MEASURING PERFORMANCE OF VARIANTS OF TCP CONGESTION CONTROL PROTOCOLS

Harjinder Kaur

CSE, GZSCCET, Dabwali Road,
Bathinda, Punjab, India,
sidhuharryab@gmail.com

Gurpreet Singh

CSE, GZSCCET, Dabwali Road,
Bathinda, Punjab, India
Gps_ynr@yahoo.com

Abstract

Today's TCP is the most used Internet protocol. Actually Transmission Control Protocol (TCP) has various advantages like reliability, connection oriented, provide flow control and congestion control mechanisms, etc. As compared to wired networks, TCP provides the various characteristics in wireless networks. In this research paper, we compare some TCP variants like, Reno, NewReno, SACK, FACK, Asym, RBP and Vegas. We compare these variants on the basis of different parameters that are total delay, jitter, average throughput, no. of dropped packets etc. and select the best variant in particular parameter.

Keywords: Reno; NewReno; SACK; FACK; Asym; RBP; Vegas.

1. Introduction

Congestion occurs when too many packets are present on the same network and it degrades the performance of the network. There are some factors that exceed responsible for the congestion like packet arrival rate, the outgoing link capacity, insufficient memory to store arriving packets, bursty traffic, speed mismatch, misbehavior of resources etc. Various TCP variants have been developed to deal with network congestion. An easy way to resolve the problem of congestion is to employ the principle of packet conservation in which stops sending a new packet until the previous one is successfully delivered to the receiver [3]. The best characteristic feature of TCP is its congestion control algorithms.

1.1. *Role of TCP*

- TCP is part of the system's kernel.
- TCP is responsible for sending/receiving packet in order (FIFO).
- TCP resends the missed packet.
- It also handles error.
- It is a connection oriented.
- It is responsible for correct delivery of data.

We use the NS-2 that is network simulator version 2 as a platform to run these TCP variants. NS-2 is basically an object-oriented script interpreter that offers some feature like simulation event scheduler and network component object libraries [2]. Fig. 1 represent the simulation cycle of NS.

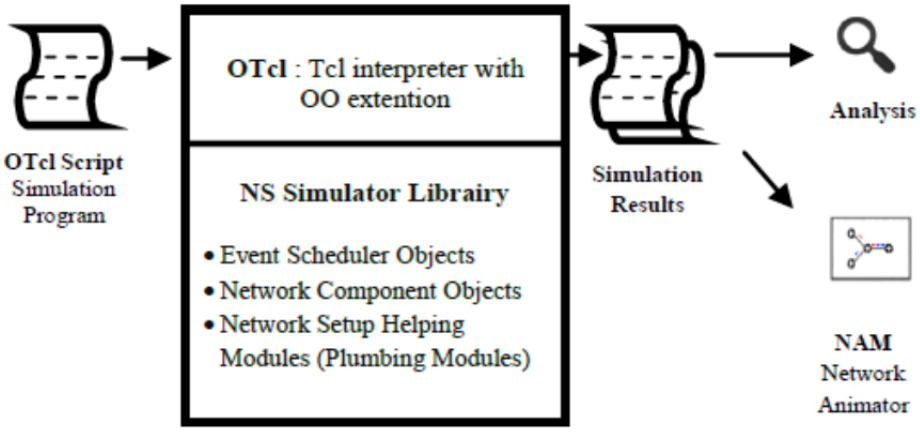

Fig.1. Simulation cycle of NS-2

2. TCP Variants

Each variation of TCP possesses some special criteria. All the variants look like same, but they have a different technique to deal with congestion. TCP's first variant was Tahoe. A new mechanism called Fast Recovery adds to TCP Tahoe, a new variant is introduced that is TCP Reno. TCP NewReno adds a newest mechanism retransmission to TCP Reno. TCP Vegas also provide its own congestion control techniques and unique retransmission. TCP FACK is same as Reno with Forward Acknowledgment. There are other kinds of TCP variants like SACK, RBP, and Asym etc.

