CS/CE/TE 6378: Advanced Operating Systems
Section 002
Project 3

Instructor: Neeraj Mittal

Assigned on: Wednesday, March 29, 2017
Due date: Wednesday, April 26, 2017

This is a group project. A group must consist of at least two and at most three students. *Code sharing among groups is strictly prohibited and will result in disciplinary action being taken.* Each group is expected to demonstrate the operation of this project to the instructor or the TA.

You can do this project in C, C++ or Java. Since the project involves socket programming, you can only use machines dcXX.utdallas.edu, where XX $\in$ {01, 02, .., 45}, for running the program. Although you may develop the project on any platform, the demonstration has to be on dcXX machines; otherwise you will be assessed a penalty of 20%.

1 Project Description

Implement a *mutual exclusion service* using two different distributed mutual exclusion algorithm: (1) Lamport's algorithm, and (2) Ricart and Agrawala's algorithm. Your service should provide two function calls to the application: csEnter() and csLeave(). The first function call csEnter() allows an application to request permission to start executing its critical section. The function call is blocking and returns only when the invoking application can execute its critical section. The second function call csLeave() allows an application to inform the service that it has finished executing its critical section.

Implementation Details: Design your program so that each process or node consists of two separate modules. The top module implements the application (requests and executes critical sections). The bottom module implements the mutual exclusion service. The two modules interact using csEnter() and csExit() functions.

Application: The application is responsible for generating critical section requests and then executing critical sections on receiving permission from the mutual exclusion service. Model your application using the following two parameters: inter-request delay and cs-execution time. The first parameter denotes the time elapsed between when a node's current request is satisfied and when it generates the next request. The second parameter denotes the time a node spends in its critical section. Assume that both inter-request delay and cs-execution time are random variables with exponential probability distribution.

Testing: Design a mechanism to *test* the correctness of your implementations. Your testing mechanism should ascertain that at most one process is in its critical section at any time. It should

be as *automated* as possible and should require minimal human intervention. *For example, visual inspection of log files to verify the correctness of the execution will not be acceptable.* You will be graded on how accurate and automated your testing mechanism is.

2 Submission Information

All the submissions will be through eLearning. Submit all the source files necessary to compile the program and run it. Also, submit a README file that contains instructions to compile and run your program.

3 Configuration Format

Your program should run using a configuration file in the following format:

The configuration file will be a plain-text formatted file no more than 100KB in size. Only lines which begin with an unsigned integer are considered to be valid. Lines which are not valid should be ignored. The configuration file will contain $n + 1$ valid lines.

The first valid line of the configuration file contains *four* tokens. The first token is the number of nodes in the system. The second token is the mean value for inter-request delay (in milliseconds). The third token is the mean value for cs-execution time (in milliseconds). The fourth token is the number of requests each node should generate.

After the first valid line, the next n lines consist of three tokens. The first token is the node ID. The second token is the host-name of the machine on which the node runs. The third token is the port on which the node listens for incoming connections.

Your parser should be written so as to be robust concerning leading and trailing white space or extra lines at the beginning or end of file, as well as interleaved with valid lines. The # character will denote a comment. On any valid line, any characters after a # character should be ignored.

You are responsible for ensuring that your program runs correctly when given a valid configuration file. Make no additional assumptions concerning the configuration format. If you have any questions about the configuration format, please ask the TA.

Listing 1: Example configuration file

```
5 20 10 1000

0 dc02 1234
1 dc03 1233
2 dc04 1233
3 dc05 1232
4 dc06 1233
```