EEL 6935: Distributed Computing Spring 2017 Instructor: J. Fortes

Homework 3A Assigned: 3/02/2017 Due: 3/13/2017 @ 11 Pm (use UF E-learning - Canvas)

Remote Procedure Call (RPC)

Preamble:

In this homework, you are required to write a C program to implement a remote time server and a client that uses RPC to fetch the time in the remote server.

A remote time server has a single method that returns the current time in the server. This is used to synchronize the nodes in a cluster that is distributed geographically. Even in general computers these days, the system time is usually synchronized with the time in a remote server over the internet.

This assignment consists of 2 parts. In part A, you will be implementing a client-server architecture using RPC. The server in this case will be a time server that accepts requests from clients and returns the current time of the server over RPC. The client will contact the server and print the current time on the console. RPC in C can be divided into three layers. The time server could be implemented in any layer, refer to the RPC programming guide attached with this assignment. While choosing the layer, you need to remember to design your system flexible so that you can reuse your implementation for part B of the assignment. Flexibility here refers to configuring the transport protocols, modifying the timeout mechanisms, ability to capture the network details of the client at the server end etc. There are few questions that you must answer before implementing the system.

Tasks:

You will be implementing a time server in C. A time server exposes one method that returns the current time. For this assignment you will be returning the date and time from the server in the following format mm/dd/yyyy:HH:MM:ss AM/PM. The server will also be printing the details of the client, specifically the IP address and the port at which the client is running on the console.

The client will invoke the remote procedure and will print the value returned by the server.

To test your implementation, you will be running two terminals (Linux). In one of the terminals, you will start the server first and in the other terminal you will run the client. In short two processes, one of the server and the other of the client. When the client sends a request, the server will print the IP address and the port number of the client in the server's terminal before returning the value to the client. Once the client receives the timestamp from the server, the client will print the value on the client's terminal.

As part of your submission, you will submit screenshots that clearly show that you have tested your setup. For the submission purpose, you will be testing your implementation by running 5 clients from different terminals. To make it more clear, you will open 6 terminals, the server will be running in one of them and the other 5 terminals will be running the client. You will execute the clients sequentially (in any order) and your screenshot should capture all the information (server and client terminals). Points will be deducted if the screenshots are not convincing.

Questions:

1. What is the layer that you would choose to implement the system to be highly flexible? What are the options you get in the chosen layer?

2. When would you choose to implement:
   a. RPC over UDP mechanism
   b. RPC over TCP mechanism

3. Is it necessary for the client to send any parameters to the time server to fetch the current time? Explain.

4. In the examples portrayed in the guide, only one parameter is being passed and one parameter is being returned by the server. How would you make a call to the server in case multiple parameters need to be passed and multiple values are being returned from the server? Explain.

```
-    ./server
-    ./client <server_hostname>
```

Figure 1

Submission (*Monday, 03/13/2017, 11 Pm*):

1. Your source code containing the client code must be named rpcClient.c and the source code containing the server code must be named rpcServer.c. The names of the other supporting files, if any, are your choices.

2. Comment every module/function in your code and use descriptive variable names. Points will be deducted if the methods are not appropriately commented.

3. Your program MUST run using the command line arguments shown in the figure 1. The hostname of the server in the test cases will be localhost.

4. There are two files that you will be uploading as part of the homework submission. A tar file and a PDF file. **Do NOT tar the PDF file**.
   a. Upload the following files within a SINGLE tar file named “hw3.tar”:
      - The source code of your program
      - A file named “makefile” to compile the program. The TA will type only “make” to compile the program. Other methods are not allowed.
      - A text file named “readme.txt” that precisely specifies the running environment including the operating system, language written, compiler version and any software needed to run your program. It also describes the program structure such

as files, classes and significant methods. If your source code is in multiple files, describe briefly the content of each file.

- All these files should be tarred into a file i.e. the following command should work to create the tar file.

  *$> tar cvf <hw3>.tar **

b. A PDF document (hw3.pdf) containing answers to the questions and your source code. The source code MUST be pasted towards the end of the PDF. Include all your modules/functions in the PDF file.

   NOTE: The PDF file MUST NOT be tarred. You will have two files to submit. A tar that contains your source code, readme file and a make file, and a PDF file that contains plots, answers to questions, definition of distance for the crescent moon and pasted text of your source code.

5. The tar file and the PDF must be uploaded to Canvas.

<u>Submission Policy:</u>

- Do NOT include binary files. Use the file names as specified above. Incorrect submission formats will lead to a grade reduction.
- All submissions are expected by the deadline specified in the homework assignment. Grade is automatically reduced by **25% for every late day**.
- Make sure to test your submitted code using the tar file. If untar, make, compile or any other command needed to executed your program do NOT work, your homework grade will be **zero.**