COL334 Assignment 4: Reliable File Transfer over UDP with Congestion Control

Aditya Jha, 2022CS11102
Jakharia Rishit, 2022CS11621

October 30, 2024

1 Introduction

This assignment aims to design and analyze a reliable file transfer protocol implemented over UDP that mimics TCP's congestion control mechanisms. Using a dumbbell topology in Mininet, two client-server pairs communicate independently over separate paths that converge through a shared bottleneck link. This setup allows us to observe the effects of network congestion, packet loss, and delay on data transmission. The implemented protocol handles congestion using additive increase and multiplicative decrease (AIMD) and relies on duplicate ACK detection and timeouts to ensure packet delivery.

This report covers the steps to implement reliable data transfer and congestion control, including assumptions, analysis of reliability and congestion control results, and difficulties encountered.

2 Reliability Analysis

Reliability in UDP-based file transfer requires handling packet loss, retransmissions, and maintaining the correct order of data. We implemented a sliding window protocol with sequence numbers to track sent and acknowledged packets and duplicate ACK detection and timeouts for lost packets. The client sends cumulative ACKs for received data, allowing the server to update its window based on the highest acknowledged sequence number.

2.1 Plot and Explanations

2.1.1 Delay Experiment

Figure 1 shows the reliability performance of TTC over varying network delays. As expected, retransmissions increase with delay due to the timeout mechanism based on estimated RTT (Round-Trip Time). In scenarios with high delays, delayed ACKs cause the server to misinterpret unacknowledged packets as lost, resulting in unnecessary retransmissions. However, Fast recovery takes more time than without fast recovery, the reasons for which are discussed below.

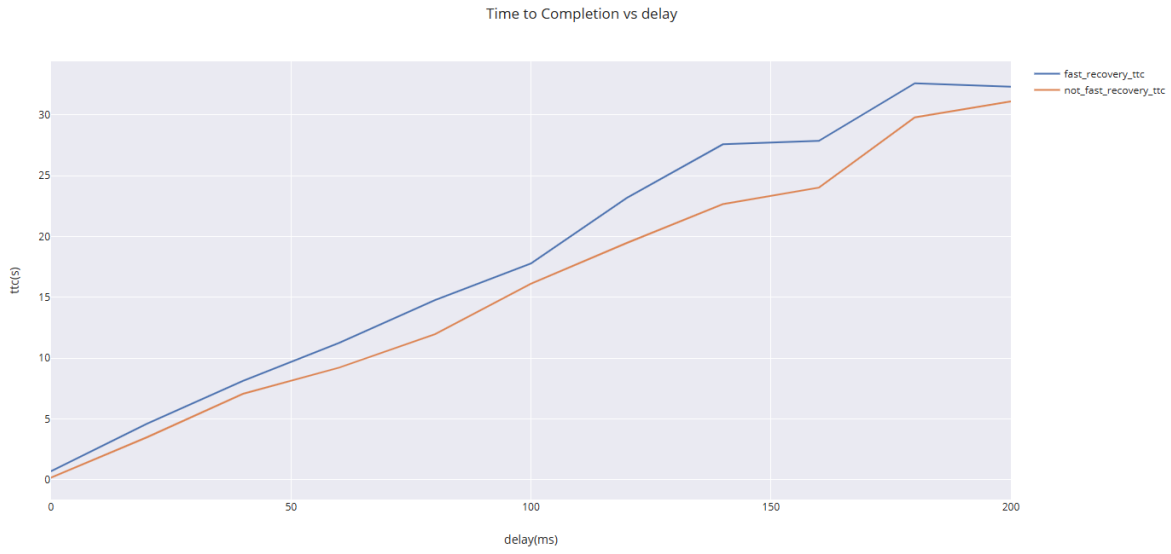

Figure 1: TTC observed with increasing network delay.

2.1.2 Loss Experiment

Figure 2 demonstrates reliability performance regarding time to completion over varying packet loss rates. With increasing loss rates, retransmissions become more frequent as lost packets trigger timeouts and duplicate ACKs, requiring retransmissions to ensure data integrity.

