A Survey on Congestion Control Mechanisms for Performance Improvement of TCP

Shital N. Karande
Department of Computer Science Engineering,
VIT, Pune, Maharashtra, India

Sanjesh S. Pawale
Department of Computer Science Engineering,
VIT, Pune, Maharashtra, India

Abstract- Transport Control Protocol (TCP) carries most of the traffic of internet, so the performance of internet depends on how TCP performs. TCP provides different features like flow control, reliability, congestion control etc. Hence one of the factors affecting the performance of TCP is Congestion Control Algorithm used. Network is a shared resource and congestion control decides how it should be shared between entities. The Transmission Control Protocol comes in many variants like TCP, Tahoe, Reno, NewReno, and Westwood and so on. Each of these variants would work differently in networks according to the parameters of that network.
In this paper, we have compared different variants of TCP. We have analyzed the behavior of these TCP variants on the basis of the when a segment should be re-transmitted and how should the sender behave when it encounters congestion and what pattern of transmissions should it follow to avoid congestion.

Keywords - TCP, Congestion control, TCP variants

I. Introduction

The most common transport protocol used in internet for data transmission is Transmission Control Protocol. Congestion in a network may occur if the load on the network is greater than the capacity of the network i.e. the number of packets sent to the network is greater than the number of packets a network can handle. Packet loss can be used to detect congestion because modern networks are reliable and rarely lose packets through hardware failure. Congestion control refers to techniques and mechanisms that can either prevent congestion, before it happens, or remove congestion, after it has happened.

There are many variants of TCP each variant being used for a specific purpose. The four variants we use in this paper are TCP Tahoe, Reno, NewReno, and Westwood.

II. TCP

TCP is a reliable connection oriented end-to-end protocol. It is a full duplex protocol, meaning that each TCP connection supports a pair of byte streams, one flowing in each direction. TCP includes a flow-control mechanism for each of these byte streams that allow the receiver to limit how much data the sender can transmit.

It contains within itself, mechanisms for ensuring reliability by requiring the receiver acknowledge the segments that it receives. The network is not perfect and a small percentage of packets are lost en route, either due to network error or due to the fact that there is congestion in the network and the routers are dropping packets. We shall assume that packet losses due to network loss are minimal and most of the packet losses are due to buffer overflows at the router. Thus it becomes increasingly important for TCP to react to a packet loss and take action to reduce congestion. TCP ensures reliability by starting a timer whenever it sends a segment. If it does not receive an acknowledgement from the receiver within the 'time-out' interval then it retransmits the segment.

III. CONGESTION CONTROL

In any network, when the number of packets in the network increases randomly, it leads to packet delay or packets may be dropped. This situation is called network congestion. In TCP, to avoid congestion, we use congestion windows, which determine the number of byes that can be outstanding at any time in the network. This is a means of stopping the link between two places from getting overloaded with too much traffic. The size of the window is calculated by estimating how much congestion is there between the sender and receiver.

Usually, sender maintains the congestion window. Acknowledgements for data sent are used by senders to infer network conditions between TCP sender and receiver. The slow-start and congestion-avoidance phases are used by a TCP sender to control the amount of outstanding data being injected into the network and are implemented by maintaining a set of state variables per TCP connection. The congestionwindow (cwnd) is a sender-side limit to the amount of data the sender can transmit into the network before receiving an acknowledgement (ACK), while the receiver's advertised window (rwnd) is a receiver-side limit to the amount of outstanding data. The minimum of cwnd and rwnd governs data transmission. Another state variable, the slow start threshold (ssthresh), is used to indicate the transition point from slow-start to congestion avoidance. Modern implementations of TCP contain four algorithms: slow-start, congestion avoidance, fast retransmit and fast recovery.

