Performance Analysis of TCP Variants

Abhishek R and Keerthi P
College of computer and information science
Northeastern University, Boston, MA
abhi2488@ccs.neu.com,keerthi4@ccs.neu.edu

Abstract

In this paper we analyze the following

- Performance of different TCP variants (Tahoe, Reno, NewReno and Vegas) based on congestion control.
- Fairness between different types of TCP variants
- Difference in queuing mechanisms like Drop-Tail and Random Early Drop(RED)

We found that TCP Vegas had less latency and packet drop compared to the other variants during congestion. NewReno variant is the most aggressive compared to other variants when put together under congestion. And Drop tail queuing was better than RED.

1. Introduction

TCP is a very popular protocol. Its first version was the Tahoe variant. Over the years with the evolution of networks and the need for better performing protocol a number of different TCP variants have popped up.

In this paper we wanted to compare few of these variants to see which of them would perform better under similar circumstances. Congestions control is one of the important features of the TCP protocol. Different variants implement it in a different way. We analyze how each variant does this under similar circumstances.

One major issue with different variants of TCP working on the same network is how fair they are to each other. We analyze the fairness factor between different TCP variants

With the amount of data a network has to handle growing exponentially, queueing is another mechanism available in every network for proper flow of traffic. We analyze two different types of queuing mechanism used, Drop-Tail and RED. We use the topology shown in figure 1.1 to conduct all the experiments which were run using the NS2 simulator.

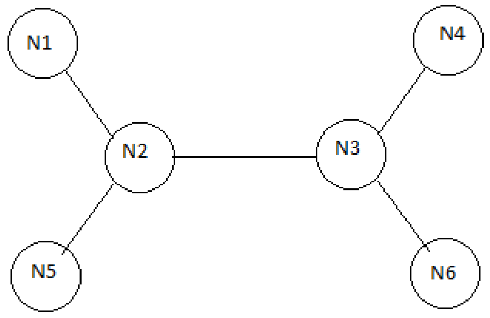


Figure 1

2. Methodology

NS2 is an open-source simulator for research and also provides simulation of protocols, such as UDP, TCP over wired and wireless networks. Its core is written in C++ with a TCL interpreter in the front end. All the scripts to run the simulations were written in TCL All the simulations were performed using NS2 network simulator. All the scripts to run the simulations were written in TCL. The generated trace files were then processed through python scripts. The output results generated by the python scripts were then put in Microsoft Excel to generate the graphs.

For the first experiment we used the above topology with 10Mbps bandwidth and delay of 10ms with DropTail as queuing algorithm. CBR flow over UDP connection between N2 and N3 nodes and FTP (File Transfer Protocol) over TCP connection between N1 and N4 nodes. In order to analyze Throughput, Average Latency and drop rate of different TCP variants such as Tahoe, Reno, NewReno and Vegas, we performed the above experiment with varying CBR rates between 1 and 10 and UDP, TCP start times. Simulation time was set to 100 seconds for TCP to stabilize.

For next experiment, same as the above experiment the links between all the nodes are set a bandwidth of 10Mbps and 10ms delay with DropTail queuing algorithm. CBR flow over UDP connection between N2 and N3 and FTP over TCP connection between N1 and N4 as well as between N5 and N6. The main of this experiment is to determine how fair TCP variants are to each other with varying CBR rate and UDP, two TCP start times. Different TCP combinations are mentioned below: (N1-N4) Reno/ (N5-N6) Reno, (N1-N4) NewReno/ (N5-N6) Reno, (N1-N4) Vegas/ (N5-N6) Vegas and (N1-N4) NewReno/ (N5-N6) Vegas. Simulation time was set to 100 seconds for TCP to stabilize.

In the third experiment all the links between the nodes are set a bandwidth of 10Mbps and 10ms delay with varying queuing algorithms i.e., DropTail and RED. FTP over TCP connection between nodes N1 and N4, whereas CBR over UDP between N5 and N6 nodes. TCP is started first, once the TCP flow is stabilized UDP is started.