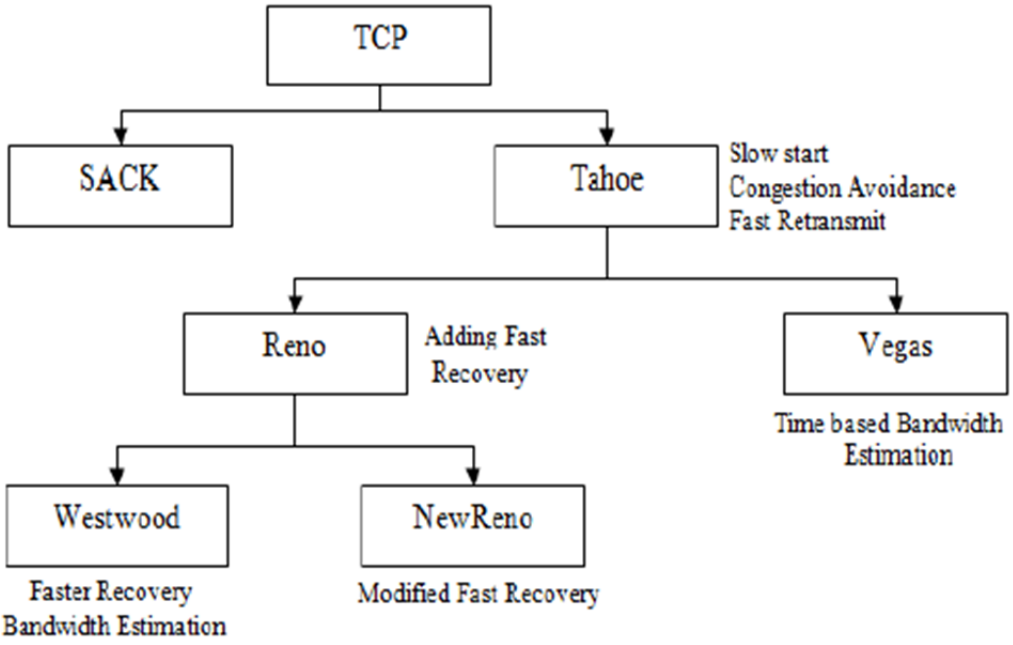

Fig.2. TCP variants

2.1. *TCP Reno*

TCP Reno has the same features that TCP Tahoe contains like slow start, congestion avoidance. TCP Reno also has a new feature "Fast Retransmit". TCP connection goes through slow start, when it starts or restarts transmission of data after a packet loss. First of all the sender sets the CWND (Congestion Widow) to 1and then increases the CWND by 1, after receiving each ACK (Acknowledgment). We send 1 packet in first round trip time (RTT), 2 packets in second and 4 packets in third RTT. We continuously increase the sending rate until congestion will not occur. When congestion encounter we decrease sending rate and start over again by setting congestion window to 1 [4]. TCP Reno allows the connection to quickly recover from isolated packet losses

through fast retransmit and recovery algorithm. We re- transmit the lost packets when three duplicate packets are received without waiting the timeout.

Fast Retransmit has the following algorithm:-

(1) When 3 duplicate ACK's received then,

(i) Set threshold = CWND/2

(ii) Retransmitting missing packets.

(iii) Set CWND = threshold + 3.

(2) After getting duplicate ACK, increase cwnd by 1.

(3) When the next ACK arrives which acknowledges new data, then set CWND = threshold [9] [12].

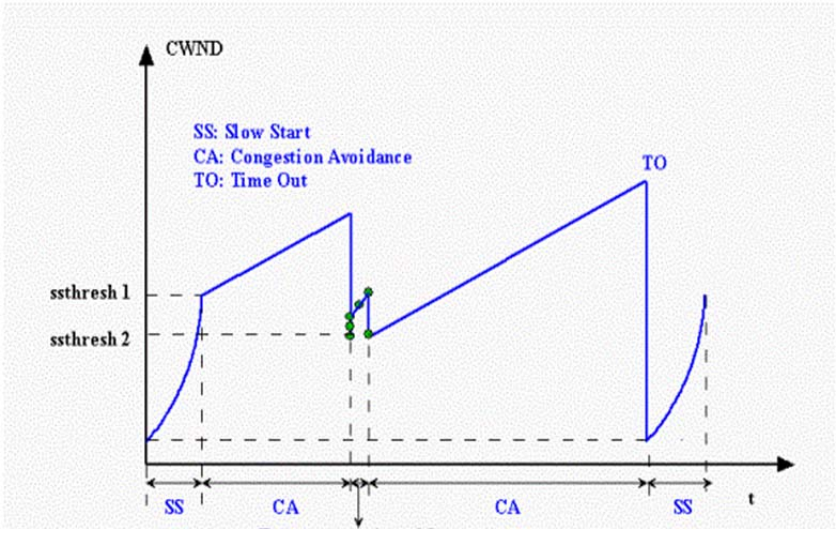

Fig.3. Fast retransmit and Fast Recovery in TCP Reno

2.1.1. Disadvantages of TCP Reno

- Reno can detect single packet loss.
- It performs very well when small packet loss.
- The window is very small when the loss occurs, then we would never receive enough duplicate acknowledgements for a fast retransmit [4].

2.2. *TCP NewReno*

TCP NewReno is extended version of TCP Reno. It is able to detect multiple packet losses and thus is much more efficient than TCP Reno. Like Reno, NewReno also enters into fast-retransmit. It exits Fast recovery when all the packets outstanding in the network are acknowledged. Partial acknowledged packets are treated as the indicator for packet loss and should be retransmitted. When multiple packets are lost from a single window, then NewReno can improved without retransmission. TCP NewReno has an advantage that it can handle multiple packet loss [4] [11].

2.2.1. Disadvantages of New Reno

- New-Reno suffers from one problem that it is taking one RTT to detect each packet loss.

2.3. *TCP SACK*

TCP SACK stands for 'Selective Acknowledgments'. It is an improved version of TCP Reno. It solves the problems faced by TCP Reno and New Reno like detecting multiple packet loss in the same window [5]. TCP SACK also has the slow start, fast retransmit and coarse grained timeout features of old TCP variants. The main feature of SACK is, it acknowledged the packets selectively rather than cumulatively. Following figure shows how packets are selectively acknowledged:

