**Code**

Client.java

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.PrintWriter;

import java.net.Socket;

import java.text.DateFormat;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.Timer;

import java.util.TimerTask;

public class Client {

static long givenTime;

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

DateFormat sdf = new SimpleDateFormat("hh:mm:ss");

givenTime = sdf.parse(args[0]).getTime() + (new Date()).getTime();

System.out.println("Current Time: "+new Date(givenTime));

Timer timer = new Timer();

timer.scheduleAtFixedRate(new TimerTask() {

@Override

public void run(){

givenTime += 100;

}

}, 0, 100);

try(

Socket sock = new Socket("localhost",6969);

BufferedReader in = new BufferedReader(

new InputStreamReader(

sock.getInputStream()

)

);

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

){

String fromServer;

while((fromServer = in.readLine()) != null){

if (fromServer.equals("send time")){

out.println(givenTime);

System.out.println("Send Time:"+new Date(givenTime));

}else if(fromServer.chars().allMatch( Character::isDigit)|| (fromServer.startsWith("-") && fromServer.substring(1).chars().allMatch( Character::isDigit))){

givenTime = Long.parseLong(fromServer);

System.out.println("New Date From Server: "+new Date(givenTime));

}else if(fromServer.equals("exit")){

System.out.println("Server shutdown exiting: ");

break;

}else{

System.out.println("From Server: "+fromServer);

}

}

}

}

}

Server.java

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.PrintWriter;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.Date;

import java.util.Timer;

import java.util.TimerTask;

import java.text.SimpleDateFormat;

import java.text.DateFormat;

import java.text.ParseException;

import java.util.ArrayList;

class Server{

static DateFormat sdf = new SimpleDateFormat("hh:mm:ss");

static long givenTime;

static Date startTime = null;

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

givenTime = sdf.parse(args[0]).getTime() + (new Date()).getTime();

Timer timer = new Timer();

timer.scheduleAtFixedRate(new TimerTask() {

@Override

public void run(){

ArrayList<Long> timeList = ClientSockets.GetTime();

long res = 0;

for(Long clientTime:timeList){

res += givenTime -clientTime;

}

res = res/Math.max(1,timeList.size());

givenTime= givenTime+res;

System.out.println("New clock time: "+new Date(givenTime));

ClientSockets.sendTime(givenTime);

}

}, 0,5000);

timer.scheduleAtFixedRate(new TimerTask() {

@Override

public void run(){

givenTime += 1;

}

}, 0, 1);

final ServerSocket serverSock = new ServerSocket(6969);

try{

while(true) new ClientSockets(serverSock.accept());

}catch(Exception e){

e.printStackTrace();

}

finally{

for(ClientSockets cs: ClientSockets.clientList){

cs.sockDos.println("exit");

}

serverSock.close();

}

}

}

class ClientSockets{

public static ArrayList<ClientSockets> clientList = new ArrayList<>();

private static int count = 0;

private static final String CLIENT="CLIENT\_";

Socket socket;

String nameString;

PrintWriter sockDos;

BufferedReader sockDis;

ClientSockets(Socket socket) throws IOException{

count++;

this.socket = socket;

this.nameString = CLIENT+count;

this.sockDos = new PrintWriter(this.socket.getOutputStream(),true);

this.sockDis = new BufferedReader(

new InputStreamReader(this.socket.getInputStream())

);

clientList.add(this);

}

public static ArrayList<Long> GetTime(){

ArrayList<Long> times = new ArrayList<>();

clientList.parallelStream().forEach((client)->{

try{

client.sockDos.println("send time");

String temp = null;

while((temp=client.sockDis.readLine()) ==null);

long time=Long.parseLong(temp);

System.out.println(client.nameString+" time: "+new Date(time));

times.add(time);

}catch(Exception e){

e.printStackTrace();

}

});

return times;

}

public static void sendTime(long refTime){

clientList.parallelStream().forEach((client)->{

try{

client.sockDos.println(refTime);

}catch(Exception e){

e.printStackTrace();

}

});

}

}

**Output**







