A Comparative Study of Different TCP Variants in Networks

Balveer Singh

Ph.D. Research Scholar, Computer Sc. Department.
NIMS University, Jaipur, India.

***Abstract*— TCP provides reliability to data transferring in all end-to-end data stream services on the internet. This protocol is utilized by major internet applications. TCP was originally created to handle the problem of network congestion collapse. This paper is prepared on the performance of different TCP variants to identify the best protocol variant for network expansion. In such context, a full comprehensive simulation environment is created for evaluating the comparative performance of TCP variants like TCP Tahoe, Reno, NewReno and TCP Vegas. From the results, TCP Reno is the most aggressive (least fair one), and highest amount of throughput. In the TCP NewReno it follows Reno's step by becoming the second most aggressive (second least fair), and second highest throughput. SACK is fair to Reno and NewReno, but it is competing with Vegas, Finally TCP Vegas shows the highest degree of fairness, TCP Vegas can adapt the changing bandwidth very well and it is robust against the fluctuation**

***Keywords*— TCP Tahoe, Reno, NewReno, SCAK, TCP Vegas.**

I. Introduction

TCP was originally made for wired links. On wired links there are very less chances of high delay [1] and corruption of data due to external parameters. Congestion is the main cause of packet loss on wired links. So, TCP was designed by keeping in mind the above parameters. As 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, NewReno and SACK and many more came into existence. TCP Tahoe is the first TCP variant which includes the first congestion control algorithm. This algorithm is developed by Jacobson and Karels in 1986. Based on the same concept presented by Jacobson and Karels, many more algorithms are then introduced. Following that, many enhancements and modifications are conducted on Tahoe, and leads to design and development of new TCP variants with different congestion window algorithms (Mo et al., 1999). The performance of TCP variants are directly affected by its own congestion control mechanisms, where the packet amount transferred over network connections is based on the work and the behaviour of the congestion control and its role in exploiting the capacity of the network path (Sarolahti, 2002). RFC 793 standardized the first TCP version with its basic configuration based on a scheme of window-based flow control. TCP Tahoe represents the second generation of TCP versions, which includes two new techniques, congestion avoidance and fast transmission. Reno is the third version of the first developed series, and it is standardized in RFC 2011, where the congestion control mechanism is further extended by fast recovery algorithm. However, versions of TCP Tahoe and Reno (and their variants) are not perfect in terms of throughput and impartiality among connections. Therefore, active research on TCP has been done, and many improvement mechanisms have been proposed [1]-[4]. Among them, a TCP Vegas version [5],[6] is one of the promising mechanisms because of its high performance. One important point is the underlying network assumed by TCP Vegas. When the original TCP Vegas was proposed [5], The RED (Random Early Detection) mechanism [8], was not consider in the operating network. TCP Vegas may or may not be effective when the router is equipped with the RED mechanism [8]. We therefore consider two packet scheduling mechanisms, the RED router as well as the conventional drop-tail router. One of the contributions in this paper is to derive analysis results of the throughput of TCP Tahoe, Reno, NewReno and Vegas in such a situation where they share the link with TCP variants. The accuracy of our analysis is validated by comparing the simulation results.

II. ANALYSIS OF TCP VARIANTS

A. TCP Variants

1) *TCP Tahoe:* A Tahoe [5] refers to the TCP congestion control algorithm which was suggested by Van Jacobson in his paper. TCP is based on a principle of conservation of packets, i.e. if the connection is running at the available bandwidth capacity then a packet is not injected into the network unless a packet is taken out as well. It implements this principle by using the acknowledgements to clock outgoing packets because an acknowledgement means that a packet was taken off the wire by the receiver. It also maintains a congestion window CWD to reflect the network capacity. It suggests that whenever a TCP connection starts or re-starts after a packet loss it should go through a procedure called slow-start. Reason for this procedure is that an initial burst might overwhelm the network and the connection might never get started. The congestion window size is multiplicatively increased that is it becomes double for each transmission until it encounters congestion. Slow start suggests that the sender set the congestion window to 1 and then for each ACK received it increase the CWD by 1. So in the first round trip time (RTT) we send 1 packet, in the second we send 2 and in the third we send 4. Thus we increase exponentially until we lose a packet which is a sign of congestion. When we encounter congestion we decrease our sending rate and we reduce congestion window to one, and start over again. The important thing is that Tahoe detects packet losses by timeouts. Sender is notified that congestion has occurred based on the packet loss.

