Dot Net Networks & TCP/IP Programming

By FADI Abdel-qader, Jordan

احترف برمجة الشبكات وبروتوكول TCP/IP — النسخة الإلكترونية
With Microsoft Visual C# & VB.NET

النسخة الإلكترونية هي ملخص لنسخة الورقية وهي نسخة مجانية بإمكانك توزيعها إلكترونيا كما تشاء، يمنع بيع النسخة الإلكترونية بأي شكل من الأشكال كما يمنع بيعها بصورة ورقية ...

النسخة الورقية هي النسخة المعتمدة من الكتاب مع تحسين وإضافة الكثير من الدروس التعليمية كما سوف تدعم البرمجة بجميع للغات الدوت نيت VB.NET & C#.NETبالإصدارتين 2003 و 2005 ...

لطلب أ والاستفسار أو التوزيع يرجى الاتصال على احد لعناوين التالية

Mobile : +962796284475
Phone: +96265055999
E-mail: fadi822000@yahoo.com
BOX: 311 Mail Code 11947 Tariq—Amman—Jordan

My online CV: http://spaces.msn.com/members/csharp2005/

الكتاب الأول في هذا المجال باللغة العربية
— النسخة الإلكترونية—

Mobile : +962796284475
Phone: +96265055999
E-mail: fadi822000@yahoo.com

تأليف فادي عبد القادر - الأردن

Version 1.0, 2005 - 2006

المقدمة:

يناقش هذا الكتاب أهم الأمور المتعلقة ببرمجة الشبكات باستخدام لغات الدوت نيت بأسلوب سلس وبسيط إذ ينتقل بك من المستوى المبتدئ إلى المتوسط إلى المتقدم بأسلوب جميل وممتع ، و يبدأ الكتاب بمقدمة عامة عن TCP/IP Models وتطبيقات Client/Server باستخدام للغات الدوت نيت كما ويحتوي على شرح مفصل عن Socket Programming وال Network Layer Protocols وبناء أنظمة متقدمة باستخدام ال Multicasting كأنظمة المؤتمرات وبرمجيات ال Remote Desktop وأنظمة التحكم عن بعد وغيرها ، كما ويحتوي على شرح مفصل لأهم برتوكولات ال Application Layer واستخداماتها في برمجيات الشبكات ، وأخيرا شرح مفصل عن طرق الحماية ووضع الصلاحيات والسياسات في برمجيات الشبكات ...

الإهداء:

اهدي هذا الكتاب إلى الطلاب والمبرمجين العرب في جميع أنحاء العالم ...

ملخص الفصول في النسخة الإلكترونية والنسخة الورقية:

النسخة الإلكترونية:

Chapter 1: An Overview on Networks & TCP/IP Programming

Chapter 2: Managed I/O: Streams, Readers, and Writers

Chapter 3: The Socket & Network Layer Programming

Chapter 4: Advanced Multicasting Systems

Chapter 5: Application Layer Programming

Chapter 6: Network Security Programming

النسخة الورقية:

سوف تحتوي النسخة الورقية على الكثير من الإضافات الجديدة ، إذ تتكون من ثلاثة أجزاء كما يلي:

الجزء الأول: ويتكون من فصلين: مهم لمبرمجي ال VB6 وال C++6 وال Java وجميع المبرمجين المنتقلين إلى الدوت نيت

Part 1: Preparation to Dot Net

Chapter 1: Dot Net Infrastructure & OOP

Chapter 2:ADO.NET

الجزء الثاني: ويتكون من ثمانية فصول: احترف بناء أنظمة الشبكات المتقدمة باستخدام تقنيات الدوت نيت

Part 2: .Net Networks & TCP/IP Programming

Chapter 3: An Overview on Networks & TCP/IP Programming

Chapter 4: Managed I/O: Streams, Readers, and Writers

Chapter 5: The Socket & Network Layer Programming

Chapter 6: Advanced Multicasting Systems

Chapter 7: Application Layer Programming

Chapter 8: Remotting & Web Services

Chapter 9: .Net Security Overview & Network Security Programming

Chapter 10: Performance Improvement & Multithreading

الجزء الثالث: تطبيقات ومشاريع عملية على أنظمة الشبكات باستخدام الدوت نيت.

Part 3: .Net Networks Applications & Real Projects

بالإضافة إلى CD يحتوي على الأمثلة والمشاريع المطروحة في الكتاب...

فهرس المحتويات لنسخة الإلكترونية:

Chapter 1
An Overview on Network & TCP/IP Programming

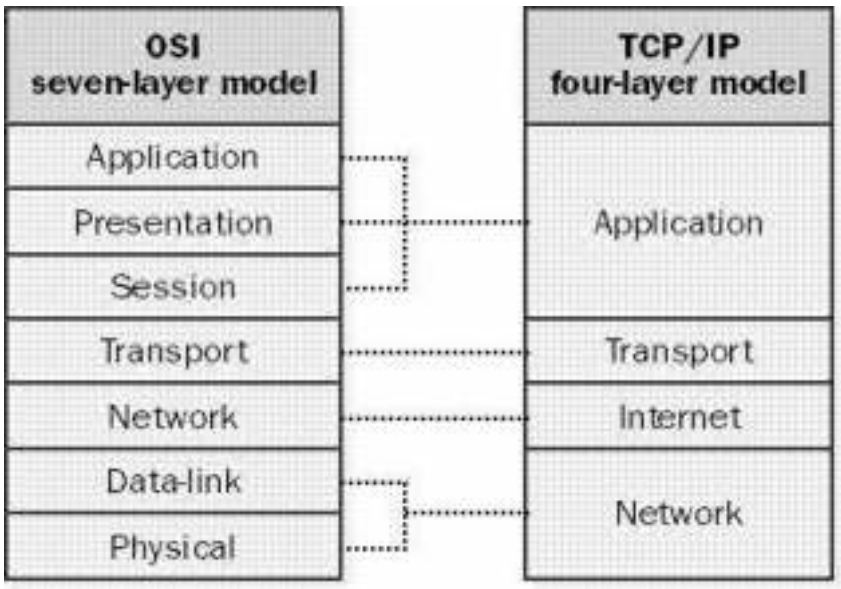
Introduction to Network and TCP/IP Programming

A. Introduction to TCP/IP Layers Programming
B. Connection Oriented Via TCP Overview
C. Connection Less Via UDP Overview
D. Streaming & Threading Overview
E. IP Multicasting Overview

بسم الله الرحمن الرحيم

1.1 : مقدمة في برمجة الشبكات و بروتوكول TCP/IP

من المعروف أن الشبكة هي مجموعة من الأجهزة متصلة مع بعضها عبر وسيلة اتصال معينة ومن هنا سيندرج لدينا التقسيم المعروف لمنظمة OSIلعملية الاتصال والتي تمر بسبعة طبقات لكل طبقة منها وظيفة معينة وتم اختصارها إلى أربعة طبقات(خمسة في بعض الكتب) في برتوكول TCP/IP وتبين الصورة المرفقة هذه الطبقات:



لإجراء عملية الاتصال بين Client و Server يلزم ما يلي :

في الجهاز المرسل Client :

تبدأ عملية توليف الرسالة المرسلة في ال Application Layer ووظيفتها هنا التعامل مع الرسالة نفسها وتحويلها من صيغة نصية إلى Data يمكن إرسالها عبر الشبكة ، ففي برمجيات الدردشة Chat يتم تحويل النص المكتوب إلى ASCII Code ثم إلى مجموعة من Binary Code توضع في مصفوفة لتجهيزها وإرسالها عبر Socket والذي يربط طبقة ال Application Layer مع بقية طبقات ال TCP/IP وهنا توضيح هذه الخطوة ...

وتبين طريقة تحويل الرسالة المكتوبة كنص باستخدام ال ASCIIEncoding Class إلى Byte Array :

C#

```
String str=Console.ReadLine();
ASCIIEncoding asen= new ASCIIEncoding();
byte[] ba=asen.GetBytes(str);
```

VB.NET

```
Dim str As String = Console.ReadLine
Dim asen As ASCIIEncoding = New ASCIIEncoding
Dim ba As Byte() = asen.GetBytes(str)
```

في نموذج OSI تم تقسيم ال upper Layers إلى ثلاثة طبقات

Application لتعامل مع البرنامج نفسه أو ما يسمى User Interface

Presentation تمثيل البيانات المرسلة وهي كما ظهرت سابقا بتحويل البيانات إلى الASCII .

Session وفيها البدء بعملية التخاطب بين الجهازين و التعريف ببعضهم البعض(فتح الجلسة) ...

أما في بروتوكول ال TCP/IP فكتفا بوجود طبقة Application والتي تقوم بعمل الطبقات الثلاث الأولى في OSI ، في session Layer يتم التعرف وفتح الجلسة بعدة خطوات وهي كما يلي :

1- إجراء الاتصال المبدئي بجهاز server عبر ال IP و ال Port المحدد وذلك بعد تحديد عملية الاتصال سواء عبر UDP أو عبر TCP

2- التعريف بنفسه وعمل ال Authentication إذا تطلب جهاز الServer ذلك

3- قبول أو رفض الجلسة ويتم ذلك بإرسال الموافقة على فتح الجلسة أو رفضها

4- بدأ الجلسة وقيام الServer بعمل Listening على الPort الخاص بالبرنامج

عندما يتم الموافقة على فتح الجلسة والبدء بعملية التخاطب يقوم جهاز المرسل Client بتحميل الرسالة إلى الطبقة الأخرى وهي هنا طبقة Transport وفي هذه الطبقة يتم تحديد طبيعة الاتصال سواء عبر TCP - Connection Protocol أو عبر ال UDP - Connectionless Protocol ففي البروتوكول الأول يتم تحديد طرفين وهما المرسل والمستقبل وPort الاتصال أما ال UDP فيتم تحديد الطرف المرسل و المستقبل (اختياري) أي انه يمكن عمل ال Broadcast بدون تحديد جهة معينة

لاستقبال الرسالة أي أن أي شخص يقوم بتصنت عبر هذا الListening Port يستطيع استقبال الرسالة ، وهنا مثال يوضح عمل هذه الطبقة باستخدام ال TCP Protocol :

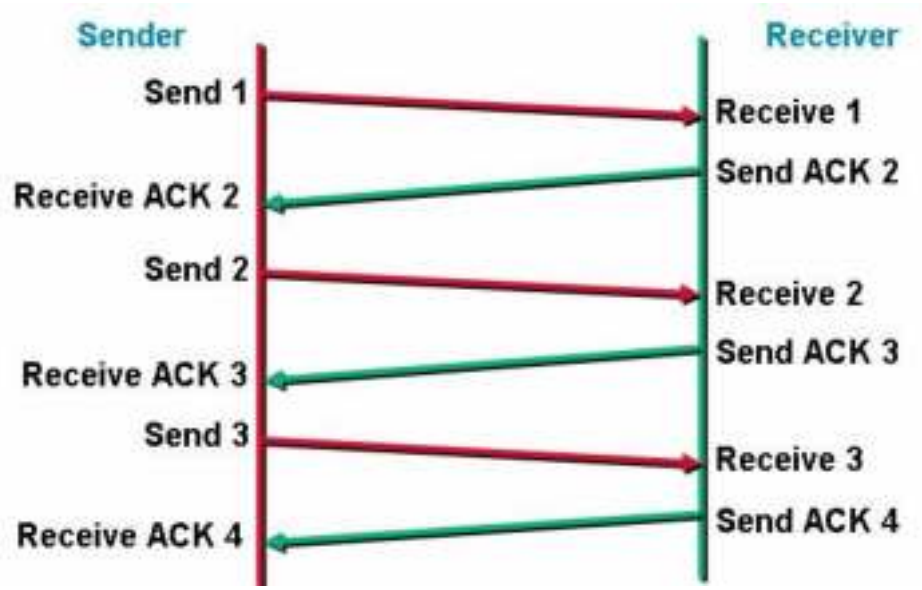
C#:

```
TcpClient tcpclnt = new TcpClient();
tcpclnt.Connect("192.168.0.2",8001);
```

VB.NET:

```
Dim tcpclnt As TcpClient = New TcpClient
tcpclnt.Connect("192.168.0.2", 8001)
```

وتتم عملية التحقق من الوصول في ال TCP كما هو موضح في الشكل التالي:



إذ أنه في كل عملية إرسال يتم إرسال رد Acknowledgment إلى المرسل يخبره فيها بوصول الرسالة، ويرسل في ال Acknowledgment Header رقم ال Packet الذي تم استقباله بنجاح ويسمى ال ACK ID.

ولإرسال الرسالة عبر الشبكة نستخدم في الدوت نت Class جاهز يقوم بهذه العملية ويسمى NetworkStream وهو المسئول عن التعامل مع وسيلة الاتصال وإرسال الرسالة إلى الطرف المعني بشكل Stream Data ، أو باستخدام ال Socket نفسه (انظر الفصل الثالث) وكمثال على ذلك:

C#:

```
NetworkStream mynetsream = tcpclnt.GetStream ();
StreamWriter myswrite = new StreamWriter (mynetsream);
myswrite.WriteLine("Your Message");
```

VB.NET:

```
Dim mynetsream As NetworkStream = tcpclnt.GetStream
Dim myswrite As StreamWriter = New StreamWriter(mynetsream)
myswrite.WriteLine("Your Message")
```

وبعد ذلك تسلم إلى Network Layer إذ تتم عنونة الرسالة ووضع عنوان المرسل والمستقبل عليها وتسلم إلى الطبقة الأدنى ليتم إرسالها عبر ال Physical Tunnel ومن الأمثلة عليها IP,IPv6,ARB-Address Resolution Protocol ...

أما بنسبة للجهاز المستقبل ال Server فيقوم بالمرور على نفس الطبقات ولكن بالعكس حيث يستلم كرت الشبكة ال Bits لتحول إلى Data link ثم Network ثم Transport ثم Application ومنها تحول من Binary إلى ASCII ومن ASCII إلى Text ..

وهذه الكود يوضح مبدأ عمل ال Server :

C#:

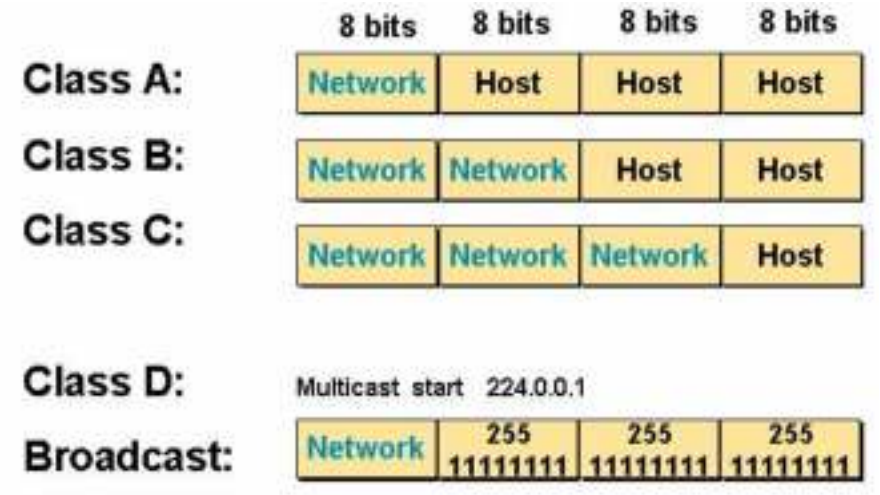
```
TcpListener myList=new TcpListener("127.0.0.1",8001);

myList.Start();
Socket s=myList.AcceptSocket();
byte[] b=new byte[100];
int k=s.Receive(b);
for (int i=0;i<k;i++)
Console.Write(Convert.ToChar(b[i]));
s.Close();
```

VB.NET:

```
Dim myList As TcpListener = New TcpListener("127.0.0.1", 8001)
myList.Start()
Dim s As Socket = myList.AcceptSocket
Dim b(100) As Byte
Dim k As Integer = s.Receive(b)
Dim i As Integer = 0
While i < k
Console.Write(Convert.ToChar(b(i)))
System.Math.Min(System.Threading.Interlocked.Increment(i), i - 1)
End While
s.Close()
```

Connectionless Sockets Via UDP : 1.2

تحدثنا سابقا عن ال TCP – Connection Oriented Protocol وبينا أن بروتوكول ال TCP هو بروتوكول موجه وهذا يعني انه يلزم احتواء ال Header الخاص به على عنوان المرسل و عنوان المستقبل كما يلزم أيضا القيام بعمليات التحقق Authentication و يدعم عمليات التحقق من الوصول باستخدام ال Acknowledgment و التسليم بشكل الصحيح .... لكن ماذا لو كان كل ذلك غير مهم بنسبة لك إذ تريد من برنامجك أن يقوم بعملية بث إذاعي Broadcast لرسالتك ولا يهمك من سوف يستلم الرسالة و أن السرعة في الإرسال و الاستقبال هي الهدف الأساسي إذا وجب عليك ترك بروتوكول ال TCP والتوجه نحو ال UDP User Datagram Protocol ويسمى أيضا بال Connectionless Protocol في هذا البروتوكول تستطيع عمل ما يسمى بال Broadcast و ال Multicast (Broad-يعني الإرسال إلى الكل و Multi-يعني الإرسال إلى مجموعة وال Uni-يعني الإرسال لواحد فقط) يوجد شرط وحيد يلزم أن تأخذه بعين الاعتبار عند استخدام ال UDP لعملية البث باستخدام Broadcast وهو أن الشبكة التي تريد عمل بث لها تتصل معها بشكل مباشر Direct Connection أي بدون وجود Router بينك وبين المستقبل إذ أن ال Router يمنع عمليات البث الإذاعي Broadcast حيث يلزم أن تكون الشبكة ضمن ال Range Class سواء A أو B أو C ، ومن المعروف أن ال IP Address مقسم إلى جزئيين الأول مخصص لشبكة Network والثاني مخصص لل HOST وكما هو موضح في الشكل التالي:



مثلا لعمل Broadcast إلى ال 10.0.0.0 – 255.0.0.0 : Address يتم ذلك كما يلي: 10.255.255.255 حيث أن الخانة الأولى (10) هي ال Network ID ويجب أن يبقى الجزء الخاص بال Network ID كما هو ويوضع ال 255 في جميع الخانات الخاصة بال HOST ID...

لاستخدام ال UDP في الدوت نيت يلزم أولا تعريف System.Net Name Space و ال System.Net.Socket لاحظ انه في ال TCP كان يلزم تعريف رقم الPort والعنوان للجهاز المستقبل أما في ال UDP فتستطيع تعريفه كما هو في TCP كما وتستطيع عمل Broadcast باستخدام IPAddress.Any بعد اشتقاق كائن من الكلاس IPEndPoint (وتعني نقطة الهدف) وتستطيع أيضا عدم تحديد رقم الPort باستخدام ال Method Bind حيث يتم تعريفها ب 0 ...

في المثال التالي يتم فتح الPort 5020والتصنت عليها ثم استلام الرسالة عبر هذا الPort وتوزيعها على الكل بدون تحديد رقم ال Port حيث يتم تسليمها على الPort المخصص لعملية البرود كاست وهو الPort صفر:

C#:

```
IPEndPoint ipep = new IPEndPoint(IPAddress.Any, 5020);
```

VB.NET

```
Dim ipep As IPEndPoint = New IPEndPoint(IPAddress.Any, 5020)
```

لتحديد نوع البرتوكول المستخدم يتم ذلك كما يلي:

C#:

```
Socket newsock = new Socket(AddressFamily.InterNetwork,
SocketType.Dgram, ProtocolType.Udp)
```

VB.NET

```
Dim newsock As Socket = New Socket(AddressFamily.InterNetwork,
SocketType.Dgram, ProtocolType.Udp)
```

ثم تمرير نقطة الهدف ورقم الPort إلى الميثود Send ...

ال Bind Method يتم وضعها في الطرف المستقبل فقط إذ تربط ال IP Address ورقم ال Port بال Socket :

C#:

```
newsock.Bind(ipep);
```

VB.NET:

```
newsock.Bind(ipep)
```

الآن تم استقبال الرسالة ونريد بثها إلى كل من يتصل مع الServer على الPort السابقة ولعمل ذلك يلزم أولا تعريف نقطة الهدف كما يلي :

C#:

```
IPEndPoint sender = new IPEndPoint(IPAddress.Any, 0);
EndPoint Remote = (EndPoint)(sender);
```

VB.NET:

```
Dim sender As IPEndPoint = New IPEndPoint(IPAddress.Any, 0)
Dim Remote As EndPoint = CType((sender), EndPoint)
```

لاحظ أن عنوان نقطة الهدف هو Any ورقم الPort صفر وهذا يعني إرسال الرسالة المستلمة إلى الكل وبما فيهم الشخص مرسل الرسالة و الServer

هنا يتم استلام الرسالة من الServer إلى الServer مرة أخرى عبر الشبكة:

C#:

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

VB.NET:

```
recv = newsock.ReceiveFrom(data, Remote)
```

لطباعة عنوان مرسل الرسالة و الرسالة نفسها:

C#:

```
Console.WriteLine("Message received from {0}:", Remote.ToString());
Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));
```

VB.NET:

```
Console.WriteLine("Message received from {0}:", Remote.ToString)
Console.WriteLine(Encoding.ASCII.GetString(Data, 0, recv))
```

نقوم هنا بإرسال رسالة ترحيبية لكل جهاز جديد يشبك على الServer نخبره بها انه تم الموافقة على دخوله ضمن الأجهزة:

C#:

```
string welcome = "Welcome Customer ...";
data = Encoding.ASCII.GetBytes(welcome);
newsock.SendTo(data, data.Length, SocketFlags.None, Remote);
```

VB.NET:

```
Dim welcome As String = "Welcome Customer ..."
Data = Encoding.ASCII.GetBytes(welcome)
newsock.SendTo(Data, Data.Length, SocketFlags.None, Remote)
```

هنا Infinity Loop الهدف منه هو عند استقبال أي رسالة في أي وقت من قبل أي جهاز يقوم الServer باستلامها وتسليمها إلى كل من هو على الشبكة ... إذا أردت تحديد عدد معين من الرسائل المستلمة تستطيع تغيير True في ال infinity loop إلى أي رقم تريده..

C#:

```
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);
 }
server.Close();
```

VB.NET:

```
While True
Data = New Byte(1024) {}
recv = newsock.ReceiveFrom(Data, Remote)
Console.WriteLine(Encoding.ASCII.GetString(Data, 0, recv))
newsock.SendTo(Data, recv, SocketFlags.None, Remote)
End While
server.Close()
```

هنا يتم إغلاق ال Socket في حالة إذا تم الخروج من Infinity Loop و لن يتم الوصول إلى هذه النقطة إلا إذا تم مقاطعته بوضع Break ضمن ال Infinity Loop وفق شرط معين أي انه في حالة استقبال رسالة أو نص رسالة معينة سيتم الخروج من Loop وسيتم إغلاق ال Socket وهذا يعني انك تستطيع إغلاق ال Server عن بعد كما يمكنك وضع جملة تشغيل أي ملف تنفيذي على الServer في حالة ورود نص معين وهكذا .

C#:

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

class SimpleUdpSrvr
{
   public static void Main()
   {
      int recv;
      byte[] data = new byte[1024];
      IPEndPoint ipep = new IPEndPoint(IPAddress.Any, 5020);
      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 Customer ...";
        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);
        }
     }
}
```

VB.NET:

```
Imports System
Imports System.Net
Imports System.Net.Sockets
Imports System.Text

Class SimpleUdpSrvr

    Public Shared Sub Main()
        Dim recv As Integer
        Dim data(1024) As Byte
        Dim ipep As IPEndPoint = New IPEndPoint(IPAddress.Any, 5020)
        Dim newsock As Socket = New Socket(AddressFamily.InterNetwork,
SocketType.Dgram, ProtocolType.Udp)

        newsock.Bind(ipep)
        Console.WriteLine("Waiting for a client...")
        Dim sender As IPEndPoint = New IPEndPoint(IPAddress.Any, 0)
        Dim Remote As EndPoint = CType((sender), EndPoint)
        recv = newsock.ReceiveFrom(data, Remote)
        Console.WriteLine("Message received from {0}:", Remote.ToString)
        Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv))
        Dim welcome As String = " Welcome Customer ..."
        data = Encoding.ASCII.GetBytes(welcome)
        newsock.SendTo(data, data.Length, SocketFlags.None, Remote)
        While True
           data = New Byte(1024) {}
           recv = newsock.ReceiveFrom(data, Remote)
```

```
            Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv))
            newsock.SendTo(data, recv, SocketFlags.None, Remote)
        End While
    End Sub
End Class
```

الآن الجزء الخاص بال Client ، يقتصر العمل هنا على قيام ال Client بإنشاء جلسة مع الServer وذلك بعد تعريفه بال IPEndPoint ورقم الPort وكما تم في السابق إلا أن الاختلاف هو في الوظيفة إذا يقتصر فقط على استقبال الرسالة من الServer وإرسال أي رساله له عبر الPort المخصص للقيام بهذه العملية انظر الكود التالي :

C#:

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

class SimpleUdpClient
{

    public static void Main()

{

 byte[] data = new byte[1024]; string input, stringData;

IPEndPoint ipep = new IPEndPoint( IPAddress.Parse("127.0.0.1"), 5020);

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

 يظهر الرسالة التاليةServerفي حالة فقدان الاتصال مع ال
  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];
 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)لكي تستطيع إرسال عدد غير محدد من الرسائل
 {
```

```
  input = Console.ReadLine();
  Exit   في حالة إذا أردت إنهاء الجلسة اكتب

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();

}

}
```

VB.NET:

```
Imports System
Imports System.Net
Imports System.Net.Sockets
Imports System.Text

Class SimpleUdpClient

    Public Shared Sub Main()
        Dim data(1024) As Byte
        Dim input As String
        Dim stringData As String
        Dim ipep As IPEndPoint = New
IPEndPoint(IPAddress.Parse("127.0.0.1"), 5020)
        Dim server As Socket = New Socket(AddressFamily.InterNetwork,
SocketType.Dgram, ProtocolType.Udp)
        Dim welcome As String = "Hello, are you there?"
        data = Encoding.ASCII.GetBytes(welcome)
        server.SendTo(data, data.Length, SocketFlags.None, ipep)
        Dim sender As IPEndPoint = New IPEndPoint(IPAddress.Any, 0)
        Dim Remote As EndPoint = CType(sender, EndPoint)
        data = New Byte(1024) {}
        Dim recv As Integer = server.ReceiveFrom(data, 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" Then
```

```
            ' break
        End If
        server.SendTo(Encoding.ASCII.GetBytes(input), Remote)
        data = New Byte(1024) {}
        recv = server.ReceiveFrom(data, Remote)
        stringData = Encoding.ASCII.GetString(data, 0, recv)
        Console.WriteLine(stringData)
    End While
    Console.WriteLine("Stopping client")
    server.Close()
  End Sub
End Class
```

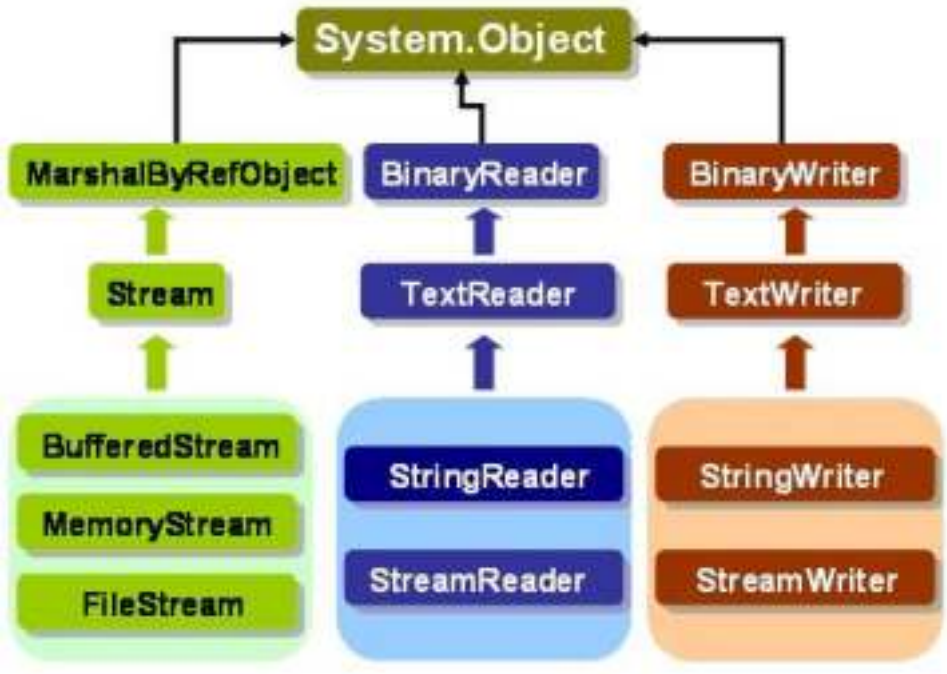
بينا في هذا الجزء كيفية التعامل مع ال UDP Connectionless Protocol وبينا الفرق بينه وبين TCP Connection Oriented Protocol ...

Streaming & Threading Overview: 1.3

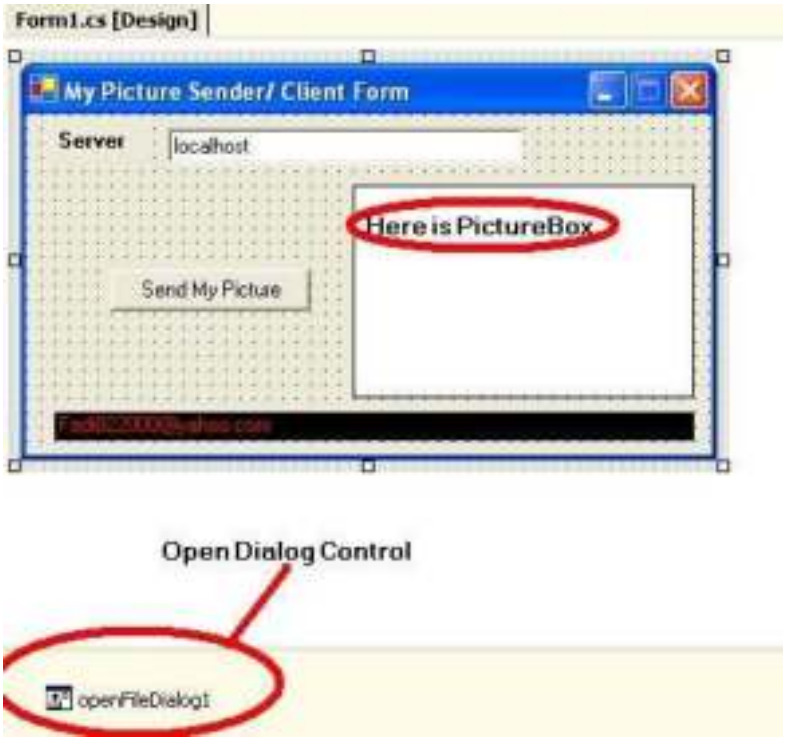
تعرفنا سابقا على أجزاء OSI و TCP/IP وبينا كيفية التعامل مع هذه الطبقات في برمجيات الشبكات ، وفي هذا الجزء سوف نبين كيفية التعامل مع ال Stream Library لإرسال Binary Data بالإضافة إلى استخدام ال Thread في برمجيات الشبكة...

أولا : ال Socket : قلنا سابقا أن الSocket هي الأداة التي يتم نقل البيانات من خلالها من جهاز إلى أخر ولاستخدامها يلزم في البداية تعريف System.Net.Sockets حيث يحتوي هذا ال Namespaces على عدد ضخم من ال Classes والتي يتم استخدامها في برمجيات الشبكة انظر الفصل الثالث.

يمكنك الStream Classes من نقل Text أو Object ،حيث بينا سابقا كيفية التعامل من الSocket لنقل Text باستخدام Stream Reader وال Stream Writer وفي هذا الجزء سنبين كيفية التعامل معه لنقل Object (أي نوع أخر من البيانات ويمكن أن يكون صورة Image أو صوت Voice أو أي شيء آخر يمكن أن يحول إلى Binary Data ..)، وكما هو الحال في نقل ال Text كنا نحول Text إلىASCII Code ثم إلى Binary أما في Object فيتم التعامل معه باستخدام ال Stream Classes والتي يتم الوصول إليها من Name System.IO Spaces وتحتوي هذه Classes على الBinary Reader و ال Binary Writer والتي تمكنك من التعامل مع أي Object وال Stream Reader وال Stream Writer والتي تمكنك من التعامل مع ال Text وال File Stream لتسهيل التعامل مع الملفات بالإضافة إلى ال Memory Stream والتي تستخدم ك Buffer لحفظ البيانات قبل إرسالها أو بعد استقبالها انظر الفصل الثاني.



حيث تساعدك هذه المكتبة على تحويل أي Object إلى Binary باستخدام ال Binary Writer لتسهل إرساله عبر الشبكة باستخدام Network Stream ثم تحويله مرة أخرى إلى Object باستخدام Binary Reader ، وكمثال تطبيقي على هذا سوف نقوم ببناء برنامج يقوم بعملية نقل Image من جهاز إلى أخر Client/Server وللبدء قم بإنشاء مشروع جديد كما في الشكل التالي :



في البداية قم بإضافة ال Namespaces التالية:

C#:

```
using System.Net.Sockets;
using System.IO;
```

VB.NET:

```
Imports System.Net.Sockets
Imports System.IO
```

للإجراء عملية الإرسال لا بد أولا من اشتقاق Instance من الكلاس MemoryStream والتي سوف نستخدمها لتخزين الصورة داخل الذاكرة بشكل مؤقت لكي نحولها لاحقا إلى مصفوفة Binary ثم إرسالها باستخدام NetworkStream عبر ال Socket إلى جهاز الServer:

C#:

```
try
{
تحديد الباث الخاص بصورة
openFileDialog1.ShowDialog ();
string mypic_path =openFileDialog1.FileName ;
pictureBox1.Image = Image.FromFile(mypic_path);

MemoryStream ms = new MemoryStream();
pictureBox1.Image.Save(ms,pictureBox1.Image.RawFormat);
تخزين الصورة ووضعها في مصفوفة من النوع بايت
byte[] arrImage = ms.GetBuffer();
ms.Close();
الاتصال بجهاز الServer عبر العنوان والPort المحدد
TcpClient myclient = new TcpClient (txt_host.Text,5020);//Connecting with
server
```

VB.NET:

```
openFileDialog1.ShowDialog
Dim mypic_path As String = openFileDialog1.FileName
pictureBox1.Image = Image.FromFile(mypic_path)
Dim ms As MemoryStream = New MemoryStream
pictureBox1.Image.Save(ms, pictureBox1.Image.RawFormat)
Dim arrImage As Byte() = ms.GetBuffer
ms.Close
Dim myclient As TcpClient = New TcpClient(txt_host.Text, 5020)
```

C#:

```
                                        إرسال الصورة المخزنة إلى جهاز الServer
NetworkStream myns = myclient.GetStream ();
BinaryWriter  mysw = new BinaryWriter (myns);
mysw.Write(arrImage);//send the stream to above address

                                        إغلاق الSocket والجلسة و ال Stream
mysw.Close ();
myns.Close ();
myclient.Close ();

}
catch (Exception ex){MessageBox.Show(ex.Message );}
```

VB.NET:

```
Try
Dim myns As NetworkStream = myclient.GetStream
Dim mysw As BinaryWriter = New BinaryWriter(myns)
   mysw.Write(arrImage)
   mysw.Close
   myns.Close
   myclient.Close
 Catch ex As Exception
Msgbox(ex.Message)
 End Try
```

ثانيا : ال Server

سوف ابدأ في هذا الجزء شرح الجزء الخاص بالServer والذي يقوم بعملية التصنت على الPort واستقبال ال Stream عبر ال Socket و قراءتها باستخدام ال Binary Reader وتحويله إلى اوبجكت (صيغته التي كان عليها قبل الإرسال) مرة أخرى ، في هذا المثال نريد استقبال صورة وفي هذه الحالة وفرت لدينا الدوت نيت خصائص جديدة في ال Controls الموجودة فيها ومن ضمنها خاصية Image.FromStream الخاصة ب ال PictureBox والتي تسهل علينا إمكانية عرض الصورة المرسلة من خلال Stream لكي يتم تحويلها من Binary Stream إلى صورة تعرض على ال Picturebox انظر المثال التالي :

C#:

```
using System.Net.Sockets ;
using System.IO;

// Objects  Declaration
TcpListener mytcpl; // Declare TCP Listener
Socket mysocket; // Declare an object from Socket Class
NetworkStream myns; //
StreamReader mysr;
void Image_Receiver()
{

mytcpl = new TcpListener (5000);// Open The Port
mytcpl.Start ();// Start Listening on That Port
mysocket = mytcpl.AcceptSocket ();// Accept Any Request From Client and
Start The Session

myns = new NetworkStream (mysocket);// Receive The Binary Data From
Port
pictureBox1.Image = Image.FromStream(myns); // Show The Image that
Resaved as Binary Stream
mytcpl.Stop();// Close TCP Session

    if (mysocket.Connected ==true)//if Connected Start Again
{

while (true)
{
Image_Receiver();// Back to First Method
}
}
}
```

VB.NET:

```
Private mytcpl As TcpListener
Private mysocket As Socket
Private pictureBox1 As System.Windows.Forms.PictureBox
Private mainMenu1 As System.Windows.Forms.MainMenu
Private menuItem1 As System.Windows.Forms.MenuItem
Private saveFileDialog1 As System.Windows.Forms.SaveFileDialog
Private myns As NetworkStream

Sub Image_Receiver()
    mytcpl = New TcpListener(5000)
    mytcpl.Start()
    mysocket = mytcpl.AcceptSocket
    myns = New NetworkStream(mysocket)
    pictureBox1.Image = Image.FromStream(myns)
    mytcpl.Stop()
    If mysocket.Connected = True Then
        While True
```

```
            Image_Receiver()
        End While
    End If
End Sub
```

ولتطبيق قم بإنشاء مشروع جديد كما في الشكل التالي :

أضف ال method السابقة في class البرنامج ثم قم باستدعائها بوضع ال
Image_Receiver()اما في ال Constructer الخاص بالبرنامج أو بحدث بدأ التشغيل
الخاص بال Form ، و الميثود التالية في حدث ال Closing الخاص بال Form :

C#:

```
private void Form1_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
{
try
{
mytcpl.Stop ();
Application.Exit();
}
catch (Exception ex) {MessageBox .Show (ex.Message );}
}
```

VB.NET:

```
Private Sub Form1_Closing(ByVal sender As Object, ByVal e As
System.ComponentModel.CancelEventArgs)
    Try
        mytcpl.Stop()
        Application.ExitThread()
        Application.Exit()
    Catch ex As Exception
Msgbox(ex.Message)
    End Try
End Sub
```

وذلك لتأكد من إغلاق الSocket عند إنهاء البرنامج ،..

