorithms for the widely used connection oriented TCP are responsible for detecting congestion and reacting to overloads in the Internet and these algorithms have proved to be the key to the Internet's operational success.

However, as link capacity grows to support the multi Gigabits per seconds (Gbps) transmission rates and emergence of new Internet applications with high-bandwidth demand, TCP's performance proves to be unsatisfactory, especially on high-speed and long distance networks. The main reason for this is the conservative behavior of TCP in adjusting its congestion window governing the senders' transmission rates [27].

Recently, several solutions were proposed to address problems of TCP by changing the way in which TCP adapts its congestion window namely BIC TCP [1], [18], CUBIC [13], FAST [5], [8], [9], HSTCP [6], [16], [19], STCP [10], LTCP [2], [25] and XCP [4]. These proposed new protocols promise to improve TCP's performance on high-speed networks significantly and are usually known as the variants of TCP for high-speed networks. While the design, of variants of TCP for high-speed networks, has received a considerable amount of interest, but less attention has been paid to thorough evaluations and the mutual comparison of these protocols. For e.g., Internet measurement studies exhibited complex behaviors and characteristics of Internet traffic [20], [23], [24], [26], [27], [30], [31], [32] and [33]. Unfortunately, existing evaluation work [27] and [32] did not include/consider these behaviors in their testing environments. Since congestion control algorithms are very sensitive to environmental variables such as background traffic [26] and propagation delays, realistic performance evaluations of TCPs for High-speed Networks [27] require creating realistic network environments where these protocols are likely to be used.

There are many factors constituting a network environment. Most frequently used factors for creating a "realistic" testing environment include static end-to-end characteristics such as (1) bottleneck bandwidth, (2) round-trip times of protocol flows being observed, (3) the network topology over the path that protocol flows of interest travel through, and (4) queue size at the bottleneck link.

These factors are more or less static and do not change over the course of the experiment. Most of existing evaluation work are based on the considerations of (1) what protocol flows of interest dynamically (i.e., as the time-varies) experience in the middle of the network path, namely the dynamic statistical properties of "background traffic" over the intermediate links and (2) the impact of background traffic to the statistical properties of such traffic. Most of these dynamic characteristics cannot be measured at end points. Yet they can greatly influence the behaviors of the protocol flows that are being observed at the end points.

There are several reasons why background traffic is important in protocol testing. First, network environments without any randomness in packet arrivals and delays are highly susceptible to the phase effect [25], a commonly observed simulation artifact caused by extreme

Manuscript received July 5, 2008.
Manuscript revised July 20, 2008.

synchronization of the network flows on the end-to-end path. A good mix of background traffic with diverse arrival patterns and delays reduce the likelihood of the phase effect. Second, a high degree of statistical multiplexing is often assumed in protocol design. For instance, the inventors of HSTCP and STCP rely on statistical multiplexing for faster convergence.

So criticizing these protocols for slow or no convergence under environments without background traffic as done in [8] is unnecessary. As today's Internet contains a high degree of statistical multiplexing, testing with no or little background traffic does not capture the actual intended behaviors of protocols in production networks. Third, as much as background traffic can influence the behavior of the protocol flows being observed, the statistical behaviors of this "passing through" aggregate traffic can also be significantly altered by the nature of the protocols flows being tested.

Thus, measuring the statistical properties of the background traffic in the middle of the network enables to study the impact. This impact is important from the perspective of fairness or backward compatibility of the protocols.

This paper is organized as follows. In Section 2 we describe various congestion control Protocols meant for supporting high data transmission rates in High Speed Networks. In Section 3 we have taken up various congestion parameters & study the in-efficiencies of these Protocols, followed by comparative analysis of these protocols in Section 4 (based on the parameters defined in section 3). Section 5 concludes the paper.

2. Various Congestion Control Protocols for High speed Networks (BIC, CUBIC, FAST, High Speed TCP, Layered TCP, Scalable TCP and XCP)

Binary Increase Congestion Control (BIC): The BIC [1] and [18] congestion control algorithm uses two window size control policies called *additive increase* and *binary search increase*. When the congestion window is large, additive increase with a large increment ensures square RTT unfairness as well as good scalability. Under small congestion windows, binary search increase supports TCP friendliness.

Binary search increase: in this case the congestion control is viewed as a searching problem in which the system gives yes/no feedback through packet loss as to whether the current sending rate (or window) is larger than the network capacity. The starting points for this search are the current *minimum window size* W_{min} and *maximum window size* W_{max}. Usually, W_{max} is the window size just before the last fast recovery (i.e. where the last packet loss occurred), and W_{min} is the window size just after the fast recovery.

The algorithm repeatedly computes the midpoint between W_{max} and W_{min} sets the current window size to the midpoint; and checks for feedback, in the form of packet losses. Based on this feedback, the midpoint is taken as the new W_{max} if there is a packet loss, and as the new W_{min} if not.

