Performance Analysis of TCP Variants

Waleed Humayon Saeed
Northeastern University, saeed.w@northeastern.edu

Tanay Mehra
Northeastern University, mehra.t@northeastern.edu

***Abstract* - This paper focuses on two experiments that focus on analyzing performance of different TCP variants. We create a simple network that consists of 6 different nodes, create congestion with the use of CBR flows, and use the corresponding trace files to extrapolate the data and determine how different variants handle congestion. We focus predominately on throughput, latency and drop rates. In experiment 1 we conduct a comparative analysis between the different variants. In experiment 2, we evaluate fairness between different TCP variants.**

1. INTRODUCTION

Modern TCP has problems with congestion. Often times a network node or link will carry a higher load than it can handle, and this leads to problems like low throughput, wasted bandwidth and increased delays. Different variants of TCP handle congestion differently, and our aim is to analyze how these variants compare against one another in a series of experiments where we simulate a network environment under varying conditions and investigate the mechanisms these variants deploy for congestion avoidance and control.

In the first experiment, we analyze the performance of four TCP variants, namely: Tahoe, Reno, NewReno and Vegas, and calculate their throughput, latency and packet drop rate to see which variant performs the best on average. We do so by creating a CBR load that creates congestion within the network, and then gathering data about the performance of each TCP variant. We then parse the data and present our findings.

In the second experiment, we analyze the fairness between 4 pairs of TCP variants to see the allocation of bandwidth between each pair. This is done in a similar fashion as experiment 1, however, we measure all metrics independently for a TCP pair. These are: Reno-Reno, NewReno-Reno, Vegas-Vegas and NewReno-Vegas.

2. METHODOLOGY

2.1 TOOLS USED

Both experiments are run using NS (version 2) which is an object-oriented, discrete event driven network simulation tool written in C++ and OTcl. NS is primarily useful for simulating local and wide area networks and allows for the user to generate trace files that show us the result of every simulation. The resulting trace files from each simulation is then parsed by a Python script and plotted into graphs using MatPlotLib.

2.2 Initial Setup and Topology

In these experiments we simulate a network with 6 nodes, numbered N1 to N6, with a CBR source at N2 and sink at N3. The bandwidth of each link is 10Mb with a standard delay of 10ms.

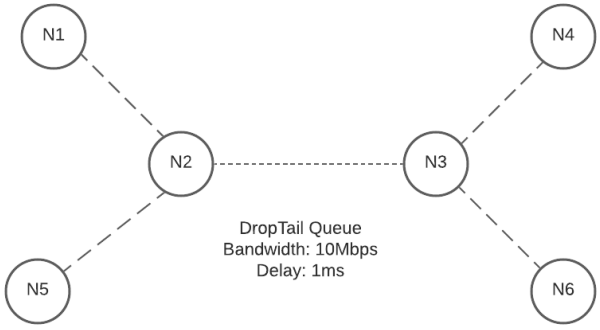


Figure 1: Network topology

We then measure the performance of each of the TCP variants using the following metrics:

$$throughput = \frac{packets\ recived\ \times\ packet\ size}{time}$$

$$latency = \frac{delay\ time\ per\ packet}{packets\ recieved}$$

$$drop\ rate = \frac{packets\ dropped}{packets\ sent}$$

Here, throughput is defined as the total number of bits that can be transmitted over a network in a certain period of time. Latency corresponds to how long it takes for a message to travel from one end of the network to the other, measured in

units of time. Finally, drop rate is defined as the percentage or ratio of packets sent that have been dropped due to congestion.

2.3 Configurations

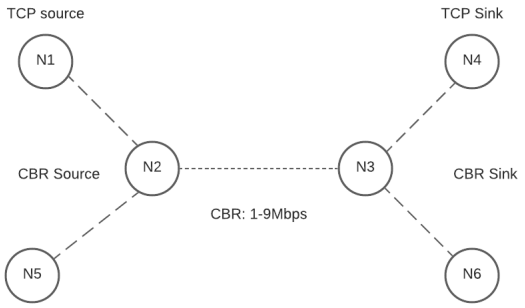


Figure 2: Experiment 1 topology

For experiment 1, we attach a TCP stream from N1, with the sink at node N4, and start a CBR stream at node N2 with the CBR sink at N3. The network uses the DropTail queuing algorithm with default bandwidth of 10Mbps and a 1ms delay. We start both TCP and CBR flows at time 0 and run the simulation for 10 seconds, varying the CBR rate ranging from 1Mbps to 9 Mbps.

