Performance Analysis of TCP Variants

Insert Subtitle Here

Eugene Carlino†
Computer Science
The Roux Institute
Portland Maine USA
carlino.eu@northeastern.edu

Eugene Carlino†
Computer Science
The Roux Institute
Portland Maine USA
carlino.eu@northeastern.edu

ABSTRACT

This paper explores the performance of various TCP variants and queuing algorithms under different network conditions. Three separate experiments were conducted using the Network Simulator 2 (NS-2) to evaluate the throughput, packet drop rate and latency of TCP streams. Experiment 1 tested the performance of TCP variants (Tahoe, Reno, NewReno, and Vegas) under various load conditions. Experiment 2 analyzed the fairness of different combinations of TCP variants. Experiment 3 focused on the influence of queuing algorithms (RED and DropTail) on TCP performance. The results showed that TCP Vegas had the highest average throughput, while TCP Reno had the lowest drop rate. Additionally, RED and SACK caused more sporadic latency, and it was observed that RED is not suitable for dealing with SACK. The paper highlights the importance of choosing the right TCP variant and queuing algorithm based on the network conditions to optimize performance.

1.Introduction

Transmission Control Protocol (TCP) is a connection-oriented protocol. Connection-oriented communication is when a connection is established between the sender and the receiver before data transmission begins. This connection is maintained throughout the entire communication session. Its job is to ensure that the data is transmitted reliably in the correct order. TCP is an example of a connection-oriented protocol in that it uses a three-way handshake before transmitting data. TCP was introduced in the 1970's and has undergone several modifications and improvements to accommodate different network environments.

1.1 TCP Tahoe

TCP Tahoe was developed in the late 1980s. It is the original congestion control algorithm. It uses a slow-start mechanism to increase the sending rate gradually. It can also detect congestion by observing packet loss. When congestion does get detected, TCP Tahoe reduces the sending rate by cutting the congestion window size. It has a slower recovery from congestion and has been seen as less efficient than the newer TCP variants.

1.2 TCP Reno

TCP Reno was developed in the early 1990s and is one of the most widely used TCP variants. It uses the same congestion control mechanism as TCP Tahoe but has a more improved fast retransmit algorithm that enables a faster recovery from packet loss. TCP Reno came with a new feature called Fast Recovery. Fast Recovery allows the sender to continue transmitting packets even after packet loss.

1.3 TCP NewReno:

TCP NewReno was developed in the late 1990s. It is an improved version of TCP Reno because it uses a more efficient fast recovery algorithm to recover from packet losses. It also uses a feature called Selective Acknowledgment (SACK). SACK allows the receiver to notify the sender about the packets that have been received successfully. This reduced the need for retransmission of packets.

1.4 TCP Vegas

TCP Vegas was developed in the mid-1990s. TCP Vegas uses a different congestion control mechanism than the one found in TCP Reno and NewReno. TCP

Vegas's mechanism focuses on measuring the round-trip time(RTT) instead of packet loss. RTT is the time it takes for a packet to travel from the sender to the receiver and back again. Specifically, TCP Vegas uses a congestion avoidance algorithm that adjusts the sending rate based on the estimated RTT of the network. [5]

It has been found that TCP Vegas performs well in high-speed networks but struggles in networks with high delay.

1.5 SACK:

SACK was a feature that was introduced in TCP NewReno. It lets the receiver send an acknowledgment to the sender with information about which packets have been received successfully. This reduces the need for the sender to resend packets that do not need to be resent. This results in improved efficiency of data transmission.

1.6 RED:

Random Early Detection is an avoidance mechanism that was introduced in the mid-1990s. It detects the early stages of network congestion before it becomes a problem by monitoring the average queue length and randomly discarding packets before the queue overflows.

In total three separate experiments were conducted. Each experiment with be discussed individually but this section will discuss the commonalities between them.

1.7 NS-2

Network-Simulator 2 (NS-2) was used to simulate the network for each experiment. NS-2 has been used by other research studies looking to evaluate network technologies and protocols. NS-2 provided flexibility with network topology and protocols to conduct each experiment. It had the protocols that this paper wished to research already built into the simulator. Network topology and protocols were specified to the simulator in a Tcl (Tool Command Language) script. The Tcl was also used to generate output. The Output is a trace file. The trace file is a text file that contains events that occurred in the simulation. To analyze the trace file python was used to parse the file. The python language was chosen because of familiarity with the Matplotlib library which was used to graph the parsed data in the trace file.

