PORTFOLIO 2

Baldur Benedikt Bårdsson Indrevoll, s364586

Hiwa Abdolahi, s364547

DAFE2410

Contents

Introduction

In this portfolio we have been tasked with creating and implementing a simple transport protocol, also known as “DATA2410 Reliable Transport Protocol” (DRTP). It is supposed to provide a reliable data delivery on top of UDP. In other words, our protocol is meant to ensure that data gets transferred reliably, in-order and without missing any data or having duplicates.

We will be implementing these to programs to start with:

DRTP, that will provide the reliable connection despite using UDP.
A file transfer application. This application will have a simple file transfer server as well as a simple file transfer client.
At last we are to make a report on the task which is what you are reading now.

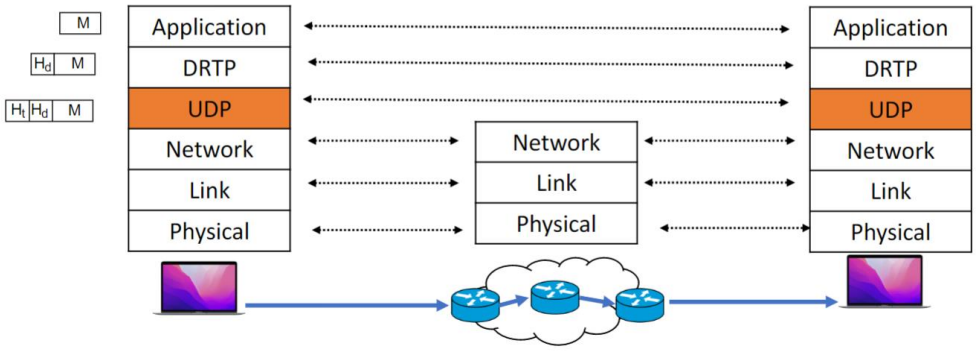
Background

In this task we will be using vscode for coding the different programs DRTP and the file transfer application. We will also be using mininet for running our code and testing it. For this we use the virtual machine “Oracle VM VirtualBox”. We have downloaded the topology from the course module “portfolio2”. As we are only two members in the group, we have met up regularly to work together on the task while also dividing the task into two when we are working from home. We have put the labs to good use to ask for guidance. We recognized that we could use some of the code from our Portfolio 1 as inspiration some places as well. We will be implementing three reliability functions for the user to chose from. These are stop_and_wait(), GBN() and SR(). We do this so that the user has more options to choose from depending on the type of file it is trying to send.

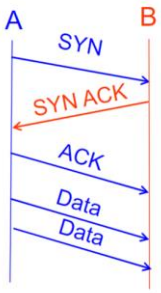
As mentioned above, we will be implementing DRTP and the file transfer application which we will be going more into detail in the next section “Implementation”.

Implementation

Starting with DRTP, our goal is to add a reliable transport over the UDP layer. The image below pictures this goal.



Similarly to Portfolio 1 we establish a connection by sending messages back and forth from server and client. Here we use a so called “Three-way handshake” to establish the connection. The sender sends a SYN to the receiver. The receiver will the get this message and send back an acknowledgment of the SYN (SYN ACK). The initial sender then sends an Ack as acknowledgement of the SYN ACK. The connection is established and data may start transferring. If however the sender does not receive a SYN ACK after sending SYN to the receiver an error is thrown.



The file transfer application (application.py) will either be invoked from the client side with the use of the -c flag, or invoked from the server side with the use of the -s flag.

The server side goes as following:

The file transfer server will receive a file that has been sent from a client using a DRTP/UDP socket and will then write this file to the file system. As it is written in the README file the arguments given are the IP address, the port number that the server is listening on and the reliability function.

