Performance Analysis of TCP Variants

Nivedita Mittal
College of Computer and Information Science,
Northeastern University,
Boston, MA
NUID: 001686609
nivi@ccs.neu.edu

Janhavi Mahajan
College of Computer and Information Science,
Northeastern University,
Boston, MA
NUID: 001686377
janhavi@ccs.neu.edu

Abstract—The purpose of this paper is to analyze the performance of different TCP variants (Tahoe, Vegas, Reno, New Reno). The performance of these TCP variants is observed on the basis of Throughput, Drop Count Rate and Latency.

I. Introduction

TCP is a reliable connection-oriented protocol and it has in-built mechanisms to handle congestion. It ensures reliability since the receiver returns an acknowledgement whenever a packet is received. However, packets are often lost due to network failures or excessive congestion issues. In this paper, we spend time analyzing the variants and understanding how they act differently during congestion. Parameters like Throughput, Drop Count and Latency are used to compare and contrast the same, and we try to answer the question "Is there an overall" best" TCP variant in this experiment, or does the" best" variant vary depending on other circumstances?". The experiments are explained briefly below:

Experiment-1:

In this Experiment, we study the TCP variants in different load conditions. A CBR flow is established along with a single TCP stream, and the performance of the TCP protocol is analyzed when congestion occurs. Then we compare the variants by plotting graphs for Throughput, Drop Count and Latency and answer various performance-oriented questions.

Experiment-2:

In this Experiment, we analyze the fairness of the TCP variants. Along with the CBR flow, we establish 2 more TCP flows. Then we use various pairs of TCP variants and compare their performances.

Experiment-3:

In this Experiment, we vary the queueing discipline for every variant. We used queueing disciplines like Drop Tail and Random Early Drop (RED). We have one TCP flow and one CBR flow. Instead of varying the rate of CBR flow, we will study the queueing disciplines used in the flow.

II. Methodology

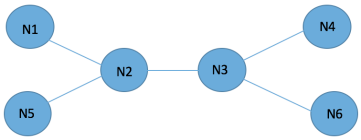

Figure 1: Network Topology

A. Network Topology

The above diagram is the network topology we applied for all the three experiments.

Our analysis of the three experiments is based on throughput, latency, drop count. We choose a queue size of 10 between N2 and N3 to handle the congestion between the two nodes.

These parameters were selected simply because they help analyze the performance of every TCP/UDP protocol. Throughput and Latency are a direct measurement of performance of the network, while drop count rate is used to capture data of dropped or missing packets. Using these three parameters, we can sufficiently estimate and compare the TCP variants without much space for error.

B. Tools Used:

We used the following tools for our experiments:

Network Simulator 2: NS is a discrete event simulator targeted at networking research. NS provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks. [1]

Shell Script:
It is a computer program designed to be run by the Unix shell, a command line interpreter.

Microsoft Excel: To plot graphs for the values obtained from the simulation

Awk Scripting: AWK is an interpreted programming language designed for text processing and typically used as a data extraction and reporting tool.

TCL files: TCL (Tool Command Language) is a scripting language commonly used for rapid scripted applications, GUIs and testing. [2]

III. Experiment 1: TCP Performance Under Congestion

In this Experiment, we analyze the performance of the four TCP variants (Tahoe, Reno, New Reno and Vegas) by varying the load conditions and observing their behavior in the congestion period.

We set up a TCP stream between N1 to N4 in the topology above, and a CBR flow between N2 and N3. Then we analyze the throughput, drop count rate and latency of the TCP variant being used, by changing the CBR bandwidth between 1 Mbps to 10 Mbps.

10 Mbps is the bandwidth at which complete bottleneck is achieved. Between this range, we observe the perfect sample size to correctly analyze our TCP variants. Later, we plot these values graphically against one another and observe the characteristics of the variants.

