Performance Analysis of TCP Variants

Rishabh Agarwal
Northeastern University, MA 02115
Email: agarwal.rish@husky.neu.edu
NUID: 001215275

Tanmay Sinha
Northeastern University, MA 02115
Email: sinha.t@husky.neu.edu
NUID: 001821288

ABSTRACT

This paper emphasizes on examining the performance of different TCP variants under the impact of congestion and queuing algorithms within a network. Various experiments were conducted using NS-2 simulator to examine their performance and figure out the throughput, packet drop rate and latency for providing congestion free network. The first experiment was to perform tests that analyze the performance of different TCP variants like Tahoe, Reno, New Reno, and Vegas in the presence of a Constant Bit Rate(CBR) flow. From the results obtained, Vegas proved to be the most effective variant in presence of a CBR flow. In the second experiment, we conducted experiments to analyze the fairness between a different pair of TCP variants under the varying CBR rate. Fairness is attained if they implement the same TCP variant when two TCP flows are present. In the third experiment, we have used DropTail and RED queuing algorithm techniques in the network topology for analyzing their behavior against SACK and Reno TCP variants. From the analyses obtained, the results in case of DropTail was better than that of RED.

1. INTRODUCTION

These days when an application is opened by multiple clients, a certain amount of data flows through a common network infrastructure. Thus, it becomes very important that a stable congestion control algorithm is implemented. Congestion control algorithms are responsible for equal distribution of the available resources among them. This tends to the inspiration driving the advancements consolidated by these congestion control calculations. This paper is an endeavor to watch the behavior of different TCP variations under the event of broadening network conditions with the end goal to settle on the most productive and solid variant among them.

Since Transmission Control Protocol centers around giving congestion control and unwavering quality over user datagram protocol, it is important to consider and assess noteworthiness of all the TCP variants against execution choosing parameters like throughput, latency, and packet drop. This paper likewise incorporates aftereffects of three distinct analyses for estimating their execution under congestion, for evaluating fairness and the criticalness of various queuing techniques on TCP variants. The distinctive TCP congestion control variants that will be talked about in this paper are Tahoe, Reno, New Reno, SACK, and Vegas.

2. METHODOLOGY

With the end goal to conduct these experiments, we have made utilization of NS2 network simulator utilized for simulating the packets over the network. NS2 gives significant help to simulation of TCP, routing, and multicast protocols over wired and remote systems. As NS2 can be composed either in C++ or Tcl for being object arranged, we have chosen Tcl as our scripting dialect for developing our network topology and planning diverse events inside the network. Because of the way that all tests require the presence of congestion inside a network, a lethargic Constant Bit Rate (CBR) application connected with UDP protocol was utilized for introducing congestion inside the network. Then again, FTP application was utilized as a traffic source for taking care of TCP transport layer protocol variants. All the duplex connections in a network topology were planned with link bandwidth of 10Mbps and delay of 10ms in the middle of packets transmission. Then again, queue limit parameter value between N2-N3 nodes has been set to an estimation of 10. In the primary experiment, we have dealt with 2 streams, one TCP flow out of N1 to N4 and another UDP flow out of N2 to N3. We set the data transmission rate of unresponsive CBR source in a scope of 1 to 10Mbps to incite congestion in the network.

In the second experiments, alongside nodes N1 and N4, nodes N5 and N6 have been arranged as TCP sender and sink individually. We made utilization of two TCP streams from N1 to N4 and N5 to N6 notwithstanding the inert UDP stream. We have increased the data transfer capacity of CBR source from 1 to 10 Mbps to analyze the fairness. In the third experiment, we have joined DropTail and RED queuing calculations in the network topology for watching their conduct against SACK and Reno TCP variants by keeping bandwidth of CBR source steady.

Every one of these analyses has been completed by utilizing NS2 simulator and Tcl scripting. When NS runs every one of the scheduled events as depicted in Tcl scripts, every one of the packets started from all the active traffic sources gets caught in a trace document. We have utilized Python as a scripting dialect to parse these records to compute throughput, latency, and packet drop rate. At that point, all these parsed records and relating results of performance deciding parameters have been utilized to plot diagrams. MS Excel was utilized to produce different diagrams and dissect the parameters.