The process repeats, until the difference between W_{max} and W_{min} falls below a preset threshold, called *the minimum increment* (S_{min}). This technique is called as *binary search increase,* and it allows bandwidth probing to be more aggressive initially when the difference from the current window size to the target window size is large, and it become less aggressive as the current window size gets closer to the target window size. A unique feature of the protocol is that its increase function is logarithmic; it reduces its increase rate, as the window size gets closer to the saturation point.

Whereas the other scalable protocols tend to increase their rates at the saturation point so that the increment at the saturation point is the maximum in the current epoch (defined to be a period between two consecutive loss events). Typically, the number of lost packets is proportional to the size of the last increment before the loss. Thus binary search increase can reduce packet loss. The main benefit of binary search is that it gives a concave response function, which goes well with that of additive increase described below.

Additive Increase: In order to ensure faster convergence and RTT-fairness, binary search increase is combined with an additive increase strategy. When the distance to the midpoint from the current minimum is too large, increasing the window size directly to that midpoint may add too much stress to the network.

When the distance from the current window size to the target in binary search increase is larger than a prescribed maximum step, called *the maximum increment* (S_{max}), the window size is increased by S_{max} until the distance becomes less than S_{max}, at which time window increases directly to the target. Thus, after a large window reduction, the strategy initially increases the window linearly, and then increases logarithmically. This combination of binary search increase and additive increase is called as *binary increase*.

Combined with a multiplicative decrease strategy, binary increase becomes close to pure additive increase under large windows. This is because a larger window results in

a larger reduction in multiplicative decrease, and therefore, a longer additive increase period. When the window size is small, it becomes close to pure binary search increase viz. a shorter additive increase period.

Advantages: The BIC TCP uses a Binary increase scheme to probe the available bandwidth efficiently [1], [18], [28]. While reaching the High throughput, BIC TCP does not increase the RTT fairness problem of standard TCP [1].

CUBIC: A New TCP-Friendly High-Speed TCP Variant

CUBIC [13] and [14] is an enhanced version of Binary Increase Congestion Control (BIC). It simplifies the BIC window control and improves its TCP-friendliness and RTT-fairness.

Although BIC achieves pretty good scalability, fairness, and stability during the current high speed environments, the BIC's growth function can still be too aggressive for TCP, especially under short RTT or low speed networks [13], [26] and [32]. Furthermore, the several different phases of window control add a lot of complexity in analyzing the protocol.

In CUBIC, the window growth function is a cubic function, whose shape is very similar to the growth function of BIC. CUBIC is designed to simplify and enhance the window control of BIC.

The growth function of CUBIC with the origin at W_{max} grows very fast upon a window reduction, 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.

The cubic function ensures the intra-protocol fairness among the competing flows of the same protocol. The function also offers a good RTT fairness property because the window growth rate is dominated by t, the elapsed time. This ensures linear RTT fairness since any competing flows with different RTT will have the same t after a synchronized packet loss (note that TCP and BIC offer square RTT fairness in terms of throughput ratio).

To enhance the fairness and stability further, window increment is clamped to be no more than S_{max} per second. This feature keeps the window to grow linearly when it is far away from W_{max}, making the growth function very much in line with BIC's as BIC increases the window additively when the window increment per RTT becomes larger than some constant.

The difference is that in CUBIC it is ensured that this linear increase of the window to be real-time dependent—when under short RTTs, the linear increment per RTT is smaller although stays constant in real time.

CUBIC is TCP friendly compared to HSTCP where it is TCP friendly when the loss rate is larger than 0.001, CUBIC has a larger area of the TCP friendly region. Further, when the RTT is very small, CUBIC is much more TCP friendly than HSTCP regardless of loss rates.

Advantages: CUBIC enhances the fairness properties of BIC while retaining its scalability and stability [13]. As the growth function is independent of RTT, its RTT fairness is guaranteed as different RTT flows will still grow their windows at the same rate [13].

Fast AQM Scalable TCP (FAST):

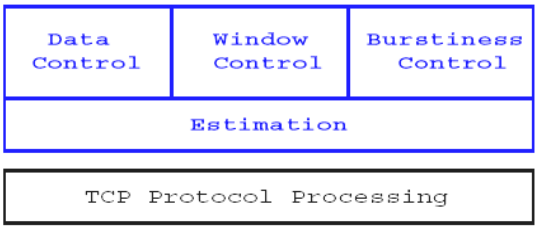

Fig. 1. FAST TCP Architecture.

In FAST [5], [8], [9] and [15], the congestion control mechanism of TCP is separated into four components as shown in Figure 1above.

These four components are functionally independent so that they can be designed separately and upgraded asynchronously.

The *data control* component determines *which* packets to transmit, *window control* determines *how many* packets to transmit, and *burstiness control* determines *when* to transmit these packets.

These decisions are made based on information provided by the *estimation* component. More specifically, the estimation component computes two pieces of feedback information for each data packet sent – a multibit queuing delay and a one-bit loss-or-no-loss indication – which are used by the other three components.

Data control selects the next packet to send from three pools of candidates: new packets, packets that are deemed to be lost (negatively acknowledged), and transmitted packets that are not yet acknowledged.

