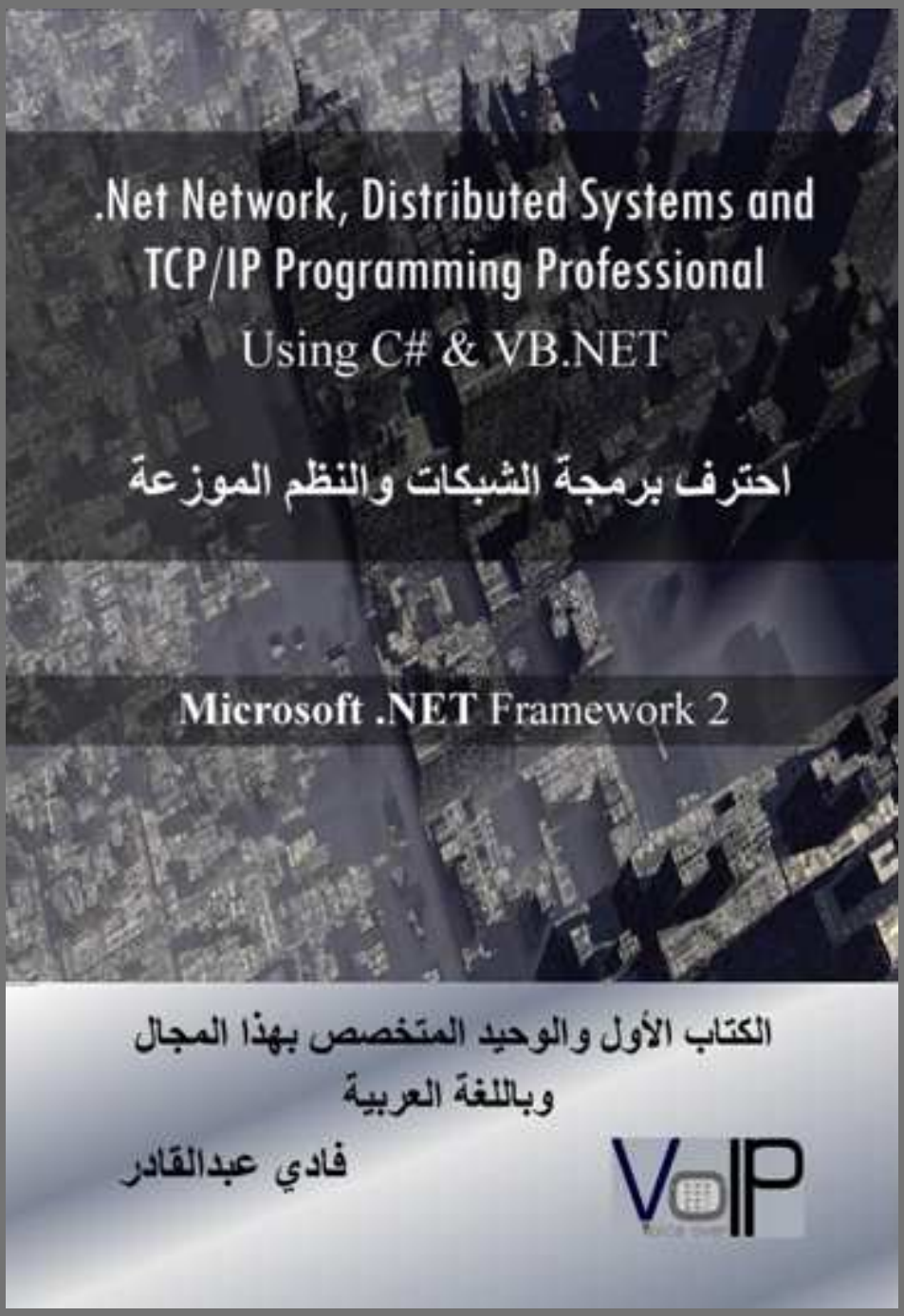
2006

أحترف برمجة الشبكات والنظم الموزعة

(النسخة الورقية الكاملة – الإصدار 2006)



SocketCoder.COM

الرخصة وحقوق النشر SCPL – SocketCoder Public License:

1 تسمى هذه الرخصة SCPL – SocketCoder Public License وقيامك بتحميل أو استخدام أي منتج من منتجات SocketCoder.Com يعتبر موافقة مسبقة من قبلك بالالتزام بتعليمات هذه الرخصة.

2 لا يوجد أي شروط تقيد عملية توزيع المنتجات الخاضعة لهذه الرخصة الكترونيا سواء من خلال البريد الإلكتروني أو المنتديات أو مواقع الإنترنت بشرط ذكر المصدر ووضع الرابط الرئيسي للمنتج.

3 تعتبر جميع المشاريع والمكتبات والأكواد والوثائق الخاضعة لهذه الرخصة مجانية بالكامل للأهداف التعليمية فقط لذلك لا يمكنك استخدامها بشكل تجاري دون إذن وموافقة من الناشر.

4 أبتدء من تاريخ 2009/11/31 تم أخضاع كتاب (احترف برمجة الشبكات والنظم الموزعة بالدوت نيت 2006) إلى رخصة SCPL وبذلك سيكون كل مافي الكتاب مجاني بالكامل بشكل ألكتروني وبذلك لا يحق لك بيعه أو بيع جزء منه أو إعادة طباعته بهدف بيعه بشكل ورقي، كذلك لا يحق لك اقتباس أي جزء منه دون مراسلة الناشر ببيان رغبتك بذلك وذكر المصدر بشكل التالي (احترف برمجة الشبكات والنظم الموزعة بالدوت نيت 2006، فادي عبد القادر، الناشر SocketCoder.Com)

حول المؤلف:

Fadi M. Abdelqader
Systems Engineer & Developer
www.SocketCoder.Com

المؤلفات الورقية حتى (2009):

- Professional Network, Distributed Systems & TCP/IP Programming In .NET Framework 1.1 & 2.0 (2006 Release).

أهم المؤلفات الإلكترونية المجانية حتى (2009):

- The *SocketCoder* e-Reference For Network, Distributed Systems And TCP/IP Programming In .NET 3.5, Arabic (SocketCoder.Com 2009)

أهم الأبحاث و المشاريع المجانية المفتوحة المصدر للأهداف التعليمية إلى (2009):

- (SocketCoder) RTP Multicasting VOIP Library (June-2008)
- (SocketCoder) Remote Desktop Controlling System (June - 2008)
- Large set of SCOL (SocketCoder Open License) learning Samples in network programming, conferencing, security & E-Learning Check www.SocketCoder.Com

المقدمة:

يناقش هذا الكتاب معظم 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 لغات الدوت نيت كما ويحتوي على شرح مفصل عن Socket Programming والـTransport Layer Protocols والـ Network Layer Programming وبناء أنظمة متقدمة باستخدام الـMulticasting كأنظمة المؤتمرات والـ Voice Chat Systems ويحتوي الكتاب على شرح لأهم الأمور المتعلقة بالـ Voice Over IP Programming والـ TAPI Telephony وبرمجيات الـ Remote Desktop وأنظمة التحكم عن بعد وغيرها بالإضافة إلى طرق تحليل والتصنت على الـ Packets المرسلة باستخدام الـ Raw Programming والـPacket Sniffing ، كما ويحتوي على شرح مفصل لأهم برتوكولات الـ Application Layer وتصميم و بناء النظم الموزعة واستخداماتها في برمجيات الشبكات، وأخيرا شرح مفصل عن طرق الحماية ووضع الصلاحيات والسياسات في برمجيات الشبكات.

قبل البدء بقراءة الكتاب:

يستخدم الكتاب الطريقة المتسلسلة في طرح ومناقشة المواضيع و الأكواد لذلك لابد من البدء بقراءة المواضيع بشكل متسلسل إذ أنه وفي كل فصل يتم شرح الأكواد الجديدة فقط، كما ويجب أن يمتلك القارئ الأساسيات البرمجية تحت أي لغة من لغات الدوت نيت قبل البدء بقراءة الكتاب.

ما هي الأمور الجديدة في هذا الكتاب ولمن هو موجه:

يطرح الكتاب معظم المواضيع المتعلقة ببرمجة الشبكات والنظم الموزعة بأسلوب مبسط للغاية ، ويعتبر الكتاب الأول والذي يناقش المواضيع المتعلقة ببرمجة برمجيات المحادثة الصوتية الـ Voice Over IP والمرئية Video Conference Systems وبناء أنظمة التعليم الإلكتروني عن بعد Remote eLearning Systems تحت منصة الدوت نيت ، خصص هذا الكتاب لطلاب والمبرمجين المهتمين بمجال برمجة الشبكات والنظم الموزعة.

ملخص الفصول:

Part1 Networks & TCPIP Programming Overview:
Chapter 1: TCP/IP Layers & Message Encapsulation Overview
Chapter 2: IPv4 & IPv6 Architecture Overview
Chapter 3: IP Multicasting Overview

Part2 Streaming:
Chapter 4: Streaming (Classes & Members)
Chapter 5: Applied Streaming in Dot Net

Part3 Transport & Network Layer Programming:
Chapter 6: Transport TCP & UDP (Classes & Members)
Chapter 7: Synchronous Socket Programming
Chapter 8: Asynchronous Socket Programming
Chapter 9: Advanced Multicasting Systems (Architecture & Conference Systems)
Chapter 10 VOIP - Voice Over IP Programming
Chapter 11 Raw Socket & Packet Sniffing Programming

Part4 Application Layer Programming:
Chapter 12 DNS Programming
Chapter 13 HTTP Programming
Chapter 14 Web Services & XML Programming
Chapter 15 Remoting & Distributed Systems Programming & Design
Chapter 16 SMTP & POP3 Programming
Chapter 17 FTP Programming

Part5 Network Security Programming:
Chapter 18 Cryptography
Chapter 19 Socket Permissions

Part6 Multithreading
Chapter 20 Multithreading (Using & Managing)

فهرس المحتويات:

Part3 Transport & Network Layer Programming:

Chapter 6: Transport TCP & UDP (Classes & Members)

- TCP Classes Members
- UDP Classes Members

Chapter 7: Synchronous Socket Programming

- Introduction to Socket Programming
- Synchronous Socket Programming
- Synchronous Socket Classes & Members

Chapter 8: Asynchronous Socket Programming

- Asynchronous Socket Class and its members
- Applied Asynchronous Socket in Dot Net

Chapter 9: Advanced Multicasting Systems

- Architecture of Multicast Socket
- Using Multicast Socket with .NET
- Multicast Conferencing Systems:
  - Full/Half Duplex Multicast Video Conferencing System.
  - Full/Half Duplex Multicast Desktop Conferencing System.
  - Full/Half Duplex Multicast Text Conferencing System

Chapter 10 VOIP - Voice Over IP Programming

- The Concept & Requirements of Voice Communication Systems
- How to Create a Voice Chat Through Dot Net Using Unmanaged API's Functions
- Testing UDP Multicasting, TCP and Thinking in SCTP to Transfer Voice Through Networks
- How to Create a Voice Conference System Using Microsoft Direct Play 9
- Create an Advanced Video/Voice Conference System By Using API & TAPI Telephony & Managed Socket Library in Dot Net.

Chapter 11 Raw Socket & Packet Sniffing Programming

- Introduction to Raw Socket & Raw Protocols
- Raw Socket Programming
  - ICMP – Internet Control Message Protocol Programming (Ping & Tracing)
  - Using ARP Protocol to Get The MAC of a Remote Machine in Dot Net
- Introduction to Packet Sniffing Applications

Part 4 Application Layer Programming:

Chapter 12 DNS Programming

- Synchronous DNS Members
- Asynchronous DNS Members

Chapter 13 HTTP Programming

- The Concept of HTTP Protocol
- Using HTTP in Dot Net
- Advanced HTTP Programming
- Using HttpWebRequest
- Using HttpWebResponse

Chapter 14 Web Services & XML Programming

- Introduction to Web services & XML
- Create A Simple Web Service Application
- Create A Simple GIS System Using (ADO.NET With Web Services)

Chapter 15 Remoting & Distributed Systems Programming & Design

- The Distributed Systems Concept & Design
  - Design a Distributed eCommerce System by Using ASP.NET and Web Services.
- Serialization Programming
  - The Serialization (Classes & Members)
  - Using BinaryFormatter & SoapFormatter to Serialize Objects & Images Through Network

- Remoting Programming
  - Remoting (Classes & Members)
  - Using Remoting Applications in Dot Net
  - Create an Advanced Distributed eLearning System (Remote Class Room)
  - Create an Advanced Remote Desktop Application With Remote (Mouse/Keyboard) Control Features

Chapter 16 SMTP & POP3 Programming

- SMTP Protocol
- SMTP Concept
- Using SMTP in Dot Net
- Advanced SMTP Programming
- POP3 Protocol Programming

Chapter 17 FTP Programming

- Introduction to FTP – File Transfer Protocol
- Create a Simple Application to Transfer Files By Using COM
- Create a Simple Application to Transfer Files By Using Web Classes
- Create a Simple Application to Transfer Files By Using Socket Programming & Streaming Libraries

Part5 Network Security Programming:

Chapter 18 Cryptography

- Cryptography in Dot Net:
  - Symmetric Encryption
    - DES (Data Encryption Standard)
  - Asymmetric Encryption
    - RSA (Rivest Shamir Adleman)
    - Digital Signature Algorithms
    - Hashing Algorithms
  - Using Security Encryption Standards With Network Applications
    - Create an Advanced RSA Client Server Chat System.

Chapter 19 Socket Permissions

- Permission Namespace Overview
- Security Action
- Socket Access property

Part6 Multithreading

Chapter 20 Multithreading Using & Managing

- Introduction to Threading in Dot Net
- Threading Classes & Members
- Multithreading & Network Applications

Appendixes (A): Network Programming Members

- System.Net Namespace
- System.Net.Socket Namespace
- Socket Option Members
- System.Threading Namespace
- TAPI Telephony Functions
- Remoting TCP / HTTP Channels Members

Appendixes (B): ASCII Code Table, References

Part1

Networks & TCP/IP Programming Overview

Chapter 1 TCP/IP Layers & Message Encapsulation Overview & Introduction to Network Programming

Chapter2 IPv4 & IPv6 Architecture Overview

Chapter3 IP Multicasting Overview

Chapter 1
TCP/IP Layers & Message Encapsulation Overview & Introduction to Network Programming

- TCP/IP Layers Encapsulation Overview

- TCP / UDP Connection Establishment

- TCP & UDP Header Encapsulation

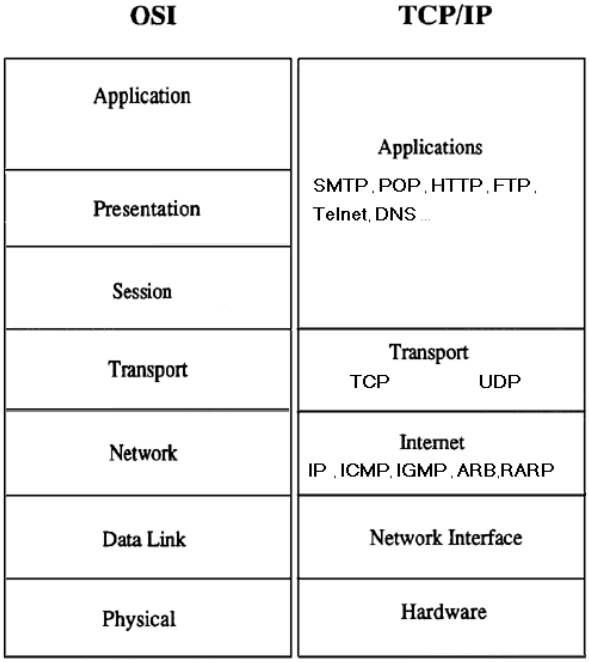
- Using TCP Connection Oriented in Dot Net to Send Unicast Messages

- Introduction to Binary Streaming in Dot Net

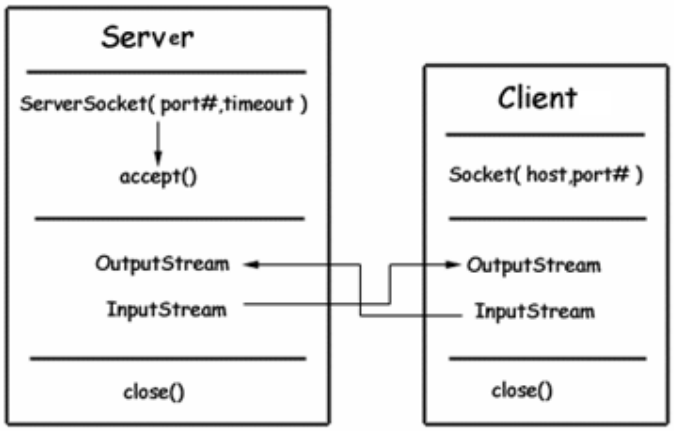
- Using UDP Connectionless in Dot Net to Send Uni & Broadcast Messages

TCP/IP Layers Encapsulation Overview : 1.1

من المعروف أن الشبكة هي مجموعة من الأجهزة متصلة مع بعضها عبر وسيلة اتصال معينة ومن هنا سيندرج لدينا التقسيم المعروف لمنظمة OSI لعملية الاتصال والتي تمر بسبعة طبقات لكل طبقة منها وظيفة معينة يتم إضافتها كـHeaders على البيانات المرسلة.وتم اختصارها إلى خمسة طبقات في برتوكول TCP/IP وتبين الصورة المرفقة هذه الطبقات:



تبدأ عملية توليف الرسالة المرسلة في الـ 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 أو الـ Bits توضع في مصفوفة لتجهيزها وإرسالها عبر Socket والذي يربط طبقة الـTransport Layer مع الـ Network Layer أو الـ Internet Layer ويوضح الشكل التالي طبيعة عمل الـSocket :



حيث يربط رقم الـPort المحدد في Transport Layer مع Destination IP في Network Layer ويقوم الـServer بالطرف المقابل بالموافقة على طلب الـClient وتقتصر وظيفة الـSocket في الـServer على ربط رقم الـPort مع الـSocket Option التي يتم تحديدها ثم البدء بعملية التصنت على الـPort الذي تم تحديده، ويمكن في هذه المرحلة وضع شروط معينة لقبول الجلسة مثل عمليات التحقق الـ Authentication أو ما شابه أو الموافقة بشكل مباشر.

