

بِسْمِ اللّٰهِ الرَّحْمٰنِ الرَّحِيْمِ

# C# Sharp Network Programming

By SUDIPTO MEGAMOLK

## فهرست

۴	..... مقدمه
۵	..... تعریف Socket
۶	..... برنامه نویسی اتصال گرای (TCP)
۱۲	..... نحوه ی ساختن کانکشن مجازی
۱۴	..... ساختن سوکت(Socket)
۲۱	..... برنامه کاربردی Client , Server
۲۵	..... برنامه نویسی بدون اتصال (Connectionless Sockets)
۳۱	..... پارامتر های ( ) SetSocketOption
۴۱	..... مدیریت Error های چندگانه
۴۵	..... چیست؟ Broad Cast
۵۰	..... MultiCast
۵۶	..... SMTP
۵۶	..... کار با برنامه Microsoft Virtual PC
۵۴	..... نحوه ی ایجاد کردن شبکه
۵۸	..... Mail Server ایجاد
۶۱	..... برنامه های ارسال Email
۷۵	..... استفاده از Sockets Helper Classes
۷۸	..... برنامه دریافت Email(POP3)
۸۵	..... Remoting

نام کتاب : C Sharp Network Programming



موضوع : برنامه نویسی شبکه



مترجم : سعید اصغری



ایمیل : Single1990@gmail.com

افرادی که از این کتاب برای یادگیری برنامه نویس تحت شبکه استفاده می کنند باید با مقدمات زبان C# آشنایی داشته باشند برای این منظور می توانید از کتاب (آموزش ویژوال C# ۲۰۰۵) گردآوری شده توسط سید محمد هاشمیان استفاده کنید. (مراجعه به اینترنت)

تمامی کدهای نوشته شده در این کتاب در محیط Visual Studio 2008 نوشته شده است این کتاب از روی کتاب Richard Blum ترجمه شده و البته ترجمش خالی از اشکال نیست.

البته این نکته رو هم بگم که در بعضی از قسمت های این کتاب نتونستم معادل فارسی برای بعضی از جمله ها رو پیدا کنم به خاطر همین از همون En استفاده کرده ام.

اگر کمی توضیحات برنامه ها گیج کنند ه یا قابل فهم نمی باشد نگران نباشید ، به مرور زمان به این اصطلاحات و نحوی عملکرد دستورات آشنا می شوید.

برای فهمیدن بعضی از قسمت ها توضیحات اضافی نوشتم، در بعضی از قسمت ها هم کد اصلی رو تغییر دادم تا راحت تر متوجه بشید.(قسمت های اضافه شده با {شروع و با} پایان داده شده )

نکته : بهتر پس از نوشتن برنامه واجرای آنها بار دیگر برنامه را Trace کرده و با قرار داد کرسر ماوس بر روی هر متغیر ، شی و ... مقادیری را که دریافت می کنند را مشاهده کنید این کار با عث می شود که با عملکرد هر یک از قسمت ها بهتر آشنا شوید

## Socket { تعریف }

در شبکه های کامپیو تری ، برنامه های متعددی در یک زمان با یکدیگر مرتبط می گردند.

زمانیکه چندین برنامه بر روی یک کامپیوتر فعال می گردند پروتکل TCP/IP می بایست از روشی به منظور تمایز یک برنامه از برنامه دیگر استفاده نماید بدین منظور از سوکت برای مشخص نمودن یک برنامه خاص استفاده می گردد.

### سوکت (Socket)

سوکت ، ترکیبی از یک آدرس IP و پورت TCP (اتصال گرا) و یا پورت UDP (بدون اتصال) است. یک برنامه ، سوکتی را با مشخص نمودن آدرس IP مربوط به کامپیوتر و نوع سرویس (TCP یا UDP) و پورتی که نشان دهنده برنامه است ، مشخص می نماید آدرس IP موجود در سوکت امکان آدرس دهی کامپیوتر مقصد را فراهم و پورت مربوطه ، برنامه ای را که داده ها برای آن ارسال می شود را مشخص می کند.

در پروتکل TCP برای به رسمیت شناختن پروتکل های مختلفی که بر روی یک ماشین در حال اجرا هستند راه حل زیر ارائه می شود

### Port Number

هر پروسه ای برای تقاضای برقراری ارتباط با پروسه ای دیگر روی شبکه ، یک شماره شناسایی برای خود بر می گزیند. به این شماره شناسایی آدرس پورت Port Number گفته می شود (برای در ک بهتر این موضوع می توانید به کتاب مرجع کامل ضد امنیت شبکه توسط امیر آشتیانی

مراجعه کنید}

## برنامه نویسی اتصال گرا (TCP)

مزیت بزرگ کتابخانه .net. داشتن IP آدرس و Port ها و به کار بردن Handle هاست دو کلاس در System.Net (NameSpace) وجود دارد. که دارای نوع هایی مختلف از IP Address های Handle می باشد.

**IPAddress-۱**

**IPEndPoint-۲**

شی IPAddress برای نمایش دادن یک IP Address منفرد می باشد. این شی دارای متدهای مختلفی برای نمایش IP Address می باشد. سازنده (Constructor) پیش فرض شی IP Address به صورت زیر می باشد.

**Public IPAddress (Long address)**

شی IPAddress دارای متدهای مختلفی به شرح زیر می باشد.

METHOD	DESCRIPTION
Equals	دو تا IP را با هم مقایسه می کند.
GetHashCode	مقدار درهم شده ای (Hash) را برای شی IPAddress برمی گرداند.
GetType	نمونه ای از یک نوع IPAddress داخلی را برمی گرداند
HostToNetworkOrder	بایتهای Host یک IPAddress را به بایتهای آدرس یک شبکه تبدیل می کند.
IsLoopBack	نشان می دهد که آیا IPAddress مطرح شده یک Loopback آدرس است یا نه
NetworkToHostOrder	به بایتهای آدرس یک شبکه را به بایتهای یک Host تبدیل می کند.
Parse	یک رشته را به یک IPAddress تبدیل می کند.
ToString	برای نمایش یک IPAddress به یک رشته استفاده می شود.

Pars() متدى است که اغلب در داخل يك IPAddress استفاده مى شود

```
IPAddress newaddress = IPAddress.Parse("192.168.0.1");
```

این غالب به شما اجازه مى دهد که فرمت String را به يك غالب استاندارد با نقطه جدا کننده تبدیل مى کند.

همچنین کلاس IPAddress برای شما چهار فیلد Readonly برای نمایش IP های خاص ارائه مى دهد.

Any : برای نمایش IP آدرس ها در يك سیستم محلی(Local) استفاده مى شود(برای تست برنامه مى تونید از این استفاده کنید )

Broadcast: برای نمایش يك آی پی Broadcast در يك سیستم محلی

LoopBack (127.0.0.1): برای نمایش آدرس سیستم استفاده مى شود

None: برای نمایش ندادن رابط شبکه در يك سیستم

در ادامه با این پارامتر ها بیشتر آشنا مى شویم.

مثال زیر چگونگی استفاده از فیلد های بالا را نشان مى دهد

```
using System.Net;

namespace test
{
    class Program
    {
        static void Main(string[] args)
        {
            IPAddress test1 = IPAddress.Parse("192.168.0.1");
            IPAddress test2 = IPAddress.Loopback;
            IPAddress test3 = IPAddress.Broadcast;
            IPAddress test4 = IPAddress.Any;
            IPAddress test5 = IPAddress.None;
            IPHostEntry ihe =
Dns.GetHostByName(Dns.GetHostName());
            IPAddress myself = ihe.AddressList[0];
            if (IPAddress.IsLoopback(test2))
                Console.WriteLine("The Loopback address is:{0}",
test2.ToString());
            else
                Console.WriteLine("Error obtaining the loopback address");
```

```
Console.WriteLine("The local IP address is :{0}\n",
myself.ToString());  
  
        if (myself == test2)
Console.WriteLine("The loopback address is the A same as local
address", myself.ToString());
        else
Console.WriteLine("loopback address is not the local
address.\n");  
  
Console.WriteLine("the test address is :{0}" ,
test1.ToString());  
  
Console.WriteLine("broadcast address:{0}" , test3.ToString());
Console.WriteLine("the any address is:{0}" , test4.ToString());  
  
Console.WriteLine("the none address is:{0}" ,
test5.ToString());  
  
    }  
}
```

خروجی به شکل زیر می باشد.

```
The Loopback address is:127.0.0.1
The local IP address is :127.0.0.1
loopback address is not the local address

the test address is :192.168.1.1
broadcast address:255.255.255.255
the any address is:0.0.0.0
the none address is:255.255.255.255
```

متدهای GetHostName و GetHostByName آی پی یک سیستم محلی را با ساختن یک

شی IPHostEntry معین می کند.

ای) است که دارای جزئیات زیادتری می باشد. Object) IPHostEntry

اما برای شروع کافی که خاصیت AddressList را یاد بگیرید. آرایه ای AddressList را می توانیم تمامی IP های یک سیستم را در آن ذخیره کنیم.

در آدرس 0.0.0.0 نمایش داده شده است این آدرس بیشتر زمانی استفاده می شود که سیستم شما دارای یک IP نمی باشد و شما می توانید با استفاده از این روش به سیستم خود یک Null Ip Address آدرس آن 255.255.255.255 می باشد ، وقتی استفاده می شود که سوکت شما ساختگی باشد.

### IPEndPoint

شی IPEndPoint برای نمایش یک IP و Port ترکیب شده با هم استفاده می شود . یک IPEndPoint زمانی استفاده می شود که به یک Socket با آدرس محلی و یا به یک آدرس دور متصل شوید .

دو تا (Constructor) برای ساختن شی IPEndPoint استفاده می شود

```
IPEndPoint(long address, int port)
IPEndPoint(IPAddress address, int port)
```

هر دو سازنده از دو پارامتر یکی IP و دیگری یک عدد صحیح که شماره پورت می باشد استفاده می کنند.

دارای متد های زیر می باشد.

METHOD	DESCRIPTION
Create	یکEndPoint به شکل یک سوکت آدرس را می سازد.
Equals	دو تا IPEndPoint را با هم مقایسه می کند.
GetHashCode	مقدار Hash را برای شی IPEndPoint برمی گرداند.
GetType	یک نمونه از IPEndpoint را بر می گرداند.
Serialize	یک نمونه سوکت آدرس از نوع IPEndPoint را می سازد که حاوی اطلاعاتی راجع IPEndPoint می باشد.
ToString	IPEndPoint را به یک رشته برای نمایش تبدیل می کند.

کلاس System.net : NameSpace کلاس خاصی است که در داخل کلاس SocketAddress وجود دارد.

آن برای نمایش یک نسخه از IPPEndPoint در شی serialized می باشد فرمت کلاس SocketAddress به شکل زیر می باشد .

۱ بایت برای نمایش AddressFamily

۱ بایت برای نمایش اندازه اشیا

۲ بایت برای نمایش شماره پورت اشیا

بایت های باقیمانده هم برای نمایش IP آدرس یک سیستم است

شما می توانید خصوصیات داخلی IPPEndPoint را به شکل زیر تنظیم کنید.

: خصوصیات یک IP را تنظیم (Set) و یا دریافت (Get) می کند AddressFamily

: نام IP آدرس را می گیرد.

Port: شماره پورت TCP یا UDP را تنظیم می کند.

هر خصوصیت می تواند اطلاعاتی در باره قسمت های اختصاصی اشیا را استخراج کند IP و Port جز خصوصیات اختصاصی و داخلی در شی IPPEndPoint به حساب می آیند.

IPPEndPoint دارای دو متده ReadOnly می باشد که شامل

MaxPort : بیشترین مقداری که می توان برای یک پورت تعیین کرد.

MinPort : کمترین مقداری که می توان برای یک پورت تعیین کرد.

به مثال زیر توجه کنید.

```
IPAddress test = IPAddress.Parse("192.168.1.1");
IPEndPoint ie = new IPPEndPoint(test, 8000);
Console.WriteLine("the IPEndPoint is : {0}", ie.ToString());
Console.WriteLine("the Address Family is :{0}",
ie.AddressFamily);
Console.WriteLine("the address is: {0}, And the A port is
:{1}\n", ie.Address, ie.Port);
Console.WriteLine("the min port number is :{0}",
IPEndPoint.MinPort);
Console.WriteLine("the max port number is :
{0}", IPEndPoint.MaxPort);
ie.Port = 80;
Console.WriteLine("The change IPEndPoint value a is:{0}",
ie.ToString());
SocketAddress sa = ie.Serialize();
Console.WriteLine("the socket address is:{0}", sa.ToString());
```

خروجی برنامه به صورت زیر می باشد.

```
the IPEndPoint is : 192.168.1.1:8000
the Address Family is :InterNetwork
the address is: 192.168.1.1, And the A port is :8000
the min port number is :0
the max port number is : 65535
The change IPEndPoint value a is:192.168.1.1:80
the socket address is:InterNetwork:16:{0,80,192,168,1,1,0,0,0,0,0,0,0,0,0}
Press any key to continue....
```

این برنامه همچنین به شما اجازه می دهد بدون آنکه یک شی جدید بسازید مقدار پورت را تغییر

ie.Port=80; دهید .

**{مثال: با توجه به دستوراتی که یاد گرفته اید می خواهیم با دادن آی پی LoopBack به برنامه}**

**نام کامپیوتر را بدست آوریم.**

```
using System.Net;
namespace test2
{
    class Program
    {
        static void Main(string[] args)
        {
IPHostEntry iphost = Dns.GetHostEntry("127.0.0.1");
string hostname = iphost.HostName;
Console.WriteLine(hostname);
        }
    }
}
```

خروجی این دستور بستگی به این دارد که نام کامپیوتر شما چه باشد در اینجا خروجی برابر Media نام کامپیوتری که برنامه را در آن اجرا کرده ام. همچنین برای پی بردن به صحت موضوع کافیه که در Run عبارت CMD را تایپ کرده و در پنجره باز شده عبارت Hostname را تایپ کرده و کلید Enter را فشار دهید تا نام کامپیوتر را مشاهده کنید. البته می توانید با ساخت یک Local Area Connection یا همان کانکشن مجازی به سیستم خود یک IP مثلا 192.168.0.1 اختصاص داده و با استفاده از این IP برنامه خود را تست کنید.

### **{ نحوه ی ساختن یک کانکشن مجازی }**

بر روی Next کلیک کنید در صفحه ی باز شده روی گزینه اول کلیک کنید سپس بر روی Next در صفحه ی بعدی در لیست مربوطه گزینه آخر را انتخاب بر روی Next کلیک کنید سپس در صفحه جدید گزینه دوم را انتخاب کرده بر روی Next کلیک کرده حال در لیست موجود Network Adapters را انتخاب بر روی Next کلیک کرده در این صفحه بر روی MicroSoftLoopBack Adapter را انتخاب کرده و سپس ۲ بار بر روی Next کلیک کنید تا عملیات به پایان برسد همان طور که

مشاهده می کنید در Notifications یک آیکن اضافه شده بر روی آن دابل کلیک کرده در صفحه ای باز شده بر روی Properties کلیک کرده در صفحه باز شده در لیست Internet Protocol(TCP/IP) گزینه This Connection Use the following item را انتخاب کرده و بر روی Properties کلیک کرده در صفحه باز شده بر روی Use the following IP Address مقدار IPAddress کلیک کرده در قسمت SubnetMask مقدار 255.255.255.0 را وارد کرده و در قسمت 192.168.0.1 را وارد کرده بر روی Ok کلیک کرده سپس در پنجره ای اولی گزینه Show Icon in Notification area when Connected را انتخاب و سپس بر روی Ok کلیک کرده و مانند پنجره ها را ببینید. به این ترتیب شما می توانید به جای آدرس 127.0.0.1 کلیک کرده و مانند پنجره ها را ببینید. به این ترتیب شما می توانید به جای آدرس 127.0.0.1 IP جدید خود را وارد کنید.

**نکته:** با ساخت این کانکشن هیچ لزومی نداره که کامپیوتر شما جزء شبکه ای باشه. این قسمت از کد برنامه قبل را به این صورت تغییر دهید.

```
IPHostEntry iphost = Dns.GetHostEntry("192.168.0.1");
```

حال می خواهیم با وارد کردن نام کامپیوتر تمامی IP های فعال سیستم نمایش داده شود.

```
using System;
using System.Text;
using System.Net;

namespace test3
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Enter a Computer Name:");
            IPHostEntry iphost;
            string s;
            try
            {
                s = Console.ReadLine();
                iphost = Dns.GetHostEntry(s);
                IPAddress[] addresses = iphost.AddressList;
                StringBuilder addressList = new StringBuilder();
                foreach (IPAddress address in addresses)
```

```
        {
            addressList.AppendFormat("IP Address: {0};",
address.ToString()));
        }
        Console.WriteLine(addressList);
    }
    catch (Exception)
    {
        Console.WriteLine("Host Not Found... ");
    }
}
```

با اجرای این کد در صورتی نام کامپیوتر به صورت صحیح وارد شود تمامی IP های سیستم توسط یک حلقه از سیستم استخراج می شود چون IP یک شی محسوب می شود با استفاده از حلقه Foreach این کار را انجام می دهیم حال اگر نام کامپیوتر اشتباه وارد شود عبارت Host Not Found نمایش داده می شود.

## ساختن سوکت (Socket)

فضای نام System.net.Socket برای شما رابط سطح پایین Winsock APIs را فراهم می کند. این بخش در C# به صورت کلاس Socket خلاصه شده است.

کلاس Socket در فضای نام System.net.Socket قرار دارد، C# مدیریت کدهای Winsock APIs (کسانی با C برنامه نوشته اند!) را فراهم می کند.

سازنده کلاس Socket به شرح زیر است.

Socket(AddressFamily *af*, SocketType *st*,ProtocolType *pt*)

فرمت پایه ای Socket مشابه تابع () در Unix می باشد

پارامتر هایی که نوع ساختن یک سوکت را مشخص می کند به شرح زیر است.

نوع شبکه را معین می کند: AddressFamily

نوع ارتباط را دادها معین می کند: SocketType

نوع یک پروتکل شبکه را مشخص می کند: ProtocolType

برای IP نرمال باید از مقدار AddressFamily.InterWork استفاده کنید منظور از IP نرمال

IP version 4 (32 بیتی) می باشد.

پارامتر های SocketType در جدول زیر توضیح داده شده است.

SOCKETTYPE	PROTOCOLTYPE	DESCRIPTION
Dgram	Udp	بیانگر یک ارتباط بدون اتصال
Stream	Tcp	بیانگر یک ارتباط اتصال گرا
Raw	Icmp	پروتکل کنترل پیغام در اینترنت
Raw	Raw	Plain IP packet communication

: مثال

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

نکته: هر SocketType باید با ProtocolType متناظر خود به کار گرفته شود.

در جدول زیر چندین مشخصه Socket که یک سوکت را می سازد نشان داده شده است.

## خصوصیات Socket

PROPERTY	DESCRIPTION
AddressFamily	دريافت AddressFamily سوکت(كه دارای انواع مختلفی می باشد که در اينجا ما فقط از InterNetwork استفاده می کنيم)
Available	دريافت مقداری از داده های آماده خواندن
Blocking	تنظيم و يا دريافت حالت Blocking در سوکت
Connected	مقداری را که نشان دهنده ارتباط سوکت با دستگاه راه دور می باشد را دريافت می کند.
Handle	يک سистем را برای سوکت دريافت می کند
LocalEndPoint	يک شی محلیEndPoint برای سوکت دريافت می کند
ProtocolType	نوع پروتکل يک سوکت را دريافت می کند.
RemoteEndPoint	اطلاعاتEndPoint راه دور را برای سوکت دريافت می کند.
SocketType	نوع سوکت را دريافت می کند.

همه ی خصوصیات سوکت به غیر از RemoteEndPoint و LocalEndPoint پس از ساخت سوکت در دسترس هستند. LocalEndPoint و RemoteEndPoint خصوصیاتی هستند که به واسطه مقید سازی سوکت استفاده می شوند.