2.Methodology

2.1 Topology

The topology seen in figure 1 was used for each experiment. A bandwidth of 10Mpbs was set on each link.

Figure 0: a visual representation of the topography used in the experiments

2.2 Experiment 1 - TCP Performance Under Congestion

In this experiment the performance of TCP variants under the influence of various load conditions were tested. The TCP Variants that were tested were the following: Tahoe, Reno, NewReno and Vegas. The topology in figure one was used in this experiment. The UDP flow was used at node 2 to node 3 to blindly send packets at a constant rate. Each TCP variant was tested by putting a single TCP stream at Node 1 to a sink at Node 4. The throughput, packet drop rate and latency of the TCP stream were looked at while varying the bandwidth of the CBR flow.

2.3 Experiment 2 - Fairness Between TCP Variants

Experiment 2 was the analysis of the fairness of TCP variants. This experiment looked at how different combinations of TCP variants worked together. In this experiment the topology in figure 1 was reused along

with the CBR flow between node 2 and node 3. In addition two TCP flows were started. One at node 1 with a sink at node 4 and another at node 5 with a sink at node 6. The pairs of TCP protocols analyzed were as follows: Reno/Reno, NewReno/Reno, Vegas/Vegas, NewReno/Vegas. The throughput, packet drop rate and latency of the TCP stream were looked at while varying the bandwidth of the CBR flow.

2.4 Experiment 3 - Influence of Queuing

Experiment 3 examined the effects of RED and DropTail queuing algorithms on network performance. The same topology depicted in Figure 1 was utilized for this experiment. Like Experiment 1, a single TCP flow was established between Node 1 and Node 4, with sinks located at Node 4. Both SACK and TCP Reno were tested in conjunction with RED and DropTail queuing disciplines, with a varying queue limit of 5. In this experiment, the TCP flows were initiated first, and after 4 seconds, the CBR flow commenced.

5 Results

5.1 Experient 1 Results

In Experiment 1, a total of 12 graphs were generated from the output of the NS-2 simulator as seen in Figure 1. These graphs display the drop rate, throughput, and average time of a TCP protocol with respect to the variance of the CBR flow's bandwidth. Figures B1-B4 illustrate the throughput of the protocols, with the total number of packets sent during the simulation on the Y-axis and the CBR bandwidth on the X-axis. Upon examining the different variants and their respective graphs, it is evident that TCP Vegas achieves the highest average throughput.

Graphs A1-A4 represent the drop rate of the TCP protocols. From these graphs, it can be observed that TCP Vegas has the lowest number of drops among all the protocols tested in the experiment. Graphs C1-C4 display the latency versus the CBR rate for each TCP variant tested in the network simulation. Each TCP protocol's latency decreased at a similar rate and started to rise once the CBR rate exceeded 7. However, TCP Vegas' latency stabilized with the lowest latency when the CBR rate was greater than 7. While this experiment did not calculate the RTT, the Vegas protocol's congestion control algorithm does take it into account. Due to Vegas having the highest throughput and the lowest average latency, it is reasonable to assume that TCP Vegas also had the lowest RTT.

Based on the results of Experiment 1, it is apparent that TCP Vegas is the most superior protocol tested in most situations. Its drop rate peaked when the CBR rate was set to 9 but quickly dropped off at 10, and it has the highest average throughput. The only aspect in which TCP Vegas did not perform well was the drop rate category. In this experiment, the peak drop rate of TCP Vegas was significantly higher than the rest (1.75% vs 0.4-3%); however, it quickly recovered and stabilized itself as the CBR rate increased.

5.2 Experiment 2 Results

For Experiment 2, a total of 12 graphs were generated, as seen in Figure 2. These graphs provide insights into the fairness between different TCP variants when sharing network resources.

As expected, the Reno-Reno and Vegas-Vegas combinations showed a high degree of fairness, with very similar packet loss, latency, and drop rates. This can be attributed to the fact that both flows use the same congestion control algorithms, leading to similar reactions to network conditions.

