PERFORMANCE ANALYSIS OF TCP VARIANTS

TEJASWINI ESWAR
Northeastern University, MA, 02115
Email: eswar.t@husky.neu.edu
NUID: 001280199

ANUP SHANDILYA
Northeastern University, MA, 02115
Email: Shandilya.a@husky.neu.edu
NUID: 001256309

Abstract: ***TCP is a reliable transport protocol from an end to end point system perspective even under congestion. This protocol is utilized by major internet applications. In this paper we have used NS-2 simulator for simulation of different TCP variants on a network topology and analyze their variance based on throughput, packet drop rate and latency. We compare different TCP protocols like Tahoe, Reno, Newreno and Vegas to test their performance under congestion and determine the fairness between them. We studied the influence of queuing disciplines like DropTail and Random Early Drop(RED) on TCP Reno and SACK.***

I. INTRODUCTION

The main reason for conducting this experiment is to understand the behavior of different TCP variants under multiple conditions. The traditional TCP model was implemented for wired networks. As the internet evolved into wireless networking, the TCP protocol needed adaptions. So, different variants like Tahoe, Reno, Newreno, Vegas and Sack came into picture. Conducting this experiment, will allow us to understand the behavior of various TCP variants under different conditions and understand their advantages and disadvantages. We conduct the experiment to determine which TCP variant is best when compared to others. From the results, TCP Vegas show the highest degree of fairness, as it can adapt to the changing bandwidth very well and is robust against fluctuations when compared with Tahoe, Reno, Newreno and Sack.

1 TCP Tahoe: This is the first variant which includes the first congestion control mechanism. It was suggested by Van Jacobson. Tahoe detects congestion by packet loss which is based on timeouts. When congestion occurs we reduce congestion window to one and start over again. Tahoe cannot differentiate a duplicate acknowledgment from timeouts.
2 TCP Reno: When a triple duplicate acknowledgement is received congestion window is halved and enters the fast recovery mode by performing fast re-transmits.
3 TCP Newreno: Newreno is a slight modification over TCP-Reno. It is able to detect multiple packet losses. TCP Newreno requires all outstanding packets to be acknowledged for TCP to transit from fast recovery mode to congestion avoidance state.
4 TCP Vegas: Vegas does not depend solely on packet loss for congestion detection. It detects congestion even before packet loss occurs. Vegas calculates the RTT of the received packets and compares it with the previously received acknowledgements. After analyzing the current RTT value, it detects congestion and resizes the queue itself.
5 TCP SACK: This variant allows data to be acknowledged in any order and allows only the missing data to be retransmitted.

II. METHODOLGY

We use NS-2 simulator for designing the network topology. It is an event based simulator, used for analyzing the behavior of network protocol by extracting the packet information from the trace file which was generated. NS supports an array of popular network protocols offering simulation results for wired and wireless networks. NS-2 separates control path implementations from data path implementations. The event scheduler and the basic network component objects in the data path are coded in C++ language, to reduce event processing time. Also, it is very cost effective as it is very costly to deploy a network containing routers, switches and data links for a network. So, it saves time and cost by simulating the network. So, we prefer NS-2 simulator for our experiment. We have used almost similar topology for all 3 experiments.

The general topology is as show below.

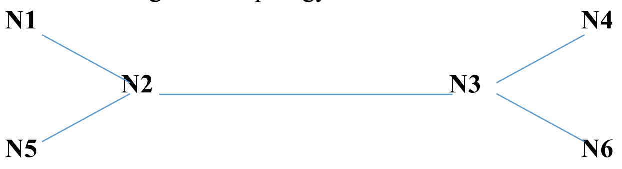

Fig 1.

N1, N2, N3, N4, N5, N6 are different nodes used in the experiment for establishing our network topology. It is interconnected with full duplex links which has a bandwidth of 10Mbps and delay of 10ms.

We have conducted 3 different experiments for understanding the TCP variants like Tahoe, Reno, Newreno, Vegas and SACK. We then generated a trace file for different conditions which are used to analyze the behavior of different variants.

Experiment 1: TCP Performance Under Congestion

In this experiment we analyze the throughput, latency and packet drop rate of different TCP variants with respect to varying CBR and analyzing the individual performance.
Throughput: It is the amount of network data that is transferred successfully from one end point to another in a given time frame.

