An Analysis of TCP Variants and Queuing Algorithms

Madhu Patar, Aisha Kothare
Khoury College of Computer Sciences, Northeastern University
Boston, Massachusetts 02115

Abstract—The project aims to analyze the performance of several TCP variants under various load conditions and queuing algorithms. It involves three experiments which are performed using NS-2 Simulator to simulate a congestion on a 6-node network topology with the 4 TCP variants proposed i.e TCP Tahoe, Reno, New Reno, Vegas. The results obtained are analysed and discussed in the paper.

Key Words: Network Simulation, TCP variants, Throughput, Latency, Queuing Algorithms

I. INTRODUCTION

The development of the transmission control protocol has many impacts, with the TCP emerging as the most widely used protocol. It ensures that data is delivered in the correct order and is transmitted reliably between communicating nodes. TCP's high responsiveness to network congestion is one of its advantages. Being a protective protocol, TCP senses incident congestion and attempts to mitigate the effects of this congestion, preventing contact from collapsing. A collection of mechanisms has been placed to detect and mitigate the effects of congestion. Since congestion control plays the key role to ensure stability of the Internet along with fair and efficient bandwidth allocation, this paper aims to compare these mechanisms, and show their differences.

Out of our four variants, TCP Tahoe organizes traffic and deals with congestion using congestion avoidance, slow start, and quick retransmit. TCP Reno adds another mechanism of Additive Increase and Multiplicative Decrease (AIMD) to the existing mechanisms in the Tahoe variant. TCP New Reno introduces us to Fast Recovery while dealing with duplicate packets. Its effectiveness is affected by repeatedly entering the Fast Recovery mode. TCP Vegas, on the other hand, senses congestion in advance and also adheres to AIMD. As soon as it detects an early congestion, it switches to congestion avoidance mode and continues to calculate the difference between actual and planned throughput.

Three experiments are conducted to analyze the performance of these variants. The first experiment is used to measure the performance of these variants by analyzing their throughput, latency and packet drop rate. The second experiment combines two of these variants by implementing them simultaneously to analyze the bandwidth distribution.The final experiment aims to determine the effect of Drop tail and RED queuing on TCP variants Reno and SACK.

We use the following formulae throughout our experiments to calculate the results:

Throughput is calculated in Mbits per second(Mbps) using the formula:

$$throughput = \frac{receivedSize}{(endTime - startTime)}$$

Where

$$receivedSize = \sum_{i=1}^{n} number\ of\ packets \times 8$$

startTime is the time at which the first packet is sent from the source node and endTime is the time when the last packet is received by the destination node.

Latency in a network is calculated as follows:

$$latency = \frac{totalDuration}{number\ of\ packets\ received} \quad (2)$$

where totalDuration is the time taken for all packets in a network to reach its destination.

Drop rate in a network is calculated as follows:

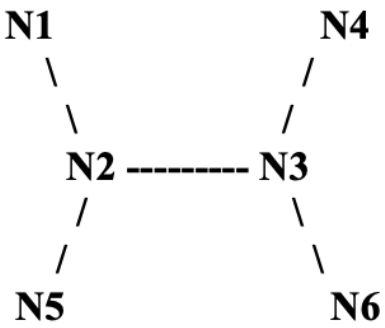
$$dropRate = \frac{number\ of\ packets\ received}{number\ of\ packets\ sent} \quad (3)$$

II. METHODOLOGY

We used the NS-2 tool to simulate the local and wide area networks and TCL for scripting tests. It is an object-oriented, discrete event driven network simulator written in C++ and OTcl. We were able to reduce the difficulty of executing the experiment files by using a Python script. A certain Network topology with 6 nodes connected by 5 duplex links was used in all experiments. The bandwidth for each node was set to 10 Mbps. DropTail and Random Early Drop (RED) queuing algorithms have also been explored.

The performance of these two TCP flows is measured in presence of each other. Pairs that are used for this experiment are Reno-Reno, Vegas-Vegas, New Reno-Vegas, New Reno-Reno. CBR flow rate is changed from 1 to 10 Mbps.