However, the Vegas-NewReno combination demonstrated a considerable degree of unfairness. NewReno was able to transmit more packets when the CBR was between 3 and 8. This disparity can be attributed to the differences in the congestion control mechanisms employed by each variant. NewReno's congestion control algorithm is reactive, adjusting its sending rate based on packet loss, while Vegas proactively adjusts its sending rate by monitoring the difference between expected and actual transmission rates.

Due to NewReno's reactive approach to congestion control, it is more aggressive in utilizing available bandwidth compared to the more conservative Vegas.

This difference in congestion control strategies results in NewReno capturing a larger share of the available bandwidth, causing an unfair distribution of network resources between the two TCP variants.

5.3 Experiment 3 Results

In Experiment 3, a total of 12 graphs were generated, illustrating the total packets sent, average time taken for a packet to be sent, and drop rate. The x-axis of these graphs represents time in seconds, rather than the variance over CBR rate as in the previous experiment.

Based on the results, it appears that each queuing discipline provides a fair bandwidth allocation for each flow. Examining the total number of packets sent in Column A reveals that the results are quite similar, with RED queuing discipline transmitting slightly more packets than DropTail.

When comparing the end-to-end latency of flows between RED and DropTail, significant differences can be observed, particularly in the average time graphs in Column B. The orange lines represent the TCP flows, and their trajectories show considerable variation. Reno-RED's latency initially decreases before increasing after the CBR flow starts, while SACK-RED and SACK-DropTail exhibit increased latency after the CBR flow begins.

To assess how the TCP flow reacts to the creation of the CBR flow, we can observe the changes in the orange lines after the 4-second mark, which is when the CBR flow starts. In all cases, changes occur at the 4-second mark. Overall, Reno-DropTail performs the best when the CBR flow begins. Additionally, the graphs indicate that DropTail does not transmit as many packets as RED and has a higher drop rate, becoming less reliable when the CBR flow is activated.

One of the goals of this experiment was to determine if RED is a suitable choice when dealing with SACK. From the results, it can be seen that the drop rate recovers quickly with the SACK-RED combination. However, as time progresses, the latency increases. If latency is a crucial factor, SACK and RED may not be the ideal combination. On the other hand, if reliability is the primary concern, SACK and RED would be the best option based on the experiment's outcomes. In conclusion, the suitability of RED and SACK as a combination depends on the specific requirements and circumstances.

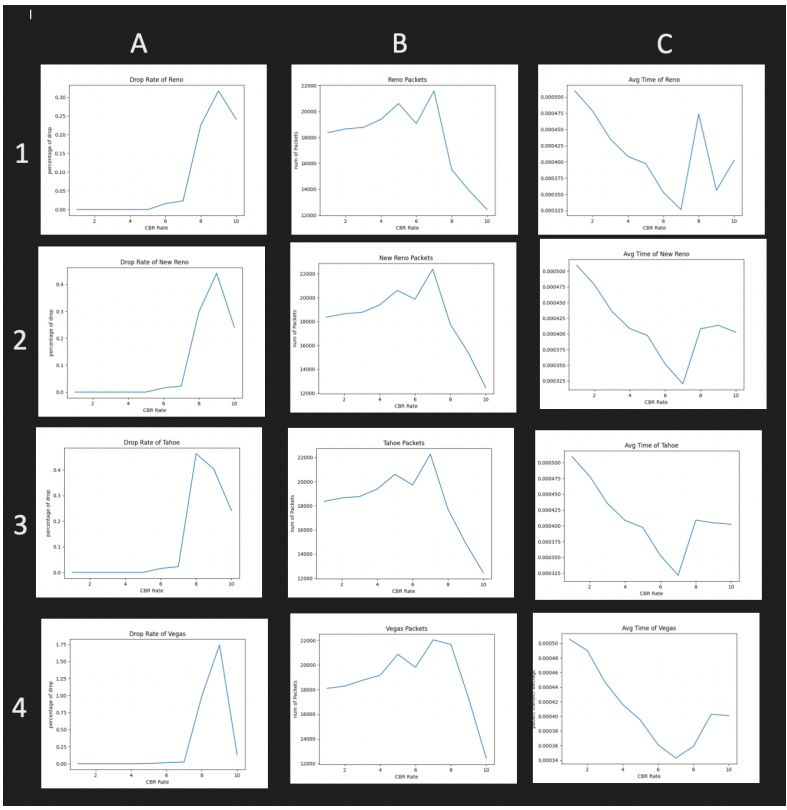

Figure 1: Contains graphs from experiment 1, please refer to attached pdf for better resolution

