**AIM -** To Implement Berkeley algorithm for clock synchronization.

**Name –** Yogita Sunil Girigosavi

**Server.java**

import java.io.\*;

import java.net.\*;

import java.util.\*;

public class Server {

public static void main(String[] args) throws IOException {

ServerSocket serverSocket = null;

Socket clientSocket = null;

PrintWriter out = null;

BufferedReader in = null;

try {

// Create a server socket on port 4444

serverSocket = new ServerSocket(4444);

} catch (IOException e) {

System.err.println("Could not listen on port: 4444.");

System.exit(1);

}

// Accept client connections and start a new thread for each one

while (true) {

try {

clientSocket = serverSocket.accept();

System.out.println("Accepted connection from " + clientSocket.getInetAddress().getHostName());

// Create input and output streams for the client socket

out = new PrintWriter(clientSocket.getOutputStream(), true);

in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

// Create a new thread to handle the client connection

Thread t = new ServerThread(clientSocket, out, in);

t.start();

} catch (IOException e) {

System.err.println("Error handling client connection.");

}

}

}

}

class ServerThread extends Thread {

private Socket clientSocket;

private PrintWriter out;

private BufferedReader in;

public ServerThread(Socket socket, PrintWriter out, BufferedReader in) {

this.clientSocket = socket;

this.out = out;

this.in = in;

}

public void run() {

try {

// Read the client's clock time

long clientTime = Long.parseLong(in.readLine());

System.out.println("Received time from " + clientSocket.getInetAddress().getHostName() + ": " + clientTime);

// Calculate the server's clock time

long serverTime = System.currentTimeMillis();

System.out.println("Server time: " + serverTime);

// Calculate the time offset

long offset = serverTime - clientTime;

System.out.println("Offset: " + offset);

// Send the time offset to the client

out.println(offset);

// Close the socket and streams

in.close();

out.close();

clientSocket.close();

} catch (IOException e) {

System.err.println("Error handling client connection.");

}

}

}

**Client.java**

import java.io.\*;

import java.net.\*;

import java.util.\*;

public class Client {

public static void main(String[] args) throws IOException {

if (args.length != 3) {

System.err.println("Usage: java Client <hostname1> <hostname2> <hostname3>");

System.exit(1);

}

// Create a list of hostnames

List<String> hostnames = Arrays.asList(args);

// Connect to each server and get the clock time

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

for (String hostname : hostnames) {

try {

// Connect to the server

Socket socket = new Socket(hostname, 4444);

System.out.println("Connected to " + hostname);

// Create input and output streams for the server socket

PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

// Send the client's clock time to the server

long clientTime = System.currentTimeMillis();

out.println(clientTime);

// Read the time offset from the server

long offset = Long.parseLong(in.readLine());

System.out.println("Offset from " + hostname + ": " + offset);

// Calculate the server's clock time

long serverTime = clientTime + offset;

Date date = new Date(serverTime);

String formattedDate = String.format("%tF %tT", date, date);

System.out.println("Server time at " + hostname + ": " + formattedDate);

times.add(date);

// Close the socket and streams

in.close();

out.close();

socket.close();

} catch (UnknownHostException e) {

System.err.println("Unknown host: " + hostname);

System.exit(1);

} catch (IOException e) {

System.err.println("Error connecting to " + hostname);

System.exit(1);

}

}

// Calculate the average clock time

long averageTime = (times.get(0).getTime() + times.get(1).getTime() + times.get(2).getTime()) / 3;

Date averageDate = new Date(averageTime);

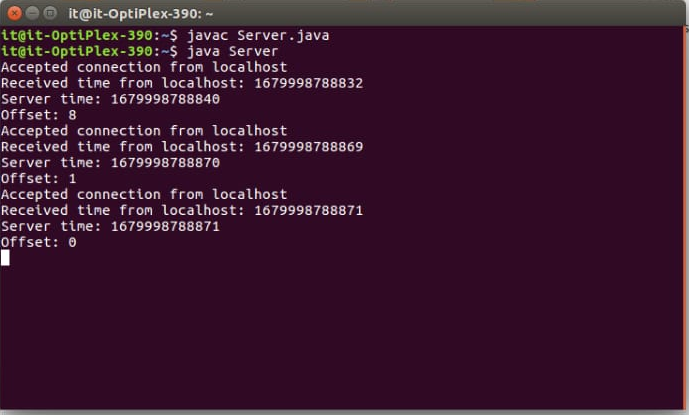
String formattedDate = String.format("%tF %tT", averageDate, averageDate);

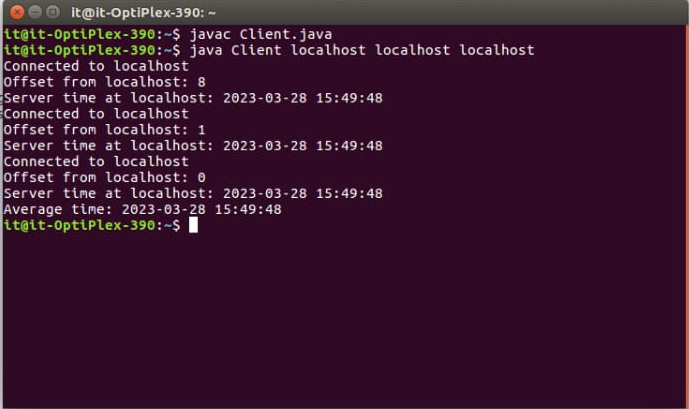
System.out.println("Average time: " + formattedDate);

}

}

**Output –**

****

****