Performance Analysis of TCP Variants

Haojun Zhu
Northeastern University
Boston, MA 02115
Email: zhu.haoj@northeastern.edu
NUID: 002979116

Marwa Elali
Northeastern University
Boston, MA 02115
Email: elali.m@northeastern.edu
NUID: 001527297

Pratik Budhiraja
Northeastern University
Boston, MA 02115
Email: budhiraja.p@northeastern.edu
NUID: 002105546

***Abstract*—The TCP protocol was designed to provide end-to-end reliability in data transfer. This protocol adds a layer of congestion control, a necessary part of the Internet. The paper presents a study on the different variants of TCP that are used widely across multiple operating systems (Tahoe, Reno, NewReno, and Vegas) and analyze their performance using different parameters (throughput, drop rate, and latency) under varying degrees of congestion, using NS-2 simulation. The analysis will help us understand the behavior of these variants in terms of congestion avoidance and fairness among them. Furthermore, we analyze the influence of different queueing disciplines between TCP variants.**

I. Introduction

The TCP protocol provides reliable data transfer and helps with the handshake connection between the client and server. The protocol also manages congestion control which is one of the data transfer issues in today's world. Each TCP variant behaves differently to handle and avoid congestion and is used in different applications. This paper aims to analyze the TCP variants in different scenarios and discuss their performance by simulating them as realistically as possible. We will use the NS-2 simulator to simulate the TCP variants under different load conditions and queueing disciplines to analyze their performance. We will conduct three experiments to compare TCP variants with each other, analyze how their performance is affected under congestion, and analyze their fairness amongst other performance parameters.

The TCP Variants referred in this paper are described as follows:

1) **TCP Tahoe**
   This is the first and simplest variant that was introduced with congestion control techniques, namely Slow Start, Fast retransmit and AIMD (Additive Increase Multiplicative Decrease). This variant uses Fast retransmit to recover from lost packets.

2) **TCP Reno**
   This variant is similar in behavior to TCP Tahoe, with addition of the Fast recovery algorithm. This algorithm uses retransmission timeout (RTO) or Fast retransmit to determine how to decrease window either by 50% (fast retransmit) or to initial (RTO) window size.

3) **TCP NewReno**
   The NewReno variant is an extension of Reno, in the sense that it resolves the problem of detecting multiple packet drops within the same congestion window. This makes it possible to detect multiple drops much faster than Reno.

4) **TCP Vegas**
   The TCP Vegas manages congestion by focusing on congestion avoidance over congestion control. It uses the round trip time, to decrease the window size before any packet drops. Using throughput calculations as well as expected throughput, their difference is compared with min and max thresholds to determine whether to increase/decrease window or do nothing.

II. Methodology

We are going to test the effects of various components on each of the TCP flows. In order to create the simulation conditions we are going to use the NS-2 simulator. It is a highly modular platform that provides support for testing various network protocols, elements routing and traffic. To parse the data we are going to use python, and excel for plotting the resulting graphs.
The first two experiments are going to test four different TCP variants - Reno, NewReno, Tahoe, and Vegas. The network topology is going to be set up as shown:

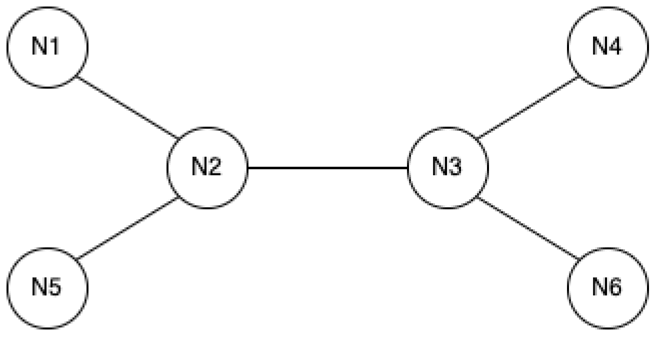

Figure 1: Network Topology

We will perform our experiment on the nodes N1, N2, N3, N4, N5 and N6. Each node is connected over a full-duplex link with bandwidth set to 10Mbps and delay of

12ms. We are going to analyze through each experiment which conditions affect each TCP stream's usage and the fairness of bandwidth usage among the TCP flows. The default queueing algorithm used in the experiments is the DropTail algorithm. It is a simple queueing algorithm used in networks to determine when to drop packets - once the queue is filled to capacity, newer packets will be dropped until the queue has enough capacity.

