Practice Problems:

1. Develop a socket program in UNIX/Linux that uses (i) TCP as the transport protocol and (ii) UDP as the transport protocol for transferring a short message between a client and server. The client sends a string (input by the user) to the server and the server prints the string on the screen after receiving it.

2. Develop a TCP-based client-server socket program for transferring a large message. Here, the message transmitted from the client to server is read from a large file. The entire message is sent by the client as a single data-unit. After receiving the file, the server sends an ACK message to the receiver. Verify if the file has been sent completely and correctly by comparing the received file with the original file ("diff" command could be used). Measure the message transfer time and throughput.

3. Develop a TCP-based client-server socket program for transferring a large message. Here, the message transmitted from the client to server is read from a large file. The message is split into short data-units which are sent one by one without waiting for any acknowledgement between transmissions of two successive data-units. Verify if the file has been sent completely and correctly by comparing the received file with the original file. Measure the message transfer time and throughput for various sizes
of data-units.

Assignment Problem

4. Develop a UDP-based client-server socket program for transferring a large message. Here, the message transmitted from the client to server is read from a large file. The message is split into short data-units which are sent by using stop-and-wait flow control. Also, a data-unit sent could be damaged with some error probability. Verify if the file has been sent completely and correctly by comparing the received file with the original file. Measure the message transfer time and throughput for various sizes of data-units. Also, measure the performance for various error probabilities and for the error-free scenario.

Choose appropriate values for parameters such as data unit size and error probability. You can simulate errors according to the frame error probability. You are free to implement the ARQ in your own way, but with stop-and-wait. For example, you may want to avoid TIMEOUTs and handle retransmissions in some other way (you may want to choose negative acknowledgement). Repeat the experiment several times and plot the average values in a report with a brief description of results, assumptions made, etc. Include the following performance figures in your report:

1) Transfer time vs error probability (0 and other values of your choice)
2) Throughput vs error probability (0 and other values of your choice)
3) Transfer time vs data unit size (error probability = 0, error probability = value of your choice)
4) Throughput vs data unit size (error probability = 0, error probability = value of your choice)