IV. CONGESTION CONTROL ALGORITHMS

A. Slow start-

Beginning transmission into a network with unknown conditions requires TCP to slowly probe the network to determine the available capacity, in order to avoid congesting the network. When a new connection is established with a host on another network, the congestion window (cwnd) is initialized to one segment. Each time an ACK is received, the congestion window is increased by one segment. The sender can transmit up to the minimum of the congestion window and the advertised window.

The sender starts by transmitting one segment and waiting for its ACK. When that ACK is received, the congestion window is incremented from one to two, and two segments can be sent. When each of those two segments is acknowledged, the congestion window is increased to four. This provides an exponential growth, although it is not exactly exponential because the receiver may delay its ACKs, typically sending one ACK for every two segments that it receives. At some point the capacity of the internet can be reached, and an intermediate router will start discarding packets. This tells the sender that its congestion window has gotten too large. Slow-start ends when the cwnd exceeds ssthresh or when congestion is detected.

B. Congestion avoidance-

TCP operates in congestion avoidance where cwnd is incremented by 1 MSS per round-trip-time (RTT). Congestion avoidance continues until congestion is detected. Congestion avoidance dictates that cwnd be incremented by segsize*segsize/cwnd each time an ACK is received, where segsize is the segment size and cwnd is maintained in bytes. This is a linear growth of cwnd, compared to slow start's exponential growth. The increase in cwnd should be at most one segment each round-trip time (regardless how many ACKs are received in that RTT), whereas slow start increments cwnd by the number of ACKs received in a round-trip time. When congestion occurs (indicated by a timeout or the reception of duplicate ACKs), one-half of the current window size (the minimum of cwnd and the receiver's advertised window, but at least two segments) is saved in ssthresh. Additionally, if the congestion is indicated by a timeout, cwnd is set to one segment (i.e., slow start). When new data is acknowledged by the other end, increase cwnd, but the way it increases depends on whether TCP is performing slow start or congestion avoidance. If cwnd is less than or equal to ssthresh, TCP is in slow start; otherwiseTCP s performing congestion avoidance. Slow start continues until TCP is halfway to where it was when congestion occurred (since it recorded half of the window size that caused the problem in step 2, and then congestion avoidance takes over. Slow start has cwnd begin at one segment, and be incremented by one segment every time an ACK is received.

C. Fast retransmit-

TCP may generate an immediate acknowledgment (a duplicate ACK) when an out- of-order segment is received. This duplicate ACK should not be delayed. After receiving a small number (usually 3) of duplicate acknowledgements for the same TCP segment (dup ACKs), the transmitter infers that a packet has been lost, and retransmits the packet without waiting for a retransmission timer to expire, resulting in higher channel utilization and connection throughput.

D. Fast recovery-

The receipt of the duplicate ACKs tells TCP more than just a packet has been lost. Since the receiver can only generate the duplicate ACK when another segment is received, that segment has left the network and is in the receiver's buffer. That is, there is still data flowing between the two ends i.e. between sender and receiver. After fast retransmit sends what appears to be the missing segment, congestion avoidance is performed instead of slow start because TCP does not want to reduce the flow abruptly by going into slow start.

V. TCP VARIANTS

A. TCP Tahoe-

TCP Tahoe is the first algorithm which employs three congestion control algorithms: slow start, congestion avoidance, fast retransmit.

Algorithm:

Slow start algorithm

```
Initialize cwnd=1
For (each segment ACKed)
        Cwnd++;
Until (congestion event or cwnd>ssthresh)
```

Congestion avoidance algorithm

```
/* cwnd>ssthresh*/
Every new ACK:
        cwnd+=1/cwnd;
Until (timeout or 3 DUPACK) /* packet loss event */
```

Fast retransmit algorithm

```
If receiving 3 DUPACK or RTO /* packet loss event */
Retransmit the packet
        ssthresh=cwnd/2;
        cwnd=1;
Perform Slow start
```

