Assignment 4 — Part 2: Congestion Control over UDP COL334/672 : Computer Networks

Rishika Garg (2023CS10820), Apurva Samota (2023CS10585)

5 November 2025

1 Introduction

Part 2 extends the reliable UDP transfer from Part 1 by adding a congestion control algorithm (CCA). The goal is to control the number of unacknowledged bytes (the congestion window, `cwnd`) to avoid overwhelming the network while maintaining high link utilization and fairness. Our implementation uses a **modular CUBIC-inspired** CCA (see code: `p2_server_refactored.py` and `p2_client_refactored.py`). The experiments below evaluate throughput, link utilization and fairness across four scenarios: fixed bottleneck bandwidths, varying random loss, asymmetric RTTs, and background UDP traffic.

2 Congestion Control Algorithm (Implementation Summary)

We implemented a CUBIC-inspired congestion controller with the following key elements:

- **Start:** initial `cwnd` = 1 MSS (MSS = 1180 B).
- **Slow Start** increases `cwnd` by the number of acked bytes per RTT until `ssthresh`.
- **CUBIC Growth:** after slow-start, cubic growth (function of elapsed epoch time) is used to increase `cwnd`; a TCP-friendly `w_tcp` is tracked and the controller uses the larger of cubic and TCP targets.
- **Loss Response:** on fast-retransmit (3 dup-ACKs) apply multiplicative decrease (w_max, ssthresh adjustments). On timeout, reset to slow-start (cwnd = 1 MSS) and backoff RTO.
- **RTO Estimation:** RTT smoothed estimator (EWMA) with dev term and RTO = EstRTT + 4*DevRTT. RTO is clamped to practical bounds and backed off on timeouts.
- **Selective ACKs (SACK):** receiver sends up to two SACK blocks to avoid retransmitting already received blocks.

3 Experimental Setup

Topology: dumbbell with two client–server pairs sharing a single bottleneck link. The file size used in experiments was ≈32.3 MB per flow (the exact size used in runs). Key metrics:

- **Link utilization** = (throughput_flow1 + throughput_flow2) / link capacity.
- **Jain's Fairness Index (JFI)** computed from observed throughputs of the two flows.

4 Results

4.1 1. Fixed Bottleneck Bandwidth (100 Mbps – 1 Gbps)

Summary (averages across runs):

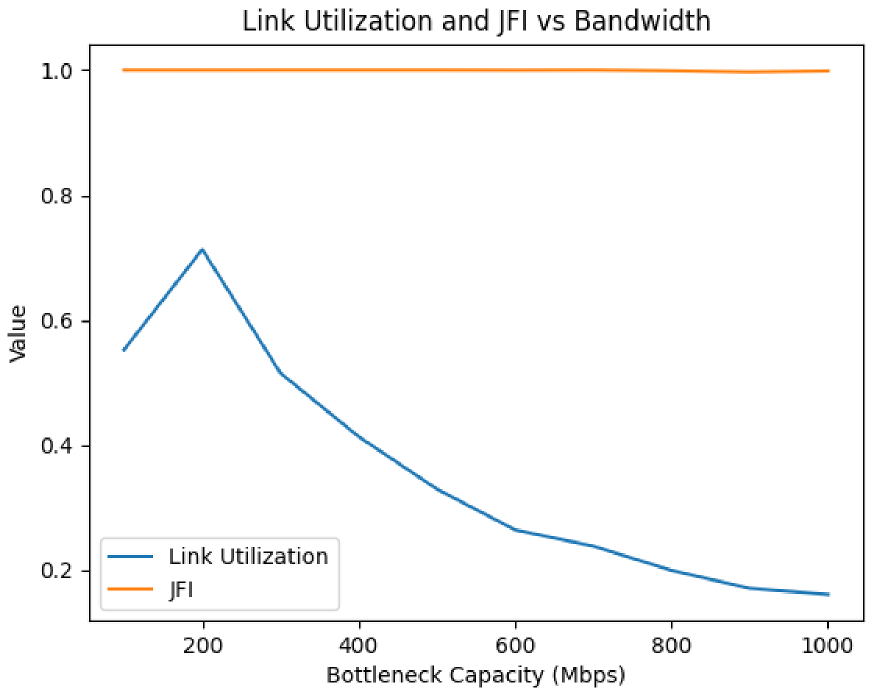

Figure 1: JFI and link utilization vs. bottleneck capacity.