في نموذج OSI تم تقسيم الـ upper Layers إلى ثلاثة طبقات:

Application لتعامل مع البرنامج نفسه أو ما يسمى User Interface

Presentation تمثيل البيانات المرسلة أو استخدام أساليب لضغط البيانات أو تشفيرها، في الدوت نيت تتم عملية تمثيل الرسالة باستخدام أي من الـ Classes ضمن الـ Encoding Class ضمن الـ System.Text Namespace وكمثال الـASCIIEncoding Class:

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)
```

Session وفيها البدء بعملية التخاطب بين الجهازين والتعريف ببعضهم البعض (فتح الجلسة) وتتم عادة في الـTCP بمجموعة من المراحل تسمى Three Way Hand Shake أما في الـUDP فيتم البدء بالإرسال دون إجراء أي من عمليات التحقق السابقة...

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

1- إجراء الاتصال المبدئي بالـ Server عبر الـ IP والـ Port المحدد وذلك بعد تحديد عملية الاتصال سواء عبر الـ UDP أو عبر الـ TCP.
2- التعريف بنفسه وعمل الـ Authentication إذا تطلب الـServer ذلك
3- قبول أو رفض الجلسة ويتم ذلك بإرسال الموافقة على فتح الجلسة أو رفضها
4- بدأ الجلسة وقيام الـServer بعمل Listening على الـPort الخاص بالبرنامج
وهنا مثال يوضح عمل هذه الطبقة باستخدام الـ TCP Protocol :

C#:
```
TcpClient tcpclnt = new TcpClient("192.168.0.2",8001);
```

VB.NET:
```
Dim tcpclnt As TcpClient = New TcpClient("192.168.0.2", 8001)
```

عندما يتم الموافقة على فتح الجلسة والبدء بعملية التخاطب يقوم جهاز المرسل الـ Client بتحميل الرسالة إلى الطبقة الأخرى وهي هنا طبقة الـ Transport وفي هذه الطبقة يتم تحديد

طبيعة الاتصال سواء عبر TCP - Connection Protocol أو عبر الـUDP Connectionless Protocol ففي البروتوكول الأول يتم تحديد طرفين وهما المرسل والمستقبل وPort الاتصال أما الـ UDP فيمكن أن يكون الطرف المستقبل كل الأجهزة Broadcast وهذا يعني أن أي شخص يقوم بتصنت عبر هذا الـ Port يستطيع استقبال الرسالة ، كما يمكن من عمل الـMulticasting (الإرسال إلى مجموعة Group)، ويتم ذلك بوضع الـBroadcast IP أو الـMulticast IP مع رقم الـPort في الـSocket (لمعرفة كيفية استخراج الـ Broadcast IP والـ Multicast IP أنظر الفصل الثاني).

ولإرسال الرسالة عبر الشبكة باستخدام الـ TCP 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 الـ Socket نفسه وكمثال على ذلك:

C#:

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

VB.NET:

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

وبعد ذلك تسلم إلى الـ Network Layer إذ تتم عنونة الرسالة ووضع عنوان المرسل والمستقبل عليها وتسلم إلى الطبقة الأدنى ليتم إرسالها عبر الـ Physical Tunnel ويحتوي الـNetwork Layer على مجموعة من البروتوكولات منها IP,IPv6,ARB-Address Resolution Protocol ولكل منها وظيفة معينة سنأتي على شرحها لاحقا...

أما بنسبة للجهاز المستقبل الـ Server فيقوم بالمرور على نفس الطبقات ولكن بالعكس حيث يستلم كرت الشبكة الـBits لتحول إلى Data link Layer ثم إلى الـ Network Layer ثم الـTransport Layer ثم الـApplication Layer ومنها تحول من Binary إلى ASCII ومن ASCII إلى Text وفي حالة استخدام الـStream Classes فإن هذه العملية تتم بشكل آلي .. ويوضح الكود التالي مبدأ عمل الـServer :

C#:

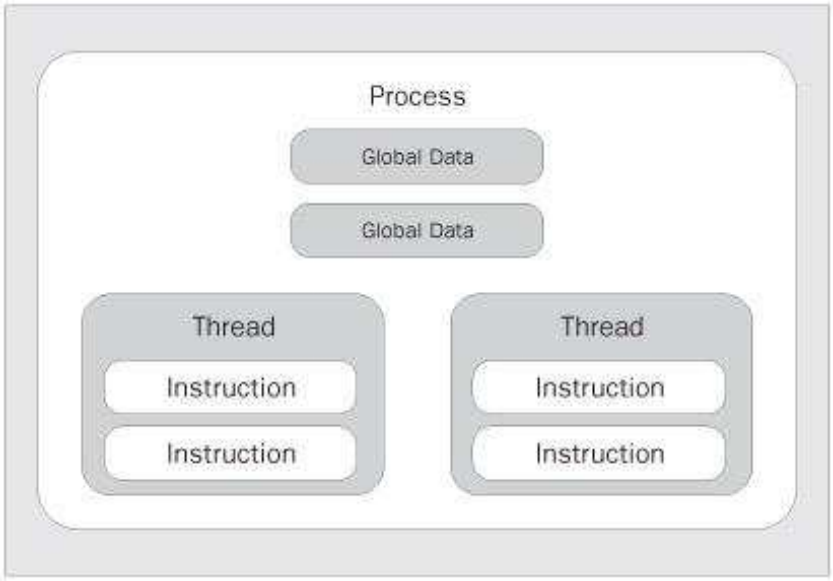
```
using System.IO;
using System.Net.Sockets;
using System.Threading;

TcpListener mylistener;
NetworkStream myntl;
public void Listener()
{
mylistener = new TcpListener(8001);
mylistener.Start();
Socket mysocket;
StreamReader str;
while (true)
        {
        mysocket  = mylistener.AcceptSocket();
        myntl = new NetworkStream(mysocket);
        str = new StreamReader(myntl);
        textbox1.Text = str.ReadToEnd();
        }
}
}
```

VB.NET:

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

Private mylistener As TcpListener
Private myntl As NetworkStream
Public Sub Listener()
   mylistener = New TcpListener(8001)
   mylistener.Start()
   Dim mysocket As Socket
   Dim str As StreamReader
      Do While True
         mysocket  = mylistener.AcceptSocket()
         myntl = New NetworkStream(s)
         str = New StreamReader(myntl)
         textbox1.Text = str.ReadToEnd()
      Loop
   End If
End Sub
```

ملاحظة هامة جدا:

سوف يؤدي الـ Infinity Loop والذي وضعناه إلى توقف البرنامج عن الاستجابة والسبب أن الـ Loop يعمل على نفس الـ Thread والمخصص للـ Form إذ لن ينفذ إي شيء إلا بعد انتهاء الـ Loop وهو ما لن يحدث أبدا إذ انه Infinity Loop، قدمت لنا الدوت نيت الحل لهذه المشكلة وهي باستخدام تكنولوجيا الـ Multithreading والتي تسمح بالمعالجة المتوازية على نفس المعالج وذلك من خلال تقسيم المهام على المعالج وعمل Session منفصلة لكل برنامج وهو ما يسمى بالـ Multitasking.. وهكذا لن يؤثر البرنامج على موارد النظام كما لن يؤثر الـ Infinity Loop على اداء البرنامج العام انظر الشكل التالي :



لاحظ انه قبل إضافة الـ Thread كان الـ Loop يعمل على منطقة الـ Global Area وهذا هو سبب عدم الاستجابة ، وبعد استخدام الـ Thread تم عمل Session أو Pipe خاص لدالة التي تحتوي على الـ Loop بحيث تعمل بشكل متوازي مع البرنامج، ولاستخدام الـ Thread يلزم أولا تعريف الـSystem.Threading Namespace :

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

VB.NET:
```
imports System.Threading
```

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

C#:
```
Thread myth;
myth= new Thread  (new System.Threading .ThreadStart(Listener));
myth.Start ();
```

VB.NET:
```
Imports System.Threading
Dim myth As Thread
```

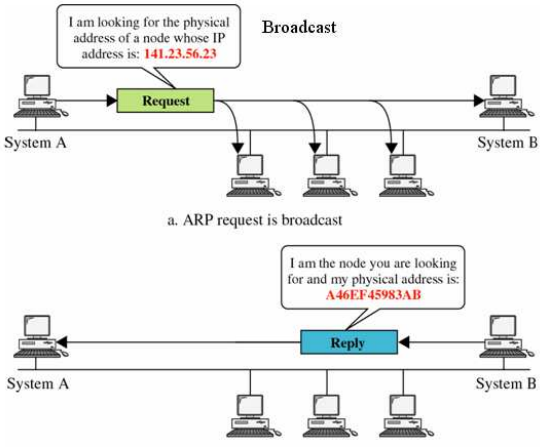
```
myth = New Thread(New System.Threading.ThreadStart(Listener))
myth.Start
```

الآن قم بإضافة myth.Aport() في حدث الـ Closing Form كما يلي:

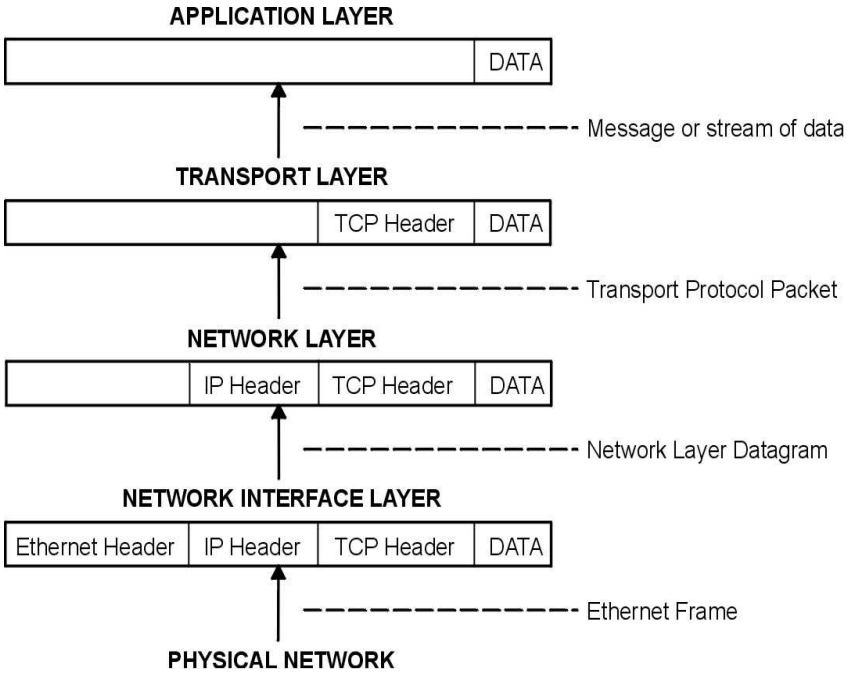
C#:
```
private void Form1_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
{
mylistener.Stop();
myth.Abort();
}
```
VB.NET:
```
Private Sub Form1_Closing(ByVal sender As Object, ByVal e As
System.ComponentModel.CancelEventArgs)
   Try
      mylistener.Stop()
      myth.Aport()
   Catch ex As Exception
      Msgbox(ex.Message)
   End Try
End Sub
```

ميزة الـ Thread رائعة جدا إذ تمكنك من تشغيل وإدارة أكثر من Thread في نفس الوقت وفي نفس البرنامج وهو ما يسمى بالـ Multithreading، انظر الفصل العشرون...

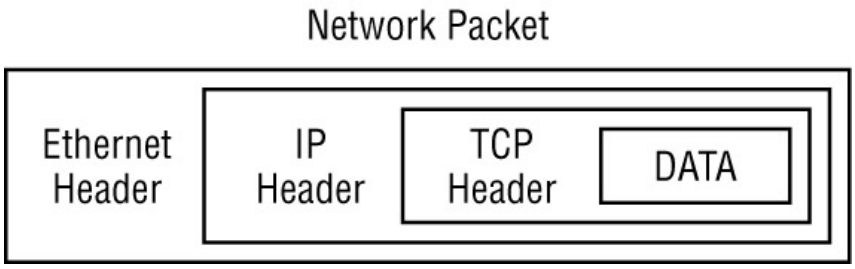
بعد إجراء عملية العنونة يقوم المرسل بسؤال عن عنوان الـMAC Address الخاص بالـServer وتتم هذه العملية عبر بروتوكول الـARP-Address Resolution Protocol ويقوم هذا البروتوكول بالتحقق من وجود الـMAC Address في الـMAC Table وفي حالة عدم وجوده يقوم بإرسالARB Broadcast Message Request إلى كل الأجهزة على الشبكة يسأل فيها عن صاحب الـIP Address المراد الإرسال له فإذا وجده يرسل الجهاز المعني Unicast ARP Replay Message يخبره فيها بعنوان الـMAC Address الخاص به وبعد استلام الرسالة يقوم بتخزين العنوان المعني في الـ MAC Table أو الـMAC Cash الموجود في الجهاز، ولكي يتم استخدامه في المرات اللاحقة ، لاحظ الشكل التالي:



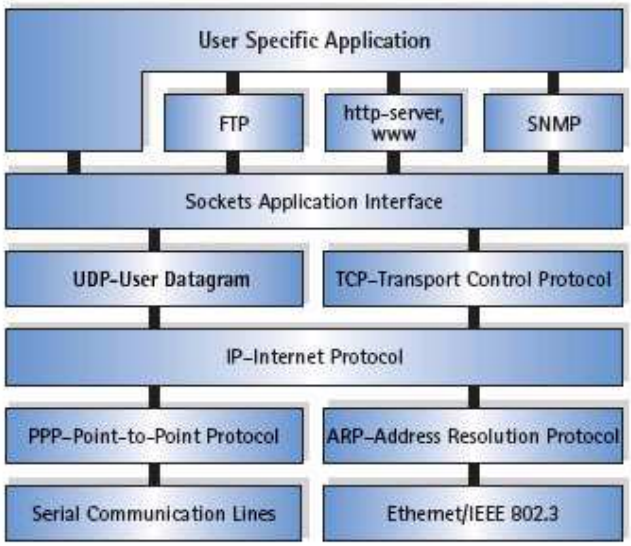
وبعد هذه المرحلة يتم تحديد نوع الـEncapsulation هل سيكون كـ Internet Encapsulation أو Ethernet Encapsulation ويتم ذلك في الـData Link Layer والشكل التالي يوضح كل هذه العمليات:



وفي النهاية يكون الشكل العام للـPacket كما يلي:



ويجب التفريق بين عملية الإرسال باستخدام الـEthernet وعملية الإرسال باستخدام الـInternet أو الـSerial Connection إذ أنه في الـEthernet تتم عملية الإرسال بعد معرفة الـMAC Address لطرف الأخر باستخدام الـARP أما في الـInternet فيتم الوصول وفق مبدأ الـPPP – Point to Point Protocol لاحظ الشكل التالي:



في كل طبقة يتم إضافة Header على الـData يحتوي على معلومات التحكم أو الـ Control Data ، ويتراوح حجم الـHeader في الـ Transport والـ Network Layer من 20 إلى 60 بايت حسب الـOptions التي يتم إضافتها إلى الـ Header.
في المثال التالي سنقوم باستخدام برنامج الـEthereal وهو برنامج يقوم بعملية الـ Packet Sniffing والذي يمكن تثبيته من الموقع الخاص به وهو:

http://www.ethereal.com/download.html

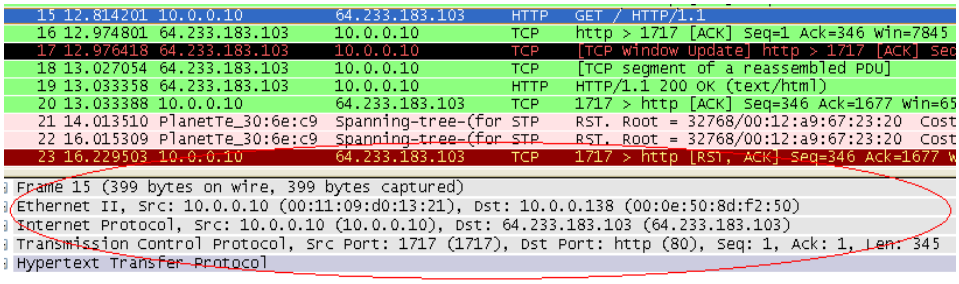
مثال على استخدام الـARB والـDNS والـHTTP :

لاحظ الشكل التالي في برنامج الـEthereal والذي يقوم بعمليات التصنت والتحليل للبيانات المارة من خلال الـNetwork Interface Card ، حيث قمنا بطلب الدخول على موقع الـGoogle باستخدام الـInternet Explorer:

8 12.623675 10.0.0.10	Broadcast	ARP	Who has 10.0.0.138? Tell 10.0.0.10
9 12.623915 10.0.0.138	10.0.0.10	ARP	10.0.0.138 is at 00:0e:50:8d:f2:50
10 12.623922 10.0.0.10	81.10.124.2	DNS	Standard query A www.google.com
11 12.682926 81.10.124.2	10.0.0.10	DNS	Standard query response CNAME www.l.google.com A 64.233.
12 12.684542 10.0.0.10	64.233.183.103	TCP	1717 > http [SYN] Seq=0 Ack=0 Win=65535 Len=0 MSS=1460
13 12.813988 64.233.183.103	10.0.0.10	TCP	http > 1717 [SYN, ACK] Seq=0 Ack=1 Win=8190 Len=0 MSS=14
14 12.814031 10.0.0.10	64.233.183.103	TCP	1717 > http [ACK] Seq=1 Ack=1 Win=65535 Len=0
15 12.814201 10.0.0.10	64.233.183.103	HTTP	GET / HTTP/1.1
16 12.974801 64.233.183.103	10.0.0.10	TCP	http > 1717 [ACK] Seq=1 Ack=346 Win=7845 Len=0
17 12.976418 64.233.183.103	10.0.0.10	TCP	[TCP Window Update] http > 1717 [ACK] Seq=1 Ack=346 Win=
18 13.027054 64.233.183.103	10.0.0.10	TCP	[TCP segment of a reassembled PDU]
19 13.033358 64.233.183.103	10.0.0.10	HTTP	HTTP/1.1 200 OK (text/html)
20 13.033388 10.0.0.10	64.233.183.103	TCP	1717 > http [ACK] Seq=346 Ack=1677 Win=65535 Len=0
21 14.013510 PlanetTe_30:6e:c9	Spanning-tree-(for STP		RST. Root = 32768/00:12:a9:67:23:20 Cost = 200000 Port
22 16.015309 PlanetTe_30:6e:c9	Spanning-tree-(for STP		RST. Root = 32768/00:12:a9:67:23:20 Cost = 200000 Port
23 16.229503 10.0.0.10	64.233.183.103	TCP	1717 > http [RST, ACK] Seq=346 Ack=1677 Win=0 Len=0

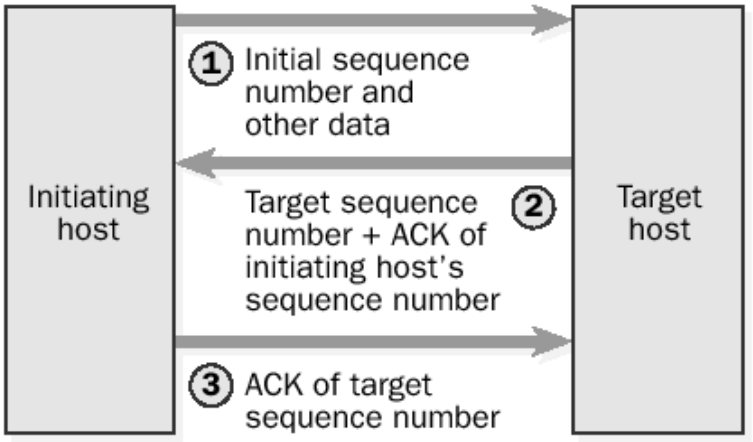
لاحظ أن أول عملية كانت السؤال عن الـMAC Address الخاص بالـGateway وهو هنا 10.0.0.138 حيث أرسلت هذه الرسالة كـBroadcast Message وفي العملية الثانية قام الـRouter بالرد على الـARP Request وأرسل عنوان الـMAC Address الخاص به بعد هذه العملية سيقوم الـClient بالاستفسار عن عنوان الـGoogle بإرسال DNS Request Query ، لاحظ أنه في المثال قد وجد الـIP Address الخاص بالـGoogle في الـ DNS

Table والموجودة ضمن نظام التشغيل وهذا يعني انه تم الدخول على الـGoogle من قبل والدليل على ذلك أن الـIP Address الخاص بالـGoogle موجود في الـDNS Table وبعد هذه العملية قام موقع الـGoogle بالرد على الطلب يخبره بوجوده وضمن العنوان المحدد لاحظ أن العمليات من 8 إلى 20 هي عمليات طلب و إرسال لمحتويات الصفحة الخاصة بموقع الـGoogle وتتم باستخدام الـ HTTP Protocol.



TCP / UDP Connection Establishment : 1.2

تمر عملية إنشاء الاتصال بمجموعة من المراحل، وفي العادة يقوم الطرف المرسل بإرسال طلب إنشاء الاتصال إلى الطرف الأخر وعند الموافقة على إجراء الاتصال يتم البدء بإرسال مجموعة من المعلومات إلى الطرف المستقبل، في بروتوكول الـ TCP تتم هذه العملية بثلاثة مراحل تسمى Three-way hand-shake وكما هو واضح في الشكل التالي:



1- حيث يقوم الطرف المرسل بتوليد رقم تسلسلي Sequence Number ويرسله إلى الـServer ويكون هذا الرقم المولد نقطة البدء لعملية الإرسال بحيث يتم زيادته بمقدار واحد (increment By One) عند كل عملية إرسال.

2- يستلم الطرف المقابل الـSequence Number ويقوم بإرسال Acknowledgment إلى المرسل مضاف إليه الرقم التسلسلي الذي تم إرساله.

3- عند هذه المرحلة يكون قد تم الموافقة على بدأ الجلسة وعندها يقوم بإرسال طلبه مرفق معه الـAcknowledgment الذي أرسل من قبل المستقبل.

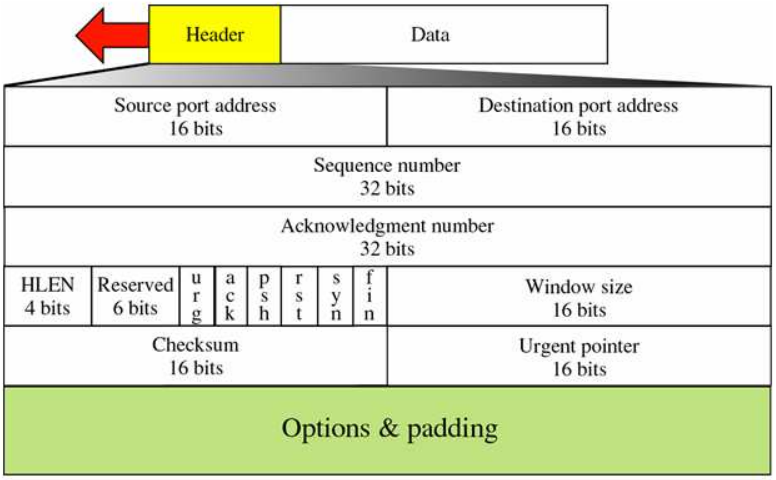
أما في بروتوكول الـ UDP فتتم بدون إرسال Acknowledgments ولا يتحقق الـUDP من عمليات الوصول كما هو الحال في الـTCP وهو ما سيتم توضيحه في الجزء التالي من هذا الفصل.

