VOL. 11, NO. 1, JANUARY 2016 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

EXPLORING PERFORMANCE OF TCP VARIANTS IN WIRED AND WIRELESS NETWORKS

Sagarika Pattnaik and Ajit Kumar Nayak

ABSTRACT

In the current Internet with a high traffic load Transmission Control Protocol (TCP) is the key transport layer protocol. In spite of resource bottlenecks and largely unpredictable user access pattern, the TCP congestion control algorithm plays an important role to make the internet usable with successful transactions. There major implementations of TCP in use are Tahoe, Reno, New Reno and SACK. In this work, extensive simulations are performed to evaluate these TCP congestion control algorithms on different deployments to study various network parameters such as effective bandwidth utilization, fair resource allocation between different delay links. Analyses of the simulations are done and they go in the favor of TCP Sack in both wired and wireless environment. The variants show some amount of difference in their performance when they are compared in wired and wireless environment. The implementations suggest mechanism for determining when a segment should re-transmit and how should the sender behave when it encounters congestion and what pattern of transmission it should follow to avoid congestion.

Keywords: TCP, Tahoe, Reno, Newreno, sack.

1. INTRODUCTION

TCP implements a window based flow control mechanism. Window based protocol means that the so called current window size defines a strict upper bound on the amount of unacknowledged data that can be in transit between a given sender receiver pair. Originally TCP's flow control was governed simply by the maximum allowed window size advertised by the receiver and the policy that allowed the sender to send new packets only after receiving the acknowledgement for the previous packet. In wired network congestion is the main cause for packet loss. Each TCP packet is associated with a sequence number, and only successfully received packets are acknowledged to the sender by sending corresponding ACK (acknowledgement) packets with sequence number of next expected packets. The reception of out of order packets indicates failure. Towards the 80's when the traffic in the Internet started growing congestion collapse occurred and it was realized, that special congestion control algorithms would be required to prevent the TCP senders from overrunning the resources of the network. In 1988, Tahoe TCP [2] was released including three congestion control algorithms; namely slow start, congestion avoidance and fast retransmit. In 1990 Reno TCP [3], provided one more algorithm called fast recovery was released. Another variant New-Reno [4, 17] was developed that improves fast retransmit and fast recovery algorithm and does not keep the pipe empty. Again another variant TCP Sack [5, 9, 15] that uses selective acknowledgement concepts have been developed.

The rest of the paper is organized as follows:

Section II describes different characteristics of TCP variants. Section III provides definition of important terms. Section IV describes the algorithms slow start, congestion avoidance and fast retransmits and fast recovery of TCP. Section V discusses the related works on this topic by different authors. Section VI does a performance study of the TCP variants in wired and wireless environment. Simulations on the variants in different scenarios and their results are evaluated in this section. Section VII draws a conclusion of all the studies done on the variants of TCP.

2. CHARACTERISTICS OF TCP VARIANTS

The Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP) are both IP transport-layer protocols. UDP is a lightweight protocol that allows applications to make direct use of the unreliable datagram service provided by the underlying IP service. TCP provides a reliable data-transfer service, and is used for both bulk data transfer and interactive data applications. TCP is the major transport protocol in use in most IP networks, and supports the transfer of over 90 percent of all traffic across the public Internet today.

TCP is the embodiment of reliable end-to-end transmission functionality in the overall Internet architecture. All the functionality required to take a simple base of IP datagram delivery and build upon this a control model that implements reliability, sequencing, flow control, and data streaming is embedded within TCP. TCP provides a communication channel between processes on each host system. The channel is reliable, full-duplex, and streaming. To achieve this functionality, the TCP drivers break up the session data stream into discrete segments, and attach a TCP header to each segment. An IP header is attached to this TCP packet, and the composite packet is then passed to the network for delivery. This TCP header has numerous fields that are used to support the intended TCP functionality. TCP has the following functional characteristics:

Unicast protocol: TCP is based on a unicast network model, and supports data exchange between precisely two parties. It does not support broadcast or multicast network models.

Connection state: TCP uses synchronized state between the two endpoints. This synchronized state is set up as part of an initial connection process, so TCP can be regarded as a connection-oriented protocol. Each local state transition is communicated to, and acknowledged by, the remote party.

Reliability: Reliability implies that the stream of octets passed to the TCP driver at one end of the connection will be transmitted across the network so that the stream is presented to the remote process as the same sequence of octets, in the same order as that generated by the sender.

This implies that the protocol detects when segments from data stream have been discarded by the network, reordered, duplicated, or corrupted. Where necessary, the sender will retransmit damaged segments so as to allow the receiver to reconstruct the original data stream. That is a TCP sender must maintain a local copy of all transmitted data until it receives an indication that the receiver has completed an accurate transfer of the data.

