Lab 8: Comparing TCP Variants using `iperf` and Wireshark

[Your Name] Roll Number¿Roll Number

November 7, 2024

1 Introduction

This lab aims to compare the performance of different TCP variants, specifically Reno and Cubic, using `iperf` for throughput measurements and Wireshark for packet capture analysis. By setting controlled delay, loss, and bandwidth constraints on the loopback interface, we aim to understand the behavior of these TCP variants under varying network conditions.

2 Experimental Setup

2.1 TCP Variant Configuration

The available TCP variants on the machine were verified using:

```
cat /proc/sys/net/ipv4/tcp_available_congestion_control
```

Reno and Cubic were chosen for this experiment, with the default TCP variant set using:

```
sudo sysctl -w net.ipv4.tcp_congestion_control=<variant>
```

2.2 Network Parameter Configuration

Network parameters were set on the loopback interface using the `tc` and `netem` utilities as follows:

- Delay: 10ms, 50ms, 100ms
- Loss: 0.1%, 0.5%, 1%
- Bandwidth: 100 Mbps

2.3 Automation Script

A Bash script was created to automate the setup of network parameters, initiate `iperf` transfers, and log throughput data. Each experiment (TCP variant, delay, and loss combination) was repeated 20 times, and results were automatically saved to CSV files for analysis.

3 Experiments

Each experiment involved transferring a 20 MB text file from an `iperf` client to server over the loopback interface. The setup varied across:

- **Delays**: 10ms, 50ms, 100ms
- **Packet Losses**: 0.1%, 0.5%, 1%
- **TCP Variants**: Reno and Cubic

4 Results

4.1 Throughput vs. Delay

The following plots show throughput versus delay for both TCP variants at each loss level.