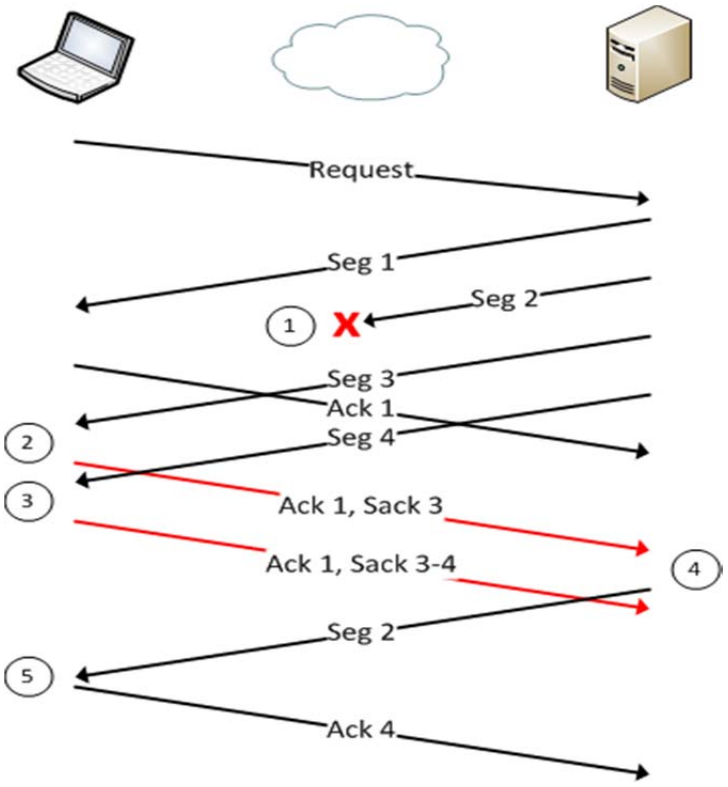

Fig.4. Selective Acknowledgement by SACK.

When the sender enters in faster recovery, then it sets the CWND to half of the current size because SACK initialize a variable pipe that keep track on how many packets are outstanding in the network. This pipe variable is reduced by 1when an ACK is received. When CWND goes larger than pipe, then it is the sign for checking which packets are un- received and send them again, otherwise send new packets [4] [9].

2.3.1. Disadvantages of SACK

- Currently selective acknowledgments are not provided by the receiver.
- Selective acknowledgment is not a very easy task.

2.4. *TCP FACK*

To improve TCP congestion control during recovery, a new algorithm is introduced that is TCP FACK. FACK or Forward Acknowledgement is on top of SACK options. During recovery FACK perform congestion control by keeping track of the amount of data outstanding in the network. The important feature of FACK is to consider the most forward selective ACK sequence number as a sign that all the previous unselected acknowledged packets were lost. FACK improve the recovery process of packets lost significantly and performance than the traditional approaches [5].

2.4.1. Disadvantages of TCP FACK

- FACK may not avoid unnecessary inflation of congestion window through delay sensing technique.

2.5. *TCP RBP*

RBP stands for Rate based pacing. Slow start problem decreases the performance. This problem is solved by sending the packets at a few paces until we may get the ACK clock running again. This rate should be depends on the fraction of previous estimates of data transfer rate and nearby estimate of accessible bandwidth, these types of modification is known as Rate Based Pacing [8].

RBP requires the following changes to TCP:

- Idle time detection and indication that RBP needs to be started.
- Bandwidth estimation.
- Calculation of the window that we expect to send in RBP and the timing between segments in that window.
- A mechanism that clocks the segments sent in RBP [10].

2.6. *TCP Asym*

TCP/Asym has the same features like the Reno and New Reno. We can say that it is evaluated from these variants of TCP [10]. It is suitable for using high volume data transmission application. Like FACK and Tahoe, Asym reset the CWND to 1 when a packet loss event happens.

2.6.1. Disadvantages of TCP Asym

- It is not suitable for medium quality and high quality real time applications.

2.7. *TCP Vegas*

TCP Vegas is modified version of TCP Reno. It introduced a modified slow start to prevent the congestion from the network and Vegas also have an advantage that it does not require duplicate ACK to detect a packet loss. It detects the congestion before packet loss.

2.7.1. New Retransmit Mechanism

Vegas has a New Retransmit mechanism. It calculates packet sending time and RTT. When a duplicate ACK is received, it checked if current packet transfer time > RTT, then it immediately retransmits the packets without waiting three duplicate ACK. TCP Vegas can detect multiple packet losses [1] .To pick another packet that has been lost previously, when a non-duplicate acknowledgement is received, one after a fresh acknowledgement and it is the first or second, then it checks the timeout values and if the packet time since it was sent exceeds the timeout value then it retransmits the packet without waiting for a duplicate acknowledgement.

2.7.2. Modified Slow-Start

Vegas have different slow start than the other variants. In other variants when connection set up they has no idea about available bandwidth and the exponential increase over shoot the bandwidth by big amount and there congestion can occur. To overcome this problem TCP Vegas increase exponentially only every other RTT and calculate the actual sending throughput to the expected and when the difference goes above the threshold, it exits the slow start and enters the congestion avoidance [12] [13].

2.7.3. *Congestion Avoidance Mechani*sm

TCP Vegas avoid the congestion very differently than other variants. A large queue is built up when it decreases the sending rate of packet as compared to required rate. For this it uses a variation of Wang and Crowcrofts, Tri-S scheme. When calculated rate is too far from the required rate it increases the use of available bandwidth, but when the calculated rate comes closer to the required rate than it decrease the usage of bandwidth [6]. Thus Vegas do not waste the bandwidth and efficiently control the congestion.

.

2.7.4. Disadvantages of TCP Vegas

- TCP Vegas cannot compete with more aggressive TCP Reno Connections.
- It may not stabilize if buffers are small [7].

3. Performance Analysis

To obtain the result we perform three experiments on TCP variants. In these experiments we use different parameters and on the basis of theses parameters we get the different values from all the TCP variants. We represent the comparison of theses TCP variants with the help of graphs. Brief description about parameters are given below:-

- **Number of packets dropped**: - It is failure of transmitting packets to arrive at their destination. It is calculated as:
  Number of packets dropped= total no. of packets sent-total no. of packets received
- **Number of packets sent**: - It is the total number of packets sent by the sender.
- **Delivery Ratio**: - To calculate delivery ratio we need the total number of packets sent and total number of packets received.
  Delivery Ratio = no. of packets successfully delivered/ total no. of packets sent
- **Average Throughput**: - throughput is the rate at which packets transferred between the sender and receiver. Where the average throughput is the rate over a longer period if time.Units of average throughput are bytes/Sec or bits/Sec.
- **Total Delay:** - Total delay is the difference between the time at which sender generated the packet and time which the receiver received the packet.
  Total delay= packet generation time/packet receiving time.
- **Total Jitter: -** Variations in delay of receiving packets, called jitter. It is the variation in latency as measured in the variability over time of the packet latency across a network
- **Average Delay: -** The average delay a packet takes to travel from sender to the receiver side node. A delay is introduced due to the queuing of packets at the interface of node, time transmission and due to buffering during route discovery [2].
- **Average Jitter: -**It is time variation between subsequent packets arrived. Main causes of jitter are network congestion or route changes.