Experiment 1
The first experiment will analyze the effect of congestion on TCP performance. Each TCP variant will be analyzed on its own with the CBR. The TCP stream begins at N1 and has a sink at N4, with a CBR stream at N2 to N3. The main variable in this experiment is the CBR rate. We will be varying the CBR starting at 1Mbps, increasing up to the bottleneck at 10Mbps. From this experiment we will analyze which variant results in the highest average throughput over time, lowest average latency, as well as lowest packet drops overall.

Experiment 2
The second experiment will be an analysis on fairness across two TCP flows. We will use a similar network topology as in experiment 1, with the addition of another TCP stream from N5 to N6, to measure how two TCP flows running together affects bandwidth usage for each variant. From this experiment we will analyze how fair each combination of TCP variants are with each other, based on the bandwidth usage of the TCP flows. By having variable start times we can further analyze the fairness among these variants.

Experiment 3
This experiment will test the influence of queueing disciplines on TCP performance. The network topology will be the same as in experiment 1. We will only be simulating two TCP variants in this experiment - Reno and SACK. The TCP will run until it reaches a steady flow, after which we will start the CBR flow. We will analyze how the different queueing algorithms affect the flows over time. Through this we can determine whether each algorithm provides fair bandwidth to each stream, as well as analysing how each algorithm affects end-to-end latency.

III. Experiment 1: TCP Performance Under Congestion

In this experiment, we test the performance of each TCP variant under various load conditions. We vary the CBR from 1Mbps to 10Mbps, to assess the throughput, drop rate, end-to-end latency under each rate. We can expect the performance of these variants to decline as each variant reaches the bottleneck capacity. However, we can analyze which variants will significantly outperform others in each measure of average throughput, drop rate, and average latency.

1) **Throughput**

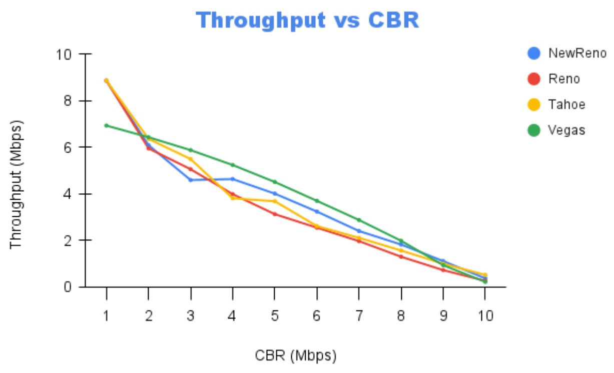

Figure 2: Throughput vs. CBR Bandwidth

All variants behave similarly as the CBR increases to the bottleneck capacity. Vegas starts with the lowest throughput while the throughput of other variants remains closely matched, but its throughput decreases in a smoother, almost linear manner as the CBR increases. Although Reno should benefit from fast recovery, it performed worse than Tahoe.

Overall, Vegas had the best average throughput for majority of the CBR rates. In the CBR range of 2-8Mbps it has the best performance and is more consistent compared to the other variants.

2) **Packet Drop rate**
We can observe that Vegas only starts to drop packets between 8-9Mbps range, whereas Tahoe is the first to begin dropping packets. Reno and NewReno have similar drop rates as Tahoe up to 9Mbps, due to the AIMD method of congestion control. NewReno reduces congestion window less often, Reno uses its fast retransmit property to determine whether to decrease window by 50% or to initial value

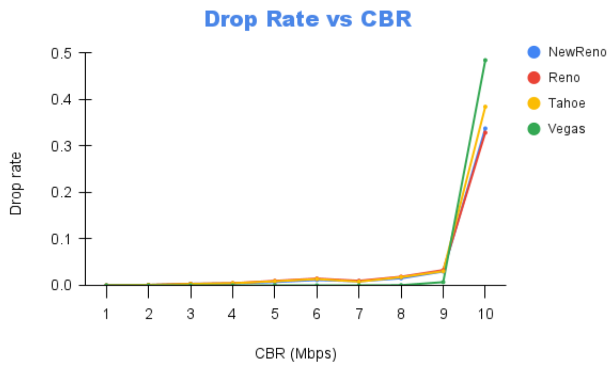

Figure 3: Packet Drop Rate vs. CBR Bandwidth

Vegas, however, with its congestion avoidance approach, is less susceptible to packet drops at lower CBR rates than the other variants. This makes Vegas have significantly lower drop rates than other variants.

