Performance Analysis of TCP Variants

Frooti Govindappa
Dept of Electronics and Computer Engineering
University of Massachusetts
Amherst, Massachusetts
fgovindappa@umass.edu

Sruthi Chilamakuri
Department of Computer Science
University of Massachusetts
Amherst, Massachusetts
schilamakuri@umass.edu

Abstract—TCP is a transport layer protocol which provides reliable and connection oriented transmission. TCP uses several mechanisms such as flow control, congestion control, retransmission, error detection to ensure reliable transmission of data. One of the major hurdles to reliable transmission is network congestion which results in packets being dropped, duplicated or delivered out of order. Several TCP variants have been developed to better handle network congestion. This paper aims to evaluate the performance of these TCP variants. In this paper, we use NS-2 to simulate TCP variants Tahoe, Reno, New Reno, Vegas, and SACK on a simple network topology and analyze their performance based on average throughput, packet drop rate and latency by varying load conditions and queuing algorithms.

Keywords—TCP, Tahoe, Reno, New Reno, Vegas, SACK, NS-2

I. INTRODUCTION

TCP or Transmission Control Protocol is a connection oriented end-to-end protocol providing reliable data transfer. The main purpose of this experiment is to study and analyze the behavior of different TCP variants under various load conditions.

Even though the original TCP worked well and was well known for its reliability, the original TCP did not provide the desired performance and could not be used in the rapidly developing and expanding Internet architecture. This spurred the evolution of several TCP variants such as Tahoe, Reno, New-Reno, Vegas, and SACK among others, each with its own applications and strengths and weaknesses. Conducting this experiment will allow us to understand the behavior of these TCP variants under different network parameters. While conducting this experiment, we discovered that Vegas performs better than other variants and same variations in a system are fair to one another.

Some of these TCP variants are discussed below:

1. TCP Tahoe

Tahoe is the simplest TCP variant. It executes slow-start and congestion avoidance mechanism and it does not have fast recovery state.

2. TCP Reno

TCP Reno is an implementation of TCP used by most networks today. It uses different congestion control algorithms. They include Congestion Avoidance mechanisms, Fast Recovery, Fast Retransmit and Slow Start.

3. TCP New Reno

New Reno was designed to improve TCP Reno's performance. TCP New Reno is a variant of Reno with an improved Fast Recovery algorithm in order to solve the timeout problem which occurs when multiple packets are lost from the same window.

4. TCP Vegas

Vegas will calculate the RTT and compare it with RTT of recently received acknowledgments. Depending on the current RTT value, Vegas detects a congestion and resizes its queue size.

5. TCP SACK

SACK allows the receiver to acknowledge non-consecutive data, which facilitates retransmission of missing data alone. This version of TCP performs poorly when multiple packets are lost from the same window of data.

II. METHODOLOGY

We used NS-2 to conduct these experiments. NS2 is an open-source simulation tool that runs on Linux. It is a discreet event simulator specifically targeted at networking research and provides extensive support for simulation of IP protocols over wired networks. We chose NS-2 as it is reliable and universally accepted. Further, as NS-2 generates an entry in the trace file for every event, a detailed packet by packet analysis can be performed to obtain accurate results.

In order to analyze the performance of TCP variants, we propose to run several experiments on the topology in Figure 1 and generalize the behavior of these protocols across a wider network.

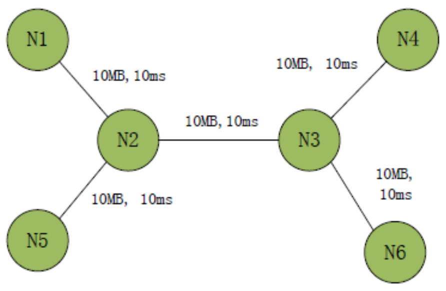

Fig1: Network Topology

Each link in the above topology is full duplex and has a bandwidth of 10 Mbps and default delay of 10 ms.

.

Experiment 1: TCP Performance Under Congestion

In this experiment we will analyze the performance of TCP variants by modifying load conditions. We will add a CBR source at N2 and a sink at N3 and then add a single TCP stream from N1 to a sink at N4. We will increase the CBR flow rate linearly from 1 Mbps to 10 Mbps to induce congestion and observe the behavior of these TCP variants.

