Birla Institute of Technology & Science, Pilani
Pilani Campus

I SEMESTER 2020-2021
Assignment-2

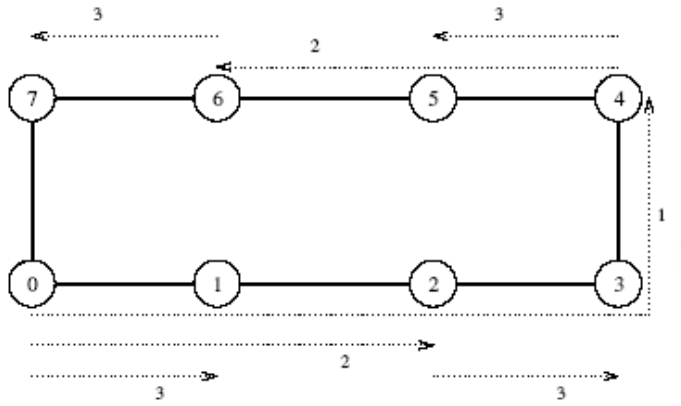
Course No.: IS F462 **Course Title:** Network Programming
Deadline: 22nd Nov 2020 **Maximum Marks:** 72M (18%)

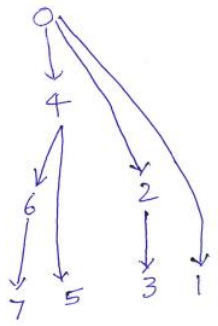
Note:

- Maximum of two students per group. Upload code in Canvas
- Name your file idno1_idno2_assignment2.tar .

P1. In this problem, mergesort algorithm needs to be implemented on a ring topology. N nodes are connected in a ring topology ($N=2^n$ where n is an integer. N is taken as CLA). The root node (specified on command line) sends half of the data to a node in the middle, which in turn sends half of the data to a node in middle of the rest. This is done until the data becomes singleton.

E.g.: Ring of 8 nodes.





Only one send() at a time.

```
Algorithm mergeSort(S, C)

        Input sequence S with n elements, comparator C

        Output sequence S sorted according to C

if S.size() > 1

        (S1, S2) ← partition(S, n/2)

        mergeSort(S1, C)        //should happen in nodex

        mergeSort(S2, C)        //should happen in nodey

        S ← merge(S1, S2)
```

Write coordinator.c run by root node and node.c by all nodes. The programs use TCP sockets for communication. Coordinator reads the file of integers, and initiates the transfer of data. All other nodes wait for the data and once received will split it among the nodes as specified above. The

coordinator prints the final sorted output to a file. Every node should print the source ip and port when they receive data. Also print the ip and port of node where they send it.

Deliverables:

- Brief Design Document (.pdf)
- coordinator.c, node.c

[24 M]

P2. TFTP (Trivial File Transfer Protocol) is used to transfer files between two machines. It doesn't need any authentication like FTP. Use UDP socket programming to implement a TFTP server that can handle Read Requests (RRQs). Write requests (WRQs) need not be implemented. TFTP protocol is defined in RFC 1350 [http://www.ietf.org/rfc/rfc1350.txt]. Use any TFTP client [command-line program 'tftp' installed on Linux machines] to test your TFTP server. The server tftpserver.c needs to achieve the following requirements:

- tftp server should accept a port number as command line argument over which it waits for incoming connections. It is a concurrent server using IO multiplexing i.e. using select () system call.

  [shell$ ./tftp 8069]

- Your server should print out information about every message received and every message sent with client identity. It uses RRQ, DATA, ACK, and ERROR messages wherever they are required. The sequence of steps required for downloading a file

1. The client C sends an RRQ (read request) to Server S (at the specified port), containing the filename and transfer mode.
2. S replies with a DATA message to C. Message is sent from a port other than the one it used for receiving RRQ and client needs to send further messages to S at this port. In each DATA, 512 bytes of data is sent.
3. All DATA messages are sequentially numbered. Once S sends DATA message, it will not send another DATA message unless it receives ACK for this DATA message. C responds to each DATA with corresponding ACK. Once S receives ACK for DATA number n, it sends DATA number n+1. If the ACK is not received within a specified time, S resends DATA.
4. C detects termination of data transfer if it receives DATA with <512 bytes size. If the total size of the file is exactly a multiple of 512, then a zero-size DATA is sent to indicate the termination of transfer.

- Your server should support binary mode (octet) of transferring data. Time-out and retransmission mechanisms should be used to ensure delivery of messages. You can use a timeout of 5 seconds for receiving acknowledgements. No of retries for retransmitting

can be utmost 3. Server should respond with appropriate ERROR message wherever it is required either when it receives RRQ or while transferring the file. See RFC for list of error options.

Deliverables:

- Brief Design Document (.pdf)
- tftpserver.c

[24]

P3. The program operates by sending UDP datagrams at increasing TTL and receiving the ICMP replies to determine the router's address at the respective TTL.

(a) Let us implement a fastertraceroute.c which finds out the addresses of the intermediate routers in parallel. There is one thread per one TTL i.e. one thread for TTL=1, another thread for TTL=2, another thread for TTL=3 etc. There is one single thread 'icmp' which reads ICMP replies over the raw socket. 'icmp' thread doesn't process the replies itself but delivers the received reply to the respective thread which has sent the datagram. Then individual threads process the reply and display the result. Design and implement this multithreaded program

(b) Given a list of domain names in a file (one per line), implement a C program findLongestCommonPath.c which finds out the longest path shared by the packets sent to these domains. The program should be using even-driven approach. It should not use threads but works like a batch mode client. It sends requests without waiting for replies for all domains. Processes replies as and when they arrive, without waiting for all the requests to be sent.

Deliverables:

- Brief Design Document (.pdf)
- fastertraceroute.c, findLongestCommonPath.c

[24]

--&--