Experiment Number 06

Aim: Develop a program for Clock Synchronization algorithms

Learning Objective: Student should be able to develop Clock Synchronization algorithms

Tools: Java

Theory:

Clock synchronization in distributed computing involves ensuring that the clocks across multiple machines in a network are aligned to a common time reference. This is crucial for coordinating actions, ordering events, and maintaining consistency in distributed systems. One widely used algorithm for clock synchronization is the Network Time Protocol (NTP). NTP employs a hierarchical system of time servers to disseminate accurate time information across the network. Each server synchronizes its clock with a higher-stratum server, ultimately obtaining time from a primary time source such as an atomic clock. Clients periodically query the time from these servers and adjust their clocks accordingly, minimizing clock skew and maintaining synchronized time across the network.

The Berkeley Algorithm

It is a clock synchronization algorithm used in distributed systems. It works by periodically synchronizing the local clocks of all machines in the network with a reference clock known as the master clock. Here's how it works:

1. **Master Selection:** One machine in the network is designated as the master, typically the one with the most accurate clock or an external time source like a GPS receiver.
2. **Measurement:** Periodically, each machine measures the difference between its local clock and the master clock. This difference is known as the clock offset.
3. **Calculation:** After collecting clock offsets from all machines, the average offset is calculated. This average represents the discrepancy between the average time of the machines and the master clock.
4. **Adjustment:** Each machine adjusts its local clock by applying the calculated average offset, thus synchronizing its time with the master clock.

This process helps minimize clock skew and ensures that all machines in the network maintain relatively synchronized time.

Clock Synchronization algorithm Berkeley's Algorithm

```
import java.util.ArrayList;
import java.util.List;
import java.util.Random;
class Machine {
    private int id;
    private int clock;
    public Machine(int id) {
        this.id = id;
        this.clock = new
Random().nextInt(100); // Initialize
clock with random value}
    public int getId() {
        return id; }
    public int getClock() {
        return clock; }
    public void setClock(int clock) {
        this.clock = clock;  }}
public class BerkeleyAlgorithm {
    private List<Machine> machines;
    public BerkeleyAlgorithm(int
numMachines) {
        machines = new ArrayList<>();
        for (int i = 0; i < numMachines;
i++) {
            machines.add(new
Machine(i));}}
    public void synchronizeClocks() {
        int sum = 0;
        for (Machine machine : machines)
{            sum += machine.getClock();}
```

```
        int average = sum /
machines.size();
        for (Machine machine : machines)
{
            machine.setClock(average);} }
    public void printClocks() {
        for (Machine machine : machines)
{
            System.out.println("Machine "
+ machine.getId() + ": Clock = " +
machine.getClock());}}
    public static void main(String[]
args) {
        BerkeleyAlgorithm algorithm = new
BerkeleyAlgorithm(5); // Number of
machines
        System.out.println("Before
synchronization:");
        algorithm.printClocks();
        algorithm.synchronizeClocks();
        System.out.println("\nAfter
synchronization:");
        algorithm.printClocks(); }}
```

Output:

```
Before synchronization:
Machine 0: Clock = 33
Machine 1: Clock = 33
Machine 2: Clock = 48
Machine 3: Clock = 93
Machine 4: Clock = 64
```

```
After synchronization:
Machine 0: Clock = 54
Machine 1: Clock = 54
Machine 2: Clock = 54
Machine 3: Clock = 54
Machine 4: Clock = 54
```

Result and Discussion:

The Berkeley Algorithm in Java enables clock synchronization among distributed machines by averaging and adjusting clock values, ensuring coordination and consistency crucial for distributed environments.

Learning Outcomes: The student should have the ability to
LO1: Describe the Clock Synchronization algorithms
LO2: Write a Program to the Clock Synchronization algorithms

Course Outcomes: Upon completion of the course students will be able to understand Clock synchronization algorithms.

Conclusion:

To conclude, the Berkeley Algorithm, implemented in Java, effectively synchronizes clocks among distributed machines by averaging and adjusting their values. This ensures coordination and consistency crucial for distributed environments, demonstrating the algorithm's practical utility in real-world applications.

For Faculty Use

Correction Parameters	Formative Assessment [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	
Marks Obtained				