3.1. *Experiment 1*

In the first experiment number of nodes are 10. We get the Table 1 from this experiment. As we can see in the table no. of packets dropped are 0 for all variants due to the simplicity in the network. TCP RBP and TCP Vegas both have lower total delay and total jitter than other TCP variant. TCP Vegas also send higher packets than the others.

Table1. Comparison of TCP variant when number of nodes = 10

Protocols→	**Asym**	**RBP**	**SACK**	**FACK**	**Reno**	**NewReno**	**Vegas**
Parameters↓							
Number of packets dropped	0	0	0	0	0	0	0
Number of TCP packets sent	3884	4031	3884	3884	3884	3884	4121
Delivery Ratio	100	100	100	100	100	100	100
Average Throughput	57676.57	57585.71	57676.57	57676.57	57676.57	57676.57	58871.43
Total Delay	2213.133	235.132	2209.294	2209.294	2213.133	2213.133	258.7978
Total Jitter	930.6553	185.5355	928.4525	928.4525	930.6553	930.6553	206.7912
Average Delay	0.569808	0.05833	0.568819	0.568819	0.569808	0.569808	0.0628
Average Jitter	0.239674	0.046386	0.239107	0.239107	0.239674	0.239674	0.057966

3.2. *Experiment 2*

In second experiment number of nodes are double. According to the Table 2 we select best TCP variant in different parameters. TCP Vegas is best in total delay, total jitter, average delay and average jitter. But RBP sends higher TCP packets and FACK has a higher average throughput. For detail, see the following table 2.

Table2. Comparison of TCP variant when the number of nodes = 20

Protocols→	**Asym**	**RBP**	**SACK**	**FACK**	**Reno**	**NewReno**	**Vegas**
Parameters↓							
Number of packets dropped	0	0	0	0	0	0	0
Number of TCP packets sent	2560	2596	2539	2566	2555	2555	2495
Delivery Ratio	100	100	100	100	100	100	100
Average Throughput	66410	64900	65864	66566	66280	66280	62375
Total Delay	1515.172	159.5237	1382.68	1232.685	1521.528	1521.528	172.5969
Total Jitter	1000.285	462.5437	797.6587	768.0233	876.4427	876.4427	144.5673
Average Delay	0.59184	0.084415	0.544577	0.480392	0.59551	0.59551	0.069177
Average Jitter	0.390889	0.076644	0.314286	0.299424	0.343165	0.343165	0.057966

3.3. *Experiment 3*

In experiment 3 we use 40 nodes named from 0-39. In this experiment complexity increases and we can see that as TCP variants drop some packets, but Vegas perform best in this parameter. TCP Reno sends higher packets and also has a higher average throughput. In other four parameters such as total delay, total jitter, average delay and average jitter, TCP RBP perform best.

Table3. Comparison of TCP variant when the number of nodes = 40

Protocols→	**Asym**	**RBP**	**SACK**	**FACK**	**Reno**	**NewReno**	**Vegas**
Parameters↓							
Number of packets dropped	5	2	3	4	7	6	0
Number of TCP packets sent	5731	6036	5881	5795	6039	6040	5773
Delivery Ratio	99.91276	99.96686	99.94899	99.93097	99.88409	99.90066	100
Average Throughput	49468	50300	50877	50123.33	52254.67	52263.33	48108.33
Total Delay	5191.998	509.5267	5007.786	3957.328	5349.767	5142.027	831.4774
Total Jitter	3791.044	462.5437	3159.615	3125.209	3460.719	3388.406	808.8977
Average Delay	0.90595	0.084415	0.85152	0.682887	0.88587	0.851329	0.144029
Average Jitter	0.661613	0.076644	0.314286	0.539387	0.573157	0.561087	0.140166

3.4. *Comparison of TCP variants*

3.4.1. Number of Nodes vs. Number of Packets Dropped

It is the difference between the total number of packets sent by the sender and total number of packets received by the receiver. Number of Nodes are related to the number of packets dropped. As we increase the no. of nodes, then the complexity of the network is automatically increased. The complexity may causes of congestion. All variants have the different performance whenever the numbers of nodes are increased. In the first experiment we checked on 10 nodes at this level no. of packets dropped by TCP variants is 0. In experiment 2 we double the no. of nodes. There's no complexity arises at this stage so no. of dropped packets is still 0 by all algorithms. Actually, in previous experiments, there is simplicity in the network. But in the last experiment when the number of nodes are 40 then packets are starting a drop as shown in the Fig. 5.

