3

Lab

Lập trình Sockets trong C#

Socket Programming in C#

Thực hành Lập trình mạng căn bản - NT106

Giảng viên biên soạn: ThS. Đỗ Thị Hương Lan

Tháng 03/2023

Lưu hành nội bộ

A. TỔNG QUAN

1. Mục tiêu

- Làm quen với Sockets và lập trình Socket trong C#
- Có khả năng viết các ứng dụng sử dụng Sockets với giao thức UDP, TCP bằng C#
- Làm quen và sử dụng Đa luồng (Multithreading) nhằm tối ưu hóa hoạt động của các ứng dụng

2. Môi trường

- Môi trường: IDE Microsoft Visual Studio, Telnet.

3. Liên quan

- Các kiến thức nền tảng về lập trình, ngôn ngữ lập trình C#, Windows Forms Application.
- Các kiến thức về Socket, TCP, UDP
- Tham khảo tài liệu (Mục E) để có kiến thức cơ bản về C#, Winforms

B. KIẾN THỨC NỀN TẢNG

1. Cơ bản về Socket và lập trình Socket

Socket là một **API (Application Programming Interface)** cung cấp các phương thức để giao tiếp thông qua mạng. Socket có thể được hiểu là điểm cuối (end-point) trong liên kết truyền thông hai chiều giữa hai chương trình.

Để lập trình mạng, sử dụng hai namespace System.Net và System.Net.Sockets.

Lớp	Namespace	Mô tả
IPAddress	System.Net	Đại diện cho Địa chỉ IP
IPEndPoint	System.Net	Đại diện cho một điểm giao tiếp
Dns	System.Net	
Socket	System.Net.Sockets	Hỗ trợ lập trình Socket
UdpClient	System.Net.Sockets	Có thể triển khai cho Client và Server của UDP
TcpClient	System.Net.Sockets	Kết nối với Server
TcpListener	System.Net.Sockets	Lắng nghe kết nối từ phía Client (TCP)

Bảng 1: Một số lớp hỗ trợ lập trình Socket trong C#

2. UDP Socket

Giao thức **UDP** là giao thức phi kết nối, nói cách khác không cần thiết lập kết nối giữa hai bên khi tiến hành trao đổi thông tin. Mặc dù UDP không tin cậy bằng giao thức TCP, nhưng tốc độ lại nhanh và dễ cài đặt. Ngoài ra có thể gửi các gói tin quảng bá (broadcast) đồng thời cho nhiều máy.

Với **UDP socket**, không cần phải thiết lập kết nối giữa hai bên trước khi truyền thông điệp. Bên gửi sẽ chỉ rõ **địa chỉ IP và số hiệu cổng** của bên nhận khi gửi thông điệp.

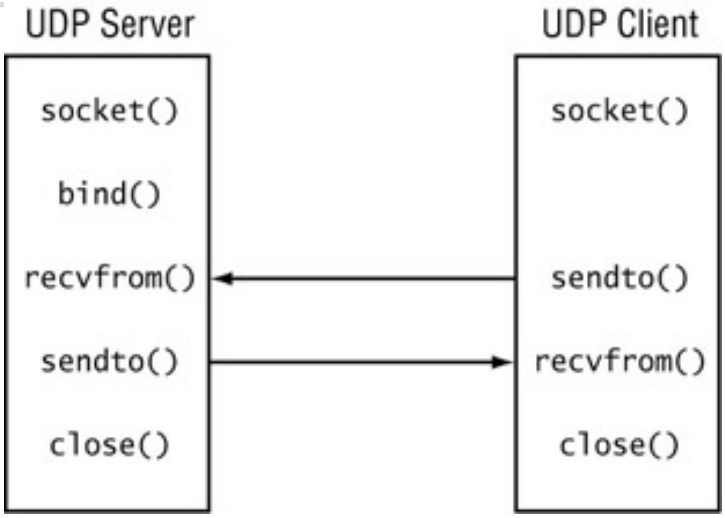
Với hai thực thể mạng đóng vai trò là client và server, quy trình xử lý được mô tả như sau:

Phía tiến trình khách (client):

- Tạo socket
- Gửi thông điệp đến server, chỉ rõ địa chỉ IP và số hiệu cổng của Server
- Nhận thông điệp của server
- Đóng socket