مثال زير خصوصیات Socket را نشان می دهد

```
using System.Net;
using System.Net.Sockets;
namespace SockProp
{
    class Program
    {
```

```
static void Main(string[] args)
{
    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("Socket type: {0}", test.SocketType);
Console.WriteLine("ProtocolType: {0}", test.ProtocolType);
test.Blocking=false;
Console.WriteLine("Blocking:{0}", test.Blocking);
Console.WriteLine("Connected:{0}", test.Connected);
test.Bind(ie);
IPEndPoint iep = (IPEndPoint)test.LocalEndPoint;
Console.WriteLine("LocalEndPoint:{0}", iep.ToString());
    test.Close();

}
}
```

خروجی به صورت زیر می باشد

Socket type: Stream

ProtocolType: Tcp

Blocking:False

Connected:False

LocalEndPoint:127.0.0.1:8000

Press any key to continue.....

متده Bind() برای محدود کردن سوکت به آدرس LoopBack سیستم می باشد.

با استفاده از متدهای داخلی سوکت شما می توانید یک ارتباط اتصال گرا به آدرس محلی یک شرکه با یک سمته راه دهنده سازند.

همانطور، که گفته شد Bind() مقدار دن سوکت به یک آدرس استفاده می شود این متدهای

Bind(EndPoint address)

به صورت ویه استفاده م شده

نکته ای، مهم دی IPEndPoint، که شاما یک آدرس و شما هست.

Listen() مه، باشد سی، از اینکه سوکت به یک آدرس محله محدود شد. شما با استفاده از متدها

به خط گوش دهید تا یک Client سعی به برقراری ارتباط نموده.

Listen(int backlog)

شکل کلی این متده است

این پارامتر تعداد ارتباطاتی را که در صف یک سیستم برای انتظار یک سرویس هستند را تعیین می کند.

بعد از listen() سرور آماده دریافت یک تقاضای ارتباط از Client می باشد. این کار توسط متده Accept()

متده Accept() متدى است که حاوی اطلاعاتی درباره یک سوکت جدیده و اینکه همه ی Connection ها برای ارتباط با آن باید آنرا فراخوانی کنند.

به این مثال توجه کنید.

```
IPHostEntry local = Dns.GetHostByName(Dns.GetHostName());
IPEndPoint iep = new IPEndPoint(local.AddressList[0],
8000);
Socket newserver = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);
newserver.Bind(iep);
newserver.Listen(5);
Socket newclient = newserver.Accept();
```

زمانی Client به سرور متصل می شود شی newserver حاوی یک ارتباط جدید که باید از این به بعد در همه ی ارتباطات بعدی با Client استفاده شود.

بعد از اینکه Client تقاضای یک ارتباط را داد . Client و Server می توانند شروع به انتقال اطلاعات نمایند . متدهای Send() و Recive() برای ارسال و دریافت اطلاعات استفاده می شوند.

هر دو متده دارای OverLoad می باشد که به شرح زیر می باشد.

نکته: به توابعی که دارای نام های مشترکی هستند ولی دارای پارامتر های متفاوتی با همدیگر می باشند به این توابع ، توابع Overloaded function می گویند.

METHOD	DESCRIPTION
Receive(byte[] data)	داده ها را دریافت کرده و در یک آرایه از نوع بایت ذخیره می کند.
Receive(byte[] data, SocketFlags sf)	خصوصیات یک سوکت و داده های دریافتی را در یک آرایه ذخیره می کند.
Receive(byte[] data, int size, SocketFlags sf)	خصوصیات یک سوکت ، اندازی داده های دریافتی را در یک آرایه از نوع بایت ذخیره می کند.
Receive(byte[] data, int offset, int size, SocketFlags sf)	خصوصیات یک سوکت ، اندازی داده های دریافتی و اینکه داده ها بر حسب Offset مرتب شده و در آرایه ذخیره می کند.
Send(byte[] data)	داده های موجود در آرایه را ارسال می کند.
Send(byte[] data, SocketFlags sf)	خصوصیات سوکت را تنظیم و داده های موجود در آرایه را ارسال می کند.
Send(byte[] data, int size, SocketFlags sf)	خصوصیات و سایز داده ای ارسالی (بر حسب بایت) را تنظیم و داده های موجود در آرایه را ارسال می کند
Send(byte[] data, int offset, int size, SocketFlags sf)	خصوصیات و سایز داده ای ارسالی (بر حسب بایت) را تنظیم و داده های رشته ای موجود در آرایه را ارسال می کند.

مقادیری که به **SocketFlags** نسبت داده می شود به شرح جدول زیر می باشد.

VALUE	DESCRIPTION
DontRoute	ارسال و دریافت داده بدون استفاده از جدول مسیر یابی
MaxIOVectorLength	یک مقدار عددی برای استفاده از ساختار WSABUF برای ارسال و دریافت داده ها
None	از استفاده نمی کند
OutOfBand	Processes out-of-band data
Partial	قسمتی از پیغام را دریافت و یا ارسال می کند
Peek	فقط به پیغام های وارد شده توجه می کند.

نکته: همچنین شما می توانید به جای استفاده از متدهای **Bind()** و **Connect()** از متدهای **Listen()** و **Accept()** استفاده کنید.

مثال

```
IPAddress host = IPAddress.Parse("192.168.1.1");
IPEndPoint hostep = new IPEndPoint(host, 8000);
Socket sock = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);
sock.Connect(hostep);
```

زمانی که یک ارتباط برقرار شد ، Client می تواند با استفاده از متدهای **Send()** و **Receive()** به ارسال و دریافت اطلاعات به پردازد. اما زمانی که ارتباط به پایان رسید شما باید **Close()** شود. برای این کار دو متدهای **Close()** و **Shutdown()** که این متدهای جلسه شما را متوقف می کنند و متدهای **Close()** و **Shutdown()** که به جلسه شما پایان می دهد.

متدهای دارای پارامترهایی می‌باشد که نحوه‌ی Shutdown شدن Socket را مشخص می‌کند. در جدول زیر این پارامترها توضیح داده شده است.

VALUE	DESCRIPTION
SocketShutdown.Both	از ارسال و دریافت داده‌ها جوگیری می‌کند
SocketShutdown.Receive	از دریافت داده‌ها جلوگیری می‌کند
SocketShutdown.Send	از ارسال داده‌ها جلوگیری می‌کند

مثال

```
sock.Shutdown(SocketShutdown.Both);
sock.Close();
```

یک مثال کاربردی از ارسال و دریافت داده‌ها بین Client و Server با استفاده از روش اتصالگر(یک برنامه چت ساده)

مراحل ساخت برنامه سمت سرور

- ساخت یک سوکت

- مقید کردن سوکت به یک IPPEndPoint محلی

- قرار دادن سوکت در حالت شنود

- دریافت تقاضا برای ارتباط با سوکت

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Net.Sockets;
namespace Server
{
    class Program
    {
        static void Main(string[] args)
        {
            int recv;
            byte[] data = new byte[2048];
            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... ");
// سوکت ساخته شده

// و منتظر دریافت یک تقاضا از سمت کلاینت می باشد

Socket client = newsock.Accept();
IPEndPoint clientEP = (IPEndPoint)client.RemoteEndPoint;

Console.WriteLine("Connected with {0} at port
{1}", clientEP.Address, clientEP.Port);
//
string welcome = "Welcome to my test server";
data = Encoding.ASCII.GetBytes(welcome);
client.Send(data, data.Length, SocketFlags.None);
// سرور پس از دریافت یک تقاضا از کلاینت یک پیغام خوشامد به کلاینت می فرستد

while (true)
{
    // در یافت کرده اگر طول آن کمتر از صفر نبود آنرا چاپ می کند

    data = new byte[1024];
    recv = client.Receive(data);
    if (recv == 0)
        break;
    Console.WriteLine(Encoding.ASCII.GetString(data, 0,
recv));
    // در این قسمت سرور پس از دریافت پیغام همان پیغام ارسال شده بیان میگرداند بدین صورت کلاینت
    // مطمئن می شود که داده هایش رسیده است
    client.Send(data, recv, SocketFlags.None);
}

Console.WriteLine("Disconnected from
{0}", clientEP.Address);
client.Close();
newsock.Close();
}
}

```

توضیح کلی برنامہ

در ابتدای آرایه از نوع بایت تعریف شده که پیغام های ورودی و خروجی در آن ذخیره می گردد چیزی که مهم است، اینست که متدهای `Send` و `Receive` با داده هایی از نوع بایت سر و کاردارند.

وداده ها باید به نحوی به صورت بایت به بایت ارسال و دریافت شوند همه می داده های منتقل شده به سوکت باید به طریقی به نوع بایت تبدیل شوند متده استفاده از Encoding.ASCII با استفاده از قابل دسترس می باشد این متده را به آرایه ای از System.text ، NameSpase بایت ها تبدیل می کند و می تواند این کار را بر عکس نیز انجام دهد .

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

با استفاده از فیلد IPAddress.Any سرور یک تقاضا را روی سیستم شما پیکره بندی می کند اگر شما می خواهید که از یک رابط خاص بسته ها را دریافت کنید می توانید از IP آن رابط استفاده کنید به این صورت

```
IPEndPoint ipep =  
new IPPEndPoint(IPAddress.Parse("192.168.1.6"), 9050);
```

بعد از معین کردن نحوی استفاده از IPEndPoint شما باید یک سوکت بسازید سپس با استفاده از متدهای Bind و Listen برای مقید کردن یک سوکت به یک IPEndPoint از آن استفاده کنید. سرانجام از متده Accept() برای جستجوی درخواست های Client استفاده کنید. متد Accept() یک شی از نوع سوکت بر می گرداند. این همان چیزی است که باید در ارتباط با یک Client استفاده شود بعد از اینکه متد Accept() تقدیم شد، یک سوکت که دارای خاصیتی به نام RemotEndPoint می باشد بر می گرداند این سوکت حاوی IP آدرس شی از نوع RemotEndPoint می باشد که شما باید با استفاده Client از تبدیل مبنا آنرا به IPEndPoint تبدیل کنید.

```
IPEndPoint clientep =(IPEndPoint)client.RemoteEndPoint;
```

بعد از اینکه سوکت شما به Client برقرار شد شما می توانید با استفاده از یک قانون کلی به نقل و انتقال داده ها به پردازید .

این دستور با عث می شود که متدهای `recv = client.Receive(data);` و `Receive` را در یافته کاراکتر های در داخل متغیر `recv` قرار دهد.

این مقدار همان اندازه داده های دریافتی می باشد.

### اما مراحل ساخت برنامه سمت Client

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Net.Sockets;
namespace Client
{
    class Program
    {
        static void Main(string[] args)
        {
            byte[] data = new byte[2048];
            string input, stringData;
            IPEndPoint ipep = new IPEndPoint(
                IPAddress.Parse("127.0.0.1"), 9050);
            Socket server = new
            Socket(AddressFamily.InterNetwork, SocketType.Stream,
            ProtocolType.Tcp);
            try
            {
                server.Connect(ipep);
            }
            catch (SocketException e)
            {
                Console.WriteLine("Unable to connect to server.");
                Console.WriteLine(e.ToString());
                return;
            }
            // کلاینت پیام خوش آمد گویی سرور را چاپ می کند
            int recv = server.Receive(data);
            stringData = Encoding.ASCII.GetString(data, 0, recv);
            Console.WriteLine(stringData);

            while (true)
            {
                // کلاینت متنی را از صفحه کلید خوانده و آنرا به سرور ارسال می کند

                input = Console.ReadLine();
                if (input == "exit")
                    break;
                server.Send(Encoding.ASCII.GetBytes(input));
            }
        }
    }
}
```

پس از ارسال متن به سرور ، سرور دوباره همان پیام را به کلاینت ارسال می کنداین کار // باعث می شود کلاینت از ارسال پیام خود به سرور مطمئن شود در صورتی که کلاینت عبارت خروج را تایپ کند برنامه خاتمه می یابد

```
        data = new byte[1024];
        recv = server.Receive(data);
        stringData = Encoding.ASCII.GetString(data, 0, recv);
        Console.WriteLine(stringData);
    }
    Console.WriteLine("Disconnecting from server...");
    server.Shutdown(SocketShutdown.Both);
    server.Close();
}
}
```

### بر نامه های بیرون اتصال (Connectionless Sockets)

برنامه نویس دو تا کار باید برای برنامه سمت سرور که ارسال و دریافت داده را انجام می دهد بکار گیرد.

- ## ۱- ساخت یک شرکت نویسندگان سوکت Socket

- مقدار دن سوکت به یک IPEndPoint

بعد از این دو کار ، سوکت می تواند به ارسال و دریافت داده بپردازد اما شما برای ارسال و دریافت داده ها نمی توانید از متدهای `Send()` و `Receive()` استفاده کنید بلکه باید از دو متدهای جدید `SendTo()` و `Receivefrom()` استفاده کنید .

متدها را به شکل آریه ای از بایت ها به یک EndPoint توسط یک متغیر ارسال م کنند.

شکا کا متابعہ صورت نہ راست

```
SendTo(byte[] data,int Offset,int Size,SocketFlags  
Flags EndPoint Remote)
```

اما متدهای Receivefrom() و Sendto() برای دریافت داده ها ارسال شده استفاده می شود غالب کلی این متدها صورت زیر می باشد.

```
Receive From(byte[] data, ref Endpoint Remote)
```

برنامه زیر یک برنامه سمت سرور که با استفاده از پروتکل UDP پیاده سازی شده است

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Net.Sockets;
namespace SimpleUdpSrvr
{
    class Program
    {
        static void Main(string[] args)
        {
            int recv;
            byte[] data = new byte[1024];
            IPEndPoint ipep = new IPEndPoint(IPAddress.Any, 9050);
            Socket newsock = new Socket(AddressFamily.InterNetwork,
            SocketType.Dgram, ProtocolType.Udp);

            newsock.Bind(ipep);
            Console.WriteLine("Waiting for a client...");
            IPEndPoint sender = new IPEndPoint(IPAddress.Any, 0);
            EndPoint Remote = (EndPoint)(sender);

            recv = newsock.ReceiveFrom(data, ref Remote);

            Console.WriteLine("Message received from {0}:",
            Remote.ToString());

            Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));
            string welcome = "Welcome to my test server";
            data = Encoding.ASCII.GetBytes(welcome);
            newsock.SendTo(data, data.Length, SocketFlags.None, Remote);

            while (true)
            {
                data = new byte[1024];
                recv = newsock.ReceiveFrom(data, ref Remote);
                Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));
                newsock.SendTo(data, recv, SocketFlags.None, Remote);
            }
        }
    }
}

```

تمام مراحل برنامه مثل برنامه Client و Server ای که با پروتکل Tcp نوشته ایم.

به غیر از تغییر `.SocketType.Dgram, ProtocolType.Udp`

نکته: زمانی شما از یک پورت برای ارتباط استفاده می کنید باید دقت داشته باشید این پورت

توسط برنامه دیگری استفاده نشده باشد برای فهمیدن این موضوع شما می توانید با استفاده از

دستور `Netstat -a` ، وضعیت پورتها را مشاهده کنید این دستور را باید در CommandPrompt وارد کرده اگر این دستور را قبل از اجرای برنامه وارد کنید لیستی از پورتهای مورد استفاده و پورتهایی که در وضعیت شنود قرار دارند را مشاهده می کنید در صورتی که پورت مورد استفاده در برنامه شما (9050) در این لیست وجود نداشت بدین معنا می باشد که این پورت آزاد می باشد و شما می توانید از آن استفاده کنید حال اگر برنامه را اجرا کنید و بعد از دستور `Netstat -a` استفاده کنید مشاهده می کنید که پورت مورد استفاده در برنامه شما به صورت زیر ظاهر می شود.

`Ud p 9050 :*.*`

برنامه سمت Client

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Net.Sockets;
namespace SimpleUdpClient
{
    class Program
    {
        static void Main(string[] args)
        {
            byte[] data = new byte[1024];
            string input, stringData;
IPEndPoint ipep = new IPPEndPoint(IPAddress.Parse("127.0.0.1"),
9050);
            Socket server = new
Socket(AddressFamily.InterNetwork, SocketType.Dgram,
ProtocolType.Udp);
            string welcome = "Hello, are you there?";
            data = Encoding.ASCII.GetBytes(welcome);
server.SendTo(data, data.Length, SocketFlags.None, ipep);
IPEndPoint sender = new IPPEndPoint(IPAddress.Any, 0);
            EndPoint Remote = (EndPoint)sender;
            data = new byte[1024];
            int recv = server.ReceiveFrom(data, ref Remote);
Console.WriteLine("Message received from {0}:", 
Remote.ToString());
            Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));
            while (true)
```

```
        {
            input = Console.ReadLine();
            if (input == "exit")
                break;
        server.SendTo(Encoding.ASCII.GetBytes(input), Remote);
        data = new byte[1024];

            recv = server.ReceiveFrom(data, ref Remote);
        stringData = Encoding.ASCII.GetString(data, 0, recv);
        Console.WriteLine(stringData);
    }
    Console.WriteLine("Stopping client");
    server.Close();
}
}
```

برای اجرا ابتدا برنامه Server و سپس برنامه Client را اجرا کنید.

نکته : زمانی که شما در برنامه Client سایز بافر داده های ارسالی خود را کم در نظر بگیرید برنامه شما نمی تواند داده هایی بیش از ظرفیت با فر را ارسال کند و برنامه شما با خطأ متوقف می شود. برنامه زیر در ک موضوع را برای شما ساده می کند.

شما این برنامه Client جدید را با برنامه [UDP Server](#) قبلي تست کنيد

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
class BadUdpClient
{
    public static void Main()
    {
        byte[] data = new byte[30];
        string input, stringData;
        IPEndPoint ipep = new IPEndPoint(
            IPAddress.Parse("127.0.0.1"), 9050);
        Socket server = new Socket(AddressFamily.InterNetwork,
            SocketType.Dgram, ProtocolType.Udp);
        string welcome = "Hello, are you there?";
        data = Encoding.ASCII.GetBytes(welcome);

server.SendTo(data, data.Length, SocketFlags.None, ipep);
        IPEndPoint sender = new IPEndPoint(IPAddress.Any, 0);
        EndPoint tmpRemote = (EndPoint)sender;
        data = new byte[30];
        int recv = server.ReceiveFrom(data, ref tmpRemote);
Console.WriteLine("Message received from {0}:",
tmpRemote.ToString());
```

```
Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));
    while (true)
    {
        input = Console.ReadLine();
        if (input == "exit")
            break;
    server.SendTo(Encoding.ASCII.GetBytes(input), tmpRemote);
        data = new byte[30];
        recv = server.ReceiveFrom(data, ref tmpRemote);
    stringData = Encoding.ASCII.GetString(data, 0, recv);
        Console.WriteLine(stringData);
    }
    Console.WriteLine("Stopping client");
    server.Close();
}
```

در این برنامه زمانی که شما در برنامه خود ۳۵ کاراکتر به بالا وارد کنید برنامه شما با خطای مواجه می‌شود و داده‌های شما گم می‌شود اگرچه شما نمی‌توانید داده‌هایی که بیشتر از سایز بافر ارسال و گم شده اند را بازیابی کنید اما می‌توانید آنها را مدیریت کنید به این صورت که اگر داده‌های ارسالی شما دارای سایزی بیشتر از سایز بافر بود شما نیز می‌توانید سایز بافر را افزایش داده این افزایش برای بار دوم صورت می‌پذیرد. برای مثال اگر دفعه‌ی اول ۳۵ کاراکتر فرستادید داده‌های شما گم می‌شود حال اگر دفعه‌ی دوم نیز همان ۳۵ کاراکتر را فرستادید داده‌های شما ارسال می‌شود در واقع سایز بافر شما متناسب با ورودی افزایش پیدا می‌کند البته شما می‌توانید همان دفعه‌ی اول سایز بافر خود را افزایش دهید. به برنامه زیر توجه کنید.

شما این برنامه Client جدید را با برنامه [UDP Server](#) قبلی تست کنید.

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
class BetterdUdpClient
{
    public static void Main()
    {
        byte[] data = new byte[30];
        string input, stringData;
        IPEndPoint ipep = new IPEndPoint(
            IPAddress.Parse("127.0.0.1"), 9050);
        Socket server = new Socket(AddressFamily.InterNetwork,
            SocketType.Dgram, ProtocolType.Udp);
        string welcome = "Hello, are you there?";
        data = Encoding.ASCII.GetBytes(welcome);
server.SendTo(data, data.Length, SocketFlags.None, ipep);
        IPEndPoint sender = new IPEndPoint(IPAddress.Any, 0);
        EndPoint tmpRemote = (EndPoint)sender;
        data = new byte[30];
        int recv = server.ReceiveFrom(data, ref tmpRemote);
        Console.WriteLine("Message received from {0}:", tmpRemote.ToString());
```

```
Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));
    int i = 30;
    while (true)
    {
        input = Console.ReadLine();
        if (input == "exit")
            break;
        int a;
        a = input.Length;
server.SendTo(Encoding.ASCII.GetBytes(input), tmpRemote);
        data = new byte[i];
        try
        {
recv = server.ReceiveFrom(data, ref tmpRemote);
stringData = Encoding.ASCII.GetString(data, 0, recv);
        Console.WriteLine(stringData);
        }
        catch (SocketException)
        {
Console.WriteLine("WARNING: data lost, retry message.");
        int v;
        v = a - i;
        i +=v ;
        }
    }
    Console.WriteLine("Stopping client");
server.Close();
}
}
```

این برنامه به این صورت می باشد که اگر سایز بافر شما ۳۰ وسایز داده های شما ۴۷ باشد تفاضل بین ۳۰ و ۴۷ را بدست می آورد ۴۷-۳۰ وسپس مقدار بدست آمده را با بافر داده که ۳۰ می باشد جمع کرد و بین صورت در بار دوم شما می توانید داده هایی را با سایز ۴۷ یا همان ۴۷ کاراکتر را ارسال کنید.