قم بإضافة الكود التالي إلى ال Save Button لكي تتمكن من تخزين الصورة المستقبلة:

C#:

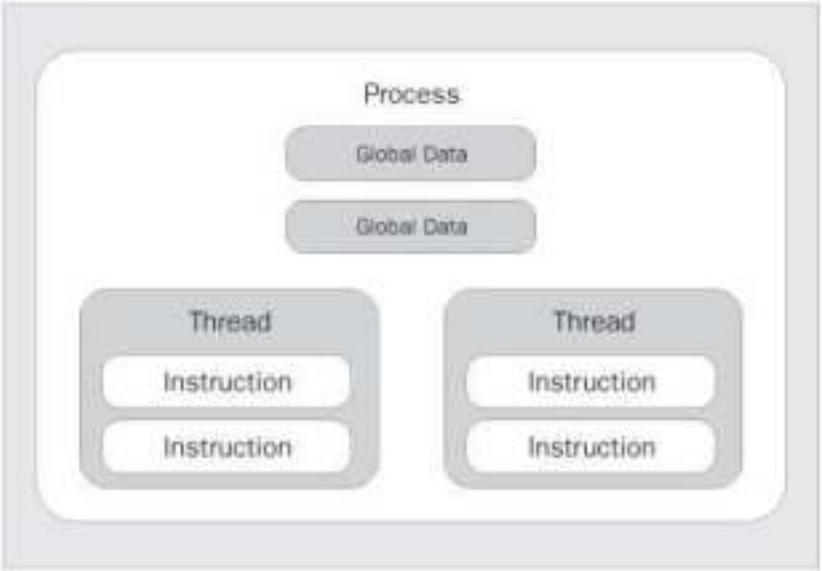
```
try
{

saveFileDialog1.Filter = "JPEG Image (*.jpg)|*.jpg" ;
if(saveFileDialog1.ShowDialog() == DialogResult.OK)
        {
string mypic_path = saveFileDialog1.FileName;
pictureBox1.Image.Save(mypic_path);
        }
}
catch (Exception){}
```

VB.NET:

```
Try
 saveFileDialog1.Filter = "JPEG Image (*.jpg)|*.jpg"
 If saveFileDialog1.ShowDialog = DialogResult.OK Then
Dim mypic_path As String = saveFileDialog1.FileName
   pictureBox1.Image.Save(mypic_path)
 End If
Catch generatedExceptionVariable0 As Exception
End Try
```

: Threading Overview

سوف يؤدي ال Infinity Loop والذي وضعناه إلى تعليق البرنامج والسبب أن ال Loop يعمل على منطقة ال Global Area والمخصصة لل Form إذ لن ينفذ شيء إلا بعد انتهاء ال Loop وهو ما لن يحدث أبدا إذ انه Infinity Loop ، قدمت لنا الدوت نيت الحل لهذه المشكلة وهي باستخدام تكنولوجيا ال Threading والتي تسمح بالمعالجة المتوازية على نفس المعالج وذلك من خلال تقسيم المهام على المعالج وعمل Session منفصلة لكل برنامج وهو ما يسمى بال Multitasking.. وهنا لا يؤثر البرنامج على موارد النظام بشكل كبير كما أن ال Loop ستعمل في Thread منفصل عن ال Thread الخاص بال Form انظر الشكل التالي :



لاحظ انه قبل إضافة ال Thread كان Loop يعمل على منطقة ال Global Area وهذا هو سبب البطء الشديد وبعد استخدام ال Thread تم عمل Session خاص لل Loop بحيث يعمل بشكل متوازي مع البرنامج ..

ولاستخدام ال Thread يلزم أولا تعريف ال System.Threading Namespace :

C#:

```
using System.Threading;
```

VB.NET:

```
imports System.Threading
```

ثم اشتقاق Instance منه وإدراج اسم الميثود التي تريد عمل Thread لها في ال Delegate الخاص بها كما يلي :

C#:

```
Thread myth;
myth= new Thread  (new System.Threading .ThreadStart(Image_Receiver));
myth.Start ();
```

VB.NET:

```
Imports System.Threading

Dim myth As Thread
myth = New Thread(New System.Threading.ThreadStart(Image_Receiver))
myth.Start
```

ألان قم بإضافة Application.ExitThread في حدث ال Closing Form كما يلي

C#:

```
private void Form1_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
{
try
{
mytcpl.Stop ();
Application.ExitThread ();
Application.Exit();
}
catch (Exception ex) {MessageBox .Show (ex.Message );}

}
```

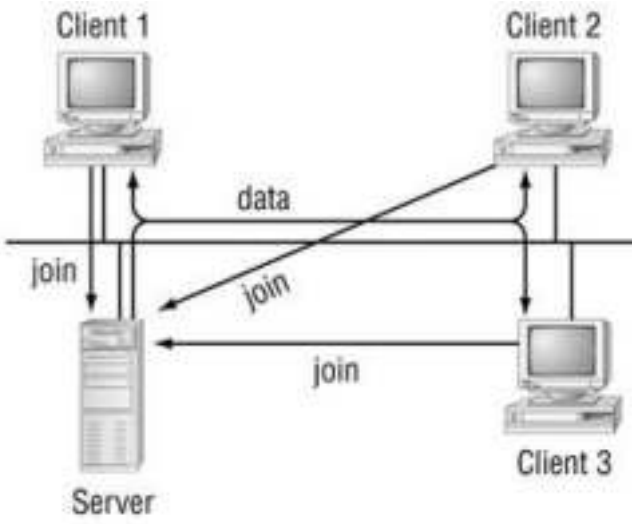
VB.NET:

```
Private Sub Form1_Closing(ByVal sender As Object, ByVal e As
System.ComponentModel.CancelEventArgs)
    Try
        mytcpl.Stop()
        Application.ExitThread()
        Application.Exit()
    Catch ex As Exception
        Msgbox(ex.Message)
    End Try
End Sub
```

ميزة ال Thread رائعة جدا إذ تمكنك من تشغيل أكثر من Thread وفي نفس الوقت وفي نفس البرنامج وهو ما يسمى بال Multithreading والذي سآتي على شرحه بتفصيل في النسخة الورقية من الكتاب.

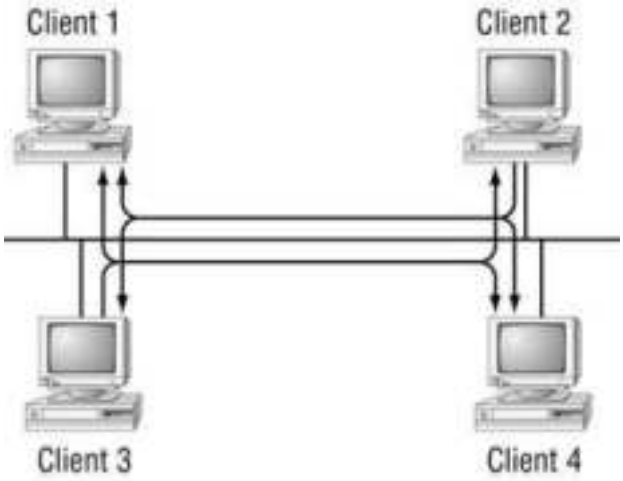
1.4: IP Multicasting واستخدامها لعمل Multicasting Group

تحدثنا سابقا عن برتوكول ال UDP وشرحنا كيفية استخدامه لعمل برود كاست حيث تستطيع عمل البرود كاست بطريقتين إما باستخدام IPAddress.Any والذي يلزمه وجود Server يقوم بعملية التصنت على الPort المحدد حيث يستقبل من خلاله أي رسالة ثم يقوم ببثها إلى كل الأجهزة أو باستخدام IPAddress.Broadcast والذي من خلاله يمكن عمل بث إلى كل الأجهزة حيث لا ضرورة لوجود جهاز Server بحيث أن الكل يمكنه التصنت على ال Port المحدد و يستقبل ويرسل من خلالها أي رسالة إلى كل الأجهزة وتشبه عملية ال Broadcast عملية البث الإذاعي حيث أن الجميع يستمع من الكل ويرسل إلى الكل ، أما إذا أردنا تقسيم الإرسال إلى مجموعات عندها يجب استخدام ال IP Multicasting وذلك بهدف استخدامه لعمل ال Multicast Group ، يعتبر هذا الموضوع من المواضيع المهمة جدا في برمجيات الشبكات ولهذا خصصت له فصل منفصل عن البقية (انظر الفصل الرابع) إذ أن اغلب برمجيات ال Conferences تعتمد عليه بشكل كبير ويعرف Multicast على انه الإرسال إلى مجموعة من المستخدمين سواء كان Managed باستخدام Client/Server حيث يكون هنالك جهاز Server في الشبكة وظيفته استقبال الرسائل من ال Clients Group ثم إرسالها إلى كامل المجموعة مرة أخرى انظر إلى الشكل التالي :



لاحظ انه يتم إرسال طلب الانضمام إلى المجموعة من قبل ال Clients وإذا وافق الServer على الطلب يقوم بضم عنوان الجهاز إلى ال IP Address List الخاصة به وتشترك كل مجموعة بنفس ال IP Multicast ويتم الإرسال إلى جميع أعضاء المجموعة التي تشترك بنفي ال IP Multicast والذي يقع ضمن ال Class D وهو مابين سابقا.

النوع الثاني ويسمى بال unmanaged - peer-to-peer Technique حيث أن كل جهاز يعمل ك server و client في نفس الوقت ولا وجود لجهاز Server مركزي مخصص لعملية الاستقبال والتوزيع حيث تتم الموافقة على طلب الانضمام إلى المجموعة بشكل تلقائي وأي جهاز في المجموعة له الحق في الانضمام ثم الاستقبال و الإرسال إلى كامل المجموعة لاحظ الشكل التالي :



تم تخصيص عناوين خاصة للـ Multicasting وهو ما يسمى بالـ IP Multicast Address وهي كما يلي :

المدى من 224.0.0.0 إلى 224.0.0.255 لشبكات المحلية LAN
المدى من 224.0.1.0 إلى 224.0.1.255 للـ Internetwork
المدى من 224.0.2.0 إلى 224.0.255.255 للـ AD-HOC Network block

قدمت الدوت نيت دعم كبير للـ IP Multicast باستخدام الـ Socket Namespace حيث يتم تعريفها باستخدام الـ الميثود SetSocketOption والتي تقوم بإدارة عمليات الانضمام والخروج من والى المجموعة multicast group (join & leave) كما تستخدم لإضافة وإلغاء العضوية AddMembership و DropMembership و يستخدم الـ UdpClient Object لتحديد رقم الـPort والذي سيتم استقبال البيانات من خلاله بالإضافة إلى تعريف الـ IP Multicasting والذي من خلاله تحدد الجهات التي سوف تستقبل الرسالة من خلال تحديد الـ Range IP الخاص بشبكات المحلية LAN حيث يستطيع أي شخص يتنصت على هذا الـPort ويستخدم نفس الـ IP Multicast استقبال هذه الرسالة ، يستخدم الكود التالي لإرسال رسالة إلى عدة جهات بحيث نستخدم رقم الـPort 5020 و ضمن الـ Group 224.100.0.1 كمثال:

C#:

```
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"),
5020);
      byte[] data = Encoding.ASCII.GetBytes("This is a test message");
      server.SendTo(data, iep);
      server.Close();
   }
}
```

VB.NET:

```
Imports System
Imports System.Net
Imports System.Net.Sockets
Imports System.Text

Class MultiSend

    Public Shared Sub Main()
Dim server As Socket = New Socket(AddressFamily.InterNetwork,
SocketType.Dgram, ProtocolType.Udp)
Dim iep As IPEndPoint = New IPEndPoint(IPAddress.Pars("224.100.0.1"),
5020)
Dim data As Byte() = Encoding.ASCII.GetBytes("This is a test message")
server.SendTo(data, iep)
server.Close()
    End Sub
End Class
```

في البداية قمنا بتعريف الSocket بتحديد الجهة التي سوف تستقبل الرسالة وهي (أي شخص يتنصت على الشبكة باستخدام ال IP Multicast Group المحدد) ثم تحديد نوع الSocket والبرتوكول المستخدم ، وبعد ذلك تحديد نقطة الهدف وذلك بوضع ال IP Multicast الذي نريد ويتبعه رقم الPort التي سيتم استقبال البيانات من خلاله:

ولإنشاء برنامج الاستقبال سوف نستخدم تعريف ال Socket نفسه ونضيف ال UdpClient Object ونسند له رقم الPort التي نريد التصنت عليه:

C#:

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

class UdpClientMultiRecv
{
    public static void Main()
  {
    UdpClient sock = new UdpClient(5020); // هذاPortالتصنت على رقم ال
    sock.JoinMulticastGroup(IPAddress.Parse("224.100.0.1"), 50);
     وهذا يعني انك سوف تتصنت على المدى المحدد

    IPEndPoint iep = new IPEndPoint(IPAddress.Any, 0);

 //  استقبال البيانات وتعبئة الرسالة في مصفوفة من النوع بايت
    byte[] data = sock.Receive(ref iep);

// التحويل إلى أسكي كود ثم طباعة الرسالة على الشاشة
    string stringData = Encoding.ASCII.GetString(data, 0, data.Length);
    Console.WriteLine("received: {0} from: {1}", stringData, iep.ToString());
    sock.Close();}}
```

VB.NET:

```
Imports System
Imports System.Net
Imports System.Net.Sockets
Imports System.Text

Class UdpClientMultiRecv

    Public Shared Sub Main()
        Dim sock As UdpClient = New UdpClient(5020)
        sock.JoinMulticastGroup(IPAddress.Parse("224.100.0.1"), 50)
        Dim iep As IPEndPoint = New IPEndPoint(IPAddress.Any, 0)
        Dim data As Byte() = sock.Receive(iep)
        Dim stringData As String = Encoding.ASCII.GetString(data, 0,
data.Length)
        Console.WriteLine("received: {0} from: {1}", stringData, iep.ToString)
        sock.Close()
    End Sub
End Class
```

لاحظ انه توجد طرق متعددة لاستقبال البيانات و إرسالها كما يمكن استخدام الكوديين السابقين في نفس البرنامج للإرسال و الاستقبال كما يمكنك إرسال Image إلى جانب النص (انظر الفصل الرابع) أو أي شيء أخر يمكن تحويله إلى Binary إذ ما عليك سوى إضافة ال **memory Stream** وال **Binary Reader** وال **Binary Writer** إلى كود الإرسال و الاستقبال كما يمكنك عمل برنامج لإرسال صورة عبر الكاميرا إلى جهات متعددة باستخدام نفس الخاصية والتي سآتي على شرحها في الفصل الرابع ...

Chapter 2
Streaming in Dot Net

2- Managed I/O: Streams, Readers, and Writers

A. Stream Classes
B. Stream Members
C. Stream Manipulation
D. Simple Remote Control Application Using StreamReader & StreamWriter Classes

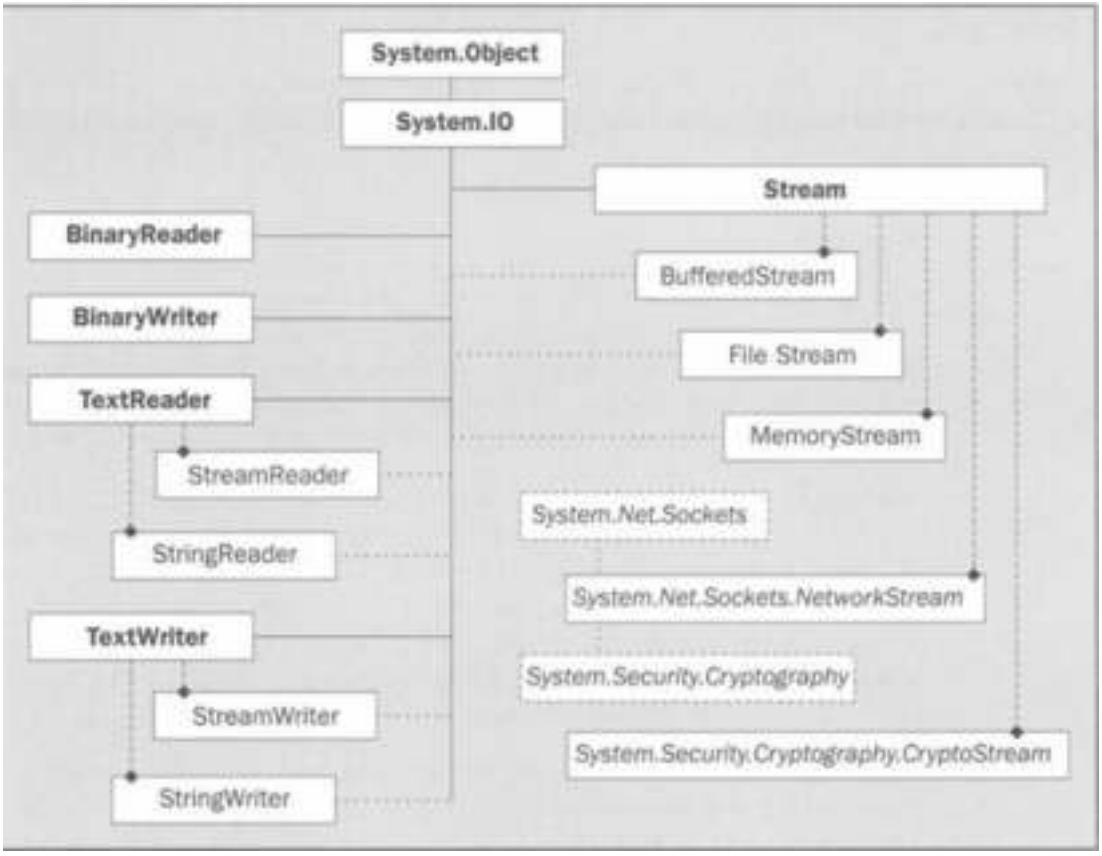
بسم الله الرحمن الرحيم

: Managed I/O: Streams, Readers, and Writers : 2.1

تحدثنا سابقا في الجزء الأول بشكل عام عن استخدامات ال Streams Library واستخدامها لإرسال Binary Data و Text Data من جهاز إلى آخر وكمثال قمنا بإرسال صورة من ال Client إلى ال Server باستخدام ال Binary Reader & Binary Writer ..
إن الهدف من إنشاء مكتبات ال Stream هو تسهيل عملية نقل البيانات من مكان إلى أخر سواء عبر الشبكة أو داخل نفس الجهاز كما هو الحال بتعامل مع الملفات أو التعامل مع الطابعة أو أي طرفية أو جهاز آخر موصول بالكمبيوتر حيث تسهل علينا عملية تحويلها إلى Byte Array وإرسالها وهو ما حل الكثير من المشاكل التي كانت تواجه المبرمجين في التعامل مع Binary Data ..
يمكن التعامل مع ال Stream بأسلوبين المتزامن **Synchronous** والغير متزامن **Asynchronous** وبشكل افتراضي تعمل جميع ال IO Streams بالأسلوب المتزامن لاكن العيب فيه هو تأثيره الشديد على أدائية النظام إذ يقوم بإغلاق ال Processing Unit في ال Thread المخصصة للبرنامج بحيث لا يسمح بتنفيذ أي أمر آخر إلا بعد الانتهاء من العملية الجارية ولا ينصح ابدآ استخدام الأسلوب المتزامن في حالة إذا كنت تتعامل مع أجهزة قراءة وكتابة بطيئة نسبيا مثل ال Floppy Disk أو ال Magnetic Tape لكنها مهمة جدا بالبرمجيات التي تعتمد على أنظمة الزمن الحقيقي أو ال Real Time Systems حيث أنها تعتمد الأسلوب المتزامن في عملية إرسال واستقبال البيانات وهو ما يمنع القيام بأي عملية أخرى إلى حين الانتهاء من تنفيذ الأمر ومن الأمثلة عليها أنظمة السحب أو الإيداع في الرصيد البنكي أو أنظمة حجز التذاكر أو شحن بطاقة الهاتف وغيرها .. طبعا في حالة إذا كان برنامجك لا يحتاج إلى وجود الخواص السابقة عندها ينصح باستخدام الأسلوب الغير متزامن Asynchronous حيث تستطيع من خلاله تنفيذ عمليات أخرى في وحدة المعالجة وبدون الحاجة لانتظار إنهاء العملية الجارية إذ يتم إنشاء Separate thread لكل عملية طلب إدخال أو إخراج مما لا يؤثر على أدائية النظام وينصح باستخدامه إذا كانت عملية القراءة أو الكتابة تجرى من خلال أجهزة بطيئة نسبيا ويمكن تميز الميثود المتزامن عن الغير متزامن في الدوت نيت بوجود كلمة Begin أو End في بداية اسم الميثود الغير متزامن وكمثال عليها BeginWrite و BeginRead و ال EndWrite و ال EndRead ..

أولا: Stream Classes

تدعم الدوت نيت عمليات ال Streams بمجموعة من ال Classes والمندرجة تحت System.IO Name Space والتي تستخدم لعمليات الإدخال و الإخراج لنقل البيانات .
تستخدم بعض ال Stream Classes ، Backing storage ، ومن الأمثلة عليها FileStream و ال BufferedStream وال MemoryStream وكذلك فإن بعضها لا يستخدم أي Back Storage ومن الأمثلة عليها ال NetworkStream والتي تستخدم لنقل ال Stream عبر الشبكة وبدون استخدام Backing Storage ، و تقسم ال Stream Classes في الدوت نيت كما في الشكل التالي :



1- BufferedStream Class : ويستخدم بشكل أساسي لحجز مقدار معين من الذاكرة بشكل مؤقت لتنفيذ عملية معينة كما تستخدم بعض البرمجيات ال Buffering لتحسين الأدائية حيث تكون كذاكرة وسيطة بين المعالجة و الإرسال أو الاستقبال وكمثال عليها برمجيات الطباعة حيث تستخدم الطابعة ذاكرة وسيطة لتخزين البيانات المراد طباعتها بشكل مؤقت ، يكمن الهدف الأساسي من استخدام ال Buffering في العمليات التي يكون فيها المعالج أسرع من عمليات الإدخال و الإخراج حيث يتم معالجة البيانات ووضعها في ال Buffer في انتظار إرسالها وهو ما يساهم في تحسين الأدائية بشكل كبير ، ويستخدم ال BufferedStream عادتا في برمجيات الشبكات مع ال NetworkStream لتخزين البيانات المراد إرسالها عبر الشبكة في الذاكرة حيث لا يستخدم هذا الكلاس Backing storage كما ذكرنا سابقا ..

بشكل افتراضي يتم حجز 4096 bytes عند استخدام ال BufferedStream ويمكن زيادتها أو تقليلها حسب الحاجة .. ويستخدم ال BufferedStream كما يلي كمثال :

C#

```
using System;
using System.Text;
using System.IO;
namespace Network_Buffering
{
   class Program
   {
      static void Main(string[] args)
      {
         ASCIIEncoding asen = new ASCIIEncoding();
         byte[] xx = asen.GetBytes("Hello Buffering");
```

```
            MemoryStream ms = new MemoryStream(xx);
            readBufStream(ms);
        }
  public static void readBufStream(Stream st)
     {
      // Compose BufferedStream
BufferedStream bf = new BufferedStream(st);
byte[] inData = new Byte[st.Length];

// Read and display buffered data
bf.Read(inData, 0, Convert.ToInt32(st.Length));
Console.WriteLine(Encoding.ASCII.GetString(inData));
        }
    }
}
```

VB.NET:

```
Imports System
Imports System.Text
Imports System.IO
Namespace Network_Buffering

    Class Program

        Shared Sub Main(ByVal args As String())
            Dim asen As ASCIIEncoding = New ASCIIEncoding
            Dim xx As Byte() = asen.GetBytes("Hello Buffering")
            Dim ms As MemoryStream = New MemoryStream(xx)
            readBufStream(ms)
        End Sub

        Public Shared Sub readBufStream(ByVal st As Stream)
            Dim bf As BufferedStream = New BufferedStream(st)
            Dim inData(st.Length) As Byte
            bf.Read(inData, 0, Convert.ToInt32(st.Length))
            Console.WriteLine(Encoding.ASCII.GetString(inData))
        End Sub
    End Class
End Namespace
```

حيث قمنا بتحويل نص إلى Byte Array باستخدام ال ASCIIEncoding وتحميله في MemoryStream عبر ال readBufStream ثم ارسلناه إلى الميثود والتي انشأناها حيث استقبلنا من خلالها ال Stream وحملناه في ذاكرة مؤقتة بستخدام الكلاس ال BufferedStream ثم قمنا بطباعة محتوياته بعد تحويله إلى نص مرة اخرى بستخدام ال Encoding.ASCII وطباعته ..

2- MemoryStream Class : وهو شبيه بعملية ال Buffring السابقة إذ يعتبر كحل جيد لتخزين البيانات بشكل مؤقت في الذاكرة قبل الإرسال أو الأستقبال حيث يغنيك عن تخزينها على شكل ملف مما يسرع العملية بشكل كبير ويستخدم كما يلي كمثال حيث استخدمنها لتخزين صورة في الذاكرة :

C#

```
MemoryStream ms = new MemoryStream();
pictureBox1.Image.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg);

byte[] arrImage = ms.GetBuffer();
ms.Close();
```

VB.NET:

```
Dim ms As MemoryStream = New MemoryStream
pictureBox1.Image.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg)
Dim arrImage As Byte() = ms.GetBuffer
ms.Close
```

3- NetworkStream Class : وكما قمنا باستخدامها سابقا ، حيث تقوم بتعامل مع ال Stream لإرساله عبر الشبكة باستخدام ال Socket ويتم استدعائها من Name Spaces System.Net.Sockets ويعتبر الكلاس NetworkStreamبأنه unbuffered إذ لا يحتوي على Backing Storage ويفضل استخدام ال BufferedStream Class معه لتحسين الأداء وتستخدم كما يلي كمثال حيث نريد إرسال الصورة التي قمنا بتخزينها في المثال السابق بذاكرة إلى جهاز أخر عبر ال Socket :

C#

```
TcpClient myclient = new TcpClient ("localhost",5020);//Connecting with
server

NetworkStream myns = myclient.GetStream ();

BinaryWriter  mysw = new BinaryWriter (myns);
mysw.Write(arrImage);//send the stream to above address
mysw.Close ();
myns.Close ();
myclient.Close ();
```

VB.NET:

```
Dim myclient As TcpClient = New TcpClient(localhost, 5020)
Dim myns As NetworkStream = myclient.GetStream
Dim mysw As BinaryWriter = New BinaryWriter(myns)
mysw.Write(arrImage)
mysw.Close
myns.Close
myclient.Close
```

4- FileStream : يتم استدعائها باستخدام System.IO Name Spaces وتستخدم بشكل اساسي في التعامل مع الملفات سواء للكتابة إلى ملف أو القراءة من ملف وتعتبر هذه الكلاس Backing Storage Class حيث تستخدم ذاكرة Buffer لتحزين البيانات بشكل مؤقت في الذاكرة لحين الإنتهاء من عملية الكتابة أو القراءة ومن الأمور الهامة فيها تحديد مسار الملف المراد القراءة منه أو الكتابة عليه وتستخدم كما يلي :

C#

```
FileStream FS = new FileStream(@"C:\MyStream.txt",
FileMode.CreateNew);// Any Action For Example CreateNew to Create Folder
```

VB.NET:

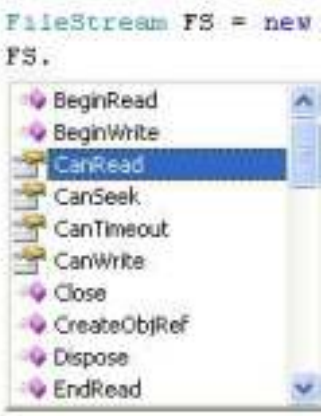
```
Dim FS As FileStream = New FileStream("C:\MyStream.txt",
FileMode.CreateNew)
```

يمكننا استخدام ال Enumeration التالية مع ال FileMode :

1- Append لإضافة نص ما إلى الملف الموجود اصلا
2- Create لإنشاء ملف جديد ويقوم بعمل overwriting في حالى إذا كان الملف موجود بشكل مسبق
3- CreateNew وهو كما في ال Create إلا انه يعطي Exception في حالة وجود الملف بشكل مسبق
4- Open لقراءة ملف ما حيث يعطي Excption في حالة عدم وجود الملف المحدد
5- OpenOrCreate في حالة إذا وجد الملف يقوم بقراءته وفي حالة عدم وجوده يقوم بإنشائه.
6- Truncate ويستخدم لحذف محتويات الملف وجعله فارغا

ثانيا : Stream Members :

هنالك مجموعة من الخواصو و الميثودس التي تشترك بها مكتبات ال Stream وهي كما يلي :



1- CanRead و CanWrite وتستخدم لمعرفة إذا كان ال Stream المستخدم يقبل عملية القراءة أو الكتابة أم لا حيث ترجع قيمة True في حالة إذا كان يقبل و False في حالة أنه لا يقبل ويستخدم عادة قبل إجراء عملية القراءة أو الكتابة لفحص مدى الصلاحية قبل المحاولة ..
2- CanSeek حيث يستخدم ال Seeking عادة لتحديد موقع ال Current Stream والعادة تدعم الكلاسات التي تستخدم Backing Storage هذه العملية مثل ال

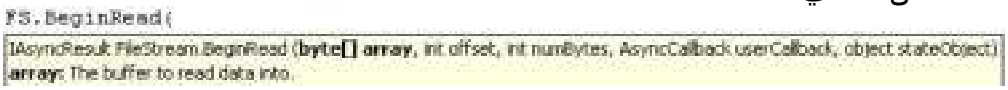
FileStream وعندها ترجع قيمة True وترجع قيمة false في حالة إذا كان ال Stream Class لا يحتوي على Backing Storage .
3- CanTimeout وترجع قيمة True في حالة إذا كان ال stream يحتوي على خاصية ال Timeout والتي تعطي وقت محدد للعملية .
4- Length وتستخدم لمعرفة حجم ال Stream بال Byte ويمكن الاستفادة منها لمعرفة نهاية ال Stream أو لتحديد حجم المصفوفة بناء على حجم ال Stream .
5- Position وتستخدم ال Get و Set لمعرفة أو تحديد الموقع ل Stream
وتشترك مكتبات ال Stream بمجموعة من الميثودس وهي كما يلي :

1- الميثودس المتزامنة Synchronous Methods :

I. Read و ReadByte وتستخدم لقراءة Stream Data وتخزينه في ال Buffer ويمكن تحديد عدد البايتات التي سيتم قراءتها باستخدام ال ReadByte كما نستطيع من خلالها معرفة نهاية ال Stream حيث ترجع ال Read قيمة 0 وال ReadByte قيمة -1 في حالة انتهاء ال Stream.
II. Write وال WriteByte وتستخدم لعملية الإرسال عبر ال Stream ويمكن تحديد عدد البايتات التي سيتم كتابتها في كل مرة باستخدام ال WriteByte.

2- الميثودس غير المتزامنة Asynchronous Methods :

I. BeginRead وال BeginWrite وتستخدم لعملية القراءة أو الكتابة باستخدام ال Stream الغير المتزامن وتأخذ خمسة بارومترات كما في الشكل التالي :



1- ال Byte Buffer والتي سوف تستخدم لعملية القراءة منه أو الكتابة عليه
2- ال offset والذي سوف يحدد فيه موقع القراءة أو الكتابة
3- ال numByte والذي سوف يتم فيه تحديد الحد الأقصى من البايتات التي سيتم كتابتها أو قراءتها
4- ال AsyncCallback وهو Optional Delegate حيث يتم استدعائه عند الانتهاء من عملية القراءة أو الكتابة
5- ال Stateobject وهي User Provided Object وتستخدم لتميز ال Read & Write Request عن غيره ال Requests .
ترجع ال Begin Methods ال IAsyncResult والذي يمثل حالة ال Stream Operation .

II. EndRead وال EndWrite وتستخدم في حالة إذا أردنا تنفيذ ال Stream Operation بعد الانتهاء من ال Stream Operation الحالي، حيث يبقى بانتظار انتهاء العملية السابقة ثم ينفذ العملية المطلوبة

وهناك بعض الميثود والتي تستخدم لإدارة ال Stream وهي :

1- Flush وتستخدم لتفريغ محتويات ال Buffer بعد إتمام العملية المحددة حيث يتم نقل محتويات ال Buffer إلى ال Destination الذي تم تحديده في Stream Object.

2- Close وتستخدم لإغلاق ال Stream وتحرير ال Resources المحجوزة من قبل ال Stream Object وينصح باستخدامها في الجزء الخاص ب Finally block ولتأكد من أن ال Stream سيتم إغلاقه وتحرير كافة الموارد في حالة حدوث أي Exception إثناء التنفيذ ولضمان عدم بقاء هذه الموارد في الذاكرة بعد إغلاق البرنامج.

3- SetLength وتستخدم لتحديد حجم ال Stream والذي نريد إرساله أو استقباله لاكن في حالة إذا كان ال Stream أقل من المحدد في ال SetLength سوف يؤدي ذالك إلى انقطاع ال Stream وعدم وصوله بشكل سليم ، لن تستطيع استخدام هذه الخاصية إلا إذا تأكدت انك تملك الصلاحية لذلك من خلال الخاصية CanWrite و CanSeek لذا ينصح بفحص الصلاحية أولا قبل تحديد حجم ال Stream .

ثالثا : Stream Manipulation:

يمكن استخدام مكتبات ال Stream لنقل Binary Data أو Text وفي العادة يتم استخدام ال BinaryReader و ال BinaryWriter لتعامل مع ال Binary Data ويتم استخدام ال StreamReader وال StreamWriter لتعامل مع ال Text ،
ويتم استخدام ال ASCIIEncoding أو UnicodeEncoding لتحويل من Stream إلى Text عند الاستقبال ومن Text إلى Stream عند الإرسال حيث تستخدم مجموعة من الميثودس وهي كما في الشكل التالي :

1- GetByteCount وهي Overloaded Method حيث تأخذ Character Array أو String وترجع عدد البايتات التي سوف نحتاجها لنقل نص معين ..
2- GetBytes لتحويل ال String إلى Byte Array حتى نستطيع إرسالها باستخدام ال Stream .
3- GetCharCount حيث تأخذ Byte Array وترجع عدد الأحرف التي سوف تكون في ال String أو في ال Character Array .
4- GetChars وتستخدم لتحويل من Byte Array إلى String وتستخدم عند استقبال البيانات من ال Stream حيث نحولها إلى نص مرة أخرى .

ولتعامل مع ال StreamReader و ال StreamWriter لنقل Text يجب أولا استدعائها من ال System.IO نيم سبيسس وتستخدم كما يلي:

StreamReader للقراءة من ملف:

C#

```
StreamReader str = File.OpenText(openFileDialog1.FileName);
textBox1.Text = str.ReadToEnd();
```

VB.NET:

```
Dim str As StreamReader = File.OpenText(openFileDialog1.FileName)
 textBox1.Text = str.ReadToEnd
```

StreamWriter للكتابة إلى ملف:

C#

```
string fname = saveFileDialog1.FileName;
StreamWriter fsave = new StreamWriter(fname);
fsave.WriteLine(textBox1.Text);
```

VB.NET:

```
Dim fname As String = saveFileDialog1.FileName
Dim fsave As StreamWriter = New StreamWriter(fname)
 fsave.WriteLine(textBox1.Text)
```

و لتعامل مع ال BinaryReader وال BinaryWriter لنقل Binary Data يتم استدعائها من ال System.IO نيم سبيسس وتستخدم كما يلي:

BinaryReader لقراءة Binary Data من ال Stream:

C#

```
NetworkStream myns = new NetworkStream(mysocket);
BinaryReader br = new BinaryReader(myns);
```

BinaryWriterr لإرسال BinaryData إلى ال Stream عبر ال Socket:

```
TcpClient myclient = new TcpClient("localhost", 5020);
NetworkStream myns = myclient.GetStream();
BinaryWriter mysw = new BinaryWriter(myns);
mysw.Write(arrImage);
```

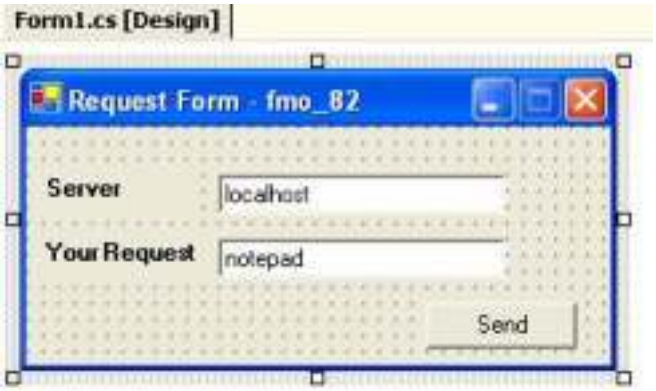
VB.NET:

```
Dim myns As NetworkStream = New NetworkStream(mysocket)
Dim br As BinaryReader = New BinaryReader(myns)
Dim myclient As TcpClient = New TcpClient("localhost", 5020)
Dim myns As NetworkStream = myclient.GetStream
Dim mysw As BinaryWriter = New BinaryWriter(myns)
 mysw.Write(arrImage)
```

Remote Control Example باستخدام ال Stream Reader & Writer:

مثال تطبيقي بسيط سوف نستخدم فيه برنامج شبيه ب Chatting لاكن سوف نستخدمه لإعطاء أوامر إلى ال Server حيث يفترض إذا قمنا بإرسال كلمة notepad إلى ال server بأن يقوم بفتح ال notepad فيه وإذا قمنا مثلا بكتابة Calc وإرسالها إلى الServer سوف يفتح الآلة الحاسبة فيه وهكذا :

أولا : إنشاء برنامج الإرسال Client : لا يختلف برنامج الإرسال عن برنامج ال Client Chat الذي قمنا بإنشائه في ال Chapter1 ويستخدم فيه كل من TCP Connection وال NetworkStream و ال StreamWriter لإجراء عملية الإرسال فباستخدام الميثود WriteLine الموجودة ضمن ال StreamWriter Object تتم عملية تحويل النص المكتوب في ال Textbox إلى مجموعة من ال Bytes ليتم إرسالها باستخدام ال NetworkStream عبر ال TCP Socket Connection إلى برنامج الServer وللبدء قم بإنشاء مشروع جديد كما في الشكل التالي :



ثم قم بإضافة Name Spaces التالية :

C#

```
using System.Net.Sockets ;
using System.IO;
```

في Send Button قم بكتابة الكود التالي:

```
try
        {
TcpClient myclient = new TcpClient (txt_host.Text,5020); // تعريف الSocket
NetworkStream myns = myclient.GetStream (); // إسناده إلى اللستريم اوبجكت
StreamWriter mysw = new StreamWriter (myns);
mysw.WriteLine(txt_msg.Text);

mysw.Close ();
myns.Close ();
myclient.Close ();
        }
catch (Exception ex) {MessageBox.Show (ex.Message );}
```

VB.NET:

```
imports System.Net.Sockets ;
imports System.IO;

Try
Dim myclient As TcpClient = New TcpClient(txt_host.Text, 5020)
Dim myns As NetworkStream = myclient.GetStream
Dim mysw As StreamWriter = New StreamWriter(myns)
 mysw.WriteLine(txt_msg.Text)
 mysw.Close
 myns.Close
 myclient.Close
Catch ex As Exception
Msgbox(ex.Message)
End Try
```

ولإنشاء برنامج ال Server والذي يعمل على استقبال ال Stream وتحويله إلى Text مرة أخرى .. قم بإنشاء مشروع جديد كما في الشكل التالي :

قم بإضافة Name Spaces التالية :

C#

```
using System.Net.Sockets ;
using System.IO;
using System.Threading;
```

ثم إضافة التعاريف التالية :

```
TcpListener mytcpl;// Objects  Declaration
Socket mysocket;
NetworkStream myns;
StreamReader mysr;
```

ثم نقوم بإنشاء ميثود جديدة كما يلي :

```
void our_Server ()
                {
mytcpl = new TcpListener (5020);// Open The Port
mytcpl.Start ();// Start Listening on That Port
mysocket = mytcpl.AcceptSocket ();// Accept Any Request From Client and
Start a Session
myns = new NetworkStream (mysocket);// Receives The Binary Data From
Port
mysr = new StreamReader (myns);// Convert Received Data to String
string order = mysr.ReadLine();

// you can add any order and Response Here
if (order=="notepad") System.Diagnostics.Process.Start("notepad");
else if (order=="calc") System.Diagnostics.Process.Start("calc");
else MessageBox.Show("Sorry Sir Your Request is not in my hand",order);

mytcpl.Stop();// Close TCP Session

if (mysocket.Connected ==true)// Looping While Connected to Receive
Another Message
        {
        while (true)
        {
        our_Server ();// Back to First Method
        }
        }
                }
```

VB.NET:

```
Private mytcpl As TcpListener
Private mysocket As Socket
Private myns As NetworkStream
Private mysr As StreamReader
```

```
Sub our_Server()
    mytcpl = New TcpListener(5020)
    mytcpl.Start()
    mysocket = mytcpl.AcceptSocket
    myns = New NetworkStream(mysocket)
    mysr = New StreamReader(myns)
    Dim order As String = mysr.ReadLine
    If order = "notepad" Then
        System.Diagnostics.Process.Start("notepad")
    Else
        If order = "calc" Then
            System.Diagnostics.Process.Start("calc")
        Else
Msgbox("Sorry Sir Your Request is not in my hand", order)
        End If
    End If
    mytcpl.Stop()
    If mysocket.Connected = True Then
        While True
            our_Server()
        End While
    End If
End Sub
```

حيث تقوم هذه الميثود بتصنت على ال Socket في حالة ورود أي Request يقوم بالموافقة عليه وإنشاء Session جديدة معه وفي حالة ورود أي بيانات عبر الSocket يتسلمها باستخدام ال StreamReader ويحولها إلى Text ثم نقوم بفحص الرسالة باستخدام الجمل الشرطية فمثلا إذا كانت الرسالة هي notepad يتم استدعائها باستخدام الميثود Start الموجودة ضمن الكلاس Process والموجودة في Name Spaces System.Diagnostics...

ولتشغيلها ضمن Thread جديد لابد من وضع تعريف ال Thread في حدث بدأ التشغيل للForm كما يلي :

C#

```
 private void Form1_Load(object sender, System.EventArgs e)
        {
Thread myth;
myth= new Thread  (new System.Threading .ThreadStart(our_Server));
myth.Start ();
        }
```

ثم قم بإضافة التالي في حدث ال Form Closing وذلك لتأكد من إغلاق الSocket وال Stream في البرنامج ..

