3

Lab

Lập trình Sockets trong C#

Working with Sockets in C#

Thực hành Lập trình mạng căn bản

GVHD: Phan Trung Phát

Lưu hành nội bộ

A. TỔNG QUAN

1. Mục tiêu

- Làm quen với Sockets trong C# và 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

- IDE Microsoft Visual Studio 2010 trở lên.

3. Liên quan

- Sinh viên cần nắm được các kiến thức nền tảng về lập trình. Các kiến thức này đã được giới thiệu trong các môn học trước và trong nội dung lý thuyết đã học do đó sẽ không được trình bày lại trong nội dung thực hành này.
- Tham khảo tài liệu (Mục E).

B. THỰC HÀNH

1. Giới thiệu về lập trình Socket và các lớp .NET cơ bản trong lập trình mạng

Socket là một API (Application Programming Interface) cung cấp các phương thức để giao tiếp thông qua mạng.

Các lớp .Net cơ bản trong lập trình mạng được cung cấp trong hai namespace System.Net và System.Net.Sockets. Hai namespace này chứa rất nhiều lớp dùng trong lập trình mạng.

Class	Namespace	Mô tả
IPAddress	System.Net	Thể hiện địa chỉ IP
IPEndPoint	System.Net	Thể hiện địa chỉ IP và Port number
TcpListener	System.Net.Sockets	Lắng nghe kết nối TCP từ Client
Socket	System.Net.Sockets	Socket
TcpClient	System.Net.Sockets	Tạo kết nối TCP từ Client

Cụ thể như:

Lớp IP Address thể hiện địa chỉ IP.

```
IPAddress ipadd = IPAddress.Parse("127.0.0.1");
```

Lớp IPEndpoint gồm địa chỉ IP và số hiệu cổng

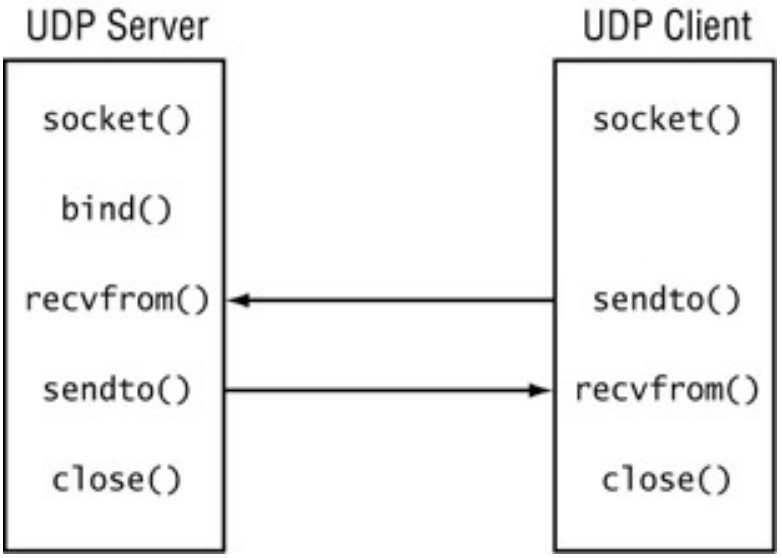
```
IPEndPoint RemoteIpEndPoint = new IPEndPoint(ipadd, 8080);
```

2. Kết nối UDP

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.

Giao thức 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.

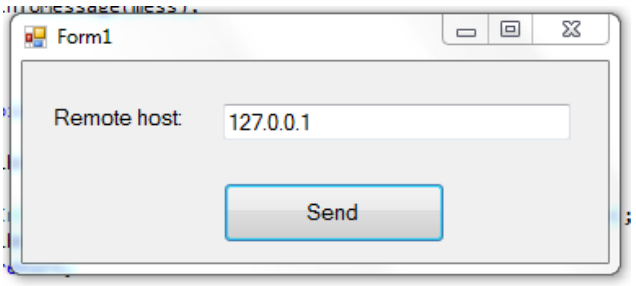
Sử dụng lớp **UDPClient** để thực hiện kết nối UDP giữa hai máy hoặc giữa 1 máy với nhiều máy.

