Distributed Computing: Spring 2024

Programming Assignment 2: Mutual Exclusion

Submission Deadline: 23rd March 2024, 21:00 hrs

1 Problem Statement

The objective of this assignment is to implement and compare the performance of Maekawa's Algorithm and Roucairol and Carvalho's Algorithm for achieving mutual exclusion in a distributed system. The above problem needs to be implemented in C++, using **sockets** or **MPI** or **zeroMQ** for communication between the cells/processes.
For simplicity, you can assume Grid quorum for implementing Maekawa's algorithm.

2 Model Specifications

You are given as input a system of n nodes (processes) which are completely connected to each other. The nodes communicate with their neighbors through messages as a part of the application which is described below.

Each process performs some local computation and then decides to enter mutual exclusion. The local computation can be simulated by sleeping for an exponential time with an average of α. Each process requests to enter the mutual exclusion k times. A process spends sometime inside the critical section (CS) that is exponentially distributed with an average of β. You can also simulate the time that a process spends inside the CS by sleeping. Once all the processes have entered and exited CS k times, you can terminate the execution. Note that $\alpha < \beta$.

Algorithm 1 Process *pi*'s execution

```
1:  procedure WORKING
2:      //Execute until the termination condition is reached
3:      for i ← 0 to k − 1 do
4:          int outCSTime ← rand(α)
5:          int inCSTime ← rand(β)
6:          //Represents pi's local processing
7:          sleep(outCSTime)
8:          //Requesting the CS
9:          reqCS()
10:         //Inside the CS
11:         sleep(inCSTime)
12:         //Releasing the CS
13:         relCS()
14:     end for
15: end procedure
```

3 Input

The input to the program will be a file, named inp-params.txt, consisting of all the parameters described above. The first line of the input file will contain the parameters :n, k, α, β.

Since we are considering that Maekawa is implementing grid quorum, you can assume that n is a square number.

4 Output

Each process should log all the messages exchanged onto a logfile. It should show all the messages sent, messages received by the process. The contents of the logfile of process p1 can be as follows for both the algorithms:

p1 is doing local computation at 10:00
p1 requests to enter CS at 10:05 for the 1st time
p1 receives p2's request to enter CS at 10:06
p1 reply to p2's request to enter CS at 10:07
.
.
.
p1 enters CS at 10:15
.
.
.
p1 leaves CS at 10:18 for the 1st time

5 Report

You have to submit a report for this assignment explaining the design and the implementation details. You must run both the algorithms, and conduct the following experiments.

Experiment 1: Message Complexity In the first experiment, we aim to compare the message complexity of MK Algorithm and RC Algorithm. The x-axis represents the number of processes involved in the system, ranging from a small number to a larger one. Meanwhile, the y-axis indicates the average number of messages exchanged per critical section entry. As the number of processes increases, we expect to observe how each algorithm scales in terms of message complexity. Vary the n from 5 to 25 processes and keep the k constant at 15.

Experiment 2: Scalability and Throughput In the second experiment, we will be focusing on scalability and throughput. Here, the x-axis denotes the level of system load, which can be quantified by the rate at which processes request access to the critical section. The y-axis measures the throughput of the system, representing the number of critical section entries completed per unit time. By plotting throughput against system load for both algorithms, we can analyze how each algorithm copes with higher demand and identify potential bottlenecks or limitations in their scalability. For this experiment let n be 10, and vary k from 5 to 25 requests per process.

Simulation Instructions: Please test your experiments on the VMs assigned to each of you. Also, please run the experiment 5 times for each point and then average it.

6 Deliverables

You have to submit the following:

1. The source files containing the two programs to execute i.e MK-⟨RollNumber⟩.cpp and RC-⟨RollNumber⟩.cpp
2. A readme.txt that explains how to execute the program.
3. The report as explained above.
4. A inp-params.txt as input file.

Zip all the files and name them as ProgAssn2-⟨rollno⟩.zip. Please follow the naming convention strictly. Otherwise, your submission will not be evaluated. Then upload the zip on the google classroom page of this course. Submit it by the above mentioned deadline.

Note: Please follow all the instructions given for the assignment strictly (input-file format, src file names, output format, reports, etc.); otherwise, the TA may not evaluate your assignment.

Evaluation Criteria: The following will be the criteria for evaluation:

1. Design: The program design along with the report as described above will carry 50 marks.
2. Execution: The program execution will carry 40 marks.
3. Code Indentation and Documentation: The indentation and documentation of the code will carry 10 marks.

Late Submission Penalty:
For each day after the deadline, your submission will be penalized by 10 marks until a maximum of 6 days beyond which your submission will not be considered.

7 Plagiarism Policy

If we find a case of plagiarism in your assignment (i.e. copying of code from each other, in part or whole), you will be awarded zero marks. Note that we will not distinguish between a person who has copied, or has allowed his/her code to be copied; both will be equally awarded zero marks for the submission. Follow below link for more information about plagiarism policy: https://cse.iith.ac.in/academics/plagiarism-policy.html