REPORT

Organization:

- Packet
    1. Create a data structure pkt with members of checksum, length, sequence number, acknowledgement number, flags, and data.
    2. Flags are stored in 3 bits as power of 2 so if we want 2 flags in one packet we store the OR operation of the 2 flags which creates a new unique number.
    3. It calculates the checksum by adding every member into a sum element then adding the carries from the sum and at last returns the one's complement of the sum.
    4. It compares checksum by calculating it and comparing it with the member cksum in the packet.
- Client
    1. Reads the client.in file and store the server IP address to connect to, port number and the requested file name.
    2. Create a UDP socket for the client to be able to connect to other sockets.
    3. Create a server address using the info before to use to connect to the server socket.
    4. Send a SYN packet to the server with the requested file name.
    5. The client waits for the server to send an acknowledgement to the SYN packet if it doesn't arrive in 2 seconds the clients times out and sends the SYN packet again until an ACK packet is arrived.
    6. After receiving the SYNACK the client sends an ACK packet to the server performing a 3-way handshake.
    7. The Stop and Wait strategy, the client opens a new file then starts to receive data packets from the server.
    8. For every new data packet, the client immediately sends an ACK packet back to the server confirming the data packet until a FIN

packet arrives then the clients know that this is the last packet and sends a FINACK packet.

9. The client immediately types the data associated with every packet in the file.
10. The GBN strategy, the client opens a new file then starts to receive data packets from the server.
11. For every new data packet, the client makes sure that the data sequence number is the same as expected then immediately sends an ACK packet back to the server confirming the data packet until a FIN packet arrives then the clients know that this is the last packet and sends a FINACK packet.
12. The client immediately types the data associated with every packet in the file.

- Server
    1. Reads the server.in file and stores the port number, seed generator number and the packet loss probability.
    2. Creates a UDP socket for the server so clients can connect to.
    3. Declares the server address using the host local address and the port number.
    4. Binds the server socket to the local address.
    5. Allocates space for the incoming client address.
    6. Stays waiting until a packet arrives with SYN flag and stores the client address the packet is coming from.
    7. After receiving the SYN packet the server immediately sends back a SYNACK packet starting the file transfer.
    8. The server starts handling the client by opening the file with the name requested by the client.
    9. Then the server starts to calculate the size of the file and the total number of packets need to be sent to the client.
    10. Using the seed to create a random number generator and with the PLP the server creates a list of the indices of the packets that will be simulated as they are lost during the transfer.
    11. The server receives the first ACK packet from the clients after sending the SYNACK packet confirming the 3-way handshake.

12. The Stop and Wait strategy, for every packet to be sent except the last packet, if it is simulated to be lost then the server doesn't send a packet and waits for an ACK packet for 2 seconds until a timeout happens.
13. After that the server sends the packet which simulates retransmitting the packet by attaching the suitable chunk of data from the file and waits to get an ACK packet for every packet.
14. For the last chunk of data to be sent the server creates a packet with FIN flag equal 1 which means this is the last packet the client should expect from the server.
15. The GBN strategy, Until the base reaches the packet before the last packet, The server creates the packet to be sent to the client by calculating its sequence number from the previous sequence number plus the length of the data where the first sequence number is random.
16. For each packet if it is to be simulated as lost then the server doesn't send it until it is timed out then send it which results in dropping the windows size to 1 MSS due to congestion.
17. For the window, the server sends a new packet as long its base number is less than the first packet base number in the window plus the size of the window which controls the congestion.
18. If the packet is sent, then the next base number is moved by one until the ACK of the packet is arrived then the base number is also moved by one.
19. If the window size is less than the threshold then every time it increases exponentially until it reaches the threshold then it increases only one by one.
20. For the last packet it is sent with FIN flag.

Major Functions:

- calcPktCksum: Calculates the checksum of the packet by adding all its members into a sum variable them adding the carries to them and storing the one's complement of the sum in the cksum member of the pkt.

- sendSYN, sendACK, sendFINACK, sendPKT, sendSYNACK: Creates a packet with the appropriate flags to send to the other side.
- recvData: For the client to receive the data segments sent by the server to write them in the file it also compares the checksum to check for any corrupt data and the flags must be either FIN or 0.
- clientHandling: For the server to open the requested file by the client, calculates its size, finds total number of packets to be sent, and calculates the indices of the packets simulated to be lost then chooses the strategy SnW or GBN then after that it closes the file and the connection.
- getChunk: Using the required index of packet it sends back the appropriate chunk of data needed for that packet which will always be of size MAX_CHUNK unless it is the last packet then it is of size of the rest of the data which must be less than or equal to MAX_CHUNK.
- getLostPktsIndices: Using the PLP and total packets it calculates the number of lost packets then stores their indices in a list to send back.

Data Structures:

- For the packets:

```
struct pkt {
    uint16_t cksum;
    uint16_t len;
    uint32_t seqno;
    uint32_t ackno;
    uint16_t flags;

    enum EFlags
    {
        FLAG_SYN    =  0x1,
        FLAG_FIN    =  0x2,
        FLAG_ACK    =  0x4,
    };
    char payload [MAX_PKTSIZE];
};
```

- Lost packets indices are stored in set<int>
- Sent packets in the window in GBN strategy are stored in a vector<pkt>

Congestion control graph:

- Stored file to draw:

Open graph.txt /usr/local/bin Save

```
 1 1 1
 2 2 2
 3 3 1
 4 4 2
 5 5 4
 6 6 8
 7 7 9
 8 8 1
 9 9 2
10 10 4
11 11 8
12 12 9
13 13 10
14 14 1
15 15 2
16 16 4
```

- Graph using gnuplot: