ISSN : 2231-0711

Performance Analysis of TCP Congestion Control Algorithms

Priyanka K. Shinde, Prof. Nitin R. Chopde,

Department Of Computer Science and Engineering,
G. H. Raisoni College of Engineering

Abstract— **The demand for fast transfer of larger volume of data, and the deployment of the network infrastructures is ever increasing. However, TCP is the dominant transport protocol of today, does not meet this demand because it favors reliability over timeliness and fails to fully utilize the network capacity due to its limitations of its conservative congestion control algorithm. The slow response of TCP in fast long distance networks leaves sizeable unused bandwidth in such networks. A large variety of TCP variants have been proposed to improve the connection's throughput by adopting more aggressive congestion control algorithms. Some of the flavors of TCP congestion control are loss-based, high-speed TCP congestion control algorithms that uses packet losses as an indication of congestion; delay-based TCP congestion control that emphasizes packet delay rather than packet loss as a signal to determine the rate at which to send packets. Some efforts combine the features of loss-based and delay-based algorithms to achieve fair bandwidth allocation and fairness among flows. A comparative analysis between different flavors of TCP congestion control namely Standard TCP congestion control (TCP Reno), loss-based TCP congestion control (HighSpeed TCP, Scalable TCP, CUBIC TCP), delay-based TCP congestion control (TCP Vegas) and mixed loss-delay based TCP congestion control (Compound TCP) is presented here in the paper.**

Key words— **Congestion control, High-speed networks, TCP.**

1. Introduction

Moving bulk data quickly over high-speed data network is a requirement of many applications. These applications require high-bandwidth links between network nodes. To maintain the stability of Internet all applications should be subjected to congestion control. TCP is well-developed, extensively used and widely available Internet transport protocol. TCP is fast, efficient and window size, thus it hurts the data rate. Standard TCP contains the congestion window that can be achieved in realistic environments. In the past few years, we notice that a surge of TCP variants address the under utilization problem responsive to network congestion conditions but TCP's AIMD congestion back-off algorithm [1] is too abrupt in decreasing the most notably due to the slow growth of TCP congestion window that makes TCP unfavorable for high BDP networks. In this paper we describe Standard TCP congestion control algorithm. The related paper is organized as related work including TCP modifications and new protocols are reviewed in section 1. Three prominent window-based high-speed TCP congestion control algorithms that use packet-loss as an implicit indication of congestion are described in section 2. Compound TCP and Zeta TCP approach is described in section 3. Further Classification of congestion control mechanism according to their properties is described in paper in section 4. Propose work is described in section 5. Finally this work is concluded in section 6.

I. Background And Related Work

The standard TCP congestion control algorithm which we refer to as TCP Reno[1] was developed in 1988. Further there is several enhancements in TCP Reno. Few modifications addressing the conservative approach of TCP to update its congestion window under congestion condition are:

i. **Loss-based TCP congestion control**: HSTCP, BIC-TCP, STCP, CUBIC-TCP, HTCP etc.
ii. **Delay-based congestion control**: TCP-Vegas, Fast-TCP , TCP-LP etc.
iii. **Learning- based TCP congestion control**: Compound TCP, Zeta TCP etc.

Most of these protocols deal with modifying the window growth function of TCP in a more scalable fashion. Tomoya[2] proposed a TCP-friendly congestion control that realizes efficient data transmission in highspeed networks, fairness with TCP Reno and fair bandwidth allocation among flows with different RTTs.]

a) *TCP Reno*

TCP Reno[2] implements the TCP's AIMD mechanism of increasing the congestion window W by one segment per round-trip time for each received ACK and halving the congestion window for each loss event per round-trip time. TCP Reno controls the congestion window as follows:

Increase:

$$W= W + 1 \div W \qquad (1)$$

Decrease:

$$W= W- 1 \div W \qquad (2)$$

When the link bandwidth does not change, TCP Reno periodically repeats the window increase and decrease.TCP Reno's congestion window in terms of packet loss rate (*p*) is defined as:

$$W_{reno} = \frac{1.22}{P^{0.5}} \qquad (3)$$

As shown above, TCP Reno places a serious constraint on the congestion window that can be achieved by TCP in realistic environments. TCP requires extremely small packet loss rate to sustain a large window which is not possible in real life networks.

b) *High-Speed TCP*

Although TCP performs very well in low to middle speed networks[11] (Kbps to several Mbps), it has very poor performance in high (tens of Mbps to Gbps) to very high (Gbps to Tbps) speed networks, as TCP is very inefficient in utilizing the high-speed network bandwidth. HighSpeed TCP (HSTCP) is a modification toTCP's congestion control mechanism for use with TCP connections with large congestion windows. HighSpeed TCP's modified response function only takes effect with higher congestion windows, it does not modify TCP behavior in environments with heavy congestion, and therefore does not introduce any new dangers of congestion collapse. HSTCP uses three parameters, WL, WH, and PH. To ensure TCP compatibility, HSTCP uses the same response function as TCP Reno when the current congestion window is WL at most, and uses the HSTCP response function when the current congestion window is greater than WL.

HSTCP response function is computed as follows:

$$W_{\text{highspeed}} = \frac{0.12}{P^{0.835}} \qquad (4)$$

It is clear from equation that HSTCP is more aggressive than TCP Reno and a HighSpeed TCP connection would receive ten times the bandwidth of a standard TCP in an environment with packet drop rate of10^{-6}, which is unfair.

c) *Scalable TCP*

Scalable TCP[13] is designed to be incrementally deployable and behaves identically to traditional TCP stacks when small windows are sufficient. Scalable TCP (STCP) and HighSpeed TCP were originally designed for high-speed backbone links, and they appear to be the major candidates for replacing in the next generation Internet the current congestion control mechanism implemented by standard TCP. STCP is a simple sender side modification to TCP congestion control, and it employs Multiplicative Increase Multiplicative Decrease (MIMD) technique. Using Scalable TCP, better utilization of a network link with the high bandwidth delay product can be achieved. If STCP is mixed with regular TCP then STCP dominates the bandwidth for sufficiently large bandwidth-delay product region. This shows unfriendliness towards standard TCP.

d) *Cubic TCP*

Cubic TCP[16] is an enhanced version of Binary Increase Congestion Control shortly BIC. It simplifies the BIC window control function and improves its TCP-friendliness and RTT fairness as BIC's growth function is too aggressive for TCP especially under short RTT or low speed networks. As the name of the protocol represents, the window growth function of CUBIC is a cubic function in terms of the elapsed time since the last loss event, whose shape is very similar to the growth function of BIC. CUBIC function provides good scalability and stability. The protocol keeps the window growth rate independent of RTT, which keeps the protocol TCP friendly under short and long RTTs. The congestion epoch period of CUBIC is determined by the packet loss rate alone. As TCP's throughput is defined by the packet loss rate as well as RTT, the throughput of CUBIC is defined only by the packet loss rate. Thus, when the loss rate is high and/or RTT is short, CUBIC can operate in a TCP mode.

ii . Delay-Based Congestion Control

Delay-based TCP congestion control algorithms like TCP Vegas attempt to utilize the congestion information Contained in packet round-trip time (RTT) samples**.**

a) *TCP Vegas*

TCP Vegas is a TCP congestion control algorithm that emphasizes packet delay, rather than packet loss, as a signal to determine the rate at which to send packets. TCP Vegas detects congestion based on increasing Round Trip Time (RTT) values of the packets in the connection unlike TCP Reno which detect congestion only after it has actually happened via packet drops. The algorithm depends heavily on accurate calculation of the Base RTT value. Base RTT is set to be the minimum of all measured RTTs; it is commonly the RTT of the first segment sent by the connection.