3. Experiments

3.1 TCP Performance under Congestion

In this experiment, TCP variants faced congestion with increasing CBR rates. Behavior of all the variants is studied with different start times i.e., UDP first TCP next, TCP first UDP next and UDP TCP started at the same time. This helped us to build a better understanding of how Tahoe, Reno, NewReno and Vegas behave under the above mentioned scenarios. The calculations and graphs provide a detailed overview of average Throughput, Latency and drop rate across varying CBR.

3.1.1 Throughput

Throughput in any medium is defined as the rate at which payload is successfully received at the destination for a given interval of time. It is a measurement of data transfer rate which can be expressed in Gbps, Kbps and Mbps.

Average Throughput for TCP variants such as Tahoe, Reno, NewReno and Vegas were calculated with increasing CBR rate. Figure 2 shows the behavior of each TCP variant in terms of throughput over increasing CBR. From our observation, there was not much difference, but TCP Vegas performed better under more congestion. The reason for Tahoe’s least throughput is, switching to slow start every time a packet is dropped where congestion size window starts from 1. Introduction of Fast-recovery and Fast-retransmit to Tahoe resulted in Reno and NewReno. According to Fast-recovery and Fast-retransmit, a packet is retransmitted if duplicate ACK is received followed by reducing the congestion window size to Sthreshold/2. Vegas on the other hand uses delay based congestion detection, which is the reason for its better performance compared to other variants. Reno, NewReno and Vegas avoid slow start unless there is a timeout, which is one of the reasons for better throughput compared to Tahoe.

Observations from the experiment:

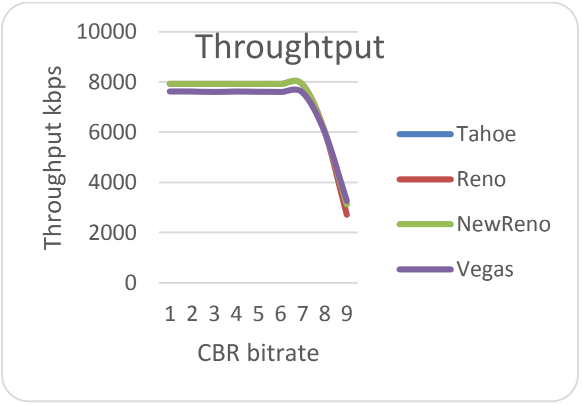


Figure 2

3.1.2 Drop rate

Packet drop rate can be noticed if channel is overloaded or when channel capacity is not fairly utilized, as a result packets are dropped before they reach destination. Drop rate is the rate at which packets are discarded in the network.

Average drop rates of all the TCP variants Tahoe, Reno, NewReno and Vegas were compared with varying CBR rates. From our observations Vegas has least Drop rate compared to other variants. Figure3 depicts Drop rate across CBR rate.

Observations from the experiment:

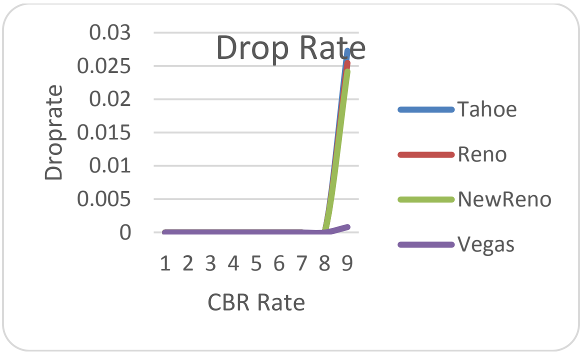


Figure 3

3.1.3 Latency

Latency is time taken by the packet to reach destination and acknowledgement of the same packet to reach source, Latency in other words can be defined as Round Trip Time.