3. NETWORK TOPOLOGY

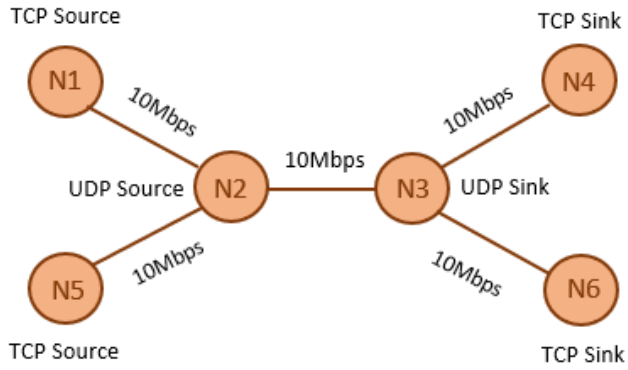

Fig1: Network Topology

The figure above shows the network topology. We have utilized a similar topology for all the experiments. We have made utilization of 6 nodes from N1 to N6 associated as appearing in Fig1 to fabricate our network fusing TCP and UDP sources. Node N1 has been arranged as a TCP sender which would be utilized for actualizing one of the TCP variants like Tahoe, Reno, New Reno, Vegas, and SACK. The relating TCP sink has been arranged at node N4 to permit packets to flow from N1 to N4. The File Transfer Protocol (FTP) application has been connected to node N1 for activating TCP sender to send packets to the TCP sink. We have even arranged node N2 to behave like a UDP source and node N3 as a UDP sink. Both the nodes N2 and N3 are associated with one another to permit UDP traffic started by Constant Bit Rate source to stream between N2 to N3 nodes. All the links have been planned with link bandwidth of 10 Mbps.

4. TCP PERFORMANCE UNDER CONGESTION

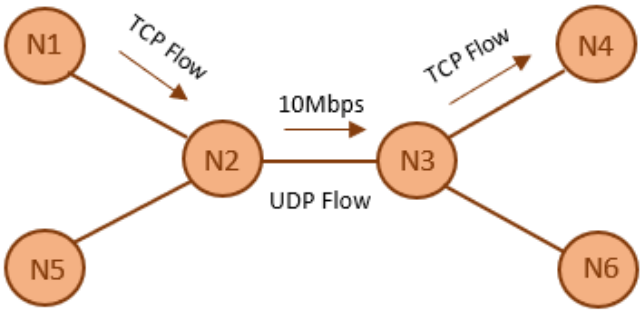

Fig2: Experiment 1 Network Topology

This experiment apprehensions with calculating the throughput, latency, and packet drop rate for a TCP variant against the consistent bit rate UDP flow. The TCP packets flow from node N1 to N4 and UDP packets initiated by CBR source travel from N2 to N3. We will upsurge the bandwidth of CBR source from 1 to 10Mbps to allow congestion to happen. As needs are, as we continue shifting the bandwidth of CBR source, we will all the while compute all the three parameters and contrast their outcomes to examine performance for TCP variants Tahoe, Reno, NewReno, and Vegas. To begin with, we will begin the UDP stream at a specific purpose of time and afterward the TCP stream in the meantime interim. In the wake of running both the occasions for 10 seconds, we would stop the simulator to check for the trace file output.

