

CS 6378: Project II

Instructor: Ravi Prakash

Assigned on: March 8, 2023

Due date: March 29, 2023

You are required to implement a tree-based quorum system as described below.

You are expected to write your own code without help from any other person or source. Any deviation from this requirement may trigger an investigation of academic misconduct.

The program must run on UTD machines (dc01, dc02, ..., dc45).

1 Requirements

1. There are seven server nodes in the system, numbered from S_1 to S_7 . The servers are logically arranged as a binary tree with S_1 as the root. The children of server S_i are nodes S_{2i} and S_{2i+1} .
2. There are five client nodes in the system, numbered from C_1 to C_5 . Each client independently generates its requests to enter the critical section by executing the following sequence of operations until twenty of its requests have been satisfied:

- (a) Waits for a period of time that is uniformly distributed in the range $[5, 10]$ time units before trying to enter the critical section.
- (b) Sends a REQUEST to all the servers and waits for GRANTS.
- (c) Once a REQUEST has been granted by **any one of the quorums of the tree**, the client can enter its critical section. A quorum of the tree is recursively defined as follows:
 - i. Root of the tree + a quorum of left subtree, or
 - ii. Root of the tree + a quorum of right subtree, or
 - iii. A quorum of left subtree + a quorum of right subtree.
 - iv. If the tree is a singleton node, then the tree's quorum consists of that node.

As you can see this tree of seven nodes can be divided into multiple quorums, not all of the same size.

- (d) On entry into the critical section the client does the following:
 - i. Prints "entering" followed by client number and current physical time on a new line,
 - ii. Let 3 units of time elapse, and then exits the critical section.
 - iii. Sends RELEASE message to all servers.
 - (e) Once a client has successfully exited the critical section twenty times it sends a *completion notification* to server S_0 .
3. The servers act as follows:
 - (a) Initially, all servers are in an unlocked state.
 - (b) If an unlocked server receives a client's request, the server gets locked by that client and sends a GRANT message to the client.
 - (c) If a locked server receives a client's request, the server cannot send a GRANT message. Instead, it adds that client's request to its queue ordered by the timestamp of the request.

- (d) If a server receives a RELEASE message from a client:
 - i. If the server is locked by that client, the server checks its queue. If there is at least one REQUEST in its queue, the server dequeues the REQUEST at the head of the queue, sends a GRANT to the corresponding client and stays in the locked state. Otherwise (i.e., queue is empty), the server becomes unlocked.
 - ii. If the server is not locked by that client, and has the client's REQUEST in its queue then it removes that REQUEST from its queue.
 - (e) S_0 brings the entire distributed computation to an end once it has received *completion notification* from all the clients.
4. There are reliable socket connections (TCP) between each pair of nodes. The messages pertaining to the mutual exclusion algorithm are sent over these connections.

2 Data Collection

For your implementation of the mutual exclusion algorithm report the following (either show it on the screen or write it to a file):

1. The total number of messages sent by each node from the beginning until it sends the *completion notification*.
2. The total number of messages received by each node from the beginning until it sends the *completion notification*.
3. For each node, report the following for each of its attempts to enter the critical section:
 - (a) The number of messages exchanged.
 - (b) The elapsed time between making a request and being able to enter the critical section (latency).
4. Rerun your experiment with other values of: (a) time between a client exiting its critical section and issuing its next request, and (b) time spent in the critical section. Report what impact do these changes have on performance, specifically latency. Also, report if you come across any deadlock situation.

3 Point Distribution

Implementation (50%): Source code of your well structured and well documented program. You may write your code in C or C++ or Java.

Correctness (50%): Output that your program produces and the performance data and its analysis.

4 Submission Information

Submission instructions will be posted on the course discussion forum on WebCT.