3) **Latency**
Latency of all variants up to CBR of 9Mbps were below 0.1. When the simulation started, Vegas had the lowest latency, and continued to have the lowest overall latency throughout the entire simulation. Other variants performed similarly, with slight fluctuations in latency at each CBR interval (5-6Mbps, and 7-9Mbps).

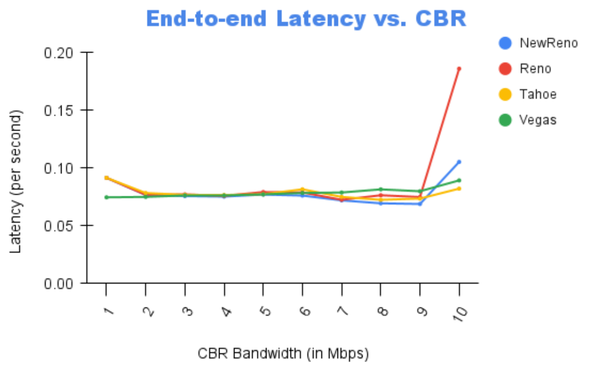

Figure 4: Latency vs. CBR Bandwidth

Reno latency increased significantly at bottleneck capacity, when compared with NewReno and Tahoe, which both had lower latency.

Due to the way Vegas handles congestion, by evaluating expected vs actual throughput, and the RTT, to increase/decrease *cwnd*, Vegas outperformed all other variants by its average latency, throughout the entire simulation.

IV. Experiment 2: Fairness Between TCP Variants

In this experiment, we test the performance of pairs of TCP variants under various load conditions. We vary the CBR from 1Mbps to 10Mbps, and start the first TCP variant before the second variant to get a more realistic scenario. We assess the throughput, drop rate, end-to-end latency under each CBR bandwidth and plot the graphs for throughput, drop rate and latency for each TCP pair to determine their fairness.

1) **NewReno/Reno**

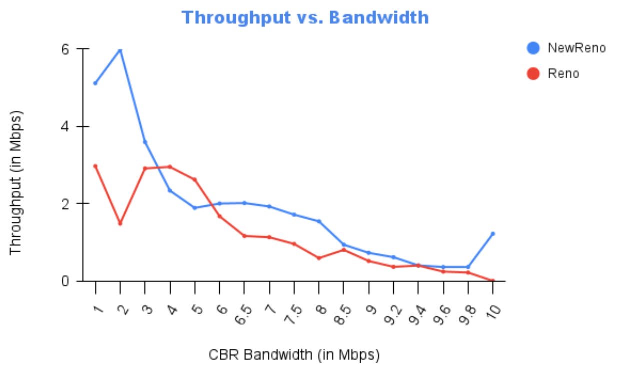

Figure 5: NR/R - Throughput vs. CBR Bandwidth

It is clearly observable that NewReno has higher overall throughput than Reno. Reno's throughput drops to 0 when the CBR is at the bottleneck capacity and NewReno is already sending packets.

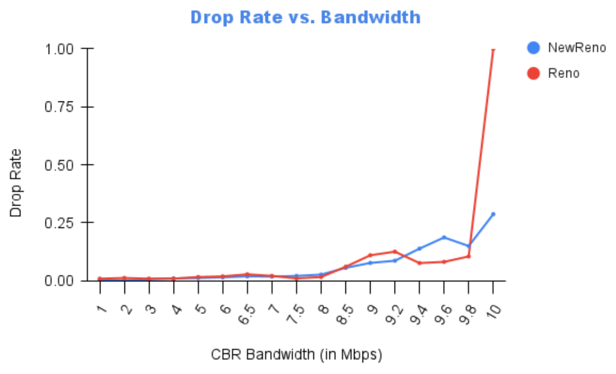

Figure 6: NR/R - Packet Drop Rate vs. CBR Bandwidth

Initial packet drop rate is low since the queues are not completely full. Reno starts to drop more packets since it handles a single packet loss scenario, exits and enters Fast recovery per packet loss, however NewReno does not exit the Fast recovery stage unless all previous packets have been acknowledged.