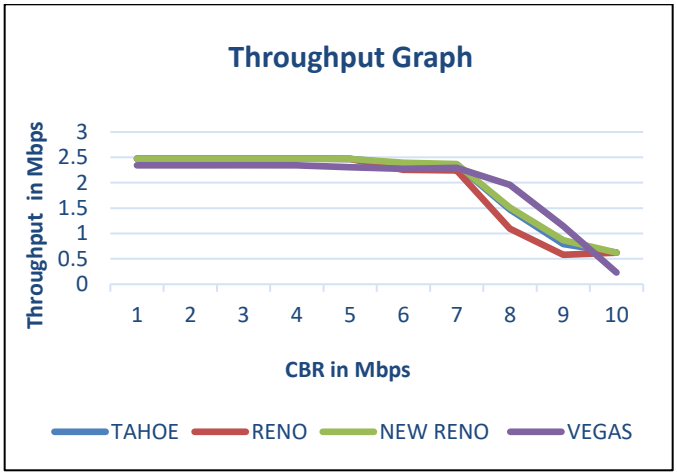

Fig3: Throughput evaluation

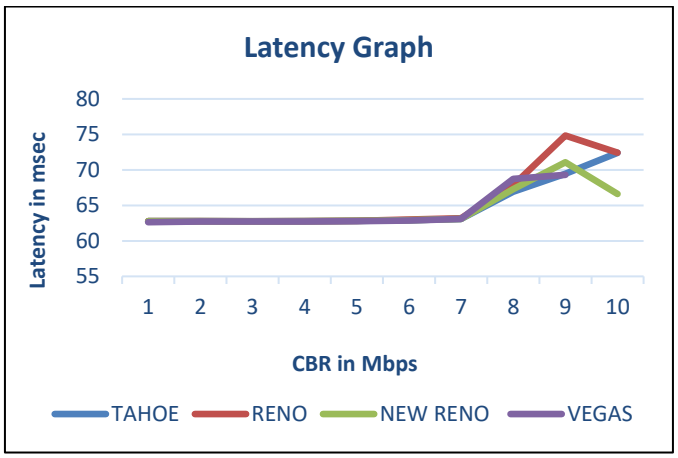

Fig4: Latency evaluation

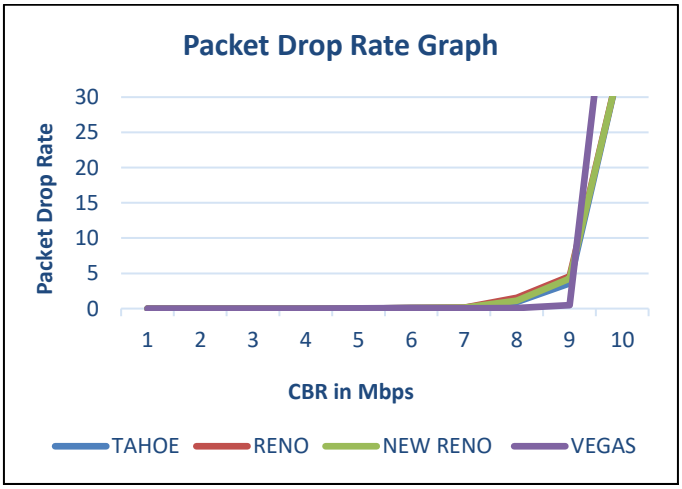

Fig5: Packet Drop Rate evaluation

Result:

In the wake of parsing the required information from the trace file utilizing Python scripting, throughput has been estimated for 4 distinctive TCP variants by changing the CBR from 1 to 10Mbps. Throughput is characterized by taking the proportion of total packet size received by TCP receiver to the total duration of the time taken for the experiment. From the throughput diagram Fig3, it is seen that all TCP variants demonstrate comparative conduct before congestion or packet drop occurs in a network. When the CBR source begins introducing congestion within a network by growing its data transmission rate, throughput began falling due to the packet drop. We notice that when CBR rate reached beyond a value of around 7.2Mbps, we got a downward graph suggesting dropping throughput. We have figured out from the throughput graph that Vegas provides the highest throughput than all other TCP variants.

The latency is one of the parameters beneficial for evaluating the performance of TCP variants. It is computed by taking the ratio of total RTT taken for all the acknowledged packets to the total number of packets acknowledged at the sender. The Fig4 demonstrates the inactivity diagram plotted for individual TCP variants against shifting CBR bandwidth. According to the chart, it is seen that the latency indicates comparative behavior up to 7.5Mbps. Be that as it may, when the CBR stream continues to increase, latency additionally increments because of an expansion in queuing and propagation delays inside nodes bringing about including the average RTT taken by every recognized packet. The latency increments and goes to the most elevated value when link bandwidth approaches the bottleneck limit. We have analyzed from latency chart that Vegas offers the lowest average latency when contrasted with others in presence of congestion.

The packet drop rate is figured by taking the proportion of collective no of packets dropped to the collective number of packets sent by the TCP sender. The Fig5 demonstrates the packet drop rate chart plotted by taking packet drop rate value for every one of the TCP variants over CBR data transfer capacity. According to the diagram, all TCP variants carry on similarly till the CBR data transmission reaches to 7.5Mbps but as the value approaches close to the connection's bottleneck limit, packet drop rate exponentially increments as congestion begins happening in the network. Since 10Mbps value has been set as bottleneck link value for CBR source, its value shoots up to an abnormal state for all the TCP variants when CBR comes to close to 10Mbps. We have seen from this diagram since the congestion bound to increment with increment in CBR data rate, the drop rate likewise increments, however, Vegas figures out how to have a lower drop rate distinguished with others.