## جلو گیری از گم شدن بسته های ارسالی

بعض ارتباطات UDP می باشد زیرا UDP یک پروتکل بدون ارتباط می باشد و راهی برای اینکه بفهمیم بسته ما به درستی ارسال شده وجود ندارد. به عنوان مثال بسیاری از بازی های رایانه ای برای ارسال اطلاعات به پروتکل UDP وابسته هستند (بازی های تحت شبکه) در مدت زمانی کوتاهی بازی کنندگان موقعیت و وضعیت خود را برای سایر بازی کنندگان ارسال می کنند اگر یک بسته در شبکه گم شود آن بسته دوباره در مدت کوتاهی ارسال می شود.

برای ارسال مجدد در برنامه های UDP دو روش وجود دارد.

### ۱- Asynchronous sockets and a Timer object

که این موضوع مورد بحث نمی باشد

و دیگری

### ۲-Synchronous sockets and setting a socket time-out value

زمانی که برنامه شما از متدهای ReceiveFrom() استفاده می کند هیچ تضمینی برای دریافت داده ها وجود ندارد شما می توانید یک زمان وقفه برای دریافت داده ها تعیین کنید که این کار را می توانید با استفاده از متدهای SetSocketOption() انجام دهید.

غالب این متدهای شکل زیر می باشد

```
SetSocketOption(SocketOptionLevel so, SocketOptionName sn, int value)
```

که هر کدام از خصوصیات آن در جدول زیر توضیح داده شده است

#### پارامتر های SocketOptionLevel

VALUE	DESCRIPTION
IP	تنظیمات IP برای سوکت ها
Socket	تنظیمات برای سوکت
Tcp	تنظیمات برای سوکت TCP
Udp	تنظیمات برای سوکت UDP

#### پارامتر های SocketOptionName

VALUE	SOCKETOPTIONLEVEL	DESCRIPTION
AcceptConnection	Socket	اگر True باشد سوکت در حال گوش دادن است.
AddMembership	IP	به گروهای کاری، یک ip اضافه می کند.
AddSourceMembership	IP	اتصال به منابع یک گروه
BlockSource	IP	داده های یک منبع را Block می کند.
Broadcast	Socket	اگر True باشد به ارسال پیغام های

		اجازه عبور می دهد.
BsdUrgent	IP	از داده های ضروری استفاده می کند فقط یک مرتبه می توان آن را سرت کرد و دیگر نمی توانید آنرا تغییر دهید.
ChecksumCoverage	Udp	می توانید مقدار Checksum را برای آن تنظیم یا از آن دریافت کنید.
Debug	Socket	اگر True باشد اطلاعات رکورد را Debug می کند.
DontFragment	IP	فیلد Fragment را در بسته های IP استفاده نمی کند.
DontLinger	Socket	سوکت را بدون انتظار برای دریافت داده ها می بندد.
DontRoute	Udp	بسته ها را به طور مستقیم به آدرس رابط ارسال می کند.
DropMembership	IP	یک IP را از گروه کاری Drops (حذف) می کند.
DropSourceMembership	IP	یک منبع را از گروه کاری Drops می کند.
Error	Socket	Error های یک وضعیت را مشخص می کند.
ExclusiveAddressUse	Socket	یک سوکت را برای دسترسی انحصاری فعال می کند.
Expedited	IP	برای تسريع داده ها استفاده می شود فقط یک مرتبه می توان آن را سرت کرد و دیگر نمی توانید آنرا تغییر دهید.
HeaderIncluded	IP	داده های ارسال شده به یک سوکت را زمانی که IP شامل هدر باشد نشان می دهد.
IPOptions	IP	تنظیمات یک IP خاص، برای بسته های دورازدسترس
IpTimeToLive	IP	زمان زنده ماندن یک بسته IP را تنظیم

		می کند (Time-to -live)
KeepAlive	Socket	TCp زمان زنده ماندن بسته رانگه می دارد.
Linger	Socket	بعد از بستن سوکت منظر داده های اضافی می باشد.
MaxConnections	Socket	حداکثر طول صف استفاده شده را تنظیم می کند.
MulticastInterface	IP	رابط را برای استفاده بسته های MultiCast تنظیم می کند.
MulticastLoopback	IP	LoopBack IP مالتی کست
MulticastTimeToLive	IP	زمان زنده بودن ip مالتی کست را تنظیم می کند .
NoChecksum	Udp	Checksum را با UDP بسته های صفر ارسال می کند.
NoDelay	Tcp	الگوریتم Nagle را برای بسته های Tcp غیر فعال می کند.
OutOfBandInline	Socket	اجازه می دهد داده ها خارج از باند دریافت شود.
PacketInformation	IP	اطلاعاتی درباره دریافت بسته بر می گردد.
ReceiveBuffer	Socket	مجموع بافر را برای بسته های دریافت شده تنظیم می کند.
ReceiveLowWater	Socket	Receives low water mark
ReceiveTimeout	Socket	زمان وقفه را دریافت می کند
ReuseAddress	Socket	به سوکت اجازه می دهد که پیش از استفاده به یک پورت مقید شود.
SendBuffer	Socket	مجموع بافر های رزرو شده ، برای ارسال بسته ها را تنظیم می کند.
SendLowWater	Socket	Sends low water mark
SendTimeout	Socket	مقدار زمان، به پایان رسیده را ارسال می کند.
Type	Socket	نوع سوکت را دریافت می کند.
TypeOfService	IP	نوع فیلد سرویس را برای ip تنظیم

		می کند.
UnblockSource	IP	سوکت را در حالت Non-Blocking تنظیم می کند.
UseLoopback	Socket	زمانی که گذرگاه فرعی شبکه قابل استفاده باشد..

و پارامتر Value مقداری را برای نام سوکت تنظیم می کند.

برای تعیین این زمان(وقفه) شما باید SetSocketOption را به صورت زیر تعریف کنید.

```
server.SetSocketOption(SocketOptionLevel.Socket,
SocketOptionName.ReceiveTimeout, 3000)
```

متده استفاده SetSocketOption از کلاس Socket منشا می شود بنابراین شما باید شی از نوع Socket را مورد استفاده قرار دهید که در اینجا این شی Server می باشد عددی که بر حسب Integer بیان شده (3000) زمان بر حسب میلی ثانیه می باشد تا دادها قبل از این زمان به سوکت برسد . برنامه زیر نحوه استفاده از متده SetSocketOption را نشان می دهد.

```
using System;
using System.Text;
using System.Net;
using System.Net.Sockets;
namespace TimeoutUdpClient
{
    class Program
    {
        static void Main(string[] args)
        {
            byte[] data = new byte[1024];
            string input, stringData;
            int recv;
            IPEndPoint ipep = new IPEndPoint(
                IPAddress.Parse("127.0.0.1"), 9050);
            Socket server = new
            Socket(AddressFamily.InterNetwork,
            SocketType.Dgram, ProtocolType.Udp);
```

```
    int sockopt =
(int)server.GetSocketOption(SocketOptionLevel.Socket, SocketOptionName.ReceiveTimeout);

Console.WriteLine("Default timeout: {0}", sockopt);
server.SetSocketOption(SocketOptionLevel.Socket,
SocketOptionName.ReceiveTimeout, 3000);

sockopt =
(int)server.GetSocketOption(SocketOptionLevel.Socket, SocketOptionName.ReceiveTimeout);

Console.WriteLine("New timeout: {0}", sockopt);
string welcome = "Hello, are you there?";
data = Encoding.ASCII.GetBytes(welcome);
server.SendTo(data, data.Length, SocketFlags.None, ipep);
IPEndPoint sender = new IPEndPoint(IPAddress.Any, 0);
EndPoint tmpRemote = (EndPoint)sender;
data = new byte[1024];
recv = server.ReceiveFrom(data, ref tmpRemote);

Console.WriteLine("Message received from
{0}:", tmpRemote.ToString());

Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));
while (true)
{
    input = Console.ReadLine();
    if (input == "exit")
        break;
server.SendTo(Encoding.ASCII.GetBytes(input), tmpRemote);
    data = new byte[1024];
    recv = server.ReceiveFrom(data, ref tmpRemote);
    stringData = Encoding.ASCII.GetString(data, 0, recv);
    Console.WriteLine(stringData);
}
Console.WriteLine("Stopping client");
server.Close();
}
}
```

برنامه در ابتدا مقدار زمان پیش فرض را دریافت کرده و سپس نمایش می دهد.

```
int sockopt =
(int)server.GetSocketOption(SocketOptionLevel.Socket,
SocketOptionName.ReceiveTimeout);
Console.WriteLine("Default timeout: {0}", sockopt);
```

متدهای `GetSocketOption` شی از نوع `Object` بر می گرداند (همان مقدار پیش فرض)

برای اینکه این شی را به یک عدد صحیح تبدیل کنیم از عبارت `int` استفاده کرده ایم.

این برنامه در مدت زمان تعیین شده داده ها را ارسال و دریافت می کند. شما این برنامه Client جدید را با برنامه [UDP Server](#) قبلی تست کنید. بعد از اینکه دو برنامه با هم ارتباط برقرار کرددند شما برنامه سمت سرور با کلید `Ctrl+C` بیندید حال متنه را در برنامه UDP Client نوشته و ارسال کنید می بینید که برنامه شما با خطا متوقف می شود چون نمی تواند با سرور ارتباط برقرار کند شما می توانید این وضعیت را کنترل کنید فقط کافی که کد عبارت برگشتی از سرور را در یک `Try` و پیغام خطای آنرا در یک `Catch` بنویسیم برای حالات دیگر نیز می توانیم این کار را انجام دهیم مثلا زمانی که Client پیغام خوشامد گویی سرور را نتواند به هر دلیلی دریافت کند به برنامه زیر توجه کنید.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Net.Sockets;
namespace ExceptionUdpClient
{
    class Program
    {
        static void Main(string[] args)
        {
            byte[] data = new byte[1024];
            string input, stringData;
            int recv;
            IPEndPoint ipep = new IPEndPoint(
                IPAddress.Parse("127.0.0.1"), 9050);

            Socket server = new
            Socket(AddressFamily.InterNetwork, SocketType.Dgram,
            ProtocolType.Udp);

            int sockopt =
(int)server.GetSocketOption(SocketOptionLevel.Socket,
SocketOptionName.ReceiveTimeout);

            Console.WriteLine("Default timeout: {0}", sockopt);
            server.SetSocketOption(SocketOptionLevel.Socket,
            SocketOptionName.ReceiveTimeout, 3000);
```

```

        sockopt =
(int)server.GetSocketOption(SocketOptionLevel.Socket,
    SocketOptionName.ReceiveTimeout);

    Console.WriteLine("New timeout: {0}", sockopt);
    string welcome = "Hello, are you there?";
    data = Encoding.ASCII.GetBytes(welcome);
server.SendTo(data, data.Length, SocketFlags.None, ipep);
IPEndPoint sender = new IPEndPoint(IPAddress.Any, 0);
EndPoint Remote = (EndPoint)sender;

    data = new byte[1024];
    try
    {
        recv = server.ReceiveFrom(data, ref Remote);
Console.WriteLine("Message received from {0}:",
Remote.ToString());
Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));
    }

    catch (SocketException)
    {

Console.WriteLine("Problem communicating with remote
server");
        return;
    }
    while (true)
    {
        input = Console.ReadLine();
        if (input == "exit")
            break;
server.SendTo(Encoding.ASCII.GetBytes(input), ipep);
        data = new byte[1024];
        try
        {
recv = server.ReceiveFrom(data, ref Remote);
stringData = Encoding.ASCII.GetString(data, 0, recv);
            Console.WriteLine(stringData);
        }
        catch (SocketException)
        {
Console.WriteLine("Error receiving message");
        }
    }
    Console.WriteLine("Stopping client");
server.Close();
}
}
}

```

این برنامه فرقی با برنامه قبلی ندارد بلکه فقط در آن خطاهای مدیریت شده اند اگر برنامه شما نتواند در مدت زمانی که تعیین شده داده ها را دریافت کند پیغام خطای آن نمایش داده می شود.

## تکرار در ارسال داده ها یی که موفق به ارسال نشده اند

دلایل زیادی وجود دارد که بسته UDP شما به مقصد نرسد که شما باید با سعی در ارسال مجدد بسته (یا پیغام ها) بسته را ارسال کنید راه ساده ای که برای انتقال داده وجود دارد این است که یک متند جداگانه برای ارسال و دریافت پیغام ها بسازید.

مراحل ساخت این متند به صورت زیر می باشد.

- ۱- ارسال یک پیغام به Host را دور
  - ۲- مدت زمان انتظار برای دریافت بسته از راه دور
  - ۳- اگر پاسخ را دریافت کردید سایز بسته را دریافت کرده واز متند خارج می شوید.
  - ۴- اگر پاسخی در مدت زمان معین دریافت نکردید مقدار retry را افزایش دهید.
  - ۵- متند retry را چک کنید اگر مقدار کمتر حد مشخص شده بود دوباره به مرحله ۱ باز می گردیم اگر مساوی بود پیغام مناسبی در Client نمایش داده می شود.
- (بخش آخر(۵) مهم ترین قسمت برنامه می باشد )

متند زیر داده ها را ارسال و دریافت می کند و شما می توانید در هر قسمت از برنامه که به ارسال و دریافت داده نیاز دارید از آن استفاده کنید.

```
private int SndRcvData(Socket s, byte[] message,EndPoint rmtdevice)
{
    int recv;
    int retry = 0;
    while (true)
    {
        Console.WriteLine("Attempt #{0}", retry);
        try
        {
            s.SendTo(message, message.Length,
SocketFlags.None, rmtdevice);
            data = new byte[1024];
            recv = s.ReceiveFrom(data, ref Remote);
        }
        catch (SocketException)
        {
            recv = 0;
        }
        if (recv > 0)
        {
            return recv;
        }
        else
        {
            retry++;
            if (retry > 4)
            {
```

```
        return 0;
    }
}
```

این متدهای سه ورودی می باشد .

- ۱- یک سوکت که پیش از متد ساخته می شود.
  - ۲- پیغام که برای ارسال به راه دور
  - ۳- یک شبیEndPoint که حاوی IP و Port

به برنامه کامل توجه کنید.

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
class RetryUdpClient
{
    private byte[] data = new byte[1024];
    private static IPEndPoint sender = new
IPEndPoint(IPAddress.Any, 0);
    private static EndPoint Remote = (EndPoint)sender;

    private int SndRcvData(Socket s, byte[] message, EndPoint
rmtdevice)
    {
        int recv;
        int retry = 0;
        while (true)
        {
            Console.WriteLine("Attempt #{0}", retry);
            try
            {
                s.SendTo(message, message.Length, SocketFlags.None,
rmtdevice);
                data = new byte[1024];
                recv = s.ReceiveFrom(data, ref Remote);
            }
            catch (SocketException)
            {
                recv = 0;
            }

            if (recv > 0)
            {
                return recv;
            }
            else
            {
                retry++;
                if (retry > 4)
```

```

        {
            return 0;
        }
    }
}

public RetryUdpClient()
{
    string input, stringData;
    int recv;
    IPEndPoint ipep = new IPEndPoint(
        IPAddress.Parse("127.0.0.1"), 9050);
    Socket server = new Socket(AddressFamily.InterNetwork,
        SocketType.Dgram, ProtocolType.Udp);
    int sockopt =
        (int)server.GetSocketOption(SocketOptionLevel.Socket,
            SocketOptionName.ReceiveTimeout);
    Console.WriteLine("Default timeout: {0}", sockopt);
    server.SetSocketOption(SocketOptionLevel.Socket,
        SocketOptionName.ReceiveTimeout, 3000);

    sockopt =
        (int)server.GetSocketOption(SocketOptionLevel.Socket,
            SocketOptionName.ReceiveTimeout);

    Console.WriteLine("New timeout: {0}", sockopt);
    string welcome = "Hello, are you there?";
    data = Encoding.ASCII.GetBytes(welcome);
    recv = SndRcvData(server, data, ipep);
    if (recv > 0)
    {
        stringData = Encoding.ASCII.GetString(data, 0, recv);
        Console.WriteLine(stringData);
    }
    else
    {
        Console.WriteLine("Unable to communicate with remote host");
        return;
    }
    while (true)
    {
        input = Console.ReadLine();
        if (input == "exit")
            break;
        recv = SndRcvData(server, Encoding.ASCII.GetBytes(input),
            ipep);
        if (recv > 0)
        {
            stringData = Encoding.ASCII.GetString(data, 0, recv);
            Console.WriteLine(stringData);
        }
        else
            Console.WriteLine("Did not receive an answer");
    }
    Console.WriteLine("Stopping client");
    server.Close();
}

```

```

    }
    public static void Main()
    {
        RetryUdpClient ruc = new RetryUdpClient();
    }
}

```

پس از اجرای برنامه RetryUdpClient با برنامه **Client** قبلی برنامه **UDP Server** با استفاده از متدهRetryUdpClient مبادرت به برقراری ارتباط می نماید در صورتی که مقدار بازگشتی از برنامه **Server** که در متغیر **Recv** بیشتر از صفر باشد برنامه داده ها را ارسال می کند اما در صورتیکه داده ها را ارسال کرد اما در مدت زمان مشخص شده (3000) میلی ثانیه جوابی از سرور دریافت نکرد (مثلًا برنامه سرور بسته شود) در داخل متده **SndRcvData** یک استثنارخ می دهد.

```

        catch (SocketException)
        {
            recv = 0;
        }
    }

```

در این صورت با صفر کردن متغیر **Recv** به برنامه می فهمانیم که اشکالی پیش آمده و شرط مذکور درست نمی باشد.

```

    if (recv > 0)
    {
        return recv;
    }
}

```

پس برنامه اجرا شده که متغیر **Retry** را یک واحد افزایش می دهد و چون برنامه هنوز از حلقه خارج نشده دوباره به ابتدای حلقه برگشته و مراحل قبل را تکرار کرده تا همان داده را ارسال کند در صورتی که برنامه نتواند داده ها را ارسال کند در مجموع ۴ بار به ارسال دوباره داده سعی می کند اگر موفق به ارسال نشود پیغام

**Did not receive an answer**

ظاهر می شود اگر تا قبل از رسیدن به ۴ بار برنامه سمت سرور را اجرا کنید می بینید که داده ها ارسال می شود.

برای درک برنامه کافی که برنامه **Client** را به صورت قدم به قدم با استفاده از کلید F11 (Trace) کنید.

