I J C T A, 8(5), 2015, pp. 1743-1757

Performance Analysis and Comparison of Congestion Control in Wired and Wireless Environment

M. Jothishkumar*, R. Baskaran**

Abstract: TCP is the most widely used protocol in the wired and wireless environment despite to the use of resource bottleneck. The algorithm of TCP congestion control is the main reason we can use the wireless Internet successfully today with a large unpredictable user access patterns.In the last decade, many congestion control algorthims have been proposed to improve the classic TCP congestion control.Among this few implementations are TCP Reno, New Reno, Vegas and SACK.Each of the algorthims simulation scenarios are carefully designed using NS2 inorder to investigate packet arrival, congestion window size, average queue size with current queue size and its drop probability.A comparitive analysis between different implementations of TCP congestion control among in the wired network shows TCP Vegas provides better perfromance in all parameters while TCP Reno and SACK are minimal in dropping probability. In wireless network TCP Reno maintains a maximum congestion window size.

Keywords: TCP Reno, New Reno, Vegas and SACK, Congestion control

1. INTRODUCTION

Over the past few decades, the internet traffic has increased to substantially and now it was important to take into consideration the TCP congestion control a fair control system into a network control.TCP is a reliable connection oriented [1] as an end-to-end protocol.TCP design philosophy has evolved considerably to develop an effective packet switching from one protocol to another which is robust and reliable.TCP mechanism is based on the receiver's requirement in order to acknowledge the packets they receive reliably. The congestion in the network leads the routers in dropping small percentage of packets (due to network error).

2. BACKGROUND AND MOTIVATION

TCP congestion control [2] is based on dynamic window adjustment. The connection begins in slow start phase with the congestion window. It twice the round trip time until the size reaches the window of a slow-start threshold. After this the window will be improved slowly at a rate of about one packet to each round trip time but it will not exceed the maximum threshold size that is advertised by the receiver. Due to congestion the network experience large queue and delays consequently the sender must retransmit the data's in order to compensate the packet losses. The packet loss is detected from the sender end when too many sources attempt to send data at high rates. It also drops the threshold to half of the present value and the window size drops to the one maximum segment size.

2.1. Problem of Congestion

Congestion [3] is a situation in Communication Networks in which too many packets are present in a part of the subnet and its subsequently leads to degrade performance. Congestion in a network may occur when the load on the network is greater than the capacity of the network.

* Research Scholar, *Email: saijoetvm@gmail.com*

** Associate Professor Department of Computer Science and Engineering Anna University, Chennai Tamil Nadu, *Email: baskaran.ramachandran@gmail.com*

Congestion Control [4] refers to techniques that can prevent either before it happens, or remove congestion after it has happened. Congestion in a network leads to performance degradation. It occurs when the load on the network is more than the capacity of the network (i.e) based on the number of packet sent. Congestion is a condition communication network where too many packets are present in a part of the subnet. The congestion in the network is sensed by the congestion control algorithms.

Engrossment of Congestion Control

Convergence, Responsiveness, Aggressiveness and smoothness are the prerequisites of congestion control algorithm. For example in internet real time applications, smoothness is desirable for a steady state of some applications in reducing its transmission rate to its fair share promptly. Similarly convergence speed is related to the aggressiveness and responsiveness indices to converge faster. In particular when there is a sudden increase in available bandwidth, it is desirable that the connection acquires the extra bandwidth quickly and responsiveness probes to increase or decrease its window size.

2.2. Slow-Start and Congestion Avoidance

Slow-start, improperly called in usage is one of the algorithms implemented by TCP entities. It is used to control the amount of data sent over the network to detect after the packet loss or at the beginning of a data transmission. The Slow-start [5] actually increases exponentially the size of the congestion and also to fill the data as soon as possible in a transmission channel.

Figure 1 shows the graphical outlook of slow-start and congestion avoidance algorithms during the transmission of datas.A retransmission timer is used for every packet when the timeout signals the loss of the packet. This retransmission is done by halving the slow-start threshold. Congestion avoidance algorithm is employed linearly by adding up to one MSS but not less than one byte after the congestion window has reached the threshold value.

2.3. Network Simulator

In this paper, various algorithms such as New Reno, Reno, Vegas and SACK are implemented by different TCP congestion control used in network simulator.NS-2 is a widely used tool to stimulate the behavior of wired and wireless networks. It also has the sources of C codes for the protocol libraries which users can edit/modify to develop custom algorthims.NS-2 is an object-oriented [5] driven event in network simulator.

2.4. TCP Reno

TCP Reno [6] retains the basic principles of TCP entities similar as Tahoe [7] such as Slow Start, Congestion Avoidance and Fast Retransmit. During the reduction of congestion window TCP Reno is not as aggressive

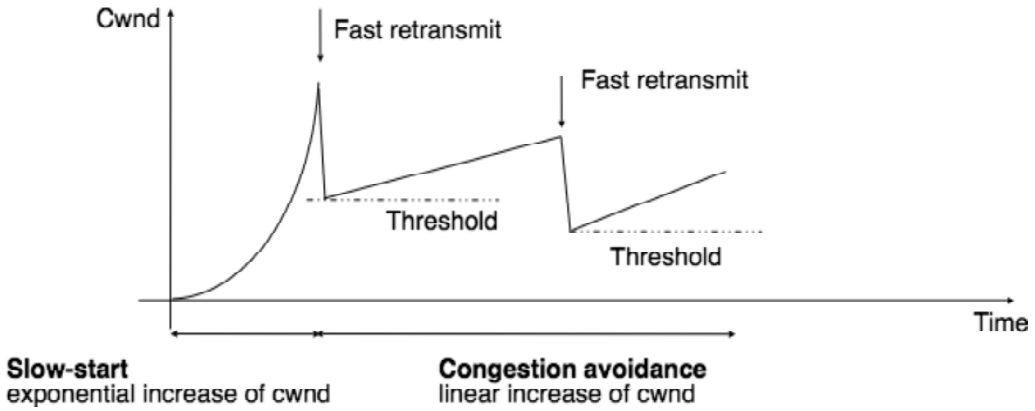

Figure 1: Slow Start and Congestion Avoidance[17]

