DSPro

Optional Practical on Distributed Systems

General guidelines

- Each exercise is worth 0,2 points out of a total of 1 points which is the credit assigned to the present homework in this academic year. You have no obligation to deliver the full homework assignment; any part thereof will count if properly explained and developed.
- Program sources must be clearly and succinctly commented
- Apply a simple OO design strategy or structured programming strategy depending on whether you are programming in Java or in C, respectively
- Include rich explanations of your design decisions and the unit tests that demonstrate the correction of your programs
- You can only submit your original work, programs and explanations. You can incorporate source code from open software projects, in which case you must cite the authors and their overall weight must be small.
- Submit the solution to **each exercise** in a **separate folder** which name must be "Exercise 1", "Exercise 2", etc. Each folder must contain one folder for the sources (**src**), another folder for the explanations (**docs**) and a last folder for the executable files (**bin**)
- Include software construction instructions and shell scripts if necessary
- Compress the complete folder structure mentioned above in a .zip file (Please, use .zip exclusively, otherwise, I might not be able to decompress the archive which might damage your grade)
- Lab B6 will remain open all the morning of Tuesday 19th for those students who might need to conduct experiments for DSPro with the Lab PCs and networks
- Submit the zip-compressed archive to **foces.informatica.unileon@gmail.com** by Wed 20th/November/2019 21:00:00 CET (UTC + 1)

1. Simple probabilistic time synchronization algorithm

Using ICMP timestamps and Raw IP Sockets, build a C program that synchronizes its host's clock (The client) with another host of your choosing (The server). The client program will fetch the server's time a number of times and after the last one, it will set its own clock in sync with the server's.

a. The program must capture the server name or IP address from the command line along with the number of times to retry synchronization with

the server.

b. The NTP client must be stopped before your program can run, just to make sure that NTP doesn't interfere with your programs, somehow

c. The source code provided in the Lab Practices of DS can be used as the basis for developing this program but you must mention in the comments that some parts of your code are not original. If you use the base code provided in the practice scripts, you must be aware that that code might contain semantic errors and, consequently, you are expected to correct them. Particularly, check that the given source program faithfully implements the Christian's time adjustment procedure explained in the lectures of the present academic year, 2019 (Observe the timestamps used for performing the adjustment).

d. Explain what tests you will perform to demonstrate that the program functions correctly. Particularly, we are interested in your building a program that continually prints out the current server time along with the client –local- time so that the user can easily observe the progress of **adjtime()** (Every 2 seconds, for example). Set an initial time difference between the two clocks not greater than 15 seconds, otherwise the time length taken by the **adjtime()** system call will be too long.

2. RFC 868 time protocol

Skim the RFC 868 time protocol and implement the client/server pairs appearing on the adjoining table. In each C/S pair, the client will request the current time to the RFC 868 time server and will print out the response. Note, in this case we are not seeking to synchronize any clocks, but only to print out the server time in the client.

Implemented over	Client	Server
TCP	C	Java
UDP	Java	C
JRMP	Java RMI	Java RMI

a. Deliver each C/S pair in a separate folder containing the corresponding sources, executable code and documentation

b. Provide extensive explanations about the considerations you made about this question

c. Since the RFC 868 time format must be the same across the three requested implementations, were you able to reuse some of the data structures or the

algorithms involved? If so, explain how you did it.

d. Explain the tests that you designed to check the correctness of the implementations

3. RMI C/S and concurrency

Write an RMI server that keeps a remote counter of certain events. Your task consists of building a Java RMI Server that implements the three following remote methods:

```
public void reset(); // Reset the counter to 0
public long int increment(); //Increments the counter (counter++)
public long int decrement(); //Decrements the counter (counter--)
```

Upon server initialization, the counter must be initialized to 0 by using method reset(). You must use only simple Java synchronization primitives such as synchronized blocks and methods:

a. Explain the general strategy that you devised for guaranteeing atomic writes on the counter variable

b. Write an RMI client program that permits testing the server

c. Provide an extensive discussion of your software design and the tests that demonstrate that it functions correctly

4. TCP timestamps option in RFC 7323

Compose a summary about the TCP timestamps based on section no. 3 of RFC 7323 titled "TCP Timestamps Option". Focus on the usefulness of this option, particularly on the hardest challenges posed by it.

5. RMI lab practice script activities

In this exercise, we seek to improve the RMI client/server practice that we did in the lab according to the following outline:

a. Clients don't know the location of the RMI server. Clients find out the

location of the RMI server by sending a request to the Local Network broadcast address and UDP port 1234. This entails the RMI server to listen for these requests when it boots. When the server receives such a broadcast request, it reacts by providing the client a response string containing the IP address and the TCP port on which the RMI registry is located.

b. When the client has fetched the IP/Port of the RMI registry, it will proceed in the manner explained in the foregoing practice script (RMI C/S). The client will invoke the remote methods and print out the results.