TCP & UDP Header Encapsulation : 1.3

في طبقة الـTransport Layer يتم التعامل مع إحدى البروتوكولين TCP أو الـUDP حيث سنحدد فيها طبيعة الإرسال فإذا كان المطلوب هو الإرسال كـStream ونوع الإرسال هو Unicast فيتم اختيار الـTCP أما في حالة كان المطلوب هو الإرسال كـBroadcast أو Multicast فيتم اختيار الـUDP لعملية الإرسال ، وسوف نبين في الجزء التالي من هذا الفصل مبدأ عملية الإرسال باستخدام بروتوكول الـTCP وبروتوكول الـUDP ...

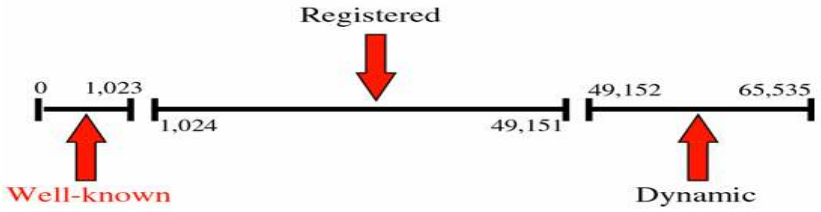
TCP Encapsulation : 1.3.1

يتميز هذا البروتوكول بدعمه لكل عمليات التحكم سواء على مستوى الـ Data Flow أو حجم الـBuffer كما يدعم عمليات التحقق من الوصول وفق الترتيب السليم Delivered on Sequence وهذا واضح من تركيب الـ Header الخاصة به أنظر إلى الشكل التالي:



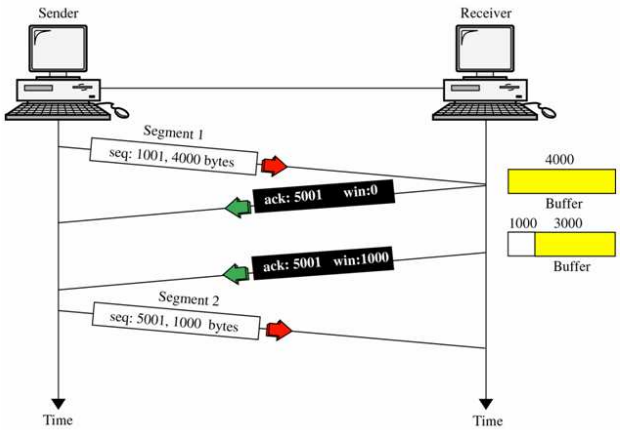
: Source & Destination Port 1.3.1.1

فـي المنطقـة المخصصـة بـ Port الإرسـال والاسـتقبال ، يـتم تحديـد الـPort Source والـPort Destination الخاصة بالبرنامج الذي يجري عملية الاتصال ومن المعروف أنه لا يمكن لأكثر من برنامج استخدام نفس رقم الـPort لكن يمكن للبرنامج الواحد استخدام أكثر من Port Address ، وبتأكيد فإن عملية اختيار الـPort ليست عشوائية إذ يجب الابتعاد عن الأرقام التي تبدأ بـ 0 وتنتهي 1023 إذ أنها أرقام لبروتوكولات معروفة ويتم استخدامها في نظام التشغيل ومن الأمثلة عليها بروتوكول الـHTTP والذي يستخدم الـPort 80 وبروتوكول الـ FTP والذي يستخدم الـPort 21 وغيره، ويفضل عند اختيار الـPort أن لا يبدأ برقم يقل عن 49.151 انظر إلى الشكل التالي:



1.3.1.2 Sequence & Acknowledgment Number :

ويحتوي كل منهما على 32 Bits ويدل هذا الرقم على رقم التسلسل للـPacket عند إرساله أو استقباله ويتم توليده عشوائيا عند بداية الاتصال أما رقم الـ Acknowledgment فيحتوي على الرقم التسلسلي للـ Packet الذي تم التأكد من وصوله وتتم هذه العملية كما في الشكل التالي:



1.3.1.3 Header Length & Validation Controls :

ويحتوي الجزء الثاني من الـHeader الخاص بالـTCP على32 Bits مقسمة على الـWindows Size 16 Bits و16Bits أخرى للـ Header Length + Validation Controls لاحظ الشكل التالي:

HLEN 4 bits	Reserved 6 bits	urg	ack	psh	rst	syn	fin	Window size 16 bits

ويحتوي الـHeader Length على حجم الـHeader الخاص بالـTCP مقسوم على 4 أي لمعرفة حجم الـHeader نضرب الـHLEN (Header Length) بـ4 ، أما الـ Validation Controls فهي 6 Controls تأخذ كل منها 1 Bit فإذا كانت قيمته 0 فهذا يعني أن هذه الأداة غير مستخدمة وإذا كانت 1 فهذا يعني أن هذه الأداة مستخدمة وكما في الشكل التالي:

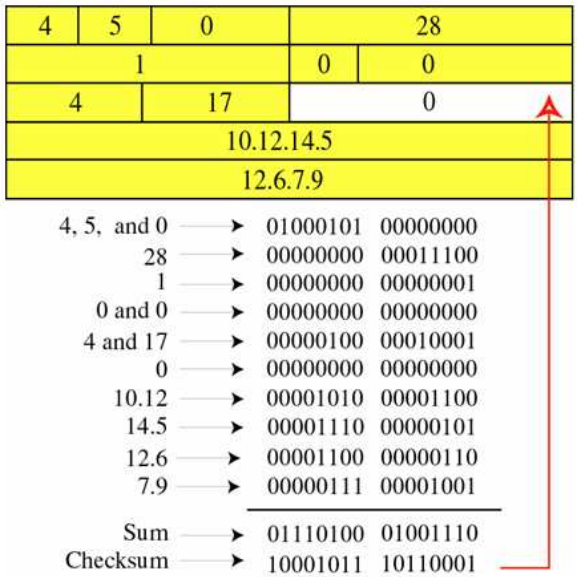
URG: Urgent pointer is valid
ACK: Acknowledgment is valid
PSH: Request for push
RST: Reset the connection
SYN: Synchronize sequence numbers
FIN: Terminate the connection

1.3.1.4 Window Size :

ويعرف فيه حجم الـPacket الذي يمكن إرساله من خلال الشبكة بناء على سرعة الوصول بين كل SYN Packet و ACK Packet، أي الوقت المستغرق لعملية التوصيل لكل Packet وقد تزيد أو تنقص بناءا على أدائية الشبكة.

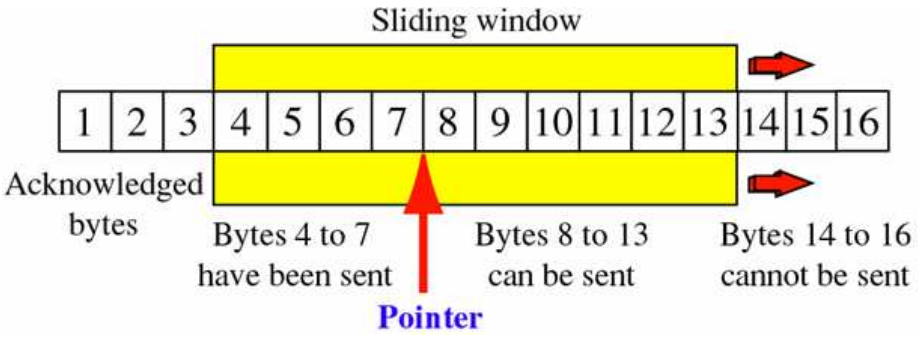
1.3.1.5 Check Sum :

وهي 16 Bits وتستخدم لعملية التحقق من وصول الـTCP Header بشكل السليم حيث يتم جمع كافة قيم الـTCP Header (كل 16 Bits لوحدها) ثم قلبها ووضع الناتج في الـ Check Sum وفي الطرف المستقبل يقوم بتأكد من الـChecksum بضرب قيمة الـHLEN بـ4 ثم مقارنة الناتج مع مقلوب الـChecksum وكما هو واضح في الشكل التالي والذي يظهر الـChecksum في الـ IP Protocol إذ انها تتم بنفس الطريقة بالـ TCP والـUDP:



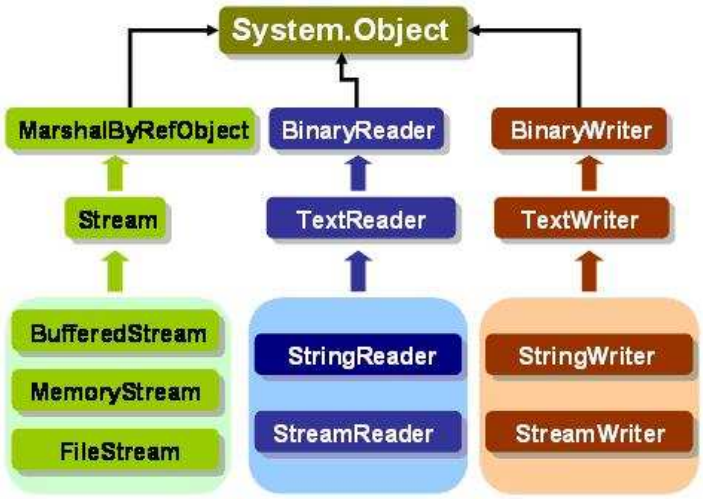
1.3.1.6 Urgent Pointer :

من المعروف أن الـData المرسلة عبر الـTCP يتم تجميعها في الـBuffer قبل أن يتم عرضها حيث يتم تحديد موقع الـData القادمة الجديدة في الـBuffer ومن هنا نحن بحاجة إلى وجود Pointer يؤشر على موقع الـData في الـBuffer وهو هنا الـUrgent Pointer لاحظ الشكل التالي والذي يوضح عملية وضع Data قادمة جديدة إلى الـBuffer الخاص بالجهاز المستقبل :

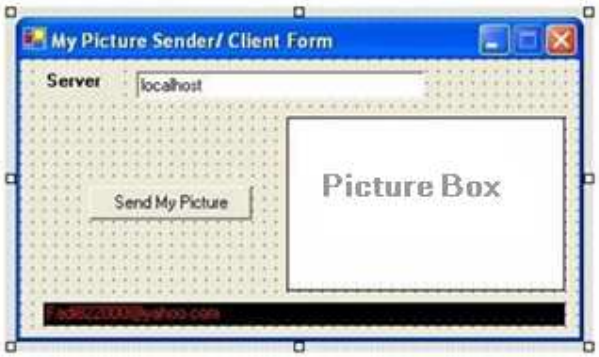


1.3.2 : Dot Net Binary Streaming Overview

يمكن من خلال الـStream Classes نقل Text أو Binary Data ،و بينا سابقا كيفية التعامل من الـSocket لنقل Text باستخدام الـ Stream Reader والـ 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 الـ Text كنا نحول الـ Text إلى ASCII Code ثم إلى Binary أما في الـ Object فيتم نقله إما كـSerialization Object (انظر الفصل الخامس عشر) أو باستخدام الـBinary Reader و الـ Binary Writer والتي تمكنك من التعامل مع أي نوع من البيانات ، كما وتستخدم الـStream Reader والـ Stream Writer لتعامل مع الـ Text والـ FileStream لتسهيل التعامل مع الملفات بالإضافة إلى الـ MemoryStream والتي تستخدم كـ Buffer لحفظ البيانات قبل إرسالها أو بعد استقبالها ، وتقوم أيضا بتحويل الـStream إلى مجموعة من الـBits والـBits إلى Stream مرة أخرى لاحظ الشكل التالي والذي يوضح تسلسل مكتبات الـStream في الدوت نيت:



وكمثال تطبيقي على هذا سوف نقوم ببناء برنامج يقوم بعملية نقل صورة Image من جهاز إلى أخر Client/Server وللبدء قم بإنشاء مشروع جديد كما في الشكل التالي :



في البداية قم بإضافة الـ Namespaces التالية:

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

VB.NET:
```
Imports System.Net.Socket
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 الـ Socket إلى الـServer:

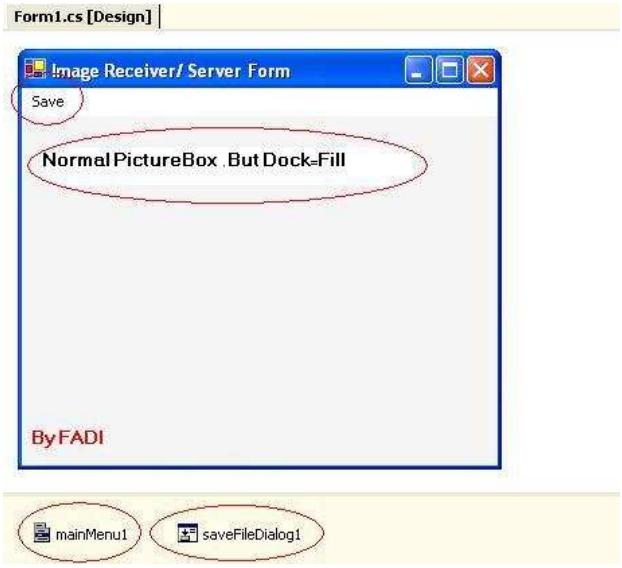
C#:
```
try
{
openFileDialog1.ShowDialog ();
string mypic_path =openFileDialog1.FileName ;
pictureBox1.Image = Image.FromFile(mypic_path);
MemoryStream ms = new MemoryStream();
pictureBox1.Image.Save(ms,pictureBox1.Image.RawFormat);
byte[] arrImage = ms.GetBuffer();
ms.Close();
TcpClient myclient = new TcpClient (txt_host.Text,5020);//Connecting
with The server
NetworkStream myns = myclient.GetStream ();
BinaryWriter  mysw = new BinaryWriter (myns);
mysw.Write(arrImage);//send the stream to The above address
mysw.Close ();
myns.Close ();
myclient.Close ();
}
catch (Exception ex){MessageBox.Show(ex.Message );}
```

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)
```

```
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
```

في الجزء الخاص بالـServer تتم عملية التصنت على الـ Port واستقبال الـ Stream عبر الـSocket وقراءتها باستخدام الـ Binary Reader وتحويله إلى Object (صيغته التي كان عليها قبل الإرسال) مرة أخرى، في هذا المثال نريد استقبال صورة وفي هذه الحالة وفرت لدينا الدوت نيت خصائص جديدة في الـ Controls الموجودة فيها ومن ضمنها Method Image.FromStream الخاصة بالـ Picture Box والتي تسهل علينا إمكانية عرض الصورة المرسلة من خلال Stream لكي يتم تحويلها من Stream إلى صورة تعرض على الـPicturebox وكما في المثال التالي:

قم بإنشاء مشروع جديد وكما في الشكل التالي :



تتم عملية استقبال الصورة كما يلي:

C#:

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

TcpListener mytcpl; // Declare TCP Listener
Socket mysocket; // Declare an object from Socket Class
NetworkStream myns; //
StreamReader mysr;
```

```
void Image_Receiver()
{
mytcpl = new TcpListener (5020);// Open The Port
mytcpl.Start ();// Start Listening
mysocket = mytcpl.AcceptSocket ();
myns = new NetworkStream (mysocket);
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
```

ولتنفيذ الـImage_Receiver Method بـ Thread منفصل عن الـ Thread الخاص بالبرنامج نقوم بتمريرها إلى الـ Thread Delegate Instance ضمن الـ Thread Class وكما 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
```

ويجب إغلاق الـ Thread والـ Socket في الـ Closing Event الخاص بالـ Form وكما يلي:

C#:

```
private void Form1_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
{
try
        {
mytcpl.Stop ();
thread.Aport();
        }
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()
      thread.Aport()
   Catch ex As Exception
Msgbox(ex.Message)
   End Try
End Sub
```

ويمكن إضافة الكود التالي إلى الـ Save Button لحفظ الصورة المستقبلة:

C#:

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

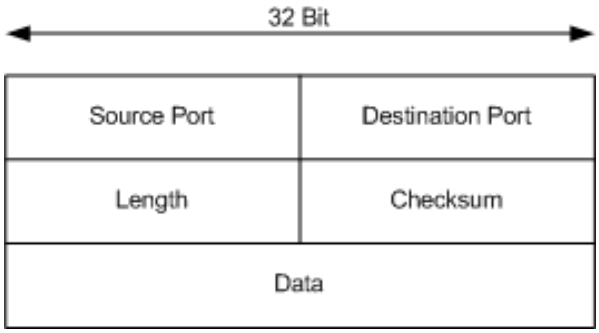
```
        {
string mypic_path = saveFileDialog1.FileName;
pictureBox1.Image.Save(mypic_path);
        }
}
catch (Exception)
        {
        }
```

VB.NET:

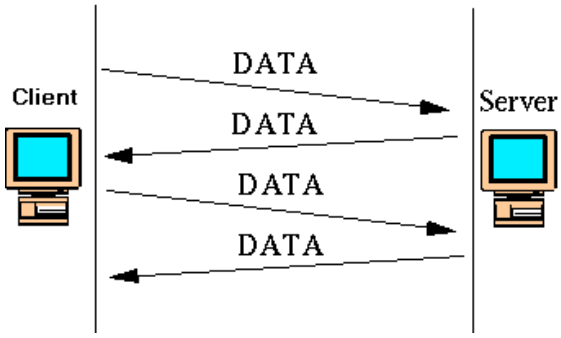
```
Try
 saveFileDialog1.Filter = "JPEG Image (*.jpg)|*.jpg"
 If saveFileDialog1.ShowDialog = DialogResult.OK Then
Dim mypic_path As String = saveFileDialog1.FileName
   pictureBox1.Image.Save(mypic_path)
 End If
Catch generatedExceptionVariable0 As Exception
End Try
```

UDP-User Datagram Protocol- Encapsulation : 1.3.3

سوف نبين في هذا الجزء طبيعة الاتصال باستخدام الـUDP حيث يتميز هذا البروتوكول بإمكانية الإرسال كـ Multicast و Broadcast بعكس الـTCP الذي يدعم الإرسال كـUnicast فقط ، لكن مشكلة هذا البروتوكول هو عدم دعمه لعمليات التحكم على مستوى الـData Flow أو حجم الـBuffer كما لا يدعم عمليات التحقق من الوصول وفق الترتيب السليم Delivered on Sequence وتعتبر هذه الأمور من أهم عيوبه ويوضح الشكل التالي التركيب العام لهذا البروتوكول:



الـCheck Sum و الـLength أو الـHeader Length هي نفسها في الـTCP لكن لاحظ عدم وجود أي من الأمور الخاصة بالـBuffer Management أو الـ Delivered On Sequence في الـHeader الخاص بالـUDP ، والمشكلة هنا أننا لا نستطيع عمل Fragmentation للـPackets حيث أن إعادة تجميعها بالترتيب الصحيح أمر غير مضمون، كما أنه لا وجود لأي Acknowledgment لتحقق من وصول البيانات أو عدم وصولها ، لاحظ الشكل التالي والذي يوضح طبيعة التراسل باستخدام الـUDP :



لاستخدام الـ UDP في الدوت نيت يلزم أولا تعريف System.Net Namespace والـSystem.Net.Socket لاحظ انه في الـ TCP كان يلزم تعريف رقم الـPort والعنوان للجهاز المستقبل أما في الـ UDP فتستطيع تعريفه كما هو في TCP كما وتستطيع عمل Broadcast باستخدام IPAddress.Any عند الاستقبال أو IPAddress.Broadcast عند الإرسال بعد اشتقاق كائن من الكلاس IPEndPoint وتستطيع أيضا عدم تحديد رقم الـPort وذلك بتمرير الرقم صفر 0 إلى الـBind Method في الـ Socket المخصصة للاستقبال.