The final experiment uses a constant CBR rate of 7 Mbps. Here, the TCP starts first and once it is steady, a constant CBR rate is introduced. Queuing algorithms Drop Tail and RED are used to analyze the performance of TCP Reno and SACK.



Here is a brief description of the topology:

1. N1 is the TCP source. It is assumed that an FTP application is running on this node
2. N2 is the source of an unresponsive UDP flow. It is a CBR (Constant Bit Rate) generator
3. N3 node is the CBR sink
4. N4 node is sink for the TCP source at N1
5. N5 is another TCP source(for Experiment 2)
6. N6 is a sink for TCP source at N5 (for Experiment 2)

CBR (Constant Bit Rate) is used between nodes N2 and N3. It uses a specific amount of bandwidth. It is not worried about the dropped packets and does not perform congestion control. It just sends packets blindly at a constant rate. This CBR is used between the nodes in all three experiments.

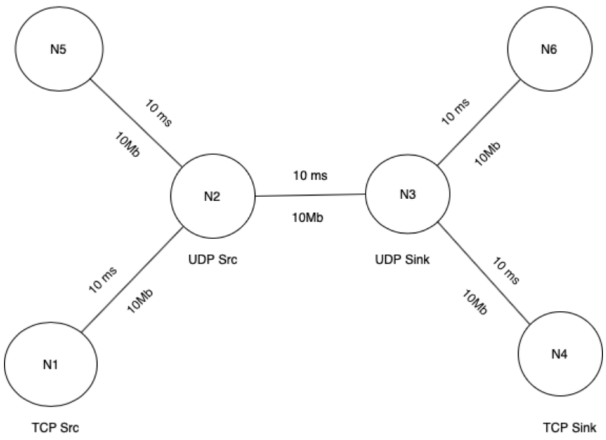
It has been observed that we can achieve a congestion window driven TCP network by using such a value. This also made sure that we did not end up with an advertised window driven TCP network. The data obtained from the execution of tests is then analyzed by the Python script to generate the final result file which gives us performance metrics i.e Latency, Throughput and Drop Rate.

Our first experiment requires us to increment CBR in every step by 1 Mbps until it reaches the bottleneck capacity for the given topology. Performance of TCP variants is then calculated by analyzing the average throughput, latency and packet drop rate.

The second experiment uses the very same topology but another TCP connection is added by attaching a TCP flow at node N5 and a sink at node N6.

III. Experiment 1

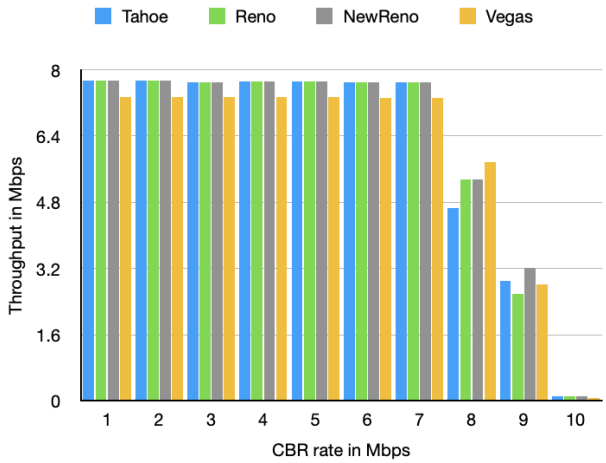
Performance of TCP Variants Under Congestion: This experiment analyzes performance of 4 TCP variants viz. Tahoe, Reno, New Reno, and Vegas to the presence of congestion. This is done by comparing the throughput, packet drop rate and end to end latency of the above mentioned TCP variants.



Experiment 1 Analysis

The graph shown below compares the throughput of TCP Tahoe, Reno, New Reno, Vegas in the presence of different CBR rates.

