laSalle ENG
Universitat Ramon Llull

Computer Engineering Area

Subject: Distributed Architecture Projects

Date: October 5th, 2021 **Academic Year:** 2021/2022

Suggested delivery date: November 16th, 2021

Proposed assignment. The distributed mutual exclusion

You have to design and implement a distributed application composed by two *heavyweight* programs: `ProcessA` and `ProcessB`. `ProcessA` has to manage three *lightweight* programs `ProcessLWA1`, `ProcessLWA2` and `ProcessLWA3`. Similarly, `ProcessB` has to manage two *lightweight* programs `ProcessLWB1` and `ProcessLWB2`. Each *lightweight* program has to live in an infinite loop that will consist off showing on the screen their program identifier 10 times waiting one second between two consecutive prints.

As all the programs will run in the same physical machine, they will compete for the same shared resource: the screen. Hence, between two *heavyweight* programs you will have to implement a *token-based* mutual exclusion. Every time a *heavyweight* process owns the token, will have to tell its associated lightweight processes `Process`A_i or `Process`B_i that they have to compete for the screen. Processes `Process`A_i have to implement the `Lamport` algorithm for mutual exclusion. Processes `Process`B_i have to implement the *Ricart and Agrawala* algorithm for mutual exclusion.

In this way, when we execute this distributed application, the screen might look as follows:

```
I am the process lightweight A1
I am the process lightweight A1
....
I am the process lightweight A1
I am the process lightweight A1
I am the process lightweight A2
I am the process lightweight A2
....
I am the process lightweight A2
I am the process lightweight A2
I am the process lightweight A3
I am the process lightweight A3
....
I am the process lightweight A3
I am the process lightweight A3
I am the process lightweight B1
I am the process lightweight B1
....
I am the process lightweight B1
I am the process lightweight B1
I am the process lightweight B2
I am the process lightweight B2
....
```

```
I am the process lightweight B2
I am the process lightweight B2
I am the process lightweight A1
I am the process lightweight A1
....
```

Although the *heavyweight* and *lightweight* processes have to run in the same physical machine, these programs must be independent and can only communicate between each other by means of *sockets*.

An example of the skeleton of a *lightweight* process is shown below:

```
...
while(1){
    waitHeavyWeight();
    requestCS();
    for (int i=0; i<10; i++){
        printf("I am the process lightweight %s\n", myID);
        espera1Segon();
    }
    releaseCS();
    notifyHeavyWeight();
}
...
```

An example of the skeleton of a *heavyweight* process is shown below:

```
...
while(1){
    while(!token) listenHeavyweight();
    for (int i=0; i<NUM_LIGHTWEIGHTS; i++)
        sendActionToLightweight();
    while(answersfromLightweigth < NUM_LIGHTWEIGHTS)
        listenLightweight();
    token=0;
    sendTokenToHeavyweight();
}
...
```