PERFORMANCE ANALYSIS OF TCP VARIANTS

Balaji Mani, Yuyang Zhang
Northeastern University
360 Huntington Avenue, Boston 02115
mani.b@husky.neu.edu , zhang.yuya@husky.neu.edu

Abstract: Transmission Control Protocol (TCP) is designed for reliable communication, but even though its reliable it couldn't perform well in large and congested networks. To overcome this several TCP variants like TCP Tahoe, Reno, NewReno, Vegas, SACK and others are proposed., here in this paper we analyzed the performance of those variants by measuring its congestion, fairness and its effect of DropTail and Random Early Detection (RED) queueing algorithms using Network Simulator (NS-2). Based on the obtained results, we calculated different parameters like average throughput, latency, drop rate and plotted graph to present best TCP variants for any particular conditions.

Introduction

The Transport layer one of the vital layer of OSI Model, consists of two main protocols TCP and UDP responsible for process to process communication. User Datagram Protocol (UDP) is connectionless protocol and it is used only when reliability is not of primary importance. On the other hand, Transmission Control Protocol (TCP) is widely known connection oriented protocol and it provides features like Congestion control, Flow control along with multiplexing and de-multiplexing of the messages. Although TCP provides reliability to the process, but it fails to provide acceptable performance in large congested networks.

To improve the TCP performance of the congested networks, different TCP variants have been proposed by researchers some of them are TCP Tahoe, Reno, NewReno, Vegas, SACK, Compound and CUBIC, these are equipped with different features like Slow Star, Fast Retransmit, Fast Recovery, Congestion Avoidance and acknowledging of out-of-order packets. We in this paper analyzed below TCP variants are as follows:

TCP Tahoe: It uses principles of Additive Increase Multiplicative Decrease (AIMD) and it detects congestion based on (cwnd) congestion window and number of unacknowledged packets. TCP Tahoe resend the ACK packet based on its associated timer, when timer associates with timer got expired then that particular packet is resent and also other parameters like slow start threshold (ssthresh), congestion window (cwnd) is updated accordingly. The slowstart is mechanism behind working of TCP Tahoe, initially TCP Tahoe exponentially increases (cwnd) to rapid phase till reaches threshold (ssthresh), Once it reaches, it then reduces the (ssthresh) to (cwnd)/2 and also set the cwnd to 1 and will start slowstart phase again.
Disadvantages: It's performance on congestion control is reduced by waiting for time-out to detect packet loss.

TCP Reno: To overcome the drawbacks of TCP Tahoe, Reno uses implementation of fast retransmit and fast recovery algorithms. Here in this approach instead of waiting for time-outs for packet loss, the retransmission happens only when sender receives three duplicate ACKs which indicates the packet has been lost and after this TCP Reno enters into Recovery phase here instead of resetting (cwnd) to 1 (only during actual timeout), it resets to (ssthresh/2) which will avoid expensive timeouts and preventing TCP not to enter slowstart phase.
Advantages: Its performance is better for less losses

TCP NewReno: It is improved version of earlier TCP variant Reno, here in this process instead of waiting for three duplicate ACKs for lost packet, TCP NewReno retransmits new packet for every new duplicate packet received. TCP NewReno makes use of partial ACK to transmit new packet resulting in improved fast retransmit process when compared to Reno. It is highly effective for single packet loss than multi packet losses.

TCP Vegas: It works based on efficient measure of Round Trip Time (RTT), due to which it has effective slowstart and better retransmission mechanisms. Unlike earlier TCP variants, TCP Vegas emphasizes on congestion avoidance algorithm on packet delay than on packet loss.

TCP SACK: It is improved version of TCP Reno used for detecting multiple packet loss. Usually in earlier variants, Sender can only learn about single lost packet per RTT but at the same time there is the possibility some other packets have been successfully received by the receiver. To overcome this SACK acknowledgment from receiver gives sender the update of the packet that has sent a positive acknowledgement.

Here in this paper, we used three experiments below to analyze the performance of TCP variants by calculation of different parameters like average throughput, latency, drop rate under different conditions like increase/decrease in size of data, congestion window cwnd and different queuing algorithms. The obtained results are plotted with respect to bit rate and time and presented at the latter section of the paper.