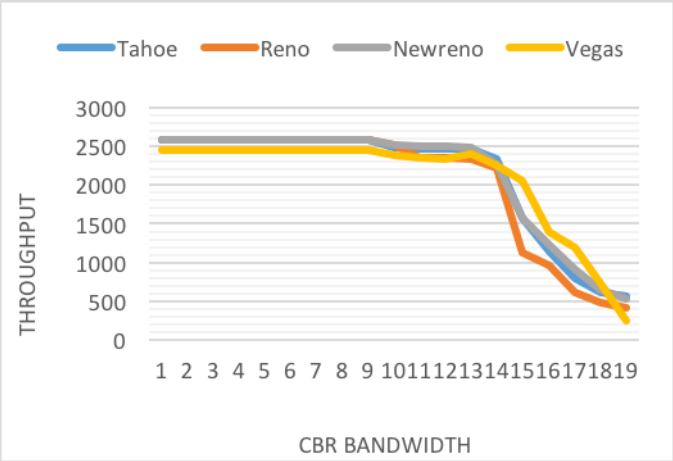

Figure 2: Throughput of various TCP variants (X-Axis: CBR Bandwidth between 1-10 in increments of 0.5 Mbps) (Y-Axis: Throughput - bits/sec)

Throughput:

Figure 2 is a plotted graph of the TCP variants throughput against the varying CBR rate. The CBR is varied between 1 Mbps and 10 Mbps. This window contains the congestion period between N2 and N3. We run the entire simulation for 61 seconds and start and end both the flows almost simultaneously. We can see from the graph that New Reno and Tahoe have an identical graph from the beginning of the simulation. It is clear that while Reno performs average from the beginning, it is unable to handle packet loss. At the CBR rate of 7 Mbps, the congestion period starts and that is when the throughput of Vegas spikes and all other variants drop. Vegas performs better than Reno/New Reno because its estimation of congestion is not dependent on packet loss.

It also introduces congestion avoidance based on the delay to further correctly estimate the bandwidth available. Reno and New Reno both have a similar graph. They both have a fast start and fast recovery. However, unlike Reno, New Reno begins retransmission on the receipt of first failure acknowledgement and does not wait for all the failure acknowledgements to arrive. Tahoe has an average throughput, since on loss of packets, it reduces its window size to 1 during slow start. Since, in the congestion window of 7-9 Mbps of CBR, minimum packets are lost, Vegas has a high throughput value in this window.

Therefore, on observing the congestion window period we conclude that Vegas has the highest average throughput, since its congestion avoidance algorithm causes the least retransmissions.

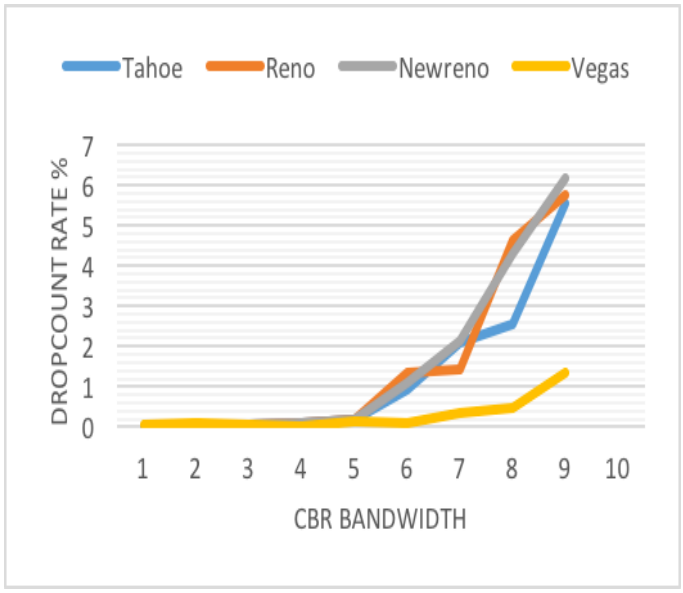

Figure 3: Drop Count Rate of various TCP variants (X-Axis: CBR Bandwidth between 1-10 in 1 Mbps increments) (Y-Axis: Percentage Drop Count Rate)

Drop Count Rate:

From figure 3, it is evident that Vegas has the lowest Drop Count Rate between all variants.

Vegas is able to detect collisions early due to Round Trip time. As the amount of traffic increases, the time for acknowledgement increases dramatically which Vegas takes as a signal, and reduces its window size. As the Round trip time reduces, Vegas detects that congestions is reducing and it increases its window size. On the other hand, other variants on detecting congestion assert their congestion avoidance algorithm. This increases their drop rate as visible in the graph.

