COMP 6320 Design and Analysis of Computer Networks

Assignment 1
Groups of 2 Due by August 31
Programming Assignment (**Group,** 100 pts)

Lab 1: Introduction to Socket Programming

First, it is assumed that by Labor Day, you have an engineering Unix account, you can compile, and you can execute C programs on Engineering Unix machines (Tux machines). You can use any computing lab to access remotely Engineering Unix machines.

Second, **find one group mate and request from me a group id (*GID*).** You must implement and test your programming assignments on a machine (Tux machines) on the engineering network system. Your work will be tested and graded on an engineering machine (Tux machine).

I strongly advise you to read beej's guide at :
http://beej.us/guide/bgnet/ *(Simple, easy-to-understand tutorial)*

FOR ALL THREE LABS THIS SEMESTER, you must use C for one of the programs (client or server) and a different language for the second one (client or server).

Example: if you write a client in C, then you must use another language for the server.
Before starting, the class must agree on the communication protocols for these labs.

Lab 11: **Introduction to Socket Programming**

The objective is to get familiar with basic socket programming function calls. You will design and implement two separate programs: a server and a client. Server and clients will run on different machines and communicate on the Internet. You will also make some basic measurements of the round trip time for requests.

Part A: Datagram socket programming (**warm-up**)

a) Write a **concurrent** datagram **echo** server **S**. This server must send back any message you send to it. The server must "listen" on port 10010+GID and could run on any machine on the Internet. The server must be able to handle concurrently multiple requests. (**server11.c**)
b) Write a datagram client (**client11b.c**) which:
   i. Accepts as command line the name of the echo server
   ii. Prompts the user to enter a string **A**
   iii. Sends the string A to Server S
   iv. Listens for a response from Server S
   v. Prints out the response received from Server S
   vi. Prints out the round trip time (time between the transmission and the reception)
c) Write a client (**client11c.c**) with **two** processes:
   i. The first process runs a loop which sends continuously the numbers from 1 to 100,000 as strings
   ii. The second process receives the responses from the server and reports any missing echo.

Part B: Report whether you there are missing echos. Report the smallest, largest, and average round trip times when using (**client11c.c**).

Lab 12: Calculator

Write a TCP calculator server (**server12.c)** and a TCP calculator client (**client12.c**).

a) The TCP client must :

i. accept as command line two 32 bit unsigned integers (***a*** and ***b***) and a character ***c*** that can be: '+', '-', 'x', or '/'.

ii. bundle ***a***, ***b***, and ***c*** into an application **message *M*.** The size of the application message ***M*** should not exceed 10 bytes.

iii. send ***M*** to the server

b) The TCP server must:

i. receive message ***M***

ii. extract a, b, and c

iii. perform the operation requested

iv. send back the result with repeating the operands and operation. The size of the message ***M*** (at the application layer) should not exceed 16 bytes.

Grading:

1) 20 points per program (3 clients and 2 servers)
2) Code does not compile on Tux machine: 0% credit
3) Code compiles on Tux machines but does work: 5% credit
4) Code compiles and interacts correctly with counterpart from the same group: 70% credit
5) **Code compiles and interacts correctly with counterpart from other groups: 100% credit**

Advice to complete these exercises:

This is just an introduction to socket programming: I advise to work **ACTIVELY** to implement these programs.

Step 1: download, compile, and execute Beej's sample programs for Section 6.3 (talker.c (client) and listener.c (server) for the datagram (UDP) sockets)
Step 2: get familiar with this code: study key socket programming calls/methods/functions
Step 3: improve the server to echo (send back) the string it received.
Step 4: improve the client to receive the echo and print it out.
Step 5: SAVE the improved versions (you may need to roll back to them when things will go bad)
Step 6: All you need now is "forming" your messages based on the specified format.

For the TCP socket programming, redo Step 1-6 using Beej's stream sample programs (Sections 6.1-6.2)

What to turn in?

1) **Hard copy** of your lab report (with the code) with group id, students names and email addresses.
2) Electronic copy of your report and code. These sources codes named as shown above and your report must me put in a folder named lab1XX where XX is your group ID (on Canvas) by only one of the groupmates name. Zip the folder and post it on Canvas. Failing to submit the proper format will result in 25% penalty.
3) Your code MUST compile and execute on engineering machines tuxXYZ
4) Your report must:
    a. state whether your code works
    b. explain the TA how to compile and execute your code
    c. Responses when applicable (quality of writing and presentation will greatly affect your final grade when your responses are correct).
    d. report bugs/problems

If the TA is unable to access/compile/execute your work, no credit will be awarded.
If the turnin instructions are not followed, 25 pts will be deducted.