Project 2: File Transfer using RBUDP and TCP

GROUP 7

Group Members & Roles

- Thaakir Fernandez (group leader) 26479443@sun.ac.za
    - Sender, Experiments, Debugging, Report
- Priyal Bhana 27040607@sun.ac.za
    - Receiver, README.md, Makefile, Report
- Gideon Daniel Botha 26894319@sun.ac.za
    - GUI, Report
- Sulaiman Bandarkar 24234850@sun.ac.za
    - Experiments, Report

Project Introduction/Overview

This project implements Reliable Blast UDP (RBUDP), a protocol designed for efficient file transfer, and compares its performance with TCP (Transmission Control Protocol). The goal is to analyze the trade-offs between the two protocols in terms of reliability, speed, and packet loss handling.
This document elaborates on the implementations used for this project and covers all the experiments performed with their corresponding results. Additionally, the features and issues encountered by all group members is given.

AI Declaration

Priyal Bhana:
I used ChatGPT as a tool to help enforce my understanding of the theory behind RBUDP and TCP and gain knowledge of how data is received differently in each protocol. I initally had trouble understanding the role of TCP in RBUDP and used ChatGPT to explain the concept of signalling with regards to file transfer protocols. This helped me to build my own framework to use as a basis when beginning to implement the Receiver.

Gideon Botha:
While implementing the GUI for this project I made use of ChatGPT to suggest ways of styling the GUI to make it more user friendly (Gave me the idea of Inline CSS styling for customising things like font size, font color and effects like buttons changing colors on click events). I also used ChatGPT right at the start of my JavaFX journey to explain how some of the objects work (such as VBoxes and HBoxes) to improve my understanding of how the different elements interact with one another.

Thaakir Fernandez:

I used ChatGpt to explain the difference between RBUDP and UDP and why it uses TCP.

Sulaiman Bandarkar:
I used Claude AI to give me an idea on how to do the experiments.

Features

All features mentioned in the project specification rubric were implemented.

File Descriptions

- **Sender** - Responsible for transmitting files using either TCP or RBUDP protocols.
- **Receiver** - Idles and waits to receive files from the sender.
- **Main** - Merges the receiver and sender into a single unit and launches the GUI.
- **ExperimentRunner** - Main launcher for all experiments.
- **Experiment1** - Compares throughput of TCP and RBUDP with different file sizes.
- **Experiment2** - Tests different RBUDP burst sizes with a fixed file size.
- **Experiment3** - Tests different RBUDP packet sizes with a fixed file size.
- **Experiment4** - Tests RBUDP performance under different packet loss rates.
- **FileGenerator** - Generates test files for the experiments.

Program Description

General Description of Implementation

This program consists of three main components and the experiments:

1. **Main Application (GUI) [Main.java]**

- Prompts the user with a login window to obtain the current user's IP address.
- Instansiates a Receiver on its own thread.
- Builds the combined GUI consisting of a Sender side on the left and a Receiver side on the right.
- Manages all the events GUI input events (such as button clicks and pressing enter on TextFields) by mapping them to their corresponding backend actions.

2. **Sender [Sender.java]**

- Reads the file, splits it into packets, and transmits it using either TCP or RBUDP.
- Handles TCP transmission by sending file data in a sequential, reliable manner.
- Implements RBUDP transmission, where packets are sent in bursts and retransmissions occur for missing or late packets.
- Communicates with the receiver via signalling using TCP control messages to ensure delivery.

3. **Receiver [Receiver.java]**

- Listens for incoming files using both TCP and RBUDP protocols.
- If the protocol used is TCP, it receives and writes the file sequentially.
- If RBUDP is used, the receiver keeps the TCP control connection open for signalling while collecting incoming packets via a datagram socket.

- It tracks received packets and requests the retransmission of any missing, late, or dropped packets using negative acknowledgements (NACKs).
- Saves the received file into a designated folder.

