CHƯƠNG 3
SOCKETS

ThS. Trần Bá Nhiệm
Website:
sites.google.com/site/tranbanhiem
Email: tranbanhiem@gmail.com

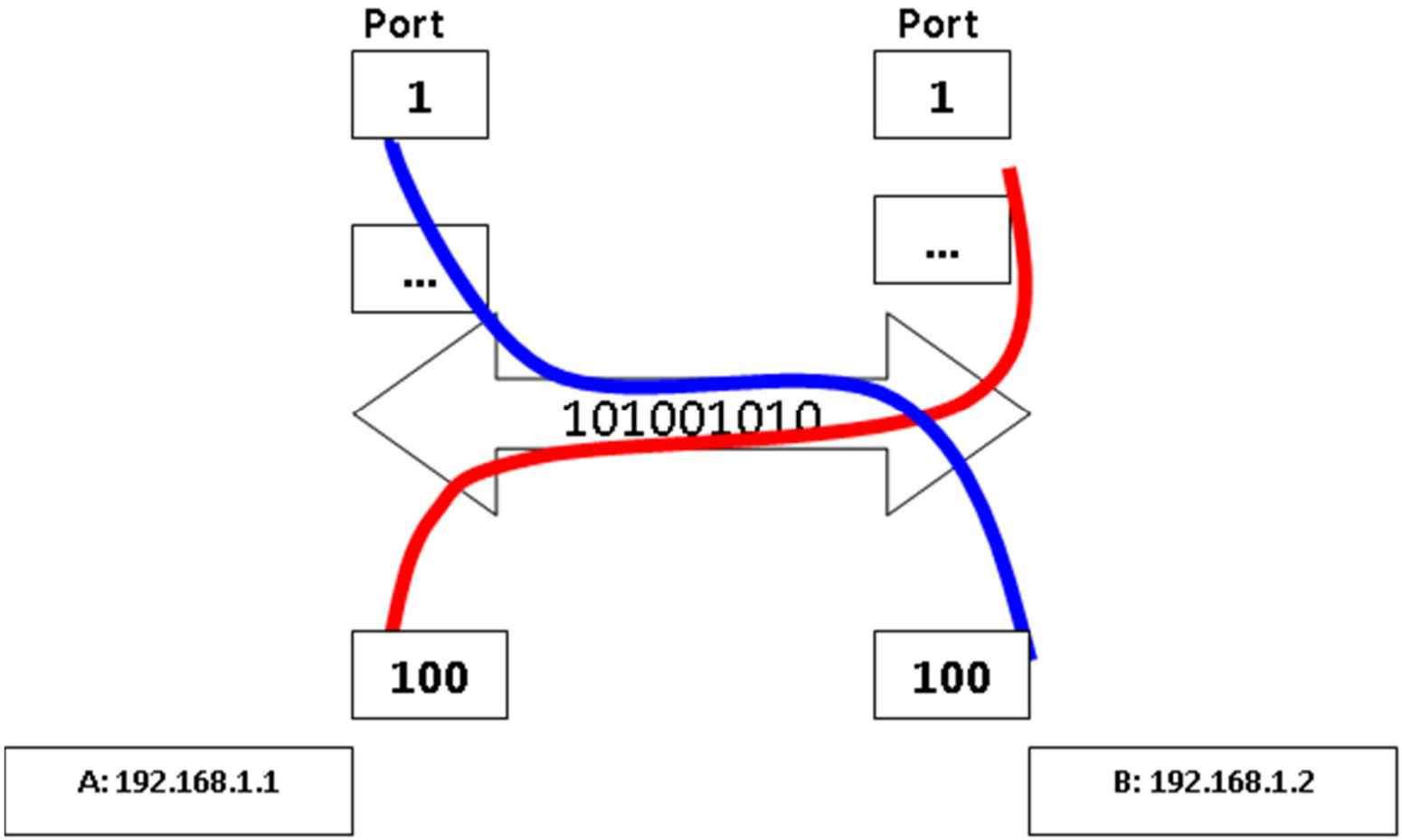
Nội dung

- Giới thiệu
- Khảo sát chức năng của các lớp Socket, UDP, TCP (TCPClient & TCPListener) và các lớp IPAddress, IPHostEntry, IPEndpoint trong lập trình mạng
- Khai báo và sử dụng các lớp UDP, TCP

Socket là gì?

- Lập trình mức socket là nền tảng của lập trình mạng
- Socket là một đối tượng thể hiện điểm truy cập mức thấp vào IP stack.
- Socket có thể ở chế độ mở, đóng hoặc một số trạng thái trung gian khác
- Socket có thể gửi, nhận dữ liệu
- Dữ liệu tổng quát được gửi theo từng khối (thường gọi là packet), khoảng vài KB/lần để tăng hiệu suất

Khái niệm địa chỉ và cổng (Address & Port)



Địa chỉ và cổng: nguyên lý

- Trong máy có rất nhiều ứng dụng muốn trao đổi với các ứng dụng khác thông qua mạng.
  - Ví dụ: có 2 ứng dụng của máy A muốn trao đổi với với 2 ứng dụng trên máy B
- Mỗi máy tính chỉ có duy nhất một đường truyền dữ liệu (để gửi và nhận)

Địa chỉ và cổng: vấn đề

- Có thể xảy ra "nhầm lẫn" khi dữ liệu từ máy A gửi đến máy B thì trên máy B không biết là dữ liệu đó gửi cho ứng dụng nào?
- Mỗi ứng dụng trên máy B sẽ được gán một số hiệu (cổng: Port), từ 0..65535.

Địa chỉ và cổng: cách giải quyết

- Khi ứng dụng trên máy A muốn gửi cho ứng dụng nào trên máy B thì chỉ việc điền thêm số hiệu cổng (vào trường RemotePort) vào gói tin cần gửi.
- Trên máy B, các ứng dụng chỉ việc kiểm tra giá trị cổng trên mỗi gói tin xem có trùng với số hiệu cổng của mình (đã được gán – chính là giá trị LocalPort) hay không? Nếu bằng thì xử lý, trái lại thì không làm gì (vì không phải là của mình).

Ứng dụng và cổng thường gặp

Port	Protocol
20	FTP data
21	FTP control
25	SMTP (email, outgoing)
53	DNS (Domain Name Service)
80	HTTP (Web)
110	POP3 (email, incoming)
143	IMAP (email, incoming)

Một số quy định

- Không bao giờ có 2 ứng dụng lại cùng dùng 1 port
- Các port từ 0 – 1023 (Well-know): dùng cho các ứng dụng quan trọng trên hệ điều hành
- Các port từ 1024 – 49151 (Registered): dành cho người lập trình (khuyến cáo tuân theo)
- Các port từ 49152 – 65535 (Dynamic): dự trữ

Lớp IPAddress

- Trên Internet mỗi một trạm (có thể là máy tính, máy in, thiết bị …) đều có một định danh duy nhất, định danh đó thường được gọi là một địa chỉ (Address).
- Địa chỉ trên Internet là một tập hợp gồm 4 con số có giá trị từ 0-255 và cách nhau bởi dấu chấm.

Lớp IPAddress

- Để thể hiện địa chỉ này, người ta có thể viết dưới các dạng sau:
  - Tên: ví dụ như May01, Server, …
  - Địa chỉ IP nhưng đặt trong một chuỗi: "192.168.1.1", "127.0.0.1“
  - Đặt trong một mảng 4 byte, mỗi byte chứa một số từ 0-255.
  - Hoặc cũng có thể là một số (long), có độ dài 4 byte. Ví dụ, với địa chỉ 192.168.1.1 ở trên thì giá trị đó sẽ là 16885952 (số ở hệ thập phân khi xếp liền 4 byte ở trên lại với nhau
    0000000100000001**10101000**11000000

Lớp IPAddress

- Như vậy, để đổi một địa chỉ chuẩn ra dạng số chúng ta chỉ việc tính toán cho từng thành phần.
- Ví dụ: Đổi địa chỉ 192.168.1.2 ra số, ta tính như sau:

2 * 256 ^ 3 + 1 * 256 ^ 2 + 168 * 256 ^ 1 + 192 * 256 ^ 0 = 33663168

Lớp IPAddress: các thành viên

Tên thuộc tính	Mô tả
Any	Cung cấp một địa chỉ IP (thường là 0.0.0.0) để chỉ ra rằng Server phải lắng nghe các hoạt động của Client trên tất cả các Card mạng (sử dụng khi xây dựng Server). Thuộc tính này chỉ đọc.
Broadcast	Cung cấp một địa chỉ IP quảng bá (Broadcast, thường là 255.255.255.255), ở dạng số Long. Muốn lấy ở dạng chuỗi, viết: Broadcast.ToString(). Thuộc tính này chỉ đọc.
Loopback	Trả về một địa chỉ IP lặp (IP Loopback, ví dụ 127.0.0.1). Thuộc tính này chỉ đọc.
Address	Một địa chỉ IP (An Internet Protocol (IP) address) ở dạng số Long. (Muốn chuyển sang dạng dấu chấm, viết : Address.ToString().