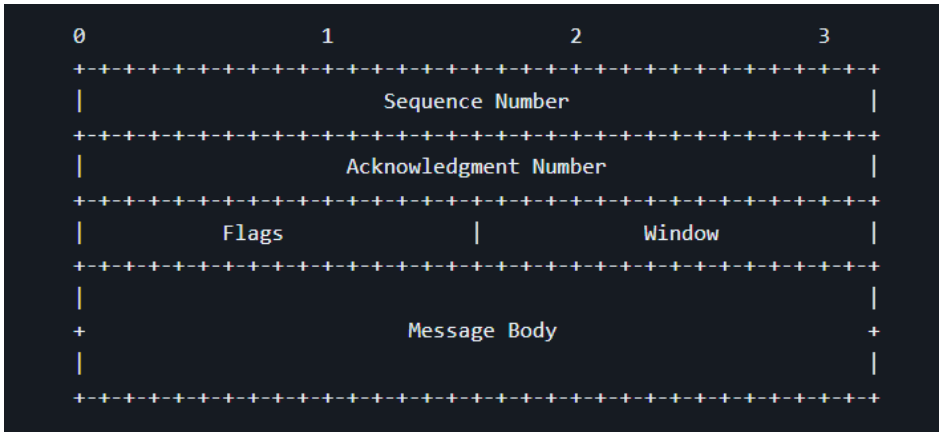
The client side goes as following:

The client will read a file from the computer’s file system and send this file to the server side using the beforementioned DRTP/UDP. In this case the arguments given include the file name, the server address and the port number. As stated in the task, it is possible to transfer two source files at the same time, however, we will not be focusing on that in this task. Therefore we will only be allowing one transfer at a time.

Furthermore, before the sender sends data over UDP it adds a header in front of the application data. The length of this header is 12 bytes, which makes for a total of 1472 bytes sent by the sender (1460 bytes of application data). This header that has been added contains a sequence number (packet

sequence number), an acknowledgment number (also known as packet acknowledgment number) and, flags and a receiver window.

Together with the header this is what it will look like:



An important note for this packet is that if the header does not contain any data whatsoever, it is an acknowledgement packet (ACK).

As said earlier; The sender and receiver perform a three-way handshake (similar to TCP) to establish a secure connection. After the sender sends an empty packet with the syn flag ON, the server responds with a packet with the SYN and ACK flags set, establishing the connection. After receiving a syn-ack packet, the sender sends an ack to the server.

When the transfer is finished, the sender sends a FIN packet. After receiving the FIN packet, the receiver sends an ACK to terminate the connection. When the sender receives the ACK, it also graciously terminates the connection.

The three different reliability functions have their own use, this is what each of them do:

Stop and wait function

The sender sends a packet and waits for an acknowledgement before sending another. If an ack is received, it transmits a new packet. The packet is resent after waiting for timeout if an ack is not received. If the sender receives a duplicate ACK (ack of the most recent received sequence), the packet is resent.

Go-Back-N function

The sender transfers data using a specified window size of 5 packets as part of the Go-Back-N method. Sequence numbers are used to represent packets; for example, packet 1 is numbered 1, packet 2 is numbered 2, and so on. If no ACK packet is received after the specified delay period, all unacknowledged packets are retransmitted, which can be adjusted to a default value of 500 ms using the socket.settimeout() method. Because a receiver delivers data in the correct order, if packets arrive out of order, there has been packet loss or network reordering. In such cases, the DRTP receiver should not acknowledge anything and should discard these packets.

Selective-Repeat function

Instead of discarding packets that arrive in the wrong order, they get placed in the receive buffer. To achieve the best results, a mix between SR and GBN is used.

Discussion

After running some tests for the different functions; SR(), GBN() and S

Conclusions

This project was demanding for being only two people working on it. We spent many labs working it out, but we might have needed even more time after all. We encountered some problems here and there where the code would not run because of either syntax error or just wrongly written code. An example being our error while writing the arguments and the reliability functions. In the end I think we managed to make it work despite some hindrances. All in all it was a good learning experience. Next time it might be better to find a few more people to work with so that the work load doesn't get so heavy on each of us.

References

- Pictures taken from 'Portfolio-2-guidelines.pdf'
- https://github.com/safiqul/Portfolio-2