**KHOA CÔNG NGHỆ THÔNG TIN**

**BÁO CÁO BÀI TẬP**

**HỌC PHẦN: NGUYÊN LÝ CÁC NGÔN NGỮ LẬP TRÌNH**

**ĐỀ TÀI: XÂY DỰNG PHẦN MỀM MÔ PHỎNG MULTI-CHAT TRÊN NỀN TẢNG C# WINFORM**

Hà Nội, tháng 5 năm 2018

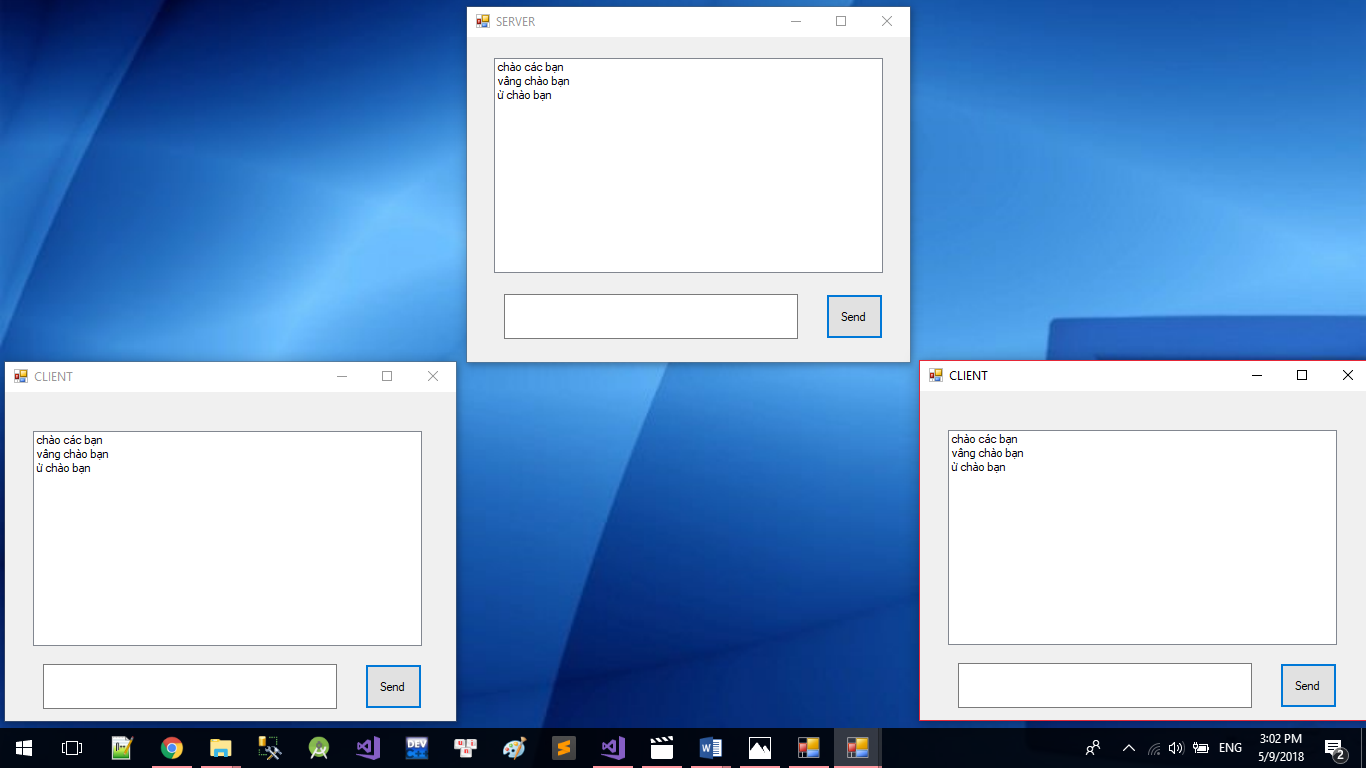
**Đánh giá của giảng viên**

………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

**Mô tả chung**

Xây dựng một ứng dụng mô phỏng việc trao đổi tin nhắn giữa 2 hoặc nhiều máy tính. Ứng dụng được xây dựng trên ngôn ngữ lập trình C# Winform với phương thức lập trình Socket, các máy tính sẽ kết nối với nhau thông qua hệ thống mạng và với mô hình server - client. Khi một máy gửi dòng tin nhắn thì những máy còn lại (đã được kết nối qua socket) sẽ nhận được dòng tin nhắn đó và chúng có thể trao đổi qua lại với nhau.