Phía tiến trình chủ (server):

- Tạo socket
- Ràng buộc socket với một số hiệu cổng cần lắng nghe
- Nhận thông điệp của client
- Gửi thông điệp cho client
- Đóng socket



Hình 1: Mô hình hoạt động Client – Server theo UDP

Để lập trình Socket theo UDP với C#, có thể sử dụng lớp **Socket** hoặc lớp **UdpClient**

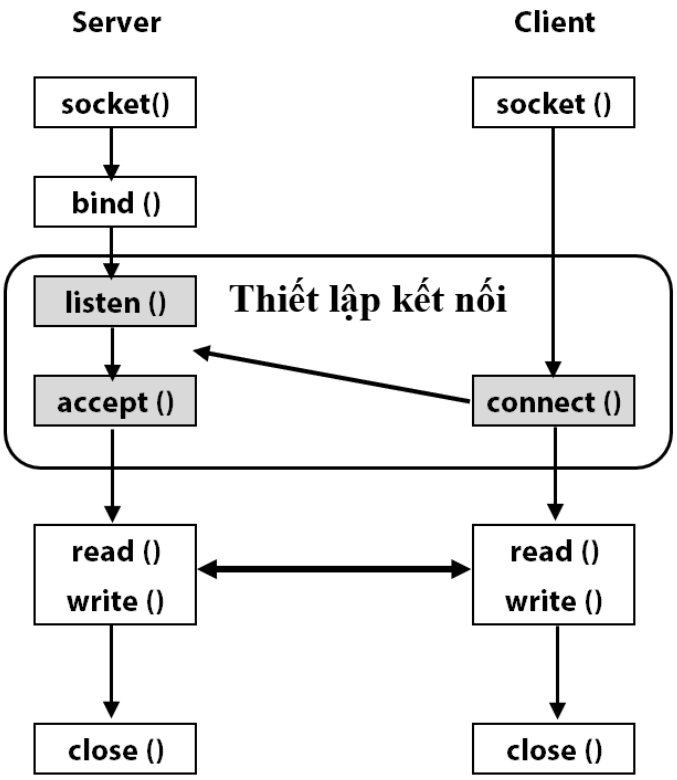
3. TCP Socket

TCP là giao thức hướng kết nối, đảm bảo dữ liệu được truyền một cách tin cậy. Trước khi trao đổi dữ liệu với nhau, trước tiên cần phải thiết lập kết nối giữa hai tiến trình mạng. Trong đó, Server phải chạy trước, lắng nghe các yêu cầu kết nối còn client sẽ kết nối tới server.

UDP	**TCP**
Hoạt động không tin cậy	Hoạt động tin cậy
Không cần thiết lập kết nối	Phải thiết lập kết nối giữa Client & Server
Cần xác định địa chỉ IP và số cổng hiệu của bên nhận trong phương thức gửi dữ liệu	Phương thức gửi dữ liệu không chứa địa chỉ và port của máy nhận
Độ trễ thấp hơn	Độ trễ cao hơn
Thường dùng cho stream, voice, game online,…	Dùng cho các ứng dụng yêu cầu tính toàn vẹn dữ liệu

Bảng 2: So sánh UDP và TCP

Các bước tiến hành trên server và client cần thực hiện khi lập trình socket theo TCP:



Hình 2: Mô hình hoạt động Client – Server theo TCP

Lập trình Socket TCP

Với C#, để lập trình Socket TCP có thể sử dụng các lớp sau:

- Phía client: Socket, TcpClient
- Phía server: Socket, TcpListener
- Để trao đổi dữ liệu: sử dụng các lớp hỗ trợ cho luồng dữ liệu. Ví dụ: NetworkStream

Thiết lập kết nối

Khi sử dụng lớp Socket (C#) để lập trình Socket theo TCP, cần cài đặt SocketType = Stream và ProtocolType = Tcp. Phía client và server cần sử dụng các phương thức để thiết lập kết nối như Listen(), Connect(), Accept().

Có thể sử dụng các lớp đặc trưng như TcpClient (dành cho client) và TcpListener (dành cho server) để lập trình xử lý dễ dàng và đơn giản hơn.