B. TCP Reno-

TCP Reno employs four Congestion control Algorithms: slow start, congestion avoidance, fast retransmit, and fast recovery. A loss event during the congestion avoidance stage, (when the congestion window - cwnd - is above ssthresh), which is identified at the sender side by triple duplicate ACK causes cwnd to drop (”deflate”) by half. This feature was added to avoid reverting to slow-start from a minimal window size. When packet loss occurs in a congested link due to buffer overflow in the intermediate routers, either the sender receives three duplicate acknowledgments or the sender’s RTO timer expires. In case of three duplicate ACKs, the sender activates TCP fast retransmit and recovery algorithms and reduces its congestion window size to half. It then linearly increases congestion window, similar to the case of congestion avoidance. This increase in transmission rate is slower than in the case of slow start and helps relieve congestion. TCP Reno fast recovery algorithm improves TCP performance in case of a single packet loss within a window of data. However, performance of TCP Reno suffers in case of multiple packet losses within a window of data.

TCP Reno implements slow start, congestion avoidance and fast retransmit same as TCP Tahoe. In addition fast recovery algorithm is added to TCP Reno.

Algorithm:

Slow start

```
        /* Same as in TCP Tahoe */
```

Congestion avoidance

```
        /* Same as in TCP Tahoe */
```

Fast retransmit

```
If receiving 3 DUPACK or RTO
        Retransmit the packet
After retransmission do not enter slow start
Enter fast Recovery
```

Fast recovery

```
        Set ssthresh=cwnd;
        Cwnd=ssthresh+3;  /* for 3 extra packets leaving the network causing DUPACK*/
        Cwnd++; /* to compensate for the one leaving the network*/
If new ACK, cwnd=ssthresh;
Return to Congestion avoidance
```

C. TCP NewReno-

It improves retransmission process during the fast recovery phase of TCP Reno. TCP NewReno can detect multiple packet losses. It does not exit the fast recovery phase until all unacknowledged segments at the time of fast recovery are acknowledged. as in TCP Reno, it overcomes reducing the congestion window size multiple times in case of multiple packet losses.

D. TCP Westwood-

TCP Westwood is a rate based algorithm extending the TCP Reno**.** TCP Westwood congestion control algorithm use a bandwidth estimation, it executed at sender side of a TCP connection. The congestion window dynamics during slow start and congestion avoidance are unchanged. The general idea is to use the bandwidth estimate BWE to set the congestion window (cwnd) and the slow start threshold (ssthresh) after a congestion episode. In TCP Westwood the sender continuously computes the connection BWE which is defined as the share bottleneck used by the connection. Thus, BWE is equal to the rate at which data is delivered to the TCP receiver. The estimate is based on the rate at which ACKs are received and on their payload. After a packet loss, the sender resets the congestion window and the slow start threshold based on BWE. The packet loss is suspected with a reception of three duplicates ACKs or timeout expiration. Another important element of this procedure is the RTT estimation. That is because the congestion window is set precisely to BWE * RTT after indication of packet loss.

Bandwidth estimation (BWE) algorithm-

$$BWE = bk = \alpha_k\, bk{-}1 + (1 - \alpha_k)\,[(bk + bk{-}1)/2]$$

Where bk = sample bandwidth

$= dk / t_k - t_{k-1}$

wheredk = amount of bytes acknowledged by ACK k,

t_k = arrival time of ACK k,

$$\alpha_k = [2\tau - \Delta(t_k - t_{k-1})] / [2\tau + \Delta(t_k - t_{k-1})]$$

where τ : is the cut- off frequency of this Tustin filter

After 3 DUPACKS

```
If receiving 3 DUPACKS
        Set ssthresh =(BWE*RTTmin) /seg_size;
        and if cwnd > ssthresh
                then set cwnd = ssthresh ;
        enter congestion avoidance
```

After Timeout

```
If RTO then set
        ssthresh = (BWE*RTTmin) /seg_size;
        if (ssthresh < 2) ssthresh =2;  end if ;
        cwin = 1;
end if
enter slow start;
```