Vegas offers the highest average throughput, least latency, and least packet drops because of its capacity to foresee congestion inside a network. It implements congestion control dependent on the packet delay understanding rather than packet drop. It checks for a timeout in an effective way and does not sit tight for the adequate no. of duplicate acknowledgments to recognize packet loss or congestion, in contrast to other TCP variants. Vegas enhances re-transmission system of Reno by monitoring sending an acknowledgment time for each fragment. By assessing a base RTT and contrasting it and the genuine RTT, it identifies congestion without sitting tight for any clock expansion or triple copy acknowledgment. On the off chance that the genuine RTT falls underneath base RTT, it increases sending window size. Additionally, on the off chance that it goes above base RTT, it decreases sending window size to anticipate congestion. Vegas does not expand its congestion window exponentially during its slow start phase. Rather, it builds a congestion window in an exponential way relying on the RTT as opposed to all other TCP variants. Because of this reason, the throughput of Vegas at an underlying phase of experiment returns a low throughput than all other TCP variants. If the distinction among actual and estimated base RTT increments over a specific edge, it leaves slow start phase and goes into the congestion avoidance. This system enables Vegas to perform superior to other TCP variations.

Conclusion:

For experiment 1, we can accomplish that TCP Vegas turns out to be the overall best variant among because of the way that it offers the most noteworthy throughput, least latency and most reduced packet drop rate under outrageous congestion conditions. Be that as it may, this situation may vary if we include another TCP source stream in our network topology close by CBR stream.

5. FAIRNESS BETWEEN TCP VARIANTS

This analysis is completed to examine the fairness between two TCP variants by the differing bandwidth of inert CBR source. Since we would manage two TCP variants, we would make utilization of complete network topology including TCP streams from nodes N1-N4 and from N5-N6 by appointing a couple of TCP variants. We would likewise be utilizing CBR flow out of N2-N3 for including congestion in the network. The different mixes that were assessed are Reno-Reno, New Reno-Reno, Vegas-Vegas, and New Reno-Vegas.

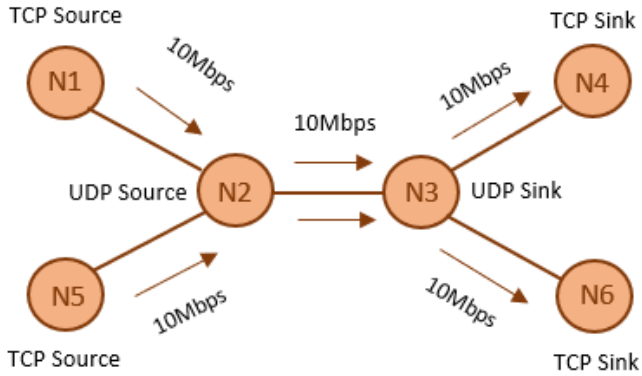

Fig6: Experiment 2 Network Topology

The second experiment spins around choosing fairness between a couple of TCP variants. We will fluctuate the bandwidth of CBR source like the first experiment to calculate values of throughput, average latency, and packet drop rate as an element of bandwidth. This experiment is for the most part being done to check if a couple of the TCP variants acts fairly to one

another. We have started both the TCP streams at 0.1 seconds while CBR stream has likewise been introduced at the equivalent time. We have built up our simulation script by distributing 10 seconds to the simulator for running the experiment. By changing the bandwidth of CBR from 1 to 10Mbps, we would compute and look at throughput, latency, and packet drop rate for each combination of variants. By following the above examination, we would be checking if the two variants get an equivalent share of network bandwidth as far as transmission capacity usage for assessing if they go about as reasonable for one another.

1. Reno-Reno

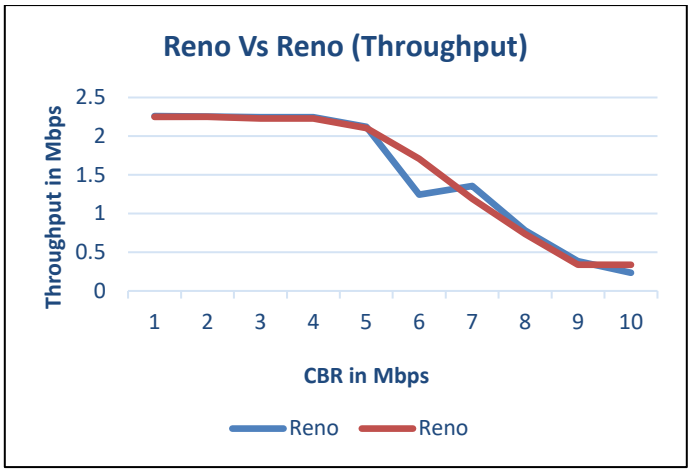

Fig7: Throughput for Reno-Reno

Results:

The above charts sketched out throughput and packet drop rate for a Reno-Reno TCP variant. From throughput diagram (Fig7), both utilize equal share of data transfer capacity by giving a similar measure of throughput. Because of this, the blend of Reno/Reno is reasonable for one another in presence of congestion.

