// Assgt 4 - Berkeley Clock Synchronisation

import java.util.ArrayList;

import java.util.List;

import java.util.Random;

import java.util.concurrent.\*;

class Node implements Callable<Integer> {

private int clockTime;

private final int nodeId;

public Node(int nodeId, int clockTime) {

this.nodeId = nodeId;

this.clockTime = clockTime;

}

@Override

public Integer call() {

System.out.println("Node " + nodeId + " has time: " + clockTime);

return clockTime;

}

public void adjustTime(int offset) {

clockTime += offset;

System.out.println("Node " + nodeId + " adjusted time to: " + clockTime);

}

}

class BerkeleyMaster {

private final List<Node> nodes;

public BerkeleyMaster(List<Node> nodes) {

this.nodes = nodes;

}

public void synchronizeClocks() throws InterruptedException, ExecutionException {

ExecutorService executor = Executors.newFixedThreadPool(nodes.size());

List<Future<Integer>> futures = executor.invokeAll(nodes);

int masterTime = 100; // Master clock time (reference)

int totalTime = masterTime;

List<Integer> times = new ArrayList<>();

// Collect times from nodes

for (Future<Integer> future : futures) {

int time = future.get();

times.add(time);

totalTime += time;

}

int averageTime = totalTime / (nodes.size() + 1);

System.out.println("Master calculated average time: " + averageTime);

// Adjust each node based on the offset

for (Node node : nodes) {

int offset = averageTime - node.call();

node.adjustTime(offset);

}

executor.shutdown();

}

}

public class BerkeleyClockSynchronisation {

public static void main(String[] args) throws InterruptedException, ExecutionException {

Random random = new Random();

List<Node> nodes = new ArrayList<>();

for (int i = 0; i < 5; i++) { // 5 nodes

int randomTime = 95 + random.nextInt(11); // Random clock time between 95 and 105

nodes.add(new Node(i + 1, randomTime));

}

BerkeleyMaster master = new BerkeleyMaster(nodes);

master.synchronizeClocks();

}

}

Output:

javac BerkeleyClockSynchronisation.java

java BerkeleyClockSynchronisation

Node 1 has time: 98

Node 3 has time: 105

Node 2 has time: 104

Node 4 has time: 99

Node 5 has time: 97

Master calculated average time: 100

Node 1 has time: 98

Node 1 adjusted time to: 100

Node 2 has time: 104

Node 2 adjusted time to: 100

Node 3 has time: 105

Node 3 adjusted time to: 100

Node 4 has time: 99

Node 4 adjusted time to: 100

Node 5 has time: 97

Node 5 adjusted time to: 100