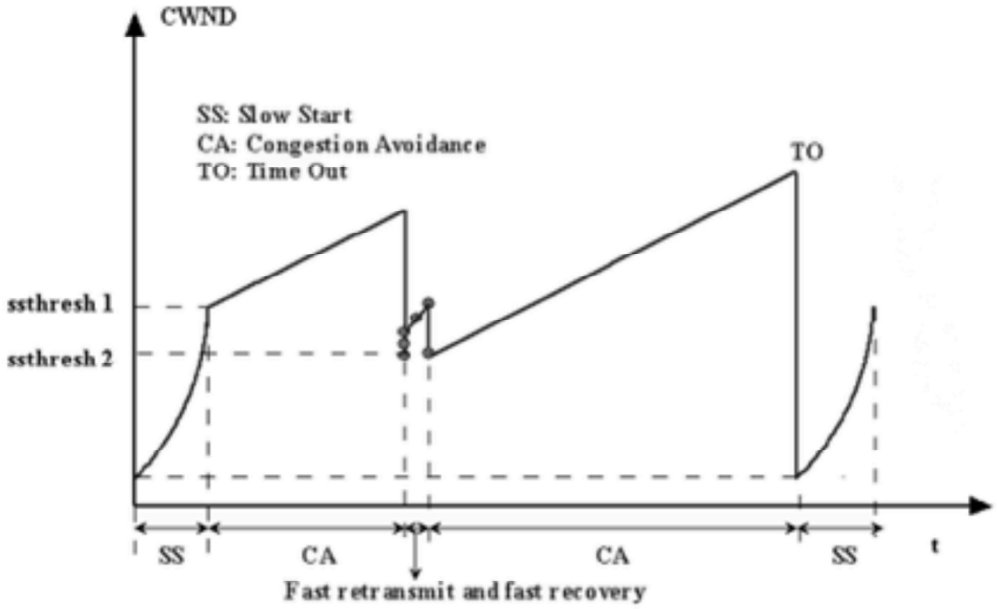

Figure 2: TCP Reno[18]

as Tahoe but implements a fast recovery algorthim.when the congestion window drop to one on the arrival of a three duplicate acknowledgement.

Figure 2 shows the arrival of 3 duplicate ACKs corresponds to light congestion in the network. If as ACK times out, slow-start is used.TCP Reno will halve the congestion window since there is no need for the congestion window to drop down drastically. Now we can set the slow-start threshold equal to the new congestion window in order to perform a fast transmit and this phase is called fast recovery. When the congestion window drop to one, the arrival of a 3 duplicate ACK will be considered as an extreme precaution. When the timeout, the losses of packets can be found out by considering the value of cwnd *(t)* and *ssth(t)* as follows.

$$\text{cwnd}(t) = 1,\ \text{ssth}(t) = (\text{cwnd}(t))/2 \quad (1)$$

cwnd(t)-used by source to limit how much data is allowed to have in transit at a given time

ssth (t)-value of the threshold at which TCP passes from the phase of slow starts in the phase of avoidance of overload.

2.5. TCP New Reno

TCP New Reno [8] is used to improve the retransmission during the fast-recovery phase of TCP Reno. During a fast recovery, for every new unsent packet from the end of the congestion window a duplicate ACK is returned to TCP New Reno inorder to keep the transmit window full. The sender assumes the partial progress of ACK points to a new hole in the sequence and the next ACK beyond the packet number is sent. The difference in the fast-recovery phase allows multiple re-transmissions in New Reno. This retransmission can be set as.

$$\text{ssthresh} = \max(\text{FlightSize}/2, 2*\text{MSS}) \quad (2)$$

$$\text{cwnd} = \text{ssthresh} + 3 * \text{MSS} \quad (3)$$

To every additional duplicate ACK received, we should increase cwnd by MSS.

2.6. TCP Vegas

TCP Vegas is a TCP congestion window avoidance avoidance algorithm emphasizing packet delay as a signal to help in determining the rate of send packets. This algorithm depends heavily on accurate calculation

of the Base RTT value. It can be viewed, when the network is not congested, the actual flow rate will be close to the expected flow rate. This difference in the flow rate can be easily transformed in to the difference between the window size and the number of acknowledge packets during round trip time by the equation.

$$Diff = (Expected\text{-}Actual)\ BaseRTT \quad (4)$$

Where Expected is the expected data rates, Actual is the actual data rate and Base RTT is the minimum round trip time.

Figure 3 shows the congestion window size based on the difference between the actual and expected data rates. TCP Vegas provide a proactive response to congestion window. It attempt to keep at α bytes less than β bytes in queue where,

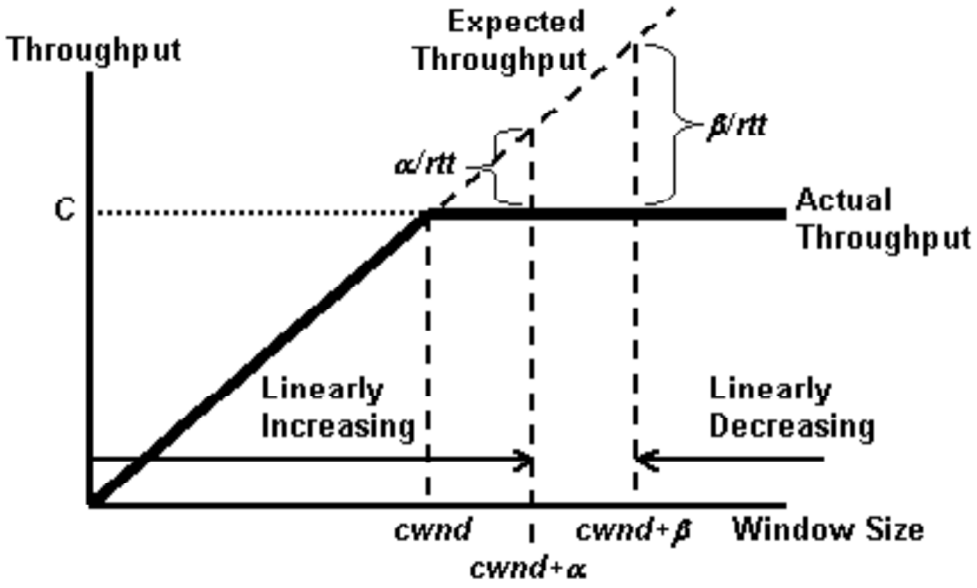

Figure 3: TCP Vegas[19]

cwnd – current congestion window size,

diff – Estimated backlog in queue,

α – low threshold for diff,

β – high threshold for diff

rtt – Actual (with congestion) round trip time.

2.7. TCP Sack

The congestion control algorithms implemented in our TCP SACK [9] are a conservative extension of Reno's congestion control. In this they use the same algorithms for increasing and decreasing the congestion window, in order to make minimal changes in the other congestion control algorithms. Adding SACK to TCP does not change the basic underlying congestion control algorithms.

The TCP SACK [10] implementation preserves the properties of Tahoe and TCP Reno of being sturdy in the presence of packets not in working order, and uses retransmit as the recovery method of last resort when timeouts. The main difference between the TCP SACK implementation and the TCP Reno implementation is the behavior of the algorithms when multiple packets of data are dropped from any one window. During Fast Recovery, TCP SACK maintains a variable called pipe that helps to estimate number of packets outstanding in the path.

