Raivat Alwar and Ritesh Patro
Project 3

Transmission Control Protocol (TCP)

Before the three way handshake to establish a connection can be started, first both the server and client side nodes need to have their sockets in a listening state. On the bootup of a server node, this is done as the first step, all the sockets are set to a listening state. In the case that a client node gets booted, it opens up one of its closed sockets and sets that specific socket to a listening state. The client then sends a packet containing the SYN flag to a specific port in the server node. The specific node is determined randomly in the python file. The server will then receive the SYN flag at the desired port and send the client a SYN_ACK packet. When the client receives this, the connection from client traveling to server is established. After the establishment, the client will send the server two different packets, the CONN_ACK and the DATA packet. The server receives the CONN_ACK packet first and establishes the connection for the server traveling to the client. By this point, the three way handshake is complete.In the case of the data transmission, the client will then send the server DATA packets and the server will send the client ACK packets in response.

Going over the client side handle of data transmission first, the initial step the client does is send as many packets as dependent on server window size and the congestion control variables. The server window size is taken from the data field in the SYN_ACK packet on the client side while the client determines the congestion control variable on initiation. These fields make it so that the packets sent into the system will be equivalent to the smaller between the two numbers. Everytime the client sends out a new unique DATA packet, the seq number, data position in sendbuff, and the ack value are saved in two hash tables with the seq number as the key. The sequence number is then added into a queue and is later used to determine if a DATA packet should be sent again. Everytime an ACK packet is received from the server, the corresponding seq number that the ACK packet is for is removed from both hashtables. The client will then call the flow control method and send out the possible number of packets it can send. When a client checks to see if it should resend a packet based on a timer, it will look at the top of the queue and see if that seq number is present as a key in either hash table. If it is, that means the ACK packet has not arrived yet and the packet has been lost in transmission. Seeing this, the client will resend the packet, and then add the sequence it just resent to the top of the queue.

On the server side, everytime the server receives a data packet, it checks to see if the seq number has been sent before. If the sequence number received is unique, the data value is then added into recievedbuff into a predetermined position based on the difference of the first sequence number sent and the received sequence. If the sequence is not unique that means the packet is a duplicate packet and the server will resend an acknowledgment. In the case the packet is unique but there is not space for the data to be stored in the buffer, the ACK will not be sent forcing the client to resend the DATA packet until the buffer space is open.

In regards to closing the connection, a three way handshake takes place. When the client is done sending data, it will send the server a FIN flag to which the server responds with a FIN_ACK flag. When the server receives the FIN flag, it will set the connection to INIT_CLOSE and send the client a FIN_ACK flag. The client, upon receiving the FIN_ACK packet, will fully close its connection. After a while the server will make it did not get any new info from the client then close its connection by changing the state from INIT_CLOSE to CLOSED.