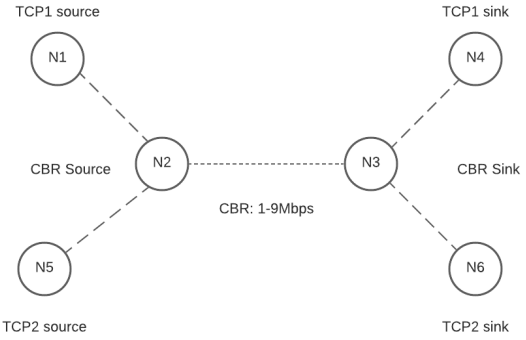


Figure 3: Experiment 2 topology

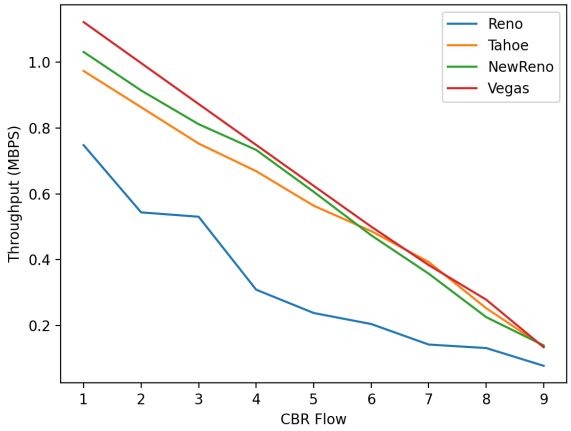
Experiment 2 follows an identical configuration to experiment 1, but with an additional TCP stream starting at node N5, with the second TCP sink being node N6. As with experiment 1, all 3 flows start at time 0 and end after 10 seconds, with the CBR rate varying from 1Mbps to 9 Mbps. However, for this setup we analyze fairness between the following variant pairs: Reno/Reno, NewReno/Reno, Vegas/Vegas and NewReno/Vegas and plot the throughput, latency and drop-rate for the simulation for each pair.

3. EXPERIMENT 1: ANALYSIS UNDER CONGESTION

Experiment 1 focused on creating a simulation of a congested network. We achieved this by setting up a CBR flow that ranged from 1 to 9 and collecting trace-files that evaluated network activity for each of these flows. We inserted ~ 1ms delays to better simulate a congested network. Then we parsed the trace-files to evaluate how different TCP variants performed under the congestion.

Our findings were as follows:

3.1 Throughput:

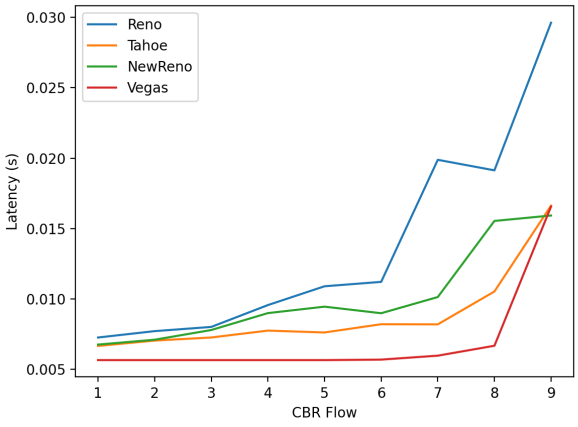


Initially, throughput for all TCP variants started stable and relatively high (>= 1Mbps), with the exception of Reno which was lower. Throughout the CBR flow and delay of 1ms, as network congestion increased all TCP variants faced deteriorating throughput.

Vegas showed highest performance for throughput, both starting out ahead of all other variants, as well as finishing above or equal to the others.

Reno had the most issues initially, but seemed to plateau around CBR = 2, before eventually following the downward trend of its counterparts. We believe that this may be attributable to Reno's fast recovery phase under congestion.

3.2 Latency:



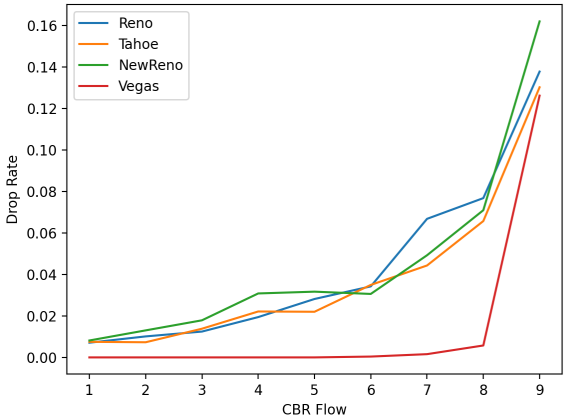
All four TCP variants began on a somewhat similar footing when it comes to latency in the network. Most notably, Vegas started off with the lowest amount of latency, whereas Reno began with the highest.

As the CBR flow begins to increase, we see an upwards trend in the latency for all TCP variants. At CBR = 3, we found that Reno, Tahoe and NewReno showed the closest degree of latency between them. Vegas, however, was still faring comparatively better.

As the CBR begins to rise closer to its limit, we noticed significant increase in latency for Reno around CBR = 6Mbps, and for NewReno and Tahoe at CBR = 7Mbps. For Vegas, we noticed a sharp rise at CBR = 8Mbps.

Overall, we noticed an upwards trajectory in latency as the CBR flow came closer to reaching its limit. Vegas performed the best in a congested network until saturation, where it suffered a steep rise in latency. Comparatively, Reno fared the worst, with two steep rises as the CBR flow closed on its limit. NewReno and Tahoe fared better than Reno overall, and ultimately converged as the CBR flow was at its maximum.

3.3 Drop Rate:



When the network does not suffer from congestion, all four variants of TCP start off at a similar level. However, with an increasingly congested network it becomes clear that Vegas performs the best in networks with high congestion, but not entirely at saturation.

At CBR = 6 we find that both Reno and NewReno rise significantly and follow this upwards trend until saturation. Tahoe fares somewhat better and does not have a major upwards spike until CBR = 7.

Vegas performs the best at lower, and higher levels of congestion until the network reaches the saturation point. At CBR = 8, Vegas suffers from significant drop rates and rises almost to the point of converging with Tahoe and Reno. This is particularly striking as the gap between Vegas and its closest counterpart is also at its highest point at that time.

Overall, we noticed an upwards trajectory in drop rates for all TCP variants, with Vegas performing the best, albeit close to Tahoe and Reno. NewReno suffered from the highest drop rate as the network reached saturation.

4. EXPERIMENT 2: ANALYSIS OF FAIRNESS BETWEEN TCP VARIANTS

Experiment 2 meanwhile focuses on investigating fairness of each of the following TCP pairs: Reno/Reno, NewReno/Reno, Vegas/Vegas, NewReno/Vegas with the former being the variant used for sending packets from N1 to N4, and the latter being used to send packets from N5 to N6.