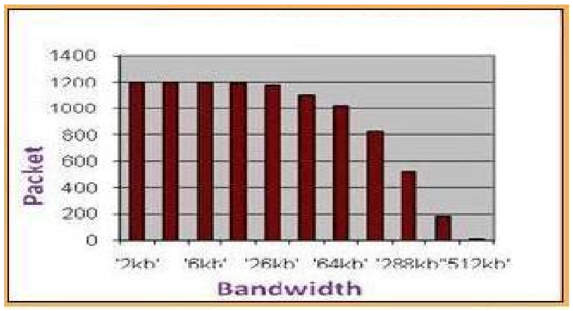

Fig. 1 Packet drop behavior for TCP Tahoe

2) *TCP Reno:* This RENO retains the basic principle of Tahoe, such as slow starts and the coarse grain retransmit timer [8]. However it adds some intelligence over it so that lost packets are detected earlier and the pipeline is not emptied every time a packet is lost. Reno requires that we receive immediate acknowledgement whenever a segment is received. The logic behind this is that whenever we receive a duplicate acknowledgment, then this duplicate acknowledgment could have been received if the next segment in sequence expected, has been delayed in the network and the segments reached there out of order or else that the packet is lost. If we receive a number of duplicate acknowledgements then it means that sufficient time have passed and even if the segment had taken a longer path, it should have gotten to the receiver by now. There is a very high probability that it was lost. So Reno suggests fast Re-transmit. Whenever we receive 3 duplicate ACK‘s we take it as a sign that the segment was lost, so we re-transmit the segment without waiting for timeout. Thus we manage to re-transmit the segment with the pipe almost full. Another modification that RENO makes is in that after a packet loss, it does not reduce the congestion window to 1. Since this empties the pipe. It enters into an algorithm which we call Fast-Recovery.

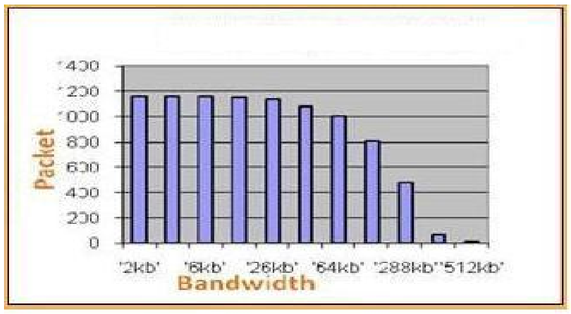

Fig. 2 Packet drop behaviour for TCP Reno

3) *TCP New Reno:* New RENO is a slight modification over TCP-RENO. It is able to detect multiple packet losses and thus is much more efficient that RENO in the event of multiple packet losses. Like RENO, New-RENO [7] also enters into fast retransmit when it receives multiple duplicate packets, however it differs from RENO in that it doesn‘t exit fast recovery until all the data which was out standing at the time it entered fast recovery is acknowledged. The fast recovery phase proceeds as in Reno, however when a fresh ACK is received then there are two cases If it ACK‘s all the segments which were outstanding when we entered fast recovery then it exits fast recovery and sets CWD to threshold value and continues congestion avoidance like Tahoe. If the ACK is a partial ACK then it deduces that the next segment in line was lost and it re-transmits that segment and sets the number of duplicate ACKS received to zero. It exits Fast recovery when all the data in the window is acknowledged.

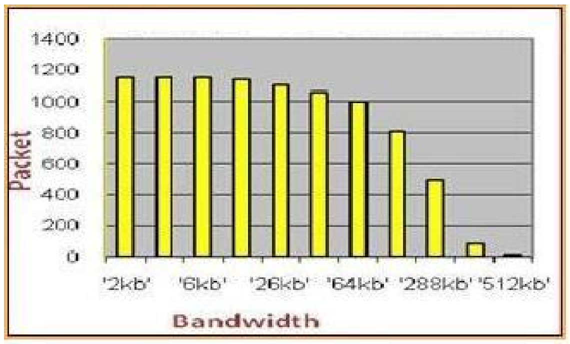

Fig. 3 Packet drop behaviour for TCP NewReno

4) *TCP Vegas:* Vegas is a TCP implementation which is a modification of Reno. It builds on the fact that proactive measure to encounter congestion is much more efficient than reactive ones. It tried to get around the problem of coarse grain timeouts by suggesting an algorithm which checks for timeouts at a very efficient schedule. Also it overcomes the problem of requiring enough duplicate acknowledgements to detect a packet loss, and it also suggests a modified slow start algorithm which prevents it from congesting the network. It does not depend solely on packet loss as a sign of congestion. It detects congestion before the packet losses occur. However it still retains the other mechanism of Reno and Tahoe, and a packet loss can still be detected by the coarse grain timeout of the other mechanisms fail.