Full duplex: TCP allows both parties to send and receive data within the context of the single TCP connection.

Streaming: Although TCP uses a packet structure for network transmission, TCP is a true streaming protocol, and application-level network operations are not transparent.

Rate adaptation: TCP is also a rate-adaptive protocol, in that the rate of data transfer is intended to adapt to the prevailing load conditions within the network and adapt to the processing capacity of the receiver. There is no predetermined TCP data-transfer rate.

3. DEFINITIONS

Segment: A segment is any TCP/IP data or acknowledgment packet (or both).

Sender Maximum Segment Size (SMSS): The SMSS is the size of the largest segment that the sender can transmit. The size does not include the TCP/IP headers and options.

Receiver Maximum Segment Size (RMSS): Size of the largest segment that the receiver is willing to accept.

Full-sized segment: a segment that contains the maximum number of data bytes permitted.

Receiver Window (rwnd): The most recently advertised receiver window.

Congestion Window (cwnd): A TCP state variable that limits the amount of data a TCP can send. At any given time, a TCP must not send data with a sequence number higher than the sum of the highest acknowledged sequence number and the minimum of cwnd and rwnd.

Initial Window (IW): The initial window is the size of the sender's congestion window after the three-way handshake is completed.

Loss Window (LW): The loss window is the size of the congestion window after a TCP sender detects loss using its retransmission timer.

4. ALGORITHMS

Different algorithms that TCP adopts are discussed in this section.

4.1 Slow start and congestion avoidance

The Slow-start algorithm [8, 9] is used at the beginning of a transfer or after repairing a loss detected by the "retransmission timer" to slowly probe the network to determine its available capacity.

IW is the initial *cwnd* with value<= *2 * SMSS* bytes and must not be more than *2* segments. The initial value of *ssthresh* may be arbitrarily high (i.e., the size of the advertised window), but it may be reduced in response to congestion. Slow-start algorithm is used when *cwnd* < *ssthresh*. When *cwnd* > *ssthresh*, the congestion avoidance algorithm is used and when *cwnd* = *ssthresh* the sender may use either of them. During *slow-start*, a *TCP* increments cwnd by at most *SMSS* bytes for each *ACK* received that acknowledges new data. During *congestion avoidance*, *cwnd* is incremented by 1 full-sized segment per *round-trip time* (*RTT*). *Slow-start* ends when *cwnd* exceeds *ssthresh* or when congestion is detected.

"Congestion avoidance" ends when congestion is detected. The formula commonly used to update *cwnd* during congestion avoidance is

$$cwnd \mathrel{+}= smss * smss / cwnd$$

This adjustment is executed on every incoming non-duplicated ACK. It provides an acceptable approximation to the underlying principle of increasing *cwnd* by 1 full-sized segment per RTT. If the formula yields *0*, the result should be rounded up to *1 byte*. Another acceptable way to increase *cwnd* during *congestion avoidance* is to count the number of bytes that have been acknowledged for new data. When the number of bytes acknowledged reaches *cwnd*, the *cwnd* is incremented by up to *SMSS* bytes.

When a *TCP* sender detects segment loss using the "retransmission timer", the value of *ssthresh* must be set to

*ssthresh = max (FlightSize/2, 2*SMSS),*

where *FlightSize* is the amount of outstanding data in the network.

Furthermore, upon a timeout *cwnd* must be set to no more that the loss window *LW*, which equals 1 full-sized segment. Therefore, after retransmitting the dropped segment the TCP sender uses the slow-start algorithm to increase the window from 1 full-sized segment to the new value of ssthresh, at which point congestion avoidance again takes over.

4.2 Fast retransmit/fast recovery

A TCP receiver should send an immediate *duplicate ACK* when an *out-of-order* segment arrives; this is to inform that a segment was received *out-of-order* and which sequence number is expected (caused by dropping, reordering or duplication in the network). In addition, a *TCP* receiver should send an immediate *ACK* when the incoming segment fills in all or part of a gap in the sequence space. This will generate more timely information for the sender recovery. The *TCP* sender should use the *fast retransmit algorithm* to detect and repair loss based on *incoming duplicate ACKs.*

After the arrival of *3 duplicate ACKs* (4 identical *ACKs* without the arrival of any other intervening packet), *TCP* performs a retransmission of what appears to be the missing segment, without waiting for the retransmission timer to expire.

