BLG433E Term Project Report
Reliable Data Transfer Protocol over UDP

Mustafa Can Çalışkan, 150200097
Yusuf Emir Sezgin, 150200066

January 26, 2025

Contents

1 Introduction

This project implements a reliable data transfer protocol over UDP, similar to TCP's functionality but built from scratch. The implementation includes both client and server components that handle packet loss through the Selective Repeat protocol, with various window sizes (1, 10, 50, 100) and error rates (1%, 5%, 10%, 20%) to test protocol performance. The system establishes connections through a three-way handshake, manages packet timeouts, and ensures reliable data delivery despite using an unreliable UDP channel. The project structure and detailed instructions for running the implementation are provided in the following sections.

2 Implementation

This section summarizes the project's file structure and instructions for running the implementation.

2.1 Project Structure

The project structure can be seen in the figure below. All experiments were conducted using this structure.

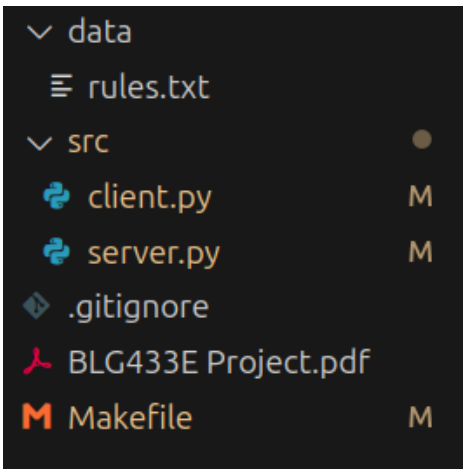

Figure 1: Project Structure

2.2 Makefile

The Makefile used for running the project can be found below. Experiments were conducted by modifying the SELECTED_ERROR_RATE and SELECTED_WIN_SIZE values within this file.

```
.PHONY: all run-server run-client clean

PYTHON = python3.12
SLEEP = sleep 5
TERMINAL = gnome-terminal --
TEXTFILE = rules.txt

# [1, 5, 10, 20]
SELECTED_ERROR_RATE = 1

# [1, 10, 50, 100]
SELECTED_WIN_SIZE = 10

all: run-server
    $(SLEEP)
    @make run-client

run-server:
    @$(TERMINAL) bash -c "$(PYTHON)
    src/server.py $(SELECTED_WIN_SIZE) $(SELECTED_ERROR_RATE);
    exec bash" & echo "Server started in new terminal" &

run-client:
    @$(TERMINAL) bash -c "$(PYTHON)
    src/client.py $(SELECTED_WIN_SIZE) $(SELECTED_ERROR_RATE) $(
        TEXTFILE);
    exec bash" & echo "Client started in new terminal" &

stop:
    @pkill -f "$(PYTHON) src/server.py" || true
    @pkill -f "$(PYTHON) src/client.py" || true
    @echo "Stopped all processes"
```

Listing 1: Makefile

2.3 Sample Text File

The sample text file "rules.txt" used can be found below. This file is located in the ./data directory.

```
### Fight Club Rules ###
1- You do not talk about Fight Club.
2- You do not talk about Fight Club.
3- If someone yells "stop!   , goes limp, or taps out, the fight is
     over.
4- Only two guys to a fight.
5- One fight at a time, fellas.
6- No shirts, no shoes.
7- Fights will go on as long as they have to.
8- If this is your first time at Fight Club, you have to fight.
9- You do not ask questions.
10- YOU DO NOT ASK QUESTIONS.
11- No excuses.
12- No lies.
13- You HAVE to trust Tyler.
```

Listing 2: rules.txt

2.4 How to Run

To run the project, simply execute "gmake all". This command will launch two separate gnome terminals for the client and server.