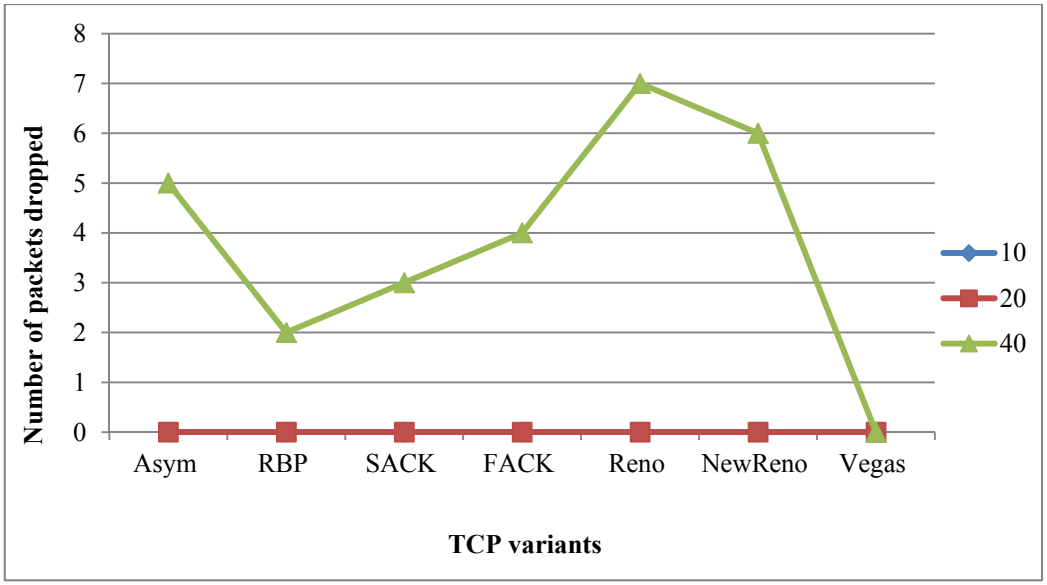

Fig.5. Number of nodes vs. Number of packets dropped

3.4.2. Number of nodes vs. Delivery ratio

We can get the delivery ratio by dividing the total number of packets, delivered to the receiver with total number of packets sent by the sender. As we can see in the Figure delivery ratio is 100 percent when numbers of nodes are 10 and 20. But whenever the number of nodes becomes 40 then delivery ratio of all variants is decrease except the TCP Vegas.

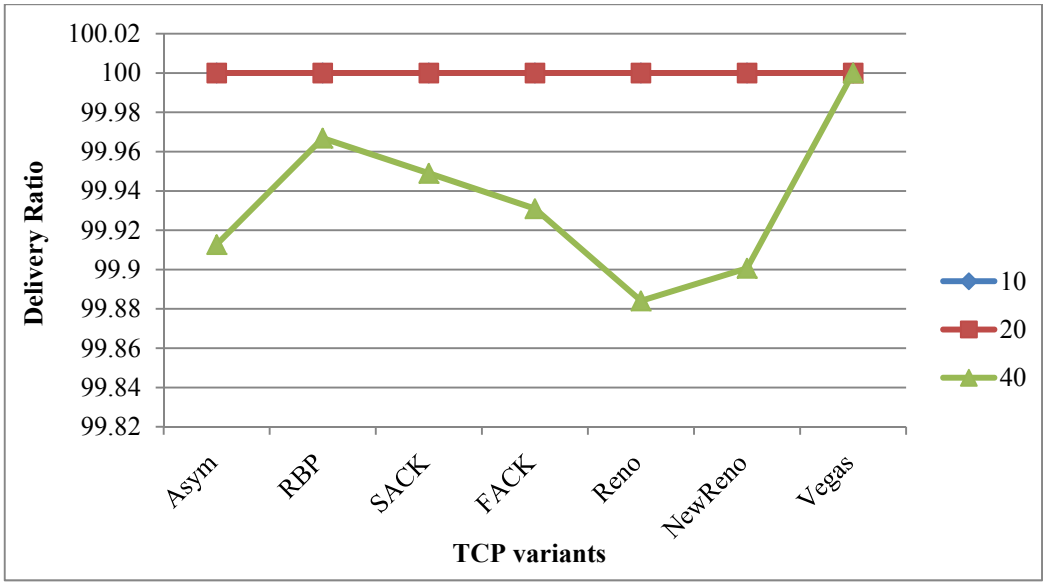

Fig.6. Number of nodes vs. Delivery ratio

3.4.3. Number of nodes vs. Number of packets sent

It is the rate of packet transmission. Number of nodes also affect the number of packets send by each node. In experiment1 where's the no. of nodes are 10 then TCP Reno, TCP New Reno, TCP SACK, TCP FACK and TCP Asym has the same output and TCP Vegas sends the highest number of packets. In second experiment no. of the nodes become 20 and TCP RBP sends the highest number of packets. Now in the last experiment TCP NewReno sends highest no of the packet.