Lớp IPAddress: các thành viên

Tên phương thức	Mô tả
AddressFamily	Trả về họ địa chỉ của địa chỉ IP hiện hành. Nếu địa chỉ ở dạng IPv4 thì kết quả là Internetwork, và InternetworkV6 nếu là địa chỉ IPv6.
Constructor	- IPAddress(Số_Long) → Tạo địa chỉ IP từ một số kiểu long - IPAddress(Mảng_Byte) → Tạo địa chỉ IP từ một mảng byte (4 byte).
GetAddressBytes	Chuyển địa chỉ thành mảng byte (4 byte).
HostToNetworkOrder	Đảo thứ tự byte của một số cho đúng với thứ tự byte trong địa chỉ IPAddress.

Lớp IPAddress: các thành viên

Tên phương thức	Mô tả
IsLoopback	Cho biết địa chỉ có phải là địa chỉ lặp hay không?
NetworkToHostOrder	Đảo thứ tự byte của một địa chỉ cho đúng với thứ tự byte thông thường.
Parse	Chuyển một địa chỉ IP ở dạng chuỗi thành một địa chỉ IP chuẩn (Một đối tượng IPAddress)
ToString	Trả về địa chỉ IP (một chuỗi) nhưng ở dạng ký pháp có dấu chấm. (Ví dụ "192.168.1.1").
TryParse (S: String)	Kiểm tra xem một địa chỉ IP (ở dạng chuỗi) có phải đúng là địa chỉ IP hợp lệ hay không? True = đúng

IPAddress: Ví dụ tạo địa chỉ

- Cách 1: *Dùng hàm khởi tạo*

```
Byte[] b = new Byte[4];
b[0] = 192;
b[1] = 168;
b[2] = 10;
b[3] = 10;
IPAddress Ip1 = new IPAddress(b);
```

IPAddress: Ví dụ tạo địa chỉ

- Cách 2: *Dùng hàm khởi tạo*

  IPAddress Ip2 = new IPAddress(16885952);
- Cách 3: *Dùng hàm khởi tạo*

  IPAddress Ip3 = IPAddress.Parse("172.16.1.1")
- Cách 4: *Thông qua tính toán*

  Long So = 192* 256^0+168* 256^1+1* 256^2 + 2*256^3;

  IPAddress Ip4 = new IPAddress(So);

IPAddress: Ví dụ kiểm tra địa chỉ

```
private void KiemTra()
{
    IPAddress ip;
    String Ip4 = "127.0.0.1";
    String Ip5 = "999.0.0.1";
    MessageBox.Show(IPAddress.TryParse(Ip4, out
   ip).ToString());
    MessageBox.Show(IPAddress.TryParse(Ip5, out
   ip).ToString());
}
```

IPAddress: Ví dụ chuyển địa chỉ hiện hành ra mảng

```
void ChuyenDoi()
{
        IPAddress Ip3 = new IPAddress(16885952);
        Byte[] b= new Byte[4];
        b = Ip3.GetAddressBytes();
        MessageBox.Show("Address: " + b[0] + "." +
    b[1] + "." + b[2] + "." + b[3])
}
```

Lớp IPEndpoint

- Trong mạng, để hai trạm có thể trao đổi thông tin được với nhau thì chúng cần phải biết được địa chỉ (IP) của nhau và số hiệu cổng mà hai bên dùng để trao đổi thông tin.
- Lớp IPAddress mới chỉ cung cấp địa chỉ IP (IPAddress), như vậy vẫn còn thiếu số hiệu cổng (Port number).
- Lớp IPEndpoint chính là lớp chứa đựng cả IPAddress và Port number.

Lớp IPEndpoint: các thành viên

Tên thuộc tính	Mô tả
Address	Trả về hoặc thiết lập địa chỉ IP cho endpoint. (Trả về một đối tượng IPAddress)
AddressFamily	Lấy về loại giao thức mà Endpoint này đang sử dụng.
Port	Lấy về hoặc thiết lập số hiệu cổng của endpoint.

Lớp IPEndpoint: các thành viên

Tên phương thức	Mô tả
IPEndPoint (Int64, Int32)	Tạo một đối tượng mới của lớp **IPEndPoint,** tham số truyền vào là địa chỉ IP (ở dạng số) và **cổng** sẽ dùng để giao tiếp.
IPEndPoint (IPAddress, Int32)	Tạo một đối tượng mới của lớp **IPEndPoint,** Tham số truyền vào là một địa chỉ **IPAddress** và số hiệu cổng dùng để giao tiếp. (Tham khảo cách tạo IPAddress ở phần trên)
Create	Tạo một endpoint từ một địa chỉ socket (socket address).
ToString	Trả về địa chỉ IP và số hiệu cổng theo khuôn dạng ĐịaChỉ: Cổng, ví dụ: 192.168.1.1:8080

Lớp IPEndpoint: ví dụ khởi tạo

```
private void TaoEndpoint()
{
    // Tạo một địa chỉ IP
    IPAddress IPAdd =
    IPAddress.Parse("127.0.0.1");
    // Truyền vào cho hàm khởi tạo để tạo
    IPEndpoint
    IPEndPoint IPep = new IPEndPoint(IPAdd,
    10000);
}
```

Lớp IPEndpoint: ví dụ khởi tạo

```
private void TaoEndPointBoiTenMay()
{
    IPAddress IPAdd = new IPAddress();
    //tạo đối tượng IP từ tên của máy thông qua phương thức tĩnh Dns.GetHostAddresses của lớp DNS
    IPAdd = Dns.GetHostAddresses("Localhost")[0];
    IPEndPoint IPep = new IPEndPoint(IPAdd, 10000);
}
```

Lớp IPEndpoint: ví dụ khởi tạo

- Lưu ý : Vì một máy tính có thể có nhiều card mạng (Interface) do vậy có thể có nhiều hơn 1 địa chỉ IP.
- Hàm GetHostAddresses sẽ trả về cho chúng ta một mảng chứa tất cả các địa chỉ đó.
- Chúng ta lấy chỉ số là 0 để chọn địa chỉ của card mạng đầu tiên.

Lớp IPHostEntry

- IPHostEntry là lớp chứa (Container) về thông tin địa chỉ của các máy trạm trên Internet.
- Lưu ý: Nó chỉ là nơi để "chứa", do vậy trước khi sử dụng cần phải “nạp" thông tin vào cho nó.
- Lớp này rất hay được dùng với lớp DNS

Lớp IPHostEntry: các thành viên

Tên thuộc tính	Mô tả
AddressList	Lấy về hoặc thiết lập danh sách các địa chỉ IP liên kết với một host.
Aliases	Lấy về hoặc thiết lập danh sách các bí danh (alias) liên kết với một host.
HostName	Lấy về hoặc thiết lập DNS name của host.

Lớp DNS

- DNS (Domain Name Service) là một lớp giúp chúng ta trong việc phân giải tên miền (Domain Resolution) đơn giản.
- Phân giải tên miền tức là: Đầu vào là tên của máy trạm thì đầu ra sẽ cho ta địa chỉ IP tương ứng của máy đó, ví dụ: ServerCNTT →192.168.3.8
- Ngoài ra lớp Dns còn có rất nhiều phương thức cho chúng ta thêm thông tin về máy cục bộ như tên, địa chỉ, v.v.

Lớp DNS: các thành viên

Tên phương thức	Mô tả
GetHostByAddress (IP As String) GetHostByAddress (IP As IPAddress) As IPHostEntry	Trả về thông tin (IPHostEntry) của trạm có địa chỉ IP được truyền vào. → Thay bằng **GetHostEntry**()
GetHostByName (Tên trạm: String) As IPHostEntry	Trả về thông tin (IPHostEntry) DNS của một trạm → Đã bị loại bỏ. Thay bằng **GetHostEntry()**
HostName	Cho ta biết tên của máy vừa được phân giải. Nếu không phân giải được thì có giá trị là địa chỉ IP.