4.1 Reno/Reno

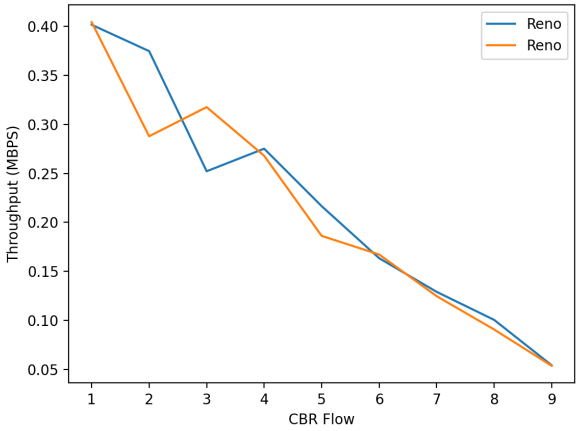


Figure 4.1a: Throughput comparison

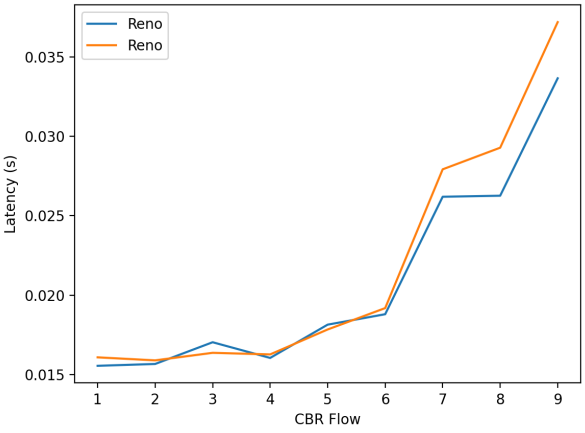


Figure 4.1b: Latency comparison

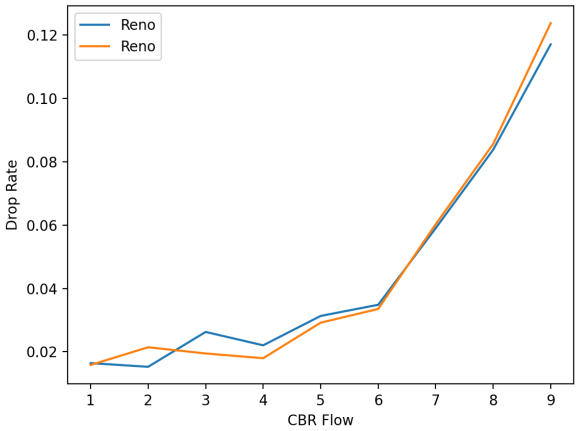


Figure 4.1c: Drop rate comparison

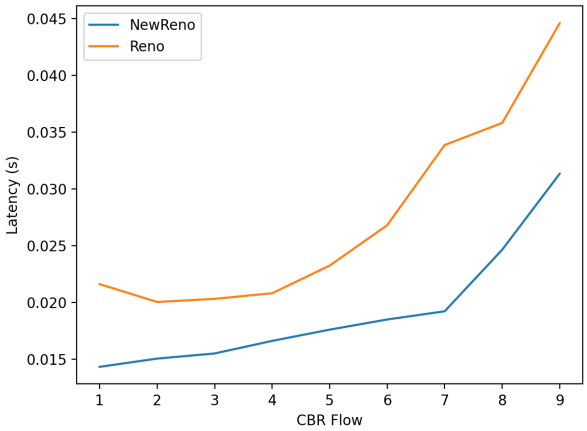


Figure 4.2b: Latency comparison

From Figure 4.1a we can see that both Reno agents tend to oscillate between having the higher share of throughput, but both ultimately converge around the part where the CBR rate is between 7-8Mbps. Figure 4.1b and 4.1c show both Reno agents having a similar Drop rate and Latency, with apparent sharp increases at the 6Mbps CBR rate mark, with one Reno agent having a slightly lower latency starting at 7Mbps. Thus, we can conclude that two Reno agents in the same network will be fair to each other.

4.2 NewReno/Reno

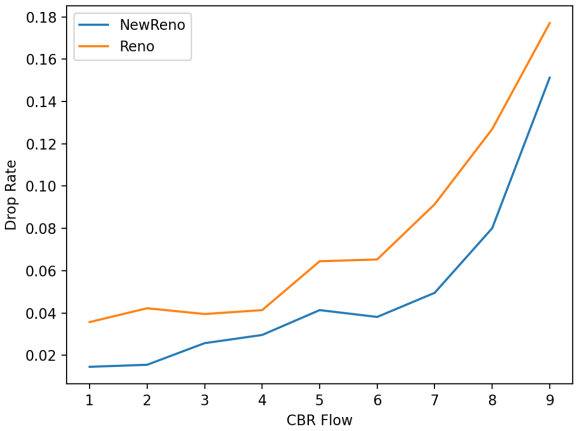


Figure 4.2c: Drop rate comparison

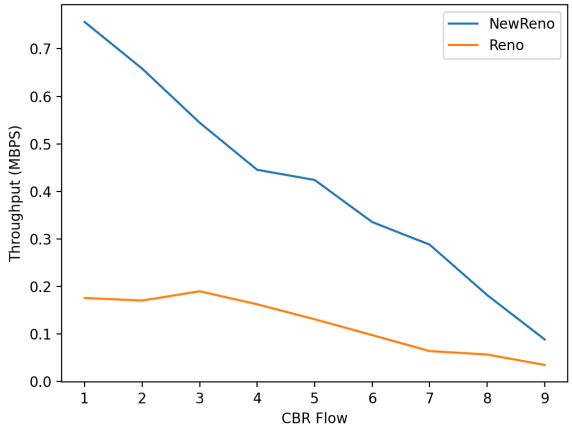


Figure 4.2a: Throughput comparison

From Figure 4.2a it is evident that there is a significant difference between the throughput for NewReno and Reno early on but seem to slowly converge as the CBR rate approaches 9Mbps. Figure 4.2b and 4.2c show the Reno agent having a higher latency and drop rate than NewReno, with the gulf between the two reaching becoming greater when the CBR flow is around 7Mbps. This shows that for smaller delays such as 1ms, NewReno will not provide a fair output when paired with Reno.