Latency: It is defined as the amount of time it takes for a packet to cross from source point to destination point and its successful reception of ACK back to the source. It was calculated to understand which TCP variant had least and highest latency.
Packet Drop Rate: It is the rate at which the packet is dropped when sent from source to destination with respect to the total no of packets sent. It is calculated to find the variant with least drop rate.
The network topology is as shown in Fig 1. A TCP flow is set from N1 to N4 with a linearly varying CBR rate which varies from 1Mbps to 10Mbps between N2 and N3 nodes.

Experiment 2: Fairness between TCP Variants

This experiment determines the fairness between two variants by linearly varying CBR rate. The network topology is as shown in Fig 1. Two TCP streams from N1 to N4 and N5 to N6 is added to the network. Also, linearly varying CBR rate from 1 to 10 is introduced between N2(source) to N3(sink). The fairness is determined by plotting a graph for each pair of TCP variant flows v/s CBR.

Experiment 3: Influence of Queuing

This experiment is to understand how queuing disciplines like DropTail and RED algorithms control the packets that are in a queue for TCP Reno and SACK variants. The network topology is as shown in Fig 1. A TCP flow is set from N1 to N4 with a constant CBR rate which is set at 5Mbps between N5 and N6 nodes.

III. ANALYSIS

EXPERIMENT 1: TCP PERFORMANCE UNDER CONGESTION
Topology:

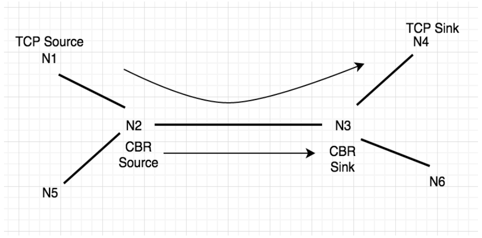

Fig 2.

The TCP flow will be one of the TCP variants like Tahoe, Reno, Newreno or Vegas. It is introduced between N1 and N4. A linearly varying CBR rate from 0 Mbps to 10 Mbps with a gradual increase in 0.1 Mbps is introduced between N2 and N3. A trace file is generated for each value of CBR and the Throughput, Latency and Packet drop rate is extracted by analyzing the trace file. The performance of TCP decreases as the CBR rate increases as TCP is a reliable protocol and will change the window only if the acknowledgements of the previous packets have been received. As CBR is an unresponsive UDP flow, it does not have any acknowledgment concept thus it keeps sending the packets into the network, and finally creating congestion. So, as the congestion in the network increases the throughput will decrease which is verified in this experiment for different TCP variants. The experiment was conducted by varying the start time of TCP flows and packet size of CBR and TCP.
These 3 graphs show throughput, latency and Packet drop rate of TCP variants with respect to CBR. The trace file generates the network data in each line. We can parse the trace file based on the sequence number for each packet and calculate the throughput of the TCP flow for each variant for a particular time of simulation.

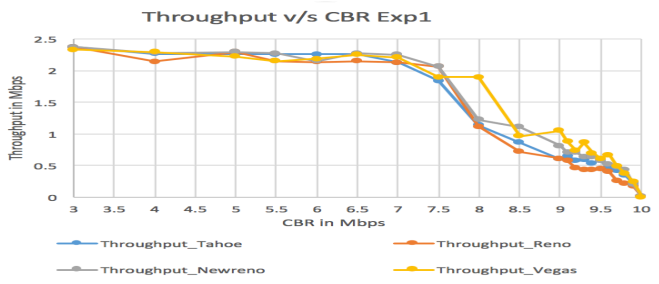

Fig 2.1 Throughput v/s CBR

For CBR value less that 7.5 Mbps the throughput of all the variants is almost similar as seen in Fig 2.1. Once the CBR value increases the network gets congested and Vegas starts performing better among the 4 variants as it can detect congestion even before packet loss occurs and Reno has the least throughput. Vegas detects congestion at a very early stage based on increasing value of RTT and adjusts the amount of data transfer during the congestion.
We conducted the T-Test to determine which is the best variant that provides a higher throughput. It was conducted for all possible combination of TCP variants.
The t value was calculated using the formula:
$T = (\text{Mean1}-\text{Mean2})/\sqrt{(\text{var1}/x1 + \text{var2}/x2)}$
Where T is the t stat value and x1 and x2 are number of values for the respective group.

variants	Tahoe-Vegas	Reno-Vegas	Newreno-Vegas
t stat	-0.397255162	-0.61674004	-0.163518094

We calculated each variants t value with respect to Vegas. As the t stat value are negative, and Vegas has best mean and least value of variance and standard deviation for almost all experiment, we can conclude that Vegas provides better throughput than other three TCP variants.

