# CS/CE/TE 6378: Project I

Instructor: Ravi Prakash

Assigned on: February 14, 2013
Due date and time: March 1, 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 tree-based *Dijkstra-Scholten* [1] termination detection algorithm. The algorithm was discussed in the class. Assume there are $n$ (given) nodes in the system. The nodes implement Lamport's logical clock with increment value 1. One of the nodes initiates the termination detection algorithm. When a node $A$ wants to send a message to another node $B$ and $A$ is not directly connected to $B$ then $A$ first establishes a TCP connection with $B$. The TCP connection is alive until $A$ receives all ACKs (as per the termination detection algorithm) from $B$.

You are given a file that contains a series of events that should be simulated on the nodes. The program is expected to read these events from the file and executes them on corresponding nodes. First line of the file gives the number of nodes in the system. Each of the subsequent lines of the file represents an event that is denoted by a tuple `<nodeid, clockval, type, [param]>`; where: `nodeid` is the node where the event should occur, `clockval` is the logical clock value at `nodeid` when the event occurs, `type` is type of the event and `param` is an optional parameter. The types of the event could be one of the following:

**INIT** Initiate the termination protocol.

**SEND** Send a message to node given in `param`.

**IDLE** The node becomes idle and it may trigger some events according to the algorithm.

**TICK** A "Tick" of the logical clock after delay given in `param` (ms).

In addition to above described events, "RECV" a message by a node is also considered as an "event".

Once the initiator detects the termination of the computation, it displays proper messages and quits. Your program should also display proper log messages either on screen or to a file.

## 4 Submission Information

The submission should be through *elearning* in the form of an archive 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] E.W. Dijkstra and C.S. Scholten. *Termination detection for diffusing computations.* Information Processing Letters, 11(1):1-4, 1980.

# 5 Program Testing

A sample input file, ”sample.txt”, is given below. However, you program will be tested on other files. The input file assumes that contiguous IDs are given to the nodes, starting from 0.

```
5
0 0 INIT
0 1 SEND 1
1 1 SEND 2
1 2 TICK 10
1 3 SEND 3
3 1 TICK 10
3 2 SEND 4
4 1 TICK 10
4 2 SEND 1
4 4 IDLE
3 4 TICK 10
3 5 IDLE
2 1 TICK 100
2 2 SEND 4
2 4 IDLE
4 6 TICK 10
4 7 IDLE
1 7 TICK 20
1 7 IDLE
0 3 TICK 10
0 4 IDLE
```