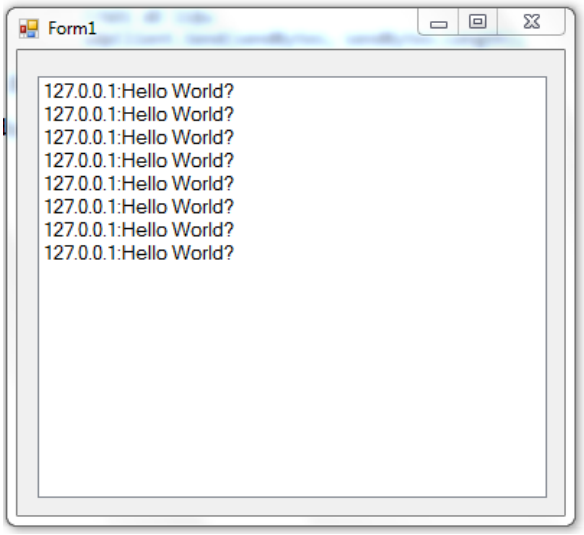


Cho **ví dụ** đơn giản sau:

Bên A – gọi là UDP client, cần kết nối đến bên B – UDP server để gửi 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 như hình minh họa bên dưới:





- 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

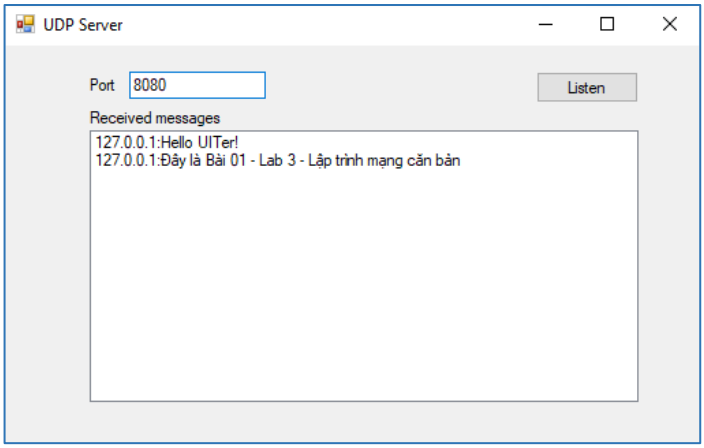
```
public void serverThread()
{
        UdpClient udpClient = new UdpClient(8080);
        while (true)
        {
            IPEndPoint RemoteIpEndPoint = new IPEndPoint(IPAddress.Any, 0);
            Byte[] receiveBytes = udpClient.Receive(ref RemoteIpEndPoint);
            string returnData = Encoding.ASCII.GetString(receiveBytes);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

Giải thích:

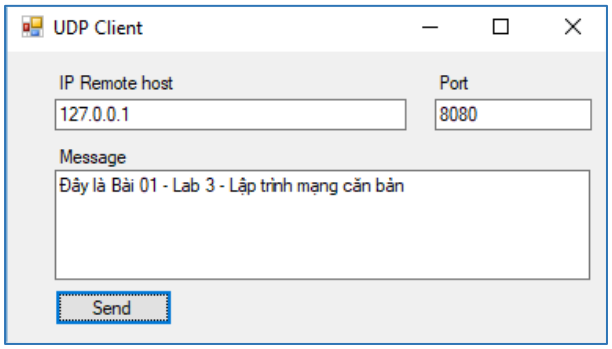
- Tại bên nhận, khai báo kết nối UDP với số hiệu cổng là 8080 và địa chỉ IPAdress là Any.
- IPAddress.Any nghĩa là socket này có thể listen từ bất kì địa chỉ IP nào dùng port 8080.
- Sự kiện load Form: Tạo một thread để chạy hàm nhận dữ liệu từ bên gửi.

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

Bài 01: Viết ứng dụng thực hiện gửi và nhận dữ liệu giữa hai bên sử dụng giao thức UDP (UDP Client và UDP Server). Người dùng ở Client sẽ chỉ định IP, port cần kết nối và thông điệp gửi đến Server. Tại Server sẽ nhận được thông điệp gửi từ Client.

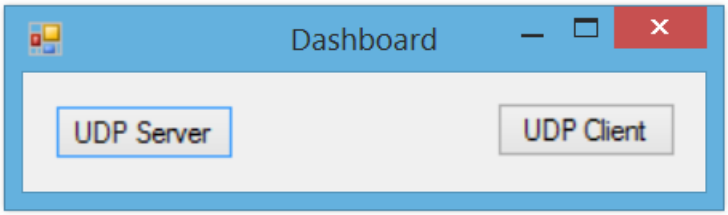


Giao diện tham khảo:



Gợi ý:

- *Tham khảo ví dụ trên 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.*
- *Client và Server thuộc cùng 1 project, có 1 Form chung để khởi tạo Client và Server (như hình bên dưới).*



Lưu ý:

- *Khai báo sử dụng các thư viện cần thiết.*
- *Trong ví dụ trên, nếu tại hàm serverThread(), ta set trực tiếp giá trị cho ListView (Thread unsafe call) thì sẽ phát sinh lỗi InvalidOperationException. Để giải quyết vấn đề trên, sinh viên cần xem lại kiến thức về Thread và sử dụng*

Thread-safe call với việc kiểm tra InvokeRequired. Sinh viên có thể tham khảo thêm tại đây.

- *Ứng dụng cần hỗ trợ gửi và nhận Tiếng Việt có dấu (UTF8).*

3. Kết nối TCP

Để đảm bảo độ tin cậy trong các ứng dụng mạng, người ta sử dụng giao thức TCP - một giao thức có kết nối.

Kết nối Server-Client với TCP/IP [1]

Khi được chạy, server cần được xác định rõ địa chỉ IP và sẽ "lắng nghe" trên một port cụ thể. Server sẽ nằm trong trạng thái này cho đến khi client gửi đến một yêu cầu kết nối. Sau khi được server chấp nhận, một connection sẽ hình thành cho phép server và client giao tiếp với nhau.

Cụ thể hơn, các bước tiến hành trên server và client mà ta cần thực hiện sử dụng giao thức TCP/IP trong C# (có thể chạy server và client trên cùng một máy):

Server:

1. Tạo một đối tượng System.Net.Sockets.TcpListener để bắt đầu "lắng nghe" trên một cổng cục bộ.