2. Vegas-Vegas

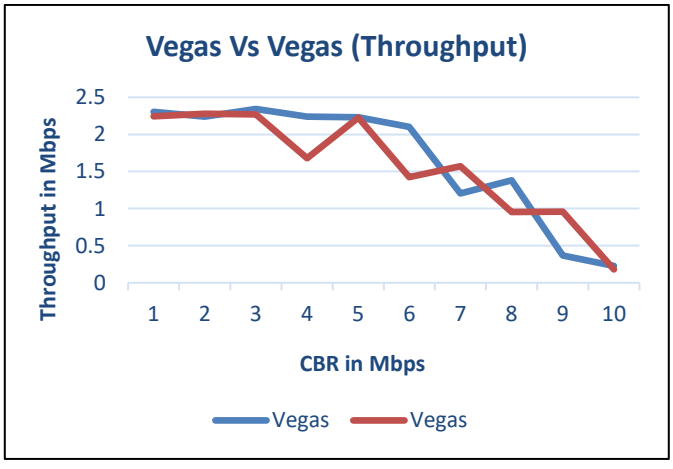

Fig8: Throughput for Vegas-Vegas

Results:

In the wake of referring to the Throughput chart of Vegas-Vegas (Fig8), we can identify with the way that the two streams share an equivalent measure of bandwidth at lower CBR rates and with an increase in congestion, both the Vegas streams gain strength over certain CBR range.

3. NewReno-Reno

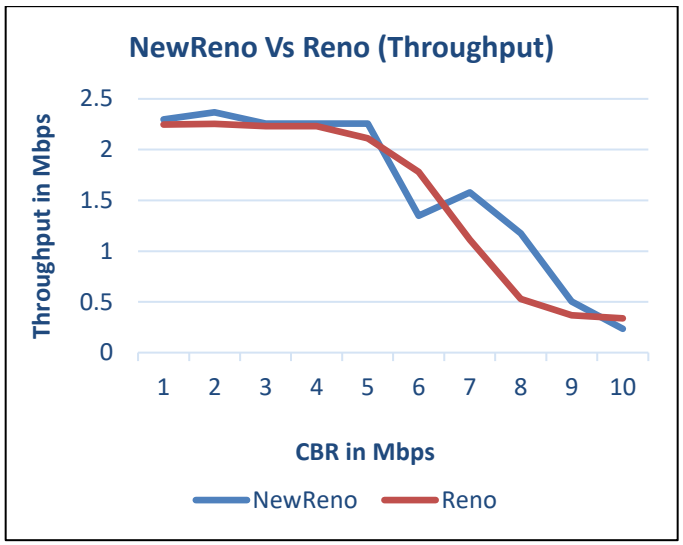

Fig9: Throughput for NewReno-Reno

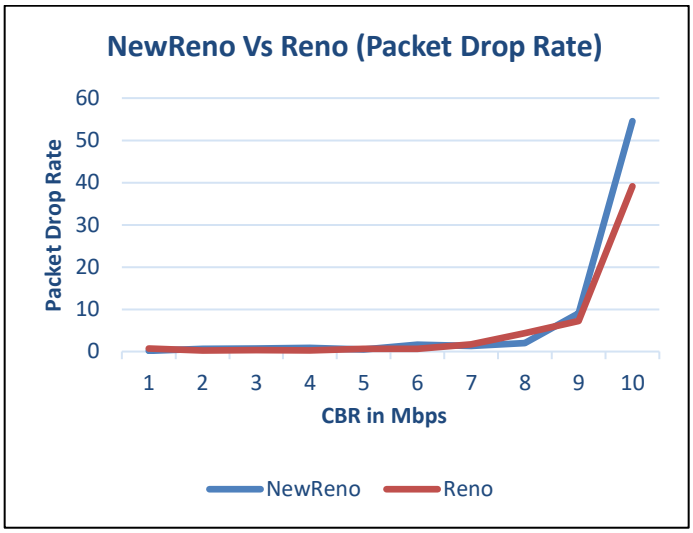

Fig10: Packet drop rate for NewReno-Reno

Results:

The mix of NewReno-Reno is seen to be fair in low CBR rates yet with an increase in congestion, NewReno supposedly is going about as unfair to Reno (Fig9). This is because of the distinction in congestion control calculation utilized by Reno and NewReno. NewReno is equipped for taking care of various packet drops with an increase in congestion. In contrast to Reno, the incomplete ACK's don't remove NewReno variant from the quick recovery mode. Under a situation of various packet drops in a single window, it stays in quick recovery until the point when all the remaining information is exchanged. We can watch the varieties in throughput because of Additive Increase Multiplicative Decrease (AIMD) congestion control calculation utilized by NewReno.

