Performance Analysis of TCP Variants

Srikar Yagavandla
NU001310900
Email: yagavandla.s@husky.neu.edu

Nagasree Yashwanth Akula
NU001375498
Email:akula.na@husky.neu.edu

Introduction:

In this paper, we perform a detailed study of a few TCP variants like TCP Tahoe, TCP Vegas, TCP Reno, TCP NewReno to understand and make a comparative study of how each TCP variant works under congestion. For this purpose, we perform three experiments in the NS-2 simulator to analyze the performance of each of the variants under varying load conditions and queueing algorithms.

In the first experiment, we compare the performance of TCP variants(TCP Tahoe, TCP Vegas, TCP Reno, TCP NewReno) under the influence of various load conditions in the form of a when a varying UDP(CBR-Constant Bit Rate).

The second experiment aims to understand how a TCP variant is fair to another when two TCP flows are installed in the network. The fairness parameters are recorded by performance metrics like throughput, end to end latency and packet drop rate.

The third experiment compares the performance of two Queueing disciplines: DropTail and Random Early Drop(RED).
The methodology, network simulation, results are explained in the following sections.

Methodology:

All the network simulations are done using the NS-2 simulator, an event driven network simulator which gives the flexibility of varying different parameters like CBR start time, TCP start time etc.. The network topology used for the purpose of the experiments is shown below.

There are a total of 6 nodes installed(N1 - N6). The bandwidth in the network is set to **10 Mbps**. For experiment 1, the delay is set as 2 ms. For experiments 2 and 3, this delay is set as 10 ms.

Performance Metrics:
The performance metrics that are used to compare the performance of each of the TCP variants are Throughput, End to End Latency, Packet Drop Rate. Each of them is described below.

Throughput: Throughput is defined as the total number of bits successfully sent over the network per unit time. It is measured in **Mbps**(Megabits per second).
Throughput = (total number of packets successfully delivered * average packet size) / time

End to End Latency: End to end latency is defined as the time taken for the source to send the packet successfully to the receiver. NS-2 trace files have an event called **'r'** standing for received which is an event for successful delivery of packets. In this paper, end to end latency and delay are used interchangeably.

End to End Latency = Σ(Delay of packets successfully delivered) / (Total number of packets delivered)
Delay of each packet delivered in defined as the time difference between the 'r' event of the packet and '+' event of the same packet. '+' here stands for an enqueue event.

Packet Drop Rate: It is defined as the ratio of packets dropped to the packets that were sent over the network.
Packet Drop Rate: (#Packets dropped)/ #Packets sent

Experiments Configuration:

We performed the experiments 1 and 2 using the DropTail queueing discipline with default queue size.
The window size used for experiment 1 and 2 is 20, which is the default size in NS-2.

For experiment 1, we added a CBR from N2 to N3, TCP Flow from N1 as source and N4 as sink. We compared the packet drop rate, throughput and end to end latency

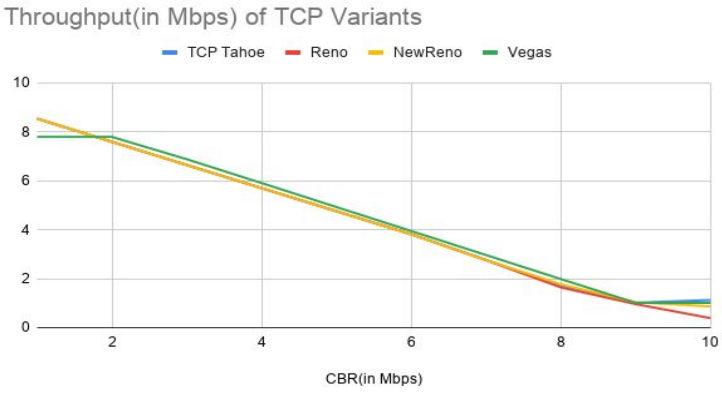
metrics of the network as a parameter of CBR by increasing it gradually from 1 to 10.