VI. COMPARATIVE ANALYSIS

Table-I Comparison of TCP variants

Sr No	TCP VARIANTS	BASE of TCP VARIANTS	ADDED FEATURES	WEAKNESS
1	TCP TAHOE	RFC 793	➢ Slow start ➢ Congestion Avoidance ➢ RTO, Duplicate- Packet loss Detection ➢ Fast retransmit- without waiting RTO	➢ It takes a complete timeout interval to detect a packet loss
2	TCP RENO	TAHOE	➢ Slow start ➢ Congestion Avoidance- one packet loss ➢ Fast Recovery- Congestion detection- dup packets, inflates congestion window by no. of dup packets ➢ Congestion Avoidance-Non dup packets	➢Doesn't handle multiple packet loss ➢Subsequent loss causes window to decrease further
3	TCP NEW RENO	RENO	➢ Modified Fast Recovery- doesn't exit from recovery phase until all data packets are acknowledged from initial congestion window ➢ Congestion window inflates with dup packets and deflates with ACK	➢New-Reno suffers from the fact that it takes one RTT to detect each packet loss.
4	TCP Westwood	RENO	➢ Modified TCP Reno ➢ Idea is to use the bandwidth estimate BWE to set the congestion window (cwnd) and the slow start threshold (ssthresh) after a congestion episode.	➢Not effectively transmit with a rate that utilizes a fair link capacity sharing

VII. CONCLUSION

In this work, we have presented a survey of various approaches to TCP congestion control that do not rely on any explicit signaling from the network. TCP Tahoe, Reno mechanisms provide varying in size of congestion window depending on ACK status, thus when packets acknowledged the window size is increased and decreased when detect lost in packets.

When TCP is used for wireless network, the packet losses often caused by sporadic losses, then performance of TCP degrades due to the activation of congestion control algorithm. So, TCP Westwood congestion algorithm designed to improve the throughput of wireless network. In TCP Westwood mechanism, bandwidth estimate BWE is used to set the congestion window (cwnd) and the slow start threshold (ssthresh) after a congestion episode.

REFERENCES

[1] M. Welzl,"Network Congestion Control", Managing Internet Traffic, Leopold Franzens, University of Innsbruck, 2005
[2] G.A. Abed, M. Ismail and K. Jumari,"A Survey on Performance of Congestion Control Mechanisms for Standard TCP Versions",Australian Journal of Basic and Applied Sciences, 2011, ISSN 1991-8178
[3] M. Kalpana1 and T. Purusothaman, "Performance Evaluation of Exponential TCP/IP Congestion Control Algorithm", International Journal of Computer Science and Network Security (IJCSNS), VOL.9 No.3, March 2009.
[4] R. Raiand and M. Shreevastava,"Performance Improvement of TCP by TCP Reno and SACK Acknowledgement", International Journal of Advanced Computer Research (ISSN:2277-7970) Volume 2 Number 1March 2012
[5] S. Hagag, A. El-Sayed,"Enhanced TCP Westwood Congestion Avoidance Mechanism (TCP Westwood New)",International Journal of Computer Applications (0975 – 8887) Volume 45– No.5, May 2012
[6] L. A. Grieco, S. Mascolo,"Performance Comparison of Reno, Vegas and Westwood+ TCP Congestion Control"
[7] Dah -Ming Chiu, Jain, R. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. Computer Networks and ISDN Systems, 17(1), (1989), 1- 14

[8] V. Jacobson and M. J. Karels, "Congestion avoidance and control", In ACM Computer Communication Review; Proceedings of the Sigcomm'88 Symposium, volume 18, pages 314–329, Stanford, CA, USA, August 1988

[9] Hanaa A. Torkey, Gamal M. Attiya and I. Z. Morsi, "Enhanced Fast Recovery Mechanism for improving TCP NewReno", Proceedings of the 18th International Conference on Computer Theory and Applications (ICCTA08), pp. 52-58, Alexandria, Egypt, 11-13 October 2008.