Window control regulates packet transmission at the RTT timescale, while burstiness control works at a smaller timescale. Burstiness control smoothes out transmission of packets in a fluid-like manner to track the available bandwidth.

Burstiness reduction limits the number of packets that can be sent when an ACK advances congestion window by a large amount.

Window pacing determines how to increase congestion window over the idle time of a connection to the target determined by the window control component. It reduces burstiness with a reasonable amount of scheduling overhead.

FAST [15] reacts to both queueing delay and packet loss. Under normal network conditions, FAST periodically updates the congestion window based on the average RTT and average queueing delay provided by the estimation component, baseRTT is the minimum RTT observed so far, and α is a positive protocol parameter that determines the total number of packets queued in routers in equilibrium along the flow's path.

In Fast, the network is modeled as a set of resources with finite capacities cl, e.g., transmission links, processing units, memory, etc., to which is referred to as "links" in the model.

The network is shared by a set of unicast flows, identified by their sources. A key departure of the protocol model from those

in the literature is that a source's *send rate* cannot exceed the *throughput* it receives.

Advantages: FAST TCP uses multi-bit information provided by queuing delay to compute the congestion window directly. This method is reported to have better responsiveness, stability, and fairness properties than the 1-bit flag loss indication used in standard, Scalable, and High Speed TCP [8], [23].

Highspeed TCP (HTCP): The HighSpeed TCP [7], [16], [17] and [19] for Large Congestion Windows was introduced by Sally Floyd [19] as a modification of TCP's congestion control mechanism for use with TCP connections with large congestion windows. It overcomes Standard TCP's difficulty of achieving a large congestion window in environments with very low packet drop rates. HighSpeed TCP proposes a small modification to TCP's increase and decrease parameters.

HighSpeed TCP tries to improve the loss recovery time of standard TCP by changing standard TCP's AIMD algorithm. This modified algorithm takes effect with higher congestion windows – i.e., when the congestion window is smaller than a given threshold, then it uses the Standard AIMD algorithm, else it uses HighSpeed AIMD algorithm [21].

In a steady-state environment, with a low packet loss rate p, Standard TCP's average congestion window is roughly 1.2/sqrt(p) segments [20]. This places a serious constraint on the congestion windows that can be achieved by TCP in realistic environments.

For example, for a Standard TCP connection with 1500-byte packets and a 100 ms round-trip time, achieving a steady-state throughput of 10 Gbps would require an average congestion window of 83,333 segments, and a packet drop rate of at most one congestion event every 5,000,000,000 packets (or equivalently, at most one congestion event every 1h:40m). If the round-trip time (RTT) is higher, the time between one congestion event and the next would need to be even greater. HighSpeed TCP does not modify TCP behavior in environments with mild to heavy congestion, and therefore does not introduce any new dangers of congestion collapse.

It is designed to have a different response in environments of very low congestion event rate, and to have the Standard TCP response in environments with packet loss rates of at most 10−2.

In environments with low packet loss rates (typically lower than 10−3), it is possible to ignore the more complex response functions that are required to model TCP performance in more congested environments with retransmit timeouts.

The Standard TCP increases its congestion window by one packet per window of data acknowledged, and halves it for every window of data containing a packet drop.

The number of round-trip times between congestion events required for a Standard TCP flow to achieve a high average throughput increases directly with the bandwidth available.

Advantages: HS-TCP is aggressive only in low-loss rate environments [30], [31]. In the single flow, which is run through the bottleneck link with varying router buffer sizes; the link utilization is very good [30], [31].

Layered TCP (LTCP): The Layered TCP [2], [25] scheme is a sender-side modification, to the congestion response function of TCP, for making it more scalable in High-Speed Networks. The congestion window response of the LTCP protocol is defined in two dimensions –

(a) At the macroscopic level, LTCP uses the concept of layering to quickly and efficiently probe for available bandwidth.

(b) At the microscopic level, it extends the existing AIMD algorithms of TCP 3 to determine the per-ACK behavior.

The aim of the LTCP protocol was to make the congestion response function scale in High speed networks under the following constraints:

1) The LTCP flows should be fair to each other (with similar RTTs).
2) The LTCP flows should be fair to TCP flows when the window is below a predefined threshold.

This threshold defines the regime in which LTCP is friendly to standard implementations of TCP.

The window scale option is used in High speed networks, to allow the receiver to advertise large window size.

This imposes a constraint on LTCP to maintain proportional fairness to TCP flows in slow networks.

In order to ensure that LTCP is fair to TCP, below the threshold, all new LTCP connections start with only one layer and behave in all respects the same as TCP. The congestion window response is modified only if the congestion window increases beyond the threshold.

When the congestion window of the LTCP flow grows beyond the LTCP window threshold, the increase behavior is modified to behave two dimensionally –

(a) If congestion is not observed over a period of time, then number of layers is increased.

(b) The per-ACK congestion window behavior of TCP is extended so that, when operating at higher layer a flow can increase its congestion window faster than when operating at a lower layer.

