# CS 6378: Project II

Instructor: Ravi Prakash

Assigned on: March 14, 2019
Due date: April 4, 2019

**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 is a file (you can choose any name for that file) that each client node can write to. Initially, the file is empty.
2. 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 $S_{2i+1}$.
3. 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 a quorum of servers and waits for GRANTS. A quorum of a binary tree of servers 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.

      Multiple quorums are possible in this system of seven servers. Each time a client makes a REQUEST, it should randomly select one of the quorums to send the REQUEST to. The goal is to select different quorums over the course of this project.
   (c) Once a REQUEST has been granted by all nodes in the selected quorum, the client can enter its critical section.
   (d) On entry into the critical section the client does the following:
      i. Opens the file mentioned above.
      ii. Writes "entering" followed by client number and current physical time in a new line at the end of the file.
      iii. Lets 3 units of time elapse, closes the file, and then exits the critical section.
      iv. Sends RELEASE message to all servers in the quorum.
   (e) Once a client has successfully exited the critical section twenty times it sends a *completion notification* to server $S_0$.

4. 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 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 is queue. If there is a 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 (queue is empty), the server becomes unlocked.

   (e) $S_0$ brings the entire distributed computation to an end once its has received *completion notification* from all the clients.

5. 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 is through eLearning.