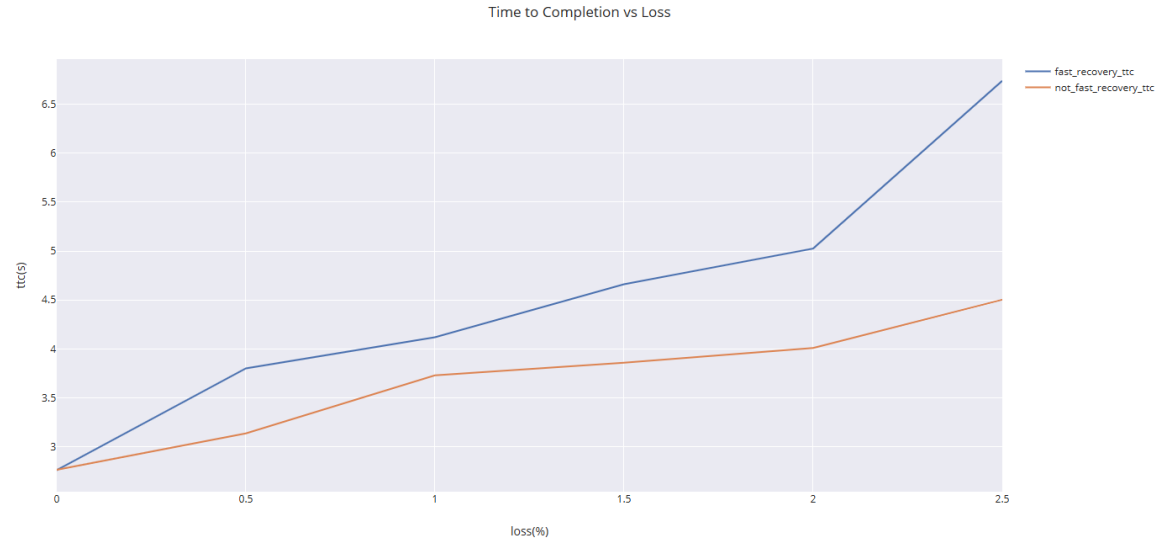

Figure 2: TTC observed with increasing network loss.

2.2 Difficulties

TTC vs delay

- Fast Recovery takes more time likely because.
  - It's triggering more frequent retransmissions than necessary
  - Each retransmission has to wait for the full network delay

  - The duplicate ACK counting mechanism may be resetting too frequently

- Without fast recovery (red line):
  - The system waits for timeouts
  - Relies on basic retransmission
  - Has less protocol overhead
  - More predictable behavior with increasing delay

TTC vs Loss

- As seen in the above graph, the fast recovery times are higher than the other, likely due to more packets being lost as there are overall more packets being retransmitted.
  - It only retransmits a single packet (ack_seq_num + 1) instead of continuing with new transmissions.
  - It resets the duplicate_ack_count to 0 after each fast recovery, which means it needs to receive three new duplicate ACKs before triggering fast recovery again.
  - Due to this, there are multiple losses and multiple retransmissions, causing a higher probability of losses due to a higher number of overall packets

3 Congestion Control Analysis

The server utilizes TCP-like congestion control to adjust its sending rate based on network congestion feedback. The congestion window (`cwnd`) starts in *Slow Start* mode, growing exponentially until reaching a threshold, after which it transitions to *Congestion Avoidance* with additive increase. Upon detecting three duplicate ACKs or a timeout, the server halves `cwnd` (multiplicative decrease) and adjusts the threshold to reduce congestion.

3.1 Plot and Explanation

3.1.1 Throughput vs Delay

Figure 3 shows the effect of increasing delay on throughput. As expected, throughput decreases with increasing delay since higher RTT values cause more frequent timeouts, limiting the congestion window growth and reducing effective transmission rates.