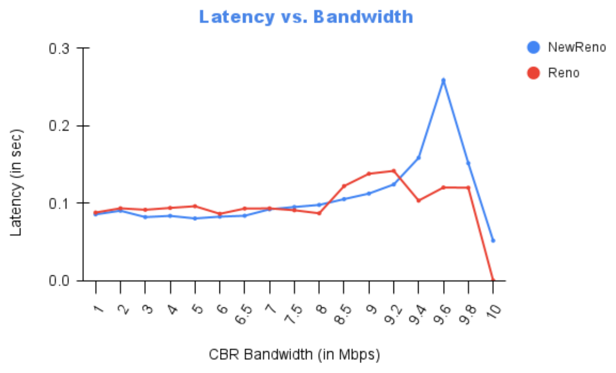

Figure 7: NR/R - Latency vs. CBR Bandwidth

Reno has a higher overall latency than NewReno, especially when CBR approaches the bottleneck capacity. Hence, we can conclude that NewReno is not completely fair to Reno. However, it does not completely affect Reno's performance.

2) **NewReno/Vegas**

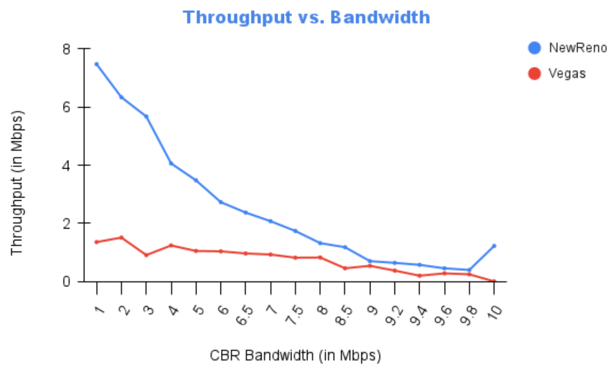

Figure 8: NR/V - Throughput vs. CBR Bandwidth

We can observe that NewReno has a higher overall throughput compared to Vegas which continues as the CBR bandwidth increases.

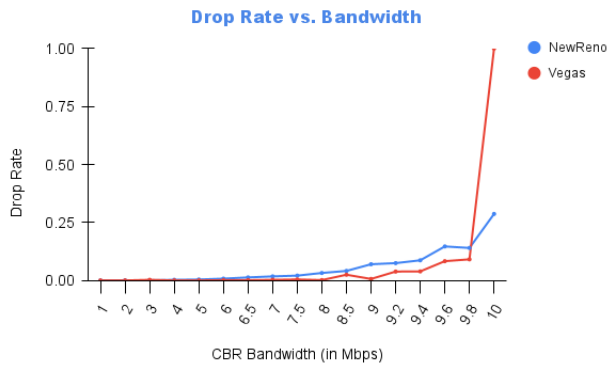

Figure 9: NR/V - Packet Drop Rate vs. CBR Bandwidth

NewReno has a lower overall packet drop rate compared to Vegas, since it runs more steadily thereby losing less number of packets than Vegas. When the CBR is at the bottleneck capacity Vegas cannot compete with NewReno and hence, drops all the packets.

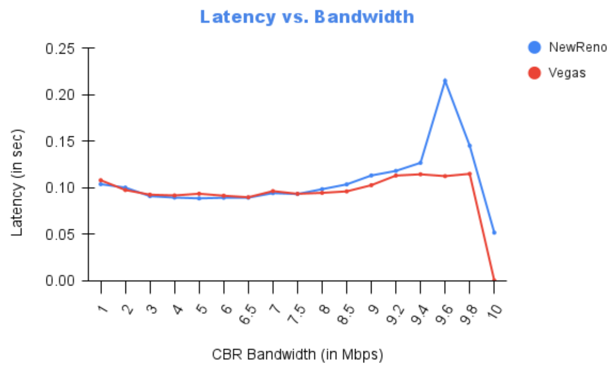

Figure 10: NR/V - Latency vs. CBR Bandwidth

NewReno and Vegas remain close in terms of latency. Around the 9.6Mbps mark, latency for NewReno peaks but stabilizes while competing with a high CBR bandwidth. When the CBR is at the bottleneck capacity, Vegas cannot compete with NewReno and drops all packets.
Clearly, NewReno is not completely fair to Vegas. Vegas can detect congestion at an early stage compared to other variants and reduces its packet sending rate, thereby providing more bandwidth to NewReno.

