Assignment 2: Reliable Data Transfer and Congestion Control

EE5150: Communication Networks

February 26, 2025

1 Objective

This assignment aims to implement a UDP client that reliably transfers 10,000 packets to a designated server while dynamically adjusting the congestion window (measured in packets) to maximize throughput. The server script simulates a bottleneck link to the end host process, incorporating specific characteristics such as link capacity (measured in packets per second), round-trip delay (in msec), packet error rate, FIFO link service, and a finite link buffer. The end-host process at the server supports a cumulative acknowledgment mechanism to enable reliable data transfer over UDP.

2 Server Behavior

The server implements the following features:

- The server listens on a specified IP and port using UDP.
- The server simulates a bottleneck link to the end host process with the following characteristics:
  - **Round-Trip Time**, RTT: The bottleneck link introduces a propagation delay of RTT seconds for every packet.
  - **Packet Error Rate, PER**: The bottleneck link is unreliable, dropping a packet randomly with a fixed probability before the packet enters the link buffer.
  - **Buffer Size,** B: The bottleneck link's finite buffer can hold up to B packets. If the buffer is full, freshly arriving packets are dropped (Droptail).
  - **Link Capacity,** C: The bottleneck link operates at a fixed capacity of C packets per second.
- The server expects packets to adhere to the following structure:

  - A 4-byte sequence number (Big-Endian format, unsigned integer) that uniquely identifies a packet.
  - Optional payload (ignored by the server in this implementation).

- At the server, the packets are processed instantaneously and a feedback is sent via cumulative acknowledgement.

- The server uses cumulative acknowledgments (ACKs) to confirm the receipt of packets:

  - Each ACK contains a 4-byte sequence number representing the highest in-order sequence number received. For example, if the server has correctly and in order received packets with sequence numbers 0, 1, and 2, and when it receives packet 3, it will send an ACK with the value 3.
  - Acknowledgements are also sent for duplicate packets, where the ACK sequence number indicates the highest in-order sequence number received.
  - If packet losses occur, acknowledgments stall until missing packets are received.

3 Client Implementation

Your client must implement the following features:

- Reliable Data Transfer

  - Send 10,000 packets to the server over UDP, each with a unique sequence number (starting from 0). The packets shall adhere to the format expected by the server.
  - Use cumulative acknowledgments from the server and timeout to detect successful reception of packets as well as losses to transmit/retransmit packets. The client shall use a go-back-N like strategy for transmission as the server expects to receive packets in order.

- Congestion Control

  - Implement a congestion window to regulate the number of packets in flight and the sending rate dynamically. For example, you may start with a congestion window size of 1 packet and increase it using an additive increase/multiplicative decrease (AIMD)-like strategy.
  - Adapt the congestion window size based on network conditions to optimize throughput.

4 Deliverables

Submit the following:

1. A Python script implementing the UDP client that is compatible with the attached server script.
2. A brief one-page report summarizing your congestion control strategy and experimental throughput results for the following link parameters:

S.No.	Capacity (PPS)	RTT (msec)	PER (%)	Buffer Size (packets)	Throughput (Pps)
1	1000	100	0	100	
2	1000	100	0	10	
3	10	1	0	1	
4	10	1	10	10	

Table 1: Network Performance Metrics

5 Grading Criteria

Your submission will be evaluated based on:

- Correct implementation of reliable data transfer.
- Effective congestion control strategy measured in terms of throughput observed for different link scenarios.