4. **Experiments [ExperimenRunner.java, Experiment1.java, Experiment2.java, Experiment3.java, Experiment4.java, FileGenerator.java]**

- Made separate to Sender and Receiver.
- **ExperimentRunner** has a main launcher for all experiments.
- It provides a command-line interface to select which experiment to run.
- Takes the receiver's IP address as a required parameter.
- Created Experiment files 1-4.
- **Experiment 1** tested the throughput of TCP versus RBUDP.
- Transfers it using TCP and measures time/throughput.
- Transfers it using RBUDP and measures time/throughput.
- Records results in a CSV file (throughput_comparison.csv).
- **Experiment 2** tests the difference of transfer rate and RBUDP throughput.
- Test transfer rate by adding in timestamps for localhost to localhost.
- Test RBUDP throughput timing how long it takes to send a file over VPN.
- Recorded data manually in CSV file (transfer_rate.csv).
- **Experiment 3** tests different RBUDP packet sizes.
- Creates a custom sender with the specified packet size.
- Measures transfer time, throughput, packet count, and retransmissions.
- Records results in a CSV file (packet_size_performance.csv).
- **Experiment 4** tests RBUDP performance under different packet loss rates.
- Creates a sender that simulates the specified packet loss rate.
- Measures transfer time and throughput.
- Calculates success rate.
- Records results in a CSV file (packet_loss_performance.csv).
- **FileGenerator** generates test files for the experiments.

The program starts when you boot up the GUI. You are then prompted by the main class to enter your IP and Username. Once you've entered that, the main class then initializes the Reciever file to start listening for files.

When you are ready to send a file, you select the file you wish to send and the main class will initialize an instance of the Sender class to send the file to the Receiver class. The Sender class takes in the filename and reads its contents into bytes to be sent either via UDP using datagram packets or via TCP using a socket. Once the file has been sent by the Sender class, the Receiver class reads in any data being sent by the Sender class.

In the case of UDP, the Receiver listens for communication via the TCP sockets and packets via the datagram sockets. If any packets get lost the Receiver will signal to the Sender that those packets need to be retransmitted. Once all the data has been received, the Receiver class signals to the then saves it into a file in the src directory and signals to the Sender that all is well and that it can now close all sockets.

Experiments

1. Experiments that compare the throughput of TCP and RBUDP data transfer.
    - Question:
      How do TCP and RBUDP protocols compare in terms of throughput when transferring files of different sizes?

    - Hypothesis:
      RBUDP will provide a higher throughput than TCP.

    - Variables:
      The Independent variable is the different-sized files being sent..
      The Dependent variable is the throughput.

    - Method:
      Generate text files of different sizes.
      Transfer each file via TCP.
      Transfer each file via RBUDP.
      Record data.

    - Results:

File Size (KB)	TCP Throughput (KB/s)	RBUDP Throughput (KB/s)	TCP Time (s)	RBUDP Time (s)
100	150.3759398	11.85817621	0.665	8.433
500	495.5401388	28.86669361	1.009	17.321
1000	698.3240223	36.18075907	1.432	27.639
5000	633.7938902	34.97041503	7.889	142.978

    - Conclusion:
      Over the files we tested, TCP was faster than RBUDP because the files we experimented on were small. The hypothesis is incorrect. The hypothesis would be correct if RBUDP was used with larger files as it would be sending many more packets than TCP trying to fill the pipe.

2. Experiments with the transfer rate of RBUDP.
    - Question:
      What is the difference between the theoretical transfer rate and the actual throughput for RBUDP?

    - Hypothesis:
      The theoretical transfer rate will yield much better performance than the actual throughput of RBUDP.

- Variables:
  Independent variable is the different sized input files.
  Dependent variables would be the transfer rate and the RBUDP throughput.

- Method:
  Generate different sized files.
  First tested from localhost to localhost to get the theoretical transfer rate with the generated files.
  Then tested over VPN to simulate a real life scenario.
  Record the data generated in both scenarios.

- Results:

File size (KB)	Transfer Rate (KB/s)	RBUDP Throughput (KB/s)	Transfer Rate Time (s)	RBUDP Time (s)
100	46.75473148	11.85817621	2.138820967	8.433
500	79.74272483	28.86669361	6.27016447	17.321
1000	80.16211093	36.18075907	12.47472139	27.639
5000	86.10906758	34.97041503	58.06589411	142.978

- Conclusion:
  The results show that the theoretical transfer rate from localhost to localhost is much faster than the actual RBUDP throughput over a VPN. This was expected as the theoretical transfer rate has no obstructions whereas the RBUDP throughput has to be sent over the VPN and Wifi which has some delay.