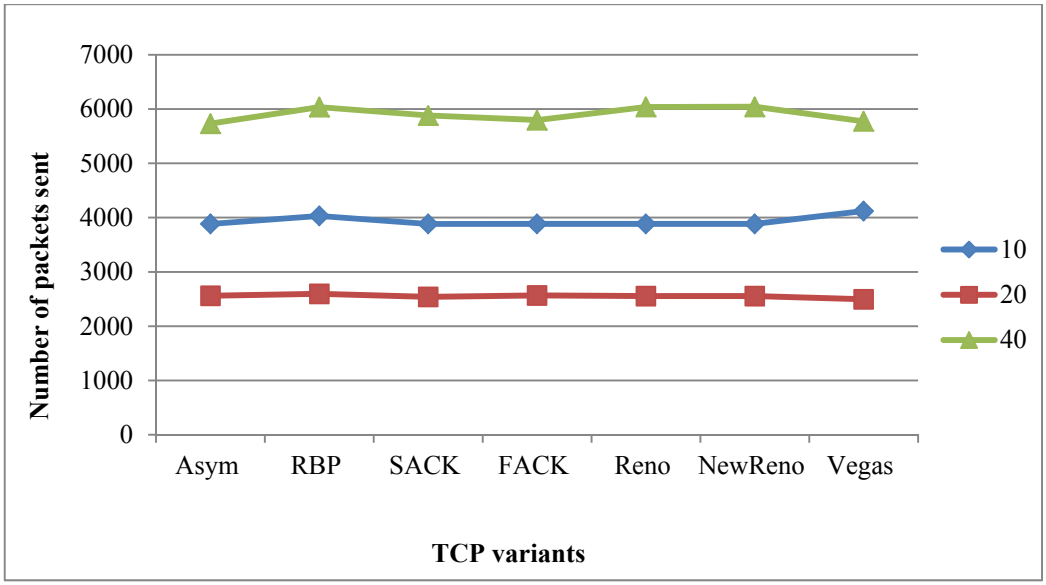

Fig.7. Number of nodes vs. Number of packets send

3.4.4. Number of nodes vs. Average throughput

Average throughput is rate over a longer period of time. It is measured in bytes/sec. In first stage every variant has the same throughput. In the experiment 2 we get the highest throughput from the all TCP variants and lowest in the experiment 3 where the no of nodes are 40. But it varies as the number of nodes increases as shown in following graph:-

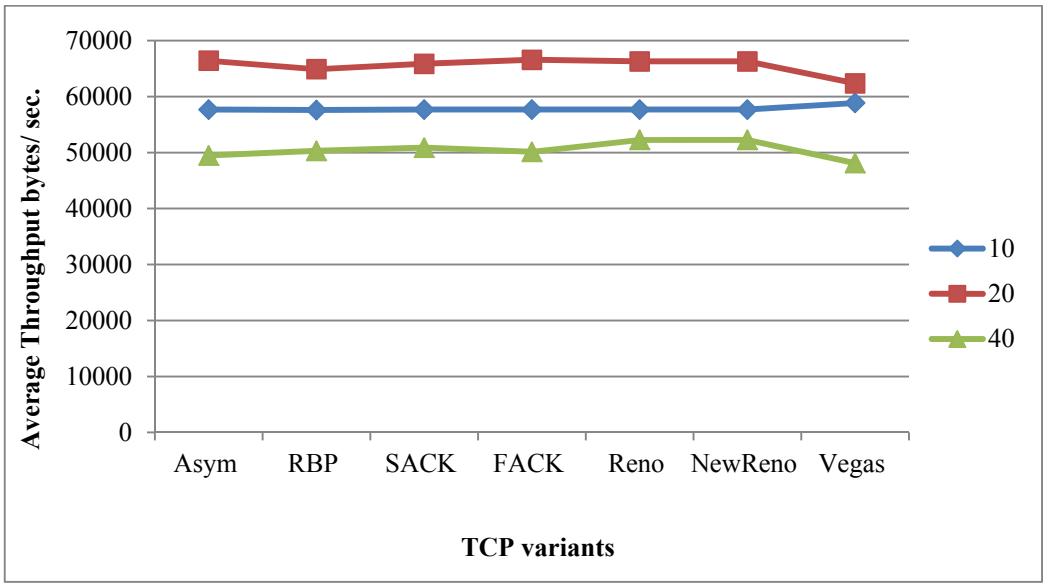

Fig.8. *Number* of nodes vs. Average throughput

3.4.5. Number of nodes vs. Total delay

It is calculated by minus the packet received time from packet generation time There are various reasons for packet delay in TCP. TCP RBP and Vegas both has a lowest delay rate than the others. Where TCP Reno is having highest delay as seen in the Fig. 9.

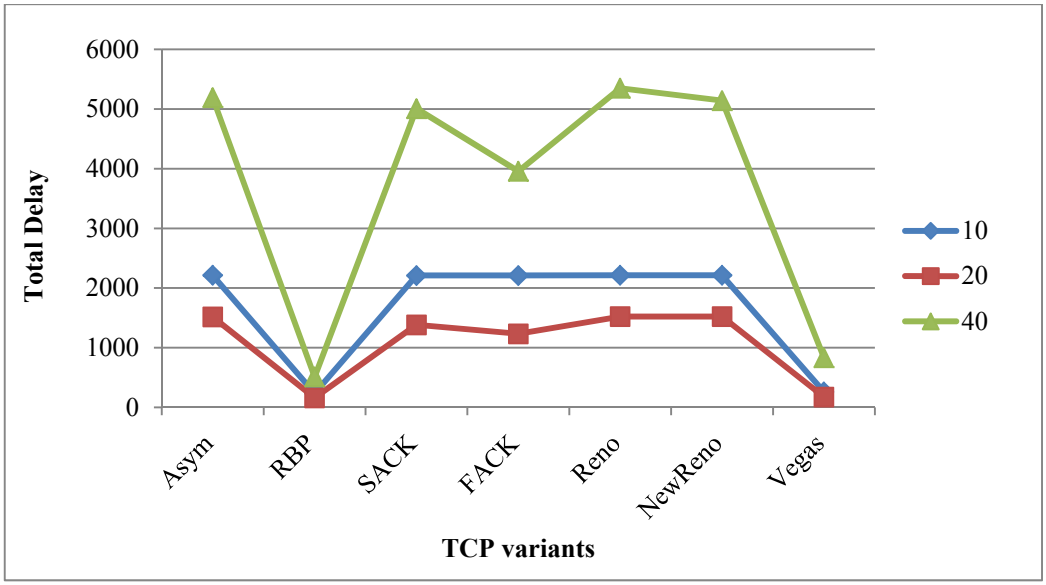

Fig.9. Number of nodes vs. Total delay

3.4.6. Number of nodes vs. Total jitter

Variations in delay of receiving packets, called jitter. In this paper, we evaluate the Total jitter that is a combination of random jitter and deterministic jitter. In the experiment 1 we can see that all variants have almost same jitter except RBP and Vegas. But when we reach at the experiment 2, jitter difference varies and in the experiment 3 where 40 nodes are used, Asym has higher jitter and RBP has lower jitter than all other variants.