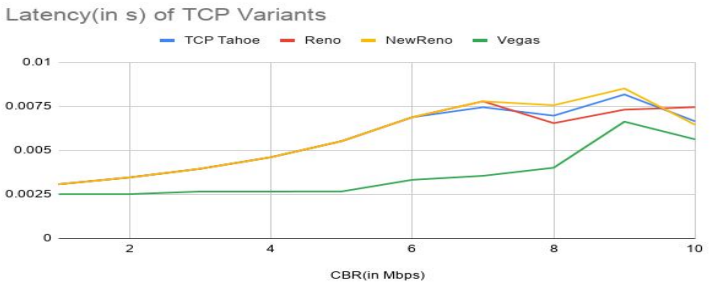
For experiment 2, we added a CBR from N2 to N3. We set one TCP flow from N1 to N4 and other TCP flow from N5 to N6. We run the experiment with 4 TCP combinations(TCP Reno/Reno, TCP NewReno/Reno, TCP NewReno/Vegas, TCP Vegas/Vegas) with one flow from N1 to N4 and other flow from N5 to N6. As a fairness indicator, we plot the throughput, end to end latency and packet drop ratio metrics on a graph to understand how they compete for resources under congestion.
In experiments 1 and 2, we start both the TCP and UDP flows at 0.1 seconds and end them at 10 seconds.

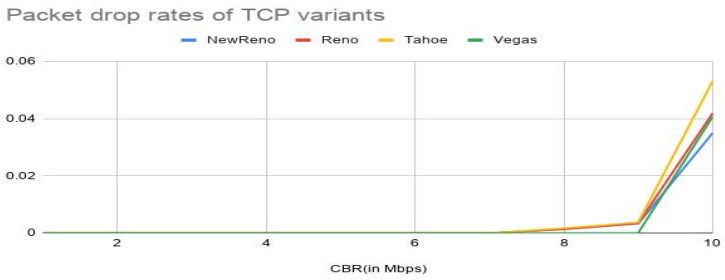
For experiment 3, we added a CBR flow from N5 to N6 and one TCP flow from N1 to N4. We compared the delay and throughput parameters by changing the queueing discipline and TCP variants. For this experiment, we consider 4 combinations of TCP RED Reno, TCP DropTail Reno, TCP RED Sack, TCP DropTail Sack.
This experiment specifically deals with queueing principles. Hence, as a configuration setting we start the TCP first, allow it to run for 5 seconds till it stabilizes. After 3 seconds, we start the CBR flow as well and record the delay when both are in play. After 17 seconds, we stopped TCP flow at 20 seconds and CBR flow at 20 seconds.

Result Analysis of Experiments

Experiment 1:
In this experiment, for all the variants, TCP flow and CBR flow are started at 0 seconds and stopped at 10 seconds. The experiment is run for 11 seconds.







Throughput:
Before the CBR flow started the throughput of all the TCP variants was fairly high, with TCP Vegas having a relatively lower value when compared to other versions because of its modified slow start mechanism.

With CBR(congestion) gradually increasing, there is a consistent drop in the throughput levels of all the variants with TCP Vegas slightly outperforming others.

Latency:
All the variants have the same latency when there is no or little congestion. With CBR flow increasing, TCP Vegas has the least latency maintaining a significant gap when compared to other TCP Variants. Under high congestion all perform poorly with high latency.

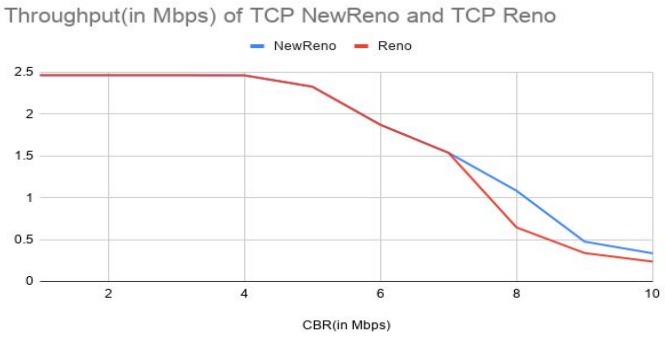
Packet Drop Rate:
Though the packet drop rates for all the variants is almost similar, under high congestion at CBR level 7 Mbps and higher, for TCP Tahoe, TCP Reno and TCP NewReno the graph starts to take off and the packet drop rate increases with increasing congestion. TCP Vegas still has the least packet drop rate even under congestion.