As in case of standard implementations of TCP, the LTCP protocol is ACK-clocked and the congestion window of an LTCP flow changes with each incoming ACK. However, an LTCP flow increases the congestion window more aggressively than the standard implementation of TCP depending on the layer at which it is operating.

Layers, on the other hand, are added if congestion is not observed over an extended period of time. To do this, a simple layering scheme is used. When the current congestion window exceeds the window corresponding to the last addition of a layer, a new layer is added.

The design of the decrease behavior is guided by a similar reasoning - for the two flows starting at different times to converge, the time taken by the larger flow to regain the bandwidth, which it gave up after a congestion event, should be larger than the time it takes the smaller flow to regain the bandwidth given up by it.

This framework provides a simple, yet scalable design for the congestion response function of TCP for the congestion avoidance phase in High speed networks. The congestion window response in slow-start is not modified, allowing the architecture to evolve with experimental slow-start algorithms.

At the end of slow-start the number of layers to operate at can easily be determined based on the window size.

Advantages: In LTCP, the macroscopic control uses the concept of layering to quickly and efficiently make use of the available bandwidth whereas microscopic control extends the existing AIMD algorithms of TCP to determine the per-ack behavior [25].

Scalable TCP (STCP): The main goal of Scalable TCP [10], [36], [37] and [38] is to improve the loss recovery time of the standard TCP. The basic idea of STCP is taken from the idea of HighSpeed TCP.

Packet loss recovery times for a traditional TCP connection (as well as HighSpeed TCP connection) are proportional to the connection's window size and RTT whereas a Scalable TCP connection's packet loss recovery times are proportional to connection's RTT only. The slow start phase of the original TCP algorithm is unmodified [21] and [22].

Like HighSpeed TCP, STCP has a threshold window size and the modified algorithm is used only when the size of the congestion window is above the threshold window size. The default threshold window size is 16 segments.

Scalable TCP is designed to be incrementally deployable and behaves identically to traditional TCP stacks when small windows are sufficient [10].

Advantages: The congestion window algorithm is scalable. The sending rate is doubled in a fixed amount of time for all rates [30], [31]. Scalable-TCP showed good performance even with small buffers. [30], [31]. When S-TCP does encounter loss, it is able to increase its window very quickly to take back the available bandwidth it had given up [31].

eXplicit Control Protocol (XCP): The XCP [3], [4], [24] and [35] is a feedback-based congestion control system that uses direct, explicit, router feedback to avoid congestion in the network. It is designed for both

scalability and generality. It performs especially well in very high delay-bandwidth product networks [3]. It uses router-assistance to accurately inform the sender of the congestion conditions found in the network.

XCP [35] divides the resource allocation function between two controllers: a congestion controller that ensures that flows use all available capacity, and a fairness controller that ensures that flows are allocated the capacity fairly. Most congestion control systems fail to make this division, much less to implement as two conceptually distinct systems. This division allows a clear exposition and implementation of two basic resource allocation functions in XCP. XCP sources send additional information about their current round-trip times and router-assigned throughput in each packet. XCP routers insert feedback into the packets that is interpreted by the sources.

In XCP, data packets carry a congestion header, filled in by the source that contains the sender's current congestion window size (H_cwnd field), the estimated RTT and a feedback field H_feedback. The H_ feedback field is the only one which could be modified at every hop (XCP router) based on the value of the two previous fields.

Basically, the H_feedback field which can take positive or negative values represents the amount by which the sender's congestion window size can be increased or decreased. On reception of data packets, the receiver copies the congestion header (which has been modified accordingly by the routers) into ACK packets sent back to the source.

It is not important that these ACK packets follow the same path than data packets since all the computations are done on the forward data path. On reception of ACK packets, the sender would update its congestion window size as follows: cwnd = max (cwnd + H_feedback; packetsize), with cwnd expressed in bytes. The core mechanism resides in XCP routers that use an efficiency controller (EC) and a fairness controller (FC) to update the value of the feedback field over the average RTT which is the control interval. The EC has the responsibility of maximizing link utilization while minimizing packet drop rate.

The EC basically assigns a feedback value proportional to the spare bandwidth S, deducted from monitoring the difference between the input traffic rate and the output link capacity, and to the persistent queue size Q (to avoid a feedback value of zero when input traffic is equal to output capacity).

Advantages: XCP [4] has very good convergence time to full utilization and fairness to other flows. Scalable for bandwidth and delay [22], [35].

3. Study the in-efficiencies of these Protocols

Researchers [3], [4], [6], [17], [21], [23], [26], [28], [30], [31] and [32] have shown that under varying Networking Environment and network traffic conditions the protocols meant for supporting high-speed data transfer, through multi gigabit links, exhibit some in-efficiencies. We are considering the following metrics for measuring the Network Performance of the protocols for High Speed Networks listed and defined above and study their inefficiencies

- Fairness [39], [41]
- Throughput [39], [41]
- Bandwidth utilization [39],
- Stability [41] is defined as the stability index of flow "*i*" is the sample standard deviation normalized by the average throughput: $S_i := \frac{1}{\bar{x}_i}\sqrt{\frac{1}{m-1}\sum_{k=1}^{m}(x_i(k)-\bar{x}_i)^2}$.
- The smaller the stability index, the less oscillation a source experiences. The stability index for interval $[0,m]$ is the average over the n active sources: $S := \frac{1}{n}\sum_{i=1}^{n} S_i$.
- **Responsiveness:** The responsiveness index [41] measures the speed of convergence when network equilibrium changes at $k = 1$, i.e., when flows join or depart.