Experiment 1: TCP Performance Under Congestion
Experiment 2: Fairness Between TCP Variants
Experiment 3: Influence of Queuing

II. Methodology

As part of Methodology is concerned, we used Event driven network simulator called NS2, it is used in simulating variety of IP network protocols such as TCP, UDP and other source behavior such as Telnet, FTP, CBR and router queue management mechanism such as DropTail and Random Early Detection (RED). We used below topology consisting of 6 Nodes for all the three experiments

Nodes Description: Here in this below topology, we used Nodes N1 and N5 as TCP source where we used FTP File Transfer Protocol applications for it. At the Nodes N4 and N6, we configure TCP Sinks to receive the TCP Data.

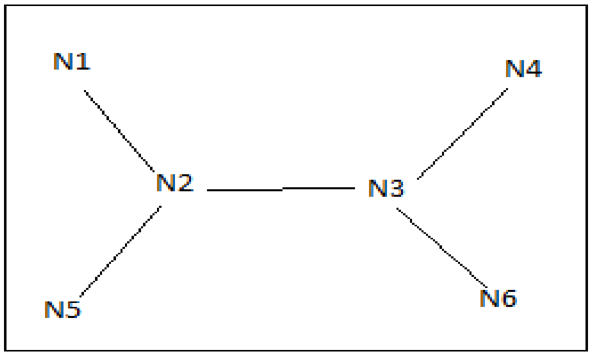

Figure 1: Network Topology

Now for the Node N2, UDP is configured and Constant Bit Rate (CBR) Traffic generator is attached to it, while at the Node N3 we configured it to be CBR Sink.

Here in these experiments we used CBR as an unresponsive UDP flow that will be sent constantly at the given specified rate varying from 1 Mbps to 10 Mbps for all TCP variants (TCP Tahoe, Reno, NewReno, Vegas, SACK). *For experiment 1 and experiment 2, We configured the bandwidth and delay of the links to be 10 Mbps and 2 ms, TCP window to 1000 and CBR packet size to be 5000. For experiment 3, we configured TCP window size to be 10000 and CBR packet size to be 1000.* For all the 3 Experiments, we compared the performance of TCP variants by varying CBR, queue length and for Queuing algorithms either DropTail or RED. Once the required results are obtained we used (gawk) to extract relevant data into .csv files and then we prepared graphs with help of Microsoft excel.

At the end of all the three experiments, we can able to answer some of the questions like, Which TCP variant(s) are able to get higher average throughput? or the lowest average latency? or Which has the fewest drops? And also other questions like fairness and which TCP variant provide fair bandwidth can also be answered after the analysis.

Calculation:
Average Throughput (Kbps) = Total # of bits received at application layer / (Total amount of time * 1024)

Average Latency (Seconds) = Sum of delay obtained for all packets / Total # of packets sent

Packet Drop Rate = (Total # of packets sent/ Total # of packets received) / Total # of packets sent

III. Experiment 1: TCP Performance Under Congestion

For the experiment 1, we configured the network topology shown in the figure 1. We configured link between each node to be bandwidth of 10 Mbps and 2 ms delay. For measuring performance under congestion we added TCP source at N1 & TCP sink at N4 and CBR source at N2 & CBR sink at N3. We varied the value of CBR Data Rate from 1 Mbps to 10 Mbps and measure the Throughput, Latency and Drop rate of TCP Variants (Tahoe, Reno, NewReno, Vegas). We used following scenarios for simulation of TCP Performance i) Starting CBR and TCP at same time with queue size of 30, ii) Starting CBR and TCP at same time with queue size of 100, iii) Starting TCP first and then CBR with queue size of 30 and finally iv) Starting CBR first and then TCP with queue size of 30.

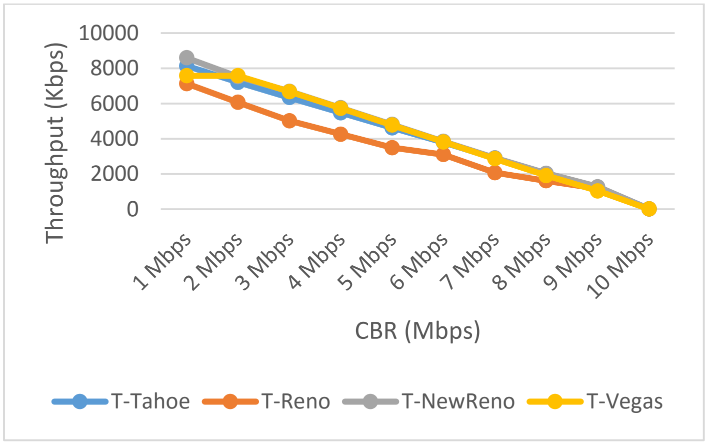

Figure 1.a, Average throughput of TCP variants vs CBR Flow with queue size 30 and starting TCP and CBR at same time

