Course on Distributed Systems at Universidad de Leon

School of Industrial, Computer and Aerospace Engineering

DSPro

Independent Study Exercises on Distributed Systems

General guidelines for solving the exercises

- Each exercise is worth 0,20 points out of a total of 1,00 points which is the credit assigned to the present homework in the course. You have no obligation to deliver the full homework assignment; any part thereof will count if properly explained and developed. *The mentioned credit does not contribute to the passing mark of the course, that is the sense in which this practice is optional.*

- Some exercises require *remote* access to two servers connected to the network in Lab B6. The servers will be started on Tuesday 28th-November-2023, one day after I give you the lectures necessary for understanding the aforementioned exercises. Keep this on mind. The specific technical data (IP addresses; port numbers; protocol numbers, etc) for accessing them will be available on this link on the given date:

  - `paloalto.unileon.es/ds/dspro2023_techdata.txt`

- The right to submit DSPro and that it be assessed is only granted to those students who have attended the practice lab sessions regularly.

- Program sources must be clearly commented.

- Apply a simple OO design strategy or structured programming strategy dependingrogramming in Java or in C, respectively.

- Include rich explanations of your design decisions and the unit tests that check the correction of your programs.

- Only your original work produced personally by you may be submitted. 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.

- Include build files written either in make, gmake or in Ant.

- Compress the solution to all of the exercises in a single .zip file (***Only .zip is accepted!!!***)

- 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 compromise a passing grade).

- Submit the zip-compressed archive to the agora task titled ***Homework #3: DS Pro 2023:***

    - *Pub. Date: 17th-November-2023*
    - *Submission deadline: 30th-November-2023 22:00*

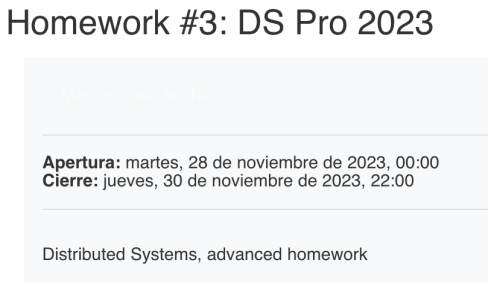

Figure 1. Agora task for DSPro 2023

1. New, simple ping utility for Linux

i. Develop a simple *ping* utility for the Linux operating system in C or in Java based on ICMP echo messages.

ii. Explain the difficulties that you have found and the development itself.

iii. Use structured programming and have your code appropriately commented.

2. ICMP Timestamp Requests over Internet

The servers in Lab B6 are not authorized to expose ICMP Timestamp facility over Internet. The ICMP Timestamp message, when sent from clients over Internet, is not finally delivered to the destination (Server) host because some of the routers comprising the path to it, will drop ICMP Timestamp Requests. In this exercise, we ask you to suggest *reasonable, sensible and feasible* strategies for circumventing this security restriction.

You might start with observing the similitude between the ICMP headers to the echo messages (Those used by the ping utility) and those to the timestamp messages, and by carefully studying the RFC to ICMP.

i. What would be necessary for passing timestamp requests/replies as echo and echo replies?

ii. Explain how you would program a server that would treat *our* ICMP echos as timestamp requests and respond accordingly.

iii. Explain your strategy for programing this in C

iv. Now, simply write a program that creates a PF_INET/RAW_SOCK socket for receiving ICMP echos and have them printed out on the server side. No ICMP response should be sent back, we simply seek out to demonstrate you ability to have echos received by your program.

v. Is this possible, actually? Discuss whether or not it is possible to have a *server* program receive IP Protocol 1 (ICMP) messages. Recall that those messages are handed to Linux's ICMP protocol module.

3. Simple probabilistic time synchronization algorithm

Using ICMP timestamps, build a C program that synchronizes its host's clock (The client) with the clock of another host of your choosing (The server). Accessing the time server entails your working in Lab B6 in-person where you'll have plenty of hosts to choose your server from.

On Tuesday (21st-Nov-23) I'll send you notice of the possibility of doing this exercise remotely

Initially, the client program will fetch the server's time 10 times with 1 min between each attempt. Each of those times it will calculate the minimum achieved Rtt and record it for later reference. Then, the program will start over a new batch of maximum 10 times fetching the server time and, whenever it achieves an Rtt

that is less than the minimum Rtt calculated earlier, it will do the synchronization of the local clock by applying the formula explained in the lectures and print out the max synchronization error, as well.

For this latter print out, first, suggest a reasonable lower bound for the one way delay ($Rtt \approx D_{fwd} + D_{back}$) assuming that the internetwork routes are fully symmetric. Maybe, use the ping utility for obtaining Rtt statistics that might help you.

The program must print out the following items:

i. The delta resulting from each timestamp reply received from the server

ii. The mean delta; the std deviation; the minimum Rtt

iii. The local time before invoking adjtime(), *i.e.,* whenever adjtime() is called to have the clock synchronized

iv. The local after invoking adjtime()

Overall procedure for checking your client:

a. Stop your NTP client altogether. Explain what you do to stop ntp. Consult the practice that we did in the course practices where we provided you details about how to stop NTP.

b. Highlight the Linux commands involved in managing the local clock that you used to perform the tests.

c. How long does adjtime() take for reaching a target time that is 5 min forward? Devise an experiment to demonstrate that your results are reasonable.

d. Explain what tests you will perform to demonstrate that the program functions correctly.

4. RMI C/S

Write a Java RMI client that checks access to two remote methods at the host at IP address 193.146.101.127 (**longStringHash**(String) and **factorial**(int)). The server

and the remote rmiregistry are deployed in the server host given above. Download the zip file containing the source tree and, afterwards unzip the zip archive:

1. You must have Java JRE and JDK installed.
2. Download zip from:

   ```
   wget http://paloalto.unileon.es/ds/lab/ant.zip
   ```

3. Unzip the archive:

   ```
   $ unzip ant; cd ant; ./install.sh
   ```

4. Skim the ant script file and use the clauses that have the client run.

   The Server and the rmiregistry port numbers will be published next Tuesday evening.

5. Article reading. Select one of the following two articles and make a *useful, short* 1-page summary of it.

1. Article on Google's transport protocol, QUIC. Highlight the differences with TCP from the viewpoint of DS.

   paloalto.unileon.es/ds/quic.pdf

2. Who unlawfully uses IPv4 addresses? Who squats IPv4 addresses?

   paloalto.unileon.es/ds/ipv4addr.pdf