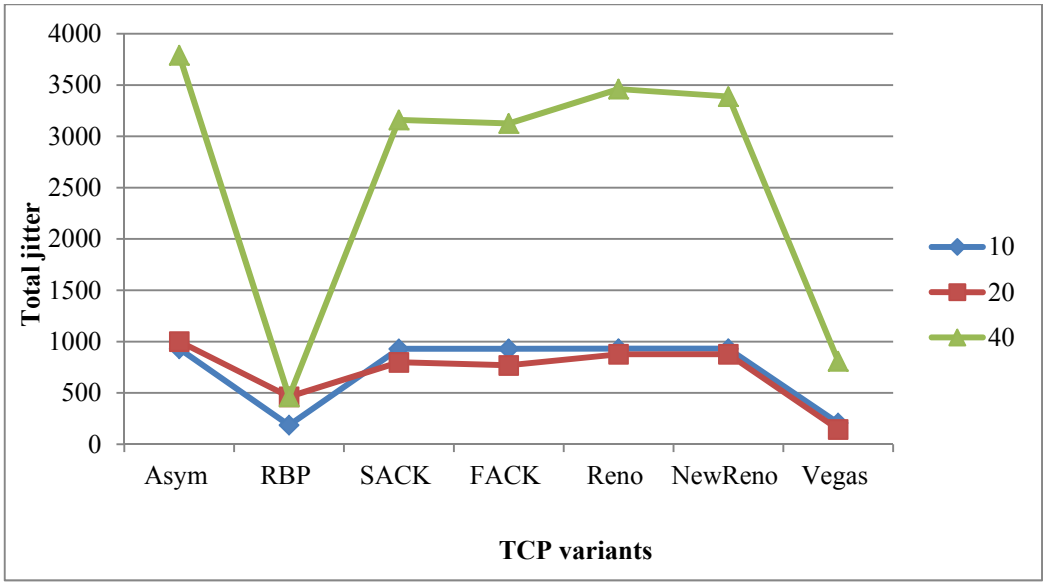

Fig.10. Number of nodes vs. Total jitter

3.4.7. Number of nodes vs. Average Delay

The average delay is based on the total delay. Fig. 11 shows that TCP RBP has lowest average delay and TCP Asym has the highest average delay.

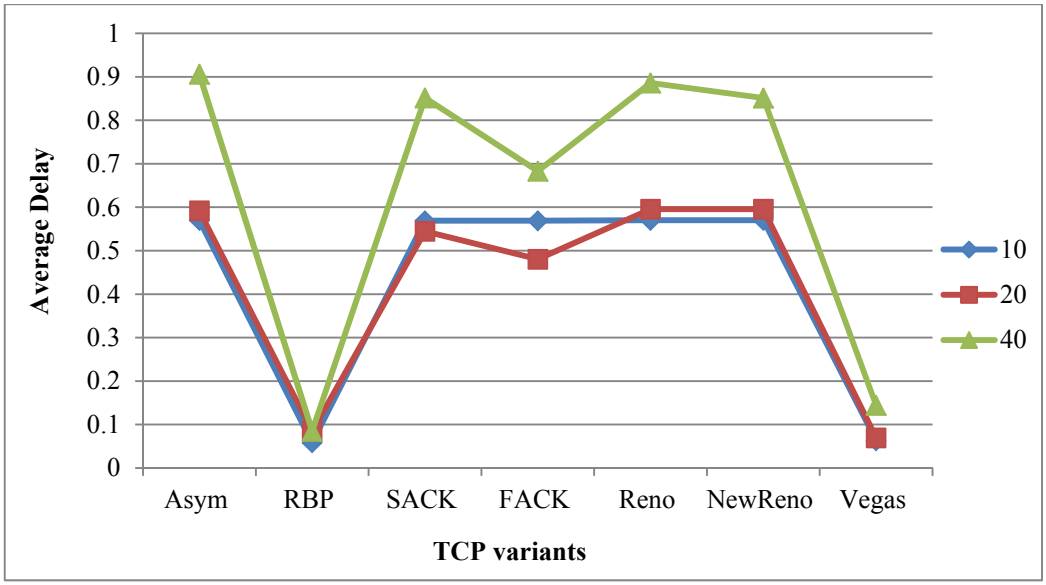

Fig.11. No. of nodes vs. Average Delay

3.4.8. Number of nodes vs. Average Jitter

Actually jitter is related to the delay because jitter is variations in the delay. So we have the almost same graph for average delay and average jitter. Similarly, in this graph TCP Asym has the highest and TCP RBP has the lowest average jitter.