After the *fast retransmit* sends what appears to be the missing segment, the *fast recovery* [10, 14] algorithm governs the transmission of new data until a *non-duplicate ACK* arrives. Duplicate *ACKs* indicate that data is flowing, then the *slow-start* algorithm is not applied. Since the receiver can only generate a *duplicate ACK* when a segment has arrived, that segment has left the network and is in the receiver's buffer, so we know it is no longer consuming network resources. Furthermore, since the *ACK clock* is preserved, the *TCP* sender can continue to transmit new segments, using a reduced *cwnd*.

5. RELATED WORKS

Fall and Floyd [10] in their work have illustrated some of the benefits of adding selective acknowledgement (SACK) to TCP. Current implementations of TCP use an acknowledgement number field that contain a cumulative acknowledgement, indicating the TCP receiver has received all the data up to the indicated byte. A selective acknowledgement allows receivers to additionally report non-sequential data they have received. When coupled with selective retransmission policy implemented in TCP senders, considerable savings can be achieved. Mao Wang, *et al* [11] have found out the effects of different routing protocols on different variants in different scenarios. Each kind of routing protocols has its own merits and flaws in some sense, so they should be applied in different conditions with respect to the area required to be covered. With the help of routing protocols, ad-hoc networks enable us to choose route in a flexible manner. Nashiry, Sumi, *et al,* [12] have worked on TCP performance Over Mobile IP Wired cum Wireless Network. M Kaur and S Singh [17] have studied the behavior of TCP over MANETs.

Reliable Transport Protocol like TCP has served well the Internet where the packet loses are mainly due to congestion, but is not ready for Mobile IP Wired cum Wireless Network where the significant loss are due to bit errors and handoffs. This paper has investigated the performance of TCP among the various TCP variants. The simulation results of this paper suggest that one TCP sender with one TCP receiver shows the best results in Mobile IP Networks.

6. SIMULATION AND RESULT EVALUATION

6.1 Simulation environment

Simulations for congestion control protocols have been performed in both wired and wireless network.

Performance measures considered in Wired Network

Effective resource utilization: It deals with how the variants are utilizing the band width after encountering a certain amount of loss.

Fairness between connections with different delays: It deals with the response of the variants to different delay factors.

Congestion window size variation: It deals with the response of the variants to congestion.

6.1.1 Effective resource utilization in Wired Network

We have provided a comparative study on the efficiency of the four TCP variants (Tahoe, Reno, New Reno and Sack) with respect to better utilization of the resources in a wired network scenario.

Simulation scenario for bandwidth utilization

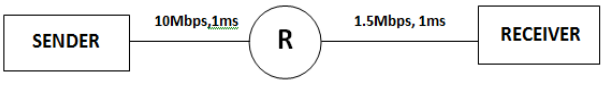

Figure-1. Scenario of bandwidth utilization.

As shown in Figure-1, three nodes are considered out of which two nodes are sender and receiver nodes represented as rectangles and the third one is the router as a circle. The router uses a finite drop tail queue with a buffer size of 6. The links are labeled with their bandwidth capacity and delay. The variants of TCP are analysed using ftp traffic. An error model is added in the slow link between the gateway and the receiver.

Result evaluation

The result of the simulation is presented in Figure-2. As can be noticed, for small amount of loss (1%) the TCP flavors do not exhibit much difference. However, for a loss of 5%, TCP Sack shows higher bandwidth utilization than Tahoe and Reno. New Reno shows better resource utilization in under load condition. In a heavy load condition again TCP Sack shows a better performance than the other three variants.

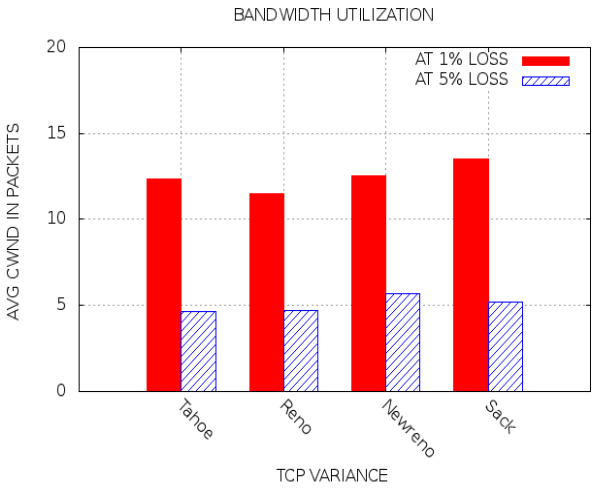

Figure-2. Bandwidth utilization.

The graph in Figure-3 shows the result of the simulation by taking different loss modules and testing the effect on different variants.