iii .Learning Based Congestion Control

Loss-based high speed algorithms are aggressive to satisfy bandwidth requirement but this aggressiveness causes TCP unfairness and RTT unfairness. Delay-based approaches provide RTT fairness but it is difficult to meet TCP fairness. Thus there is another approach i.e. learning based approaches that address the problems in the two approaches.

a) *Compound TCP*

Compound TCP integrates a scalable delay-based component into the standard TCP congestion avoidance algorithm. This scalable delay-based component has a fast window increase function when the network is under-utilised and reduces the sending rate when a congestion event is sensed. To implement Compound TCP maintains the following state variables; cwnd (congestion window), dwnd (delay window), awnd
(Receiver advertised window).

b) *ZETA TCP*

The accurate and rapid detection of packet loss capabilities of Zeta TCP[21] is especially valuable with the explosive growth of mobile networks. The flaky last mile fading channel to the mobile devices creates frequent bulk packet loss. Such loss triggers standard and delay-based TCP to jam more packets into the network, which actually causes more problems. ZetaTCP, analyzes the situation intelligently and allows rapid and efficient recovery from packet loss and enables a smoother transmission and maximum throughput.

I. Classification

The following is one possible classification according to the following properties:

1. The type and amount of feedback received from the network: Loss (L); delay (D); single-bit (S) or multi-bit (M) explicit signals
2. Incremental deploy ability on the current Internet: Sender needs modification (S); receiver needs modification (R); routers/gateways need modification (G)

3. The aspect of performance it aims to improve: high bandwidth delay product networks (B); lossy links (L); fairness (F); advantage to short flows (S); variable-rate links (V); speed of covergenec(C)
4. The fairness criterion it uses: max-min (M), proportional (P), "minimum potential delay" (D), Other (O)

Some well-known congestion avoidance mechanisms are classified[19][20] by this scheme are as follows:

TABLE I
CLASSIFICATION OF CONGESTION AVOIDANCE MECHANISM

Variant	Feedback	Changes	Benefits	Fairness
(New)Reno	L	-	-	D
Vegas	D	S	Less loss	P
High Speed	L	S	B	O
BIC	L	S	B	O
CUBIC	L	S	B	O
H-TCP	L	S	B	O
FAST	D	S	B	P
Compound TCP	L/D	S	B	P
Westwood	L/D	S	L	O
Jersey	L/D	S	L	O
CLAMP	M	G/R	V	M
TFRC	L	S/R	No Retransmission	D
XCP	M	S/G/R	BLFC	M
VCP	M(2 bits)	S/G/R	BLF	P

II. PROPOSE WORK

TCP is relied upon to carry more than 90% of all Internet traffic. It has become essential to stay with today's demands of impatient end-users to effectively delivering latency sensitive web applications. In order to keep up with these demands, various TCP optimization approaches have been developed now. Applying optimization techniques to standard, loss-based TCP provides some improvement. But as long as network speed is governed by loss, a high data-rate and stable throughput will be impossible to achieve. A more modern delay-based approach can provide some fundamental improvement to dealing with network latency, but for the most part in it remains a static. An advanced learning-based approach is needed that is capable of observing session characteristics on-the-fly in order to apply it intelligentelly, session-specific transport optimizations. Advanced learning-based TCP is now use by hundreds of companies and millions of users to accelerate the latency-sensitive applications. It is able to provide these capabilities in a completely transparent manner.

