ISSN (Online): 2230-7893
www.IJCEM.org

Experimental Study of TCP Congestion Control Algorithms

Kulvinder Singh

Asst. Professor, Department of Computer Science & Engineering,
Vaish College of Engineering,
Rohtak, Haryana, India
kuvinderbhuna@gmail.com

Abstract

The ongoing TCP congestion algorithm doesn't acquire full advantage of high delay, large-scale bandwidth environments. The forth chapter involves assessing alternative transmission control protocol congestion algorithms and comparing them with the presently employed congestion algorithm. The destination was to discover if an alternative algorithm could allow for broader throughput with minimum impact on active network traffic. The alternative congestion algorithms employed were Scalable TCP and HighSpeed TCP. Network research laboratory experiments were execute to record the functioning of each algorithm under different network forms.

Keywords: *TCP, Congestion Throughput, Algorithm.*

1 Introduction

The Transmission Control Protocol (TCP) is the almost widely utilized congestion control protocol on the web. Most Internet applications suchlike HTTP, FTP, IMAP, POP and SMTP all use TCP to allow for a reliable end to end connection. TCP provides the mechanisms that provide data to be transferred across networks that are dynamic and have a large variety of resources. For instance congestion control keeps the network resources from being overloaded without the need for specified information about network resources. This allows for the network to be very scalable and autonomous which has most likely been the reason for the success of the web. Without transmission control protocol, network resources like core links could easily get congested or underutilized. TCP purposes to keep the utilization of the link as high as possible by slowing down and speeding up each single connection sharing the link[7]. This same system is also what keeps links from being overloaded.

2 TCP Congestion Control Algorithms

2.1 Standard TCP

Standard TCP uses congestion control algorithms described in RFC2581[1]. The algorithms used are:

Slow Start

Congestion Avoidance

Fast Retransmit

Fast Recovery.

A TCP connection is always using one of these four algorithms throughout the life of the connection.

2.1.1 Slow Start

After a TCP connection is established, initial congestion algorithm used is the Slow Start algorithm. During the slow start, congestion window is incremented by one segment for every new ACK received. The connection remains in slow start until one of three events occurs.

The first event is when, congestion window reaches the slow start threshold. The connection then use congestion avoidance algorithm, after the congestion window is greater than, or equal to, the slow start threshold. The slow start threshold, ssthresh, is a variable used to determines when connection should changes

from the slow start algorithm to congestion avoidance algorithm. The initial value of ssthresh is set to an arbitrarily high value[1], this value is not usually reached during initial slow start after a new connection is established.

The second event is receiving the duplicate ACKs for same data. Upon receiving three duplicate ACKs, the connection uses fast retransmit algorithm. The last event that can occur during slow start is a timeout. If a timeout occurs, congestion avoidance algorithm is used to adjust congestion window and slow start threshold[5].

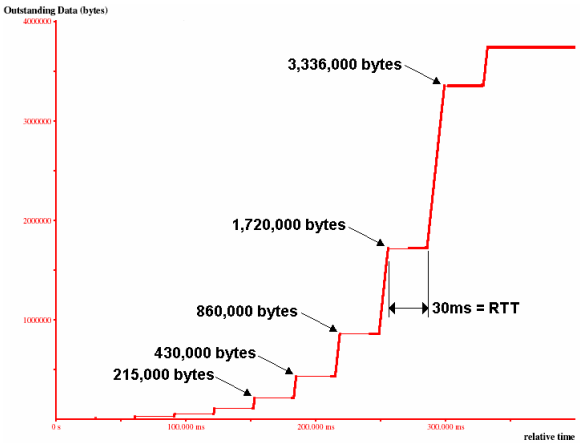

Figure 1. Example of Slow Start.

In figure 1, an example of slow start for a TCP connection is shown. The 30ms delay between increments is round trip time (RTT) for given example. The figure shows the outstanding data bytes for working connection. Observe that how the outstanding data doubles every RTT, this indicates that congestion window is double each RTT. The outstanding data levels off at 3,740,000 bytes. For given example, this is the maximum outstanding data possible given by bandwidth-delay product. The bandwidth-delay product is bandwidth multiplied by delay; In this case of example, it is 1Gb/s * 30ms = 30Mb or approximately 3.75MB. The bandwidth delay product gives the number of bytes it takes to fill the channel that are using.