في المثال التالي يتم فتح الـPort 5020 والتصنت عليها ثم استلام الرسالة عبر هذا الـ Port وتوزيعها على الكل:

1)

C#:

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

VB.NET

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

وتتم عملية إنشاء الـSocket وتحديد نوع البرتوكول المستخدم كما يلي:

2)

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 نمرر IPEndpoint Object إلى الدالة Send ...
في الـ Bind Method والتي يتم وضعها في الطرف المستقبل فقط يتم استخدامها لربط الـ IP Address ورقم الـPort بالـ Socket :

3)

C#:

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

VB.NET:

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

الآن تم استقبال الرسالة ونريد بثها إلى كل من يتصل مع الـServer على الـ Port السابقة ولعمل ذلك يلزم أولا تعريف IPEndPoint Object كما يلي :

4)

C#:

```
IPEndPoint sender = new IPEndPoint(IPAddress.Any, 0);
EndPoint Remote = (EndPoint)(sender);
recv = newsock.ReceiveFrom(data, ref Remote);
```

VB.NET:

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

ولطباعة عنوان مرسل الرسالة و الرسالة نفسها:

5)

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 يقوم بشبك مع الـServer لنخبره بها انه تم الموافقة على انضمامه:

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)
```

الهدف من الـ Infinity Loop : إبقاء عملية الاستقبال مستمرة بالإضافة إلى أنه عند استقبال أي رسالة في أي وقت من قبل أي جهاز ، يقوم الـServer باستلامها وتسليمها إلى كل من هو على الشبكة ... إذا أردت تحديد عدد معين من الرسائل المستلمة تستطيع تغيير الـ True في الـ 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()
```

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

C#:

```
using System;
using System.Net;
using System.Net.Socket;
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.Socket
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
```

ثانيا الجزء الخاص بالـ Client ، يقتصر العمل هنا على قيام الـ Client بإنشاء جلسة مع الـServer وذلك بعد تعريفه بالـ IPEndPoint ورقم الـPort وكما تم في السابق.

C#:

```
using System;
using System.Net;
using System.Net.Socket;
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);
  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)
 {
  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.Socket
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
```

ويمكن تطبيق الـ Broadcast Address على الـ UDP Socket وذلك بعد تفعيلها من الـSocket Options (لمزيد من المعلومات عن الـ SocketOption Class أنظر Appendixes (A) في المرفقات) وتتم العملية كما يلي كمثال لإرسال Broadcast Message إلى كل الأجهزة المتصلة على الشبكة المحلية:

C#: - Client

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
public class Broadcast
{
        public static void Main()
        {
        Socket sock = new Socket(AddressFamily.InterNetwork,
SocketType.Dgram, ProtocolType.Udp);
        IPEndPoint iep = new IPEndPoint(IPAddress.Broadcast, 5000);
        byte[] data = Encoding.ASCII.GetBytes("Hello All");
        sock.SetSocketOption(SocketOptionLevel.Socket,
SocketOptionName.Broadcast, 1);
        sock.SendTo(data, iep);
        sock.Close();
        }
}
```

VB.NET:- Client

```
Imports System
Imports System.Net
Imports System.Net.Sockets
Imports System.Text
Public Class Broadcast
Public Shared Sub Main()
  Dim sock As Socket = New Socket(AddressFamily.InterNetwork,
SocketType.Dgram, ProtocolType.Udp)
 Dim iep As IPEndPoint = New IPEndPoint(IPAddress.Broadcast,
5000)
    Dim data As Byte() = Encoding.ASCII.GetBytes("Hello All")
      sock.SetSocketOption(SocketOptionLevel.Socket,
SocketOptionName.Broadcast, 1)
      sock.SendTo(data, iep)
      sock.Close()
   End Sub
End Class
```

C#: - Server

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
public class RecvBroadcst
{
        public static void Main()
```

```
        {
Socket sock = new
Socket(AddressFamily.InterNetwork,SocketType.Dgram,
ProtocolType.Udp);
IPEndPoint iep = new IPEndPoint(IPAddress.Any, 5000);
sock.Bind(iep);
EndPoint ep = (EndPoint)iep;
Console.WriteLine("Ready to receive...");
byte[] data = new byte[1024];
int recv = sock.ReceiveFrom(data, ref ep);
string stringData = Encoding.ASCII.GetString(data, 0, recv);
Console.WriteLine("received: {0}  from: {1}",stringData,
ep.ToString());
        }
}
```

VB.NET:- Server

```
Imports System
Imports System.Net
Imports System.Net.Sockets
Imports System.Text
Public Class RecvBroadcst
Public Shared Sub Main()
      Dim sock As Socket = New Socket(AddressFamily.InterNetwork,
SocketType.Dgram, ProtocolType.Udp)
      Dim iep As IPEndPoint = New IPEndPoint(IPAddress.Any, 5000)
      sock.Bind(iep)
      Dim ep As EndPoint = CType(iep, EndPoint)
      Console.WriteLine("Ready to receive...")
Dim data As Byte() = New Byte(1023)
Dim recv As Integer = sock.ReceiveFrom(data, ep)
Dim stringData As String = Encoding.ASCII.GetString(data, 0, recv)
 Console.WriteLine("received: {0}  from: {1}", stringData,
ep.ToString())

   End Sub
End Class
```

وهكذا بينا مراحل الاتصال باستخدام الـTCP/IP واستخدام الـTCP والـUDP في الدوت نيت، سيتم الحديث في الفصل التالي عن Network Layer Encapsulation ومعمارية بروتوكول الـIPv4 والـIPv6 ...

Chapter 2
IPv4 & IPv6 Architecture Overview

- IPv4 Architecture

- Classful IP Address
    i. Unicast IP
    ii. Broadcast IP
    iii. Multicast IP

- CIDR Nation Overview

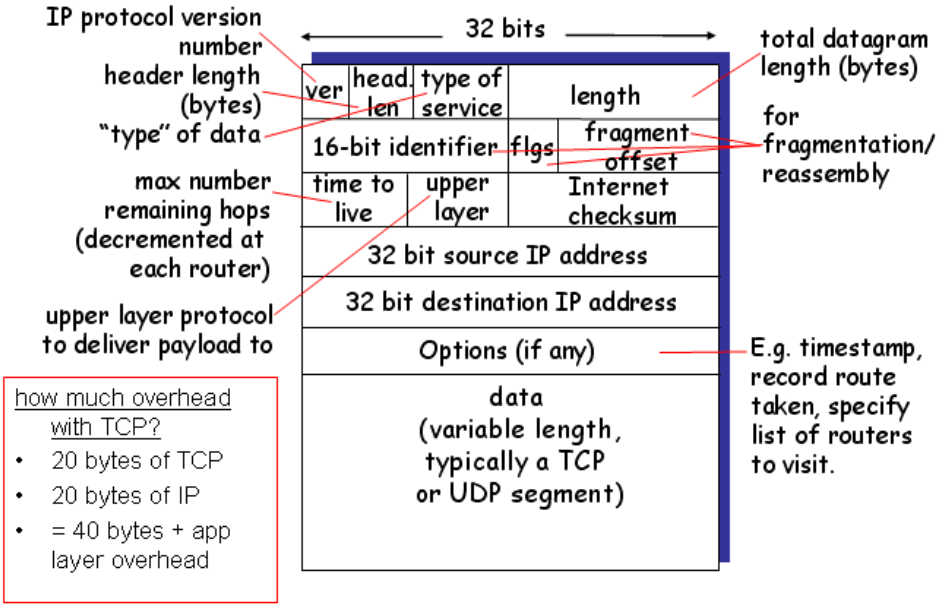
- IPv6 Architecture Overview

:The Internet Protocol

في الـNetwork Layer تتم عملية عنونة الـPacket باستخدام بروتوكول الإنترنت IP، وتتم هذه المرحلة بناءا على البروتوكول المستخدم في الـTransport Layer فإذا تم استخدام الـTCP عندها لا نستطيع إلا أن يكون العنوان المستخدم هو Unicast أما في حالة كان الـUDP هو البروتوكول المستخدم عندها نستطيع وضع عنوان Unicast أو Multicast أو Broadcast وسوف نأتي في الجزء التالي من هذا الفصل على شرح تركيب بروتوكول الإنترنت IPv4 وطرق وضع عناوين الـUnicast والـMulticast والـBroadcast فيه:

IPv4 Architecture : 2.1

يوضح الشكل التالي التركيب العام لبروتوكول الإنترنت IPv4 :



يتراوح حجم الـHeader الخاص بالـIPv4 من 20 إلى 60 Bytes بناءا على الـOptions المستخدم فإذا لم يتم استخدام الـOptions عندها سيكون حجم الـHeader ثابت وهو 20 Bytes وفي هذه الحالة يمكننا معرفة حجم الـData المرسلة عبر الـTCP كما يلي:

Data = total Length – (20 Bytes for IPv4 Header+ 20 Bytes For TCP Header)

وتقسم الـ20 Bytes الخاصة بالـIPv4 Header كما يلي:

4 Bits يوضع فيها الـVersion المستخدم وهو هنا IPv4 ويتم تمثيله 0100 في الـBinary.
4 Bits للـHeader Length ويوضع في حجم الـHeader مقسوم على 4
8 Bits للـType of Services حيث يمثل فيها مدى جودة الخدمة المطلوبة للبروتوكول المستخدم في الـApplication Layer ، ومنها Minimum Delay أو الـ Maximum Services وغيرها...
32 Bits وتستخدم لعمليات الـFragmentation وإعادة ترتيب الـFragments
8 Bits وتستخدم لتحديد الـ TTL – Time to Live ويعبر عن عدد الـHops أو الـRouters التي يسمح للـPacket المرور من خلالها إلى أن تنتهي ، والعدد الافتراضي لها 16 hops وعند مرورها بكل Router يتم طرح 1 من قيمتها.

8 Bits أخرى لتحديد نوع البروتوكول المستخدم في الـUpper Layer سواء TCP أو UDP ...

16 Bits للـ Checksum ولا تختلف طريقة حسابه عن الطريقة المستخدمة في الـTCP أو الـUDP والتي شرحناها في الفصل الأول.

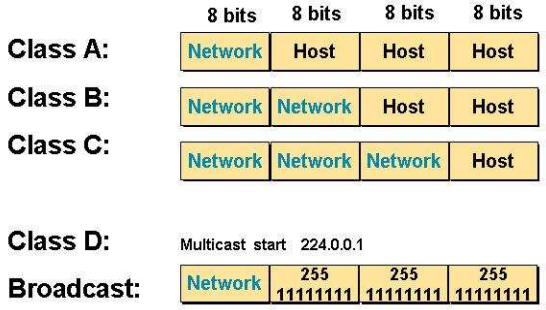
32 Bits لتحديد عنوان الجهاز المرسل

32 Bits أخرى لتحديد عنوان الجهاز المستقبل

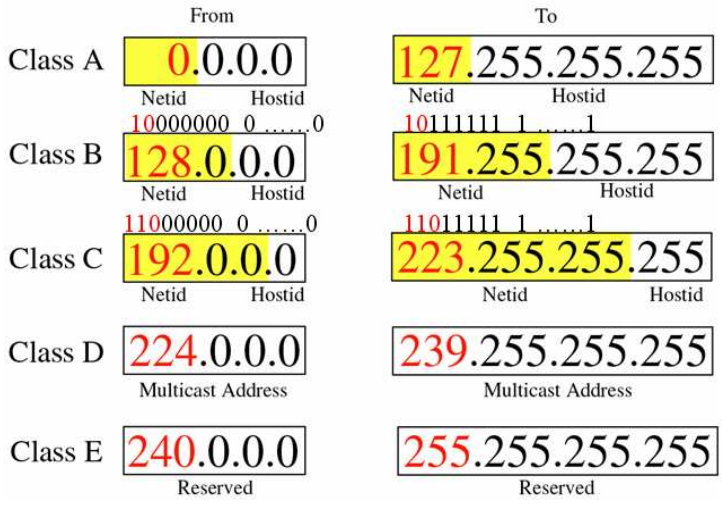
وسف نبين في الجزء التالي من هذا الفصل طريقة توليد الـUnicast والـBroadcast والـUnicast باستخدام الـClassful والـ CIDR - Classless InterDomain Routing :

Classful IP Address : 2.1.1

تعتبر عملية العنونة باستخدام الـClassful بسيطة جدا ويقسم فيها العنوان إلى جزأين، يعبر الجزء الأول عن عنوان الـ Network ID والجزء الثاني عن الـHost ID وكما هو موضح في الشكل التالي:



حيث يبدأ الـ Class A من 1 إلى 126 ويكون قيمة الـBit الأول في الـBinary صفر 0 ويبدأ Class B من 128 إلى 191 ويكون قيمة الـ Bit الأول في الـBinary واحد 1 ويبدأ Class C من 192 إلى 223 ويكون قيمة الـ Bit الأول والثاني في الـBinary واحد واحد 1 1 ... أما Class D فيبدأ من 224 وتكون الثلاثة Bits الأولى منه 111 ويستخدم لتعبير عن الـ Multicast Group ...والشكل التالي يوضح هذه التقسيمات:



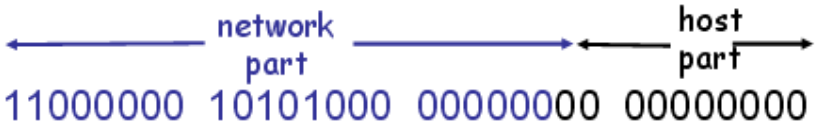
ولاستخراج عنوان الـ Broadcast من أي من التقسيمات السابقة يتم تعبئة الـBits الخاصة بالـHost ID بواحد وكمثال لمعرفة الـ Broadcast Address للعنوان 10.0.0.1 يجب أولا تحديد إلى أي Class ينتمي هذا العنوان، ومن الواضح أنه ينتمي إلى Class A لأن الجزء الخاص بالـ Network ID يبدأ بـ10 وهو بين 1 والـ126، إذا الجزء الخاص بالـHost ID

هو 0.0.1 ولتحويله إلى Broadcast Address نضع قيمة 255 في الجزء الخاص بالـ Host ID ويصبح العنوان كما يلي: 10.255.255.255 وهو الـ Broadcast Address للـ Class A في الـClassful Nation ...

لكن المشكلة في الـClassful Nation أنه محدود إلى درجة كبيرة فمثلا القيمة العظمى لعدد العناوين للـ Class A هو Host 2^{32-8} ، وكان الحل بإمكانية دمج الـSubnets لجعل إمكانية توليد عناوين أكثر للـHost Part وهو ما يسمى بالـClassless ...

CIDR - Classless InterDomain Routing IP Address : 2.1.2

وتتم هذه العملية باستئجار مجموعة من الـBits الخاصة بالـNetwork Part وضمها إلى الـHost Part وكمثال لتوليد 1024 عنوان جديد من العنوان 192.168.1.0/24
نقوم باستئجار 2 Bits من الـ24 Bits الخاصة بالـNetwork ID عندها يصبح الـSubnet كما يلي : 192.168.1.0/22 ويصبح للـ Host ID ، 10 Bits = 32-22 أي 2^{10} Addresses وكما هو واضح في الشكل التالي:



ولمعرفة الـ Broadcast الخاص بالعنوان الجديد 192.168.1.0/22 نقوم بتقسيم الـ 22 Bits على العنوان حيث كل جزء يأخذ 8 Bits وكما يلي:

8	8	6	
192	168	1	0
11111111	11111111	11111100	00000000

ثم نحول الـ one's في الجزء الثالث إلى عشري وسيكون في المثال 252 وهذا يعني أن الـAddresses Range ستبدأ بـ 192.168.252.0 وستنهي بـ192.168.255.254
إذا سيكون الـ Broadcast IP هو : 192.168.255.255 ...

لاحظ أن مشكلة المحدودية للعناوين قد حلت بشكل جزئي باستخدام الـCIDR لكن مازالت الإمكانيات محدودة إذا أردنا توليد عناوين لملايين من الأجهزة وكان الحل في الـIPv6 حيث زاد فيه حجم العناوين من 32 Bits إلى 128 Bits والذي سيتم الحديث عنه في الجزء التالي من هذا الفصل.

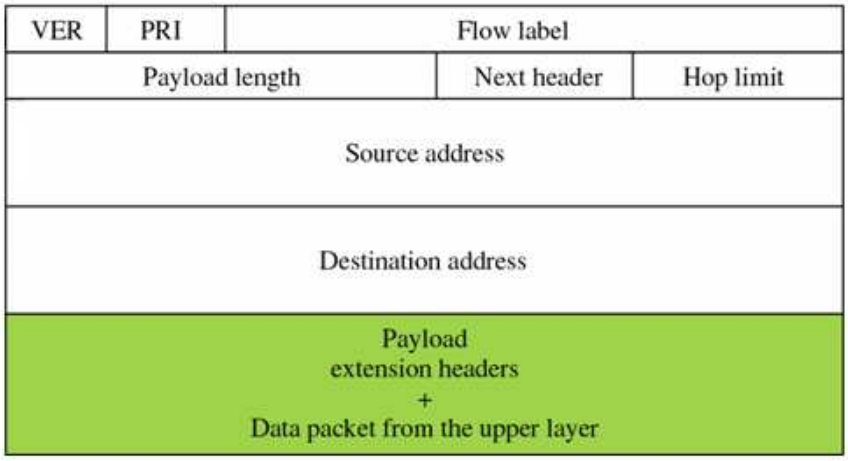
IPv6 Architecture : 2.2

الـIPv6 وهو الجيل التالي للـIPv4 حل الـIPv6 الكثير من المشاكل التي تواجه الـIPv4 ونلخصها بمجموعة من النقاط:

- الحاجة إلى عناوين أكثر حيث أن القيمة العظمى للـIPv4 Address تبقى محدودة مع زيادة الطلب على عناوين الـIP's في العالم.
- قد يؤدي الـIPv4 إلى مجموعة من المشاكل وخاصة في Routing Table والتي قد تم حلها في الـIPv6
- مشكلة الـSecurity في الـIPv4 حيث لم يدعم أي من عمليات التشفير والتحقق Authentication على مستوى الـNetwork Layer وقد حلت هذه المشكلة في الـIPv4

باستخدام بروتوكول الـIPSec حيث يتم تشفير الـIP والـPort على مستوى الـSocket لكن أصبحت الحاجة ملحة لجعل هذه الـSecurity مدمجة على مستوى الـNetwork Layer .
- تطوير مبدأ الـBroadcast حيث تم تطويره إلى الـany cast إذ يتم إرسال رسالة واحدة إلى كل جهاز على الشبكة وفي حالة وجد الجهاز المعني يتوقف الـRouter عن الإرسال ، والهدف في هذه الطريقة إيجاد طريقة تخفض من الـBandwidth المستخدم عند البحث عن جهاز ما على الشبكة.

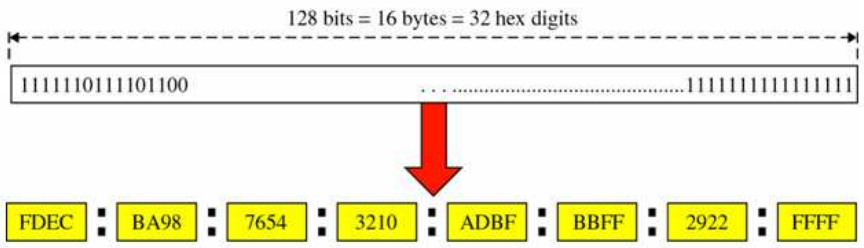
حل الـIPv6 كل هذه المشاكل حيث دعم ما يعادل Addresses 2^{128} وهو رقم كبير جدا كما قلل من حجم الـRouting Table مما سرع عمليات التوجيه Routing كما دعم عمليات الـAuthentication والتشفير على مستوى الـNetwork Layer كم تم حذف الـ Type of Services وحل محلها الـPriority أي الأولويات، ويوضح الشكل التالي الـHeader الخاصة بالـIPv6 :



يستخدم الـIPv6 الـ hexadecimal بدلا من الـDecimal في الـIPv4 لتمثيل العنوان وتكون الصيغة العامة له كما يلي كمثال:

69dc:8864:ffff:ffff:0:1280:8c0a:ffff

لاحظ انه يتكون من ثمانية منازل بدلا من 4 بالـIPv4 ، في كل منزلة يوضع بها 16 Bits.
ولتمثيله في الـHexadecimal نعطي 4 Digits لكل منزلة فيه وكما في الشكل التالي:

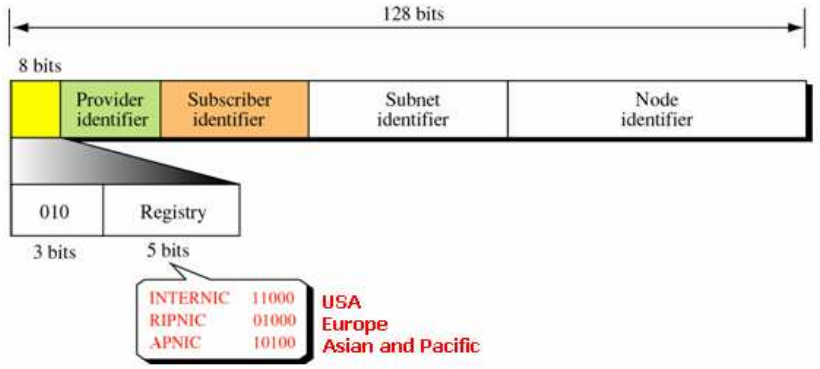


وتقسم العناوين في الـIPv6 إلى نوعين Provider-Based Addresses و Geographic-based Addresses حيث يقوم الـISP’s بتوزيع العناوين على الـClients باستخدام الـStandard الخاص Provider-Based Addresses وأما الـ Geographic-based Addresses فهو مخصص لإعطاء العناوين الدولية ، أي انه سيكون لكل دولة رمز خاص يكون في بداية العنوان وكما يلي:

- Provider-Based Addresses
    a) Registry ID
    b) Provider ID
    c) Subscriber ID
    d) Subscriber Subnet
    e) Host Number

- Geographic-based Addresses
    a) Registry ID
    b) World Zone
    c) Country, City, etc.