3) **Reno/Reno**

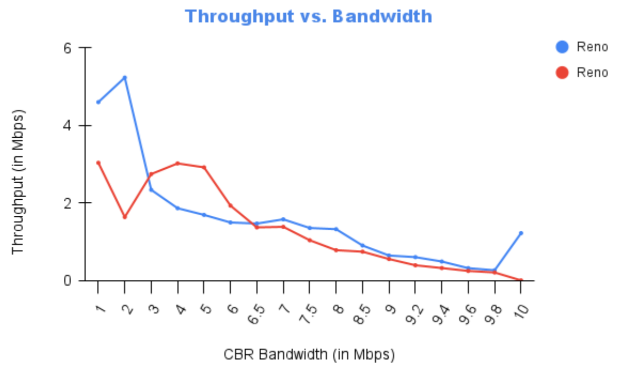

Figure 11: R/R - Throughput vs. CBR Bandwidth

Both Reno variants start at different timings, thus causing the fluctuations in the throughput rates. However, both flows remain closely matched.

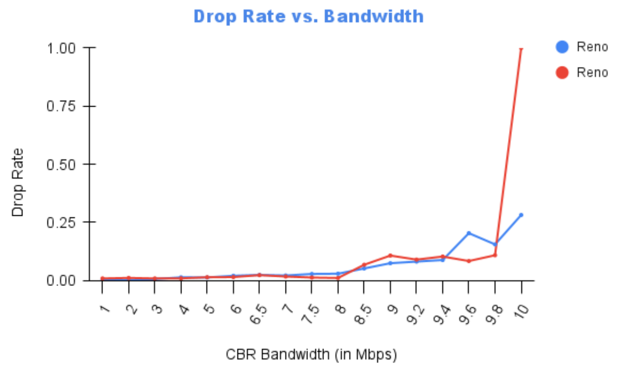

Figure 12: R/R - Packet Drop Rate vs. CBR Bandwidth

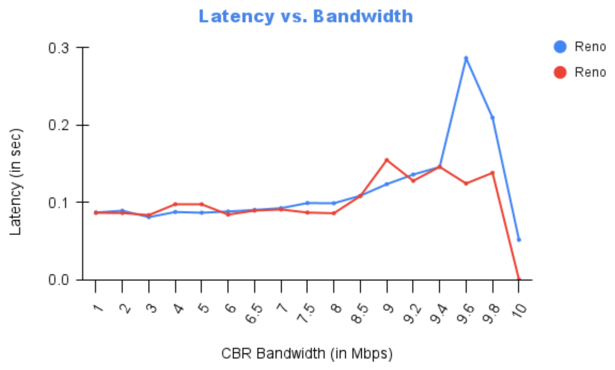

Figure 13: R/R - Latency vs. CBR Bandwidth

Similarly, in the case of drop rate and latency, there is some fluctuation between the two TCP flows. However, this can be accounted for, since when one of the variants has started and has reached a certain level of throughput, the other variant cannot compete due to bandwidth capacity constraints. Hence, we can clearly conclude that both the TCP Reno variants behave fairly for throughput and drop rates.

4) **Vegas/Vegas**

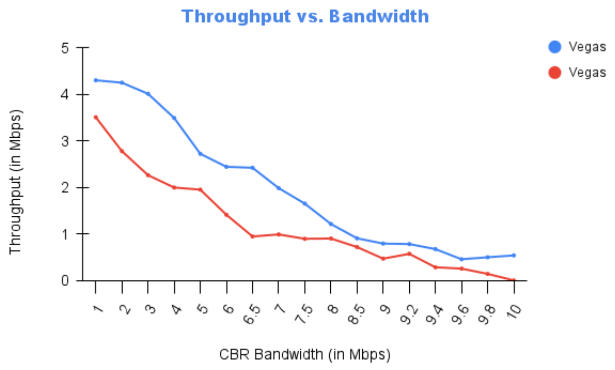

Figure 14: V/V - Throughput vs. CBR Bandwidth

We can observe that the performance parameters of Vegas vs. Vegas is similar to having simultaneous TCP Reno flows, as described above.