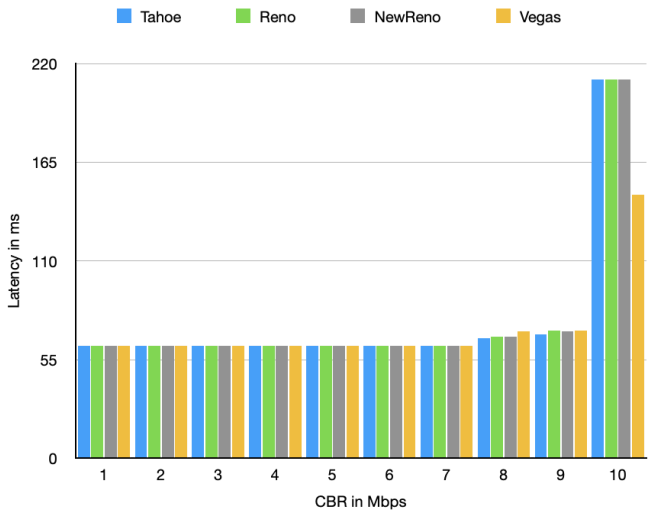
As seen in the graph below, when compared between TCP Tahoe, Reno, New Reno, Vegas, then Vegas has an overall better throughput.This is due to its wise changes in slow start, congestion avoidance and retransmission algorithms.



Graph 1: Throughput of TCP variants in varying CBR Rate

Throughput of New Reno is degraded as compared to Vegas due to throttling of the congestion window. TCP Tahoe has the lowest value of throughput, as it starts from a slow start phase every time after retransmission.

The graph shown below compares the end-to-end latency of the TCP variants discussed above.

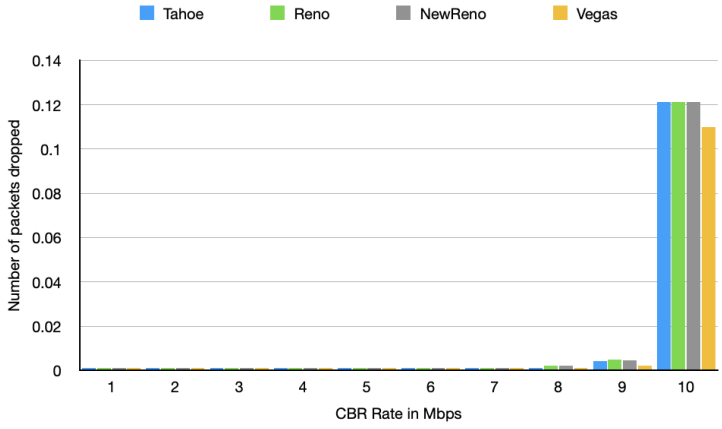


Latency of Variants with varying CBR rate

It shows that after entering the congestion avoidance phase, all the TCP flows increase rapidly. TCP flow of Vegas has the smallest latency and the rest of three flows have similar latency which are approximately 20-25% higher than that of Vegas.

Since Vegas employs an efficient algorithm to measure incipient congestion which detects congestion before the packet losses occur, it is able to detect and avoid the congestion more easily and efficiently.

The next in the line is to study the amount of packets dropped in varying CBR rates. The graph clearly shows that there are minimum packet drops for all TCP variants.



Packet drop rate for varying TCP agents

Vegas has the minimum number of dropped packets when compared to other variants. Vegas has its first dropped packet after the CBR Rate reaches bottleneck capacity.

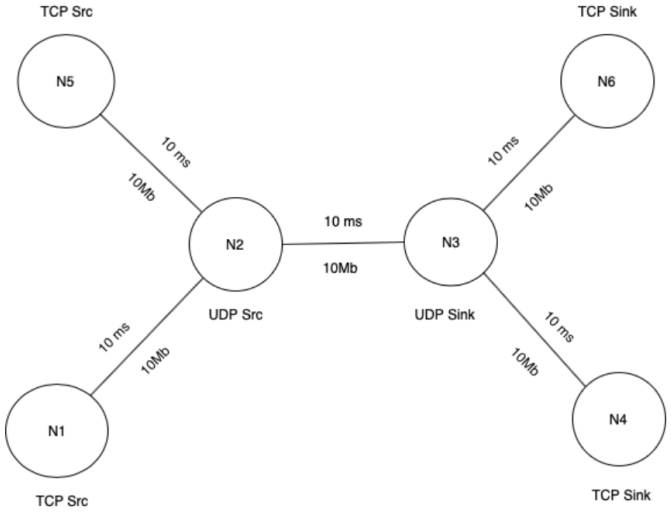
It also has the best latency and the least packet drop rate when compared with other variants. Whereas Tahoe, Reno and New Reno had more latency and packet drop rate. Thus the performance of TCP Vegas was better than other variants. It had a decent throughput with a very good latency and minimum packet drop rate.