Gửi nhận dữ liệu

Tạo luồng để gửi nhận dữ liệu: sử dụng phương thức GetStream() để dựng NetworkStream

Khi thực hiện gửi và nhận dữ liệu:

- Gửi: dùng phương thức Write() để ghi vào NetworkStream
- Nhận: dung phương thức Read() để đọc từ NetworkStream

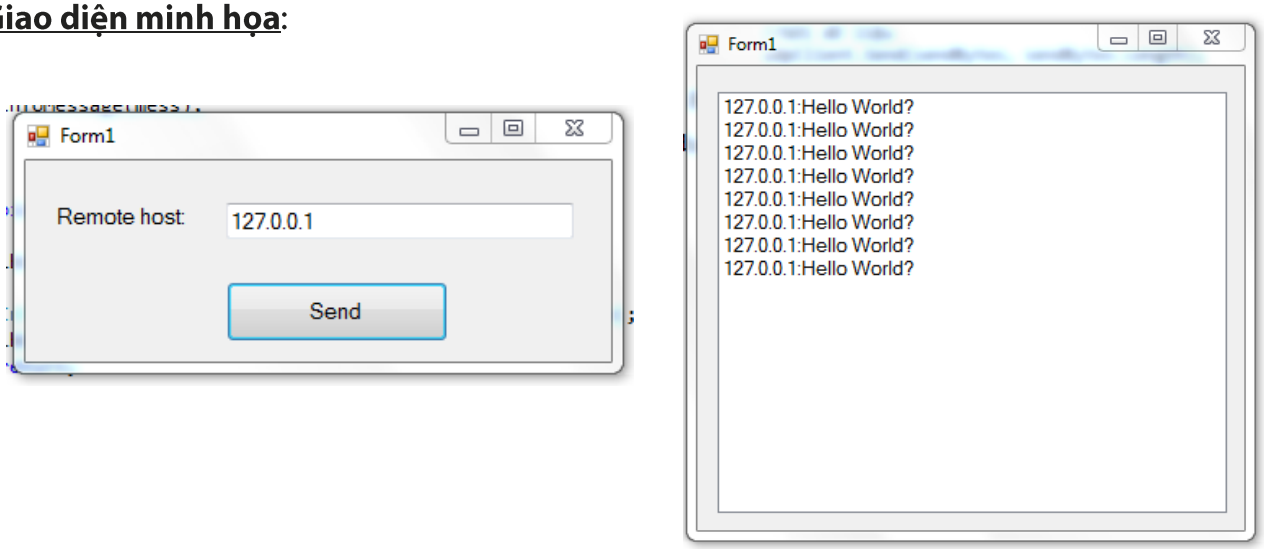
C. VÍ DỤ MINH HỌA

1. Chương trình UDP (Client – Server) cơ bản

Yêu cầu: Viết chương trình (Client – Server) theo UDP, cho phép Client gửi chuỗi lời chào đến Server "Hello World?

Bên A – gọi là UDP client, cần kết nối đến bên B – UDP server để gởi dữ liệu. Trong ví dụ này cho dữ liệu là chuỗi "Hello World?". Sử dụng địa chỉ IP Loopback và port mặc định là 8080. Mỗi khi bên A nhấn nút Send, "Hello World?" sẽ được gửi sang bên B.

Giao diện minh họa:



Hình 3: Giao diện chương trình UDP Client – Server

- Bên A

  Sự kiện cho nút **Send**:

```
private void button1_Click(object sender, EventArgs e)
{
    //Tạo kết nối UDP
  UdpClient udpClient = new UdpClient();
  //Do ý đồ muốn gởi dữ liệu là "Hello World?" sang bên nhận. Nên cần chuyển chuỗi Hello World sang kiểu byte
  Byte[] sendBytes = Encoding.ASCII.GetBytes("Hello World?");
    //Gởi dữ liệu mà không cần thiết lập kết nối với Server
  udpClient.Send(sendBytes, sendBytes.Length, tbHost.Text, 8080);
}
```

- Bên B:

➢ Sử dụng ListView để nhận dữ liệu từ bên A gởi.