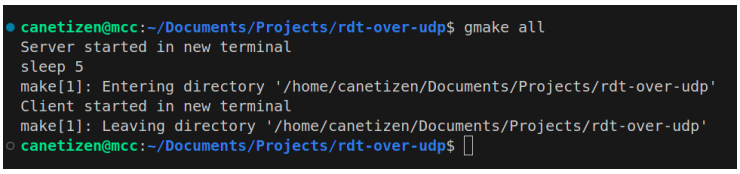

Figure 2: Terminal Description

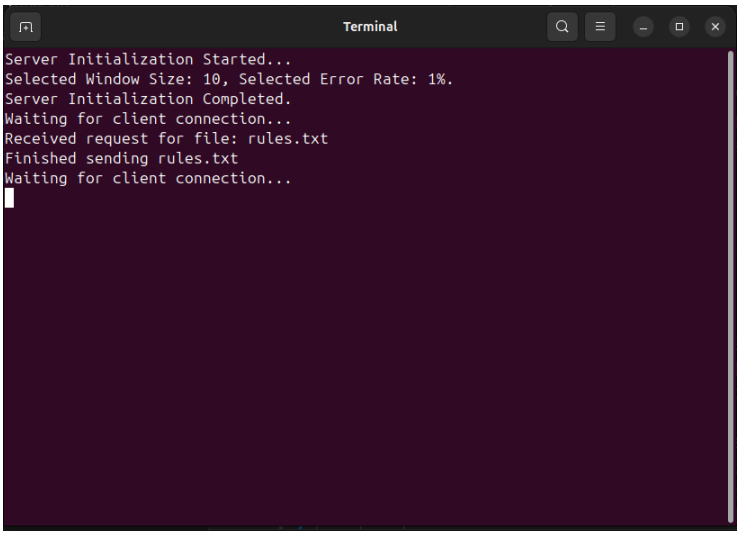

Figure 3: Server Description

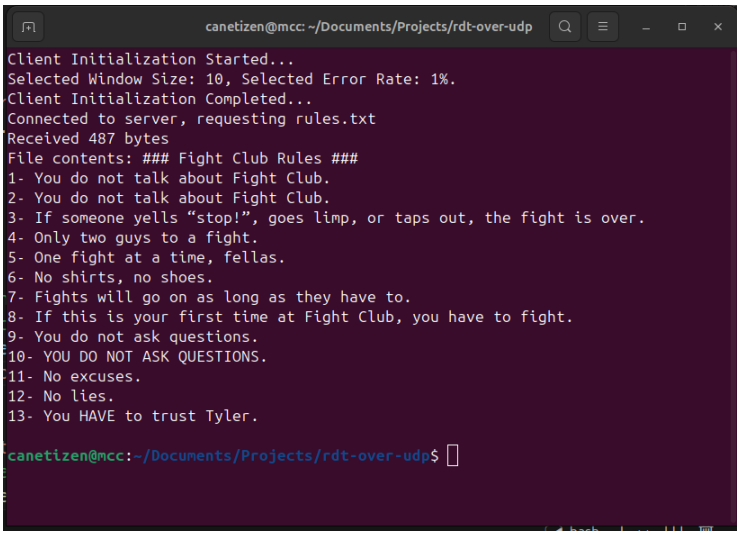

Figure 4: Client Description

3 Results

The transmission characteristics for different parameter changes are detailed separately in each subsection.

3.1 Window Size = 1

3.1.1 Error Rate = 1%

The data sending was not completed due to too short window size.

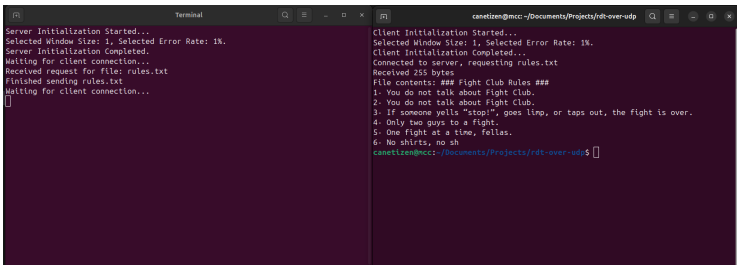

Figure 5: Window Size = 1, Error Rate = 1

3.1.2 Error Rate = 5%

The data sending was not completed due to too short window size.

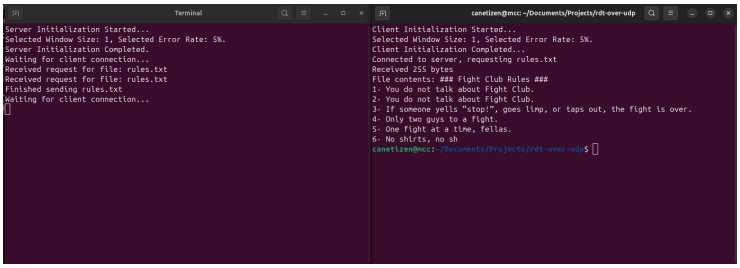

Figure 6: Window Size = 1, Error Rate = 5

3.1.3 Error Rate = 10%

The data sending was not completed due to too short window size.