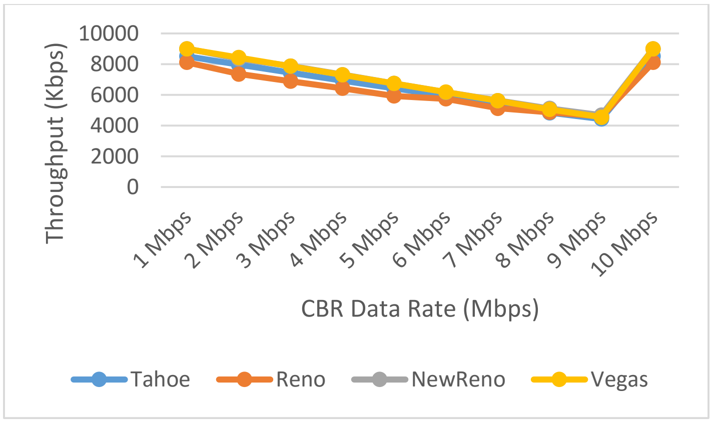

Figure 1.b, Average throughput of TCP Variants vs CBR data flow with queue size 30 and starting TCP first

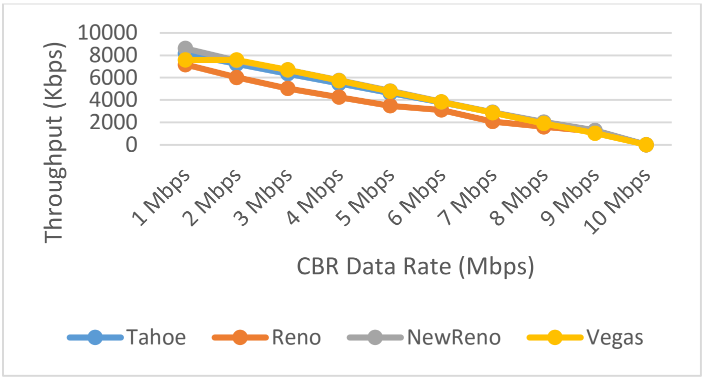

Figure 1.c, Average throughput of TCP Variants vs CBR data flow with queue size 30 and starting CBR first

From figures 1.a to 1.c, We have analyzed that average throughput of TCP Vegas is high when compared to other TCP variants, this is because TCP Vegas implements delay based congestion avoidance method when compared to ACK method in other TCP variants. Next to TCP Vegas we analyzed that TCP NewReno Variant performs better in terms of average throughput. This is because TCP NewReno uses improved Fast Retransmit mechanism to resend the new packet for each partial ACK received.

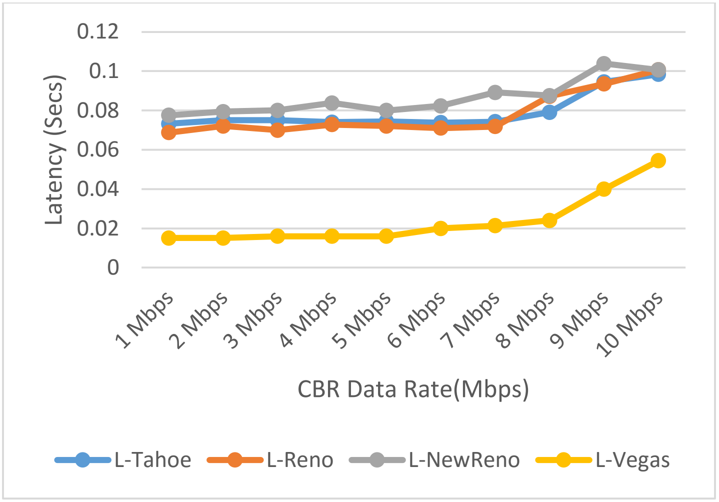

Figure 1.d, Average latency of TCP variants vs CBR data rate with queue size and starting TCP & CBR at same time

