

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

**🙚🞻🙘**



**BÁO CÁO CUỐI KỲ**

**MÔN: CÁC HỆ THỐNG PHÂN TÁN**

**GIẢNG VIÊN HƯỚNG DẪN : Thạc sĩ Lê Ngọc Bảo**

**THỰC HIỆN ĐỀ TÀI :**

* **DƯƠNG CHẤN NINH – N19DCCN117 – D19CQCNPM01-N**

**TPHCM - Tháng 12 - 2022**

1. **Tên đề tài:**

* Mạng nội bộ của nhóm, 1 máy tính được xem là Time server. Xây dựng ứng dụng mô hình client/server đồng bộ đồng hồ vật lý các máy trạm theo thời gian đồng hồ time server.

1. **Mô tả cách thực hiện:**

* Sử dụng ngôn ngữ Java trên môi trường Eclipse để thực hiện đề tài. Máy chủ server là 1 đối tượng được tạo từ lớp **ServerSocket** trong Java, các máy client là các đối tượng được tạo từ lớp **Socket**.
* Các máy client sẽ yêu cầu server gửi thông tin về thời gian của server về client để client tiến hành đồng bộ thời gian.
* Thời gian của máy client sẽ được tạo ngẫu nhiên, thời gian của máy server sẽ là thời gian của nước Việt Nam.
* Chương trình sử dụng thư viện JFrame để mô tả quá trình đồng bộ một cách tường minh hơn.

1. **Chương trình thực hiện:**

* File Server.java:

**package** server\_client;

**import** java.io.DataInputStream;

**import** java.io.DataOutputStream;

**import** java.net.ServerSocket;

**import** java.net.Socket;

**import** java.text.SimpleDateFormat;

**import** java.util.Calendar;

**public** **class** Server{

**private** ServerSocket serverSocket;

**private** SimpleDateFormat timeFormat;

**private** DataInputStream din;

**private** DataOutputStream dout;

**public** Server(ServerSocket serverSocket) {

**this**.serverSocket = serverSocket;

}

**public** **void** startServer() {

**try** {

setTimeFormat(**new** SimpleDateFormat("HH:mm:ss"));

**while**(!serverSocket.isClosed()) {

Socket socket = serverSocket.accept();

System.***out***.println("A new client has connected!");

din = **new** DataInputStream(socket.getInputStream());

dout = **new** DataOutputStream(socket.getOutputStream());

String str="",str2="";

str=din.readUTF();

**if**(str.equals("Request Time")) {

str2 = timeFormat.format(Calendar.*getInstance*().getTime());

System.***out***.println(str2);

dout.writeUTF(str2);

dout.flush();

}

}

} **catch** (Exception e) {

e.printStackTrace();

}

}

**public** **void** closeServerSocket() {

**try** {

**if**(serverSocket != **null**) {

serverSocket.close();

}

} **catch** (Exception e) {

e.printStackTrace();

}

}

**public** ServerSocket getServerSocket() {

**return** serverSocket;

}

**public** **void** setServerSocket(ServerSocket serverSocket) {

**this**.serverSocket = serverSocket;

}

**public** SimpleDateFormat getTimeFormat() {

**return** timeFormat;

}

**public** **void** setTimeFormat(SimpleDateFormat timeFormat) {

**this**.timeFormat = timeFormat;

}

**public** DataInputStream getDin() {

**return** din;

}

**public** **void** setDin(DataInputStream din) {

**this**.din = din;

}

**public** DataOutputStream getDout() {

**return** dout;

}

**public** **void** setDout(DataOutputStream dout) {

**this**.dout = dout;

}

}

* File MyFrame.java

**package** server\_client;

**import** java.awt.Color;

**import** java.awt.FlowLayout;

**import** java.awt.Font;

**import** java.awt.Toolkit;

**import** java.io.DataInputStream;

**import** java.io.DataOutputStream;

**import** java.io.IOException;

**import** java.net.Socket;

**import** java.net.UnknownHostException;

**import** javax.swing.JButton;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JOptionPane;

**import** java.util.Random;

**public** **class** MyFrame **extends** JFrame{

/\*\*

\*

\*/

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**private** JLabel timeLabel;

**private** JButton button;

**private** String time;

Random rd = **new** Random();

**private** **int** hours = rd.nextInt(24), minute = rd.nextInt(59), second = rd.nextInt(59);

MyFrame(String tittle){

**this**.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

**this**.setTitle(tittle);

**this**.setLayout(**new** FlowLayout());

**this**.setSize(350, 200);

**this**.setResizable(**false**);

**this**.setLocation((Toolkit.*getDefaultToolkit*().getScreenSize().width - getSize().width) / 2,

(Toolkit.*getDefaultToolkit*().getScreenSize().height - getSize().height) / 2);

timeLabel = **new** JLabel();

timeLabel.setFont(**new** Font("Verdana", Font.***PLAIN***, 50));

timeLabel.setForeground(**new** Color(0x00FF00));

timeLabel.setBackground(Color.***black***);

timeLabel.setOpaque(**true**);

button = **new** JButton();

button.setBounds(200, 100, 100, 50);

button.setText("Request Time");

button.addActionListener(e -> updateTime());