  Let $x_i(k)$ be the running average by period $k \le m$:

  $$\bar{x}_i(k) := \frac{1}{k}\sum_{t=1}^{k} x_t(k)$$

  Then $x_i(m) = x_i$ is the average over the entire interval $[1,m]$.

3.1 Observed Inefficiencies

BIC-TCP: On an early packet drop, BIC-TCP may suffer in its initial ascent due to an incorrect estimate of the maximum link capacity [30]. In spite of its *Fast Convergence* strategy, BIC-TCP may not be able to share the link capacity fairly with other competing flows [30], [42], [43], [45].

CUBIC: CUBIC is found to be slow in increasing its *cwnd* to fully occupy the link. It suffers a lot, more than BIC TCP, on an early packet drop in its first ascent because of the incorrect estimation of maximum link capacity [30], [42].

FAST: FAST suffers unfairness and instability in small buffer or long delay networks regardless of background traffic types [26]. The major deficiency of FAST TCP is that it has a control parameter "α" that needs to be set up manually and its performance can be affected by reverse traffic [26]. FAST exhibits poor performance whenever the router queue size is smaller than "α" [30], [42], [43].

HSTCP: It has been found to be not so friendly with common TCP flows [31]. HSTCP trades stability for fairness; that is, while its fairness is good independent of background traffic types, larger variance in the flow sizes and RTTs of background flows causes the protocol to induce a higher degree of global loss synchronization among competing flows, lowering link utilization and stability [26]. HSTCP converges very slowly [15], [29]. HSTCP can increase the RTT bias of TCP [1], [42], [43], [45].

LTCP: Research is underway to study its inefficiencies.

STCP: The Scalable TCP may not converge to fairness equilibrium [11], [29]. Scalable TCP can increase the RTT bias of TCP [1].

XCP: The high dependence of XCP [4] on the returned ACK packets for maintaining a coherent view of the network conditions. This dependence has high impact on the XCP performances in case of ACK losses on the reverse path, making XCP very unstable if used as it is on dynamic, very high-speed networks. XCP needs to maintain RTT per connection, Needs router participation, deployment might prove to be difficult, unfair toward connections with longer RTT, malicious Sender can falsify the header and the feedback calculation may go erroneous [22],[42],[43],[44] and [45].

3.2 Proposed Solution

We have proposed a solution in [39], [40] in the form of a flowchart and a model respectively for detecting and controlling the congestion also to improve on the inefficiencies observed as in 3.1. The simulations for the proposed solution will be carried out using Network Simulator NS2 version 2.31.

4. Comparative Analysis (based on parameters viz. Throughput, Fairness, Stability, Bandwidth Utilization and Queuing Delay)

The Comparative Analysis table has been presented in **Appendix B** at the end of this Paper.

5. Conclusion

In general, most of the high-speed protocols are not fair when competing with other high-speed protocols [31]. Intra-protocol fairness suffers when the flows are started at different times, due to slower convergence times [31]. The performance of S-TCP and FAST do not depend upon the competing flow, but rather are dependent only upon their own operation [31]. S-TCP is too aggressive in obtaining bandwidth, even when competing with another S-TCP flow [31].

Acknowledgements

The authors thank the authorities of the Dravidian University, Kuppam, India, who provided opportunities and resources for carrying out this research and for the liberal grants extended for research activities in the Dept. of CS at the University. The first author further thanks the Management and also the Principal of the New Horizon College of Engineering, Panathur Post, Bangalore, India for their support and encouragement extended to him to pursue research in the chosen field of study.

References

[1] Lisong Xu, Khaled Harfoush, and Injong Rhee "Binary Increase Congestion Control (BIC) for Fast Long-Distance Networks", IEEE, INFOCOM, 2004.

[2] Sumitha Bhandarkar, Saurabh Jain and A. L. Narasimha Reddy "LTCP: A Layered Congestion Control Protocol for Highspeed Networks" ACM SIGCOMM Computer Communication Review, Volume 36 , Issue 1, pp.: 41 – 50, 2006. http://portal.acm.org/citation.cfm?id=1111322.1111332

[3] D. Katabi, M. Handley, C. Rohrs. "Congestion Control for High Bandwidth-Delay Product Networks". ACM SIGCOMM 2002.

[4] B.Even, Y.Li, D.J.Leith "Evaluating the Performance of TCP Stacks for High-Speed Networks", Hamilton Institute, Ireland, 2006. www.hamilton.ie/net/pfldnet2006.pdf

[5] T. Cui and L. Andrew. "FAST TCP simulator module for ns-2", version 1.1, 2004. Available at http://www.cubinlab.ee.mu.oz.au/ns2fasttcp.