➢ Bắt sự kiện xảy ra khi Form được Load lên

- Hàm **serverThread**: đón nhận dữ liệu từ bên A gởi sang và hiện lên ListView

```
public void serverThread()
{
        UdpClient udpClient = new UdpClient(8080);
        while (true)
        {
            IPEndPoint RemoteIpEndPoint = new IPEndPoint(IPAddress.Any, 0);
            Byte[] receiveBytes = udpClient.Receive(ref RemoteIpEndPoint);
            string returnData = Encoding.ASCII.GetString(receiveBytes);
            string mess = RemoteIpEndPoint.Address.ToString() + ":" +
                    returnData.ToString();
           //Viết hàm InfoMessage để hiển thị thông điệp lên List View
            InfoMessage(mess);
        }
}
```

Giải thích:

- Tại bên nhận, khai báo kết nối UDP với số hiệu cổng là 8080 và địa chỉ IPAdress là Any.
- IPAddress.Any nghĩa là socket này có thể listen từ bất kì địa chỉ IP nào dùng port 8080.
- Sự kiện load Form: Tạo một thread để chạy hàm nhận dữ liệu từ bên gởi.

```
private void Form1_Load(object sender, EventArgs e)
        {
            //Xử lý lỗi InvalidOperationException
          CheckForIllegalCrossThreadCalls = false;
          Thread thdUDPServer = new Thread(new ThreadStart(serverThread));
          thdUDPServer.Start();
        }
```

D. BÀI TẬP

Các bài thực hành dưới đây yêu cầu viết chương trình dưới dạng *Windows Forms App*.

Sinh viên có thể tùy biến cách sắp xếp giao diện khác sao cho hợp lý.

1. Bài 01 – UDP Client – Server

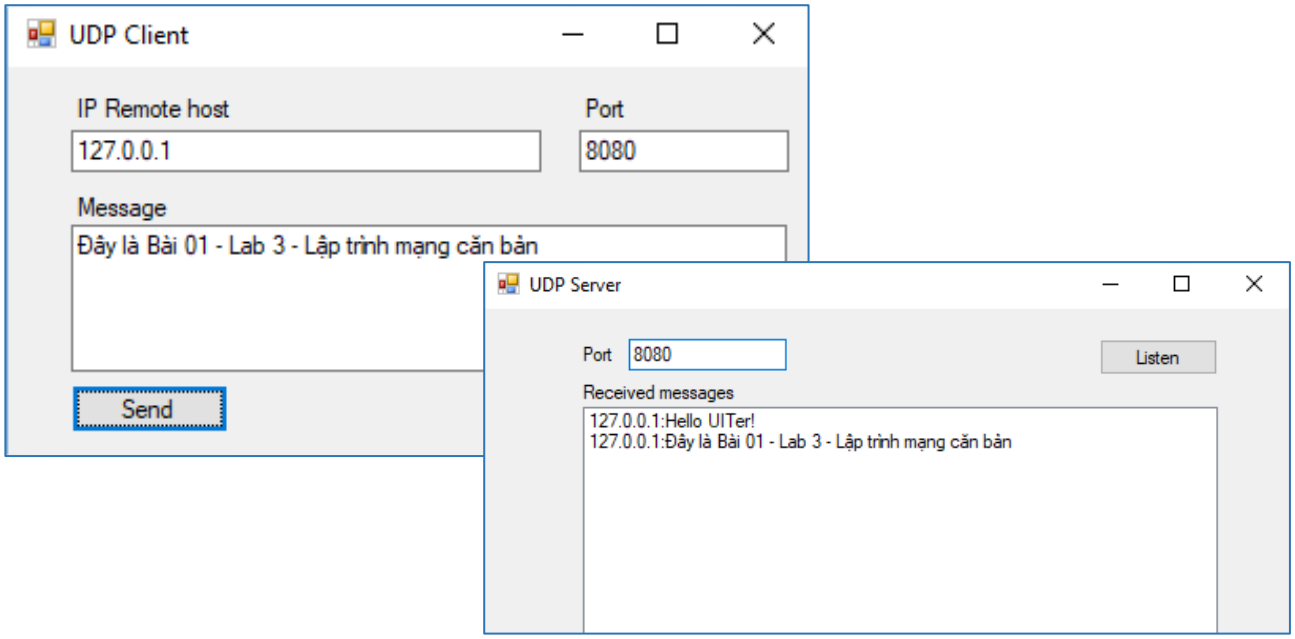
Yêu cầu:

Viết ứng dụng thực hiện gởi và nhận dữ liệu giữa hai bên sử dụng giao thức UDP (UDP Client và UDP Server)

