

CHƯƠNG 2: LẬP TRÌNH MẠNG TRONG .NET FRAMEWORK

2.1. Socket hướng kết nối (TCP Socket)

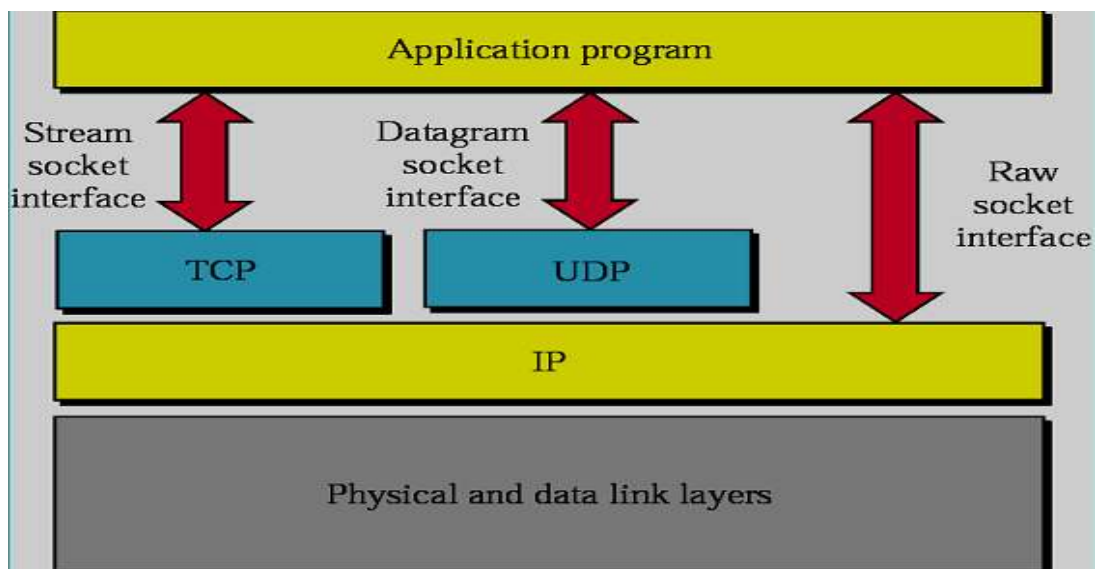
Socket là một giao diện lập trình ứng dụng (API) mạng

Thông qua giao diện này chúng ta có thể lập trình điều khiển việc truyền thông giữa hai máy sử dụng các giao thức mức thấp là TCP, UDP...

Socket là sự trừu tượng hoá ở mức cao, có thể tưởng tượng nó như là thiết bị truyền thông hai chiều gửi – nhận dữ liệu giữa hai máy tính với nhau.

■ Các loại Socket

- ☐ Socket hướng kết nối (TCP Socket)
- ☐ Socket không hướng kết nối (UDP Socket)
- ☐ Raw Socket



■ Đặc điểm của Socket hướng kết nối

- ☐ Có 1 đường kết nối ảo giữa 2 tiến trình
- ☐ Một trong 2 tiến trình phải đợi tiến trình kia yêu cầu kết nối.
- ☐ Có thể sử dụng để liên lạc theo mô hình Client/Server
- ☐ Trong mô hình Client/Server thì Server lắng nghe và chấp nhận một yêu cầu kết nối
- ☐ Mỗi thông điệp gửi đi đều có xác nhận trở về
- ☐ Các gói tin chuyển đi tuần tự

■ Đặc điểm của Socket không hướng kết nối

- ☐ Hai tiến trình liên lạc với nhau không kết nối trực tiếp
- ☐ Thông điệp gửi đi phải kèm theo địa chỉ của người nhận
- ☐ Thông điệp có thể gửi nhiều lần
- ☐ Người gửi không chắc chắn thông điệp tới tay người nhận
- ☐ Thông điệp gửi sau có thể đến đích trước thông điệp gửi trước đó.

■ Số hiệu cổng của Socket

- ❑ Để có thể thực hiện các cuộc giao tiếp, một trong hai quá trình phải công bố số hiệu cổng của socket mà mình sử dụng.
- ❑ Mỗi cổng giao tiếp thể hiện một địa chỉ xác định trong hệ thống. Khi quá trình được gán một số hiệu cổng, nó có thể nhận dữ liệu gửi đến cổng này từ các quá trình khác.
- ❑ Quá trình còn lại cũng yêu cầu tạo ra một socket.

2.1.1. Giới thiệu về *NameSpace System.Net* và *System.Net.Sockets*

- Cung cấp một giao diện lập trình đơn giản cho rất nhiều các giao thức mạng.
- Có rất nhiều lớp để lập trình
- Ta quan tâm lớp *IPAddress*, *EndPoint*, *DNS*, ...
- Lớp *IPAddress*
- Một số Field cần chú ý:
 - *Any*: Cung cấp một địa chỉ IP để chỉ ra rằng Server phải lắng nghe trên tất cả các Card mạng
 - *Broadcast*: Cung cấp một địa chỉ IP quảng bá
 - *Loopback*: Trả về một địa chỉ IP lặp
 - *AddressFamily*: Trả về họ địa chỉ của IP hiện hành
- Lớp *IPAddress*
- Một số phương thức cần chú ý:
 - Phương thức khởi tạo
 - [IPAddress\(Byte\[\]\)](#)
 - [IPAddress\(Int64\)](#)
 - *IsLoopback*: Cho biết địa chỉ có phải địa chỉ lặp không
 - *Parse*: Chuyển IP dạng chuỗi về IP chuẩn
 - *ToString*: Trả địa chỉ IP về dạng chuỗi
 - *TryParse*: Kiểm tra IP ở dạng chuỗi có hợp lệ không?
- Lớp *EndPoint*
- Một số phương thức cần chú ý:
 - Phương thức khởi tạo
 - *EndPoint (Int64, Int32)*
 - *EndPoint (IPAddress, Int32)*
 - [Create](#): Tạo một EndPoint từ một địa chỉ Socket
 - [ToString](#) : Trả về địa chỉ IP và số hiệu cổng theo khuôn dạng ĐịaChỉ: Cổng, ví dụ: 192.168.1.1:8080
- Lớp *DNS*
- Một số thành phần của lớp:
 - *HostName*: Cho biết tên của máy được phân giải
 - *GetHostAddress*: Trả về tất cả IP của một trạm
 - *GetHostEntry*: Giải đáp tên hoặc địa chỉ truyền vào và trả về đối tượng *IPHostEntry*

- GetHostName: Lấy về tên của máy tính cục bộ
- Namespace System.Net.Sockets
 - Một số lớp hay dùng: TcpClient, UdpClient, TcpListener, Socket, NetworkStream, ...
- Để tạo ra Socket
 - Socket(AddressFamily *af*, SocketType *st*, ProtocolType *pt*)

SocketType	Protocoltype	Description
Dgram	Udp	Connectionless communication
Stream	Tcp	Connection-oriented communication
Raw	Icmp	Internet Control Message Protocol
Raw	Raw	Plain IP packet communication

```

using System.Net;
using System.Net.Sockets;
class SockProp {
    public static void Main() {
        IPAddress ia = IPAddress.Parse("127.0.0.1");
        IPEndPoint ie = new IPEndPoint(ia, 8000);
        Socket test = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);
        Console.WriteLine("AddressFamily: {0}", test.AddressFamily);
        Console.WriteLine("SocketType: {0}", test.SocketType);
        Console.WriteLine("ProtocolType: {0}", test.ProtocolType);
        Console.WriteLine("Blocking: {0}", test.Blocking);
        test.Blocking = false;
        Console.WriteLine("new Blocking: {0}", test.Blocking);
        Console.WriteLine("Connected: {0}", test.Connected);
        test.Bind(ie);
        IPEndPoint iep = (IPEndPoint)test.LocalEndPoint;
        Console.WriteLine("Local EndPoint: {0}", iep.ToString());
        test.Close();
        Console.ReadKey();
    }
}

```

2.1.2. Viết chương trình cho phía máy chủ

- Viết chương trình cho phía máy chủ
 - ☐ Tạo một Socket
 - ☐ Liên kết với một IPEndPoint cục bộ
 - ☐ Lắng nghe kết nối
 - ☐ Chấp nhận kết nối
 - ☐ Gửi nhận dữ liệu theo giao thức đã thiết kế

❑ Đóng kết nối sau khi đã hoàn thành và trở lại trạng thái lắng nghe chờ kết nối mới

■ Viết chương trình cho phía máy chủ

```
IPEndPoint iep = new IPEndPoint(IPAddress.Any, 9050);
Socket newsock = Socket(AddressFamily.InterNetwork,
    SocketType.Stream, ProtocolType.Tcp);
newsock.Bind(iep);
newsock.Listen(10);
Socket client = newsock.Accept();
//Gửi nhận dữ liệu theo giao thức đã thiết kế
.....
newsock.Close();
```

Chương trình Server:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Net;
using System.Net.Sockets;
class Server {
    static void Main(string[] args) {
        IPEndPoint iep = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 2008);
        Socket server = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
        ProtocolType.Tcp);
        server.Bind(iep);
        server.Listen(10);
        Console.WriteLine("Cho ket noi tu client");
        Socket client = server.Accept();
        Console.WriteLine("Chap nhan ket noi tu: {0}",
        client.RemoteEndPoint.ToString());
        string s = "Chao ban den voi Server";
        //Chuyen chuoi s thanh mang byte
        byte[] data = new byte[1024];
        data = Encoding.ASCII.GetBytes(s);
        //gui nhan du lieu theo giao thuc da thiet ke
        client.Send(data,data.Length,SocketFlags.None);
        while (true) {
            data = new byte[1024];
            int recv = client.Receive(data);
            if (recv == 0) break;
            //Chuyen mang byte Data thanh chuoi va in ra man hinh
            s = Encoding.ASCII.GetString(data, 0, recv);
            Console.WriteLine("Clien gui len: {0}", s);
            //Neu chuoi nhan duoc la Quit thi thoat
            if (s.ToUpper().Equals("QUIT")) break;
            //Gui tra lai cho client chuoi s
            s = s.ToUpper();
            data = new byte[1024];
```

```

        data = Encoding.ASCII.GetBytes(s);
        client.Send(data, data.Length, SocketFlags.None);
    }
    client.Shutdown(SocketShutdown.Both);
    client.Close();
    server.Close();
}
}

```

2.1.3. *Viết chương trình cho phía máy khách*

■ Viết chương trình cho phía máy khách

- ☐ Xác định địa chỉ của Server
- ☐ Tạo Socket
- ☐ Kết nối đến Server
- ☐ Gửi nhận dữ liệu theo giao thức đã thiết kế
- ☐ Đóng Socket

■ Viết chương trình cho phía máy khách

```

IPEndPoint ipep = new IPEndPoint(IPaddress.Parse("192.168.1.6"), 9050);
Socket server = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
    ProtocolType.Tcp);

```

```
server.Connect(ipep);
```

Chương trình Client:

```

using System;
using System.Collections.Generic;
using System.Text;
using System.Net;
using System.Net.Sockets;

```