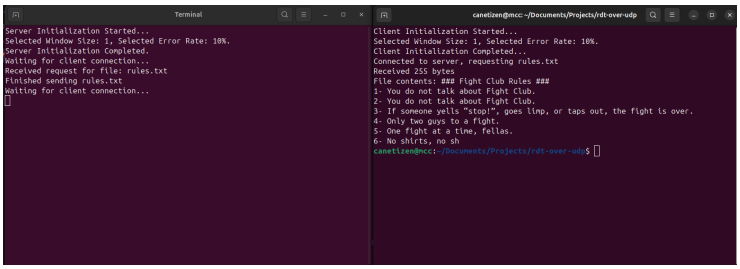

Figure 7: Window Size = 1, Error Rate = 10

3.1.4 Error Rate = 20%

The data sending was not happened due to error.

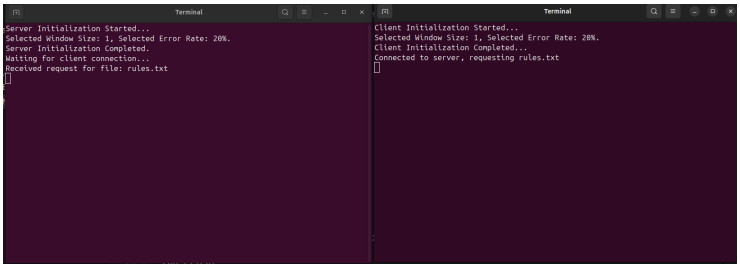

Figure 8: Window Size = 1, Error Rate = 20

3.2 Window Size = 10

3.2.1 Error Rate = 1%

The data sending completed without error.

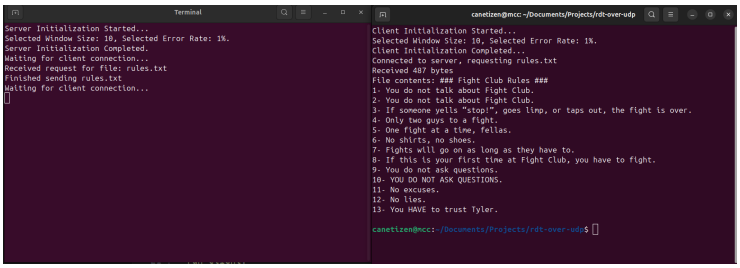

Figure 9: Window Size = 10, Error Rate = 1

3.2.2 Error Rate = 5%

The data sending completed without error.

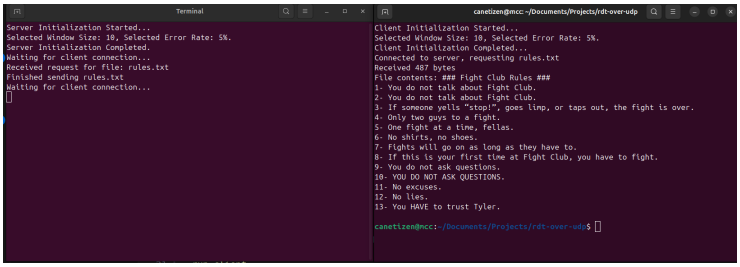

Figure 10: Window Size = 10, Error Rate = 5

3.2.3 Error Rate = 10%

The data sending completed without error.

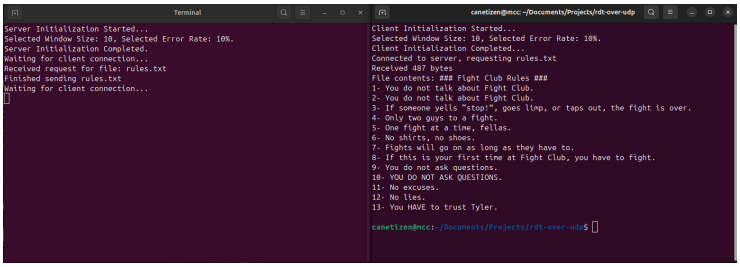

Figure 11: Window Size = 10, Error Rate = 10

3.2.4 Error Rate = 20%

The data sending completed without error.