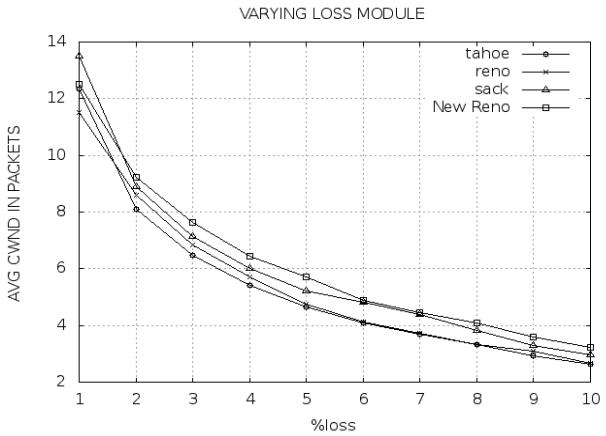

Figure-3. Congestion window vs Loss.

6.1.2 Fairness between connections with different delays

The scenario in Figure-4 uses six nodes for the evaluation. S1 and S2 are source nodes, S3 and S4 are receiver nodes and R1 and R2 are two routers (finite buffer switches). The propagation delay of the link that connects R2 and S4 is denoted by X.The links are labeled with their capacities and propagation delays. Same version TCP is used on one bottleneck link. Test is performed with respect to fair share the bandwidth while maintaining different connection delays.

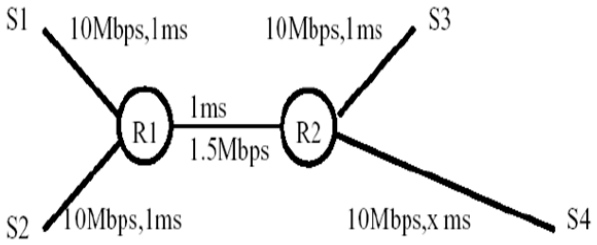

Figure-4. Scenario for fairness check.

Table-1. Showing average cwnd (Congestion Window) for varying delays.

Delay(X) (in ms)	**Avgcwnd** (in pkts)
1	24.6744
22	23.1866
50	21.1026
120	16.8212
200	13.1099

Result evaluation

The four variants respond to the varying delay in a similar manner. The Table-1 shows the result of simulation as average congestion window at different delays. The behavior of the variants is checked for both low and high delay conditions. At higher delay the variants show a biased behavior. At shorter delay the variants occupy a larger bandwidth than at higher delay. The reason is while a source does not detect any congestion it continues to increase its window size by 1 during 1 round trip time(RTT), thus connections with shorter delays can increase their window size faster than connections with longer delays and occupy a larger bandwidth as can be seen from Table-1.

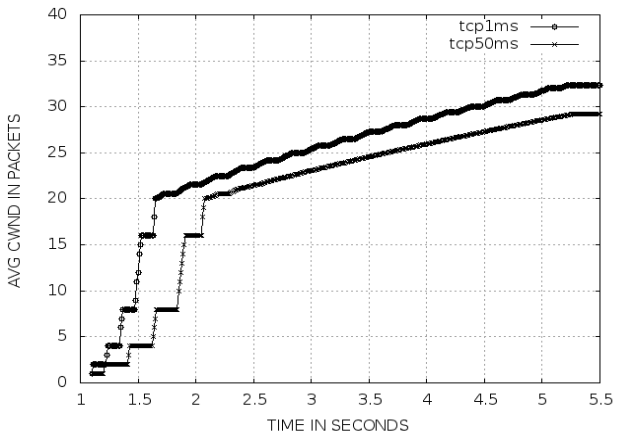

Figure-5. Varying delay conditions.

Tahoe, Reno, New Reno and Sack show a biased behavior as delay changes. Source does not detect congestion. Connections with shorter delays can update their window size faster and occupy larger bandwidth than connections with longer delays. Sack is following the same window increasing policy. It is also biased as delay changes.

6.1.3 Congestion window size variation

The TCP variants show difference in their behavior only if there is a packet drop. For one packet loss the TCP flavors do not show much difference. For more than 1 packet loss in one congestion window the variants show their efficiency according to their respective algorithms.

Simulation scenario

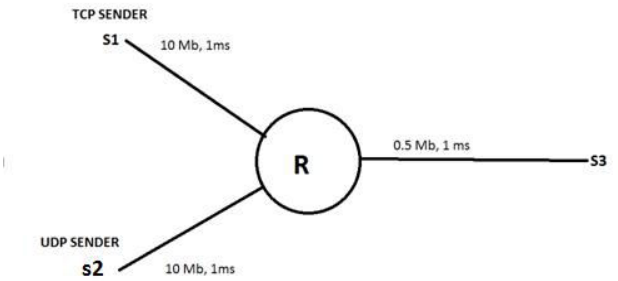

Figure-6. Scenario for evaluating congestion window size variation.

