**REPORT**

The project includes the following files: -

**Helper.h/.c** – to support common function used across all files(read and write files)

**TCPHeader.h/.c** – Contains the TCPHeader header and the necessary functions to create TCPHeader nodes, create linked list, get back contents from linked list.

**SlidingWindow.h/.c** - Contains the Sliding window and the necessary functions to create Sliding window nodes, create linked list, append the linked list,get back contents from linked list and delete the linked list

**Client.h/.c** – Preforms operation of getting the data packets and buffer and send acknowledgements.

**Server.h/.c** – performs the operation of sending the data receiving acknowledgments , calculating timeout and retransmitting the packet, re-transfer the packet if dup-ack received.

**Project Implementation:-**

**Header structure:-**

**typedef struct tcpHeader{**

**unsigned long seqNum; /\*sequence Number\*/**

**unsigned long ackNum; /\*acknowledgment Number\*/**

**int order; /\*the order of packet\*/**

**unsigned short ackFlag; /\*ack flag\*/**

**unsigned short finFlag; /\* fin flag to close the connection\*/**

**char \*data; /\* data pointer\*/**

**struct tcpHeader \*next;**

**struct tcpHeader \*prev;**

**}TCPHeader;**

* The sequence number keeps track of the number of bytes it has transferred
* Acknowledgement number keeps track the next byte of data required.
* The order is used for wrap-around function. The order keeps incrementing every time the sequence number wraps up.
* The ackFlag is to decide if the packet is acknowledgement or data
* The finFlag is to terminate a connection.

**Steps:-**

1) Client sends first request to the server for a file

2) The server receives the file and creates a linked list of TCP header with the data in it.

3) The server puts this linked list into sliding window linked list

4) The TCP header linked list is terminated by fin flag enabled tcp header.

5) The servers starts sending the packets and waiting for acks in different thread.

6) The server sends from the last acked packet till the window size.

7) As and when the acks are received the ackflag in the sliding window is updated and the sliding window is moved. Also the congestion window is updated. The RTT is also calculated for each packet. Timeout is also calculated.

8) If three ack are received for a packet then the packet is resent, also the congestion window and ssthresh is reduced.

9) It the ack is not received the packet is assumed to be lost and the time out occurs and the packet is resent. Congestion window is calculated.

10) The client receives the packet and send the corresponding acks.

11) If the client receives the packet out of order , the packet is buffered and last in-order packet acknowledgement is sent again.

10) After all the data packets are sent and acknowledgement are received in server the fin packet is sent. The server thread is closed.

11) The client replies with a FIN-ACK packet for a FIN packet.

12) The receiver thread in server closes the thread when FIN-ACK is received.

**TimeOut Implementation :**

When sending each packet a thread is created from it to resend the packet. However it will send after a delay of time(timeout). After the delay it will check if the ack is received. It will resend the packet only if the ack is not received.

**Fast Recovery Implementation:**

Every time a acknowledgement is received a counter is maintained. If the counter becomes three then a dup-ack is met and the packet is resent.

**Congestion Window Implementation:**

Initially the congestion window starts with one(slow start). After congestion window is greater than ssthreash then the transition happens from slow start to congestion avoidance. When the duplicate or timeout occurs the congestion window is updated. At timeout the congestion window goes back to slow start

**Sliding Window Implementation :**

I have used the concept of mutex and conditional mutex to implement the sliding window. After the packets is transmitted the send is locked and the receive is open. When an ack is received then the receive window is locked and the send window is open. The next packet in the sliding window unsent is sent and locked again. This happens till the end of the packets and the fin packet is sent after all packets.

**Variable packet loss:**

This is to test the fast recovery. Using the rand() function different packets are dropped randomly and this will send duplicate acks to the sender. Packets are retransmitted.

**High Latency Communication:**

This is to test the timeout. Random packets are delayed for about 2sec. This may exceed the timeout period and the packets would be resent from the server.

**Data Split and rejoining:**

After the file contents are extracted, the file contents are split into chunks and put in the tcp header. They are created in linked list. Each tcp header is then put into a sliding window linked list. Each tcp packet is sent to the receiver. The tcp is then extracted at the receiver end. Appended to the sliding window. After all the packets are received then the tcp packets are received from the sliding windows created a linked list out of header. Then this is extracted again to be written to a file

**Sequence number wrap-around protection:**

My implementation is different form the book for the above concept. I have introduced a new flag called 'order' in TCP header. This flag will keep the order of the sequence number. It initially starts with zero. Every time a wrap around occurs the order is updated . So the next set of sequence numbers is updated with order number 1 till the wrap-around occurs.

The receiver would buffer based on both sequence number and order. This will handle the sequence number wrap-around protection. To demonstrate this I have startted the sequence number close to the max value of long. So after a few packets the wrap-around occurs and the sequence start from zero again.