Report 2: Reliable Blast User Datagram Protocol (RBUDP)

Department of Mathematical Sciences
Computer Science Division
University of Stellenbosch
7600 Stellenbosch
August 2022

1 INTRODUCTION

There are many protocols that can be used to transfer files over a network. In this project, you will implement RBUDP, a protocol for data transfer. You will measure its performance in comparison with TCP which in turn will help you understand the differences and trade-offs between these protocols and how to analyse them.

2 PROJECT MEMBERS INFORMATION

GROUP MEMBERS:
[Takunda, Charles, Mudima, 23969156]
[Ben, Sibusiso, Mahlangu, 23629428]

INTRODUCTION / OVERVIEW

This program aims to show how files are transferred between a server and a client using two main protocols for file/data transfers namely the RBUDP, which uses DatagramSockets and DatagramPackets within its domain to send and receives files and TCP, which uses the basic Socket to establish connection between the sender and receiver so to exit the process of file/data transfer. It constitutes of two main classes namely the Send.java and Receive.java. These files work concurrently to establish a network of receiving and sending files/data of any type between online users.

UNIMPLEMENTED FEATURES

- Handling lost packages when sending via RBUDPs: The current program does not cater for any lost packages during the process of sending and receiving data.

ADDITIONAL FEATURES IMPLEMENTED

- A pop-up GUI to indicate that the receiver's side is ready to receive any data from the sender

DESCRIPTION OF FILES

Our project contains the following two files:
***Receiver.java*:** contains different functions for each protocol (RBUDP and TCP) that they use to receive and detect any available signal from a sender that is ready to send any data and reads the bytes sent to display the correct received file.
Send.java: contains several functions for each protocol (RBUDP and TCP) that they use to send data and for UDP, a special function that caters for sequences numbers.

EXPERIMENT 1

TITLE: Comparing the throughput of TCP and RBUDP

DESCRIPTION: Throughput is the number of successfully received packets in a unit time and it is represented in bps. TCP attempts to provide both an efficient utilization and a fair share of network resources. UDP is a fast transmission protocol used by most of the real-time applications as it is suitable for delay sensitive applications like video and audio transmission. In this experiment we compared the performance of TCP and UDP through.

HYPOTHESIS: TCP is considered one of the most important protocols on the Internet, whereas UDP is a minimal message-oriented Transport Layer protocol. UDP is a much faster, simpler, simpler, and efficient protocol.

VARIABLES:

1. **Independent variables**: Throughput values
2. **Dependent variables:** TCP and UDP

METHOD

This experiment was carried via two terminals where one terminal had the receiver, and the other terminal had the sender. Our program only allows for one file transfer to be sent, at a time, either using UDP or TCP. This then meant that we had to send files separately using the different modes of data transfer. The throughput values were then measured and recorded, after transmission was completed.

Server/ Client	CPU	Memory	DISK speed	OS
Server	Core i3	4GB	250 MB/s	Linux
Client	Core	4GB	250 MB/s	Linux

Both the client and the server were being run one terminal hence one computer, that is why we have the same specifications for both the server and client.

