Distributed Systems

Assignment 3 – Clock Synchronization
Assignment given on: 22nd Sep 2020
Due Date: 30th Sep 2020

Assignment 3 – Clock Synchronization using Berkeley Algorithm.

The objective of this assignment is to understand about clock synchronization in a distributed setting. You will implement the Berkeley algorithm.

There are two kinds of nodes in the system: a single master node, and a set of slave nodes. Periodically, the master queries all of the slave nodes for their local time and each node replies with their current time. The master then computes the time difference between its local clock and that of each of the slaves (deriving a set of delta values, one per slave). Next, the master computes average time difference between all the clock times received and the clock time given by th master's system clock itself. Then, for each slave the master computes a time correction value by computing the difference between avg and the delta value for the slave. Finally, the master adds the correction value to its local clock and sends correction value to each slave for the slaves to adjust their clocks.

Implementation requirements:

- The master must also synchronize its time along with the slaves.
- As long as a slave is alive its time should eventually be synchronized (with times of the other alive slaves and the master).
- UDP must be used for messaging.
- Each node in your implementation should maintain and synchronize a process-local clock (not the system clock).
- Each node should periodically log its local time to stdout, particularly before and after synchronization.
- You must use the GoVector library to track the partial ordering between important events (e.g., message sends/receives and synchronization events) in your distributed system and to log these events to a log file. Use the ShiViz tool to visualize the GoVector-generated log.
- You can run multiple clients on local machine. Install Terminator and run multiple clients using terminator terminal.

Assumptions you can make:

- The master node does not fail

Solution spec:

Write a single go program that acts as a time master or a time slave, depending on the command line options passed to the program. Given a -m flag in the first position the program should behave as a time master and expect the following arguments on the rest of the command line:

- ip:port : address that the master should use to communicate with all of the slaves
- time : the process-local time that the master should be initialized with
- slavesfile : a file with as many lines as there are slaves, each line contains an "ip:port" address of a slave
- logfile : filename to which GoVector log messages will be written

Given a -s flag in the first position the program should behave as a time slave and should expect the following three arguments on the rest of the command line

- ip:port : address that the slave should listen on for server's messages
- time : the process-local clock value that the slave should be initialized with
- logfile : filename to which GoVector log messages will be written

What to Submit

A single zip file containing:

- Source files
- Slaves file
- README File
- Go Vector log and ShiViz screen shot.

Grading – 20 Marks

- Master-Clinet.go – 15 Marks
- GoVector log and ShiViz screen shot – 4 Marks
- Readme – 1 Mark

Submission Details:

1. Please read the questions carefully and complete it.

2. Make a directory with name <Your_Roll_Number> and copy your all program (source code) and output file to that folder.

3. Implement the solution using GoLang.

4. Please take screen shot of output, name it with its question number and put it in a same folder.

5. Test well before submission. Follow some coding style uniformly. Provide proper comments in your code.

6. Submit only through moodle and well in advance. Any hiccups in the moodle/Internet at the last minute is never acceptable as an excuse for late submission. Submissions through email will be ignored.

7. Please attach a readme.txt file

8. Please zip your folder and submit to moodle within 30-09-2020 (23:55 PM)