The variable pipe is incremented by one when the sender sends a new packet or retransmits the old packet. It is decremented by one when the sender receives a duplicate ACK packet. Now TCP SACK option will be reporting that new data has been received at the receiver.

Figure 4 shows when the sender receives three duplicate ACK's, it retransmits the first lost segment, and increases the CWND one to each duplicate ACK it receives. During this retransmission segment it uses the SACK information to retransmit the lost segment before sending any new segments.

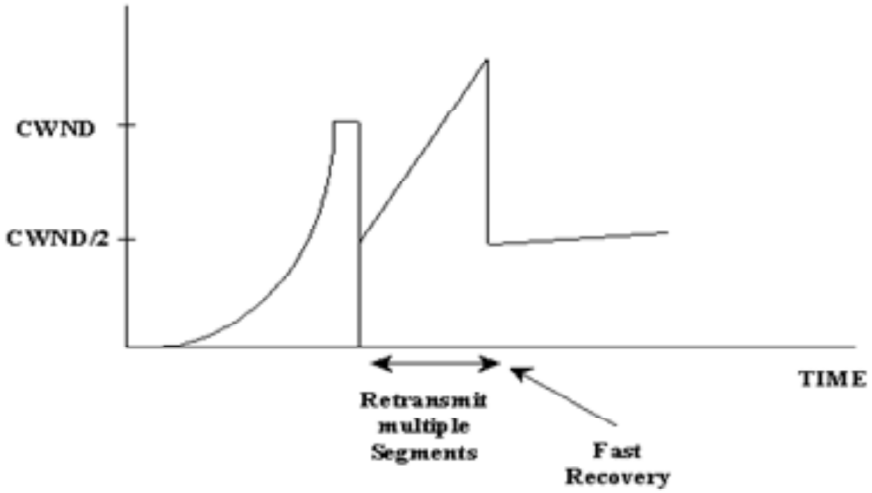

Figure 4: TCP SACK[20]

3. SIMULATION BASED ANALYSIS

This section describes the simulation tool, network topology, Simulation results and simulation parameter. The Performances of congestion control are evaluated on the basis of five performance metrics mentioned below.

3.1. Simulation Tool

Here the simulation of TCP Congestion control algorithm is done by using the network simulator (NS2) software due to its simplicity and availability. NS is a discrete event simulator targeted at network research. NS provides substantial support for simulation of congestion control over wired and wireless network. NS2 is written in C++ and OTCL for data per event packets.OTCL are used for the periodic and triggered event.NS2 include a network animator called nam animator, which provides traffic and topology generation. Post processing provides simple trace analysis.

3.2. Performance Metrics

We have used different parameters for measuring the performance of each protocol they are listed below in detail.

1) *Congestion Window Size:* It helps to determine the number of bytes standing in the queue at any time.
2) *Drop Probability:* It can be analyzed by the maximum threshold, minimum threshold and mark probability denominator.
3) *Average Queue Size:* It can be found by comparing the previous average and current size of the queue using a formula.

$$\text{Average} = (\text{old\_average}*(1-1/2\text{^n})) + (\text{current\_queue\_size}*1/2\text{^n}) \quad (5)$$

Where *n* is the exponential weight factor

4) *Packet Arrival:* The rate at which packets arrive with respect to the data rate at which they are serviced.

$$I = al/R \tag{6}$$

Where *R* is the link speed, *l* is the packet size and *a* is the offered load in packets/second.

3.3. Network Topology and Simulations Parameter

Network topology and simulation parameter are used in this paper to analyze the performance of TCP congestion control algorithms over a wired and wireless network. The different parameters are packets arrival, congestion window, average queue size, current queue size and drop probability. This topology consists of six nodes in which three nodes at each side of the bottleneck link. Here three nodes are acting as TCP source and three nodes are acting as TCP sink. Both the routers are performed by applying the congestion control algorithm. This is generated by network animator tool. The topology in our NS-2 is shown in figure 5 and Table 1.

Network topology consists of six nodes in which three nodes on each side of the bottleneck link. Here three nodes are acting as a TCP source and three nodes are acting as a TCP sink since both the routers are

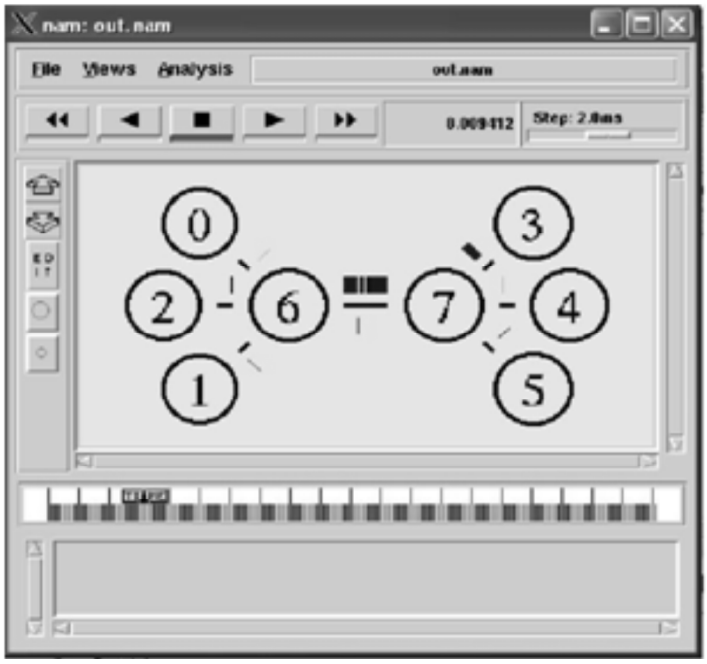

Figure 5: Network Topology

Table 1
Simulation Parameters

Parameters	*Value*
Packet Size	1200 bytes
Interface Queue	Drop Tail
Bandwidth	2 Mbps and 5 Mbps
Round Trip Time	10ms ~ 300ms
Simulation Time	10 ms to 50 ms
Date rate for FTP, TCP	8000 kb,1 Mb
Routing protocol	DSDV, AODV and DSR

applying the congestion control algorithm. This topology is generated by the network animator tool, after running TCL script by considering the following parameters table.

3.4. Simulation Results

1) *Analysis of Packet Arrival*

Figure 6 shows the occurrence packet arrival of the TCP congestion control algorithm. Table II shows the packet arrival in different congestion control algorithm with 10 ms to 50 ms time interval.

It is also observed that TCP Reno and New Reno are comparatively better performance than TCP SACK.TCP Vegas is better than the other congestion control algorithms.