```

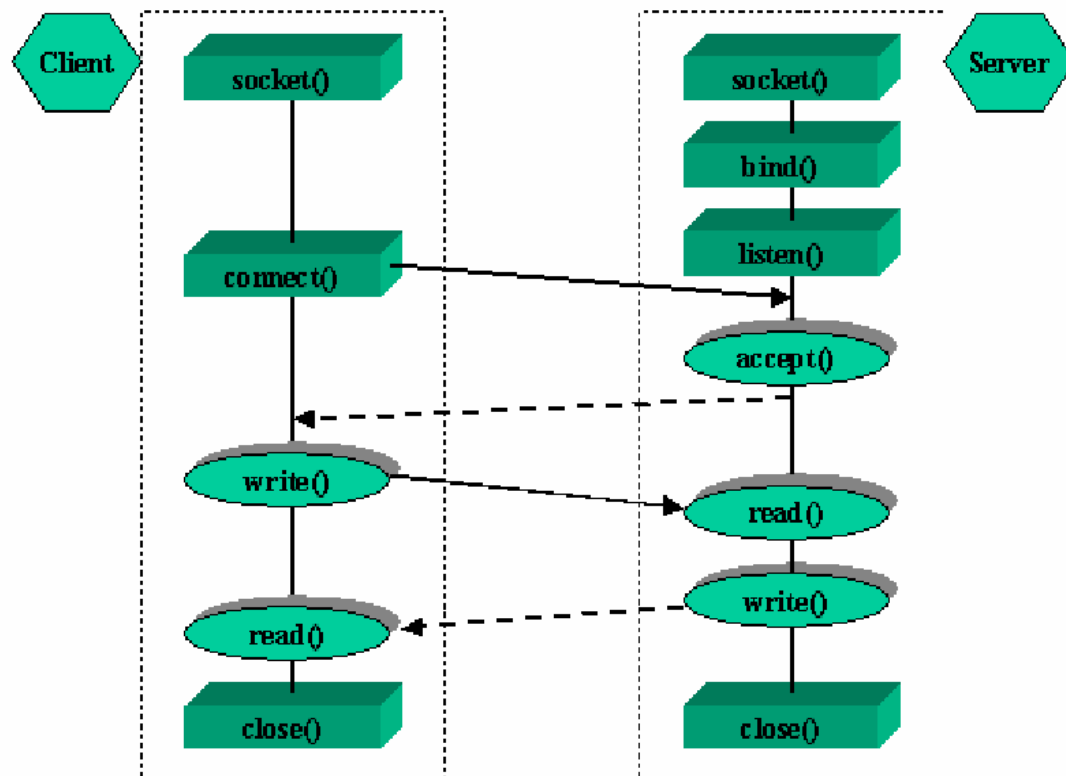
class Client {
    static void Main(string[] args) {
        IPEndPoint iep = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 2008);
        Socket client = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);
        client.Connect(iep);
        byte[] data = new byte[1024];
        int recv = client.Receive(data);
        string s = Encoding.ASCII.GetString(data, 0, recv);
        Console.WriteLine("Server gui: {0}", s);
        string input;
        while (true) {
            input = Console.ReadLine();
            //Chuyen input thanh mang byte gui len cho server
            data = new byte[1024];
            data = Encoding.ASCII.GetBytes(input);
            client.Send(data, data.Length, SocketFlags.None);
            if (input.ToUpper().Equals("QUIT")) break;
            data = new byte[1024];
            recv = client.Receive(data);

```

```

        s = Encoding.ASCII.GetString(data, 0, recv);
        Console.WriteLine("Server gui: {0}", s);
    }
    client.Disconnect(true);
    client.Close();
}
}

```



2.1.4. Sử dụng các luồng nhập xuất với Socket

Từ Socket ta có thể tạo ra luồng để nhập xuất với Socket đó là sử dụng lớp `NetworkStream`

Property	Description
<code>CanRead</code>	Is true if the <code>NetworkStream</code> supports reading
<code>CanSeek</code>	Is always false for <code>NetworkStreams</code>
<code>CanWrite</code>	Is true if the <code>NetworkStream</code> supports writing
<code>DataAvailable</code>	Is true if there is data available to be read

Ví dụ chương trình Client/Server sử dụng `NetworkStream` để gửi và nhận dữ liệu
Chương trình Client sử dụng `NetworkStream`:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Net.Sockets;

```

```

class Program {
    static void Main(string[] args) {
        IPEndPoint iep = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 2009);
        Socket client = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
        ProtocolType.Tcp);
        client.Connect(iep);
        NetworkStream ns = new NetworkStream(client);
        byte[] data = new byte[1024];
        while (true) {
            string input = Console.ReadLine();
            data = Encoding.ASCII.GetBytes(input);
            ns.Write(data, 0, data.Length);
            if (input.ToUpper().Equals("QUIT")) break;
            data = new byte[1024];
            int rec = ns.Read(data, 0, data.Length);
            string s = Encoding.ASCII.GetString(data, 0, rec);
            Console.WriteLine(s);
        }
        client.Close();
    }
}

```

Chương trình Server sử dụng NetworkStream:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Net.Sockets;
class Program {
    static void Main(string[] args) {
        IPEndPoint iep=new IPEndPoint(IPAddress.Parse("127.0.0.1"),2009);
        Socket server = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
        ProtocolType.Tcp);
        server.Bind(iep);
        server.Listen(10);
        Socket client = server.Accept();
        byte[] data;
        NetworkStream ns = new NetworkStream(client);
        while (true) {
            data = new byte[1024];
            int rec = ns.Read(data, 0, data.Length);
            string s = Encoding.ASCII.GetString(data, 0, rec);
            Console.WriteLine(s);
            data = new byte[1024];
            s = s.ToUpper();
            if (s.Equals("QUIT")) break;
            data = Encoding.ASCII.GetBytes(s);
            ns.Write(data, 0, data.Length);
        }
    }
}

```

```

        client.Close();
        server.Close();
    }
}

```

Trên cơ sở của `NetworkStream` ta có thể nối thêm các luồng để nhập xuất theo hướng ký tự như `StreamReader`, `StreamWriter`

Sau đây là một ví dụ về chương trình Client/Server sử dụng luồng nhập xuất, chương trình Server chép phép Client gửi lên yêu cầu, nếu yêu cầu là `GetDate` không phân biệt chữ hoa chữ thường thì Server trả về cho Client ngày hiện tại, nếu yêu cầu là `GetTime` không phân biệt hoa thường thì Server trả về giờ hiện tại, nếu là `Quit` thì Server ngắt kết nối với Client, không phải các trường hợp trên thì thông báo không hiểu lệnh.

Chương trình phía Client:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Net.Sockets;
using System.IO;
using System.Threading;
class DateTimeClient {
    static void Main(string[] args) {
        IPEndPoint iep = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 9999);
        Socket client = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
                                   ProtocolType.Tcp);

        client.Connect(iep);
        NetworkStream ns = new NetworkStream(client);
        StreamReader sr = new StreamReader(ns);
        StreamWriter sw = new StreamWriter(ns);
        while (true) {
            string input = Console.ReadLine();
            sw.WriteLine(input);
            sw.Flush();
            if (input.ToUpper().Equals("QUIT")) break;
            string kq = sr.ReadLine();
            Console.WriteLine(kq);
        }
        sr.Close();
        sw.Close();
        ns.Close();
        client.Close();
    }
}

```

Chương trình phía Server:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;

```



```

using System.Net.Sockets;
using System.IO;
class DateTimeServer {
    static void Main(string[] args) {
        IPEndPoint iep = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 2009);
        Socket server = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
                                   ProtocolType.Tcp);

        server.Bind(iep);
        server.Listen(10);
        Socket client = server.Accept();
        NetworkStream ns = new NetworkStream(client);
        StreamReader sr = new StreamReader(ns);
        StreamWriter sw = new StreamWriter(ns);
        string kq="";
        while (true) {
            string s = sr.ReadLine();
            s=s.ToUpper();
            if (s.Equals("QUIT")) break;
            if (s.Equals("GETDATE"))
                kq = DateTime.Now.ToString("dd/MM/yyyy");
            else
                if (s.Equals("GETTIME"))
                    kq = DateTime.Now.ToString("hh:mm:ss");
                else
                    kq = "Khong hieu lenh";
            sw.WriteLine(kq);
            sw.Flush();
        }
        sr.Close();
        sw.Close();
        client.Close();
    }
}

```

2.2. Socket không hướng kết nối (UDP Socket)

- Chương trình phía máy chủ
 - ☐ Tạo một Socket
 - ☐ Liên kết với một IPEndPoint cục bộ
 - ☐ Gửi nhận dữ liệu theo giao thức đã thiết kế
 - ☐ Đóng Socket
- Chương trình phía máy khách
 - ☐ Xác định địa chỉ Server
 - ☐ Tạo Socket
 - ☐ Gửi nhận dữ liệu theo giao thức đã thiết kế
 - ☐ Đóng Socket

2.2.1. Viết chương trình cho phía máy chủ

```
using System;
```

```

using System.Collections.Generic;
using System.Text;
using System.Net;
using System.Net.Sockets;
class Program {
    static void Main(string[] args) {
        IPEndPoint iep = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 2008);
        Socket server = new Socket(AddressFamily.InterNetwork, SocketType.Dgram,
ProtocolType.Udp);
        server.Bind(iep);
        //tao ra mot Endpot tu xa de nhan du lieu ve
        IPEndPoint RemoteEp = new IPEndPoint(IPAddress.Any, 0);
        EndPoint remote=(EndPoint)RemoteEp;
        byte[] data = new byte[1024];
        int recv = server.ReceiveFrom(data, ref remote);
        string s = Encoding.ASCII.GetString(data, 0, recv);
        Console.WriteLine("nhan ve tu Client: {0}", s);
        data = Encoding.ASCII.GetBytes("Chao client");
        server.SendTo(data, remote);
        while (true) {
            data=new byte[1024];
            recv = server.ReceiveFrom(data, ref remote);
            s = Encoding.ASCII.GetString(data, 0, recv);
            if (s.ToUpper().Equals("QUIT")) break;
            Console.WriteLine(s);
            data=new byte[1024];
            data=Encoding.ASCII.GetBytes(s);
            server.SendTo(data,0,data.Length,SocketFlags.None,remote);
        }
        server.Close();
    }
}

```

2.2.2. Viết chương trình cho phía máy khách

```

using System;
using System.Collections.Generic;
using System.Text;
using System.Net;
using System.Net.Sockets;

class Program {
    static void Main(string[] args) {
        IPEndPoint iep = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 2008);
        Socket client = new Socket(AddressFamily.InterNetwork, SocketType.Dgram,
ProtocolType.Udp);
        String s = "Chao server";
        byte[] data = new byte[1024];
        data = Encoding.ASCII.GetBytes(s);
        client.SendTo(data, iep);
        EndPoint remote = (EndPoint)iep;
    }
}

```

```

data = new byte[1024];
int recv = client.ReceiveFrom(data, ref remote);
s = Encoding.ASCII.GetString(data, 0, recv);
Console.WriteLine("Nhan ve tu Server{0}",s);
while (true) {
    s = Console.ReadLine();
    data=new byte[1024];
    data = Encoding.ASCII.GetBytes(s);
    client.SendTo(data, remote);
    if (s.ToUpper().Equals("QUIT")) break;
    data = new byte[1024];
    recv = client.ReceiveFrom(data, ref remote);
    s = Encoding.ASCII.GetString(data, 0, recv);
    Console.WriteLine(s);
}
client.Close();
}
}

```

Sử dụng Socket không hướng kết nối viết chương trình chat giữa 2 máy như sau: (Sau này chúng ta có thể sử dụng lớp UdpClient)

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;
using System.Threading;

public partial class Form1 : Form {
    private Socket udp1;
    private IPEndPoint ipremote, iplocal;
    public Form1() {
        InitializeComponent();
        CheckForIllegalCrossThreadCalls = false;
    }
    private void btStart_Click(object sender, EventArgs e) {
        udp1 = new Socket(AddressFamily.InterNetwork, SocketType.Dgram,
        ProtocolType.Udp);
        iplocal = new IPEndPoint(IPAddress.Parse("127.0.0.1"),
        int.Parse(txtLocalPort.Text));
        udp1.Bind(iplocal);
        ipremote = new IPEndPoint(IPAddress.Parse(txtIp.Text),
        int.Parse(txtRemotePort.Text));
        Thread tuyen = new Thread(new ThreadStart(NhanDL));
        tuyen.Start();
    }
}