This will be performed for different variants (Tahoe, Reno, New Reno and Vegas) under the following test conditions to create randomness:

a) Start and end CBR and TCP flows at once
b) Vary start and end times of both flows
c) Start CBR flow once TCP is stable
d) Start CBR flow when TCP in slow start

The performance of the TCP variants will be evaluated based on three factors: average throughput, latency and packet drop rate as a function of the bandwidth used by the CBR flow and relevant graphs will be plotted to help us identify the best TCP variant.

1)Average throughput (Kbps) =

$$\frac{\text{Number of packets received at destination} * 8}{\text{Run time} * 1000}$$

2) Packet Drop Rate =

$$\frac{\text{Total number of packets dropped}}{\text{Run time}}$$

3) Latency =

$$\frac{\text{Total round trip time}}{\text{Number of received acknowledgements}}$$

Experiment 2: Fairness Between TCP Variants

In this experiment, we will compare the fairness between TCP variants (Reno, New Reno, and Vegas) in the same congested network. We will add a CBR source at N2 and a sink at N3 and then add two TCP streams from N1 to N4 and N5 to N6 respectively. We will repeat the experiments using the following combinations of TCP variants:

a) Reno/Reno
b) New Reno/Reno
c) Vegas/Vegas
d) New Reno/Vegas

The fairness of the TCP variants will be evaluated based on three factors: average throughput, latency and packet drop rate, and relevant graphs will be plotted.

Experiment 3: Influence of Queuing

In this experiment, we will study the effect of queuing methods such as DropTail and Random Early Drop (RED) on overall throughput of TCP variants (Reno and SACK). We will add one TCP flow from N1 to N4 and once the flow is stable we will add another flow, CBR (over UDP), from N5 to N6.

We will plot graphs measuring the performance of TCP and CBR flows over time and verify fairness of each queuing discipline to the TCP and CBR flows. We will compare the impact of DropTail and RED on end to end latency of flows. We will also determine which queuing method is suitable for each TCP variant.

III. RESULTS

A. Experiment 1: TCP Performance Under Congestion

The network topology and flow setup for this experiment is shown in Figure 2.

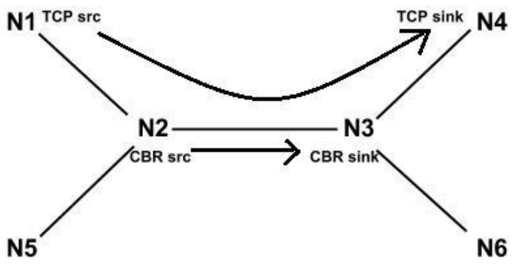

Fig2: Experiment 1 Network Topology

As stated in the section of methodology, we have compared the performance of four mentioned TCP variants, Tahoe, Reno, New Reno and Vegas. In detail, we had set up a CBR flow from N2 to N3 with changing rates that range from 1MB to 10MB with step of 1MB, which is used to generate traffic competing resources with TCP flow. We had also set up a FTP TCP flow from N1 to N4, which will also trespass N2 to N3. It means that TCPs' performance will be impacted by the CBR flow and might cause congestion. TCP packets start from node 1 and enter the queue. They are then removed from the queue before they are received at node 2. The same process happens when the packet is transmitted to node 3. But there may be some instances where packets are dropped due to congestion in the network. If node 1 does not receive ACK from the same packet sequence that it transmitted, it may retransmit the packet until the ACK is received. Once the packet reaches node 3, it is send to node 4 and a ACK from node 4 is sent to node 3

1) Throughput:

The successfully received TCP packets were observed at TCP sink. The total number bytes received were calculated till the last packet was received at that node. We get the simulation result as shown in Figure 3.

