Comparative Analysis of TCP Variants

Samkeet Shah
shah.sam@husky.neu.edu
Northeastern University

Fan Yang
yang.fan7@husky.neu.edu
Northeastern University

ABSTRACT
In this paper, we aim to do a comparative analysis of TCP variants Tahoe, Reno, NewReno, Vegas and SACK and study their behavior under various simulated network environments. We run simulations to observe the effects of congestion, queueing and fairness on these TCP variants.

INTRODUCTION
Transmission Control Protocol (TCP) is a connection-oriented protocol that provides a safe and reliable medium of communication between various process and machines across the internet. Over the years, various improvements have been made to the TCP protocol. Each variant has some improvements over the previous versions, such as better congestion control, packet retransmission etc. TCP manages congestion, reliable delivery of data and splitting of data into packets and reassembling them at the destination. TCP runs at the Transport layer of the Open System Interconnection (OSI) model.

METHODOLOGY AND SET-UP
We use Network Simulator (NS-2) tool, which an openly available to under the GNU GPLv2 license for research and use[2]. The setup involves creating a topology as shown in Fig. 1.

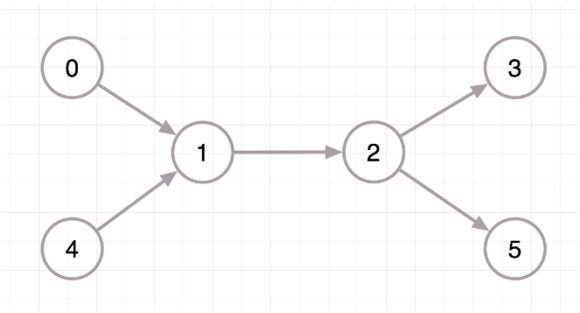

Figure 1. Network Topology

(The value of the nodes are incremented by 1 in the labelling used in the actual experiments)
The topology and simulation environment is set up as follows.

1. Node 0 is used as a TCP source which runs an FTP application to Node 3.
2. Node 1 runs over a UDP protocol, a CBR that varies from 0 to MAX (10Mbps) in Experiment 1 and 2.
3. In experiment 2 another FTP application runs over TCP from Node 4 to Node 5.
4. Similarly in Experiment 3, a CBR flow starts from Node 4 to Node 5 in experiment over UDP.
5. Each link is a duplex link with a 10Mb bandwidth and a 5ms delay.
6. The CBR packet size used in all experiments is 1000 byte and the queue limit for link N1-N2 is 10.

We performed the following experiments:

1. For experiment 1, we performed 2 scenarios, in the first scenario the CBR flow was in the direction of TCP flow. In the second scenario, the CBR flow was opposite to the TCP flow. It is run for 15 seconds. We observer the effects of external traffic (UDP in this case) on TCP throughput.
2. In the second experiment, we had two scenarios, one where we run the network topology with the given scenarios. In the second scenario, we simultaneously start all four TCP variants from the same node. This experiment was run for 15 seconds. In this experiment we run 2 FTP applications on TCP from different sources to observe the fairness between them.
3. For the last experiment, to detect the effects of Queuing algorithms on TCP flow, we start TCP before CBR and keep the CBR rate constant for experiment three. The last experiment is run for 30 seconds where we observe the effects of queuing algorithms DropTail and RED on different TCP variants.

We conducted each experiment scenario for many times, to get the average data. Then we used Excel to plot the graphs and stats for visualization and verification. Furthermore, we applied T-tests to arrive at and prove our conclusions. This will be a statistical approach to verifying the results.

ANALYSIS

TCP Performance Under Congestion.
In this experiment we study the effect of external traffic on TCP throughput. The network topology is shown in Fig. 1. We have a TCP connection set up from node 0 to node 3. A constant bit rate (CBR) source is added from node 1 to node 2. CBR runs on top of UDP protocol, which is a connectionless protocol. We perform the experiment as follows, we start the TCP traffic before the CBR flow at

N1. An FTP application is running at node 0 which generates the TCP traffic. We gradually vary the CBR rate from 0 to maximum value (10 megabits per second).