RESULTS

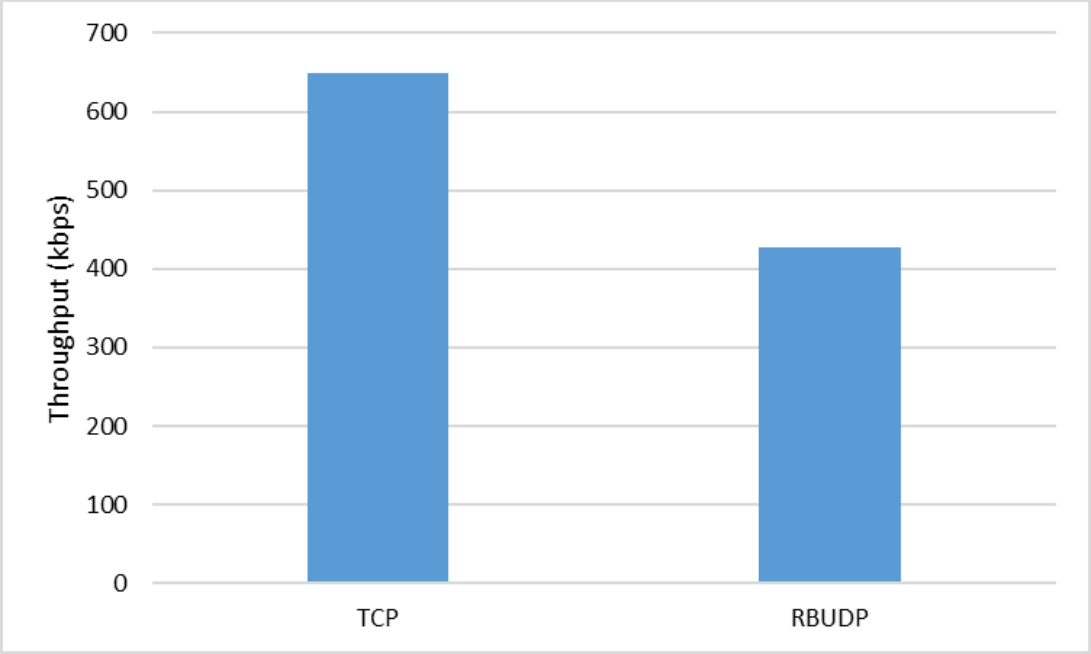


Figure 1Throughput comparison of TCP and UDP

CONCLUSION

It was noted that TCP had a higher throughput in comparison to RBUDP. TCP throughput comes 649 kbps and UDP throughput comes 428 kbps. With RBDUP user has to measure the available bandwidth before transfer. In study throughput is a main comparative metric.

EXPERIMENT 2

TITLE: Transfer rate of RBDUP

DESCRIPTION: In this experiment we transferred different types of files using the UDP method to test the rate which these files would be transferred

HYPOTHESIS: UDP packets are transmitted at a rate of 64 Kbps here, and 80-byte UDP packets are transmitted at a rate of 100 packets/sec.

VARIABLES

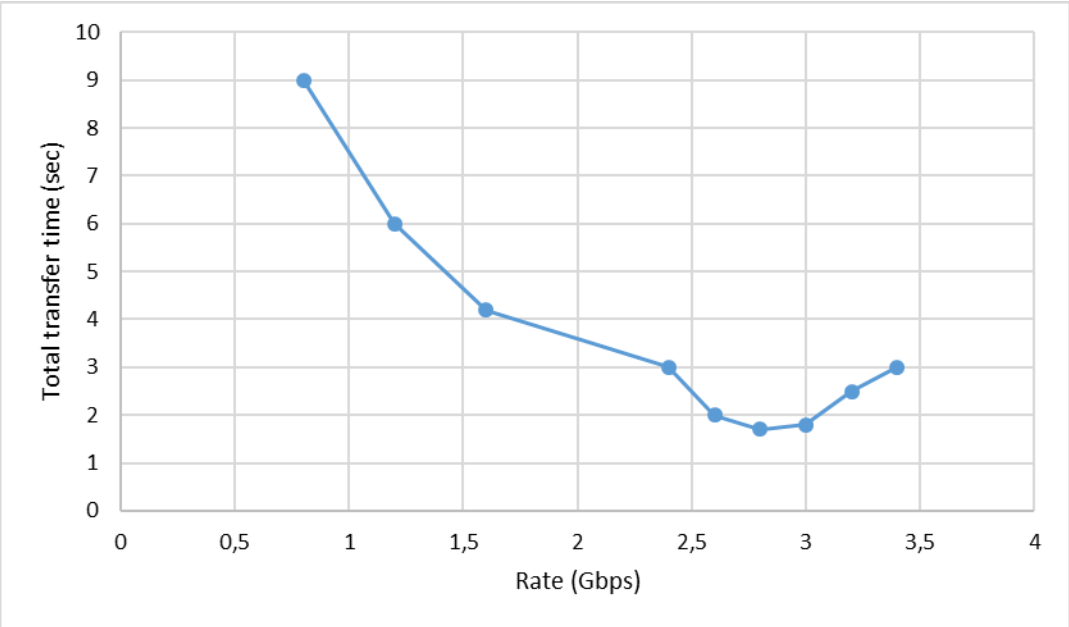
1. **Independent variables**: Total transfer time (ms)
2. **Dependent variables:** Rate

METHODS

We transferred different file size ranging from 2kb-2mb. We carried our time measurements based on when the file was fully received in the directed folder.