4. NewReno - Vegas

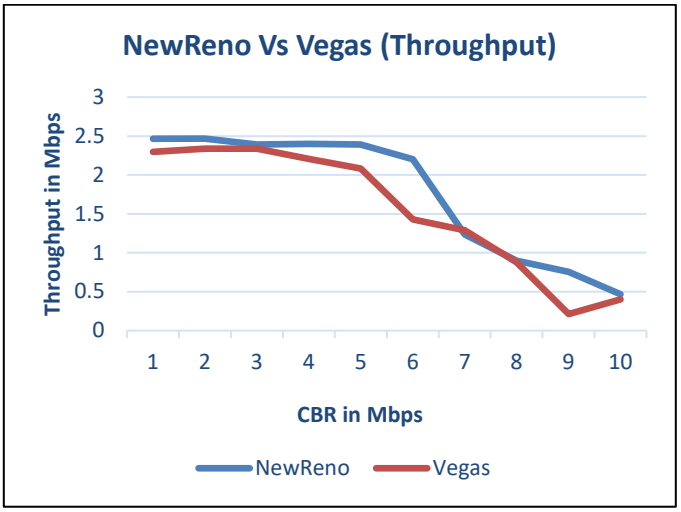

Fig11: Throughput for NewReno-Vegas

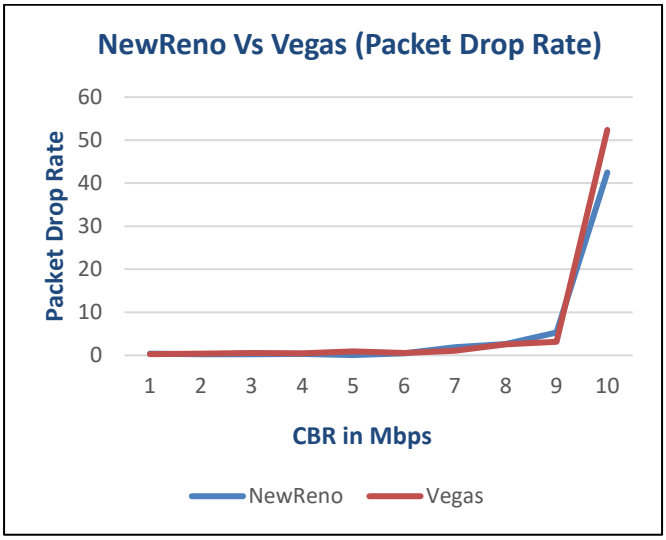

Fig12: Packet drop rate for NewReno-Vegas

Results:

The above diagrams show throughput and packet drop rate values for NewReno and Vegas TCP variant. From Fig11, it was noticed that NewReno offers higher throughput than Vegas. New Reno upsurges its congestion window until the point that it experiences a packet drop which is the principal sign for congestion. Then again, Vegas does not wait for the clock expansion or triple duplicate ACKs to happen for congestion detection. It handles congestion by evaluating the delay in the link and progressively changing the congestion window. When Vegas predicts the congestion, it reduces the packet transmission rate. By following this methodology, Vegas recognizes congestion in the link and prevents an expansion in the congestion window. In this way, other TCP variants like NewReno gets more data transmission to give higher throughput. Thus, New Reno gives off an impression of being unfair to Vegas.

Conclusion:

We can finish up from this experiment that a similar match of TCP variants for the most part act fair for one another although an alternate combination of TCP variant tends to act unfairly to one another because of the distinctive congestion control component sent by them.

6. INFLUENCE OF QUEUEING

This experiment is performed to investigate the effect of queue administration calculation in a network by keeping the bandwidth of the CBR source steady. We would deal with DropTail and RED (Random Early Drop) queuing calculations to figure throughput for Reno and SACK TCP variants. In Drop tail, when the queue gets completely filled up, it drops ensuing arriving packets until the point that the queue offers adequate space for accommodating incoming traffic. Whereas, RED conveys issues tending to unfair distribution of buffer space among various data streams.