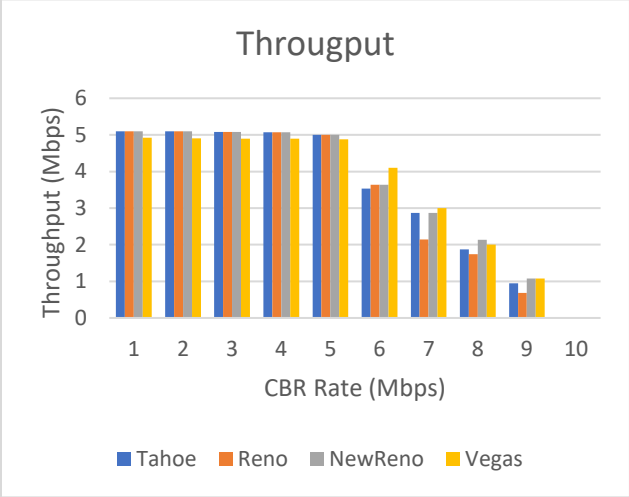

Figure 2. TCP Throughput Comparison

We can observe from Fig. 2 that TCP Vegas has the most consistent higher throughput among all other TCP variants. Initially TCP Vegas uses slow start hence it takes time to reach maximum flow rate. Vegas uses slow start to avoid congestion hence it has a low throughput initially. After CBR rate starts, due to congestion other variants suffer packet drop where as Vegas achieves better throughput by using the available bandwidth for maximum throughput[1].

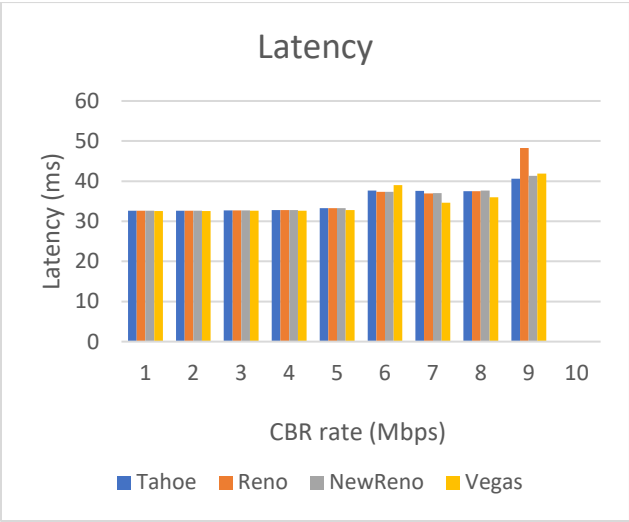

Figure 3. Latency Comparison of TCP Variants

NewReno has a high throughput as well. Towards higher CBR rates, NewReno almost equals Vegas' throughput rate. It can be observed from Fig. 3 that TCP Vegas has the lowest Latency overall. The latency increases when the CBR flow starts, which can be noticed as a slight rise in the graph at Bandwidth value 5 – 6Mbps.

But TCP Vegas recovers from the congestion to maintain a low latency as compared to other variants. TCP Reno has the highest latency since it waits for three ACKs before retransmitting a packet.