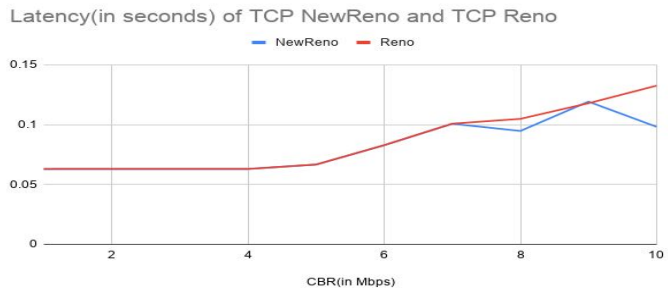
Though TCP Vegas initially has lower throughput because of its modified slow start algorithm where it increases window size for every other RTT, it performs well when compared to other variants as congestion increases. This is because of its new retransmission algorithm where it does not wait for 3 duplicate ACKs and congestion avoidance mechanism which makes it the most preferable choice under congestion. TCP Vegas also has the lowest latency average latency and fewest drops.

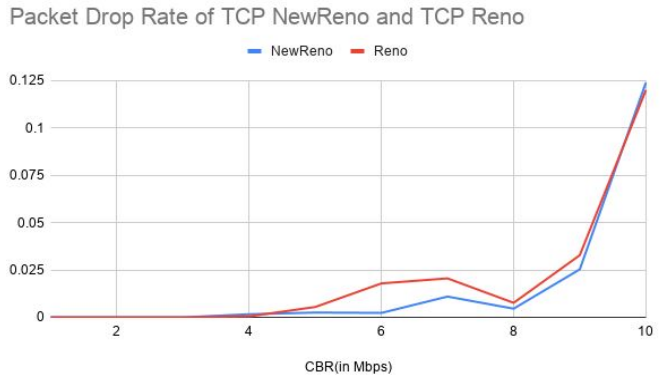
When there is no congestion, TCP NewReno, TCP Reno, TCP Tahoe perform equally well.

Experiment 2:

All combinations of experiment 2 are performed with 10 Mbps bandwidth and 10 ms delay between N2 and N3. The TCP Flow 1, TCP Flow 2 and CBR Flow are all started at 0 seconds and stopped at 10 seconds. The whole experiment is run for 10 seconds.