For this experiment, we have kept the CBR rate steady at 6Mbps. The aggregate span of this experiment is 25 seconds. The TCP stream begins at 0 seconds and goes ahead till 25 seconds though the UDP stream begins at 5 seconds and stops at 25 seconds. The TCP window size is chosen as 150 and maximum cwnd measure is 200. After setting up every one of these parameters, average throughput and average latency are computed after some time rather than CBR rate for Reno and SACK TCP variants.

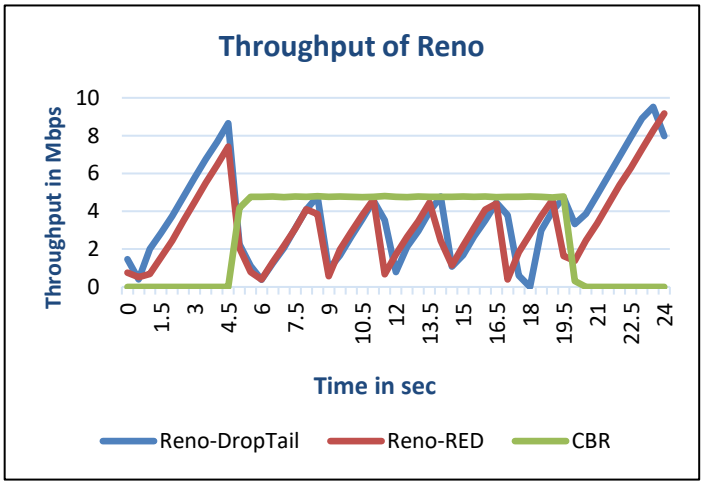

Fig13: Throughput for Reno DropTail and RED

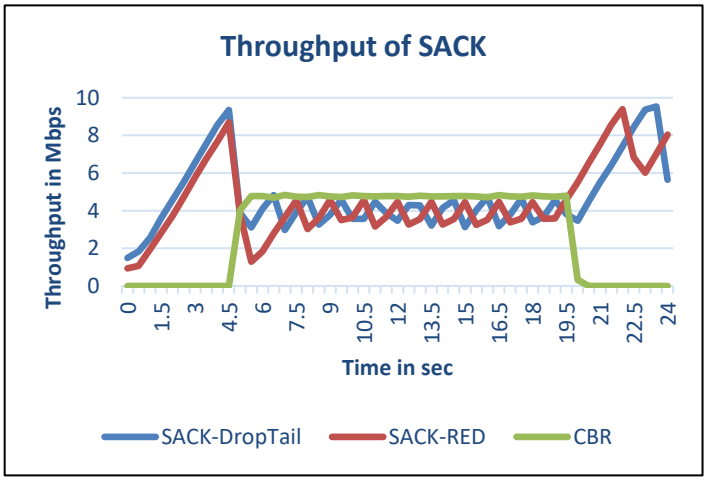

Fig14: Throughput for SACK DropTail and RED

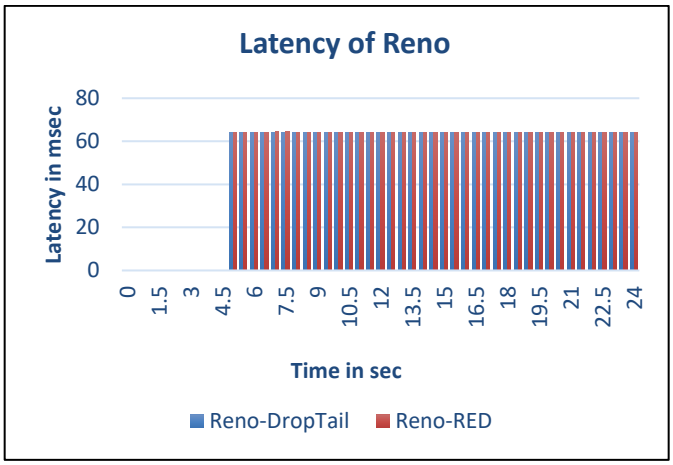

Fig15: Latency for Reno DropTail and RED

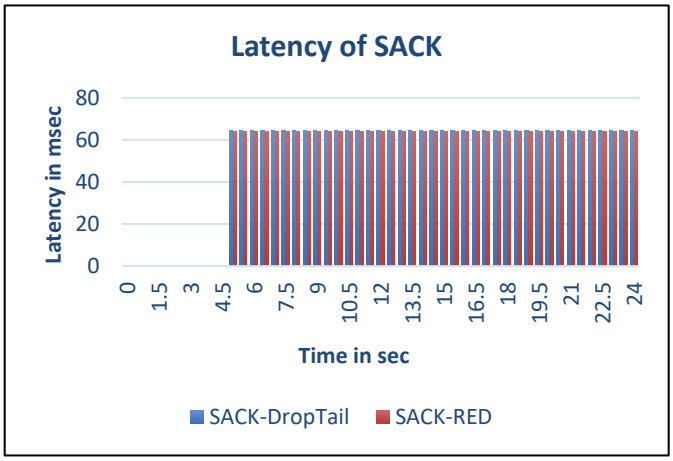

Fig16: Latency for SACK DropTail and RED

Results:

The Fig13 and Fig14 speak to the throughput performance of Reno and SACK separately in presence of DropTail and RED calculations. Random Early Drop(RED) is a queue management algorithm which monitors the normal queue size and as needs are, performs resulting actions on packets (either drop or check) on the basis of probabilities. It takes a shot at the phenomenon of keeping the queue size between a minimum and maximum threshold values. When the queue estimate goes over the greatest limit value, RED calculation notices congestion and begins dropping packets with a likelihood proportional to the normal queue size and the aggregate number of transmitted packets. This methodology of RED for recognizing congestion drive TCP stream to go into congestion avoidance state even before the queue turns out to be full. Thus, RED deals with giving fair usage of bandwidth.

Then again, the Drop Tail queueing algorithm does not check for any queue size for setting any confinement on packets transmission. It enables packets to stream constantly until the queue turns out to be full. At whatever point the queue goes full, DropTail begins dropping packets. This technique prevents DropTail from giving fair utilization of BW. Also, the RED algorithm, for the most part, begins dropping packets even before the queue turns out to be full. Subsequently, the end-to-end latency viewed for RED is more than that of DropTail. When the CBR stream begins introducing in the network, congestion happens which thus forces packets getting dropped for the two kinds of streams prompting a decreased throughput for that moment.