```
private void Form1_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
{
try
        {
mytcpl.Stop ();
Application.ExitThread ();
Application.Exit();
```

```
        }
catch (Exception ex) {MessageBox .Show (ex.Message );}
```

VB.NET:

```
Private Sub Form1_Load(ByVal sender As Object, ByVal e As
System.EventArgs)
    Dim myth As Thread
    myth = New Thread(New System.Threading.ThreadStart(our_Server))
    myth.Start()
End Sub

Private Sub Form1_Closing(ByVal sender As Object, ByVal e As
System.ComponentModel.CancelEventArgs)
    Try
        mytcpl.Stop()
        Application.ExitThread()
        Application.Exit()
    Catch ex As Exception
         Msgbox(ex.Message)
    End Try
End Sub
```

Chapter 3
The Socket & Network Layer Programming

The Socket & Network Layer Programming

A. Socket Programming
B. Socket Class Members
C. TCP & UDP Classes Members
D. Asynchronous Sockets

بسم الله الرحمن الرحيم

3.1: Socket & Network Layer Programming :

في هذا الجزء سوف نبين بشكل أكثر تفصيلا عن برمجة طبقة ال Network Layer وهي التي يتم التعامل معها لإرسال واستقبال البيانات بعد تحويلها من و إلى Stream عبر الشبكة، قمنا سابقا باستخدام ال TCP و UDP للإرسال وللاستقبال وبينا الفرق بينهما وفي هذا الجزء سوف نتحدث عن ال Socket Programming وال TCP & UDP Classes وفي النهاية سوف نتحدث عن Asynchronous Socket .

أولا: Socket Programming

من المعروف أن الSocket هي الأداة التي يتم نقل البيانات من خلالها من جهاز إلى أخر ولاستخدامها يلزم في البداية تعريف Name Space System.Net.Sockets حيث يحتوي هذا Name Space على عدد ضخم من ال Classes والتي يتم استخدامها في برمجيات الشبكة وسوف نتحدث عن أهما وهو Socket Class إذ يمكننا بمن التعامل مع ال TCP أو ال UDP أو مع أي نوع أخر من البرتوكولات بشكل مباشر ويتكون ال Socket من ثلاثة بارومترات كما يلي: Object Method

C#:

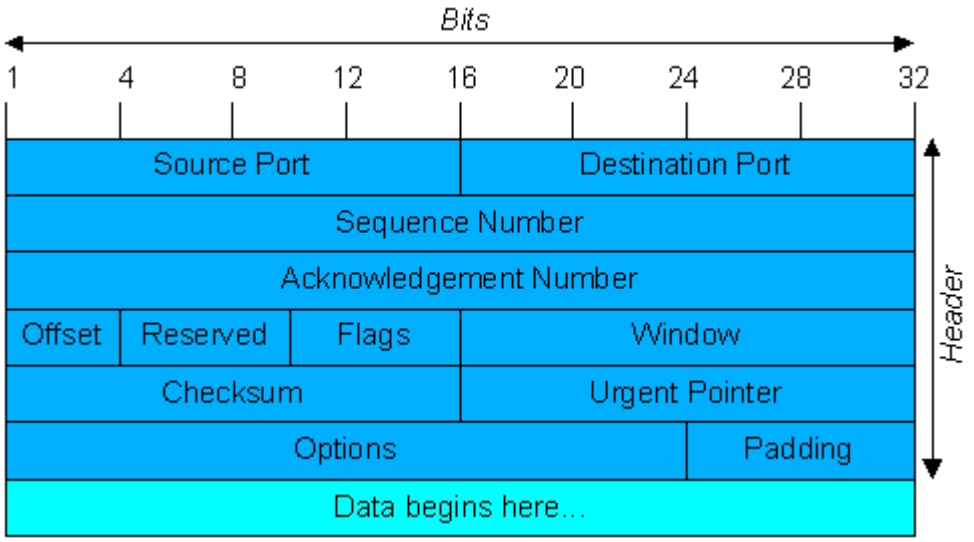
```
Socket MySocket = new Socket(AddressFamily. , SocketType., ProtocolType.);
```

VB.NET:

```
Dim MySocket As Socket = New Socket(AddressFamily, SocketType,
ProtocolType)
```

حيث يتم في البارومتر الأول تحديد نوعية ال IP Address والذي سوف تتعامل معه ويعطيك عدد كبير من الخيارات ومنها IPX والمستخدم في شبكات ال Novel أو ATM والمستخدم في شبكات ال ATM Networks أو NetBIOS Address وغيرها ... ومن أهم هذه الخيارات الInterNetwork وهو ما نستخدمه بشكل دائم مع البرمجيات الخاصة بالشبكات ويعرف على أن نوع IP هو من النوع IPv4 وهو المعتاد مع نظام مايكروسوفت وأغلب أنظمة التشغيل المعروفة حاليا وفي المستقبل القريب جدا سيتم الإستغناء عنه وليحل محله ال IPv6، في البارومتر الثاني يتم تحديد نوع ال Socket أي هل سوف نستخدم Stream لإرسال البيانات أو شيء اخر وعادة ما يتم استخدام ال Stream لهذه المهمة حيث اننا سنعتمد نمطية التراسل من النوع Stream ، وأخيرا نحدد نوع البروتوكول المستخدم للإتصال فهل هو من النوع UDP أو TCP أو بروتوكولات أخرى مثلICMP Internet Control Message Protocol أو IGMP Internet Group Management Protocol أو اننا نريد مثلا إنشاء ال Socket لتعريف IP Security Header بإختيار IPSecAuthenticationHeader وغيرها وسوف نأتي على شرح مثل هذه الأمور لاحقا إنشاء الله، وهنا سوف نختار ال TCP أو UDP ومن المعروف أن برتوكول ال TCP هو برتوكول موجه وهذا يعني إجراء عملية التحقق من الوصول والتوصيل إلى شخص ما محدد أما برتوكول ال UDP فهو برتوكول سريع نسبيا و لاكنه لا يدعم عملية التحقق من الوصول السليم للبيانات المرسلة وهو مفيد جدا لإجراء عملية البث الإذاعي Broadcast وإنشاء مجموعات البث Multicast Group وهو ما شرحناه في الجزء الأول والثاني أنظر إلى الشكل التالي ويبين فيه ال Header الخاص بال TCP وال Header الخاص بال UDP ولاحظ الفرق بينهما:

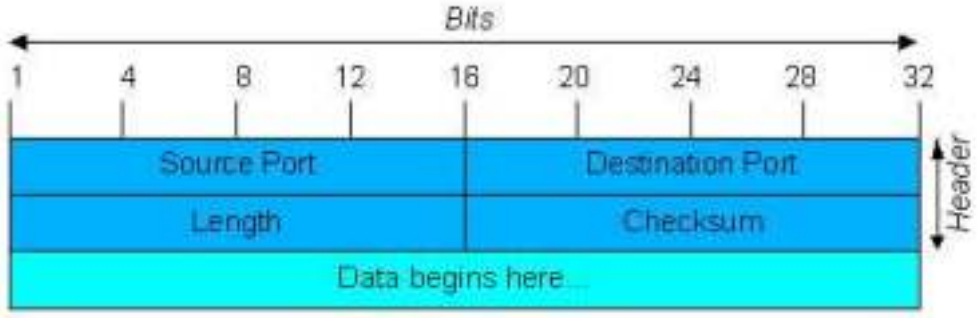
أولا ال TCP Header ويتكون من 32 Bits للبكت الواحد حيث يتم فيه تخزين عنوان المرسل في 16 Bits والمستقبل في 16 Bits والرقم التسلسلي للبكت في 32 Bits ورقم التحقق بالإضافة إلى ال Checksum وفي النهاية يتم وضع الجزء الخاص بالبيانات :



TCP Header

Data Offset: 4 bits the number of 32 bit words in the TCP Header. This indicates where the data begins. The TCP header (even one including options) is an integral number of 32 bits long.
Window: The number of data octets beginning with the one indicated in the acknowledgment field which the sender of this segment is willing to accept.
...

ثانيا ال UDP Header ويتكون من 32 Bits من البيانات للبكت الواحد ويحتوي على عنوان المرسل 16 Bits أما المستلم و ال Checksum فهما اختياريان وبشكل افتراضي لا يتم استخدامهم في عملية الإرسال:

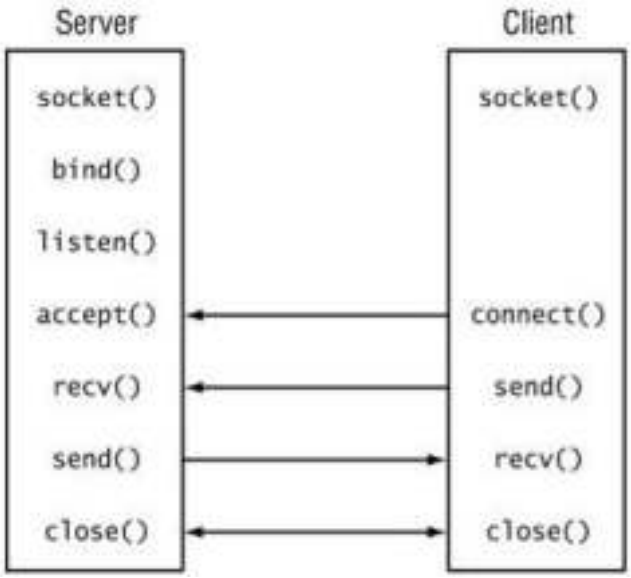


UDP Header

The Checksum in UDP Header. 16 bits.
Computed as the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the UDP header, and the data, padded as needed with zero bytes at the end to make a multiple of two bytes. If the checksum is cleared to zero, then checksuming is disabled. If the computed checksum is zero, then this field must be set to 0xFFFF.
...

استخدام ال Socket Programming لإنشاء TCP Connection :

تمر عملية الاتصال باستخدام ال TCP Socket Connection بمجموعة من المراحل وهي كما في الشكل التالي :



إذ تبدأ العملية في ال Client و ال server بإنشاء ال Socket كما يلي :

C#:

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

VB.NET:

```
Dim MySocket As Socket = New Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp)
```

ثم ربط ال Socket مع الكمبيوتر الحالي باستخدام الميثود Bind وتستخدم فقط عند الاستقبال وكما يلي :

C#:

```
IPEndPoint ip = new IPEndPoint(IPAddress.Any, 5020);
MySocket.Bind(ip);
```

VB.NET:

```
Dim ip As IPEndPoint = New IPEndPoint(IPAddress.Any, 5020)
 MySocket.Bind(ip)
```

ثم القيام بعملية التصنت على الPort المحدد باستخدام الميثود listen ويمكنك تحدد عدد الأجهزة التي سيتم قبولها ولوضع عدد غير محدد نمرر له الرقم -1 ثم نقوم بالموافقة على الاتصال باستخدام الميثود accept وكما يلي :

C#:

```
MySocket.Listen(-1);
MySocket.Accept();
```

VB.NET:

```
MySocket.Listen(-1)
 MySocket.Accept
```

ويتم استقبال البيانات من خلال الميثود Receive حيث تعبئ البيانات في مصفوفة من النوع Byte وكما يلي :

C#:

```
byte[]Received=new byte[1024];
MySocket.Receive(Received);
```

VB.NET:

```
Dim Received(1024) As Byte
 MySocket.Receive(Received)
```

وهنا قمنا بإنشاء Connection من النوع TCP وبتعريفها على الPort(5020 كمثال) حيث يتم ربطها بال Socket باستخدام الميثود Bind وقمنا بتعريف Listen لا نهائي العدد -1 ..

ولتعريف برنامج الإرسال TCP Client باستخدام ال Socket لابد من تعريف الSocket مرة أخرى وإسناد عنوان الServer ورقم الPort بنقطة الهدف IPEndPoint ثم إرسال البيانات باستخدام الميثود Send وتتم عملية الإرسال بما تم تعريفه في ال socket حيث سنستخدم Stream Socket وكما يلي :

C#:

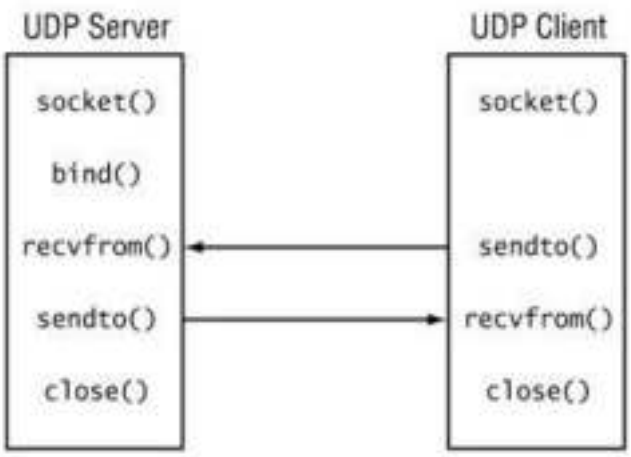
```
String str = Console.ReadLine();
ASCIIEncoding asen = new ASCIIEncoding();
byte[] msg = asen.GetBytes(str);

Socket MySocket = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);
IPEndPoint remote = new IPEndPoint(IPAddress.Parse("192.168.1.101"),
5020);
MySocket.Connect(remote);
MySocket.Send(msg);
MySocket.Close();
```

VB.NET:

```
Dim str As String = Console.ReadLine
Dim asen As ASCIIEncoding = New ASCIIEncoding
Dim msg As Byte() = asen.GetBytes(str)
Dim MySocket As Socket = New Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp)
Dim remote As IPEndPoint = New
IPEndPoint(IPAddress.Parse("192.168.1.101"), 5020)
 MySocket.Connect(remote)
 MySocket.Send(msg)
 MySocket.Close
```

استخدام ال Socket Programming لإنشاء UDP Connectionless :

تمر عملية الاتصال باستخدام ال UDP Socket Connection بمجموعة من المراحل وهي كما في الشكل التالي :



وتتشابه عملية الاتصال كما في ال TCP إذ تبدأ العملية في ال Client و ال server بإنشاء ال Socket كما يلي :

C#:

```
Socket MySocket = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Udp);
```

VB.NET:

```
Dim MySocket As Socket = New Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Udp)
```

ثم ربط ال Socket مع الكمبيوتر الحالي باستخدام الميثود Bind وتستخدم فقط عند الاستقبال وكما يلي :

C#:

```
IPEndPoint sender = new IPEndPoint(IPAddress.Any, 5020);
MySocket.Bind(sender);
```

VB.NET:

```
Dim sender As IPEndPoint = New IPEndPoint(IPAddress.Any, 5020)
 MySocket.Bind(sender)
```

ولاستقبال البيانات نستخدم الميثود ReceiveFrom حيث نعرف في البداية End Point Reference بناء على ما تم تعريفه في السابقة ونمرره ك reference مع مصفوفة ال Byte إلى الميثود ReceiveFrom ومن ثم نستطيع تحويل المصفوفة إلى String من خلال الميثود GetString الموجودة ضمن الكلاس ASCII وكما يلي :

C#:

```
int recv;
byte[] data = new byte[1024];
EndPoint Remote = (EndPoint) (sender);
recv = newsock.ReceiveFrom(data, ref Remote);
Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));
```

VB.NET:

```
Dim recv As Integer
Dim data(1024) As Byte
```

```
Dim Remote As EndPoint = CType((sender), EndPoint)
 recv = newsock.ReceiveFrom(data, Remote)
 Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv))
```

ويتم في الإرسال استخدام الميثود SendTo حيث نمرر لها البيانات بعد تحويلها من String إلى Byte Array وحجم البيانات المرسلة إذ يمكننا معرفته من خلال الميثود Length ونوع ال Flags حيث نريد عمل Broadcast لرسالة المرسلة واخيرا نمرر له ال EndPoint Object وكما يلي:

C#:

```
string welcome = "Hello All";
data = Encoding.ASCII.GetBytes(welcome);
newsock.SendTo(data, data.Length, SocketFlags.Broadcast, Remote);
```

VB.NET:

```
Dim welcome As String = "Hello All"
 data = Encoding.ASCII.GetBytes(welcome)
 newsock.SendTo(data, data.Length, SocketFlags.Broadcast, Remote)
```

يمكن وضع هذا الأكواد في Infinity While Loop بحيث لا تنتهي أو يمكن تحديدها بعدد معين من عمليات الإرسال والاستقبال ..

ثانيا: Socket Classes Members :

1- IPAddress Class : ويستخدم لتعريف IP Address حيث يمكن إسناده إلى ال IPEndPoint كمثال والصيغة العامة له كما يلي:

C#:

```
IPAddress newaddress = IPAddress.Parse("192.168.1.1");
```

VB.NET:

```
Dim newaddress As IPAddress = IPAddress.Parse("192.168.1.1")
```

ويمكن الإختيار بين اربعة خيارات في تحديد العنوان وهي كما يلي :

Any ويستخدم لتمثيل أي عنوان متاح على الشبكة

Broadcast ويستخدم لتمثيل البث الإذاعي لجميع الأجهزة على الشبكة

Loopback ويستخدم لتمثيل العنوان المعروف لل loopback وهو 127.0.0.1

None ويستخدم في حالة عدم وجود Network Interfase في النظام

كما يدعم مجموعة من الميثود وأهمها :

Equals يستخدم هذا الميثود بشكل عام للمقارنة بين tow Objects وهنا سيستخدم للمقارنة بين عنوانين ويرجع True إذا كانا متشابهين و False إذا كانا مختلفين.

GetHashCode وتستخدم لإرجاع العنوان إلى صيغة Hash Code

HostToNetworkOrder ويرجع الجزء الخاص بال Network من العنوان

NetworkToHostOrder ويرجع الجزء الخاص بال Host من العنوان

2- **IPEndPoint Class :** حيث استخدمناه لتحديد العنوان وال Port لل Host والذي نريد الاتصال به والصيغة العامة له كما يلي :

C#:

```
IPEndPoint end = new IPEndPoint(IPAddress.Parse("192.168.1.1"), 5020);
```

VB.NET:

```
Dim end As IPEndPoint = New IPEndPoint(IPAddress.Parse("192.168.1.1"),
5020)
```

مجموعة الخواص التي تدعم في ال Socket Class وهي كما يلي:

AddressFamily ويرجع مجموعة العناوين المعرفة على ال Socket

Available ويرجع حجم البيانات الجاهزة للقراءة من ال Socket

Blocking ويعطي Get أو Set لمعرفة إذا كان ال socket يستخدم ال Blocking Mode أم لا

Connected وتستخدم هذه الخاصية بكثرة لمعرفة إذا كان ال Socket متصل مع ال Remote Host أم لا

Handle ويستخدم لمعرفة نظام التشغيل الذي يتعامل مع ال Socket

ProtocolType ويستخدم لمعرفة البروتوكول الذي يستخدم في ال Socket

RemoteEndPoint ويرجع معلومات عن ال Socket الذي يستخدم مع ال Remote Host

وكمثال لاستخداماتها:

C#:

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

```
        fmo.Close();
    }

}
```

VB.NET:

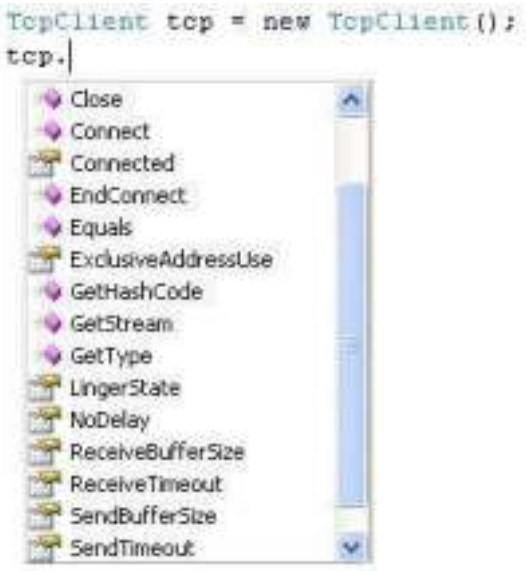
```
imports System
imports System.Net
imports System.Net.Sockets

Public Shared Sub Main()
    Dim ia As IPAddress = IPAddress.Parse("127.0.0.1")
    Dim ie As IPEndPoint = New IPEndPoint(ia, 8000)
    Dim fmo As Socket = New Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp)
    Console.WriteLine("AddressFamily: {0}", fmo.AddressFamily)
    Console.WriteLine("SocketType: {0}", fmo.SocketType)
    Console.WriteLine("ProtocolType: {0}", fmo.ProtocolType)
    Console.WriteLine("Blocking: {0}", fmo.Blocking)
    fmo.Blocking = False
    Console.WriteLine("new Blocking: {0}", fmo.Blocking)
    Console.WriteLine("Connected: {0}", fmo.Connected)
    fmo.Bind(ie)
    Dim iep As IPEndPoint = CType(fmo.LocalEndPoint, IPEndPoint)
    Console.WriteLine("Local EndPoint: {0}", iep.ToString)
    fmo.Close()

End Sub
```

حيث سترجع المعلومات التالية:

```
AddressFamily: InterNetwork
SocketType: Stream
ProtocolType: Tcp
Blocking: True
new Blocking: False
Connected: False
Local EndPoint: 127.0.0.1:8000
Press any key to continue . . .
```

ثالثا: TCP & UDP Classes Members :

أولا ال Classes الخاصة بال TCP Connection Oriented Protocol :



TcpClient Class-1: حيث تحتوي على مجموعة من ال Methods وال Properties وهي كما يلي:

أولا: أهم الميثود الخاصة بها TCPClient Methods :

Connect: وتستخدم لأجراء عملية الاتصال مع ال server حيث نمرر فيها عنوان ال IP الخاص بال Server و رقم الPort وكما يلي:

C#:

```
TcpClient tcp = new TcpClient();
tcp.Connect(IPAddress.Parse("192.168.1.1"),5020);
```

VB.NET:

```
Dim tcp As TcpClient = New TcpClient
 tcp.Connect(IPAddress.Parse("192.168.1.1"), 5020)
```

Close: لإنهاء الاتصال مع ال TCP Socket.
EndConnect: لإنهاء Asynchronies Connection حيث ترجع Asynchronies Result.
GetStream: ويستخدم لقراءة ال Stream من ال Socket في عملية الإرسال و الاستقبال.

ثانيا: أهم الخصائص TCPClient Properties :

LingerState : وتأخذ get أو Set لتحديد أو معرفة ال Linger Time
NoDelay: وتأخذ get أو Set لتحديد أو معرفة إذا كان هناك وقت معين لتأخير أم لا
ExclusiveAddressUse: وتأخذ get أو Set لتحديد أو معرفة الSocket يسمح باستخدام ال Client Port أم لا.
SendBufferSize و ReceiveBufferSize: وتأخذ get أو Set لتحديد أو معرفة حجم ال Buffer المستخدم في ال stream والمعرف في TCP Client Object.
SendTimeout و ReceiveTimeout: وتأخذ get أو Set لتحديد أو معرفة الوقت المتاح لعملية الإرسال أو الإستقبال حيث يعطي Time Out في حالة أنه لم يجد الطرف الأخر خلال فترة زمنية معينة.

TcpListener Class-2: حيث تحتوي على مجموعة من ال Methods وال Properties وهي كما يلي:

```
TcpListener tcp_Listener = new TcpListener(IPAddress.Any,5020);
tcp_Listener.
```

أولا: أهم الميثود الخاصة بها TcpListener Methods :

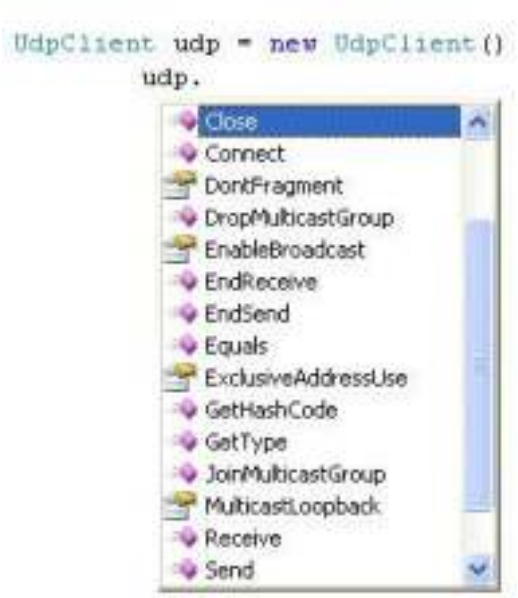
AcceptSocket: وتستخدم لقبول عملية الاتصال مع ال Client.
Start : وهي Overloaded Method حيث انه في حالة تمرير رقم إليها يتم تحديد عدد الأجهزة التي تسمح بوجودها في الطابور أو ال Qouee وبدون تحديد رقم معين يصبح ال Qouee غير محدد.
Stop : وتستخدم لإغلاق عملية التصنت ويفضل وضعها في ال Finally عند استخدام ال Try و ال Catch حتى يتم إنهاء عملية التصنت في حالة حدوث أي Exception.

ثانيا: أهم الخصائص في TcpListener:

LocalEndpoint : حيث يرجع ال IP ورقم الPort المستخدم في ال LocalEndpoint المحدد.
Server: ومن خلالها نستطيع الوصول إلى كل الخصائص و الميثود في ال TCP Server والتي شرحناها سابقا مثل ال Accept وال Sendto وال Receive و Listen وغيرها

ثانيا ال Classes الخاصة بال UDP Connectionless Protocol :

UdpClient Class-1: وتستخدم لتعريف UDP Datagram Protocol Connection قمنا سابقا بتعريفها والتعامل معها وفي هذا الجزء سنبين أهم محتوياتها وهي كما يلي :



ومن أهم الميثود والخصائص الخاصة بها :

JoinMulticastGroup و DropMulticastGroup: لضم أو إلغاء عنوان أو مجموعة من العناوين من ال Multicast Group.
EnableBroadcast: وتأخذ Get أو Set لتفعيل ال Broadcasting في ال socket.
MulticastLoopback: وتأخذ Get أو Set لمعرفة أو تحديد ال Mlticast Loopback.

2-MulticastOption Class: ويستخدم في ال Multicasting حيث يتم تخزين IP Address List لتعامل معها في Multicast Group لعمل Join و Drop لأي Multicast Group وتستخدم كما يلي كمثال لإضافة عضوية لاستقبال رسائل Multicast :

أولا نعرف ال UDP Socket وكما يلي :

C#:
```
mcastSocket = new Socket(AddressFamily.InterNetwork,SocketType.Dgram,
ProtocolType.Udp);
```

VB.NET:
```
mcastSocket = New Socket(AddressFamily.InterNetwork, SocketType.Dgram,
ProtocolType.Udp)
```

ثانيا نقوم بتعريف Address List ثم نسند إليها ال IP الذي نريد إدخاله في ال Group أو نجعل ال User يدخل العنوان بنفسه نربطها بالسكوت باستخدام الميثود Bind وكما يلي :

C#:
```
IPAddress localIPAddr = IPAddress.Parse(Console.ReadLine());
mcastSocket.Bind(IPlocal);
```

VB.NET:
```
Dim localIPAddr As IPAddress = IPAddress.Parse(Console.ReadLine)
 mcastSocket.Bind(IPlocal)
```

ثالثا نقوم بتعريف ال Multicast Option ونسند لها العنوان المحدد كما يلي:

C#:

```
MulticastOption mcastOption;
mcastOption = new MulticastOption(localIPAddr);
```

VB.NET:

```
Dim mcastOption As MulticastOption
 mcastOption = New MulticastOption(localIPAddr)
```

ومن ثم نضيف التغير على ال حيث تأخذ هذه الميثود ثلاثة بارومیترات الأول لتحديد مستوى التغيير على IP أو على IPv6 أو على Socket أو TCP أو UDP وفي حالتنا هذه سوف نستخدم التغير على IP إذ ما نريده هو ضم IP إلى Multicast Group وفي البارومیتر الثاني نحدد نوع التغيير حيث نريد إضافة عضوية ويمكن الاختيار بين إضافة عضويه AddMembership أو إلغاء عضوية DropMembership وأخيرا نسند إليه ال MulticastOption Object والذي قمنا بإنشائه و كما يلي:

C#:

```
mcastSocket.SetSocketOption(SocketOptionLevel.IP,
SocketOptionName.AddMembership,mcastOption);
```

VB.NET:

```
mcastSocket.SetSocketOption(SocketOptionLevel.IP,
SocketOptionName.AddMembership, mcastOption)
```

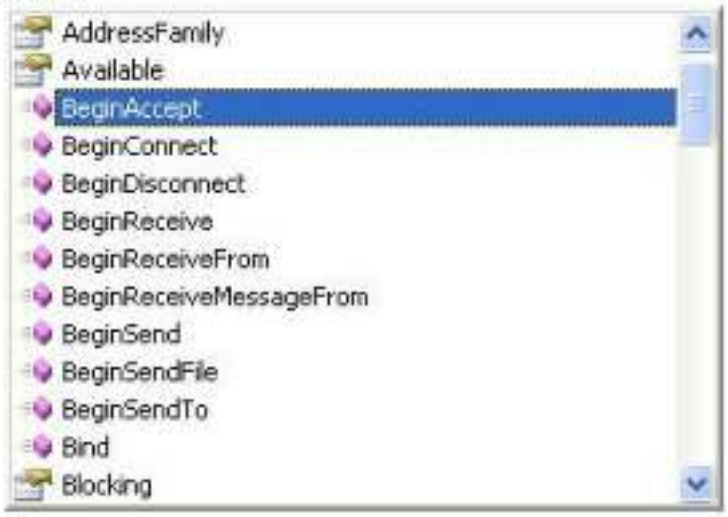
3.2: Asynchronous Sockets Programming :

سوف نتحدث في هذا الجزء عن استخدام ال Asynchronous Socket بشكل أكثر تفصيلا عما تحدثنا به سابقا وسوف نطبق مجموعة من الأمثلة العملية على استخدام الاتصال الغير متزامن في برمجيات الشبكات ...
من المعروف أن الاتصال المتزامن مهم جدا في البرمجيات التي تحتاج إلى العمل في الزمن الحقيقي حيث لا يسمح باستخدام الاتصال لأمر آخر إلى بعد انتهاء العملية الجارية واستخدامه مهم جدا في العمليات التي تتطلب مثل هذه الأمور لا كن لا ينصح أبدا استخدامه في حالة إذا كانت الجهة المستقبلة للبيانات تستخدم Slow Connection كاعتماد الشبكة على ال Dialup لربط الجهازين المرسل مع المستقبل أو في حالة إذا كان هنالك مجموعة كبيرة من المستخدمين تستخدم الServer حيث يمنع الأسلوب المتزامن بقية المستخدمين على الشبكة من إجراء عملية الإرسال في حالة كون ال Server يستقبل بيانات من جهاز آخر ، وفي هذه الحالة ينصح باستخدام الاتصال الغير المتزامن إذ يعتبر مهم جدا في حالة إذا أردنا من البرنامج القيام بعدة مهام وعلى نفس ال Thread وباستخدام نفس ال Connection ، أو كما ذكرنا سابقا في حالة إذا كان الاتصال بطيء نسبيا أو انه يوجد عدد مستخدمين يستخدمون نفس ال Server ..

أولا : Asynchronous Socket Class and its members

تدعم الدوت نيت الاتصال غير المتزامن بمجموعة من ال methods الموجودة ضمن ال Socket Class والتي يتم استدعائها من ال System.Net.Socket Namespaces وقد ميزت الدوت نيت هذه الميثودس بوجود ال Begin في بداية أسم الميثود، ولكل Begin Method يوجد End Method مقابلة لها والتي تستخدم لإرجاع callback result عند انتهاء ال Begin Method من التنفيذ وهي كما يلي:

```
Socket MySocket = new Socket(AddressFamily
MySocket.b
```



1- BeginAccept و تستخدم لقبول ال Client Request وإسناده إلى ال Object AsyncCallback وباستخدام هذه الطريقة سوف يتمكن ال Server من استقبال عدد من ال Clients Requests في نفس الوقت وبدون الحاجة لانتظار الانتهاء من العملية الجارية حيث يتم في كل مرة استدعاء الميثود باستخدام ال AsyncCallback Delegate وتستخدم كما يلي كما يلي:

C#:

```
m_mainSocket = new Socket(AddressFamily.InterNetwork,
SocketType.Stream,ProtocolType.Tcp);
IPEndPoint ipLocal = new IPEndPoint (IPAddress.Any, 5020);

m_mainSocket.Bind (ipLocal);
m_mainSocket.Listen (10);
m_mainSocket.BeginAccept (new AsyncCallback (Client_request_method),
null);
```

VB.NET:

```
m_mainSocket = New Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp)
Dim ipLocal As IPEndPoint = New IPEndPoint(IPAddress.Any, 5020)
 m_mainSocket.Bind(ipLocal)
 m_mainSocket.Listen(10)
 m_mainSocket.BeginAccept(New AsyncCallback(Client_request_method),
Nothing)
```

حيث سيتم إضافة ال Client Request في Callback Reference منفصل عن السابق وهنا لابد من إنشاء method لاستقبال ال Client Request وإنهاء ال Client Accepted باستخدام الميثود EndAccept Object :

C#:

```
public void Client_request_method(IAsyncResult ar)
   {
Socket listener = (Socket)ar.AsyncState;
Myclient = listener.EndAccept(ar);
Myclient.Send(/* data to be send*/ );
listener.BeginAccept(new AsyncCallback(Client_request_method), listener);
Console.WriteLine("Socket connected to {0}",
client.RemoteEndPoint.ToString());    }
```

VB.NET:

```
Dim listener As Socket = CType(ar.AsyncState, Socket)
 Myclient = listener.EndAccept(ar)
 Myclient.Send
 listener.BeginAccept(New AsyncCallback(Client_request_method), listener)
 Console.WriteLine("Socket connected to {0}",
client.RemoteEndPoint.ToString)
```

في Dot Net 2005 أصبحت الBeginAccept Method تأخذ عدة أشكال كما يلي:

الشكل الأول في الدوت نيت 2003 و 2005 وتأخذ AsyncCallBack Delegate و State Object لإرجاع معلومات عن حالة ال Request في ال Socket وكما يلي:

```
MySocket.BeginAccept(AsyncCallback , object state)
```

الشكل الثاني في الدوت نيت 2005 حيث يمكنك فيه تحديد حجم البيانات المستلمة
MySocket.BeginAccept(int Data_ Receive_Size , AsyncCallback , object state)

الشكل الثالث في الدوت نيت 2005 حيث بمكن فيه تحديد ال Accepted Socket
MySocket.BeginAccept(Socket accept_Socket ,int Data_ Receive_Size , AsyncCallback , object state)

2- BeginConnect وتستخدم لبدأ Asynchronous Connection على ال Socket ورقم الPort المحدد حيث يسند لها ال IPEndPoint وال Asynchronous Callback وال State Object وكما يلي:
MySocket.BeginConnect(EndPoint IP,Syncallback Result,object state)

وتستخدم كما يلي كمثال:

C#:
```
Socket MySocket = new Socket (AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);
IPEndPoint ipend = new IPEndPoint(IPAddress.Parse("192.168.1.101"),
5020);

 MySocket.BeginConnect(ipend, new AsyncCallback(Connected), MySocket);
```

VB.NET:
```
Dim MySocket As Socket = New Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp)
Dim ipend As IPEndPoint = New
IPEndPoint(IPAddress.Parse("192.168.1.101"), 5020)
```

في ال Connected Method يتم تحديد ال CallBack Socket كما يلي:

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

VB.NET:
```
Public Shared Sub Connected(ByVal iar As IAsyncResult)
    Dim sock As Socket = CType(iar.AsyncState, Socket)
    Try
        sock.EndConnect(iar)
    Catch generatedExceptionVariable0 As SocketException
        Console.WriteLine("Unable to connect to host")
    End Try
```

```
End Sub
```

3- BeginReceive وتستخدم لإستقبال بيانات من ال Client وتخزينها في Byte Array والصيغة العامة لها كما يلي:

```
MySocket.BeginReceive(Byte[] buffer,int offset, SocketFlags,AsyncCallback,
object sate)
```

ويستخدم كما يلي كمثال:

C#:

```
byte[] data = new byte[1024];
 MySocket.BeginReceive(data, 0, data.Length, SocketFlags.None, new
AsyncCallback(ReceivedData), MySocket);

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

VB.NET:

```
Dim data(1024) As Byte
MySocket.BeginReceive(data, 0, data.Length, SocketFlags.None, New
AsyncCallback(ReceivedData), MySocket)

Sub ReceivedData(ByVal iar As IAsyncResult)
   Dim remote As Socket = CType(iar.AsyncState, Socket)
   Dim recv As Integer = remote.EndReceive(iar)
   Dim receivedData As String = Encoding.ASCII.GetString(data, 0, recv)
   Console.WriteLine(receivedData)
End Sub
```

كما تستخدم الميثود BeginReceiveFrom لإستقبال البيانات من موقع محدد باستخدام ال UDP حيث يضاف إلى التركيب السابق IPEndPoint Refrance Object .

4- BeginSend وتستخدم لإرسال بيانات إلى الطرف المستقبل عبر ال Asynchronous Socket والصيغة العامة لها كما يلي:

```
MySocket.BeginSend (Byte[] buffer,int offset, SocketFlags,AsyncCallback,
object sate)
```

وتستخدم كما يلي كمثال:

C#:

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

```
byte[] data = Encoding.ASCII.GetBytes("Hello Word");
MySocket.BeginSend(data, 0, data.Length, SocketFlags.None,
new AsyncCallback(SendData), MySocket);
```

VB.NET:

```
Private Shared Sub SendData(ByVal iar As IAsyncResult)
    Dim server As Socket = CType(iar.AsyncState, Socket)
    Dim sent As Integer = server.EndSend(iar)
End Sub
Dim data As Byte() = Encoding.ASCII.GetBytes("Hello Word")
MySocket.BeginSend(data, 0, data.Length, SocketFlags.None, AddressOf
SendData, MySocket)
```

كما تستخدم الميثود BeginSendto لإرسال البيانات إلى Remote Host محدد باستخدام ال UDP حيث يضاف إلى التركيب السابق IPEndPoint Refrance Object .

5- كما تم إضافة مجموعة من الميثود الجديدة في الدوت نيت 2005 وهي:

BegonDiconnect لإنهاء الاتصال و **BeginSendFile** لإرسال ملف و ال **BeginReceiveMessageFrom** والتي تستخدم لإستقبال عدد محدد من البيانات وتخزينها في مكان محدد في ال Bufer ..

تأخذ ال**BeginSendFile** التركيب التالي:

```
MySocket.BeginSendFile(string filename,AsyncCallback Asyn,object state)
```

وال **BeginReceiveMessageFrom** التركيب التالي:

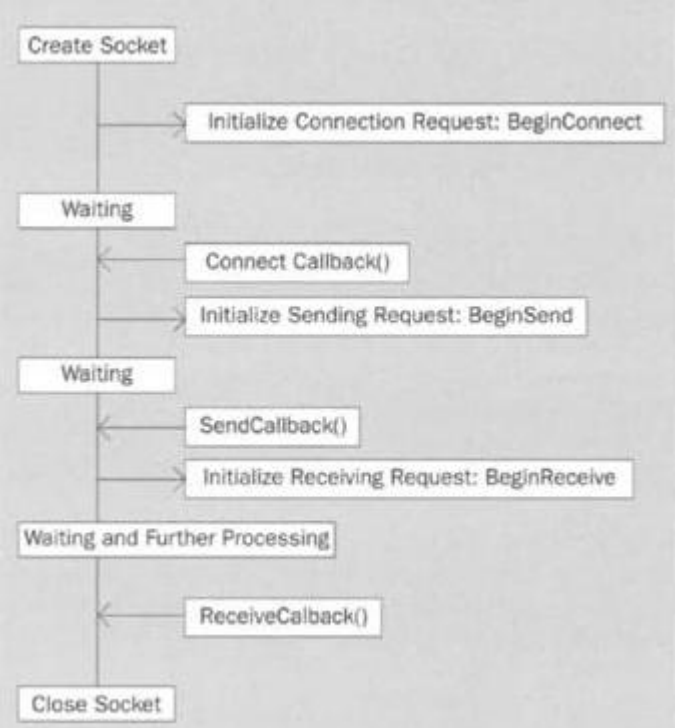
```
MySocket.BeginReceiveMessageFrom(byte Buffer ,int offset,int
size,SocketFlags sf,ref EndPoint,AsyncCallback ascb,object state)
```

وال **BegonDiconnect** التركيب التالي:

```
MySocket.BeginDisconnect(bool reuseSocket,AsyncCallback ascb,object state)
```

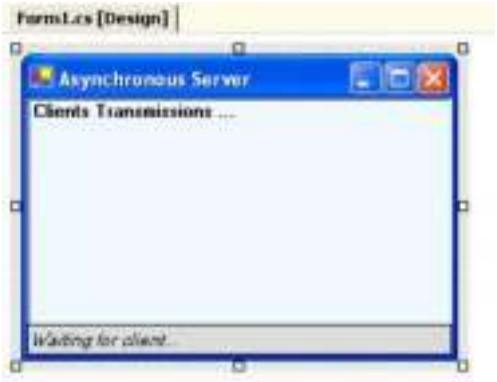
ثانيا: تطبيقات ال Asynchronous Socket في الدوت نت :

تمر عملية الاتصال الغير متزامن بمجموعة من المراحل تبدأ بإنشاء الSocket Object في ال Server Side بعد ذلك يتم تعريف ال BeginConnect لبدأ Asynchronous Connection على ال Socket حيث يتم إسناد IPEndPoint Object والMethod Asynchronous Callback وال State Object لها وتبدأ في هذه الحالة عملية الاتصال بال Socket ، وبعد ذلك تمرر إلى ال BeginAccept لقبول ال Client Request حيث يتم قبول الطلب ويرسل Acknowledgement إلى ال Client ليعلمه فيها بقبول الجلسة وإمكانية البدء للإرسال و يستطيع ال Client بعد الموافقة على الجلسة البدء بالإرسال باستخدام الميثود BeginSend ويستقبل ال Server الرسالة من ال Client باستخدام الميثود BeginReceive وكما ذكرنا سابقا فإن لكل عملية Begin تقابلها الميثود End للاستعداد لإجراء عملية أخرى على نفس ال Thread في البرنامج وهو ما ميز الاتصال الغير متزامن عن الاتصال المتزامن.