2. Đợi và chấp nhận kết nối từ client với phương thức AcceptSocket(). Phương thức này trả về một đối tượng System.Net.Sockets.Socket dùng để gửi và nhận dữ liệu.
3. Thực hiện giao tiếp với client.
4. Đóng Socket.

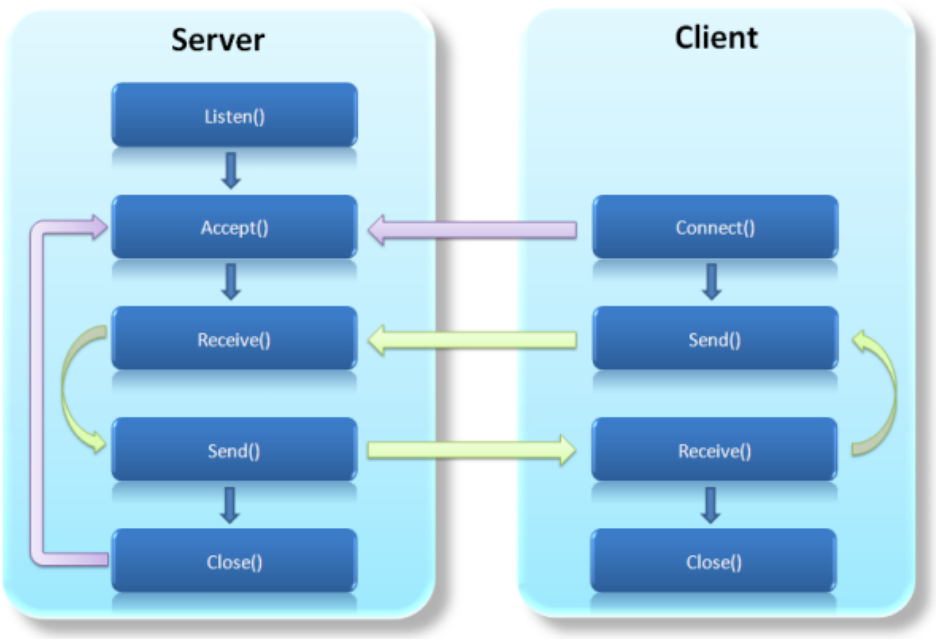
Thông thường quy trình này sẽ được đặt trong một vòng lặp (lặp lại bước 2) để chấp nhận nhiều kết nối cùng lúc (sử dụng Thread) hoặc các kết nối lần lượt.

[1] https://yinyangit.wordpress.com/2011/06/22/socket-communication-with-tcp-client-server/

Client:

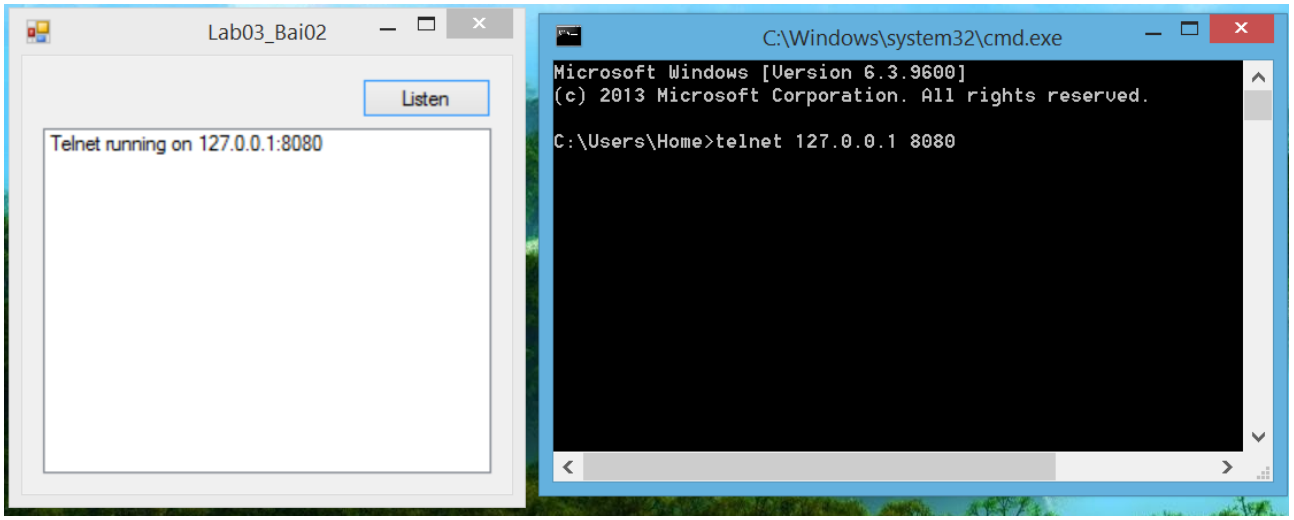
1. Tạo một đối tượng System.Net.Sockets.TcpClient
2. Kết nối đến server với địa chỉ và port xác định với phương thức TcpClient.Connect().
3. Lấy luồng (stream) giao tiếp bằng phương thức TcpClient.GetStream().
4. Thực hiện giao tiếp với server.
5. Đóng luồng và socket.

Quy trình này có thể được minh họa theo mô hình sau:



Bài 2: Viết chương trình lắng nghe dữ liệu từ dịch vụ Telnet sử dụng kết nối TCP (sử dụng lớp Socket) với mô tả sau:

1. Chạy chương trình.
2. Nhấn nút Listen.
3. Mở CMD gõ lệnh: **telnet <IP của máy> 8080**
4. Vào màn hình telnet, gõ thông điệp tùy ý, chương trình sẽ nhận và hiện lên form.
5. Xem hình mẫu.



Gợi ý:

- Cần có TCP Server lắng nghe kết nối.
- Để có thể telnet đến địa chỉ IP của máy tại cổng 8080, ta phải mở port 8080 lắng nghe kết nối TCP đến port 8080, do đó khi nhấn nút Listen có nghĩa là ta đang 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();
        }
```

