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

Instructor: Neeraj Mittal

Assigned on: Thursday October 29, 2015
Due date: Thursday December 3, 2015

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 Lamport's distributed mutual exclusion 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.

Use the mutual exclusion service to implement a *totally ordered broadcast service* in which every process delivers broadcast messages sent using the service in the *same* order. Note that even the sender will deliver its own broadcast message it sends using the service. The broadcast service should support two function calls: tobSend() and tobReceive(). The first function call tobSend(String m) allows an application to broadcast a message m to all processes (including itself). The function call is non-blocking and returns as soon as the broadcast service has copied the message into its buffer. The second function call tobReceive() allows an application to deliver a message. The function call is blocking and returns only when there is a message waiting to be received.

Implementation Details: Design your program so that each process or node consists of three separate modules organized in a hierarchical fashion. The top module implements the application (broadcasts and delivers messages). The middle module implements the totally ordered broadcast service. The bottom module implements the mutual exclusion service. The top two modules interact using tobSend() and tobReceive() functions. The bottom two modules interact using csEnter() and csExit() functions. *It should be possible to swap the implementation of a module in your program with our own implementation of the same module, and your program should still compile and run correctly!*

Application: Each node broadcasts a certain number of messages one-by-one using the totally ordered broadcast service. Note that multiple nodes may invoke tobSend() function concurrently. A message contains a number chosen uniformly at random. Every node maintains a separate output file that stores the values received by that node in the order in which they are received with each value stored on a separate line.

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 **three** tokens. The first token is the number of nodes in the system. The second token is the number of messages each node will broadcast. The third token is the minimum delay between two consecutive invocations of tobSend() function by the same node. 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 10 # numNodes numMessages

0 dc02 1234   # nodeID hostName listenPort
1 dc03 1233
2 dc04 1233
3 dc05 1232
4 dc06 1233
```

4 Output Format

If the configuration file is named `<config_name>.txt` and is configured to use n nodes, then your program should output n output files, named in according to the following format: `<config_name>-<node_id>.out`, where `node_id` $\in \{0, ..., n-1\}$.

The output file for process j should be named `<config_name>-j.out` and should contain the following: If there are n nodes in the system, and each node sent k messages, then there should be nk lines in total. The i^{th} line should consist of the i^{th} random number which was received by process j via totally-ordered broadcast.

Listing 2: Example output file

```
12356
1887
13482
5257
...
```

During the demo, the TA will ask you to write a simple command which tests the correctness of your project. You should be prepared to do this. You may write a script ahead of time and turn it in with your project. It is possible to test your project on `csgrads1` from the command line using a single command.