Lớp DNS: các thành viên

Tên phương thức	Mô tả
GetHostAddresses (IP_Or_HostName: String) as IPAddress()	Trả về tất cả các địa chỉ IP của một trạm.
GetHostEntry (IP_Or_HostName As String) GetHostEntry (IP As IPAddress)	Giải đáp tên hoặc địa chỉ IP truyền vào và trả về một đối tượng **IPHostEntry** tương ứng.
GetHostName As String	Lấy về tên của máy tính cục bộ.
Resolve (Hostname: String)	Chuyển tên của máy hoặc địa chỉ IP thành **IPHostEntry** tương ứng. → Đã bị bỏ, thay bằng **GetHostEntry()**

Lớp DNS: các thành viên

- Lưu ý: Đây là các **phương thức tĩnh**, do vậy khi gọi thì gọi trực tiếp từ tên lớp mà không cần phải khai báo một đối tượng mới của lớp này.
- Ví dụ: Dns.Resolve, Dns.GetHostname, Dns.GetHostEntry, v.v…

Lớp DNS: ví dụ 1

```
private void ShowIPs()
{
        // Lấy tất cả địa chỉ IP của máy
         IPAddress[] add = Dns.GetHostAddresses("Nhiem-PC");
        foreach (IPAddress ip in add) {
           MessageBox.Show(ip.ToString());
        }
        //for (int i = 0; i < add.Length; i++)
        //{
        //    MessageBox.Show(add[i].ToString());
        //}
}
```

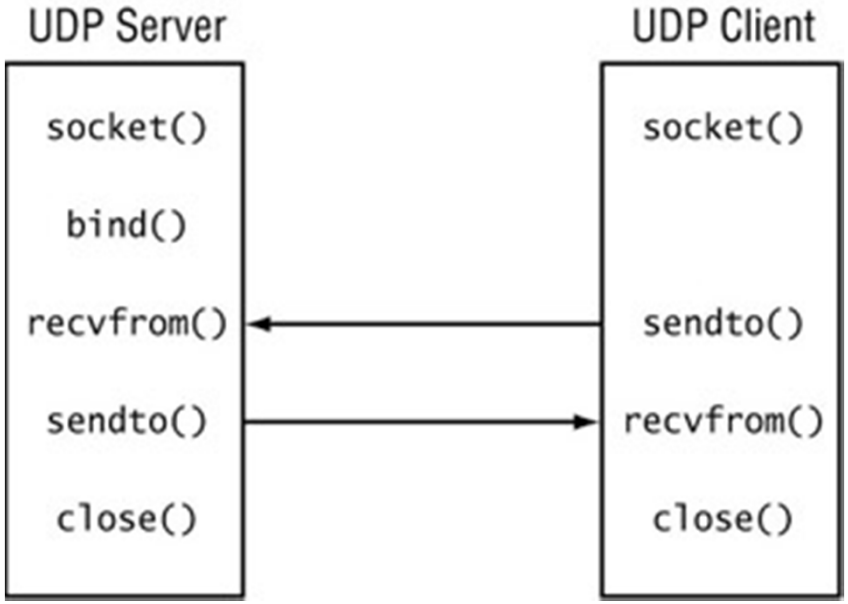
Lớp DNS: ví dụ 2

```
private void CreatIPHostEntry() {
        IPHostEntry iphe1, iphe2, iphe3;
        IPAddress ipadd =
IPAddress.Parse("127.0.0.1");
        iphe1 = Dns.GetHostEntry("Notebook");
        iphe2 = Dns.GetHostEntry("127.0.0.1");
        iphe3 = Dns.GetHostEntry(ipadd);
        MessageBox.Show(iphe1.HostName);
        MessageBox.Show(iphe2.HostName) ;
        MessageBox.Show(iphe3.HostName) ;
}
```

Lớp UDPClient

- Giao thức UDP (User Datagram Protocol hay User Define Protocol) là một giao thức phi kết nối (Connectionless)
- Nói cách khác là không cần thiết lập kết nối giữa hai bên khi tiến hành trao đổi thông tin.
- Giao thức này không tin cậy bằng giao thức TCP nhưng tốc độ lại nhanh và dễ cài đặt. Ngoài ra, với giao thức UDP ta còn có thể gửi các gói tin quảng bá (Broadcast) cho đồng thời nhiều máy

Lớp UDPClient: trình tự kết nối



Lớp UDPClient: các thành viên

Tên phương thức, thuộc tính	Mô tả
UdpClient ()	Tạo một đối tượng (thể hiện) mới của lớp UDPClient.
UdpClient (AddressFamily)	Tạo một đối tượng (thể hiện) mới của lớp UDPClient. Thuộc một dòng địa chỉ (AddressFamily) được chỉ định.
UdpClient (LocalPort: Int32)	Tạo một **UdpClient** và gắn (bind) một cổng cho nó.
Active	Lấy về hoặc thiết lập giá trị cho biết kết nối với máy ở xa đã được tạo ra chưa. Kiểu dữ liệu là bool
Client	Lấy về hoặc thiết lập các socket

Lớp UDPClient: các thành viên

Tên phương thức	Mô tả
UdpClient (IPEndPoint)	Tạo một **UdpClient** và gắn (bind) một IPEndpoint (gán địa chỉ IP và cổng) cho nó.
UdpClient (Int32, AddressFamily)	Tạo một **UdpClient** và gán số hiệu cổng, AddressFamily
UdpClient (Remotehost: String, Int32)	Tạo một **UdpClient** và thiết lập với một trạm từ xa mặc định.
UdpClient (IPEndPoint)	Tạo một **UdpClient** và gắn (bind) một IPEndpoint (gán địa chỉ IP và cổng) cho nó.

Lớp UDPClient: các thành viên

Tên phương thức	Mô tả
BeginReceive	Nhận dữ liệu không đồng bộ từ máy ở xa.
BeginSend	Gửi không đồng bộ dữ liệu tới máy ở xa
Close	Đóng kết nối.
Connect	Thiết lập một Default remote host.
EndReceive	Kết thúc nhận dữ liệu không đồng bộ ở trên
EndSend	Kết thúc việc gửi dữ liệu không đồng bộ ở trên
Receive (EndPoint của máy ở xa) As Byte()	Nhận dữ liệu (đồng bộ) do máy ở xa gửi. (Đồng bộ có nghĩa là các lệnh ngay sau lệnh Receive chỉ được thực thi nếu Receive đã nhận được dữ liệu về, còn nếu nó chưa nhận được thì nó vẫn cứ chờ (blocking))
Send	Gửi dữ liệu (đồng bộ) cho máy ở xa.

Lớp UDPClient: chương trình chat

- Khởi tạo ứng dụng VC#, tạo project mới, có 1 form, 4 textbox với tên: txtIpAddress, txtLocalPort, txtPort, txtSend; 2 listbox tên lstReceived, lstSent và 2 nút lệnh btnConnect, btnSend
- Khai báo một số thông tin như sau:

Lớp UDPClient: chương trình chat

```
string _localPort = "10";
string _remotePort = "1000";
UdpClient _applications = new UdpClient();
Thread _thread;
bool _exit = false;
delegate void
ClearCacheReceivedData(string Data, string
RemoteHost);
```

Lớp UDPClient: chương trình chat

```
private void OnbtSendClick(object sender, EventArgs e)
        {
            IPAddress ip;
            if (!IPAddress.TryParse(txtIpAddress.Text, out ip))
                MessageBox.Show("Hãy nhập chính xác IP của người nhận",
"Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);
            else
            {
                SentData();
                lstSent.Items.Insert(0, txtSend.Text);
                txtSend.Clear();
            }
        }
```

Lớp UDPClient: chương trình chat

```
private void ReceivedData(string Data, string RemoteHost)
        {
            if (lstReceived.InvokeRequired)
            {
                ClearCacheReceivedData clearCacheReceivedData = new
ClearCacheReceivedData(ReceivedData);
                lstReceived.Invoke(clearCacheReceivedData, new object[] {
Data, RemoteHost });
                return;
            }
            string msg = "";
            msg = "(Người gửi: " + RemoteHost + ")" + Data;
            lstReceived.Items.Insert(0, msg);
        }
```

Lớp UDPClient: chương trình chat

```
private void SentData()
        {
            byte[] msg;
            msg = System.Text.Encoding.UTF8.GetBytes(txtSend.Text);
            _applications.Send(msg, msg.Length, txtIpAddress.Text,
int.Parse(_remotePort));
        }
```