Yêu cầu chi tiết

- Client sẽ chỉ định IP, port cần kết nối và thông điệp gửi đến Server
- Server nhận và hiển thị thông điệp gửi từ Client

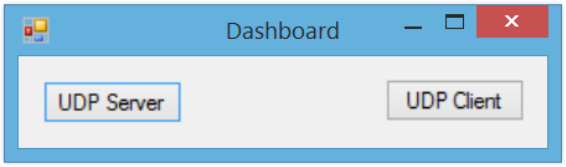
Giao diện minh họa:



Hình 4: Giao diện chương trình gửi nhận theo UDP

Gợi ý:

- Tham khảo ví dụ và thay đổi các dữ liệu liên quan đến địa chỉ IP, Port và thông điệp
- Địa chỉ IP 127.0.0.1 là địa chỉ loopback. Để viết ứng dụng truyền thông qua mạng Lan, sử dụng địa chỉ IP Private phù hợp.
- Có thể chọn 1 trong 2 cách sau:
  - Mỗi Client – mỗi Server thuộc 1 project khác nhau, khi chạy bấm chọn chuột phải vào tên Project > Chọn Debug > Chọn Start new instance
  - Client và Server thuộc cùng 1 project, có điều hướng để khởi tạo Client và Server



2. Chương trình TCP Server đơn giản

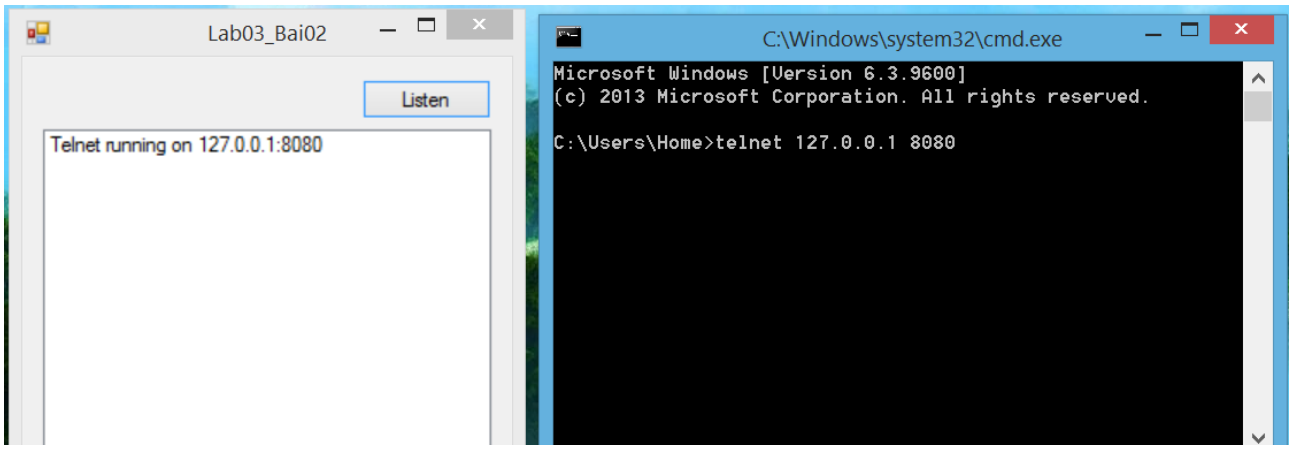
Yêu cầu:
Viết chương trình phía Server trình lắng nghe dữ liệu từ dịch vụ Telnet sử dụng kết nối TCP

Yêu cầu chi tiết:

Mô tả luồng hoạt động:

- Chạy chương trình
- Nhấn nút Listen
- Mở CMD gõ lệnh: **telnet <IP của máy> port**
- Vào màn hình telnet, gõ thông điệp tùy ý, chương trình sẽ nhận và hiện lên form

Giao diện minh họa:



Hình 5: Giao diện chương trình nhận dữ liệu qua Telnet

Gợi ý:

- Cần có TCP Server lắng nghe kết nối
- Để có thể telnet đến địa chỉ IP của máy tại cổng 8080, cần phải mở port 8080 lắng nghe kết nối TCP đến port 8080, do đó khi nhấn nút Listen có nghĩa là thực hiện lắng nghe kết nối tại địa chỉ IP của máy và cổng là 8080
- Sự kiện cho nút Listen:

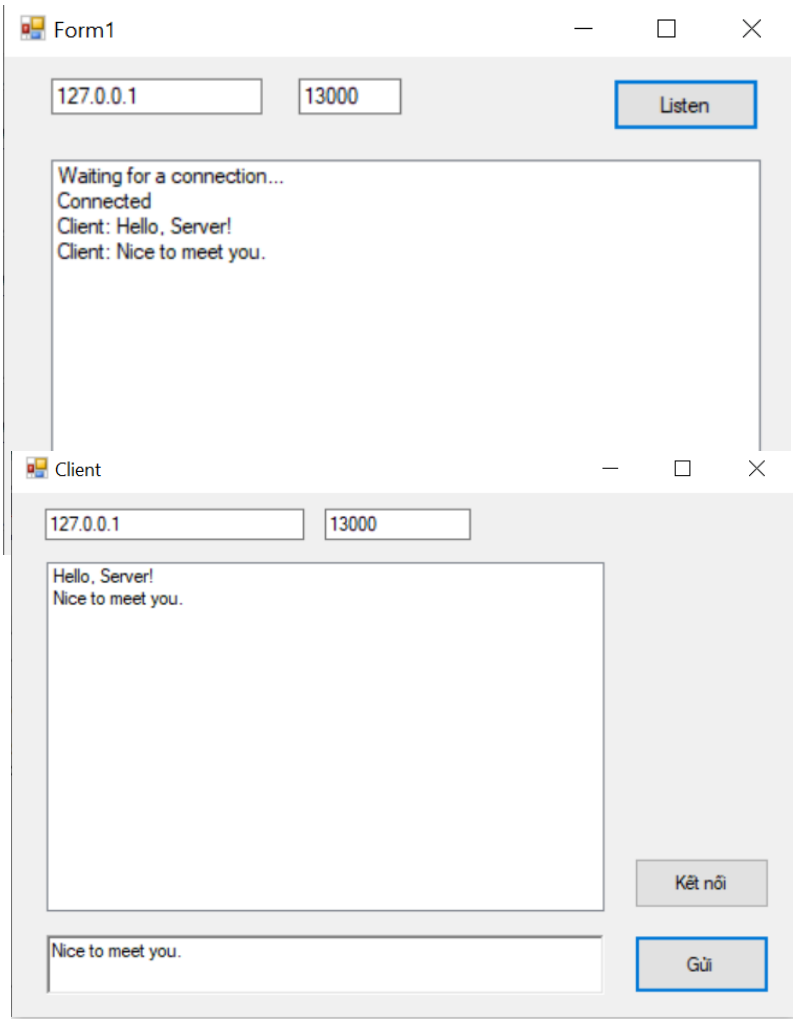
```
private void StartListen(object sender, EventArgs e)
{
    //Xử lý lỗi InvalidOperationException
    CheckForIllegalCrossThreadCalls = false;
    Thread serverThread = new Thread(new ThreadStart(StartUnsafeThread));
    serverThread.Start();
}
```

```
void StartUnsafeThread()
 {
      int bytesReceived = 0;
       // Khởi tạo mảng byte nhận dữ liệu
       byte[] recv = new byte[1];
       // Tạo socket bên gửi
       Socket clientSocket;
// Tạo socket bên nhận, socket này là socket lắng nghe các kết nối tới nó tại địa chỉ IP
của máy và port 8080. Đây là 1 TCP/IP socket.
//AddressFamily: Với địa chỉ Ipv4 cần chọn AddressFamily.InterNetwork
//SocketType: kiểu kết nối socket, ở đây dùng luồng Stream để nhận dữ liệu
       Socket listenerSocket = new Socket(
                  AddressFamily.InterNetwork,
                  SocketType.Stream,
                  ProtocolType.Tcp);
       IPEndPoint ipepServer = new IPEndPoint(IPAddress.Parse("127.0.0.1"),8080);
// Gán socket lắng nghe tới địa chỉ IP của máy và port 8080
      listenerSocket.Bind(ipepServer);
       // bắt đầu lắng nghe. Socket.Listen(int backlog)
      listenerSocket.Listen(-1);
       //Đồng ý kết nối
      clientSocket = listenerSocket.Accept();
       //Nhận dữ liệu
      listView.Items.Add(new ListViewItem("New client"));
       while (clientSocket.Connected)
       {
            string text = "";
           do
             {
                bytesReceived = clientSocket.Receive(recv);
                text += Encoding.ASCII.GetString(recv);
                } while (text[text.Length - 1] != '\n');
               listView.Items.Add(new ListViewItem(text));
            }
          listenerSocket.Close();
        }
```