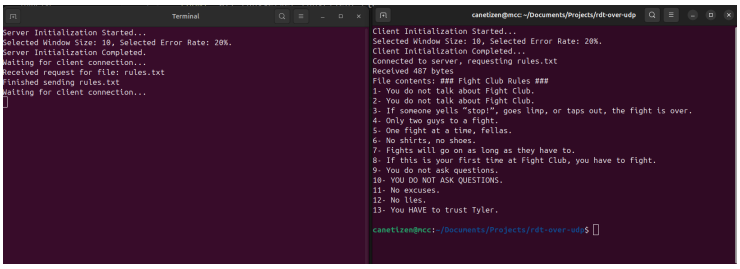

Figure 12: Window Size = 10, Error Rate = 20

3.3 Window Size = 50

3.3.1 Error Rate = 1%

The data sending completed without error.

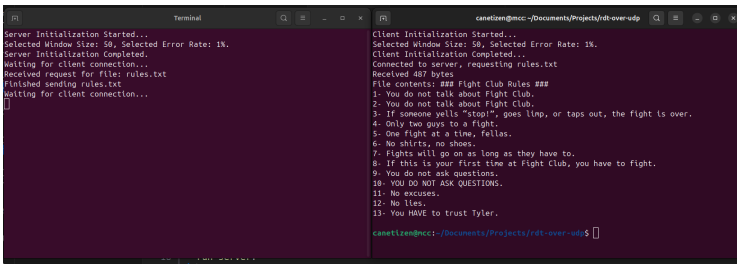

Figure 13: Window Size = 50, Error Rate = 1

3.3.2 Error Rate = 5%

The data sending completed without error.

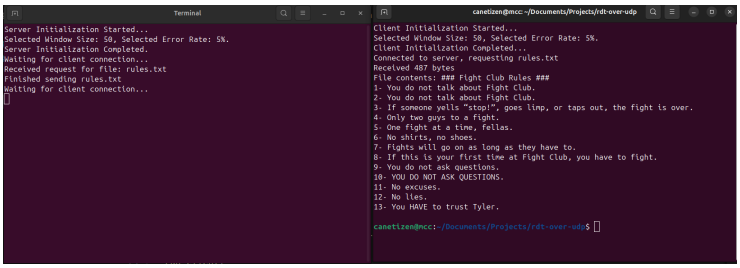

Figure 14: Window Size = 50, Error Rate = 5

3.3.3 Error Rate = 10%

The data sending completed without error.

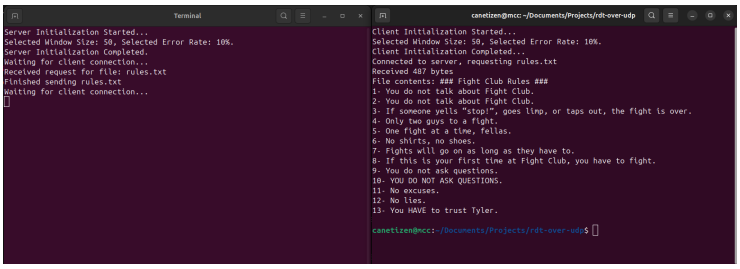

Figure 15: Window Size = 50, Error Rate = 10

3.3.4 Error Rate = 20%

The data sending completed without error.

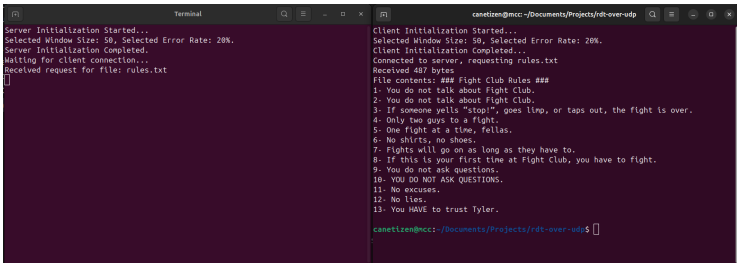

Figure 16: Window Size = 50, Error Rate = 20

3.4 Window Size = 100

3.4.1 Error Rate = 1%

The data sending completed without error.