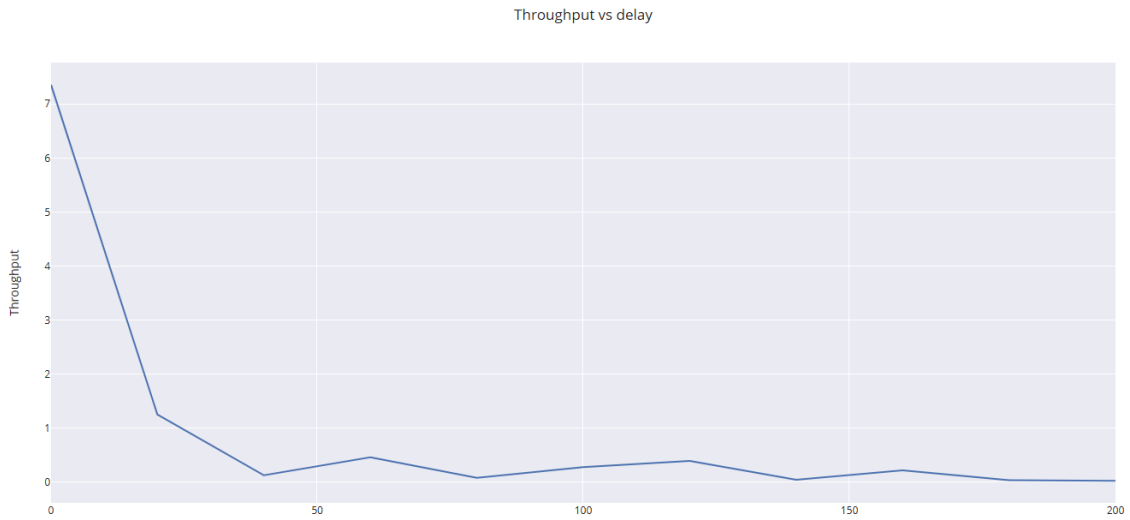

Figure 3: Throughput observed with increasing network delay(ms).

3.1.2 Throughput vs Loss

Figure 4 displays the impact of packet loss on throughput. Throughput decreases as packet loss increases since lost packets lead to more frequent retransmissions and slower `cwnd` growth, particularly under congestion.

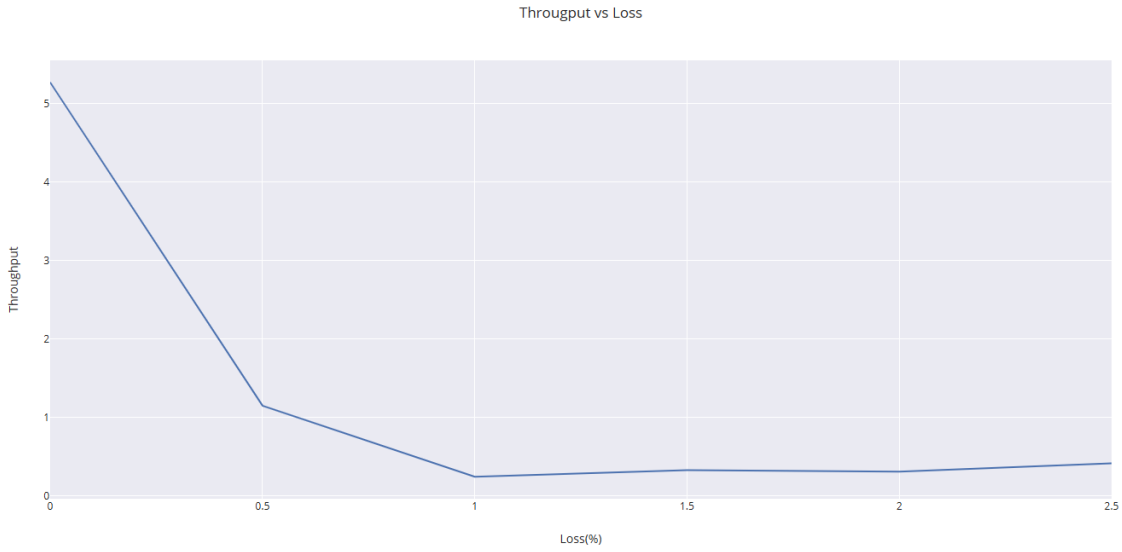

Figure 4: Throughput observed with increasing network packet loss.