3. Chương trình gửi nhận dữ liệu với TCP (1S – 1C)

Yêu cầu:
Viết ứng dụng thực hiện gửi và nhận dữ liệu sử dụng giao thức TCP (TCP Client và TCP Listener

Giao diện minh họa:



Hình 6: Giao diện chương trình chat 1 client – 1 server

Gợi ý: Sử dụng cách tạo Server ở bài 2, chỉ cần viết thêm ở phía Client.

Thực hiện theo các bước sau (Với phía Client)

1. Tạo một đối tượng System.Net.Sockets.TcpClient
2. Kết nối đến server với địa chỉ và port xác định với phương thức TcpClient.Connect()
3. Lấy luồng (stream) giao tiếp bằng phương thức TcpClient.GetStream().
4. Thực hiện giao tiếp với server.
5. Đóng luồng và socket.

```
 //1 Tạo đối tượng TcpClient
TcpClient tcpClient = new TcpClient();
 //2 Kết nối đến Server với 1 địa chỉ Ip và Port xác định
IPAddress ipAddress = IPAddress.Parse("127.0.0.1");
IPEndPoint ipEndPoint = new IPEndPoint (ipAddress, 8080);
tcpClient.Connect (ipEndPoint);
 //3 Tạo luồng để đọc và ghi dữ liệu dựa trên NetworkStream
NetworkStream ns = tcpClient.GetStream()
 //4 Dùng phương thức Write để gửi dữ liệu đến Server
Byte[] data = System.Text.Encoding.ASCII.GetBytes("Hello server\n");
ns.Write(data, 0, data.Length);
 //5 Dùng phương thức Write để gửi dữ liệu mang dấu hiệu kết thúc cho Server
biết và đóng kết nối
Byte[] data = System.Text.Encoding.ASCII.GetBytes("quit\n");
ns.Write(data, 0, data.Length);
ns.Close();
tcpClient.Close();
```

Ghi chú: Có thể tách thành 3 Hàm **khởi tạo, kết nối** (1,2,3) và **hàm gửi dữ liệu** (4) và **đóng kết nối** (5) riêng tương ứng với các action khác nhau (New Client, Send Data, Close Client). Gợi ý khai báo sẵn các đối tượng TcpClient, NetworkStream trong Class, và dùng this để gọi đến các đối tượng này.