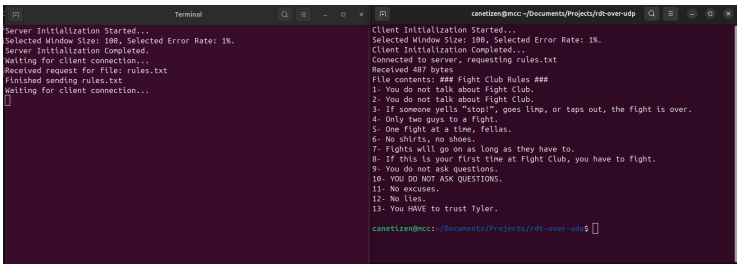

Figure 17: Window Size = 100, Error Rate = 1

3.4.2 Error Rate = 5%

The data sending completed without error.

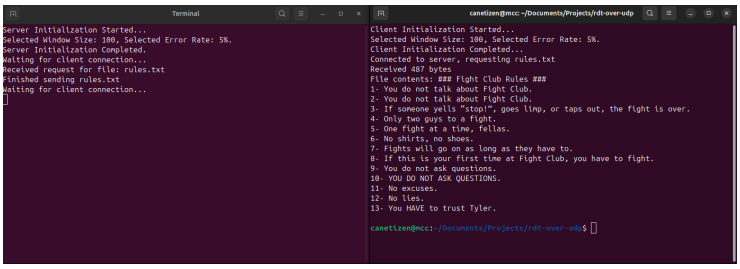

Figure 18: Window Size = 100, Error Rate = 5

3.4.3 Error Rate = 10%

The data sending completed without error.

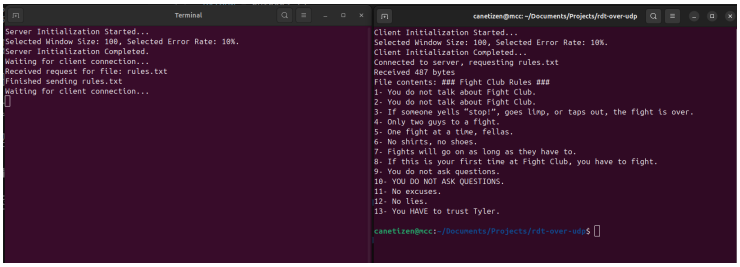

Figure 19: Window Size = 100, Error Rate = 10

3.4.4 Error Rate = 20%

The data sending completed without error.

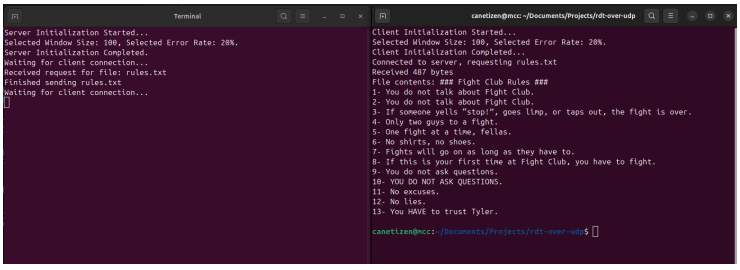

Figure 20: Window Size = 100, Error Rate = 20

4 Discussions

Our observations indicate that when the window size is too small relative to the amount of data being transmitted, the client may fail to receive the entire dataset due to buffer overflow. This limitation arises because the Selective Repeat protocol requires sufficient buffer space to store out-of-order packets until they can be correctly reassembled.

Additionally, the error rate significantly affects the transmission process, as a higher error rate increases the probability of unacknowledged packets. This results in more frequent retransmissions, leading to potential delays and inefficiencies in data delivery. In our experiment, we encountered an error once, likely due to the fact that we executed each test case only once.

Furthermore, we observed that beyond a window size of 10, the results remained consistent. We attribute this behavior to the use of a relatively short text file, where larger window sizes did not provide additional benefits due to the limited amount of data being transferred.

5 Conclusion

In this project, we successfully implemented a reliable data transfer protocol over UDP using the Selective Repeat algorithm. Our experiments with varying window sizes (1, 10, 50, 100) and error rates (1%, 5%, 10%, 20%) demonstrated that window size significantly impacts transmission reliability, with sizes below 10 proving insufficient for reliable data transfer. While higher error rates increased retransmission frequency, the protocol maintained data integrity for window sizes of 10 and above. The implementation effectively handled connection establishment through three-way handshaking and managed packet timeouts, achieving reliable data delivery despite UDP's inherent unreliability. This project highlights the importance of proper window sizing in reliable data transfer protocols and demonstrates how selective repeat can overcome the limitations of an unreliable transport layer.