4.3 Vegas/Vegas

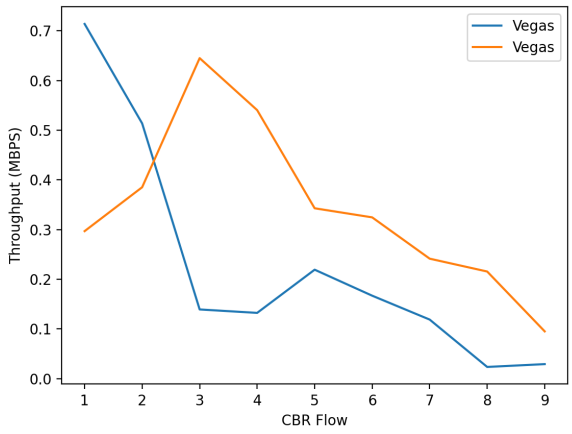


Figure 4.3a: Throughput comparison

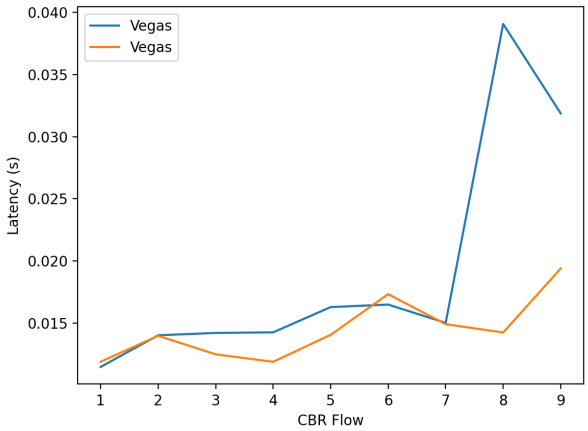


Figure 4.3b: Latency comparison

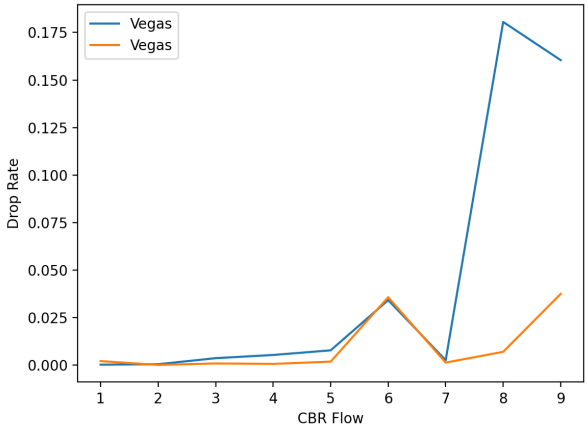


Figure 4.3a shows that while one Vegas agent will start off having a higher throughput, it will soon start losing a major share of the bandwidth, especially at around 3Mbps CBR, where the difference between the two Vegas agents is fairly large. Figures 4.3b and 4.3c show that the drop rate and latency for the same Vegas agent also seems to surge between 7-8Mbps after which is slowly decreases. This shows us that two Vegas agents will not be fair in a network with smaller delays.

