COL 334/672: Assignment 3 - Milestone 1 Report

Parth Thakur, 2021CS50615
Amish Kansal, 2021CS50622

October 15, 2023

1 Introduction

In this milestone, the goal was to send and receive UDP packets in accordance with the specified protocol. The server provided information about the total size of the data to be received. The client's task was to reliably receive the data, overcoming the challenges posed by a lossy network and the server's leaky bucket filtering mechanism.

2 Implementation Details

The code was written in Python, leveraging the socket and hashlib libraries. The client sends requests to the server and waits for a response. If no response is received within a specified timeout, the client resends the request.

2.1 Reliable Data Reception

To ensure the reliable reception of data, the client:

- Sends a "SendSize" request to the server to get the total size of the data.
- Sequentially requests data packets starting from an offset. If the expected packet is not received or if the received packet's offset and size do not match the expected values, the client resends the request.
- Introduces sleep intervals between requests to avoid overwhelming the server.

2.2 Hash Verification

After receiving all the data packets, the client computes the MD5 hash of the received data and sends it to the server for verification.

2.3 Graphical Analysis

The client logs the time and offset for each data request sent and received. This data is used to generate graphs showing the sequence-number trace and a zoomed-in view of the trace.

3 Results and Observations

The attached figures show the sequence-number trace and its zoomed-in view. The blue dots represent the time and offset of the requests sent by the client, while the orange dots represent the corresponding replies received from the server.

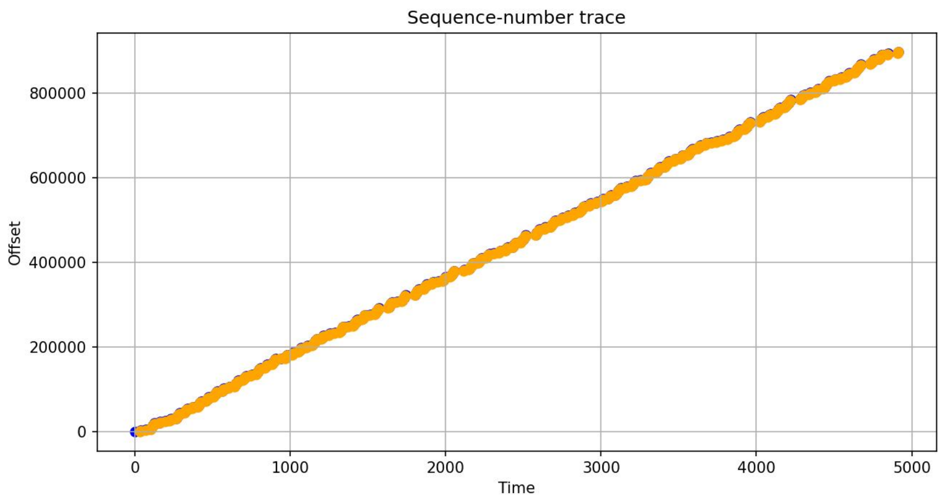

Figure 1: Sequence-number trace

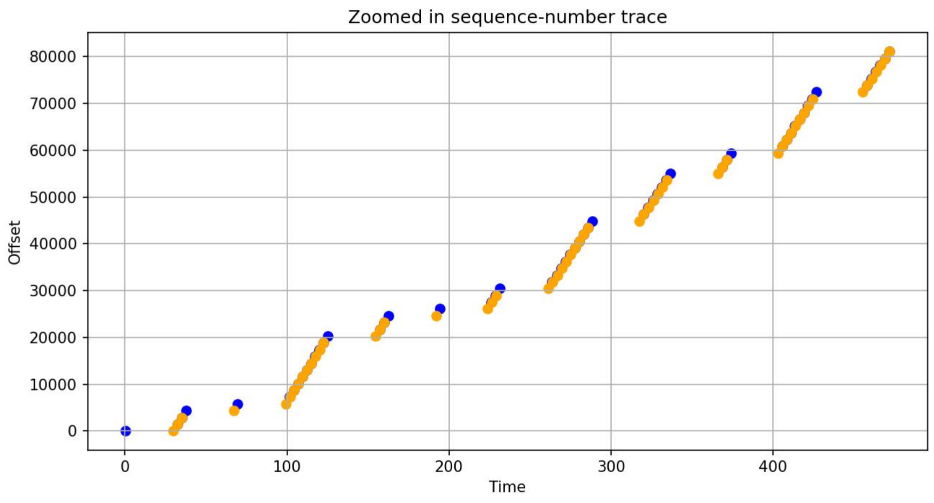

Figure 2: Zoomed in sequence-number trace

The client was able to extract the entire file (880000 bytes) in close to 5 seconds with the total penalty being 2 and the MD5 hash was confirmed by the server.

Several requests were skipped by the server as seen in the server logs. The client was able to detect this and re-request the skipped packets.

4 Conclusion

The client was able to reliably receive data from the server by retransmitting requests when necessary and adjusting the request rate to avoid overwhelming the server. The generated graphs provided insights into the client-server interaction dynamics and the reliability mechanisms in place.

5 Future Work

For the upcoming milestones, the focus will be on optimizing the request rate to maximize throughput without getting ”squished” by the server’s leaky bucket filter and on handling variable server rates.