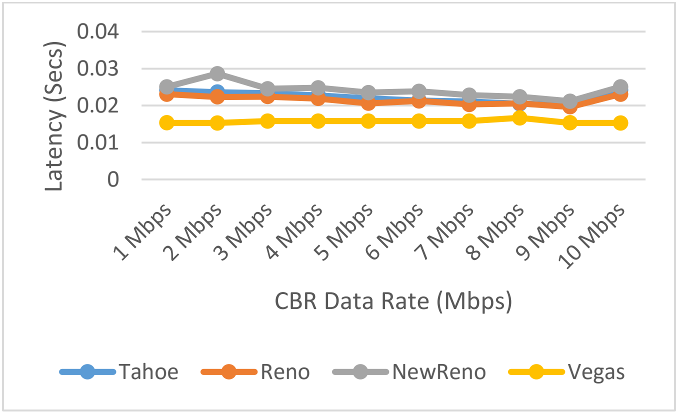

Figure 1.e, Average latency of TCP variants vs CBR data rate with queue size 100 and starting TCP first

From figures 1.d and 1.e, we analyzed TCP Vegas showing better performance yielding less latency, this is due to its delay based congestion avoidance approach and its measurement of RTT, while other TCP variants follows ACK method leading to heavy packet loss.

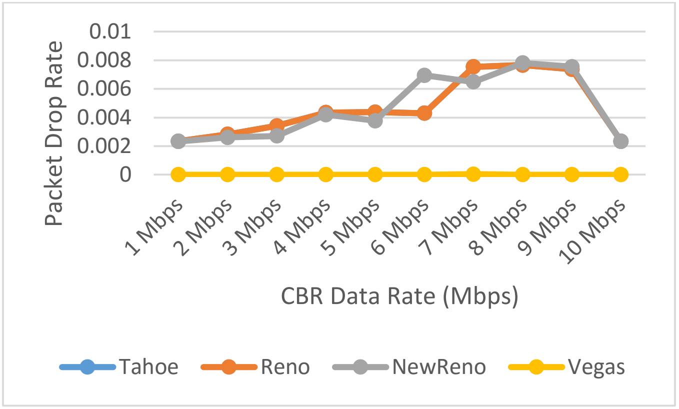

Figure 1.f, Packet Drop rate of TCP variants vs CBR data rate with queue size 30 and starting TCP first

For figure 1.f, we again found TCP Vegas to be performing better to have less or zero drop rate. This is because TCP Vegas probes and detects the congestion in channel very quickly because of delay based congestion avoidance approach and also it found that for various scenarios like "Starting TCP first and starting CBR first and both at the same time" Drop rate of Vegas founds to be of near zero drop rate, but in the same time other TCP variants suffers from varying congestion window sizes due to which packet loss occurrence is higher.

We also cross verified the obtained values by *performing T-Test for all TCP variants (Tahoe, Reno, NewReno and Vegas) where the ideal value of p-value should have to be <0.5, After performing T-Test, we obtained following values, for Vegas- p =0.0046, NewReno- p=0.0075, Reno- p=0.001218 and Tahoe – p=0.007968.* After analyzing these values, we concluded that all the obtained values are stable with respect to each other of TCP variants.

Finally, at end of Experiment 1, We came to conclusion that TCP Vegas shows better performance in all Throughput, Latency and Drop Rate when compared to other TCP variants (Tahoe, Reno and NewReno).

III. Experiment 2: Fairness Between TCP Variants

In Experiment 2, we used same network topology that we used in experiment 1, instead we additionally used one more TCP flow configuring TCP source and sink at nodes N5 and nodes N6 respectively. As in experiment 1, CBR rate is varied from 1 Mbps to 10 Mbps to measure fairness and latency among the TCP variants (Newreno_Reno, NewReno_Vegas, Reno_Reno, Vegas_Vegas)

From figures 2.a and 2.b for TCP NewReno and Reno combination, we analyzed that NewReno throughput is performing better than Reno and it uses unfair bandwidth with

Reno. This is because NewReno uses improved Fast Retransmit mechanism so that it sends a new packet for every single partial ACK received, when compared to Reno using Fast Recovery phase as it resends packets only on receiving three duplicate ACKs. Similarly, for the same above reasons NewReno performs better with low latency than Reno.

