ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 6, Issue 10, October 2017

Congestion Control Techniques in Transport Layer for Wired Connections

Sweeti Sah, Aman Verma, Ghanshaym Chaurasia, Jitendra Kurmi*

BBAU, Vidya Vihar Raibareily Road Lucknow, India

Abstract: This paper presents the comparative analysis of TCP Congestion Control Techniques including Tahoe, Reno, New Reno, SACK, Vegas and how these techniques different from each other. When any packet is being lost or timeout occurs, these techniques come into role and what is the effect on throughput, efficiency, performance when compared with TCP Vegas.

Keywords: Efficiency, Throughput, TCP tahoe, TCP reno, TCP new reno, TCP sack, TCP vegas

I. INTRODUCTION

TCP is connection oriented end to end transmission protocol. Reliability of packet is ensured by receiving the acknowledgment segment within the timeout interval by the receiver node. Packet loss can be because of the delay, timeout, buffer overflow and etc. We assume the loss due to network is minimal but due to buffer overflow is more at router [1]. So these techniques are introduced to deal with congestion and how they react and take appropriate action and improve throughput, efficiency. There are few components:

Slow Start
The congestion window start with size = 1 and grows on exponentially until it reaches it threshold value.

Additive Increase Multiplicative Decrease (AIMD)
When congestion window size reaches threshold value then it decreases the congestion window multiplicatively further do linear increment. Fast retransmit and fast recovery are other two components.

TCP Tahoe
Suggested by Van Jacobson in 1988 [1]. Start with slow start mechanism, congestion window size = 1. Network capacity can be determined by congestion window [1]. As we send data packet we get an acknowledgement then increment the congestion window size and keep on sending the data until reaches threshold value and move into congestion avoidance phase, there it keep on sending the data and after getting an acknowledgment it just increment congestion window = congestion window +1/ congestion window and we keep on sending unless loss or time out occurs. After getting three duplicate acknowledgement it moves into Fast Retransmit state and send the missing packet. Set the threshold value as congestion window/2 and congestion window = 1, move to slow start phase. In case of timeout in slow start and congestion avoidance phase, it moves into Retransmission timeout phase, when all acknowledgement is received from retransmission timeout phase to slow state phase for whatever packet is being send. The process repeat and so on (Fig. 1).

TCP Reno
First few steps of TCP Reno are same as TCP Tahoe. When it is in Fast Retransmit state [2], it moves immediately to Fast Recovery state and set threshold = congestion window/2 and congestion window = threshold, after that sends missing packet (Fig. 2).

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 6, Issue 10, October 2017

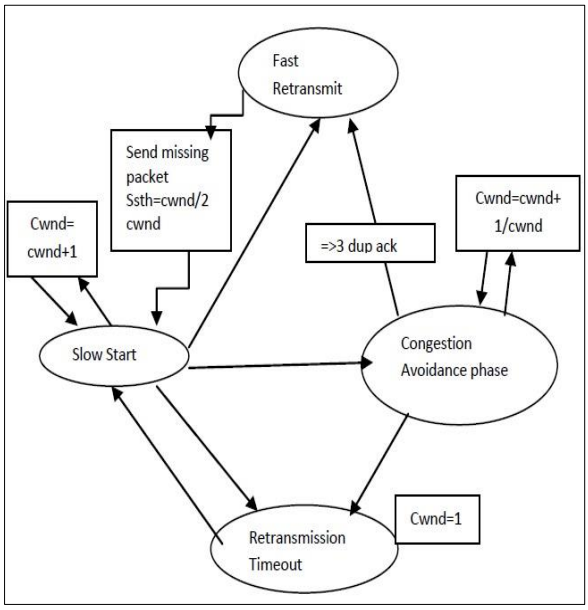

Fig. 1. TCP Tahoe.

From Fast Recovery state after receiving duplicate acknowledgment, it increases congestion window = congestion window +1 and keep on sending the data and move to congestion avoidance phase when there are no duplicate acknowledgements left by setting congestion window = threshold.

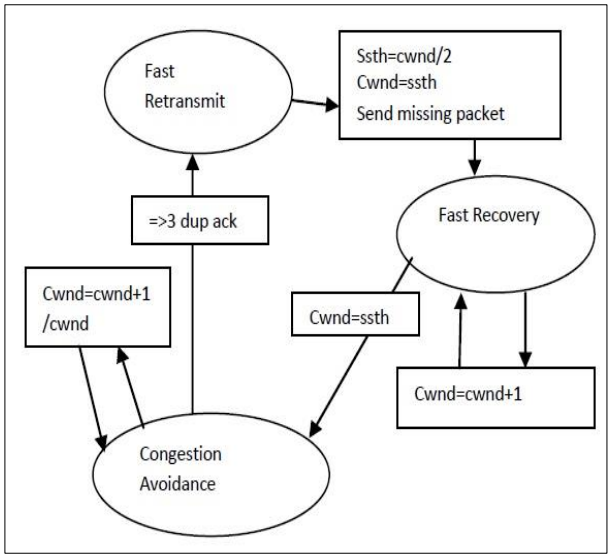

Fig. 2. TCP Reno.

