Introduction

In this paper, we present the result of our study on the performances of different TCP variants. The NS-2 network simulator is used as the main tool for this study. We compared the performances of TCP Tahoe, Reno, NewReno and Vegas under different situations. In experiment 1, we analyze the performance of each one under different congestions. In experiment 2, we simulate two TCP flows to analyze how well they share bandwidth and their fairness to each other. In experiment 3, how queueing mechanism used in buffer impacts the TCP performance is studied. At the end of the paper, the key results from these experiments are highlighted.

Methodology

We conducted the simulation in NS-2. NS-2 is an object oriented network simulator which can simulate TCP, routing and multicast protocols over LANs and WANs. In terms of TCP’s performances, following properties are studied:

1. Throughput, the amount of successfully transferred data per second. The unit is in MB/s
2. Average RTT, the average round-trip delay time of packets. The unit is in second.
3. Drop rate, the percentage of package dropped. The unit is in percent.

The simulation of network topology, flow starting time, buffer size and queuing mechanism are set up by TCL scripts. The trace results give by NS-2 are exported to standard output and parsed by a Java program which also calculates throughput, average RTT and drop rate. We also conducted T-test for our results in order to check the significance of their statistical difference.

Experiment 1

In this experiment, we use a network topology as shown below. One TCP flow is sent from N1 to N4, and one CBR flow is sent from N2 to N3. The bandwidth of the links of N1-N2, N2-N3 and N3-N4 are all 10mbps. The experiment starts with CBR flow rate at 1mbps, it is increased in 1 mbps every iteration until bottleneck capacity 10mbps is reached. This experiment is conducted with following TCP variants: Tahoe, Reno, NewReno and Vegas. For each TCP variant, we run the experiment 10 times, in order to calculate T-value. The results of this experiment are the average results of 10 times. Fig 1.2-1.4 show the results in the form of throughput, drop rate and RTT.

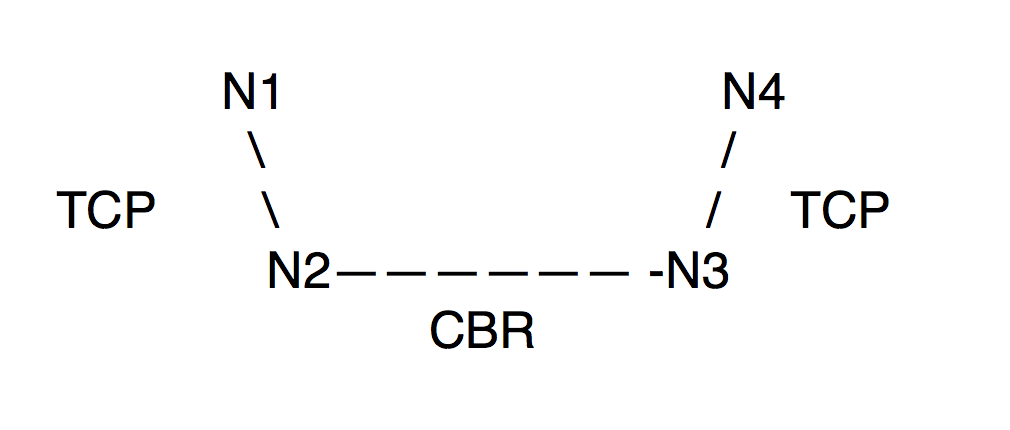


Fig. 1.1 Topology for experiment 1

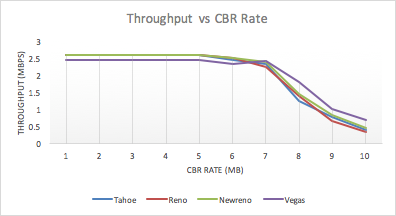


Fig. 1.2 Throughput of all TCP variants vs CBR rate

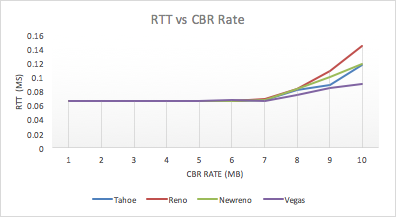


Fig. 1.3 RTT of all TCP variants vs CBR rate

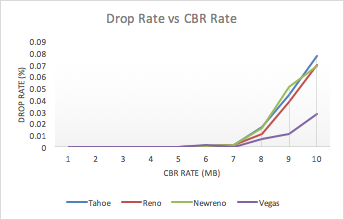


Fig. 1.4. DropRate of all TCP variant vs CBR rate

| T-Value | Tahoe | Reno | NewReno | Vegas |
| --- | --- | --- | --- | --- |
| Tahoe | 0 | 8.4079 | 5.8644 | 5.9734 |
| Reno | 8.4079 | 0 | 16.747 | 5.9188 |
| NewReno | 5.8644 | 16.747 | 0 | 21.955 |
| Vegas | 5.9734 | 5.9188 | 21.955 | 0 |

Table 1.1 . T-Value for throughput

Table 1.1 shows the T-value for the experiment, which is produced by running the experiment than 10 times. With p-value at 0.001, and degree of freedom at 18, if T-value is larger than 3.92, then the null hypotheses that the two TCPs have the same throughput can be rejected. As the value in the above table shows, we can safely conclude all the TCPs in our experiment have significantly different throughput. In the following paragraphs of this section, we will discuss the causes of these differences.

***Tahoe***, from the experiment it shows Tahoe has a relatively low throughput and high drop rate especially when the CBR rate is high. This is due to the fact that Tahoe takes a full retransmission time out to detect a packet loss. This causes many packets to be transmitted in vain.

***Reno***, unlike Tahoe, Reno enters fast retransmit once it receives three duplicate ACKs. As a result, it would not wasted many packets which would be discarded by the receiver. Therefore we can see it has a much better drop rate than Tahoe. However, due to the fast retransmission of Reno, it can send packets at a fast rate even when the network is pretty congested, which can cause many packets to be stranded in buffer for a long time. Therefore, it has a longer RTT than other TCPs.

***NewReno***, pretty similar to Reno, NewReno has one major difference which is it does not exit fast retransmission until all the data in the pipeline has been acknowledged. Due to this difference, NewReno generally is more aggressive than Reno. As a result, we can see that it has a high RTT and high drop rate. However, when the congestion is low its throughput is the best among all the TCPs.

***Vegas***, it uses a much more accurate mechanism to estimate RTT and decide retransmission timeout. We can see this has an enormous benefit when congestion is high. Among all the TCPs, Vegas has the best performance in terms of drop rate and RTT. However, Vegas uses additive increases in the congestion window, which makes it less aggressive than the other TCPs. This is reflected in the relatively lower throughput of it when the congestion is low. Whereas, when the congestion is high, Vegas’ better RTT estimation mechanism makes it outperformed all the other TCP in throughput.