4. Chương trình Chatroom đơn giản (1 S – n C)

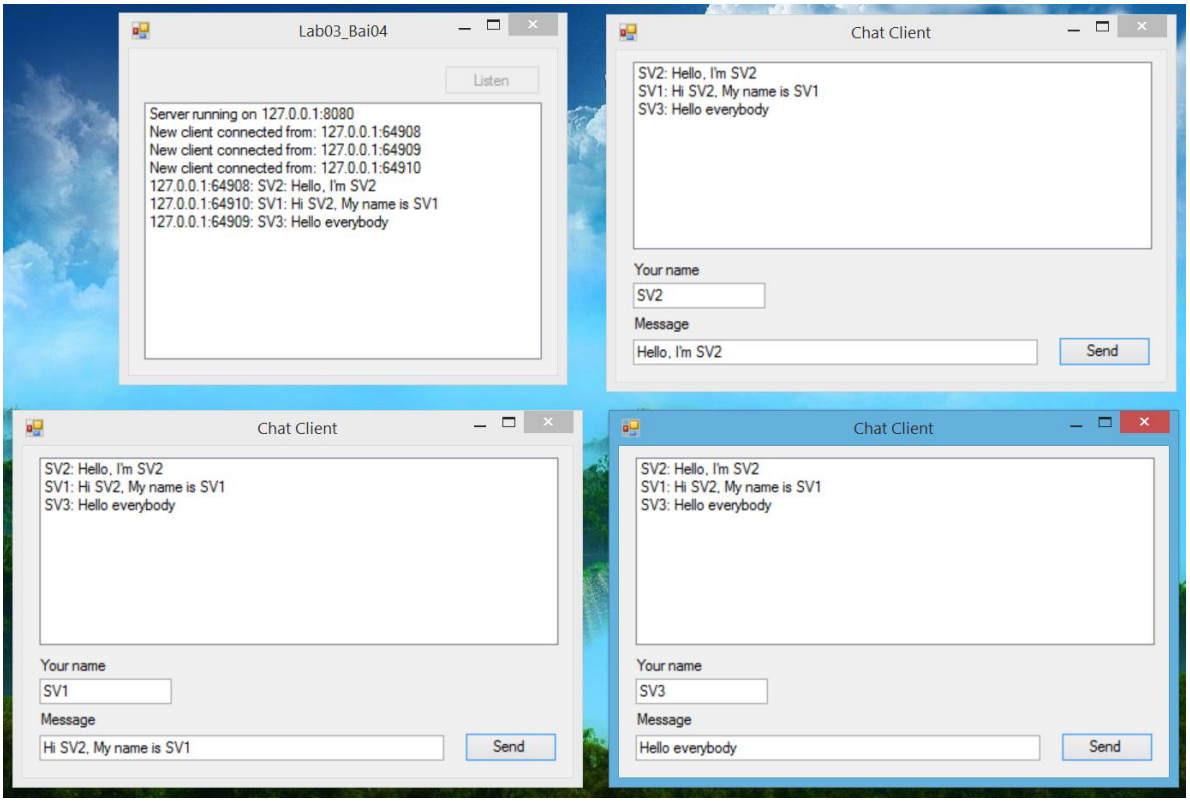
Yêu cầu:

Viết chương trình Chat Room/ Gửi và nhận dữ liệu. Mỗi người dùng sẽ có một tài khoản, khi một người dùng gửi tin nhắn thì tất cả đều sẽ nhận được tin nhắn đó

Yêu cầu chi tiết:

- Điền một tên bất kỳ để phân biệt với những người dùng khác
- Client kết nối đến server
- Server chỉ có nhiệm vụ quản lý các kết nối và broadcast tin nhắn đến các client khác cùng kết nối vào server.

Giao diện minh họa:



Hình 7: Giao diện chương trình chat n client – 1 server

5. Chương trình Chatroom nâng cao

Yêu cầu:
Viết chương trình Chat Room nâng cao cho phép quản lý phòng chat, tài khoản

Yêu cầu chi tiết:

- Thiết kế giao diện đẹp mắt, dễ sử dụng
- Cho phép người dùng đăng ký / đăng nhập
- Người dùng được phép lựa chọn hoặc khởi tạo phòng chat mới (Ví dụ: vào phòng chat bằng mã code)
- Khi client tham gia hoặc thoát phòng chat, các client khác nhận được thông báo

E. YÊU CẦU & ĐÁNH GIÁ

1. Yêu cầu

- Sinh viên thực hành và nộp bài **cá nhân** theo thời gian quy định.
- Bài nộp: **Source code** (nén) và 1 **File .pdf** báo cáo tóm lược các bài tập liên quan kèm hình ảnh minh họa

Toàn bộ project đặt vào 1 file nén (.rar/.zip)	**File .PDF screenshot**
LabX-MSSV-HọTênSV	**LabX-MSSV-HọTênSV.pdf**
Ví dụ: *Lab1-16520901-NguyenVanA*	Ví dụ: *Lab1-16520901-NguyenVanA.pdf*

2. Đánh giá kết quả

- Tiêu chí đánh giá:
  - Chương trình chạy được, hoàn thành các yêu cầu cơ bản: **+70%.**
  - Có kiểm tra các điều kiện ràng buộc khi nhập dữ liệu, code "sạch" [2], đặt tên biến rõ ràng: **+30%.**
  - Nộp bài không đầy đủ; lỗi, không chạy được; nộp trễ; sao chép code bạn khác, nguồn có sẵn: *xử lý tùy theo mức độ* **(- (10 → 100)%)**

F. THAM KHẢO

[1] Microsoft (2018). C# Guide. [Online] Available at: https://docs.microsoft.com/en-us/dotnet/csharp/

[2] Martin, R. C. (2009). *Clean code: a handbook of agile software craftsmanship*. Pearson Education.

HẾT