The scenario prepared for evaluating congestion window size variation uses two source nodes (S1, S2) as shown in Figure-6, one receiver node(S3) and one finite buffer drop tail gate way(R) indicated by a circle. Source S1 uses TCP protocol and ftp as traffic. Source S2 uses UDP protocol and CBR traffic. The links are labeled with their respective bandwidth and delay.

Result evaluation

From the simulations we found that congestion window of TCP Tahoe and Reno drops to zero and moves to slow-start when more than one packet are dropped in one window as shown in Figure-7. In Figure-8 New Reno shows a similar behavior as Reno. TCP SACK maintains the same window size as the value after the first packet drop and returns to a higher window size than both Reno and Tahoe and is presented in Fig. 9.TCP SACK shows a better performance as can be seen from Figure-10. We can say that the algorithm of TCP SACK performs better in the case of more than one packet is dropped in one window.

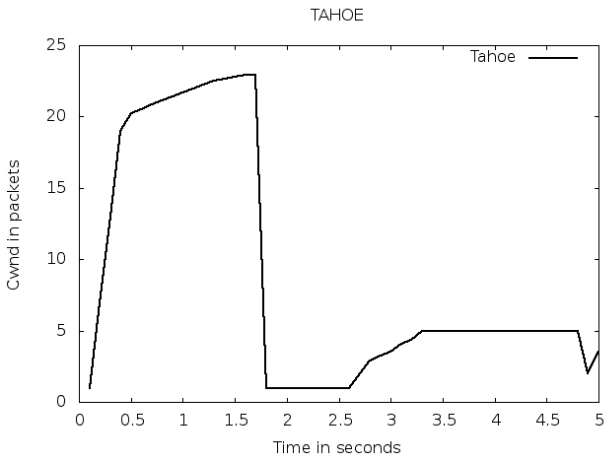

Figure-7. Congestion window of Tahoe TCP on packet loss.

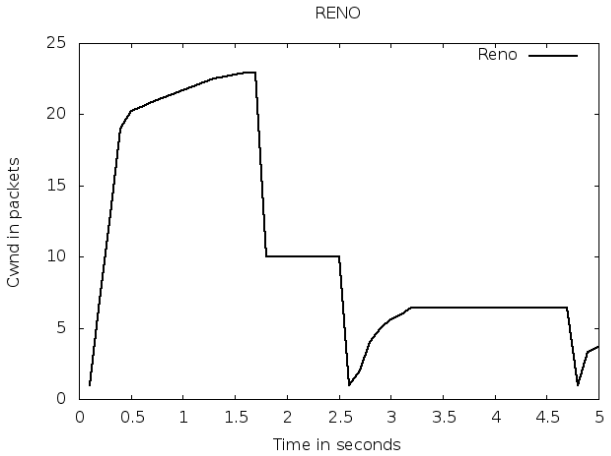

Figure-8. Congestion window of Reno TCP on packet loss.

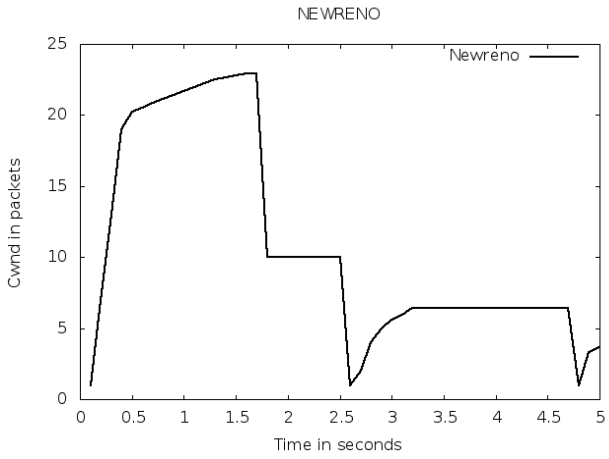

Figure-9. Congestion window of Newreno TCP on packet loss.

6.2 Performance of TCP variants in wireless network

Differences between the characteristics of wired and wireless networks have significant impact on TCP performance. TCP was designed and optimized to perform well in wired networks. Wireless links are subjected to considerable packet losses due to link errors, delay variations, and long sudden delays violate TCP's essential design assumptions. Improving TCP performance in wireless networks has been an ongoing research activity [12]. Taking into consideration of the current demand TCP was implemented in the Wireless environment. Here we have undergone a performance study of the TCP variants in the wireless environment and provided a detailed simulation result of their behavior in different network scenarios.