لاحظ الشكل التالي:



وهكذا بينا في هذا الفصل مبدأ عمل الـIPv4 والـIPv6 وكيفية توليد عناوين الـUnicast والـBroadcast باستخدام الـ Classful والـ CIDR Nation .

سيتم الحديث في الفصل التالي عن الـ IP Multicasting واستخدامها لعمل الـMulticast Group في الدوت نيت ...

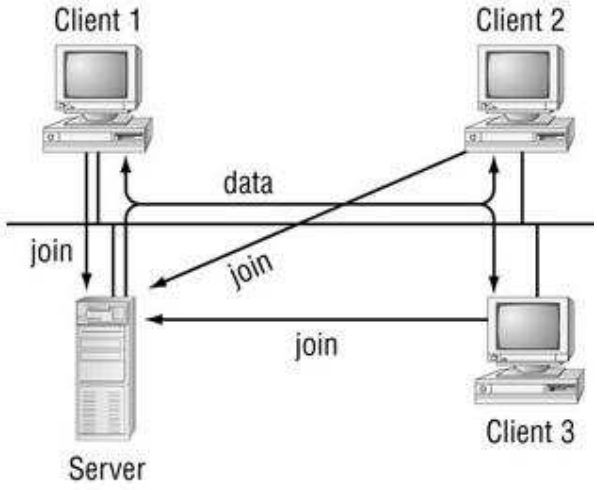
Chapter 3
IP Multicasting Programming Overview

- IP Multicasting Overview

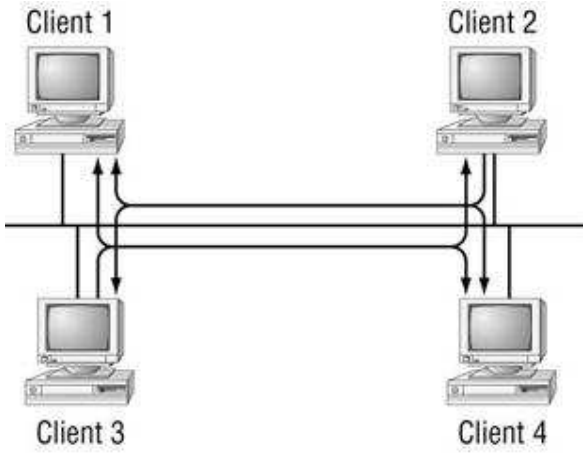
- Using IP Multicasting in Dot Net to Create a Multicast Groups

: IP Multicasting Overview :3.1

تحدثنا في الفصل الأول عن برتوكول الـ UDP وشرحنا كيفية استخدامه لعمل برود كاست حيث تستطيع عمل البرود كاست بطريقتين إما بجود Server يقوم بعملية التصنت على الـPort المحدد و يستقبل من خلاله أي رسالة ثم يقوم ببثها إلى كل الأجهزة أو باستخدام الـIPAddress.Broadcast في طرف المرسل والذي من خلاله يمكن عمل بث إلى كل الأجهزة حيث لا ضرورة لوجود جهاز Server بحيث أن الكل يمكنه التصنت على الـ Port المحدد و يستقبل ويرسل من خلالها أي رسالة إلى كل الأجهزة وتشبه عملية الـBroadcast عملية البث الإذاعي حيث أن الجميع يستمع من الكل ويرسل إلى الكل ، أما إذا أردنا تقسيم الإرسال إلى مجموعات عندها يجب استخدام الـ IP Multicasting وذلك بهدف استخدامه لعمل الـ Multicast Group، يعتبر هذا الموضوع من المواضيع المهمة جدا في برمجيات الشبكات ولهذا خصصت له فصل منفصل عن البقية (انظر الفصل التاسع) إذ أن اغلب برمجيات الـ 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 الـ Client ثم إرسالها إلى كامل المجموعة مرة أخرى انظر إلى الشكل التالي :



لاحظ انه يتم إرسال طلب الانضمام إلى المجموعة من قبل الـ Clients وإذا وافق الـServer على الطلب يقوم بضم عنوان الجهاز إلى الـ IP Address List الخاصة به وتشترك كل مجموعة بنفس الـ IP Multicast ويتم الإرسال إلى جميع أعضاء المجموعة التي تشترك بنفس الـ IP Multicast والذي يقع ضمن الـ Class D وهو ما بيناه في الفصل السابق.
النوع الثاني ويسمى بالـ unmanaged - peer-to-peer Technique حيث أن كل جهاز يعمل كـ 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 والإرسال 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

3.2: IP Multicasting واستخدامها لعمل Multicasting Group

قدمت الدوت نيت دعم جيد للـ IP Multicast باستخدام الـ Socket Namespace حيث يتم تعريفها باستخدام الدالة SetSocketOption والذي نعرف من خلاله الـ IGMP Protocol والذي يقوم بإدارة عمليات الانضمام والخروج من والى المجموعة multicast group (join & leave) كما يستخدم لإضافة وإلغاء العضوية AddMembership و DropMembership بالإضافة إلى العمليات المتعلقة بالـ Multicast Routing ، ويمكن استخدام الـUdpClient 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 ، 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.Socket;
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.Socket
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 الـSocket وتحديد الجهة التي سوف تستقبل الرسالة وهي (أي Socket يتنصت على الشبكة باستخدام الـ IP Multicast Group المحدد) ثم تحديد نوع الـSocket والبرتوكول المستخدم ...

ولإنشاء برنامج الاستقبال سوف نستخدم تعريف الـ Socket نفسه ونضيف الـ UdpClient Object ونسند له رقم الـ Port التي نريد التصنت عليه ، ويمكن أيضا تحديد الـTTL للـMulticast Packet ويعبر عن عدد الـRouters التي يسمح للـPacket بالمرور من خلالها وفي هذا المثال حددناها بـ 50 Hops:

C#:

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

class UdpClientMultiRecv
{
    public static void Main()
  {
UdpClient sock = new UdpClient(5020);
sock.JoinMulticastGroup(IPAddress.Parse("224.100.0.1"), 50);

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

```
byte[] data = sock.Receive(ref iep);
string stringData = Encoding.ASCII.GetString(data, 0, data.Length);
Console.WriteLine("received: {0} from: {1}", stringData,
iep.ToString());
sock.Close();}}
```

VB.NET:

```
Imports System
Imports System.Net
Imports System.Net.Socket
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 صورة إلى جانب النص أو أي شيء أخر يمكن تحويله إلى Binary إذ ما عليك سوى إضافة الـ **memory Stream** لتمثيل الصورة كـ Byte Array إلى كود الإرسال و الاستقبال كما يمكنك عمل برنامج لإرسال أوامر إلى الـ Clients لتحكم بهم عن بعد وكمثال عمل Shutdown لجميع الأجهزة التي تعمل على نفس الـ Multicast Group ، ويمكنك أيضا عمل برنامج لإرسال صورة عبر الكاميرا إلى جهات متعددة باستخدام نفس الخاصية والتي سآتي على شرحها في الفصل التاسع Advanced Multicast Systems ...

Part 2
Streaming in Dot Net

Chapter 4
Streaming in Dot Net
Managed I/O: Streams, Readers, and Writers

- Stream Classes
- Stream Members
- Stream Manipulation

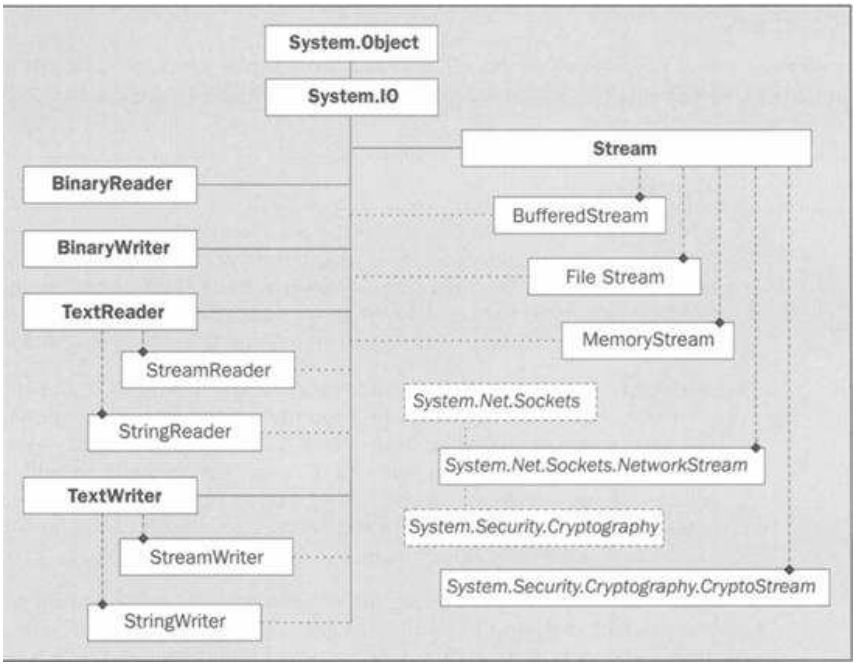
2.1 : Managed I/O: Streams, Readers, and Writers

إن الهدف من إنشاء مكتبات الـ Stream تسهيل عملية نقل البيانات من مكان إلى آخر سواء عبر الشبكة أو ضمن الـLocal Computer 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 الـStream بأسلوبين المتزامن **Synchronous** والغير متزامن **Asynchronous** وبشكل افتراضي تعمل جميع الـIO Streams بالأسلوب المتزامن لكن العيب فيه هو تأثيره الشديد على أدائية النظام إذ يقوم بإغلاق الـProcessing Unit في الـThread 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 الـFloppy Disk أو الـMagnetic Tape لكنها مهمة جدا بالبرمجيات التي تعتمد على أنظمة الزمن الحقيقي أو الـReal 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 الدوال المتزامن عن الغير متزامن في الدوت نيت بوجود كلمة Begin أو End في بداية اسم الدالة الغير متزامن وكمثال عليها BeginWrite و BeginRead و الـEndWrite و الـEndRead ..

أولا: Stream Classes

تدعم الدوت نيت عمليات الـStreams بمجموعة من الـClasses والمندرجة تحت System.IO Namespace والتي تستخدم لعمليات الإدخال و الإخراج لنقل البيانات . تستخدم بعض الـStream Classes ، Backing storage ، ومن الأمثلة عليها FileStream و الـBufferedStream والـMemoryStream وكذلك فإن بعضها لا يستخدم أي Back Storage ومن الأمثلة عليها الـNetworkStream والتي تستخدم لنقل الـStream عبر الشبكة وبدون استخدام Backing Storage ، و تقسم الـStream 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 الـBuffering لتحسين الأدائية حيت تكون كذاكرة وسيطة بين المعالجة والإرسال أو الاستقبال وكمثال عليها برمجيات الطباعة حيث تستخدم الطابعة ذاكرة وسيطة لتخزين البيانات المراد طباعتها بشكل مؤقت ، و يكمن الهدف الأساسي من استخدام الـBuffering في العمليات التي يكون فيها المعالج أسرع من عمليات الإدخال و الإخراج حيث يتم معالجة البيانات ووضعها في الـBuffer في انتظار إرسالها وهو ما يساهم في تحسين الأدائية بشكل كبير ، ويستخدم الـBufferedStream عادتا في برمجيات الشبكات مع الـNetworkStream 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 الـBufferedStream ويمكن زيادتها أو تقليلها حسب الحاجة.. ويستخدم الـBufferedStream 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 الـASCIIEncoding وتحميله في عبر الـMemoryStream ثم ارسلناه إلى الدالة readBufStream والتي انشأناها حيث استقبلنا من خلالها الـStream وحملناه في ذاكرة مؤقتة باستخدام الكلاس الـBufferedStream ثم قمنا بطباعة محتوياته بعد تحويله إلى نص مرة اخرى باستخدام الـ Encoding.ASCII وطباعته..

2- **MemoryStream Class** : وهو شبيه بعملية الـBuffring السابقة إذ يعتبر كحل جيد لتخزين البيانات بشكل مؤقت في الذاكرة كـ Stream Data قبل الإرسال أو الأستقبال حيث يغنيك عن تخزينها على شكل ملف مما يسرع العملية بشكل كبير ويستخدم ايضا لإجراء عمليات التحويل من Byte إلى Stream والعكس، 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** :ويستخدم لتعامل مع الـStream لإرساله عبر الشبكة باستخدام الـSocket ويتم استدعائها من System.Net.Socket Namespaces ويعتبر الكلاس NetworkStreamبأنه unbuffered إذ لا يحتوي على Backing Storage ويفضل استخدام الـBufferedStream Class معه لتحسين الأداء وتستخدم كما يلي كمثال حيث نريد إرسال الصورة التي قمنا بتخزينها في المثال السابق بذاكرة إلى جهاز آخر عبر الـSocket :

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 Namespaces وتستخدم بشكل اساسي في التعامل مع الملفات سواء للكتابة إلى ملف أو القراءة من ملف ويستخدم ايضا لتحويل الملف إلى Byte Array وحتى يسهل نقله عبر الشبكة و يعتبر هذا 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 الـEnumeration التالية مع الـFileMode :

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 الـCreate 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.

لمزيد من المعلومات حول استخدام الـFileStream انظر الفصل السابع عشر FTP Programming .

ثانيا : Stream Members :

هنالك مجموعة من الخواص و الـMethods (Members) التي تشترك بها مكتبات الـStream وهي كما يلي :

1- **CanRead** و **CanWrite** وتستخدم لمعرفة إذا كان الـStream 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 الـSeeking عادة لتحديد موقع الـCurrent Stream وترجع هذه الخاصية قيمة True في حالة كان الـStream يدعم الـSeeking و False في حالة أنه لا يدعم الـSeeking وفي العادة تدعم الـClasses التي تستخدم Backing Storage هذه العملية مثل الـFileStream وعندها ترجع قيمة True .

3- **CanTimeout** وترجع قيمة True في حالة إذا كان الـstream يحتوي على خاصية الـTimeout والتي تعطي وقت محدد للعملية .

4- **Length** وتستخدم لمعرفة حجم الـStream بالـByte ويمكن الاستفادة منها لمعرفة نهاية الـStream أو لتحديد حجم المصفوفة بناء على حجم الـStream .

5- **Position** وتستخدم الـGet و Set لمعرفة أو تحديد موقع الـ Stream

وتشترك مكتبات الـStream بمجموعة من الـMethods وهي كما يلي :

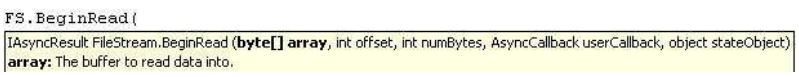
1- الدوال المتزامنة Synchronous Methods :

I. **Read** و **ReadByte** وتستخدم لقراءة Stream Data وتخزينه في الـBuffer ويمكن تحديد عدد البايتات التي سيتم قراءتها باستخدام الـReadByte كما نستطيع من خلالها معرفة نهاية الـStream حيث ترجع الـRead قيمة 0 والـReadByte قيمة -1 في حالة انتهاء الـStream.

II. **Write** والـ**WriteByte** وتستخدم لعملية الإرسال عبر الـStream ويمكن تحديد عدد البايتات التي سيتم كتابتها في كل مرة باستخدام الـWriteByte.

2- الدوال غير المتزامنة Asynchronous Methods :

I. **BeginRead** والـ**BeginWrite** وتستخدم لعملية القراءة أو الكتابة باستخدام الـStream الغير المتزامن وتأخذ خمسة باروميترات كما في الشكل التالي :



1- الـ**Byte Buffer** والتي سوف تستخدم لعملية القراءة منه أو الكتابة عليه

2- الـ**offset** وهو المؤشر أو الـ Pointer والذي نستطيع تحديد موقع القراءة أو الكتابة من وعلى الـBuffer.

3- الـ**numByte** والذي سوف يتم فيه تحديد الحد الأقصى من البايتات التي سيتم كتابتها أو قراءتها.
4- الـ**AsyncCallback** وهو Optional Delegate حيث يتم استدعائه لإنهاء عملية القراءة أو الكتابة بدون حجز الـ Thread وحتى يبقى الـ Stream في وضعية الانتظار .
5- الـ**Stateobject** وهو User Provided Object ويستخدم لتميز الـ Read & Write Request عن غيره من الـRequests .
ترجع الـBegin Methods الـIAsyncResult والذي يمثل حالة الـ Stream Operation .

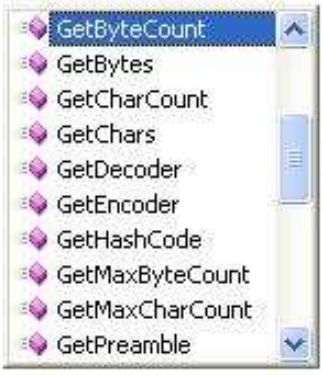
وهناك بعض الدوال والتي تستخدم لإدارة الـStream وهي :

1- **Flush** وتستخدم لتفريغ محتويات الـBuffer بعد إتمام العملية المحددة حيث يتم نقل محتويات الـBuffer إلى الـ**Destination** الذي تم تحديده في الـ Stream Object.
2- **Close** وتستخدم لإغلاق الـStream وتحرير الـResources المحجوزة من قبل الـStream Object وينصح باستخدامها في الجزء الخاص بـ Finally block ولتأكد من أن الـStream 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 الـStream والذي نريد إرساله أو استقباله لكن في حالة إذا كان الـStream أقل من المحدد في الـSetLength سوف يؤدي ذالك إلى انقطاع الـStream 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 الـStream .

ثالثا : Stream Manipulation:

يمكن استخدام مكتبات الـStream لنقل Binary Data أو Text وفي العادة يتم استخدام الـBinaryReader و الـBinaryWriter لتعامل مع الـData Binary ويتم استخدام الـStreamReader والـStreamWriter لتعامل مع الـText ، ويتم استخدام الـASCIIEncoding أو UnicodeEncoding لتحويل من Stream إلى Text عند الاستقبال ومن Text إلى Stream عند الإرسال حيث تستخدم مجموعة من الدوال وهي كما في الشكل التالي :