2.1.2Congestion Avoidance

The congestion avoidance algorithm consists of two parts,

Additive Increment (AI)

Multiplicative Decrement (MD)

It referred to as AIMD. When congestion avoidance occurss, additive increment is used to adjust congestion window after receiving new ACKs. Multiplicative decrement is used to adjust congestion window after a congestion event occurs.

2.1.3 Additive Increment

After receiving an ACK for new data, congestion window is increment by

(MSS)2/Cwnd,

where MSS is maximum segment size, this formula is known as additive increment. The goal of additive increment is to open congestion window by a maximum of one MSS per RTT. Additive increment can be described by using the equation (1):

$$Cwnd = Cwnd + a*MSS2/Cwnd \quad (1)$$

where the value of a is a constant, a = 1.

2.1.4 Multiplicative Decrement

Multiplicative decrement occurs after a congestion event, such as a lost packet or a timeout. After a congestion event occurs, the slow start threshold is set to half current congestion window. This update to slow start threshold follows equation (2):

$$ssthresh = (1 - b)*FlightSize \quad (2)$$

FlightSize is equal to amount of data that has been sent but not yet ACKed and b is a constant, b = 0.5. The congestion window is adjusted accordingly. After a timeout occurs, congestion window is set to one MSS and slow start algorithm is reuse. The fast retransmit and fast

recovery algorithms covers congestion events due to lost packets.

2.1.5 Fast Retransmit

The fast retransmit algorithm was design to quick recovery from a lost packet before a timeout occurs. When sending data, fast retransmit algorithm is used after receiving three duplicate ACKs for same segment. After receiving duplicate ACKs, sender immediately resend the lost packets, to avoid a timeout, and then use the fast recovery algorithm.

2.1.6 Fast Recovery

Fast Recovery algorithm is used to avoid setting the congestion window equal to one MSS segment after a drop packet occurs. After a drop occur, multiplicative decrement portion of congestion avoidance is used to update the slow start threshold. Next, the congestion window is set to a new value of ssthresh. After the window size is decremented, additive increment is used to reopen congestion window.

Figure 2 shows the example of outstanding data for a TCP connection and illustrates the performance of congestion control. The connection starts using slow start algorithm until the channel is filled and then switches congestion avoidance.

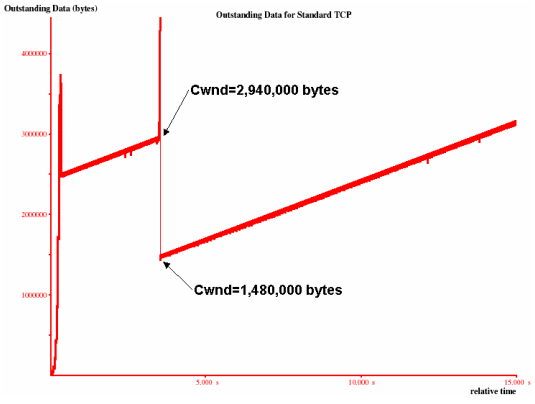

Figure 2. Standard TCP Congestion Control.

In this example, a single drop occurs around three seconds. Fast Retransmit and Recovery algorithms are used to resend lost segment and then cut congestion window in half[4]. Additive increment is used after the drop to reopen the congestion window.

2.2 HighSpeed TCP

The HighSpeed TCP was proposed by Sally Floyd as a sender side alternative congestion control algorithm[2,3]. HighSpeed TCP attempts to improve performance of the TCP connections with large congestion windows. Another goal of HighSpeed TCP is to behaves similarly to a Standard TCP when using small congestion windows.

HighSpeed TCP uses same slow start and fast retransmit and recovery algorithms that are implemented in the Standard TCP. The modifications were only made to the congestion avoidance algorithm.