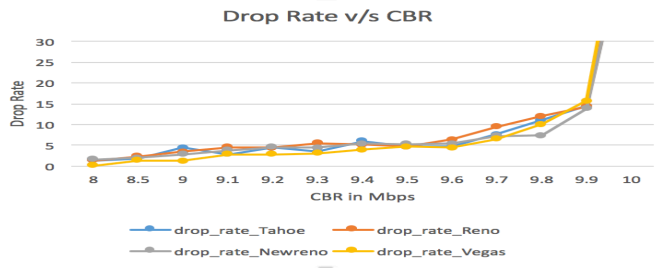

Fig 2.2 Drop Rate v/s CBR

We can find the packet drop rate from the trace file by parsing the data and counting the number of traces which has the event field equal to 'd' and if my packet type is TCP. The packet drop rate under varying CBR rate is as show in the Fig 2.2. Vegas drops few packets until its bottleneck is reached as it increases its window size and doubles the input from its previous value. Once it reaches the limit, it's window size decreases drastically and hence its drop rate shoot's up.

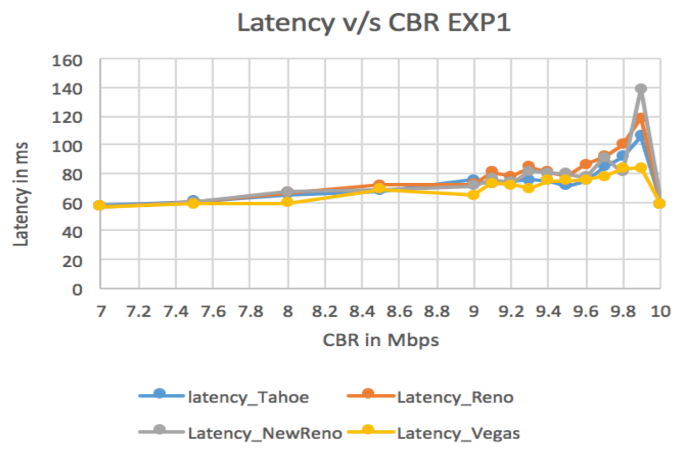

Fig 2.3 Latency v/s CBR

The average Latency of each variant was calculated based on the actual RTT of the packets. We parsed the trace file and calculated the sent and ACK received time of the packet to calculate the latency.
According to the graph in Fig 2.3 TCP Vegas has the least latency as it detects congestion even before packet loss occurs. Vegas calculates the RTT of the received packets and compares it with the previously received acknowledgements. After analyzing the current RTT value, it detects congestion and resizes the queue itself.

So based on our experiments in this network topology, TCP Vegas performs better than other three variants as it has higher throughput, least drop rate and least latency.

EXPERIMENT 2: FAIRNESS BETWEEN TCP VARIANTS

Topology:

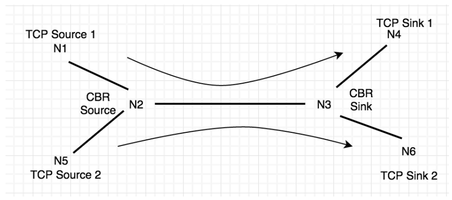

Fig 3.

In this experiment we have introduced two TCP flows from N1 to N4 and N5 to N6. Also, we have introduced a linearly varying CBR from 0 Mbps to 10 Mbps with a gradual increase in 0.1 Mbps between N2 and N3.