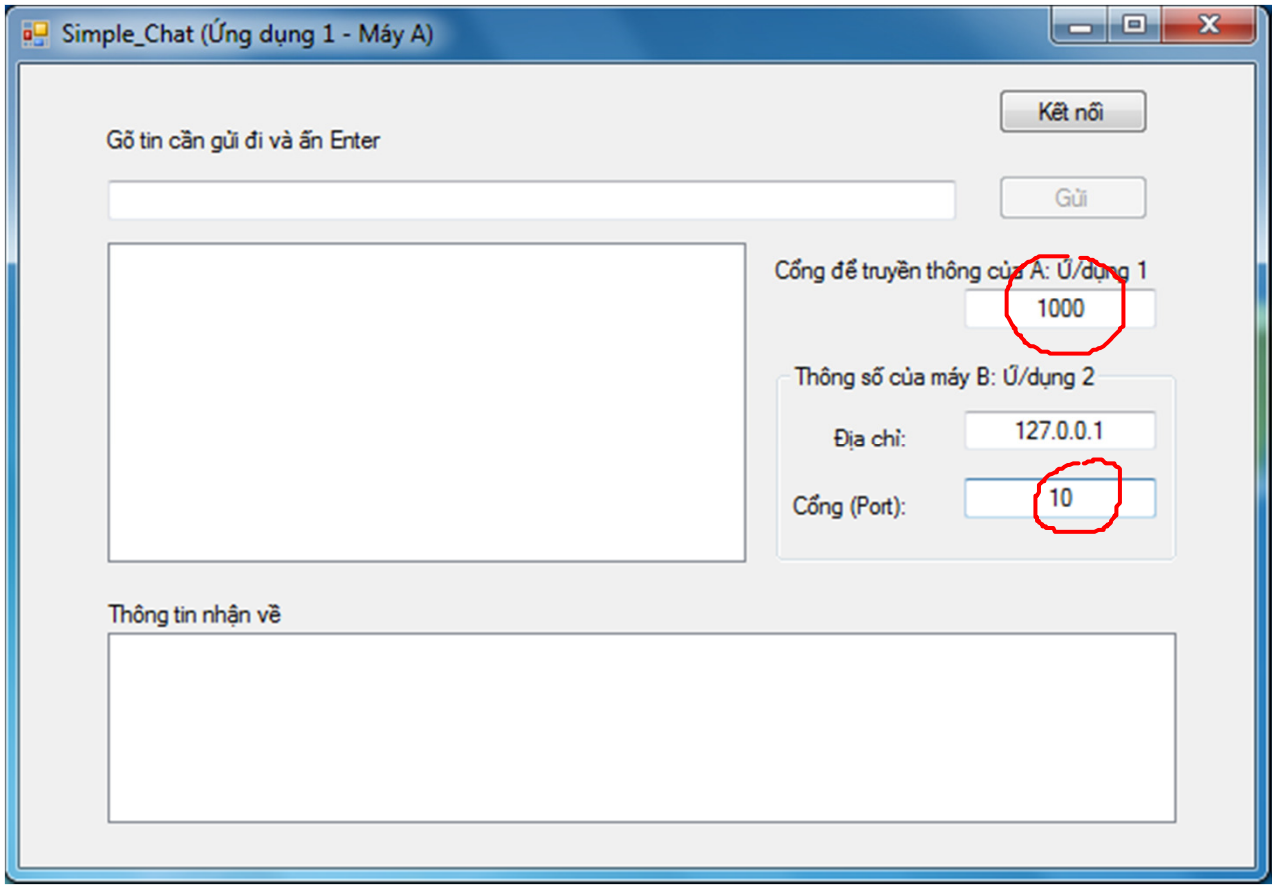
Lớp UDPClient: chương trình chat

```
private void Explore()
    {
        IPAddress ip;
        byte[] msg;
        string str = "";
        ip = Dns.GetHostEntry(_remotePort).AddressList[0];
        IPEndPoint ep = new IPEndPoint(ip, Convert.ToInt16(_remotePort));
        while (_exit == false)
        {
            Application.DoEvents();
            if (_applications.Available > 0)
            {
                msg = _applications.Receive(ref ep);
                str = System.Text.Encoding.UTF8.GetString(msg);
                ReceivedData(str, ep.Address.ToString());
            }
        }
    }
```

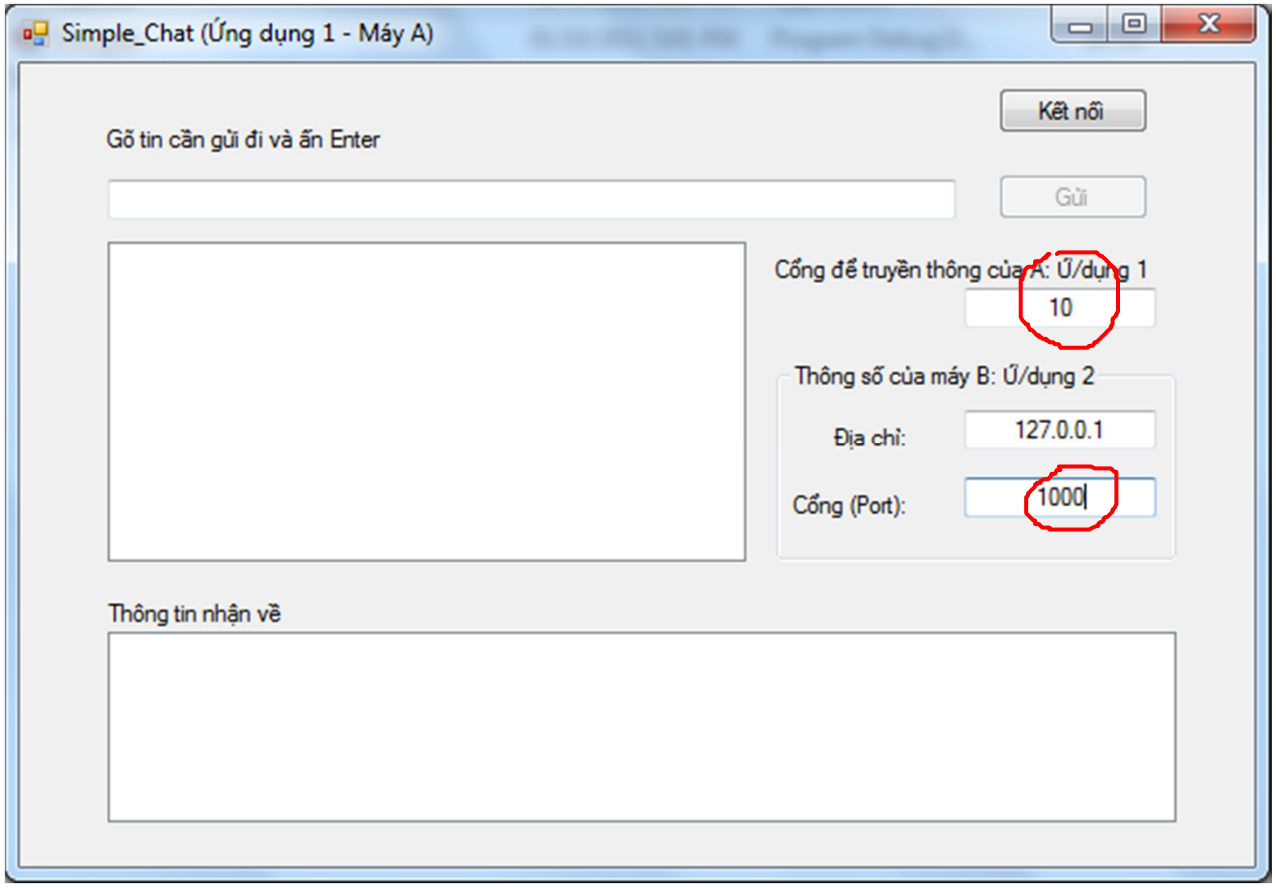
Lớp UDPClient: chương trình chat

```
private void btnConnect_Click(object sender, EventArgs e)
        {
            _localPort = this.txtLocalPort.Text;
            _remotePort = this.txtPort.Text;
            _applications = new UdpClient(int.Parse(_localPort));
            _thread = new Thread(Explore);
            _thread.Start();
            this.btSend.Click += OnbtSendClick;
            this.btSend.Enabled = true;
            this.btnConnect.Enabled = false;
            txtIpAddress.ReadOnly = txtLocalPort.ReadOnly =
txtPort.ReadOnly = true;
        }
```