- Hàm xử lý

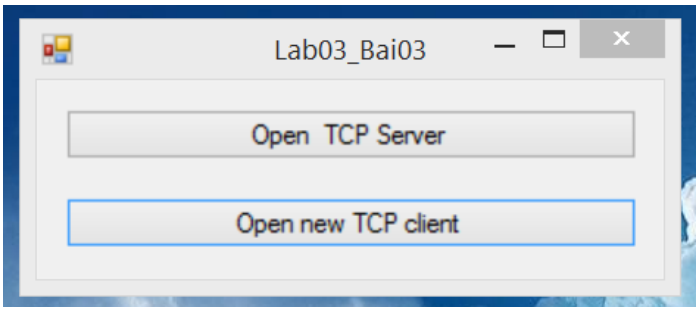
```
void StartUnsafeThread()
        {
            int bytesReceived = 0;
                // Khởi tạo mảng byte nhận dữ liệu
            byte[] recv = new byte[1];
```

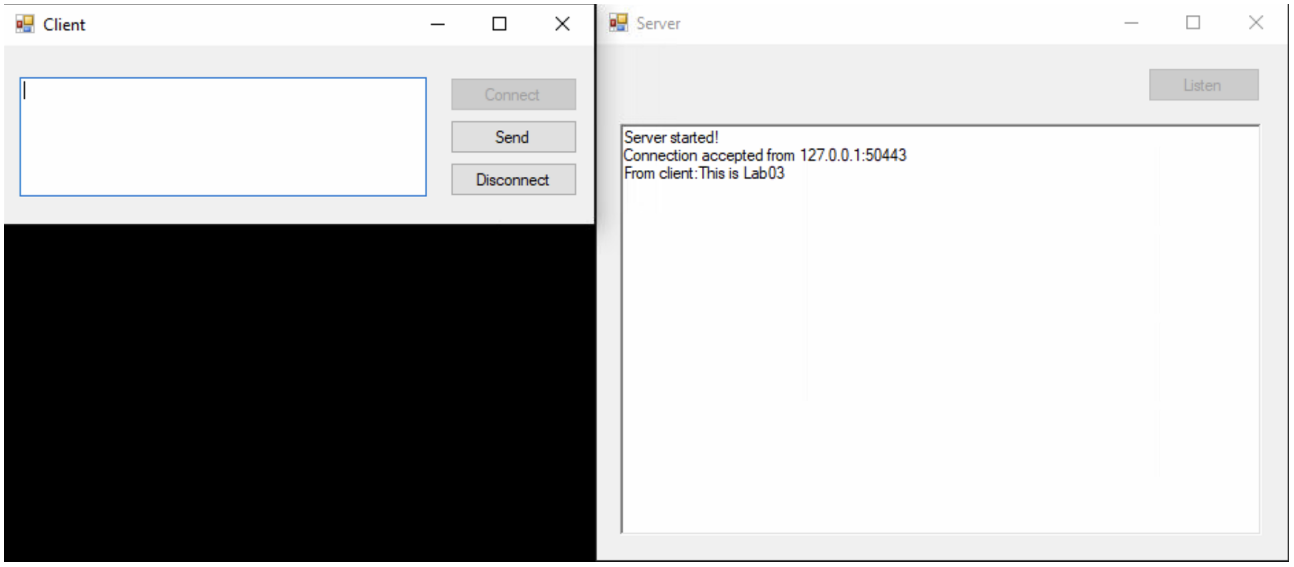
```
                // Tạo socket bên gởi
            Socket clientSocket;
        // Tạo socket bên nhận, socket này là socket lắng nghe các kết nối tới nó tại địa chỉ IP của máy và port 8080. Đây là 1 TCP/IP socket.
        //AddressFamily: trả về họ địa chỉ của địa chỉ IP hiện hành. Ở đây là địa chỉ Ipv4 nên chọn AddressFamily.InterNetwork
        //SocketType: kiểu kết nối socket, ở đây dùng luồng Stream để nhận dữ liệu
        //ProtocolType: sử dụng giao thức kết nối nào, ở đây sử dụng kết nối TCP
        // Ba tham số của hàm đi với nhau khi ta thực hiện kết nối TCP.
            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)
        // với backlog: là độ dài tối đa của hàng đợi các kết nối đang chờ xử lý
            listenerSocket.Listen(-1);
                //Đồng ý kết nối
            clientSocket = listenerSocket.Accept();
                //Nhận dữ liệu
           listViewCommand.Items.Add(new ListViewItem("New client connected"));
           while (clientSocket.Connected)
          {
           string text = "";
           do
            {
              bytesReceived = clientSocket.Receive(recv);
              text += Encoding.ASCII.GetString(recv);
```

```
            } while (text[text.Length - 1] != '\n');
            listViewCommand.Items.Add(new ListViewItem(text));
        }
        listenerSocket.Close();
    }
```

Bài 3: 1 Server – 1 Client Viết ứng dụng thực hiện gởi và nhận dữ liệu sử dụng giao thức TCP (TCP Client và TCP Listener). Server lắng nghe kết nối và thông điệp từ Client.

1. Chạy Server
2. Nhấn nút Listen
3. Khởi tạo Client
4. Gửi thông điệp từ Client đến Server
5. Server nhận thông điệp và hiện lên form.
6. Xem hình mẫu.





Gợi ý: Sử dụng cách tạo Server ở bài trên, 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: Tách thành 3 Hàm **khởi tạo, kết nối** và **hàm gửi dữ liệu** và **đóng kết nối** riêng tương ứng với các action khác nhau (Connect, Send, Disconnect). Khai báo sẵn các đối tượng trong Class, và dùng this để gọi đến các đối tượng này.

