CS/CE/TE 6378: Project II

Instructor: Ravi Prakash

Assigned on: March 7, 2013
Due date and time: March 23, 2013, 11:59 pm

This is an individual project and you are expected to demonstrate its operation to the instructor and/or the TA.

1 Requirements

1. Source code must be in the C /C++ /Java programming language.
2. The program must run on UTD lab machines (`net01, net02, ..., net50`).

2 Client-Server Model

In this project, you are expected to use *client-server* model of computing. You will need to know thread and/or socket programming and its APIs for the language you choose. Make sure that each node is running on a single machine (`netXX`). Please get familiar with basic UNIX commands to run your program on *netXX*.

3 Description

Implement Maekawa's distributed mutual exclusion algorithm with deadlock handling [1].There are N nodes in the system, numbered from 0 to $N-1$. The requirements for each node are:

1. Each node communicates through persistent TCP connections to other nodes.
2. It reads an input file ("input.txt") and populates its quorum. The file also contains inter-request time (IR) and critical section time (CST), both in milliseconds. IR is the time between an exit from the critical section and the next request for critical section entry. CST represents the time interval that each process will spend in the critical section, once it gets the access.
3. Each process generates M number of CS requests (given in "input.txt").
4. Once it gets access to CS, it sends a message to a designated node, *CSNode*. The message contains `<NODE-ID,REQ-ID,CST>`. The CS execution finishes when the process receives ACK from *CSNode*.
5. Node zero has additional responsibility to send an initial message to all other nodes to start the execution. It also collects and display the collected data from all nodes (including itself) and indicates termination of the program.

4 Critical Section

Implement a separate program, *CSNode*, that runs on a separate machine. *CSNode* accepts a TCP connection on a listening socket, spawn a thread to execute the CS request. It opens a file "logs.txt". For execution, it does following in the given sequence:

1. Write `<SENDER-ID,REQ-ID,START-TIME>` to the file.
2. Sleeps CST amount of time.
3. Write `<SENDER-ID,REQ-ID,FINISH-TIME>` to the file.
4. Sends ACK to the sender notifying that CS request is done.

5 Data Collection

For your implementation of the mutual exclusion algorithm, each node must collect the following:
For each of its CS request:

1. Number of REQUESTs sent.
2. Number of REPLYs received.
3. Number of RELEASEs sent.
4. Number of FAILs received.
5. Number of ENQUIREs received.
6. Number of YEILDs sent.

Once a process finished M CS requests it send the collected data to node zero.

6 Submission Information

The submission should be through *elearning* in the form of an archive (.zip, .tar or .gz) consisting of:

1. File(s) containing only the source code.
2. The README file, which describes how to run your program.

NOTE: Do not submit unnecessary files.

References

[1] Maekawa Mamoru, *A $\sqrt{N}$ algorithm for mutual exclusion in decentralized systems*, ACM Trans. Comput. Syst., pg.145–159, vol.-3, No.2, 1985.

7 Program Testing

A sample input file, ”input.txt”, is given below. However, you program will be tested on other files. The input file assumes that zero based continuous IDs are given to the nodes.