وبناء على المفاهيم السابقة سوف نقوم الآن بإنشاء برنامج Client/Server Chatting يعتمد على ال Asynchronous Socket لإرسال واستقبال البيانات .

وللبدء قم بإنشاء مشروع جديد كما في الشكل التالي:



سوف نستخدم ال Namespaces التالية:

C#:

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

VB.NET:

```
Imports System.Net
Imports System.Net.Sockets
Imports System.Text
```

في ال Global Declaration (أي بعد تعريف ال Main Class)قم بإضافة التعاريف التالية:

C#:

```
public class Form1 : System.Windows.Forms.Form
{
Socket server = new Socket(AddressFamily.InterNetwork,SocketType.Stream,
ProtocolType.Tcp);
IPEndPoint iep = new IPEndPoint(IPAddress.Any, 5020);
private byte[] data = new byte[1024];
private int size = 1024;
```

VB.NET:

```
Public Class Form1 Inherits System.Windows.Forms.Form

Private server As Socket = New Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp)
 Private iep As IPEndPoint = New IPEndPoint(IPAddress.Any, 5020)
 Private data As Byte() = New Byte(1024) {}
 Private size As Integer = 1024
```

في ال Form Load قم بإضافة الكود التالي حيث سنعرف Connection يعتمد على ال TCP ويعمل على الPort 5020 ثم تعريف عملية قبول الاتصال باستخدام ال BeginAccept :

C#:

```
private void Form1_Load(object sender, System.EventArgs e)
{
server = new Socket(AddressFamily.InterNetwork,SocketType.Stream,
ProtocolType.Tcp);
IPEndPoint iep = new IPEndPoint(IPAddress.Any, 5020);
server.Bind(iep);
server.Listen(5);
server.BeginAccept(new AsyncCallback(AcceptConn), server);
}
```

VB.NET:

```
Private Sub Form1_Load(ByVal sender As Object, ByVal e As
System.EventArgs)
    server = New Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp)
    Dim iep As IPEndPoint = New IPEndPoint(IPAddress.Any, 5020)
    server.Bind(iep)
    server.Listen(5)
    server.BeginAccept(New AsyncCallback(AcceptConn), server)
End Sub
```

ثم إنشاء Accept Callback Method والذي سيتم فيه إنهاء ال Accepted Request باستخدام ال EndAccept Method وبعد ذلك إرسال Acknowledgement إلى ال Client تخبره فيها بقبول الطلب وترسل باستخدام ال BeginSend Method كما يلي:

C#:

```
void AcceptConn(IAsyncResult iar)
{
Socket oldserver = (Socket)iar.AsyncState;
Socket client = oldserver.EndAccept(iar);
conStatus.Text = "Connected to: " + client.RemoteEndPoint.ToString();
string stringData = "Welcome to my server";
byte[] message1 = Encoding.ASCII.GetBytes(stringData);
client.BeginSend(message1, 0, message1.Length, SocketFlags.None,new
AsyncCallback(SendData), client);
}
```

VB.NET:

```
Sub AcceptConn(ByVal iar As IAsyncResult)
    Dim oldserver As Socket = CType(iar.AsyncState, Socket)
    Dim client As Socket = oldserver.EndAccept(iar)
    conStatus.Text = "Connected to: " + client.RemoteEndPoint.ToString
    Dim stringData As String = "Welcome to my server"
    Dim message1 As Byte() = Encoding.ASCII.GetBytes(stringData)
    client.BeginSend(message1, 0, message1.Length, SocketFlags.None, New
AsyncCallback(SendData), client)
End Sub
```

ثم إنشاء Send Callback method لإنهاء ال BeginSend وكما يلي:

C#:

```
void SendData(IAsyncResult iar)
{
Socket client = (Socket)iar.AsyncState;
int sent = client.EndSend(iar);
client.BeginReceive(data, 0, size, SocketFlags.None,new
AsyncCallback(ReceiveData), client);
}
```

VB.NET:

```
Sub SendData(ByVal iar As IAsyncResult)
    Dim client As Socket = CType(iar.AsyncState, Socket)
    Dim sent As Integer = client.EndSend(iar)
    client.BeginReceive(data, 0, size, SocketFlags.None, New
AsyncCallback(ReceiveData), client)
End Sub
```

ثم إنشاء Receive Callback method لإنهاء ال BeginReceive وكما يلي:

C#:

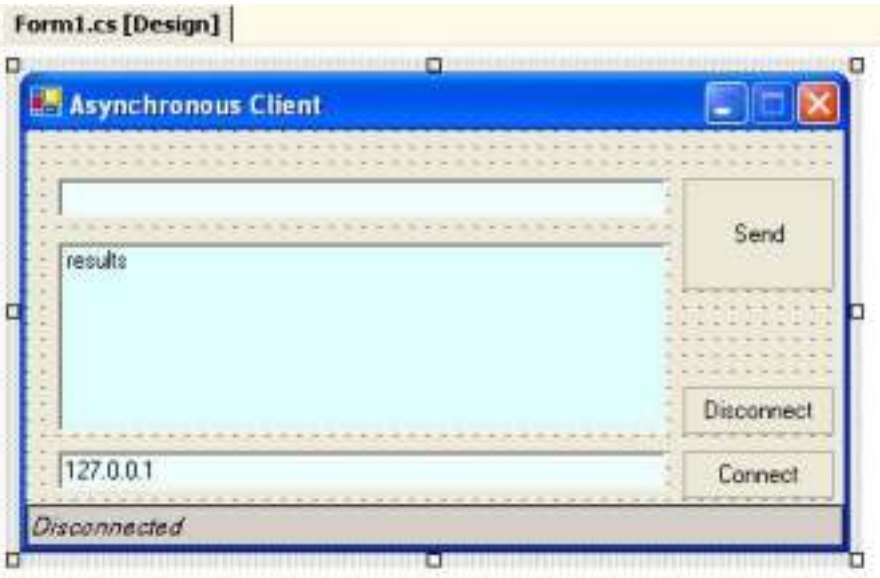
```
void ReceiveData(IAsyncResult iar)
{
Socket client = (Socket)iar.AsyncState;
int recv = client.EndReceive(iar);
        if (recv == 0)
        {
```

```
                client.Close();
                conStatus.Text = "Waiting for client...";
                server.BeginAccept(new AsyncCallback(AcceptConn), server);
                return;
            }
string receivedData = Encoding.ASCII.GetString(data, 0, recv);
results.Items.Add(receivedData);
byte[] message2 = Encoding.ASCII.GetBytes(receivedData);
client.BeginSend(message2, 0, message2.Length, SocketFlags.None,new
AsyncCallback(SendData), client);
}
```

VB.NET:

```
Sub ReceiveData(ByVal iar As IAsyncResult)
    Dim client As Socket = CType(iar.AsyncState, Socket)
    Dim recv As Integer = client.EndReceive(iar)
    If recv = 0 Then
        client.Close()
        conStatus.Text = "Waiting for client..."
        server.BeginAccept(New AsyncCallback(AcceptConn), server)
        Return
    End If
    Dim receivedData As String = Encoding.ASCII.GetString(data, 0, recv)
    results.Items.Add(receivedData)
    Dim message2 As Byte() = Encoding.ASCII.GetBytes(receivedData)
    client.BeginSend(message2, 0, message2.Length, SocketFlags.None, New
AsyncCallback(SendData), client)
End Sub
```

وهنا قد تم الانتهاء من برنامج ال Server والآن سوف نقوم بإنشاء برنامج ال Client وللبدء قم بإنشاء مشروع جديد كما في الشكل التالي:



سوف نستخدم ال Namespaces التالية:

C#:
```
using System.Net;
using System.Net.Sockets;
using System.Text;
```

VB.NET:
```
imports System.Net
imports System.Net.Sockets
imports System.Text
```

في ال Global Declaration (أي بعد تعريف ال Main Class)قم بإضافة التعاريف التالية:

C#:
```
public class Form1 : System.Windows.Forms.Form
{
        private Socket client;
        private byte[] data = new byte[1024];
        private int size = 1024;
```

في ال Connect Button قم بكتابة الكود التالي:

C#:
```
conStatus.Text = "Connecting...";
Socket newsock = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);
IPEndPoint iep = new IPEndPoint(IPAddress.Parse(textBox1.Text), 5020);
newsock.BeginConnect(iep, new AsyncCallback(Connected), newsock);
```

VB.NET:
```
Private client As Socket
Private data As Byte() = New Byte(1024) {}
Private size As Integer = 1024


conStatus.Text = "Connecting..."
Dim newsock As Socket = New Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp)
Dim iep As IPEndPoint = New IPEndPoint(IPAddress.Parse(textBox1.Text),
5020)
newsock.BeginConnect(iep, New AsyncCallback(Connected), newsock)
```

ثم قم بإنشاء Callback Connect method كما يلي:

C#:
```
void Connected(IAsyncResult iar)
{
client = (Socket)iar.AsyncState;
        try
        {
client.EndConnect(iar);
conStatus.Text = "Connected to: " + client.RemoteEndPoint.ToString();
```

```
client.BeginReceive(data, 0, size, SocketFlags.None, new
AsyncCallback(ReceiveData), client);
        }
catch (SocketException)
                {
        conStatus.Text = "Error connecting";
                }
}
```

VB.NET:

```
Sub Connected(ByVal iar As IAsyncResult)
    client = CType(iar.AsyncState, Socket)
    Try
        client.EndConnect(iar)
        conStatus.Text = "Connected to: " + client.RemoteEndPoint.ToString
        client.BeginReceive(data, 0, size, SocketFlags.None, New
AsyncCallback(ReceiveData), client)
    Catch generatedExceptionVariable0 As SocketException
        conStatus.Text = "Error connecting"
    End Try
End Sub
```

ثم إنشاء Receive Callback method لإنهاء ال BeginReceive وكما يلي:

C#:

```
void ReceiveData(IAsyncResult iar)
{
Socket remote = (Socket)iar.AsyncState;
int recv = remote.EndReceive(iar);
string stringData = Encoding.ASCII.GetString(data, 0, recv);
results.Items.Add(stringData);
}
```

VB.NET:

```
Sub ReceiveData(ByVal iar As IAsyncResult)
    Dim remote As Socket = CType(iar.AsyncState, Socket)
    Dim recv As Integer = remote.EndReceive(iar)
    Dim stringData As String = Encoding.ASCII.GetString(data, 0, recv)
    results.Items.Add(stringData)
End Sub
```

ثم إضافة الكود التالي في ال Send Button :

C#:

```
try
{
byte[] message = Encoding.ASCII.GetBytes(newText.Text);
newText.Clear();
client.BeginSend(message, 0, message.Length, SocketFlags.None,new
AsyncCallback(SendData), client);
newText.Focus();
}
catch(Exception ex){MessageBox.Show(ex.Message);}
```

VB.NET:

```
Try
Dim message As Byte() = Encoding.ASCII.GetBytes(newText.Text)
 newText.Clear
 client.BeginSend(message, 0, message.Length, SocketFlags.None, New
AsyncCallback(SendData), client)
 newText.Focus
Catch ex As Exception
Msgbox(ex.Message)
End Try
```

ثم إنشاء Send Callback method لإنهاء ال BeginSend وكما يلي:

C#:

```
void SendData(IAsyncResult iar)
{
        try
        {
        Socket remote = (Socket)iar.AsyncState;
        int sent = remote.EndSend(iar);
remote.BeginReceive(data, 0, size, SocketFlags.None, new
AsyncCallback(ReceiveData), remote);
                        }
catch(Exception ex){MessageBox.Show(ex.Message);}

}
```

VB.NET:

```
Sub SendData(ByVal iar As IAsyncResult)
   Try
      Dim remote As Socket = CType(iar.AsyncState, Socket)
      Dim sent As Integer = remote.EndSend(iar)
      remote.BeginReceive(data, 0, size, SocketFlags.None, New
AsyncCallback(ReceiveData), remote)
   Catch ex As Exception
Msgbox(ex.Message)
   End Try
End Sub
```

ثم إنشاء Receive Callback method لإنهاء ال BeginReceive وكما يلي:

C#:

```
void ReceiveData(IAsyncResult iar)
{
        try
        {
Socket remote = (Socket)iar.AsyncState;
int recv = remote.EndReceive(iar);
string stringData = Encoding.ASCII.GetString(data, 0, recv);
results.Items.Add(stringData);
        }
        catch(Exception ex){MessageBox.Show(ex.Message);}
}
```

VB.NET:

```
Sub ReceiveData(ByVal iar As IAsyncResult)
   Try
      Dim remote As Socket = CType(iar.AsyncState, Socket)
      Dim recv As Integer = remote.EndReceive(iar)
      Dim stringData As String = Encoding.ASCII.GetString(data, 0, recv)
      results.Items.Add(stringData)
   Catch ex As Exception
Msgbox(ex.Message)
   End Try
End Sub
```

وكما لاحظنا فإن برنامج ال Client لا يختلف كثيرا عن برنامج ال Server حيث أما نعرف في ال Server ال Socket Connection وال BeginAccept Method في ال Client فنعرف ال Socket Connection و ال BeginConnect Method وتبقى عملية الإرسال والاستقبال هي نفسها في Server وال Client ...

Chapter 4
Advanced Multicasting Systems

Advanced Multicasting Systems

A. Architecture of Multicast Sockets
B. Using Multicast Sockets with .NET
C. Multicast Conferencing Systems:
    1. Full/Half Duplex Multicast Video Conferencing System.
    2. Full/Half Duplex Multicast Desktop Conferencing System.
    3. Full/Half Duplex Multicast Text Conferencing System

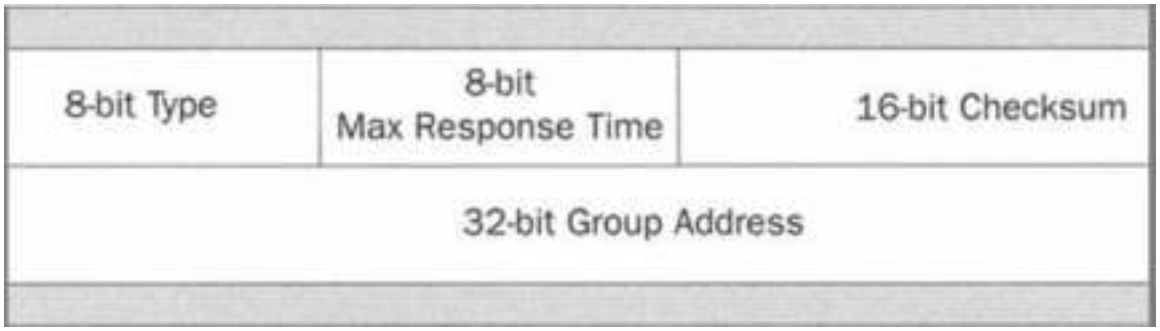
بسم الله الرحمن الرحيم

4.1: Advanced Multicasting Systems :

قمنا سابقا بتعريف ال Multicasting وبينا الفرق بينها وبين ال Broadcasting وبينا أنواعها وكيفية التعامل معها في الدوت نيت وفي هذه الجزء سوف نتحدث عنها بشكل أكثر تفصيلا وذلك لأهميتها الكبيرة في برمجيات الشبكات وخاصة برمجيات ال Conferencing...

أولا : Architecture of Multicast Sockets :

من المعروف انه يتم التعامل مع ال Multicasting عبر برتوكول ال UDP وباستخدام ال Class D Subnet Mask وتتم عملية إدارة المجموعات باستخدام برتوكول ال – IGMP Internet Group Management Protocol والذي هو جزء من ال Internet Protocol Model وكما يتضح من الشكل التالي فإن برتوكول ال IGMP يحتوي على عمليات التحقق من الوصول السليم للبيانات (حيث يتم إرسال حجم البيانات الكلي لرسالة وهي اختيارية إذ يمكن إلغائها بوضع الرقم صفر) ، و تحتوي أيضا على ال TTL Time to Live والذي يحدد فيه العمر الافتراضي لكل رسالة، ونوع العملية الإدارية (ضم إلى مجموعة ، إلغاء من مجموعة ، أو إرجاع معلومات عن المجموعة Membership Query) وأخيرا عنوان المجموعة التي يتم تحديدها برمجيا ضمن ال Range المحدد لل Class D .



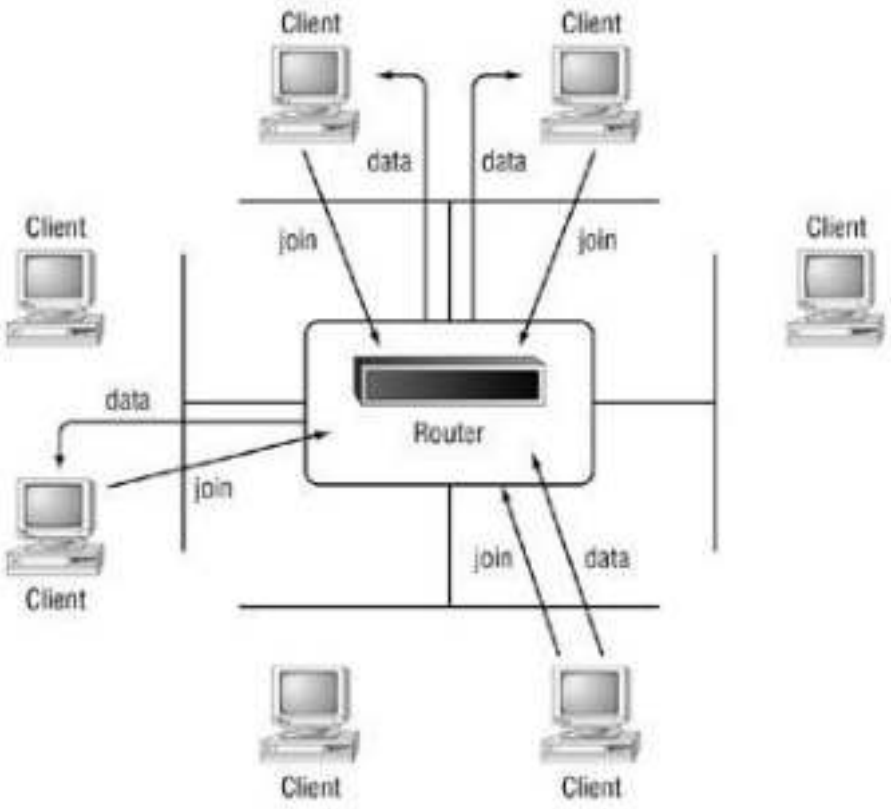
وتم تخصيص ال Range في ال Multicasting من 224.0.0.0 إلى 239.255.255.255 ونستطيع تحديده بثلاثة طرق فإما بشكل يدوي Static أو Dynamic أو على أساس ال Scope-Relative وبشكل عام تستخدم هذه التوزيعات كما يلي كمثال:
التخصيص 224.0.0.1 ويستخدم في جميع الشبكات المحلية فقط حيث لا يتم تمريره إلى شبكة أخرى عبر ال Router أما إذا أردنا التمرير إلى شبكات أخرى عبر ال Router فنستخدم التخصيص 224.0.0.2و لاكن بشرط استخدام نفس ال Subnet في الشبكات الأخرى ... ولمعرفة جميع التخصيصات لل Multicasting انظر الرابط التالي:

http://www.iana.org/assignments/multicast-addresses

يتم نقل ال Multicast Packets بين ال Backbone Tunnels باستخدام ال Unicast Tunnel حيث يتم إرسالها من داخل الشبكة إلى ال Router و ترسل من Router إلى أخر عبر ال Backbone Tunnel باستخدام أسلوب ال Unicast وهو ما يوفر الكثير من ال Bandwidth في الشبكة حيث ترسل نسخة واحدة إلى ال Router ويقوم هو بتوزيعها على الأجهزة باستخدام ال Unicast المشكلة الوحيدة في ال Multicast هو انه يعتمد بشكل كامل على استخدام ال UDP Connectionless Protocol.

ويمكننا استخدام ال Multicasting في ثلاثة أنواع من الشبكات وهي شبكات ال Peer to Peer حيث لا وجود لجهاز Server والكل يستقبل و يرسل من و إلى ال Group الذي

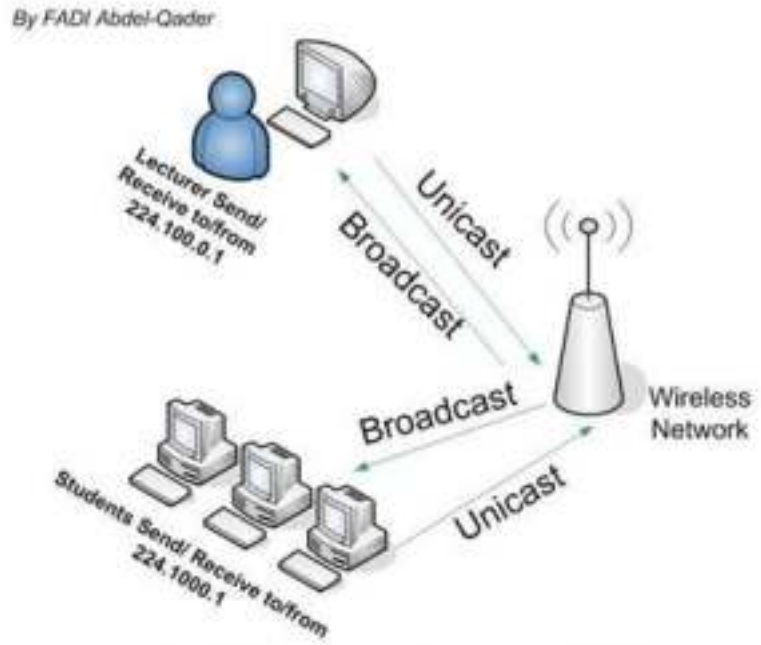
هو فيه، والنوع الثاني Server Based Network حيث يتم إرسال رسالة واحدة إلى ال Server ويقوم ال Server بتوزيعها على بقية الأجهزة في الشبكة ، أما النوع الثالث فيتم من خلال ال Router ، وكما يتضح من الشكل التالي فإن عملية الإرسال تتم بعد انضمام ال Client إلى المجموعة التي تملك ال IP Multicast ويرسل ال Client رسالة واحدة إلى ال Router حيث يقوم ال Router بتوزيعها على الأجهزة في المجموعة مستخدما ال Routing Table.



وكما كان الحال في الإرسال باستخدام ال Broadcasting يتم الإرسال في Multicasting من جهاز محدد إلى مجموعة معينة وليس إلى الكل كما في ال Broadcast ، حيث تُكون كل مجموعة من الأجهزة Group خاص ويتم التخصيص كما ذكرنا سابقا وفق ال IP Multicasting حيث تمتلك كل مجموعة نفس ال IP Multicast ويوجد عدة أشكال لل Multicasting ومن الأمثلة عليها الإرسال إلى مجموعة one to Group و الإرسال إلى أكثر من مجموعة one to Multi Group :

1 – الإرسال مجموعة One to Group:

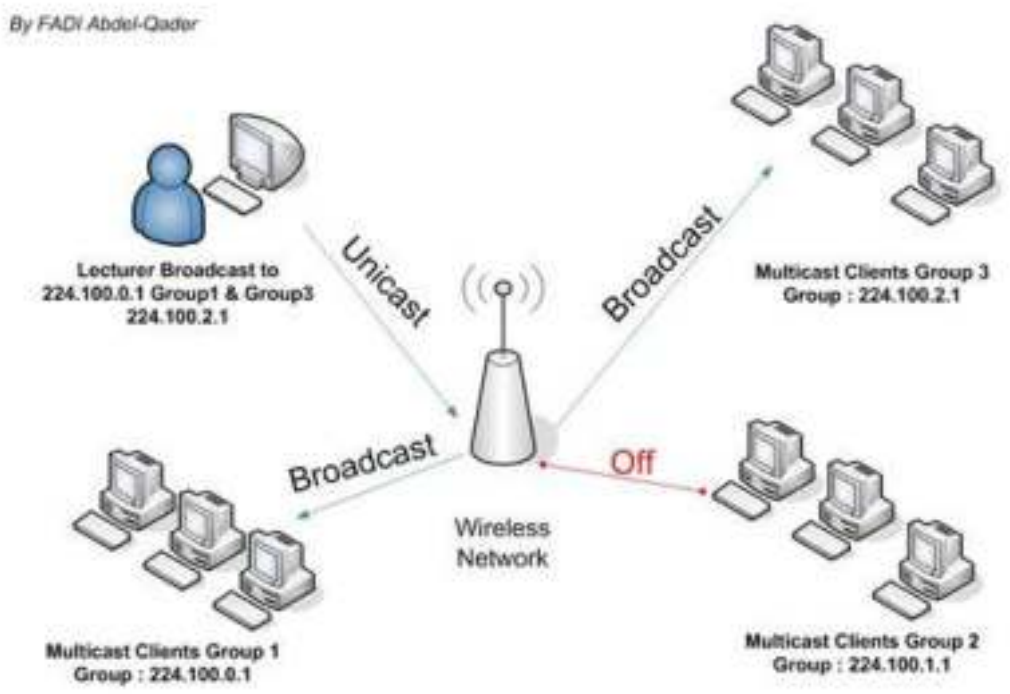
وفيه يملك ال Sender User نفس ال IP Multicasting الذي يملكه ال Receiver Users ويتم الإرسال من داخل ال Group إلى جميع أعضائه حيث ترسل ك Unicast إلى ال Access Point حيث يقوم بتوزيعها على كافة الأعضاء في المجموعة بأسلوب ال Broadcast وكما في الشكل التالي:



Full Duplex Multicast System for one Group

2- الإرسال إلى أكثر من مجموعة One to Multi-Groups:

وفيه قد يكون ال IP Multicasting للـ Sender User مختلف عن Receiver Users ويتم الإرسال من User داخل ال Group إلى المجموعة الذي هو عضو منها وإلى مجموعات أخرى ، ويتم تحديدها باستخدام Address List للمجموعات التي نريد الإرسال لها ...



Half Duplex Multicast System for Multi-Groups

ثانيا :Using Multicast Sockets with .NET:

شرحنا سابقا كيفية التعامل مع ال Multicasting في الدوت نيت وتعرفنا على ال Members وال Classes الخاصة بها وهنا سوف نبين بشيء من التفصيل هذه العمليات ونطبق عليها مجموعة من الأمثلة وبعد ذلك سنقوم ببناء نظام Conference System معتمدا على ال Multicasting ...

من العمليات الأساسية في التعامل مع ال Multicasting :

1- الانضمام أو الخروج من مجموعة Drop Group || Joining :

لا تلزم عملية الانضمام إلى ال Multicast Group أي عمليات تحقق سوى التصنت على ال port وال IP Multicasting المحدد ، ويتم ذلك بعد تعريف udpClient Object وباستخدام ال JoinMulticastGroup Method يتم تعريف ال IP Multicasting الذي سوف ننضم إليه وكما يلي:

C#:
```
UdpClient sock = new UdpClient(5020);
sock.JoinMulticastGroup(IPAddress.Parse("225.100.0.1"), 50);
IPEndPoint iep = new IPEndPoint(IPAddress.Any, 0);
```

VB.NET:
```
Dim sock As UdpClient = New UdpClient(5020)
sock.JoinMulticastGroup(IPAddress.Parse("225.100.0.1"), 50)
Dim iep As IPEndPoint = New IPEndPoint(IPAddress.Any, 0)
```

وكما يلي لإلغاء عملية الانضمام من مجموعة:

C#:
```
sock.DropMulticastGroup(IPAddress.Parse("225.100.0.1"));
```

VB.NET:
```
sock.DropMulticastGroup(IPAddress.Parse("225.100.0.1"))
```

إذ تستخدم ال Methods **JoinMulticastGroup و DropMulticastGroup** لضم أو إلغاء عنوان أو مجموعة من العناوين من ال Multicast Group ، وباستخدام Class MulticastOption: يمكننا تخزين IP Address List لتعامل معها في Multicast Group لعمل Join و Drop لأي Multicast Group وتستخدم كما يلي كمثال لإضافة عضوية لاستقبال رسائل Multicast :

أولا نعرف ال UDP Socket وكما يلي :

C#:
```
mcastSocket = new Socket(AddressFamily.InterNetwork,SocketType.Dgram,
ProtocolType.Udp);
```

VB.NET:
```
mcastSocket = New Socket(AddressFamily.InterNetwork, SocketType.Dgram,
ProtocolType.Udp)
```

ثانيا نقوم بتعريف Address List ثم نسند إليها ال IP الذي نريد إدخاله في ال Group أو نجعل ال User يدخل العنوان بنفسه نربطها بالسكوت باستخدام الميثود Bind وكما يلي :

C#:

```
IPAddress localIPAddr = IPAddress.Parse(Console.ReadLine());
mcastSocket.Bind(IPlocal);
```

VB.NET:

```
Dim localIPAddr As IPAddress = IPAddress.Parse(Console.ReadLine)
mcastSocket.Bind(IPlocal)
```

ثالثا نقوم بتعريف ال Multicast Option ونسند لها العنوان المحدد كما يلي:

C#:

```
MulticastOption mcastOption;
mcastOption = new MulticastOption(localIPAddr);
```

VB.NET:

```
Dim mcastOption As MulticastOption
MulticastOption(localIPAddr New = mcastOption)
```

ومن ثم نضيف التغير على SetSocketOption حيث تأخذ هذه الميثود ثلاثة بارومترات الأول لتحديد مستوى التغيير على IP أو على IPv6 أو على Socket أو TCP أو UDP وفي حالتنا هذه سوف نستخدم التغير على IP إذ ما نريده هو ضم IP إلى Multicast Group وفي البارومتر الثاني نحدد نوع التغيير حيث نريد إضافة عضوية ويمكن الاختيار بين إضافة عضويه AddMembership أو إلغاء عضوية DropMembership وأخيرا نسند إليه ال MulticastOption Object والذي قمنا بإنشائه و كما يلي:

C#:

```
mcastSocket.SetSocketOption(SocketOptionLevel.IP,
SocketOptionName.AddMembership,mcastOption);
```

VB.NET:

```
Dim mcastOption As MulticastOption
mcastOption = New MulticastOption(localIPAddr)
```

2- الإرسال إلى مجموعة Sending Data to a Multicast Group:

حتى نستطيع الإرسال باستخدام ال IP Multicasting لابد أولا من تعريف ال Socket باستخدام ال UDP Connection وإسناد ال IP Multicasting ورقم ال Port إلى ال IPEndPoint Object ... ونستطيع الإرسال باستخدام ال sendto method حيث نسند لها ال data as Bytes Array وال IPEndPoint Object وكما يلي لإرسال رسالة نصية:

C#:

```
Socket server = new Socket(AddressFamily.InterNetwork,SocketType.Dgram,
ProtocolType.Udp);
IPEndPoint iep = new IPEndPoint(IPAddress.Parse((“225.100.0.1”), 5020);
byte[] data = Encoding.ASCII.GetBytes(msg.Text);
server.SendTo(data, iep);
server.Close();
msg.Clear();
msg.Focus();
```

VB.NET:

```
Dim server As Socket = New Socket(AddressFamily.InterNetwork,
SocketType.Dgram, ProtocolType.Udp)
Dim iep As IPEndPoint = New IPEndPoint(IPAddress.Parse("225.100.0.1"),
5020)
Dim data As Byte() = Encoding.ASCII.GetBytes(msg.Text)
server.SendTo(data, iep)
server.Close
msg.Clear
msg.Focus
```

ولإرسال Binary Data كإرسال صورة مثلا لابد من استخدام ال Memory Stream لتخزين الصورة في الذاكرة على هيئة Stream ثم تحويلها إلى Byte Array وبعد ذلك إرسالها باستخدام ال sendto Method وكما يلي:

C#:

```
MemoryStream ms = new MemoryStream();
PictureBox1.Image.Save(ms,System.Drawing.Imaging.ImageFormat.Jpeg);
byte[] arrImage = ms.GetBuffer();
ms.Close();
Socket server = new Socket(AddressFamily.InterNetwork,SocketType.Dgram,
ProtocolType.Udp);
IPEndPoint iep = new IPEndPoint(IPAddress.Parse("225.100.0.1"), 5020);
server.SendTo(arrImage,iep);
```

VB.NET:

```
Dim ms As MemoryStream = New MemoryStream
PictureBox1.Image.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg)
Dim arrImage As Byte() = ms.GetBuffer
ms.Close
Dim server As Socket = New Socket(AddressFamily.InterNetwork,
SocketType.Dgram, ProtocolType.Udp)
Dim iep As IPEndPoint = New IPEndPoint(IPAddress.Parse("225.100.0.1"),
5020)
server.SendTo(arrImage, iep)
```

3- الاستقبال من مجموعة Receiving Data From a Multicast Group:

حتى نستطيع الاستقبال من مجموعة لابد أولا من تحديد ال IP Multicast الخاص بالمجموعة و الانضمام إليه ثم استقبال البيانات باستخدام ال Receive Method ويتم ذلك كما يلي لاستقبال رسالة نصية وعرضها في list Box:

C#:

```
UdpClient sock = new UdpClient(5020);
sock.JoinMulticastGroup(IPAddress.Parse("225.100.0.1"), 50);
IPEndPoint iep = new IPEndPoint(IPAddress.Any, 0);

byte[] data = sock.Receive(ref iep);
string stringData = Encoding.ASCII.GetString(data, 0, data.Length);
listBox1.Items.Add(iep.Address.ToString() +" :_  "+stringData );
```

VB.NET:

```
Dim sock As UdpClient = New UdpClient(5020)
```

```
sock.JoinMulticastGroup(IPAddress.Parse("225.100.0.1"), 50)
Dim iep As IPEndPoint = New IPEndPoint(IPAddress.Any, 0)
Dim data As Byte() = sock.Receive(iep)
Dim stringData As String = Encoding.ASCII.GetString(data, 0, data.Length)
listBox1.Items.Add(iep.Address.ToString + " :_ " + stringData)
```

ولاستقبال صورة نستخدم ال memory Stream لاستقبال البيانات من ال Receive Method وتخزينها في الذاكرة على هيئة Stream Data ثم تحويلها إلى صورة مرة أخرى باستخدام ال image.FromStream Method وكما يلي:

C#:

```
UdpClient sock = new UdpClient(5020);
sock.JoinMulticastGroup(IPAddress.Parse("225.100.0.1"));
IPEndPoint iep = new IPEndPoint(IPAddress.Any, 0);

byte[] data = sock.Receive(ref iep);
MemoryStream ms = new MemoryStream(data);
pictureBox1.Image = Image.FromStream(ms);
sock.Close();
```

VB.NET:

```
Dim sock As UdpClient = New UdpClient(5020)
sock.JoinMulticastGroup(IPAddress.Parse("225.100.0.1"), 50)
Dim iep As IPEndPoint = New IPEndPoint(IPAddress.Any, 0)
Dim data As Byte() = sock.Receive(iep)
Dim stringData As String = Encoding.ASCII.GetString(data, 0, data.Length)
listBox1.Items.Add(iep.Address.ToString + " :_ " + stringData)
```

ملاحظات هامة في استخدام ال Multicasting في برمجيات الشبكات :

1- من الملاحظ أننا لا نستطيع استخدام ال Network Stream لعملية إرسال ال Multicasting إذ يتطلب استخدامها وجود TCP Socket Connection وهو غير متاح في ال Multicasting ويستعاض عنها باستخدام ال memory Stream لإرسال Binary Stream عبر ال sendto method ...

2- لا يمكنك استخدام ال Multicasting ك loopback في حالة عدم وجود شبكة أو اتصال لذلك لن تستطيع تجربة أي من تطبيقات ال Multicasting في حالة عدم اتصالك بالشبكة.

3- يمكن لكل جهاز أن ينضم إلى أكثر من مجموعة بحيث يستقبل من جهات متعددة، كذلك يستطيع الإرسال إلى عدة مجموعات.

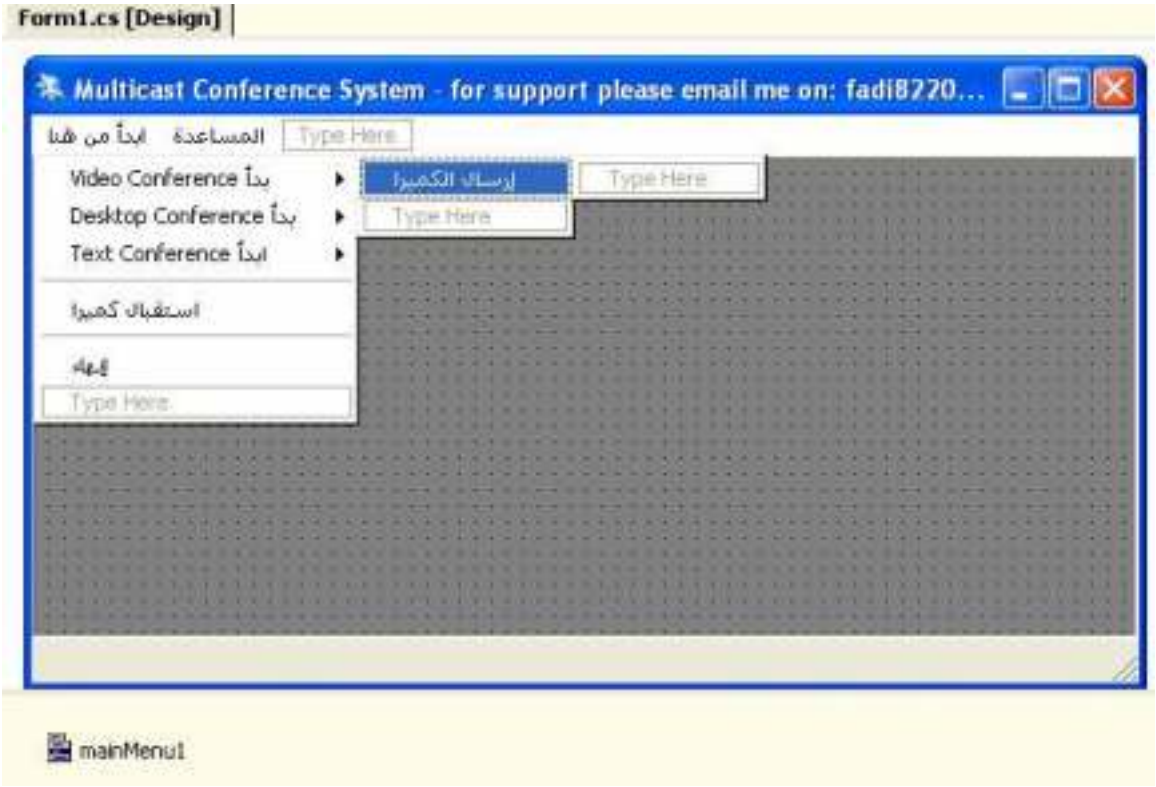
4- في العادة تكون السعة المسموحة لإرسال ال Multicasting Data عبر ال sendto Method محدودة لذلك يمكنك استخدام ال Binary Reader & Writer وال Stream Reader & Writer لإرسال والاستقبال بدلا منها ...