Observations:

- The CUBIC-based sender maintains **very high fairness** (JFI ≈ 1.0) across bandwidths: both flows share the bottleneck nearly equally.
- Link utilization decreases with increasing link capacity (in these tests throughput per flow did not scale linearly with capacity). This happens because the experimented file size and per-flow sending dynamics limit achieved throughput at very high capacities in our testbed (Mininet host/cpu constraints and limited run time).

- Peak utilization appears around 200 Mbps in these runs; at larger capacities, absolute throughput per flow plateaus so utilization drops.

4.2 2. Varying Random Loss (0% – 2%)

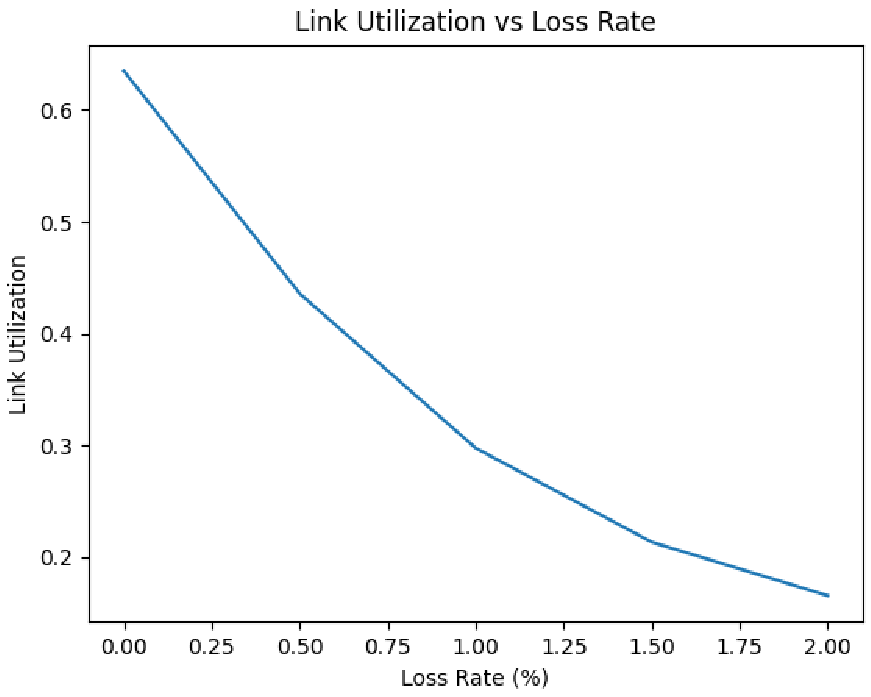

Figure 2: Link utilization vs. random loss rate.

Observations:

- Throughput (hence link utilization) drops rapidly with small amounts of random loss. At 1% loss utilization is already under 30% of link capacity.
- CUBIC recovers from loss using multiplicative decrease and then cubic growth, but random losses cause frequent reductions in cwnd and significant retransmission overhead.
- This result highlights sensitivity of loss-based CCAs to non-congestion losses (common in experimental networks).

4.3 3. Asymmetric RTTs (Client2 delay varied: 5 ms – 25 ms)

