PERFORMANCE BASED COMPARISON OF TCP VARIANTS 'TAHOE, RENO, NEWRENO, SACK' IN NS2 USING LINUX PLATFORM

[1] HARDIK V. MIYANI, [2] VISHV B. KUKADIYA,
[3] MR. KAPIL S. RAVIYA,[4]MR.DHRUMIL SHETH

[1,2] Student, Electronics & Communication Engineering Dept., C.U.Shah College of Engineering and Technology, Wadhwancity
[3]Assistant Professor, Electronics & Communication Engineering Dept., C.U.Shah University

erhvmiyani@gmail.com, vishv.92@gmail.com, raviyakapil@gmail.com , dhrumildeep@gmail.com

ABSTRACT: *TCP/IP protocol, which was formerly developed for wired links, is now an inseparable part of the Internet. Hence, its competence on wireless links could play a significant role in the performance of the Internet. The use of original TCP/IP protocol on wireless links in spreading the Internet has encountered some serious performance issues, the reason being the wired links are very less prone to channel errors and more affected by congestion. There is no way in TCP to distinguish the correct reason for losses hence losses are not treated distinctively. The research of more than 25 years has gone through different variants of TCP, out of which most up-to-date variant TCP SACK (Selective Acknowledgement) is the most resourceful. Its potential to avoid redundant retransmissions based on SACK information accessible at TCP sender. It should be noticed that even TCP SACK is powerless of judging the concrete cause of loss i.e. corruption or congestion. In this paper we will vary different parameters in our scripts and observe the performance of throughput and its graphs. And we will conclude that which one is the better.*

KEYWORDS*: *Congestion window, Corruption, Throughput, Fast retransmit, Fast recovery, Slow Start Threshold

1. INTRODUCTION[1]

TCP was mainly used in wired links. Because wired links have very less chances of high delay and data corruption of due to external parameters. Congestion is the main reason of packet loss on wired links. So, TCP was designed by keeping in mind the above parameters. As Technology upgrades, wireless and heterogeneous networks came into the existence, due to the requirement of reliable protocol in TCP/IP model in internet, TCP was adopted as it was on wired links. Wireless links have severe problem of variable and high delay with high Bit Error Rate (BER). So initially, unmodified old TCP started to perform badly on wireless links. To deal with the problems of wireless links, a research started in the field of TCP and modifications were done according to the requirements to improve the performance. Variants named Tahoe, Reno, New Reno and SACK and many more came into existence.

2. BASICS ABOUT TCP VARIANTS

2.1 TAHOE TCP[4]

Tahoe by Jacobson assumed that congestion signals are represented by lost segments. It was assumed by Jacobson that losses due to packet corruption are much less probable than losses due to buffer overflows on the network. Therefore, on a loss, the sender should lower its share of the bandwidth. This is done by reducing its cwnd to half of the size at which the loss was found. The reasoning behind this value of a half is that the decrease in throughput should be equal to the multiplicative increase of queue length in the network upon congestion. The implementation of this multiplicative decrease is through the use of a tcp variable called ssthresh. Upon a loss, half of the value of cwnd just before the loss is recorded in ssthresh. The connection then resorts back to slow start by setting cwnd to 2 segments. Slow start grows the cwnd exponentially until it reaches ssthresh from which it will do congavoid until the same thing happens again until the connection is terminated. In order to determine that a packet is lost, we must time the delay of the packet; from the sender putting into the network and the time at which we receive the ack for that packet. This value is known as the round trip time (RTT). From this value (and the aggregation of timed pairs), we can use a Retransmission Time-Out (RTO). If an ack is not received before this RTO, then the sender should be confident that the packet is lost and should therefore resend the segment to enable reliable delivery and movement of the window.

LIMITATIONS OF TCP TAHOE:

TCP Tahoe does not deal well with multiple packet drops within a single window of data.

2. RENO TCP

TCP TAHOE + FAST RECOVERY = TCP RENO

TCP Reno introduced major improvements over Tahoe by changing the way in which it reacts to detecting a loss through duplicate acknowledgements. The idea is that the only way for a loss to be detected via a timeout and not via the receipt of a dupack is when the flow of packets and acks has completely stopped - This would be an indication of heavy congestion.