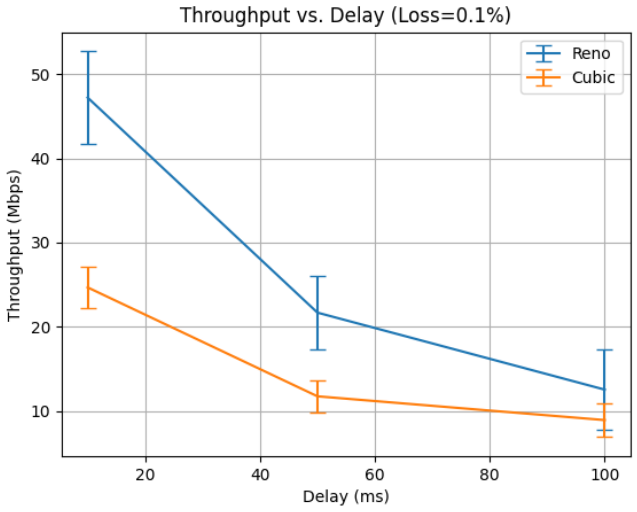

Figure 1: Throughput vs. Delay (Loss = 0.1%) for Reno and Cubic

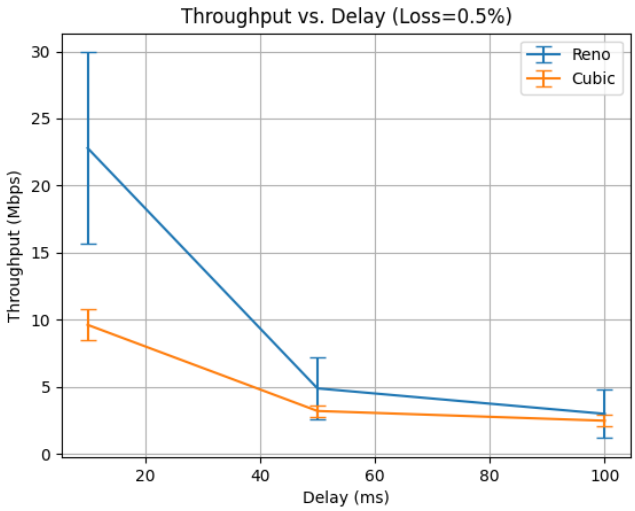

Figure 2: Throughput vs. Delay (Loss = 0.5%) for Reno and Cubic

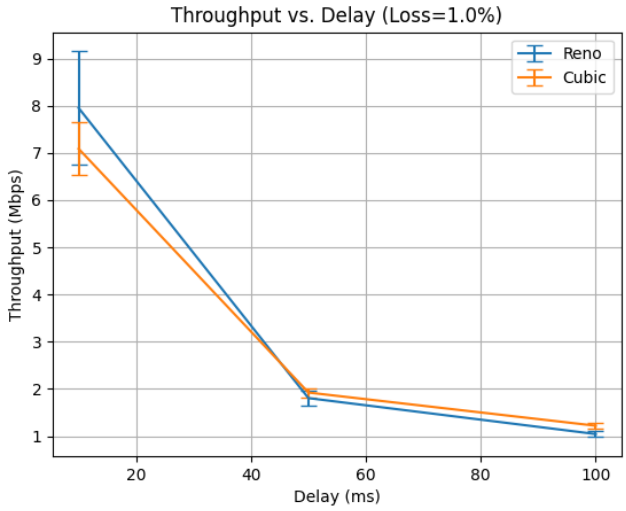

Figure 3: Throughput vs. Delay (Loss = 1%) for Reno and Cubic

4.2 Throughput vs. Loss

The following plots show throughput versus loss for both TCP variants at each delay level.