```

```

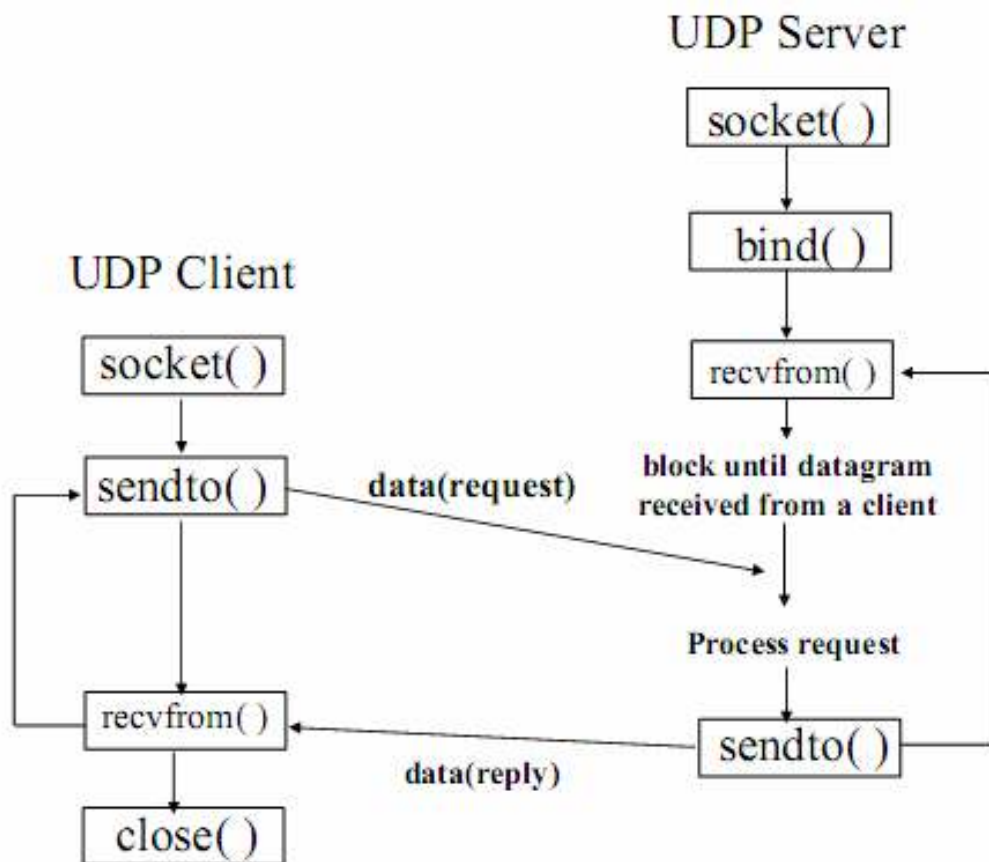
}

private void btSend_Click(object sender, EventArgs e) {
    byte[] data = new byte[1024];
    data = Encoding.ASCII.GetBytes(txtSend.Text);
    udp1.SendTo(data, ipremote);
}

private void NhanDL() {
    while (true) {
        byte[] data = new byte[1024];
        IPEndPoint ipe = new IPEndPoint(IPAddress.Any, 0);
        EndPoint remote = (EndPoint)ipe;
        int rec = udp1.ReceiveFrom(data, ref remote);
        string s = Encoding.ASCII.GetString(data, 0, rec);
        txtNoidung.Text += s + "\r\n";
    }
}

private void button1_Click(object sender, EventArgs e) {
    MessageBox.Show(txtSend.Text.Substring(0, txtSend.Text.IndexOf(" ")));
}
}

```



2.2.3. Sử dụng lớp *System.IO.MemoryStream* để tạo vùng đệm nhập xuất

2.3. Sử dụng các lớp hỗ trợ được xây dựng từ lớp Sôket

2.3.1. Lớp *TCPClient*




Mục đích của lớp *UDPClient* ở trên là dùng cho lập trình với giao thức UDP, với giao thức này thì hai bên không cần phải thiết lập kết nối trước khi gửi do vậy mức độ tin cậy không cao. 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, MS.NET cung cấp hai lớp có tên là *TCPClient* và *TCPListener*.

- Các thành phần của lớp *TcpClient*



+ Phương thức khởi tạo:


Constructor Method	
Name	Description
TcpClient ()	Tạo một đối tượng TcpClient . Chưa đặt thông số gì.
TcpClient (EndPoint)	Tạo một TcpClient và gán 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, 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.

+ Một số thuộc tính:

	Name	Description
	Available	Cho biết số byte đã nhận về từ mạng và có sẵn để đọc.
	Client	Trả về Socket ứng với <i>TCPClient</i> hiện hành.
	Connected	Trạng thái cho biết đã kết nối được đến Server hay chưa ?

+ Một số phương thức:

	Name	Description
	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	<p>Trả về NetworkStream để từ đó giúp ta gửi hay nhận dữ liệu. (Thường làm tham số khi tạo StreamReader và StreamWriter) .</p> <p>Khi đã gắn vào StreamReader và StreamWriter rồi thì ta có thể gửi và nhận dữ liệu thông qua các phương thức Readln, writeline tương ứng của các lớp này.</p>
---	---------------------------	--

Ta sử dụng lớp TcpClient viết lại chương trình DateTimeClient như sau:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Net.Sockets;
using System.IO;
using System.Threading;
class DateTimeClient {
    static void Main(string[] args) {
        IPEndPoint iep = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 9999);
        TcpClient client = new TcpClient();
        client.Connect(iep);
        StreamReader sr = new StreamReader(client.GetStream());
        StreamWriter sw = new StreamWriter(client.GetStream());
        while (true) {
            string input = Console.ReadLine();
            sw.WriteLine(input);
            sw.Flush();
            if (input.ToUpper().Equals("QUIT")) break;
            string kq = sr.ReadLine();
            Console.WriteLine(kq);
        }
        sr.Close();
        sw.Close();
        client.Close();
    }
}
```

2.3.2. Lớp TCPListener






TCPListener 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.

Các thành phần của lớp TcpListener:

+ Phương thức khởi tạo:

Constructor method	
Name	Description
<u>TcpListener (Port: Int32)</u>	Tạo một TcpListener và lắng nghe tại cổng chỉ định.
<u>TcpListener (IPEndPoint)</u>	Tạo một TcpListener với giá trị Endpoint truyền vào.
<u>TcpListener (IPAddress, Int32)</u>	Tạo một TcpListener và lắng nghe các kết nối đến tại địa chỉ IP và cổng chỉ định.

+ Các phương thức khác

	Name	Description
	<u>AcceptSocket</u>	Chấp nhận một yêu cầu kết nối đang chờ.
	<u>AcceptTcpClient</u>	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)
	<u>Pending</u>	Cho biết liệu có kết nối nào đang chờ đợi không ? (True = có).
	<u>Start</u>	Bắt đầu lắng nghe các yêu cầu kết nối.
	<u>Stop</u>	Dừng việc nghe.

Sử dụng lớp TcpListener ta viết lại chương trình DateTime Server như sau:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Net.Sockets;
using System.IO;
class DateTimeServer{
    static void Main(string[] args) {
        IPEndPoint iep = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 2009);
        TcpListener server = new TcpListener(iep);
        server.Start();
        TcpClient client = server.AcceptTcpClient();
        StreamReader sr = new StreamReader(client.GetStream());
        StreamWriter sw = new StreamWriter(client.GetStream());
        string kq="";
        while (true) {
```

```

string s = sr.ReadLine();
s=s.ToUpper();
if (s.Equals("QUIT")) break;
if (s.Equals("GETDATE"))
    kq = DateTime.Now.ToString("dd/MM/yyyy");
else
    if (s.Equals("GETTIME"))
        kq = DateTime.Now.ToString("hh:mm:ss");
    else
        kq = "Khong hieu lenh";
sw.WriteLine(kq);
sw.Flush();
}
sr.Close();
sw.Close();
client.Close();
}
}

```

2.3.3. Lớp *UDPClient*

Giao thức UDP (User Datagram Protocol hay User Define Protocol) là một giao thức phi kết nối (Connectionless) có nghĩa là một bên có thể gửi dữ liệu cho bên kia mà không cần biết là bên đó đã sẵn sàng hay chưa ? (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.

Trong .NET, lớp **UDPClient** (nằm trong `System.Net.Sockets`) đóng gói các chức năng của giao thức UDP.

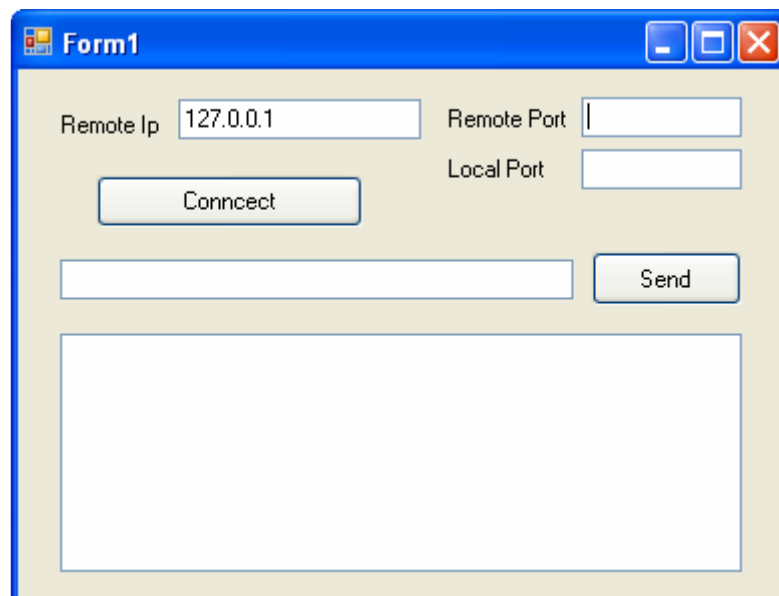
Constructor method	Description
UdpClient ()	Tạo một đối tượng (thể hiện) mới của lớp UDPClient .
UdpClient (AddressFamily)	Tạo một đối tượng (thể hiện) mới của lớp UDPClient . Thuộc một dòng địa chỉ (AddressFamily) được chỉ định.
UdpClient (Int32)	Tạo một UdpClient và gắn (bind) một cổng cho nó.
UdpClient (IPEndPoint)	Tạo một UdpClient và gắn (bind) một IPEndPoint (gán địa chỉ IP và cổng) cho nó.
UdpClient (Int32, AddressFamily)	Tạo một UdpClient và gắn số hiệu cổng, AddressFamily

UdpClient (String, Int32)		Tạo một UdpClient và thiết lập với một trạm từ xa mặc định.
PUBLIC Method		
	Name	Description
	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	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 – dù chỉ một chút – thì nó vẫn cứ chờ (blocking))
	Send	Gửi dữ liệu (đồng bộ) cho máy ở xa.

Ví dụ sử dụng UdpClient viết chương trình Chat giữa 2 máy:

Do chương trình ở 2 máy là như nhau ta chỉ cần viết một chương trình copy ra để sử dụng.