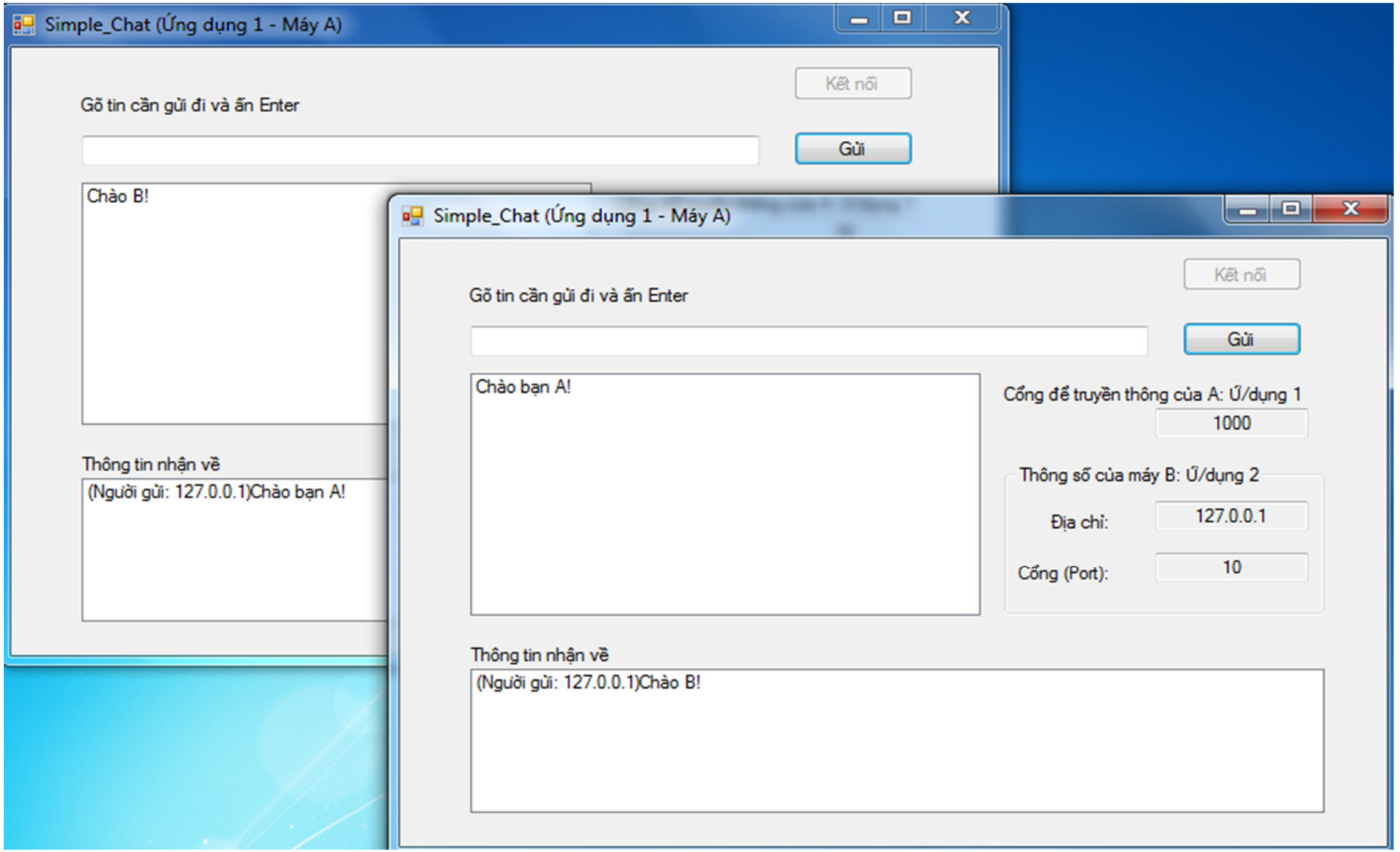
Lớp UDPClient: chương trình chat



Lớp UDPClient: chương trình chat



Lớp UDPClient: chương trình chat



Lớp UDPClient: tổng kết

- Khi muốn gửi dữ liệu qua mạng bằng lớp UDPClient, chúng ta theo cách đơn giản nhất như sau:
  - Tạo một UDPClient và gán cho nó một số hiệu cổng. Ví dụ: UDPClient udp = new UDPClient(1000)
  - Tạo một địa chỉ IP ứng với địa chỉ của máy mà ta muốn giao tiếp bằng IPEndPoint hoặc IPAddress hoặc IPHostEntry. (Lưu ý: Nếu dùng DNS.GetHostEntry thì ta có thể truyền vào là tên của máy. Sau đó muốn lấy địa chỉ thì: DNS.GetHostEntry("Tên_Máy").Address[0])

Lớp UDPClient: tổng kết

- Gửi dữ liệu đi:
  - Bước 1: Chuyển chuỗi thành mảng byte
  - Bước 2: Gọi phương thức Send, trong đó truyền địa chỉ IP của máy ở xa mà ta vừa tạo ở 2 và thêm vào số hiệu cổng mà máy ở xa đang dùng để nhận dữ liệu.
- Khi nhận: Dùng phương thức Receive để nhận dữ liệu về. Khi đó chúng ta cần tạo một đối tượng IPEndPoint với địa chỉ và số hiệu cổng của máy ở xa mà chúng ta muốn nhận dữ liệu. Phương thức này trả về dữ liệu ở dạng mảng byte, do vậy để chuyển sang dạng chuỗi ký tự thì cần dùng lớp Encoding để chuyển đổi.

Lớp TCPClient

- Để đảm bảo độ tin cậy trong các ứng dụng mạng, người ta còn dùng một giao thức khác, gọi là giao thức có kết nối: TCP (Transport Control Protocol). Trên Internet chủ yếu là dùng loại giao thức này, ví dụ như Telnet, HTTP, SMTP, POP3… Để lập trình theo giao thức TCP, .NET cung cấp hai lớp có tên là TCPClient và TCPListener.

Lớp TCPClient: các thành viên

Tên phương thức	Mô tả
TcpClient ()	Tạo một đối tượng **TcpClient**.
TcpClient (IPEndPoint)	Tạo một **TcpClient** và gắn cho nó một EndPoint cục bộ. (Gán địa chỉ máy cục bộ và số hiệu cổng để sử dụng trao đổi thông tin về sau)
TcpClient (RemoteHost: String, RemotePort: Int32)	Tạo một đối tượng **TcpClient** và kết nối đến một máy có địa chỉ và số hiệu cổng được truyền vào. RemoteHost có thể là địa chỉ IP chuẩn hoặc tên máy.

Lớp TCPClient: các thành viên

Tên thuộc tính	Mô tả
Available	Cho biết số byte đó nhận về từ mạng và có sẵn để đọc.
Client	Trả về Socket ứng với TCPClient hiện hành.
Connected	Trạng thái cho biết đã kết nối được đến Server hay chưa

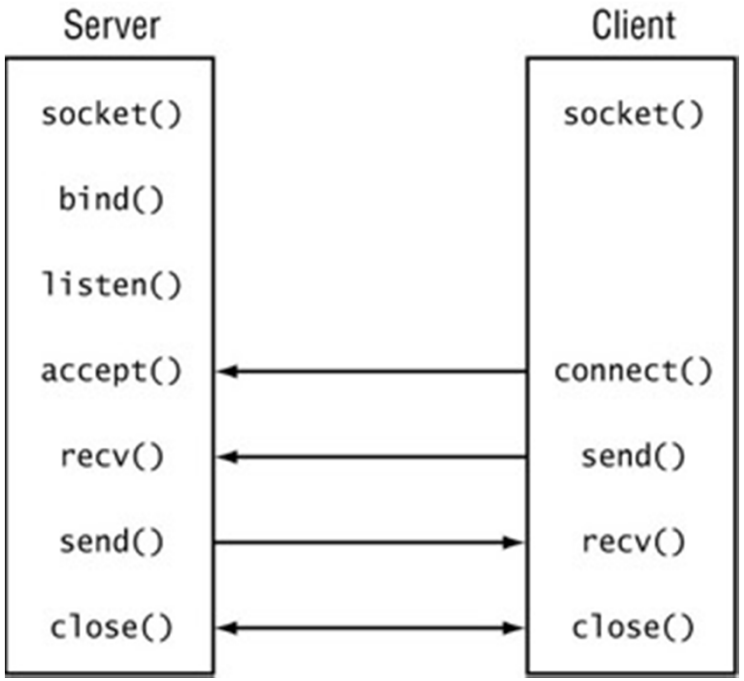
Lớp TCPClient: các thành viên

Tên phương thức	Mô tả
Close	Giải phóng đối tượng **TcpClient** nhưng không đóng kết nối.
Connect (RemoteHost, Port)	Kết nối đến một máy TCP khác có tên và số hiệu cổng.
GetStream	Trả về NetworkStream để từ đó giúp ta gửi hay nhận dữ liệu. (Thường làm tham số khi tạo StreamReader và StreamWriter) . Khi đó gắn vào StreamReader và StreamWriter rồi ta có thể gửi và nhận dữ liệu thông qua các phương thức ReadIn, WriteIn tương ứng của các lớp này.

Lớp TCPClient: trình tự kết nối

- Bước 1: Tạo một đối tượng TCPClient
- Bước 2: Kết nối đến máy chủ (Server) dùng phương thức Connect
- Bước 3: Tạo 2 đối tượng StreamReader (Receive) và StreamWriter (Send) và **"nối"** với GetStream của TCPClient
- Bước 4:
  - Dùng đối tượng StreamWriter.Writeline/Write vừa tạo ở trên để gửi dữ liệu đi.
  - Dùng đối tượng StreamReader.Readline/Read vừa tạo ở trên để đọc dữ liệu về.
- Bước 5: Đóng kết nối.

Lớp TCPClient: trình tự kết nối



Lớp TCPClient

- Nếu muốn gửi/nhận dữ liệu ở mức byte (nhị phân) thì dựng NetworkStream (truyền GetStream cho NetworkStream)

So sánh TCP & UDP

TCP	UDP
Hoạt động tin cậy	Hoạt động không tin cậy
Phải thiết lập kết nối giữa client & server	Không cần thiết lập kết nối
Dữ liệu gửi không chứa địa chỉ và port của máy nhận	Phải xác định địa chỉ và port của máy nhận trong dữ liệu gửi

Chương trình chat dùng TCPClient

```
class Connection
{
        TcpClient tcpClient;
        private Thread thrSender;
        private StreamReader srReceiver;
        private StreamWriter swSender;
        private string currUser;
        private string strResponse;
        public Connection(TcpClient tcpCon)  {
            tcpClient = tcpCon;
            thrSender = new Thread(AcceptClient);
            thrSender.Start();
       }
```

Chương trình chat dùng TCPClient

```
private void CloseConnection()
        {
            tcpClient.Close();
            srReceiver.Close();
            swSender.Close();
        }
```

Chương trình chat dùng TCPClient

```
private void AcceptClient()
{
        srReceiver = new
System.IO.StreamReader(tcpClient.GetStream());
        swSender = new
System.IO.StreamWriter(tcpClient.GetStream());
        currUser = srReceiver.ReadLine();
        if (currUser != "")   {
            if (ChatServer.htUsers.Contains(currUser) == true)   {
                swSender.WriteLine("0|This username already exists.");
                swSender.Flush();
                CloseConnection();
                return;
            }
```

Chương trình chat dùng TCPClient

```
        else if (currUser == "Administrator") {
                swSender.WriteLine("0|This username is
reserved.");
                swSender.Flush();
                CloseConnection();
                return;
            }
            else  {
                swSender.WriteLine("1");
                swSender.Flush();
                ChatServer.AddUser(tcpClient, currUser);
            }
```