MOTIVATION FOR IMPROVING RENO:

In the Internet, packets are often transmitted in bursts (bursty nature of tcp etc). As a result, losses also often happen in bursts. This is primarily due to FIFO (drop tail) queues in routers. The fundamental problem is that Fast Retransmit assumes that only one segment was lost. This can result in loss of ack clocking and timeouts if more than one segment is lost.

LIMITATIONS OF TCP RENO:

Reno encounters several problems with multiple packet losses in a window of data (usually in the order of half a window). This usually happens when invoking Fast Retransmit and Fast Recovery.

3. NEW RENO TCP[3]

TCP RENO + RECOVERY OF MULTIPLE PACKET LOSS = NEW RENO TCP

A modification of Reno lead to New-Reno TCP which shows that Reno can be improved without the addition of Selective ACKs but still suffers without it. Here, the wait for a retransmit timer is eliminated when multiple packets are lost from a window. New Reno is the same as Reno but with more intelligence during fast recovery. It utilizes the idea of partial acks: when there are multiple packet drops, the acks for the retransmitted packet will acknowledge some, but not all the segments send before the Fast Retransmit. In TCP Reno, the first partial ACK will bring the sender out of the fast recovery phase. This will result in the requirement of timeouts when there are multiple losses in a window, and thus stalling the tcp connection. In New Reno, a partial ack is taken as an indication of another lost packet and as such the sender retransmits the first unacknowledged packet. Unlike Reno, partial acks don't take New Reno out of Fast Recovery. This way, it retransmits one packet per RTT until all the lost packets are retransmitted and avoids requiring multiple fast retransmits from a single window of data.

4. SACK TCP

The main difference between the SACK TCP, New Reno and Reno is in case of multiple packet drops from one window of data. TCP with selective acknowledgement is designed to provide information about the loss of multiple segments in a window of data. The receiver uses the SACK option to inform the sender of all successfully received packets but not cumulatively acknowledged. The sender uses this information to retransmit selectively only the packets that were lost.

5. SIMULATION SCENARIO

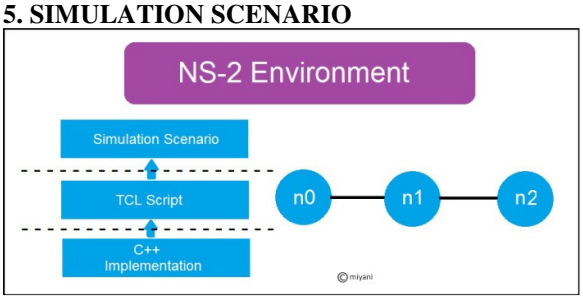


Figure: 1 Simulation Scenario

6. READING PARAMETERS

For performance measurement, parameters like delay, drop/error rate, and file size are kept variable for different variants. TCPTRACE software is used to trace actual path of data transfer for analysis. XPLOT software can be used to plot graphs of different quantities like sequence numbers, round trip time, congestion window, throughput etc.

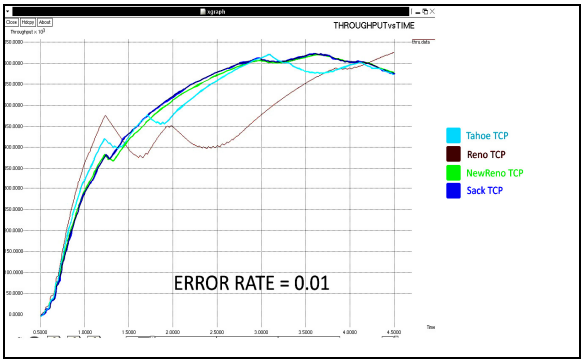
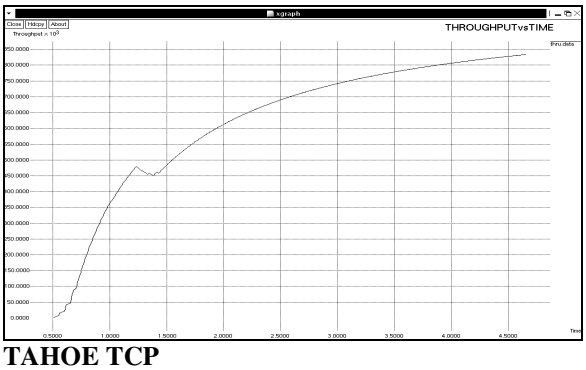
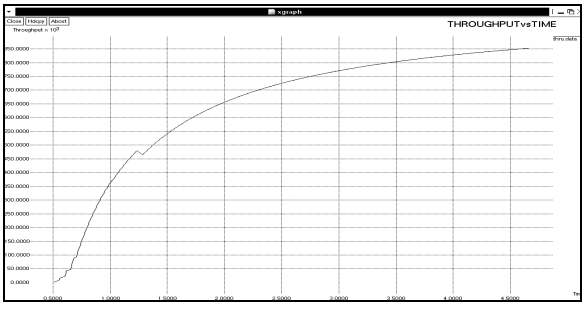