```
ASCIIEncoding asc = new ASCIIEncoding();
asc.
```



1- **GetByteCount** وهي Overloaded Method حيث تأخذ Character Array أو String وترجع عدد البايتات التي سوف نحتاجها لنقل نص معين ..

2- **GetBytes** لتحويل الـString إلى Byte Array حتى نستطيع إرسالها باستخدام الـStream .
3- **GetCharCount** حيث تأخذ Byte Array وترجع عدد الأحرف التي سوف تكون في الـString أو في الـCharacter Array .
4- **GetChars** وتستخدم لتحويل من Byte Array إلى String وتستخدم عند استقبال البيانات من الـStream حيث نحولها إلى نص مرة أخرى .

ولتعامل مع الـStreamReader و الـStreamWriter لنقل Text يجب أولا استدعائها من الـNamespace System.IO 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 الـBinaryReader والـBinaryWriter لنقل Binary Data يتم استدعائها من الـNamespace System.IO ، حيث تستخدم BinaryReader لقراءة Binary Data من الـStream والـ BinaryWriterr لإرسال BinaryData إلى الـStream عبر الـSocket وكما يلي كمثال:

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

وهكذا بينا أهم مكتبات الـStream في الدوت نيت وطرق التعامل معها ، والفرق بين الـStreams المتزامن والغير متزامن ، سوف نطبق في الفصل التالي مجموعة من الأمثلة العملية على الـSteaming في بيئة الدوت نيت...

Chapter 5
Applied Streaming in Dot Net

- Create a Simple Remote Control Application Using StreamReader & StreamWriter Classes

- Create a Remote Desktop Application By Using TCP Streaming Connection

- Create a Simple Application to Store & Read Images (Binary Data) in Microsoft Access & Microsoft SQL Server Database Management System By Using Streams Library & ADO.NET

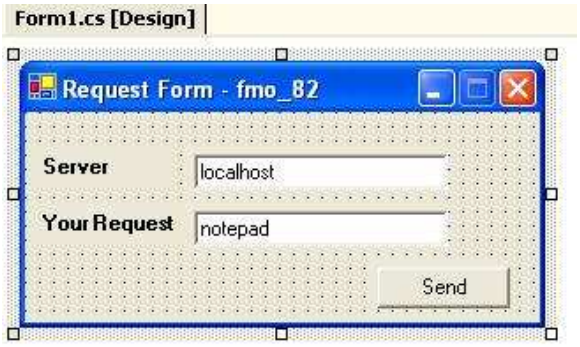
المقدمة:

سوف نناقش في هذا الفصل مجموعة من التطبيقات باستخدام الـ TCP Streaming ومنها نظام لتحكم عن بعد بإطفاء وتشغيل البرامج وإطفاء وتشغيل الأجهزة ، ونظام آخر لمراقبة سطح المكتب عن بعد ، وأخيرا طريقة تخزين الـBinary Data في قواعد البيانات Access و SQL Server Database باستخدام مكتبات الـStreams.

Remote Control Example 5.1 باستخدام الـStream Reader & Writer:

مثال تطبيقي بسيط سوف نستخدم فيه برنامج شبيه بـ Chatting لكن سيتم استخدامه لإعطاء أوامر إلى الـServer حيث يفترض إذا قمنا بإرسال كلمة notepad إلى الـserver بأن يقوم بفتح الـnotepad فيه وإذا قمنا مثلا بكتابة Calc وإرسالها إلى الـServer سوف يفتح الآلة الحاسبة فيه وهكذا :

أولا : إنشاء برنامج الإرسال Client : لا يختلف برنامج الإرسال عن برنامج الـ Client Chat الذي قمنا بإنشائه في الـChapter1 ويستخدم فيه كل من TCP Connection والـNetworkStream و الـStreamWriter لإجراء عملية الإرسال فباستخدام الدالة WriteLine الموجودة ضمن الـStreamWriter Object تتم عملية تحويل النص المكتوب في الـTextbox إلى مجموعة من الـBytes ليتم إرسالها باستخدام الـNetworkStream عبر الـTCP Socket Connection إلى برنامج الـServer وللبدء قم بإنشاء مشروع جديد كما في الشكل التالي :



ثم قم بإضافة الـ Namespaces التالية :

C#
```
using System.Net.Socket ;
using System.IO;
```
VB.NET:
```
imports System.Net.Socket
imports System.IO
```

في Send Button قم بكتابة الكود التالي:

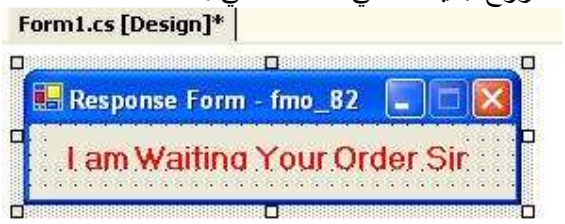
C#:
```
try
{
TcpClient myclient = new TcpClient (txt_host.Text,5020);
NetworkStream myns = myclient.GetStream ();
StreamWriter mysw = new StreamWriter (myns);
mysw.WriteLine(txt_msg.Text);
```

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

VB.NET:

```
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 الـServer والذي يعمل على استقبال الـStream وتحويله إلى Text مرة أخرى .. قم بإنشاء مشروع جديد كما في الشكل التالي :



ثم قم بإضافة الـ Namespaces التالية :

C#

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

VB.NET:

```
imports System.Net.Socket
imports System.IO
imports System.Threading
```

ثم إضافة التعاريف التالية إلى منطقة الـ Global Declaration :

C#:

```
TcpListener mytcpl;
Socket mysocket;
NetworkStream myns;
StreamReader mysr;
```

VB.NET:

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

وإنشاء دالة الـServer والتي ستستقبل الـCommands من الـClients :

C#:

```
void our_Server ()
                {
mytcpl = new TcpListener (5020);
mytcpl.Start ();
mysocket = mytcpl.AcceptSocket ();
myns = new NetworkStream (mysocket);
mysr = new StreamReader (myns);
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();

if (mysocket.Connected ==true)    {
        while (true)
        {
        our_Server ();
        }
        }
}
```

VB.NET:

```
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 الدالة بتصنت على الـSocket في حالة ورود أي Request يقوم بالموافقة عليه وإنشاء Session جديدة معه وفي حالة ورود أي بيانات عبر الـSocket يتسلمها باستخدام الـStreamReader ويحولها إلى Text ثم نقوم بفحص الرسالة باستخدام الجمل الشرطية فمثلا إذا كانت الرسالة هي notepad يتم استدعائها باستخدام الدالة Start الموجودة ضمن الكلاس Process والموجودة في System.Diagnostics Namespaces...

ولتشغيلها ضمن Thread جديد لابد من وضع تعريف الـThread في حدث بدأ التشغيل للـForm كما يلي :

C#

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

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
```

ثم قم بإضافة التالي في حدث الـClosing Form وذلك لتأكد من إغلاق الـSocket والـStream وإنهاء الـThread في البرنامج ..

C#:

```
private void Form1_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
{
try
{
mytcpl.Stop ();
myth.Aport();
```

```
}
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()
myth.Aport();
   Catch ex As Exception
        Msgbox(ex.Message)
   End Try
End Sub
```

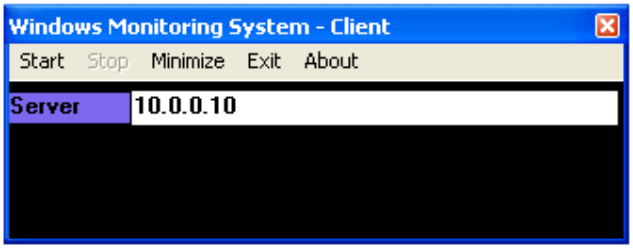
كما يمكننا بنفس الخطوات السابقة إنشاء برنامج يقوم بإطفاء الـComputer Clients عن بعد وذلك بتنفيذ الملف Shutdown.EXE (والمرفق مع الكتاب) عند ورود كلمة Shutdown أو تنفيذ أمر الـ RPC – Remote Procedure Call والذي يأتي مع نظام التشغيل Windows 2000/XP .

5.2 مثال لإنشاء نظام Remote Desktop لإرسال صورة سطح المكتب إلى الـ Server:

في هذا المثال سنقوم بإنشاء برنامج Remote Desktop بسيط لنقل صورة سطح المكتب من جهاز إلى آخر باستخدام الـ TCP والـ Stream Library :

أولا إنشاء برنامج الـ Client :

لتطبيق سنقوم بإنشاء مشروع جديد كما في الشكل التالي:



سوف نستخدم الـ Namespaces التالية:

```
System.Net
System.Net.Socket
System.IO
```

تتم عملية التقاط صورة سطح المكتب باستخدام دوال الـ API إذ انه لا يوجد مكتبة معينة في الدوت نيت لإجراء هذه العملية وقد قمت بكبسلة هذه الدوال في الملف CaptureScreen.dll لتسهيل استخدامها لاحقا ، ولالتقاط صورة سطح المكتب نستخدم الدالة CaptureScreen.GetDesktopImage() حيث ترجع صورة سطح المكتب كـ Image ، ولإرسالها لابد من تحويل هذه الصورة إلى Stream ووضعها في الـ Buffer قبل عملية الإرسال ، وتتم هذه العملية باستخدام الـ MemoryStream Class والذي شرحناه سابقا وكما يلي:

C#:

```
MemoryStream ms = new MemoryStream();
Image img  = CaptureScreen.GetDesktopImage();
img.Save(ms,System.Drawing.Imaging.ImageFormat.Jpeg);
byte[] arrImage = ms.GetBuffer();
```

VB.NET:

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

ولإرسال الصور الملتقطة إلى الجهاز الآخر لابد من تعريف Socket وكما يلي في حالة استخدام TCP Socket :

C#:

```
TcpClient myclient;
MemoryStream ms;
NetworkStream myns;
```

```
BinaryWriter  mysw;

myclient = new TcpClient (Server_IP.Text,5020);
myns = myclient.GetStream ();
mysw = new BinaryWriter (myns);
mysw.Write(arrImage);

ms.Close();
mysw.Close ();
myns.Close ();
myclient.Close ();
```

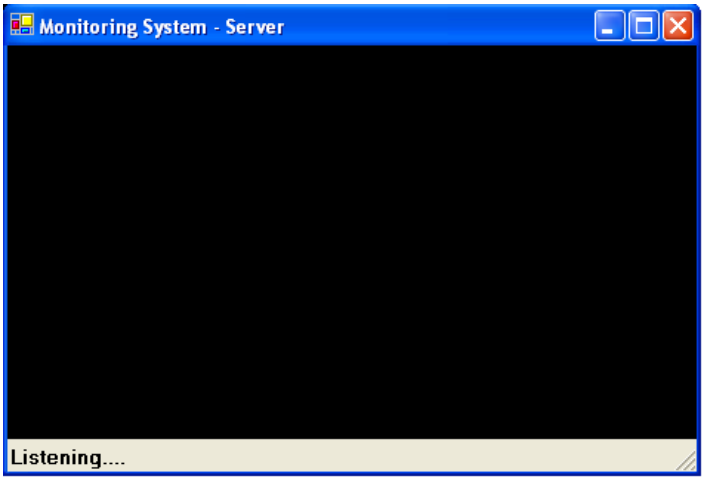
VB.NET:

```
Dim myclient As TcpClient
Dim ms As MemoryStream
Dim myns As NetworkStream
Dim mysw As BinaryWriter

myclient = New TcpClient (Server_IP.Text,5020)
myns = myclient.GetStream ()
mysw = New BinaryWriter (myns)
mysw.Write(arrImage)

ms.Close()
mysw.Close ()
myns.Close ()
myclient.Close ()
```

ويمكننا وضع هذا الكود ضمن Infinity Loop Method ومن ثم إنشاء New Thread لها ، أو نضعها في Timer ويحدد الـ Interval فيه بناء على سرعة الإرسال التي نحتاجها من جهة ومقدرة الشبكة من جهة أخرى ، وكشبكة LAN فإن 100 MS تعتبر ممتازة.

ثانيا إنشاء برنامج الـ Server :

ولتطبيقه سنقوم بإنشاء مشروع جديد كما في الشكل التالي:



سوف نستخدم الـ Namespaces التالية:

```
System.Net
System.Net.Socket
System.IO
```

ثم نعرف الـ TCP Socket والـ Thread والـ TCP Listener كما يلي:

C#:

```
Thread myth;
TcpListener mytcpl = new TcpListener (5020);
Socket mysocket;
NetworkStream myns;
```

VB.NET:

```
Dim myth As Thread
Dim mytcpl As TcpListener = New TcpListener(5020)
Dim mysocket As Socket
Dim myns As NetworkStream
```

ثم نقوم بتعريف دالة الاستقبال وكما يلي:

C#:

```
void ServerMethod ()
{
        try
        {

mytcpl.Start ();
```

```
mysocket = mytcpl.AcceptSocket ();
myns = new NetworkStream (mysocket);

pictureBox1.Image = Image.FromStream(myns);
mytcpl.Stop();

if (mysocket.Connected ==true)
        {
while (true)
                {
                ServerMethod ();
                }
        }
myns.Flush();

}
catch (Exception){}
}
```

VB.NET:

```
Private Sub ServerMethod()
   Try

      mytcpl.Start()
      mysocket = mytcpl.AcceptSocket()
      myns = New NetworkStream(mysocket)

      pictureBox1.Image = Image.FromStream(myns)
      mytcpl.Stop()

      If mysocket.Connected = True Then
         Do While True
            ServerMethod()
         Loop
      End If
      myns.Flush()

   Catch e1 As Exception
   End Try
End Sub
```

ولتشغيل الـ Thread نضع الكود التالي في حدث بدأ التشغيل للـ Form :

C#:

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

VB.NET:

```
myth= New Thread (New System.Threading.ThreadStart(AddressOf
ServerMethod))
myth.Start ()
```

وهكذا بينا طريقة إرسال صورة سطح المكتب من جهاز إلى آخر ، سوف نبين في الجزء الخاص بالنظم الموزعة (الفصل الخامس عشر) كيفية إضافة خاصية التحكم بالـ Mouse والـKeyboard للـRemoteServer وذلك باستخدام دوال الـAPI.

5.3 مثال لتخزين صورة في قاعدة بيانات Access و Server SQL باستخدام الـADO.NET ومكتبات الـStream :

سنستخدم في هذا المثال الـ Memory Stream Class لتمثيل صورة بالذاكرة على شكل Stream Data حيث يمكننا تحويلها إلى Bytes لاحقا باستخدام GetBuffer Method وكما يلي:

C#:

```
try
{
MemoryStream stream = new MemoryStream();
pictureBox.Image.Save(stream,System.Drawing.Imaging.ImageFormat.
Jpeg);
          byte[] arr = stream.GetBuffer();
          Store_it (arr);
         }
catch(Exception ex)
         {
MessageBox.Show(ex.Message);
         }
}
```

VB.NET

```
Try
Dim stream As MemoryStream = New MemoryStream
pictureBox.Image.Save(stream,System.Drawing.Imaging.ImageFormat.
Jpeg)
Dim arr As Byte() = stream.GetBuffer()
Store_it (arr)
Catch ex As Exception
MessageBox.Show(ex.Message)
End Try
```

أولا استخدام Microsoft Access :

بعد تحويل الصورة إلى Byte Array سنرسلها إلى الـ Store_it Method والتي سنستخدم من خلالها الـOleDbParameter لتمرير الـByte Array إلى الـDatabase وكما يلي:

C#:

```
public void Store_it (byte[] content)
{

try
{

oleDbConnection1.Open();
OleDbCommand insert = new OleDbCommand( "insert into img values
(?)",oleDbConnection1);
insert.Parameters.Add(new OleDbParameter("@pic",
OleDbType.Binary)).Value = content;
insert.ExecuteNonQuery();

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

finally {oleDbConnection1.Close();}

 }
```

VB.NET

```
Public Sub Store_it(ByVal content As Byte())
   Try
      oleDbConnection1.Open()
      Dim insert As OleDbCommand = New OleDbCommand("insert
into img values (?)", oleDbConnection1)
      insert.Parameters.Add(New OleDbParameter("@pic",
OleDbType.Binary)).Value = content
      insert.ExecuteNonQuery()
   Catch ex As Exception
      MessageBox.Show(ex.Message)
   Finally
      oleDbConnection1.Close()
   End Try
End Sub
```

وتتم عملية قراءة الصورة من قاعدة باستخدام الطريقة المعتادة لقراءة البيانات لكن يجب أن يتم تحويل الـBinary Data والتي سيتم وضعها في الـDataset إلى Stream مرة أخرى وتتم هذه العملية كما يلي:

C#:

```
oleDbDataAdapter1.SelectCommand.CommandText = "select * from
img";
```

```
oleDbDataAdapter1.Fill(dsPictures1);
byte[] arrPicture = ((byte[]) (dsPictures1.Tables[0].Rows[0]["pic"]));
MemoryStream ms = new MemoryStream(arrPicture);
pictureBox2.Image = Image.FromStream(ms);
```

VB.NET

```
oleDbDataAdapter1.SelectCommand.CommandText = "select * from
img"
oleDbDataAdapter1.Fill(dsPictures1)
Dim arrPicture As Byte() =
(CType(dsPictures1.Tables(0).Rows(0)("pic"), Byte()))
Dim ms As MemoryStream = New MemoryStream(arrPicture)
pictureBox2.Image = Image.FromStream(ms)
```

حيث قمنا بعمل Casting لمخرجات الـDataset وتخزينها في Byte Array وبعد ذلك وبتمرير الـArray إلى الـMemoryStream Class ، سيتم تحويل الـByte Array إلى Stream مرة أخرى وعندها يمكننا عرضها على PictureBox باستخدام الـ FromStream Method ...

ثانيا استخدام Microsoft SQL Server :

لا تختلف عملية التخزين بقاعدة بيانات SQL Server عن الـMicrosoft Access سوى انه يتم استخدام الـ SqlParameter بدلا من OleDbParameter ، كما أن صيغة الـConnection String ستختلف بين الـAccess Database والـ SQL Server Database حيث يمرر للأول مسار الملف الخاص بقاعدة البيانات أما الثاني فيمرر له اسم الجهاز الذي يحتوي على الـ SQL Server وكما يلي كمثال:

لإنشاء الـ Connection String:

C#:

```
string SQL_CONNECTION_STRING =
"Server=SQL_SERVER_NAME;DataBase=DB_NAME;Integrated
Security=SSPI";
```

VB.NET

```
Dim SQL_CONNECTION_STRING As String =
"Server=SQL_SERVER_NAME;DataBase=DB_NAME;Integrated
Security=SSPI"
```

ثم يمرر إلى الـ Connection Object كما يلي:

C#:

```
SqlConnection My_Connection = new
SqlConnection(connectionstring);
```

VB.NET

```
Dim My_Connection As SqlConnection = New
SqlConnection(connectionstring)
```

لتخزين الصورة في قاعدة البيانات SQL Server :

C#:

```
public void Store_it (byte[] content)
{
        try{
My_Connection.Open();
SqlCommand insert = new SqlCommand( "insert into img values
(@pic)",My_Connection);
insert.Parameters.Add(new SqlParameter("@pic",
SqlDbType.Binary)).Value = content;
insert.ExecuteNonQuery();
                        }
catch(Exception ex)
        {
        MessageBox.Show(ex.Message);
        }
finally {My_Connection.Close();}
}
```

VB.NET

```
Public Sub Store_it(ByVal content As Byte())
   Try
      My_Connection.Open()
      Dim insert As SqlCommand = New SqlCommand("insert into img
values (@picID)", My_Connection)
      insert.Parameters.Add(New SqlParameter("@pic",
SqlDbType.Binary)).Value = content
      insert.ExecuteNonQuery()
   Catch ex As Exception
      MessageBox.Show(ex.Message)
   Finally
      My_Connection.Close()
   End Try
End Sub
```

ولعرض الصورة في Picture Box مرة أخرى :

C#:

```
SqlDataAdapter.SelectCommand.CommandText = "select * from img";
SqlDataAdapter.Fill(dsPictures1);
byte[] arrPicture = ((byte[]) (dsPictures1.Tables[0].Rows[0]["pic"]));
MemoryStream ms = new MemoryStream(arrPicture);
```

```
pictureBox2.Image = Image.FromStream(ms);
```

VB.NET

```
SqlDataAdapter.SelectCommand.CommandText = "select * from img"
SqlDataAdapter.Fill(dsPictures1)
Dim arrPicture As Byte() =
(CType(dsPictures1.Tables(0).Rows(0)("pic"), Byte()))
Dim ms As MemoryStream = New MemoryStream(arrPicture)
pictureBox2.Image = Image.FromStream(ms)
```

سيتم الحديث في الجزء التالي عن استخدام Transport Layer & Network Layer Programming وبرمجة بروتوكولاتها في بيئة الدوت نيت.

Part 3
Transport & Network Layer Programming

Chapter6 Transport TCP & UDP (Classes & Members)

Chapter7 Synchronous Socket Programming

Chapter8 Asynchronous Socket Programming

Chapter9 Advanced Multicasting Systems

Chapter10 Voice Over IP Programming

Chapter11 Raw Socket Programming

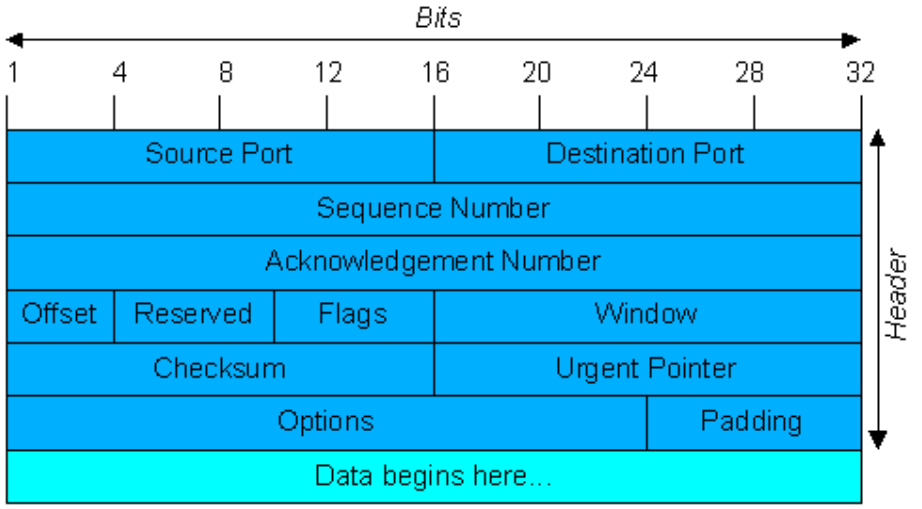
Chapter 6
Transport TCP & UDP
(Classes & Members)

- TCP Classes Members
- UDP Classes Members

: TCP & UDP Classes Members : 6

سوف نتحدث في هذا الفصل عن الـ TCP Classes والـ UDP Classes واهم الـMembers لكل منهما والتي يتم التعامل معها في الـ Transport Layer Classes ، ومن المعروف أن في هذه الطبقة إذا استخدمنا الـTCP يمكن التحكم بخصائص الإرسال ومنها الـOffset والـ Flags بالإضافة إلى وضع وتحديد الـ Source و الـDestination Port لدى الطرف المرسل والمستقبل.