TCP NewReno_Reno:

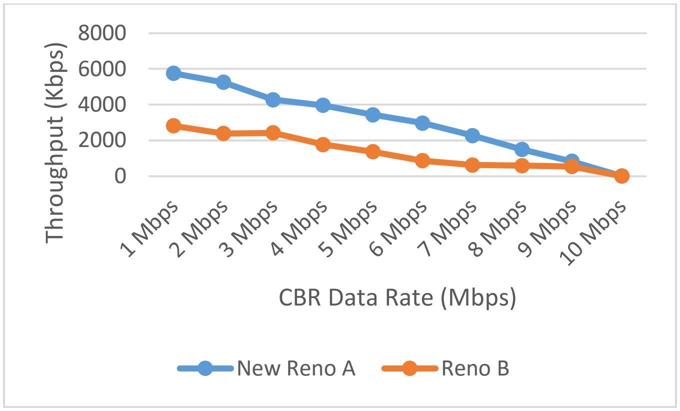

***Figure 2.a, Average throughput of TCP New Reno vs Reno with varying CBR data rate, Queue size 50, starting both TCP and CBR at same time*.**

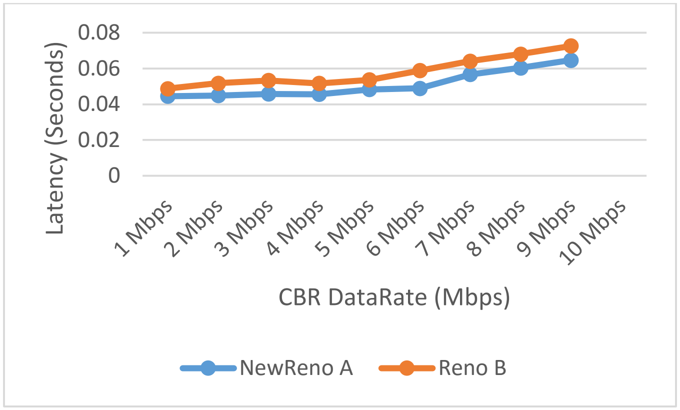

Figure 2.b, Average Latency of TCP New Reno vs Reno with varying CBR data rate, Queue size 50, starting both TCP and CBR at different time

For the Figures 2.c and 2.d, in case of throughput, TCP NewReno is performing better and it unfairly shares bandwidth with Vegas. This is because due to difference in mechanisms used, while TCP NewReno uses Fast Retransmit mechanism to quickly retransmit packet whenever there is single partial ACK received and in case of TCP Vegas uses delay based congestion avoidance mechanism where it detects congestion with help of delay and RTT measurement, on considering latency, Vegas performance is better since it uses better congestion avoidance approach than NewReno.

TCP NewReno_Vegas:

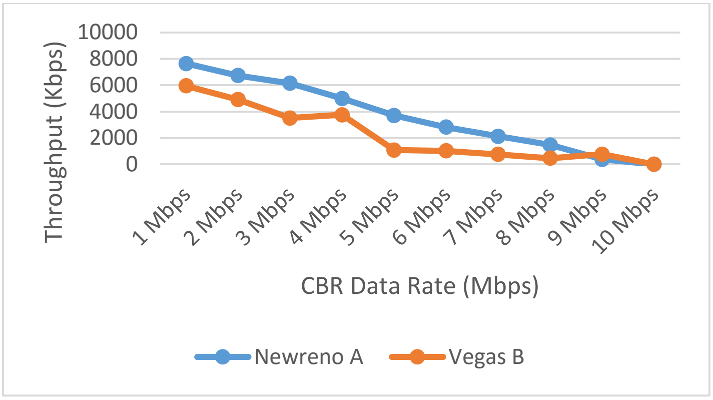

Figure 2.c, Average throughput of TCP New Reno vs Vegas with varying CBR data rate, Queue size 50, starting both TCP and CBR at same time with different RTT for TCP A

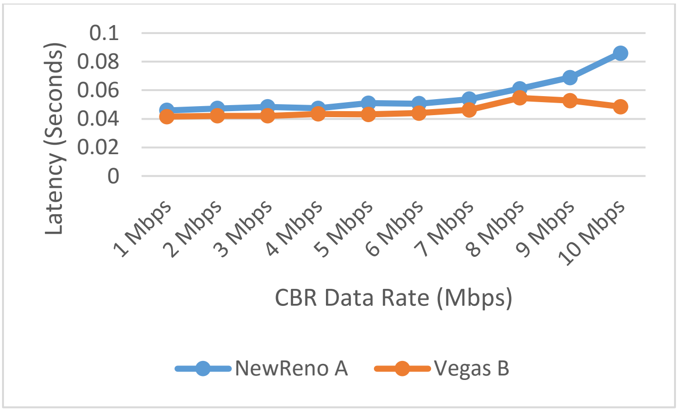

Figure 2.d, Average Latency of TCP New Reno vs Vegas with varying CBR data rate, Queue size 50, starting both TCP and CBR at same time