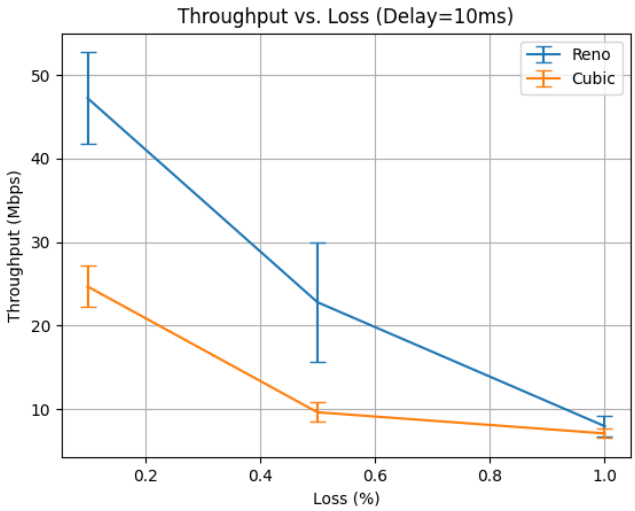

Figure 4: Throughput vs. Loss (Delay = 10ms) for Reno and Cubic

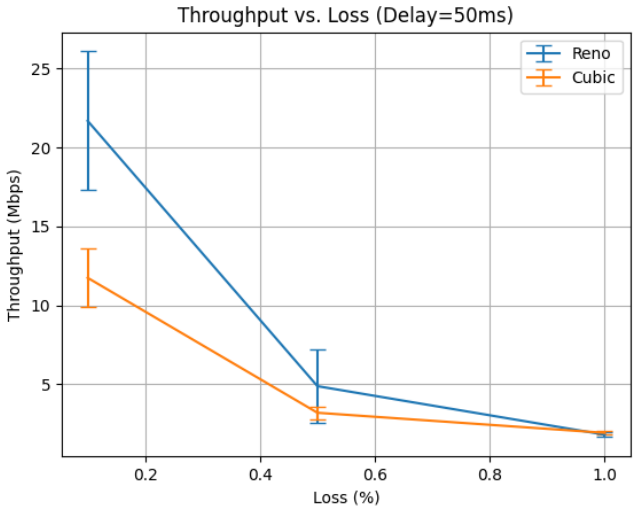

Figure 5: Throughput vs. Loss (Delay = 50ms) for Reno and Cubic

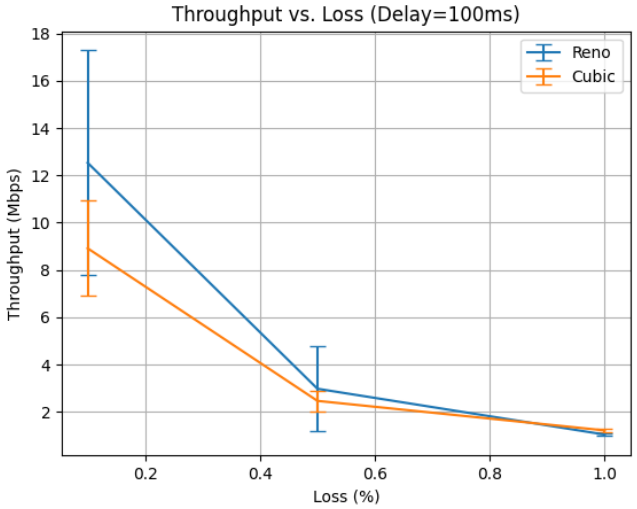

Figure 6: Throughput vs. Loss (Delay = 100ms) for Reno and Cubic

5 Analysis and Discussion

5.1 Throughput vs. Delay

Increase in delay results in an increase in RTT which in turn delays ACKs from the receiver and slow down the sender's ability to increase the CWND or initiate new packets, thus reducing the rate of data flow. $Throughput = CWND/RTT$ As RTT increases, throughput inversely decreases.

5.2 Throughput vs. Loss

TCP interprets packet loss as a sign of network congestion. TCP responds to packet loss by reducing its congestion window (CWND), the amount of unacknowledged data that can be sent before waiting for an ACK. In variants like Reno and Cubic, CWND decreases with every loss, reducing the rate at which data is transmitted. When a packet is lost, TCP also initiates a retransmission, which uses additional time and bandwidth.

5.3 Relative position of Cubic vs Reno

Cubic is more aggressive but is lower in throughput graphs. This might be because Cubic is designed to handle high-speed, high-latency networks more efficiently than Reno by having a cubic window growth pattern. However, when delay and loss both increase, Cubic's window adjustments can lead to overshoots or unstable growth. It might then back off quickly, dropping throughput as it

tries to regain stability. Cubic may drop the congestion window more significantly than Reno does in response to repeated packet loss in this setting, leading to performance oscillations and further reducing its average throughput.

5.4 Wireshark Analysis

For Delay = 10ms, Loss = 0.1% and Delay = 100ms, Loss = 1%, Wireshark was used to capture the "`bytes_in_flight`" data for each variant. Annotate the Wireshark plots, marking regions where Reno and Cubic were in Slow Start and Congestion Avoidance, as well as any packet loss events.