أولا الـTCP Header ويتكون من 32 Bits للـ Packet الواحد حيث يتم فيه تخزين عنوان Port المرسل في 16 Bits والمستقبل في 16 Bits والرقم التسلسلي في 32 Bits ورقم التحقق بالإضافة إلى الـChecksum وفي النهاية يتم وضع الجزء الخاص بالبيانات وهو ما تم شرحه في الفصل الأول بتفصيل، لاحظ الشكل التالي:



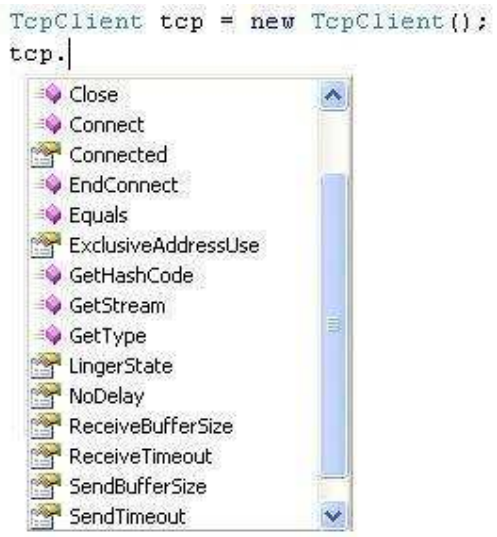
TCP Header

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 الـUDP Header ويتكون من 32 Bits من البيانات للـ Packet الواحد ويحتوي على عنوان المرسل 16 Bits والمستلم 16 Bits و الـChecksum والـ Total Length:

UDP Header

أولا الـClasses الخاصة بالـ TCP Connection Oriented Protocol :



1-TcpClient Class: ويستخدم لتعريف Simple TCP Socket إذ يمرر له الـIPAddress والـ Port للجهة التي نريد الاتصال معها ويحتوي هذا الكلاس على مجموعة من الـMethods والـ Properties ونلخصها بالتالي:

أولا: TCPClient Methods :

Connect: وتستخدم لأجراء عملية الاتصال مع الـserver حيث نمرر فيها عنوان الـIP الخاص بالـServer ورقم الـPort 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 الـTCP Socket.
GetStream: ويستخدم لقراءة الـStream من الـSocket في عملية الإرسال و الاستقبال وتستخدم عادة لربط الـSocket مع الـ NetworkStream Class.

ثانيا: أهم الخصائص TCPClient Properties :

LingerState : وتأخذ get أو Set لتحديد أو معرفة الـLinger 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 إذا كان الـSocket يسمح باستخدام الـClient Port أم لا ، وتستخدم عادة مع الـ SocketOption Class (لمزيد من المعلومات أنظر الـ (A) Appendixes)
SendBufferSize و ReceiveBufferSize: وتأخذ get أو Set لتحديد أو معرفة حجم الـBuffer المستخدم في الـstream والمعرف في TCP Client Object.

SendTimeout و ReceiveTimeout: وتأخـذ get أو Set لتحديـد أو معرفـة الوقـت المتاح لعملية الإرسال أو الإستقبال حيث يعطي Time Out في حالة أنه لم يجد الطرف الأخر خلال فترة زمنية معينة.

2-TcpListener Class: حيث تحتوي على مجموعة من الـMethods والـProperties وهي كما يلي:

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

أولا: أهم الدوال في الـ TcpListener:

AcceptSocket: وتستخدم لقبول عملية الاتصال مع الـClient.

Start : وهي Overloaded Method حيث انه في حالة تمرير رقم إليها يتم تحديد عدد الأجهزة التي تسمح بوجودها في الـQouee ، وفي حالة عدم تمرير رقم معين تصبح الـQouee غير محدد.

Stop : وتستخدم لإغلاق عملية التصنت ويفضل وضعها في الـFinally عند استخدام الـTry و الـCatch وحتى يتم إنهاء عملية التصنت حتى لو حدث أي Exception.

ثانيا: أهم الخصائص في TcpListener:

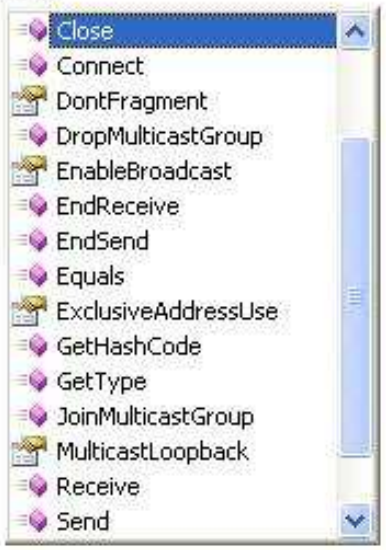
LocalEndpoint : حيث يرجع الـIP ورقم الـPort المستخدم في الـLocalEndpoint المحدد.

Server: ومن خلالها نستطيع الوصول إلى كل الخصائص و الدوال في الـTCP Server والتي شرحناها سابقا مثل الـAccept والـSendto والـReceive و Listen وغيرها

ثانيا الـClasses الخاصة بالـ 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 :

```
UdpClient udp = new UdpClient()
udp.
```



ومن أهم الدوال والخصائص الخاصة بها :

JoinMulticastGroup و DropMulticastGroup: لضـــم أو إلغـــاء عنـــوان أو مجموعة من العناوين من الـMulticast Group.
EnableBroadcast: وتأخذ Get أو Set لتفعيل الـBroadcasting في الـsocket.
MulticastLoopback: وتأخذ Get أو Set لمعرفة أو تحديد الـMlticast Loopback.

MulticastOption Class-2: ويستخدم في الـMulticasting 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 الـUDP 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 الـIP الذي نريد ضمه في الـGroup أو نجعل الـUser يدخل العنوان بنفسه بعد ذلك نربطها بالـSocket باستخدام الدالة 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 الـMulticast Option ونسند لها العنوان المحدد كما يلي:

C#:
```
MulticastOption mcastOption;
mcastOption = new MulticastOption(localIPAddr);
```

VB.NET:
```
Dim mcastOption As MulticastOption
 mcastOption = New MulticastOption(localIPAddr)
```

ولتفعيلها يجب تعريفها في الـ SocketOptions Method و تأخذ هذه الدالة ثلاثة بارومیترات الأول لتحديد مستوى التغيير على IP أو على IPv6 أو على Socket أو TCP أو UDP ، (لمزيد من المعلومات أنظر الـ A Appendixes في المرفقات) ، 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 الـMulticastOption Object والذي قمنا بإنشائه و كما يلي:

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

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

وهكذا بينا أهم الخصائص والتي يمكن التحكم بها في بيئة الدوت نيت والمتعلقة بالـTransport Layer Protocols و الـ Network Layer Protocols، في الفصل التالي سيتم الحديث بشكل مفصل عن بقية الـ Members والتي يتم التعامل معها في الـNetwork Layer Protocols وطرق التعامل مع الـ Synchronous Socket والـAsynchronous Socket.

Chapter 7
Network Layer & Synchronous Socket Programming

- Introduction to Socket Programming
- Synchronous Socket Programming
- Synchronous Socket Classes & Members

7: Network Layer & Synchronous Socket Programming :

في هذا الجزء سوف نبين بشكل أكثر تفصيلا عن برمجة طبقة الـNetwork Layer والـSocket وهي التي يتم التعامل معها لإرسال واستقبال البيانات بعد تحويلها من و إلى Stream عبر الشبكة، قمنا سابقا باستخدام الـTCP والـ UDP للإرسال وللاستقبال وبينا الفرق بينهما وفي هذا الجزء سوف نتحدث عن الـSocket Programming والـClasses والـMembers الخاصة بالـSocket Class..

7.1: Introduction to Socket Programming

من المعروف أن الـ Socket هي الأداة التي يتم نقل البيانات من خلالها من جهاز إلى أخر ولاستخدامها يلزم في البداية تعريف System.Net.Socket Namespace حيث يحتوي هذا Namespace على عدد كبير من الـClasses والتي يتم استخدامها في برمجيات الشبكة وسوف نتحدث عن أهمها وهو Socket Class إذ يمكننا من التعامل مع الـTCP أو الـUDP أو مع أي نوع أخر من البرتوكولات بشكل مباشر ويتكون الـSocket Object Method من ثلاثة باروميترات كما يلي:

C#:

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

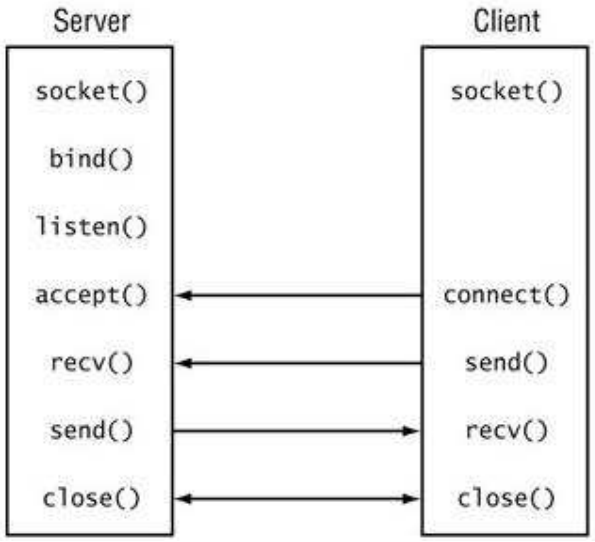
VB.NET:

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

حيث يتم في الباروميتر الأول تحديد نوعية الـIP Address والذي سوف تتعامل معه ويعطيك عدد كبير من الخيارات ومنها IPX والمستخدم في شبكات الـNovel أو ATM والمستخدم في شبكات الـATM Networks أو NetBIOS Address وغيرها ... ومن أهم هذه الخيارات الـInterNetwork وهو ما نستخدمه بشكل دائم مع البرمجيات الخاصة بالشبكات ويعرف على أن نوع الـ IP هو IPv4 وهو المعتاد مع نظام مايكروسوفت وأغلب أنظمة التشغيل المعروفة حاليا وفي المستقبل القريب جدا سيتم الإستغناء عنه وليحل محله الـIPv6 (لمزيد من المعلومات انظر الفصل الثاني)، في الباروميتر الثاني يتم تحديد نوع الـSocket أي هل سوف نستخدم Stream أو Dgram لإرسال البيانات أو Raw Socket (ويستخدم الـ Raw Socket مع البروتوكولات التي لا تطلب التعامل مع Port محدد ومن الأمثلة عليها الـ ICMP لمزيد من المعلومات انظر الفصل الخامس عشر) وبشكل دائم يتم استخدام الـStream في حالة إستخدام الـ TCP والـ Dgram في حالة استخدام الـUDP ، وأخيرا نحدد نوع البروتوكول المستخدم للإتصال هل هو من النوع UDP أو TCP أو بروتوكولات أخرى مثل الـICMP Internet Control Message Protocol أو الـ IGMP Internet Group Management Protocol أو اننا نريد مثلا إنشاء الـSocket لتعريف IP Security Header بإختيار IPSecAuthenticationHeader وغيرها، وهنا سوف نختار الـTCP أو UDP ومن المعروف أن برتوكول الـTCP هو برتوكول موجه وهذا يعني إجراء عملية التحقق من الوصول والتوصيل إلى شخص ما محدد أما برتوكول الـUDP فهو برتوكول سريع نسبيا و لكنه 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 الفصل الأول والثاني والثالث .

7.2 استخدام الـSynchronous Socket Programming لإنشاء TCP Connection:

تمر عملية الاتصال باستخدام الـTCP Socket Connection بمجموعة من المراحل وكما في الشكل التالي :



إذ تبدأ العملية في الـClient و الـserver بإنشاء الـSocket وكما 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 الـSocket مع الكمبيوتر الحالي باستخدام الدالة 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 الـPort المحدد باستخدام الدالة listener ويمكنك تحدد عدد الأجهزة التي سيتم قبولها ولوضع عدد غير محدد نمرر له الرقم -1 ثم نقوم بالموافقة على الاتصال باستخدام الدالة accept وكما يلي :

C#:
```
MySocket.Listen(-1);
MySocket.Accept();
```

VB.NET:
```
MySocket.Listen(-1)
 MySocket.Accept
```

ويتم استقبال البيانات من خلال الدالة 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 الـPort(5020 كمثال) حيث يتم ربطها بالـSocket باستخدام الدالة Bind وقمنا بتعريف Listen لا نهائي العدد -1..

ولتعريف برنامج الإرسال TCP Client باستخدام الـSocket لابد من تعريف الـSocket وإسناد عنوان الـServer ورقم الـPort بالـIPEndPoint Instance Object ثم إرسال البيانات باستخدام الدالة Send وتتم عملية الإرسال بما تم تعريفه في الـsocket حيث سنستخدم Stream Socket وكما يلي :

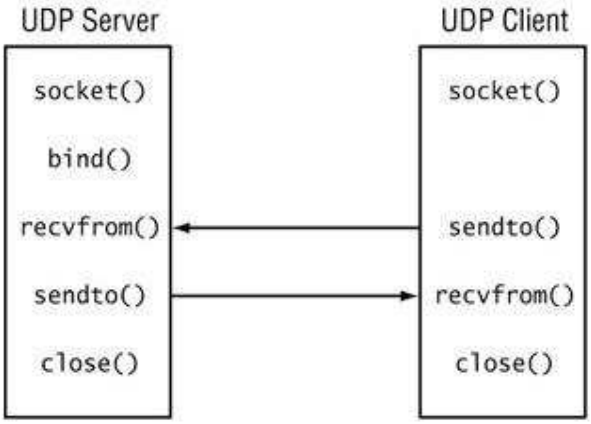
C#:
```
String str = Console.ReadLine();
ASCIIEncoding asen = new ASCIIEncoding();
byte[] msg = asen.GetBytes(str);
Socket MySocket = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);
IPEndPoint remote = new
IPEndPoint(IPAddress.Parse("192.168.1.101"), 5020);
MySocket.Connect(remote);
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
```

7.3 استخدام الـSynchronous Socket Programming لإنشاء UDP Connectionless :

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



وتشبه عملية استخدام الـUDP الـTCP إلا حد كبير إذ تبدأ العملية في الـClient و الـServer بإنشاء الـSocket كما يلي :

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

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

ثم ربط الـSocket مع الكمبيوتر الحالي باستخدام الدالة 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 الدالة ReceiveFrom حيث نعرف في البداية EndPoint Reference بناء على ما تم تعريفه في السابق ونمرره كـ Reference Object مع مصفوفة

الـByte إلى الدالة ReceiveFrom ومن ثم نستطيع تحويل المصفوفة إلى String من خلال الدالة GetString الموجودة ضمن ASCII Class 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 الدالة SendTo حيث نمرر لها البيانات بعد تحويلها من String إلى Byte Array وحجم البيانات المرسلة إذ يمكننا معرفته من خلال الخاصية Length ونوع الـFlags واخيرا نمرر له الـEndPointObject والذي يعرف الـ IP Address والـ Port لـSocket وكما يلي كمثال لإرسال Broadcast Message :

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)
```

7.4 Synchronous Socket Classes Members:

1- IPAddress Class : ويستخدم لتعريف الـ IPAddress حيث يمكن إسناده إلى الـIPEndPoint كمثال ويمكن أن نكون من خلاله Array of Address في حالة إرسال رسالة ما إلى مجموعة أما الصيغة العامة له فهي كما يلي:

C#:
```
IPAddress newaddress = IPAddress.Parse("192.168.1.1");
```

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

ويمكن الاختيار بين اربعة خيارات في تحديد العنوان وهي كما يلي :
Any ويستخدم لتمثيل أي عنوان متاح على الشبكة (عند الإستقبال ويعني الإستقبال من الكل).
Broadcast ويستخدم لتمثيل البث الإذاعي لجميع الأجهزة على الشبكة (عيث يقوم بعملية حساب الـIP Broadcast دون تدخل المستخدم ، لمعرفة كيفية حساب الـBroadcast أنظر الفصل الثاني).
Loopback ويستخدم لتمثيل العنوان المعروف للـ loopback وهو 127.0.0.1

كما يدعم مجموعة من الدوال وأهمها :

Equals تستخدم هذه الدالة بشكل عام للمقارنة بين tow Objects وهنا ستستخدم للمقارنة بين عنوانين وترجع True إذا كانا متشابهين و False إذا كانا مختلفين.

GetHashCode وتستخدم لإرجاع العنوان إلى صيغة Hash Code

HostToNetworkOrder ويرجع الجزء الخاص بالـNetwork من العنوان

NetworkToHostOrder ويرجع الجزء الخاص بالـHost من العنوان

IPEndPoint Class : حيث استخدمناه لتحديد العنوان والـPort للـ 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 الـSocket Class وهي كما يلي:

AddressFamily ويرجع مجموعة العناوين المعرفة على الـSocket

Available ويرجع حجم البيانات الجاهزة للقراءة من الـSocket

Blocking ويعطي Get أو Set لمعرفة إذا كان الـsocket يستخدم الـBlocking Mode أم لا

Connected وتستخدم هذه الخاصية بكثرة لمعرفة إذا كان الـSocket متصل مع الـRemote Host أم لا

ProtocolType ويستخدم لمعرفة البروتوكول الذي يستخدم في الـSocket

RemoteEndPoint ويرجع معلومات عن الـSocket الذي يستخدم مع الـ Remote Host

وكمثال لاستخداماتها:

C#:

```
using System;
using System.Net;
using System.Net.Socket;
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.Socket

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 . . .
```

وهكذا بينا كيفية برمجة الـ Synchronous Socket في بيئة الدوت نيت ، سوف نتحدث في الفصل التالي عن برمجة Asynchronous Socket في بيئة الدوت نيت.

Chapter 8
Asynchronous Socket

- Asynchronous Socket Class and its members

- Applied Asynchronous Socket in Dot Net

8: Asynchronous Socket Programming :

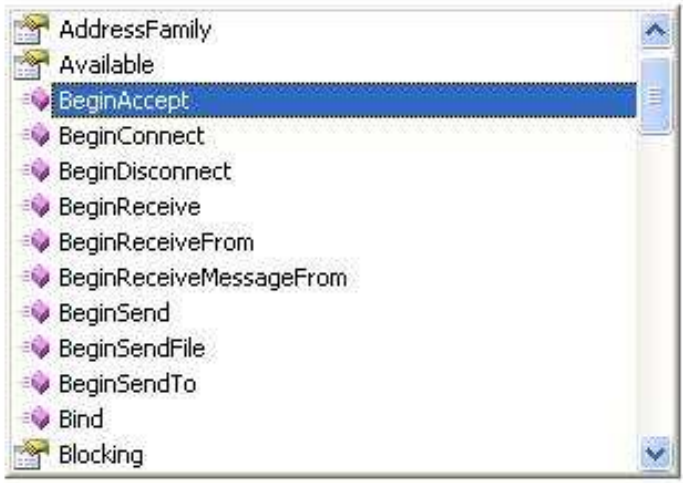
سوف نتحدث في هذا الجزء عن استخدام الـ Asynchronous Socket بشكل أكثر تفصيلا عما تحدثنا به سابقا وسوف نطبق مجموعة من الأمثلة العملية على استخدام الاتصال الغير متزامن في برمجيات الشبكات ...

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 لكن لا ينصح أبدا استخدامه في حالة إذا كانت الجهة المستقبلة للبيانات تستخدم Slow Connection كاعتماد الشبكة على الـDialup لربط الجهازين المرسل مع المستقبل أو في حالة إذا كان هنالك مجموعة كبيرة من المستخدمين تستخدم الـServer حيث يمنع الأسلوب المتزامن بقية المستخدمين على الشبكة من إجراء عملية الإرسال في حالة كون الـServer 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 الـThread وباستخدام نفس الـConnection ، أو كما ذكرنا سابقا في حالة إذا كان الاتصال بطيء نسبيا أو انه يوجد عدد مستخدمين يستخدمون نفس الـServer ..

أولا Asynchronous Socket Class and its members :

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

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



1- BeginAccept و تستخدم لقبول الـClient Request وإسناده إلى الـ Object AsyncCallback وباستخدام هذه الطريقة سوف يتمكن الـServer من استقبال عدد من الـClients Requests في نفس الوقت وبدون الحاجة لانتظار الانتهاء من العملية الجارية حيث يتم في كل مرة استدعاء الدالة باستخدام الـAsyncCallback 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 الـClient Request في Callback Reference منفصل عن السابق وهنا لابد من إنشاء method لاستقبال الـClient Request وإنهاء الـ Client Accepted Object باستخدام الدالة EndAccept:

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 الـBegin Accept Method تأخذ عدة أشكال كما يلى:

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