Table 2
Packet Arrival

S. No	*Time*	*Reno (bytes)*	*New Reno*	*Vegas*	*SACK*
1	10 ms	200	203.75	212.5	192.5
2	20 ms	406.25	412.5	432.5	392.5
3	30 ms	612.5	621.25	652.5	593.75
4	40 ms	818.75	828.75	868.75	796.25
5	50 ms	1018.75	1030	1081.25	987.5

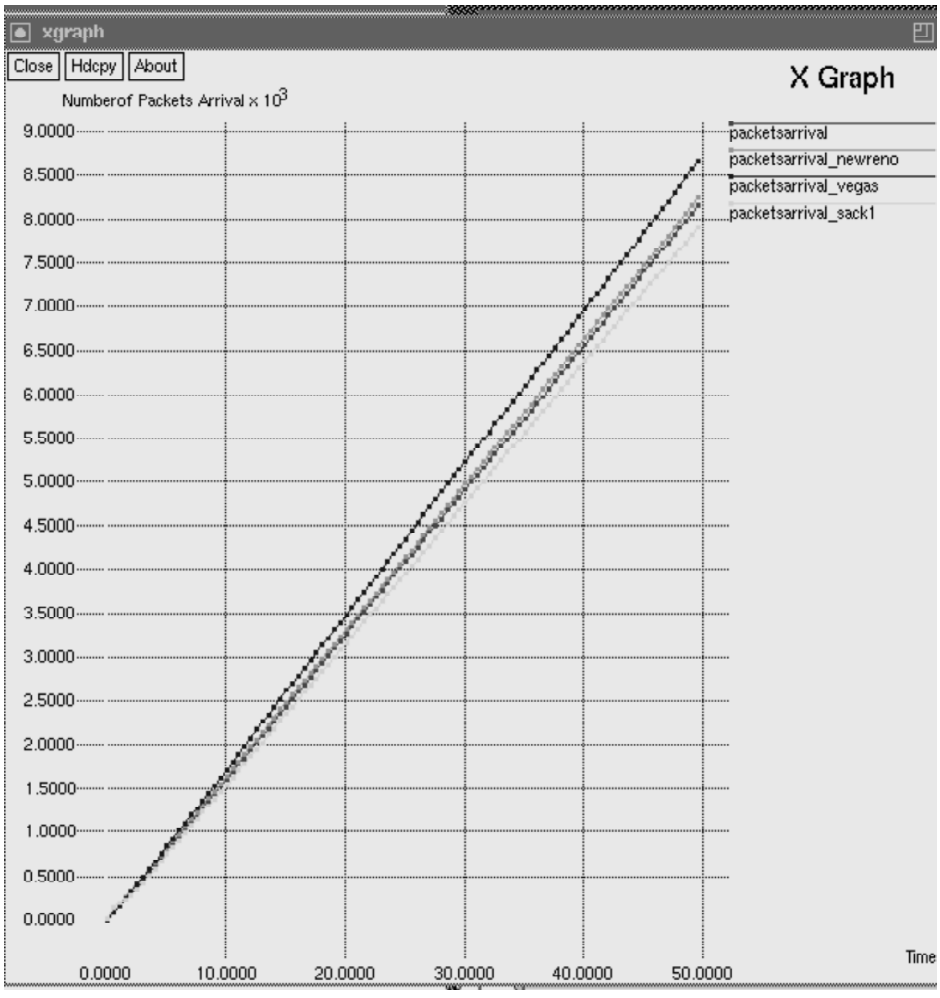

Figure 6: Packet Arrival

2) *Analysis of Packet Drop*

A Key to determine the happenings in the network is the real time analysis of dropping of packet since we can't wait for the packet to be re-sent.

Figure 7 shows the exponential growth of packet drop for various algorithm used in TCP congestion control. Table III shows the linear growth of queue size for TCP SACK and Vegas and a packet drop for TCP Reno and TCP New Reno at the initial stage 10 ms with an average interval of 50 ms in the end. After a subsequent interval TCP SACK shows better performance than the other congestion control algorithm.

Table 3
Packet Drop

S. No	*Time*	*Reno (bytes)*	*New Reno*	*Vegas*	*SACK*
1	10 ms	153	175	133	134
2	20 ms	364	363	307	259
3	30 ms	544	568	476	390
4	40 ms	712	748	630	535
5	50 ms	897	936	801	653