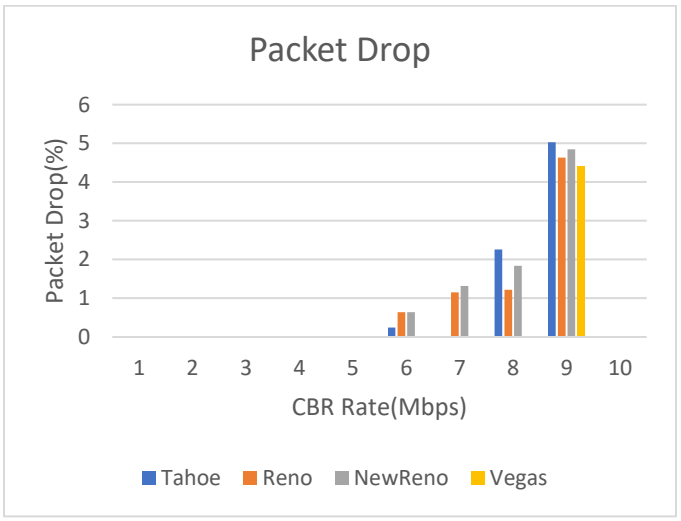

Figure 4. Packet Drop Comparison of TCP Variants

It is evident from Figure 4 that TCP Vegas has the lowest Drop rate among all other variants. When the CBR flow starts, the packet drop increases for all variants but it rises sharply for all NewReno, Reno and Tahoe and less steeply for Vegas. Vegas monitors the RTT time for its packets and hence adjusts the packet flow to avoid congestions and thereby has a lower drop rate. Thus we can conclude from the observations that when considered as a whole throughput, latency and drop-rate, Vegas outperforms all other variants.

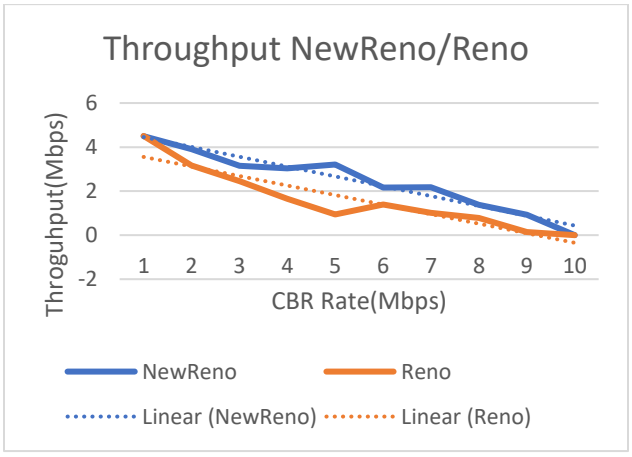

Figure 5. Throughput Comparison NewReno/Reno

Fairness Between TCP Variants

In this experiment we analyze the effect of multiple TCP flows originating from different sources on each other and determine the fairness between them. The network simulation consists of a topology with 6 nodes (N0 – N5). TCP flows have been set up between N0-N3 and N4-N5. A constant bit rate flow (CBR) is set up originating at N2 to N3. The TCP flows are started before the CBR flow and the CBR flow is gradually varied from 0 to MAX (10Mbs) to observe the effects on thoughput, latency and drop-rate of each of the TCP flows. We analyze the following combinations NewReno/Reno, NewReno/Vegas, Reno/Reno, Vegas/Vegas.

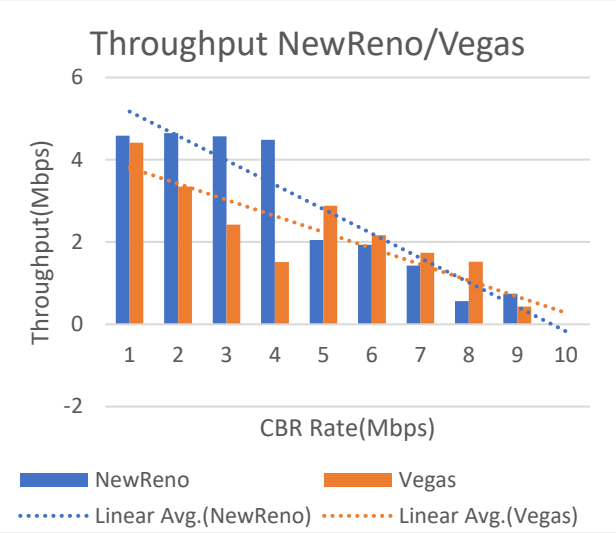

Figure 6. Throughput Comparison NewReno/Vegas

In case of NewReno/Reno, we observe that NewReno drastically affects the Reno throughput. From Figure 4 the Linear averages plotted, clearly indicate that NewReno is not fair to Reno.