TCP New Reno

It extends fast Recovery state phase and remain in Fast Recovery state until all data in pipe before detecting three duplicate acknowledgement are acknowledged [3]. Able to avoid the problem of multiple packet loss problem.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 6, Issue 10, October 2017

TCP Sack

It report non continuous block of data. After the detection of packet loss, more than one lost packets can sent in one Round Trip Time. Acknowledgement of packet is done selectively for maximum utilization. Whenever the sender enters into Fast Recovery state, a variable will be initialized which will estimate how much data is outstanding in the network. It will set congestion window as half the current size. For every acknowledgement it receives it reduces the pipe by one and retransmit a segment and increment by one. Whenever the pipe goes smaller than congestion window. It will check which segment is not received and send them again. If there are no segment outstanding then it will send a new packet [4,5]. Thus in one RTT more than one segment can be send.

TCP Vegas

It is proactive in nature. It detects early packet loss. It is more efficient than all the above mentioned and also overcomes the problem of requiring enough duplicate acknowledgements to detect packet loss. It does not wait for three duplicate acknowledgement [6] to send the lost packet. It keeps the track of all the segment that is being send and also calculate the estimation of Round Trip Time by keeping the track that how much time it is going to take to receive an acknowledgement back.

TCP Vegas is different compared to other implementation during Congestion Avoidance phase. Instead of detecting the congestion by loss of segment, it detect by decreasing sending rate compared to expected rate as a result of large queues that is building inside the routers. It uses a variation of Wang and Crowcroft; s Tri-S scheme [6].

Comparision

Comparision for TCP Tahoe, TCP Reno, TCP New Reno, TCP New Reno, TCP Sack and TCP Vegas as shown in Table 1.

	Solution	**Problem**
TCP TAHOE	**Slow Start and Congestion**	Complete Timeout Interval to detect Packet Loss
	Avoidance	Cumulative ACK
	Increase window size	Cwnd = 1 when packet loss
	Fast Retransmit	Inefficient
	Detects Congestion	Pipeline emptied
TCP RENO	Cwnd = Cwnd/2	Inefficient for Multiple
	Immediate ACK	Packet Loss
	Packet loss is detected earlier, whenever there is three duplicate ack, i.e. the sign of one packet loss. After Fast Retransmit state, it enters into Fast Recovery state.	
	Pipeline is full Efficient than Tahoe	
TCP NEW RENO	Detect Multiple Packet Loss	Takes one RTT (Round Trip Time) to detect one packet loss
	Extends Fast Recovery Phase until all data in pipe before detecting three duplicate ACK are acked.	
TCP SACK	Retransmission of more than one lost packet per RTT.	Not Easy
	Not acknowledged cumulatively but selectively.	
	Sender retransmit only the segments that have actually been lost.	
TCP VEGAS	Overcomes the problem of getting three duplicate ACK for One packet loss.	Cannot Compete with more aggressive TCP Reno connection
	Proactive	Rerouting path may change propagation delay
	Efficient	Adjust sending rate
	Detects Congestion before Packet Loss	Performance may degrade in asymmetric network
	Detects Multiple Packet Loss faster	Treats all packet loss as Random loss

Table 1. Comparision for TCP Tahoe, TCP Reno, TCP New Reno, TCP New Reno, TCP Sack and TCP Vegas.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 6, Issue 10, October 2017

II. PROPOSED SOLUTION

The proposed solutions can be implemented using ns2. The throughput and Efficiency of TCP Vegas can be improved when symmetrical network is used and by creating an algorithm that could distinguish between packet loss and random loss. In the algorithm of TCP Vegas there is a calculation of Expected Sending Rate and Actual Sending rate. Additionally we can calculate the time of Actual Sending the data packet and estimating the Expected Time of sending and receiving the data packet when it stored in buffer.

III. CONCLUSION

Hence TCP Vegas is more efficient than TCP Tahoe, Reno, New Reno, Sack by improving the throughput as it detect the packet loss before it occur and extending the re-transmission mechanism of RENO. TCP Vegas do not waste bandwidth by transmitting too high at data rate. And also when connection starts TCP Vegas has no idea of available bandwidth.

REFERENCES

[1] V. Jacobson, "Congestion Avoidance and Control", SIGCOMM Symposium no Communication Architecture and protocols.
[2] V. Jacobson, "Modified TCP Congestion Control and Avoidance Alogrithms", Technical Report, 1990.
[3] S. Floyd, T. Henderson, "The New- Reno Modification to TCP's Fast Recovery Algorithm", RFC 2582, 1999.
[4] O. Ait-Hellal, E. Altman, "Analysis of TCP Reno and TCP Vegas", 1997.
[5] K. Fall, S. Floyd, "Simulation Based Comparison of Tahoe, Reno and SACK TCP", 1996.
[6] LS. Brakmo, LL. Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet", IEEE Journal on Selected Areas in Communication, vol.13, pp.1465-1490, 1995.