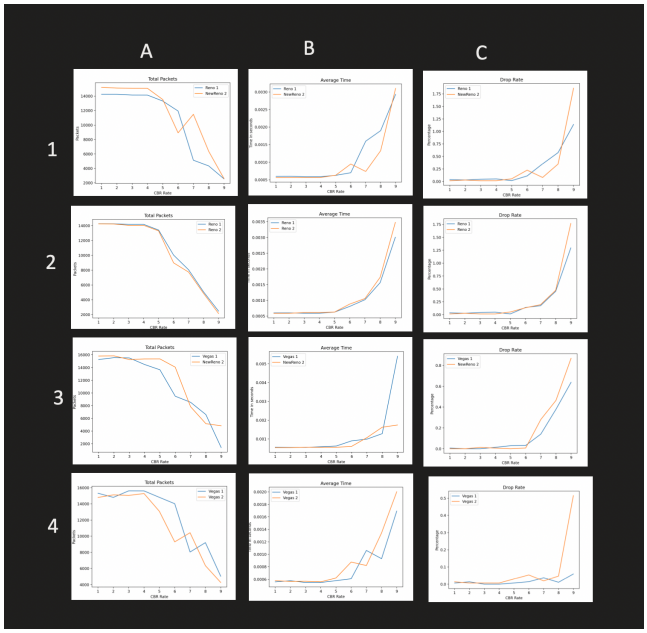

Figure 2: Contains graphs from experiment 1, please refer to attached pdf for better resolution

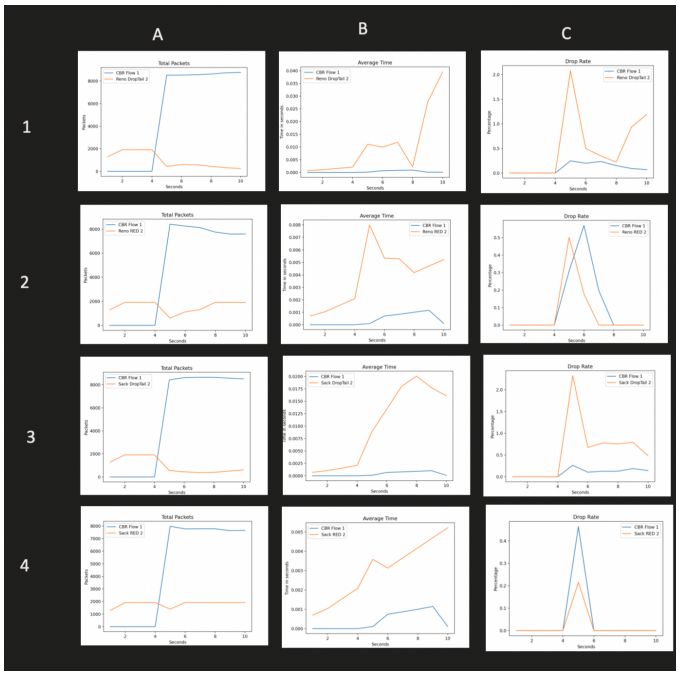

Figure 3: Contains graphs from experiment 3

6.Conclusion

This study investigated the performance of different TCP variants and their interactions under various conditions. The three experiments conducted focused on different aspects of TCP performance. Experiment 1 evaluated the performance of TCP variants under congestion. The results from experiment 1 show that TCP Vegas outperformed the other TCP variants. Experiment 2 evaluated the fairness between TCP variants. The results of experiment 2 revealed that the same-protocol combinations showed a high degree of fairness while the Vegas-NewReno pairing displayed considerable unfairness due to the differences in their congestion control mechanisms. Experiment 3 evaluated the influence of queueing algorithms on the overall network performance. The results suggested that the choice between RED and DropTail and their combination with SACK depends on the specific requirements of the network.