I. **Reno/Reno**:

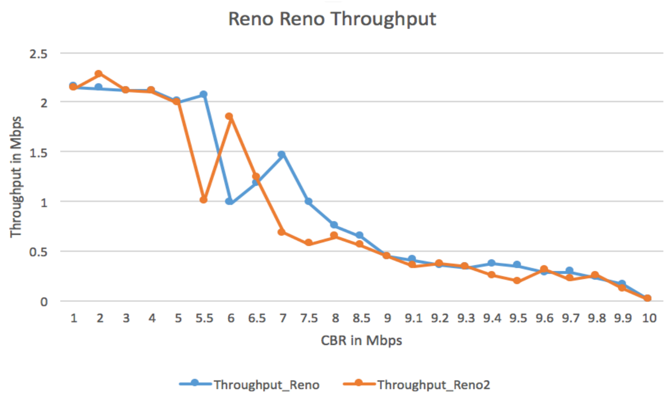

Fig 3.1.1 Reno/Reno Throughput v/s CBR

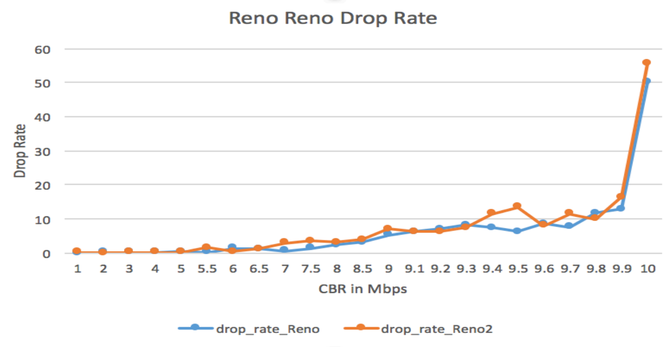

Fig 3.1.2 Reno/Reno Drop Rate v/s CBR

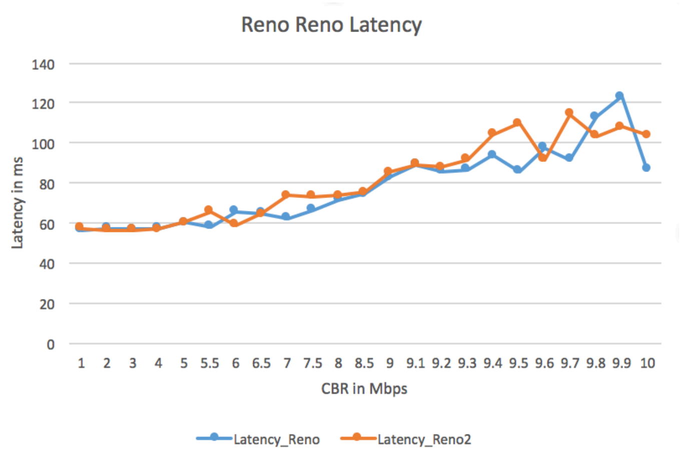

Fig 3.1.3 Reno/Reno Latency v/s CBR

Both TCP flows TCP1 and TCP2 are set as Reno. The Fig 3.1.1 3.1.2 3.1.3 show how fair they are to each other in perspective of throughput, packet drop rate and latency respectively when introduced in the same network.

According to the graphs we can see that, the two TCP Reno flows utilizes the bandwidth alternatively. This is true because we have introduced a constant CBR flow and when one TCP flow has higher throughput, the other flow automatically suppresses because of channel bandwidth capacity. So, we can conclude that both the TCP flows are fair to each other. Thus we can conclude that two Reno flows are fair to each other.

II. **Newreno/Reno**:

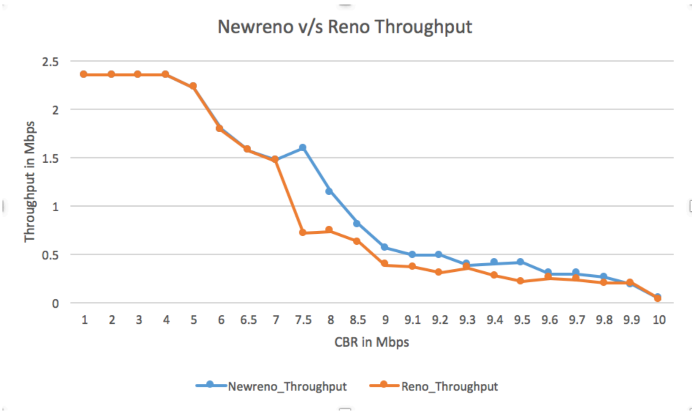

Fig 3.2.1 Newreno/Reno Throughput v/s CBR.