Therefore, we infer since Vegas is able to deal with congestion accurately using RTT, it has the least dropped packets in the simulation.

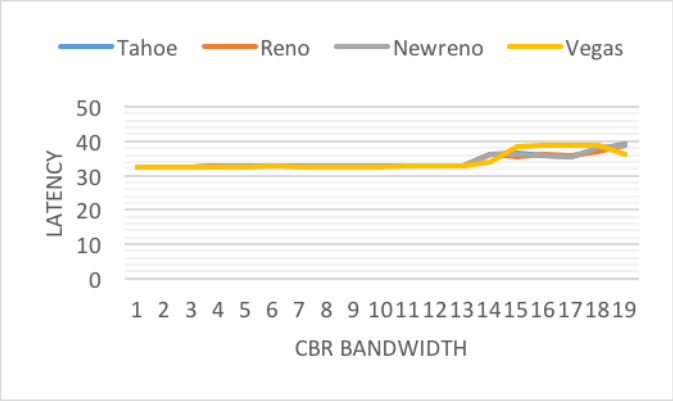

Figure 4: Latency of various TCP variants (X-Axis: CBR Bandwidth between 1-10 in increments of 0.5 Mbps) (Y-Axis: Latency - seconds)

Latency:

For accurate measure of latency, we calculate the one-pass time for every packet. Based on the total number of packets sent we calculate the average latency and compare it for all variants. Since Vegas deals with congestion by using RTT, and is successful at correctly detecting whether congestion is easing or not, it has the lowest latency which is evident from figure 4.

Incase of other variants, when congestion is detected, the window size is reduced, packet drops occur and the average latency increases per packet.

Best TCP Variant?

Based on our experiments, where the simulation is run for 61 seconds for the specific topology, Vegas is the best variant. The main reasons for the same are highlighted below:

- It is a variant of TCP Reno, which follows a proactive approach to deal with congestion rather than reactive.
- It slow start algorithm prevents excessive congestion of the network by excessive retransmissions.
- It does not depend on packet loss as a form of congestion. Using the RTT (Round Trip Time) it pre-empts that congestion may be occurring and starts congestion avoidance.

IV. Experiment 2: Fairness Between TCP Variants

In this experiment, we analyze the fairness between the TCP variants by plotting throughput, packet loss rate and latency of each TCP flow as a function of bandwidth used by the CBR flow. We run the simulation for 61 seconds and set a TCP flow from N1 to N4 and second TCP flow from N5 to N6.

A. NewReno Reno

In New Reno Reno combination, we infer that New Reno is being unfair to Reno. Because New Reno is a slight modification over TCP-Reno, it is able to detect multiple packet losses and thus is much more efficient that Reno in this event. [4]

In this simulation, the average throughput of New Reno is larger than that of Reno by 60 bits/seconds. Also average latency of New Reno is lower than Reno's latency by 6 seconds, and its drop count is also lower.

Thus it is clear from our observation that New Reno is in-fact unfair to Reno. (See Figure 5 and 6)

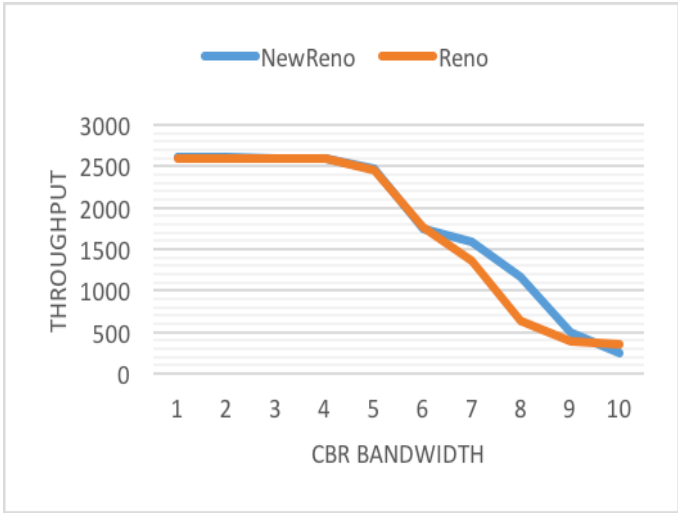

Figure 5: Throughput of New Reno / Reno (X-Axis: CBR Bandwidth between 1-10 in 1 Mbps increments) (Y-Axis: Throughput - bits/sec)

