Lab 9

1. Simulate a simple stop-and-wait protocol where the sender sends a frame, and the receiver acknowledges the reception. Experiment with different timeout values(atleast with 5 different values) and analyse their impact on the throughput performance of the stop-and-wait protocol.
Graph: X axis: timeout values Y axis: throughput
2. Implement Go-Back-N and selective repeat protocol simulations. Create a sender and receiver that can handle multiple frames in the window. Introduce frame loss and errors into the simulation. Analyse how these protocols handle these situations and also analyse the impact of different window sizes (atleast with 5 different values) on these protocol's efficiency.

Graph 1: X axis: window size Y axis: Throughput
Graph 2: X axis: window size Y axis: Delay

Lab 8

To analyze the performance of Pure Aloha, Slotted Aloha and CSMA MAC Protocols against increasing frame sizes(200,300,400) in a given network.

Lab 7

Compare the performance of any two static routing protocols with the following topologies: (i) Linear (ii) Random (iii) Grid . Assume the following:
Number of nodes: 10 nodes for linear topology and 25 nodes for Random and Grid topologies
Traffic : FTP
Queue : Drop Tail
Number of Source : 50% of node
Simulation Time: 100 Sec.

Plot the graphs for the following metrics:

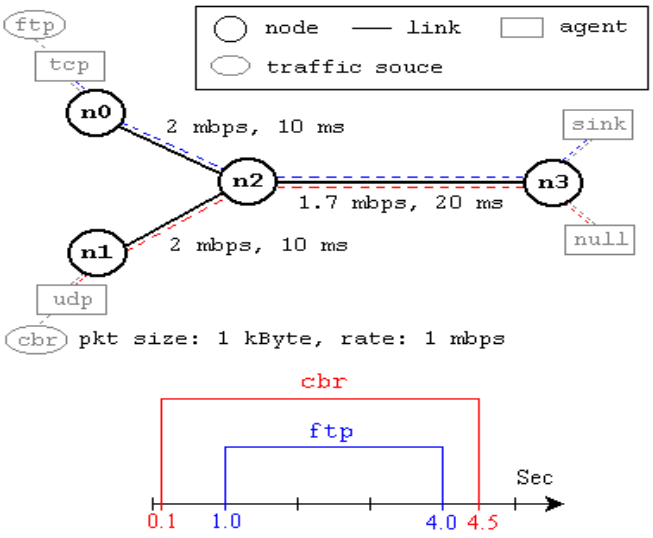
X axis: Simulation time interval (20, 40, 60, 80 and 100sec)
Y Axis: PDR, PLR, routing control overhead and delay

Lab 6

Simulate TCP Reno, Tahoe and Vegas for hight, medium and low traffics with differing number of nodes (10,20,30,40,50) and plot the following metrics for the differing traffics:
1. Control Overhead
2. Packet delivery ratio (PDR)
3. Retransmission rate (RT)

Lab 5

1) Simulate the following Network. Find the total number of packets transmitted and received.
   Sources: no and n1
   Destination: n3
   Total simulation time: 5 sec
   Interface Queue type: Droptail
   CBR Traffic from n1 to n3, started at 0.1 sec and stopped at 4.5sec
   FTP traffic from n0 to n3, started at 1 sec and stopped at 4 sec.



2) Simulate the network with varying number of nodes (10,15,20,25,30,40) to compare TCP and UDP protocol in terms of the following performance metrics.
   1) Throughput
   2) Packet loss
   3) End-to-end Delay
   4) Fairness Index

Lab 4

Simulate any 3 topologies (of which one must be hybrid topology) with N nodes for 100secs. The nodes are numbered sequentially started from 1. In all the topologies, the odd numbered stations are sources and even numbered nodes are destinations. Assume that all source nodes transmit the fixed size data packets from the starting of the simulation till the end of the simulation.

Plot (i) Graph 1: Throughput in Y-axis Vs No. of nodes (5,10,15,20, 25) in X-axis
(ii) Graph2: Packet Loss in Y-axis Vs No. of nodes (5,10,15,20, 25) in X-axis

Assume your own values for the rest of the parameters.

Lab 3

Select *any two* out of the following questions:

1) Implement an auction marketplace. An auctioneer has a set of items to be auctioned and users can buy an item. Before an item is sold, it should be properly put out for auction - the auctioneer starts with the basic price, and users keep quoting higher and higher prices till no one agrees for a higher price and one of them decides to buy it.

2) Implement an auth service using sockets. Clients can create / sign up an account. Once signed in, you can store key value pairs which you can return as per demand. Implement some form of hashing of password and encryption for kay value paid using a library. Also allow clients to login and upload files in the server and set proper access - either to the whole public or only to certain specific users.

3) Design a program using socket programming that implements the hexapawn game. It is a mini-version of chess. It is played on a 3x3 board. The goal of each player is to either advance a pawn to the opposite end of the board or leave the other player with no legal moves, either by stalemate or by having all of their pieces captured. Write appropriate server and client programs for the game and simulate the game playing.

Lab 2

1) Write a simple TCP iterative server and client to evaluate the given expression. The client enters an infix expression and sends to the server. There may or may not be spaces before and after the operators. For example, all of the following are valid expressions to enter: “13 + 42*5”, “10+2/4”, “5 + 6 - 3”. The server evaluates the postfix expression and sends the result back to the client. The client displays the result on the screen. It then prompts the user to enter the next expression.

2) Implement a mini chat app using socket programming. Clients must be able to chat with each other (as users can in WhatsApp and InstaGram). The server must supervise all these chats and enable communication between them.

3) Implement a 2 player Tic Tac Toe Game using socket programming. There should be at least 2 programs - client playing the game and the server, and a supervising server conducting the game. It should display proper result on game completion.

Lab 1

a). Create three programs, two of which are clients to a single server. Client1 will send a character to the server process. The server will decrement the letter to the next letter in the alphabet and send the result to client2. Client2 prints the letter it receives and then all the processes terminate.
b). Write a socket program to enable client1 to send a float value to the server. The server process should increase the value of the number it receives by a power of 1.5. The server should print both the value it receives and the value that it sends. Client2 should print the value it receives from the server.

c). Send datagrams with two arrays of integers (only even numbers) to a server. The server should check the data, whether there are odd and/or fraction numbers. If it is not, the server sums the elements of each array and puts the sum in a third array that is returned to the client. If the server discovers that the arrays have erroneous then the server does not reply. A timeout period should be established by the client such that retransmission occurs after the period expires.
d) Implement a port scanner using socket programming. The port scanner checks a number of ports (for instance, from 1 to 1026) to see if they are open (a server is listening on that port number) or closed (a server is not listening on that port number)