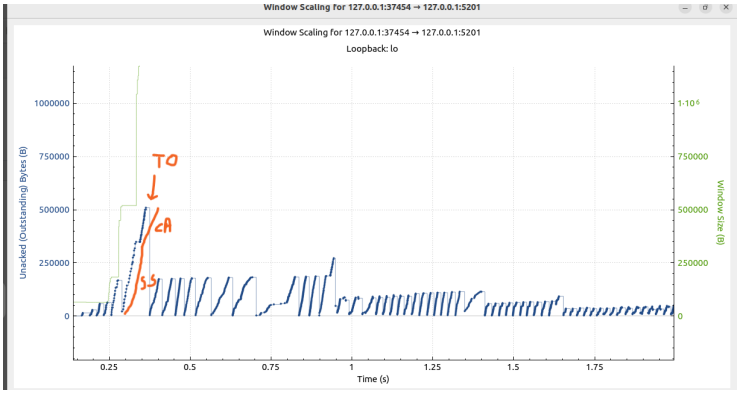

Figure 7: TCP Reno: Wireshark analysis for Delay = 10ms, Loss = 0.1%

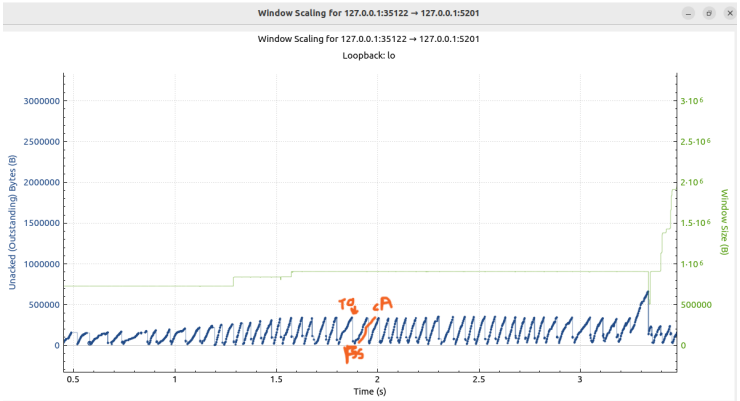

Figure 8: TCP Cubic: Wireshark analysis for Delay = 10ms, Loss = 0.1%

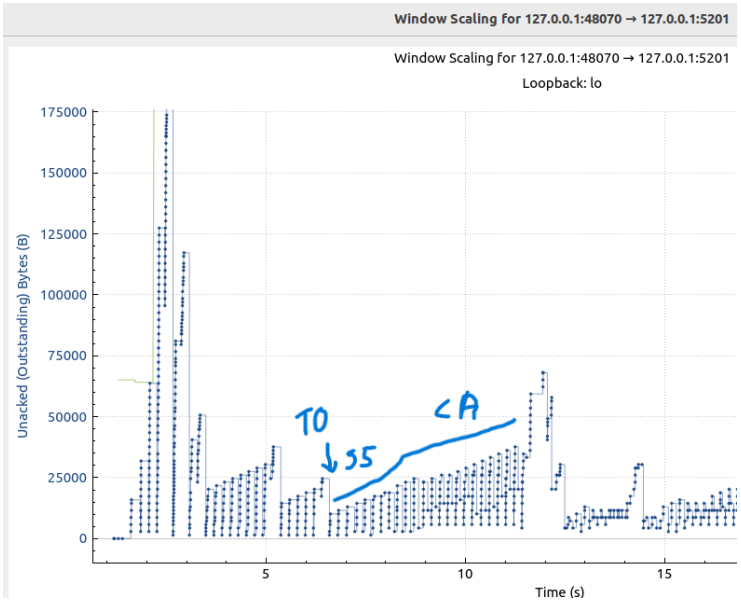

Figure 9: TCP Reno: Wireshark analysis for Delay = 100ms, Loss = 1%

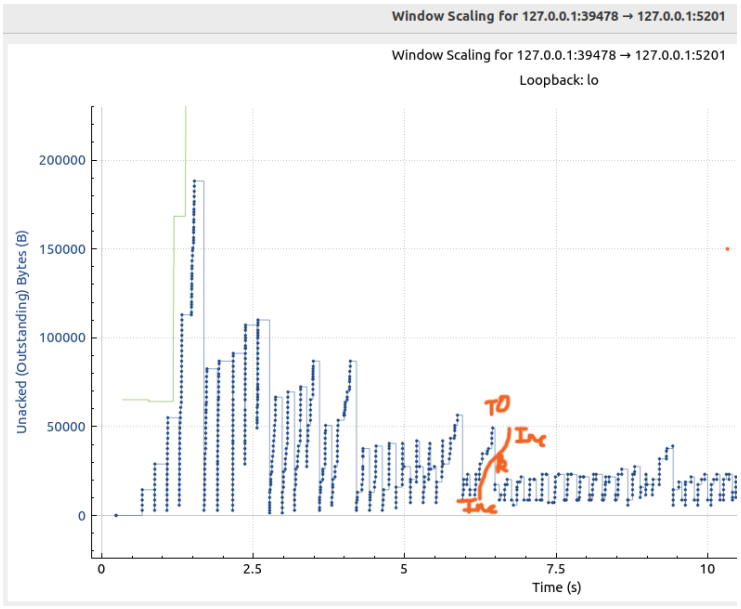

Figure 10: TCP Cubic: Wireshark analysis for Delay = 100ms, Loss = 1%

6 References

- Linux `tc` command documentation: `https://netbeez.net/blog/how-to-use-the-linux-traffic-co`
- TCP congestion control in Linux: `https://www.kernel.org/doc/Documentation/networking/tcp.txt`
- Confidence Intervals: `https://www.mathsisfun.com/data/confidence-interval.html`