5- تتم عملية اختيار ال IP Multicast وفق لل Network Topology التي تملكها لذلك لابد من التقيد بالعناوين المحددة وهو ما بينته سابقا ..

ثالثا تطبيق مشروع نظام المؤتمرات Multicasting Conferencing Systems:

في هذا التطبيق سوف نفترض وجود غرفة صفية حيث يقوم المحاضر بإلقاء المحاضرة عن بعد أمام طلابه إذ نريد هنا جعل الطلاب يرون الأستاذ وكما يستطيع الأستاذ رؤية طلابه بالإضافة إلى إمكانية عرض المحاضرة على ال Power Point Slides كما يستطيع الطلاب التحدث مع الأستاذ باستخدام Text Chatting ...

سوف نقوم هنا بتقسيم نظام المؤتمرات إلى ثلاثة أنظمة رئيسية وهي نظام مؤتمرات الفيديو ونظام مؤتمرات سطح المكتب ونظام المؤتمرات النصية، في البداية سوف نقوم بعمل الشاشة الرئيسية للبرنامج و كما في الشكل التالي:



1- Full/Half Duplex Multicast Video Conferencing System :

وفرت لنا Microsoft مجموعة من ال Classes الخارجية والتي تتعامل مع ال DirectX 9 مباشرة حيث نستطيع استخدامها لتعامل مع الكاميرا أو ال Scanner أو الصوت أو أي طرفية أخرى وفي هذا التطبيق سوف نستخدم الDirect Show Dot Net Classes لالتقاط صورة عبر الكاميرا وعرضها على ال Picture box حيث نستطيع إرسالها لاحقا إلى ال Multicast Group باستخدام ال memory Stream وال Sendto method وهو ما بيناه سابقا ..

وحتى نستطيع استخدامها سوف نضم ال Direct Show Classes إلى المشروع وكما يلي:

وحتى نتعامل معها سوف نستدعيها باستخدام :

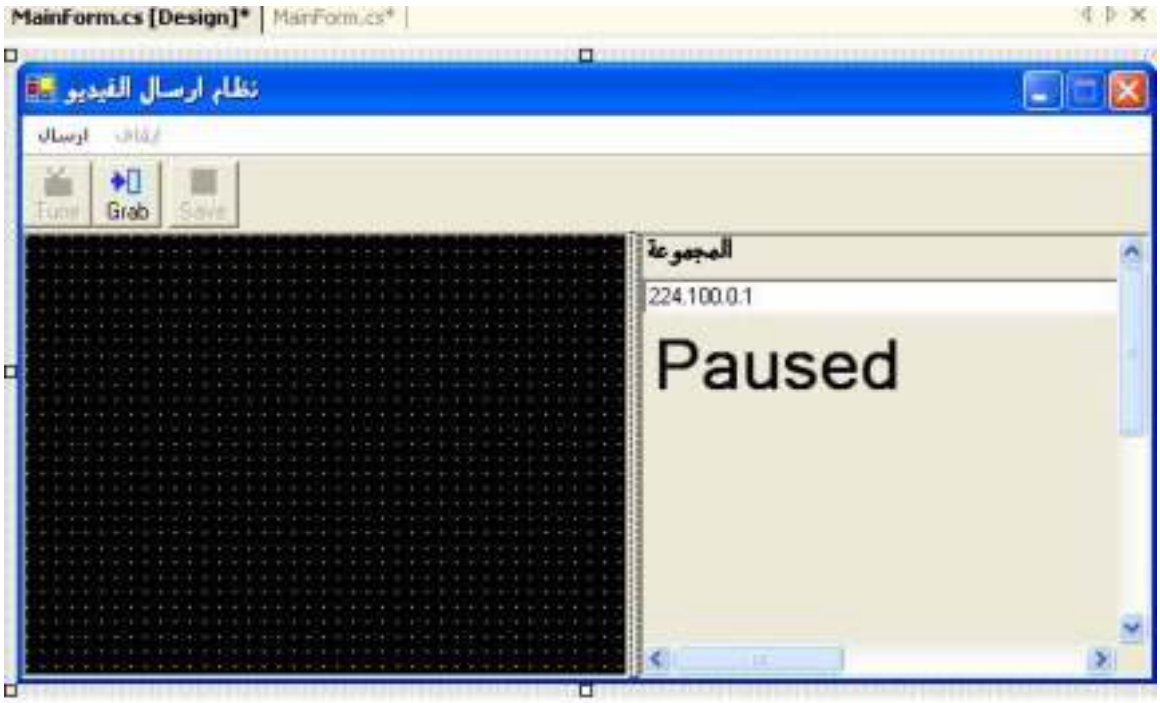
C#:

```
using DShowNET;
using DShowNET.Device;
```

VB.NET:

```
imports DShowNET
imports DShowNET.Device
```

وسيكون شكل برنامج الإرسال عبر الكاميرا كما في الشكل التالي:



سوف نستخدم ال DeviceSelector Class لإختيار جهاز الإدخال عند بداية تشغيل البرنامج وكما يلي:

C#:

```
DeviceSelector selector = new DeviceSelector( capDevices );
selector.ShowDialog( this );
```

```
dev = selector.SelectedDevice;
```

VB.NET:

```
Dim selector As DeviceSelector = New DeviceSelector(capDevices)
selector.ShowDialog(Me)
dev = selector.SelectedDevice
```

و لإلتاط الصورة عبر الكاميرا سوف نقوم بإنشاء method جديدة كما يلي :

C#:

```
void OnCaptureDone()
{
        try {
Trace.WriteLine( "!!DLG: OnCaptureDone" );
toolBarBtnGrab.Enabled = true;
int hr;
if( sampGrabber == null )return;
hr = sampGrabber.SetCallback( null, 0 );
int w = videoInfoHeader.BmiHeader.Width;
int h = videoInfoHeader.BmiHeader.Height;
if( ((w & 0x03) != 0) || (w < 32) || (w > 4096) || (h < 32) || (h > 4096) )
return;
int stride = w * 3;
GCHandle handle = GCHandle.Alloc( savedArray, GCHandleType.Pinned );
int scan0 = (int) handle.AddrOfPinnedObject();
scan0 += (h - 1) * stride;
Bitmap b = new Bitmap( w, h, -stride, PixelFormat.Format24bppRgb, (IntPtr)
scan0 );
handle.Free();
savedArray = null;
Image old = pictureBox.Image;
pictureBox.Image = b;
if( old != null ) old.Dispose();
toolBarBtnSave.Enabled = true;}
        catch( Exception){}
}
```

VB.NET:

```
Private Sub OnCaptureDone()
Try
Trace.WriteLine("!!DLG: OnCaptureDone("
toolBarBtnGrab.Enabled = True
Dim hr As Integer
If sampGrabber Is Nothing Then
Return
End If
hr = sampGrabber.SetCallback)Nothing(0 ،
Dim w As Integer = videoInfoHeader.BmiHeader.Width
Dim h As Integer = videoInfoHeader.BmiHeader.Height
If)) w And& H03) <> 0 (OrElse) w < 32 (OrElse) w > 4096 (OrElse) h < 32 (
OrElse) h > 4096 (Then
Return
```

```
End If
Dim stride As Integer = w * 3
Dim handle As GCHandle = GCHandle.Alloc(savedArray,
GCHandleType.Pinned(
Dim scan0 As Integer = CInt)handle.AddrOfPinnedObject(()
scan0 += (h - 1) * stride
Dim b As Bitmap = New Bitmap(w, h, -stride, PixelFormat.Format24bppRgb ،
New IntPtr(scan0((
handle.Free()
savedArray = Nothing
Dim old As Image = pictureBox.Image
pictureBox.Image = b
If Not old Is Nothing Then
old.Dispose()
End If
toolBarBtnSave.Enabled = True
Catch e1 As Exception
End Try
End Sub
```

ثم عمل Timer وإضافة الكود التالي فيه لاستمرار عملية التقاط الصورة:

C#:

```
int hr;
int size = videoInfoHeader.BmiHeader.ImageSize;
savedArray = new byte[ size + 64000 ];
```

VB.NET:

```
Dim hr As Integer
Dim size As Integer = videoInfoHeader.BmiHeader.ImageSize
savedArray = New Byte(size + 64000) {}
```

ولإرسال الصورة إلى الطرف الأخر سوف نستخدم method إرسال الصورة ونضعه في Timer وكما يلي:

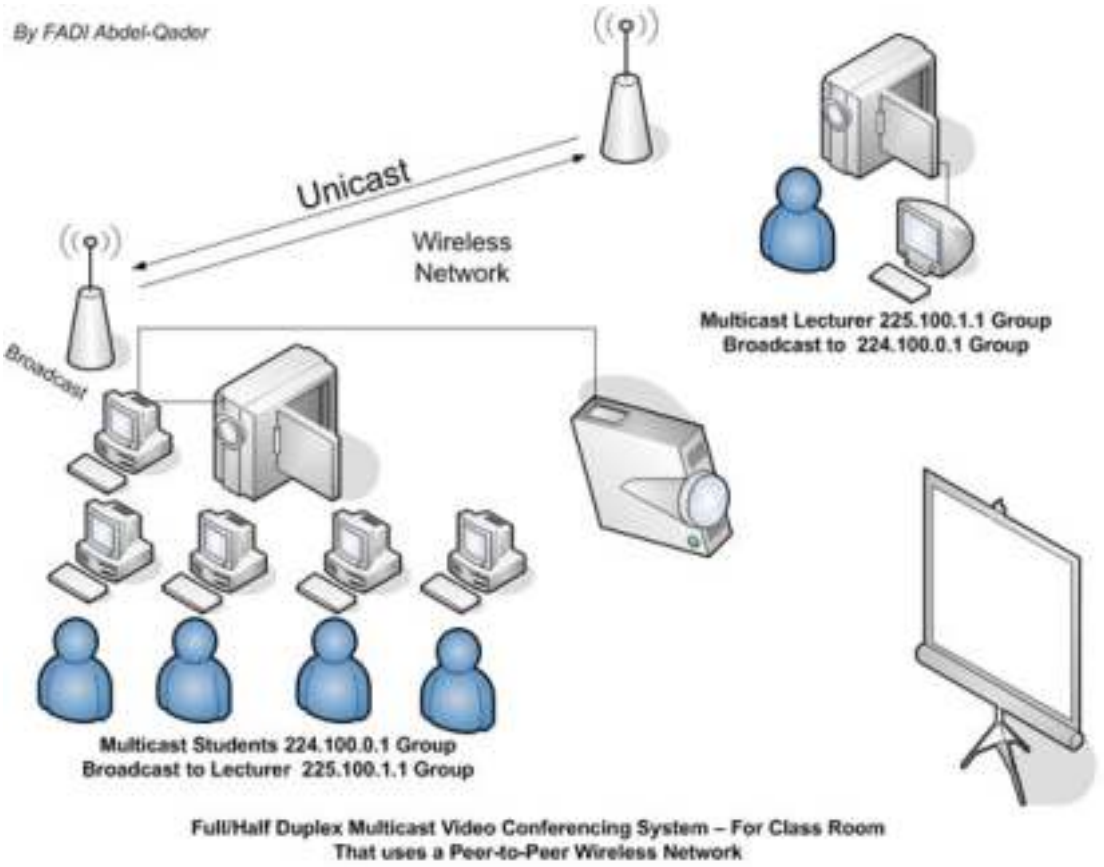
C#:

```
try
{
MemoryStream ms = new MemoryStream();
pictureBox.Image.Save(ms,System.Drawing.Imaging.ImageFormat.Jpeg);
byte[] arrImage = ms.GetBuffer();
ms.Close();
Socket server = new Socket(AddressFamily.InterNetwork,SocketType.Dgram,
ProtocolType.Udp);
IPEndPoint iep = new IPEndPoint(IPAddress.Parse(textBox1.Text), 5020);
server.SendTo(arrImage, iep);
server.Close();}
catch (Exception){}
```

VB.NET:

```
Try
Dim ms As MemoryStream = New MemoryStream
 pictureBox.Image.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg)
Dim arrImage As Byte() = ms.GetBuffer
 ms.Close
Dim server As Socket = New Socket(AddressFamily.InterNetwork,
SocketType.Dgram, ProtocolType.Udp)
Dim iep As IPEndPoint = New IPEndPoint(IPAddress.Parse(textBox1.Text),
5020)
 server.SendTo(arrImage, iep)
 server.Close
Catch generatedExceptionVariable0 As Exception
End Try
```

وهنا يستطيع المحاضر إرسال الصورة عبر الكاميرا إلى طلابه كما سوف يتمكن من رؤية طلابه عبر الكاميرا وسوف نفترض هنا استخدامه لشبكة لا سلكية حيث سيرسل البيانات إلى ال Access Point بأسلوب ال Unicast وسوف يتولا ال Access Point توزيع البيانات إلى جميع الأعضاء المنضمين إلى ال Multicast Group ويرسلها لهم باستخدام ال Broadcast وكما في الشكل التالي:



وكما نلاحظ في الشكل السابق فإن المحاضر ينضم إلى مجموعتين مجموعة الأساتذة وهي 225.100.1.1 حيث سيستقبل صورة طلابه عليها، ومجموعة الطلاب

224.100.0.1 والتي سوف يرسل الصورة إليها .. وكما نلاحظ ايضا فإن عملية الإرسال بين ال Access Point1 وال Access Point2 تتم باستخدام ال Unicast ...
وحتى يستطيع الطلاب رؤية أستاذهم والأستاذ رؤية طلابه ، لابد من إنشاء برنامج الاستقبال حيث سنستخدم نفس ال method التي شرحنها سابقا لاستقبال الصورة وللبدء قم بعمل New Form جديد كما في الشكل التالي:

سوف نستخدم ال Namespaces التالية لاستقبال الصورة من ال Multicast Group :

C#:

```
using System.Net.Sockets ;
using System.Net;
using System.IO;
using System.Threading;
```

VB.NET:

```
imports System.Net.Sockets
imports System.Net
imports System.IO
imports System.Threading
```

ثم قم بكتابة method الاستقبال كما يلي:

```
void Image_Receiver()
{
        UdpClient sock = new UdpClient(5020);
        sock.JoinMulticastGroup(IPAddress.Parse(textBox1.Text));
        IPEndPoint iep = new IPEndPoint(IPAddress.Any, 0);
```

```
        byte[] data = sock.Receive(ref iep);
        MemoryStream ms = new MemoryStream(data);
        pictureBox1.Image = Image.FromStream(ms);
        sock.Close();
}
```

VB.NET:

```
Sub Image_Receiver()
    Dim sock As UdpClient = New UdpClient(5020)
    sock.JoinMulticastGroup(IPAddress.Parse(textBox1.Text))
    Dim iep As IPEndPoint = New IPEndPoint(IPAddress.Any, 0)
    Dim data As Byte() = sock.Receive(iep)
    Dim ms As MemoryStream = New MemoryStream(data)
    pictureBox1.Image = Image.FromStream(ms)
    sock.Close()
End Sub
```

وحتى نستدعيها لابد من استخدام ال Threading حتى لا يتأثر نظام التشغيل بعملية الاستقبال ، وحتى نقوم بذلك قم بعمل Timer وضع فيه الكود التالي لاستخدام ال Threading :

C#:

```
Thread myth;
myth= new Thread  (new System.Threading .ThreadStart(Image_Receiver));
myth.Start ();
```

VB.NET:

```
Dim myth As Thread
myth = New Thread(New System.Threading.ThreadStart(Image_Receiver))
myth.Start
```

وحتى تتمكن من تخزين الصورة الملتقطة عبر الكاميرا على هيئة JPEG Image File قم بإنشاء saveFileDialog واستدعيه كما يلي:

C#:

```
try
{
        saveFileDialog1.Filter = "JPEG Image (*.jpg)|*.jpg" ;
        if(saveFileDialog1.ShowDialog() == DialogResult.OK)
        {

        string mypic_path = saveFileDialog1.FileName;
        pictureBox1.Image.Save(mypic_path);
        }
}
catch (Exception){}
```

VB.NET:

```
Try
 saveFileDialog1.Filter = "JPEG Image (*.jpg)|*.jpg"
 If saveFileDialog1.ShowDialog = DialogResult.OK Then
Dim mypic_path As String = saveFileDialog1.FileName
   pictureBox1.Image.Save(mypic_path)
```

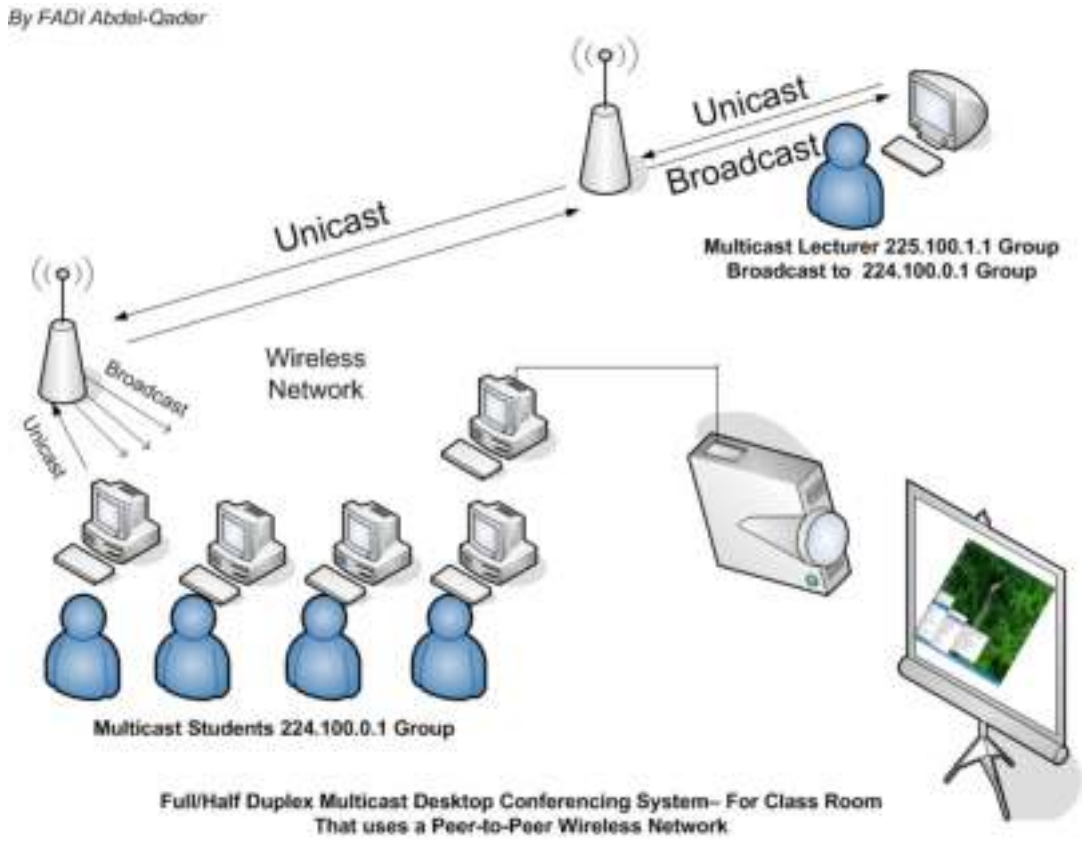
```
 End If
Catch generatedExceptionVariable0 As Exception
End Try
```

وهنا قد تم الانتهاء من المشروع الأول وهو ال Video Conference System ، وحتى يستطيع المحاضر عرض المحاضرة باستخدام برنامج ال Power Point سوف نقوم بعمل مشروع مؤتمرات سطح المكتب ...

:Full/Half Duplex Multicast Desktop Conferencing System -2

الهدف من هذا المشروع هو تمكين الأستاذ من عرض المحاضرة باستخدام برنامج ال Power Point حيث سترسل صورة سطح المكتب من جهاز الأستاذ إلى أجهزة الطلبة ، ولا تختلف عملية الإرسال عن البرنامج السابق في شيء سوى إنشاء Classes لتقوم بالتقاط صورة سطح المكتب ومن ثم إرسالها إلى ال Multicast Group ومن ثم استقبالها وعرضها على الطلاب باستخدام Data Show Projector ...

وهنا مخطط عمل البرنامج :



وكما نلاحظ من الشكل التالي فإن الأستاذ يقوم بشرح المحاضرة على جهازه الشخصي ويرسل الصورة إلى الطلاب وكما نلاحظ أيضا فإن هذه العملية هي أحادية الاتجاه وكما يمكن جعلها باتجاهين Full || Half Duplex لاكن لابد من إنشاء مجموعة جديدة لعملية الإرسال من الطالب إلى الأستاذ حيث يعرض الأستاذ محاضرته ويرسلها إلى مجموعة الطلاب ويستطيع أحد الطلاب عرض جهازه على الأستاذ إذ يرسل الصورة إلى مجموعة الأستاذ ...

ولإنشاء برنامج إرسال صورة سطح المكتب قم بعمل New Form جديد كما في الشكل التالي:

في البداية سوف نقوم بعمل Three Classes لالتقاط صورة سطح المكتب وكما يلي:

أولا PlatFormInvokeGDI32.cs لالتقاط صورة سطح المكتب باستخدام ال GDI+ وال API:

C#:

```
using System;
using System.Runtime.InteropServices;
namespace SampleGrabberNET
{

//This class shall keep the GDI32 APIs being used in our program.
public class PlatformInvokeGDI32
{

#region Class Variables
public  const int SRCCOPY = 13369376;
#endregion

#region Class Functions
[DllImport("gdi32.dll",EntryPoint="DeleteDC")]
public static extern IntPtr DeleteDC(IntPtr hDc);

[DllImport("gdi32.dll",EntryPoint="DeleteObject")]
public static extern IntPtr DeleteObject(IntPtr hDc);

[DllImport("gdi32.dll",EntryPoint="BitBlt")]
public static extern bool BitBlt(IntPtr hdcDest,int xDest,int
yDest,int wDest,int hDest,IntPtr hdcSource,int xSrc,int ySrc,int RasterOp);

[DllImport ("gdi32.dll",EntryPoint="CreateCompatibleBitmap")]
public static extern IntPtr CreateCompatibleBitmap(IntPtr hdc,
int nWidth, int nHeight);

[DllImport ("gdi32.dll",EntryPoint="CreateCompatibleDC")]
```

```
public static extern IntPtr CreateCompatibleDC(IntPtr hdc);

[DllImport ("gdi32.dll",EntryPoint="SelectObject")]
public static extern IntPtr SelectObject(IntPtr hdc,IntPtr bmp);
#endregion

#region Public Constructor
public PlatformInvokeGDI32()
{
}
#endregion
}}
```

VB.NET:

```
System Imports
System.Runtime.InteropServices Imports
SampleGrabberNET Namespace

.This class shall keep the GDI32 APIs being used in our program'
PlatformInvokeGDI32 Class Public

"Class Variables" Region#
13369376 = Integer As SRCCOPY Const Public
End Region#

"Class Functions" Region#
<("DllImport("gdi32.dll",EntryPoint:="DeleteDC>_
IntPtr As (IntPtr As hDc ByVal)DeleteDC Function Shared Public
Function End

<("DllImport("gdi32.dll",EntryPoint:="DeleteObject>_
IntPtr As (IntPtr As hDc ByVal)DeleteObject Function Shared Public
Function End

<("DllImport("gdi32.dll",EntryPoint:="BitBlt>_
As xDest ByVal ،IntPtr As hdcDest ByVal)BitBlt Function Shared Public
As hDest ByVal ،Integer As wDest ByVal ،Integer As yDest ByVal ،Integer
As ySrc ByVal ،Integer As xSrc ByVal ،IntPtr As hdcSource ByVal ،Integer
Boolean As (Integer As RasterOp ByVal ،Integer
Function End

<("DllImport ("gdi32.dll",EntryPoint:="CreateCompatibleBitmap>_
ByVal ،IntPtr As hdc ByVal)CreateCompatibleBitmap Function Shared Public
IntPtr As (Integer As nHeight ByVal ،Integer As nWidth
Function End

<("DllImport ("gdi32.dll",EntryPoint:="CreateCompatibleDC>_
IntPtr As (IntPtr As hdc ByVal)CreateCompatibleDC Function Shared Public
Function End
<("DllImport ("gdi32.dll",EntryPoint:="SelectObject>_
```

```
As bmp ByVal ،IntPtr As hdc ByVal)SelectObject Function Shared Public
IntPtr As (IntPtr
Function End
End Region#

"Public Constructor" Region#
Public Sub New()
Sub End
End Region#
Class End
Namespace End
```

ثانيا PlatformInvokeUSER32.cs إذ سوف نستخدمها مع ال Class السابق لالتقاط صورة سطح المكتب باستخدام ال user32 API :

C#:

```
using System;
using System.Runtime.InteropServices;
namespace SampleGrabberNET
{
// This class shall keep the User32 APIs being used in our program.
        public class PlatformInvokeUSER32
        {

                #region Class Variables
                public  const int SM_CXSCREEN=0;
                public  const int SM_CYSCREEN=1;
                #endregion

                #region Class Functions
                [DllImport("user32.dll", EntryPoint="GetDesktopWindow")]
                public static extern IntPtr GetDesktopWindow();

                [DllImport("user32.dll",EntryPoint="GetDC")]
                public static extern IntPtr GetDC(IntPtr ptr);

                [DllImport("user32.dll",EntryPoint="GetSystemMetrics")]
                public static extern int GetSystemMetrics(int abc);

                [DllImport("user32.dll",EntryPoint="GetWindowDC")]
                public static extern IntPtr GetWindowDC(Int32 ptr);

                [DllImport("user32.dll",EntryPoint="ReleaseDC")]
                public static extern IntPtr ReleaseDC(IntPtr hWnd,IntPtr hDc);

                #endregion

                #region Public Constructor
                public PlatformInvokeUSER32()
                {
                }
                #endregion
```

```
        }
        //This structure shall be used to keep the size of the screen.
        public struct SIZE
        {
                public int cx;
                public int cy;
        }
}
```

VB.NET:

```
System Imports
System.Runtime.InteropServices Imports
SampleGrabberNET Namespace
.This class shall keep the User32 APIs being used in our program '
PlatformInvokeUSER32 Class Public

"Class Variables" Region
0=Integer As SM_CXSCREEN Const Public
1=Integer As SM_CYSCREEN Const Public
End Region#

"Class Functions" Region#
<("DllImport("user32.dll", EntryPoint:="GetDesktopWindow>_
IntPtr As ()GetDesktopWindow Function Shared Public
Function End

<("DllImport("user32.dll",EntryPoint:="GetDC>_
IntPtr As (IntPtr As ptr ByVal)GetDC Function Shared Public
Function End

<("DllImport("user32.dll",EntryPoint:="GetSystemMetrics>_
Integer As (Integer As abc ByVal)GetSystemMetrics Function Shared Public
Function End

<("DllImport("user32.dll",EntryPoint:="GetWindowDC>_
IntPtr As (Int32 As ptr ByVal)GetWindowDC Function Shared Public
Function End

<("DllImport("user32.dll",EntryPoint:="ReleaseDC>_
(IntPtr As hDc ByVal ،IntPtr As hWnd ByVal)ReleaseDC Function Shared Public
IntPtr As
Function End

End Region#

"Public Constructor" Region#
Public Sub New()
Sub End
End Region#
Class End
.This structure shall be used to keep the size of the screen'
SIZE Structure Public
```

```
Integer As cx Public
Integer As cy Public
Structure End
Namespace End
```

ثالثا: CaptureScreen.cs والتي سوف نستخدمها بشكل مباشر في البرنامج حيث يتعامل مع ال PlatFormInvokeGDI32 Class وال PlatformInvokeUSER32 Class :

C#:

```
using System;
using System.Drawing;

namespace SampleGrabberNET
{
//This class shall keep all the functionality for capturing the desktop.
        public class CaptureScreen
        {
                #region Public Class Functions
                public static Bitmap GetDesktopImage()
                {
//In size variable we shall keep the size of the screen.
        SIZE size;

//Variable to keep the handle to bitmap.
IntPtr hBitmap;
//Here we get the handle to the desktop device context.
IntPtr hDC =
PlatformInvokeUSER32.GetDC(PlatformInvokeUSER32.GetDesktopWindow());
//Here we make a compatible device context in memory for screen device
context.
IntPtr hMemDC = PlatformInvokeGDI32.CreateCompatibleDC(hDC);
//We pass SM_CXSCREEN constant to GetSystemMetrics to get the X
coordinates of screen.
size.cx=PlatformInvokeUSER32.GetSystemMetrics(PlatformInvokeUSER32.SM
_CXSCREEN);
//We pass SM_CYSCREEN constant to GetSystemMetrics to get the Y
coordinates of screen.
size.cy=PlatformInvokeUSER32.GetSystemMetrics(PlatformInvokeUSER32.SM
_CYSCREEN);
//We create a compatible bitmap of screen size using screen device context.
hBitmap = PlatformInvokeGDI32.CreateCompatibleBitmap(hDC, size.cx,
size.cy);
//As hBitmap is IntPtr we can not check it against null. For this purspose
IntPtr.Zero is used.
if (hBitmap!=IntPtr.Zero)
        {
//Here we select the compatible bitmap in memory device context and keeps
the reference to Old bitmap.
IntPtr hOld = (IntPtr) PlatformInvokeGDI32.SelectObject(hMemDC, hBitmap);
//We copy the Bitmap to the memory device context.
```

```
PlatformInvokeGDI32.BitBlt(hMemDC, 0, 0,size.cx,size.cy, hDC, 0, 0,
PlatformInvokeGDI32.SRCCOPY);
//We select the old bitmap back to the memory device context.
PlatformInvokeGDI32.SelectObject(hMemDC, hOld);
//We delete the memory device context.
PlatformInvokeGDI32.DeleteDC(hMemDC);
 //We release the screen device context.
PlatformInvokeUSER32.ReleaseDC(PlatformInvokeUSER32.GetDesktopWindo
w(), hDC);//Image is created by Image bitmap handle and stored in local
variable.
Bitmap bmp = System.Drawing.Image.FromHbitmap(hBitmap);
//Release the memory to avoid memory leaks.
PlatformInvokeGDI32.DeleteObject(hBitmap);
//This statement runs the garbage collector manually.
GC.Collect();//Return the bitmap
return bmp;
}//If hBitmap is null return null.
return null;
}
#endregion
}
}
```

VB.NET:

```
Imports System
Imports System.Drawing

Namespace SampleGrabberNET
'This class shall keep all the functionality for capturing the desktop.
        Public Class CaptureScreen
                #Region" Public Class Functions"
                Public Shared Function GetDesktopImage ()As Bitmap
'In size variable we shall keep the size of the screen.
        Dim size As SIZE

'Variable to keep the handle to bitmap.
Dim hBitmap As IntPtr
'Here we get the handle to the desktop device context.
Dim hDC As IntPtr =
PlatformInvokeUSER32.GetDC(PlatformInvokeUSER32.GetDesktopWindow(()
'Here we make a compatible device context in memory for screen device
context.
Dim hMemDC As IntPtr = PlatformInvokeGDI32.CreateCompatibleDC(hDC(
'We pass SM_CXSCREEN constant to GetSystemMetrics to get the X
coordinates of screen.
size.cx=PlatformInvokeUSER32.GetSystemMetrics(PlatformInvokeUSER32.SM
CXSCREEN( _
'We pass SM_CYSCREEN constant to GetSystemMetrics to get the Y
coordinates of screen.
size.cy=PlatformInvokeUSER32.GetSystemMetrics(PlatformInvokeUSER32.SM
CYSCREEN( _
```

```
'We create a compatible bitmap of screen size using screen device context.
hBitmap = PlatformInvokeGDI32.CreateCompatibleBitmap(hDC, size.cx,
size.cy(
'As hBitmap is IntPtr we can not check it against null. For this purspose
IntPtr.Zero is used.
If Not hBitmap.Equals(IntPtr.Zero (Then
'Here we select the compatible bitmap in memory device context and keeps
the reference to Old bitmap.
Dim hOld As IntPtr = CType)PlatformInvokeGDI32.SelectObject(hMemDC,
hBitmap), IntPtr(
'We copy the Bitmap to the memory device context.
PlatformInvokeGDI32.BitBlt(hMemDC, 0, 0,size.cx,size.cy, hDC, 0, 0,
PlatformInvokeGDI32.SRCCOPY(
'We select the old bitmap back to the memory device context.
PlatformInvokeGDI32.SelectObject(hMemDC, hOld(
'We delete the memory device context.
PlatformInvokeGDI32.DeleteDC(hMemDC(
' We release the screen device context.
PlatformInvokeUSER32.ReleaseDC(PlatformInvokeUSER32.GetDesktopWindo
w(), hDC' (Image is created by Image bitmap handle and stored in local
variable.
Dim bmp As Bitmap = System.Drawing.Image.FromHbitmap(hBitmap(
'Release the memory to avoid memory leaks.
PlatformInvokeGDI32.DeleteObject(hBitmap(
'This statement runs the garbage collector manually.
GC.Collect' ()Return the bitmap
Return bmp
End If' If hBitmap is null return null.
Return Nothing
End Function
#End Region
End Class
End Namespace
```

وحتى نستطيع التحكم في حجم الصورة سوف نكتب ال method التالية:

C#:

```
public Bitmap ResizeBitmap( Bitmap b, int nWidth, int nHeight )
{
Bitmap result = new Bitmap( nWidth, nHeight );  using( Graphics g =
Graphics.FromImage( (Image) result ) )    g.DrawImage( b, 0, 0, nWidth,
nHeight );
 return result;
}
```

VB.NET:

```
Public Function ResizeBitmap(ByVal b As Bitmap, ByVal nWidth As Integer,
ByVal nHeight As Integer) As Bitmap
    Dim result As Bitmap = New Bitmap(nWidth, nHeight)
    ' Using
    Dim g As Graphics = Graphics.FromImage(CType(result, Image))
    Try
```

```
        g.DrawImage(b, 0, 0, nWidth, nHeight)
    Finally
        CType(g, IDisposable).Dispose()
    End Try
    Return result
End Function
```

سوف نستخدم ال Namespaces التالية في البرنامج لتعامل مع ال Multicasting :

C#:

```
using System.Net;
using System.Net.Sockets;
using System.IO;
```

VB.NET:

```
imports System.Net
imports System.Net.Sockets
imports System.IO
```

ثم نقوم بعمل Timer لالتقاط صورة سطح المكتب و إرسالها إلى ال Multicast Group المحدد :

C#:

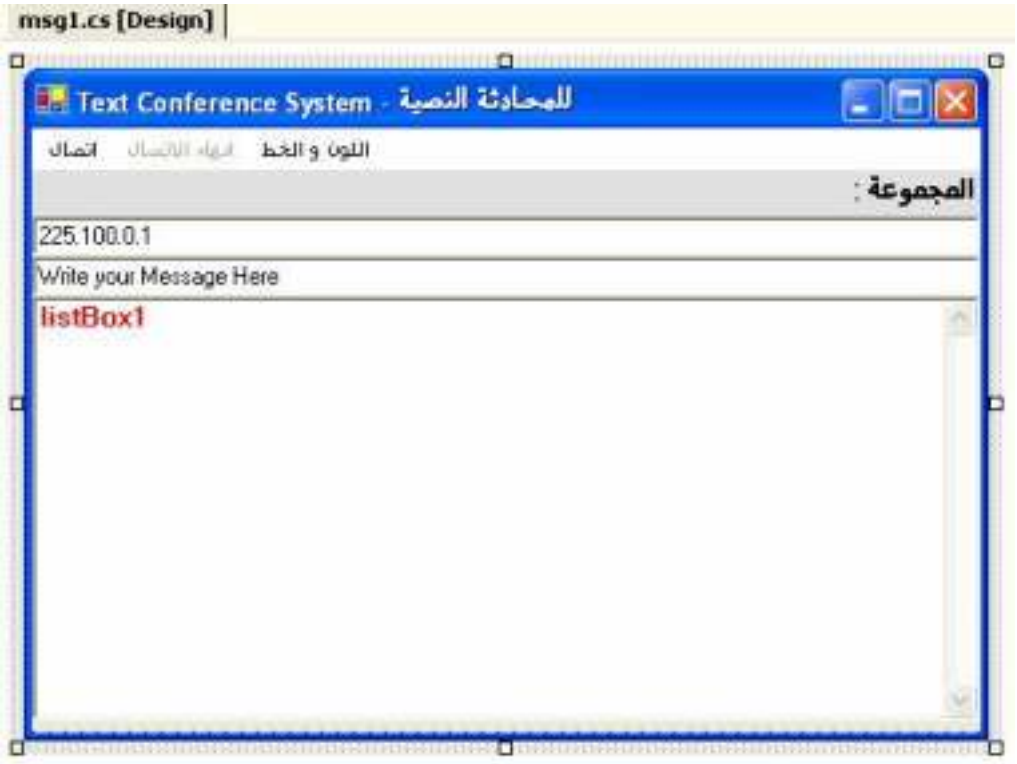
```
 Bitmap bt = new Bitmap(CaptureScreen.GetDesktopImage());
picScreen.Image = ResizeBitmap(bt, 352, 200 );
MemoryStream ms = new MemoryStream();
picScreen.Image.Save(ms,System.Drawing.Imaging.ImageFormat.Jpeg);
byte[] arrImage = ms.GetBuffer();
ms.Close();
Socket server = new Socket(AddressFamily.InterNetwork,
SocketType.Dgram, ProtocolType.Udp);
IPEndPoint iep = new IPEndPoint(IPAddress.Parse(textBox1.Text), 5020);
server.SendTo(arrImage,iep);
server.Close();
```

VB.NET:

```
Dim bt As Bitmap = New Bitmap(CaptureScreen.GetDesktopImage)
picScreen.Image = ResizeBitmap(bt, 352, 200)
Dim ms As MemoryStream = New MemoryStream
picScreen.Image.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg)
Dim arrImage As Byte() = ms.GetBuffer
ms.Close
Dim server As Socket = New Socket(AddressFamily.InterNetwork,
SocketType.Dgram, ProtocolType.Udp)
Dim iep As IPEndPoint = New IPEndPoint(IPAddress.Parse(textBox1.Text),
5020)
server.SendTo(arrImage, iep)
server.Close
```

:Full/Half Duplex Multicast Text Conferencing System -3

وحتى يستطيع الطلبة التحدث إلى الأستاذ باستخدام ال Text Chat Multicast Conference System سوف نقوم بإنشاء New Form جديد وكما في الشكل التالي:



ثم قم بإضافة ال Namespaces التالية:

C#:

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

VB.NET:

```
imports System.Net
imports System.Net.Sockets
imports System.Text
imports System.Threading
```

سوف نستخدم ال method التالية لإجراء عملية الإرسال حيث سترسل الرسالة عند الضغط على ال Enter بعد كتابة الرسالة في ال Textbox المخصص :

C#:

```
private void msg_KeyPress(object sender,
System.Windows.Forms.KeyPressEventArgs e)
{if(e.KeyChar == '\r'){
try{
Socket server = new Socket(AddressFamily.InterNetwork,SocketType.Dgram,
ProtocolType.Udp);
IPEndPoint iep = new IPEndPoint(IPAddress.Parse(txt_host.Text), 5020);
```

```
byte[] data = Encoding.ASCII.GetBytes(msg.Text);
server.SendTo(data, iep);
server.Close();
msg.Clear();
msg.Focus();
}
catch(Exception){}}}
```

VB.NET:

```
Private Sub msg_KeyPress(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyPressEventArgs)
    If e.KeyChar = Microsoft.VisualBasic.Chr(13) Then
        Try
            Dim server As Socket = New Socket(AddressFamily.InterNetwork,
SocketType.Dgram, ProtocolType.Udp)
            Dim iep As IPEndPoint = New
IPEndPoint(IPAddress.Parse(txt_host.Text), 5020)
            Dim data As Byte() = Encoding.ASCII.GetBytes(msg.Text)
            server.SendTo(data, iep)
            server.Close()
            msg.Clear()
            msg.Focus()
        Catch generatedExceptionVariable0 As Exception
        End Try
    End If
End Sub
```

وسوف نستخدم الميثود التالية لعملية الاستقبال حيث ستعرض الرسالة المستقبلة في list Box مخصص:

C#:

```
public void server()
{
try
{
UdpClient sock = new UdpClient(5020);
sock.JoinMulticastGroup(IPAddress.Parse(txt_host.Text), 50);
IPEndPoint iep = new IPEndPoint(IPAddress.Any, 0);

byte[] data = sock.Receive(ref iep);
string stringData = Encoding.ASCII.GetString(data, 0, data.Length);
listBox1.Items.Add(iep.Address.ToString() +" :_  "+stringData );
sock.Close();
listBox1.Focus();
msg.Focus();
myth.Abort();
}catch(Exception){}}
```

VB.NET:

```
Public Sub server()
   Try
      Dim sock As UdpClient = New UdpClient(5020)
      sock.JoinMulticastGroup(IPAddress.Parse(txt_host.Text), 50)
      Dim iep As IPEndPoint = New IPEndPoint(IPAddress.Any, 0)
      Dim data As Byte() = sock.Receive(iep)
      Dim stringData As String = Encoding.ASCII.GetString(data, 0,
data.Length)
      listBox1.Items.Add(iep.Address.ToString + " :_ " + stringData)
      sock.Close()
      listBox1.Focus()
      msg.Focus()
      myth.Abort()
   Catch generatedExceptionVariable0 As Exception
   End Try
End Sub
```

ولاستدعائها لابد من استخدام ال Threading ، قم بعمل Timer واستدعي فيه ال method السابقة باستخدام ال Thread وكما يلي:

C#:

```
Thread myth;
myth= new Thread  (new System.Threading .ThreadStart(server));
myth.Start ();
```

VB.NET:

```
Dim myth As Thread
myth = New Thread(New System.Threading.ThreadStart(server))
myth.Start
```

سوف نشغل ال Timer عند الضغط على زر الاتصال باستخدام timer1.Enabled = true
وفي زر إنهاء الاتصال قم بإضافة الكود التالي:

C#:

```
timer1.Enabled = false;
txt_host.ReadOnly = false;
msg.Enabled=false;
        try
        {
        Socket server = new Socket(AddressFamily.InterNetwork,
        SocketType.Dgram, ProtocolType.Udp);
IPEndPoint iep = new IPEndPoint(IPAddress.Parse(txt_host.Text), 5020);
  byte[] data = Encoding.ASCII.GetBytes("has Left the Room");

server.SendTo(data, iep);
server.Close();
msg.Clear();
msg.Focus();
        }
catch(Exception){}
```

VB.NET:

```
timer1.Enabled = False
txt_host.ReadOnly = False
msg.Enabled = False
Try
Dim server As Socket = New Socket(AddressFamily.InterNetwork,
SocketType.Dgram, ProtocolType.Udp)
Dim iep As IPEndPoint = New IPEndPoint(IPAddress.Parse(txt_host.Text),
5020)
Dim data As Byte() = Encoding.ASCII.GetBytes("has Left the Room")
 server.SendTo(data, iep)
 server.Close
 msg.Clear
 msg.Focus
Catch generatedExceptionVariable0 As Exception
End Try
```

Chapter 5
Application Layer Programming

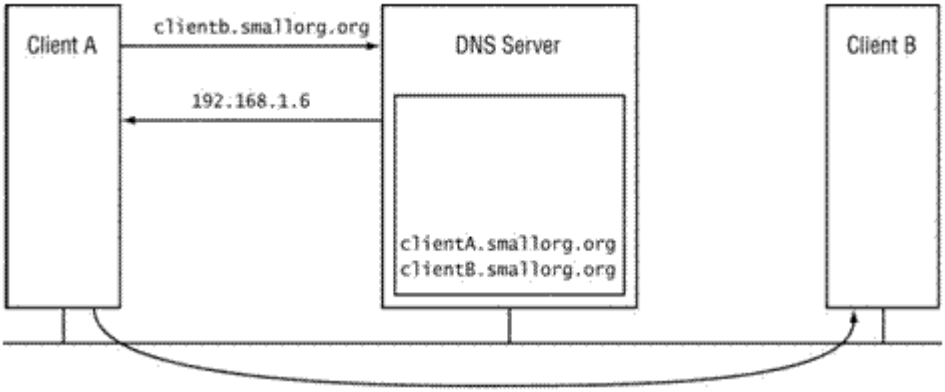
Application Layer Programming

A. DNS Programming
B. SMTP Programming
C. POP3 Programming
D. HTTP Programming
E. Web Services & XML Programming
F. FTP Programming

بسم الله الرحمن الرحيم

5.1: DNS Programming :

تعتبر خدمة DNS واحدة من أهم الخدمات التي تستخدم في الإنترنت والشبكات بشكل عام، وتختصر وظيفة DNS بالقيام بعملية ترجمة ال Domain Name إلى Domain IP من وإلى العكس ويتم ذلك من خلال مجموعة كبيرة جدا من مزودات DNS (والتي تقوم بتحديث قاعدة البيانات الخاصة بها كل فترة معينة) ، تبدأ هذه العملية بقيام الـClient A بطلب ال Domain الخاص بال Client B وذلك بإدخال Domain Name الخاص به - حيث تم مسبقا قيام ال Client B بتعريف نفسه في قاعدة البيانات الخاصة ب DNS Server - كما يحتوي كل Client على قاعدة بيانات تحتوي على عناوين ال Domains وتسمى بال local DNS حيث يقوم بالبحث بداخلها على عنوان Domain من خلال Domain Name فإذا لم يجده يقوم بطلب عنوان الدومين من ال DNS Server وبعد إيجاده يقوم ال DNS Server بإرسال العنوان إلى ال Client ويقوم بدوره بتخزين العنوان في Local DNS الخاص به ، انظر إلى الشكل التالي:



في الدوت نيت يمكننا التعامل مع DNS باستخدام System.Net Name Space والتي تحتوي على جميع ال DNS Classes والتي تحتوي على كل ال Methods الخاصة ب DNS وتقسم هذه الميثودس إلى قسمين متزامن Synchronous Methods و غير متزامن Asynchronous Methods وهي كما يلي:

أولا الميثودس المتزامنة Synchronous Methods وهي :

GetHostName والتي تستخدم لجلب اسم الهوست وترجع هذه الميثود قيمة String تحتوي على ال Computer Name ولا تأخذ هذه الميثود أي باراميترات ويمكن استخدامها كما يلي :

C#:

```
string hostname = Dns.GetHostName();
```

VB.NET:

```
Private hostname As String = Dns.GetHostName
```

الميثود GetHostByName و الميثود GetHostByAddress وتستخدم كل منها كما يلي:

C#:

```
IPHostEntry host_ip = Dns.GetHostByName(Computer_Name); // لجلب العنوان باستخدام الاسم
IPHostEntry host_name = Dns.GetHostByAddress(IP_Address); // لجلب الاسم باستخدام العنوان
```

VB.NET:

```
Private host_ip As IPHostEntry = Dns.GetHostByName(Computer_Name)
Private host_name As IPHostEntry = Dns.GetHostByAddress(IP_Address)
```

الميثود Resolve وهي Overloaded Method حيث ترجع Host Name إذا أرسلت لها IP Address وترجع Host Address إذا أرسلت لها Host Name في ال IPHostEntry ولا يختلف استخدامها عن استخدام الميثودس السابقة .
وهذا المثال يبين طريقة استخدامها :

C#:

```
using System;
using System.Net;

class FMO_DNS
{
        public static void Main()
        {
IPHostEntry IPHost = Dns.Resolve("www.yahoo.com"); // الدومين الذي نريد معرفة الأي بي الخاص به
Console.WriteLine(IPHost.HostName); // جلب اسم الدومين بالكامل

IPAddress[] addr = IPHost.AddressList; // وضع قائمة العناوين في مصفوفة

                for(int i= 0; i < addr.Length ; i++) // طباعة عناصر المصفوفة
                {Console.WriteLine(addr[i]);}}}
```

VB.NET:

```
Imports System
Imports System.Net

Class FMO_DNS

   Public Shared Sub Main()
      Dim IPHost As IPHostEntry = Dns.Resolve("www.yahoo.com")
      Console.WriteLine(IPHost.HostName)
      Dim addr As IPAddress() = IPHost.AddressList
      Dim i As Integer = 0
      While i < addr.Length
         Console.WriteLine(addr(i))
         System.Math.Min(System.Threading.Interlocked.Increment(i), i - 1)
      End While
   End Sub
End Class
0
```

ثانيا الميثودس غير المتزامنة Asynchronous Methods :

وتبدأ عادة بكلمة Begin أو End ومن الأمثلة عليها :

BeginGetHostByName و BeginResolve و EndGetHostByName و EndResolve

طبيعة عملها كما هو الحال في الميثودس المتزامنة لكنها تختلف بكون انه لا يشترط تنفيذها لإكمال عمل البرنامج في حين المتزامن لا تسمح بالإنتقال إلى الخطوة الثانية في البرنامج إلا في حالة انتهاء عملها وقد تسبب هذه السيئة بخفض البريفورمانس بشكل عام في البرنامج لذلك ينصح باستخدام الطريقة الغير متزامنة وتستخدم كما يلي : Begin___

public static IAsyncResult BeginResolve(string *hostname*,
AsyncCallback *requestCallback*, object *stateObject*)

حيث يتم وضع الهوست نيم في الباروميتر الأول و الباروميتر الثاني يعرف فيه ال delegate وتسمح لك بتمرير مدخلات إلا delegate ، ويستخدم End____ كما يلي :

public static IPHostEntry EndResolve(IasyncResult *ar*)

وهنا مثال شامل و بسيط يقوم بجلب جميع ال IP's الموجودة على الشبكة حيث يعمل على جلب ال host names من خلال الخاصية ProcessStartInfo من خلال الميثود StandardOutput حيث يتم تحويله إلى host name ثم يتم تحويل GetMachineNamesFromProcessOutput ثم تخزينها في Collicaion الأسماء إلى عناوين من خلال الميثود Dns.Resolve .. طبعا يتم استخدام ال StreamReader لقراءة ال collection الخاص بال ProcessStartInfo وهذا هو المثال :

C#:

```
using System;
using System.IO;
using System.Diagnostics;
using System.Net;
using System.Collections.Specialized;

namespace NetworkIPs
{
        public class Names

        {
                public StringCollection GetNames()

                {
ProcessStartInfo _startInfo = new ProcessStartInfo("net","view");
                        _startInfo.CreateNoWindow = true;
                        _startInfo.UseShellExecute = false;
                        _startInfo.RedirectStandardOutput = true;
                        Process _process = Process.Start(_startInfo);
                        StreamReader _reader = _process.StandardOutput;
                        StringCollection _machineNames =
GetMachineNamesFromProcessOutput(_reader.ReadToEnd());
                        StringCollection _machineIPs = new StringCollection();
                        foreach(string machine in _machineNames)
                        {
                                _machineIPs.Add(IPAddresses(machine));
                        }
```

```
                return _machineIPs;
            }
            private static string IPAddresses(string server)
            {
                try
                {
System.Text.ASCIIEncoding ASCII = new System.Text.ASCIIEncoding();
                    // Get server related information.
                    IPHostEntry heserver = Dns.Resolve(server);
                    //assumin the machine has only one IP address
                    return heserver.AddressList[0].ToString();
                }
                catch
                {
return "Address Retrieval error for " + server;
                }
            }
            //string manipulations
            private StringCollection
GetMachineNamesFromProcessOutput(string processOutput)
            {
string _allMachines = processOutput.Substring( processOutput.IndexOf("\\"));
                StringCollection _machines= new StringCollection();
while(_allMachines.IndexOf("\\") != -1 )
                {
_machines.Add(_allMachines.Substring(_allMachines.IndexOf("\\"),
_allMachines.IndexOf(" ",_allMachines.IndexOf("\\")) -
_allMachines.IndexOf("\\")).Replace("\\",String.Empty));
_allMachines = _allMachines.Substring(_allMachines.IndexOf("
",_allMachines.IndexOf("\\") + 1));
                }
                return _machines;
            }
        }

        public class Runner
        {
            static void Main()
            {
                Names _names = new Names();
                StringCollection names = _names.GetNames();
                foreach(string name in names)
                Console.WriteLine(name);
                Console.ReadLine();
            }
        }
```

VB.NET:

```
Imports System
Imports System.IO
Imports System.Diagnostics
Imports System.Net
Imports System.Collections.Specialized


Public Class Names

    Public Function GetNames() As StringCollection

        Dim _startInfo As ProcessStartInfo = New ProcessStartInfo("net",
"view")
        _startInfo.CreateNoWindow = True
        _startInfo.UseShellExecute = False
        _startInfo.RedirectStandardOutput = True
        Dim _process As Process = Process.Start(_startInfo)
        Dim _reader As StreamReader = _process.StandardOutput
        Dim _machineNames As StringCollection =
GetMachineNamesFromProcessOutput(_reader.ReadToEnd())
        Dim _machineIPs As StringCollection = New StringCollection
        For Each machine As String In _machineNames
            _machineIPs.Add(IPAddresses(machine))
        Next machine
        Return _machineIPs
    End Function
    Private Shared Function IPAddresses(ByVal server As String) As String
        Try
            Dim ASCII As System.Text.ASCIIEncoding = New
System.Text.ASCIIEncoding
            ' Get server related information.
            Dim heserver As IPHostEntry = Dns.Resolve(server)
            'assumin the machine has only one IP address
            Return heserver.AddressList(0).ToString()
        Catch
            Return "Address Retrieval error for " & server
        End Try
    End Function
    'string manipulations
    Private Function GetMachineNamesFromProcessOutput(ByVal
processOutput As String) As StringCollection
        Dim _allMachines As String =
processOutput.Substring(processOutput.IndexOf("\"))
        Dim _machines As StringCollection = New StringCollection
        Do While _allMachines.IndexOf("\") <> -1
            _machines.Add(_allMachines.Substring(_allMachines.IndexOf("\"),
_allMachines.IndexOf(" ", _allMachines.IndexOf("\")) -
_allMachines.IndexOf("\")).Replace("\", String.Empty))
            _allMachines = _allMachines.Substring(_allMachines.IndexOf(" ",
_allMachines.IndexOf("\") + 1))
```

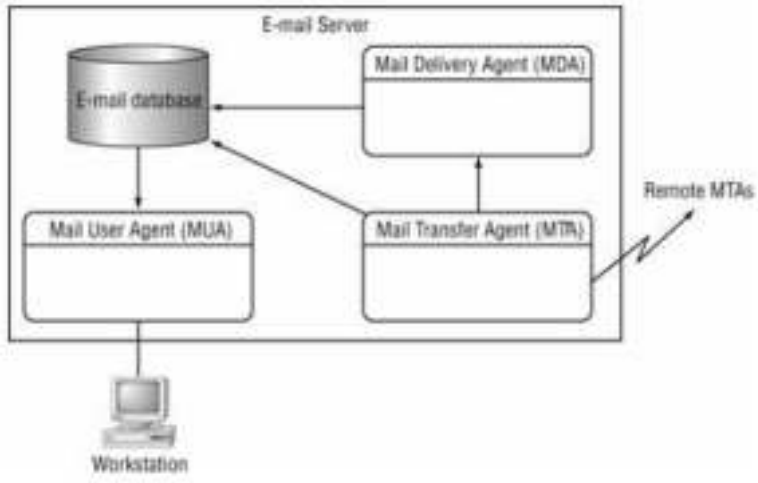
```
        Loop
        Return _machines
    End Function
End Class

Public Class Runner
    Shared Sub Main()
        Dim _names As Names = New Names
        Dim names As StringCollection = _names.GetNames()
        For Each name As String In names
            Console.WriteLine(name)
        Next name
        Console.ReadLine()
    End Sub
End Class
```

SMTP & POP3 Programming :5.2

تحدثنا في الجزء السابق عن برمجة برتوكول DNS والمسئول عن عملية ترجمة Domain من اسم نطاق إلى IP وبالعكس وبينا كيفية القيام بهذه العملية في سي شارب ، في هذا الجزء سوف نتحدث عن برمجة بعض البروتوكولات الأخرى لطبقة ال Application Layer وهما هنا ال SMTP والمسئول عن إرسال الرسائل عبر البريد الإلكتروني و ال POP3 والمسئول عن عملية توصيل الرسالة إلى الزبون من خلال عمل Download لها من ال Mail Server وفي الجزء اللاحق سوف نتحدث عن ال HTTP Programming والذي يستخدم بشكل أساسي في تصفح ال Web ، مع العلم انه يوجد بروتوكولات كثيرة سوف آتي على شرحها عند الحاجة ..

الجزء الأول: SMTP – Simple Mail Transfer Protocol Programming

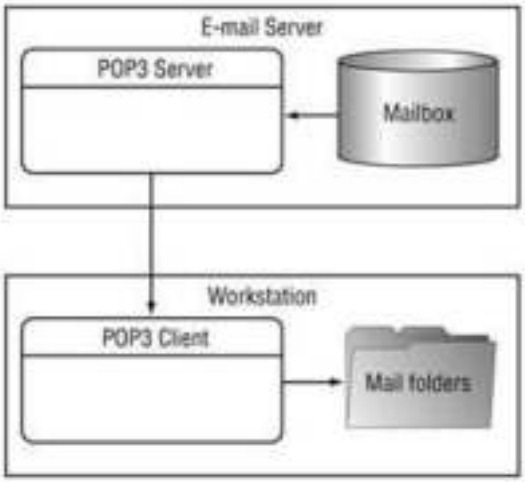
من المعروف أن ال Mail Server يقوم بتجزئة عمليات إرسال و استقبال البريد الإلكتروني عبر الإنترنت إلى ثلاثة أجزاء وهي كما في الشكل التالي :



Message Transfer Agent – MTA والمسئول عن الإرسال Outgoing والتوصيل Incoming للرسائل

Message Delivery Agent -MDA و المسئول عن عمليات ال filtering والتأكد من وصول الرسالة

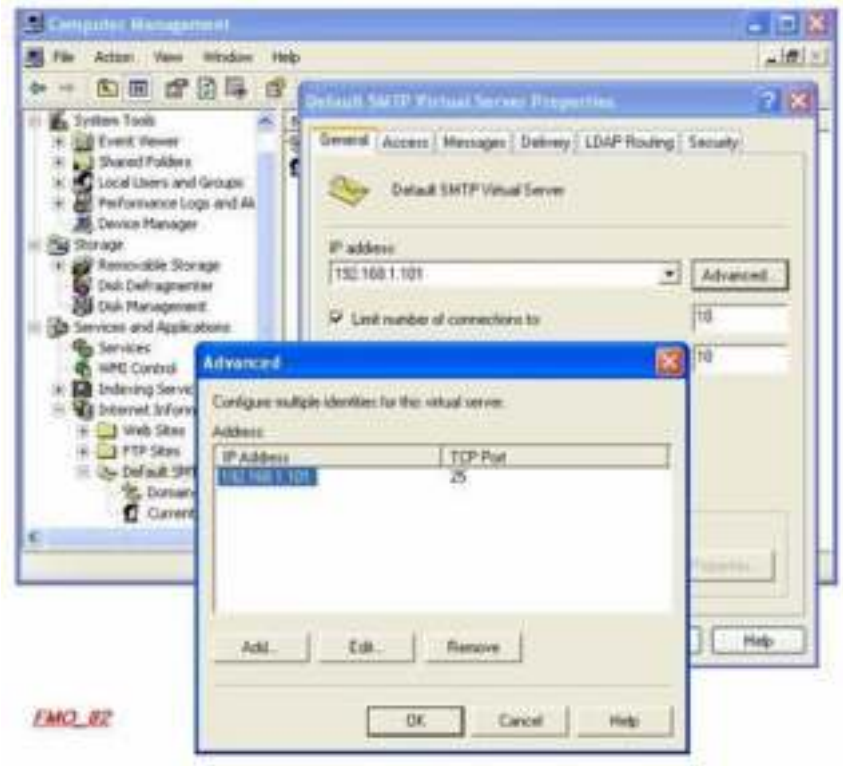
Message User Agent -MUA والمسئول عن عملية قراءة و تخزين الرسالة في Database لدى المستقبل Client وتتم هذه العملية باستخدام بروتوكول POP - Post Office Protocol انظر إلى الشكل التالي :



و يستخدم برتوكول ال SMTP Simple Mail Transfer Protocol بشكل أساسي في ال MTA أي عمليات إرسال Outgoing وتوصيل Incoming الرسائل .

لتطبيق يجب أولا التأكد من أنك تملك حساب SMTP من ال Internet Provider الخاص بك تستطيع تجربة ال Account الخاص بك من خلال برنامج ال Outlook Express الموجود مع ال Windows إذا كنت لا تملك حساب SMTP تستطيع تجربة البرنامج من خلال إنشاء Virtual SMTP Server عن طريق ال IIS وذلك بتثبيتها من : Control Panel >> Add/Remove Programs تأكد من تفعيل كل من ال IIS وال SMTP كما في الشكل التالي :

ثم إعداد الServer من ال IIS كما في الشكل التالي :



تدعم الدوت نيت استخدام بروتوكول ال SMTP من خلال Name Space System.Web.Mail و تحتوي على الكلاس SmtpMail والتي من خلالها نستخدم الميثود Send والتي تستخدم لإرسال الرسالة عبر الPort 25 وهو الPort المخصص لبروتوكول SMTP و تعتبر الميثود Send " overloaded Method " حيث تأخذ عدة أشكال إذ بإمكانك استخدامها مع براميتر واحد إلى أربعة باراميترات ، وبشكل افتراضي نستخدم البرامترات التالية :

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

البراميتر الأول يوضع فيه عنوان المرسل والثاني يوضع فيه عنوان المرسل إليه و البراميتر الثالث لعنوان الرسالة والرابع لنص الرسالة .

ولعمل برنامج يقوم بإرسال البريد الإلكتروني قم بإنشاء New Form كما في الشكل التالي:

ثم قم بإضافة System.Web.Mail Name Space ، (إذا لم تظهر لديك Mail. قم بإدراج System.Web Name Space إلى ال References) ثم قم بكتابة الكود التالي :

لا تنسى إضافة Name Space هذا في بداية البرنامج

C#:

```
using System.Web.Mail;
```

VB.NET:

```
imports System.Web.Mail;
```

ثم كتابة الكود هذا في زر الإرسال

C#:

```
try
{
string from = textBox1.Text;
string to = textBox2.Text;
string subject = textBox3.Text;
string body = textBox4.Text;
SmtpMail.SmtpServer = textBox5.Text;
SmtpMail.Send(from, to, subject, body);
}
catch (Exception ex) {MessageBox.Show(ex.Message);}
```

VB.NET:

```
Try
Dim from As String = textBox1.Text
 Dim to As String = textBox2.Text
Dim subject As String = textBox3.Text
Dim body As String = textBox4.Text
 SmtpMail.SmtpServer = textBox5.Text
```

```
 SmtpMail.Send(from, to, subject, body)
Catch ex As Exception
Msgbox(ex.Message)
End Try
```

ملاحظة هامة جدا :

هذا الكود يعمل بشكل جيد، لاكن يجب التأكد من تفعيل ال SMTP من ال IIS كما ذكر في السابق وقم بوضع IP الخاص ب ال SMTP (والذي تم تعريفه مسبقا في SMTP Virtual Server)بال SMTP Server Textbox ، يجب التأكد أيضا من ال SMTP Server لديك يدعم استخدام المكتبة CDO2 - Microsoft Collaboration Data Objects Version 2 وإلا سوف تحصل على Exception يخبرك بأنه لا يستطيع الوصول إلى CDO2 Object ، في العادة يتم استخدمها مع Windows XP و Windows 2000 وتعمل بشكل افتراضي عند تثبيت ال SMTP Virtual Server أو مع Microsoft Exchange Server2003 أما إذا كنت تستخدم Exchange Version 5 أو 5.5 فسوف تحصل على ال Exception السابق الذكر .

الجزء الأكثر تقدم: SMTP Advanced Programming

يعتبر المثال السابق مثال بسيط لإرسال رسائل عبر SMTP باستخدام CDO2 ، وفي العادة عند إنشاء برامج مثل برنامج ال Outlook يتم استخدام ال HTML Format بالإضافة إلا إمكانية إرسال ملحقات وطبعا يعطيك عدة خيارات لإرسال و استقبال البريد الإلكتروني هل باستخدام ال HTTP أو ال POP3 ... وهنا سوف نقوم بإنشاء برنامج بسيط يقوم بإرسال واستقبال البريد الإلكتروني باستخدام ال SMTP و POP3 بنسبة لاستخدام ال POP3 فيجب أن يتوفر لديك حساب POP3 من ال ISP الخاص بك أو أن تقوم بتثبيت Microsoft Exchange Server2003 على جهازك وإعداده بحيث يستخدم ال POP3 إذ عندها سوف تحتاج لوجود Domain Controller مثبت على الجهاز و Windows 2003 Server بالإضافة إلى تثبيت الActive Directory عليه.

قدمت الدوت نيت دعم ممتاز لاستخدام هذه الخواص وذلك من خلال Name Space System.Web.Mail وباستخدام الكلاس MailMessage لدعم ال HTML Format و الكلاس MailAttachment لدعم إمكانية إرسال ملحقات مع الرسالة ولاكن لبرمجة ال POP3 يلزم استخدام System.Net.Sockets Name Space و System.Net و System.IO حيث يتم عمل Session خاص مع الServer للقيام بعملية تفحص وجود رسائل جديدة وفي حالة وجودها يقوم بتعبئتها في List Box أو Treelist حسب الحاجة وعند الضغط على إحداها يقوم ال Client بعمل Download لرسالة من ال Mail Server ولعمل Advanced SMTP eMail Sender قم بأخذ Object من الكلاس MailMessage كما يلي :

C#:

```
using System.Web.Mail;

try
{
 MailMessage mm = new MailMessage();
```

ثم إضافة الكود التالي وكما في السابق

```
   mm.From = textBox1.Text;
      mm.To = textBox2.Text;
  // mm.Cc = لاحظ انه يمكنك من الإرسال لأكثر من شخص هذه حسب الحاجة
  // mm.Bcc =
   mm.Subject = textBox3.Text;
   mm.Headers.Add("Reply-To", "fadi822000@yahoo.com"); //  لوضع أي إضافات
تريدها مع الرسالة
   mm.Headers.Add("Comments", "This is a test HTML message");
   mm.Priority = MailPriority.High; // يمكنك وضع خيارات أهمية الرسالة
```

```
   mm.BodyFormat = MailFormat.Html; // نوع الفورمات المستخدم
   mm.Body = "<html><body><h1>" + textBox4.Text + "</h1></html>";
   SmtpMail.Send(mm);
}
catch (Exception ex) {MessageBox.Show(ex.Message);}
```

VB.NET:

```
imports System.Web.Mail;

Try
Dim mm As MailMessage = New MailMessage
 mm.From = textBox1.Text
 mm.To = textBox2.Text
 mm.Subject = textBox3.Text
 mm.Headers.Add("Reply-To", "fadi822000@yahoo.com")
 mm.Headers.Add("Comments", "This is a test HTML message")
 mm.Priority = MailPriority.High
 mm.BodyFormat = MailFormat.Html
 mm.Body = "<html><body><h1>" + textBox4.Text + "</h1></html>"
 SmtpMail.Send(mm)
Catch ex As Exception
Msgbox(ex.Message)
End Try
```

لاحظ أن جسم الرسالة يستخدم كود ال HTML وهذا يمكنك من وضع أي لون أو حجم أو أي شيء يمكن عمله باستخدام ال HTML (راجع قسم ال HTML بالمنتدى لتعرف على هذه اللغة السكربتية الرائعة) ، ولجعل البرنامج قادر على إرسال ملحقات يجب استخدام الكلاس MailAttachment وإدراج اسم الملف فيه وكما يلي بالكود :

C#:

```
MailAttachment myattach =
new MailAttachment("Your_Attached_File_path.extension",
MailEncoding.Base64);
```

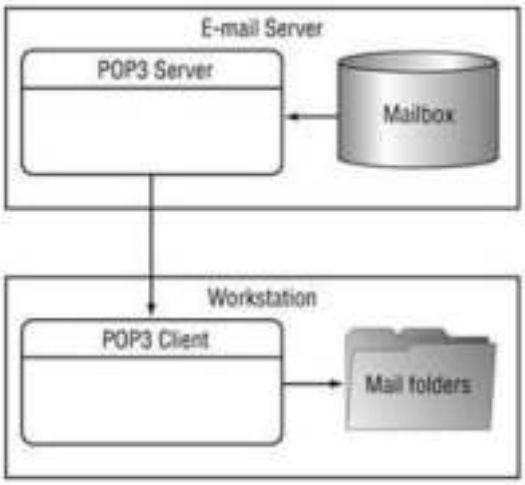
وهنا قد انتهينا من عمل برنامج ال SMTP بشكل كامل ، طبعا عملية ال Design وغيرها تعتمد على حسب ذوق وذكاء وخبرة المبرمج.

VB.NET:

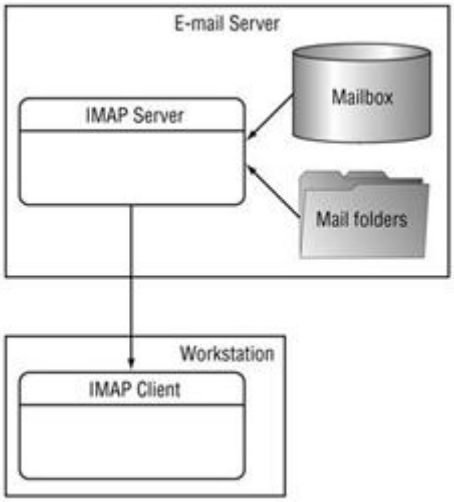
```
Dim myattach As MailAttachment = New
MailAttachment("Your_Attached_File_path.extension", MailEncoding.Base64)
```

ثانيا : POP3- Post Office Protocol Version 3 Programming

كما تحدثنا سابقا فإن وظيفة برتوكول ال POP3 والذي يعمل في جزء ال MUA - Mail User Agent على Port 110 ضمن برتوكول ال TCP تكمن في كونه المسئول عن عملية توصيل الرسالة إلى الزبون Client من خلال عمل Download لها من ال Mail Server حيث تحفظ الرسائل في ال Mail Folder والموجود أساسا في جهاز ال Client أنظر إلى الشكل التالي:



ومن البدائل لل POP3 برتوكول IMAP – Interactive Mail Access Protocol فمن خلاله يستطيع المستخدم إنشاء Mail Folder خاصة به ولاكن في ال Mail server وليس في جهاز الزبون وتعتبر هذه من ميزات ال IMAP وسيئاته بنفس الوقت إذ أن قراءة الرسالة تتم مباشرة من خلال ال Server حيث تستطيع قراءتها من أكثر من Client ولاكن المشكلة فيه هي تحكم مدير خادم الرسائل Mail Server Administrator بحجم ال Mail Folder إذ تكون في العادة سعتها محدودة أنظر إلى الشكل التالي :



لاحظ أن ال Mail Folder يقع ضمن Mail Server ويتم قراءته بعد التحقق Authentication من اسم المستخدم وكلمة المرور لاكن كما قلنا فإن مشكلته تكمن في محدودية سعة ال Mail Folder لذا ينصح لشركات الكبيرة استخدام ال POP3 كونه غير محدود السعة فالذي يتحكم في السعة هو ال Client ولا دخل ل Mail Server Administrator بها.

وبما أننا قررنا اعتماد ال POP3 لعملية قراءة الرسائل سوف نبدأ ببرمجته إذ يلزم الأمر استخدام System.Net.Sockets Name Space و System.Net و System.IO حيث يتم عمل Session خاص مع الServer باستخدام ال Socket للقيام بعملية تفحص وجود

رسائل جديدة وفي حالة وجودها يقوم بتعبئة عناوينها في List Box أو Treelist خاص حسب الحاجة وعند الضغط على إحداها يقوم ال Client بعمل Download لرسالة من ال Mail Server إلى ال Mail Folder ثم عرضها في Textbox.

ولتطبيق قم بإنشاء New Form جديد كما يظهر في الشكل التالي :

ثم قم بإضافة Name Spaces التالية :

C#:

```
using System.Net;
using System.Net.Sockets;
using System.IO;
```

VB.NET:

```
imports System.Net
imports System.Net.Sockets
imports System.IO
```

لاحظ انه يتم التعامل مع ال Socket وال Stream لإنشاء Session مع الServer باستخدام برتوكول ال TCP وقراءة الرسالة من ال POP3 Server .

ثم قم بإضافة التعاريف التالية في بداية البرنامج (أي بعد تعريف الكلاس الرئيسي – في منطقة ال Global Declaration) :

C#:

```
public TcpClient Server;// اشتقاق اوبجكت من برتوكول التي سي بي وذلك بهدف
 إنشاء الجلسة
public NetworkStream NetStrm;// سوف نستخدمه لإرسال معلومات المستخدم
public StreamReader  RdStrm; //  3 البوب من الواردة المعلومات لقراءةServer
public string Data; // لاستخدامها في معرفة عدد الرسائل
public byte[] szData; //  3 البوب من الواردة البيانات لتخزينServer
public string CRLF = "\r\n";// لاستخدامها في البرنامج لعمل سطر جديد..
```

VB.NET:

```
Public Server As TcpClient
Public NetStrm As NetworkStream
Public RdStrm As StreamReader
Public Data As String
Public szData As Byte()
Public CRLF As String = "" & Microsoft.VisualBasic.Chr(13) & "" &
Microsoft.VisualBasic.Chr(10) & ""
```

في ال Connect Button قم بإضافة الكود التالي :

C#:

```
// create server POP3 with port 110
//  المخصص وهو Port110 عبر الServerلإنشاء سيشن مع البوب
Server = new TcpClient(POPServ.Text,110);
try
{
// initialization
NetStrm = Server.GetStream();
RdStrm= new StreamReader(Server.GetStream());
Status.Items.Add(RdStrm.ReadLine());

// Login Process
// إدخال اسم المستخدم وكلمة المرور وتمريرها إلى البوب Server
Data = "USER "+ User.Text+CRLF;
szData = System.Text.Encoding.ASCII.GetBytes(Data.ToCharArray());
NetStrm.Write(szData,0,szData.Length);
Status.Items.Add(RdStrm.ReadLine());
Data = "PASS "+ Passw.Text+CRLF;
// بعد التأكد من اسم المستخدم وكلمة المرور يتم قراءة صندوق الوارد الخاص
بالمستخدم
szData = System.Text.Encoding.ASCII.GetBytes(Data.ToCharArray());
NetStrm.Write(szData,0,szData.Length);
Status.Items.Add(RdStrm.ReadLine());

Send STAT command to get information ie: number of mail and size
                لمعرفة عدد الرسائل الموجودة في POP3 Server باستخدام الأمر STAT
Data = "STAT"+CRLF;
szData = System.Text.Encoding.ASCII.GetBytes(Data.ToCharArray());
NetStrm.Write(szData,0,szData.Length);
Status.Items.Add(RdStrm.ReadLine());
```

قم بإضافة الكود التالي إلى ال Disconnect Button :

```
// Send QUIT command to close session from POP server
Data = "QUIT"+CRLF;
szData = System.Text.Encoding.ASCII.GetBytes(Data.ToCharArray());
NetStrm.Write(szData,0,szData.Length);
Status.Items.Add(RdStrm.ReadLine());
//close connection
NetStrm.Close();
RdStrm.Close();
```

VB.NET:

```
Server = New TcpClient(POPServ.Text, 110)
NetStrm = Server.GetStream
RdStrm = New StreamReader(Server.GetStream)
Status.Items.Add(RdStrm.ReadLine)
Data = "USER " + User.Text + CRLF
szData = System.Text.Encoding.ASCII.GetBytes(Data.ToCharArray)
NetStrm.Write(szData, 0, szData.Length)
Status.Items.Add(RdStrm.ReadLine)
Data = "PASS " + Passw.Text + CRLF
szData = System.Text.Encoding.ASCII.GetBytes(Data.ToCharArray)
NetStrm.Write(szData, 0, szData.Length)
Status.Items.Add(RdStrm.ReadLine)
Data = "STAT" + CRLF
szData = System.Text.Encoding.ASCII.GetBytes(Data.ToCharArray)
NetStrm.Write(szData, 0, szData.Length)
Status.Items.Add(RdStrm.ReadLine)
Data = "QUIT" + CRLF
szData = System.Text.Encoding.ASCII.GetBytes(Data.ToCharArray)
NetStrm.Write(szData, 0, szData.Length)
Status.Items.Add(RdStrm.ReadLine)
NetStrm.Close
RdStrm.Close
```

ولقراءة الرسائل من صندوق الوارد(بشكل افتراضي سيتم قراءة الرسالة الأخيرة) قم بإضافة الكود التالي إلى ال Read Last Come Email Button :

C#:

```
string szTemp;
Message.Clear();
try
        {

// retrieve mail with number mail parameter
Data = "RETR 1"+CRLF; // لتحديد رقم الرسالة المراد قراءتها

szData = System.Text.Encoding.ASCII.GetBytes(Data.ToCharArray());
NetStrm.Write(szData,0,szData.Length);
szTemp = RdStrm.ReadLine(); // تخزين الرسالة بشكل مؤقت حتى يتم طباعتها
        if(szTemp[0]!='-')
        {
        while(szTemp!=".")
                {
        Message.Text += szTemp+CRLF;
        szTemp = RdStrm.ReadLine();
        }
                }
        else
        {Status.Items.Add(szTemp);}
        }
catch(InvalidOperationException err){Status.Items.Add("Error:
"+err.ToString());}
```

VB.NET:

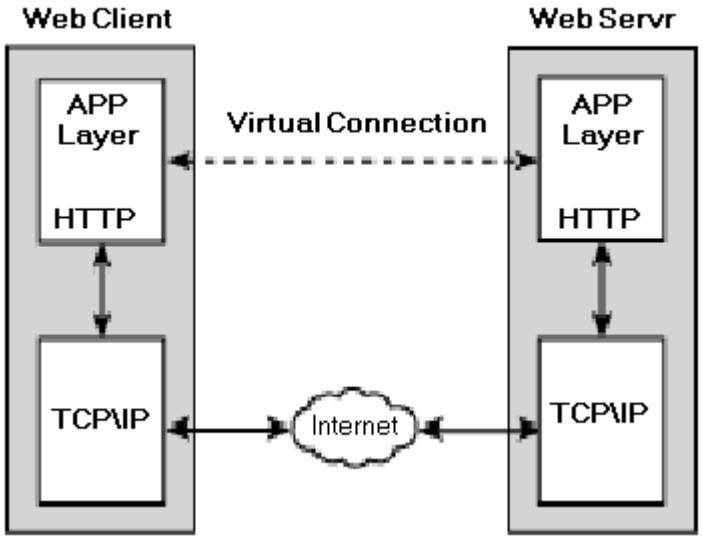
```
Dim szTemp As String
Message.Clear
Try
 Data = "RETR 1" + CRLF
 szData = System.Text.Encoding.ASCII.GetBytes(Data.ToCharArray)
 NetStrm.Write(szData, 0, szData.Length)
 szTemp = RdStrm.ReadLine
 If Not (szTemp(0) = "-"C) Then
   While Not (szTemp = ".")
     Message.Text += szTemp + CRLF
     szTemp = RdStrm.ReadLine
   End While
 Else
   Status.Items.Add(szTemp)
 End If
Catch err As InvalidOperationException
 Status.Items.Add("Error: " + err.ToString)
End Try
```

وهنا قد قمت بشرح كيفية عمل البوب 3 وبرمجته في الدوت نت وهذا مثال بسيط تستطيع البدء منه لعمل مشروع كامل شبيه بال Outlook الخاص بميكروسوفت حيث تستطيع استخدام ملف ال DLL الخاص بالإنترنت إكسبلورر لعرض الرسائل الواردة بدلا من عرضها على شكل HTML Code كما تستطيع عمل Tree List لوضع الرسائل الواردة حيث يكون لكل رسالة رقم تسلسلي يتم وضعه في الكود السابق لقراءتها حيث استخدمت الرقم 1 بشكل افتراضي والذي يقوم بقراءة الرسالة الأخيرة الواردة ،

HTTP – Hyper Text Transfer Protocol Programming :5.3

تتلخص وظيفة ال HTTP بشكل عام على انه البرتوكول المستخدم لتوصيل طلب المستخدم User Request إلى الويب Server ثم قيام ال web server بالرد على ال Request والذي يسمى ب Server Response وبتأكيد تستطيع نقل جميع أشكال ال (Multimedia) من النص وصورة و صوت و فيديو وغيره .. من ال Web Server إلى ال Client Application باستخدام Byte Stream object.