Hình ảnh của nó như sau:



```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
```

```

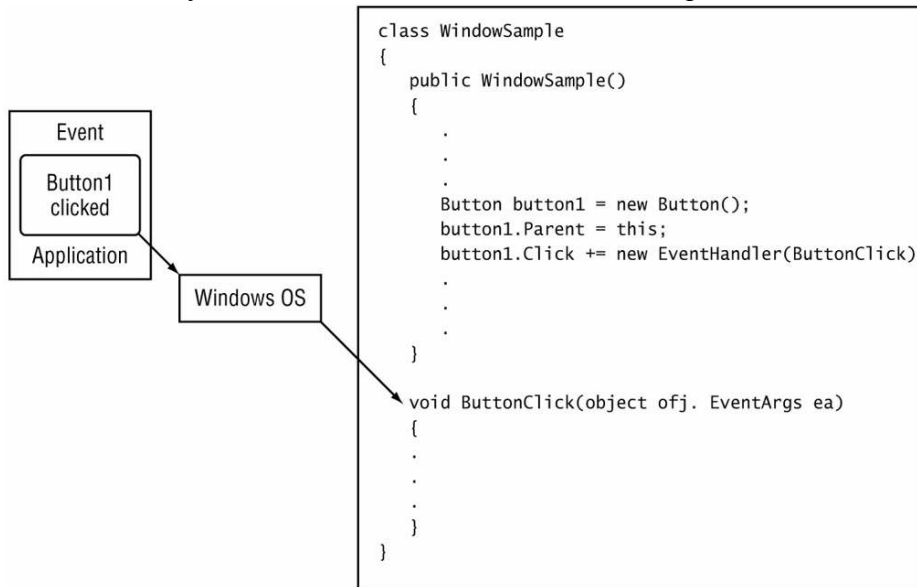
using System.Text;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;
using System.Threading;
namespace UdpChat {
    public partial class Form1 : Form {
        public Form1() {
            InitializeComponent();
            CheckForIllegalCrossThreadCalls = false;
        }
        private void btSend_Click(object sender, EventArgs e) {
            UdpClient send = new UdpClient();
            IPEndPoint iepRemote = new IPEndPoint(IPAddress.Parse(txtIp.Text),
int.Parse(txtRemotePort.Text));
            byte[] data = new byte[1024];
            data = Encoding.UTF8.GetBytes(txtSend.Text);
            send.Send(data, data.Length, iepRemote);
            txtReceive.Text += "Sender: " + txtSend.Text + "\r\n";
            txtSend.Clear();
            if (txtSend.Text.ToUpper().Equals("QUIT")) this.Dispose();
        }
        private void btConnect_Click(object sender, EventArgs e) {
            Thread tuyen = new Thread(new ThreadStart(NhanDI));
            tuyen.Start();
        }
        private void NhanDI() {
            UdpClient receiver = new UdpClient(int.Parse(txtLocalPort.Text));
            IPEndPoint iep = new IPEndPoint(IPAddress.Any, 0);
            while (true) {
                byte[] data = new byte[1024];
                data = receiver.Receive(ref iep);
                string s = Encoding.UTF8.GetString(data);
                if (s.Trim().ToUpper().Equals("QUIT")) break;
                txtReceive.Text += "Receiver: " + s + "\r\n";
            }
        }
    }
}

```

2.4. Socket không đồng bộ

2.4.1. Mô hình xử lý sự kiện của windows

Mô hình xử lý sự kiện của Windows được thể hiện qua hình sau:



Như vậy thông qua mô hình này ta có thể ủy nhiệm cho một thủ tục nào đó thực hiện khi sự kiện xảy ra trên các Control

Ví dụ:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

```

```

namespace EventDemo {
    public partial class Form1 : Form {
        public Form1() {
            InitializeComponent();
            button1.Click += new EventHandler(NhanTiep);
        }

        private void button1_Click(object sender, EventArgs e) {
            MessageBox.Show("Bac da nhan em");
        }
        private void NhanTiep(object sender, EventArgs e) {
            MessageBox.Show("Bac lai nhan em roi");
        }
    }
}

```

Ở ví dụ trên chúng ta ngoài sự kiện Click của button 1 chúng ta thêm một sự kiện khi button1 được nhấn đó là sự kiện NhanTiep.

2.4.2. Sử dụng Socket không đồng bộ

Để lập trình không đồng bộ với Socket chúng ta sử dụng các phương thức cho việc sử dụng bất đồng bộ

Các phương thức cho việc lập trình bất đồng bộ được chia làm 2 lại thường bắt đầu bằng Begin và End:

- ❑ Phương thức bắt đầu bằng Begin, bắt đầu một chức năng và được đăng ký với phương thức AsyncCallback
- ❑ Bắt đầu bằng End chỉ chức năng hoàn thành khi AsyncCallback được gọi.

Requests Started By...	Description of Request	Requests Ended BY...
BeginAccept()	To accept an incoming connection	EndAccept()
BeginConnect()	To connect to a remote host	EndConnect()
BeginReceive()	To retrieve data from a socket	EndReceive()
BeginReceiveFrom()	To retrieve data from a specific remote host	EndReceiveFrom()
BeginSend()	To send data from a socket	EndSend()
BeginSendTo()	To send data to a specific remote host	EndSendTo()

- Để chấp nhận kết nối bất đồng bộ ta sử dụng phương thức BeginAccept() và EndAccept() như sau:

- ❑ Phương thức BeginAccept() và EndAccept()
`IASyncResult BeginAccept(AsyncCallback callback, object state)`
`Socket EndAccept(IAsyncResult iar);`

- ❑ Thường được dùng như sau:

```
Socket sock = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
                                                                    ProtocolType.Tcp);
```

```
IPEndPoint iep = new IPEndPoint(IPAddress.Any, 9050);
```

```
sock.Bind(iep);
```

```
sock.Listen(5);
```

```
sock.BeginAccept(new AsyncCallback(CallAccept), sock);
```

Trong đó phương thức CallAccept thường được viết như sau:

```
private static void CallAccept(IAsyncResult iar) {
    Socket server = (Socket)iar.AsyncState;
    Socket client = server.EndAccept(iar);
    ...
}
```

- Để thiết lập kết nối theo cách bất đồng bộ chúng ta sử dụng phương thức BeginConnect() và EndConnect() như sau:

- ❑ Phương thức BeginConnect() và EndConnect()

```

Socket newsock = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);
IPEndPoint iep = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 9050);
newsock.BeginConnect(iep, new AsyncCallback(Connected), newsock);

```

Trong đó phương thức Connected thường được viết như sau:

```

public static void Connected(IAsyncResult iar) {
    Socket sock = (Socket)iar.AsyncState;
    try {
        sock.EndConnect(iar);
    } catch (SocketException) {
        Console.WriteLine("Unable to connect to host");
    }
}

```

- Để gửi dữ liệu bất đồng bộ chúng ta làm như sau:

+ Phương thức BeginSend() và EndSend()

+ BeginSend()

```

IAsyncResult BeginSend(byte[] buffer, int offset, int size, SocketFlags sockflag,
AsyncCallback callback, object state)

```

Ví dụ:

```

sock.BeginSend(data, 0, data.Length, SocketFlags.None, new
AsyncCallback(SendData), sock);
+ EndSend()

```

```

int EndSend(IAsyncResult iar)

```

Trong đó phương thức SendData thường được viết như sau:

```

private static void SendData(IAsyncResult iar) {
    Socket server = (Socket)iar.AsyncState;
    int sent = server.EndSend(iar);
}

```

Tương tự như giao thức hướng kết nối nếu ta sử dụng gửi dữ liệu theo giao thức không hướng kết nối chúng ta cũng thực hiện tương tự như sau:

❑ Phương thức BeginSendTo() và EndSendTo()

```

IAsyncResult BeginSendTo(byte[] buffer, int offset, int size, SocketFlags
sockflag, EndPoint ep, AsyncCallback callback, object state)

```

Ví dụ:

```

IPEndPoint iep = new EndPoint(IPAddress.Parse("192.168.1.6"), 9050);
sock.BeginSendTo(data, 0, data.Length, SocketFlags.None, iep, new
AsyncCallback(SendDataTo), sock);
int EndSendTo(IAsyncResult iar)

```

- Để nhận dữ liệu bất đồng bộ ta thực hiện như sau:

+ Nhận dữ liệu với giao thức hướng kết nối:

❑ Phương thức BeginReceive và EndReceive()

```
sock.BeginReceive(data, 0, data.Length, SocketFlags.None, new
AsyncCallback(ReceivedData), sock);
```

Với ReceivedData được định nghĩa như sau:

```
void ReceivedData(IAsyncResult iar) {
    Socket remote = (Socket)iar.AsyncState;
    int recv = remote.EndReceive(iar);
    string receivedData = Encoding.ASCII.GetString(data, 0,
recv);
    Console.WriteLine(receivedData);
}
```

+ Nhận dữ liệu bất đồng bộ với giao thức không hướng kết nối.

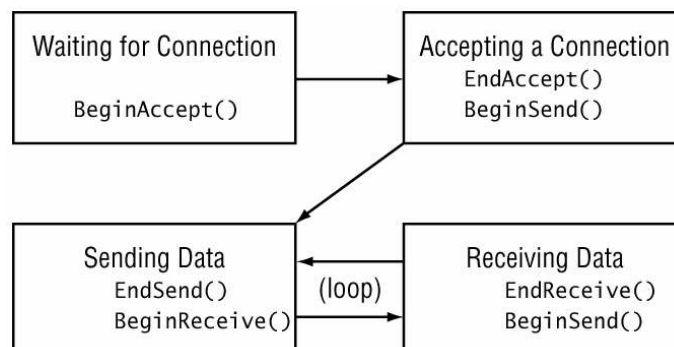
❑ Phương thức BeginReceiveFrom() and EndReceiveFrom()

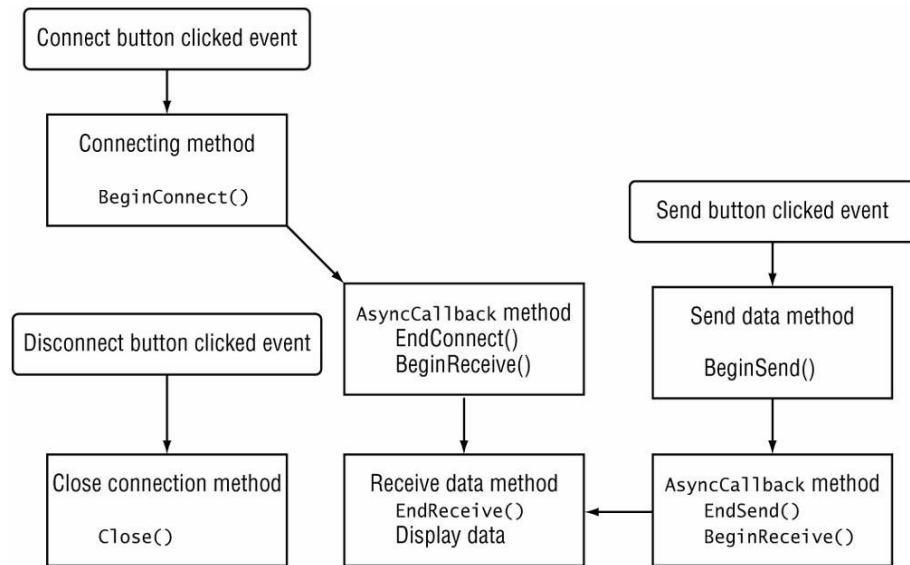
```
sock.BeginReceive(data, 0, data.Length, SocketFlags.None, ref iep, new
AsyncCallback(ReceiveData), sock);
```

```
void ReceiveData(IAsyncResult iar){
    Socket remote = (Socket)iar.AsyncState;
    int recv = remote.EndReceiveFrom(iar);
    string stringData = Encoding.ASCII.GetString(data, 0,
recv);
    Console.WriteLine(stringData);
}
```

2.4.3. Ví dụ về Socket không đồng bộ

Sau đây chúng ta sẽ sử dụng các phương thức không đồng bộ viết chương trình Client/Server theo Socket bất đồng bộ, mỗi khi Client gửi dữ liệu lên Server, nó sẽ in ra và gửi trả lại cho Client. Mô hình của client/server sử dụng các phương thức bất đồng bộ như sau:





Chương trình phía Client:

```

using System;
using System.Drawing;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Windows.Forms;
class AsyncTcpClient:Form
{
    private TextBox newText;
    private TextBox conStatus;
    private ListBox results;
    private Socket client;
    private byte[] data = new byte[1024];
    private int size = 1024;
    public AsyncTcpClient()
    {
        Text = "Asynchronous TCP Client";
        Size = new Size(400, 380);
        Label label1 = new Label();
        label1.Parent = this;
        label1.Text = "Enter text string:";
        label1.AutoSize = true;
        label1.Location = new Point(10, 30);
        newText = new TextBox();
        newText.Parent = this;
        newText.Size = new Size(200, 2 * Font.Height);
        newText.Location = new Point(10, 55);
        results = new ListBox();
        results.Parent = this;
        results.Location = new Point(10, 85);
        results.Size = new Size(360, 18 * Font.Height);
        Label label2 = new Label();
        label2.Parent = this;
    }
}
  
```

```

label2.Text = "Connection Status:";
label2.AutoSize = true;
label2.Location = new Point(10, 330);
conStatus = new TextBox();
conStatus.Parent = this;
conStatus.Text = "Disconnected";
conStatus.Size = new Size(200, 2 * Font.Height);
conStatus.Location = new Point(110, 325);
Button sendit = new Button();
sendit.Parent = this;
sendit.Text = "Send";
sendit.Location = new Point(220,52);
sendit.Size = new Size(5 * Font.Height, 2 * Font.Height);
sendit.Click += new EventHandler(ButtonSendOnClick);
Button connect = new Button();
connect.Parent = this;
connect.Text = "Connect";
connect.Location = new Point(295, 20);
connect.Size = new Size(6 * Font.Height, 2 * Font.Height);
connect.Click += new EventHandler(ButtonConnectOnClick);
Button discon = new Button();
discon.Parent = this;
discon.Text = "Disconnect";
discon.Location = new Point(295,52);
discon.Size = new Size(6 * Font.Height, 2 * Font.Height);
discon.Click += new EventHandler(ButtonDisconOnClick);
}
void ButtonConnectOnClick(object obj, EventArgs ea)
{
    conStatus.Text = "Connecting...";
    Socket newsock = new Socket(AddressFamily.InterNetwork,
        SocketType.Stream, ProtocolType.Tcp);
    IPEndPoint iep = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 9050);
    newsock.BeginConnect(iep, new AsyncCallback(Connected), newsock);
}
void ButtonSendOnClick(object obj, EventArgs ea)
{
    byte[] message = Encoding.ASCII.GetBytes(newText.Text);
    newText.Clear();
    client.BeginSend(message, 0, message.Length, SocketFlags.None,
        new AsyncCallback(SendData), client);
}
void ButtonDisconOnClick(object obj, EventArgs ea)
{
    client.Close();
    conStatus.Text = "Disconnected";
}
void Connected(IAsyncResult iar)
{

```



```

client = (Socket)iar.AsyncState;
try
{
    client.EndConnect(iar);
    conStatus.Text = "Connected to: " + client.RemoteEndPoint.ToString();
    client.BeginReceive(data, 0, size, SocketFlags.None,
        new AsyncCallback(ReceiveData), client);
} catch (SocketException)
{
    conStatus.Text = "Error connecting";
}
}
void ReceiveData(IAsyncResult iar)
{
    Socket remote = (Socket)iar.AsyncState;
    int recv = remote.EndReceive(iar);
    string stringData = Encoding.ASCII.GetString(data, 0, recv);
    results.Items.Add(stringData);
}
void SendData(IAsyncResult iar)
{
    Socket remote = (Socket)iar.AsyncState;
    int sent = remote.EndSend(iar);
    remote.BeginReceive(data, 0, size, SocketFlags.None,
        new AsyncCallback(ReceiveData), remote);
}
public static void Main()
{
    Application.Run(new AsyncTcpClient());
}
}

```

Chương trình phía Server:

```

using System;
using System.Drawing;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Windows.Forms;
class AsyncTcpSrvr : Form
{
    private TextBox conStatus;
    private ListBox results;
    private byte[] data = new byte[1024];
    private int size = 1024;
    private Socket server;
    public AsyncTcpSrvr()
    {
        Text = "Asynchronous TCP Server";
        Size = new Size(400, 380);
    }
}

```

```

results = new ListBox();
results.Parent = this;
results.Location = new Point(10, 65);
results.Size = new Size(350, 20 * Font.Height);
Label label1 = new Label();
label1.Parent = this;
label1.Text = "Text received from client:";
label1.AutoSize = true;
label1.Location = new Point(10, 45);
Label label2 = new Label();
label2.Parent = this;
label2.Text = "Connection Status:";
label2.AutoSize = true;
label2.Location = new Point(10, 330);
conStatus = new TextBox();
conStatus.Parent = this;
conStatus.Text = "Waiting for client...";
conStatus.Size = new Size(200, 2 * Font.Height);
conStatus.Location = new Point(110, 325);
Button stopServer = new Button();
stopServer.Parent = this;
stopServer.Text = "Stop Server";
stopServer.Location = new Point(260, 32);
stopServer.Size = new Size(7 * Font.Height, 2 * Font.Height);
stopServer.Click += new EventHandler(ButtonStopOnClick);
server = new Socket(AddressFamily.InterNetwork,
    SocketType.Stream, ProtocolType.Tcp);
IPEndPoint iep = new IPEndPoint(IPAddress.Any, 9050);
server.Bind(iep);
server.Listen(5);
server.BeginAccept(new AsyncCallback(AcceptConn), server);
}
void ButtonStopOnClick(object obj, EventArgs ea)
{
    Close();
}
void AcceptConn(IAsyncResult iar)
{
    Socket oldserver = (Socket)iar.AsyncState;
    Socket client = oldserver.EndAccept(iar);
    conStatus.Text = "Connected to: " + client.RemoteEndPoint.ToString();
    string stringData = "Welcome to my server";
    byte[] message1 = Encoding.ASCII.GetBytes(stringData);
    client.BeginSend(message1, 0, message1.Length, SocketFlags.None,
        new AsyncCallback(SendData), client);
}
void SendData(IAsyncResult iar)
{
    Socket client = (Socket)iar.AsyncState;

```

```

int sent = client.EndSend(iar);
client.BeginReceive(data, 0, size, SocketFlags.None,
    new AsyncCallback(ReceiveData), client);
}
void ReceiveData(IAsyncResult iar)
{
    Socket client = (Socket)iar.AsyncState;
    int recv = client.EndReceive(iar);
    if (recv == 0)
    {
        client.Close();
        conStatus.Text = "Waiting for client...";
        server.BeginAccept(new AsyncCallback(AcceptConn), server);
        return;
    }
    string receivedData = Encoding.ASCII.GetString(data, 0, recv);
    results.Items.Add(receivedData);
    byte[] message2 = Encoding.ASCII.GetBytes(receivedData);
    client.BeginSend(message2, 0, message2.Length, SocketFlags.None,
        new AsyncCallback(SendData), client);
}
}
public static void Main()
{
    Application.Run(new AsyncTcpSrvr());
}
}

```

2.4.4. Sử dụng các phương thức Non-blocking

Để lập trình bất đồng bộ chúng ta có thể sử dụng các phương thức Non – bloking như phương thức Poll() và phương thức Select:

+ Phương thức Poll()

```
bool Poll(int microseconds, SelectMode mode);
```

❑ SelectRead: Poll() trả về true nếu một trong những điều kiện sau được thoả:

- Nếu phương thức Accept() thành công
- Nếu có dữ liệu trên Socket
- Nếu kết nối đã đóng

❑ SelectWrite: Poll() trả về true nếu thoả một trong những điều kiện sau:

- Phương thức Connect() thành công
- Nếu có dữ liệu trên Socket để gửi

❑ SelectError: Poll() trả về true nếu một trong những điều kiện sau được thoả:

- Nếu phương thức Connect() thất bại
- Nếu có dữ liệu ngoài băng thông chuẩn gửi đến nhưng thuộc tính OutOfBandInline không được thiết lập là true.

+ Phương thức Select():

Socket.Select(IList checkRead, IList checkWrite, IList checkError, int microseconds)

- *checkRead* monitors the specified sockets for the ability to read data from the socket.
- *checkWrite* monitors the specified sockets for the ability to write data to the socket.
- *checkError* monitors the specified sockets for error conditions.

Ví dụ ứng dụng Server sử dụng phương thức Poll()

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
class TcpPollSrvr
{
    public static void Main()
    {
        int recv;
        byte[] data = new byte[1024];
        IPEndPoint ipep = new IPEndPoint(IPAddress.Any,
            9050);
        Socket newsock = new
            Socket(AddressFamily.InterNetwork,
                SocketType.Stream, ProtocolType.Tcp);
        newsock.Bind(ipep);
        newsock.Listen(10);
        Console.WriteLine("Waiting for a client...");
        bool result;
        int i = 0;
        while (true)
        {
            i++;
            Console.WriteLine("polling for accept#{0}...", i);
            result = newsock.Poll(1000000, SelectMode.SelectRead);
            if (result)
            {
                break;
            }
        }
        Socket client = newsock.Accept();
        IPEndPoint newclient =
            (IPEndPoint)client.RemoteEndPoint;
        Console.WriteLine("Connected with {0} at port {1}",
            newclient.Address, newclient.Port);

        string welcome = "Welcome to my test server";
        data = Encoding.ASCII.GetBytes(welcome);
        client.Send(data, data.Length,
            SocketFlags.None);
    }
}
```

```

i = 0;
while (true)
{
    Console.WriteLine("polling for receive #{0}...", i);
    i++;
    result = client.Poll(3000000, SelectMode.SelectRead);
    if (result)
    {
        data = new byte[1024];
        i = 0;
        recv = client.Receive(data);
        if (recv == 0)
            break;

        Console.WriteLine(
            Encoding.ASCII.GetString(data, 0, recv));
        client.Send(data, recv, 0);
    }
}
Console.WriteLine("Disconnected from {0}",
    newclient.Address);
client.Close();
newsock.Close();
}
}

```

Sau đây chúng ta sẽ viết một chương trình Server sử dụng phương thức Select() để chấp nhận 2 kết nối đến từ client và xử lý từng kết nối.

