COMPUTER NETWORKS (CS F303)
LAB-SHEET – 10
Topic: Practice problems on Socket Programming and Wireshark Packet Analysis

Exercise: 1 Client-server file transfer using UDP as the transport protocol and implementing Stop-and-Wait protocol at application level.

Write client and server programs to upload a given file **(input.txt**) from client to the server using UDP as the transport layer protocol and Stop and Wait ARQ protocol at application layer to attain reliable communication.

Your program MUST include following features:

1. Each line of the **input.txt** file should be transmitted separately using Stop-and-Wait protocol by encapsulating it in the form of a packet structure. (*Note that the length of each line of the file may be different and hence the packet size would vary*.)
2. In addition to the payload, the packet structure should contain following information in the form of a header.
   a. The size (number of bytes) of the payload.
   b. The Seq. No. (Required for Stop and Wait protocol).
   c. Whether the packet is the last packet or not?
   d. The packet is DATA or ACK. In this way, you can utilize the same packet structure for both DATA send by the client and ACK send by the server.
3. The packet loss should be implemented at the server. The server will randomly drop a received packet and would not send ACK back to the client. The packet drops rate (PDR) should again be defined as **#define macro**, and its value can be kept as 10% for the submission purpose. Assume the ACKs are not lost. Hence no need to implement ACK losses.
4. The client should handle the retransmission of lost packets. You are free to choose any of the following techniques to implement retransmission:
   a) Explicit timers
   b) Non-blocking socket and receive function calls
   c) Explicit NACKs (Negative ACKs) sent by the server(receiver) (*Note: Similar to ACK packets NACK packets are also not lost*)
5. The duration of retransmission time should be defined as a macro and can be set 2 seconds for the submission purpose. You should keep a copy of the transmitted packet to facilitate re-transmission, instead of re-constructing a new packet from the input file again.
6. At server side, you **should not** store the packets in a temporary buffer and write the entire buffer to the file at the end. Instead, directly write the received packets in the output file (**output.txt**) as they are received.
7. The client and server should produce the traces (using printf statement) of every sent (including retransmission) and receive events in the following format:
8.

Client Trace	Server Trace
`SENT DATA:Seq.No---of size---Bytes`	`RCVD DATA:Seq.No---of size---Bytes`
`RCVD ACK:for PKT with Seq.No. ---`	`SENT ACK: for PKT with Seq.No.---`
`RESENT DATA: Seq.No---of size---Bytes`	`DROP DATA:Seq.No.--- of size---Bytes`

Exercise-2: Traffic capturing and analysis using Wireshark.

Use Wireshark to capture packets from your Network Card. Here is what you need to do:

i. Close all opened running windows and applications (web browser, email program and other applications that typically send data to the Internet). This will minimize the amount of traffic you are going to capture.

ii. Open Wireshark program, then open web browser and start capturing packets on your active network card.

iii. Type www.google.com In the address bar (URL bar) of your web browser and search for anything you want. Browse top two-2-3 results given by google search. Further, click on URLs and images in the web pages. In summary, you are supposed browse the web for about a minute and capture network traffic to be analyzed in in this assignment. Stop Wireshark capture.

iv. Close the browser completely.

v. Start analyzing packets and protocols.

Answer following questions. Provide the details of steps used by you to get the answer and also the screen shots of the outcome of these results to justify your answer in a single file named as your_BITS_id.pdf.

1) What is the duration of your packet capture in seconds? What about the start and end time of the capture expressed in hh:mm:ss?

2) How long did it take from when the HTTP GET message was sent until the HTTP OK reply was received for the webpages (at least 3) you visited in your web browser?

3) What is the Internet (IP) address of the URLs you visited and what is the Internet address of your computer?

4) What is the IP address of the DNS server you are connecting to?

5) List the application layer protocols that you see in protocols field that are using UDP and TCP respectively.

6) Locate TCP handshake segments and find the sequence number of SYN, SYN+ACK and ACK messages of all the TCP connections made by your computer.

7) Find out all incoming (received by your machine) http traffic.

8) Find out the list of all TCP connections which have been reset. Provide appropriate reason for connection reset. **Packet of previously closed connection is arriving now, so connection is established once again to receive that packet.**

9) List all TCP segments which are send and received by your machine having header length more than 20 bytes. Give the appropriate reason for header length larger than the default size.
Options field is being used here so it is greater than the default size

10) List all the duplicate ACK TCP segments.

11) Provide the sequence number of any one out-of-order TCP segment captured in your trace file.

12) How many number of HTTP request (i.e., GET and POST) messages did your browser send?

13) Find out all the traffic between your machine and a particular (of your choice) web site (IP address).

14) Calculate the throughput of all the TCP connection involved in question 13.