TCP Reno_Reno:

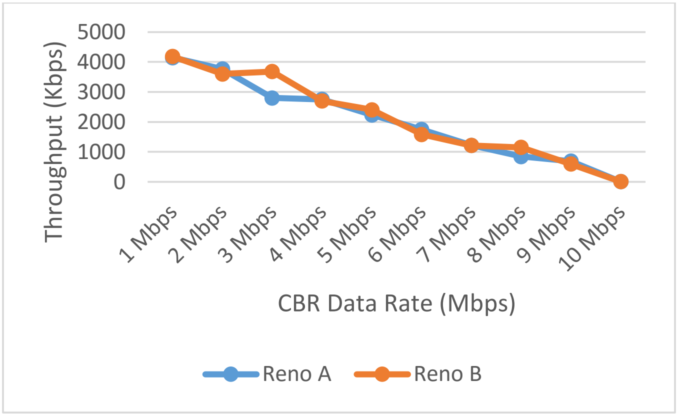

Figure 2.e, Average throughput of TCP Reno vs Reno with varying CBR data rate, Queue size 50, starting both TCP and CBR at same time

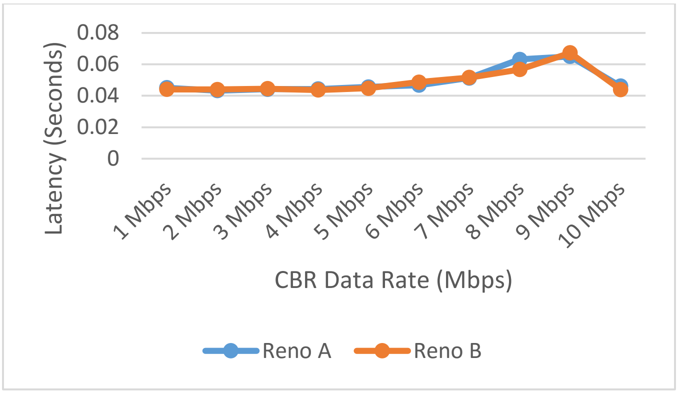

Figure 2.f, Average Latency of TCP Reno vs Reno with varying CBR data rate, Queue size 50 , starting both TCP and CBR at same time

From figures 2.e and 2.f, we analyzed that both TCP Reno is performing fairly by sharing common bandwidth, both Reno possess more or less similar values as it both uses same congestion avoidance approach as a result both reaches maximum rate and it also it gets reduced when there is timeout, similarly performance of Latency and Drop rate is due to same reasons.

TCP Vegas_Vegas:

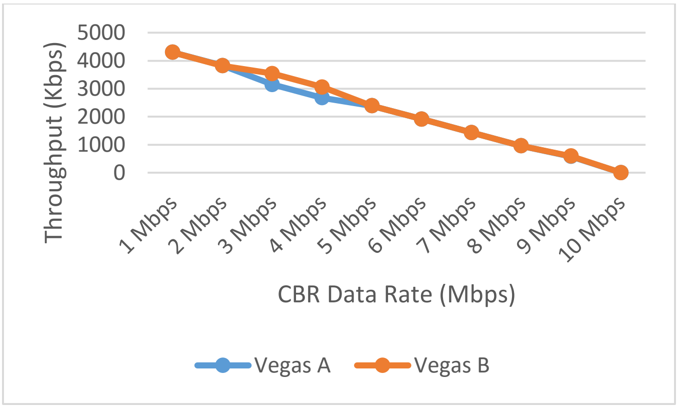

Figure 2.g, Average throughput of TCP Vegas vs Vegas with varying CBR data rate, Queue size 50, starting both TCP and CBR at same time