## مدیریت Error های چندگانه

شما می توانید Error های چند گانه ای که تولید می شوند را مدیریت کنید. منظور از Error های چند گانه **CodeError** می باشد.

شما می توانید برای هر Error که در Catch رخ می دهد دلیل آنرا به صورت یک پیغام مناسب با Error تولید شده به کاربر نمایش دهید.  
در زیر لیستی از Error های که ممکن است به وجود آیند را توضیح داده ایم.

ERROR CODE	DESCRIPTION
10004	وقفه فراخوانی تابع
10003	دسترسی غیر مجاز
10014	آدرس ناصحیح
10022	ورودی نا معتبر
10024	سوکت های باز زیادی وجود دارد
10035	منابع به طور موقت غیرقابل دسترس
10036	عملیاتی هم اکنون در حال پیشرفت بوده
10037	عملیاتی پیش از این در حال پیش رفت بوده
10038	عملیات Socket برقرار نیست
10039	آدرس مقصد لازم است
10040	پیغام شما طولانی است
10041	نوع پروتکل شما برای سوکت غلط است
10042	تنظیمات پروتکل شما صحیح نیست
10043	پروتکل شما پشتیبانی نمی شود
10044	نوع سوکت شما پشتیبانی نمی شود
10045	عملیات شما پشتیبانی نمی شود
10046	پروتکل Family پشتیبانی نمی شود
10047	شما توسط AddressFamily پشتیبانی نمی شود
10048	آدرس پیش از این استفاده شده
10049	قادر به اختصاص دادن آدرس نیست
10050	شبکه از کار افتاده است
10051	شبکه در دسترس نیست
10052	ارتباط شبکه شما Reset شده
10053	نرم افزار سبب خاتمه ارتباط شده است
10054	ارتباط شما توسط رقیب ریست شده
10055	فضای بافر در دسترس نیست
10056	سوکت شما پیش از این متصل بوده
10057	سوکت شما مرتبط نیست
10060	ارتباط خارج از زمان(timed out) (وقفه در ارتباط)

10061	ارتباط قبول نشده
10064	در دسترس نیست Host
10065	به سوی Host نیست Route
10067	پردازش طولانی است
10091	زیر سیستم شبکه غیر قابل دسترس
10101	Graceful shutdown in progress
10109	نوع کلاس پیدا نمی شود
11001	پیدا نشده Host

به برنامه Client زیر توجه کنید  
در این برنامه از دو CodeError استفاده شده است.

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
class BestUdpClient
{
    private byte[] data = new byte[1024];
    private static IPEndPoint sender = new
    IPEndPoint(IPAddress.Any, 0);
    private static EndPoint Remote = (EndPoint)sender;
    private static int size = 30;
    private static int AdvSndRcvData(Socket s, byte[] message,
    EndPoint rmtdevice)
    {
        int recv = 0;
        int retry = 0;
        while (true)
        {
            Console.WriteLine("Attempt #{0}", retry);
            try
            {
                s.SendTo(message, message.Length, SocketFlags.None,
                rmtdevice);
                data = new byte[size];
                recv = s.ReceiveFrom(data, ref Remote);
            }
            catch (SocketException e)
            {
                if (e.ErrorCode == 10054)
                    recv = 0;
                else if (e.ErrorCode == 10040)
                {
```

```

        Console.WriteLine("Error receiving packet");
        size += 10;
        recv = 0;
    }
}
if (recv > 0)
{
    return recv;
}
else
{
    retry++;
    if (retry > 4)
    {
        return 0;
    }
}
}
}

public static void Main()
{
    string input, stringData;
    int recv;
    IPEndPoint ipep = new IPEndPoint(
        IPAddress.Parse("127.0.0.1"), 9050);
    Socket server = new Socket(AddressFamily.InterNetwork,
        SocketType.Dgram, ProtocolType.Udp);
    int sockopt =
(int)server.GetSocketOption(SocketOptionLevel.Socket,
    SocketOptionName.ReceiveTimeout);

    Console.WriteLine("Default timeout: {0}", sockopt);
    server.SetSocketOption(SocketOptionLevel.Socket,
    SocketOptionName.ReceiveTimeout, 3000);
    sockopt =
(int)server.GetSocketOption(SocketOptionLevel.Socket,
    SocketOptionName.ReceiveTimeout);

    Console.WriteLine("New timeout: {0}", sockopt);
    string welcome = "Hello, are you there?";
    data = Encoding.ASCII.GetBytes(welcome);
    recv = AdvSndRcvData(server, data, ipep);

    if (recv > 0)
    {
        stringData = Encoding.ASCII.GetString(data, 0, recv);
        Console.WriteLine(stringData);
    }
    else
    {
        Console.WriteLine("Unable to communicate with remote host");
        return;
    }
    while (true)
}

```

```

    {
        input = Console.ReadLine();
        if (input == "exit")
            break;
        recv = AdvSndRcvData(server, Encoding.ASCII.GetBytes(input),
        ipep);

        if (recv > 0)
        {
            stringData = Encoding.ASCII.GetString(data, 0,
            recv);
            Console.WriteLine(stringData);
        }
        else
            Console.WriteLine("Did not receive an
answer");
        }
        Console.WriteLine("Stopping client");
        server.Close();
    }
}

```

**BetterUdpClient**

شما می توانید کد size += 10; که باعث می شود سایز بافر افزایش یابد را مثل برنامه تغییر داده تا سایز بافر شما متناسب با داده های ورودی افزایش پیدا کند.

## Broad Cast چیست؟

درواقع زمانی که بخواهید یک پیغام را برای همه ی کامپیوتر های شبکه تا یک محدوده ی مجاز ارسال کنید به این کار Broad Cast می گویند.

برای ارسال BroadCast نمی توان از پروتکل TCP استفاده کرد زیرا در ارتباط TCP دو دستگاه باید با یکدیگر یک ارتباط خصوصی داشته باشند و نمی توان بسته های Broad Cast استفاده می کرد بدین منظور از پروتکل UDP استفاده می شود. فرمت IP حاوی دو نوع BroadCast می باشد.

Local BroadCast - ۱

Global BroadCast - ۲

برنامه نویس های شبکه از Local BroadCast برای ارسال پیغام ها در یک Subnet خاص استفاده می کنند.

به شما اجازه می دهد که یک بسته را به همه ی آدرس ها در یک شبکه Global BroadCast ارسال کنید.

این نکته مد نظر داشته باشید که روتور ها اجازه عبور چنین بسته ای را در شبکه نمی دهند و بسته شما در شبکه گم می شود.

یک IP از دو بخش تشکیل شده است یکی آدرس شبکه و دیگری آدرس هاست می باشد. برای مثال اگر IP از کلاس C باشد ۱۹۲.۱۶۸.۱.۱ آدرس BroadCast آن به این صورت می باشد ۱۹۲.۱۶۸.۱.۲۵۵ . در این قسمت مختصراً به IP می پردازیم .(مطلوب این بخش از کتاب خود آموز TCP/IP ، MCSE از انتشارات نص بر گرفته شده ) قالب یک IP حاوی ۳۲ بیت می باشد یک آدرس IP را بطور معمول و استاندارد، به صورت چهار عدد ده دهی که با نقطه از هم جدا می شود نشان می دهد . عدد این دهی که با نقطه از هم جدا می شود نشان می دهد . قسمتی از آدرس که نشان دهنده ای ابزار روی یک شبکه خاص است را ID شبکه و قسمتی را که خود ابزار را نشان می دهد ID میزبان می نامند.

### کلاس های شبکه

کلاس A هشتایی اول را برای شناسایی ID شبکه و بقیه هشتایی ها برای میزبان می باشد منظور از هشت تایی ، در حالت باینری می باشد . کلاس B: آدرس های کلاس B خیلی پرکار هستند آنها هشتایی اول و دوم آدرس را برای ID های شبکه و دو تا هشتایی بعدی را برای میزبان ها استفاده کرده اند . کلاس C : آدرس های کلاس C سه هشتایی اول را برای ID های شبکه و هشتایی باقیمانده را برای میزبانها استفاده می کنند .

{}

### نکته

تمام آدرس های کلاس A با اولین بیت تنظیم شده به صفر شروع می شود (در حالت باینری) تمام آدرس های کلاس B با اولین بیت آدرس تنظیم شده به ۱ و دومین بیت ۰ شروع می شود همه آدرس های کلاس C با د و بیت اول تنظیم شده به ۱۱ و سومین بیت ۰ می باشد

	پایان	شروع
کلاس A	1	127
کلاس B	128	191
کلاس C	192	223

شبکه ID

میزبان ID

کلاس A

0																				
---	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

شبکه ID

میزبان ID

کلاس B

1	0																			
---	---	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

شبکه ID

میزبان ID

کلاس C

1	1	0																		
---	---	---	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

دو کلاس دیگر وجود دارد که کلاس D برای Multi Cast و کلاس E که به صورت رزرو می باشد

کلاس D در محدوده ی 224 تا 239 با چهار بیت 1110 در ابتدا تنظیم شده است و کلاس E در محدوده ی 240 تا 247 می باشد

برای تبدیل یک آدرس به صورت باینری مثل 135.55.108.75 به صورت زیر عمل می کنیم

	128	64	32	16	8	4	2	1
135	1	0	0	0	0	1	1	1
55	0	0	1	1	0	1	1	1
108	0	1	1	0	1	1	0	0
75	0	1	0	0	1	0	1	1

همانطور که مشاهده می کنید در بیت اول 135 در حالت باینری به صورت 10 می باشد که ما به راحتی می توانیم تشخیص دهید که این IP در کلاس B قرار دارد و چون کلاس B دارای دو تا هشتایی ID شبکه و دو تا هشتایی host شبکه می باشد IP پخش همگانی(BroadCast) آن به صورت 135.55.255.255 می باشد.

به موضوع اصلی خودمان بر می گردیم کتابخانه Net. حاوی عناصری برای ارسال و دریافت بسته های BroadCast می باشد. برنامه زیر نحوه ارسال یک بسته BroadCast را نشان می دهد .

```
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();
    }
}
```

برنامه را اجرا می کنیم . با اینکه برنامه اجرا می شود اما نمی تواند با استفاده از متده SendTo اقدام به ارسال یک بسته BroadCast کند و در این خط با خطا مواجه می شود . برای این کار شما باید یک سری تنظیمات انجام دهید تا به سوکت بفهمانید که این یک بسته بroadcast می باشد. برای ایجاد این نوع تنظیم باید از متده SetSocketoption که قبلا هم از آن استفاده کرده ایم به کار گیریم نحوه ای اسفاده از این متده به صورت زیر است.

```
Socket sock = new
Socket(AddressFamily.InterNetwork, SocketType.Dgram,
ProtocolType.Udp);
sock.SetSocketOption(SocketOptionLevel.Socket, SocketOptionName
.Broadcast, 1);
```

عدد ۱ در SetSocketOption به این معنی که شما مجوز عبور بسته BroadCast را داده اید و عدد صفر به این معنی می باشد که شما اجاز ارسال بسته Broad Cast را از آن صلب کرده اید. برنامه زیر اصلاح شده برنامه‌ی قبلی (Client) می باشد.

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
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("192.168.0.255"), 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();
        Console.ReadKey();
    }
}
```

در این برنامه از دو سطح Global و Local استفاده شده است که می توانید با Trace کردن برنامه مقادیر هر شی را بررسی کنید

اما نحوه‌ی دریافت بسته‌های BroadCast به صورت زیر می‌باشد(Server)

```

using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
class RecvBroadcst
{
    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();
        Console.ReadKey();
    }
}

```

در این برنامه نام کامپیوتر شما برای Server ارسال می‌شود (شما باید اول برنامه Server را اجرا کنید و سپس برنامه Client را اجرا کنید)  
نکته: قبل از اجرای برنامه باید Connection مجازی که ساخته بودید (LoopBackAdapter) را با IP 192.168.0.1 مقدار دهی کنید تا برنامه شما نتیجه مناسب را نشان دهد همچنان می‌توانید خطاهای تولید شده را توسط دستور try و Catch مدیریت کنید.

## MultiCast

با استفاده از Broad Cast شما می‌توانید اطلاعات را به همه‌ی دستگاه‌ها در یک Subnet ارسال کنید بنابراین اطلاعات به یک Subnet محلی ارسال می‌شوند (IP MultiCasting) راهی که به برنامه شما اجازه می‌دهد بسته‌ها را به مجموعه‌ای از Subnet‌ها را ارسال کنید.  
در واقع با IPMultiCast، ما کامپیوتر‌ها را در یک گروه قرار می‌دهیم و سپس به ارسال بسته‌ها در این گروه اقدام می‌کنیم.

IPMultiCast طرحی که از یک محدوده IP خاص برای یک گروه استفاده می‌کند.  
هر گروه Multi Cast حاوی یک گروه کامپیوتر می‌باشد که به یک IP گوش می‌دهد.  
رنج MultiCast از 239.255.255 تا 244.0.0.1 برای نمایش گروه‌های IP Address می‌باشد

برطبق RFC3171 گروه‌ها به صورت زیر تقسیم شده است.

**{ تو ضیحات:  
RFC چیست؟ }**

RFC ها یا (Internet Request For Comment) مستنداتی هستند که اطلاعات کاملی درباره TCP/IP و زیر پروتکل هایش در اختیار ما می‌گذارند  
مستندات RFC از طریق اینترنت قابل دسترس هستند بهترین سایت آرشیو Internic از طریق آدرس [www.internic.net](http://www.internic.net) قایل دسترس هستند برای شروع مطالعات خود درباره RFC ها یک ایندکس خوب درباره RFC ها پیدا کنید. RFC ها ظاهرا متون خشکی به نظر می‌رسد اما آنها شامل توصیف کاملی از استانداردهای پایه‌ای TCP/IP هستند.

محدوده	تخصیص داده شده
224.0.0.0 - 224.0.0.255	Local network control block
224.0.1.0 - 224.0.1.255	Internetwork control block
224.0.2.0 - 224.0.255.0	AD-HOC block
224.1.0.0 - 224.1.255.255	ST multicast groups
224.2.0.0 - 224.2.255.255	SDP/SAP block
224.252.0.0 - 224.255.255.255	DIS transient block
225.0.0.0 - 231.255.255.255	Reserved
232.0.0.0 - 232.255.255.255	Source-specific multicast block
233.0.0.0 - 233.255.255.255	GLOP block
234.0.0.0 - 238.255.255.255	Reserved
239.0.0.0 - 239.255.255.255	Administratively scoped block

هر IPAddress برای یک پروژه خاص تعیین شده است برای مثال آدرس 224.0.0.1, 224.0.0.2 برای ارتباط گروه‌هایی از روترهای رزرو شده است

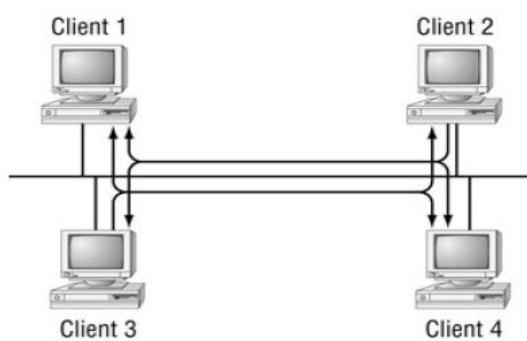
## تکنیک های Multi Cast

دو تکنیک برای کنترل جلسه های MultiCast وجود دارد

۱-تکنیک PeerToPeer که در آن همه کلاینت ها می توانند پیغام ها را به بقیه کلاینت ها در گروه ارسال کنند

۲-سرور مرکزی که پیغام ها را برای گروه های کلاینت ارسال می کند.

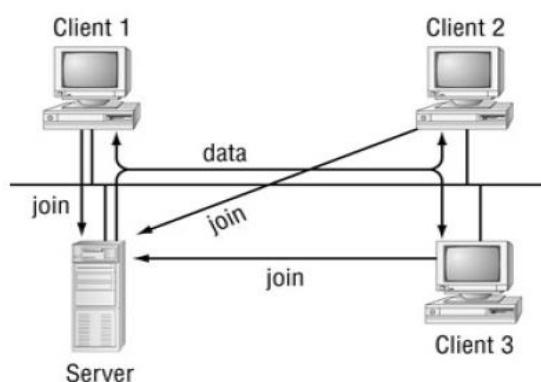
در گروه های مالتی کست Client همه گروه های Client با هم مساوی هستند  
هر Client در گروه توانایی تغییر پیغام سایر Client ها در گروه را دارا می باشد.



## Central Server

یک دستگاه که همه گروه های Multicast در شبکه را کنترل می کند.

فردی که تقاضای اتصال به یک گروه MultiCast را می دهد باید مجوز آنرا از سرور مرکزی دریافت کند اگر مجوز از سرور مرکزی برای MultiCast رد شود بسته های MultiCast به Client ارسال نمی شود.



## ارسال بسته های MultiCast از طریق Routers

اگر چه بسته های MultiCast می توانند در سرتاسر شبکه عبور کنند اما اتفاقی که ممکن است در این بخش بیفتد، در قسمت Router ها می باشد به صورت پیش فرض بیشتر روترها اجازه عبور بسته های MultiCast را نمی دهند.

Internet Group Management Protocol (IGMP) که به روترها کمک کنند تا بسته های SubNet های مختلف عبور دهد.

## C# Socket Multicasting

کلاس IPMultiCast ، Socket پشتیبانی می کند

دو تا انتخاب برای استفاده از MultiCasting وجود دارد

۱- اضافه کردن سوکت به گروه MultiCast

۲- حذف سوکت از گروه MultiCast

کلاس MulticastOption معین می کند که سوکت می تواند به گروه MultiCast اضافه و یا حذف شود. این کلاس دارای دو سازنده (Constructors) می باشد

MulticastOption( IPAddress )

MulticastOption( IPAddress , IPAddress )

برای مثال اگر شما نیاز داشته باشید که یک سوکت را به یک گروه MultiCast اضافه کنید باید به شکل زیر استفاده کنید

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

در واقع IP های MultiCast فقط توسط برنامه نویس در برنامه به سیستم ها اختصاص داده می شود

برنامه زیر نحوه **در یافت بسته های MultiCast** را نشان می دهد.

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
```

```

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();
        Console.ReadKey();
    }
}

```

برنامه نوشته شده به غیر از خط `MulticastOption` تکرا ری می باشد و نیازی به توضیح اضافی نمی باشد.

برنامه زیر نیز بسته های `MultiCast` را ارسال می کند.

```

using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
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();
        Console.ReadKey();
    }
}

```

ابتدا برنامه دریافت را اجرا کرده و بعد برنامه ارسال را اجرا کنید.

اگر می خواهد عملکرد برنامه را بهتر متوجه شوید باید در یک شبکه واقعی دو نسخه متفاوت از برنامه Recv را بر روی دو کامپیوتر مختلف اجرا کنید مشاهده می کنید که پیغام ها در هر دو برنامه مشاهده می شود حال اگر IPMulticast یکی از برنامه های Recv را از 224.0.0.1 به یک IPMulticast دیگر تغیر دهید پس از جرای دوباره برنامه ها و بعد از آن اجرای برنامه Send مشاهده می کنید که یک برنامه Recv آن پیغام را دریافت می کند و آن هم برنامه ای که 224.0.0.1 آن باشد چون برنامه Send فقط به این گروه بسته را ارسال می کند

شما نمی توانید دو نسخه از برنامه Recv را در یک کامپیوتر اجرا کنید چون هر دو از یک پورت استفاده می کنند و برنامه ها نمی توانند به یک پورت مشترک وصل شوند.

### تعیین مقدار TTL برای Multicast

TTL یا همان زمان زنده ماندن بسته ، زمانی که یک بسته از یک روتور عبور می کند تا به مسیر مورد نظر برسد این مقدار پس از عبور از هر روتور کاهش می یابد حال اگر این مقدار صفر شود این بسته منقضی شده و به مقصد نمی رسد . در واقع این عمل برای کاهش سربار بر روی خطوط می باشد. هر چند خود روتور ها یک مقدار پیش فرض برای بسته در نظر می گیرد اما شما نیز می توانید مقدار مورد نظر خود را با استفاده از متده SetSocketOption تعیین کنید . (در واقع TTL مشخص می کند که یک بسته به چه تعداد(Hope) می تواند از روتور ها عبور کند)

به برنامه SendMulticast زیر توجه کنید.

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
class NewMultiSend
{
    public static void Main()
    {
        Socket server = new Socket(AddressFamily.InterNetwork,
        SocketType.Dgram, ProtocolType.Udp);
        IPEndPoint iep = new IPEndPoint(IPAddress.Any, 9050);
        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();
    }
}
```

}

برنامه Send را با برنامه Recv قبلی تست کنید  
قبل از Test برنامه ها کانکشن مجازی که ساخته بودید را فعال Enable کنید.