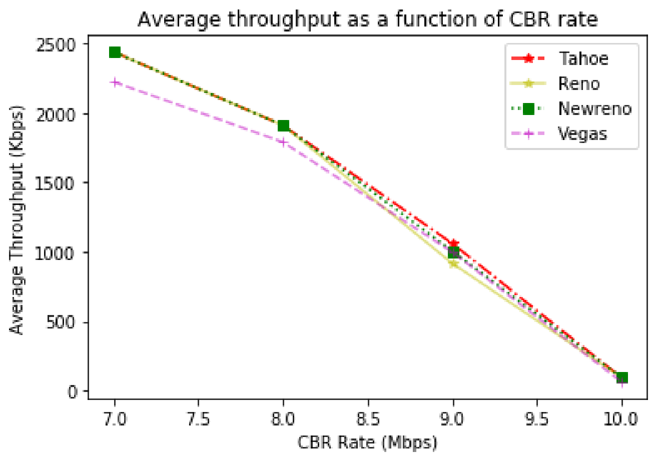

Fig3: Throughput V/S Bandwidth

As the bandwidth of CBR increases, the throughput of Tahoe decreases due to packet drops and it enters the slow start stage and reduces the congestion window to 1 whose value will not grow unless the TCP receives ACK's for all the sent packets and hence it will remain in the slow start for longer duration of time.

The Reno enters and exists the fast recovery mode for each packet in case of multiple drops in one window because of which its overall throughput decreases.

The New Reno throughput decreases a little but shows better performance under high congestion since it stays in the fast recovery stage until it receives ACKs for all the packets that were present when New Reno entered the slow start stage.

From the figure3, we can see that initial throughput values of TCP Vegas is less than other variant's throughput value because Vegas first calculates the base RTT which helps it to figure out how much bandwidth is available to it and predicts congestion well in advance hence it shows a better performance .

2) Latency:

The successfully received TCP packets at the TCP sink were observed. The Time-stamp and the sequence number were saved in the associative arrays. The ACK packets of the received packets from the TCP sink were observed at the TCP source and the same parameters were saved in the associative arrays. In this way RTT was calculated with the help of which Latency was calculated. The average latencies of the TCP variants are depicted in Figure 4.

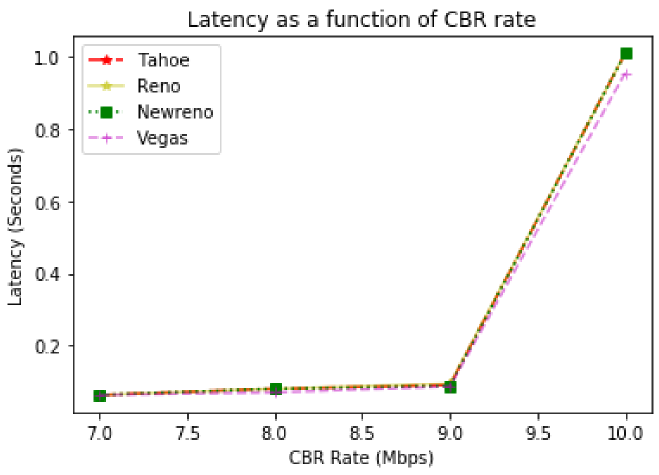

Fig4: Latency V/S Bandwidth

TCP Vegas has the least latency because time taken for ACK packets of TCP Vegas is very less hence RTT is less, making its performance much better. New Reno takes one RTT to detect each loss if multiple packets are lost in on window. This leads to increase in queuing delays and thus high latency. In Tahoe, packet loss is detected only when the retransmission timer expires. Tahoe uses GO-BACK_N windowing technique in which, when the packet drop occurs the number of packets retransmitted during congestion increases leading to larger queuing delays. Reno has high latency because it detects congestion after 3 ACK packets, hence RTT will be optimized late.