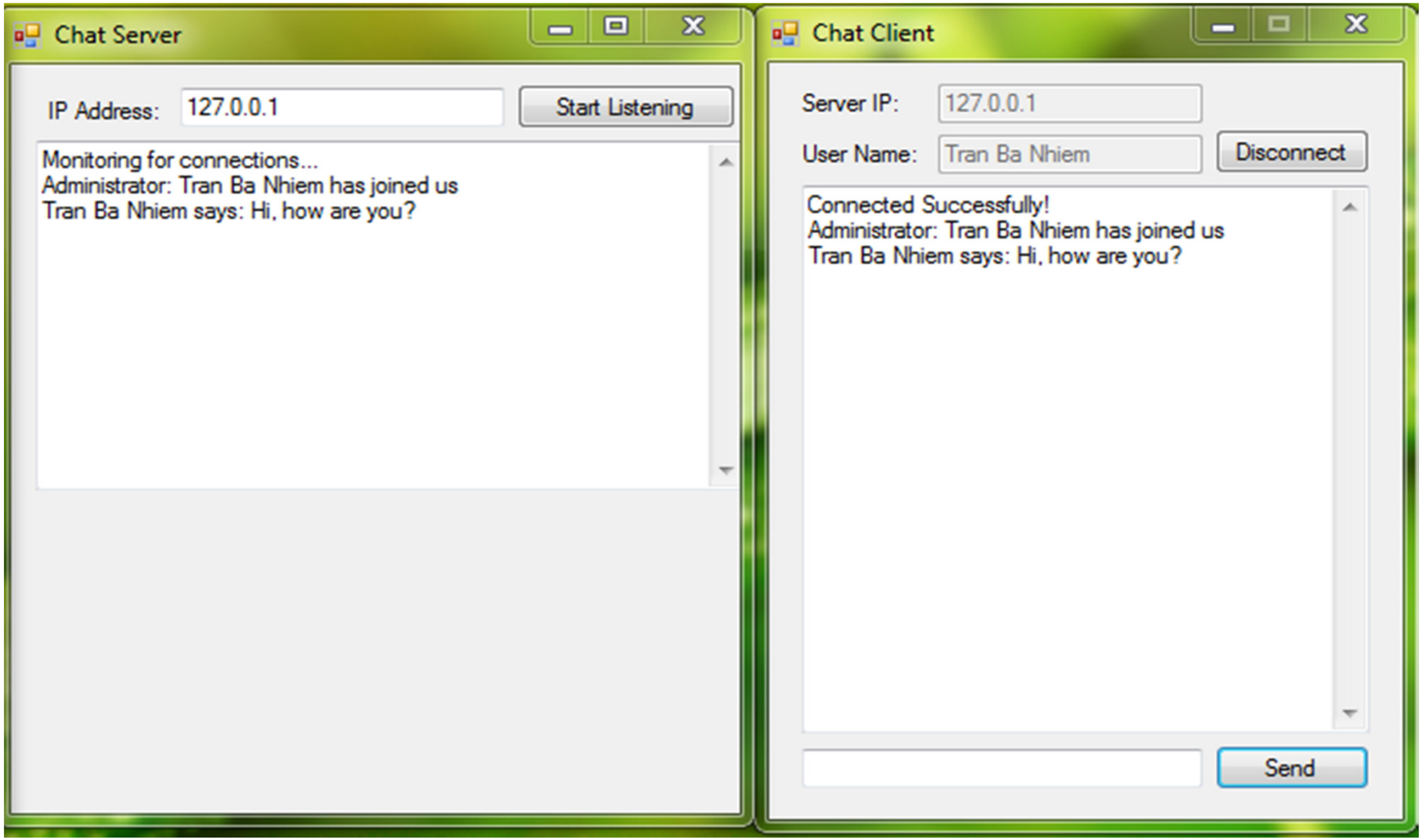
Chương trình chat dùng TCPClient

```
            }
            else {
                CloseConnection();
                return;
            }
            try {
                while ((strResponse =
srReceiver.ReadLine()) != "")  {
                    if (strResponse == null) {
                        ChatServer.RemoveUser(tcpClient);
                    }
```

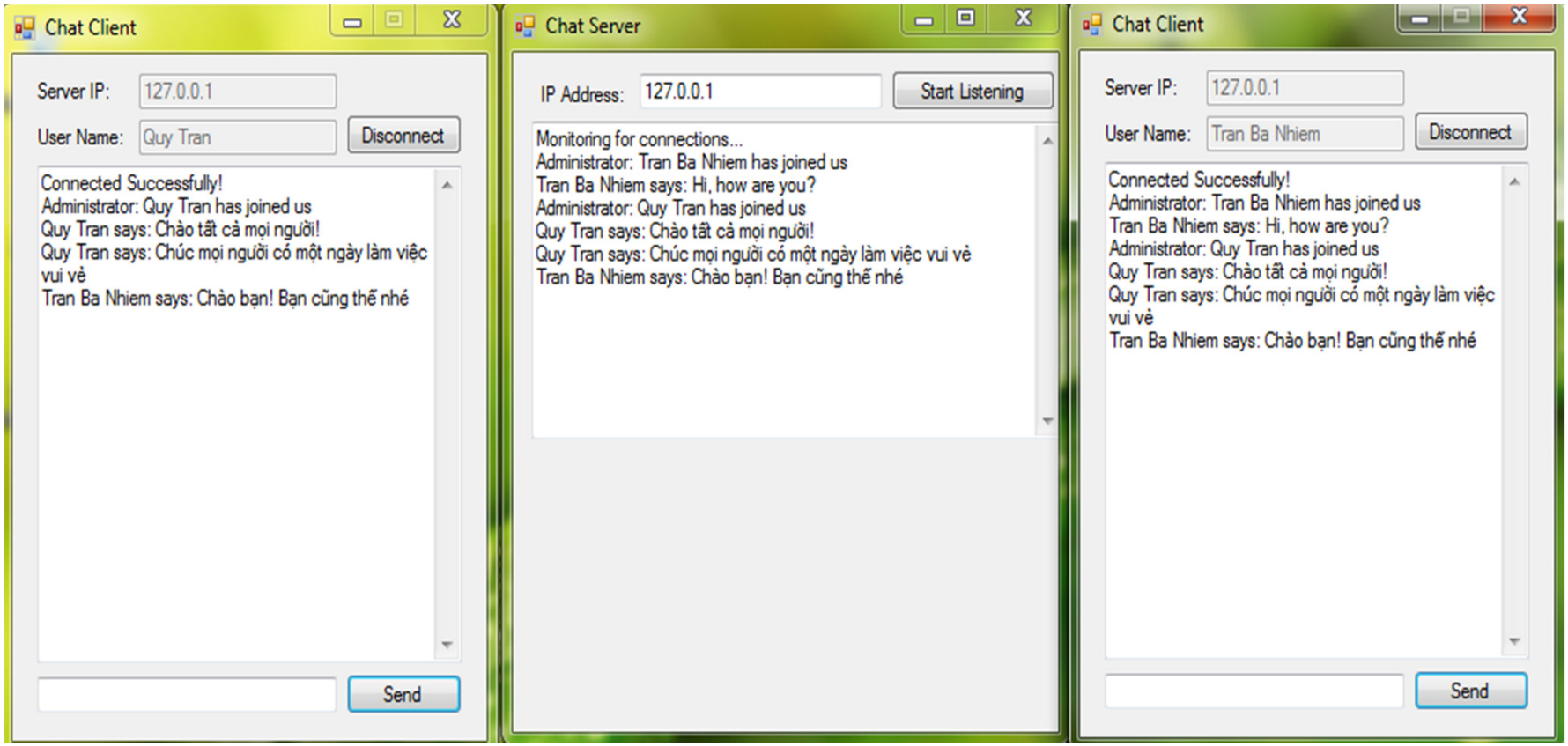
Chương trình chat dùng TCPClient

```
            else {
                ChatServer.SendMessage(currUser,
strResponse);
                }
            }
        }
        catch {
            ChatServer.RemoveUser(tcpClient);
        }
    }
```