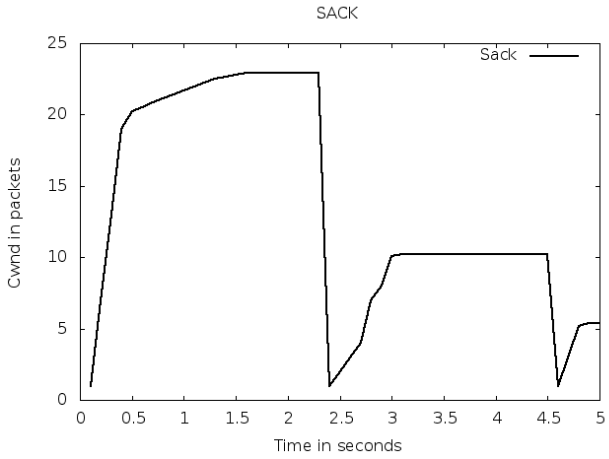

Figure-10. Congestion window of Sack Tcp on packet loss.

6.2.1 Protocols in wireless network

There are many routing protocols are designed for wireless networks in last few decades. They are broadly classified into two categories, the first one being the on-demand and the other category of protocols are called table driven. We have chosen a table driven protocol for our simulations calledDestination Sequenced Distance Vector (DSDV) which is an extension of Distance Vector protocol used for wired networks.

DSDV: Destination sequenced distance vector

DSDV is a table driven routing protocol developed for wireless networks. Each node has a routing table that indicates for each destination, which is the next hop and number of hops to the destination. Each node periodically broadcasts routing updates. A sequence number is used to tag each route.

When a neighbour node receives a routing information update, it compares the freshness of the update to the currently used routing information by comparing the associated sequence numbers, and the most recent information is used to update its routing table. However, if the sequence numbers of the old and the new information are the same, it selects the better routing information according to an applied metric. Alternatively, the receiver of the routing information update message may wait for some time until it receives another update message about the same destination, and if this new message has a better metric value than that of the already received one, it may use it instead.

6.2.2 Performance measures in wireless network

The parameters like fairness and congestion window size variation have been studied. The type of queue used is drop tail with length of 50. Anomni antenna with two ray ground and Mac/802_11 as MAC layer protocol is used for the simulation.

Fairness of TCP variants with different delays

In this simulation, we have used the simple scenario as shown in Figure-11. Intentionally we kept the network size small to make our representation clear. However, it may be scaled to any size. The network contains four nodes where node 0 and node 3 are the senders and node 1 is the receiving node. Node 2 works as the router. TCP is used as the transport protocolat node 0 for sending and UDP is used for node 3.Varying delay is introduced at the link layer.The wireless links are shown by dotted lines.

Simulation scenario

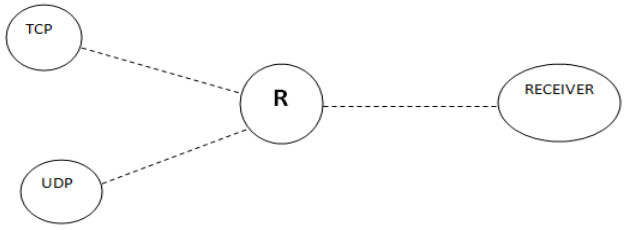

Figure-11. Wireless environment scenario.

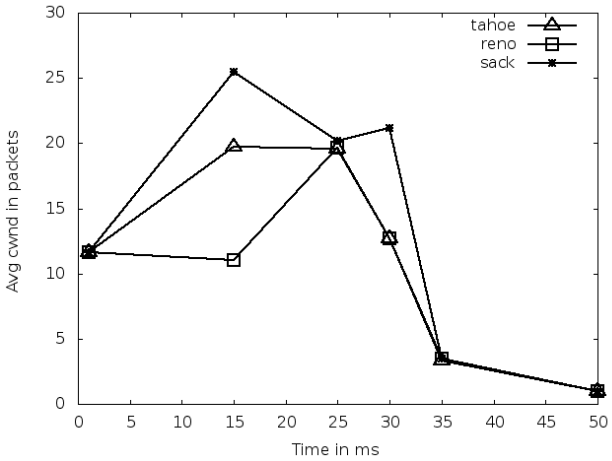

Figure-12. Average congestion window for varying delays.