Figure: 2 X-Graph for Error Rate = 0.01

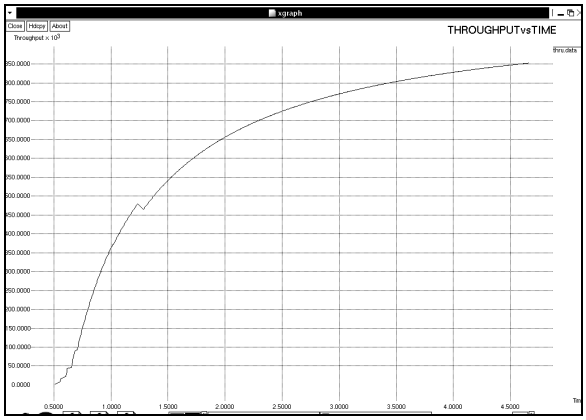
Similarly we can get ERROR RATE = 0.001 and x-Graphs are shown below,



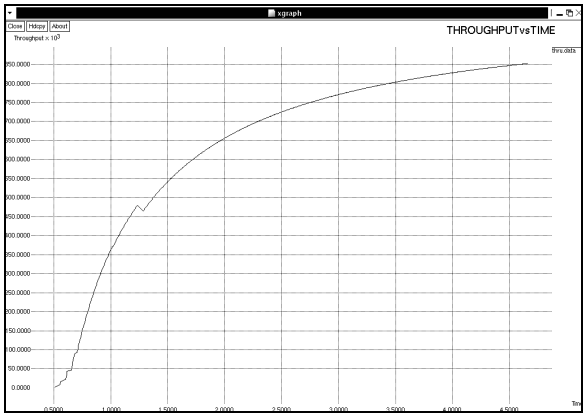
TAHOE TCP



RENO



NEW RENO



SACK TCP

7. CONCLUSION

New-Reno TCP is a variant of Reno with a little modification within Fast Recovery algorithm. This was done in order to solve the timeout problem when multiple packets are lost form the same window. Note that higher performances were obtained due to the little modification of Reno TCP. Although New Reno solves the timeout problem when multiple packets are lost form the same window, it can retransmit only one packet per Round Trip Time. SACK provides even better performance and gives us the idea that why it is used currently used everywhere. It uses two algorithm of fast retransmit and fast recovery together so we get this enhanced data rates.

In case of Reno & SACK both, as delay on wireless link increases, throughput decreases gradually, as round trip time for each data packet increases before reaching to destination node as well as for each acknowledgement packet to reach to the source node.

In case of respective drop/error rate, average throughput maintained at higher digit in case of SACK compared to Reno. So, performance degradation is higher in Reno compared to SACK.

8. REFERENCES

1. Kevin Fall & Sally Floyd, "Simulation – based comparisons of Tahoe, Reno and SACK TCP"

2. S.Floyd & T. Henderson, "The NewReno modification to TCP's Fast Recovery algorithm", RFC 2582 April, 1999

3. Data Communication and Networking (Mc Graw Hill, Author: Forouzan, 5th Edition)

4. T.V.Lakshman, U. Madhow, and B. Suter, "Window-based error recovery and flow control with a slow acknowledgment channel: A study of TCP/IP performance," in Proc. IEEE Infocom 1997.

5. A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," in Proc. ACM SIGCOMM'89.