IV. Experiment 2

Fairness Between TCP Variants: Several OS use various different TCP implementations, and it would only be fair to compare and check whether each TCP variant is fair to the other when presented in an ideal situation. Whether they are actually fair or not would be interesting to find out. We analyzed the fairness between three TCP variants - Reno, NewReno and Vegas.

TABLE I
TCP AGENT CONFIGURATIONS

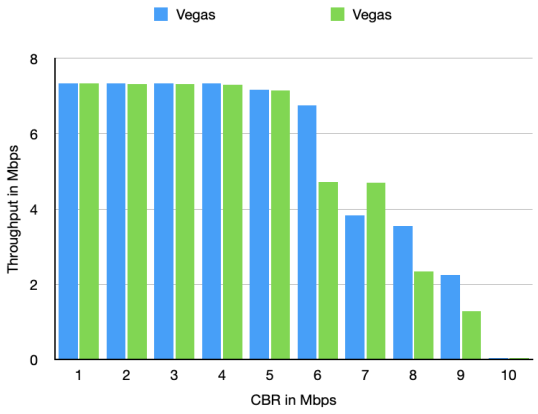
1 → 4	5 → 6
Reno	Reno
New Reno	Reno
Vegas	Vegas
New Reno	Vegas



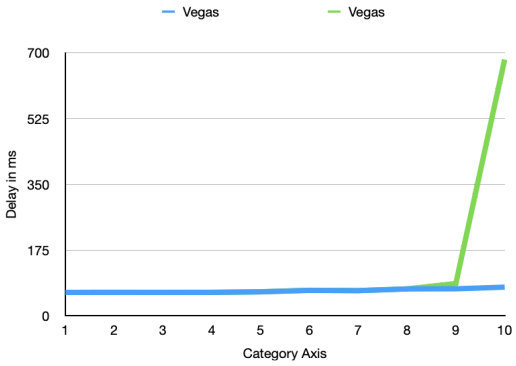
Experiment 2 Analysis

The following combinations were tested to observe their performance against one another:

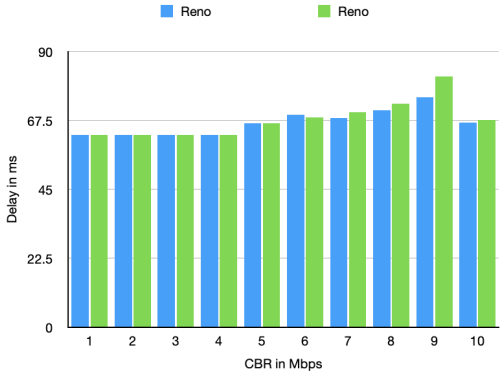
1. Reno/Reno
2. New Reno/Reno
3. Vegas/Vegas
4. New Reno/Vegas



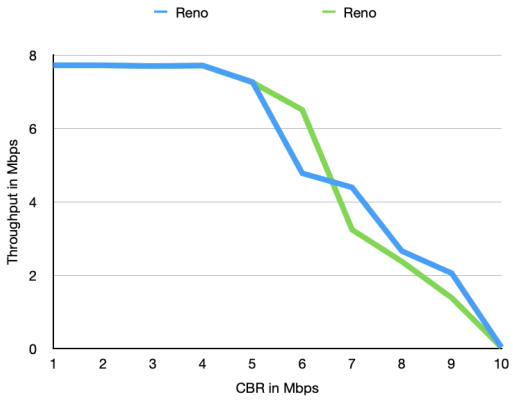
Throughput of TCP variants Vegas/Vegas in varying CBR Rate