The diagrams (Fig14 and Fig16) show the performance of SACK with RED is poor when contrasted with others. Since SACK recognizes congestion on the event of a packet loss, the RED algorithm takes a shot at the approach of dropping packet even before the queue is full. Thus, SACK with RED is not a good alternative over DropTail.

7. CONCLUSION

Diverse sorts of TCP variants carry on differently relying on the level of network congestion and the presence of various kinds of flow in the network. Considering experiment 1, we analyzed the performance of four TCP variants under differentiating load conditions in presence of a CBR stream. For this situation, we concluded that Vegas gives the best execution as far as throughput, packet drop rate, and latency, to the way that TCP Vegas identifies congestion in a network, dependent on the packet delay experienced rather than packet drop.

An extensive range of applications make utilization of TCP as transport layer protocol and every application may not incorporate a similar sort TCP variant during implementation to control congestion. Thus, with the end goal to have TCP variant to give the best performance we should know which mix of TCP variants perform well with one another with the end goal to get fair use of bandwidth. The outcomes got from experiment 2 concludes that the blend of TCP variants taking care of similar sort of congestion control algorithms are comparatively fair.

In experiment 3, we examined the impact of various queueing networks like DropTail and RED against Reno and SACK TCP variants. RED guarantees fairness inside a network by arranging the probabilistic methodology of packet dropping which additionally came about into getting the reduced throughput. This demonstrates RED can be utilized with different users that offer huge date. IP can be one of the users of RED.

Future extension identifies with tending to numerous other TCP variants like BIC, CUBIC, and so on that can enhance the congestion control in networks alongside their individual performances. Since TCP Vegas has turned out as the most proficient decision, we can check the Vegas performance parameter nearby RED queuing technique. Indeed, even examination of such two diverse queuing procedures on two distinctive networking topology sounds inspiring.

8. REFERENCES

[1] http://nile.wpi.edu/NS/
[2] https://www.isi.edu/nsnam/ns/tutorial/index.html
[3]http://zfadlullah.org/files/coursematerial/informationengineering/Additional/ns_kiso.pdf
[4]http://www.mathcs.emory.edu/~cheung/Courses/558a/Syllabus/6-transport/TCP.html
[5] https://en.wikipedia.org/wiki/TCP_congestion_control