UDP-RDT-with Selective Repeat-Protocol

	Member Names	ID Nos.
Group A	Nipun Wahi	2018A7PS0966H
	Hrithik Kulkarni	2018A7PS0278H
	Ameetesh Sharma	2018A7PS0167H
	Mir Ameen Mohideen	2018A7PS0487H
	Nielless Acharya	2018A7PS0207H
Group B	Swapnil Agarwal	2017B3A71343H
	Thribhuvan Reddy M	2017B3A71116H
	Arundhan Reddy M	2017B3A70889H
	Uttam Singh	2017B4A70683H
	Srinivas Konduri	2017B3A70746H

Assumptions:

We are implementing file transfer using UDP as the transport layer protocol, where we have to ensure the reliability of data transfer, which is a significant apprehension.
Here are some salient points of our implementation:

- The file will be divided into packets of fixed size, which is pre-defined
- The sender sends n number of packets at a time with sequence numbers 1,2,3….n
- Selective repeat ARQ will be in the process once a packet departs from the sender
- The receiver will send an acknowledgement for the first non received packet
- Packets received ahead of time will not be stored in a buffer at the receiver's end
- After receiving the acknowledgement from the receiver, the sender will then either retransmit the missing packet or send the next series of packets, depending on the case.
- UDP checksum will be calculated and tallied to ensure that the packet is not corrupted. Retransmission of packets will happen if it is corrupted.
- Timeout to receive an acknowledgement is fixed and assumed to be 2 seconds for the sake of simplicity.

Working:

- The initial connection between the client and the server will be through the 3-way-handshaking protocol.
- First, the client sends a syn packet and waits for the synack packet to be received. Until then, it will keep sending syn to the server after a fixed amount of time.
- When it receives the synack packet from the server, the client will stop sending syn and send an acknowledgement. The server, until it receives the acknowledgement will keep sending synack packets. Clients would handle duplicate packets, but acknowledgement for all of them would be sent.
- The client will then send a message to the server. The packet will contain the following segments:
    - Sequence number
    - ACK number
    - Flag (to tell between the first packets and the actual data packets)
    - Checksum
    - Data packet
- The encoding of this packet will be as follows
    - The first 1 byte would be a flag and would be treated as uint8.

    - The following 4 bytes would be sequence number and would be treated as uint32
    - The following 4 bytes would be ack number and would be treated as uint32
    - The following 4 bytes would be the size of the payload and would be treated as uint32
    - The last 1 byte would be the checksum and would be treated as uint8
    - All of the above headers would be in Big Endian.
    - The checksum would be calculated by XORing all the bytes in the packet(including data payload) except the checksum
    - Next would be the payload or data, which would take the size number of bytes
- The 1-byte flag would signify the type of packet, We have chosen five flags, 0 would signify syn packet, 1 would signify ack packet,2 would signify synack,3 would signify fin packet,4 would signify a send packet (This is the packet that contains the payload).
- The client will send a packet and wait for an acknowledgement(except when it sends an ack, there are no acknowledgements for ack). If it receives the correct acknowledgement, then a packet is confirmed to be received by the server, and it will stop resending the previous packet. Once waiting for the acknowledgement of the packet it has sent, if the client receives a packet it has already received, it ignores the packet but sends its acknowledgement.
- The syn packet would have 0 as the seq number, 0 as the size, 0 as the ack number and no data.
- The ack packet would contain 0 as the sequence number and the following sequence number it wants from the other machine as the ack number.
- Similarly, the synack packet would also have sequence number 0 and acknowledgement as 1.
- For each new send packet that is to be sent, the sequence number would be incremented by 1. From the sequence number, we can find out if the packet is a duplicate or a new packet, and if it's a new packet and we are expecting an ack for the previous packet we sent, then the new packet would be ignored.
- We will send more than one packet at a time in a window and keep them in a buffer, and until we receive an acknowledgement of all packets. We retransmit when there are multiple acks and the acks which it is corresponding to is present in the buffer. The acks are treated as cumulative ack and ack seq number is 1 more than the packet its acknowledging.
- This scheme would handle packet loss due to the retransmission of packets until an ack has been received. The packets' order wouldn't matter since out of order packets are

ignored at client side. Any changes in packets' values would make the checksum invalid, and we won't give acknowledgement for invalid packets.