**CSE 5306 Distributed Systems Spring 2022 Project-2**

Name: Adithya Venkatesh

ID: 1002032198

Email: [axv2198@mavs.uta.edu](mailto:axv2198@mavs.uta.edu)

I have neither given nor received unauthorized assistance on this work

Sign: Adithya Venkatesh

Date: 04/10/22

**PART-1 (10 pts)**

In this we randomly initialize the logical clock of individual processes and use Berkeley’s algorithm to synchronize these clocks to the average clock. We can select any process as the time daemon to initiate the clock synchronization. After the synchronization, each process prints out its logical clock to check the result of synchronization.

Diagram

Description automatically generated

**Berkeley’s Algorithm**

* A single node is taken as master node, it periodically sends request for time to all the other client node
* It uses Christian’s algorithm to fetch clock time
* The Master node calculate the average time difference between all the clocks and the clock time given by master node itself.
* The average time is then sent to all the clients after adding with its own time over the network.

**Implementation and Execution Details**

* We have three nodes in which one is the Daemon and two slave nodes.
* All the clock will have a random number assigned to each other and Daemon request for time from slave nodes
* All the three nodes are connected using socket using clientConnect() function in server side
* SynchronousAllClocks() function will calculate the average of each client time and uses Berkley’s algorithm to find the time and sends it to all the child node.
* The child node will correct it’s time according to the Daemon.

**PART-2 (20 pts)**

In this part we have to Implement the vector clock for your distributed system. You can create two threads for each process, one for sending messages to other nodes and one for listening to its communication port. Communication among nodes can be done using RPC or using sockets. Once a process sends a message, it should print its vector clock before and after sending the message. Similarly, once a process receives a message, it should print its vector clock before and after receiving the message.

A picture containing text, clock, watch

Description automatically generated

**Vector Clock Algorithm**

* Initially all clocks are zero.
* The local clock value for a process is incremented at least once before each local atomic event (like a database write, or sending a message).
* Each time a process sends a message, it includes a copy of its own timestamp vector.
* Each time a process receives a message, it increments its own logical clock in the vector by one and updates each element in its vector by taking the maximum of the value in its own vector clock and the value in the vector in the received message (for every element).

**Implementation and Execution Details**

* There are three process p1,p2,p3 which are communicating with each other using sockets
* Two threads are created to communicate in each process to communicate with each other, one is to send and one is to receive
* Thevector clock is printed before and after sending of message.
* The socket will be up and running all the time till there is a manual intervention