Project Documentation

The Simple Transport layer acronym-ed as STCP, is a reliable in-order data transmission protocol. It insures the reliability and in-order transmission of data packets requested by one host from another on top of both reliable and unreliable network.

Design :-

The STCP being a duplex transport layer protocol is capable of doing both receiving and sending for a single host. For each host it will maintain the information regarding the sender and receiver states and buffers. The maximum window size that it supports is 3072 bytes with the maximum payload in each segments being 536 bytes. The sequence number space for the segments can range from 0-4294967296. It works as below :-

Initial Handshake :-

To begin the data transfer both the sides needs to do connection initiation to setup a connection between two which helps in future data transmission. This connection setup is done with the 3-way Handshake mechanism in which they decide an initial sequence number in the range of 0-255 both including. 3-way handshake is initiated from active side by sending SYN packet. The passive side on getting the SYN sends SYN-ACK packet which is a SYN piggybacked with and ACK packet. Finally, the active end sends an ACK packet thus establishing a connection between 2 hosts. The transport layer waits for 2 seconds to hear for an event from Network layer. If either side timeout waiting for ACK then it retransmits SYN packet considering the earlier packet to be lost. The retransmission takes place at a maximum of 6 times after which the connection is closed considering the network layer failure.

Data Transfer :-

After the connection has been setup between 2 hosts one can request data from other side. In our case client application requests a file from the server application. These communication is done using the initial sequence number on which both sides agreed upon. STCP maintains receive buffer *(rcvrDataBuffer)* and send buffer (*sndrDataBuffer)* having size of 3072 bytes for each hosts to support the duplex connection. The state variable for sender buffer are *sendBase (the first unacknowledged sequence number), nextSeqNum(the next sequence number that should be used to transfer any data within window size), sendBufferBaseInfo (the base index of the sender buffer where the datas from application are stored).* The state variables for receiver buffer are *expectedSeqNumber (the sequence number expected to receive), selfRcvWindowSize (empty space in it's own receiver buffer), rcvBufferBaseInfo (index of receiver buffer from where data will be transferred to application) and currentRcvrWindowSize (advertised window size of the other host).* I am also maintaining a *rcvrWindow* in which index corresponding to the data bytes in receive buffer is marked 1, this is used for the in-order delivery of data to the application and making receiver capable of storing the out of order data segments.

Sender's Action :- Sender will receive the data from Application depending upon its own free space in buffer and the advertised receive window size of other end. It can accept at max 3072 bytes of data from the application. As soon as it receives any data it stores it inside the send buffer and transfers those forming the segments of maximum size of 536 bytes. It splits the data received from the application and sends those in multiple packets to the receiver. It starts timer as soon as its sends the data and wait for ACK. If timeout happens then it retransmits the data for a maximum of 6 times. On receiving ACK it moves its *sendBase* to the sequence number for which it receives the ACK.

Receiver's Action :- Receiver accepts all the data whether it be in-order or out-of-order and stores it inside its receive buffer. It discards any data that has already been acknowledged or falls out of receivers window size. STCP discards any *options* if present inside the received segments. Before sending the data to application it does a lookup in the *rcvrWindow* to find the length of in-order data received and sends an ACK. It also sends the ACK for the with the expected data sequence number for the out of order data segments.

I have used the below mentioned functions to implement the proper handling of all the scenarios and make the code modular :-

a) ***setTimerForUnackedData*** - It is used to set the bool flag inside the context_t structure used to indicate whether timer is running or not.
b) ***isTimerValueSet*** – It is used to check if a timer is running.
c) ***startTimer*** – It is used to start the timer and set the bool flag specific to timer inside the context_t. I have used alarm() to implement this using 1 sec as timeout value.
d) ***stopTimer*** – It is used to stop the timer and unset the bool flag.
e) ***handleTimerExpiry*** - It is used to retransmit the dropped data packet and the FIN packet if ACK is not received for either of them. It retransmits at max 6 times.
f) ***createStcpHeader*** – It is used to create the header for the segments that will be sent on the network.
g) ***getEmptySenderBufferSize*** – It is used to get the empty size of the sender buffer.
h) ***storeDataIntoBuffer*** – It is used to store the data inside the sender and receiver buffer.
i) ***setReceivedBytesInReceiverWindow*** – It is used to set the index position corresponding to the received data bytes in the receive buffer.
j) ***sendDataToApplication -*** It is used to send all the in-order data to the application layer and then updating the variables like receive window size, rcvBufferBaseInfo and expected sequence number.
k) ***sendAcknowledgementPacket*** – It is used to create and send the ACKNOWLEDGEMENT segments. ACK segements in STCP doesn't consume any sequence number.

Connection Termination – Connection termination in STCP happens by 4-way handshake mechanism. It can be initiated from either end. For example:-FIN from active side, ACK from passive side, FIN from passive side and ACK from active side. After sending FIN packet timer is started and if ACK is not received on either side before timeout then FIN segments are retransmitted for maximum 6 times after which the connection is closed.

Some Facts:-
STCP switches between various states which are as follows:-
CSTATE_DEFAULT, CSTATE_SYNSENT, CSTATE_SYNRCVD, CSTATE_SYNACKSENT, CSTATE_SYNACKRCVD, CSTATE_ACKSENT, CSTATE_ACKRCVD, CSTATE_ESTABLISHED, CSTATE_FINWAIT_1, CSTATE_FINWAIT_2, CSTATE_CLOSE_WAIT, CSTATE_LAST_ACK, CSTATE_CLOSING, CSTATE_TIME_WAIT
STCP has the endianess compatibility i.e. it will work both on machines supporting little endian or big endian architecture. To enable few printf statements throughout the code one needs to uncomment the line ***(//#define print 1).***