Figure4 depicts average Latency versus CBR bit rate. Average throughput of Tahoe, Reno, NewReno and Vegas were compared with varying CBR rates, from our observations Vegas has least average latency when compared to the rest because it is based on delay based congestion detection, so like other variants vegas doesn't have to wait for 3 or many duplicate ACKS to detect and retransmit a lost packet. Besides Vegas has an enhanced slow start phase which minimizes the effect of congestion on its performance. It retransmits the lost packet with receipt of first duplicate ACK by comparing transmitting time with RTT estimate without having to wait for 3 duplicate ACKs or RTO.

Observations from the experiment:

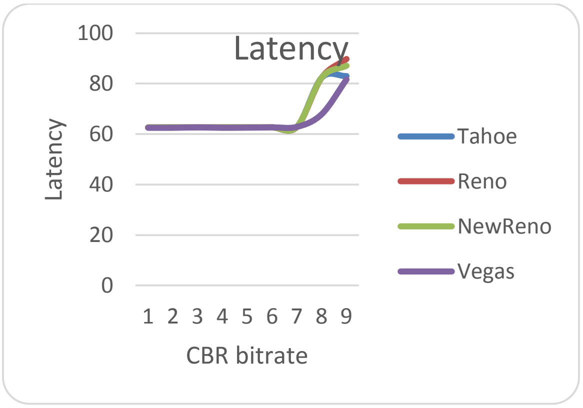


Figure 4

3.2 Fairness between TCP variants

The main aim of this experiment is to determine how different TCP variants are fair to each other with CBR introduced to create congestion.

3.2.1 Reno/Reno

The two Reno variants seems to be fair to each other with increasing CBR, which could be seen from the below mentioned graphs. From figures 5 it is evident that the two TCP variants have almost same average throughput, drop rate and latency. Even with increasing the CBR bitrate bandwidth utilization by these two variants was equal and they are pretty fair to each other.

Observations:

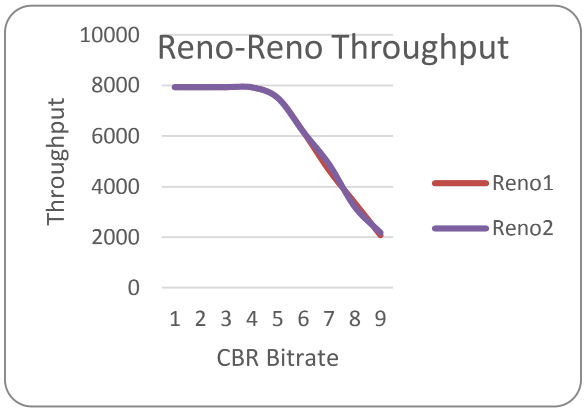
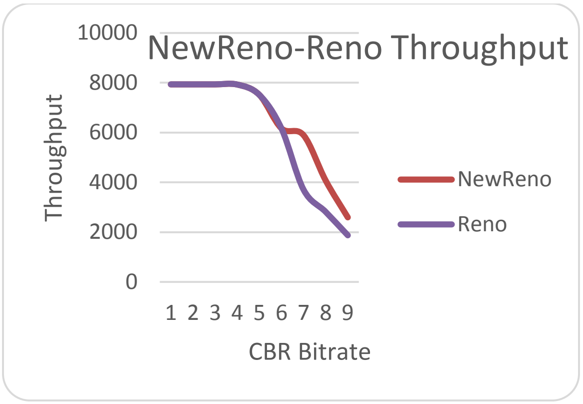
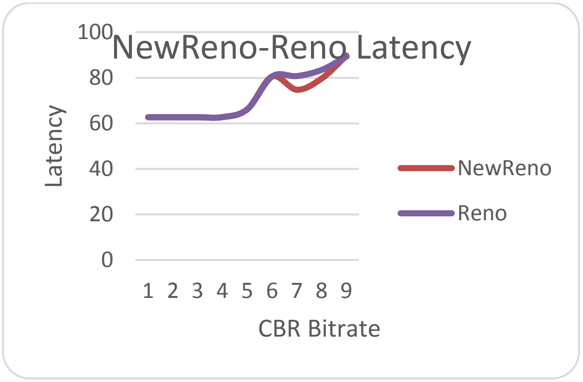