III. CONCLUSION

TCP Reno was commonly implemented algorithm. Most others are competing proposals which still need some evaluation. Starting with 2.6.8 the Linux kernel switched the default implementation was again changed to CUBIC in the 2.6.19 version. FreeBSD uses New Reno as the default algorithm. However, it supports a number of other choices. When the per-flow product of bandwidth and latency increases, regardless of the queuing scheme, TCP prone to instability and becomes inefficient. This becomes increasingly important as the Internet evolves to incorporate very high-bandwidth optical links.TCP Interactive allows applications to subscribe to TCP events and it respond accordingly enabling various functional extensions to TCP from outside TCP layer. Most of the TCP congestion schemes work internally. Zeta-TCP detects the congestions from both the loss rate measures and latency, and applies different CWND back off strategies based on the likelihood of the congestions to maximize its goodput. It also has a couple of other improvements to accurately detect the packet losses, accelerate/control the inbound (download) traffic and avoiding RTO retransmission.

REFERENCES

[1]. V. Jacobson, “Congestion avoidance and control,” the ACM SIGCOMM'88.
[2].Tomoya Hatano, Hiroshi Shigeno and Ken-ichi Okada,“TCP friendly congestion control for highSpeed Network” IEEE, 2007.
[3]. David X. Wei, Cheng Jin, Steven H. Low, and Sanjay Hedge, “Fast TCP: Motivation, Architecture, Algorithms, Performance”, IEEE/ACM transactions on networking, 2006.
[4]. W. Stevens, “TCP Slow Start, Congestion Avoidance,Fast Retransmit, and Fast Recovery Algorithms”, RFC2001, Jan. 1997.
[5]. M. Allman, V. Paxson, and W. Stevens, “TCP Congestion Control”, RFC 2581, Apr. 1999.
[6]. S. Floyd and T. Henderson, “The NewReno modification to TCP's fast recovery algorithm”, RFC 2582, Apr. 1999.
[7]. V. Jacobson, R. Braden, and D. Borman, “TCP extensions for high performance,” RFC 1323, May 1992
[8]. M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, “TCP selective acknowledgment options,” RFC 2018, Oct. 1996.
[9]. Jon Postel, “Transmission Control Protocol,” RFC 793 September 1981,
[10]. Allman, M., Balakrishnan, H. and S. Floyd, “Enhancing TCP's Loss Recovery using Limited Transmit:, RFC 3042, January 2001.
[11]. S. Floyd: “HighSpeed TCP for Large Congestion Windows”, RFC 3649, December 2003
[12]. Lisong Xu, Khaled Harfoush, and Injong Rhee: “Binary Increase Congestion Control for Fast, Long DistancE Networks”, 2003.
[13]. Tom Kelly: “Scalable TCP: Improving performance in high-speed wide area networks”; ACM SIGCOMM IJCSA Issue 1, Volume 2, 200837Computer Communication review, April 2003.
[14]. Hamed Vahdet Nejad, Mohammad Hossien Yaghmaee, Hamid Tabatabaee, “Fuzzy TCP: Optimizing TCP Congestion Control”, IEEE, 2006
[15]. M. Allman, S. Floyd, C.Partridge, “Increasing TCP's initial window”, September 1998
[16]. Injong Rhee, and Lisong Xu: “CUBIC: A New TCPFriendly High-Speed TCP Variant”, Sangtae Ha, Injong Rhee, Lisong Xu.
[17]. D. Leith, and R. Shorten: “H-TCP: TCP Congestion Control for High Bandwidth-Delay Product Paths”,June20, 2005
[18].Lefteris Mamata, Tobias Harks, and Vassils Tsaoussidis: “Approaches to congestion control in packet networks”,JIE VOL, 1 NO. 1, January 2007.
[19]. Peng Yang, Member, IEEE, Juan Shao, Wen Luo, Lisong Xu, Member, IEEE, Jitender Deogun, Member, IEEE, and Ying Lu, Member, IEEE, ”TCP Congestion Avoidance Algorithm Identification”,IEEE, 2013.
[20]. Jingyuan Wang, Jiangtao Wen, Fellow, IEEE, Yuxing Han, Jun Zhang, Chao Li, and Zhang Xiong “TCP-FIT:An Improved TCP Congestion Avoidance Algorithm for Heterogeneous Networks” IEEE, 2011.