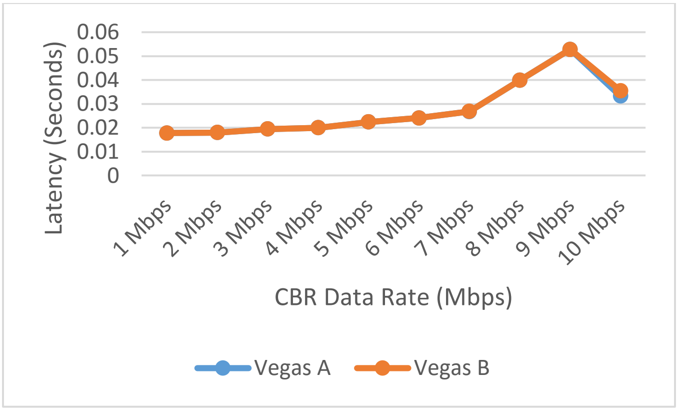

Figure 2.h, Average Latency of TCP Vegas vs Vegas with varying CBR data rate, Queue size 50 , starting both TCP and CBR at same time

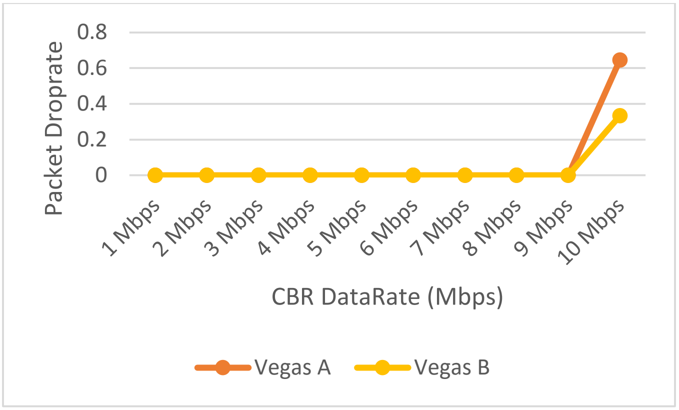

Figure 2.i, Packet Drop rate of TCP Vegas vs Vegas against varying CBR Data rate with different RTT of queue size 50

From the figures 2.g, 2.h and 2.i, with combination of same TCP variant Vegas is fairly sharing bandwidth and its throughput, latency and droprate is almost same. This is because TCP Vegas uses delay based congestion avoidance approach of detecting congestion in earlier.

From the results of experiment 2, We conclude that TCP achieve fairness in terms of bandwidth, latency and droprate only when there is same TCP variant used and not in combination of different TCP variants. This is because of difference in congestion control algorithms used by them. We also conclude that NewReno throughput performance is far better when it has been used with Reno and Vegas in sending data.

III. Experiment 3: Influence of Queuing

For the experiment 3, we used the same network topology mentioned in figure 1, here in this experiment we studied the performance of TCP on the effect of different queuing algorithms DropTail and RED for TCP RENO and SACK. Here we configured TCP source and sink at N1 and N4, CBR source and sink at node N5 and N6 respectively. For the testing scenario we start TCP source first and allow it to be stabilized and also fixed CBR rate to 6Mbps

From the figures 3.a and 3.b, we analyzed that throughput of Reno and SACK is higher without CBR flow, while when CBR flow gets started throughput of both SACK and Reno gets reduced drastically for both queueing algorithms (DropTail and RED). Also we observed that TCP SACK faces big drop in bandwidth for both queuing algorithms this is because of its congestion control mechanism and also it is observed that RED receives high throughput when compared to DropTail because as it receives queue size earlier and thereby avoiding TCP global synchronization.

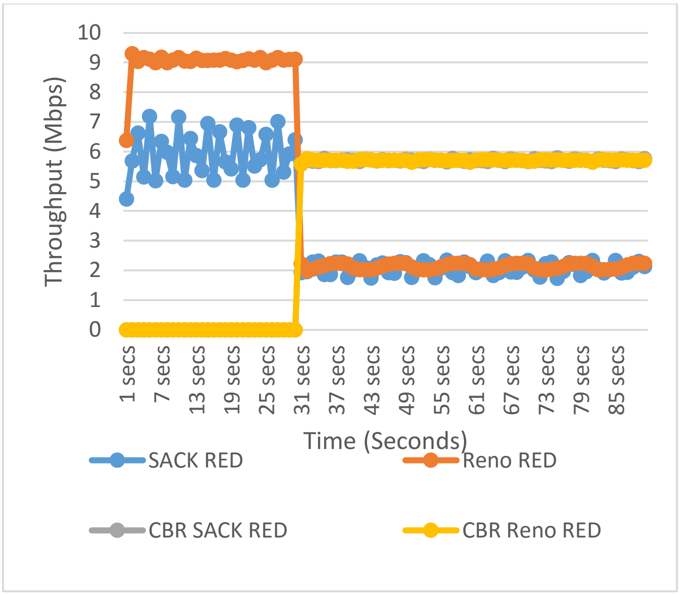

Figure 3.a Throughput (Mbps) vs Time (Seconds) for RED algorithm – TCP Reno and TCP SACK