3) Packet Drop rate

The total number of TCP packets dropped at all the nodes was counted using a counter with the help of which Packet Drop Rate was calculated. The packets drop rate under varying CBR traffic is shown in Figure 5

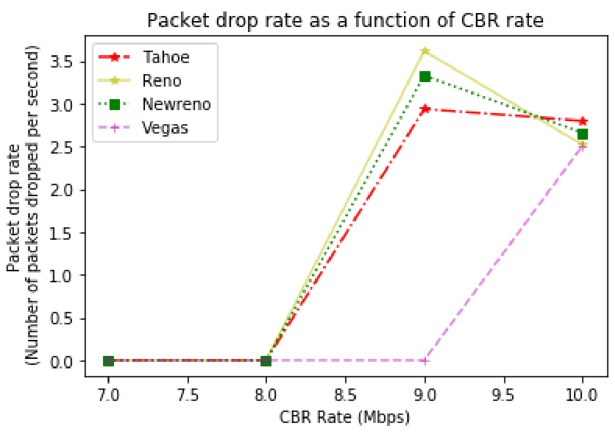

Fig5: Packet Drop Rate V/S Bandwidth

TCP Vegas has the least drop rate as compared to all other variants as Vegas detects congestion in its initial stage by calculating the base RTT and comparing it with RTT of every packet it sends. Reno performs least as it will come to know about congestion after 3 repeated ACKs.

B. Experiment 2: Fairness between TCP Variants

The network topology and flow setup for this experiment is shown in Figure 6. The CBR flow and the TCP flow were started at the same time instance. The TCP variant 1 packets travel between node 1 and node 4 while TCP variant 2 packets travel between node 5 and node 6. The CBR Rate is varied from 1Mbps to 10Mbps between node 2 and node 3.

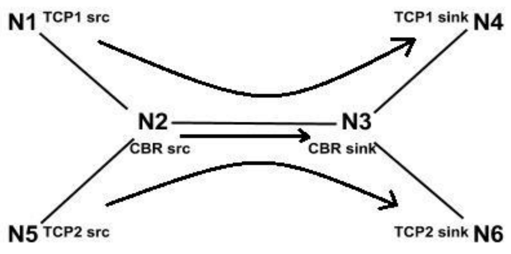

Fig6: Experiment 2 Network Topology

The fairness of variants to one another is analyzed by plotting graphs for throughput and latency for each flow with respect to CBR flow rate for the following pairs.

A. **Reno/Reno**

a) Throughput:

According to the network topology in Figure 6, both TCP1 and TCP2 link are assigned as Reno. From the simulation results we know that both the TCP Reno flows are fair to each other in terms of bandwidth. This is because both the flows follow the same algorithm of congestion control. The throughput of both the TCP Reno flows decreases in a similar fashion under congestion. when one TCP variant has more throughput, the other one suppress due to the bandwidth capacity, and vice versa. Hence, the two TCP agents can alternatively utilize the bandwidth. Thus it is fair for the two Reno TCP variants to be on the same system.

b) Latency:

From the simulation results we know that both the TCP flows will try to reach equilibrium

B. **New-Reno/Reno**

a) Throughput:

Now, we assign New-Reno to TCP1 and Reno to TCP2 link in the fig 6. In a combination of New Reno and Reno, the former has a better throughput utilization of the channel as compared to Reno. This is because New Reno can handle multiple packet drops when congestion increases in the network. New Reno does not come out of the fast recovery phase unless all packets present at the time when it entered fast recovery phase are acknowledged. As a result, New Reno will have an advantage and so is unfair as compared to Reno.

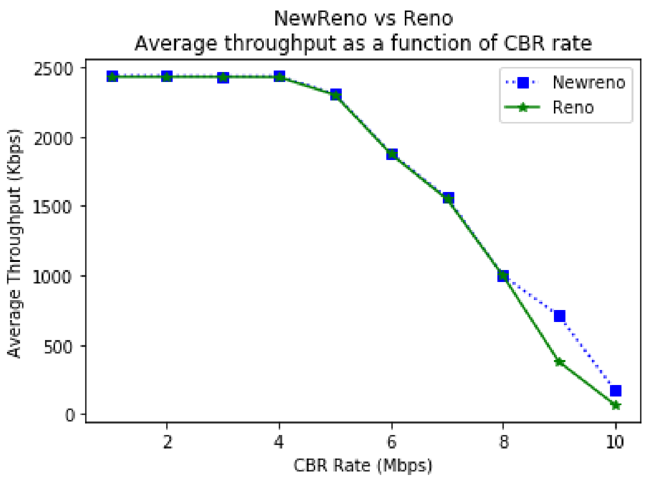

Fig 7: New Reno vs Reno (Throughput)

b) Latency:

In Reno, there will be a lot of ACKs before retransmission thereby increasing the net RTT which in turn gives high latency which can be seen in the figure8. Hence performance of New Reno is better than Reno.