Figure 5

3.2.2 NewReno/Reno

When conducted the experiment with NewReno as one of the variants and Reno as the other variant. NewReno turned out to be more aggressive than Reno where congestion increased and packet drop rate increased. NewReno recovered faster than Reno, allowing it to take up bandwidth faster than Reno. From figures 6 it is evident that NewReno has higher throughput and lower latency when compared to Reno. This behavior of NewReno is due to improvements in its Fast-retransmit phase where multiple re-transmission of packets is allowed. New Reno having more bandwidth leads it to having slightly better latency and lesser drop rates as well.





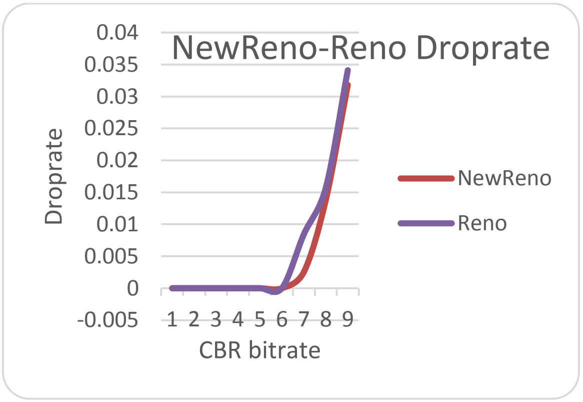


Figure 6

3.2.3 Vegas/Vegas

Here we put two TCP Vegas connections simultaneously on the topology. A CBR flow is introduced to create congestion. Then we calculated the throughput, latency and drop rates. Since both variants follow the same recovery rules, the flows were pretty much fair to each other. Figure 7 shows the throughput that each variant had for different CBR bit rates. We can see that both Vegas flows are pretty much the same.

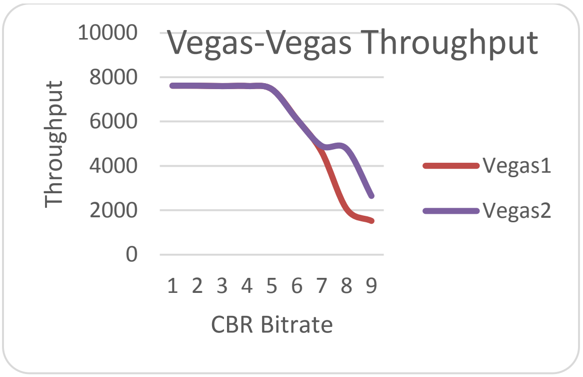


Figure 7

3.2.4 NewReno/Vegas

When conducted an experiment with NewReno and Vegas on the either of the links of the network, it was evident NewReno or Vegas are not fair to each other. From figure 8 and 9 it is obvious that throughput of NewReno is more compared to Vegas because they follow different congestion control mechanisms. NewReno increases/decreases congestion window based on loss whereas Vegas increases/decreases congestion window with change in delay/RTT. NewReno follows the strategy where it aggressively increases its window size if it detects any lost packets in the channel, thus taking up more bandwidth because of which this would lead to increased delay/ RTT for Vegas. This is one of the reasons for NewReno being unfair to Vegas, but when considered individually Vegas has better throughput compared to NewReno.

