PROJECT REPORT

# Assumptions:-

* Window size in reliable UDP comprises of ‘number of packets’ unlike ‘number of bytes’ in TCP. One packet is of 1500 bytes which is equal to the MTU value.
* The following string would not be part of any of the files to be transmitted over reliable UDP: **‘!!!<<<NO\_DATA>>>!!!’**. The string is being used to indicate no data in the packet transmitted.
* Terms ‘Sender’ and ‘Receiver’ would be used interchangeably with ‘Server’ and ‘client’, respectively.

# Implementation details:-

## Header:-

Defined a Structure (struct udp\_packet) available to both the sender and receiver containing following parameters:-

**Header Parameters:-**

* Source Port (‘src\_port’ serialized to 4 bytes):- Would be equal to the port number used by the client to connect to server.
* Destination Port (‘dest\_port’ serialized to 4 bytes):- Would be typically same as the source port.
* Sequence Number (‘seq\_num’ serialized to 4 bytes):- Sequence number of the packet.
* Acknowledgement Number (‘ack\_num’ serialized to 4 bytes):- Acknowledgement Number in case the packet is marked as ACK.
* Header Length (‘head\_length’ serialized to 4 bytes):- Length of the header which is fixed at 22 bytes (Total length of all header parameters).
* ACK Flag (‘is\_ack’ serialized to 1 byte):- Acknowledgement flag indication.
* FIN Flag (‘is\_fin’ serialized to 1 byte):- Finish Flag to be set by the sender indicating completed transfer of packets.

**Data Parameter:-**

* Data [‘data’ serialized to (MTU – Header Length = 1478) bytes]:- Contains the part of data fetched from the file to be transmitted. If no data is present, the server/ client would add ‘!!!<<<NO\_DATA>>>!!!’.

## Sliding Window:-

* An initial window size is taken as an input parameter for the server and client.
* Initially the client waits for packets from the server.
* The Sender sends ‘window size’ packets to the receiver and waits for an ACK from receiver.
* The receiver buffers all the packets, irrespective of whether they are in order or not.
* The receiver sends a Cumulative ACK for a missing packet if there is any, or the next packet in the sequence to the sender.
* The sender then sends the next ‘window size’ number of packets.

## RTT and Timeout calculation:-

* On arrival of an ACK, the server calculates the new Estimated RTT and timeout values following the Simplified Estimated RTT Calculation as provided.
* E.g. On arrival of ACK 3, Server would use the sampled RTT for Packet 2 and calculate the stated parameters.
* The values are printed the in program execution output.

E.g.:-

Received ACK: 13

---------- Calculating Estimated RTT and Timeout ---------- Packet 12 | Send time: 1444598142:801081 | Ack Time: 1444598142:801131

Sampled RTT: 50

Estimated RTT: 530.007874

RTT Deviation: 983.199219

Updated Timeout: 4462

## Congestion Control:-

* Slow start and Congestion Avoidance phases have been implemented.
* The transmission starts in slow start, wherein the window size would increase by 1 MSS (MSS = 1 Packet) on arrival of every ACK.
* The transmission would enter Congestion Avoidance if:-
  + Server receives 3 duplicate ACKs.
  + Server retransmits a packet after timeout.
* The change in the phase is marked by the following text in output:-

!!! Transmission entering Congestion Avoidance Mode !!!

* The window size is incremented in a controlled way by using the formula:-
  + window\_size = window\_size + MSS(MSS/window\_size)
* Integer value of the new window size is considered.

## Packet Loss and High Latency simulation:-

* Packet loss simulation has been implemented for sender and High latency simulation has been implemented for the receiver.
* These simulations have been programmed to be enabled/ disabled based on command line parameters (Details in the README).
* Packet Loss simulation is achieved based on a drop percentage provided and High Latency simulation has been achieved by adding arbitrary delays while sending every 3rd ACK.

Some observations:-

1. For a transmission with no drop and no latency:-

Number of packets transmitted in slow start: **11**

Number of packets transmitted in Congestion Control phase: **0**

Percentage of packets transmitted in slow start: **100.000000 percent**

Percentage of packets transmitted in Congestion Control phase: **0.000000 percent**

1. For a transmission with drop percentage 30% at server for 15 KB file:-

Number of packets transmitted in slow start: **3**

Number of packets transmitted in Congestion Control phase: **8**

Percentage of packets transmitted in slow start: **27.272728 percent**

Percentage of packets transmitted in Congestion Control phase: **72.727272 percent**

1. For a transmission with drop percentage 30% at server and a high latency enabled at client for 15 KB file:-

Number of packets transmitted in slow start: **0**

Number of packets transmitted in Congestion Control phase: **11**

Percentage of packets transmitted in slow start: **0.000000 percent**

Percentage of packets transmitted in Congestion Control phase: **100.000000 percent**

1. For a transmission with no drop at server and a high latency enabled at client for 15 KB file:-

Number of packets transmitted in slow start: **0**

Number of packets transmitted in Congestion Control phase: **11**

Percentage of packets transmitted in slow start: **0.000000 percent**

Percentage of packets transmitted in Congestion Control phase: **100.000000 percent**