يعمل برتوكول ال HTTP على ال Application Layer وهذا يعني استخدامه بشكل مباشر من واجهة المستخدم كما هو الحال في DNS,SMTP,POP3,FTP انظر إلى الشكل التالي:



أولا : Downloading From Web Server

نستطيع التعامل مع ال Web Server في الدوت نيت باستخدام الكلاس WebClient الموجود في System.Net Name Space إذ تقدم لنا جميع الإمكانيات لتوصيل طلب الزبون و الرد عليه User Request & Server Response وتدعم ال WebClient Class ثلاثة Methods لتحميل البيانات من ال Web Server وهي:

1- **DownloadData** ووظيفتها جلب البيانات من ال Web Server وتخزينها في Byte Array وتعرض على شكل HTML Code وتستخدم كما يلي كمثال :

C#:

```
using System;
using System.Net;
using System.Text;
class DownloadData_Method
{
public static void Main ()
{
WebClient wc = new WebClient();
byte[] response =
wc.DownloadData("http://www.google.com");
Console.WriteLine(Encoding.ASCII.GetString(response));
}
}
```

VB.NET:

```
Imports System
Imports System.Net
Imports System.Text

Class DownloadData_Method

    Public Shared Sub Main()
        Dim wc As WebClient = New WebClient
        Dim response As Byte() = wc.DownloadData("http://www.google.com")
        Console.WriteLine(Encoding.ASCII.GetString(response))
    End Sub
End Class
```

2- **DownloadFile** ووظيفتها نقل ملف ما من ال Web Server وتخزينها مباشرة في Local Computer وهو سهل الاستخدام جدا إذ ما عليك سوا تمرير موقع الملف والمكان الذي تريد تخزين الملف فيه ويستخدم كما يلي كمثال :

C#:

```
using System;
using System.Net;

class DownloadFile_Method
{
public static void Main ()
{
WebClient wc = new WebClient();
string filename = "C:\\ra.zip";

Console.WriteLine("Download in Progress Please Waite...");

wc.DownloadFile("http://www.personalmicrocosms.com/zip/ra.zip", filename);

Console.WriteLine("file downloaded");
}
}
```

VB.NET:

```
Imports System
Imports System.Net
Imports System.Text

Class DownloadData_Method

    Public Shared Sub Main()
        Dim wc As WebClient = New WebClient
        Dim response As Byte() = wc.DownloadData("http://www.google.com")
        Console.WriteLine(Encoding.ASCII.GetString(response))
    End Sub
End Class
```

3- **OpenRead** ووظيفتها إنشاء Read Only Stream بين الزبون والServer لجلب بيانات من URL محدد وتخزينه في Stream Object بعد تمرير ال URL للموقع الذي تريد عرضه وباستخدام الميثود ReadLine نستطيع عرض البيانات المخزنة في ال Stream Object على شكل HTML Code .
ملاحظة : تستخدم الميثود Peek لمعرفة نهاية ال Stream Object .

C#:

```
using System;
using System.IO;
using System.Net;

class OpenRead_Method
{
public static void Main ()
{
WebClient wc = new WebClient();
string response;

Stream strm = wc.OpenRead("http://www.google.com");
StreamReader sr = new StreamReader(strm);

while(sr.Peek() > -1)
{
response = sr.ReadLine();
Console.WriteLine(response);
}
sr.Close();
}
}
```

VB.NET:

```
Imports System
Imports System.IO
Imports System.Net

Class OpenRead_Method

    Public Shared Sub Main()
        Dim wc As WebClient = New WebClient
        Dim response As String
        Dim strm As Stream = wc.OpenRead("http://www.google.com")
        Dim sr As StreamReader = New StreamReader(strm)
        While sr.Peek > -1
            response = sr.ReadLine
            Console.WriteLine(response)
        End While
        sr.Close()
    End Sub
End Class
```

ويحتوي ال **WebClient Class** على مجموعة من ال **Properties** والتي تستخدم لجلب معلومات عن ال Web Host مثل property ResponseHeaders والذي يستخدم

لجلب معلومات هامة عن ال web host مثل عدد ال Headers ونوع ال cash control
واسم ال Server و نوع ال Encoding المستخدم وغيرها من المعلومات الهامة،
ويستخدم كما يلي كمثال:

C#:

```
using System;
using System.Net;

class ResponseHeaders_property
{
public static void Main ()
{
WebClient wc = new WebClient();
byte[] response =
wc.DownloadData("http://www.google.com");
WebHeaderCollection whc = wc.ResponseHeaders;
Console.WriteLine("header count = {0}", whc.Count);
for (int i = 0; i < whc.Count; i++)
{
Console.WriteLine(whc.GetKey(i) + " = " + whc.Get(i));
}
}
}
```

VB.NET:

```
Imports System
Imports System.Net

Class ResponseHeaders_property

    Public Shared Sub Main()
        Dim wc As WebClient = New WebClient
        Dim response As Byte() = wc.DownloadData("http://www.google.com")
        Dim whc As WebHeaderCollection = wc.ResponseHeaders
        Console.WriteLine("header count = {0}", whc.Count)
        Dim i As Integer = 0
        While i < whc.Count
            Console.WriteLine(whc.GetKey(i) + " = " + whc.Get(i))
            System.Math.Min(System.Threading.Interlocked.Increment(i), i - 1)
        End While
    End Sub
End Class

//Output:
//header count = 6
//Cache-Control = private
//Content-Type = text/html
//Set-Cookie = PREF=ID=6ae22f44980c5d78…
//7JRA; expires=Sun, 17-Jan-2038 19:14:
//Server = GWS/2.1
//Transfer-Encoding = chunked
//Date = Wed, 23 Nov 2005 10:10:58 GMT
```

ثانيا : Uploading to Web Server

يدعم ال WebClient أربعة Methods لتحميل البيانات إلى ال Web Server وهي :

1- **OpenWrite** ويستخدم لإرسال Stream Data إلى ال Web Server وذلك بعد تمرير عنوان ال URL للملف والنص الذي نريد كتابته على ال Web Page طبعا يجب أن تملك الصلاحيات لذلك ويستخدم كما يلي كمثال :

C#:

```
using System;
using System.IO;
using System.Net;
method_class OpenWrite
{
public static void Main ()
      {
WebClient wc = new WebClient();
string data = "<h1>Welcome to My Page</h1>";
Stream strm = wc.OpenWrite("C:\\mypage.html");
StreamWriter sw = new StreamWriter(strm);
sw.WriteLine(data);
sw.Close();
strm.Close();
      }
}
```

VB.NET:

```
Imports System
Imports System.IO
Imports System.Net

Class OpenWrite_method

   Public Shared Sub Main()
      Dim wc As WebClient = New WebClient
      Dim data As String = "<h1>Welcome to My Page</h1>"
      Dim strm As Stream = wc.OpenWrite("C:\mypage.html")
      Dim sw As StreamWriter = New StreamWriter(strm)
      sw.WriteLine(data)
      sw.Close()
      strm.Close()
   End Sub
End Class
```

2 – **UploadData** ويستخدم لنقل محتويات مصفوفة من النوع Byte إلى ال Web Server وهذا يعني انك تستطيع من خلالها رفع أي نوع من البيانات مثل النص الصور الفيديو وغيره إلى ال web server بعد تحويلها إلى Byte Array ويستخدم كما يلي كمثال :

C#:

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

Method_class UploadData
{
public static void Main ()
{
WebClient wc = new WebClient();
string data = "This is The Text Before Converted it to Byte";
byte[] dataarray = Encoding.ASCII.GetBytes(data);
wc.UploadData("C:\\mydata.txt", dataarray);
}
}
```

VB.NET:

```
Imports System
Imports System.Net
Imports System.Text

Class UploadData_Method

    Public Shared Sub Main()
        Dim wc As WebClient = New WebClient
        Dim data As String = "This is The Text Before Converted it to Byte"
        Dim dataarray As Byte() = Encoding.ASCII.GetBytes(data)
        wc.UploadData("C:\mydata.txt", dataarray)
    End Sub
End Class
```

3- **UploadFile** وتستخدم هذه الميثود لرفع ملف من ال Local Computer إلى ال Web Host وهي بسطة الاستخدام جدا وتستخدم كما يلي كمثال :

C#:

```
using System;
using System.Net;

class UploadFile_Method
{
public static void Main ()
      {
WebClient wc = new WebClient();
wc.UploadFile("http://www.yoursite.com", "C:\\myfile.html");
      }
}
```

VB.NET:

```
Imports System
Imports System.Net

Class UploadFile_Method

    Public Shared Sub Main()
        Dim wc As WebClient = New WebClient
        wc.UploadFile("http://www.yoursite.com", "C:\myfile.html")
    End Sub
End Class
```

4- **UploadValues** وتستخدم لرفع **Collection** من البيانات والـ values الخاصة بها إلى الويب Server وذلك بعد تحويل الـ **Collection** إلى Byte Array ولتعريف **Collection** نستخدم الكلاس NameValueCollection الموجود في Name Space System.Collections.Specialized وبعد تعريفه نستخدم الميثود add لإضافة الـ Collection جديد.. وتستخدم كما يلي كمثال :

C#:

```
using System;
using System.Collections.Specialized;
using System.Net;
using System.Text;

class UploadValues_Method
{

public static void Main ()
{
WebClient wc = new WebClient();
NameValueCollection nvc = new NameValueCollection();
nvc.Add("firstname", "Fadi");
nvc.Add("lastname", "Abdel-qader");
byte[] response =
wc.UploadValues("http://localhost/mypage.aspx", nvc);
Console.WriteLine(Encoding.ASCII.GetString(response));
}
}
```

VB.NET:

```
Imports System
Imports System.Collections.Specialized
Imports System.Net
Imports System.Text

Class UploadValues_Method

    Public Shared Sub Main()
        Dim wc As WebClient = New WebClient
        Dim nvc As NameValueCollection = New NameValueCollection
        nvc.Add("firstname", "Fadi")
```

```
        nvc.Add("lastname", "Abdel-qader")
        Dim response As Byte() =
wc.UploadValues("http://localhost/mypage.aspx", nvc)
        Console.WriteLine(Encoding.ASCII.GetString(response))
    End Sub
End Class
```

ثالثا :المواضيع الأكثر تقدما في ال HTTP Programming:

يعتبر هذا الجزء من أهم الأجزاء في برمجة تطبيقات Web Client Applications والذي سوف نتحدث فيه عن استخدام كل من ال HttpWebRequest Class و ال HttpWebResponse Class :

1- استخدام HttpWebRequest Class :

يحتوي هذا الكلاس على مجموعة من ال Properties والتي تستخدم بشكل أساسي في تطبيقات ال Web Client Applications لإنشاء مثل :

1- استخدام خاصية ال Web Proxy : والتي نمرر فيها عنوان ال Proxy Server ورقم الPort حتى نستطيع التعامل مع ال HTTP Web Requests من خلف Proxy Server أو Firewall ويتم تعريف ال Proxy Server Prosperity كما يلي كمثال :

C#:

```
using System;
using System.Net;

class ProxyServer_Property
{

public static void Main ()
        {
HttpWebRequest hwr = (HttpWebRequest)WebRequest.Create(
"http://www.google.com");

WebProxy proxysrv = new
WebProxy("http://proxy1.server.net:8080");
hwr.Proxy = proxysrv;
        }
}
```

VB.NET:

```
Imports System
Imports System.Net

Class ProxyServer_Property

    Public Shared Sub Main()
        Dim hwr As HttpWebRequest =
CType(WebRequest.Create("http://www.google.com"), HttpWebRequest)
        Dim proxysrv As WebProxy = New
WebProxy("http://proxy1.server.net:8080")
        hwr.Proxy = proxysrv
    End Sub
End Class
```

نعرف في البداية ال HttpWebRequest Object ثم نعرف WebProxy Object من الكلاس webProxy ونسند له عنوان ال Proxy Server ورقم الPort وبعد ذلك نستطيع إسناده إلى أي اوبجكت باستخدام الخاصية Proxy التي تكون موجودة عادة في جميع HttpWebRequest Objects ..

2- استخدام ال HttpWebrequest لإرسال بيانات إلى الويب Server باستخدام ال Streams وتستخدم كما يلي كمثال :

C#:

```
HttpWebrequest hwr =
(HttpWebRequest)WebRequest.Create("http://localhost");
Stream strm = hwr.GetRequestStream();
StreamWriter sw = new StreamWriter(strm);
sw.WriteLine(data);
```

VB.NET:

```
Dim hwr As HttpWebrequest = CType(WebRequest.Create("http://localhost"),
HttpWebRequest)
Dim strm As Stream = hwr.GetRequestStream
Dim sw As StreamWriter = New StreamWriter(strm)
sw.WriteLine(data)
```

بعد تعريف ال HttpWebRequest Object نقوم بتعريف Stream Object ونسند له ال Request Stream من خلال الميثود GetRequestStream .

2 - استخدام HttpWebResponse Class:

تستخدم ال **HttpWebResponse Object** لإرجاع بيانات من الويب Server إلى ال Client حيث نستخدم الميثود **GetResponse** و الميثود **BeginGetResponse** لهذه العملية ولا يوجد فرق في وظيفة هذه الميثودس سوى أن **BeginGetResponse** تعتبر **asynchronous Method** .

يحتوي ال **HttpWebResponse Object** على عدد من ال **Properties** وهي :

1- **CharacterSet** : وتستخدم لتحديد نوع ال Character Set
2- **ContentEncoding** : وتستخدم لعملية ال encoding
3- **ContentLength** : وتستخدم لمعرفة حجم الرد
4- **ContentType** : لتحديد نوع ال Respone
5- **Cookies** : لتعامل مع ال Cookies ولستخدامها يجب أولا إنشاء ملف Cookie فارغ وتعريفه كما يلي كمثال :

C#:

```
HttpWebRequest hwr =
(HttpWebRequest)WebRequest.Create(http://www.amazon.com);
hwr.CookieContainer = new CookieContainer();
```

وذلك قبل ال HTTP Request ثم نسنده إليه كما يلي :

```
HttpWebResponse hwrsp = (HttpWebResponse)hwr.GetResponse();
hwrsp.Cookies = hwr.CookieContainer.GetCookies(hwr.RequestUri);
```

VB.NET:

```
Dim hwr As HttpWebRequest =
CType(WebRequest.Create("http://www.amazon.com"), HttpWebRequest)
hwr.CookieContainer = New CookieContainer

Dim hwrsp As HttpWebResponse = CType(hwr.GetResponse,
HttpWebResponse)
hwrsp.Cookies = hwr.CookieContainer.GetCookies(hwr.RequestUri)
```

6- **Headers** : لمعرفة ال HTTP Headers

7- **LastModified** : يرجع فيه وقت وتاريخ أخر تعديل

8- **Method** : لمعرفة الميثود والتي تستخدم في ال HTTP Response

9 – **ProtocolVersion** : لمعرفة ال HTTP Version

10 – **ResponseUri** : ال URL الخاص بServer

11 – **Server** : لمعرفة اسم الServer

12 – **StatusCode** : لمعرفة نوع ال Coding امستخدم

13 – **StatusDescription** : لإرجاع Text يحتوي على حالة ال HTTP

تحدثنا في الجزء السابق عن برمجة ال HTTP وبينا فيه كيفية التفاعل بين ال web server وال client ويعتبر هذا الجزء مكمل لما تحدثنا عنه سابقا، تتلخص وظيفة استخدام ال web services بإمكانية الاستفادة من ال Methods الموجودة بال web server داخل برنامج الزبون وباستخدام برتوكول ال SOAP وهو اختصار ل Simple Object Access Protocol يتم نقل ال Result من ال web Services server إلى ال Client بعد تحويلها إلى ال XML - extensible Markup Language حيث تنقل عبر برتوكول ال HTTP إلى جهاز الزبون والهدف من استخدامه هو تسهيل وصول ال Data من ال web server إلى ال Client من خلال ال firewalls والبيئات المختلفة إذ أن جميع بيئات الشبكات تدعم برتوكول ال HTTP والذي يعمل على الPort 80 . ولا تختلف لغة ال XML عن ال HTML إذ تستخدم نفس القواعد في ال HTML وهي مجموعة من ال Elements وال Attributes مثل ال </> <> لاكن تتميز بمرونة اكبر وكمثال عليها :

```
<myStuff>
<myName>FADI Abdel-qader</myName>
<myTelephone>+962796...</myTelephone>
<myEmail>fadi822000@yahoo.com</myEmail>
<myAge>23</myAge>
<mySex>M</mySex>
</myStuff>
```

ويتم استدعائها في الدوت نيت باستخدام System.xml Name Spaces حيث يتم قراءتها باستخدام الميثود Load الموجود في ال XmlDocument Class كما يلي :

C#:

```
using System.Xml;
// Then you can Read any XML File as Below:
XmlDocument xDoc = new XmlDocument();
xDoc.Load(@"C:\myinfo.xml");
XmlNodeList name = xDoc.GetElementsByTagName("myName");
XmlNodeList telephone = xDoc.GetElementsByTagName("myTelephone");
XmlNodeList email = xDoc.GetElementsByTagName("myEmail");
XmlNodeList age = xDoc.GetElementsByTagName("myAge");
XmlNodeList sex = xDoc.GetElementsByTagName("mySex");

MessageBox.Show(
"Name: " + name[0].InnerText +"\n"+
"Telephone: " + telephone[0].InnerText +"\n"+
"Email: "+ email[0].InnerText +"\n"+
"Age: "+ age[0].InnerText +"\n"+
"sex: "+ sex[0].InnerText +"\n"
```

VB.NET:

```
Dim xDoc As XmlDocument = New XmlDocument
xDoc.Load("C:\myinfo.xml")
Dim name As XmlNodeList = xDoc.GetElementsByTagName("myName")
Dim telephone As XmlNodeList =
xDoc.GetElementsByTagName("myTelephone")
Dim email As XmlNodeList = xDoc.GetElementsByTagName("myEmail")
Dim age As XmlNodeList = xDoc.GetElementsByTagName("myAge")
Dim sex As XmlNodeList = xDoc.GetElementsByTagName("mySex")
Msgbox("Name: " + name(0).InnerText + "" & Microsoft.VisualBasic.Chr(10)
& "" + "Telephone: " + telephone(0).InnerText + "" &
Microsoft.VisualBasic.Chr(10) & "" + "Email: " + email(0).InnerText + "" &
Microsoft.VisualBasic.Chr(10) & "" + "Age: " + age(0).InnerText + "" &
Microsoft.VisualBasic.Chr(10) & "" + "sex: " + sex(0).InnerText + "" &
Microsoft.VisualBasic.Chr(10) & "")
```

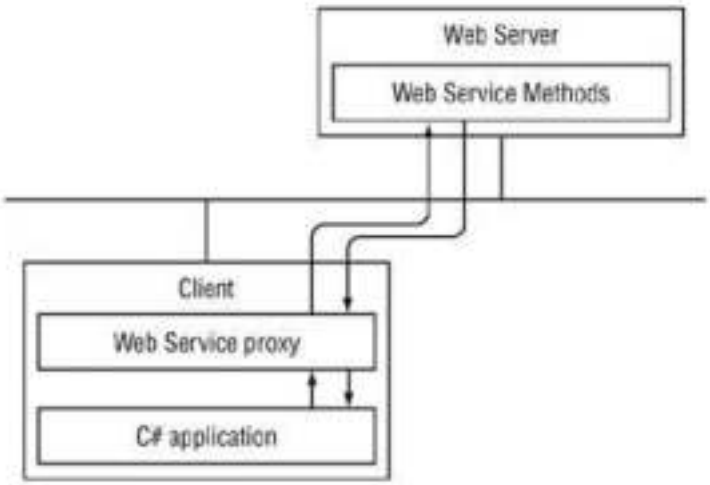
تمر عملية استخدام ال web services بثلاثة مراحل وهي :

1- The web service server : والذي يتم من خلاله إرسال واستقبال البيانات عبر برتوكول ال SOAP باستخدام ال IIS وال ASP.NET .

2- The proxy object : والذي يسمح للClient بإرسال و استقبال البيانات من وإلى ال web Services Server حيث يتم تعريفه في ال HttpWebRequest من خلال الكلاس WebProxy وهو ما بينته في الجزء السابق.

3- The client application : وهو الواجهة الخاصة بزبون والتي يتم ربطها بال Web Services Server

كما في الشكل التالي :



ولإنشاء web services server نقوم بعمل مشروع ASP.NET Web Services جديد
ونستدعي System.Web.Services Name Spaces ثم نقوم بتوريث الكلاس
WebService للكلاس الرئيسي للمشروع وكما يلي كمثال:

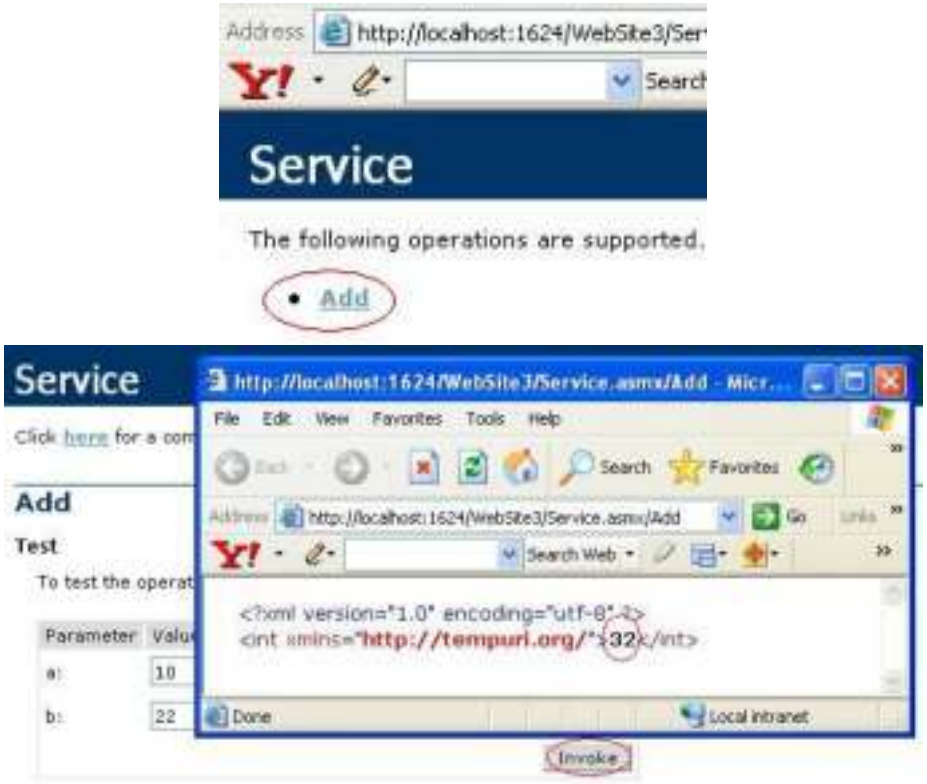
C#:

```
using System;
using System.Web;
using System.Web.Services;
using System.Web.Services.Protocols;

[WebService(Namespace = "http://my_url.com/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
public class Service : System.Web.Services.WebService
{
    public Service () {}
    [WebMethod]
  public int Add(int a, int b)
  {
   return a + b;
  }

    }
```

VB.NET:

```
Imports System
Imports System.Web
Imports System.Web.Services
Imports System.Web.Services.Protocols

<WebService(Namespace="http://my_url.com/")> _
<WebServiceBinding(ConformsTo=WsiProfiles.BasicProfile1_1)> _
Public Class Service
    Inherits System.Web.Services.WebService

    Public Sub New()
    End Sub

    <WebMethod()> _
    Public Function Add(ByVal a As Integer, ByVal b As Integer) As Integer
        Return a + b
    End Function
End Class
```

حيث يتم استقبال قيمتين A و B وبعد ذلك يقوم بإرجاع ناتج جمع القيمة الأولى مع القيمة الثانية إلى ال Client على شكل XML باستخدام برتوكول ال SOAP وكما يظهر في الشكل التالي :



ولإنشاء برنامج ال Client يجب أولا تحويل الكلاس السابق إلى Dll File و إرفاقه بال Client Resources ويتم استخدامه كما يلي :

C#:

```
using System;
class Client_side
{
   public static void Main(string[] argv)
   {
      My_main_class mm = new My_main_class();
      int x = Convert.ToInt16(argv[0]);
      int y = Convert.ToInt16(argv[1]);
      int sum = mm.Add(x, y);
      Console.WriteLine(sum);
   }
}
}
```

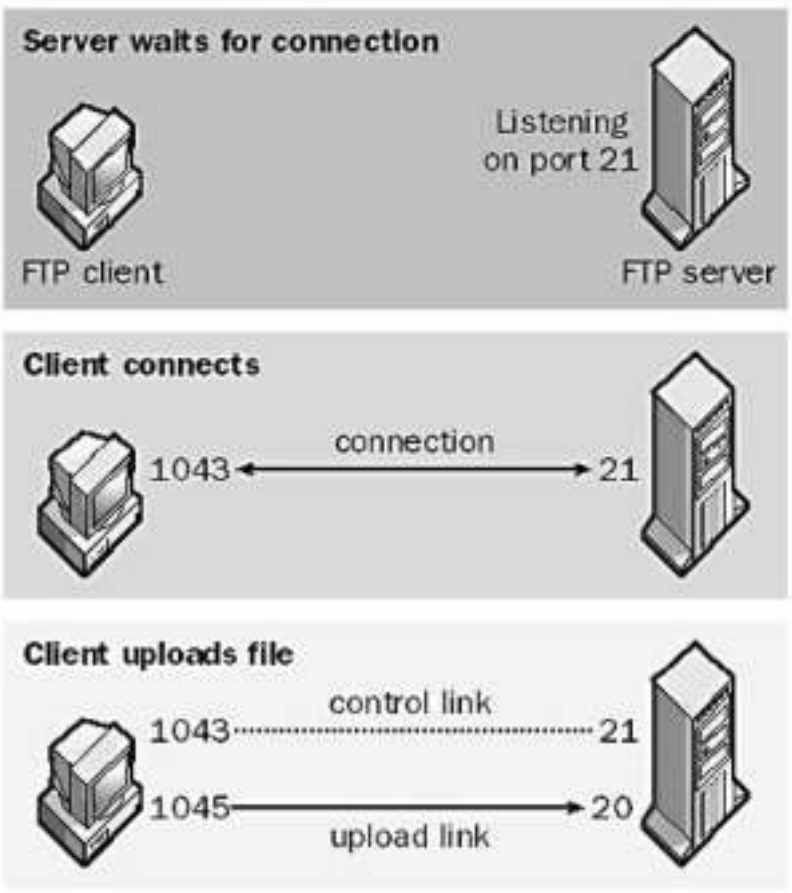
VB.NET:

```
Class Client_side

    Public Shared Sub Main(ByVal argv As String())
        Dim mm As My_main_class = New My_main_class
        Dim x As Integer = Convert.ToInt16(argv(0))
        Dim y As Integer = Convert.ToInt16(argv(1))
        Dim sum As Integer = mm.Add(x, y)
        Console.WriteLine(sum)
    End Sub
End Class
```

وهكذا بينا الأساسيات في ال Web services وسوف تجد كافة التفاصيل في النسخة الورقية من الكتاب...

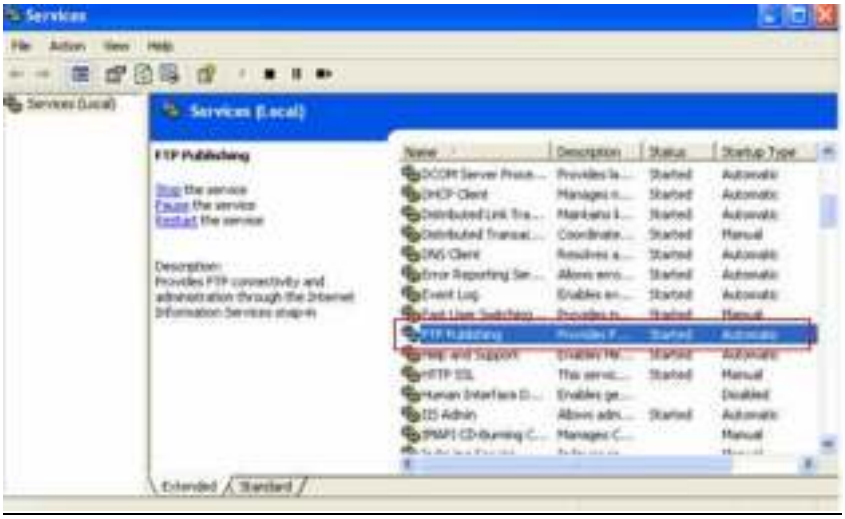
: FTP – File Transfer Protocol Programming 5.5

سوف نبدأ هنا بشرح برتوكول أخر من برتوكولات ال Application Layer وهو برتوكول ال FTP والذي يستخدم بشكل أساسي في عملية تنزيل downloading و رفع uploadingالملفات من و إلى ال FTP Server وكالعادة في اغلب برمجيات الشبكات و التي تعتمد على وجود Client/Server حيث يقوم الServer بتصنت على الPort المخصص لل FTP وهو الPort 21 باستخدام ال TCP Connection Oriented Protocolحيث يبقى الServer بوضع الانتظار لورود طلب من ال Client بإنشاء Session معه وبعد إجراء عمليات التحقق Authentication والتأكد من الصلاحيات يتم الموافقة على البدء بالجلسة حيث يتم تحديد رقم الPort والذي سوف يتم استقبال البيانات من خلاله ويتم الإرسال إلى جهاز الزبون عبر الPort 20 في الServer وتتضح هذه العملية كما في الشكل التالي :

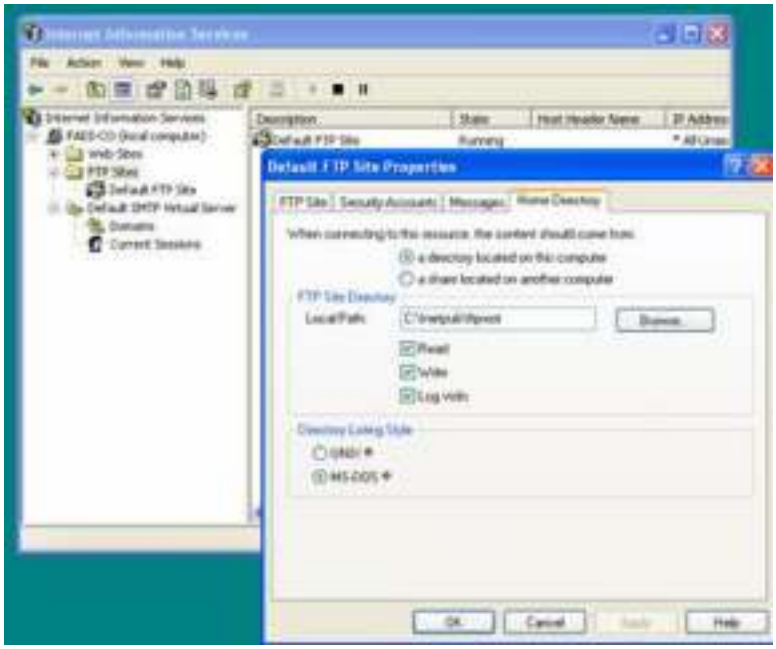


<u>ملاحظة:</u> لتفعيل خدمة ال FTP لديك بحيث يعمل جهازك ك FTP Server يجب أولا التأكد من أن ال FTP Services مثبتة لديك مع ال IIS و كما يظهر في الشكل التالي :

ومن ثم التأكد من تفعيلها ب Services من Control Panel ثم Administrative Tools ثم Services وكما يظهر في الشكل التالي :

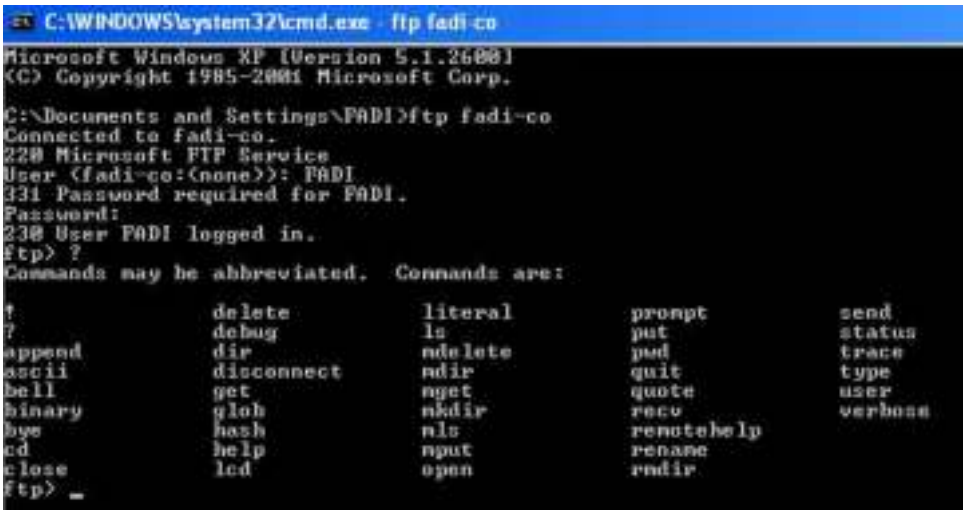


ثم التأكد منه في ال IIS بحيث يظهر كما في الشكل التالي :



أولا : FTP Commands

تشبه عملية الاتصال و الاستخدام لل FTP عملية ال Telnet إلى حد كبير حيث يدعم برتوكول ال FTP مجموعة من الأوامر والتي يتم من خلالها عملية التخاطب مع الServer أو مع ال Remote Host وتضح هذه العملية كما في الشكل التالي :



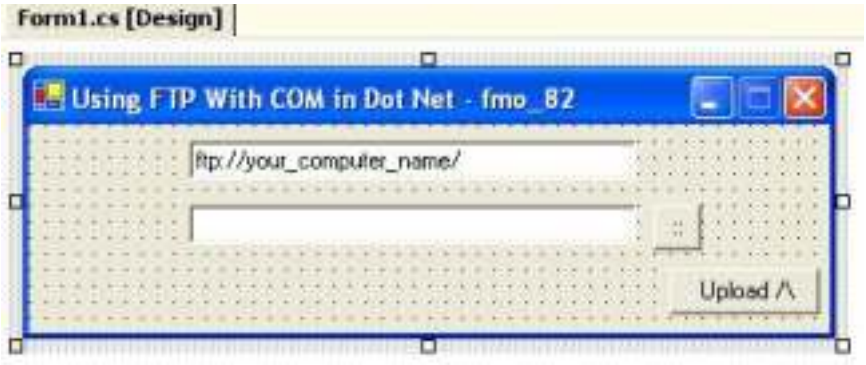
وهنا شرح لأهم ال FTP Commands:

USER <username> & PASS <password>	مطلوبة لعملية التحقق لإنشاء الجلسة
RETR <*filename*> أو RECV	ويستخدم لتنزيل ملف من الServer بعد تحديد اسم الملف
STOR <*filename*> أو SEND	ويستخدم لرفع الملف إلى الServer بعد تحديد اسم الملف
TYPE <*type indicator*>	لتحديد طبيعة أو هيئة البيانات التي يتم نقلها وكما يلي : -a ASCII –e EBCDIC – I for Binary - L<Byte Size> Data والذي سيتم

نقله	
لتحديد نوع الجلسة سواء Passive أو Active إذ انه في حالة ال Passive يتم تفعيل الاتصال فقط في حالة ورود أو رفع أي ملف من و إلى الServer .	PASV
لفحص حالة الاتصال و uploading & Downloading	Status أو STAT
وهي كما هو متعارف عليه في التعامل مع الملفات و المجلدات في نظام ال DOS	Delete , cd , mkdir , rename ...
لإنهاء الجلسة مع ال Remote Host	QUIT أو Close

ثانيا : التعامل مع ال FTP في الدوت نت باستخدام COM Components :

تدعم الدوت نيت استخدام ال FTP عبر ITC – Internet Transfer Control وهو جزء من ال COM Components Controls وللبدء قم بإنشاء New Windows Application كما في الشكل التالي :



ثم قم بإضافة Name Spaces التالية :

C#:

```
using System.IO;
using System.Reflection;
```

VB.NET:

```
imports System.IO
imports System.Reflection
```

ثم إضافة الكود التالي إلى ال Upload Button :

C#:

```
private void button1_Click(object sender, System.EventArgs e)
{
FileInfo thisFile = new FileInfo(tbFile.Text);
Type ITC;
object[] parameter= new object[2];
object ITCObject;
ITC = Type.GetTypeFromProgID("InetCtls.Inet");
ITCObject = Activator.CreateInstance(ITC);
parameter[0] = (string)tbServer.Text;
```

```
parameter[1] = (string)"PUT " + thisFile.FullName + " /" +
thisFile.Name;
ITC.InvokeMember("execute", BindingFlags.InvokeMethod,null, ITCObject,
parameter);}
```

VB.NET:

```
Private Sub button1_Click(ByVal sender As Object, ByVal e As
System.EventArgs)
    Dim thisFile As FileInfo = New FileInfo(tbFile.Text)
    Dim ITC As Type
    Dim parameter(2) As Object
    Dim ITCObject As Object
    ITC = Type.GetTypeFromProgID("InetCtls.Inet")
    ITCObject = Activator.CreateInstance(ITC)
    parameter(0) = CType(tbServer.Text, String)
    parameter(1) = CType("PUT ", String) + thisFile.FullName + " /" +
thisFile.Name
    ITC.InvokeMember("execute", BindingFlags.InvokeMethod, Nothing,
ITCObject, parameter)
End Sub
```

تم في البداية تعريف ال ITC من خلال ال Type Class والموجود ضمن Name Space
System.Reflection ثم عرفنا Array من النوع Object وذلك لاستخدامها في تمرير
اسم الملف و ال FTP Server إلى الميثود InvokeMember والموجودة ضمن ال ITC
... Control Object
سوف تجد الملف الذي سيتم رفعه في المجلد :
C:\Inetpub\ftproot

ثالثا : التعامل مع ال FTP في الدوت نت باستخدام ال Web Class :

يمكن برمجة ال FTP باستخدام web Class والموجودة ضمن Name Spaces System.Net وتشبه عملية التعامل معه كما في التعامل مع ال WebRequest و ال webResponse و التي تعاملنا معها في برمجة ال HTTP حيث يمكننا Classes الاستفادة منها لتعامل مع ال FTP Protocol وهي كما يلي :

- WebClient إذ تم دعم dot net Framework 2 استخدام الكلاس WebClient والذي يدعم التعامل مع ال FTP والذي يتم استدعائه من System.Net Name Spaces ويتم تعريفه كما يلي :

C#:

```
using System;
using System.Net;

namespace Web_Client
{
   class Program
   {
      public static void Main(string[] args)
      {
         string filename = "ftp://ms.com/files/dotnetfx.exe";
         WebClient client = new WebClient();
         client.DownloadFile(filename, "dotnetfx.exe");

      }

   }
}
```

VB.NET:

```
Imports System
Imports System.Net
Namespace Web_Client

   Class Program

      Public Shared Sub Main(ByVal args As String())
         Dim filename As String = "ftp://ms.com/files/dotnetfx.exe"
         Dim client As WebClient = New WebClient
         client.DownloadFile(filename, "dotnetfx.exe")
      End Sub
   End Class
End Namespace
```

- FtpRequestCreator ويستخدم لتسجيل وبدأ العمل مع ال FTP ويعرف كما يلي :

C#:
```
using System;
using System.Net;

namespace FTP
{
   public class FtpRequestCreator : IWebRequestCreate
   {
      public FtpRequestCreator()
      {
      }

      public System.Net.WebRequest Create(System.Uri uri)
      {
         return new FtpWebRequest(uri);
      }
   }
}
```

VB.NET:
```
Imports System
Imports System.Net
Namespace FTP

   Public Class FtpRequestCreator
      Implements IWebRequestCreate

      Public Sub New()
      End Sub

      Public Function Create(ByVal uri As System.Uri) As
System.Net.WebRequest
         Return New FtpWebRequest(uri)
      End Function
   End Class
End Namespace
```

- FtpWebRequest ويستخدم لعمل download or upload a file on an FTP server
ويتم تعريفها كما يلي :

C#:
```
using System;
using System.Net;

namespace FTP
{
   public class FtpWebRequest : WebRequest
   {
      private string username = "Fadi";
```

```
        internal string password = "fff";
        private Uri uri;
        private bool binaryMode = true;
        private string method = "GET";

        internal FtpWebRequest(Uri uri)
        {
            this.uri = uri;
        }

        public string Username
        {
            get { return username; }
            set { username = value; }
        }

        public string Password
        {
            set { password = value; }
        }

        public bool BinaryMode
        {
            get { return binaryMode; }
            set { binaryMode = value; }
        }

        public override System.Uri RequestUri
        {
            get { return uri; }
        }

        public override string Method
        {
            get { return method; }
            set { method = value; }
        }

        public override System.Net.WebResponse GetResponse()
        {
            FtpWebResponse response = new FtpWebResponse(this);
            return response;
        }
    }
}
```

VB.NET:

```
Imports System
Imports System.Net
Namespace FTP

    Public Class FtpWebRequest
        Inherits WebRequest
        Private username As String = "Fadi"
        Friend password As String = "fff"
        Private uri As Uri
        Private binaryMode As Boolean = True
        Private method As String = "GET"

        Friend Sub New(ByVal uri As Uri)
            Me.uri = uri
        End Sub

        Public Property Username() As String
            Get
                Return username
            End Get
            Set(ByVal value As String)
                username = value
            End Set
        End Property

        Public WriteOnly Property Password() As String
            Set(ByVal value As String)
                password = value
            End Set
        End Property

        Public Property BinaryMode() As Boolean
            Get
                Return binaryMode
            End Get
            Set(ByVal value As Boolean)
                binaryMode = value
            End Set
        End Property

        Public Overloads Overrides ReadOnly Property RequestUri() As
System.Uri
            Get
                Return uri
            End Get
        End Property

        Public Overloads Overrides Property Method() As String
            Get
                Return method
```

```
            End Get
            Set(ByVal value As String)
                method = value
            End Set
        End Property

        Public Overloads Overrides Function GetResponse() As
System.Net.WebResponse
            Dim response As FtpWebResponse = New FtpWebResponse(Me)
            Return response
        End Function
    End Class
End Namespace
```

- FtpWebResponse ويستخدم لعملية الرد من قبل الServer ويتم تعريفها كما يلي:

C#:

```
using System;
using System.IO;
using System.Net;
using System.Net.Sockets;

namespace FTP
{
    public class FtpWebResponse : WebResponse
    {
        private FtpWebRequest request;
        private FtpClient client;

        internal FtpWebResponse(FtpWebRequest request)
        {
            this.request = request;
        }
    }
}
```

VB.NET:

```
Imports System
Imports System.IO
Imports System.Net
Imports System.Net.Sockets
Namespace FTP

    Public Class FtpWebResponse
        Inherits WebResponse
        Private request As FtpWebRequest
        Private client As FtpClient

        Friend Sub New(ByVal request As FtpWebRequest)
            Me.request = request
        End Sub
    End Class
End Namespace
```

- FtpWebStream ويستخدم لتعريف ال Stream والذي سوف يستخدم لعملية النقل ويعرف بشكل مبدئي كما يلي :

C#:

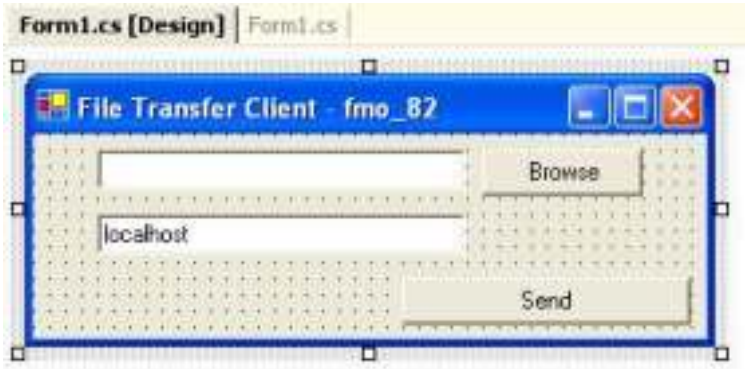
```
using System;
using System.IO;
using System.Net.Sockets;

namespace FTP
{
    internal class FtpWebStream : Stream
    {
        private FtpWebResponse response;
        private NetworkStream dataStream;

 public FtpWebStream(NetworkStream dataStream, FtpWebResponse
response)
        {
            this.dataStream = dataStream;
            this.response = response;
        }
    }
}
```

VB.NET:

```
Imports System
Imports System.IO
Imports System.Net.Sockets
Namespace FTP

    Friend Class FtpWebStream
        Inherits Stream
        Private response As FtpWebResponse
        Private dataStream As NetworkStream

        Public Sub New(ByVal dataStream As NetworkStream, ByVal response
As FtpWebResponse)
            Me.dataStream = dataStream
            Me.response = response
        End Sub
    End Class
End Namespace
```

رابعا : مثال تطبيقي لرفع ملف من جهاز Client إلى جهاز Server باستخدام ال Stream وال Socket:

في هذا الجزء سوف نقوم بإنشاء برنامجين Client / Server وبتعامل مع ال Stream Library سوف نقوم بتحويل الملف إلى Byte Array و إرساله عبر ال Stream باستخدام ال Socket و TCP Connection ، ولبرمجة الجزء الخاص بالإرسال أو ال Client قم بإنشاء مشروع جديد كما في الشكل التالي :



سوف نستخدم Name Spaces التالية :

C#:

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

VB.NET:

```
imports System.IO
imports System.Net
imports System.Net.Sockets
imports System.Text
```

في ال Send Button قم بكتابة الكود التالي :

C#:

```
try
{
Stream fileStream = File.OpenRead(textBox1.Text);
// Alocate memory space for the file
byte[] fileBuffer = new byte[fileStream.Length];
fileStream.Read(fileBuffer, 0, (int)fileStream.Length);
// Open a TCP Connection and send the data
TcpClient clientSocket = new TcpClient(textBox2.Text,8880);
NetworkStream networkStream = clientSocket.GetStream();
networkStream.Write(fileBuffer,0,fileBuffer.GetLength(0));
networkStream.Close();
}
catch (Exception ex){MessageBox.Show(ex.Message);}
```

VB.NET:

```
Try
Dim fileStream As Stream = File.OpenRead(textBox1.Text)
Dim fileBuffer(fileStream.Length) As Byte
 fileStream.Read(fileBuffer, 0, CType(fileStream.Length, Integer))
Dim clientSocket As TcpClient = New TcpClient(textBox2.Text, 8880)
Dim networkStream As NetworkStream = clientSocket.GetStream
 networkStream.Write(fileBuffer, 0, fileBuffer.GetLength(0))
 networkStream.Close
Catch ex As Exception
Msgbox(ex.Message)
End Try
```

قمنا في البداية بقراءة الملف الذي نود إرساله وتخزينه ب Stream Object وحتى نستطيع إرساله عبر ال Socket لابد من تحويله إلى مصفوفة من النوع Byte وقمنا بتسميته ب fileBuffer ثم تعبئته باستخدام الميثود Read والموجودة ضمن fileStream وبعد ذلك قمنا بإنشاء اتصال مع ال Server باستخدام ال TCP Connection حيث تم إرسال محتويات ال fileBuffer إلى ال Server باستخدام NetworkStream Class ...

ولبرمجة جزء Server وهو المسئول عن استقبال الملف وتخزينه قم بإنشاء مشروع جديد كما يظهر في الشكل التالي :

سوف نستخدم Name Spaces التالية :

C#:

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

VB.NET:

```
imports System.Threading
imports System.Net
imports System.Net.Sockets
imports System.Text
imports System.IO
```

ثم إضافة ال Method التالية وليكن اسمها handlerThread وكما يلي :

C#:

```
public void handlerThread()
{
Socket handlerSocket = (Socket)alSockets[alSockets.Count-1];
NetworkStream networkStream = new
NetworkStream(handlerSocket);
int thisRead=0;
int blockSize=1024;
Byte[] dataByte = new Byte[blockSize];
lock(this)
{
// Only one process can access
// the same file at any given time
Stream fileStream = File.OpenWrite(@"c:\upload");

while(true)
{
thisRead=networkStream.Read(dataByte,0,blockSize);
fileStream.Write(dataByte,0,thisRead);
if (thisRead==0) break;
fileStream.Close();
}
lbConnections.Items.Add("File Written");
handlerSocket = null;
}
```

VB.NET:

```
Public Sub handlerThread()
    Dim handlerSocket As Socket = CType(alSockets(alSockets.Count - 1),
Socket)
    Dim networkStream As NetworkStream = New
NetworkStream(handlerSocket)
    Dim thisRead As Integer = 0
    Dim blockSize As Integer = 1024
    Dim dataByte(blockSize) As Byte
    SyncLock Me
        Dim fileStream As Stream = File.OpenWrite("c:\upload")
        While True
            thisRead = networkStream.Read(dataByte, 0, blockSize)
            fileStream.Write(dataByte, 0, thisRead)
            If thisRead = 0 Then
                ' break
            End If
            fileStream.Close()
        End While
        lbConnections.Items.Add("File Written")
        handlerSocket = Nothing
    End SyncLock
End Sub
```

ثم قم بكتابة ميثود أخرى جديدة وذلك لفتح TCP Connection على الPort 8880
وهو افتراضي والتصنت عليها وليكن اسمها listenerThread وكما يلي :

C#:

```
public void listenerThread()
{
80);8TcpListener tcpListener = new TcpListener(8
tcpListener.Start();
while(true)
{
Socket handlerSocket = tcpListener.AcceptSocket();
if (handlerSocket.Connected)
{
lbConnections.Items.Add(handlerSocket.RemoteEndPoint.ToString() +
" connected.");
lock (this)
{
alSockets.Add(handlerSocket);
}
ThreadStart thdstHandler = new
ThreadStart(handlerThread);
Thread thdHandler = new Thread(thdstHandler);
thdHandler.Start();
}
}
}
```

VB.NET:

```
Public Sub listenerThread()
    Dim tcpListener As TcpListener = New TcpListener(8880)
    tcpListener.Start()
    While True
        Dim handlerSocket As Socket = tcpListener.AcceptSocket
        If handlerSocket.Connected Then
            lbConnections.Items.Add(handlerSocket.RemoteEndPoint.ToString +
" connected.")
            SyncLock Me
                alSockets.Add(handlerSocket)
            End SyncLock
            Dim thdstHandler As ThreadStart = New ThreadStart(handlerThread)
            Dim thdHandler As Thread = New Thread(thdstHandler)
            thdHandler.Start()
        End If
    End While
End Sub
```

ثم قم بإضافة الكود التالي إلى حدث بدأ تشغيل البرنامج Form Load :

C#:

```
private void Form1_Load(object sender, System.EventArgs e)
{
IPHostEntry IPHost = Dns.GetHostByName(Dns.GetHostName());

lbConnections.Text = "My IP address is " +
IPHost.AddressList[0].ToString();

alSockets = new ArrayList();

Thread thdListener = new Thread(new ThreadStart(listenerThread));
thdListener.Start();}
```

VB.NET:

```
Private Sub Form1_Load(ByVal sender As Object, ByVal e As
System.EventArgs)
    Dim IPHost As IPHostEntry = Dns.GetHostByName(Dns.GetHostName)
    lbConnections.Text = "My IP address is " + IPHost.AddressList(0).ToString
    alSockets = New ArrayList
    Dim thdListener As Thread = New Thread(New
ThreadStart(listenerThread))
    thdListener.Start()
End Sub
```

باستخدام ال Thread تم تنفيذ ال listenerThread Method والتي قمنا فيها بتعريف ال tcpListener وتفعيله على الPort 8880 حيث سيتم قبول أي طلب يأتي من ال Client على هذا الPort وبعد ذلك استدعاء الميثود handlerThread والتي سيتم فيها استقبال ال Stream Data وتخزينها في Byte Array ثم قراءتها وتخزينها في المكان المحدد وباستخدام ال fileStream.Write حيث مررنا له ال Stream والذي يحتوي على اسم الملف thisRead وال dataByte Array ...

Chapter 6
Network Security Programming

Network Security Programming

Dot Net Security Namespaces Overview

1. Cryptography
2. Permission

بسم الله الرحمن الرحيم

6.1 : Network Security Programming :

تتلخص الفكرة من الأمن بحماية البيانات من الدخول غير المخول unauthorized Access
باستخدام عدة أساليب وأهمها :

- Data Encryption & Decryption التشفير وفك التشفير
- Authentications التحقق من هوية الشخص مرسل الرسالة
- Set Policies & Permissions تحديد وتنفيذ السياسات و الصلاحيات

دعمت في الدوت نيت جميع أساليب الحماية التي ذكرناها سابقا باستخدام ال Security Namespaces والتي تحتوي على مجموعة ضخمة من المكتبات الفرعية وهي كما في الشكل التالي

أولا : Cryptography Namespace Overview :

Cryptography in .NET: وهي المكتبة التي تهتم بكل ما يخص عمليات تشفير وفك تشفير البيانات من Clear Text إلى Cipher Text وبالعكس وتستخدم بشكل أساسي لتشفير البيانات قبل عملية الإرسال وفك تشفيرها عند الاستلام ، ونستطيع تقسيم طرق التشفير فيها إلى ثلاثة أقسام رئيسية هي:

A- Symmetric algorithms: الأسلوب المتماثل وفيه يستخدم المفتاح السري ذاته لعملية التشفير وفك التشفير وهي طريقة سريعة لإجراء عملية التشفير وفك التشفير لا كنها ليست آمنة كطريقة الغير المتماثلة ودعمت الدوت نيت التشفير المتماثل بمجموعة من ال Algorithms Classes وهي:

- الكلاس الذي يدعم التشفير باستخدام ال DES-Data Encryption Standard : DESCryptoServiceProvider
- الكلاس الذي يدعم RC2 Algorithms : RC2CryptoServiceProvider
- الكلاس الذي يدعم Rijndael Managed Algorithms : RijndaelManaged

الطريقة المعتادة في التشفير بالأسلوب المتماثل هي تشفير الرسالة وإرسالها عبر الشبكة لاكن باستخدام هذه الطريقة فإن نسبة الخطأ التي قد تكون عالية جدا وقد نفقد بعض هذه البيانات مما يؤدي إلى فقد الرسالة أو قد تسرق وتجرى عليها عمليات لمحاولة فك الشيفرة ناهيك عن الحجم الهائل التي قد تحجزه من ال Network Bandwidth..وتم حل هذه المشكلة بجعل عملية التشفير تتم على مستوى ال Stream نفسه ويستخدم لهذه العملية ال **CryptoStream Class** حيث يتم استخدام مفتاحين لتشفير مفتاح التشفير Encryption Kay ومفتاح لفك التشفير **IV** Installation Victor Decryption ويشترط استخدام نفس المفتاحين في عملية التشفير وفك التشفير ويستخدم الكلاس السابق مع ال **MemoryStream** أو **FileStream** حيث نمرر له ال Stream Data ونوع التشفير سواء DES أو TripleDES أو

RC2 ، وكمثال سوف نستخدم ال TripleDES إذيجب أن يتكون كلا المفتاحين من 16 Bits ...

Symmetric Stream Encryption Example:

C#:

```
byte[] Key = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09,0x10,
0x11, 0x12, 0x13, 0x14, 0x15, 0x16};
byte[] IV =  {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09,0x10,
0x11, 0x12, 0x13, 0x14, 0x15, 0x16};

string phrase = msg.Text;
MemoryStream ms = new MemoryStream();

TripleDESCryptoServiceProvider tdes = new
TripleDESCryptoServiceProvider();
CryptoStream csw = new CryptoStream(ms,tdes.CreateEncryptor(Key, IV),
CryptoStreamMode.Write);

csw.Write(Encoding.ASCII.GetBytes(phrase), 0, phrase.Length);
csw.FlushFinalBlock();

byte[] cryptdata = ms.GetBuffer();

textBox1.Text=Encoding.ASCII.GetString(cryptdata, 0, (int)ms.Length);
```

VB.NET:

```
Dim Key As Byte() = {&H1, &H2, &H3, &H4, &H5, &H6, &H7, &H8, &H9,
&H10, &H11, &H12, &H13, &H14, &H15, &H16}
Dim IV As Byte() = {&H1, &H2, &H3, &H4, &H5, &H6, &H7, &H8, &H9, &H10,
&H11, &H12, &H13, &H14, &H15, &H16}

Dim phrase As String = msg.Text
Dim ms As MemoryStream = New MemoryStream()

Dim tdes As TripleDESCryptoServiceProvider = New
TripleDESCryptoServiceProvider()
Dim csw As CryptoStream = New CryptoStream(ms,
tdes.CreateEncryptor(Key, IV), CryptoStreamMode.Write)

csw.Write(Encoding.ASCII.GetBytes(phrase), 0, phrase.Length)
csw.FlushFinalBlock()

Dim cryptdata As Byte() = ms.GetBuffer()

textBox1.Text=Encoding.ASCII.GetString(cryptdata, 0, CInt(ms.Length))
```

Symmetric Stream Decryption Example:

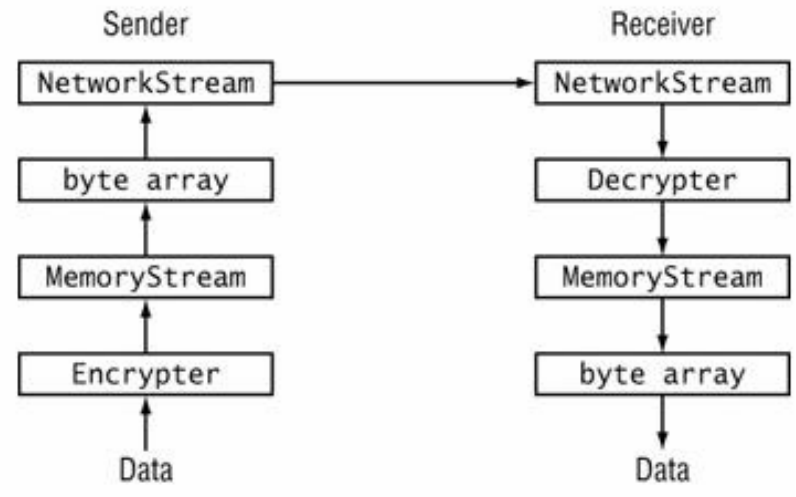
C#:

```
byte[] Keyy = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09,0x10,
0x11, 0x12, 0x13, 0x14, 0x15, 0x16};
byte[] IVv =  {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09,0x10,
0x11, 0x12, 0x13, 0x14, 0x15, 0x16};
ms.Position = 0;
byte[] data = new byte[1024];
CryptoStream csr = new CryptoStream(ms,tdes.CreateDecryptor(Keyy,
IVv),CryptoStreamMode.Read);
int recv = csr.Read(data, 0, data.Length);
string newphrase = Encoding.ASCII.GetString(data, 0, recv);
textBox1.Text=newphrase;
```

VB.NET:

```
Dim Keyy As Byte() = {&H1, &H2, &H3, &H4, &H5, &H6, &H7, &H8, &H9,
&H10, &H11, &H12, &H13, &H14, &H15, &H16}
Dim IVv As Byte() = {&H1, &H2, &H3, &H4, &H5, &H6, &H7, &H8, &H9,
&H10, &H11, &H12, &H13, &H14, &H15, &H16}
ms.Position = 0
Dim data As Byte() = New Byte(1023) {}
Dim csr As CryptoStream = New CryptoStream(ms,
tdes.CreateDecryptor(Keyy, IVv), CryptoStreamMode.Read)
Dim recv As Integer = csr.Read(data, 0, data.Length)
Dim newphrase As String = Encoding.ASCII.GetString(data, 0, recv)
textBox1.Text=newphrase
```

في برمجيات الشبكات نقوم في البداية بتشفير البيانات المرسلة باستخدام أي من الأساليب السابقة لتشفير ثم نحول البيانات المشفرة إلى Stream لإرسالها عبر ال Socket باستخدام ال Network Stream ، ثم يقوم الطرف المستقبل باستقبال الرسالة باستخدام ال Network Stream عبر ال Socket ، عملية فك التشفير تكون كما هي الخوارزمية المستخدمة ثم تحمل الرسالة إلى ال memory stream وتخزن في Byte Array عندها يمكن أن تحول إلى رسالة مرة أخرى وكما في الشكل التالي:



B- Asymmetric algorithms: الأسلوب الغير متماثل وهو أكثر أمانا من الأسلوب المتماثل إذ تشفر البيانات باستخدام مفتاح عام Public Kay ولفك التشفير يستخدم مفتاح خاص Private Kay ويكون هناك علاقة بين المفتاحين ويستخدم 128 Bits لتشفير وهو أفضل أساليب التشفير للبيانات ودعمت الدوت نيت التشفير الغير متماثل والذي يدعم تشفير المفتاح الخاص Private Kay باستخدام Tow Algorithms Classes وهي:

1- DSACryptoServiceProvider for Digital Signature Algorithm

التواقيع الرقمية :والهدف منها التحقق من هوية الشخص مرسل الرسالة وكمثال يقوم المرسل بتوليد ملخص لرسالة باستخدام الHash Function وبعد ذلك يقوم بتشفير ملخص الرسالة الذي تم توليده لتكوين المفتاح الخاص والذي سيستخدم كتوقيع رقمي للمرسل ثم يرسل المفتاح العام مع الرسالة، أما بما يتعلق بالمستلم فيقوم بفك تشفير الملخص باستخدام المفتاح العام ويجب أن يتم ذلك باستخدام نفس الخوارزمية التي اتبعها المرسل في تشفير الملخص، فإذا كان ملخص الرسالة التي ولدها المستلم هي نفسها التي ولدها المرسل عندها يتحقق من أن الشخص مرسل الرسالة هو نفسه .
في البداية سوف ننشئ instance من ال DSACryptoServiceProvider لتوليد المفتاح العام والخاص ثم نكون ال Hash sign Value ونخزنه في Byte Array ولفحصه نولد hash sign value جديد ونقارنه بالسابق فإذا تشابها عندها نقرر أن الشخص هو نفسه صاحب الرسالة المرسلة وكما يلي:

C#:

```
using System;
using System.Security.Cryptography;

class DSACSPSample
{

        static void Main()
        {
                try
                {
//Create a new instance of DSACryptoServiceProvider to generate
//a new key pair.
DSACryptoServiceProvider DSA = new DSACryptoServiceProvider();

//The hash value to sign.
byte[] HashValue =
{59,4,248,102,77,97,142,201,210,12,224,93,25,41,100,197,213,134,130,135}
;
//The value to hold the signed value.
byte[] SignedHashValue = DSASignHash(HashValue,
DSA.ExportParameters(true), "SHA1");

//Verify the hash and display the results.
if(DSAVerifyHash(HashValue, SignedHashValue, DSA.ExportParameters(false),
"SHA1"))
{Console.WriteLine("The hash value was verified.");}
else
{Console.WriteLine("The hash value was not verified.");}}
catch(ArgumentNullException e)
{Console.WriteLine(e.Message);}
```

```
}
public static byte[] DSASignHash(byte[] HashToSign, DSAParameters
DSAKeyInfo, string HashAlg)
{
        try
        {
//Create a new instance of DSACryptoServiceProvider.
DSACryptoServiceProvider DSA = new DSACryptoServiceProvider();

//Import the key information.
DSA.ImportParameters(DSAKeyInfo);

//Create an DSASignatureFormatter object and pass it the
//DSACryptoServiceProvider to transfer the private key.
DSASignatureFormatter DSAFormatter = new DSASignatureFormatter(DSA);

//Set the hash algorithm to the passed value.
DSAFormatter.SetHashAlgorithm(HashAlg);

//Create a signature for HashValue and return it.
return DSAFormatter.CreateSignature(HashToSign);
}
catch(CryptographicException e)
{Console.WriteLine(e.Message);return null;}
}
```

VB.NET:

```
Imports System
Imports System.Security.Cryptography

Friend Class DSACSPSample

   Shared Sub Main()
      Try
         'Create a new instance of DSACryptoServiceProvider to generate
         'a new key pair.
         Dim DSA As DSACryptoServiceProvider = New
DSACryptoServiceProvider()

         'The hash value to sign.
         Dim HashValue As Byte() = {59, 4, 248, 102, 77, 97, 142, 201, 210,
12, 224, 93, 25, 41, 100, 197, 213, 134, 130, 135}
         'The value to hold the signed value.
         Dim SignedHashValue As Byte() = DSASignHash(HashValue,
DSA.ExportParameters(True), "SHA1")

         'Verify the hash and display the results.
         If DSAVerifyHash(HashValue, SignedHashValue,
DSA.ExportParameters(False), "SHA1") Then
            Console.WriteLine("The hash value was verified.")
         Else
```

```
            Console.WriteLine("The hash value was not verified.")
        End If
    Catch e As ArgumentNullException
        Console.WriteLine(e.Message)
    End Try
End Sub
Public Shared Function DSASignHash(ByVal HashToSign As Byte(), ByVal
DSAKeyInfo As DSAParameters, ByVal HashAlg As String) As Byte()
    Try
        'Create a new instance of DSACryptoServiceProvider.
        Dim DSA As DSACryptoServiceProvider = New
DSACryptoServiceProvider()

        'Import the key information.
        DSA.ImportParameters(DSAKeyInfo)

        'Create an DSASignatureFormatter object and pass it the
        'DSACryptoServiceProvider to transfer the private key.
        Dim DSAFormatter As DSASignatureFormatter = New
DSASignatureFormatter(DSA)

        'Set the hash algorithm to the passed value.
        DSAFormatter.SetHashAlgorithm(HashAlg)

        'Create a signature for HashValue and return it.
        Return DSAFormatter.CreateSignature(HashToSign)
    Catch e As CryptographicException
        Console.WriteLine(e.Message)
        Return Nothing
    End Try
End Function
```

C#:

```
public static bool DSAVerifyHash(byte[] HashValue, byte[] SignedHashValue,
DSAParameters DSAKeyInfo, string HashAlg)
{
    try
    {
//Create a new instance of DSACryptoServiceProvider.
DSACryptoServiceProvider DSA = new DSACryptoServiceProvider();

//Import the key information.
DSA.ImportParameters(DSAKeyInfo);

//Create an DSASignatureDeformatter object and pass it the
//DSACryptoServiceProvider to transfer the private key.
DSASignatureDeformatter DSADeformatter = new
DSASignatureDeformatter(DSA);

//Set the hash algorithm to the passed value.
```

```
DSADeformatter.SetHashAlgorithm(HashAlg);

//Verify signature and return the result.
return DSADeformatter.VerifySignature(HashValue, SignedHashValue);
        }
catch(CryptographicException e){Console.WriteLine(e.Message);return
false;}}}
```

VB.NET:

```
Public Shared Function DSAVerifyHash)ByVal HashValue As Byte ،()ByVal
SignedHashValue As Byte ،()ByVal DSAKeyInfo As DSAParameters ،ByVal
HashAlg As String (As Boolean
Try
'Create a new instance of DSACryptoServiceProvider.
Dim DSA As DSACryptoServiceProvider = New DSACryptoServiceProvider()

'Import the key information .
DSA.ImportParameters(DSAKeyInfo(

'Create an DSASignatureDeformatter object and pass it the
'DSACryptoServiceProvider to transfer the private key.
Dim DSADeformatter As DSASignatureDeformatter = New
DSASignatureDeformatter(DSA(

'Set the hash algorithm to the passed value.
DSADeformatter.SetHashAlgorithm(HashAlg(

'Verify signature and return the result .
Return DSADeformatter.VerifySignature(HashValue, SignedHashValue(
Catch e As CryptographicException
        Console.WriteLine(e.Message(
        Return False
End Try
End Function
```

2- RSACryptoServiceProvider

ويستخدم في إجراء التشفير وفك التشفير الغير متماثل وهو non inherited Class في البداية سوف ننشئ instance جديد من ال RSACryptoServiceProvider وذلك لتوليد المفتاح العام والخاص ونرفق المفتاح العام مع الرسالة ومن ثم يقوم المستلم بفك الرسالة باستخدام المفتاح الخاص وتتم كما في الشكل التالي:

تشفر الرسالة باستخدام مفتاح عام وخاص (العام يرسل مع الرسالة) والخاص يتفق عليه الطرفان المرسل والمستقبل

وهنا مثال توضيحي لطريقة التشفير وفك التشفير باستخدام ال RSA Algorithm :

C#:

```
using System;
using System.Security.Cryptography;
using System.Text;

class RSACSPSample
{

        static void Main()
        {
                try
                {
//Create a UnicodeEncoder to convert between byte array and string.
UnicodeEncoding ByteConverter = new UnicodeEncoding();

//Create byte arrays to hold original, encrypted, and decrypted data.
        byte[] dataToEncrypt = ByteConverter.GetBytes("Data to Encrypt");
        byte[] encryptedData;
        byte[] decryptedData;

//Create a new instance of RSACryptoServiceProvider to generate
//public and private key data.
        RSACryptoServiceProvider RSA = new RSACryptoServiceProvider();

//Pass the data to ENCRYPT, the public key information
//(using RSACryptoServiceProvider.ExportParameters(false),
//and a boolean flag specifying no OAEP padding.
```

```
encryptedData = RSAEncrypt(dataToEncrypt,RSA.ExportParameters(false),
false);

//Pass the data to DECRYPT, the private key information
//(using RSACryptoServiceProvider.ExportParameters(true),
//and a boolean flag specifying no OAEP padding.
decryptedData = RSADecrypt(encryptedData,RSA.ExportParameters(true),
false);

//Display the decrypted plaintext to the console.
Console.WriteLine("Decrypted plaintext: {0}",
ByteConverter.GetString(decryptedData));
        }
catch(ArgumentNullException) {Console.WriteLine("Encryption failed.");}
        }
```

VB.NET:

```
Imports System
Imports System.Security.Cryptography
Imports System.Text

Friend Class RSACSPSample

    Shared Sub Main()
        Try
            'Create a UnicodeEncoder to convert between byte array and string.
            Dim ByteConverter As UnicodeEncoding = New UnicodeEncoding()

            'Create byte arrays to hold original, encrypted, and decrypted data.
            Dim dataToEncrypt As Byte() = ByteConverter.GetBytes("Data to
Encrypt")
            Dim encryptedData As Byte()
            Dim decryptedData As Byte()

            'Create a new instance of RSACryptoServiceProvider to generate
            'public and private key data.
            Dim RSA As RSACryptoServiceProvider = New
RSACryptoServiceProvider()

            'Pass the data to ENCRYPT, the public key information
            '(using RSACryptoServiceProvider.ExportParameters(false),
            'and a boolean flag specifying no OAEP padding.
            encryptedData = RSAEncrypt(dataToEncrypt,
RSA.ExportParameters(False), False)

            'Pass the data to DECRYPT, the private key information
            '(using RSACryptoServiceProvider.ExportParameters(true),
            'and a boolean flag specifying no OAEP padding.
            decryptedData = RSADecrypt(encryptedData,
RSA.ExportParameters(True), False)
```

```
            'Display the decrypted plaintext to the console.
            Console.WriteLine("Decrypted plaintext: {0}",
ByteConverter.GetString(decryptedData))
        Catch e1 As ArgumentNullException
            Console.WriteLine("Encryption failed.")
        End Try
    End Sub
```

ننشئ الميثود التي ستقوم بتشفير الرسالة:

C#:

```
static public byte[] RSAEncrypt(byte[] DataToEncrypt, RSAParameters
RSAKeyInfo, bool DoOAEPPadding)
{
        try{
//Create a new instance of RSACryptoServiceProvider.
RSACryptoServiceProvider RSA = new RSACryptoServiceProvider();

//Import the RSA Key information. This only needs
//to include the public key information.
RSA.ImportParameters(RSAKeyInfo);

//Encrypt the passed byte array and specify OAEP padding.
//OAEP padding is only available on Microsoft Windows XP or
//later.
return RSA.Encrypt(DataToEncrypt, DoOAEPPadding);
                }
//Catch and display a CryptographicException
//to the console.
catch(CryptographicException e){Console.WriteLine(e.Message);return null;}
}
```

VB.NET:

```
Shared Public Function RSAEncrypt)ByVal DataToEncrypt As Byte ،()ByVal
RSAKeyInfo As RSAParameters ،ByVal DoOAEPPadding As Boolean (As Byte()
        Try
'Create a new instance of RSACryptoServiceProvider.
Dim RSA As RSACryptoServiceProvider = New RSACryptoServiceProvider()
'Import the RSA Key information. This only needs
'to include the public key information.
RSA.ImportParameters(RSAKeyInfo(
'Encrypt the passed byte array and specify OAEP padding  .
'OAEP padding is only available on Microsoft Windows XP or
'later  .
Return RSA.Encrypt(DataToEncrypt, DoOAEPPadding(
'Catch and display a CryptographicException
'to the console.
Catch e As CryptographicException
        Console.WriteLine(e.Message(
        Return Nothing
End Try
End Function
```

ننشئ الميثود التي ستقوم بفك تشفير الرسالة:

C#:

```
static public byte[] RSADecrypt(byte[] DataToDecrypt, RSAParameters
RSAKeyInfo,bool DoOAEPPadding)
{
try
{
//Create a new instance of RSACryptoServiceProvider.
RSACryptoServiceProvider RSA = new RSACryptoServiceProvider();
//Import the RSA Key information. This needs
//to include the private key information.
RSA.ImportParameters(RSAKeyInfo);

//Decrypt the passed byte array and specify OAEP padding.
//OAEP padding is only available on Microsoft Windows XP or
//later.
return RSA.Decrypt(DataToDecrypt, DoOAEPPadding);
                }
//Catch and display a CryptographicException
//to the console.
catch(CryptographicException e){Console.WriteLine(e.ToString());return null;}
}}
```

VB.NET:

```
Shared Public Function RSADecrypt)ByVal DataToDecrypt As Byte ،()ByVal
RSAKeyInfo As RSAParameters ،ByVal DoOAEPPadding As Boolean (As Byte()
Try
'Create a new instance of RSACryptoServiceProvider.
Dim RSA As RSACryptoServiceProvider = New RSACryptoServiceProvider()
'Import the RSA Key information. This needs
'to include the private key information.
RSA.ImportParameters(RSAKeyInfo(

'Decrypt the passed byte array and specify OAEP padding  .
'OAEP padding is only available on Microsoft Windows XP or
'later  .
Return RSA.Decrypt(DataToDecrypt, DoOAEPPadding(
'Catch and display a CryptographicException
'to the console.
Catch e As CryptographicException
        Console.WriteLine(e.ToString(()
        Return Nothing
End Try
End Function
```

C- Hashing algorithms: وهو أقوى الأساليب البرمجية لتشفير البيانات إذ يستخدم فيه 512 bits algorithm كحد أقصى بدلا من 128 bits باستخدام Message MAC Digest Algorithms وهنا لن تستطيع فك تشفير الرسالة وإرجاعها إلى حالتها السابقة ويستخدم بشكل أساسي لتوليد ال Passwords وفي توليد التواقيع الرقمية Digital Signature وفي اغلب الحالات تستخدم لتخزين كلمة المرور Password في ال Database بشكل امن.

ويستخدم ال SHA1Managed و ال SHA256Managed وال SHA384Managedوال SHA512Managed لتعريف Hash Object ومنه نستخدم ال ComputeHash Method لتوليد ال hash code وتخزينه في byte Array وكما يلي كمثال:

C#:

```
SHA1Managed shaM1 = new SHA1Managed ();
byte[] my_kay1= ASCIIEncoding.ASCII.GetBytes("convert this text to hash code");
byte[] hashed_kay1 = shaM1.ComputeHash(my_kay1);
MessageBox.Show(ASCIIEncoding.ASCII.GetString(hashed_kay1));

SHA256Managed shaM2 = new SHA256Managed();
byte[] my_kay2= ASCIIEncoding.ASCII.GetBytes("convert this text to hash code");
byte[] hashed_kay2 = shaM2.ComputeHash(my_kay2);
MessageBox.Show(ASCIIEncoding.ASCII.GetString(hashed_kay2));

SHA384Managed shaM3 = new SHA384Managed ();
byte[] my_kay3= ASCIIEncoding.ASCII.GetBytes("convert this text to hash code");
byte[] hashed_kay3 = shaM3.ComputeHash(my_kay3);
MessageBox.Show(ASCIIEncoding.ASCII.GetString(hashed_kay3));

SHA512Managed shaM4 = new SHA512Managed ();
byte[] my_kay4= ASCIIEncoding.ASCII.GetBytes("convert this text to hash code");
byte[] hashed_kay4 = shaM4.ComputeHash(my_kay4);
MessageBox.Show(ASCIIEncoding.ASCII.GetString(hashed_kay4));
```

VB.NET:

```
Dim shaM1 As SHA1Managed = New SHA1Managed
Dim my_kay1 As Byte() = ASCIIEncoding.ASCII.GetBytes("convert this text to hash code")
Dim hashed_kay1 As Byte() = shaM1.ComputeHash(my_kay1)
Msgbox(ASCIIEncoding.ASCII.GetString(hashed_kay1))
Dim shaM2 As SHA256Managed = New SHA256Managed
Dim my_kay2 As Byte() = ASCIIEncoding.ASCII.GetBytes("convert this text to hash code")
Dim hashed_kay2 As Byte() = shaM2.ComputeHash(my_kay2)
Msgbox(ASCIIEncoding.ASCII.GetString(hashed_kay2))
Dim shaM3 As SHA384Managed = New SHA384Managed
Dim my_kay3 As Byte() = ASCIIEncoding.ASCII.GetBytes("convert this text to hash code")
Dim hashed_kay3 As Byte() = shaM3.ComputeHash(my_kay3)
Msgbox(ASCIIEncoding.ASCII.GetString(hashed_kay3))
```

```
Dim shaM4 As SHA512Managed = New SHA512Managed
Dim my_kay4 As Byte() = ASCIIEncoding.ASCII.GetBytes("convert this text to hash code")
Dim hashed_kay4 As Byte() = shaM4.ComputeHash(my_kay4)
Msgbox(ASCIIEncoding.ASCII.GetString(hashed_kay4))
```

ثانيا : Permission Namespace Overview :

وتدعم ال Permission Namespace في الدوت نيت ثلاثة أنواع من الصلاحيات وهي ال Socket permissions وال Identity Permissions وال Role- based permissions ...

Socket Permission: وتمكنك من تحديد صلاحيات استخدام ال Socket في برمجيات الشبكات باستخدام SocketPermission و SocketPermissionAttribute ضمن ال System.Net وال System.Security.Permissions Namespaces وكمثال نستطيع منع Client Host Address معين من الاتصال مع ال Listener Application ، ويتم ذلك بتعريف SocketPermission Attribute نحدد فيها نوع العملية وال Access Kind و عنوان ال Host الذي سيطبق عليه ال Permission ورقم ال Port ونوع ال Transport سواء موجه أو غير موجه TCP أو UDP.

نريد في هذا المثال منع اتصال الAddress 127.0.0.1 loopback بال Socket عبر جميع ال Ports وبغض النظر عن نوع ال Socket المستخدم.

```
[SocketPermission(SecurityAction.Deny, Access="Connect",
Host="127.0.0.1",Port="All", Transport="All")]
```

يمكننا الSecurityAction object من تحديد نوع العملية التي نريدها وكما يلي:

Assert: وتعني السماح Client Host معين من إجراء عملية محددة
Demand: وتعني تطبيق الصلاحيات على جميع ال Classes التي تقع في منطقة ال Stack أعلى ال Defined Abstract
Deny: وتعني منع ال Client Host من إجراء عملية معينة.
InheritanceDemand : وفيها تطبق الصلاحيات على ال Class الذي سيرث ال Class الحالي.
PermitOnly: وفيه يمنع جميع ال Access عدا ال Client User المحدد.
...

وفي الAccess property نحدد نوع عملية المنع أو السماح وتأخذ خيارين هما :

Accept لمنع أو السماح ل Client Socket من عمل Binding مع ال IP Address و ال Port المحدد.
Connect لمنع أو السماح ل Client Socket من عمل connect مع ال Remote Host المحدد.

في ال Host وال Port نحدد عنوان ال Host الذي سيطبق عليه ال Permission و رقم ال Port التي يتصل بها (في ال Port property نستطيع تمرير كلمة all لدلالة على تطبيق الصلاحية على جميع ال Ports)

وأخيرا نحدد ال Transport property والتي سنعرف فيها نوع ال Socket المستخدم وتأخذ الخيارات التالية:
All بدون تحديد نوع ال Socket إذ تطبق هذه ال Permession على جميع ال Socket Types.
Connectionless إذا كانت ال Socket تستخدم Datagram Protocols وكمثال بروتوكول UDP.
ConnectionOriented إذا كانت ال Socket تستخدم Oriented Protocols وكمثال بروتوكول TCP.
TCP إذ تستطيع تحديده مباشرة.
UDP إذ تستطيع تحديده مباشرة.

سوف نأتي على شرح كافة تفاصيل ال Permission Namespace وبقية ال Security Namespaces في النسخة الورقية من الكتاب.

The End of ebook

نـهاية النسخة الإلكترونية

سوف تجد الكثير من الإضافات والمواضيع الجديدة في النسخة الورقية ، لا تتردد أبدا في طلبها...

Fadi822000@yahoo.com
http://spaces.msn.com/members/csharp2005