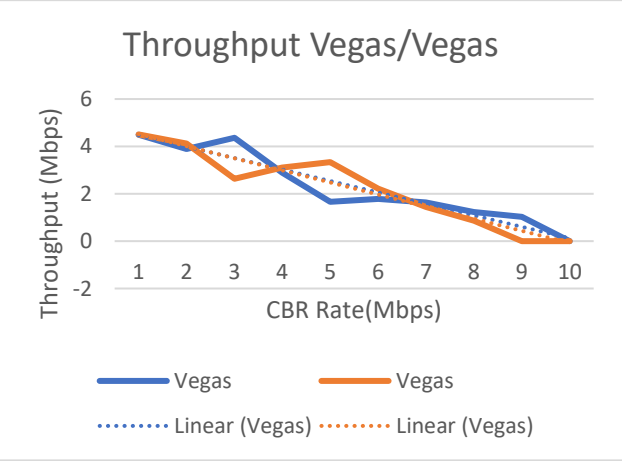

Figure 8. Throughput Comparison Vegas/Vegas

NewReno uses Fast Retransmit to recovery from congestion, hence we observe the unfair bandwidth use by for the NewReno/Reno pair. Reno waits for three ACKs before retransmitting a packet, hence it's performance is reduced drastically.

For NewReno/Vegas, both variants have some advantages over other variants. NewReno supports fast retransmit and fast recovery, where as Vegas user slow start to avoid congestion. From Fig. 6, NewReno uses a higher part of the bandwidth. Initially, due to slow start Vegas throughput is low. But since Vegas uses slow for congestion avoidance, by determining the available bandwidth. Hence the throughput declines steadily as compared to the sharper decline for NewReno with increasing CBR rate.

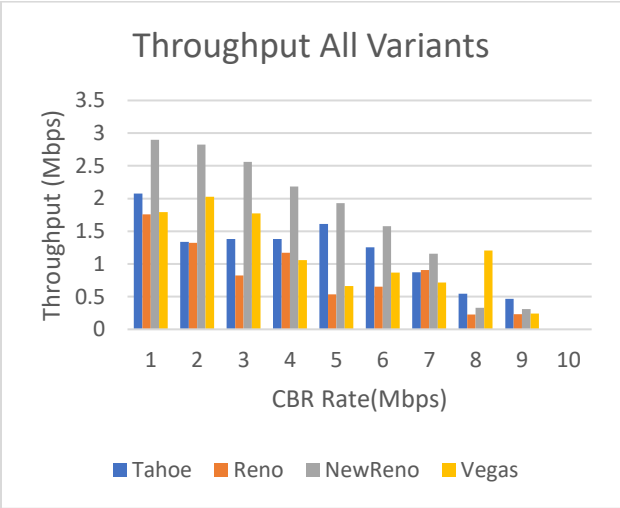

Figure 9. Fairness Comparison - Variants from Same Source

Reno/Reno and Vegas/Vegas we observe from Fig. 7 and Fig.8 respectively that, both combinations are fair. We can observe from Fig. 9 that the Linear trendline for Reno/Reno gradually converge to coincide. The throughput of both Reno sources remains almost same. As seen in Fig. 8, the Linear Avg. trendlines for both Vegas sources are overlapping, which indicates the Vegas/Vegas combination is a fair combination with an average equal share of bandwidth for both sources.

We conducted another test where we started all 4 TCP Variants (Tahoe, Reno, NewReno/Vegas) from the same source (Node N0) and observed the fairness. The results as shown in Fig. 9 indicate that initially NewReno dominates the bandwidth in the channel. But as CBR rate increases, all throughputs suffer. But Vegas throughput increases overall as compared to other variants, by using congestion avoidance technique.