## (Simple Mail Transfer Protocol)SMTP

{ در این بخش شما با پروتکل SMTP و نحوه‌ی برنامه نویسی برای این پروتکل که مربوط به ارسال Email می‌باشد آشنا می‌شوید قبل از اینکه به برنامه نویسی در این بخش بپردازیم می‌خواهیم با راه اندازی یک MailServer در Windows2003 آشنا شویم تا نتایج برنامه‌هایی که می‌نویسیم را مشاهده کنیم.

(داخل پرانتز)

این برنامه به شما اجازه می‌دهد که در کنار سیستم عامل خود اقدام به نصب یک سیستم عامل دیگر به طور مجازی نمایید به این صورت شما می‌توانید از سیستم عامل مجازی به همراه سیستم عامل اصلی خود، به طور همزمان استفاده کنید و از مزایای هر دو سیستم عامل استفاده کنید که یکی از این مزایا، ایجاد شبکه مجازی بین دو سیستم عامل نصب شده می‌باشد.

برای شروع برنامه Virtual PC(2004) را اجرا کرده پس از اجرا در کادر باز شده بر روی گزینه New کلیک کرده در صفحه‌ی جدید بر روی Next کلیک کنید در صفحه‌ی بعدی بدون تغییر، گزینه‌ی پیش فرض را قبول کرده (Create a Virtul Machine) بر روی Next کلیک کنید یک نام دلخواه انتخاب کرده بر روی Next کلیک کنید در صفحه‌ی بعدی از لیست کشویی Next Windows Server 2003 (Operation System) گزینه‌ی انتخاب کرده بر روی Adjusting the Ram کلیک کرده در صفحه‌ی بعد دو گزینه برای انتخاب وجود دارد، گزینه‌ی انتخاب کرده در این قسمت مقدار حافظه برای تخصیص به سیستم عامل را مشخص کنید. برای

مثال اگر RAM شما 256 می‌باشد مقدار 128 یعنی نصف RAM خود را به این سیستم عامل اختصاص دهید بر روی گزینه Next کلیک کرده در صفحه‌ی بعدی نیز Next را انتخاب کنید بعد

مسیر ذخیره سازی سیستم عامل(پس از نصب سیستم عامل ، یک فایل مشخص با پسوند Vhd به وجود می آید) را مشخص کنید بعد Next و بعد Finish را انتخاب کنید.

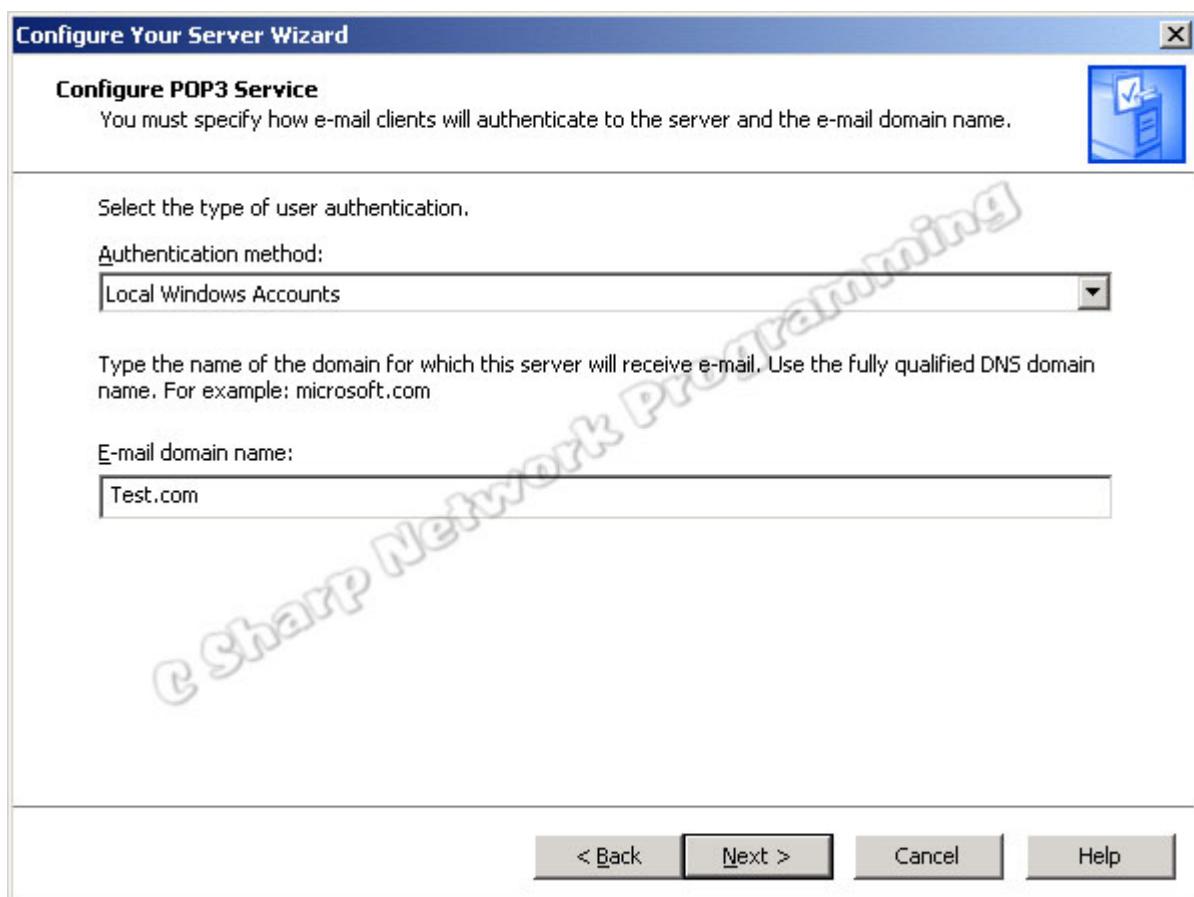
همانطور که مشاهده می کنید یک گزینه به لیست اضافه می شود و در سمت راست دکمه Start فعال شده قبل از اینکه دکمه Start را کلیک کنید Cd ویندوز 2003 را در CDRom قرار داده و بعد گزینه Start را کلیک کنید . نصب Win2003 با خودتون.

### نحوه ایجاد کردن شبکه

پس از نصب Win2003 سیستم مجازی شما یک بار ریست میشه پس از بالا امدن سیستم مجازی در سیستم عامل اصلی (XP) کانکشن مجازی که ساخته بودید را اجرا (دابل کلیک ) کنید (از پنجره NetworkConnection طرز به وجود آوردن LoopBackAdapter را قبل توضیح دادم پس از فعال شدن آن IP سیستم اصلی(XP) خود را برابر 192.168.0.2 قرار دهید و در قسمت Preferred DNS Server IP مقدار رو به رو را وارد کنید 192.168.0.1 حال وارد ویندوز مجازی شده و بر روی کانکشن که به علامت زرد در آمده کلیک کنید آنرا برابر 192.168.0.1 در قسمت Preferred DNS Server را وارد کنید برای اینکه صحت ارتباط را چک کنید در ویندوز مجازی (Server) خود وارد Command Prompt شده و دستور Ping 192.168.0.2 را وارد کنید اگر عبارت ..... را مشاهده کردید مشکلی نیست . همین کار را برای ویندوز اصلی خود (XP) انجام دهید ولی این بار به صورت ping 192.168.0.1 وارد کنید.

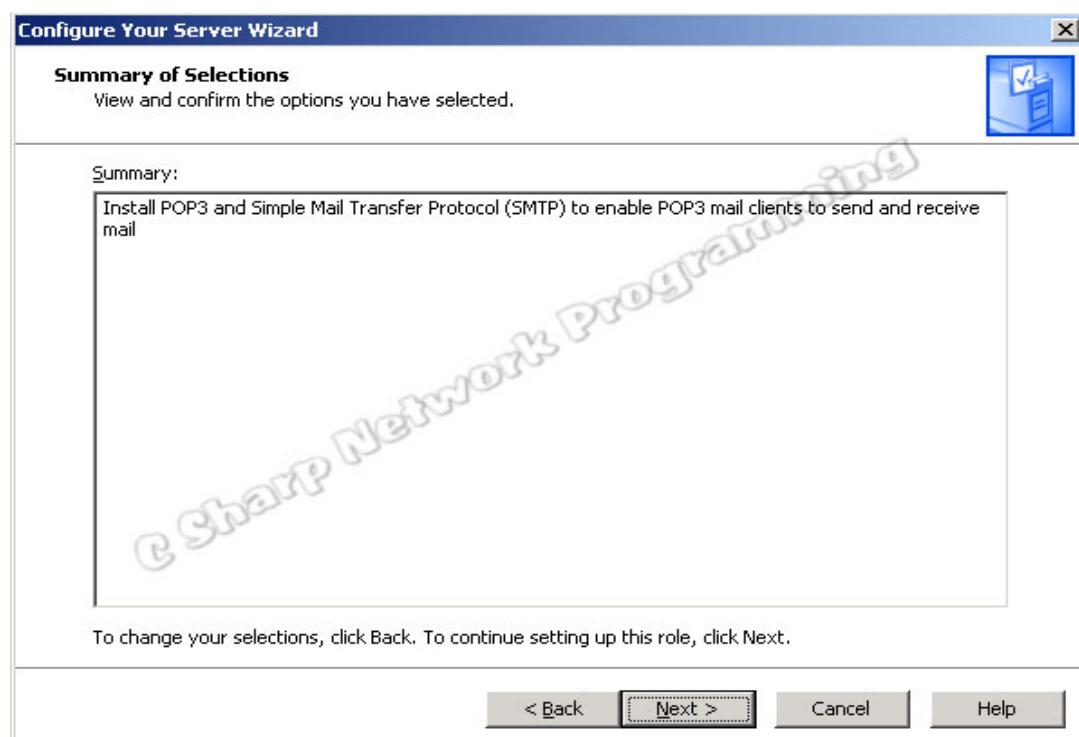
## ایجاد Mail Server

حال دوباره به win2003 بازگشته(برای اینکه Virtaul PC خود را از برنامه Mouse آزاد کنید کلید Alt را نگه دارید) از منوی Start گزینه Mange Your Server را انتخاب کرده در پنجره باز شده گزینه Add or Remove را انتخاب کرده بر روی Next کلیک کرده پس از چند ثانیه در صفحه بعدی گزینه Custom Configuration را انتخاب کرده بر روی ویرایش Next کلیک کرده و در این صفحه گزینه MailServer(POP3,SMTP) را انتخاب کرده روی Next کلیک کنید در این صفحه یک نام دامین مثلا Test.com را وارد کنید



بر روی Next کلیک کنید

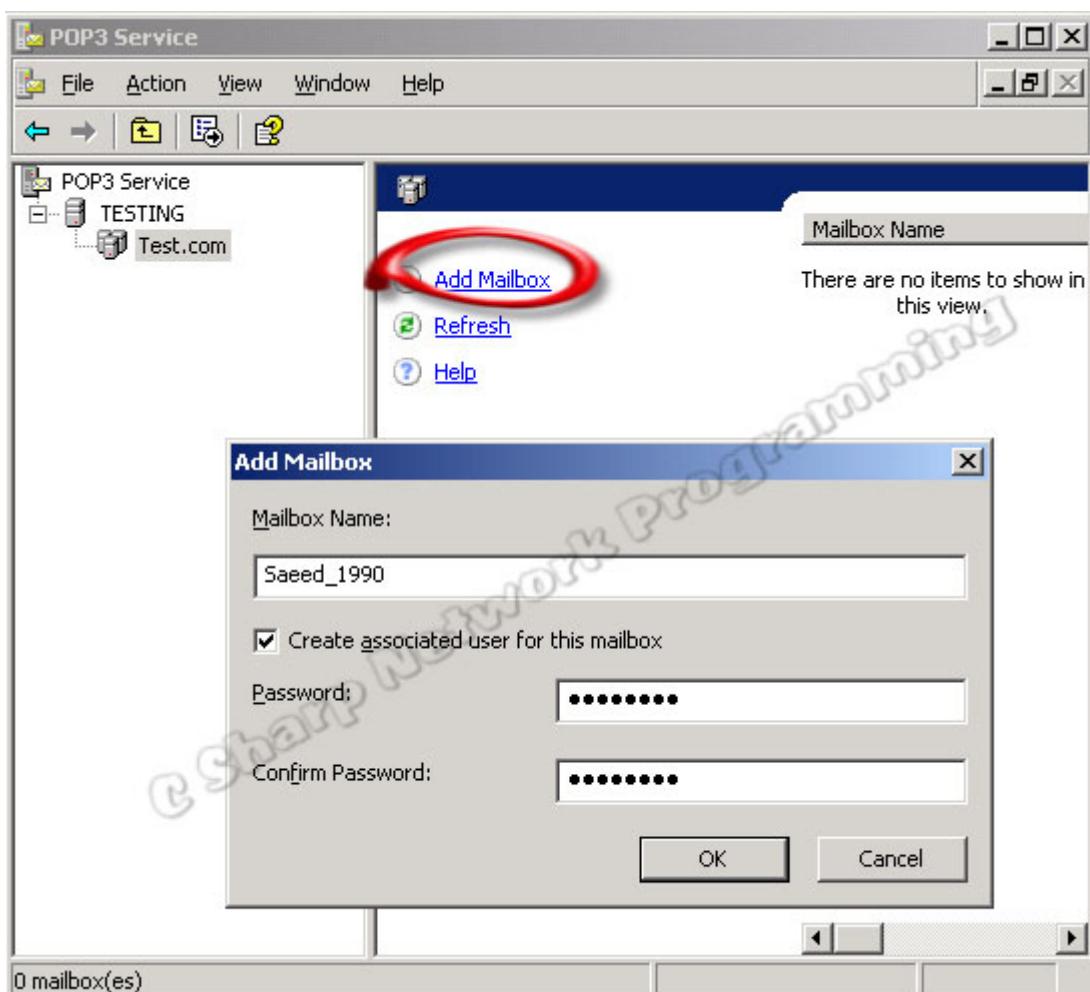
ویندوز 2003 CD را در CDRom نصب آغاز گردد



در انتها نیز بر روی Finish کلیک کنید.

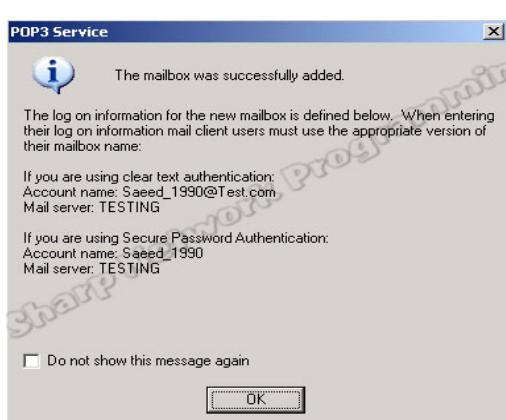


حال برای ساختن یک MailAccount در Win2003 وارد منوی Start→AdministrativeTools→POP3Service شده باز شده بروی AddMailBox کلیک کنید.



در قسمت Mailbox Name یک نام وارد کنید و در قسمت Password یک کلمه عبور (کلمه عبور باید ترکیبی از کلید ها مثل (Shift+A) واز ۶ کاراکتر بیشتر باشد ) بر روی ok کلیک کنید

حال شما یک Email به نام  
Saeed\_1990@Test.com  
ساخته اید {



## برنامه های ارسال Email

خوب بر می گردیم سر موضوع اصلی کلا پروتکل SMTP برای ارسال Email و پروتکل POP3

برای دریافت Email می باشد

برای ارسال Email روشی کلاسی برای ارسال Email می باشد System.Net.Mail

برای ارسال Email دو تا فرمت وجود دارد

Send(MailMessage message)

Send(string from, string to, string subject, string body)

فرمت اول به شما اجازه ارسال یک شی از نوع MailMessage را می دهد (در ادامه توضیح

داده میشه) فرمت دوم به شما اجازه می دهد که به صورت دستی فیلد های Mail را تنظیم کنید

: آدرس Email فرد ارسال کننده From

: آدرس گیرنده Email که می تواند بیش از یکی باشد To

: موضوع Email Subject

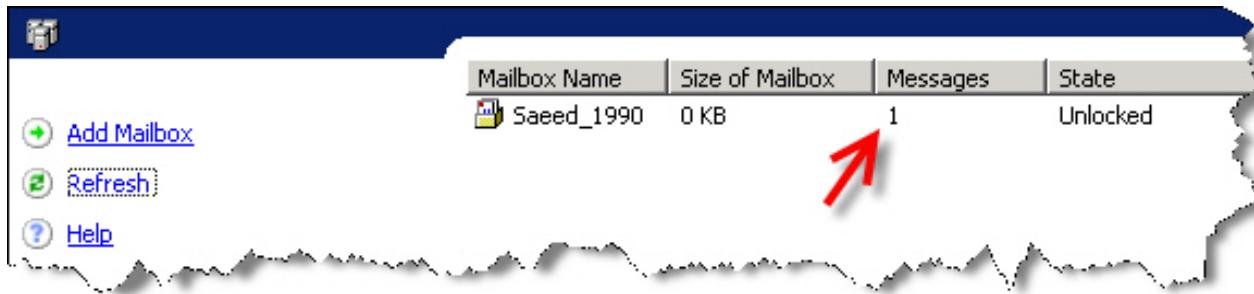
و پارامتر آخر (Body) بدنی اصلی پیغام را تشکیل می دهد که فرمت آن می تواند Text یا HTML باشد

به برنامه ارسال Email به MailServer که به وجود آوردم توجه کنید

```
using System;
using System.Net;
using System.Net.Mail;

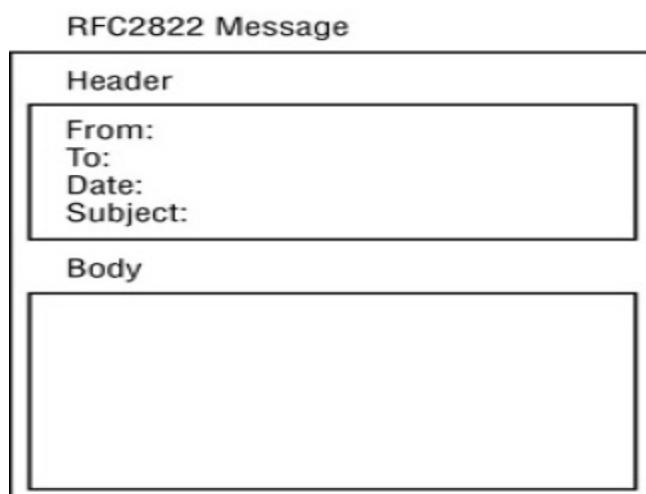
class MailTest
{
    public static void Main()
    {
        string from = "Ali@yahoo.com";
        string to = "Saeed_1990@Test.com";
        string subject = "This is a test mail message";
        string body = "Salam saeed in mail faghat baray Teste..";
        SmtpClient sc=new SmtpClient();
        try
        {
            sc.Host = "192.168.0.1";
            sc.Send(from, to, subject, body);
            Console.WriteLine("Send Email... ");
            Console.ReadKey();
        }
        catch
        {
            Console.WriteLine("Error");
            Console.ReadKey();
        }
    }
}
```

در آن قرار گرفته را مشخص کرده ایم بقیه خطوط هم که واضح می باشد برنامه را اجرا کنید همانطور که در شکل زیر مشاهده می کنید یک Email از طرف Ali@yahoo.com به Saeed\_1990@test.com ارسال شده است



### فرمت RFC2822 طبق Email

طبق RFC2822 هر پیغام باید حاوی دو بخش جداگانه باشد  
۱-عنوان (Header) که حاوی اطلاعات پیام در mail می باشد.  
۲-body (بدنه) که حاوی متن Email می باشد.