**Giao diện của chương trình**



**Phần code phía Server**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Net.Sockets;

using System.Net;

using System.Threading;

using System.IO;

using System.Runtime.Serialization.Formatters.Binary;

namespace CHAT

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

CheckForIllegalCrossThreadCalls = false;//tránh việc đụng độ khi sử dụng tài nguyên giữa các thread

Connect();

}

private void Form1\_FormClosed(object sender, FormClosedEventArgs e)

{

Close();

}

IPEndPoint IP;

Socket server;

//khai báo 1 list các client

List<Socket> clientList;

//kết nối đến server

void Connect()

{

clientList = new List<Socket>();//khởi tạo 1 list nhiều client

//khởi tạo địa chỉ IP và socket để kết nối

IP = new IPEndPoint(IPAddress.Any, 1997);

server = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.IP);

//đợi kết nối từ client

server.Bind(IP);

//tạo 1 luồng lăng nghe từ client

Thread Listen = new Thread( () => {

try

{

while (true)

{

server.Listen(100);

Socket client = server.Accept();//nếu lăng nghe thành công thì server chấp nhận kết nối

clientList.Add(client);//thêm các client được server accept vào list

//tạo luồng nhận thông tin từ client

Thread receive = new Thread(Receive);

receive.IsBackground = true;

receive.Start(client);

}

}

/\*khi kết nối đến n client mà có 1 client disconnect thì server sẽ chạy vòng lặp while liên tục để

chương trình ko bị crash\*/

catch

{

IP = new IPEndPoint(IPAddress.Any, 1997);

server = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.IP);

}

} );

Listen.IsBackground = true;

Listen.Start();

}

//đóng kết nối đến server

void Close()

{

server.Close();

}

//gửi dữ liệu

void Send(Socket client)

{

//nếu textboc khác rỗng thì mới gửi tin

if ( (client != null) && (txbMessage.Text != string.Empty) )

{

client.Send(Serialize(txbMessage.Text));

}

}

//nhận dữ liệu

void Receive(object obj)

{

Socket client = obj as Socket;

try

{

while (true)

{

//khởi tạo mảng byte để nhận dữ liệu

byte[] data = new byte[1024 \* 5000];

client.Receive(data);

//chuyển data từ dạng byte sang dạng string

string message = (string)Deseriliaze(data);

//khi 1 client gửi thì cả server và các client (ngoại trừ thằng client vừa gửi) cùng nhận đc

foreach(Socket item in clientList)

{

if(item != null && item != client )

{

item.Send(Serialize(message));

}

}

AddMessage(message);

}

}

catch

{

clientList.Remove(client);

client.Close();

}

}

//add mesage vào khung chat

void AddMessage(string s)

{

lsvMessage.Items.Add(new ListViewItem() { Text = s });

}

//Hàm phân mảnh dữ liệu cần gửi từ dạng string sang dạng byte để gửi đi

byte[] Serialize(object obj)

{

//khởi tạo stream để lưu các byte phân mảnh

MemoryStream stream = new MemoryStream();

//khởi tạo đối tượng BinaryFormatter để phân mảnh dữ liệu sang kiểu byte

BinaryFormatter formatter = new BinaryFormatter();

//phân mảnh rồi ghi vào stream

formatter.Serialize(stream, obj);

//từ stream chuyển các các byte thành dãy rồi cbi gửi đi

return stream.ToArray();

