**Logo, company name

Description automatically generated**

**2225-CSE-5306-001 - DISTRIBUTED SYSTEMS**

**PROJECT- 2**

I have neither given nor received unauthorized assistance on this work.

**Sign:**  Srinath Kalvagadda,1001967654                             Date: 07/16/2022

Sai Subhash Yalamadala,1002031729 Date: 07/16/2022

**Introduction**

Goal of this programming assignment is to develop an n-node distributed system that implements Vector clock.

A Logical clock is used to monitor the nodes when they send or receive messages. We are using 3 processes A,B,C that belong to the machine to mimic a distributed system with different machines. Since each process has a unique port number they can act as individual machines for testing our distributed system with.

We use Berkeley Algorithm for synchronizing logical clock at all nodes.

**Berkeley Algorithm** assumes that none of the nodes of a network have an accurate time source, so it uses another algorithm called **Cristian’s algorithm** to ping all the nodes from a master node(which will be selected using leader election algorithm) periodically to fetch clock time of each node.

Diagram

Description automatically generated

Source: <https://www.geeksforgeeks.org/berkeleys-algorithm/>

Programming Language used : **JAVA**

**Vector Clocks in Distributed Systems**

Vector Clock is an algorithm that generates partial ordering of events and detects causality violations in a distributed system. These clocks expand on Scalar time to facilitate a causally consistent view of the distributed system, they detect whether a contributed event has caused another event in the distributed system. It essentially captures all the causal relationships. This algorithm helps us label every process with a vector (a list of integers) with an integer for each local clock of every process within the system. So, for N given processes, there will be vector/ array of size N.

**How does the vector clock algorithm work?**

* Initially, all the clocks are set to zero.
* Every time, an Internal event occurs in a process, the value of the processes logical clock in the vector is incremented by 1
* Also, every time a process sends a message, the value of the processes logical clock in the vector is incremented by 1.

Chart

Description automatically generated

**Source**: <https://www.geeksforgeeks.org/vector-clocks-in-distributed-systems/>

**PART 1**

Suppose the logical clock on each machine represents the number of messages has been sent and received by this machine. It is actually a counter used by the process (or the machine emulator) to count events. Randomly initialized the logical clock of individual processes and used Berkeley’s algorithm to synchronize these clocks to the average clock. 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.

To Implement the logical clock of individual processes and use Berkeley’s algorithm to synchronize these clocks to the average clock.

* Each server will be running and will be ready to send and receive messages among servers.
* When the client is executed to send the messages then the timer will be updated to the average clock in all the servers.
* Average clock is calculated in the client, and it will be synchronized.
* The logical clock is printed in all the server to check the result of synchronization.

**PART 2**

Created 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.

To Implement the vector clock that supports communication among nodes.

we used **Java** and **Socket.**

* Each server will be running and will be ready to send and receive messages among servers.
* After selecting the server that you want to send the message the vector clock will be printed before and after sending the messages.
* For example, if you want to send the message from server 1 to server 2 then the vector clock is printed before and after sending the messages in server1. Vector clock is automatically printed in server2 also before and after receiving the messages.

**What you have learned**

* Understanding how Berkeley Algorithm, Cristian’s Algorithm works.
* How Java Sockets are implemented.
* Got an brief idea how techniques like vector clocks can be used in distributed systems.

**What issues you have encountered**

* Having no prior knowledge about Berkeley Algorithm made it cumbersome to implement.
* Using multiple threads, to maintain and monitor different processes was more challenging than expected.

Srinath Kalvagadda has worked on Part 1.

Sai Subhash has worked on Part 2.

.