Chương trình Select Server:

```

using System;
using System.Collections;
using System.Net;
using System.Net.Sockets;
using System.Text;
class SelectTcpSrvr
{
    public static void Main()
    {
        ArrayList sockList = new ArrayList(2);
        ArrayList copyList = new ArrayList(2);
        Socket main = new Socket(AddressFamily.InterNetwork,
            SocketType.Stream, ProtocolType.Tcp);
        IPEndPoint iep = new IPEndPoint(IPAddress.Any, 9050);
        byte[] data = new byte[1024];
        string stringData;
    }
}

```

```

int recv;
main.Bind(iep);
main.Listen(2);
Console.WriteLine("Waiting for 2 clients...");
Socket client1 = main.Accept();
IPEndPoint iep1 = (IPEndPoint)client1.RemoteEndPoint;
client1.Send(Encoding.ASCII.GetBytes("Welcome to my server"));
Console.WriteLine("Connected to {0}", iep1.ToString());
sockList.Add(client1);
Console.WriteLine("Waiting for 1 more client...");
Socket client2 = main.Accept();
IPEndPoint iep2 = (IPEndPoint)client2.RemoteEndPoint;
client2.Send(Encoding.ASCII.GetBytes("Welcome to my server"));
Console.WriteLine("Connected to {0}", iep2.ToString());
sockList.Add(client2);
main.Close();
while (true)
{
    copyList = new ArrayList(sockList);
    Console.WriteLine("Monitoring {0} sockets...", copyList.Count);
    Socket.Select(copyList, null, null, 10000000);
    foreach (Socket client in copyList)
    {
        data = new byte[1024];
        recv = client.Receive(data);
        stringData = Encoding.ASCII.GetString(data, 0, recv);
        Console.WriteLine("Received: {0}", stringData);
        if (recv == 0)
        {
            iep = (IPEndPoint)client.RemoteEndPoint;
            Console.WriteLine("Client {0} disconnected.", iep.ToString());
            client.Close();
            sockList.Remove(client);
            if (sockList.Count == 0)
            {
                Console.WriteLine("Last client disconnected, bye");
                return;
            }
        }
    }
    else
        client.Send(data, recv, SocketFlags.None);
}

```

```

    }
}
}
}

```

Chương trình Client:

```

using System;
using System.Collections;
using System.Net;
using System.Net.Sockets;
using System.Text;
class SelectTcpClient
{
    public static void Main()
    {
        Socket sock = new Socket(AddressFamily.InterNetwork,
            SocketType.Stream, ProtocolType.Tcp);
        IPEndPoint iep = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 9050);
        byte[] data = new byte[1024];
        string stringData;
        int recv;
        sock.Connect(iep);
        Console.WriteLine("Connected to server");
        recv = sock.Receive(data);
        stringData = Encoding.ASCII.GetString(data, 0, recv);
        Console.WriteLine("Received: {0}", stringData);
        while (true)
        {
            stringData = Console.ReadLine();
            if (stringData == "exit")
                break;
            data = Encoding.ASCII.GetBytes(stringData);
            sock.Send(data, data.Length, SocketFlags.None);
            data = new byte[1024];
            recv = sock.Receive(data);
            stringData = Encoding.ASCII.GetString(data, 0, recv);
            Console.WriteLine("Received: {0}", stringData);
        }
        sock.Close();
    }
}

```

2.5. Sử dụng Thread trong các ứng dụng mạng

Một số khái niệm

- Đa nhiệm (Multitasking): Là khả năng hệ điều hành làm nhiều công việc tại một thời điểm
- Tiến trình (Process): Khi chạy một ứng dụng, hệ điều hành sẽ cấp phát riêng cho ứng dụng đó bộ nhớ và các tài nguyên khác. Bộ nhớ và tài nguyên vật lý riêng biệt này được gọi là một tiến trình. Các tài nguyên và bộ nhớ của một tiến trình thì chỉ tiến trình đó được phép truy cập.
- Tuyến (Thread): Trong hệ thống, một tiến trình có thể có một hoặc nhiều chuỗi thực hiện tách biệt nhau và có thể chạy đồng thời. Mỗi chuỗi thực hiện này được gọi là một tuyến (Thread). Trong một ứng dụng, Thread khởi tạo đầu tiên gọi là Thread sơ cấp hay Thread chính.

2.5.1. Sử dụng Thread trong chương trình .Net

Để sử dụng Thread trong .Net ta sử dụng Namespace System.Threading

- Một số phương thức thường dùng

Public Method Name	Mô tả
Abort()	Kết thúc Thread
Join()	Buộc chương trình phải chờ cho thread kết thúc (Block) thì mới thực hiện tiếp (các câu lệnh đứng sau Join).
Resume()	Tiếp tục chạy thread đã bị tạm ngưng - suspended.
Sleep()	Static method : Tạm dừng thread trong một khoảng thời gian.
Start()	Bắt đầu chạy (khởi động) một thread. Sau khi gọi phương thức này, trạng thái của thread chuyển từ trạng thái hiện hành sang Running.
Suspend()	Tạm ngưng (ngủ) thread. (Phương thức này đã bị loại khỏi phiên bản VS.NET 2005)

- Một số thuộc tính thường dùng:

Public Property Name	Mô tả
CurrentThread	This static property: Trả về thread hiện hành đang chạy.
IsAlive	Trả về giá trị cho biết trạng thái thực thi của thread hiện hành.
IsBackground	Sets or gets giá trị cho biết là thread là background hay foreground thread.
IsThreadPoolThread	Gets a value indicating whether a thread is part of a thread pool.
Priority	Sets or gets giá trị để chỉ định độ ưu tiên (dành nhiều hay ít CPU cho thread). Cao nhất là 4, thấp nhất là 0.

Public Property Name	Mô tả
ThreadState	Lấy về trạng thái của thread (đang dừng, hay đang chạy...)

- Tạo một tuyến trong C#

```

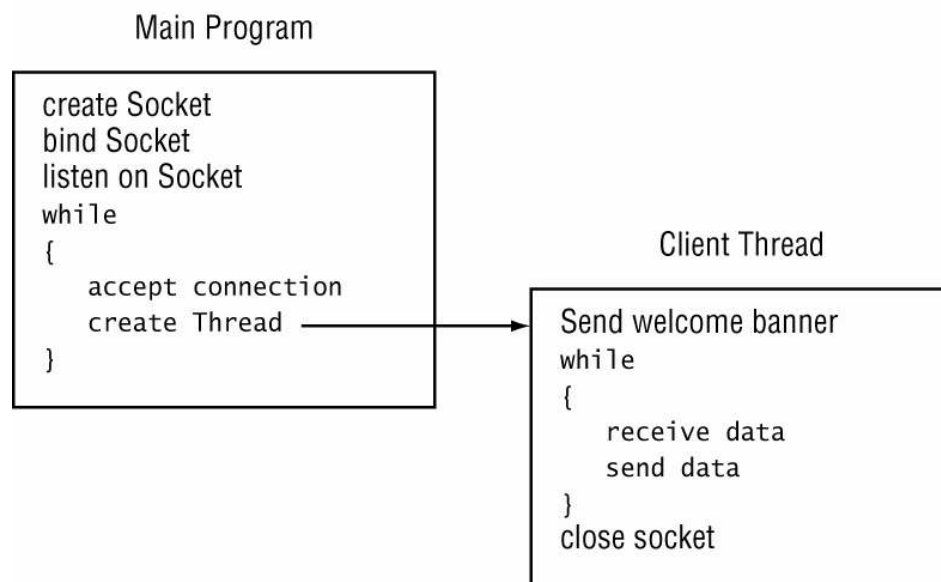
.....
Thread newThread=new Thread(new ThreadStart(newMethod));
.....
}
void newMethod() {
...
}

```

2.5.2. Sử dụng Thread trong các chương trình Server

- Đa tuyến hay được ứng dụng trong các chương trình Server, các chương trình đòi hỏi tại một thời điểm chấp nhận nhiều kết nối đến từ các Client.

- Để các chương trình Server có thể xử lý nhiều Client tại một thời điểm ta có mô hình ứng dụng đa tuyến như sau:



Sau đây chúng ta viết lại chương trình DateTimeServer có sử dụng Thread như sau:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using System.IO;
class Program {
    static void Main(string[] args) {
        IPEndPoint iep = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 2009);
        TcpListener server = new TcpListener(iep);
        server.Start();
        while (true) {

```

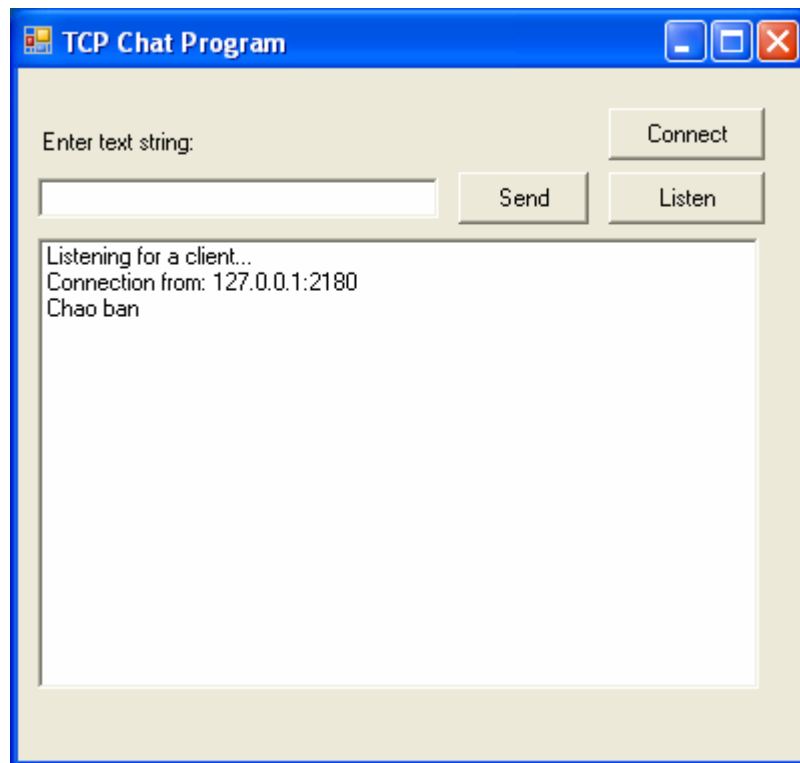
```

        //chap nhan ket noi
        TcpClient client= server.AcceptTcpClient();
        //Tao ra tuyen moi de xu ly moi Client
        new ClientThread(client);
    }
    server.Stop();
}
}
class ClientThread {
    private Thread tuyen;
    private TcpClient client;
    public ClientThread(TcpClient client) {
        this.client = client;
        tuyen = new Thread(new ThreadStart(GuiNhanDL));
        tuyen.Start();
    }
    private void GuiNhanDL() {
        StreamReader sr = new StreamReader(client.GetStream());
        StreamWriter sw= new StreamWriter(client.GetStream());
        string kq="";
        while(true)
        {
            string s=sr.ReadLine();
            s=s.ToUpper();
            if(s.Equals("QUIT")) break;
            if(s.Equals("GETDATE"))
                kq=DateTime.Now.ToString("dd/MM/yyyy");
            else
                if(s.Equals("GETTIME"))
                    kq=DateTime.Now.ToString("hh:mm:ss");
                else
                    kq="Khong hieu lenh";
            sw.WriteLine(kq);
            sw.Flush();
        }
        client.Close();
    }
}
}

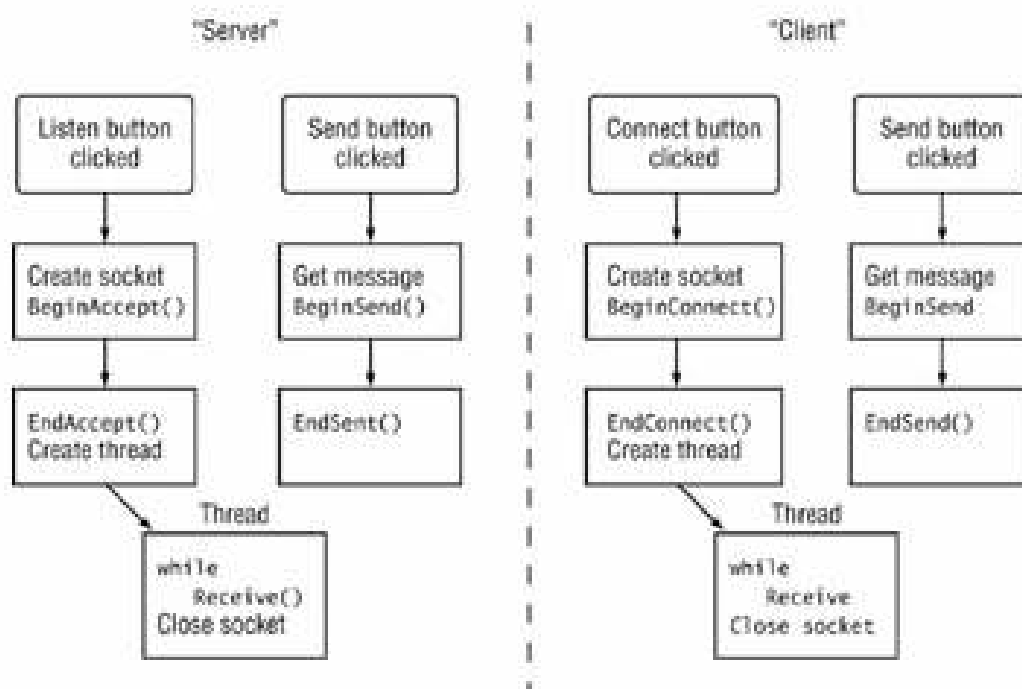
```

2.5.3. Sử dụng Thread để gửi/nhận dữ liệu

Ứng dụng đa tuyến trong việc gửi nhận dữ liệu chúng ta viết chương trình Chat theo giao thức TCP như sau:



Ứng dụng mô hình xử lý sự kiện của Windows và đa tuyến và Socket không đồng bộ ta chỉ cần viết một chương trình sau đó dịch ra, ta chạy ứng dụng nhấn Listen nó sẽ lắng nghe trong vai trò Server còn khi ta chạy và nhấn Connect nó sẽ đóng vai trò Client và kết nối tới Server.



Văn bản chương trình như sau:

```

using System;
using System.Drawing;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;

```

```

using System.Windows.Forms;
class TcpChat:Form
{
    private static TextBox newText;
    private static ListBox results;
    private static Socket client;
    private static byte[] data = new byte[1024];
    public TcpChat()
    {
        Text = "TCP Chat Program";
        Size = new Size(400, 380);

        Label label1 = new Label();
        label1.Parent = this;
        label1.Text = "Enter text string:";
        label1.AutoSize = true;
        label1.Location = new Point(10, 30);
        newText = new TextBox();
        newText.Parent = this;
        newText.Size = new Size(200, 2 * Font.Height);
        newText.Location = new Point(10, 55);
        results = new ListBox();
        results.Parent = this;
        results.Location = new Point(10, 85);
        results.Size = new Size(360, 18 * Font.Height);
        Button sendit = new Button();
        sendit.Parent = this;
        sendit.Text = "Send";
        sendit.Location = new Point(220,52);
        sendit.Size = new Size(5 * Font.Height, 2 * Font.Height);
        sendit.Click += new EventHandler(ButtonSendOnClick);
        Button connect = new Button();
        connect.Parent = this;
        connect.Text = "Connect";
        connect.Location = new Point(295, 20);
        connect.Size = new Size(6 * Font.Height, 2 * Font.Height);
        connect.Click += new EventHandler(ButtonConnectOnClick);
        Button listen = new Button();
        listen.Parent = this;
        listen.Text = "Listen";
        listen.Location = new Point(295,52);
        listen.Size = new Size(6 * Font.Height, 2 * Font.Height);
        listen.Click += new EventHandler(ButtonListenOnClick);
    }
    void ButtonListenOnClick(object obj, EventArgs ea)
    {
        results.Items.Add("Listening for a client...");
        Socket newsock = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
            ProtocolType.Tcp);
    }
}

```

```

IPEndPoint iep = new IPEndPoint(IPAddress.Any, 9050);
newsock.Bind(iep);
newsock.Listen(5);
newsock.BeginAccept(new AsyncCallback(AcceptConn), newsock);
}
void ButtonConnectOnClick(object obj, EventArgs ea)
{
    results.Items.Add("Connecting...");
    client = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
        ProtocolType.Tcp);
    IPEndPoint iep = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 9050);
    client.BeginConnect(iep, new AsyncCallback(Connected), client);
}
void ButtonSendOnClick(object obj, EventArgs ea)
{
    byte[] message = Encoding.ASCII.GetBytes(newText.Text);
    newText.Clear();
    client.BeginSend(message, 0, message.Length, 0,
        new AsyncCallback(SendData), client);
}
void AcceptConn(IAsyncResult iar)
{
    Socket oldserver = (Socket)iar.AsyncState;
    client = oldserver.EndAccept(iar);
    results.Items.Add("Connection from: " + client.RemoteEndPoint.ToString());
    Thread receiver = new Thread(new ThreadStart(ReceiveData));
    receiver.Start();
}
void Connected(IAsyncResult iar)
{
    try
    {
        client.EndConnect(iar);
        results.Items.Add("Connected to: " + client.RemoteEndPoint.ToString());
        Thread receiver = new Thread(new ThreadStart(ReceiveData));
        receiver.Start();
    } catch (SocketException)
    {
        results.Items.Add("Error connecting");
    }
}
void SendData(IAsyncResult iar)
{
    Socket remote = (Socket)iar.AsyncState;
    int sent = remote.EndSend(iar);
}
void ReceiveData()
{
    int recv;

```

```

string stringData;
while (true)
{
    recv = client.Receive(data);
    stringData = Encoding.ASCII.GetString(data, 0, recv);
    if (stringData == "bye")
        break;
    results.Items.Add(stringData);
}
stringData = "bye";
byte[] message = Encoding.ASCII.GetBytes(stringData);
client.Send(message);
client.Close();
results.Items.Add("Connection stopped");
return;
}
public static void Main()
{
    Application.Run(new TcpChat());
}
}

```

2.5.4. Sử dụng ThreadPool trong các chương trình .Net

Method	Description
BindHandle()	Binds an operating system handle to the thread pool
GetAvailableThreads()	Gets the number of worker threads available for use in the thread pool
GetMaxThreads()	Gets the maximum number of worker threads available in the thread pool
QueueUserWorkItem()	Queues a user delegate to the thread pool
RegisterWaitForSingleObject()	Registers a delegate waiting for a WaitHandle object
UnsafeQueueUserWorkItem()	Queues an unsafe user delegate to the thread pool but does not propagate the calling stack onto the worker thread
UnsafeRegisterWaitForSingleObject()	Registers an unsafe delegate waiting for a WaitHandle object

```

using System;
using System.Threading;
class ThreadPoolSample
{

```

```

public static void Main()
{
    ThreadPoolSample tps = new ThreadPoolSample();
}
public ThreadPoolSample()
{
    int i;
    ThreadPool.QueueUserWorkItem(new WaitCallback(Counter));
    ThreadPool.QueueUserWorkItem(new WaitCallback(Counter2));
    for(i = 0; i < 10; i++)
    {
        Console.WriteLine("main: {0}", i);
        Thread.Sleep(1000);
    }
}
void Counter(object state)
{
    int i;
    for (i = 0; i < 10; i++)
    {
        Console.WriteLine(" thread: {0}", i);
        Thread.Sleep(2000);
    }
}
void Counter2(object state)
{
    int i;
    for (i = 0; i < 10; i++)
    {
        Console.WriteLine(" thread2: {0}", i);
        Thread.Sleep(3000);
    }
}
}

```

2.5.5. Sử dụng ThreadPool trong các chương trình Server

```

using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;
class ThreadPoolTcpSrvr
{
    private TcpListener client;
    public ThreadPoolTcpSrvr()
    {
        client = new TcpListener(9050);
        client.Start();
        Console.WriteLine("Waiting for clients...");
        while(true)
        {
            while (!client.Pending())
            {
                Thread.Sleep(1000);
            }
            ConnectionThread newconnection = new ConnectionThread();
            newconnection.threadListener = this.client;
            ThreadPool.QueueUserWorkItem(new
                WaitCallback(newconnection.HandleConnection));
        }
    }
    public static void Main()
    {
        ThreadPoolTcpSrvr tpts = new ThreadPoolTcpSrvr();
    }
}

```

```

    }
}
class ConnectionThread
{
    public TcpListener threadListener;
    private static int connections = 0;
    public void HandleConnection(object state)
    {
        int recv;
        byte[] data = new byte[1024];
        TcpClient client = threadListener.AcceptTcpClient();
        NetworkStream ns = client.GetStream();
        connections++;
        Console.WriteLine("New client accepted: {0} active connections",
            connections);
        string welcome = "Welcome to my test server";
        data = Encoding.ASCII.GetBytes(welcome);
        ns.Write(data, 0, data.Length);
        while(true)
        {
            data = new byte[1024];
            recv = ns.Read(data, 0, data.Length);
            if (recv == 0)
                break;

            ns.Write(data, 0, recv);
        }
        ns.Close();
        client.Close();
        connections--;
        Console.WriteLine("Client disconnected: {0} active connections",
            connections);
    }
}

```

2.6. Kỹ thuật IP Multicasting

2.6.1. Broadcasting là gì?

Broadcast, tiếng Việt gọi là quảng bá. Trong hệ thống mạng hữu tuyến, quảng bá là thuật ngữ dùng để chỉ việc gửi một gói thông tin đến tất cả các nút mạng trong mạng. Để thực hiện hình thức quảng bá, địa chỉ đến của gói tin sẽ là địa chỉ quảng bá.

Có hai loại là: Local Broadcast và Global Broadcast

2.6.2. Sử dụng Broadcasting để gửi dữ liệu đến nhiều máy trong mạng cục bộ

Gửi gói dữ liệu Broadcast

```

using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
class BadBroadcast {
    public static void Main() {
        Socket sock = new Socket(AddressFamily.InterNetwork,
            SocketType.Dgram, ProtocolType.Udp);
        IPEndPoint iep = new IPEndPoint(IPAddress.Broadcast, 9050);
        byte[] data = Encoding.ASCII.GetBytes("This is a test message");
        sock.SendTo(data, iep);
        sock.Close();
    }
}

```



```
}
```

Chúng ta phải thiết lập như sau:

```
class Broadcast {
    public static void Main() {
        Socket sock = new Socket(AddressFamily.InterNetwork, SocketType.Dgram,
            ProtocolType.Udp);
        IPEndPoint iep1 = new IPEndPoint(IPAddress.Broadcast, 9050);
        IPEndPoint iep2 = new IPEndPoint(IPAddress.Parse("192.168.1.255"), 9050);
        string hostname = Dns.GetHostName();
        byte[] data = Encoding.ASCII.GetBytes(hostname);
        sock.SetSocketOption(SocketOptionLevel.Socket,
            SocketOptionName.Broadcast, 1);
        sock.SendTo(data, iep1);
        sock.SendTo(data, iep2);
        sock.Close();
    }
}
```

Nhận gói dữ liệu Broadcast