[6] Junsoo Lee, Stephan Bohacek, Jo˜ao P. Hespanha, Katia Obraczka "A Study of TCP Fairness in High-Speed Networks", University of Southern California, white paper 2007. http://whitepapers.silicon.com/0,39024759,60300997p,00.htm.

[7] S. Floyd, S. Ratnasamy, and S. Shenker "Modifying TCP's congestion control for high speeds", May 2002.

[8] C. Jin, D. X. Wei, and S. H. Low, Hegde, S. FAST TCP: Motivation, Architecture, Algorithms, Performance". IEEE/ACM Transactions on Networking, Volume 14, Issue 6, pp. :1246 – 1259, Dec. 2006.

[9] C. Jin, D. X. 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". *IEEE Network*, 19(1) pp. 4–11, January/February 2005.

[10] T. Kelly. "Scalable TCP: Improving performance in highspeed wide area networks". In *Proceedings of PFLDnet*, Geneva, Switzerland, Feb. 2003.

[11] D. Leigh, R. Shorten, et al. H-TCP ns-2 implementation, 2005.

[12] ns the Network Simulator.
http://www.isi.edu/nsnam/ns/.

[13] I. Rhee and L. Xu. "CUBIC: A new TCP-friendly high-speed TCP variant". In *Proceedings of PFLDnet*, Lyon, France, Feb. 2005.

[14] I. Rhee and L. Xu. "Simulation code and scripts for CUBIC", 2005. http://www.csc.ncsu.edu/faculty/rhee/export/bitcp/cubic-script/script.htm.

[15] C. Jin, D. Wei, S. H. Low, G. Buhrmaster, J. Bunn, D. H. Choe, R. L. A. Cottrell, J. C. Doyle, H. Newman, F. Paganini, S. Ravot, and S. Singh, "FAST Kernel: Background theory and experimental results," in *First International Workshop on Protocols for Fast Long-Distance Networks*, February 2003..

[16] E. Souza and D. A. Agarwal. "A HighSpeed TCP study: Characteristics and deployment issues". Technical Report LBNL-53215, 2003.

[17] K. Tokuda, G. Hasegawa, and M. Murata. "Performance analysis of HighSpeed TCP and its improvements for high throughput and fairness against TCP Reno connections". In *High-Speed Networking Workshop (HSN)*, 2003.

[18] L. Xu, K. Harfoush, and I. Rhee. "Binary increase congestion control for fast, long distance networks". In *Proceedings of IEEE INFOCOM*, Hong Kong, Mar. 2004.

[19] S. Floyd. "Highspeed TCP for large congestion window", Internet Draft draft-floyd-tcp-highspeed-01.txt, February 2003.

[20] S. Floyd and K. Fall. "Promoting the use of end-to-end congestion control in the Internet". IEEE/ACM Transactions on Networking, 7(4):458–472, August 1999.

[21] Eric He, Michael Welzl and Pascale Vicat-Blanc Primet "A Survey of Transport Protocols other than Standard TCP" Data Transport Research Group, GFD-I.055, 2005.

[22]http ://www.cs.ucla.edu/NRL/hpi/tcpw/tcpw_papers/part1-congestion-control.pdf

[23] Eric He, Rajkumar Kettimuthu, Sanjay Hegde, Michael Welzl, Jason Leigh, Chaoyue Xiong and Pascale Vicat-Blanc Primet, "Survey of Protocols and Mechanisms for Enhanced Transport over Long Fat Pipes", Data Transport Research Group,2003/2004. http://www.globus.org/alliance/publications/papers/Survey.pdf

[24] D. M. Lopez-Pacheco and C. Pham "Robust Transport Protocol for Dynamic High-Speed Networks: enhancing the XCP approach" in Proc. of IEEE MICCICON, 2005.

[25] Sumitha Bhandarkar, Saurabh Jain and A. L. Narasimha Reddy, "Improving TCP Performance in High Bandwidth High RTT Links Using Layered Congestion Control", International Workshop on Protocols for Fast Long-Distance Networks, February 2005. http://whitepapers.silicon.com/0,39024759,60303395p,00.htm

[26] Sangtae Ha, Long Le, Injong Rhee and Lisong Xu "Impact of background traffic on performance of high-speed TCP variant protocols" S. Ha et al., Computer Networks 51, pp. 1748–1762, 2007.

[27] Sangtae Ha, Yusung Kim, Long Le, Injong Rhee, and Lisong Xu* "A step toward realistic evaluation of high-speed TCP protocols" , 2006. http://www.csc.ncsu.edu/faculty/rhee/export/bitcp/asteppaper. htm.

[28] Yunhong Gu, Xinwei Hong, and Robert L. Grossman, "Experiences in Design and Implementation of a High Performance Transport Protocol ", Supercomputing, Proceedings of the ACM/IEEE SC2004 Conference, pp. 22, 2004.