Result evaluation

It is observed from Figure-12 that in wireless environment again Sack is showing a better performance as delay changes. All the four variants are biased, but among them Sack is having its average congestion window higher than the others at different delays. In case of wired network the four variants were biased in a similar manner to delay changes but in wireless the case is not the same. At low delay and at delay at 50 ms all the four variants are equally biased.

Congestion window size variation

The same simulation topology as in Figure-11 is used for this simulation also.

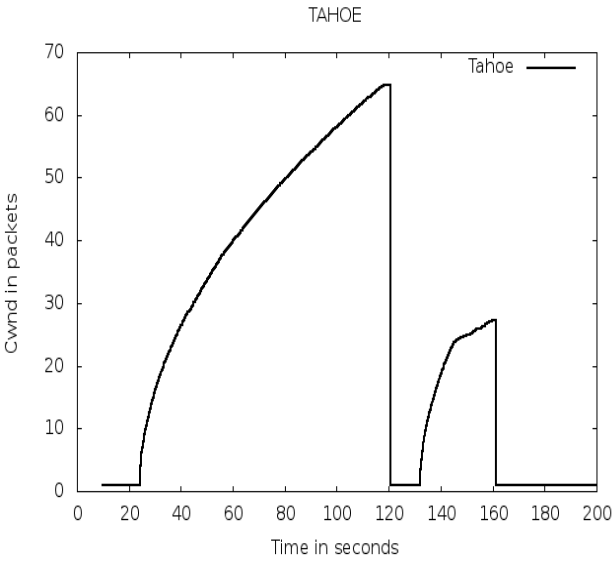

Figure-13. Congestion window of Tahoe TCP.

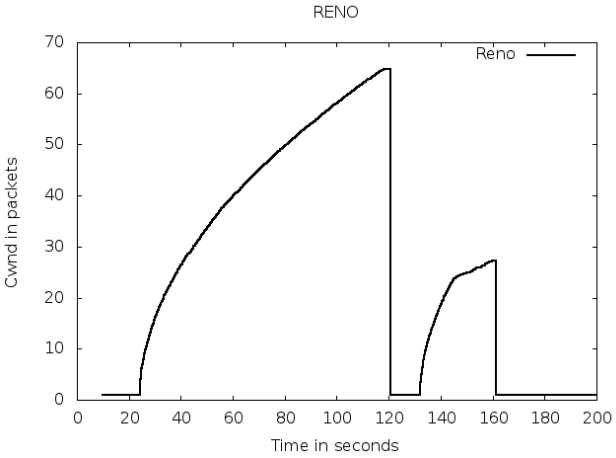

Figure-14. Congestion window of Reno TCP.

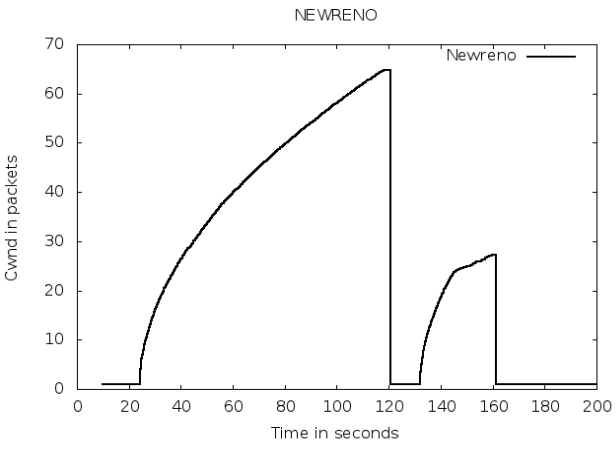

Figure-15. Congestion window of Newreno TCP.

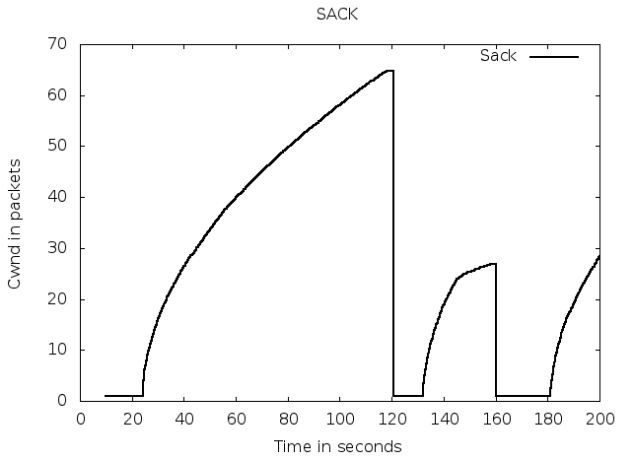

Figure-16. Congestion window of Sack TCP.