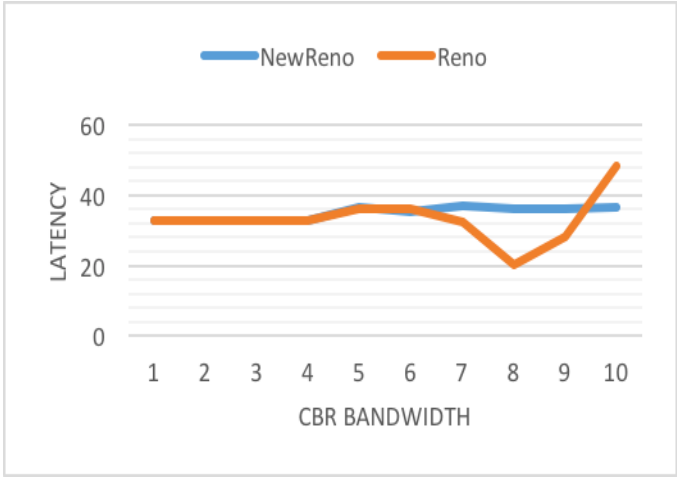

Figure 6: Latency of New Reno/Reno (X-Axis: CBR Bandwidth between 1-10 in 1 Mbps increments) (Y-Axis: Latency - seconds)

B. NewReno Vegas

Like Reno, New-Reno also enters into fast-retransmit when it receives multiple duplicate packets. However, it differs from Reno in the sense that it doesn't exit fast-recovery until all the data, which is out standing at the time it entered fast recovery is acknowledged. Due to this behavior of New Reno, it is unfair to Vegas, as TCP New Reno consumes more bandwidth. Vegas extends on the re-transmission mechanism of Reno. It keeps track of when each segment was sent and it also calculates an estimate of the RTT.
Thus in this scenario, it is observed that New Reno is unfair to Vegas. (See Figure 7)

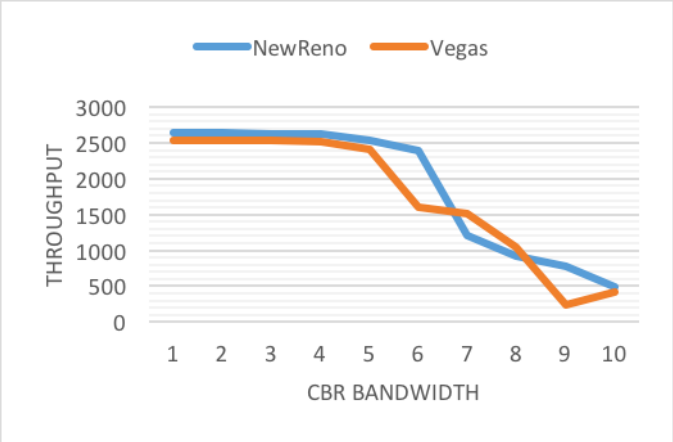

Figure 7: Throughput of New Reno/Vegas (X-Axis: CBR Bandwidth between 1-10 in 1 Mbps increments) (Y-Axis: Throughput - bits/sec)

C. Reno Reno

In Reno Reno combination, we infer that TCP flows are fair to each other. The average throughput, average latency and the drop count for both the flows are the same. This is so because, as they are the same variant, and equally aggressive, they remain fair to one another. (See Figure 8)

