PROTOCOL SPECIFICATION

This protocol thrives over UDP protocol to bring principles of reliable data transfer by constructing additional features in the middle layer over the UDP socket to transfer files between a sender and a receiver. This is somewhat similar to stop and wait protocol.

Events occurring at the SENDER side:

1. The given file is read and each line is made into a packet, with the line string as the payload.
2. Connection is setup with the receiver, using the receiver address and port number, by creating a socket, using mySocket() .
3. The file is sent in parts of 4096 bytes in the form of custom created packets.
4. When a packet is sent, a thread is created which will wait to receive the Ack for this packet, thus implementing Multithreading.
5. While this thread is waiting for Ack , other thread is busy sending packets on time-out event which is finally terminated when the former thread receives an uncorrupt Ack.
6. If the packet is lost, its thread retransmits the packet and waits for an Ack again, in which case a duplicate packet may be recived by receiver which is handled by the sequence number.
7. Steps 3,4,5 and 6 are repeated until the file is fully sent.

Events occurring at the RECEIVER side:

1. The receiver sets up a a socket using mySocket() and binds to the user provided port number.
2. The receiver then waits to receive packets from the sender.
3. If a packet is received , it is checked for errors and an Ack is sent to sender with the sequence numbered packet.
4. If a packet with sequence number that was earlier received is received again, it is acknowledged as a duplicate packet, and an Ack is sent as the sender could have lost the previous Ack.
5. The received bytes are decoded and added to the hitherto received file.

Packet Structure:

1. The packets are implemented as a typical objects in python
2. Each of the fields are to be filled upon packet construction by passing the arguments to the constructor.
3. There are two types of packets: Payload packets and Ack Packets
4. The Payload packets consists of the following fields :
    i. Payload
    ii. Sequence number
    iii. Checksum
    iv. Is_ack (to determine if it is an ack)

5. The Ack packet consists of the following fields:
    i. Ack Number
    ii. Is_ack
    iii. Checksum

Strategies to tackle various situations:

1. Packet Loss: The packet loss is handled by waiting for Acks
2. Packet corruption : The packets are included with the checksum field which is checked for against the on spot calculated checksum at the receiver side.
3. Duplicate Acks: The duplicates are handled by the sequence number as sequences other than the anticipated ack sequences are discarded.
4. Packet re-ordering: The packets are bound to reach in order as there are only two sequence numbers used alternatively.
5. Packet retransmission scenarios: the packet is retransmitted either when there is packet loss, Ack drops and corrupted Acks.

Assumptions in Application layer:

1. The file to be sent is present in the current working directory
2. The file to be received is always wanted in present directory of the receiver

Assumptions in Network Layer:

1. The assumption made here was the reiciever is up while the sender sends file and hence the sender doesn't have to keep sending the first packet again and again upon no acks from receiver.

Additional Scenario:

1. The middle layer is designed such that it can change its policy of reliable data transfer into plain UDP accordingly if the user permits unreliability.

github repository: https://github.com/rohankumar1999/toy-application.git

Team members:
B.N.ROHAN KUMAR - 2017A7PS0024H
NARKEDAMILLY BASWATH - 2017A7PS0033H
G.SAI DHEERAJ REDDY - 2017A7PS1701H
V.KARTHIK REDDY - 2017A7PS0155H
P.VAISHNAVI - 2017A7PS0017H