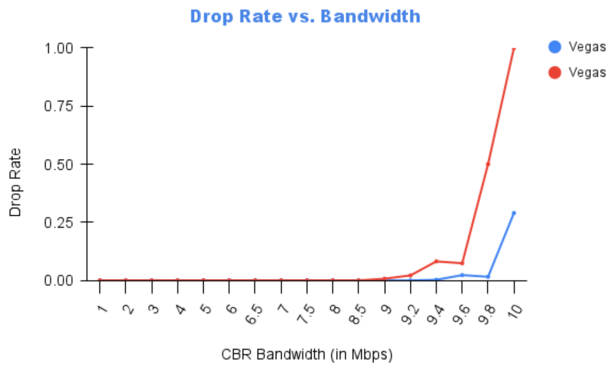

Figure 15: V/V - Packet Drop Rate vs. CBR Bandwidth

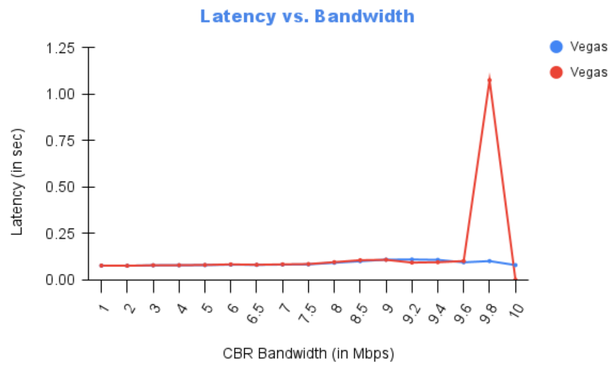

Figure 16: V/V - Latency vs. CBR Bandwidth

Vegas/Vegas shows a similar behaviour when comparing throughput, drop rate and latency, since when one of the variants has started and has reached a certain level of throughput, the other variant cannot compete due to bandwidth capacity constraints. Hence, we can conclude that both the TCP Vegas variants behave fairly for throughput and drop rates.

From the experiments conducted above, we can conclude that if we introduce a similar TCP variant as a second flow into the network, it is fair with the other TCP flow. However, the same cannot be said with different combinations of TCP variants. Each variant is linked with its own congestion avoidance and packet loss retransmission technique. For instance, TCP Vegas considers the network to be always congested and reduces its transmission window, thus it gets dominated by NewReno (as discussed above).

V. EXPERIMENT 3: INFLUENCE OF QUEUEING

In this experiment, we are testing the effect on TCP performance when using different queueing algorithms. We analyze the performance of two TCP variants (Reno, SACK) using the following queueing algorithms - *DropTail* which is the default queueing algorithm for experiments 1 and 2. Even though this algorithm is easy to implement, it has issues with synchronization, and *RED* (Random Early Detection). The simulation runs for 30s, since running for only 10s produced results that were not conclusive, and so increasing simulation time allows the throughput to reach a steady state before introducing the CBR flow at 5s. We will analyze the average throughput, and average latency over time.

1) **Throughput**
We waited until TCP reached a steady flow before introducing CBR in order to see how the introduction of CBR would affect TCP. When CBR is introduced at 5s, throughput for all variants decreases significantly, and then continues to fluctuate for the rest of the simulation.