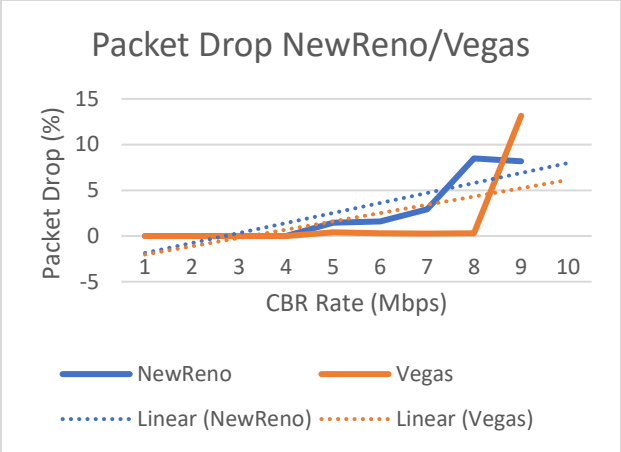

Figure 12. Packet Drop (Experiment 2) NewReno/Vegas

Hence we can conclude that NewReno combined with Reno and Vegas causes an unfair use of bandwidth for other variants. Vegas' property of using slow start results in detecting the available bandwidth for transmission and hence the consequent TCP throughput is low for Vegas when combined with NewReno. Similarly Reno suffers with low throughput rate when combined with NewReno, since NewReno used Fast Retransmit. But Reno/Reno and Vegas/Vegas combinations have a fair bandwidth utilization for both sources.

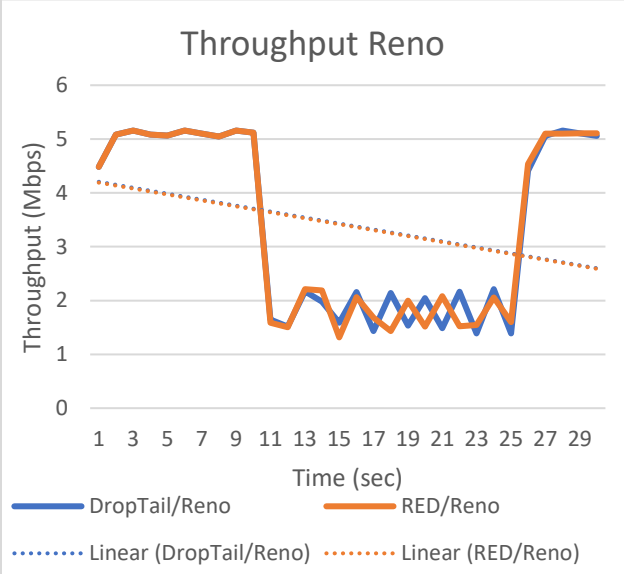

Figure 10. Throughput plot for Reno under RED/DropTail

Experiments 3. Influence of Queueing

In this experiments we study the effect of Queueing algorithms DropTail and RED on the overall performance of TCP variants. Queueing algorithms determine how packets are stored and forwarded during congestion. Hence we study the how they affect TCP variants Reno and SACK. We use the network topology as mentioned previously. But now we have TCP flow from N1 to N4 and CBR flow from N4-N5.

As it can be observed from Fig. 10, the throughput due to RED and DropTail on TCP Reno, is almost the same, but at some points RED performs better towards the end of simulation. The latency plot as shown in Fig. 11, clearly indicates that TCP Reno under RED has a significantly lower latency as compared to DropTail. This is primarily due to, RED's property of using probability to determine if packet should be dropped. Hence RED is better for Reno as compared to DropTail queueing algorithm.

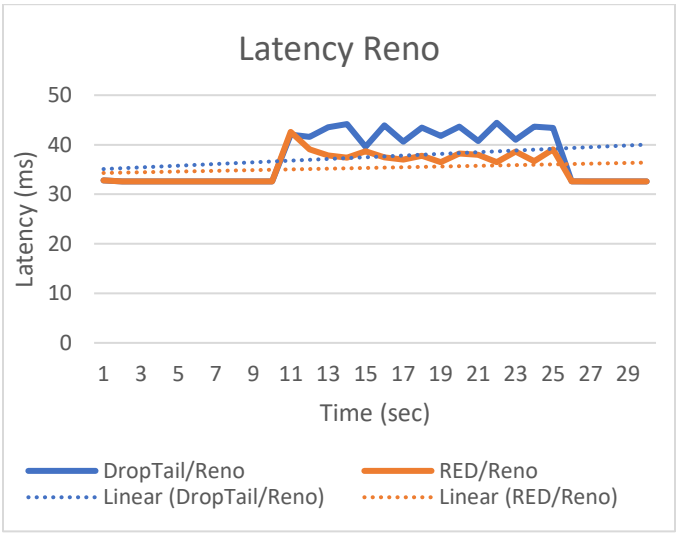

Figure 11. Latency plot for Reno under RED/DropTail

For TCP SACK, the queueing algorithm, RED performs slightly poor till the CBR flow starts. After that the performance of SACK under RED improved as compared to DropTail. This can be observed from Fig. 12.