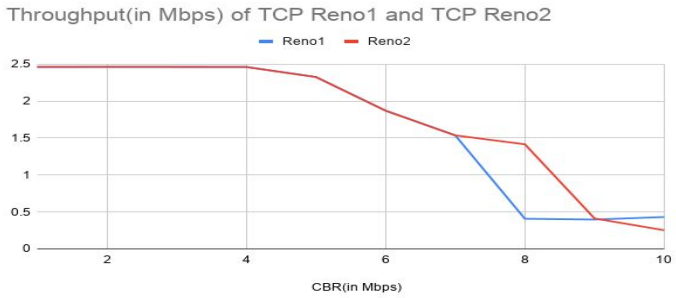


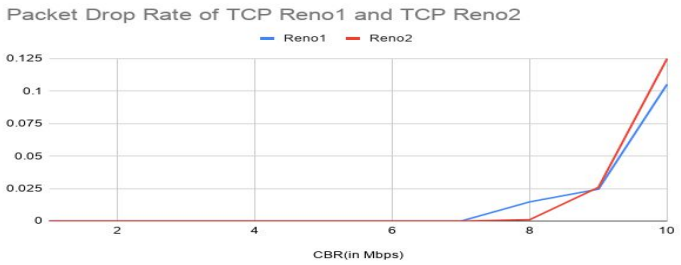


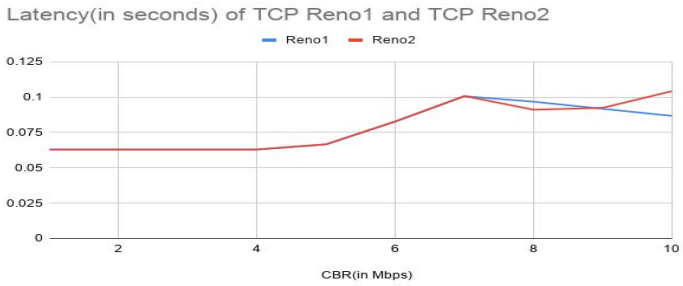


TCP Reno/NewReno

TCP Reno and TCP NewReno perform equally well, produce equal throughput, delay and packet loss rate and are fair to each other before congestion. As CBR levels increase, CBR and TCP flow utilize the bandwidth allocated which results in reduction of overall throughput of the TCP flow. When CBR hits 7 Mbps, congestion in the network starts to increase which leads to multiple packet losses thereby leading to decrease in overall performance of the network. Since TCP New Reno performs well under multiple packet losses scenario, it has higher throughput when compared to TCP Reno, hence the difference in throughputs around 8 seconds mark.
For the same reasons above stated, TCP New Reno reacts well to congestion and multiple packet losses hence the RTT of the packets and packet drop rate is lesser than TCP Reno during congestion.





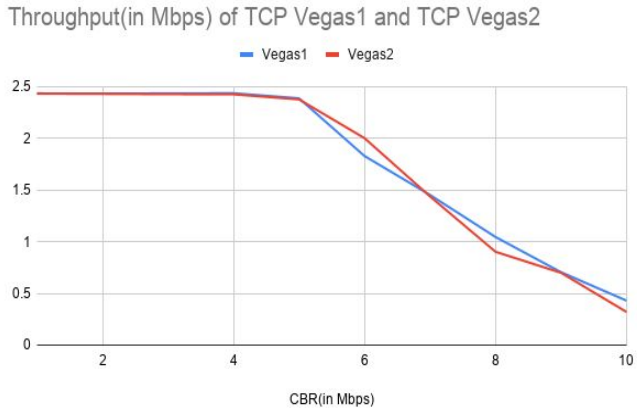


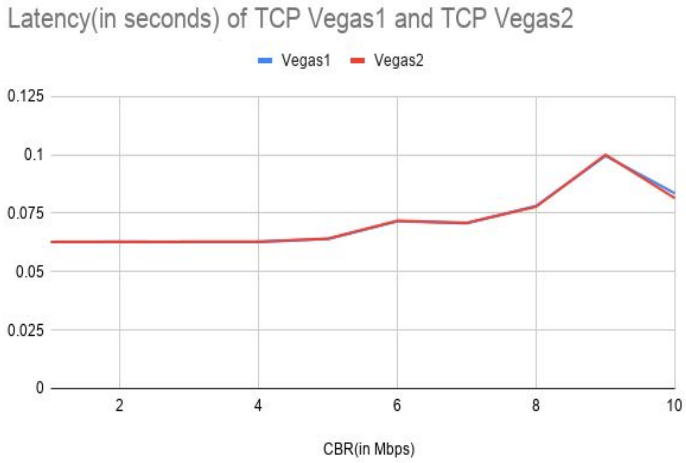
Reno/Reno

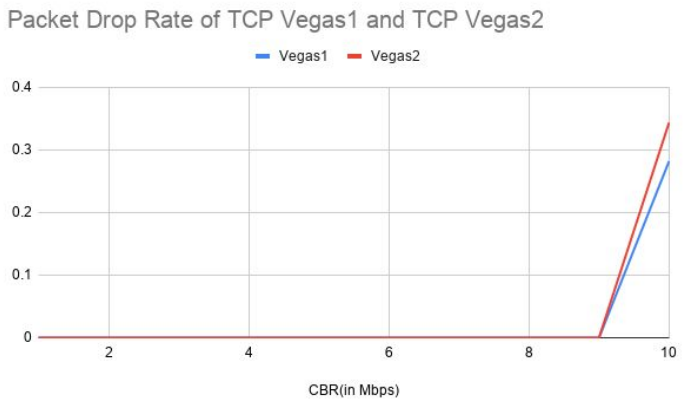
When the CBR is low,as expected both the links have zero drop rates, equally higher throughput and equal delays.

Throughput:
Around the 8 seconds mark, where the congestion is at its peak level, one TCP Reno flow suppresses the other TCP Reno flow .

This behaviour continues till the CBR 10 level mark which is an indication that both the links seem to be working fair to each other even during congestion.
TCP Reno 1 and TCP Reno 2 for most of the time seem to be fair to each other even during congestion. The throughput, latency and packet drop rate vary slightly during 7-9 Mbps CBR levels but stabilizes and is fair to each other there after.







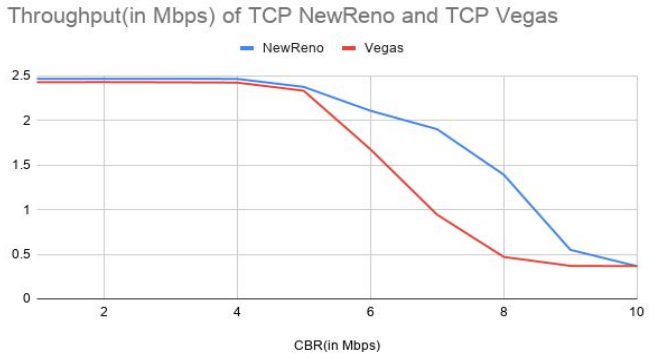
Vegas/Vegas

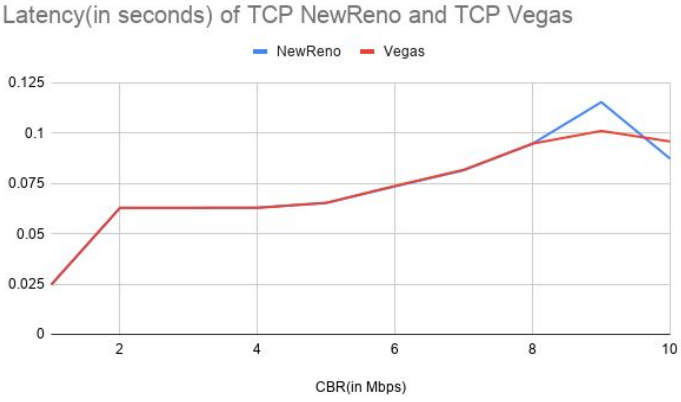
Throughput:

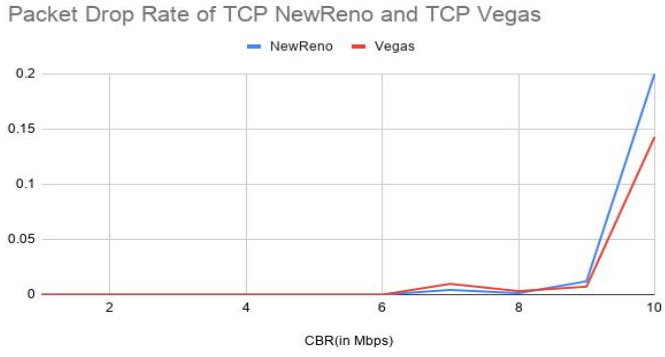
As we can see in the figure, two Vegas get fair throughput when competing with each other for the same congested network.

Delay:
The two of the Vegas running on the same network have the same trends towards congestion which is again being fair to each other.

Packet Drop Rate:
TCP Vegas generally has a lower packet drop rate even in extreme congestion. Only when the CBR bandwidth hits 10 Mbps, it has packet drops. This is exactly what is happening with both the flows. They seem to overlap with each other in packet drops, throughput and delay which means they are fair to each other.