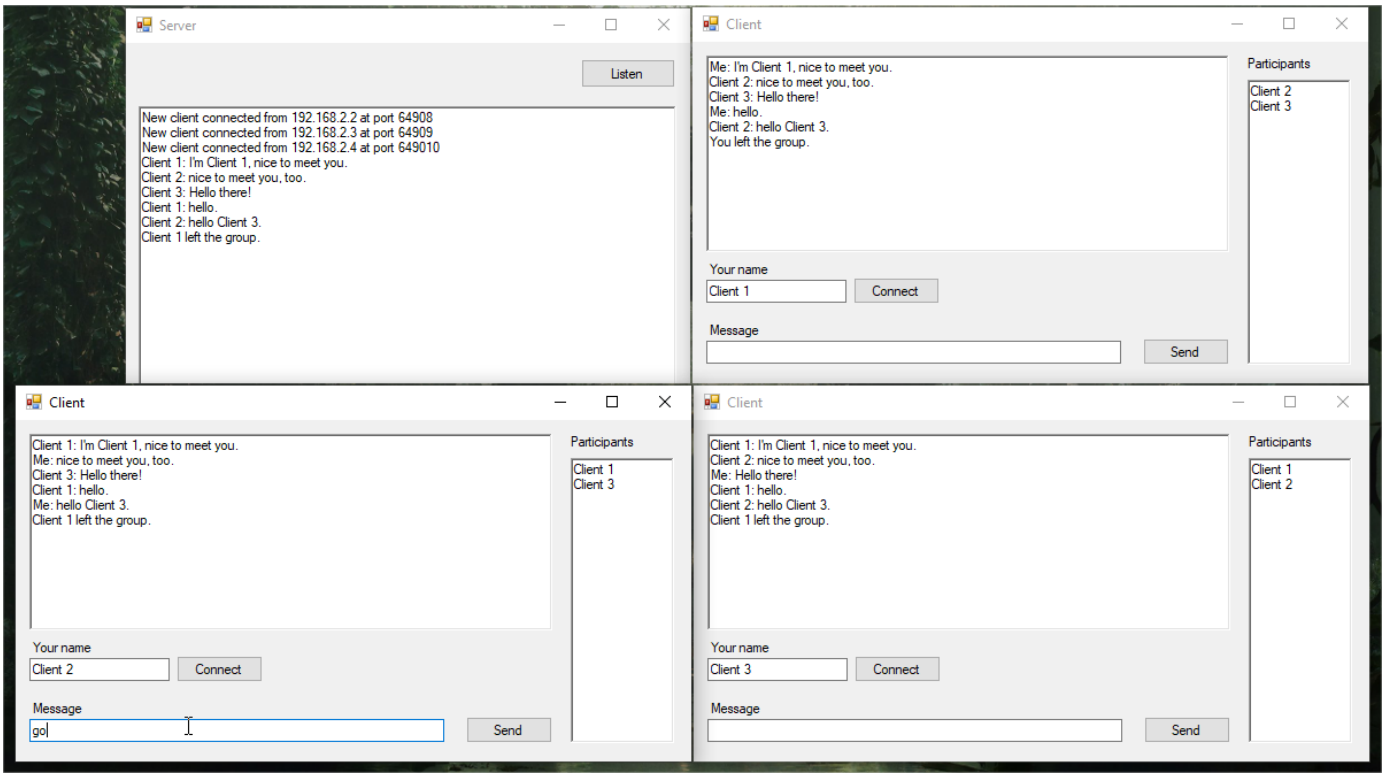
Bài 4: 1 Server – Multi Client **Quản lý phòng vé** (phiên bản số 3). Lấy ý tưởng và kế thừa từ bài 5 – bài thực hành số 2, dữ liệu về phòng vé mặc định đã có sẵn từ file. Dữ liệu được thống nhất lưu trữ tại Server. Các chức năng tương tự, tuy nhiên thông tin vé được đặt là đồng bộ trong hệ thống đối với các Client. Nếu vé đã được đặt ở 1 quầy thì các quầy khác không đặt được vé đó nữa (hệ thống sẽ thông báo hoặc sử dụng các properties của control để giới hạn) một cách thời gian thực.

