Desing

As with the previous projects, functionality was prioritized over readability. As a result, project 3 was not modularized even though I want it to modularized the project. However, the parts corresponding to each project have been separated from each other and they have comments in the code as well in the debugger to help with readability. In order to allow a client and a server to establish a connection, a socket structure was used, and this socket also describes the state of the current connection. Once a client connects to the server, the listening socket is copied and a new socket is created to allow for communication. This new socket is added to a socket list in the server and client side. The way to identify a socket is by its source and destination port values. To send the data, the fact that data is being sent from zero to a transfer size implies that the data is already in order. So it is not being checked if the data is in order. For the sliding protocol mechanism, the sliding protocol checks if there is enough space in the buffer and the maximum transfer size has not been reached yet and the maximum TCP payload size has not been reached either. If the three conditions mentioned before are met, data transmission continues. The space that is available in the buffer is the data that is sent. Since nesC does not allow dynamically to create timers that correspond to each packet, an array of data was sent in the window slide mechanism instead of sending multiple packets. In addition, it may be hard  also to differentiate which timer corresponds to each packet. So if a packet is lost, the packet just gets resend. When we check for the sequence number to see whether a packet was lost or is a duplicate, the packages with a lost sequence numbers are the ones that gets resend. For connection tear down, the connection was handled in a three way reverse handshake. It is realized that some packets are lost, but the majority make it to the destination. Especially some packets may not make it tot eh destination if a FIN packet is sent before all the regulars packets. Should this happens, the packets just get resend.

Discussion questions

Q1. It seems there is a tradeoff. If security is important, then a choosing a random initial sequence number will confuse any potential attacker that is eavesdropping the connection as the attacker will have trouble identifying the next sequence numbers. However, having random sequence numbers can also jeopardize the detection of lost sequence numbers. E.g. If the initial sequence number is one, and the next sequence number is 5, we know that the previous packets were lost. Thought this should be handled in the network layer.

Q2. The buffer should be big enough to handle all data regular transmissions. So far, the received data resides in the buffer and it is not being sent to the application layer. So the buffer needs to be big enough to handle all data connection. 16 bit is the size of the current buffer.

Q3. The maximum number of sockets will be used immediately as each SYN will create a new socket in the server to host the connection. One way to handle this would be to have a maximum number of connections allowed per host.

Q4. The connection may remain open indefinitely. A timer could be used to close the connection after certain period of time.