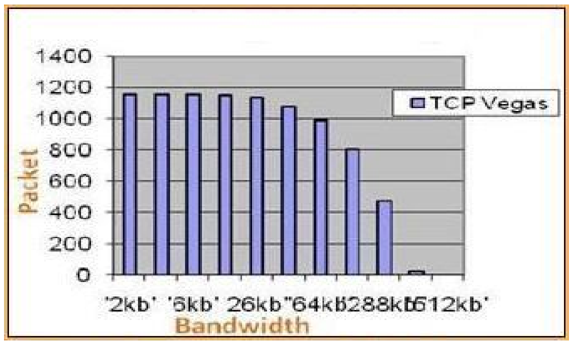

Fig. 4 Packet drop behavior for TCP vegas

III. PERFORMANCE EVALUATION

A. Packet Vs Bandwidth

Channel bandwidth has been varied each time and the number of packets was counted at destination during entire simulation period. As bandwidth of channel increases, number of packet received also increases by different magnitude for different variants. Highest number of packet is obtained for NewReno ASYM(NewReno in the asymmetry channel) because of its ‘Fast Recovery’ and ‘Open Loop’ congestion control mechanism. But TCP Tahoe, Reno, NewReno, sack1and Vegas behaves identically with bandwidth variation which generate a common curve.

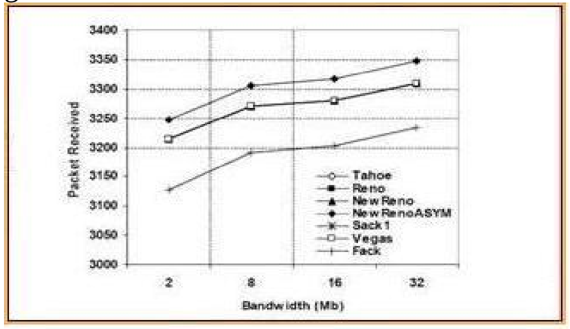

Fig. 5 Packet Vs bandwidth

B. Packet Vs Propagation Delay

Although number of packet received is inversely proportional to propagation delay, ‘Vegas’ has the best performance as depicted due to its improved retransmission technique. In Vegas, ‘acknowledgment’ is merged with ‘data packet’ (which is called piggybacking) instead of separate transmission .It saves fifty percent (50%) time than normal TCP implementation since 'acknowledgement' passing does not take extra time for it. That is why it can transmit more data.

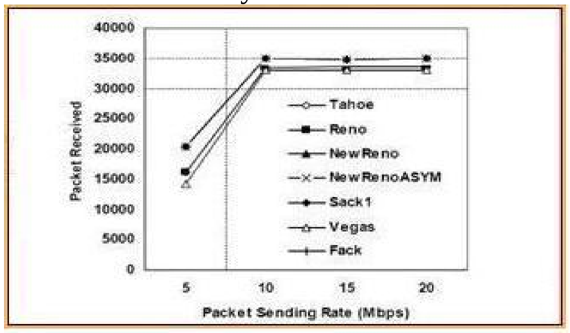

Fig. 6 Packet Vs Propagation delay

IV. ROUND TRIP TIME BEHAVIOR

Round Trip Time is a very important metric for measuring network performance of particular protocol. RTT is the amount of time that is needed for a packet to be sent to the far most nodes and receiving the acknowledgment for that packet. By gathering the data for TCP variants we depict the following figure Fig. 7. from where we can say that TCP Vegas performs better because the required time to receive acknowledgment from the destination node is less compared to other TCP variants.

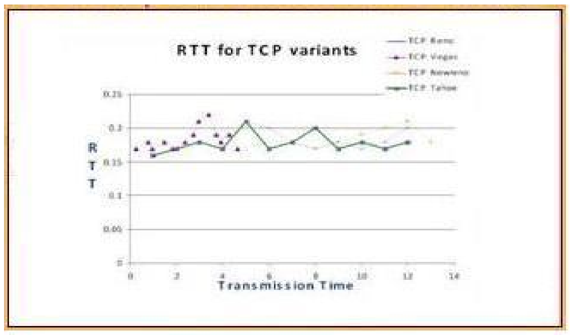

Fig. 7 RTT for TCP variants

We can say that TCP Vegas performs better because the required time to receive acknowledgment from the destination node is less compared to other TCP variants.