3.1.3 Fairness - JFI vs Link Latency

Fairness was assessed using Jain's Fairness Index (JFI) to measure the sharing of network resources between the two client-server pairs. Figure 5 illustrates JFI values over varying network latencies, indicating how fairly the congestion control algorithm allocates bandwidth between flows. Higher JFI values (closer to 1) signify fair bandwidth distribution.

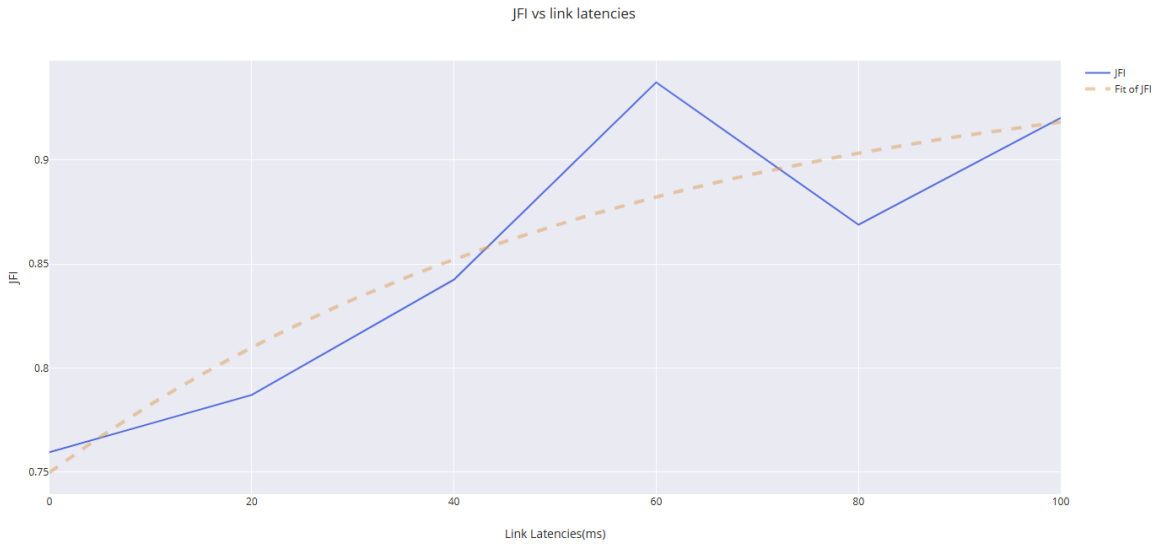

Figure 5: Jain's Fairness Index observed with increasing network latency.

3.2 Explanation of JFI:

1 **Increasing Fairness with Delay (0-60 ms):**

- As the network delay increases from 0 ms to 60 ms, JFI generally improves (from 0.76 to 0.94), indicating that the competing flows share resources more fairly.
- In lower-delay environments, the congestion window (cwnd) grows quickly due to the faster receipt of ACKs, allowing one flow to potentially dominate bandwidth briefly before the other flow adjusts.
- With moderate delays, the congestion window growth rate slows down slightly, giving both flows a better chance to reach equilibrium and share bandwidth more evenly.

2 **JFI Stabilizes:**

- Uptill 100 ms, JFI returns to a high value of 0.92, showing that, at extremely high delays, fairness again improves slightly. This could be because both flows experience enough delay to adjust to the network conditions similarly, leading to a stable equilibrium.

4 Conclusion

This assignment demonstrated the complexity of implementing reliable data transfer over UDP, particularly when simulating TCP-like congestion control. Our results show that RTT, network delay, and congestion control parameters significantly influence file transfer reliability and fairness. Future improvements could include developing a more adaptive timeout algorithm and adding selective acknowledgment (SACK) support for efficient retransmissions.