.

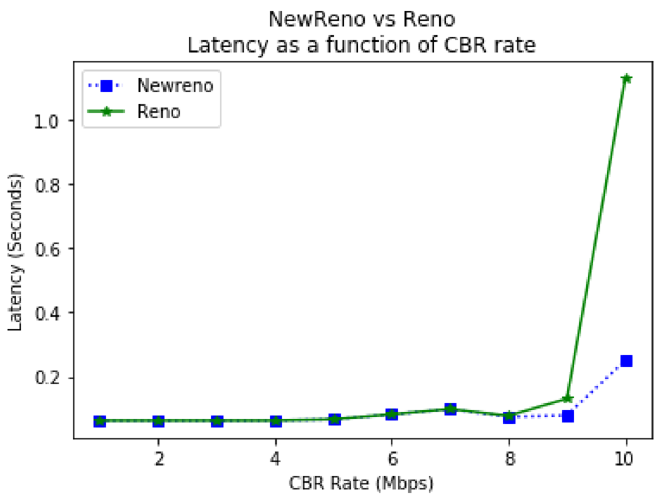

Fig8: New Reno vs Reno (Latency)

C. **Vegas/Vegas**

To both TCP1 and TCP2 links on Figure 6, TCP Vegas is assigned. From the simulation results we know that both Vegas flows are fair to each other. When one of the TCP Vegas flow say Vegas_1 detects the occurrence of congestion in the network, it immediately backs-off by reducing the number of packets being sent on the network. Thus, the other TCP Vegas flow say Vegas_2 utilizes the bandwidth of the network and vice-versa. Therefore, we can see throughput of both the flows growing and then falling down. But the overall throughputs of both the flows are almost the same. This combination can be used where equal traffic flow is required in a network.

D. **NewReno/Vegas**

a) Throughput:

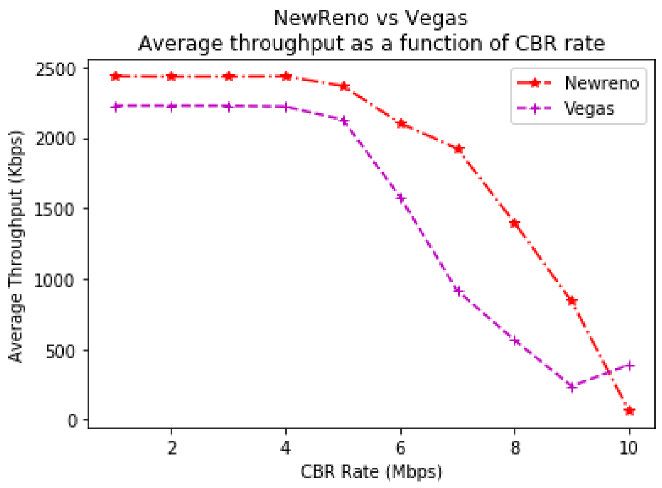

Fig 9: New Reno vs Vegas (Throughput)

On Figure 6, we now assign New Reno to TCP1 and Vegas to TCP2 link. TCP Vegas is more efficient and has a better performance. But, in combination with New Reno, its performance takes a hit. Figure 9 shows that New Reno is unfair to Vegas. This is because as the New Reno traffic in the network increases, TCP Vegas detects congestion and reduces the number of packets it sends into the network, thus giving the network bandwidth to the other flow. But as the CBR traffic in the network increases the throughput of New Reno reduces as the number of packet drops increases. But at the same time the throughput of Vegas improves as it is a congestion detection mechanism.

b) Latency:

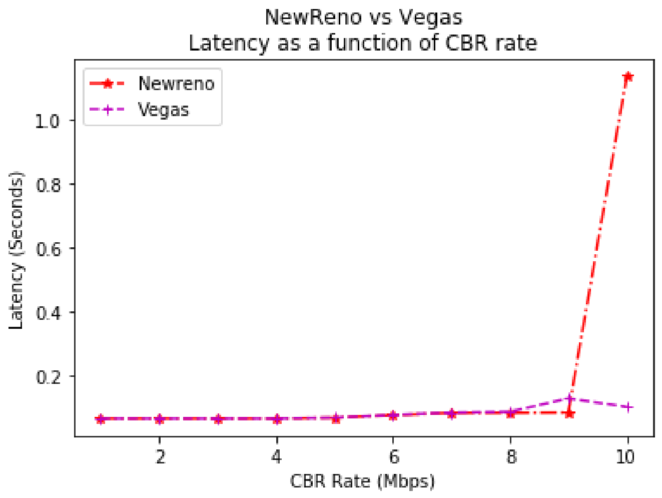

Fig10: New Reno vs Vegas (Latency)

New Reno stays in the fast recovery stage until it receives ACKs for all the packets that were present when it entered the slow start stage which results in high latency than Vegas. Hence the performance of Vegas is better than New Reno which is shown in figure10.

C. Experiment 3: Influence of Queuing:

The network topology and flow setup of experiment 3 are as Figure 11. This experiment is performed to analyze the influence of different queuing mechanisms like DropTail and Random Early Detection (RED).

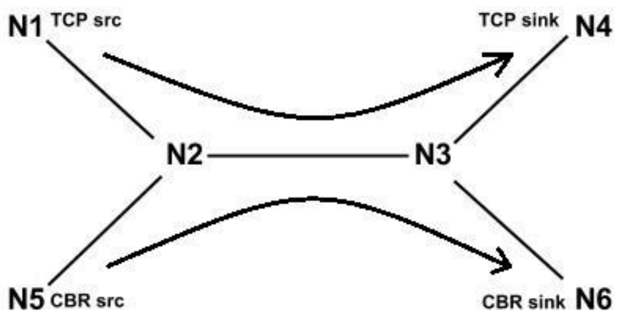

Fig11: Experiment 3 Network Topology

We recreate four sets of TCP variations and queue types: Reno with DropTail, Reno with RED, SACK with DropTail, and SACK with RED. The experiment procedure depends on the Throughput and Latency of TCP and CBR stream. Concerning the network topology from Figure 11 we assign one of the TCP variants (Reno or SACK) between N1 and N4 and CBR between N5 and N6. The CBR flow is started once the TCP flow is ready.

A) Throughput

Figure 12-13 shows the influence between two TCP queuing disciplines. The Throughputs of Reno and SACK with RED queuing algorithm are smaller compared to the Throughputs of Reno and SACK with DropTail queuing algorithm because, in RED, packets are dropped by the statistical algorithm. Whereas in DropTail packets are dropped independently when queue gets full irrespective of flow type.