Latency of TCP variants Vegas/Vegas in varying CBR Rate



Latency of TCP variants Reno/Reno in varying CBR Rate

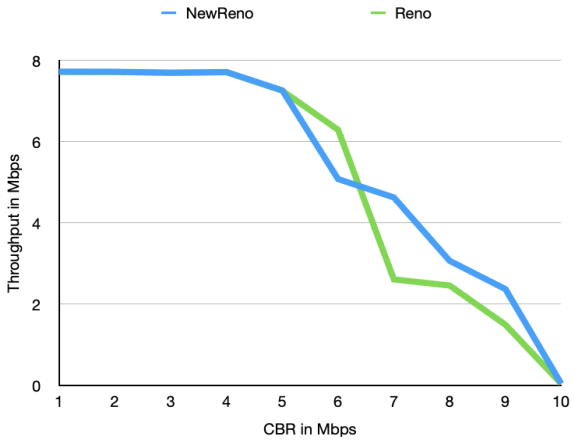


Throughput of TCP variants Reno/Reno in varying CBR Rate

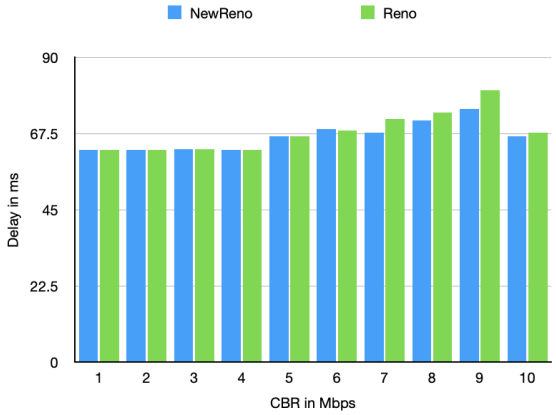
As per the graphs alongside, the fairness between Vegas/Vegas changes after CBR Rate reaches 5 Mbps. The latency is altered after CBR rate 9Mbps. After this we can see that one variant starts performing better than the other.

Due to the non aggressive protocol design of Vegas, it improves the overall throughput of the complete network. In the case of similar TCP variants deployed in the network, we observe that overall throughput for Reno/Reno is lower compared to Vegas/Vegas.

The latency readings above indicate that the two Vegas running on the same network tend to behave in a similar fashion but deviate on reaching knee point. Where, one fares better than the other. The same being true for a Reno/Reno combination too, but with instances where one of them spikes over the other alternatively.



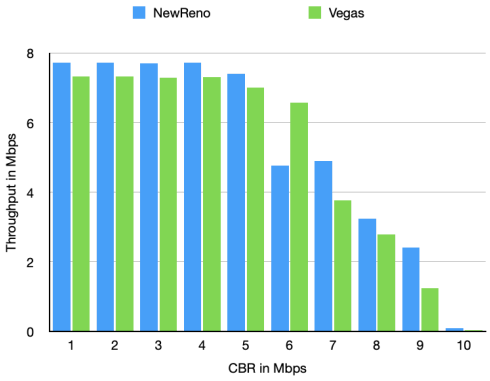
Throughput of TCP variants New Reno/Reno in varying CBR Rate



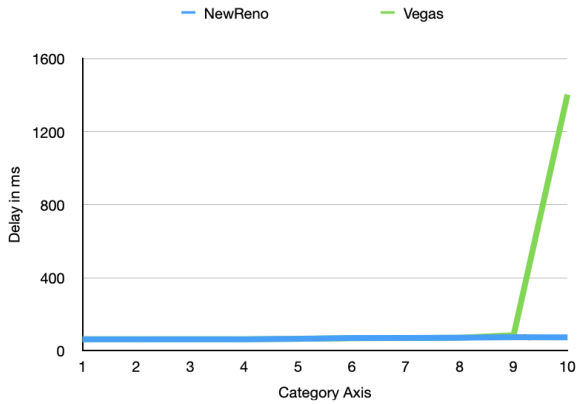
Latency of TCP variants New Reno/Reno in varying CBR Rate

In the New Reno/Vegas graph, we can infer that New Reno dominates Vegas. The fairness is commendable only till the CBR Rate reaches 6 Mbps. We can notice that after this, New Reno dominates Vegas. This occurs since Vegas implements Round Trip Time for computing the network congestion.

Hence the throughput is reduced by CBR when it senses congestion due to another TCP source.
We observe a similar increase in throughput for the TCP variants when a steady flow is used.