V. QUEUES FOR TCP PACKETS

TCP variants packets are waiting to the bottleneck node because of the narrow bottleneck bandwidth. We have shown queuing data behaviour according to transmission time. From the simulated data we plotted the following figure Fig. 8. For TCP Vegas a certain amount of data is waiting at the bottleneck link where as for TCP NewReno, the amount of data is waiting at the bottleneck link is near about 6000.As a result we can say that TCP Vegas provide the better performance compared to TCP Tahoe, TCP Reno and TCP NewReno.

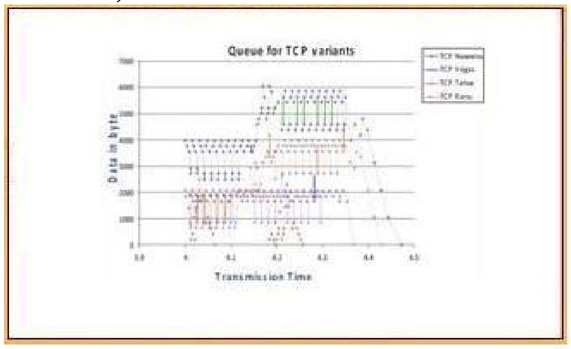

Fig. 8 Queue for TCP variants

VI. CONCLUSION

We have calculated the performance of four TCP variants; they are TCP Tahoe, TCP Reno, TCP New Reno and TCP Vegas. After analyzing the performance from simulated data. We have found that TCP Vegas is better than other TCP variants for sending data and information. In essence, TCP Vegas dynamically increases or decreases transmission of data according to the window size of sending packets of observed RTTs, whereas TCP Tahoe and Reno continue to increase their window size until packet loss is detected. Simulation, implementation and experimentation conclude that TCP Vegas can provide higher throughput than any other TCP variants. We conclude stating that the performance of TCP Vegas is much better than any other TCP variants where congestion is occurred in two point junction. Comparison of TCP variants with respect to other remaining network parameters would be an important future attempt. Such type of analysis is very helpful for selecting the appropriate TCP in a certain platform.

REFERENCES

[1] C.E. Perkins and E.M. Royer, *"Ad-hoc On-Demand distance vector routing"*, Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, LA, pp. 90-100, February 1999.

[2] D. Johnson, D. Maltz and Y. Hu., *"The dynamic source routing protocol for mobile ad hoc networks"*, IETF MANET Working Group, Internet Draft, 2003.

[3] M.K.J. Kumar and R.S. Rajesh, *"Performance analysis of MANET routing protocols in different mobility models"*, IJCSNS International Journal of Computer Science and Network Security, vol. 9 No.2, pp 22-29, Feb 2009

[4] K.Kathiravan, Dr. S. Thamarai Selvi, A.Selvam "*Tcp Performance Analysis For Mobile Ad Hoc Network Using Ondemand Routing Protocols*".

[5] JACOBSON, V. *Congestion avoidance and control*. In Proceedings of SIGCOMM '88 (Stanford, CA, Aug. 1988), ACM.

[6] Laxmi Subedi, Mohamadreza Najiminaini, and Ljiljana Trajkovi *Performance Evaluation of TCP Tahoe, Reno, Reno with SACK, and NewReno using OPNET Modeler*.

[7] S.Floyd, T.Henderson "*The New- Reno Modification to TCP's Fast Recovery Algorithm*" RFC 2582, Apr 1999.

[8] O. Ait-Hellal, E.Altman "*Analysis of TCP Reno and TCP Vegas*".

[9] Suhas Waghmare et. al "*Comparative Analysis of different TCP variants in a wireless environment*", 978-1-4244-8679-3/11 ©2011 IEEE

[10] Z.Wang and J. Crowcroft, "*Eliminating Periodic Packet Losses in 4.3--Tahoe BSD TCP congestion control*," ACM Computer Communication Review, vol.22, pp. 9-16, April 1992.

[11] Michel Perloff and Kurt Reiss, "*Improvements to TCP performance*," Communications of ACM, vol.38, pp. 90-100, February 1995.

[12] Matthew Mathis and Jamshid Mahdavi, "*Forward acknowledgment: Refining TCP congestion control*," ACM SIGCOMM Computer Communication Review, vol.26, pp. 281-291, October 1996.

[13] Go Hasegawa and Masayuki Murata and Hideo Miyahara, "*Fairness and stability of congestion control mechanisms of TCP*," in Proceedings of IEEE INFOCOM'99, pp. 1329-1336, March 1999.

[14] Lawrence S. Brakmo, Sean W. O'Malley, and Larry L. Peterson *TCP Vegas: New Techniques for Congestion Detection and Avoidance*, SIGCOMM' 94, 1994.