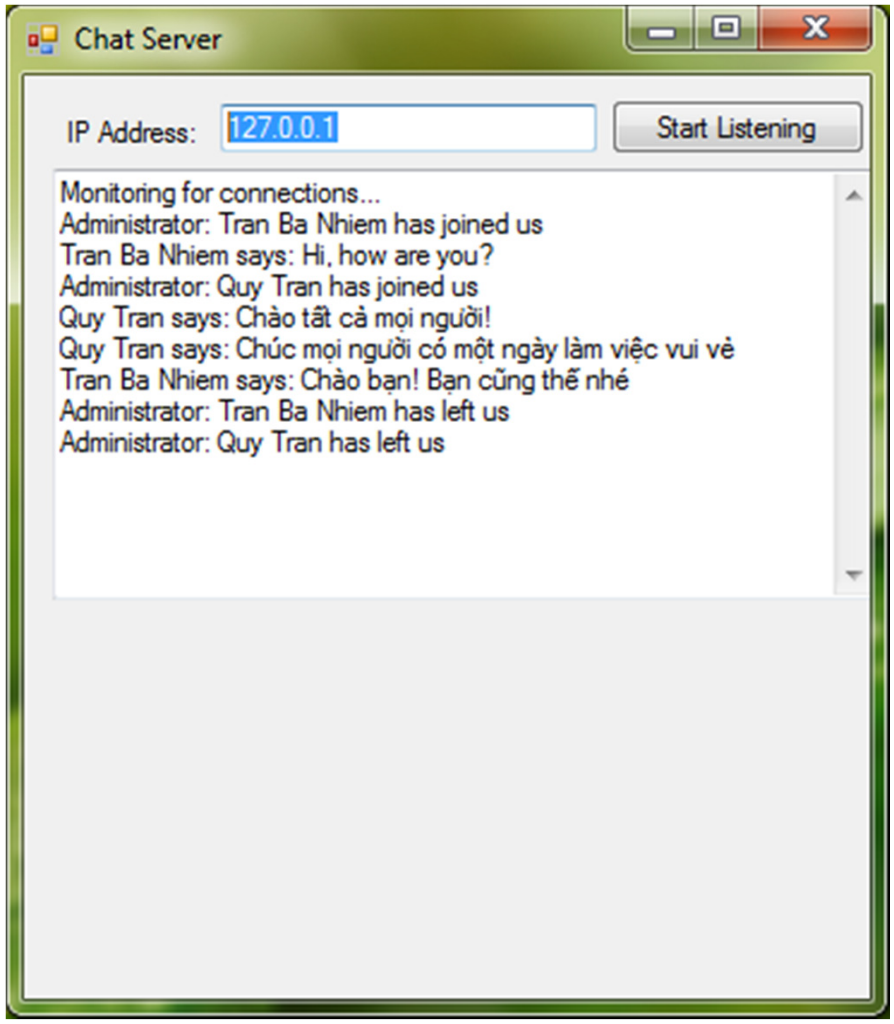
Lớp TCPClient: chương trình chat



Lớp TCPClient: chương trình chat



Lớp TCPClient: chương trình chat



Lớp TCPClient: ví dụ

- Tạo một TCP Client và kết nối đến server (FTP Server), sau đó gửi 1 chuỗi

```
using System.Net.Sockets;
using System.Net;
using System.IO;
public class Form1 {
    // Tạo địa chỉ ứng với 127.0.0.1
    Long DiaChi = 1 * 256 ^ 3 + 127 * 256 ^ 0
```

Lớp TCPClient: ví dụ

```
        // Tạo một IPEndPoint từ địa chỉ IP và cổng
(TCPClient cần một IPEndPoint)
        IPEndPoint LocalEP = new IPEndPoint(DiaChi,
1000);  '// cho cục bộ (client)
        // Tạo một đối tượng TCP ứng với địa chỉ và cổng ở
trên
        TcpClient tcp = new TcpClient(LocalEP);
        // Hai luồng nhập và xuất dùng để đọc/ghi vào kết
nối TCP
        StreamWriter Ghi;
        StreamReader Doc;
```

Lớp TCPClient: ví dụ

```
private void Form1_Load(…) {
        tcp.Connect("localhost", 21);
        MessageBox.Show(tcp.Connected)
        Doc = new StreamReader(tcp.GetStream());
        Ghi = new StreamWriter(tcp.GetStream());
        //Gửi thử một chuỗi cho server (FTP Server)
        Ghi.Writeline("User nhiemtb");
        Ghi.Flush();
        // Đọc dữ liệu do Server gửi về
        String S = Doc.ReadLine();
        MessageBox.Show("Dữ liệu gửi từ server : " + S);
}
```

Lớp TCPClient: ví dụ

```
    private void Gui_Du_Lieu(String Data)
    {
        Ghi.WriteLine(Data);
        Ghi.Flush();
    }
}
```

Nhận xét

- Ở ví dụ trên ta thấy rằng việc gửi thì có thể thực hiện nhiều lần với việc gọi nhiều lần phương thức Gửi_Dữ_Liệu. Tuy nhiên, đối với việc nhận dữ liệu thì ta chỉ thực hiện một lần. Trong trường hợp nếu ta muốn nhận dữ liệu bất cứ khi nào có dữ liệu về thì cần áp dụng kỹ thuật "Thăm dò" và “Kích hoạt sự kiện" như trong phần UDPClient.

Nhận xét

- Ý tưởng thực hiện như sau:
  - Bước 1: Tạo một TCPClient
  - Bước 2: Kết nối
  - Bước 3: Tạo một luồng mới, luồng này "chuyên theo dõi" xem có dữ liệu mới về hay không (chỉ việc kiểm tra bộ đệm (đối tượng StreamReader.EndOfStream = True/False). Nếu bộ đệm không rỗng (có dữ liệu mới) thì giá trị EndOfStream sẽ bằng False. Khi có dữ liệu trong bộ đệm ta kích hoạt (Raise) sự kiện Có_Dữ_Liệu lên. Trong sự kiện này ta sẽ viết các lệnh xử lý.

Viết chương trình Telnet

```
Imports System.Net.Sockets
Imports System.Net
Imports System.IO
Imports System.Threading
Public Class frmTelnet
        '/// Tạo một đối tượng TCPClient
        Dim tcp As New TcpClient()
        '/// Hai luồng nhập và xuất dùng để ghi vào kết nối TCP
```

Viết chương trình Telnet

```
        Dim Ghi As StreamWriter
        Dim Doc As StreamReader
        '/// Tạo một thread chuyên thăm dò dữ
liệu
        Dim Th As Thread
        '/// Cờ báo hiệu khi thoát. Để tránh việc
lặp vô hạn
        Dim Thoat As Boolean = False
        Public Event Dữ_Liệu_Về(ByVal Data As
String)
```

Viết chương trình Telnet

```
Sub Thăm_Dò()
        Dim S As String
        Do While Thoat = False
                Application.DoEvents()
                If Doc.EndOfStream = False Then
                        S = Doc.ReadLine
                        RaiseEvent Dữ_Liệu_Về(S)
                End If
        Loop
End Sub
```

Viết chương trình Telnet

```
Private Sub frmTelnet_Dữ_Liệu_Về(ByVal Data As String)
Handles Me.Dữ_Liệu_Về
        lstreceived.Items.Insert(0, Data)
End Sub

Private Sub Form1_Load(ByVal sender As Object, ByVal e
As System.EventArgs) Handles Me.Load
        Dim RPort As Long =
Integer.Parse(txtRemotePort.Text)
        Dim IpEnd As New
IPEndPoint(IPAddress.Parse(txtRemoteHost.Text), RPort)
```

Viết chương trình Telnet

```
        '/// Kết nối tới máy chủ
        tcp.Connect(IpEnd)
        Doc = New StreamReader(tcp.GetStream())
        Ghi = New StreamWriter(tcp.GetStream())
        Th = New Thread(AddressOf Thăm_Dò)
        Th.Start()
End Sub
```

Viết chương trình Telnet

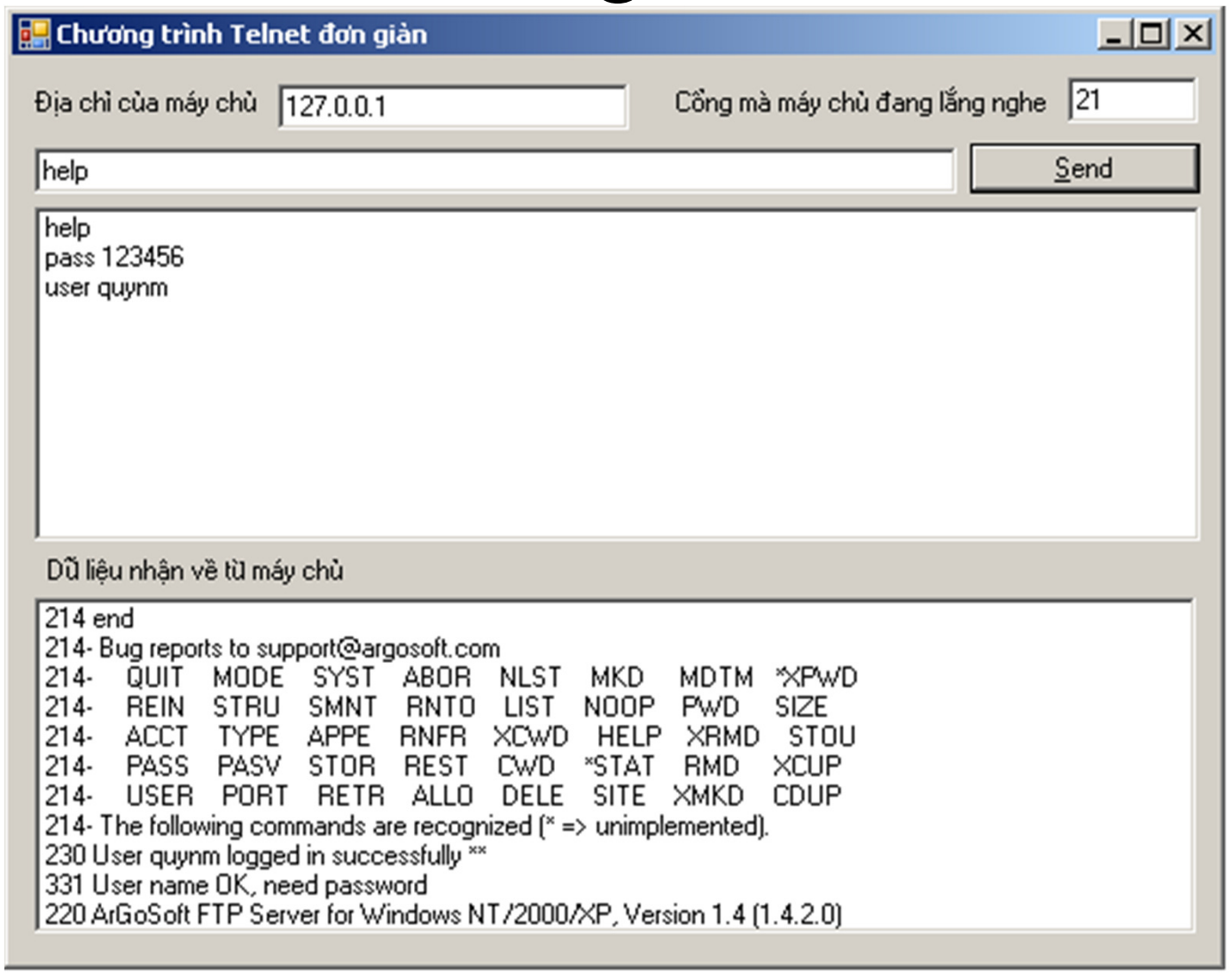
```
        Private Sub cmdSend_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
cmdSend.Click
                Gui_Du_Lieu(txtMsg.Text)
        End Sub


        Private Sub frmTelnet_FormClosing(ByVal sender
As Object, ByVal e As
        System.Windows.Forms.FormClosingEventArgs)
Handles Me.FormClosing
                Thoat = True
        End Sub
```

