NATIONAL INSTITUTE OF TECHNOLOGY, WARANGAL
(An Institution of National Importance)
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
MID SEMESTER EXAMINATIONS, September, 2023
III B. Tech (Computer Science & Engineering) II Semester
CS304 : COMPUTER NETWORKS

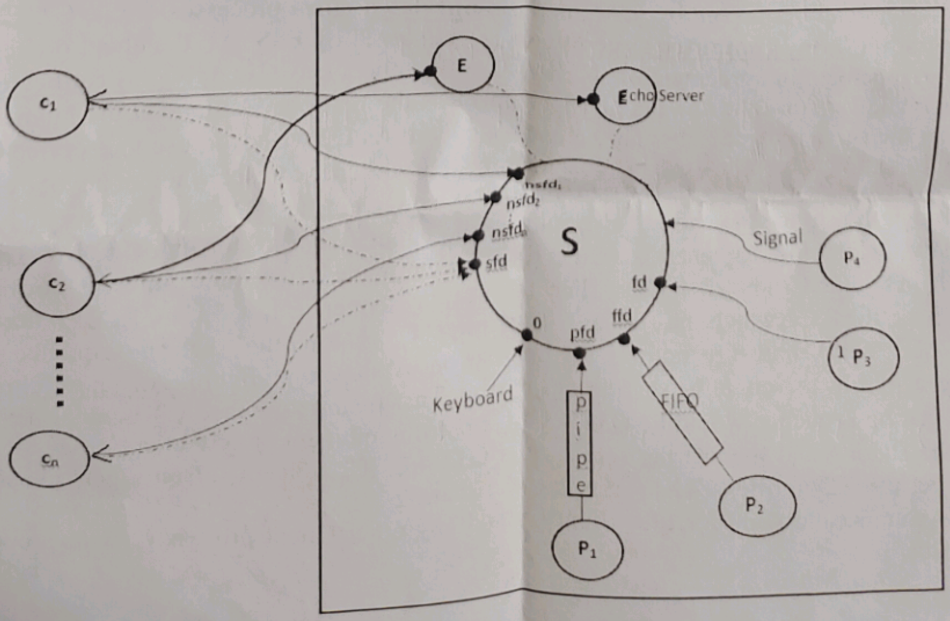
Date: 29– 09 - 2023 Time: 4.30 PM – 6.30 PM Max. Marks: 30

N.B.: Answer ALL questions
All programs/functions/code segments are to be written in C/C++.

1. Give the frame format of Ethernet and explain the significance of each field.
Within five collisions, what is the probability that the random value that a station chooses is 9 in binary exponential back off method?
A cell phone with only one mobile number (10 digits) is getting identified uniquely for connection. Then why we should have two address numbers i.e. Ethernet address of 48-bits, and IP address of 32-bits for uniquely identifying a computer? Can't we have a facility (arrangement) of using only oneaddress? If yes, then explain how? If no, then also clarify.
The first word of your answer should be either 'yes' or 'no' compulsory. Otherwise, the answer won't be evaluated.

2. **All-in-One Server**
A server process S gets input data from processes P1, P2, P3 and from keyboard(standard input). P1 is connected through pipe and P2 is connected through FIFO(named pipe) to S. Process P3 sends its standard output to S. The server S also listens for connection requests from connection-oriented clients on a socket file descriptor sfd and it accepts any such requests. If S gets data from keyboard or P1 or P2 or P3, it sends the same data to all connected socket file descriptors (nsfds). If S gets a signal from process P4, then it handovers an already accepted client connection on first-cum-first basis to a separate newly created echo sever process E. From then onwards the client will be served by E only. Likewise, whenever S gets a signal from P4, a separate echo sever process E will be created to serve an already connected client on first-cum-first basis

First write clear steps/flowchart, then write pseudo code program/program for the server process S only. The program should not use threads.
(No need of writing programs for P1, P2, P3, P4, Echo Server, and client.
No need of writing #include statements and socket address initialization statements).



F Si Ci

3. **Facilitator Super Server:** A Facilitator super server process – F, Service server process –S_i will be running in the same computer system. All Client processes – C_i are in different computer systems. 7.

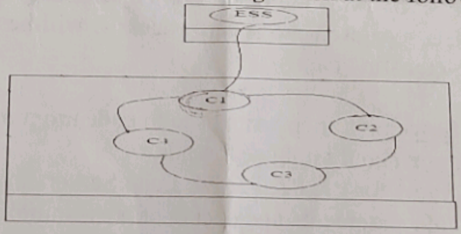
Process F initially does not contain any service points (sfds), but opens only one well-known(port) connectionless sfd. 'A developer codes S_i.C or S_i.CPP keeping in mind the working logic of process F. A user/developer **types input into** the process F in the format of (port number(m), /pathname/S_i.exe) soon after he stores/loads the S_i.exe file in the pathname. Process F reads this input and creates the process S_i. From then onwards, process S_i is ready to serve clients on port number m. Now process F includes a new service point, i.e. sfd$_i$ into the existing service points, i.e., sfds. This new sfd$_i$ is for this particular service S_i. All S_i services are connection-oriented only. That means, process F supposes to facilitate services (S_1, S_2, S_3, …,S_n) by listening to n socket fds. As soon as a client connect request arrives for a service (say S_i), {in other words, as soon as process F **observes** request for service S_i}, process F **notifies** process S_i. Then process S_i takes care of the client to serve by a service function thread.

A client can know the list of services available at process F, by sending an enquiry message (connection-less) to the well-known port of F. The process F sends the list. Further, whenever a new sfdi gets added, process F sends a message about this new sfdi (i.e its port number and brief service description of Si) to all the clients who have contacted/consulted it so far.

(**Hint**:setsockopt(sfd,SOL_SOCKET,SO_REUSEADDR||SO_REUSEPORT,&temp,size(int)) enables port reusability).

First write clear steps/flowchart, then write pseudo code program/program for process F and process S_i (a simple service functionality) with proper system calls.
No need of writing #include statements and socket address initialization statements.

4. **Exclusive Special Server (ESS)-circular sharing:** Look at the following figure.



8080
Si.exe
C1

ESS is a very special server such that **only one of** the client processes running at a particular computer system can get served by it. . Suppose, **unrelated** client processes of C1, C2, C3, C4 running at the **same computer system** wish to get served by ESS. All the client process know the arrangement of agreement of sharing. The first client (say **C1**) **only gets connected** to ESS as it knows (from the sharing arrangement) it is the first one. Assume at the moment C1 is using the service of ESS though the connected sfd. As C2 wishes to use the service of ESS, it joins in the circular sharing. After some time, C1 will check to see whether anybody is there in the sharing arrangement, and as C2 is there , so C1 notifies C2 to use the service of ESS (through the **same sfd**) and C1 also joins in the circular sharing for its next turn. That means now the sharing sequence is C2,C3,C4,C1. And again after some time C2 notifies C3(which has joined soon after C2). Then the sharing sequence is C3,C4,C1,C2. After a while C3 notifies C4(which has joined soon after C3). Now the sharing sequence is C4,C1,C2,C3, and C4 in turn notifies C1, and so on as shown in figure. Any new client process can join in the circle of sharing at the rear. That means, suppose C3 is using ESS, then the sharing sequence would be C3,C4,C1,C2. At this moment, a new client process C7 joins is the sharing and the sharing sequence is C3,C4,C1,C2,C7. So, C7 only gets chance after C2, because C3 notifies C4 as part of circular sharing.

Write program/pseudo code for first client C1 and any other Client process C_i with proper system calls.