




MAY 16, 2023

PORTFOLIO 2  
DATA2410

SOBAN ALI & SANDER HARALD SÆVIK  
S362045 & S362067  
Oslo Metropolitan University



1	INTRODUCTION	3
2	BACKGROUND	4
3	IMPLEMENTATION	4
4	DISCUSSION	5
5	CONCLUSIONS	6
6	REFERENCES	6

# 1 Introduction

In this project, we need to create a reliable transport protocol called DRTP that will make sure that data is transmitted without any errors, missing data, or duplicates over the UDP network. To achieve this, we need to develop two applications: a file transfer server and a file transfer client. The client application reads a file and sends it to the server application using the DRTP/UDP network, where the server writes the received file to the file system.

To make this possible, we also need to implement three different reliability functions, namely Go-Back-N (GBN), Selective Repeat (SR), and Stop-and-Wait (saw). Users can choose any of these reliability functions as per their requirements. Overall, this project requires us to create a robust and efficient system that can securely transmit files over the network without any errors or loss of data.

The DRTP protocol includes a custom header that adds 12 bytes to the application data, where the header contains a sequence number, an acknowledgment number, flags, and a receiver window for flow control. The DRTP packet is sent over UDP, and each packet includes 1460 bytes of data, plus 12 bytes of the custom DRTP header. The DRTP protocol supports two types of packets: data packets, which include the application data, and acknowledgment packets, which confirm the receipt of data packets up to a specified sequence number. The DRTP protocol supports three flag bits, which include the SYN, ACK, and FIN flags, and a reset flag. The three-way handshake is used to establish a reliable connection, and the connection is closed gracefully using the FIN and ACK packets.

The file transfer client application sends the file over the network to the file transfer server application, which writes the file to the file system. The file transfer client application supports two test cases: packet loss and normal operation. The file transfer server application supports two test cases: skipping an ACK to trigger retransmission at the sender-side and normal operation. The file transfer client and server applications use the same reliability function chosen by the user.

Overall, the project requires the implementation of a custom reliable transport protocol (DRTP) on top of UDP and the development of two applications, file transfer client and file transfer server, to transfer files over the DRTP/UDP network. The project supports three reliability functions: Go-Back-N (GBN), Selective Repeat (SR), and Stop-and-Wait (SW). The project also supports two test cases for the file transfer client and server applications, which include packet loss and normal operation, and skipping an ACK to trigger retransmission at the sender-side and normal operation, respectively.

In this project, we need to create a reliable data transfer protocol (DRTP) on top of UDP and develop a file transfer application comprising a client and server component. We will create a custom DRTP header and establish a three-way handshake protocol for connection setup and termination. To ensure reliability, users can select from three reliability functions, namely Go-Back-N, Selective Repeat, and Stop-and-Wait. We will pay close attention to details such as sequence and acknowledgment numbers, window size, and flag fields.

In summary, this project involves designing and implementing a reliable data transfer protocol and file transfer application, while meeting specific requirements and constraints.

## 2 Background

The project involves implementing a reliable data transfer protocol called the DATA2410 Reliable Transport Protocol (DRTP) on top of UDP. The protocol ensures reliable data delivery in order without missing data or duplicates. The implementation involves two programs: DRTP, which provides reliable connections over UDP, and a file transfer application comprising a simple file transfer client and server.

Sender adds a 12-byte header to the application data before sending it over UDP, containing the sequence number, acknowledgment number, flags, and receiver window.

A DRTP packet without data is only an acknowledgment packet. If the packet contains data, it also includes a message body. The flag field comprises the SYN, ACK, FIN, and Reset flags.

When two devices want to establish a reliable connection, they use a three-way handshake. The sender sends a packet with the SYN flag on, and the receiver sends a packet back with the SYN and ACK flags set. This lets the sender know that the connection has been established, and they send an ACK to confirm. When the transfer is done, the sender sends a FIN packet, and the receiver responds with an ACK to confirm it received the FIN packet, then closes the connection.

## 3 Implementation

First, we start by handling the modules we need to use, first we have socket for communication then struct for the binary data, we then need to use argparse for -c and -s in client and server. Sys is for the functionality, meanwhile time is needed for time purpose tasks. Random is there for random numbers.

Then we have the argparse command lines. We have c for client, and s as server as mentioned previously. Then we have l and p for the server address and the host address. Moving on we have -r as the reliable method we are going to use whether it is stop and wait, selective repeat or gbn.

We firstly start by declaring Header, and max data length, and also the header format. Then we will start on the function. The function create\_packet takes in several values: ``sequence_number``, ``acknowledgment_number``, ``flags``, ``data``, and ``window``. If ``flags`` is not provided, it sets default values for SYN, ACK, and FIN flags. It then calculates the flag value using bitwise operators. The function also calculates the length of the data, creates a header using the ``struct.pack()`` method-

First of all, we have the gbn\_client function which creates a Go-Back-N (GBN) client that establishes a connection with a server using a three-way handshake. It then sends packets to the server in the order of their sequence numbers and waits for ACK packets. If an ACK packet is not received within a certain timeout period, the client retries the unacknowledged packets.