Throughput of TCP variants New Reno/Vegas in varying CBR Rate



Latency of TCP variants New Reno/Vegas in varying CBR Rate

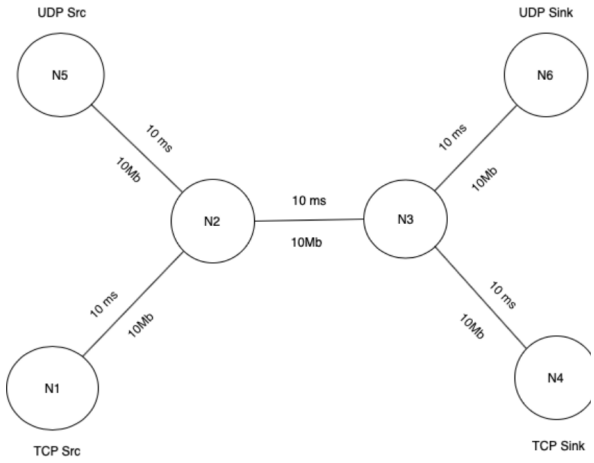
It has been observed that if New Reno coexists with Reno or Vegas, then NewReno has better throughput than the other two flavors of TCP. New Reno does not wait for a fixed number of packets to be dropped unlike Reno and hence NewReno offers better performance compared to Reno.

According to the above graphs, the throughput and latency of both variants are similar with the exception of the CBR Rate of 8 Mbps, when New Reno dominated over Reno.

Thus, we can conclude that if NewReno is along with Vegas, Reno, then NewReno has superior throughput than the other two TCP variants. Since NewReno does not wait for a certain number of packets to be dropped, it performs better than Reno.

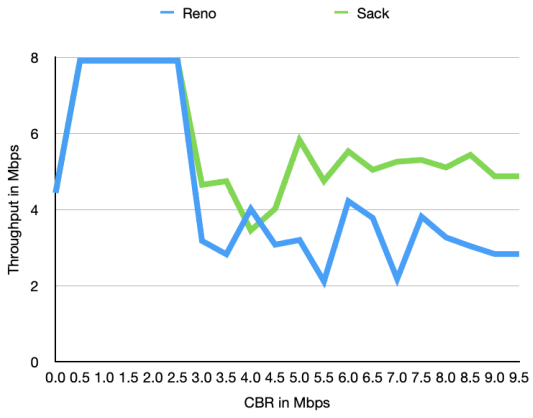
V. Experiment 3

Influence of Queuing: Queuing is crucial for quick and efficient packet transmission in networks. This experiment focuses on two queueing algorithms - DropTail and Random Early Detection (RED). We have worked on Reno and SACK variants of TCP.

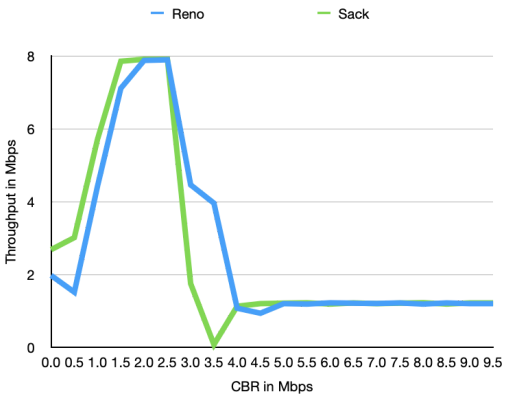


Experiment 3 Analysis

Let us approach the Droptail Queuing technique first. It can be clearly understood from the graph below, that the throughput rises first. This is due to lack of CBR source and the TCP flow tries to slowly increase to 8 Mbps bandwidth. But as the CBR rate of 8 Mbps is started the throughput reduces.
The throughput and thus performance of Reno surpasses that of SACK in this experiment.



