Project 3 Part 2 - UDP Solano Report

Nicholas Myers ID: 917617010 Section: A01

Filenames: **receiver_solano.py**
sender_solano.py

---

1. Explain your implementation of sequence and acknowledge numbers. Why do both sender and the receiver maintain separate sequence and acknowledgement numbers? (3 points)

- First, both the client and server in their initial headers generated a random number bounded by the recommendation in the RFC (0 - 4294967295). They then appended this random number as their sequence number to start. Their respective acknowledgement numbers were based on the received sequence number, and would increment accordingly to however much data was successfully received. For example, if the clients' sequence number was 1000 and sent 100 bits to the server, the server's acknowledgement number, after successfully processing it, would then become 1100. Thus for the client, once they received the servers ACK, their acknowledgement number would be the ACKs sequence number + 1. Then if the client sent another message, its sequence number would increment by the # of bits it is sending and so on. The sender and receiver maintain different sequence and acknowledgement numbers since the acknowledgment number is the sequence number of the next message the receiver expects to get. If the clients' sent messages sequence number was equal to the received ACKs acknowledgement number, then it knows the receiver successfully got the data.

2. Explain and justify the additional header fields you needed to add compared to part 1 for this implementation. (3 points)

- The only additional header fields I added in this part compared to part 1 were seqNum and ackNum (sequence number and acknowledgement number). Because part 2 implements a stop and go protocol, our client needed to ensure if the sent messages sequence number was equal to the received acknowledgements acknowledgement number. That way we would know if the receiver successfully got our data, thus we needed to add the seqNum and ackNum header fields that would dynamically change as messages were sent and received.

3. How would the performance of this file transfer have been affected if we did not use separate welcoming and connection sockets? (4 points)

- It would severely reduce the efficiency for multi-client file transfer as a new client would have to wait for the original client to finish its file transfer and close its connection before it could allow a new client to connect. This is because the server's socket would start its while loop to connect to an incoming connection with a 3-way handshake, then if we did not use another socket, it would create another while loop to accept all the data from the client until disconnected. Since the server would be in this nested while loop in the socket, if another client attempted to connect, it would not receive anything back as the server's socket is already binded to and connected to another clients' port and ip and would need to wait until the current client disconnects to attempt to connect with the server.