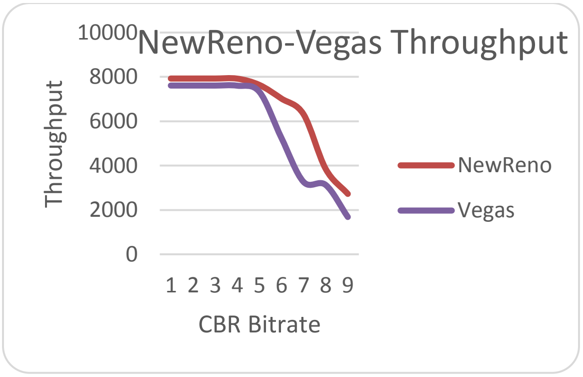
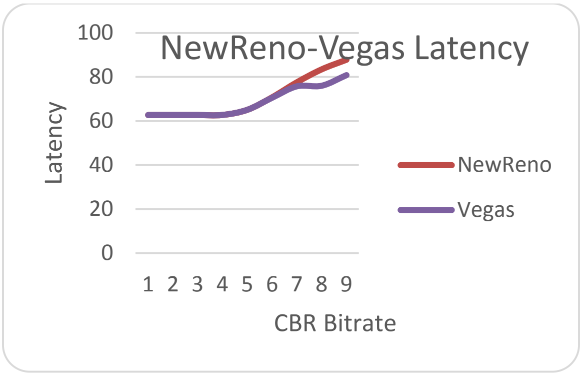


Figure 8



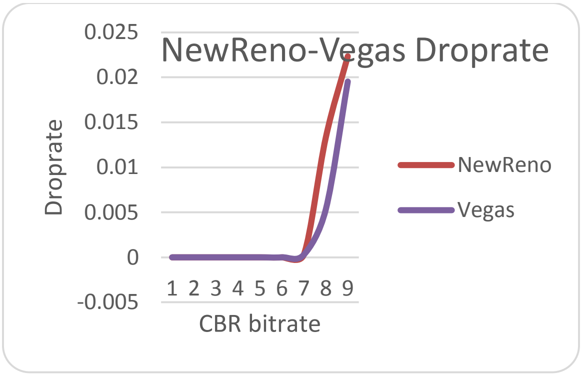


Figure 9

3.3 Influence of queuing

For this experiment we tested two different queuing algorithms DropTail and RED. This was to help us understand which queuing delay provides fair bandwidth and better end-to-end latency. SACK and Reno are the two TCP variants that were tested with both the algorithms.

Droptail is a simple queuing algorithm which is based on FIFO, i.e. if the queue gets filled up with buffered packets, Droptail simply drops/discards the packet trying to entering the queue.

RED (Random Early Detection) on the other hand is based on average queueing length. RED statistically drops packets from flows before it reaches its hard limit. A packet being discarded or queued up depends on average queue length over minimum threshold value.

TCP Reno was implemented over DropTail and RED with constant CBR being introduced at the 150th second, which is after the flow stabilizes. From the output we could see that there wasn't much difference between both the algorithms. DropTail was only a little better compared to RED.

TCP Sack was also implemented over DropTail/RED and constant CBR. From the results it is evident that SACK DropTail's performance is better compared to RED's in terms of Throughput and loss of packets probably because of SACK's high packet loss rate in RED.

As discussed earlier UDP is not fair to other variants in sharing channel bandwidth, thus end-to-end delay is in inversely proportional to bandwidth. Besides there was no remarkable change in latency of either flows.

Observations from the experiment:

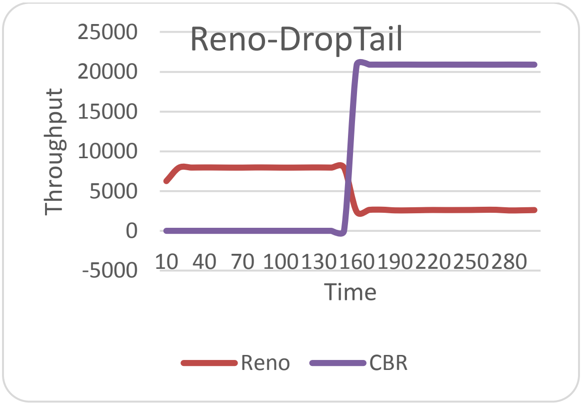


Figure 10

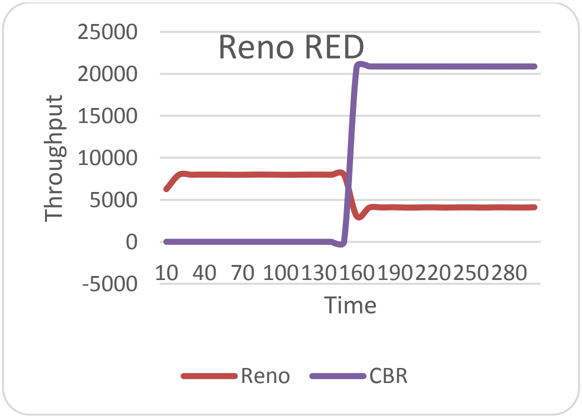


Figure 11

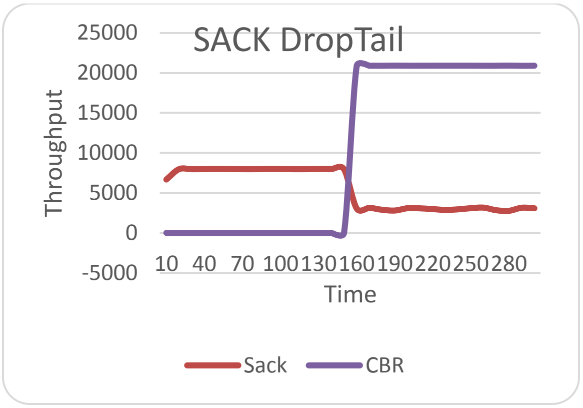


Figure 12

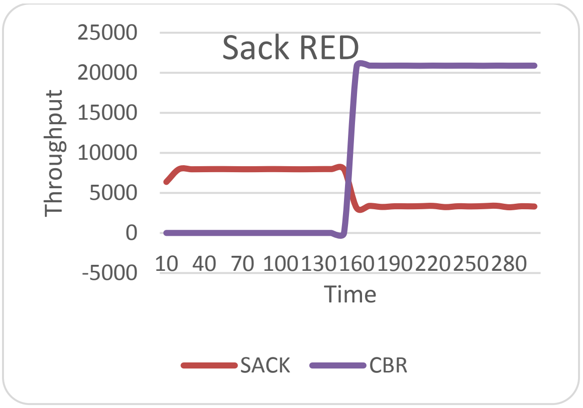


Figure 13

4. Conclusion

In the first experiment, TCP Vegas performed a little better with better throughput, lower latency and low drop rate when congestion levels increased.

In the second experiment, we found that NewReno was more aggressive when run together with a different TCP variant because it aggressively increases its cwnd size during fast-retransmit allowing it to unfairly take up more bandwidth before the other TCP variants.

In the third experiment, TCP Reno/SACK performed better when the queuing algorithm DropTail was used when compared to RED, Sack performed slightly better when DropTail was used than RED.

Out experiments have revealed how diverse the number options are then it comes to the construction of a network. There are so many factors that a network administrator has to take in to build a network for a particular requirement. For example even though TCP Vegas seems to be the best TCP variant, When it comes to being deployed on a network with different types of TCP variants, it is not aggressive enough when it comes to recovery after packet loss.

5. References

[1] K.Fall, S.Floyd "Simulation Based Comparison of Tahoe, Reno and SACK TCP".

[2] Madiha Kazmi, Azra Shamim, Nasir Wahab, and Fozia Anwar "Comparison of TCP Tahoe, R e no, New Reno, Sack and Vegas in IP and MPLS Networks under Constant Bit Rate Traffic"

[3] S.Floyd, T.Henderson "The New- Reno Modification to TCP's Fast Recovery Algorithm" RFC 2582, Apr 1999.

[4] L.S.Brakmo, L.L. Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet"

[5] O. Ait-Hellal, E.Altman "Analysis of TCP Reno and TCP Vegas".