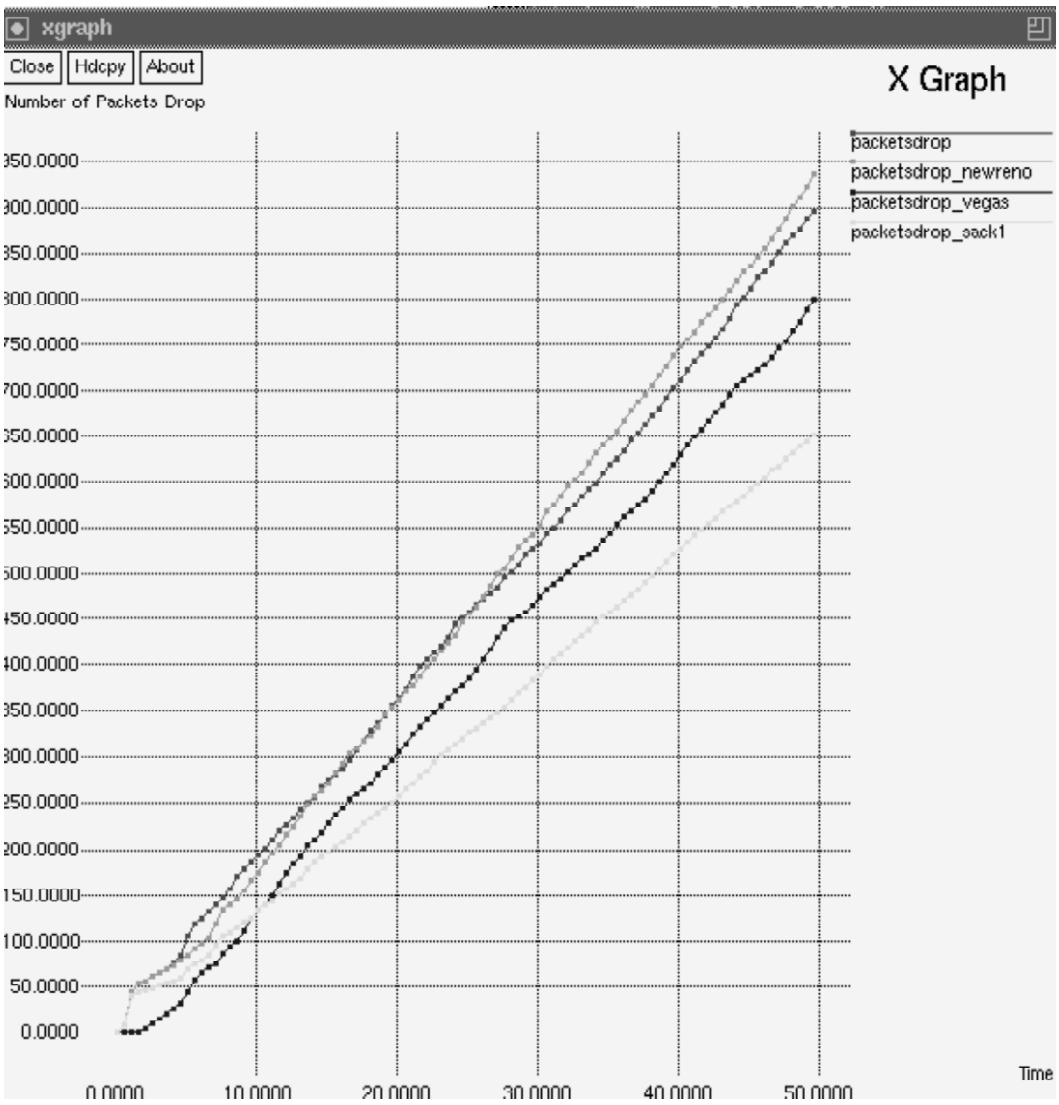

Figure7: Packet Drop

3) *Analysis of Average Queue Size*

The average queue size depends on the previous average with the current size of the queue. The calculation formula is given below:

$$Avg = o^* (1\text{-}2\text{-}n) + c^* (2\text{-}n) \quad (7)$$

Where *n* is the user-configurable exponential weight factor, o is the old average and *c* is the length of current queue. The previous average will be more important for high values of n. Peaks and Lows in queue length will be smoothed by a high value.

Figure 8 shows the exponential growth of queue size for TCP SACK and TCP New Reno at the initial stage. At the interval starting time of 10 ms Table IV shows the linear growth of average queue size for TCP Reno, New Reno, Vegas and SACK. After the subsequent interval time TCP Vegas shows better performance when compared to other congestion control algorithm.

Table 4
Average Queue Size

S. No	*Time*	*Reno (bytes)*	*New Reno*	*Vegas*	*SACK*
1	10 ms	7613.23	7720.82	7393.85	7918.22
2	20 ms	7434.01	8131.3	8295.59	7658.25
3	30 ms	7980.88	8623.24	8234.75	7581.86
4	40 ms	7313.83	8112.9	7553.21	7622.49
5	50 ms	7366.43	7824.03	8082.37	7730.91

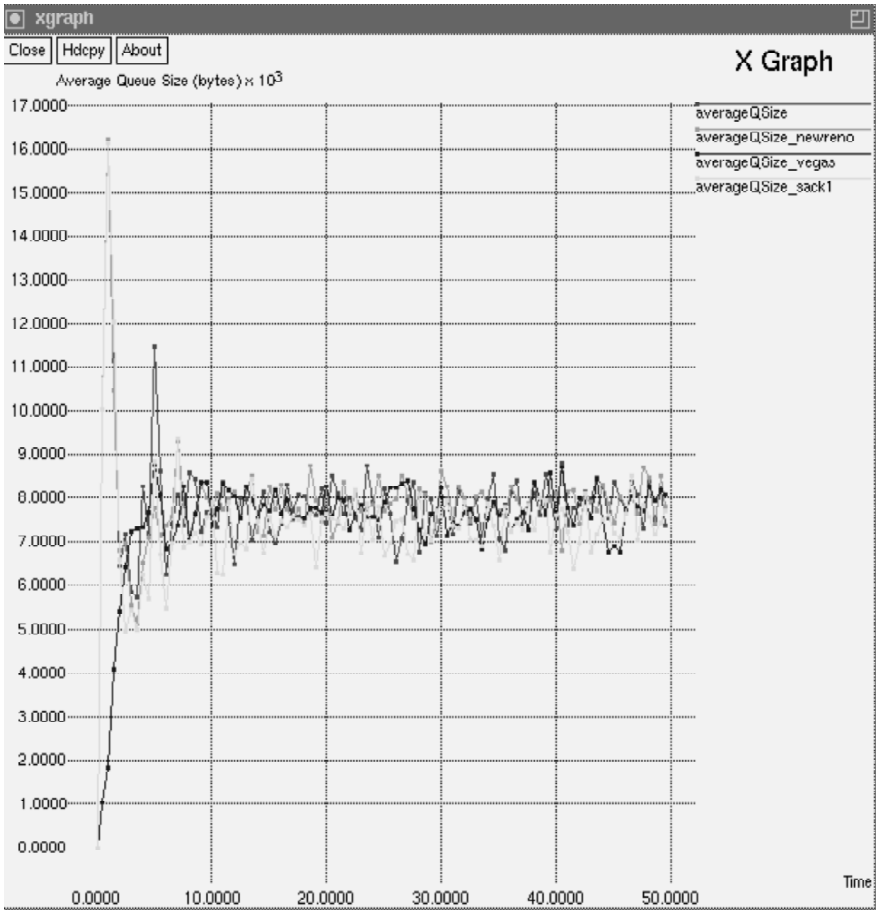

Figure 8: Average Queue Size

4) *Analysis of Current Queue Size*

Figure 9 shows the exponential growth of queue size for TCP Vegas at the initial stage. At the interval starting time of 10 ms Table 5 shows the exponential growth of average queue size for TCP New Reno, Vegas. After the subsequent interval time TCP Reno shows better performance when compared to other congestion control algorithm.