**this**.add(timeLabel);

**this**.setVisible(**true**);

**if** (tittle.equals("Client"))

**this**.add(button);

setTime();

}

**public** **void** setTime() {

String hours = **new** String();

String minute = **new** String();

String second = **new** String();

**while**(**true**) {

**if**(**this**.hours < 10)

hours = "0" + Integer.*toString*(**this**.hours);

**else** hours = Integer.*toString*(**this**.hours);

**if**(**this**.minute < 10)

minute = "0" + Integer.*toString*(**this**.minute);

**else** minute = Integer.*toString*(**this**.minute);

**if**(**this**.second < 10)

second = "0" + Integer.*toString*(**this**.second);

**else** second = Integer.*toString*(**this**.second);

time = hours + ": " + minute + ": " + second;

timeLabel.setText(time);

**try** {

**if**(**this**.second < 59) **this**.second++;

**else** {

**this**.second = 0;

**if**(**this**.minute < 59) **this**.minute ++;

**else** {

**this**.minute = 0;

**if** (**this**.hours < 23) **this**.hours++;

**else** **this**.hours = 0;

}

}

Thread.*sleep*(1000);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

}

**public** **void** updateTime() {

Socket s;

**try** {

s = **new** Socket("192.168.1.22",9999);

DataInputStream din=**new** DataInputStream(s.getInputStream());

DataOutputStream dout=**new** DataOutputStream(s.getOutputStream());

String str="",str2="";

str = "Request Time";

dout.writeUTF(str);

dout.flush();

**while**(**true**){

str2=din.readUTF();

**if**(!str2.isEmpty()) {

System.***out***.println(str2);

str2.replace(" ", "");

String [] unitsOfClock = str2.split(":");

**this**.hours = Integer.*parseInt*(unitsOfClock[0]);

**this**.minute = Integer.*parseInt*(unitsOfClock[1]);

**this**.second = Integer.*parseInt*(unitsOfClock[2]);

**this**.time = str2;

**break**;

}

}

dout.close();

s.close();

} **catch** (UnknownHostException e) {

e.printStackTrace();

} **catch** (IOException e) {

JFrame frame = **new** JFrame("JOptionPane showMessageDialog example");

JOptionPane.*showMessageDialog*(frame, "Can't connect to server! Server is not operating now.", "Connect error", JOptionPane.***ERROR\_MESSAGE***);

}

}

}

* File Client.java

**package** server\_client;

**public** **class** Client{

**public** **static** **void** main(String[] args) {

**new** MyFrame("Client");

}

}

* File ServerRun.java

**package** server\_client;

**import** java.net.ServerSocket;

**public** **class** ServerRun **implements** Runnable{

**private** ServerSocket serverSocket;

**private** Server server;

**public** ServerRun(ServerSocket serverSocket) {

**this**.serverSocket = serverSocket;

**this**.server = **new** Server(**this**.serverSocket);

}

@Override

**public** **void** run() {

**this**.server.startServer();

}

}

* File ProccessingRequestsOfClients.java

**package** server\_client;

**import** java.io.IOException;

**import** java.net.ServerSocket;

**public** **class** ProccessingRequestsOfClients {

**public** **static** **void** main(String[] args) {

ServerSocket serverSocket;

**try** {

serverSocket = **new** ServerSocket(9999);

ServerRun serverRun = **new** ServerRun(serverSocket);

ServerClockRun serverClockRun = **new** ServerClockRun();

Thread thread = **new** Thread(serverClockRun);

thread.start();

**for** (**int** i = 0; i <= 3; i++) {

Thread myThreadForServerRun = **new** Thread(serverRun);

myThreadForServerRun.start();

}

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

}

* File ServerClockRun.java

**package** server\_client;

**public** **class** ServerClockRun **implements** Runnable{

@Override

**public** **void** run() {

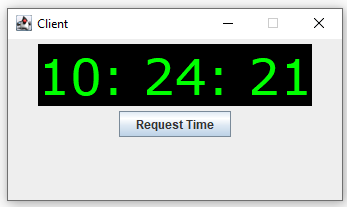
**new** MyFrame("SERVER");

}

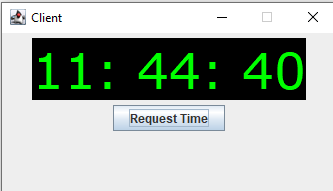
}

1. **Kết quả:**

* Thời gian máy client sau khi chạy:

****

* Thời gian máy client sau khi yêu cầu máy chủ gửi thời gian về để đồng bộ:

****

1. **Link github của source code của đề tài:** https://github.com/ninhdng37/CacHeThongPhanTanJ07D19