NewReno/Vegas

Throughput: In the throughput graph of TCP NewReno/Vegas, both the flows get the same throughput till the 5 second mark meaning they are fair to each other till this point. At this point, with CBR levels increasing gradually the maximum share of the bandwidth is utilized by TCP NewReno where TCP Vegas is suppressed hence the throughput gap between the flows.

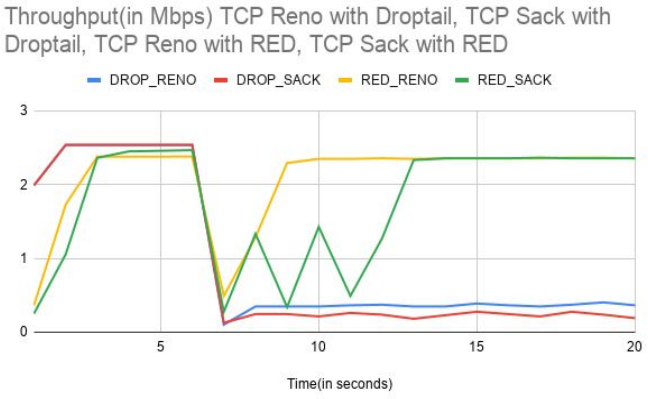
Latency: In the end to end latency graph of TCP NewReno/Vegas, both the flows have equal end to end latencies till high the congestion point. At this point, there is a sudden spike in the latency of TCP NewReno but latency of TCP Vegas remains stable.

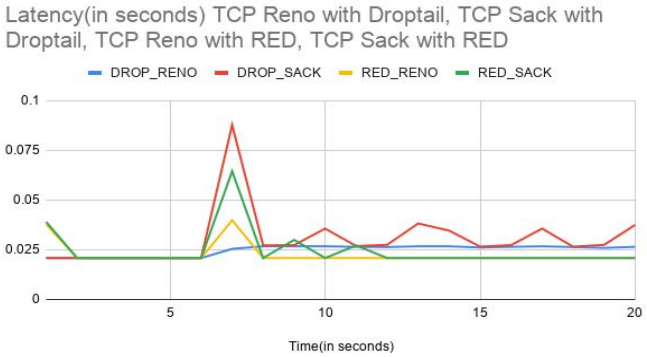
Packet Drop Rate:

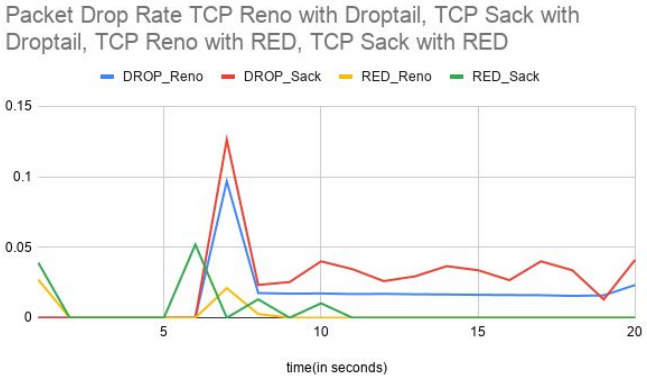
From the packet drop rate graph of TCP NewReno/Vegas, both the flows share equal packet drop rate till the 6 seconds mark. At this point, there is a slight hiccup in the levels of both the flows but the one with TCP Vegas seems more affected than TCP NewReno but eventually they switch and stay fair to each other.

Experiment 3:

In this experiment, for all the combinations, the TCP flow is started first at 0 seconds allowing the TCP flow to stabilize, then the CBR flow is started at 6 seconds. TCP flow and CBR flow are stopped at 20 seconds.







Average End to End Latency:

Before the CBR stream, all the TCP streams have a low and uniform delay without any oscillations. RED queueing protocols(TCP Sack and TCP Reno) start off with really high delay when compared to the rest of the combinations. The TCP flow stabilizes there after till 6 seconds. After the 6 second point, with the introduction of CBR flow with 8 Mbps bandwidth, there is a sudden spike in the levels of latency in all the combinations with TCP Sack with Droptail setting affected the most. Droptail queueing in general gets affected the most with congestion. The spike at 6 seconds clearly shows this. After this point, TCP protocols with Droptail oscillate but never really stabilize whereas TCP protocols with RED queueing get affected only slightly with the introduction of CBR flow and stabilize immediately.
With regards to end to end latency, TCP protocols with Droptail perform the worst under congestion and TCP protocols with RED get affected insignificantly and recover almost immediately.