#### Date فیلد

این فیلد برای اینکه تشخیص داده بشه پیام ارسال شده در چه تاریخی و زمانی (Time) ارسال شده است مورد استفاده قرار می گیرد.

این فیلد حاوی کلمه‌ی کلیدی Date

Date: Wed, 21 Aug 2002 18:30:00 -0500  
(زمان ارسال یک Email برای کاربران مهم می باشد)

## فیلد های صادر کننده پیام

From: mailbox-list

Sender: mailbox

Reply-To: mailbox-list

مقدار mailbox-list می تواند حاوی یک mailAddress یکتا یا چند mailAddress می باشد. که با کاما از هم جدا شده اند باشد.

در این بخش توضیحات مربوط به فیلد ها از کتاب آقای احسان ملکیان(اصول مهندسی اینترنت) بر گرفته شده است.

From: در جلوی این فیلد آدرس mail نویسنده نامه درج می شود.  
Sender: جلوی این فیلد آدرس Mail کسی را که نامه را حقیقتا ارسال کرده مشخص می شود  
(فرد به گفته کسی دیگر نامه را ارسال کرده و شخصی که ارسال کننده است یک واسطه می باشد)  
ReplyTo: این فیلد زمانی بکار می رود که نویسنده اصلی نامه تمایلی به دریافت پاسخ آن نامه نداشته باشد بلکه بخواهد پاسخ نامه ی ارسالی توسط شخص ثالثی دریافت شود.

## فیلد های آدرس مقصد

To: address-list

Cc: address-list

Bcc: address-list

مقدار address-list می تواند بیش از یک آدرس Mail که با کاما از هم جدا شده اند را شامل شود.

To: آدرس شخص گیرنده نامه  
Cc: در جلوی این فیلد آدرس Mail شخص دیگری که قرار است یک Copy از این نامه را داشته باشد نوشته می شود.

Bcc: این فیلد دقیقا همانند فیلد قبلی است با این تفاوت که گیرندهای نامه از این موضوع که شخص دیگری این نامه را دریافت کرده مطلع نخواهد شد.

### فیلد های شناسایی

Message-ID: msg-id

In-Reply-To: msg-id

References: msg-id

Msg-id: مقداری که می تونه ترکیبی از حروف بزرگ و اعداد برای شناسایی پیام باشد.  
Message-ID: یک شماره منحصر به فرد برای آنکه بتوان بعدا به آن شماره استناد کرد.  
In-Reply-To: در جلوی این فیلد ، شماره نامه ای قرار می گیرد که نامه فعلی در پاسخ به آن نامه ارسال شده است.(مثل شماره نامه های اداری)

References: در جلوی این فیلد شماره نامه های دیگری قرار می گیرد که نامه فعلی به موضوع آن نامه مرتبط است.

### فیلد های اطلاعات

Subject: subject-text

Comments: comment-text

Keywords: phrase-text

Subject: یک کلمه یا جمله کوتاه که مضمون نامه را برای خواننده آن مشخص می کند.  
Comments: فیلدي که توضیحات بیشتری در مورد بدنه اصلی نامه را شامل می شود.  
Keywords: برخی از کلمات کلیدی که با متن و موضوع نامه مرتبط است و برنامه های نامه خوان می توانند آنها را ملاک دسته بندی یا جستجو قرار بدهد.

کتابخانه .NET کلاسی به نام MailMessage دارد که می توانید فرمت RFC2822 را برای ارسال Email با کلاس SmtpClient پیاده سازی کنید.

پیاده سازی این کلاس به صورت زیر می باشد

```
MailMessage newmessage = new MailMessage();
```

کلاس MailMessage دارای مشخصات زیر می باشد.

PROPERTY	DESCRIPTION
Attachments	ضمیمه کردن فایل به پیغام
Bcc	تنظیم آدرس وجودا کردن آن با سمی کالن برای استفاده Bcc
Body	بدنہ Mail را مشخص می کند.
BodyEncoding	نوع کد بندی بدنہ Mail را مشخص می کند.
IsBodyHtml	نوع بدنہ ی پیام را مشخص می کند (Text, Html)
Cc	تنظیم آدرس وجودا کردن آنبا سمسکالن برای استفاده Cc
From	تنظیم آدرس برای استفاده از From
Headers	تنظیم سفارشی مقادیر فیلد Header
Subject	موضوع نامه
Priority	اولویت پیام ها را مشخص می کند.
To	تنظیم آدرس وجودا کردن آن با سمی کالن برای استفاده از To

### ویژگی Header

نحوه استفاده از متده Add به صورت زیر می باشد

```
newmessage.Header.Add("Reply-To", "testing@myisp.net");
```

همچنین نحوه استفاده از فیلد Date به صورت زیر است

```
DateTime mydate = DateTime.Now;
```

```
newmessage.Header.Add("Date", mydate.ToString());
```

### خصوصیات Body

دو خصیصات IsbodyHtml و BodyEncoding که فرمت بدنہ ی پیام را تعیین می کنند.

خصوصیت `BodyEncoding` در `NameSpace.Text` روبه رو فرمت متن را که می تواند مقادیر ASCII ، UTF8 ، UTF7 باشد نوع پیش از نوع `Text` می باشد. `IsBodyHtml` که می تواند `Text` یا `Html` باشد . `True` و `false`

### خصوصیات Priority

مشخصه `Priority` به شما اجازه می دهد که آنرا در فیلد `Header` تنظیم کنید اگر چه این فیلد استاندارد نیست اما بسیاری از `MailClient` ها تشخیص می دهند که این `Email` دارای اهمیت بیشتری برای خواندن می باشد.

: این متده سه مقدار زیر را دریافت می کند.

: این پیام مستلزم رسیدگی فوری می باشد.

: یک پیام عادی می باشد.

: این پیام تصادفی می باشد(مانند یک پیام تجاری)

نحوه استفاده این خصوصیات به صورت زیر است

```
newmessage.Priority = MailPriority.High;
```

با استفاده از خصوصیت `Attachment` شما می توانید یک فایل را ضمیمه نامه الکترونیکی خود بکنیدشما می توانید بیش از یک شی از کلاس `Attachment` را استفاده کنید.

```
Attachment at = new Attachment("C:\\1.bmp");
message.Attachments.Add(at);
```

در ادامه بیشتر درباره این موضوع بحث می کنیم

به برنامه زیر توجه کنید

```

using System;
using System.Net.Mail;
using System.Text;
class MailT
{
    public static void Main()
    {
        MailAddress From1 = new MailAddress("Ali@Yahoo.com");
        MailAddress Tol = new
MailAddress("Saeed_1990@Test.com", "Saeed asgary");
        SmtpClient SM = new SmtpClient();
        MailMessage message = new MailMessage(From1, Tol);
        message.CC.Add("Behzad@test.com");
        message.Bcc.Add("maryam@Test.com");
        message.Subject = "This is a fancy test message";
        message.Headers.Add("Reply-To", "sara@Test.com");
        message.Headers.Add("Comments", "This is a test HTML
message");
        message.Priority = MailPriority.High;
        message.IsBodyHtml = true;
        message.Body = "<html><body><h1>This is a test
message</h1><h2>This message A should have HTML-type
formatting</h2>Please use an HTML-capable viewer.";
        try
        {

            SM.Host = "192.168.0.1";
            SM.Send(message);
            Console.WriteLine("Send Mail....");
            Console.ReadKey();
        }
        catch
        {
            Console.WriteLine("This device is unable to send
Internet messages");
            Console.ReadKey();
        }
    }
}

```

در برنامه بالا برای ذخیره آدرس Email استفاده میشے که می تونیم این آدرس ها را به شی از نوع MailMessage نسبت دهیم و برای انتقال به آدرس ها از نوع SmtpClient که وظیفه آن ارسال Email می باشد استفاده کنیم.

نکته : قبل از اینکه برنامه VirtualPc را اجرا کنید مطمئن شوید که کانکشن مجازی در ویندوز اصلی فعال باشد در غیر این صورت ممکن شبکه‌ی شما برقرار نشود.

## Mail Attachments

در SMTP برای اینکه یک فایل را بتوانید انتقال دهید باید قبل از رسیدن به MailServer داده های باینری به ASCII تبدیل شود. MailClient باید توانایی این تبدیل را داشته باشد. دو تا تکنیک برای تبدیل داده‌ها باینری به متن وجود دارد یکی فرمت uuencode و دیگری سیستم نامه رسان توسعه یافته در اینترنت یا (MIME) بیشتر سیستم‌های Mail اجازه‌ی استفاده از هر دو تکنیک را می‌دهند.

### uuencode

سالها پیش از اینکه اینترنت به محبوبیت برسد مدیران Unix داده‌های باینری خود را از طریق خطوط ارتباطی توسط مودم به صورت اسکی ارسال می‌کردند برنامه‌های آنها داده‌های باینری را به متون اسکی تبدیل می‌کرد. که به آن uuencode می‌گفتند. ۳ to 4 Unix to Unix مخفف uuencode می‌باشد برنامه‌های استفاده می‌کردند آنها ۳ بایت باینری را به ۴ بایت کد اسکی تبدیل می‌کردند.

بخش MIME بر گرفته از کتاب اصول مهندسی اینترنت آقای احسان ملکیان

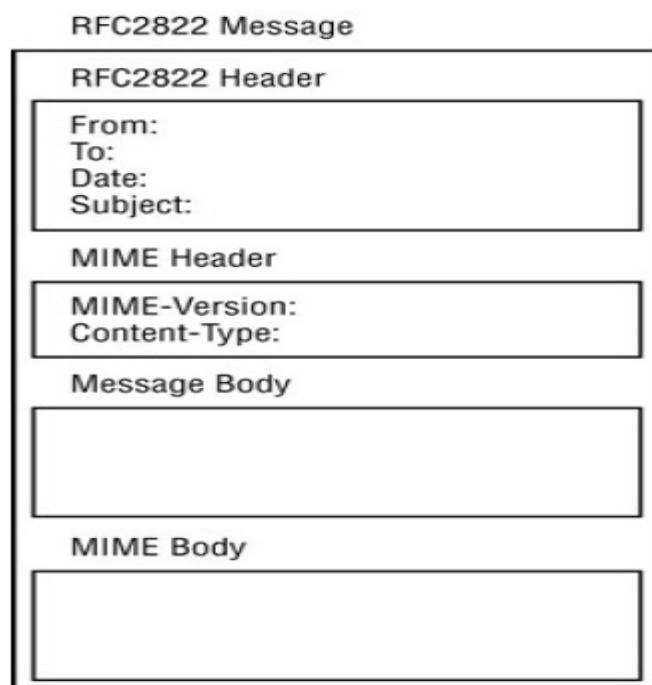
### (Multipurpose Internet Mail Extensions)MIME

در این استاندارد ایده اصلی آن بوده است که بدون پشت پا زدن به استاندارد RFC822 که در آن زمان بسیار محبوبیت داشت روشی ابداع شود تا بتوان فایلهای غیر اسکی همانند فایلهای اجرایی، صدا و تصویر به گونه‌ای در بدنه نامه قرار گیرد که براساس سرویس دهنده‌های قدیمی قابل ارسال و دریافت باشد. در این صورت بدون نیاز به تغییر سرویس دهنده‌های قبلی، فقط باید برنامه نامه خوان درسمت کاربر عوض شود که هزینه کمی را به کاربر تحمیل می‌کند.

استاندارد MIME پنج فیلد جدید در سرآیند نامه تعریف کرده است که این فیلدها در جدول زیر معرفی شده اند

### The MIME Message Header Fields

FIELD	DESCRIPTION
MIME-Version	شماره نسخه MIME
Content-Transfer-Encoding	یک سطر که مضمون کلی نامه را مشخص می کند.
Content-ID	یک مشخصه یا یک شماره منحصر به فرد
Content-Description	طریقه‌ی کد گذاری محتوای نامه
Content-Type	نوع و محتوای نامه



### MIME-Version فیلد

این فیلد به برنامه نامه خوان در سمت کاربر تفهیم می کند که این نامه الکترونیکی با استاندارد سازماندهی و ارسال شده است. در ضمن نسخه استاندارد MIME را نیز مشخص

می نماید .نامه هایی که این فیلد را در سرآیند نامه نداشته باشند (همانند نامه های قدیمی با استاندارد RFC822 ) بصورت نامه های تماماً متنی با کدهای آسکی تلقی می شوند.

#### **Content-Description فیلد**

متنی که در جلوی این فیلد قرار می گیرد مضمون و محتوای نامه را مشخص می کند .گیرنده نامه با استفاده از این فیلد می تواند تشخیص بدهد که آیا رمزگشائی و خواندن پیام ارزشمند است یا نه .

#### **Content-ID فیلد**

این فیلد که مشابه فیلد Message-Id در استاندارد RFC822 است شماره یا رشته های است منحصر به فرد، که می توان به عنوان شماره نامه در نامه های بعدی به آن استناد کرد .

#### **Content-Transfer-Encoding فیلد**

در جلوی این فیلد عبارتی قرار می گیرد که به برنامه نامه خوان در سمت کاربر تفهیم می کند که چه قاعدهای را برای دیکود کردن بدنۀ نامه بکار ببرد . بگونه ای که اشاره شد بر خلاف استاندارد RFC822 در بدنۀ نامه های مبتنی بر استاندارد MIME می تواند کدهای غیر آسکی ، فایلهای صدا تصویر یا کلأ هر فایل دودویی قرار بگیرد .بنابراین در مقصد قبل از نمایش محتوای نامه ، باید قسمت بدنۀ آن پردازش و دیکود شود .اگر بدون رمزگشائی ، نامه را نگاه کنید یکسری کاراکترهای نامفهوم خواهد دید .

انواع کدگذاری در استاندارد MIME به شرح زیر است

METHOD	DESCRIPTION
7-bit	استاندارد 7 بیت برای متن اسکی
8-bit	استاندارد 8 بیت برای متن اسکی
binary	داده های باینری Raw
ietf-token	نשانه گذاری تعمیم یافته بر حسب RFC2045
x-token	Two characters, X- or x-, followed (with no intervening space) by any token

دو نوع کد گذاری دیگر وجود دارد که در زیر به صورت خلاصه به توضیح آنها پرداخته ایم

**کد گذاری Base64** : این روش که به آن ASCII Armor هم گفته می شود در موقعی کاربرد دارد که بخواهید یک فایل داده ( مثل یک فایل اجرائی یا فایل تصویر (را در بدنه نامه جا سازی نمایید . در چنین موقعی بهترین راه حل ممکن آن است که فایل به نحوی به کاراکترهای ASCII تبدیل شده و درون متن نامه قرار بگیرد ) . برای روشن شدن قضیه فرض کنید نامه ای نوشته و به آن فایل تصویر ضمیمه می کنید. این فایل تصویر بصورت کدهای ASCII در متن نامه قرار می گیرد و شما می توانید آن کدها را توسط یک ویرایشگر ساده مثل Notepad (بییند)

روش تبدیل فایلهای باینری به کدهای ASCII بشرح زیر انجام می شود  
از درون فایل داده (سه بایت سه بایت جدا می شود(سه بایت مجموعاً 24 بیت خواهد شد)  
این 24 بیت به چهار قسمت شش بیتی تقسیم می گردد . هر قسمت شش بیتی مجموعاً 64 حالت دارد(از صفر تا 63) که برای حالت صفر کarakter 'A' و برای 1 کarakter 'B' و به همین ترتیب تا 25 که 'Z' قرار می گیرد برای 26 تا 51 به ترتیب 'a' تا 'z' و برای 52 تا 61 به ترتیب کارکتر های 0 تا

۹ برای ۶۲ تا ۶۳ به ترتیب '+' و '/' قرار می گیردبا روش فوق هر فایل دودویی به حالت متنی تبدیل می شود .در مقصد برنامه نامه خوان وقتی گزینه زیر را در متن ببیند به راحتی آنرا به حالت اصلی برخواهد گرداند:

#### Content-Transfer-Encoding: base64

دقت کنید هر فایل که به روش فوق کد شود فقط شامل حروف کوچک و بزرگ انگلیسی ، کاراکترهای ۰ تا ۹ و علامت های + و / خواهد بودو در مقصد هر کاراکتر که به غیر از کاراکترهای ذکر شده لابلای آن وجود داشته باشد به سادگی حذف خواهد شد؛ چون در استاندارد MIME هر سطر می تواند حداکثر هزار کاراکتر باشد لذا پس از تبدیل یک فایل دودویی به حالت ASCII می توان در هر جای متن کد '\n' را اضافه کرد تا هر سطر زیر هزار کاراکتر باشد؛ این کدها در مقصد حذف خواهند شد .این نکته نیز قابل توجه است که طول یک فایل دودویی پس از تبدیل به حالت ASCII Armor طول فایل حداقل با ضریب  $1/3333 \cong 4/3$  افزایش پیدا می کند.

#### کد گذاری : quoted-printable

استفاده از روش قبل برای تبدیل فایلهایی که تعداد کمی کاراکتر با کد بالای ۱۲۸ دارند راه مناسبی نیست چون طول فایل بیهوده افزایش می یابد .در این روش برای کاراکترهایی که کد آنها زیر ۱۲۸ است خود کاراکتر، ولی برای آنها که کدشان بین ۱۲۸ تا ۲۵۵ است ابتدا علامت '=' و بعد دو کاراکتر معادل کد مبنای ۱۶ آن درج می شود.

این روش کد گذاری فقط زمانی مفید است که نسبت کاراکترهای بالای ۱۲۸ در متن بسیار کم باشد چرا که هر کاراکتر بالای ۱۲۸ پس از تبدیل با سه کاراکتر جاینشین خواهد شد.

## فیلد Content-Type

آخرین فیلد سرآیند در استاندارد MIME یکی از کاربردی ترین فیلدها خواهد بود و مشخصات محتوای نامه را تشریح خواهد کرد. بعنوان مثال در قسمت سرآیند یک نامه این فیلد می تواند بصورت زیر تنظیم شده باشد:

Content-Type: Video/Mpeg

به معنای آنکه محتوای بدن فایلی ویدئویی با قالب MPEG و بالطبع نرم افزار نامه خوان باید قادر باشد ضمن استخراج آن از متن ، ابزار نمایش آنرا هم بارگذاری نماید.

انواع که یک نامه الکترونیکی در بر می گیرد شامل زیر است .

نوع **Text**: به معنای آنست که نامه از نوع متنی است . و شامل سه نوع مختلف می باشد.

Plain: نامه معمولی با کارکتر ASCII

Enriched: نامه در غالب یک زبان نشانه گذاری شده ارسال می شود.

html: فرمت متن از نوع تگ های Html

Message: متن به منظور انتقال تکه تکه شده است.

RFC822: متن تنظیم شده در استاندارد rfc822

Partial : متن طولانی به منظور انتقال تکه تکه شده است.

external-body : متن پیام باید از شبکه اینترنت بارگذاری شود.

Image : یک فایل تصویری با غالب JPEG , GIF

Video : فایل ویدئویی با قالب MPEG

Audio : فایل صوتی با قالب snd

Application: برنامه های نظری صفحه گسترده و ورد و غیره ...