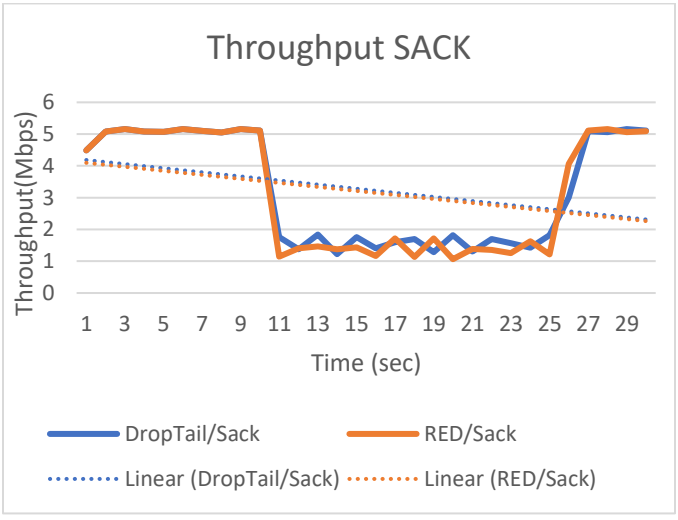

Figure 12. Throughput for SACK under RED/DropTail

The latency plot for TCP SACK shown in Fig. 13 clearly indicates that SACK performs better under RED.

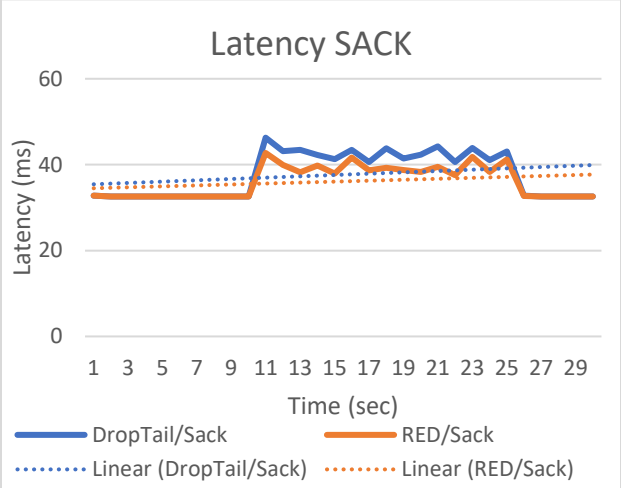

Figure 12. Latency for SACK under RED/DropTail

CONCLUSION

We can infer from our experiments that TCP Vegas performs better overall, as compared to other variants as seen in experiment 1. However when paired with another TCP flow such as NewReno we observe that NewReno causes unfair use of transmission channel hence Vegas performance suffers. But as we can see from Fig. 12 the packet drop rate for NewReno always remains higher as compared to Vegas, since Vegas uses Congestion avoid techniques to determine available bandwidth for transmission in the channel.

For the second experiment, NewReno paired with any other variants caused a drop in performance of other variants. Performance of TCP Reno is very low in terms of throughput, latency primarily due to advantages of other variants and since Reno uses 3 ACKs before packet retransmission, which significantly increases the latency.

From the third experiment, RED algorithm proves to be better for both SACK and Reno as compared to DropTail, since RED drops before fully queued to slow down sender for a better congestion control.

REFERENCES

[1] “A Comparative Analysis of TCP Tahoe, Reno, New-Reno, SACK and Vegas”, EECS Instructional and Electronics, University of California, Berkley.

[2] https://en.wikipedia.org/wiki/Ns_(simulator)