[29] D. J. Leith and R. Shorten. "H-TCP Protocol for High-Speed Long Distance Networks". PFLDnet '04, Feb. 2004. http://dsd.lbl.gov/DIDC/PFLDnet2004/talks/Leith-slides.pdf.

[30] Pankaj Sharma, "Performance Analysis of High-Speed Transport Control Protocols", Master of Science (Computer Science) Thesis, Clemson University, August 2006. http://www.cs.odu.edu/~mweigle/papers/sharma-thesis06.pdf

[31] Michele C. Weigle, Pankaj Sharma, and Jesse R. Freeman IV, "Performance of Competing High-Speed TCP Flows" Networking 2006. Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems, pp. 476-487,2006. http://www.cs.odu.edu/~mweigle/papers/networking06.pdf

[32] Yee-Ting Li, Douglas Leith and Robert N. Shorten, "Experimental Evaluation of TCP Protocols for High-Speed Networks", IEEE/ACM Transactions on Networking, pp.: 1109 – 1122, Oct. 2007. www.ieeexplore.ieee.org/iel5/90/4346537/04346548.pdf

[33] Sally Floyd, Vern Paxson, "Difficulties in simulating the Internet", IEEE/ACM Transactions on Networking 9 (4), 2001.

[34] E. Altman, K. Avrachenkov, C. Barakat, A.A. Kherani, B.J. Prabhu "Analysis of MIMD congestion control algorithm for High Speed Networks" Computer Networks: The International Journal of Computer and Telecommunications Networking Volume 48 , Issue 6, pp.: 972 – 989, 2005.

[35] For XCP:

http://www.isi.edu/nsnam/ns/doc/node238.html

[36] R. El Khoury and E. Altman "Analysis of Scalable TCP" In Proceedings of the International Working Conference on Performance Modelling and Evaluation of Heterogeneous Networks (HET-NETS), July 2004. http://citeseer.ist.psu.edu/719259.html

[37] E. Altman, K. E. Avrachenkov, B. J. Prabhu "Fairness in MIMD Congestion Control Algorithms", In Proceedings of the IEEE INFOCOM, 2005. http://www.inria.fr/rrrt/rr5312.html

[38] E. Altman, K. Avrachenkov, C. Barakat, A.A. Kherani, B.J. Prabhu "Analysis of MIMD Congestion Control Algorithm for High Speed Networks", Elsevier, Computer Networks Volume 48, Issue 6, pp.: 972-989, 2005. linkinghub.elsevier.com/retrieve/pii/S1389128605000022

[39] K. Satyanarayan Reddy, C. Lokanatha Reddy "A Survey on Congestion Control Mechanisms in High Speed Networks" has been published in the *International Journal of Computer Science and Network Security* (IJCSNS) Vol. 8 No. 1, pp. 187 – 195, January 2008. http://paper.ijcsns.org/07_book/200801/20080126.pdf

[40] K. Satyanarayan Reddy, C. Lokanatha Reddy "An Efficient Explicit Congestion Reduction in High Traffic High Speed Networks through Automated Rate Controlling" abstract appeared in the International Conference ICSTAORIT – 2006 – XXVI ISPS CONFERENCE Page no. 125 with Paper Id : IT-45 held at Tirupati, India during the period 7th January 2007 to 9th January 2007.

[41] Cheng Jin David X. Wei Steven H. Low, "FAST TCP: Motivation, Architecture, Algorithms, Performance" IEEE Infocom 2004.

http://www.ieee-infocom.org/2004/Papers/52_2.PDF

[42] adrien Bullot, R. Les Cottrell and Richard Hughes-Jones, "Evaluation of Advanced TCP Stacks on Fast Long-Distance Production Networks" Journal of Grid Computing, Volume 1, Number 4, pp. 345-359, December 2003. http://www.springerlink.com/content/v7463765348771g6/

[43] Saverio Mascolo, Francesco Vacirca "Issues in Performance Evaluation of New TCP Stacks in High Speed Networks" white paper on Bandwidth Issues, znet, 2005. http://www.zdnet.com.au/whitepaper/0,2000063328,22437228p-16001408q,00.htm

[44] D. M. Lopez-Pacheco, C. Pham, "Robust Transport Protocol for Dynamic High-Speed Networks: enhancing the XCP approach" ICON, 2005.

http://web.univ-pau.fr/~cpham/Paper/icon05.pdf

[45] R. Les Cottrell, Saad Ansari, Parakram Khandpur, Ruchi Gupta, Richard Hughes-Jones, Michael Chen, Larry McIntosh, Frank Leers, "Characterization and Evaluation of TCP and UDP-based Transport on Real Networks", Presented at the Third International Workshop on Protocols for Fast Long-

Authors

†**K. Satyanarayan Reddy** received his M.Sc.(Mathematics) & M.Phil. (Mathematics) **D**egrees from Nagpur University, Maharashtra State, and M. Tech. (Computer Applications) from **I**ndian **S**chool of **M**ines, **Dhanbad**, Jharkhand in 1987, 1988 and 2000 respectively. He is currently associated with Information Science & Engineering department of New Horizon College of Engineering, Bangalore, Karnataka State, India. He is a Research Scholar in the Dept. of Computer Science at Dravidian University, Kuppam, AP, India and is pursuing his Ph.D. Degree in Computer Science. His current areas of research are Congestion Control in High Speed Networks and Data Communications.