: متن دارای چند قسمت با قالبهای متفاوت است **Multipart**

: متن دارای چند قسمت است که ترتیب مشخص دارد **Mixed**

: متن دارای چند قسمت با قالبهای متفاوت است **Parallel**

: متن دارای چند قسمت با قالبهای متفاوت است **Alternative**

: متن شامل چند قسمت است و هر قسمت از نوع RFC822 است **Digest**

خوب می ریم سراغ برنامه نویسی

برای اینکه بتوانیم یک فایل رو به Email خودمون Attachment کنیم از کلاس

استفاده می کنیم همچنین برای مقدار دهی فایل Content-Type باید از

استفاده می کنیم using System.Net.Mime;

برنامه زیر نحوهی ضمیمه کردن یک فایل به Email را نشان می دهد.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net.Mail;
using System.Net.Mime;

namespace MailAttachTest
{
    class Program
    {
        static void Main(string[] args)
        {
            Attachment at = new Attachment(@"C:\1.TXT",
MediaTypes.Text.Plain);

            MailAddress From1 = new
MailAddress("Ali@yahoo.com");

            MailAddress to = new
MailAddress("Saeed_1990@test.com");

            MailMessage newmessg = new MailMessage(From1,to);
            newmessg.Subject = "A test mail attachment message";
            newmessg.Priority = MailPriority.High;
            newmessg.Headers.Add("Comments", "This message
attempts to send a binary attachment");
            newmessg.Attachments.Add(at);
        }
    }
}
```

```
newmesssg.Body = "Here's a test file for you to try";
    try
    {
        SmtpClient Sm = new SmtpClient();
        Sm.Host = "192.168.0.1";
        Sm.Send(newmesssg);
        Console.WriteLine("Send a Message....");
Console.ReadKey();
    }
    catch
    {
        Console.WriteLine("This device cannot send
Internet messages");
    }
}
```

در برنامه بالا چون فایل ما از نوع `text` بود با استفاده از کد `MediaTypeNames.Text.Plain`

مشخص کردیم که برنامه ما یک متن ساده می باشد و سپس با استفاده از کد

```
اضافه MailMessage فایل را به شی newmessg.Attachments.Add(at); کردهیم
```

و در انتهای فایل رو ارسال کردیم.

خوب تا این قسمت با نحوه‌ی ارسال یک Mail و همچنین ضمیمه کردن فایل به آن آشنا شدید  
قبل از اینکه بخواهیم با پروتکل Pop3 برای دریافت Email هایمان از Mailserver آشنا شویم  
به مسائل دیگر برنامه نویسی که پیش زمینه‌ی کار برای بکار گیری پروتکل POP3 می باشد می

## Sockets Helper Classes

## TCPClient

متد های کلاس TCPClient در ساختن برنامه های تحت شبکه استفاده می شود این متد ها از نوع مدل اتصال گرا هستند متد TCPClient مثل یک برنامه سوکت، اما تعداد زیادی از مراحل بهم پیوستن، یا امتر ها در این کلاس ساده شده است.

سه راه برای ساختن یک شی از نوع **TCPClient** وجود دارد

۱-استفاده از متده است (Connect) برای اتصال به یک Host خاص

```
TcpClient newclient = new TcpClient();
newclient.Connect("www.google.com", 8000);
```

در ابتدا یک شی **TCPClient** ساخته شده و سپس به یک آدرس و یک پورت محدود می شود.

۲-استفاده از یک آدرس محلی برای ساخت یک سوکت به صورت زیر

```
IPAddress ia = Dns.GetHostByName(Dns.GetHostName()).AddressList[0];
IPEndPoint iep = new IPEndPoint(ia, 1023);
TcpClient newClient = new TcpClient(iep);
```

۳-اگر می خواهید به یک آدرس و یک پورت مشخص متصل شوید دیگر نیازی به متده

Connect() نیست.

```
TcpClient newClient = new TcpClient("192.168.0.1", 8000);
```

شما می توانید با استفاده از **TCPClient** که ساخته اید اقدام به ارسال و دریافت داده ها نماید

متده GetStream() از شی Networkstream به شما اجازه میدهد که داده خود را بر حسب

بایت در یافت و یا ارسال کنید.

شی Networkstream دارای دو متده باشد که به صورت جریانی ، اطلاعات را از روی سوکت

می خواند و یا می نویسد این دو متده Read() و Write() می باشد.

مثال:

```
TcpClient newClient = new TcpClient("192.168.0.1", 8000);
NetworkStream ns = newClient.GetStream();
byte[] outbyte = Encoding.ASCII.GetBytes("Testing..");
byte[] inbyte;
ns.Write(outbyte, 0, outbyte.Length);
ns.Read(inbyte, 0, inbyte.Length);
string instring = Encoding.ASCII.GetString(inbyte);
```

```

Console.WriteLine(instring);
ns.Close();
newClient.Close();

```

حتما باید Close NetworkStream را کنید.

### TcpListener

یک سوکت در سمت TCPLISTENER شبیه یک Socket Client در سمت TcpClient برنامه Server می باشد.

این کلاس دارای سه سازنده می باشد

TCPLISTENER(int Port)..... مقید به یک پورت .....

TCPLISTENER(IPEndPoint ie) ..... endpoint مقید به یک

TCPLISTENER(IPAddress addr,int port) ip و پورت خاص

یکی از متدهای مورد استفاده در TCPLISTENER که برای دریافت تقاضا

از سمت TCPCLIENT می باشد و دیگری متده استفاده در Start برای گوش دادن به تقاضای ورودی می باشد.

```

TcpListener newserver = new TcpListener(9050);
newserver.Start();
TcpClient newclient = newserver.AcceptTcpClient();
NetworkStream ns = newclient.GetStream();
byte[] outbytes = Encoding.ASCII.GetBytes("Testing");
ns.Write(outbytes, 0, outbytes.Length);
byte[] inbytes = new byte[1024];
ns.Read(inbytes, 0, inbytes.Length);
string instring = Encoding.ASCII.GetString(inbytes);
Console.WriteLine(instring);
ns.Close();
newclient.Close();
newserver.Stop();

```

کدهای بالا فقط برای آشنایی از نحوه استفاده از TCPLISTENER و TCPCLIENT می باشد.

### UdpClient

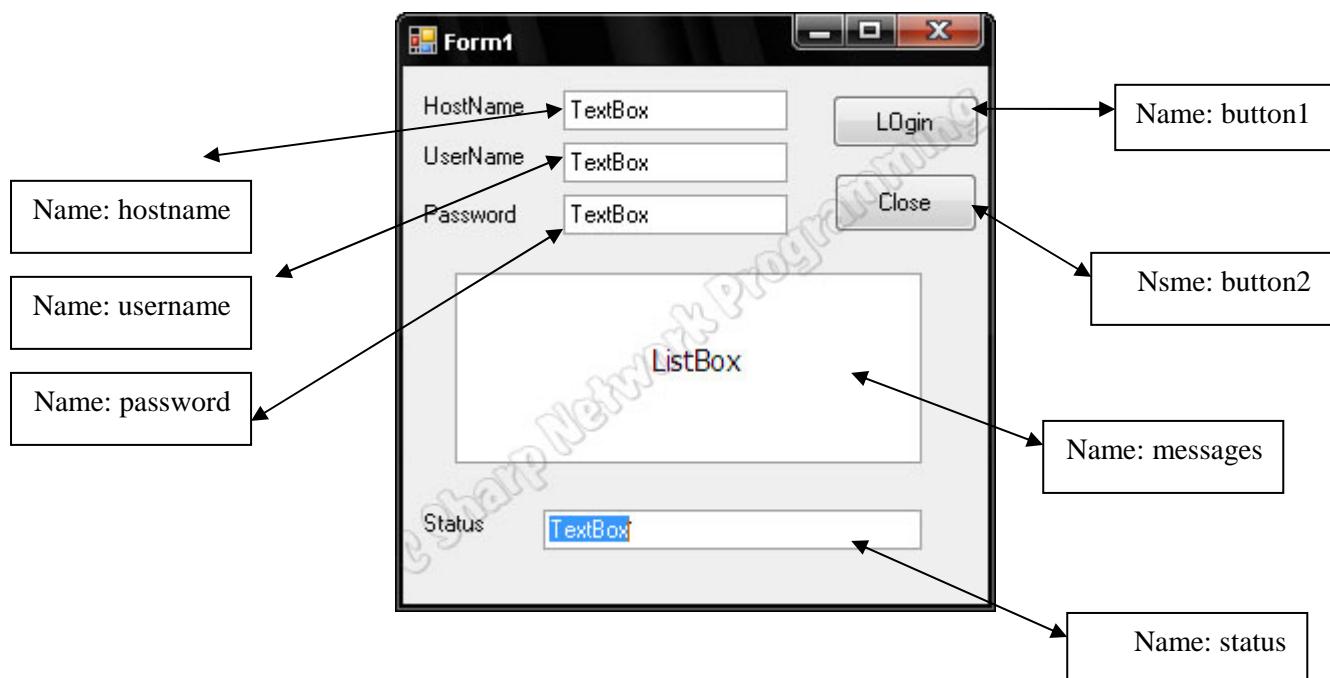
همچون کلاس های بالا برای ارتباطات بدون اتصال نیز کلاس UdpClient در نظر گرفته شده نحوه استفاده از این کلاس به صورت زیر است.

```

    UdpClient newconn = new UdpClient(8000);
    aIPEndPoint remoteClient = new IPEndPoint(IPAddress.Any, 0);
    byte recv = newconn.Receive(ref remoteClient);
    string data = Encoding.ASCII.GetString(recv);
    Console.WriteLine("from :{0}", remoteClient.ToString());
    newconn.Close();
  
```

اما در رابطه با پروتکل POP3 ، که اين پروتکل وظيفه اش دریافت نامه های ارسال شده از صندوق الکترونیکی شما می باشد. پورت استفاده شده عدد 110 که پورتی ثابت برای اين پروتکل می باشد که در برنامه های نامه خوان استفاده می شود در زیر، برنامه ای را بدین منظور طراحی خواهیم داد.

فرمی به شکل زیر طراحی کنید



کد مربوط به برنامه

```

using System;
using System.Drawing;
using System.IO;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;
using System.Windows.Forms;
  
```

```

namespace Pop3
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

    }
    1     private TcpClient mailclient;
    2     private NetworkStream ns;
    3     private StreamReader sr;
    4     private StreamWriter sw;
    5     class ShowMessage : Form
    {
    5     5     public ShowMessage(NetworkStream ns, string messnumber)
    6         {
    6             StreamReader sr = new StreamReader(ns);
    7             StreamWriter sw = new StreamWriter(ns);
    8             string response;
    9             Text = "Message " + messnumber;
    10            Size = new Size(400, 380);
    11            ShowInTaskbar = false;
    12            TextBox display = new TextBox();
    13            display.Parent = this;
    14            display.Multiline = true;
    15            display.Dock = DockStyle.Fill;
    16            display.ScrollBars = ScrollBars.Both;
    17            sw.WriteLine("retr " + messnumber); //Retrieve entire
    17             message
    18            sw.Flush();
    19            response = sr.ReadLine();
    20            while (true)
    20             {
    21                 response = sr.ReadLine();
    22                 if (response == ".")
    22                  break;
    23                 display.Text += response + "\r\n";
    23             }
    24             }
    24     void loginandretr()
    {
    25         string response;
    26         string from = null;
    27         string subject = null;
    28         int totmessages;
    29         try
    {
    30             mailclient = new TcpClient(hostname.Text, 110);
    30         }
    catch (SocketException)
    {
    31             status.Text = "Unable to connect to server";
    31         }
    }
}

```

```

                return;
            }
        32         ns = mailclient.GetStream();
        33         sr = new StreamReader(ns);
        34         sw = new StreamWriter(ns);
        35         response = sr.ReadLine(); //Get opening POP3 banner
        36         sw.WriteLine("User " + username.Text); //Send username
        37         sw.Flush();
        38         response = sr.ReadLine();
        39         if (response.Substring(0, 3) == "-ER")
        {
        40             status.Text = "Unable to log into server";
            return;
        }
        41         sw.WriteLine("Pass " + password.Text); //Send password
        42         sw.Flush();
        try
        {
        43             response = sr.ReadLine();
        }
        catch (IOException)
        {
        44             status.Text = "Unable to log into server";
            return;
        }
        45         if (response.Substring(0, 4) == "-ERR")
        {
        46             status.Text = "Unable to log into server";
            return;
        }
        47         sw.WriteLine("stat"); //Send stat command to get
number of messages
        48         sw.Flush();
        49         response = sr.ReadLine();
        50         string[] nummess = response.Split(' ');
        60         totmessages = Convert.ToInt16(nummess[1]);
        61         if (totmessages > 0)
        {
        62             status.Text = "you have " + totmessages + " messages";
        }
        else
        {
        63             status.Text = "You have no messages";
        }
        64             for (int i = 1; i <= totmessages; i++)
        {
        65         sw.WriteLine("top " + i + " 0"); //read header of each
message
        66         sw.Flush();
        67         response = sr.ReadLine();
        while (true)
        {
        68             response = sr.ReadLine();
            if (response == ".")
        
```

```

                break;
70             if (response.Length > 4)
71             {
72                 if (response.Substring(0, 5) == "From:")
73                     from = response;
74                 if (response.Substring(0, 8) == "Subject:")
75                     subject = response;
76             }
77         messages.Items.Add(i + " " + from + " " + subject);
    }

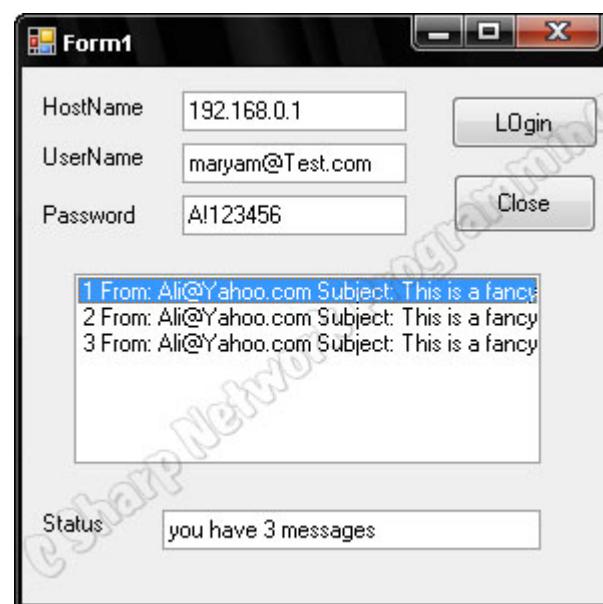
private void button1_Click(object sender, EventArgs e)
{
    status.Text = "Checking for messages...";
    loginandretr();
}

private void button2_Click(object sender, EventArgs e)
{
    if (ns != null)
    {
        sw.Close();
        sr.Close();
        ns.Close();
        mailclient.Close();
    }
    Close();
}

private void messages_MouseDoubleClick(object
sender, MouseEventArgs e)
{
    string text = (string)messages.SelectedItem;
    string[] textarray = text.Split(' ');
    ShowMessage sm = new ShowMessage(ns, textarray[0]);
    sm.ShowDialog();
}
}
}

```

پس اجرای برنامه ، در هر یک از TextBox مقادیر مورد نظر را وارد نموده و سپس بر روی Login کلیک کنید .



در انتهای اگر بر روی هر یک از Email های در یافته دابل کلیک کنید مشاهده می کنید که مورد نظر در یک کادر جدید باز شده و حاوی مشخصات Email شما می باشد و از جمله متن اصلی (Body)



اما توضیح قسمت های مختلف این برنامه

برای خواندن خطوطی از اطلاعات به شکل یک فایل متنی که از استاندارد کد گذاری UTF8 استفاده می کند.

کاراکترها را به صورت جریانی می نویسد ، نوع کد گذاری UTF8 خود کلاس Stream برای ورودی و خروجی به صورت بایت طراحی شده به خاطر همین از دو کلاس بالا برای خواندن و نوشتن استفاده کرده ایم چون یک Email همانطور که قبلا گفتیم بر مبنای کاراکتر می باشد.

در خط یک برنامه، یک شی از نوع TCPClient برای برقراری ارتباط با سرور تعریف کرده ایم در خط ۲ یک شی برای خواندن اطلاعات به صورت جریانی از روی سوکت تعریف کرده ایم و خط ۳ و ۴ هم شی هایی برای خواندن و نوشتن(Send) اطلاعات استفاده کرده ایم.

در خط ۲۴ یک تابع به نام Loginandretr تعریف شده که نقش اصلی در این برنامه را بازی می کند در خط ۳۰ از شی MailClient که ساخته بودیم آدرس IP سرور بعلاوه شماره پورت را دریافت کرده و اقدام به برقراری ارتباط می کنیم در خط ۳۱ در صورتی که قادر به برقراری ارتباط نباشد با پیغامی به کاربر گزارش داده می شود .

در خط ۳۲ پس از برقراری ارتباط ، داده ها ی مورد نیاز برای تراکنش لازم تا برقراری ارتباط کامل در یافت می شود در خط ۳۵ درون متغیر Response مقداری به شکل زیر ذخیره می شود

+ok Microsoft windows pop3 Servic Version 1.0.0

بعد در خط ۳۶ مقدار UserName به سمت سرور ارسال می شود در خط ۳۸ در متغیر Response مقدار +ok قرار می گیرد.

در خط ۴۱ نیز مقدار pass به سرور ارسال می شود اما در صورتی که درست نباشد در خط ۴۳ پیغام ERR logon Failure - درون متغیر Response قرار می گیرد.

و در خط ۴۵ بررسی می شود که آیا ۴ حرف اولی که در متغیر Response ذخیره شده با -ERR برابر می باشد یا نه. در صورتی که شرط درست باشد پیغام خطای داده واز تابع loginandretr خارج می شود.

اما در صورتی که همه چیز درست باشد در خط ۴۷ با استفاده از فرمان stat از سرویس دهنده Mail تقاضای مشخص کردن وضعیت صندوق پستی را خواهیم خواست قبل از ادامه کار به توضیحات زیر توجه کنید) این بخش، فرمان ها از کتاب اصول مهندسی اینترنت احسان ملکیان) پروتکل Pop3 دارای یکسری فرامین برای برقراری ارتباط با MailServer می باشد.  
هر فرمانی که کاربر صادر می کند با یکی از دو پاسخ +ok و یا ERR به معنای پذیرش و انجام فرمان، و یا ERR به معنای رد درخواست و عدم پذیرش می باشد.

فرمان	عملکرد
USER	برای مشخص کردن شناسه کاربری
PASS	برای مشخص کردن کلمه عبور
STST	تقاضای مشخص کردن وضعیت صندوق پستی (شامل تعداد و حجم نامه ها)
LIST	ارائه فهرستی از شاخص پیامهای رسیده
RTER	تقاضای در یافت نامه هایی که اندیس آنها مشخص شده است
DELE	تقاضای حذف نامه
NOOP	برای آگاهی از فعل بودن طرف مقابل
RSET	برگرداندن پیام های حذف شده
QUIT	نهایی کردن و اعمال تغییرات داده شده و قطع ارتباط

خوب اما متند() برای حذف داده های SW از حافظه می باشد.  
اما در خط ۴۹ پس از برقراری ارتباط کامل و قرار گرفتن پیغام +ok و با فاصله، تعداد نامه های رسیده ۳+ok در متغیر Response نوشته می شود که در خط ۵۰ عدد ۳ جدا شده و به عنوان شمارنده ی حلقه و همچنین نشان دادن تعداد نامه های رسیده در کادر Status نمایش داده

می شود. در خط ۶۰ تعداد نامه های رسیده و در خط ۶۸ به بعد اطلاعات هر نامه شامل تاریخ – متن فرستنده و .... استخراج می شود.

و در انتهای با استفاده از تابع ShowMessage اطلاعات توسط یک کادر جدید نمایش داده می شود.

## Remoting

.NET فریم ورک به آسانی قادر که با برنامه های توزیع شده ارتباط برقرار بکند. Remoting توسط.NET پشتیبانی می شود و به برنامه های کاربردی اجازه می دهد که کلاس ها و متدها خود را بر روی کامپیوتر های شبکه به اشتراک بگذارد(مشابه مفهوم Remoting با Host) با WebService فراخوانی کنند. برای اینکه بتوانیم (کلاس ها ، متدها ، اشیا ...) را بر روی شبکه بفرستیم باید آنها را Serialize کنیم (یا مثلا بخواهیم یک کلاس را در یک فایل ذخیره کنیم باشد آنرا Deserialize بکنیم. بعد آنرا ذخیره کنیم).

{ زمانی که شما قصد ارسال اطلاعاتی نظیر (کلاس و متدهای غیره ) از طریق شبکه داشته باشید شما فقط می توانید از طریق کابل شبکه مقدار String (صفرویک) را ارسال کنید در واقع فرآیند تبدیل یک Object را String به Serialization می گویند. و عمل تبدیل Object به String می گویند } دو کلاسی که برای انتقال اطلاعات به صورت Serializ مورد استفاده قرار می گیرند. (SoapFormatter و BinaryFormatter) می باشد.

## استفاده از کلاس **Serialization**

سه مرحله لازم برای Serialize کردن یک کلاس و ارسال آن بر روی شبکه لازم می باشد

۱- ساخت یک شی کتابخانه (library)

۲- نوشتن یک برنامه Sender (ایجاد یک کلاس Serialized برای ارسال کردن به صورت Stream (جریانی از صفر و یک ها))

۳- نوشتن یک برنامه Reciver برای خواندن داده های Stream و تبدیل دوباره آنها به حالت اول (متدها Object اولیه)

## ساختن کلاس **Serialized**

همانطور که قبلاً گفته‌یم هر کلاسی که بر روی شبکه ارسال می شود باید Serialize شود این کلاس باید حاوی بر چسب [Serializable] باشد در واقع این بر چسب به کامپایلر اعلام می کند که این کلاس قابل Serialization شدن می باشد. کلا عبارتهای که قبل از کلاس و یا متدهای نوشته می شوند به آنها Attribute می گویند.

در زیر نحوه ساختن یک کلاس Serialize نشان داده شده است.  
پروژه جدیدی را باز کرده در قسمت Solution Explorer راست کلیک کرده از گزینه **Add Class** را انتخاب کنید و کدهای زیر را در آن قرار دهید.