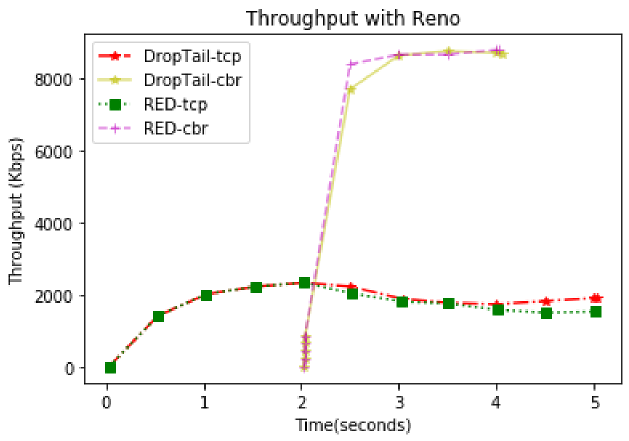

Fig12: Throughput with Reno

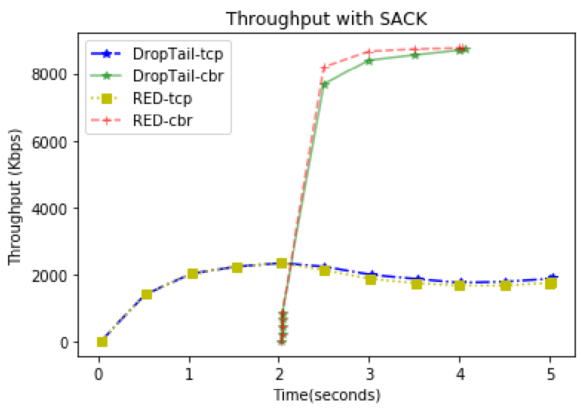

Fig13: Throughput with SACK

B) *Latency*

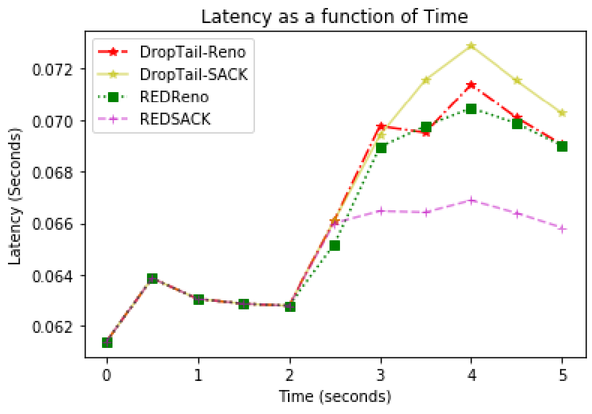

Fig14: Latency (Reno and SACK)

From the figure 14, we know that Reno and SACK with RED queuing algorithm have the lowest latency when compared with the Reno and SACK with DropTail queuing algorithm because RED will try to reduce the congestion better than DropTail hence there will be less traffic making RTT more optimized.

1) Does each queuing discipline provide fair bandwidth to each flow?

No, DropTail does not provide fair bandwidth. Because of its algorithm when the queue is full it will simply drops the incoming packets which will prevent the fair utilization of the bandwidth.

RED provides fair bandwidth compare to DropTail. Initially TCP utilizes maximum bandwidth but when CBR is initialized, the TCP throughput reduces over a course of time and CBR utilizes the predefined bandwidth.

2) How does the end-to-end latency for the flows differ between DropTail and RED?

The number of packets dropped in DropTail is more than that of RED. This is because in RED queuing algorithm packets are dropped based on the statistical algorithm. Whereas in DropTail packets are dropped independently when queue gets full irrespective of flow type

Hence, the end-to-end latency for DropTail is higher compared to that of RED.

3) How does the TCP flow react to the creation of the CBR flow?

With the help of figure 12-13 it is evident that when the CBR flow starts the throughput of TCP reduces due to congestion in the network.

4) Is RED a good idea while dealing with SACK?

Yes, RED is a good idea while dealing with SACK because from figure 13 we know that, before the start of CBR, the throughput for SACK is excellent showing full utilization of bandwidth. RED also prevents bias against bursty traffic which makes it an asset while dealing with SACK.

IV CONCLUSION

This paper has explored the use of NS-2 to study and compare different TCP variants by analyzing the throughput, latency, and packet drop rate within a congested network to scale the performance, fairness, and influence of queuing.. From the above conducted experiments, we can conclude:

In Experiment 1, depending upon the simulation analysis, we can conclude that TCP Vegas performs the best in terms of throughput, latency and packet drop rate for most of the times, especially under high congestion conditions in the network.

In Experiment 2, we can conclude from the simulation analysis that when two flows use the same TCP variant they are fair to each other. But when two different TCP flows are running at once, they are not fair to each other in terms of bandwidth utilization.

In Experiment 3, we observed from the simulation analysis that the TCP variants perform better in terms of throughput when DropTail queuing algorithm is used but gives small latency periods when RED queuing algorithm is used. Thus use of DropTail and RED is a trade-off between throughput and latency.

V REFERENCS

1) http://ieeexplore.ieee.org/document/7044985/

2) https://www.omicsonline.org/open-access/scenario-based-performance-analysis-of-variants-of-tcp-using-nssimulator-1-0976-4860-1-223-233.pdf

3) http://tldp.org/HOWTO/Adv-Routing-HOWTO/lartc.adv-qdisc.red.html

,