Assignment COMP3331 Report

The language I chose to use on my COMP3331 programming assignment was Python 3 for its ease of use and readability.

Organization of code

The STP system I have implemented does not utilize any child directories or helper files. All helper and main functions are contained within the Sender.py and Receiver.py files.

Overall Program Design

My program closely follows the structure of a TCP transport system. The sender is able to send files with ‘Stop and Wait’ through both a reliable and unreliable channel, as well as using a sliding window protocol to send data through a reliable channel.

The receiver can receive all incoming data, and copy this over to a file on the receiver’s end.

To send a file, the sender must have access to the file. First the sender will send a SYN segment to the receiver to initiate a connection. If this segment fails to reach the receiver 4 times, the sender will respond with a RESET segment and be unable to send the file.

If, however, the receiver receives the SYN segment, it will respond with an ACK and open a connection to the sender. Again, depending on the rlp, the receiver may be unable to send this ack, to simulate loss of acks in the real world.

If a successful two-way handshake takes place, the sender will read the file, 1000 bytes at a time, since that is the largest amount of data the channel can support. The sender will then consecutively send DATA segments, while filling the ‘window” array. This array simulates an actual TCP sliding window. If at any point the window array is full (determined by the value of the window_size attribute), the sender will temporarily stop reading input. Segments can leave the window array when a corresponding ack has been received for the segment.

The sender will also maintain a timer for the oldest unacknowledged segment. A timer is started whenever the size of the window reaches 1 (when the first segment has been sent, or a segment has been removed) and whenever an ack is received and the window still contains other segments.

The send function is not exited until all acks have been received for all sent segments.

The “listen” thread handles all functions to do with receiving an ack from the receiver. It is able to differentiate between SYNACKs, FINACKS and normal ACKs. In the case of a SYNACK, the sender establishes that a connection between the sender and receiver works. When receiving a FINACK, the sender closes the connection and ends the program.

When it receives a normal ACK, it checks initially for a duplicate ack. The sender maintains a count for the number of duplicate acks it has received for a specific segment. If the sender has received three duplicate acks for one segment, it will resend the oldest unacknowledged statement, the one in the start of the window.

In the case of a stop and wait protocol, the sender will send each segment, and wait until the corresponding ACK has been returned.

The receiver operates in a similar way, however it is the receiver that manages “packet loss” and “ACK loss”, determined by the flp and rlp.

The receiver also maintains a buffer, in case a segment is lost but the previous segment is sent. In the case of a skipped segment, all new segments that are received are stored in the buffer. When the correct segment is received, the buffer is emptied in order into the file.

Data Structure Design

The main data structures used in STP are queues and lists. The sender maintains a “window” queue, to simulate the sliding window in TCP and the receiver maintains a buffer array to store any out of order segments.

The sender utilizes multi-threading to allow for data to be sent and received simultaneously as well as for a timer to be set in the background, to monitor the oldest unacknowledged segment.

Other than this, the data structures used in the program are relatively simple.

Design Tradeoffs

This interpretation of STP would not be possible with a few design tradeoffs.

Firstly, the receiver often sends ACKs far too quickly, especially since the receiver and sender are on the same machine. This rapid retrieval of ACKs often confuses the sender when using the sliding window protocol. To mitigate this, I have used a time.sleep() function at the start of the receiver’s main function. This allows for the segments to be sent consecutively rather than basically one by one when receiving each corresponding ACK.

This interpretation of STP is also unable to correctly send and retrieve data when using the sliding window protocol over an unreliable channel. It is able to work in cases where only acks are lost, or if only a few segments in the middle are lost, however if multiple segments are lost or if the first few segments are lost, the program is unable to function.