```
using System;
[Serializable]
public class SerialEmployee
{
    public int EmployeeID;
    public string LastName;
    public string FirstName;
    public int YearsService;
    public double Salary;
    public SerialEmployee()
    {
        EmployeeID = 0;
        LastName = null;
        FirstName = null;
        YearsService = 0;
        Salary = 0.0;
    }
}
```

این کلاس حاوی اطلاعات پایه ای استفاده می باشد

خوب ما مرحله اول را پیاده سازی کردیم اما مرحله دوم ساخت یک برنامه Sender می باشد  
بعد از مرحله ای یک شما باید داده های کلاسی که ساخته اید را Serialize کنید.

کلاس SoapFormatter برای تبدیل داده ها به Xml Serialize که در قالب یک فایل xml برای انتقال اطلاعات بین برنامه ها مورد استفاده قرار می گیرد برای اطلاعات می باشد. }  
بیشتر در این مورد به کتاب سید محمد هاشمیان مراجعه شود {  
از کلاس Serialize برای هم برای Deserialize کردن و هم برای BinaryFormatter کردن داده ها مورد استفاده قرار می گیرد.

این دو کلاس به ترتیب زیر می باشد NameSpace

```
using System.Runtime.Serialization.Formatters.Soap;
using System.Runtime.Serialization.Formatters.Binary;
```

به برنامه SoapTest توجه کنید کد های زیر را به پروژه قبلی که شروع کرده بودیم اضافه کنید

```
using System;
using System.IO;
using System.Runtime.Serialization;
using System.Runtime.Serialization.Formatters.Soap;
class SoapTest
{
    public static void Main()
    {
        SerialEmployee emp1 = new SerialEmployee();
        SerialEmployee emp2 = new SerialEmployee();
        emp1.EmployeeID = 1;
        emp1.LastName = "Blum";
        emp1.FirstName = "Katie Jane";
        emp1.YearsService = 12;
        emp1.Salary = 35000.50;
        emp2.EmployeeID = 2;
        emp2.LastName = "Blum";
        emp2.FirstName = "Jessica";
        emp2.YearsService = 9;
        emp2.Salary = 23700.30;
        Stream str = new FileStream(@"C:\soapttest.xml",
        FileMode.Create, FileAccess.ReadWrite);
        IFormatter formatter = new SoapFormatter();
        formatter.Serialize(str, emp1);
        formatter.Serialize(str, emp2);
        Console.ReadKey();
        str.Close();
```

```

    }
}

```

کلاس زیر قرار دارد NameSpace نیز در IFormatter

```
using System.Runtime.Serialization;
```

شی FileStream یک فایل XML را برای ذخیره سازی اطلاعات Serialize شده ایجاد می کند .

نکته : اگر پس از اجرای برنامه ، برنامه شما در خط

```
using System.Runtime.Serialization.Formatters.Soap;
```

پیغام خطا داد در پنجره Solution Explorer بر روی References کلیک راست کرده و گزینه Add Reference کلیک کنید در پنجره باز شده در تب .NET گزینه Add Reference

```
System.Runtime.Serialization.Formatters.Soap
```

را پیدا کرده و بر روی OK کلیک کنید تا مشکل برطرف شود .

هر چند برنامه بالا اطلاعات را Serialize کرد اما هنوز آنها را ارسال نکرده است .

برنامه زیر کامل شده بی برنامه بی بالا می باشد

```
using System;
[Serializable]
public class SerialEmployee
{
    public int EmployeeID;
    public string LastName;
    public string FirstName;
    public int YearsService;
    public double Salary;
    public SerialEmployee()
    {
        EmployeeID = 0;
        LastName = null;
        FirstName = null;
        YearsService = 0;
        Salary = 0.0;
    }
}
```

Add Class



```

using System;
using System.IO;
using System.Net;
using System.Net.Sockets;
using System.Runtime.Serialization;
using System.Runtime.Serialization.Formatters.Binary;
class BinaryDataSender
{
    public static void Main()
    {
        SerialEmployee emp1 = new SerialEmployee();
        SerialEmployee emp2 = new SerialEmployee();
        emp1.EmployeeID = 1;
        emp1.LastName = "Blum";
        emp1.FirstName = "Katie Jane";
        emp1.YearsService = 12;
        emp1.Salary = 35000.50;
        emp2.EmployeeID = 2;
        emp2.LastName = "Blum";
        emp2.FirstName = "Jessica";
        emp2.YearsService = 9;
        emp2.Salary = 23700.30;

        TcpClient client = new TcpClient("127.0.0.1", 9050);
        IFormatter formatter = new BinaryFormatter();
        NetworkStream strm = client.GetStream();
        formatter.Serialize(strm, emp1);
        formatter.Serialize(strm, emp2);
        Console.ReadKey();
        strm.Close();
        client.Close();
    }
}

```

برنامه بعد از اینکه داده ها را Serialize کرد آنها با استفاده از شی از نوع

TCPClient به سمت مقصد ارسال می کند .

نکته: برای ارسال داده ها به صورت Serialize شما باید حتما از روش اتصال گرا (TCP) استفاده

نمایید.

## واما مرحله سوم نوشتن برنامه Reciver

پروژی جدید را ایجاد نمایید پس از ایجاد پروژه در پنجره **Solution Explorer** بر روی نام پروژه خود کلیک راست کرده گزینه **Add→Class** را انتخاب کنید و کدهای زیر را اضافه کنید.

```
using System;
[Serializable]
public class SerialEmployee
{
    public int EmployeeID;
    public string LastName;
    public string FirstName;
    public int YearsService;
    public double Salary;
    public SerialEmployee()
    {
        EmployeeID = 0;
        LastName = null;
        FirstName = null;
        YearsService = 0;
        Salary = 0.0;
    }
}
```

پس از نوشتن کدها دوباره یک کلاس جدید ایجاد کنید و کدهای زیر را در آن قرار دهید

```
using System;
using System.Reflection;
using System.Runtime.Serialization;
sealed class OverrideBinder : SerializationBinder
{
    public override Type BindToType(string assemblyName, string typeName)
    {
        typeName = "SerialEmployee";
        assemblyName = Assembly.GetExecutingAssembly().FullName;

        Type typeToDeserialize = null;

        typeToDeserialize = Type.GetType(String.Format("{0}, {1}", typeName, assemblyName));

        return typeToDeserialize;
    }
}
```

دلیل نوشتن کد های بالا بعدا شرح داده می شود

حالا برای اینکه بتوانیم داده های ارسال شده از برنامه قبلی را دریافت کرده (Deserialize) به

صورت زیر عمل می کنیم.

```

using System;
using System.IO;
using System.Xml.Serialization;
using System.Net;
using System.Net.Sockets;
using System.Runtime.Serialization;
using System.Runtime.Serialization.Formatters.Binary;
using System.Reflection;

class BinaryDataRcvr
{

    public static void Main()
1     { TcpListener server = new
      TcpListener(IPAddress.Parse("127.0.0.1"), 9050);
2     server.Start();
      Console.WriteLine("Listing...");
3     TcpClient client = server.AcceptTcpClient();
4     NetworkStream strm = client.GetStream();
5     IFormatter formatter = new BinaryFormatter();
6     SerialEmployee empl;
7     SerialEmployee emp2;

8     formatter.Binder = new OverrideBinder();
9     empl = (SerialEmployee)formatter.Deserialize(strm);
      Console.WriteLine("Serialized.....\n");
10
      Console.WriteLine("empl.EmployeeID = {0}", empl.EmployeeID);
      Console.WriteLine("empl.LastName = {0}", empl.LastName);
      Console.WriteLine("empl.FirstName = {0}", empl.FirstName);

      Console.WriteLine("empl.YearsService={0}", empl.YearsService);
      Console.WriteLine("empl.Salary = {0}\n", empl.Salary);

      emp2 = (SerialEmployee)formatter.Deserialize(strm);

      Console.WriteLine("emp2.EmployeeID = {0}", emp2.EmployeeID);
      Console.WriteLine("emp2.LastName = {0}", emp2.LastName);
      Console.WriteLine("emp2.FirstName = {0}", emp2.FirstName);
      Console.WriteLine("emp2.YearsService ={0}", emp2.YearsService);
      Console.WriteLine("emp2.Salary = {0}", emp2.Salary);

      Console.ReadKey();
}

```

```

        strm.Close();
        server.Stop();
    }
}

```

در خط ۱ شی از نوع `TcpListener` ایجاد می شود در خط ۲ برنامه به خط گوش داده تا تقاضایی از سمت برنامه `Sender` را دریافت کند در خط ۳ به در خواست برنامه پاسخ داده می شود.

در خط ۴ داده های ارسال شده از روی سوکت خوانده شده و در شی از نوع `Stream` قرار داده می شود در خط ۵ شی از نوع `BinaryFormatter` برای `Deserialize` کردن داده ها تعریف می شود در خطوط ۶ و ۷ دو شی از روی کلاس `SerialEmployee` ساخته می شود.

اما خط ۸، هنگامی که یک برنامه را کامپایل می کنید، کد `MSIL` تولید شده در فایل هایی به نام اسembly ذخیره می شوند. این فایل ها دارای پسوند `Exe` و یا شامل کتابخانه ای از کلاسها و توابع برای استفاده در برنامه‌ی دیگر می باشد که این گونه فایلها دارای پسوند `dll` می باشد. در برنامه بالا چون برنامه اطلاعات را از برنامه دیگری دریافت می کند به اطلاعات مربوط به اسembly آن برنامه نیاز دارد که این اطلاعات به صورت زیر است.

`Name <,Culture = CultureInfo> <,Version = Major.Minor.Build.Revision> <,  
StrongName> <,PublicKeyToken>`

این اطلاعات با استفاده از کلاس `OverrideBinder()` به شی `formatter` نسبت داده می شود. کار استخراج نوع اسembly را دستور زیر از فضای نام `Using system.Reflection;` انجام می دهد.

```
assemblyName = Assembly.GetExecutingAssembly().FullName;
```

در خط ۹ نیز اطلاعات `serialize` شده `Deserialize` می شود و به انواع مختلف از نوع کلاس `Object` `Deserialize` می شود. (چون خروجی متده `SerialEmployee` با استفاده از تبدیل مبنا آنرا به یک یک `Object` تبدیل کرد)

```
emp1 = (SerialEmployee)formatter.Deserialize(strm);
```



از خط ۱۰ به بعد نیز این اطلاعات چاپ می شود.

زمانی که شما از دو برنامه قبلی برای ارسال و دریافت اطلاعات استفاده می کنید انتظار دارید که همه ی داده های شما دریافت شده و Deserialize شوند اما این یک الزام نیست به این معنی مثلا زمانی که متدهای Deserialize() مقدار فضای لازم (بایت) برای انجام کار را در اختیار نداشته باشد یک استثنای خطا می دهد برای این کار شما باید مقدار فضای هر داده می Serialize شده را به برنامه Reciver ارسال کنید شما می توانید این کار را با استفاده از شی از نوع MemoryStream انجام دهید. این شی داده می Serialize شده را در حافظه نگه می دارد . که می توان سایز داده های ذخیره شده در آن را بدست آورد.

به برنامه کامل شده DataSender زیر توجه کنید

پروژه جدیدی را ایجاد کنید یک کلاس به برنامه می خود اضافه کنید

```
using System;
[Serializable]
public class SerialEmployee
{
    public int EmployeeID;
    public string LastName;
    public string FirstName;
    public int YearsService;
    public double Salary;
    public SerialEmployee()
    {
        EmployeeID = 0;
        LastName = null;
        FirstName = null;
        YearsService = 0;
        Salary = 0.0;
    }
}
```

---

```
using System;
using System.IO;
using System.Net;
using System.Net.Sockets;
using System.Runtime.Serialization;
using System.Runtime.Serialization.Formatters.Soap;
class BetterDataSender
{
    public void SendData(NetworkStream strm, SerialEmployee emp)
    {
        IFormatter formatter = new SoapFormatter();
        MemoryStream memstrm = new MemoryStream();
        formatter.Serialize(memstrm, emp);
        byte[] data = memstrm.GetBuffer();
        int memsize = (int)memstrm.Length;
        byte[] size = BitConverter.GetBytes(memsize);
        strm.Write(size, 0, 4);
        strm.Write(data, 0, memsize);
        strm.Flush();
        memstrm.Close();
    }
    public BetterDataSender()
    {
        SerialEmployee emp1 = new SerialEmployee();
        SerialEmployee emp2 = new SerialEmployee();
        emp1.EmployeeID = 1;
        emp1.LastName = "Blum";
        emp1.FirstName = "Katie Jane";
        emp1.YearsService = 12;
        emp1.Salary = 35000.50;
        emp2.EmployeeID = 2;
        emp2.LastName = "Blum";
        emp2.FirstName = "Jessica";
        emp2.YearsService = 9;
        emp2.Salary = 23700.30;
        TcpClient client = new TcpClient("127.0.0.1", 9050);
        NetworkStream strm = client.GetStream();
        SendData(strm, emp1);
        SendData(strm, emp2);
        strm.Close();
        client.Close();
    }
    public static void Main()
    {
        BetterDataSender bds = new BetterDataSender();
        Console.ReadKey();
    }
}
```

این خط ; باعث می شود که اطلاعات formatter.Serialize(memstrm, emp) را در حافظه قرار بگیرد.

متدهای GetBuffer() اطلاعاتی که در شی `MemoryStream` قرار دارد را استخراج می کند.

در فضای نام `System.IO`; قرار دارد `MemoryStream`

خط `int memsize = (int)memstrm.Length;` سایز یا همان مقدار اطلاعات موجود در شی `memstrm` را استخراج می کند.

در خط `byte[] size = BitConverter.GetBytes(memsize);` فضای لازم بر حسب بایت توسط متدهای `BitConverter` محاسبه می شود. و به سمت برنامه Receiver تحویل داده می شود.

برای نوشتن برنامه Reciver پروژه‌ی جدیدی را ایجاد کنید و دو کلاس به برنامه اضافه کرده و کدهای زیر را در آن قرار دهید

### Class1

```
using System;
[Serializable]
public class SerialEmployee
{
    public int EmployeeID;
    public string LastName;
    public string FirstName;
    public int YearsService;
    public double Salary;
    public SerialEmployee()
    {
        EmployeeID = 0;
        LastName = null;
        FirstName = null;
        YearsService = 0;
        Salary = 0.0;
    }
}
```

---

## Class 2

```

using System;
using System.Reflection;
using System.Runtime.Serialization;
sealed class OverrideBinder : SerializationBinder
{
    public override Type BindToType(string assemblyName, string
typeName)
    {
        typeName = "SerialEmployee";
assemblyName = Assembly.GetExecutingAssembly().FullName;
Type typeToDeserialize = null;
typeToDeserialize = Type.GetType(String.Format("{0}, {1}",
typeName, assemblyName));

        return typeToDeserialize;
    }
}

```

---

```

using System;
using System.IO;
using System.Net;
using System.Net.Sockets;
using System.Runtime.Serialization;
using System.Runtime.Serialization.Formatters.Soap;
class BetterDataRcvr
{
    private SerialEmployee RecvData(NetworkStream strm)
    {
        MemoryStream memstrm = new MemoryStream();
        byte[] data = new byte[4];
        int recv = strm.Read(data, 0, 4);
        int size = BitConverter.ToInt32(data, 0);
        int offset = 0;
        while (size > 0)
        {
            data = new byte[1024];
            recv = strm.Read(data, 0, size);
            memstrm.Write(data, offset, recv);
            offset += recv;
            size -= recv;
        }
        IFormatter formatter = new SoapFormatter();
        formatter.Binder = new OverrideBinder();
        memstrm.Position = 0;
        SerialEmployee emp =
(SerialEmployee)formatter.Deserialize(memstrm);
        memstrm.Close();
        return emp;
    }
    public BetterDataRcvr()

```

```

{
    TcpListener server = new TcpListener(9050);
    server.Start();
    Console.WriteLine("listening... ");
    TcpClient client = server.AcceptTcpClient();
    NetworkStream strm = client.GetStream();
    SerialEmployee empl = RecvData(strm);
    Console.WriteLine("empl.EmployeeID = {0}", empl.EmployeeID);
    Console.WriteLine("empl.LastName = {0}", empl.LastName);
    Console.WriteLine("empl.FirstName = {0}", empl.FirstName);
    Console.WriteLine("empl.YearsService={0}", empl.YearsService);
    Console.WriteLine("empl.Salary = {0}\n", empl.Salary);
    SerialEmployee emp2 = RecvData(strm);
    Console.WriteLine("emp2.EmployeeID = {0}", emp2.EmployeeID);
    Console.WriteLine("emp2.LastName = {0}", emp2.LastName);
    Console.WriteLine("emp2.FirstName = {0}", emp2.FirstName);
    Console.WriteLine("emp2.YearsService={0}", emp2.YearsService);
    Console.WriteLine("emp2.Salary = {0}", emp2.Salary);
    strm.Close();
    server.Stop();
}

public static void Main()
{
    BetterDataRcvr bdr = new BetterDataRcvr();
    Console.ReadKey();

}
}

```

برنامه بالا با استفاده از متده RecvData داده های ارسال شده از برنامه Sender را دریافت می کند. مقدار داده های ارسال شده ۴ بایت می باشد این مقدار نشان می دهد که چند بایت به خاطر عمل Serialize شدن ، ایجاد شده است. زمانی که همه بایت ها دریافت شد اطلاعات وارد شی Deserialize می شود و بعد عمل انجام می شود.

البته به غیر از این مباحثت گفته شده مباحثت دیگری همچون

Asynchronous Sockets

به کار گیری Threads در برنامه نویسی تحت شبکه

ICMP

SNMP

HTTP

Active Directory

Security

و... وجود دارد.

امیدوارم که این کتاب برای شما مفید واقع شده باشد و با تحقیقات بیشتر در مورد مباحثت گفته

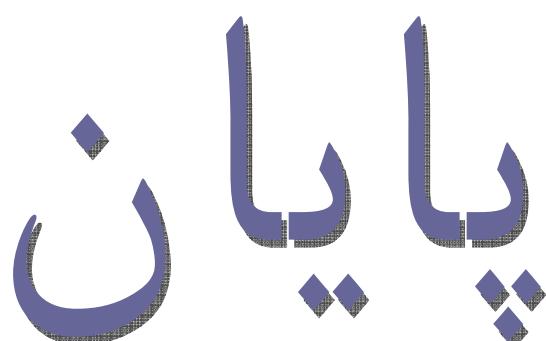
شده بتوانید برنامه های مفید و کاربردی بهتری بنویسید.

صبر کلید رسیدن است و کامیابی

سرانجام کسی که صبر کند <<امام علی(ع)>>

بهترینها را در خودتان خواهید یافت

وقتی در دیگران خوبی بجوئید <<مارتین والش>>



تابستان ۱۳۸۹