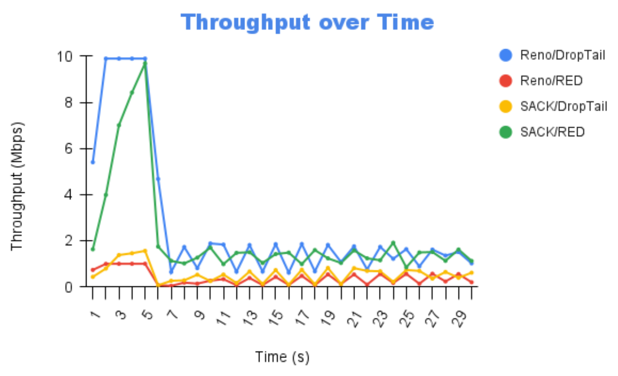

Figure 17: Throughput vs. Time

Reno using DropTail outperformed Reno using RED significantly throughout the simulation. However, SACK performed significantly better using RED rather than DropTail. Reno reaches a steady flow much quicker, taking only 1s to get a steady throughput, whereas SACK continually increases its throughput. However the results show that before introducing CBR, SACK can have a potentially higher throughput than Reno overall.

For Reno using DropTail, and SACK using RED, their throughput reaches up to 10Mbps before introducing CBR, after which it experiences a sharp drop to below 2Mbps, which is a much bigger drop. These results indicate that the DropTail algorithm is much more optimal to use for TCP Reno, as it outperforms Reno using RED by a significant difference. As well, SACK would perform better using the RED algorithm, when compared with SACK using DropTail.

2) **Latency**

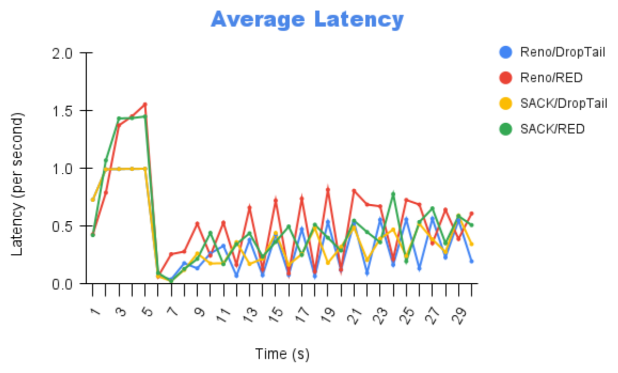

Figure 18: Latency vs. Time

With throughput, we found that each of the TCP variants was more suited to one of the queueing algorithms over another. However, analysing latency results indicate that the type of queueing algorithm produced similar results in both variants. When CBR is introduced, all variants under both queueing disciplines had their latency drop to the same level at the 6s mark and begin to fluctuate continuously for the rest of the simulation. The variants using the DropTail algorithm have lower latency overall, as well as having less aggressive fluctuations after CBR starts, while the variants using the RED algorithm have a much higher latency overall throughout the simulation, as well as less stable fluctuations.

VI. Conclusion

After carrying out all three experiments, our simulations have produced results that help identify how each TCP variant performs under our various conditions. We have analyzed the mappings of our experiment results to measure performance using throughput, latency and drop rate.

Through experiment 1, we found that TCP Vegas outperformed all other variants overall, although experienced the most packet drops at the bottleneck.

In experiment 2, we found that NewReno outperformed Reno in most aspects, but had a higher latency when the CBR rate starts reaching the bottleneck. NewReno had better throughput when competing with Vegas. Vegas drop rate shoots up at bottleneck capacity. NewReno is not completely fair to Vegas. Reno with Reno, and Vegas with Vegas, had the most stable performance, with both streams performing at similar levels and having similar patterns during the simulation.

Experiment 3 showed us that DropTail is the best queueing algorithm to give lower latency, however throughput performance was dependent on the TCP variant used. Reno performed much better using DropTail, and SACK was more suited to use RED.

These experiments give a good analysis however we can carry out further experiments, as well as varying many more conditions to gain even more conclusive findings for specific use cases. As networks are constantly experiencing changes in their conditions due to interference from various other flows among other factors, network performance may not always behave as predicted and can be susceptible to more fluctuations in performance.

References

[1] Simulation-based Comparisons of Tahoe, Reno, and SACK TCP by Kevin Fall and Sally Floyd

[2] TCP Tahoe, Reno, NewReno, Vegas and SACK comparison - $https://www.isi.edu/nsnam/ns/doc/node387.html$

[3] An Overview of Performance Comparison of Different TCP Variants in IP and MPLS Networks - $https://link.springer.com/chapter/10.1007/978-3-642-22185-9_11$