Name: **Akshaan Kakar**
ID: **804029538**

Name: **Justin Hamilton**
ID:

CS 118 Project 2 - Go-Back-N over UDP

1. Description of Protocol:

For this project, we implemented the Go-Back-N protocol atop the User Datagram Protocol. Go-Back-N is a reliable pipelined protocol which used a congestion window of a specified size along with cummulative ACKs from the receiver to indicate the receipt of a package. The window size can be specified to the server program in our implementation. We defined a structure called a **segment** which carried the packet data along with header information. The segment had the following members:

```
struct segment {
     int seq_no;
     int ack_no;
     int fin;
     int data_len;
     char* data[MAX_PACKET_SIZ];
}
```

- `seq_no` was used to hold the sequence number for the segment.
- `ack_no` was used to hold the acknowledgement number for the segment.
- `fin` held 1 or 0 to indicate whether the packet was a fin packet or not respectively.
- `data_len` held the size of data in the segment.
- `data[MAX_PACKET_SIZ]` was a buffer for the actual packet data.
- `MAX_PACKET_SIZ` was defined to be 1024 bytes

The overall description of the steps followed by our protocol is as follows:

- The server instantiates a UDP socket and is ready to accept UDP datagrams.
- The client sends a request to the server on a specified port. The data in this segment is the name of the requested file.
- The server splits the file into a vector of segments and then sends the first window of segments to the client and starts a timer.
- The client returns ACK segments for each packet received.
    - If the packet received by the client is marked lost or corrupted, the client sends the last in order packet's ACK back to the server.
- The server receives ACKs and moves the window forward for each ACK and sends the next packet in the vector and resets the timer.
    - If the ACK received by the server is lost or corrupted, the server waits for a higher numbered ACK since ACKs are cummulative. If no higher number ACK is received before a timeout of 5s, the server re-sends all the packets since the last in order ACKed packet.
- When the last packet is ACKed by the client, the server sends a segment with fin set to 1.
- The client responds with a FINACK.
- The server responds with another FINACK and the connection is closed.

<u>2. Difficulties faced:</u>

- The program logic was hard to follow at times, due to the asynchronous behavior introduced by Non-Blocking I/O and the select() calls used to implement the timer.
    - This was solved by visualizing the state machines that represent the GO-Back-N protocol.
- The program was hard to debug since both the client and the server had to be run simultaneously and so errors were hard to localize.
    - This was solved mostly by dumping output to standard out and checking where the issues were.

<u>3. Compiling:</u>

The server and client can be compiled using the **make** command. The server and client can be run as follows:

• Server (sender): ./server port_number congestion_window_size prob_loss prob_corruption

• Client (receiver): ./client server_hostname server_port_number filename prob_loss prob_corruption

.