Rate (Gbps)	Total transfer time
0,8	9
1,2	6
1,6	4,2
2,4	3
2,6	2
2,8	1,7
3	1,8
3,2	2,5
3,4	3

RESULTS



CONCLUSION

The results show a gradual steady decrease in time as this was noted when file sizes where changed. Our UDP would stop sending data once an IO Exception occurred, and this resulted in the sender repeatedly attempting to send the data again.

EXPERIMENT 3

TITLE: Experiments with the packet size used in RBUDP

HYPOTHESIS: As we increase the number of packets size used, the rate at which the files are sent and received increases also.

VARIABLES

1. **Independent variable**: Rate of transfer files(mb/s)
2. **Dependent variable**: packet size(bytes)

METHOD

This experiment was carried out through using different packet sizes ranging from 0 byte to 893440 bytes. The rate of transfer was recorded each time when a packet size was sent through

RESULTS

Packet size(bytes)	Rate of transfer files (Mbps)
0	12.6
0.27	22.5
57	12.4
23884	8.5
65504	6.7
111698	10.4

CONCLUSION

If the packet size is less than 8 bytes, the UDP will not send that data through since is below the minimum packet size which is 8 bytes thus the rate of transfer cannot be measured, any packet size in the range of eight and 65535 bytes are sent through and the rate of transfer is slightly around the same rate and lastly, packet sizes that are greater than 65535 bytes, some bytes are not read in so it is inconclusive to give a transfer rate there due to lost bytes

EXPERIMENT 4

TITLE: Experiments using varying packet loss rates. If you are not comparing RBUDP and TCP, you may use random drop. Alternatively, if you are comparing TCP and RBUDP, you must artificially cause congestion in the network

HYPOTHESIS

UDP is more likely to have a higher packet loss rate in comparison to the packet loss of TCP due to multicasting data simultaneously

VARIABLES

1. **Independent variable**: number of files sent simultaneously
2. **Dependent variable**: packet loss rate (%)

METHOD

The methodology behind the experiment was to use a software called Wireshark to do the detailed analysis on the packet loss rate for the UDP and TCP. So, the software was run, and data was record as we increase the number of files sent simultaneously, firstly a file was sent, then two files, followed by four files so forth simultaneously

RESULTS

For TCP, the packet loss rate when a file was sent was at 0.03%, when two files were sent it was 0.02 and for four files it was at 0.04%, while for UDP for the first file was at 4, 06%, when two files were sent it was at 6,07% and when four files when sent it was at 8,97%

CONCLUSION

The packet loss rate was found to be significantly small close to 0% when using the Transmission Control Protocol(TCP) mainly because it has congestion control system that resends data when it was not fully transferred and the ability to maintains its stable state during the process of sending and receiving files .On the other hand, the RBUDP packet loss rate increases as we increase the number of files sent simultaneously due to not having a congestion control system.

ISSUES ENCOUNTERED

- The program failing to send and receive multiple files on one run, like to send and receive again, you will need to close the current running process and start again.
- When sending via RBUDP, the files sent is the same as the file received physically but when generating a files id data, it shows different file ids (that long codeword)

SIGNIFICANT DATA STRUCTURES

1. An array data structure, this simplified the way we send and receive the bytes of data
2. DatagramSockets, that simplified the way for sending and receiving files
3. DatagramPackets, that simplifies the way handle lost packages and re sending them when a corrupted file was detected.

COMPILATION

1. Firstly, ensure that you have the files in the same folder/directory
2. Run javac *.java to compile all the files
3. Java Receive first to show that you are ready to receive some files
4. java Send to choose a file you want to send

EXECUTION

1. After compiling the files as directed form the previous sub section
2. Java Receive first to show that you are ready to receive some files
3. Choose which file protocol you want to use to receive the file by typing 1 for TCP and 4 for UDP in the command line
4. java Send to send files
5. Choose which file protocol you want to use to send the files from the provided buttons on the GUI
6. Then your file will be sent, and the receiver will display a pop up GUI with the filename that was sent.
7. If send via UDP, on the command line, the transaction between sequence numbers will be displayed.

REFERENCES

Computer Network Lab (2021) *udp file transfer.* [online video] Available at: https://youtu.be/ezblDWct3Gs. [Accessed 14/08/2022]

GeeksforGeeks (2021). *Datagrampackets Programming in Java.[*online].GeeksforGeeks. Available at: https://www.geeksforgeeks.org/datagrampackets-programming-in-java/?ref=lbp. [Accessed 10/08/2022]