Performance Analysis of TCP Variants

Omkar Sambhus and Gaurav Durgule

Northeastern University
361 Huntington Avenue, Boston MA 02120
omkar469@ccs.neu.edu, gaurav1@ccs.neu.edu

1 Introduction

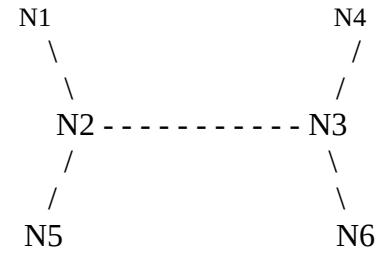
In this paper we analyze the performance of the four TCP variants namely Tahoe, Reno, New Reno and Vegas under the situations of high congestion and delay.
The original design of TCP (Tahoe) worked reliably, but it was unable to provide acceptable performance in large and congested networks. For example, the conventional TCP (Tahoe) waits for an RTT amount of time whenever a packet is lost and this is the case for every lost packet. Only after the time out does the TCP conclude that the network is congested and accordingly implements its congestion control measures. Hence we compare the Tahoe variant of the TCP which is the original version of TCP with the newly proposed variants like Reno, New Reno and Vegas.

2 Methodology

For the purpose of analysis we have used the Network Simulator version 2 (ns2) on Linux OS. NS2 was chosen for the following reasons.

1. Protocol Support
   NS2 supports all the protocols and their variants required for the experiments. (TCP, UDP, TCP-Reno, TCP- NewReno, TCP – Vegas, CBR and FTP)
2. Cost Factor
   Linux OS and hence NS2 is free to use for educational and research purposes.

The basic topology which we have used is the dumbbell topology which is as follows:



Section 3 is about gauging the fairness of the four TCP variants under various load conditions. We have assigned a constant bit rate (CBR) flow from N2 to N3. It is like an unresponsive UDP flow which uses a certain slice of bandwidth without bothering about the controlling congestion. A TCP flow is assigned from N1 to N4. N2-N3 is the common link for CBR and TCP flows and this vulnerable for congestion and bottlenecking. With this arrangement, all the four variants were run with varying delays and CBR bandwidths. The results were observed for a delay of 10ms and CBR rates of 2, 4, 6, 8 and 10 mb/s. The queuing algorithm used was DropTail and the queue size was kept as 10 which is the default queue size. Each link has a constant bandwidth of 10 mb/s. The simulation has be carried out for a period of 12 seconds. In the first set of experiments the UDP flow starts at 0.1 seconds and ends at 12.0 seconds whereas the TCP flow starts at 3.0 seconds and ends at 9.0 seconds.
The next set of experiments includes two TCP flows over a CBR flow. Hence we have assigned another TCP flow from N5 to N6 to the previous arrangement.
Section 4 is about the understanding the influence of queuing on the TCP variants. Two queuing algorithms were used namely, DropTail and Random Early Drop (RED). They are algorithms that control how packets in a queue are treated. The topology used is the same as before. However, here the UDP flow is from N5 to N6. In these set of experiments we start the TCP flow first. Once that is steady, the CBR flow is started. The purpose of this is to understand the behavior of TCP under a scenario where the channel experiences a sudden burst of UDP packets.

3 Results for Experiment 1

Experiment 1 revealed a plethora of vital numbers which give an idea of gauging the fairness of the four TCP variants under various load conditions.

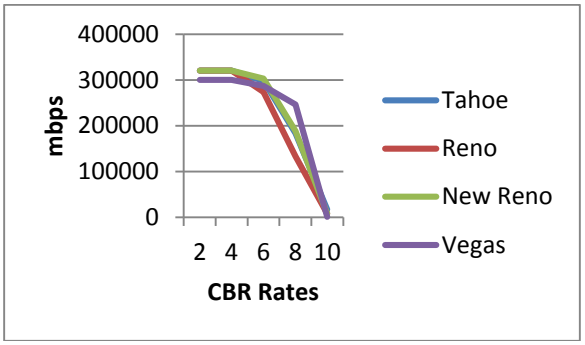


Figure 3.1

Figure 3.1 portrays the performance of the four TCP variants, with varying CBR rates (2,4,6,8,10 mbps), in terms of the bandwidth used by them. Vegas starts off with a relatively lower share of bandwidth but gradually picks up and as compared to the other variants it gets the maximum share of bandwidth at CBR rates 6 and 8 mbps which is when the congestion takes place. Thus Vegas has the best performance in terms of bandwidth during congestion. Surprisingly Reno's performance is inferior to Tahoe. The reason for this could be the Fast Recovery mechanism for packet losses.

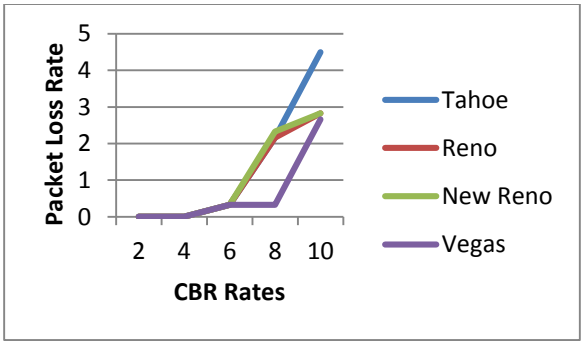


Figure 3.2

In terms of the packet loss rate in the presence of a UDP flow, Vegas trumps the other variants again as shown in Figure 3.2 by having the lowest packet loss rate. Tahoe has the maximum packet loss rate and Reno and New Reno show similar behavior. This is observed due to the difference in congestion handling mechanisms of the variants.