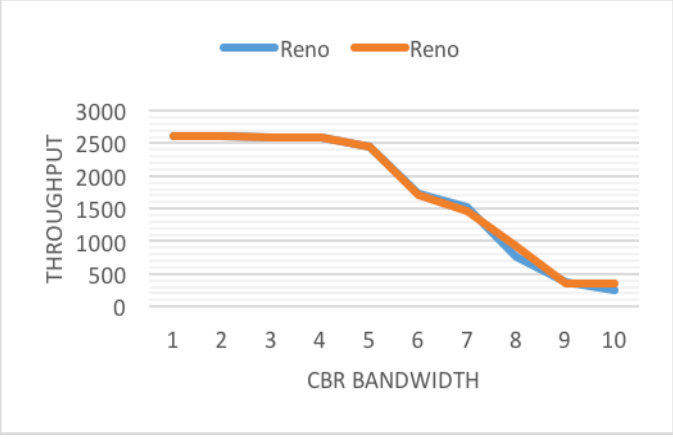

Figure 8: Throughput of Reno Reno (X-Axis: CBR Bandwidth between 1-10 in 1 Mbps increments) (Y-Axis: Throughput - bits/sec)

D. Vegas Vegas

In the Vegas Vegas combination, we infer that TCP flows are unfair to each other. In the congestion period (CBR rate 7 to 9), if the throughput for one TCP flow becomes high then the TCP flow for the other TCP flow becomes low. Their behavior is in contrast to the previous behavior of Reno/Reno because, Vegas as a variant is extremely aggressive, and does not share bandwidth with any other TCP flow. That being said, an interesting observation is that since both flows are Vegas, they alternate their hold of the bandwidth and are actually able to achieve similar average throughputs throughout the entire run. (See Figure 9)

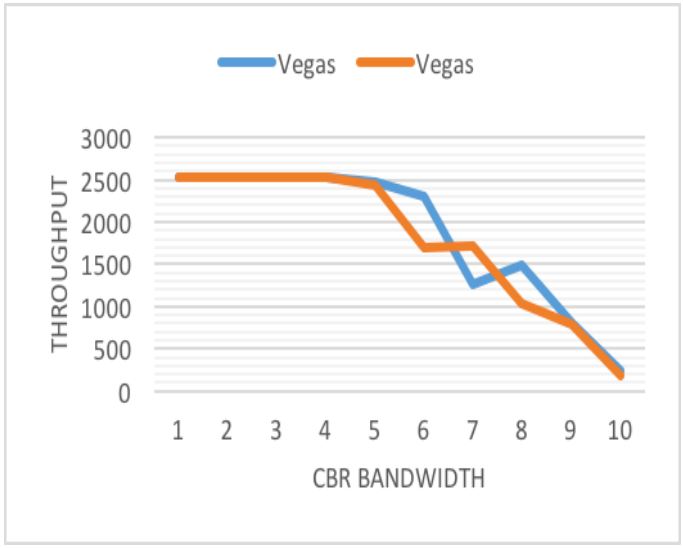

Figure 9: Throughput of Vegas Vegas (X-Axis: CBR Bandwidth between 1-10 in 1 Mbps increments) (Y-Axis: Throughput - bits/sec)

V. Experiment 3: Influence of Queuing

In this experiment, instead of varying the CBR bandwidth rate, we vary the queueing discipline for two TCP variants (Reno and SACK) and compare their respective results. This experiment consists of one TCP flow setup between N1 and N4 and one CBR flow between N5 and N6. The two queueing disciplines studied are Drop Tail and Random Early Drop (RED). The graphs are plotted for the throughput performance of TCP and CBR flow for each queueing discipline.

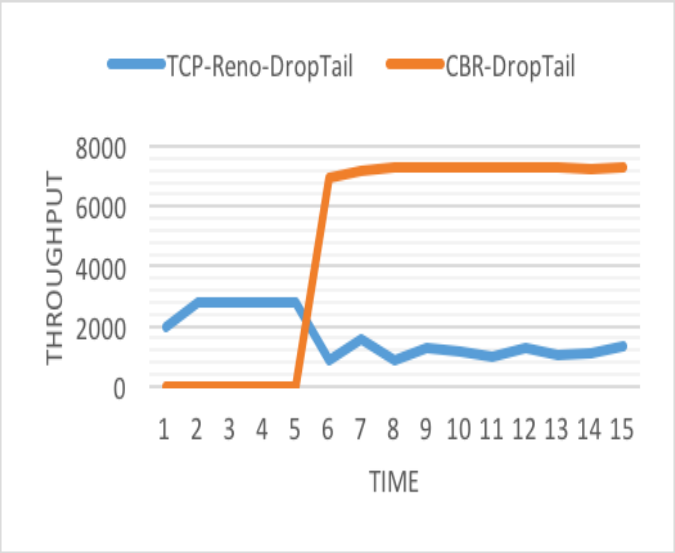

Figure 10: TCP Reno and CBR flow using Drop Tail
(X-Axis: Time-seconds)
(Y-Axis: Throughput - bits/sec)