Table 5
Current Queue Size

S. No	*Time*	*Reno (bytes)*	*New Reno*	*Vegas*	*SACK*
1	10 ms	4144	8880	9936	2960
2	20 ms	10065	10064	6072	2368
3	30 ms	7696	3552	8280	5328
4	40 ms	9472	5328	3864	7696
5	50 ms	5328	6512	6072	7104

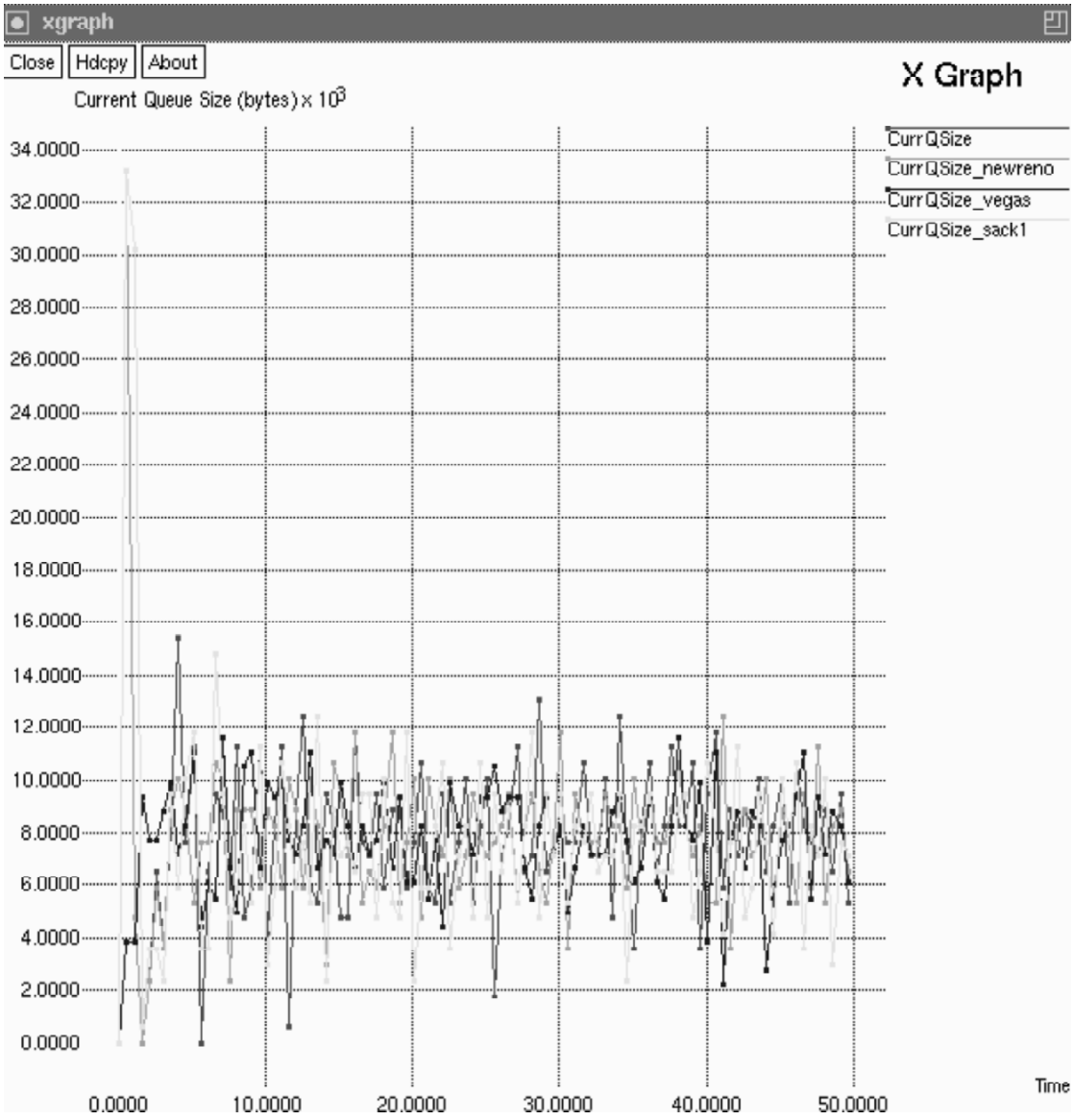

Figure 9: Current Queue Size

5) ***Analysis of Drop probability***

Drop probability is counted from the last discarded packet to the number of packets arriving consequently without any discard. It has been analyzed that if count increases, then drop probability will also increase. So the dropping probability can be calculated using Px (avg) as follows.

$$Px(avg) = Pmax*(avg-Tthr\ (min)/\ (Tthr\ (max)-Tthr\ (min) \quad (8)$$

Where Px is the temporary probability which is varied from 0 to Pmax.Tthr (max) is maximum threshold

Below Table 6 shows the minimal dropping probability for of TCP congestion control algorithms such as Reno, New Reno, Vegas and SACK with a subsequent interval of 10 ms.From the table we came to know that TCP Reno and SACK is minimal at the beginning and shows gradual increase than the other algorithms i.e., Vegas and New Reno.

Table 6
Dropping Probability

S. No	*Time*	*Reno*	*New Reno*	*Vegas*	*SACK*
1	10 ms	0.11358	0.12649	0.09787	0.15079
2	20 ms	0.09868	0.17575	0.19547	0.11899
3	30 ms	0.11292	0.23478	0.18817	0.10982
4	40 ms	0.09627	0.17354	0.10638	0.11471
5	50 ms	0.09732	0.13884	0.16988	0.12771