Average Throughput:

Before the CBR stream, all the tcp streams have a good throughput without any oscillations. The TCP flows look stabilized till 6 seconds. At this point with the introduction of CBR flow of 8 Mbps, there is a drastic drop in the throughput levels of all the TCP flows with Droptail queueing mechanisms suffering the most. Over time,

TCP protocols(Sack and Reno) recover well to see the throughput rising back to where it was initially even under congestion where TCP protocols with Droptail queueing mechanism never really recover.

RED performs better than Droptail for the following reasons. In the conventional tail drop algorithm, a router or other network component buffers as many packets as it can, and simply drops the ones it cannot buffer. If buffers are constantly full, the network is congested. Tail drop distributes buffer space unfairly among traffic flows. Tail drop can also lead to TCP global synchronization - all TCP connections "hold back" simultaneously, and then try to push packets into the links simultaneously. Networks become under-utilized and flooded—alternately, in waves.
RED addresses these issues by preemptively dropping packets before the buffer becomes completely full. It uses predictive models to decide which packets to drop.

Hence the performance of RED queueing discipline is significantly higher than Droptail queueing discipline.

With the introduction of CBR flow, the TCP flow sees a major hit with reduced throughput and increased end to end latency and packet drops for the reasons stated above.

From the throughput, delays and packet drop rate graphs, it is clear that TCP Sack with RED recovers almost quickly after a spike at 6 seconds mark and packet drop and latency stabilize thereafter. Though TCP Sack with RED takes times to recover as seen with 2-3 increase and decrease throughput levels, it eventually catches up with the other queueing disciplines. It is not advisable to use Droptail with TCP Sack as is evident from the graph because of higher delays, slow recovery and drastic drop in throughput.

Under 0 congestion, when there is only TCP flow in the network, both the queueing disciplines work fairly with respect to allocation of bandwidth. But when CBR flow starts at 6 seconds, it is clear that CBR flow takes a share of the resources of the network. After this point, Droptail queueing discipline as visible from the throughput graph almost never recovers whereas RED queueing discipline coupled with TCP Reno recovers immediately and goes back to where it was before CBR flow. TCP Sack with RED however after a few oscillations eventually at 13 second mark recovers to gain maximum throughput.

Conclusion:

Results obtained in the above experiments withhold the major intuitions regarding the performance of various TCP variants under congestion. From experiment 1, we conclude that though other TCP variants like TCP Tahoe, TCP NewReno, TCP Reno perform better than TCP Vegas before congestion, during high congestion and network traffic TCP Vegas outperforms other TCP variants.
Fairness as a parameter with respect to two different TCP protocols in different flows is a function of multiple parameters like throughput, packet loss rate, latency, bandwidth-delay product. From experiment 2, when two flows share the resources of the same network, one flow may suppress the other. This was evident in the case of TCP Reno/Reno and TCP NewReno/Reno. Fairness can be tested by varying other dependent parameters like delay and start and stop times of each flow.
In general, it would seem that the RED Queue management system performs far better than the traditional Droptail, because it preemptively drops packets from the queue even before the queue becomes full. When the queue is empty incoming packets and enqueued and as the queue fills up the probability of the incoming packets being dropped increases and becomes 1 when the queue is full. However, RED is much more complicated to implement and its adoption rates on the real world internet is actually very low.

References

[1]https://david.choffnes.com/classes/cs4700sp17/papers/tcp-sim.pdf
[2] NS3: NewReno vs Vegas vs Veno TCHAYE-KONDI JUDE - tchaye59@gmail.com Subject:http://kializer.com/3x3U
[3] https://www.ripublication.com/acst17/acstv10n6_17.pdf