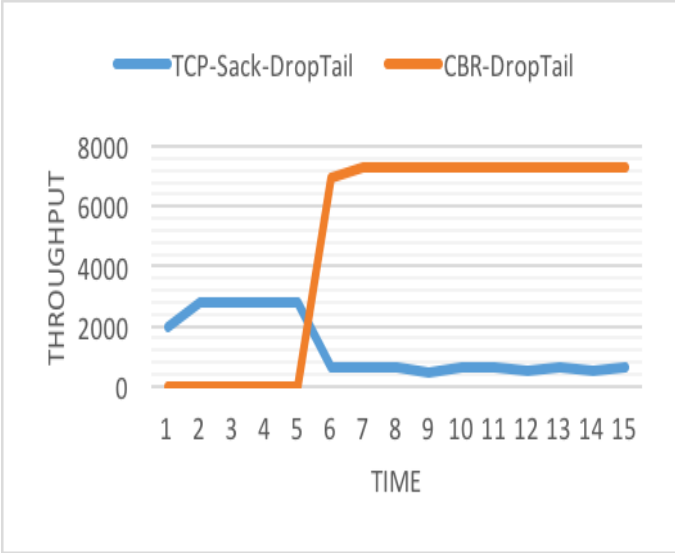

Figure 11: TCP Sack and CBR flow in Drop Tail
(X-Axis: Time-seconds)
(Y-Axis: Throughput - bits/sec)

Fairness between flows using Drop Tail/RED Queueing Technique

The Drop Tail queueing technique follows the FIFO method and when the queue is full, begins to drop packets till space is available for more packets in queue.

It is evident from the graph that CBR handles Drop Tail queueing mechanism better when it begins at time = 5. The reason for this is that since Drop Tail has burst losses, very often many packets from same flow are lost. This paralyses the TCP Reno fast retransmit mechanism and causes large fall in throughput. CBR on the other hand is a UDP based flow and is easily able to handle packet losses. Hence its throughput is high.

Since the congestion control algorithm for SACK is basically the same as implemented for TCP Reno, it also shares a similar drop in performance as TCP Reno. Similar results are seen when using RED queueing mechanism. RED is a different queueing mechanism than Drop Tail, in that it uses statistical probabilities to drop packets from its queue. This probability depends on the number of packets in the queue and the number of packets from a particular source.

Since TCP (Reno and SACK) is a reliable protocol, it retransmit packets when drop occurs. This increases the overall number of packets from same source in the buffer for RED, causing more drop packets for that source.

Based on the above analyses and observations from figure 11, it is evident that CBR in in fact unfair while using Drop Tail or RED queueing mechanism against TCP Reno or SACK.

Difference in End to End Latency for flows between Drop Tail and RED

End to End Latency is inversely proportional to Throughput. Using the analysis of the graphs we plotted for throughput, it is observed that both TCP Reno and SACK perform best with Drop Tail. This is because RED is pre-empting a congestion using statistical probability and asking the Node to not transmit any more packets. This leads to excessive drops just to prevent congestion, which increases end to end latency. (See Figure 12)