Reno vs Sack Throughput using DropTail Queuing

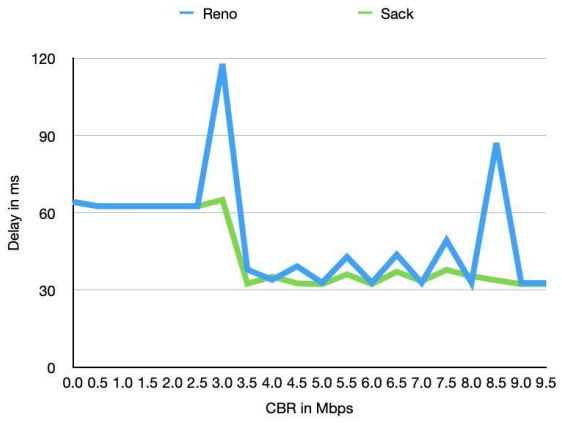


Reno vs Sack Throughput using Red Queuing

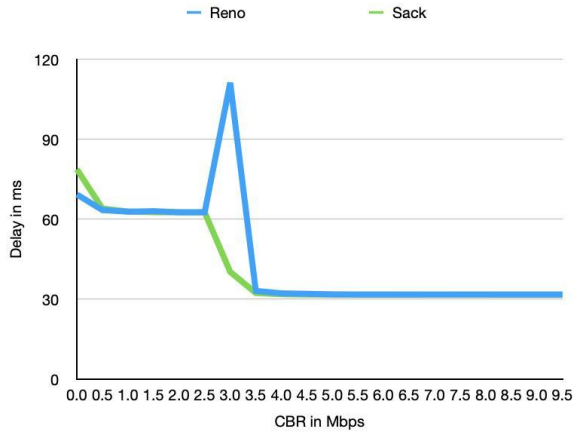
But when throughput of Reno and SACK are compared, SACK wins it by a close margin. The throughput is the same for these two variants till the CBR flow is started. But as CBR initiates, Reno's throughput takes a hit.

Next, we proceed to the RED queuing algorithm. This experiment uses the same topology as used by Drop Tail queuing.

However, a performance difference does exist and this when the CBR source is started. Initially the throughput takes a small hit, but over time, TCP Reno and SACK perform better. RED queuing performs better when the CBR source is present and qualifies as a better queuing algorithm.



Reno vs Sack Delay using DropTail Queuing



Reno vs Sack Delay using Red Queuing

Latency in the TCP connections seems so sharply elevated once the CBR flow begins but gradually stabilizes over time. We observe that both the queueing techniques have similar latency results however RED offers more stability with lower latency than its counterpart. This could be because of how RED handles packets.
With higher traffic and a full buffer, DropTail drop packets while RED considers probabilities to manage new packets arriving.

VI. Conclusion

It was very interesting to study and observe the subtle differences exhibited by the various TCP variants under several scenarios.
For the specific topology of six nodes, after experimenting and observing the behavior of the four variants of TCP i.e. Tahoe, Reno, NewReno and Vegas under different load conditions, we can conclude that Vegas performs the best. It presents better throughput, lesser drop rate and lower latency when a congestion is brought to a link.

In terms of how fair a TCP variant is to the other contenders, it is observed that the combination shows high fairness compared to the combination of dissimilar pairs irrespective of which link. The combination of differing pairs would obviously not perform better as one of them would always be aggressive towards the other and focus on bettering its own performance.

Queuing algorithms definitely do play a role in the throughput and delay of a network. The RED algorithm shows better performance than DropTail in networks with high traffic. This would be because RED adapts a strategy to determine how to drop packets instead of dropping. everything once the buffer size is reached. SACK offers better throughput and stability and consistency than Reno.

TCP's congestion control, which enables the protocol to change the end-to-end communication rate to the bandwidth on the bottleneck connection, is a critical design problem. It is capable of poor performance in high-bandwidth networks due to its slow response time and large congestion windows as packet loss is interpreted as a sign of congestion by congestion control algorithms. Random and high-probability glitches, as well as sporadic communication, characterize wireless networks.
Since TCP lacks a mechanism to distinguish between congestion and wireless random losses, the latter can result in significant throughput degradation. As a result, improving TCP congestion management is critical to meeting current requirements in new applications.
In the near future, we would further like to analyse the behaviour of other TCP variants viz. BIC, CUBIC, compound and so on, to compare their performances as well.

VII. References

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

[2]Improvement of TCP connection throughput over noisy environments by Alexander Cheskis.