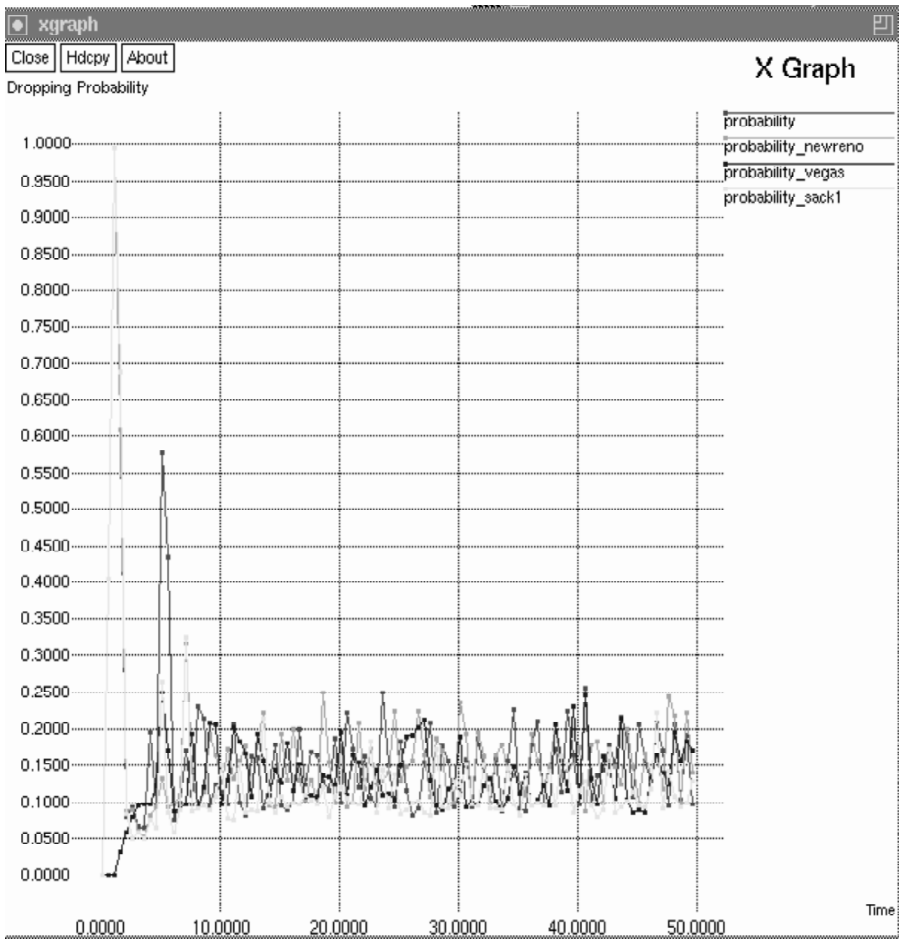

Figure 10: Dropping Probability

6) Analysis of Congestion Window Size

Congestion window actually tells the number of outstanding packets in a congested window at one time. If its value is less than the congestion window size it means that network is not congested and greater value means that network is congested. In order to calculate the congested window size we use some in built variables that give us the size of the congested window in a particular time.

The Calculation formula is given below

$$Cwnd = Cwnd = MSS*(MSS/Cwnd)$$

Where Cwnd is the congestion window and MSS is the maximum segment size.

Table 7 shows the consistent maintenance of congestion window size for TCP Vegas. After the subsequent interval time TCP Reno gradually maintains the minimum congestion window size, which will affect the link to increase the data rate. At the end of time interval TCP Reno, New Reno and SACK are come with a congestion size of 1 that also shown on figure 11.

Table 7
Congestion Window Size

S. No	*Time*	*Reno*	*New Reno*	*Vegas*	*SACK*
1	10 ms	1	2.9	4	1
2	20 ms	1.5	2.1	2	2.9
3	30 ms	1.3	1	2.5	1
4	40 ms	1.4	1	2.5	3.55
5	50 ms	1	1.2	2.5	1

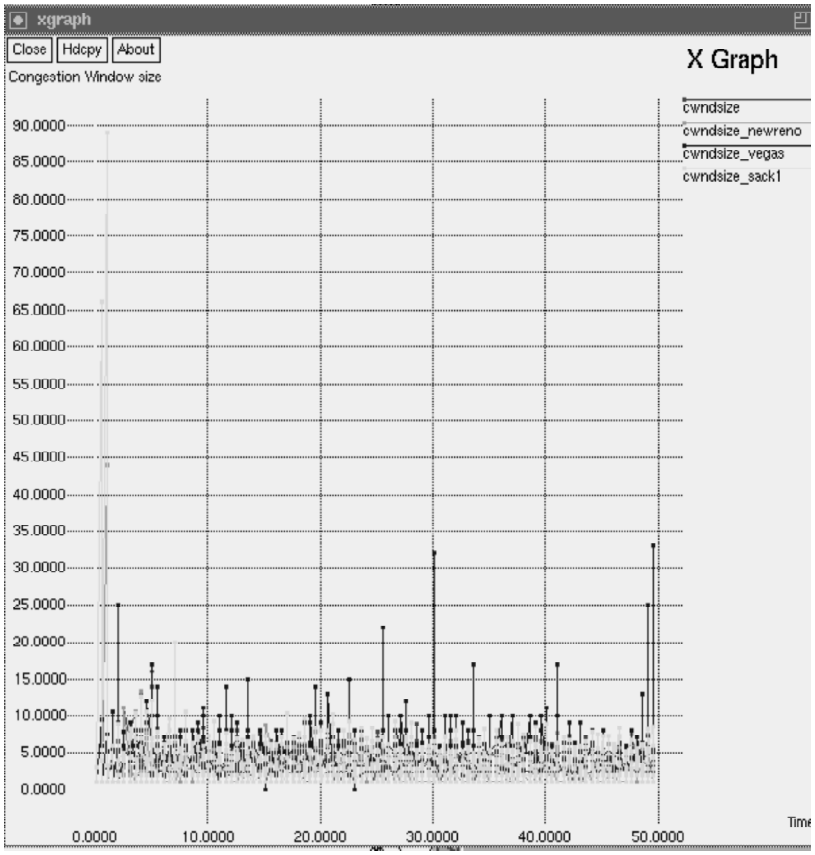

Figure 11: Congestion Window Size

3.5. Simulation Results of Wireless Network:

1) *About Wireless Network*

A wireless network [13], which uses high-frequency radio waves rather than wired network to communicate between nodes. Wireless networks are reliable, but when interfered within it will reduce the range and the quality of the interference signal. Interference signal will be caused by other devices operating on the similar radio frequency. So it is very hard to control the addition of new devices on the same frequency.

Table 8
Network Parameters

Parameter	*Value*
Channel	Wireless Channel
Adhoc Routing	DSDV
Wireless Error rate	5%
LAN bandwidth	10 Mbps
Application	FTP
Mac Standard	802.11
Ifq	Drop Tail

2) *Analysis of Congestion Window*

In wireless networks, to ensure a good TCP performance, it is necessary to limit the TCP congestion window size thereby reducing the probability of congestion loss. This congestion loss is considered to be the result of network congestion. As a precaution the congestion window size is reduced dramatically.Table 9 shows the consistent maintenance of congestion window size for TCP Reno.Figure 12 shows the gradual maintenance of congestion window size in TCP algorithms.

Table 9
Congestion Window Size in Wireless Network

S. No	*Time*	*Reno*	*New Reno*	*Vegas*	*Tahoe*
1	2 ms	20.24	1	2	1
2	4 ms	22.49	12	2	1
3	6 ms	23.75	20.54	2	1
4	8 ms	27.02	10	2	1
5	10 ms	29.83	11.31	4.20	1

4. CONCLUSION

In this paper, the authors focused on detailed evaluation and comparison of TCP Reno, New Reno, Vegas and SACK in both wired and wireless networks. By taking different parameters like congestion window size, average queue size, current queue size, packet arrival and dropping probability are implemented through NS2 simulation. From the evaluation of wired network TCP Vegas could achieve the higher arrival of the packet at given bandwidth.TCP SACK shows a minimum drop of packets when compared to other congestion control algorithm.TCP Vegas shows better performance of average queue size in wired networks. While analyzing the current queue size TCP Reno shows the improved performance when weigh against others.TCP Reno and SACK show minimum dropping probability.TCP Vegas shows the consistent maintenance of congestion window size in wired networks. From the evaluation of wireless network TCP Reno shows the best performance in congestion window size when compare to others.