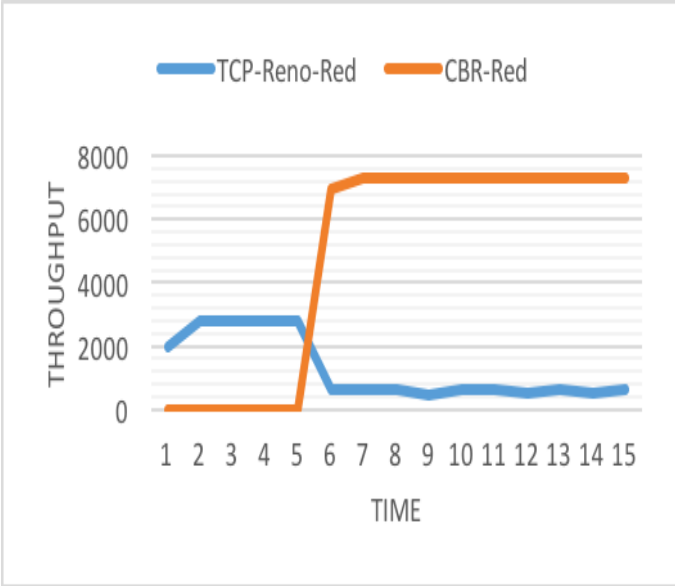

Figure 12: TCP Reno and CBR using RED
(X-Axis: Time-seconds)
(Y-Axis: Throughput - bits/sec)

TCP Flow reaction to CBR Flow

The TCP Flow begins first (in both cases) and CBR flow begins at time=5. When the CBR flow begins, it is evident from the graph that the TCP flow finds its difficult to gain bandwidth and loses its upper hand to CBR. The main reason for this is that CBR is a UDP based flow and is not reliable. Therefore, losing packets (short bursts or otherwise) does not decrease its overall performance. However, in case of TCP, excessive loss of packets hampers its retransmit mechanism, and causes large drop in performance. (See Figure 13)

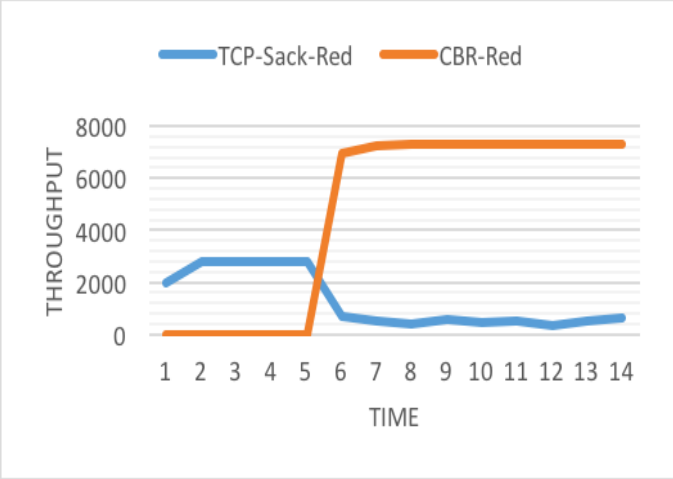

Figure 13: TCP Sack and CBR using RED
(X-Axis: Time-seconds) (Y-Axis: Throughput - bits/sec)

SACK using RED

When we compare the graphs for SACK using RED and SACK using Drop Tail, it is evident that performance for SACK is better when using Drop Tail. (See Figure 14)

VI. Conclusion

In this paper, we have analyzed the performance of TCP Tahoe, Reno, New Reno, Vegas variants based on the throughput, latency and drop count rate. This experiment is based on the wired network topology and we conclude the following:

- TCP Vegas has the highest throughput under congestion.
- New Reno is unfair with other TCP variant and Vegas is unfair to itself in the congestion period.
- Drop Tail queuing gives better throughput for TCP Reno over SACK as compared to RED queuing in a network with one TCP and a CBR flow.

Our performance analysis can be a basis for choosing a TCP variant in the wired network. This also gives a direction to perform analysis of the same variants in different type of networks like wireless network etc. Performance analysis can be done more accurately with parameters used in the above paper along with other network parameters such as Jitter, Error rate etc.

References

[1] http://www.isi.edu/nsnam/ns/

[2] https://en.wikipedia.org/wiki/Tcl

[3] http://inst.eecs.berkeley.edu/~ee122/fa05/projects/Project2/SACKRENEVEGAS.pdf

[4] http://ijcttjournal.org/Volume4/issue-8/IJCTT-V4I8P204.pdf

[5] http://www.cs.cmu.edu/~uhengart/infocom00.pdf

[6] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.19.6783&rep=rep1&type=pdf

[7] Comparison of TCP congestion control mechanisms Tahoe, Newreno and Vegas. Digvijaysinh B Kumpavat Prof. Paras S Gosai Prof. Vyomal N Pandya.