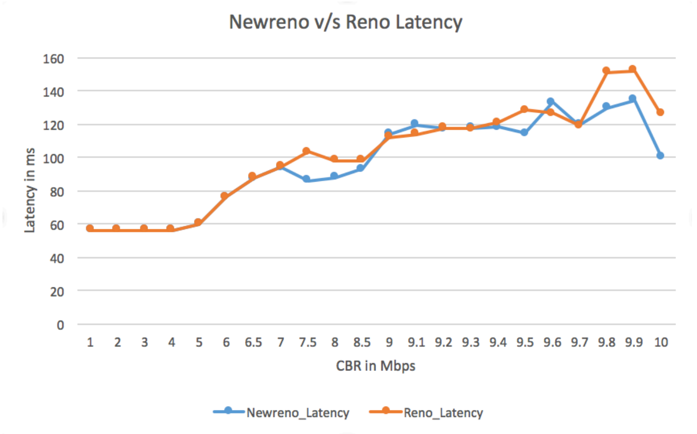

Fig 3.2.2 Newreno/Reno Latency v/s CBR

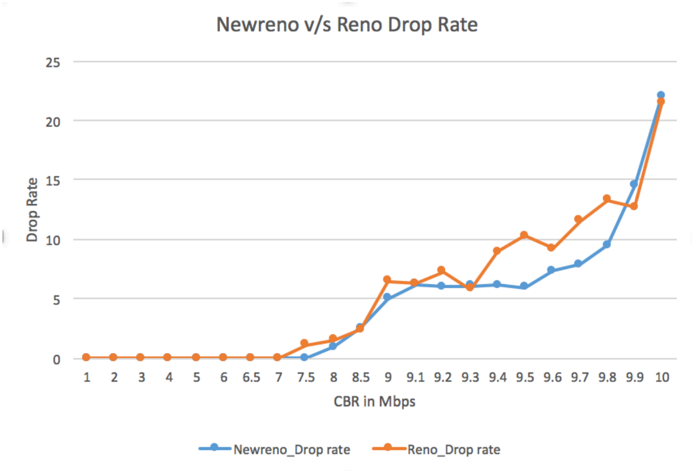

Fig 3.2.3 Newreno/Reno Drop Rate v/s CBR

Now we assign TCP1 to Newreno and TCP2 to Reno. The Fig 3.2.1, 3.2.2 and 3.2.3 shows how fair they are to each other in perspective of throughput, latency and packet drop rate when introduced in the same network.

According to the graphs, we can see that Newreno and Reno also utilize the bandwidth alternatively. This is true because we have introduced a constant CBR flow and when one TCP flow has higher throughput, the other flow automatically suppresses because of channel bandwidth capacity. So, we can conclude that both the TCP Newreno and Reno are fair to each other.