Then we have the sr\_client function. The selective repeat starts by creating a socket and setting some variables, such as the sequence number, expected acknowledgment number, bytes sent, and window size. It then sends a SYN packet to the server and waits for a SYN-ACK packet in response. Once it receives the SYN-ACK packet, it sends an ACK packet to the server to establish a connection.

When the connection is established, the function sends each packet to the server and waits for an acknowledgment. If an acknowledgment is not received within a specified timeout, it retransmits the packet. If a packet is successfully acknowledged, it updates the sequence number and bytes sent.

Moving on we have the last client which is the stop and wait client. The "Stop-and-Wait" protocol sender sends only one packet at a time and waits for an acknowledgment (ACK) from the receiver before sending the next packet.

In this implementation, the client sends a SYN packet to initiate a three-way handshake with the server. Once the handshake is established, the client starts sending data packets to the server. For each data packet, the client waits for an ACK packet from the server before sending the next data packet. If the ACK is not received within a certain time period (timeout), the client retransmits the packet.

Finally, the client sends a FIN packet to close the connection with the server and waits for a FIN-ACK packet from the server. Once the FIN-ACK packet is received, the client sends an ACK packet to confirm the closure of the connection and then closes the socket.

Lastly, we have the server which does;

- Server makes a UDP and waits for the SYN packet to start a three way handshake.
- When SYN is perpetrated, the server will send SYN-ACK to the client to perform a three-way handshake.
- When the handshake has gone through the server waits for data packets and sends ACK packets.
- Server keeps in mind the expected seq number for the next data and deletes older packets.
- When a FIN packet is received, the server sends a FIN-ACK packet to the client to complete the three-way handshake.
- When FIN is perpetrated, the server will send a FIN-ACK packet to the client to complete the handshake.

## 4 Discussion

When it comes to transferring files, there are different protocols that can be used. In this case, the Go-Back-N protocol is faster than the Stop-and-Wait protocol.

In network protocols such as GBN, there is a concept referred to as window size. This value represents the number of unacknowledged packets that can be sent out before an acknowledgement is requested from the receiver. For instance, if the window size is set to 5, then the sender may transmit up to 5 packets without needing to wait for corresponding acknowledgments. This technique helps promote faster file transfer by reducing any idle time of the network.

Network protocols such as GBN rely on acknowledgment (ACK) to ensure successful data transmission. A skipped or lost ACK can cause problems with the protocol. For example, if acknowledgments are lost, the sender does not know if the receiver received all packets successfully. This can cause the sender to resend the packet, causing a delay in transmission. In the GBN protocol, the sender expects her BACK from the receiver for each packet sent. ACK helps the sender decide which packet to remove from the buffer and send a new packet. If no ACK is received, the sender assumes the packet was lost or not

received by the receiver, triggering retransmission of the unacknowledged packet or the entire window, depending on the implementation.

When packets are sent over a network, they are usually numbered in sequence so that they can be properly classified when they reach their destination. However, sometimes packets are intentionally skipped or dropped due to network problems. This can cause packets to arrive out of order and can affect transmission reliability and performance.

For example, protocols such as Go-Back-N and Selective-Repeat typically send packets with consecutive sequence numbers. If a packet is skipped, the sender will not receive an acknowledgment until the missing packet is received. On the other hand, the receiver may need to buffer out-of-order packets until the missing packet arrives. This can introduce latency and increase the overall latency of transmissions.

To handle out-of-order delivery, protocols such as GBN and selective repeat use mechanisms such as sequence number tracking and buffer management. These mechanisms help the receiver to reconstruct the transmitted data in the correct order, even if the packets arrive out of order. However, frequent out-of-order or skipped sequence numbers can increase buffer requirements and cause buffer overflow problems.

Overall, understanding how protocols deal with out-of-order delivery is critical to ensuring reliable and efficient data transmission over networks.

## 5 Conclusions

Simply put, packets are like the communication building blocks in network protocols. They are used to transmit information between internet-connected devices such as computers and servers. When you initiate a communication session with a server, your device sends her a special packet called a SYN packet. The server responds with her SYN-ACK packet confirming it is ready for communication. Data is then sent and received between the device and the server in the form of packets. Each packet contains a unique sequence number and confirmation number to ensure that everything arrives safely. Each time a packet is sent from the device to the server, the server sends back an ACK packet to confirm receipt of the packet and request the next packet in sequence. Once all the data has been transferred, the device sends her FIN packet to signal the end of the session, and the server responds with her FIN-ACK packet to confirm that the session has ended. Packet-based communications are very important for being able to exchange data reliably and efficiently over long distances, and are therefore an important part of the Internet.

## 6 References (Optional)

safiqu. (2023). *2410/header.py at main · safiqu/2410*. GitHub.

<https://github.com/safiqu/2410/blob/main/header/header.py>

safiqu. (2023, April 28). *safiqu/Portfolio-2*. GitHub. <https://github.com/safiqu/Portfolio-2>