3. Experiments with the packet size used in RBUDP.
   - Question:
     Does varying the packet size affect the throughput of bytes per second for different files?

   - Hypothesis:
     As the packet size increases the throughput will increase and the transfer times will decrease.

   - Variables:
     The Independent variable is the packet size because it was changed to see the effects on the other measurements.
     The dependent variables are the transfer time and the throughput.

   - Method:
     Generated 2 input files.
     Ran each one with different packet sizes and timed their transfer times to be recorded.
     Record the data.

- Results:
  <u>File of 2000KB</u>

Packet Size (bytes)	Transfer Time (s)	Throughput (B/s)
1000	25.88044173	38.63921684
2000	29.73686263	67.25659075
5000	113.0458258	44.22985071
10000	24.38369855	410.1100569

  <u>File of 11000KB</u>

Packet Size (bytes)	Transfer Time (s)	Throughput (B/s)
1000	146.6775387	6.817676442
2000	304.2880872	6.572718697
5000	110.451493	45.26874072
10000	102.2161312	97.83191638

- Conclusion:

  The result for the 2000KB file, all the data is out of order because making a packet size more than the file size gives obscure results. It cannot send more bytes than how big the actual size of file is.

  The results for the 11000KB file, reflects the hypothesis for the throughput increasing as the packet size increases but the transfer time has an anomaly for packet size 2000 bytes.

  In conclusion, packet size does increase throughput and decreases transfer time only when the transfer file size is greater than the packet size.

4. Experiments using increasing packet loss rates.
   - Question:
     How does increasing packet loss rates influence RBUDP?

   - Hypothesis:
     The higher the packet loss rate, the lower the success rate would be. It will also decrease transfer time and the throughput..

   - Variables:
     Independent variable is the packet loss rate.
     The dependent variable would be the packet success rate.

- Method:
  1000KB transfer file generated.
  The experiment runs with different loss rates.
  Record data.

- Results:

Loss Rate (%)	Transfer Time (s)	Throughput (KB/s)	Packets Sent	Retransmissions	Success Rate (%)
0	36.049	27.74002053	3289	1072	67.40650654
1	45.901	21.78601773	3978	1761	55.73152338
10	62.888	15.90128482	3802	1585	58.31141504
50	134.855	7.415372066	4956	2739	44.73365617
100	193.526	5.167264347	35472	35472	0

- Conclusion:
  As the results show that increasing the packet loss rate really affects the transfer time, throughput and success rate negatively.

5. Experiments using increasing burst sizes.
   - Question:
     How is RBUDP affected by changing the burst size?

   - Hypothesis:
     As the burst size is increased then the throughput will increase.
   - Variables:
     Independent variable is the burst size.
     Dependent variable is the throughput.

   - Method:
     Generate test files.
     Create a CustomBurstSizeSender with the specified burst size.
     Measure transfer time, throughput, and retransmissions.
     Record the data.

   - Results:

Burst Size (bytes)	Transfer Time (s)	Throughput (KB/s)

50	101.292	9.872447972
100	54.202	18.44950371
200	54.221	18.44303868
400	35.091	28.4973355
800	28.659	34.89305279
1600	27.786	35.98934715

- Conclusion:
  The hypothesis is correct because as the burst size increases, the transfer time decreases and the throughput increases.

Issues Encountered

- It took us some time for us to understand what the project was about. The project specification was to vague with regards to how RBUDP differs from TCP.
- We encountered threading issues within the Receiver class when we attempted to have the TCP and UDP listening simultaneously - a method we eventually scrapped.
- We struggled with the design choice on the GUI implementation - whether to have a combined GUI implementation of Sender and Receiver or to have them on separate GUI implementations.
- We had Git issues that caused the working transferring of files to start to malfunction because members in the group had different commits and versions of the file.

Design

- We decided to have a combined GUI implementation for the Sender and Receiver.