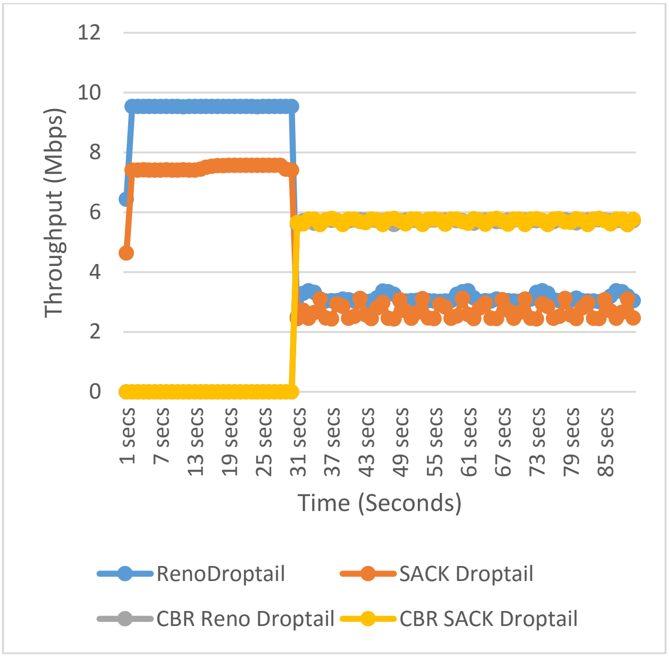

Figure 3.b Throughput (Mbps) vs Time (Seconds) for DropTail algorithm – TCP Reno and TCP SACK

Latency: At end of this experiment, we found out the latency to be as follows: 1). RENO DropTail: 0.0140261 Sec, 2). SACK DropTail: 0.0141688 Sec, 3). RENO RED: 0.0102386 Sec, 4.) SACK RED: 0.0109362 Sec. From the obtained values it is found that with RED queuing algorithm TCP SACK and RENO latency performs much better when compared to DropTail algorithm, this is because DropTail drops the packet only when the queue is full but RED receives average queue size earlier and also by avoiding TCP global synchronization.

IV. Conclusion:

With Experiments 1 to 3, We analyzed Performance of TCP variants (TCP Tahoe, Reno, NewReno and Vegas) under congestion, fairness and also under the influence of queuing.

From the experiment 1, We varied the data flow of CBR rate from 1 Mbps to 10 Mbps under different queue size, TCP window size and CBR packet size. At end of experiment 1, it is found that TCP Vegas is performing better than other variants in terms of throughput, latency and droprate. Next to TCP Vegas, we also found that TCP New Reno is performing well because of its Fast Retransmit mechanism. In the experiment 2, we studied that fairness is good when there is same combination of TCP variants (Reno/Reno) , (Vegas/Vegas), but fairness is unfair when there is different combination of TCP variants, this is because of difference in congestion control mechanisms that they have employed, finally in the experiment 3, we found out that under influence of queueing algorithm Random Early Detection (RED), TCP variant SACK and Reno is performing better both in terms of Throughput and latency than DropTail algorithm, this is because RED receives average queue size in earlier and also it avoids TCP global synchronization when compared to DropTail where it drops out packet when buffer size is full. Overall based on application needs one should choose best TCP variant to get better end user experience.

For Future works, we planned to cover experiments to use queueing algorithm Random Early Detection (RED) for all process only using TCP related process and DropTail algorithm for all only UDP related traffic and we also planned to extend our experiment by implementing other TCP variants like CUBIC, BIC, Compound and then to analyze its Performance under Congestion, fairness and queueing algorithm influence of it.

V. References:

I. Simulation-based Comparisons of Tahoe, Reno, and SACK TCP by Kevin Fall and Sally Floyd

II. http://nile.wpi.edu/NS/

III.http://zfadlullah.org/files/course-material/information-engineering/Additional/ns_kiso.pdf