III. **Vegas/Vegas**:

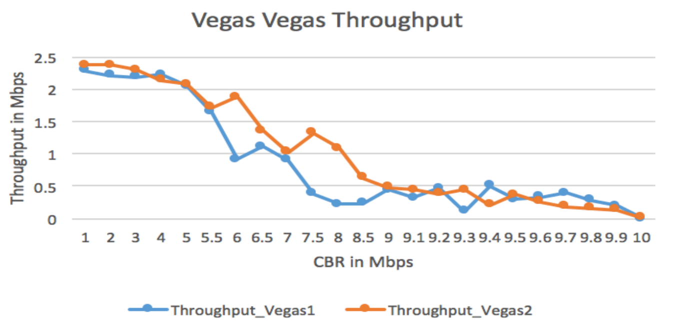

Fig 3.3.1 Vegas/Vegas Throughput v/s CBR

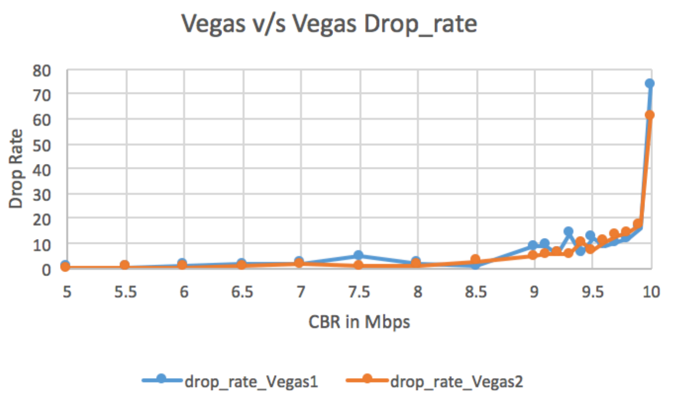

Fig 3.3.2 Vegas/Vegas Drop Rate v/s CBR

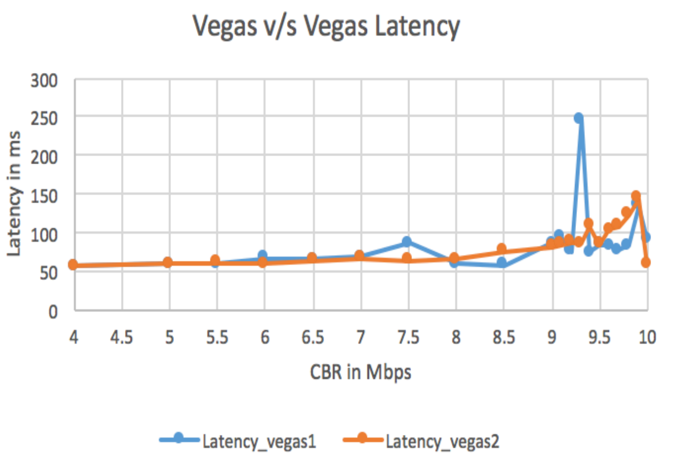

Fig 3.3.3 Vegas/Vegas Latency v/s CBR

Both TCP links TCP1 and TCP2 are set as Vegas. The Fig 3.3.1, 3.3.2 and 3.3.3 show how fair they are to each other in perspective of throughput, latency and packet drop rate when introduced in the same network.

As with the earlier cases the two TCP flows utilizes the bandwidth alternatively. So, Vegas and Vegas are fair to each other.

IV. Newreno/Vegas:

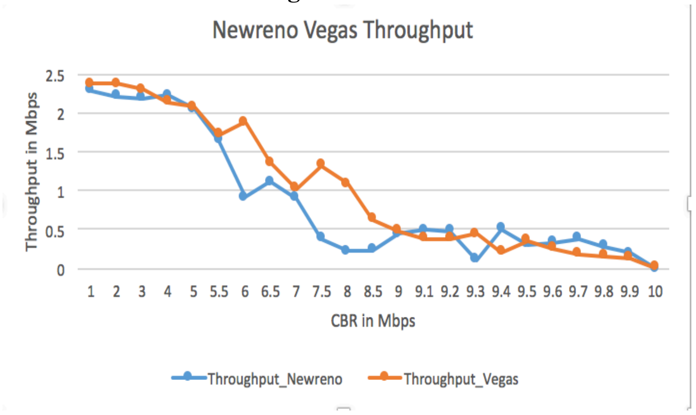

Fig 3.4.1 Newreno/Vegas Throughput v/s CBR

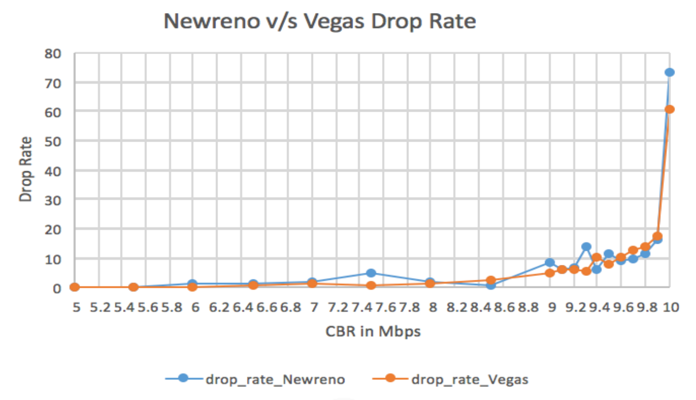

Fig 3.4.2 Newreno/Vegas Drop Rate v/s CBR

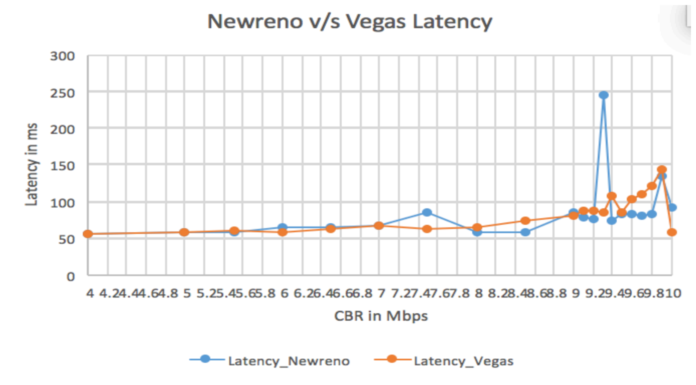

Fig 3.4.3 Newreno/Vegas Latency v/s CBR

Now we assign TCP1 to Newreno and TCP2 to Vegas. The graphs in Fig 3.4.1, 3.4.2 and 3.4.3 show the Throughput, Drop Rate and Latency respectively for this pair of flow.
With the increase in CBR value, the throughput of the Newreno becomes higher than Vegas. The latency of Newreno shoot's up when CBR rate is at 9.2Mbps, otherwise they are almost similar to each other.