In the second set of experiments, we monitor and analyze the performance of these variants in a topology with two TCP flows and a UDP flows.

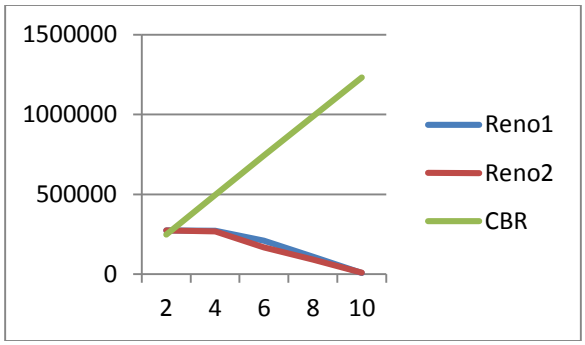


Figure 3.3

Figure 3.3 shows the bandwidth utilization for two TCP Reno flows running over a CBR flow. The bandwidth is evenly shared by the two Reno flows which were expected.

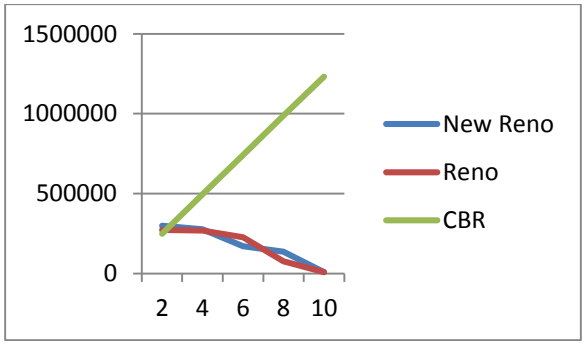


Figure 3.4

Figure 3.4 shows the bandwidth utilization for a Reno flow and a New Reno flow over a CBR flow. As it can be seen in the graph, New Reno takes up more of the bandwidth as compared to Reno when congestion occurs at CBR rates 6 to 9 mbps. The Fast Recovery mechanism enables the New Reno to send packets faster and hence utilize the bandwidth better as compared to Reno.

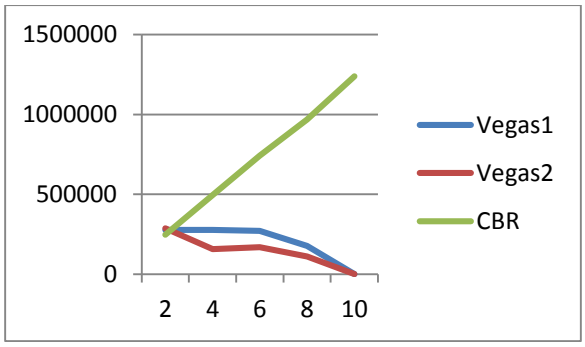


Figure 3.5

Figure 3.5 shows the bandwidth utilization for two Vegas flows over a CBR flow. Surprisingly, one Vegas flow utilizes more of the bandwidth consistently as compared to the other irrespective of the congestion. Since both the TCP Vegas flows are

started at the exact same instant, it so happens that one flow takes up the slice of bandwidth that it requires and not much is left for the other Vegas flow. Hence it takes up whatever is available rather than getting the bandwidth that it needs. A CBR is continuously running which takes up a slice of the bandwidth consistently. This is the reason due to which the second Vegas flow ends up getting a lesser amount of bandwidth.

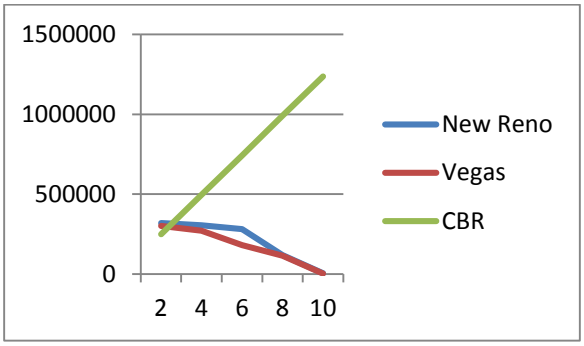


Figure 3.6

Figure 3.6 shows the bandwidth utilization of a New Reno flow and a Vegas flow over a CBR flow. The New Reno dominates in the early stages when there is no congestion, but as soon as the congestion occurs at CBR rate of 6, Vegas out performs New Reno and starts using more of the bandwidth. This is because of the when the network is not congested, the actual flow rate will be close to the expected flow rate. Otherwise, the actual flow rate will be smaller than the expected flow rate. TCP Vegas, using this difference in flow rates, estimates the congestion level in the network and updates the window size accordingly [2].

References

[1]http://www.articlesbase.com/networks-articles/network-simulator-2-ns2-4065753.html

[2] EVALUATION OF DIFFERENT TCP CONGESTION
CONTROL ALGORITHMS USING NS-2 Hui (Hilary) Zhang (hzhang@sfu.ca)
Zhengbing Bian (zbian@cs.sfu.ca)
http://www.sfu.ca/~zbian/courses/cmpt885/