††**Lokanatha C. Reddy** earned M.Sc.(Maths) from Indian Institute of Technology, New Delhi; M.Tech.(CS) with Honours from Indian Statistical Institute, Kolkata; and Ph.D.(CS) from Sri Krishnadevaraya University, Anantapur. Earlier worked at KSRM College of Engineering, Kadapa (1982-87); Indian Space Research Organization (ISAC) at Bangalore (1987-90). He is the Head of the Computer Centre (on leave) at the Sri Krishnadevaraya University, Anantapur (since 1991); and a Professor of Computer Science and Dean of the School of Science & Technology at the Dravidian University, Kuppam (since 2005). His active research interests include Real-time Computation, Distributed Computation, Device Drivers, Geometric Designs and Shapes, Digital Image Processing, Pattern Recognition and Networks.

APPENDIX - B

Comparative Analysis of various High Speed Network Protocols

Metrics ▼ ▼ ▼	High Speed Network Protocols ►►►						
	BIC [1], [18]	**CUBIC** [13], [14]	**FAST** [5],[8],[9], [15], [41]	**HS TCP** [16], [17], [19], [21], [27], [30], [31]	**LTCP** [2], [25], [27]	**STCP** [10], [34], [36], [37], [38]	**XCP** [3], [24], [35]
Fairness	BIC exhibits good fairness in presence of Background traffic [27].	The Cubic function ensures intra-protocol fairness among the competing flows of the same protocol [21].	When compared to all other TCP variant protocols for highspeed networks FAST exhibits unfairness in presence of Background traffic [27].	It is very gentle, its unfriendliness increases with packet drop rates and with window size greater than optimal [42]. It exhibits very good fairness in presence of Background traffic [27].	Fairness Index for LTCP is high across different RTTs and different number of flows [2]., [25], [45].	It is very aggressive for short distance links and becomes gentle for Long distance links [21], with slight increase in buffer size it provides very low fairness [43].	It exhibits good fairness mechanism [21].
Bandwidth Utilization	BIC exhibits good Bandwidth utilization both in case of single flow and two flows [27].	It exhibits good Bandwidth utilization especially for small Bandwidth Delay Product networks. Even in presence of Background traffic it performs extremely well [27].	Experimentally it has been established that in presence of Background traffic the bandwidth utilization of FAST reduces considerably. It is due to noise in Round Trip Time (RTT) estimation [27] .	The Bandwidth Utilization improves even during the Bursty Traffic. But in presence of Background traffic the bandwidth utilization of HTCP is lower than the other protocols under consideration.	The window adaptation of LTCP is much more efficient in utilizing the link bandwidth in High-speed Networks [2], [45].	The Bandwidth utilization of STCP (and that of HTCP & FAST) is significantly lower than that of other protocols under consideration [27].	XCP exhibits high bandwidth utilization independent of the delay, by adjusting its aggressiveness according to Round Trip Delay [3].
Through-put	It achieves good throughput when the buffer size is less than the Bandwidth Delay Product also in presence of background traffic [27].	It achieves good throughput when the buffer size is less than the Bandwidth Delay Product.	Throughput remains optimal but in presence of Reverse /Background traffic the throughput gets affected [26].	In case of Bursty Traffic the throughput improves but Fairness becomes an issue.	LTCP offers an improvement of a factor of about 8 for the achievable throughput compared to TCP [2].	It exhibits significant improvement in the throughput.in presence of background traffic [27].	It exhibits good throughput for different flows having 'substantially different RTT's [3].
Stability	For small window size the Stability is good [42].	It provides good stability and scalability	For small window size the Stability is good [42]. Fast exhibits good stability when different RTT high-speed flows are competing [27].	For small window size the Stability is good [42]. HTCP becomes unstable in long RTT networks, also when different RTT high-speed flows are competing [27].	The LTCP exhibits stability for short RTT [2], [45].	In absence of UDP cross traffic it exhibits best stability but in presence of UDP traffic it becomes Unstable [42].	It is found to be stable for any link capacity, feedback delay or no. of sources [21]. It becomes unstable due to ACK loss on reverse path [44].
Responsive-ness	As the propagation delay increases the cwnd's converge increasingly slowly, not reaching Fairness for long time [32].	Cubic can exhibit slow convergence following network disturbances such as the start-up of new flows [29].	It exhibits strange behavior of convergence under different networks and traffic conditions [32].	It has been shown that the flows do converge to fairness, but that the convergence time can be long more so when the propagation delay increases [32].	Results indicate that LTCP has promising convergence properties [25].	It has been shown that the cwnds either do not converge to fairness or converge very slowly [32].	It separates the fairness controller from the efficiency controller to have better control over the Responsiveness of the traffic.