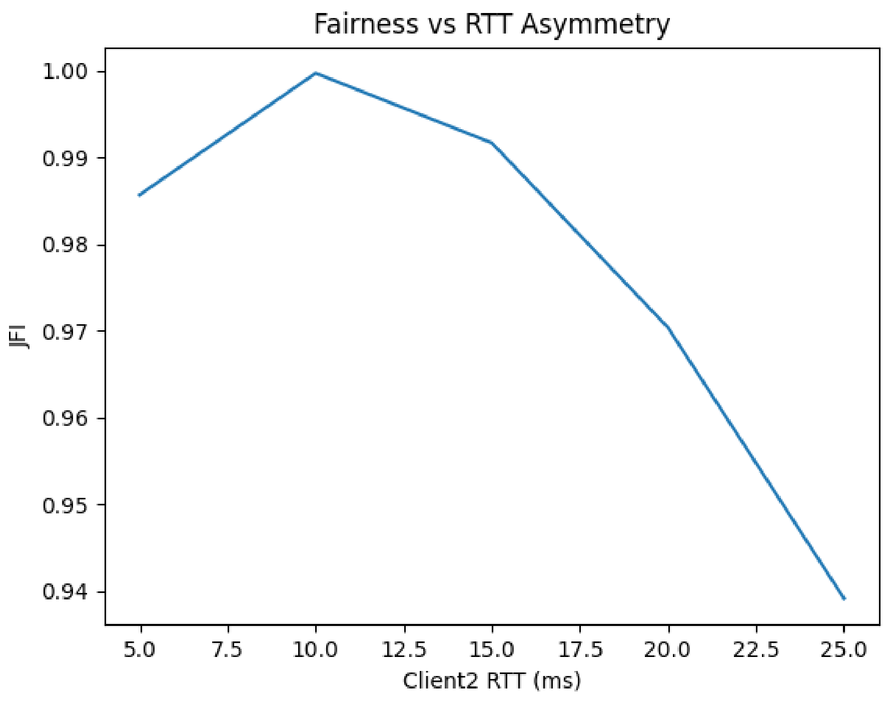

Figure 3: RTT (Client2) vs JFI.

Observations:

- Fairness remains high for small RTT differences; slight degradation appears at larger asymmetries (JFI drops to ∼0.94 at 25 ms).
- Link utilization showed some variability: higher RTT asymmetry sometimes increased measured utilization (likely due to timing interactions between cwnd growth and the measurement interval).
- Overall, the CUBIC-like controller is reasonably RTT-fair in this range, but extreme RTT differences can affect short-term fairness.

4.4 4. Background Bursty UDP Traffic

We simulated a short-bursty UDP background flow with ON/OFF behavior and varied the average OFF duration (labels in data: 0.5, 0.8, 1.5) to represent heavy → light background load.

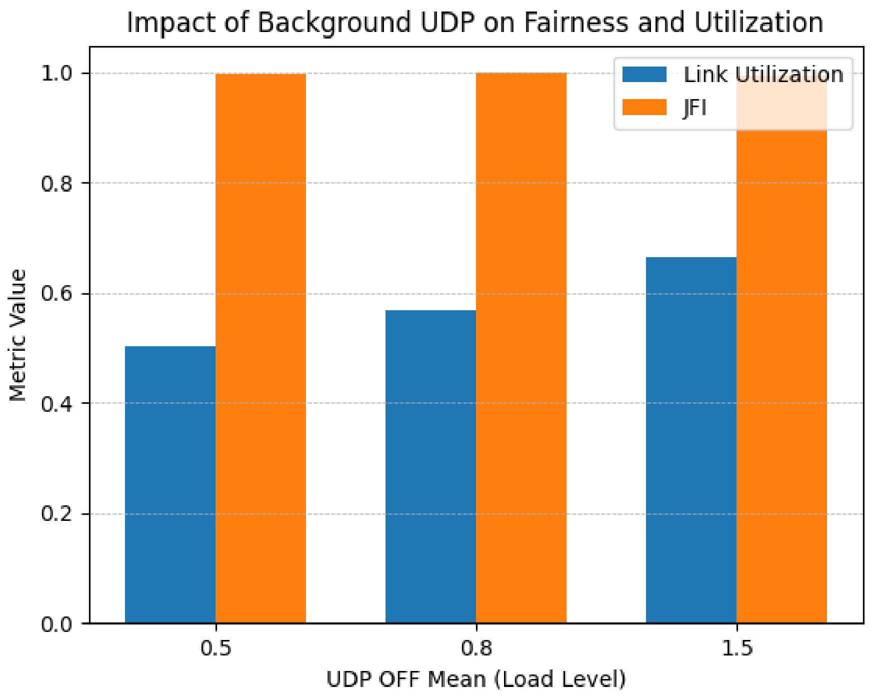

Figure 4: Background UDP load: utilization and fairness.

Observations:

- Heavy background load (shorter OFF mean) reduced fairness slightly but kept it close to 1.0 in our tests — CUBIC flows still share the leftover capacity fairly.
- Link utilization increased under heavier UDP load (more bytes delivered overall), but available capacity for TCP-esque flows decreased.
- Bursty UDP can cause transient queueing and packet loss at the bottleneck, invoking loss response in the CCA and temporarily reducing cwnd.

5 Conclusions

- The modular CUBIC-inspired CCA delivers **high fairness** across a variety of conditions (JFI $\approx$ 0.94–1.00 in our experiments).
- **Link utilization** is sensitive to random packet loss; even 1% loss severely reduces effective throughput in this setup.
- Asymmetric RTTs and bursty background UDP traffic affect short-term fairness and throughput; however, the implemented controller remains robust under moderate asymmetry and background load.

- Practical testbed limitations (Mininet host CPU, small file sizes, measurement intervals) can limit observed scalability at very high link capacities; these factors should be considered when interpreting utilization values at 500+ Mbps.