}

//Hàm gom mảnh các byte nhận được rồi chuyển sang kiểu string để hiện thị lên màn hình

object Deseriliaze(byte[] data)

{

//khởi tạo stream đọc kết quả của quá trình phân mảnh

MemoryStream stream = new MemoryStream(data);

//khởi tạo đối tượng chuyển đổi

BinaryFormatter formatter = new BinaryFormatter();

//chuyển đổi dữ liệu và lưu lại kết quả

return formatter.Deserialize(stream);

}

//gửi tin cho nhiều client

private void btnSend\_Click(object sender, EventArgs e)

{

foreach(Socket item in clientList)

{

Send(item);

}

AddMessage(txbMessage.Text);

txbMessage.Clear();

}

}

}

**Phần code phía Client**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Net;

using System.Net.Sockets;

using System.IO;

using System.Runtime.Serialization.Formatters.Binary;

using System.Threading;

namespace CLIENT

{

public partial class frmCLIENT : Form

{

public frmCLIENT()

{

InitializeComponent();

CheckForIllegalCrossThreadCalls = false;//tránh việc đụng độ khi sử dụng tài nguyên giữa các thread

Connect();

}

private void frmCLIENT\_FormClosed(object sender, FormClosedEventArgs e)

{

Close();

}

IPEndPoint IP;

Socket client;

//kết nối đến server

void Connect()

{

//IP là địa chỉ của server.Khởi tạo địa chỉ IP và socket để kết nối

IP = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 1997);

client = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.IP);

//bắt đầu kết nôi. Nếu ko kết nối được thì hiện thông báo

try

{

client.Connect(IP);

}

catch

{

MessageBox.Show("Lỗi kết nối", "Lỗi", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

//tạo luồng lắng nghe server khi vừa kết nối tới

Thread listen = new Thread(Receive);

listen.IsBackground = true;

listen.Start();

}

//đóng kết nối đến server

void Close()

{

client.Close();

}

//gửi dữ liệu

void Send()

{

//nếu textboc khác rỗng thì mới gửi tin

if(txbMessage.Text != string.Empty)

{

client.Send(Serialize(txbMessage.Text));

}

}

//nhận dữ liệu

void Receive()

{

try

{

while (true)

{

//khai báo mảng byte để nhận dữ liệu dưới mảng byte

byte[] data = new byte[1024 \* 5000];

client.Receive(data);

//chuyển data từ dạng byte sang dạng string

string message = (string)Deseriliaze(data);

AddMessage(message);

}

}

catch

{

Close();

}

}

//add mesage vào khung chat

void AddMessage(string s)

{

lsvMessage.Items.Add(new ListViewItem() { Text = s });

txbMessage.Clear();

}

//Hàm phân mảnh dữ liệu cần gửi từ dạng string sang dạng byte để gửi đi

byte[] Serialize (object obj)

{

//khởi tạo stream để lưu các byte phân mảnh

MemoryStream stream = new MemoryStream();

//khởi tạo đối tượng BinaryFormatter để phân mảnh dữ liệu sang kiểu byte

BinaryFormatter formatter = new BinaryFormatter();

//phân mảnh rồi ghi vào stream

formatter.Serialize(stream, obj);

//từ stream chuyển các các byte thành dãy rồi cbi gửi đi

return stream.ToArray();

}

//Hàm gom mảnh các byte nhận được rồi chuyển sang kiểu string để hiện thị lên màn hình

object Deseriliaze (byte[] data)

{

//khởi tạo stream đọc kết quả của quá trình phân mảnh

MemoryStream stream = new MemoryStream(data);

//khởi tạo đối tượng chuyển đổi

BinaryFormatter formatter = new BinaryFormatter();

//chuyển đổi dữ liệu và lưu lại kết quả

return formatter.Deserialize(stream);

}

private void btnSend\_Click(object sender, EventArgs e)

{

Send();

AddMessage(txbMessage.Text);

}

}

}

----------------------------------------HẾT--------------------------------------------