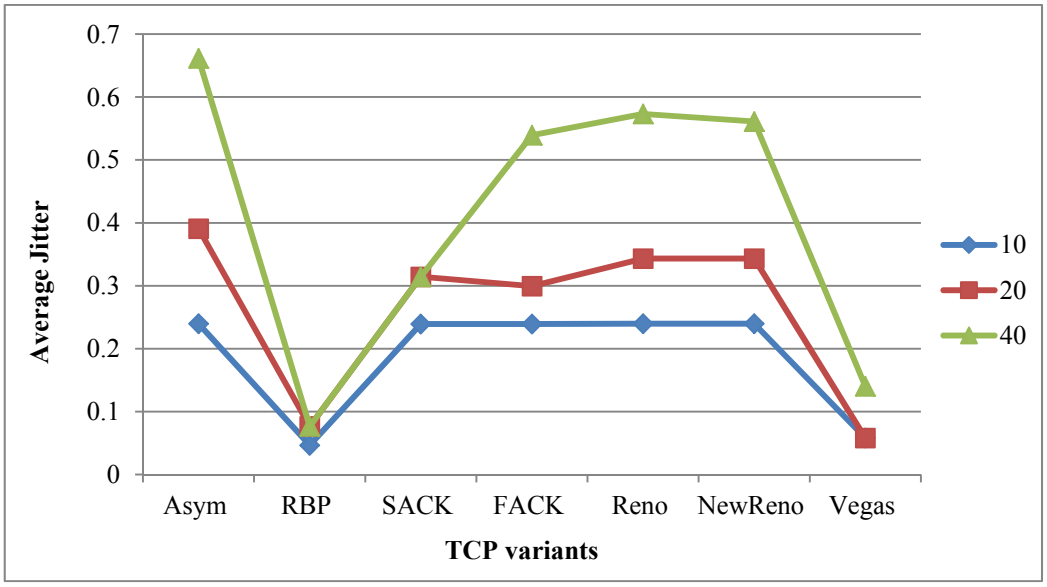

Fig.12. Number of nodes vs. Average Jitter

4. Result

Here we select the best TCP on the basis if above mentioned parameters. Each variant has different performance of different parameters. If a variant has low performance in a parameter, then it can be possible that the variant has the highest performance in another parameter. Table 4 shows the brief description about it.

PARAMETERS	BEST TCP
Number of Nodes vs. Number of Packets Dropped	Vegas
Number of Nodes vs. Number of packets sent	NewReno
Number of Nodes vs. Average Throughput in bytes/sec	NewReno
Number of Nodes vs. Delay	RBP
Number of Nodes vs. Jitter	RBP

Table 4. Best TCP variant in different parameters.

5. Conclusion

In this paper, we work on seven TCP protocols, TCP Reno, TCP New Reno, TCP SACK, TCP FACK, TCP Asym, TCP RBP and TCP Vegas. All these variants have different techniques to control congestion. We compare TCP variants using different parameters and select the best one on the basis of these parameters. This type of analysis is very helpful for selecting the best TCP protocol in required platforms.

References

[1] Bathla, N.; Kaur, A.; Singh, G. (2014): Relative Inspection on TCP variants Reno, NewReno, Sack, Vegas in AODV. International Journal of Research in Engg. & Applied Science, vol. 4, pp. 1-12.

[2] Chhabra, A.; Dhiman, A.; Joshi, M.; (2014): Performance Evaluation of Variants of TCP Based on Buffer Management over WiMAX. International Journal of Computer Science & Communication Networks, vol. 4, pp. 67-75.

[3] Fahmy, S.; Karwa, T.P. (2001): TCP congestion control: Overview and survey of ongoing research. Computer Science Technical Reports, pp. 1-12.

[4] Fall, K.; Floyd, S. (1996): Simulation-Based Comparison of Tahoe, Reno and SACK TCP. Computer Communication Review, vol. 26, pp.5-21.

[5] Gital, A. Y.; Ismail, A. S.; Chiroma, H. (2014): Performance Evaluation of TCP Cve Based on Cloud Computing Model. Journal of Theoretical and Applied Information Technology, vol. 70 No.1, pp. 9-18.

[6] Ho, C. Y.; Chen, Y. - C.; Chan, Y. - C.; Ho, C. - Y. (2008): Fast retransmit and fast recovery schemes of transport protocols: A survey and taxonomy. Computer Networks, vol. 52, pp. 1308–1327.

[7] Islam, M. S.; Kashem, M.A.; Sadid, W. H.; Rahman, M. A.; Islam, M. N.; Anam, S. (2009): TCP Variants and Network Parameters: A Comprehensive Performance Analysis. Proceedings of the International MultiConference of Engineers and Computer Scientists, vol. 1, pp. 978-988.

[8] Kandukuri, P.; Reddy, P. D. K.; Dr. Sam, R. P. (2016): Rate Based Pacing with Various TCP Variants. International Journal of Trend in Research and Development, vol. 3, pp. 356-359.

[9] Kaur, A.; Singh, G.; Singh, B. (2011): Evaluation of congestion control variants of TCP by strolling propagation delay in NS-2. National conference on Conveying Technologies Beyond 2020 (CTB-2020) & International Journal for Applied Engg. & Research, vol. 6, pp. 655-658.

[10] Kamboj, R; Singh, G.; (2015): VARIOUS TCP OPTIONS FOR CONGESTION EVASION. International Journal of Advanced Research in Computer Engineering & Technology (IJARCET). vol. 4 Issue 4, pp. 1534-1539.

[11] Khan, M. N. I.; Ahmed, R.; Aziz, M. T. (2012): A Survey of TCP Reno, New Reno and SACK over Mobile Ad-Hoc Network. International Journal of Distributed and Parallel Systems (IJDPS), vol. 3, pp. 49-63.

[12] Kushwaha, D. S.; Singh, V. K.; Singh, S.; Sharma, S.; (2015): Study of TCP Variants over Wireless Network. International Journal of Electrical, Electronics and Data Communication, ISSN: 2320-2084, vol. 3, pp. 62-67.

[13] Singh, G; Kumar, D.; Kaur, A. (2009): Simulation based comparison of performance metrics for various TCP extensions using NS-2. IEEE Sponsored International Conference on Innovative Technologies (ICIT-09), pp. 106-110.