2.2.1 Congestion Avoidance

The HighSpeed TCP still uses as AIMD congestion avoidance algorithm. The changes made involves adjusting the increment and decrement parameters, more specifically a and b parameters in equations (1) and (2) respectively. New parameters are found in a table and are based on current congestion window in MSS segments, given in equation (3).

$$w = Cwnd/MSS \quad (3)$$

The table is created using equations (4) and (5):

$$a(w) = (HW2 * HP * 2 * b(w)) / (2 - b(w)) \quad (4)$$

$$b(w) = (((HD - 0.5) * (\log(w) - \log(HW))) / (\log(LW) - \log(w))) + 0.5 \quad (5)$$

HW = 83000, HP = 10-7, LW = 38, and HD = 0.1. The first 5 values generated for the HighSpeed TCP table are given below in Table 1.

w	a(w)	b(w)
38	1	0.50
118	2	0.44
221	3	0.41
347	4	0.38
495	5	0.37

Table 1. HighSpeed TCP Table.

2.2.2 Additive Increment

Equation (1) can be rewritten as equation (6) to show the HighSpeed TCP's additive increment formula.

$$Cwnd = Cwnd + a(w)*MSS2/Cwnd \quad (6)$$

The goal of HighSpeed TCP's additive increment is to open congestion window by a(w) in each RTT. Equation (6) allows for large congestion windows to open faster than equation (1).

2.2.3 Multiplicative Decrement

Highspeed TCP follows the same multiplicative decrement algorithm as Standard TCP except for the following changes to equation (2).

$$ssthresh = (1 – b(w))*FlightSize \quad (7)$$

The congestion window is resize same as in Standard TCP after detecting a lost packet, using fast recovery, or a timeout occur.

Figure 3 shows the outstanding data plot for a connection using HighSpeed TCP and shows how congestion control algorithm function. Example shows that the same slow start algorithm is used. A single drop occurs at around on 1 second. This lead to fast retransmit and recovery algorithms being used. At the time of drop Cwnd ~ 3.6MB. Using equation (3) and Table 1, w = 408 segments, so b(w) = 0.37. After the congestion window is reduced to ~2.4MB or w = 262, a(w) = 4. A slight bend in linear increment is observed after around 2.5 seconds. This bend is when w becomes greater than 347 and uses a(w) = 5.

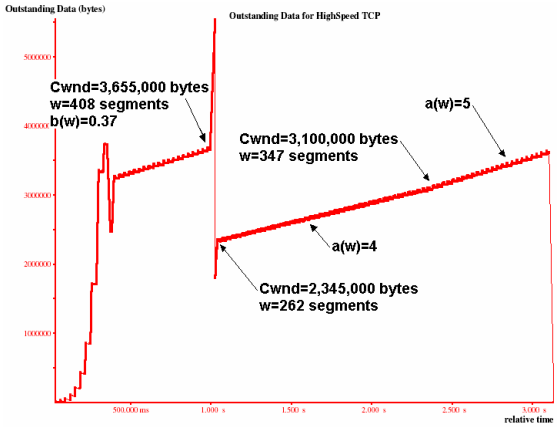

Figure 3. Example of HighSpeed TCP Congestion Control.

2.3 Scalable TCP

Tom Kelly proposed Scalable TCP as another alternative sender side congestion control algorithm[1]. The goal of Scalable TCP is to quickly recover from short congestion periods.

2.3.1 Congestion Avoidance

Scalable TCP uses a different congestion avoidance algorithm than Standard TCP. Scalable TCP uses a multiplicative increment multiplicative decrement (MIMD) rather than the AIMD of Standard TCP.

2.3.2 Multiplicative Increment

The multiplicative increment occurs when standard additive increment would normally occurs. In equation (8) shows the formula used to adjust congestion window after receiving a new ACK.

$$Cwnd = Cwnd + a*Cwnd \quad (8)$$

where a is adjustable, the value of a used was 0.02.

2.3.3 Multiplicative Decrement

The multiplicative decrement is same as Standard TCP except that the value of b in

equation (2) is adjustable, the value of b used 0.125.