Result evaluation

The result of congestion window size simulation are provided in Figures 13-16. The congestion window of Sack TCP shows a better behavior than the other variants. It is able to build up its congestion window to a higher value after packet drop than other variants. Sack TCP due to its algorithm is able to handle packet drop in a better way.

7. CONCLUSIONS

After performing a simulation study of the different variants from different aspects in different scenarios both in wired and wireless environment, we have come to a conclusion that TCP with selective acknowledgement (SACK) is most promising. But in under load condition in wired environment the variant New Reno shows better bandwidth utilization than the other variants. It is also seen that the variants are also biased with longer delays. Again the SACK TCP is performing well in handling packet losses as can be seen from their window size variation. In wireless environment the variants are showing some differences in their behavior than the simulations done in wired environment. In wired environment all TCP variants are equally biased but in wireless network the variants show varying responses to the changing delay.

In this paper we have explored the fundamental restrictions imposed by the lack of selective acknowledgments in TCP. We assume that the addition of selective acknowledgments to TCP will open the way to further developments of the TCP protocol. The addition of selective acknowledgments allows additional improvements to TCP, in addition to improving the congestion control behavior when multiple packets are dropped in one window of data. TCP with SACK shows that SACK and explicit wireless loss notification both result in substantial performance improvements for TCP over lossy links. Several researchers are exploring the use of SACK, coupled with the explicit notification of non-congestion-related losses, for lossy environments such as satellite links.

The SACK option will allow the TCP protocol to be more intelligent in other ways as well. SACK opens the way to further advances of TCP's congestion control algorithms. The SACK implementation in our simulator could be improved in its robustness to reordered packets during Fast Recovery. Works are still to be done regarding the loss of re-transmitted packets. Researches has to be done regarding avoidance of unnecessary retransmit time outs to correct unnecessary fast retransmits resulting from reordered or delayed packets , and to assist the development of viable mechanism for corruption notification. Further research is open in the fields like CBR or HTTP traffic instead of FTP traffic, different TCP variants such as TCP Vegas, with more packet drops.

REFERENCES

[1] Fall and K.Vardhan, The ns Manual, December 13, 2003, http://www.isi.edu/nsnam/ns/.

[2] V. Jacobson and M.J. Karels. 1988. Congestion avoidance and control. Proceedings of Sigcomm `88, 18(4) standford, CA.

[3] V. Jacobson. 19990. Modified TCP Congestion Control and Avoidance Alogrithms. Technical Report.

[4] S. Floyd and T. Henderson. 1999. The New-Reno Modification to TCP's Fast Recovery Algorithm. RFC 2582.

[5] S. Floyd, J. Mahdavi, M. Mathis and M. Podolsky. 2000. An Extension to the Selective Acknowledgement (SACK) Option for TCP. RFC2883.

[6] M.Allman and V. Paxson. 2009. TCP congestion Control. RFC 5681.

[7] S. Floyd. 2001. A retport on recent developments in TCP congestion control. Communications Magazine IEEE. 39(4).

[8] A Abed, Md Ismail and K. Jumari. 2012. Exploration and Evaluation of traditional TCP Congestion Control techniques. Journal of King Saud University.

[9] S.Floyd. 2004. New Reno modification to TCP's Fast Recovery Algorithm. Request for Comments: 3782.

[10] Kevin Fall and Sally Floyd. 1996. Simulation based comparison of TCP Tahoe, Reno and Sack. Newsletter, ACM, SIGCOMM, Computer communication review. 26(3).

[11] Ren Mao, Haobing Wang, Li, Li and Fei Ye. 2001. TCP over Ad Hoc Networks (NS2 simulation analysis).

[12] Md.Asif Nashiry, Shauli Sarmin Sumi, Subrata Kumar Das and Md. Kamrul Islam. 2012. TCP performance Over Mobile IP Wired cum Wireless Network. Global Journal of Computer Science and Technology. Vol. 12.

[13] Hui (Hilary) Zhang and Zhengbing Bian. 2002. Evaluation of different TCP congestion Control Algorithms using NS2.

[14] Matthew Mathis, Jamshid Mahdavi, Sally Floyd and Allyn Romanow. 1996. TCP Selective Acknowledgment Options. RFC 2018.

[15] Eitan Altman and Tania Jiminez. 2003. NS simulator for the beginners.

[16] J.P. Van den Burg. 2010. New Reno and Sack TCP implemented in UMTS networks; A performance analysis. University of Twente.

[17] Manpreet Kaur and Sandeep Singh Kang. 2014. A survey of routing protocols using TCP variants over MANETs. International Journal of Engineering and Computer Science.