From the above experiment, it is good to postulate that TCP variants of similar kind are usually fair to each other. However, different variants of TCP are unfair to each other. But, Newreno and Vegas variants are not fair to each other. As the Vegas detects congestion even before packet drop occurs, it gets suppressed by Newreno. And once Newreno becomes dominant, Vegas will always consider that there is congestion in the network and send lesser data into the network and stays predominant between the two variants.

EXPERIMENT 3: INFLUENCE OF QUEUING

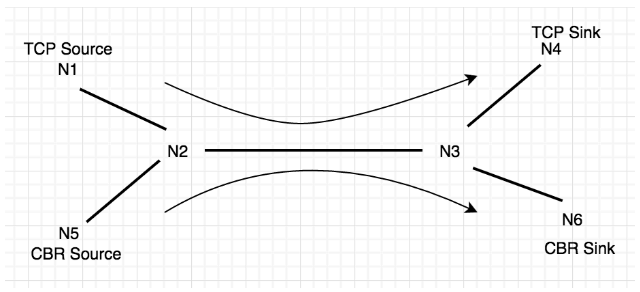

Fig 4

In this experiment we have introduced a TCP flow from N1 to N4 and a constant CBR between N5 and N6. We have tested the experiment under 2 different cases.

1- We have started TCP and CBR at the same time.
2- We have checked how TCP flow occurs when CBR is not introduced.

We have checked the TCP flow (Reno &SACK) for two different queuing disciplines DropTail and RED. The total duration of the experiment is 30 seconds. We introduce the CBR flow at 5th second and stop CBR flow at 25th seconds. We start the TCP flow from 0th second and stop it at 30th second. The bandwidth of every link is 10Mbps and delay of 10ms. TCP window size is 120, max window is 150, and CBR stream rate is 5Mbps. We calculate the average bandwidth and latency for the TCP streams.

I Average Throughput:

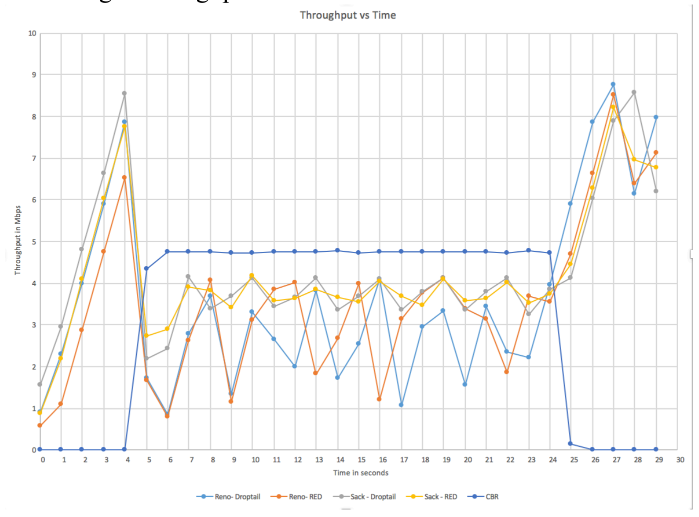

Fig 4.1 Throughput v/s Time

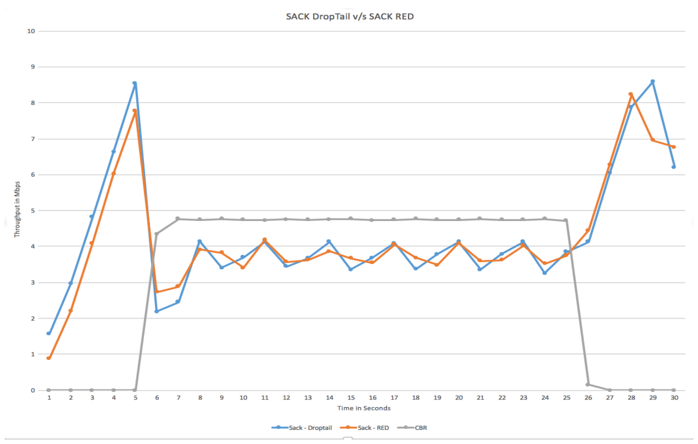

Fig 4.2 Throughput TCP SACK v/s Time

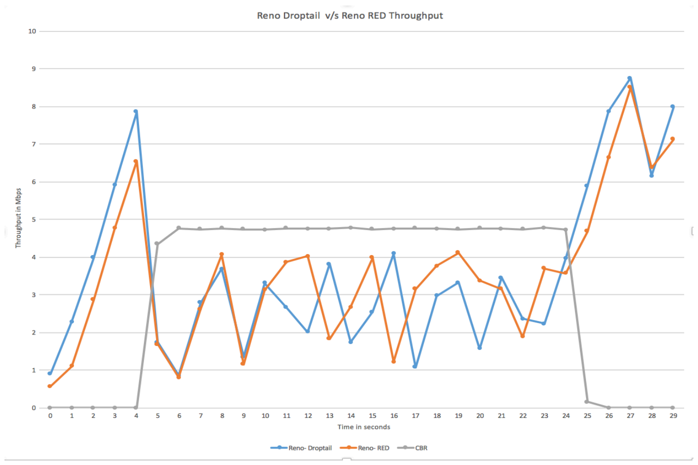

Fig 4.3 Throughput TCP Reno v/s Time

DropTail is a simple active queuing management algorithm, which does not differentiate traffic from different source. As soon as the queue fills up, it will drop the subsequent packets arrived. RED monitors the average queue size and takes actions on the incoming packets based on statistical data.
After CBR flow is introduced at the 5th second, RED TCP SACK flow will have a higher throughput than the DropTail. So, RED has a higher throughput performance as compared to DropTail for TCP SACK. In case of RED TCP Reno flow has a higher throughput than DropTail. So, RED has a higher throughput performance as compared to DropTail for SACK and RENO due to its ability to gauze the congestion based on the existing packets in the queue and action accordingly. For TCP SACK, RED and DropTail are fair to each other because SACK accepts acknowledgements in any order and retransmits only the missing data. But, in case of TCP Reno, RED provides fair bandwidth as Reno accepts ACK in order and RED sets the queue size based on statistical data analysis, thus providing better bandwidth and is unfair to DropTail.

II Average Latency:
After CBR flow is introduced at the 5th second, the latency of all the TCP variants shoots up. When DropTail algorithm is used the latency is greater when compared to RED. This is because, RED defines a limit over how bursty the traffic is to be allowed in the queue unlike DropTail.

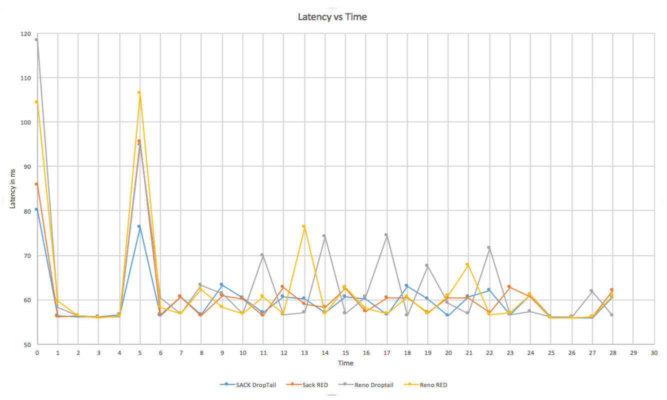

Fig 4.4 Latency v/s Time

From the graph in Fig 4.1 we can see that the throughput is very high when there is no CBR flow in the network till 5sec. The CBR flow does not take congestion into consideration, and hence sends the packet into the network. When there is a TCP flow along with CBR, then TCP flow backs off and reduces its window size. So, TCP reduces its sending rate with the occurrence of congestion. Also, TCP SACK works well with RED and TCP Reno should not be used with DropTail according to our results.

IV. CONCLUSION

This paper has introduced us to the network simulation using NS-2 simulator for a simple network topology. We analyzed the TCP variant by analyzing the throughput, latency and packet drop rate when congestion is introduced in the network. These results cannot be justified as a full-fledged result to determine the best TCP variant, as the network topology used for this experiment is very simple and, a complex topology might result in a different conclusion. The result of this experiment is as follows:

1. TCP Vegas is the best variant among the different TCP variants.
2. TCP variants of similar kind are usually fair to each other. And variants of different kinds are generally unfair to each other.
3. When DropTail algorithm is used the latency is greater when compared to RED, but in case of throughput RED provides higher bandwidth for TCP SACK and RENO.

REFERENCES:
1] http://www.isi.edu/nsnam/ns/ns-documentation.html
2] Evaluation of Different TCP Congestion Control Algorithm using NS-2, Hui Zhang & Zhengbing Bian
3] A Comparative Study of Different TCP Variants in Network, Balveer Singh
4] Wikipedia, Congestion-avoidance algorithm.
5] Simulation based Comparison of Tahoe, Reno and SACK TCP, Kevin Fall & Sally Floyd.