Figure 4 shows an example of a Scalable TCP connection and congestion control algorithm that it use. The connection starts in the slow start algorithm until channel is filled. The connection uses the multiplicative increment portion of congestion avoidance to adjust congestion window. After a single drop occur around 1.4 seconds, fast retransmit and recovery algorithms are used to cut congestion window by 0.125, the value of b, and congestion avoidance is used again to reopen congestion window.

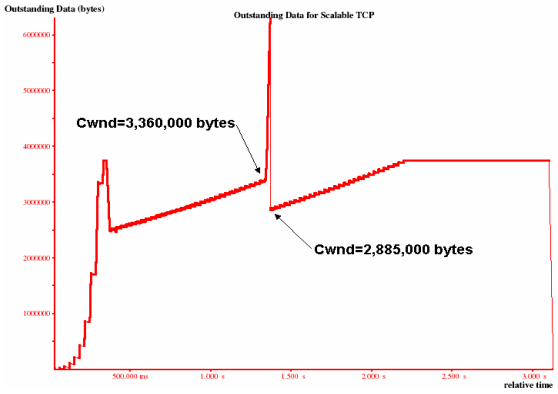

Figure 4. Example of Scalable TCP Congestion Control.

3 Summary of TCP Congestion Control Algorithms

	Increment	Decrement	Algorithm
Standard TCP	a*MSS^2/Cwnd	b*FlightSize	AIMD
HighSpeed TCP	a(w)*MSS^2/Cwnd	b(w)*FlightSize	AIMD
Scalable TCP	a*Cwnd	b*FlightSize	MIMD

Table 2 gives an overview of the congestion control algorithms used.

4 Conclusions:

To summarize, almost encouragingly, in a newly design space where control rules apply history information, window based congestion control mechanisms can be TCP friendly, and still provide smoothness as well as better transient behaviour. They can solve the problem evoked by slowly responsive congestion controls. Applied that equation based congestion control schemes use longer history, we consider comparisons between equation based schemes and our scheme remain an interesting for future work[6].

HighSpeed TCP and Scalable TCP implements simple changes to the currently used congestion control algorithm. These changes have both a positive and negative effects on the existing network traffic. Each algorithm provides higher channel utilization for high speed and long delay environment. However, the alternative algorithms do not shares channel equally, when mixed with Standard TCP traffic. In a homogenous environment, the overall channel utilization and sharing between streams increments as compared to a mixed environment. Future work is needed to study the effects of more than two competing streams.

References:

[1] M. Allman, V. Paxson, and W. Stevens. TCP Congestion Control, Internet RFC 2581, April 1999.

[2] S. Floyd. HighSpeed TCP for Large Congestion Windows. Internet Draft < draft-ietf-tsvwg-highspeed-01.txt>, August 2003.

[3] T. Kelly. Scalable TCP: Improving Performance in Highspeed Wide Area Networks, URL http://www.lce.eng.cam.ac.uk /~ctk21/scalable/, December 2002

[4] High-Speed TCP, URL http://www.hep.man.ac.uk/u/garethf/hstcp

[5] Scalable TCP, URL http://www.lce.eng.cam.ac.uk /~ctk21/scalable/

[6] K. Ramakrishnan and S. Floyd, A proposal to add explicit congestion notification (ECN) to IP." RFC 2481, January 1999.

[7] M. Christiansen, K. Jeray, D. Ott, and F. D. Smith, Tuning RED for web traffic," in Proc. of ACM SIGCOMM'00, (Stockholm, Sweden), September 2000.

About Author: Kulvinder Singh received the M.Tech.(CSE) degree in 2006 and the M.Phil.(CS) degree in 2008 from Ch. Devi Lal University Sirsa(Haryana), India. He is pursuing Ph.D. in Computer Science. At present he is working as Assistant Professor in Vaish College of Engineering, Rohtak, India. He is having more than 6 years of experience both in industry and academia. He is a member of IEC. He presents many research papers in national and international conferences. He published many research papers in various International Journals. His interest areas are Networking, Web Security, Internet Congestion and Fuzzy Database.