Viết chương trình Telnet

```
        Private Sub Gui_Du_Lieu(ByVal Data
As String)
                Ghi.WriteLine(Data)
                Ghi.Flush()
                lstSent.Items.Insert(0,
txtMsg.Text)
        End Sub
End Class
```

Viết chương trình Telnet



Lớp TCPListener

- TCPListerner là một lớp cho phép người lập trình có thể xây dựng các ứng dụng Server (Ví dụ: như SMTP Server, FTP Server, DNS Server, POP3 Server hay server tự định nghĩa ….). Ứng dụng server khác với ứng dụng Client ở chỗ nó luôn luôn thực hiện lắng nghe và chấp nhận các kết nối đến từ Client

Lớp TCPClient: các thành viên

Tên phương thức	Mô tả
TcpListener (Port: Int32)	Tạo một **TcpListener** và lắng nghe tại cổng chỉ định.
TcpListener (IPEndPoint)	Tạo một **TcpListener** với giá trị Endpoint truyền vào.
TcpListener (IPAddress, Port: Int32)	Tạo một **TcpListener** và lắng nghe các kết nối đến tại địa chỉ IP và cổng chỉ định.
Active	Trả về một giá trị cho biết TcpListener đang lắng nghe kết nối từ client. Kiểu bool
Server	Trả về socket

Lớp TCPClient: các thành viên

Tên phương thức	Mô tả
AcceptSocket	Chấp nhận một yêu cầu kết nối đang chờ.
AcceptTcpClient	Chấp nhận một yêu cầu kết nối đang chờ. (Ứng dụng sẽ dừng tại lệnh này cho đến khi nào có một kết nối đến)
Pending	Cho biết liệu có kết nối nào đang chờ đợi không? (True = có).
Start	Bắt đầu lắng nghe các yêu cầu kết nối.
Stop	Dừng việc nghe.

Lớp TCPClient: ví dụ xây dựng một ứng dụng Server đơn giản

```
Imports System.Net.Sockets
Imports System.Net
Imports System.IO
Imports System.Threading
Public Class frmServer
        Dim TCPServer As New System.Net.Sockets.TcpListener(21)
        Dim Thoat As Boolean = False
        Dim Clients(100) As TcpClient
        Dim CurrClient As Integer = 0
        Sub Xử_Lý_Kết_Nối()
                Dim LastClient As Integer = CurrClient - 1
                Dim Con As TcpClient = Clients(LastClient)
                Dim Doc As New StreamReader(Con.GetStream)
                Dim Ghi As New StreamWriter(Con.GetStream)
                Dim S As String
```

Lớp TCPClient: ví dụ xây dựng một ứng dụng Server đơn giản

```
        While Thoat = False
                Application.DoEvents()
                If Doc.EndOfStream = False Then
                        S = Doc.ReadLine
                        '// Xử lý tại đây:
                        S = S.ToUpper
                        Ghi.WriteLine(S) '//Gửi lại Client...
                        Ghi.Flush()
                End If
        End While
End Sub
```

Lớp TCPClient: ví dụ xây dựng một ứng dụng Server đơn giản

```
Sub Nghe_Kết_Nối()
        Do While Thoat = False
                Clients(CurrClient) =
TCPServer.AcceptTcpClient()
                CurrClient += 1
                'MsgBox ("Đã có " & (CurrClient + 1) & " kết
nối !")
                Dim Th As New Thread(AddressOf
Xử_Lý_Kết_Nối)
                Th.Start()
        Loop
End Sub
```

Lớp TCPClient: ví dụ xây dựng một ứng dụng Server đơn giản

```
    Private Sub frmClose(ByVal s As Object, ByVal e As
    FormClosingEventArgs) Handles Me.FormClosing
            Thoat = True
    End Sub

    Private Sub Form1_Load(ByVal s As Object, ByVal
    e As EventArgs) Handles Me.Load
            TCPServer.Start()
            Nghe_Kết_Nối()
    End Sub
End Class
```

Dùng TCP/IP để truyền file

- Để đọc file, tạo một stream tương ứng
- Đọc nội dung file vào buffer array
- Gửi buffer array trên đường truyền

```
Stream fileStream = File.OpenRead(tbFilename.Text);
// Alocate memory space for the file
byte[] fileBuffer = new byte[fileStream.Length];
fileStream.Read(fileBuffer, 0, (int)fileStream.Length);
 // Open a TCP/IP Connection and send the data
TcpClient clientSocket = new TcpClient(tbServer.Text,8080);
NetworkStream networkStream =  clientSocket.GetStream();
networkStream.Write(fileBuffer,0,fileBuffer.GetLength(0));
networkStream.Close();
```

Debugging network code

- Debugging là một phương pháp quan trọng để theo dõi và giải quyết các lỗi phát sinh trong khi viết chương trình
- Nên dùng cấu trúc try/catch

```
try {
 serverSocket.Bind(ipepServer);
 serverSocket.Listen(-1);
}
catch(SocketException e) {
 MessageBox.Show(e.Message);
}
```

Debugging network code

- Để định vị các trục trặc trong các ứng dụng multithreaded, cơ chế theo vết (tracing) đóng vai trò cực kỳ quan trọng
- Nên dùng System.Diagnostics.Trace
- Hoặc các phát biểu dạng Console.WriteLine

Bài tập

1. Viết chương trình UDP đặt ở hai máy thực hiện công việc sau: Khi một ứng dụng gửi chuỗi "OPEN#<Đường dẫn >" thì ứng dụng trên máy kia sẽ mở file nằm trong phần <đường dẫn>. Khi một ứng dụng gửi chuỗi "SHUTDOWN" thì ứng dụng kia sẽ tắt máy tính.

Bài tập

2. Viết chương trình UDP (ứng dụng A) đặt trên một máy thực hiện các công việc sau: Khi một ứng dụng (B) gửi một chuỗi chữ tiếng Anh thì ứng dụng A sẽ gửi trả lại nghĩa tiếng Việt tương ứng. Nếu từ tiếng Anh không có trong từ điển (từ điển ở đây chỉ có 3 từ Computer, RAM, HDD) thì ứng dụng A gửi trả lại chuỗi "Not found".

Bài tập

3. Viết chương trình SMTP server
4. Viết chương trình client/server trong đó, khi client di chuyển chuột thì server cũng di chuyển chuột theo. (dựng các hàm API về SetCursorPos…)
5. Viết chương trình Client/Server: Khi client gửi chuỗi "shutdown", "restart" thì Server sẽ tắt máy và khởi động tương ứng. (dựng hàm API ExitWindow…)

Bài tập

6. Viết chương trình kiểm tra xem máy 192.168.1.1 có dịch vụ FTP đang chạy hay không?
7. Viết chương trình kiểm tra xem máy "Servercntt" có dịch vụ FTP đang chạy hay không?
8. Viết chương trình Telnet ở trên linh hoạt hơn (có thể thay đổi tên máy, cổng…)
9. Viết chương trình Server giải đáp tên miền. Nếu máy khách gửi tên máy thì server sẽ gửi về địa chỉ IP. (danh sách này tự tạo ra – khoảng 3 cặp để minh họa).

Bài tập

10. Viết chương trình Client/Server. Khi Client gửi đường dẫn của tập tin nằm trên máy server thì server gửi trả cho Client nội dung của tập tin đó.
11. Cài đặt các chương trình đã minh họa trong bài giảng của chương bằng ngôn ngữ C# hoặc VB.NET