# CS/CE/TE 6378: Project I

Instructor: Ravi Prakash

Assigned on: February 6, 2018
Due date and time: February 20, 2018, 11:59 pm

This is an individual project and you are required to demonstrate its operation to the instructor and/or the TA to get credit for the project.

## 1 Requirements

1. Source code must be in the C /C++ /Java programming language.
2. The program must run on UTD lab machines (`dc01, dc02, ..., dc45`).

## 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. It can be assumed that each process (server/client) is running on a single machine (`dcXY`). Please get familiar with basic UNIX commands to run your program on *dcXY*.

## 3 Description

Design and implement a distributed system which consists of *three* server processes and *five* client processes. Assume that each file is replicated on all the servers, and all replicas of a file are consistent in the beginning. A client can perform a READ or WRITE operation on the files. READ operation involves only one server that has the target file and the server is chosen randomly by a client. WRITE involves all servers that have a copy of the target file and all of them should be updated in order to keep the replicas consistent. READ/WRITE on a file can be performed by only one client at a time. However, different clients are allowed to concurrently perform a READ/WRITE on different files. In order to ensure this, implement Lamport's mutual exclusion algorithm among clients so that no READ/WRITE violation could occur. The operations on files (hosted by servers) can be seen as *critical section* executions.

To host files, create separate directory for each server. The servers must support following operations and reply to the clients with appropriate messages –

1. ENQUIRY : A request from a client for information about the list of hosted files.
2. READ: A request to read last line from a given file.
3. WRITE: A request to append a string to a given file.

The set of files does not change during your program's execution. Also, assume that there is no server failure during your program's execution.

The clients must be able to do the following –

1. Gather information about the hosted files by querying the servers and keep the metadata for future.
2. Append a string $< client\_id,\ Timestamp >$ to a file, $f_i$ during WRITE. Here $client\_id$ is the name of the client, and $Timestamp$ is the value of the client's local clock when the write request is generated. This must be done to all replicas of $f_i$.
3. Read last line of a file, $f_i$, during READ.

Instead of submitting READ/WRITE requests through a command line interface, write an application that periodically generates READ/WRITE requests for a randomly chosen file from the set of files stored by the servers.

Pertaining to socket programming, close all open sockets when your program exits/terminates. This may save you some valuable time during the test runs and more importantly, from a panic situation during the project demonstration.

Displaying appropriate log messages to the console or to a file is very important for testing, debugging and analyzing the correctness of your program.

# 4 Submission Information

The submission should be through eLearning in the form of an archive consisting of:

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

**NOTE: Do not submit unnecessary files.**