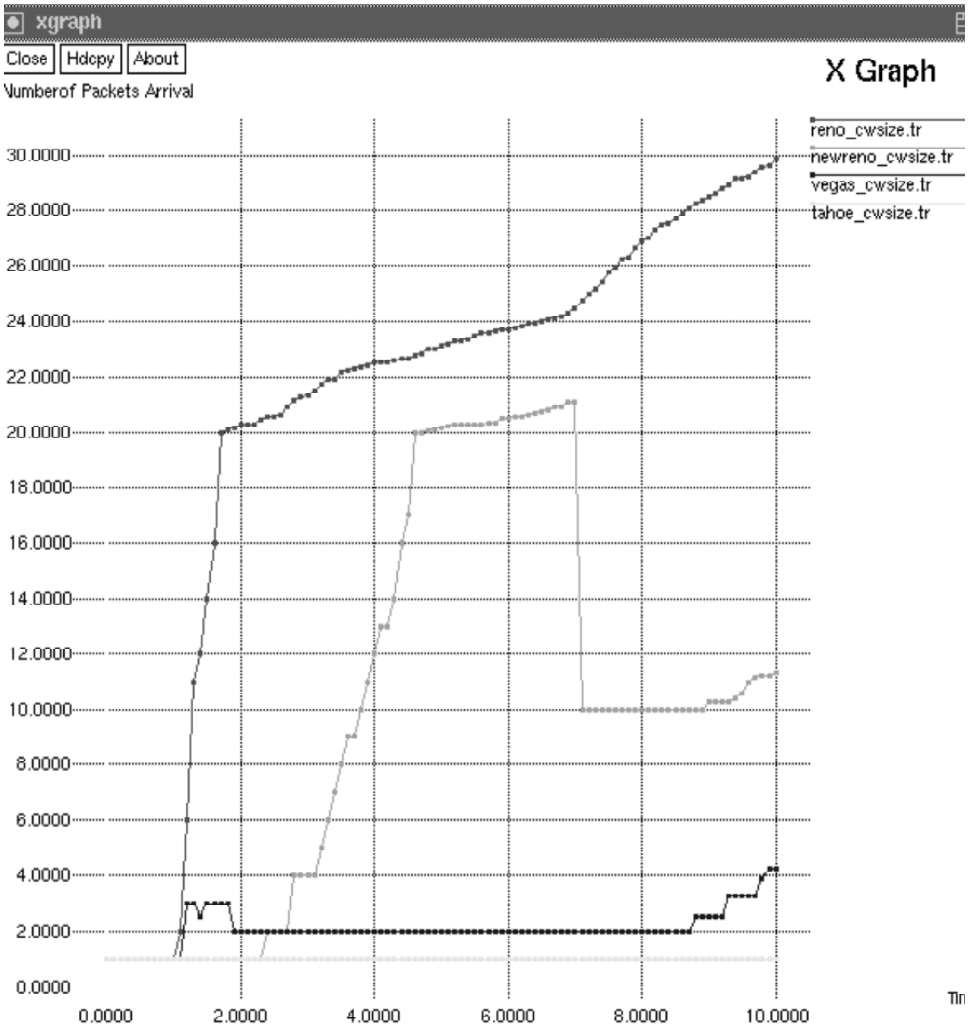

Figure 12: Congestion Window Size in Wireless Network

Reference

[1] J. Postel. Transmission Control Protocol. RFC 793, September 1980.

[2] D. Bansal and H. Balakrishnan. Binomial Congestion Control Algorithms.In *Proc. of IEEE INFOCOM*, pages 631–640, Anchorage, USA, April 2001

[3] P. Gevros, J. Crowcroft, P. Kirstein, and S. Bhatti, "Congestion *control* mechanisms and the best effort service model," IEEE Network, vol. 15, no. 3 pp 16-26, May 2001.

[4] E. Altman, K. Avrachenkov, and C. Barakat. A Stochastic Model of TCP/IP with Stationary Random Losses. In *Proc. of ACM SIGCOMM*, pages 231–242, Stockholm, Sweden, August 2000.

[5] Jitender Sharma and Amit Kumar Garg-"Analysis of Tahoe: A TCP Variant", International Journal of Engineering and Advanced Technology(IJEAT), ISSN: 2249-8958, Volume-1, Issue-2, December2011.

[6] The Network Simulator-NS-2. Information Sciences Institute, University of Southern California, USA, 2006.www.isi.edu/nsnam/ns/

[7] V. Jacobson. Berkeley TCP evolution from 4.3-Tahoe to 4.3 Reno. In *Proc. of the 18th Internet Engineering Task Force*, Vancouver, Canada, August 1990

[8] V. Jacobson. Congestion Avoidance and Control. In *Proc. of ACM SIGCOMM*, pages 314–329, Stanford, CA, USA, August 1988.

[9] A. Medina, M. Allman, and S. Floyd. Measuring the Evolution of Transport Protocols in the Internet. *Computer Communications Review*, 35(2):37–51, April 2005.

[10] M.Mathis, S.Floyd. TCP Selective Acknowledgement Op-tions (SACK) http://tools.ietf.org/html/rfc2018.

[11] K. Fall and S. Floyd. Simulation-Based Comparisons of Tahoe, Reno, and Sack TCP. *ACM Computer Communication Review*, 26(3): 5–21, July 1996.

[12] A Comparative Analysis of TCP Tahoe, Reno, ew-Reno, SACK and Vegas L.S. Brakmo & L.L.Peterson IEEE Journal on Selected Areas in Communication, vol. 13[1995],(1465-1490).

[13] "TCP Performance Enhancement using ECN & Snoop Protocol for Wi-Fi Network", Manish D Chawhan and Dr. Avichal Kapur.The 2nd IEEE International Conference on Computer and Network Technology (ICCNT'10), 2010, Bangkok, Thailand.

[14] "The Impacts of Transmission Topology Control on the European Electricity Network" Han, J.; Papavasiliou, A. Power Systems, IEEE Transactions on Year: 2016, Volume: 31.

[15] "M21TCP: Overcoming TCP incast congestion in data centres" Adesanmi, A.; Mhamdi, L. Cloud Networking (CloudNet), 2015 IEEE 4th International Conference on Year: 2015.

[16] Performance evaluation of congestion-aware routing protocols for underwater sensor networks with multimedia data Jain, S.; Pilli, E.S.; Govil, M.C.; Rao, D.V. Underwater Technology (UT), 2015 IEEE Year: 2015.

[17] http://cnp3book.info.ucl.ac.bee/2nd/html/protocols/congestion.html

[18] http://www2.ensc.sfu.ca/~ljilja/cnl/presentations/grace/modeling_TCP/sld012.htm

[19] www.soi.wide.ad.jp/class/20060035/slides/07/17.html

[20] http://www.cse.wustl.edu/~jain/cis788-97/ftp/tcp_over_atm/index.html