ASSIGNMENT – III DATED 1-Sep-17

PREREQUISITE: You should have completed Assignment-II and understood the problems in that assignment.

POINTS TO NOTE:
a) Students whose roll numbers are ending with ODD digits, should complete the ODD numbered programs below and submit. Similarly students whose roll numbers are ending with EVEN digits, should complete the EVEN numbered programs below and submit
b) Programs should be complete in ALL respects and COMPILE AND RUN SUCCESSFULLY on a LINUX environment
c) Code should not be lifted out of internet and submitted and neither should it be copied between friends. Explanations on the implementation will be sought before awarding the credits for the assignment.
d) The assignment should be submitted to the email id iiitdmcn@gmail.com FROM your IIITDM account ON OR BEFORE 18-Sep-17

PROGRAMS:

1) Write a TFTP server using UDP that should understand the TFTP protocol. Implement a baisic TFTP features such as listing of files and download feature.Refer the TFTP protocol manual for more details.

2) Develop a HTTP Web server implementation that will be able to run FOREVER without getting affected by any interruption from a terminal. The Web server should also conform to the following specifications:

http://www.w3.org/Protocols/rfc2616/rfc2616.txt

Out of the above specification, implement ONLY the GET Optionr alone in your server. You may test your server with any standard browser like Chrome or Firefox. Alternatively, you can write a small client program that will accept a URL in the format of:

http://<domain name>:<Port number>/<file name> or

http://<IP Address>:<Port number>/<filename>

The client should establish a connection to your WEB server running on a particular port (default port number to be used when port number is not specified is last 4 digits of RollNo Id).

The data returned by your web server should be displayed correctly in your web browser or in your client program.

3) Implement a Client and a Server program meeting the following requirements.

Client specification: Your client should accept as a parameter from the command line: 1) the address of the server. If there are two parameters on the command line, then the two parameter is the port number to connect to at the server side (If 2nd parameter is not specified, assume the default port number to connect to is 4567). The order of the parameters in the command line should be in the order mentioned above.

The client should send the command in a TCP message to the server. Then the client should receive messages from the server and print all lines to stdout until no more lines are received. (Note that you can write the client output to a file by redirecting the output from the command line in Unix.)

The client, like a standard FTP client, should be able to also send the following commands to server and get the output of those commands back from server.

- `ls` to list the files in the current directory.
- `put <file_name>` to transfer a file from the client to the server.
- `get <file_name>` to transfer a file from the server to the client.

The output received from server for each of the commands should be displayed on terminal of the client.

Server specification: Your server should accept as a single optional parameter from the command line the number of the port to listen to. The server should receive the name of the file from the client. If the file cannot be opened in the current directory, then the server should return the string "File not found." to the client. If the file can be opened, then the server should send each line of the file to the client until the end of the file is reached. After sending the file, the server should close the file and then continue to listen for more client requests, in a concurrent server fashion. Your server should run indefinitely without any association with any terminal.

You server should print out appropriate informative messages as it executes. You should do appropriate error detection for the TCP socket calls. You may test to program to see if it runs correctly by performing a "diff" on the original file and the one that has been received via your FTP program. For convenience, you may assume that the files transferred by your program have line lengths that are 80 characters or less.

4) Write a DNS client and a DNS server. The client should accept two command line parameters, the first is the name (or IP address) of your DNS server, second is a hostname. Client should use a UDP DNS request to find the IP address of the hostname listed as the second command line parameter. Create your own **resolv.conf** to allow the client to know about your own DNS server or just hardcode the dns server ip.