Distributed Systems Project, Spring 2016, First exercise

Select this assignment if (your student ID modulo 4 + 1) equals 1.

Assignment 1: Lamport clocks
Write a program that implements Lamport clocks.

Specifications
Command line interface:
program configuration_file line
where configuration_file is the name of the configuration file and line is
the line of this client (see below).

The configuration file has an undetermined number of lines, each in the
following format:
id host port
where id is an integer used in the manner described below, host is either the
hostname or the IP address of the client and port is the port on which it is
listening at that address. Your client program takes as argument one integer,
which indicates its own ID, i.e., the line in the configuration that indicates what
port this client should use. The client can ignore the hostname for its own line,
but needs to use the other lines to know who are the other clients in the system.
There is no upper limit to the number of lines in the configuration file. If the
argument given to the program does not exist in the configuration file, your
program is allowed to crash immediately.

NOTE: It's easiest to have all the clients run on the same machine during early
development.
NOTE: You can develop your programs in any environment, but it must also be
runnable on the Ukkko cluster.

Running of the Program
Make sure all programs are running before having them start executing the
algorithm. In the following, we use terms node and program interchangeably,
since individual programs are intended to simulate different nodes.

Each node can freely choose to have either a local event or send a message to one
other node. Choose randomly with equal probabilities between these two
possibilities.
  • For local event, increment local Lamport clock by a random amount
    selected uniformly at random between 1 and 5.
  • For a message, select another node uniformly at random and send a
    message to that node with the correct clock value. The receiving node
    must execute Lamport's algorithm to update its clock correctly.

Every node should run for 100 events after which it is allowed to terminate. If a
node happens to attempt to communicate with a departed node, have it fail in
some graceful manner.
Output Format
Each individual program should produce an output of its run. Use the following syntax for output:

- Local event: `l n`, where $n$ is the amount by which the clock was increased
- Sending a message: `s r l`, where $r$ is the receiving node ID and $l$ is the clock value sent in the message.
- Receiving a message: `r s t n`, where $s$ is the sender of the message, $t$ is the timestamp that was in the message, and $n$ is the clock value after running Lamport’s algorithm.

**NOTE:** Because we use partly automated tools for checking the output, your output must match exactly the format above. Do not add any other text or lines. Failure to comply will lead to a reduced grade.

Guidelines
You are free to choose any programming language, but we recommend using a higher level language, e.g., Ruby or Python, even if you have to learn the language from scratch during the assignment.

The actual contents of the messages are irrelevant, as long as you are able to implement the required functionality. No interoperability between groups is required so you are free to choose the format.

Deliverables
Program source code with documentation. Code must be commented and documentation should describe your solution, the execution process, and instructions on how to run it.

Timeline
The assignment is due on January 28th at 10:00. No extensions will be given.

Return
Store all the files in a directory that has same name as your username. Zip this directory, name the zip-file “username_DSP16_EX1.zip”, and return the zip-file via Moodle. Please indicate clearly your name and student ID in every source code file.