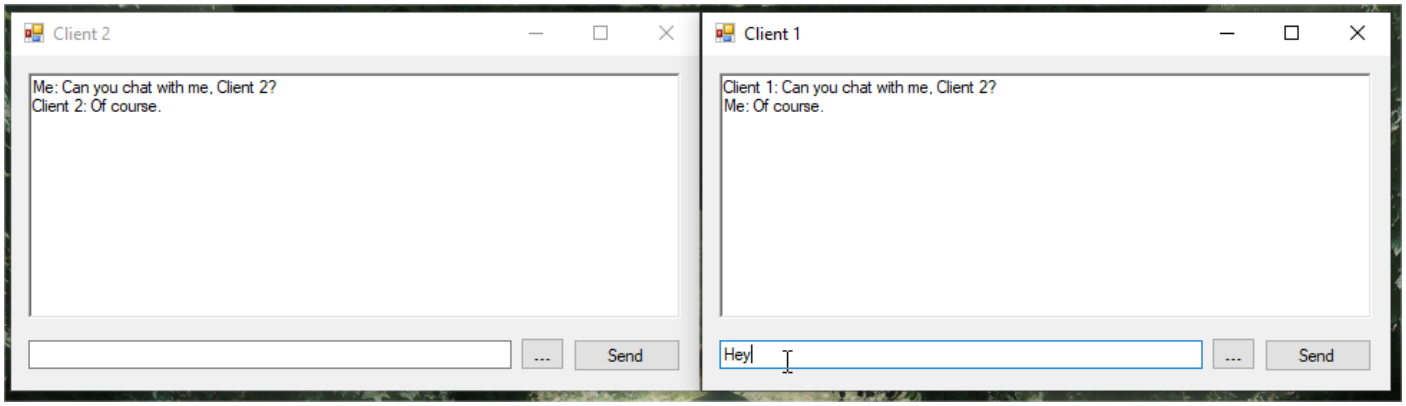
Bài 5: 1 Server – Multi Client Viết chương trình Chat Room/ Gửi và nhận dữ liệu sử dụng TCP Client và TCP Listener. Mỗi người dùng sẽ có một tài khoản, khi một người dùng gửi tin nhắn thì tất cả mọi người còn lại đều sẽ nhận được tin nhắn đó. Thêm vào đó là tính năng nhắn tin riêng, người dùng có thể chọn một người dùng khác để nhắn tin riêng. Cho phép gửi tin nhắn dạng hình ảnh (.jpg và .png) và file (.txt).

Gợi ý:

- Chỉ cần điền một tên bất kỳ để phân biệt với những người dùng khác (Không cần register, login).

- Client kết nối đến server. Server chỉ có nhiệm vụ quản lý các kết nối và broadcast tin nhắn đến các client cần thiết cùng kết nối vào server.





C. YÊU CẦU & NỘP BÀI

1. Yêu cầu

- Có form điều hướng để mở các form liên quan.
- Các giao diện ở trên chỉ mạng tính chất minh họa, sinh viên tiến hành thiết kế giao của riêng mình đảm bảo các tiêu chí: dễ nhìn, thể hiện hết được các yêu cầu cần thực hiện, đẹp.

- Code "sạch" [2], đặt tên biến rõ ràng.
- 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.*

2. **Đánh giá kết quả**

- Sinh viên thực hành và nộp bài theo **Nhóm (Nhóm trưởng nộp)** tại website môn học theo thời gian quy định.
- Source code được nộp tại GitHub và báo được nộp dưới định dạng:

Toàn bộ các file liên quan đặt vào 1 file nén (.zip) với tên theo quy tắc sau:

Mã lớp-LabX-MSSV1-MSSV2

Ví dụ: *NT106.M21.MMCL.1-Lab03-25520001-25520002*

D. 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.

[3] DNS: https://msdn.microsoft.com/en-us/library/system.net.dns(v=vs.110).aspx

[4] system.net.sockets: https://msdn.microsoft.com/en-us/library/system.net.sockets(v=vs.110).aspx

[5] Socket: https://msdn.microsoft.com/en-us/library/system.net.sockets.socket(v=vs.110).aspx

[6] IPHostEntry: https://msdn.microsoft.com/en-us/library/system.net.iphostentry(v=vs.110).aspx

[7] IPAddress: https://msdn.microsoft.com/en-us/library/system.net.ipaddress(v=vs.110).aspx

[8] IPEndpoint :

https://msdn.microsoft.com/en-us/library/system.net.ipendpoint(v=vs.110).aspx

HẾT