Assignment 3: Developing a Reliable Transport Layer Protocol using Raw Sockets

Objective:

The objective of this assignment is to develop a reliable transport layer protocol on the top of the IP layer. Note that IP layer provides unreliable datagram delivery - there is no guarantee that a packet transmitted using IP will be eventually delivered to the receiver, because of any of the following reasons:

1. The packet may get lost during end-to-end delivery - this can happen due to many reasons, like buffer overflow, forwarding error etc.
2. The packet is transmitted to the end host, but the content of the packet may get corrupted.

The broad objective of this assignment is to develop a transport layer protocol with following features,

a) *Connection oriented packet delivery* - before initiating a session, an end-to-end connection is established to ensure that an end-to-end path exists between the source and the destination

b) Ensure end-to-end reliability - End-to-end reliability is ensured via explicit acknowledgement of packet reception.

We call this protocol as **Reliable Transport Layer Protocol (RTLP)**. The unix socket programming API provides a raw socket (SOCK_RAW) framework to directly access the IP layer services. Your task is to develop RTLP on top of the IP layer services supported by raw socket APIs.

Description:

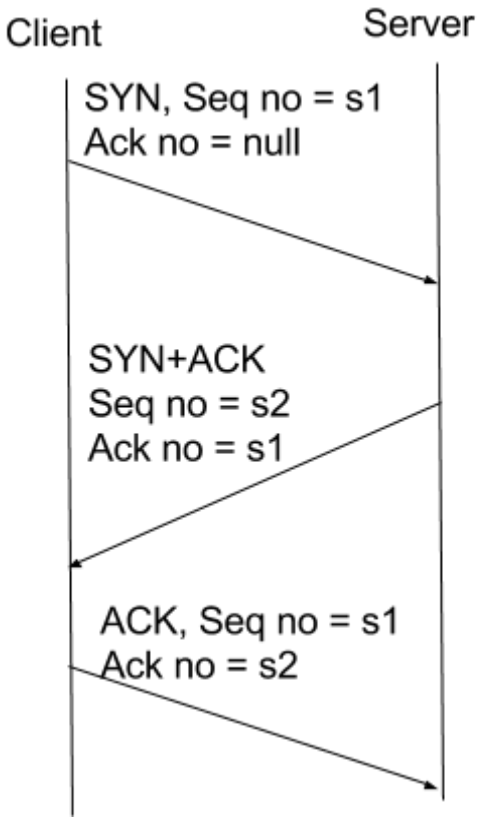
First you need to define a header structure for RTLP. RTLP header is 16 bytes and it contains following fields:

- Source port : 2 bytes
- Destination port: 2 bytes
- Sequence number: 4 bytes
- Acknowledgement number: 4 bytes
- Checksum: 4 bytes

Part A: Connection Establishment: RTLP uses a three way handshaking for connection establishment at the beginning of a data transmission session.

1. The client sends a SYN packet to the server. The SYN packet contains an initial sequence number, say s1, (which is a randomly generated integer).
2. The server sends back with an SYN+ACK packet indicating that it agrees with the connection (ACK), and it also wants to open a connection (SYN) towards the client. Therefore, it generates another sequence number, say s2. For this SYN+ACK packet, sequence number = s2 and acknowledgement number = s1.

3. The client, on receiving the SYN+ACK packet, returns back another ACK packet where acknowledgement number = s2.



In this part of the assignment, your task is to implement the three way handshake mechanism.

Part B: Error Control through Explicit Acknowledgements: RTLP uses explicit acknowledgement mechanism to inform the sender that the receiver has received the data sent by the sender. This is done through the sequence number and acknowledgement number. RTLP uses byte sequence number that indicates how many bytes have been transmitted, whereas the acknowledgement number indicates how many bytes have been acknowledged. Note that the receiver acknowledges every individual packets explicitly. The initial sequence number for the server and the client is established during the connection establishment procedure as discussed before.

Assume that in a RTLP session, the initial sequence number is 1224 for the client and 3342 for the server (s1=1224, and s2=3342). Here is an example of sequence of packet transmissions.

1. Client sends a packet of 24 bytes. So, in the RTLP header of the data packet, sequence number = (1224+24) = 1248, Acknowledgement number = 3342. The sender also keeps this data in a buffer, called RTLP sender buffer, until it receives the corresponding acknowledgement.
2. On receiving the packet, the server sends an acknowledgement (ACK), where Acknowledgement number = 1248, indicating that it has received the 24 bytes of data correctly. Note that the sequence number field for this ACK packet is 3342, as the

server is not sending any data. If it wants to send data with this packet, it can do so by modifying the sequence number field appropriately.

3. Now there can be two options:
    a. **The client receives the acknowledgement.** On receiving the acknowledgement, the client understands that all 24 bytes of the packet has been transmitted successfully, and it deletes the packet from RTLP sender buffer. If the next packet is of size 100 bytes, then for that packet, sequence number = (1248+100) = 1348.
    b. **The client does not receive the acknowledgement.** Now, if the client does not receive the acknowledgement within a timeout period (think of the value of a suitable timeout period), it resends the packet from the RTLP packet buffer, with sequence number = 1248.

So, here are your tasks for this part of the assignment.

1. Implement the explicit acknowledgement mechanism with byte sequence number.
2. Implement the timeout mechanism. Vary the timeout value to check its impact over packet transmission. Note that if this timeout value is less, you'll receive duplicate packets. On the other hand, for a large timeout value, the packet transmission delay may increase.

Part C: Checksum Computation: RTLP header contains a checksum value. Before sending a RTLP packet, the sender computes the checksum over following fields (together):

I. Source port
II. Destination port
III. Sequence number
IV. Acknowledgement number
V. Data

You can use any simple hash function for the checksum computation. The sender includes the checksum in the RTLP header. The receiver, on reception of the packet, first computes the checksum over all the above five fields together, and matches with the received checksum. If the checksum value matches, it accepts the packet, otherwise, it drops the packet. The checksum ensures that the neither the header nor the data has been modified during packet transmission.

In Part C, your task is to implement the checksum computation and matching procedure in RTLP.

Part D: Connection Termination. The connection termination procedure at the end of a RTLP session is similar to the connection establishment procedure and uses three way handshaking. If the client wants to terminate the session, it sends a FIN packet to the server. In response, the server sends back with a FIN+ACK packet (similar to the SIN+ACK one), and finally the client replies with an ACK packet. This closes a RTLP session.

Your task is to implement the three way handshaking mechanism for terminating a RTLP session.

Test Cases:
This is a simple test case that you can use to check the working of RTLP. However, this does not cover all the possible test scenarios, and you should think of developing the test scenarios accordingly.

1. Run a RTLP server and a RTLP client at two different physical machines.
2. RTLP client sends periodic ECHO_REQUEST messages, where the message data is “ECHO REQ N” where N is the request number, N=1,2,3,...
3. RTLP server replies back with ECHO_RESPONSE messages, where the message data is “ECHO RES N+1”. For example, in reply to “ECHO REQ 10”, the server should send “ECHO RES 11”.
4. Vary the Acknowledgement timeout value and check for what value of the timeout, you get duplicate messages.
5. **Food for Thought:** Can you see any relation between acknowledgement timeout and message transmission/propagation delay?