```
class RecvBroadcast {
    public static void Main() {
        Socket sock = new Socket(AddressFamily.InterNetwork, SocketType.Dgram,
            ProtocolType.Udp);
        IPEndPoint iep = new IPEndPoint(IPAddress.Any, 9050);
        sock.Bind(iep); EndPoint ep = (EndPoint)iep;
        Console.WriteLine("Ready to receive..."); byte[] data = new byte[1024];
        int recv = sock.ReceiveFrom(data, ref ep);
        string stringData = Encoding.ASCII.GetString(data, 0, recv);
        Console.WriteLine("received: {0} from: {1}", stringData, ep.ToString());
        data = new byte[1024]; recv = sock.ReceiveFrom(data, ref ep);
        stringData = Encoding.ASCII.GetString(data, 0, recv);
        Console.WriteLine("received: {0} from: {1}", stringData, ep.ToString());
        sock.Close();
    }
}
```

2.6.3. Multicasting là gì?

Một địa chỉ multicast cho phép thiết bị gửi dữ liệu tới một tập xác định trước các host, được biết đến như các nhóm multicast, trong các mạng con khác nhau.

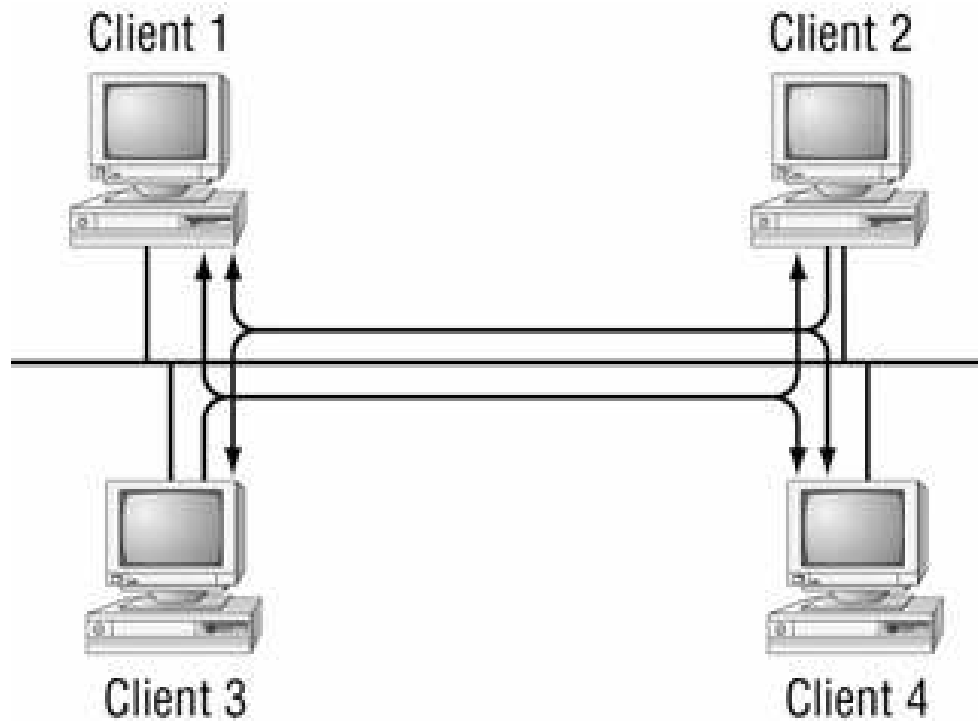
Một số địa chỉ Multicast

Địa chỉ multicast	Chức năng
224.0.0.0	Địa chỉ cơ sở
224.0.0.1	Tất cả các hệ thống trên mạng con này
224.0.0.2	Tất cả các Router trên mạng con này

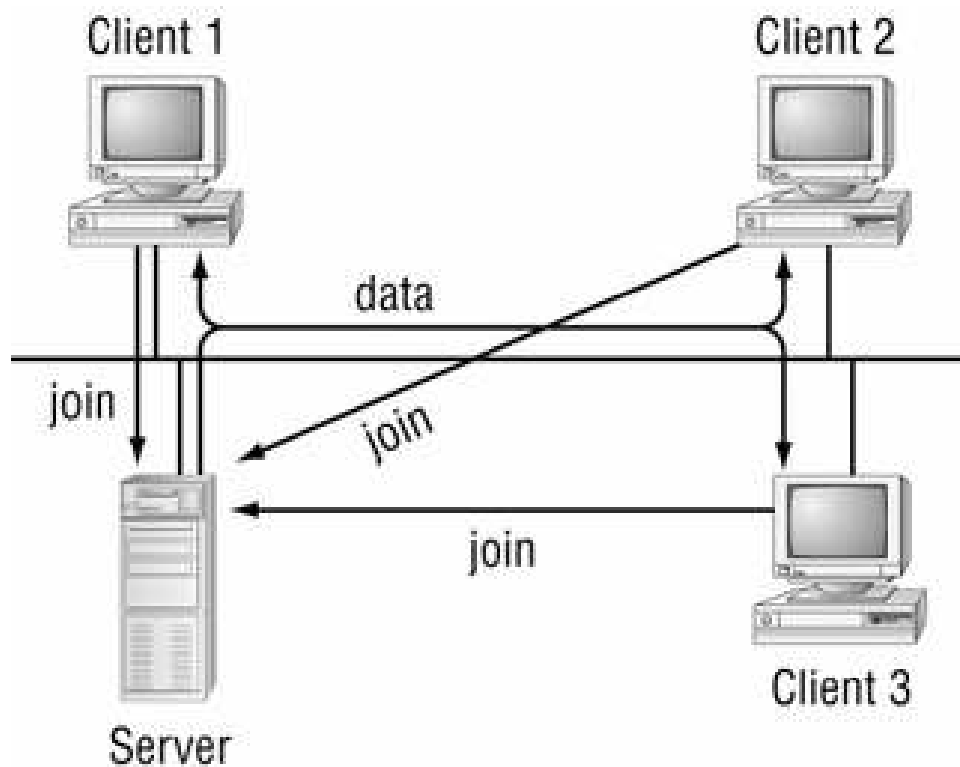
224.0.0.5	Các DR trong OSPF
224.0.1.9	Nhóm địa chỉ RIPv2
224.0.1.24	Nhóm địa chỉ WINS server

Có 2 kỹ thuật Multicast được sử dụng

+ Peer to Peer



+ Central Server



2.6.4. Socket Multicasting trong .Net

- Sử dụng phương thức SetSocketOption()

- Socket option có thể được sử dụng để
 - ❑ Thêm một Socket vào nhóm Multicast
 - ❑ Loại một Socket khỏi nhóm Multicast
 - ❑ SetSocketOption(SocketOptionLevel,SocketOptionName, optionValue)
 - ❑ SocketOptionName
 - AddMembership
 - DropMembership
 - Sử dụng phương thức SetSocketOption()
 - Socket option có thể được sử dụng để
 - ❑ optionValue là một đối tượng của lớp MulticastOption
- ```
MulticastOption(IPAddress) MulticastOption(IPAddress,IPAddress)
```
- Ví dụ thêm một Socket vào nhóm Multicast 224.100.0.1
 

```
sock.SetSocketOption(SocketOptionLevel.IP,
SocketOptionName.AddMembership, new
MulticastOption(IPAddress.Parse("224.100.0.1"));
```

#### **Gửi dữ liệu Multicast**

```
class MultiSend{
 public static void Main() {
 Socket server = new Socket(AddressFamily.InterNetwork,
 SocketType.Dgram, ProtocolType.Udp);
 IPEndPoint iep = new IPEndPoint(IPAddress.Parse("224.100.0.1"), 9050);
 byte[] data = Encoding.ASCII.GetBytes("This is a test message");
 server.SendTo(data, iep);
 server.Close();
 }
}
```

#### **Nhận dữ liệu Multicast**

```
class MultiRecv{
 public static void Main() {
 Socket sock = new Socket(AddressFamily.InterNetwork, SocketType.Dgram,
 ProtocolType.Udp);
 Console.WriteLine("Ready to receive...");
 IPEndPoint iep = new IPEndPoint(IPAddress.Any, 9050);
 EndPoint ep = (EndPoint)iep;
 sock.Bind(iep);
 sock.SetSocketOption(SocketOptionLevel.IP,
 SocketOptionName.AddMembership,
 new MulticastOption(IPAddress.Parse("224.100.0.1")));
 byte[] data = new byte[1024];
 int recv = sock.ReceiveFrom(data, ref ep);
 string stringData = Encoding.ASCII.GetString(data, 0, recv);
 Console.WriteLine("received: {0} from: {1}", stringData, ep.ToString());
 sock.Close();
 }
}
```

## Gửi dữ liệu Multicast với TTL

```
class NewMultiSend{
 public static void Main() {
 Socket server = new Socket(AddressFamily.InterNetwork,
 SocketType.Dgram, ProtocolType.Udp);
 IPEndPoint iep = new IPEndPoint(IPAddress.Any, 9051);
 IPEndPoint iep2 = new IPEndPoint(IPAddress.Parse("224.100.0.1"), 9050);
 server.Bind(iep);
 byte[] data = Encoding.ASCII.GetBytes("This is a test message");
 server.SetSocketOption(SocketOptionLevel.IP,
 SocketOptionName.AddMembership,
 new MulticastOption(IPAddress.Parse("224.100.0.1")));
 server.SetSocketOption(SocketOptionLevel.IP,
 SocketOptionName.MulticastTimeToLive, 50);
 server.SendTo(data, iep2);
 server.Close();
 }
}
```

### ■ Multicast với lớp UdpClient

- ☐ JoinMulticastGroup()

- ☐ DropMulticastGroup()

### ■ JoinMulticastGroup() là phương thức overload

- ☐ JoinMulticastGroup(IPAddress)

- ☐ JoinMulticastGroup(IPAddress, int)

```
class UdpClientMultiSend{
 public static void Main() {
 UdpClient sock = new UdpClient();
 IPEndPoint iep = new IPEndPoint(IPAddress.Parse("224.100.0.1"), 9050);
 byte[] data = Encoding.ASCII.GetBytes("This is a test message");
 sock.Send(data, data.Length, iep);
 sock.Close();
 }
}

class UdpClientMultiRecv
{
 public static void Main()
 {
 UdpClient sock = new UdpClient(9050);
 Console.WriteLine("Ready to receive...");
 sock.JoinMulticastGroup(IPAddress.Parse("224.100.0.1"), 50);
 IPEndPoint iep = new IPEndPoint(IPAddress.Any, 0);
 byte[] data = sock.Receive(ref iep);
 string stringData = Encoding.ASCII.GetString(data, 0, data.Length);
 Console.WriteLine("received: {0} from: {1}", stringData, iep.ToString());
 sock.Close();
 }
}
```

## 2.7 Bài tập áp dụng

```
class MulticastChat : Form{
 TextBox newText;
 ListBox results;
 Socket sock;
 Thread receiver;
 IPEndPoint multiep = new IPEndPoint(IPAddress.Parse("224.100.0.1"), 9050);
 public MulticastChat() {
 Text = "Multicast Chat Program";
 Size = new Size(400, 380);
 Label label1 = new Label();
 label1.Parent = this;
 label1.Text = "Enter text string:";
 label1.AutoSize = true;
 label1.Location = new Point(10, 30);
 newText = new TextBox();
 newText.Parent = this;
 newText.Size = new Size(200, 2 * Font.Height);
 newText.Location = new Point(10, 55);
 results = new ListBox();
 results.Parent = this;
 results.Location = new Point(10, 85);
 results.Size = new Size(360, 18 * Font.Height);
 Button sendit = new Button();
 sendit.Parent = this;
 sendit.Text = "Send";
 sendit.Location = new Point(220, 52);
 sendit.Size = new Size(5 * Font.Height, 2 * Font.Height);
 sendit.Click += new EventHandler(ButtonSendOnClick);
 Button closeit = new Button();
 closeit.Parent = this;
 closeit.Text = "Close";
 closeit.Location = new Point(290, 52);
 closeit.Size = new Size(5 * Font.Height, 2 * Font.Height);
 closeit.Click += new EventHandler(ButtonCloseOnClick);
 sock = new Socket(AddressFamily.InterNetwork, SocketType.Dgram,
 ProtocolType.Udp);
 IPEndPoint iep = new IPEndPoint(IPAddress.Any, 9050);
 sock.Bind(iep);
 sock.SetSocketOption(SocketOptionLevel.IP,
 SocketOptionName.AddMembership,
```

```

 new MulticastOption(IPAddress.Parse("224.100.0.1")));
receiver = new Thread(new ThreadStart(packetReceive));
receiver.IsBackground = true;
receiver.Start();
}
void ButtonSendOnClick(object obj, EventArgs ea) {
 byte[] message = Encoding.ASCII.GetBytes(newText.Text);
 newText.Clear();
 sock.SendTo(message, SocketFlags.None, multiep);
}
void ButtonCloseOnClick(object obj, EventArgs ea) {
 receiver.Abort();
 sock.Close();
 Close();
}
void packetReceive() {
 EndPoint ep = (EndPoint)multiep;
 byte[] data = new byte[1024];
 string stringData;
 int recv;
 while (true) {
 recv = sock.ReceiveFrom(data, ref ep);
 stringData = Encoding.ASCII.GetString(data, 0, recv);
 results.Items.Add("from " + ep.ToString() + ": " + stringData);
 }
}
public static void Main() {
 Application.Run(new MulticastChat());
}
}

```