4.4 NewReno/Vegas

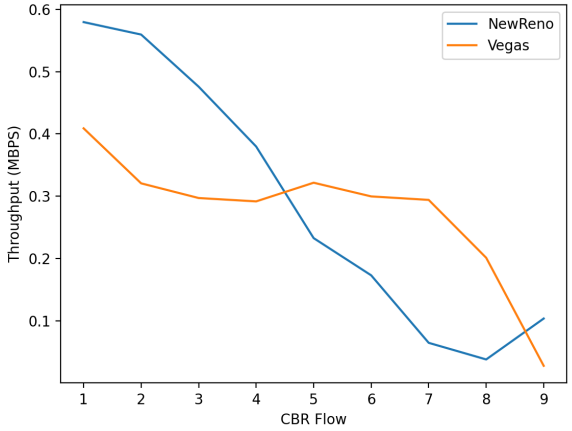


Figure 4.4a: Throughput comparison

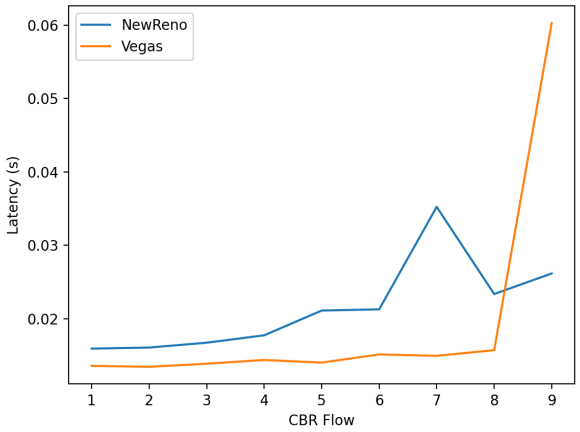


Figure 4.4b: Latency comparison

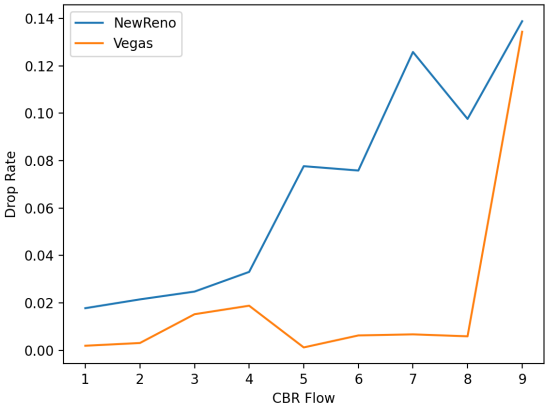


Figure 4.4c: Drop rate comparison

Finally, the fourth part of our second experiment shows the difference between NewReno and Vegas TCP on the same network. Figure 4.4a shows NewReno having a much higher throughput, but then decreasing as Vegas' share of the bandwidth increases, thus causing Vegas to have a higher throughput around 4-5Mbps. However, after 8Mbps NewReno upper bounds Vegas. Figure 4.4b shows that NewReno will have a higher latency than Vegas, and will have a short spike around 7Mbps CBR, whereas Vegas' latency will stay consistent with the increased CBR flow, with a drastic spike after 8Mbps where it overtakes NewReno. Figure 4.4c shows that throughout the simulation, with sudden sharp increases around between 4-6Mbps. Ultimately, we can see that the drop rate seems to decrease while Vegas' drop rate takes a major hike while being consistently low around the 8Mbps mark. These figures also indicate that NewReno and Vegas are not fair in the same network.

5. CONCLUSION

In this paper, through both experiments we learn that TCP performance under simulation is variable and can be influenced by several factors. We attempt to create a viable simulation through multiple runs and CBR flows within a certain range to simulate more accurate network scenarios. We utilize NS-2 to create these simulations and produce accurate trace files of the network activity. We then parse these files and conduct analysis on the data. From this analysis we can conclude:

1. In a network simulation with a single TCP node, Vegas performs better in terms of latency, drop rates and throughput than Reno, NewReno and Tahoe.
2. The simulation results of Experiment 2 indicate that in a same network with smaller delays, Reno/Reno will be the fairest pairing, and it stays consistently fair at higher delays like 10ms as well. Meanwhile, NewReno does not seem to be fair with Reno or Vegas, as it seems to take a major share of the bandwidth during congestion. Two Vegas agents may also not be fair to eachother, but only during low delays like 1ms. At higher delays the chances of two Vegas agents being fair is higher.

REFERENCES

[1] L.L. Peterson & B.S. Davie, "Computer Networks: A Systems Approach," *Fifth Edition*, pp 44-46.

[2] Kevin Fall & Sally Floyd, "Simulation -based comparisons of Tahoe, Reno, and SACK TCP"

[3] Sarah Edwards, Xuan Liu & Niky Rega, "Creating Repeatable Computer Science and Networking Experiments on Shared, Public Testbed"