```
MySocket.BeginAccept(Socket accept_Socket ,int Data_ Receive_Size ,
AsyncCallback , object state)
```

2- BeginConnect وتستخدم لربط الـClient مع الـServer أو الـRemote Machine كـAsynchronous Connection حيث يسند لها الـIPEndPoint Instance Object (يحدد فيه الـ IP Address ورقم الـPort) ويسند لها أيضا الـAsynchronous Callback والـState 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(Connect_him),
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)
MySocket.BeginConnect(ipend, new AsyncCallback(Connect_him),
MySocket)
```

في الـConnect_him Method يتم تحديد الـCallBack Socket كما يلي:

C#:

```
    public static void Connect_him (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 الـClient وتخزينها في Byte Array والصيغة العامة لها كما يلي:

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

وتستخدم 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 الدالة BeginReceiveFrom لإستقبال البيانات من موقع محدد باستخدام الـUDP حيث يضاف إلى التركيب السابق IPEndPoint Refrance Object .

4- BeginSend وتستخدم لإرسال بيانات إلى الطرف المستقبل عبر الـ 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 الدالة BeginSendto لإرسال البيانات إلى Remote Host محدد باستخدام الـUDP حيث يضاف إلى التركيب السابق IPEndPoint Refrance Object .

5- كما تم إضافة مجموعة من الدوال الجديدة في الدوت نيت 2005 وهي:
BegonDiconnect لإنهاء الاتصال و **BeginSendFile** لإرسال ملف والـ**BeginReceiveMessageFrom** والتي تستخدم لإستقبال عدد محدد من البيانات وتخزينها في مكان محدد في الـBufer ..

التركيب التالي:

تأخذ الـ **BeginSendFile:**

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

والـ**BeginReceiveMessageFrom** التركيب التالي:

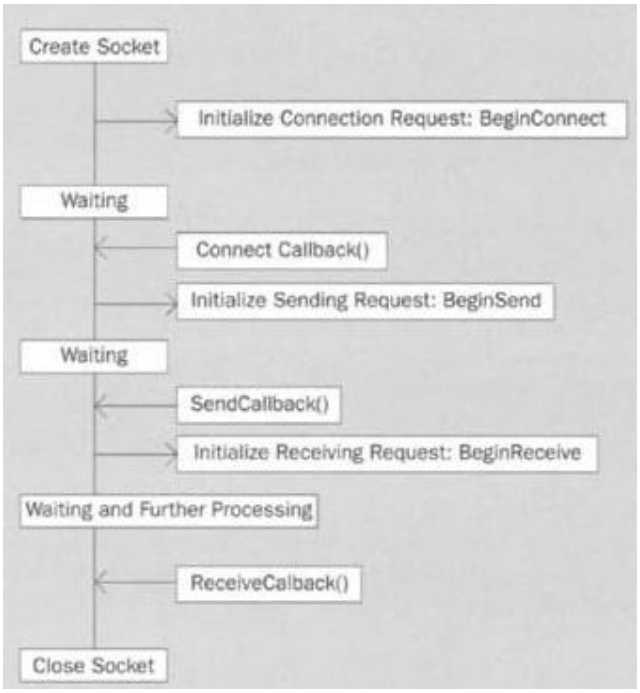
```
MySocket.BeginReceiveMessageFrom(byte Buffer ,int offset,int
size,SocketFlags sf,ref EndPoint,AsyncCallback ascb,object state)
```

والـ**BegonDiconnect** التركيب التالي:

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

ثانيا: تطبيقات الـAsynchronous Socket في الدوت نيت :

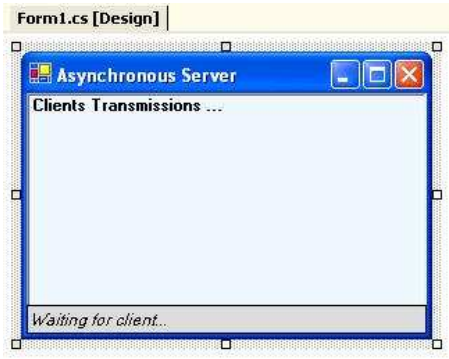
تمر عملية الاتصال الغير متزامن بمجموعة من المراحل تبدأ بإنشاء الـSocket Object في الـServer Side بعد ذلك يتم تعريف الـBeginConnect لبدأ Asynchronous Connection على الـSocket حيث يتم إسناد IPEndPoint Object والـMethod Asynchronous Callback والـState Object لها وتبدأ في هذه الحالة عملية الاتصال بالـSocket ، وبعد ذلك تمرر إلى الـBeginAccept لقبول الـClient Request حيث يتم قبول الطلب ويرسل Acknowledgement إلى الـClient ليعلمه فيها بقبول الجلسة وإمكانية البدء للإرسال و يستطيع الـClient بعد الموافقة على الجلسة البدء بالإرسال باستخدام الدالة BeginSend ويستقبل الـServer الرسالة من الـClient باستخدام الدالة BeginReceive وكما ذكرنا سابقا فإن لكل عملية Begin تقابلها الدالة End للاستعداد لإجراء عملية أخرى على نفس الـThread في البرنامج وهو ما ميز الاتصال الغير متزامن عن الاتصال المتزامن.



نلاحظ في الشكل السابق بقاء الـ Server بوضعية الإنتظار إذ أنه وفي كل عملية طلب للـRequest ما يتم إنهائة باستخدام الـ Endxxx Method وتستدعى هذه الـMethod باستخدام الـ Delegate المخصص لإجراء عملية الإتصال الغير متزامن وهي AsyncCallback وتشبه هذه العملية إلى حد كبير عملية الدخول المباشر إلى المعالجة إذ لا تنتظر إنهاء الـ Server لعملية المعالجة الحالية في حالة ورود طلب جديد وهكذا يسلم أي طلب جديد إلى الـ Endxxx Method لإنهائه وإبقاء حالة الـ Server بوضع الإنتظار ...

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

وللبدء قم بإنشاء مشروع جديد كما في الشكل التالي:



سوف نستخدم الـNamespaces التالية:

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

VB.NET:
```
Imports System.Net
Imports System.Net.Socket
Imports System.Text
```

في الـGlobal Declaration (أي بعد تعريف الـMain 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 الـForm_Load قم بإضافة الكود التالي حيث سنعرف Connection يعتمد على الـTCP ويعمل على الـPort 5020 ثم تعريف عملية قبول الاتصال باستخدام الـBeginAccept :

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 الـ Accepted Request باستخدام الـEndAccept Method وبعد ذلك إرسال Acknowledgement إلى الـClient تخبره فيها بقبول الطلب وترسل باستخدام الـBeginSend 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 الـBeginSend 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 الـBeginReceive وكما يلي:

C#:

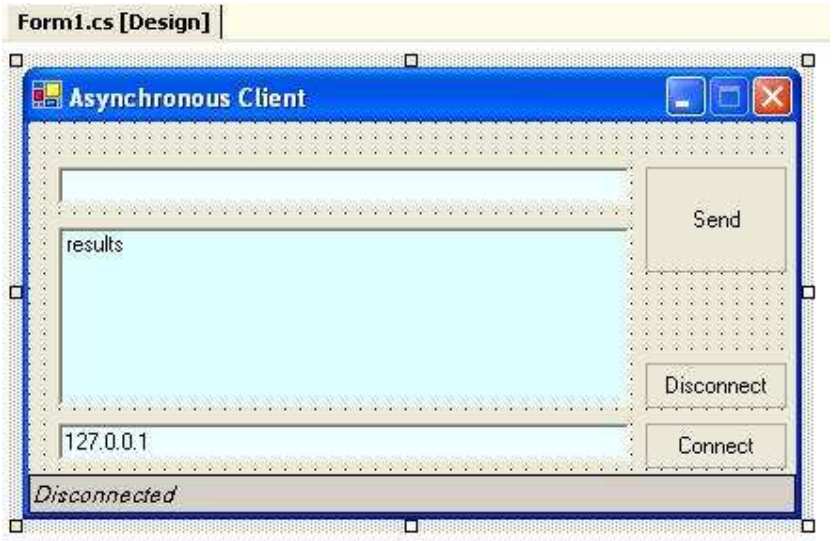
```
void ReceiveData(IAsyncResult iar)
{
Socket client = (Socket)iar.AsyncState;
int recv = client.EndReceive(iar);
        if (recv == 0)
        {
                client.Close();
                conStatus.Text = "Waiting for client...";
        server.BeginAccept(new AsyncCallback(AcceptConn), server);
                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 الـServer والآن سوف نقوم بإنشاء برنامج الـClient وللبدء قم بإنشاء مشروع جديد كما في الشكل التالي:



سوف نستخدم الـNamespaces التالية:

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

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

في الـGlobal Declaration (أي بعد تعريف الـMain 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 الـConnect Button قم بكتابة الكود التالي:

```
.
.
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 الـBeginReceive 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 الـSend 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 الـBeginSend 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 الـBeginReceive 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 الـClient لا يختلف كثيرا عن برنامج الـServer حيث نعرف في الـServer الـSocket Connection والـBeginAccept Method أما في الـClient فنعرف الـSocket Connection و الـBeginConnect Method وتبقى عملية الإرسال والاستقبال هي نفسها في Server والـClient ...

سيتم الحديث في الفصل التالي عن طرق برمجة أنظمة الـ Multicasting Conference Systems في بيئة الدوت نيت.

Chapter 9
Advanced Multicasting Systems

- Architecture of Multicast Socket

- Using Multicast Socket with .NET

- Multicast Conferencing Systems:

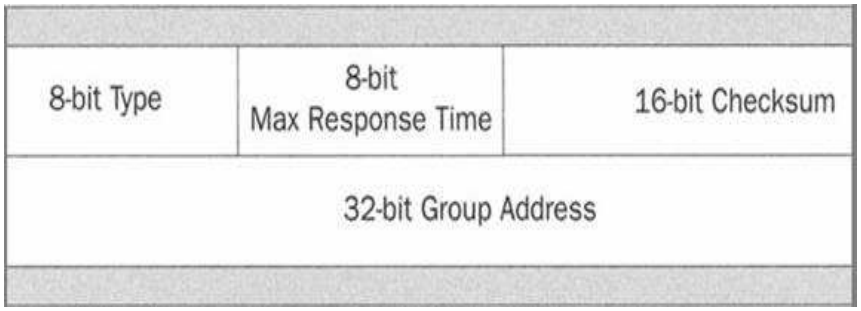
    1. Full/Half Duplex Multicast Video Conferencing System.
    2. Full/Half Duplex Multicast Desktop Conferencing System.
    3. Full/Half Duplex Multicast Text Conferencing System

9: Advanced Multicasting Systems :

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

أولا : Architecture of Multicast Socket :

من المعروف انه يتم التعامل مع الـMulticasting عبر برتوكول الـUDP وباستخدام الـClass D Subnet Mask وتتم عملية إدارة المجموعات باستخدام برتوكول الـ – IGMP Internet Group Management Protocol والذي هو جزء من الـ Internet Protocol Model وكما يتضح من الشكل التالي فإن برتوكول الـIGMP 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 الـTTL Time to Live والذي يحدد فيه العمر الافتراضي لكل رسالة بناء على عدد الـ Hops التي سيمر منها الـ Packet، ونوع العملية الإدارية (ضم إلى مجموعة ، إلغاء من مجموعة ، أو إرجاع معلومات عن المجموعة Membership Query) وأخيرا عنوان المجموعة التي يتم تحديدها برمجيا ضمن الـRange المحدد للـ Class D .



وتم تخصيص الـRange في الـIP Multicasting من 224.0.0.0 إلى 239.255.255.255 ونستطيع تحديده بثلاثة طرق إما بشكل يدوي Static أو Dynamic أو على أساس الـScope-Relative وبشكل عام تستخدم هذه التوزيعات كما يلي كمثال:

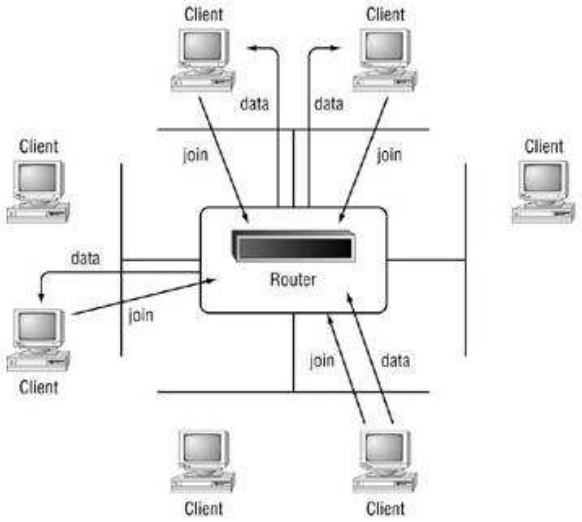
كمثال التخصيص 224.0.0.1 يستخدم في جميع الشبكات المحلية حيث لا يتم تمريره إلى شبكة أخرى عبر الـRouter. لمعرفة جميع التخصيصات للـ Multicasting انظر الرابط التالي الخاص بمنظمة الـ Iana:

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

يتم نقل الـMulticast Packets بين الـBackbone Tunnels كـ Unicastحيث يتم إرسالها من داخل الشبكة إلى الـRouter و ترسل من Router إلى أخر عبر الـBackbone Tunnel بأسلوب الـUnicast وهو ما يوفر الكثير من الـBandwidth في الشبكة حيث ترسل نسخة واحدة إلى الـRouter ويقوم هو بتوزيعها على الأجهزة كـBroadcast ، المشكلة الوحيدة في الـMulticast هو انه يعتمد بشكل كامل على استخدام الـ UDP Connectionless Protocol.

ويمكننا استخدام الـMulticasting في ثلاثة أنواع من الشبكات وهي شبكات الـ Peer to Peer حيث لا وجود لجهاز Server والكل يستقبل و يرسل من و إلى الـGroup الذي هو فيه،

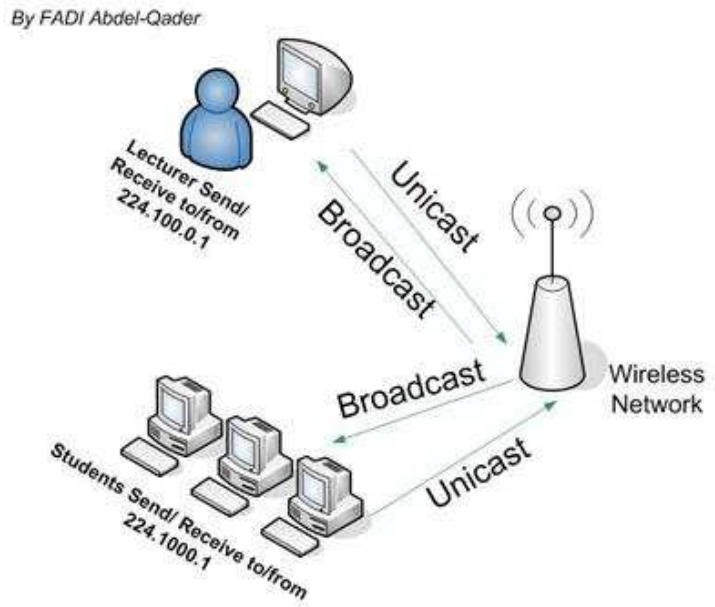
والنوع الثاني Server Based Network حيث يتم إرسال رسالة واحدة إلى الـServer ويقوم الـServer بتوزيعها على بقية الأجهزة في الشبكة، أما النوع الثالث فيتم من خلال الـRouter، وكما يتضح من الشكل التالي فإن عملية الإرسال تتم بعد انضمام الـClient إلى المجموعة التي تملك الـIP Multicast ويرسل الـClient رسالة واحدة إلى الـRouter حيث يقوم الـRouter بتوزيعها على الأجهزة في المجموعة مستخدما الـRouting Table.



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

1 – الإرسال من واحد إلى مجموعة One to Group:

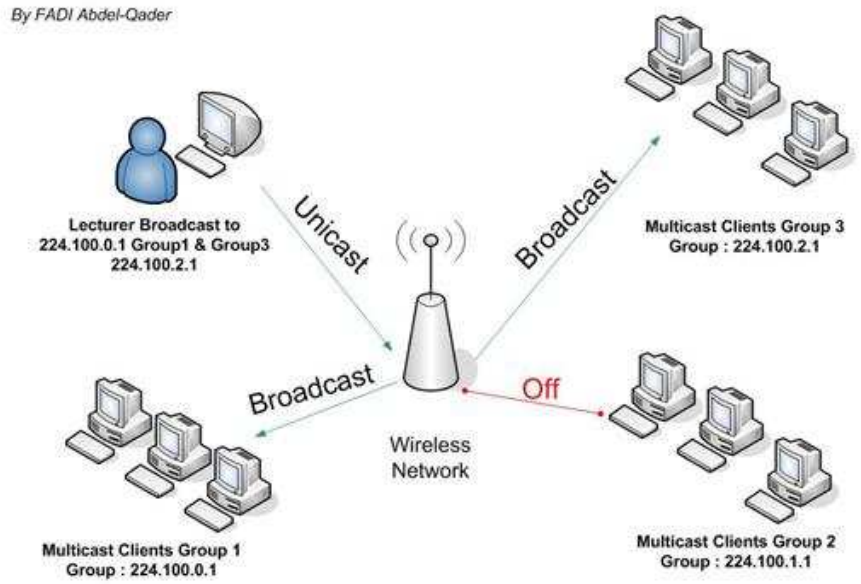
وفيه يملك الـSender User نفس الـIP Multicasting الذي يملكه الـReceiver Users ويتم الإرسال من داخل الـGroup إلى جميع أعضائه حيث ترسل كـ Unicast إلى الـAccess Point حيث يقوم بتوزيعها على كافة الأعضاء في المجموعة بأسلوب الـBroadcast وكما في الشكل التالي:



Full Duplex Multicast System for one Group

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

وفيه قد يكون الـIP Multicasting للـ Sender User مختلف عن Receiver Users ويتم الإرسال من User داخل الـGroup 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 Socket with .NET:

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

من العمليات الأساسية في التعامل مع الـMulticasting :

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

لا تلزم عملية الانضمام إلى الـMulticast Group أي عمليات تحقق سوى التصنت على الـport والـMulticasting IP المحدد ، ويتم ذلك بعد تعريف udpClient Object وباستخدام الـJoinMulticastGroup Method يتم تعريف الـMulticasting IP 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 الـ Methods **JoinMulticastGroup** و **DropMulticastGroup** لضم أو إلغاء عنوان أو مجموعة من العناوين من الـMulticast 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 الـUDP 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 الـIP الذي نريد إدخاله في الـGroup أو نجعل الـUser يدخل العنوان بنفسه نربطها بالسكوت باستخدام الدالة 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 الـMulticast 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 الدالة ثلاثة باروميترات 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 الباروميتر الثاني نحدد نوع التغيير حيث نريد إضافة عضوية ويمكن الاختيار بين إضافة عضويه AddMembership أو إلغاء عضوية DropMembership وأخيرا نسند إليه الـMulticastOption 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 الـMulticasting IP لابد أولا من تعريف الـ Socket Object باستخدام الـUDP Connection وإسناد الـMulticasting IP ورقم الـPort إلى الـIPEndPoint Object ... ونستطيع الإرسال باستخدام الـsendto method حيث نسند لها الـdata as Bytes Array والـIPEndPoint 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 الـMemory Stream لتخزين الصورة في الذاكرة على هيئة Stream ثم تحويلها إلى Byte Array وبعد ذلك إرسالها باستخدام الـsendto Method وكما يلي:

C#:

```
MemoryStream ms = new MemoryStream();
PictureBox1.Image.Save(ms,System.Drawing.Imaging.ImageFormat.Jp
eg);
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 الـIP Multicast الخاص بالمجموعة و الانضمام إليه ثم استقبال البيانات باستخدام الـReceive 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 الـmemory Stream لاستقبال البيانات من الـReceive Method وتخزينها في الذاكرة على هيئة Stream Data ثم تحويلها إلى صورة مرة أخرى باستخدام الـimage.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 الـMulticasting في برمجيات الشبكات :

1- مـن الملاحـظ أننـا لا نسـتطيع اسـتخدام الـNetwork Stream لعمليـة إرسـال الـMulticasting إذ يتطلب استخدامها وجود TCP Socket Connection وهو غير متاح في الـMulticasting ويستعاض عنها باستخدام الـmemory Stream لإرسال Binary Stream عبر الـsendto method ...

2- لا يمكنك استخدام الـMulticastingكـloopback في حالة عدم وجود شبكة أو اتصال لذلك لن تستطيع تجربة أي من تطبيقات الـMulticasting في حالة عدم اتصالك بالشبكة.

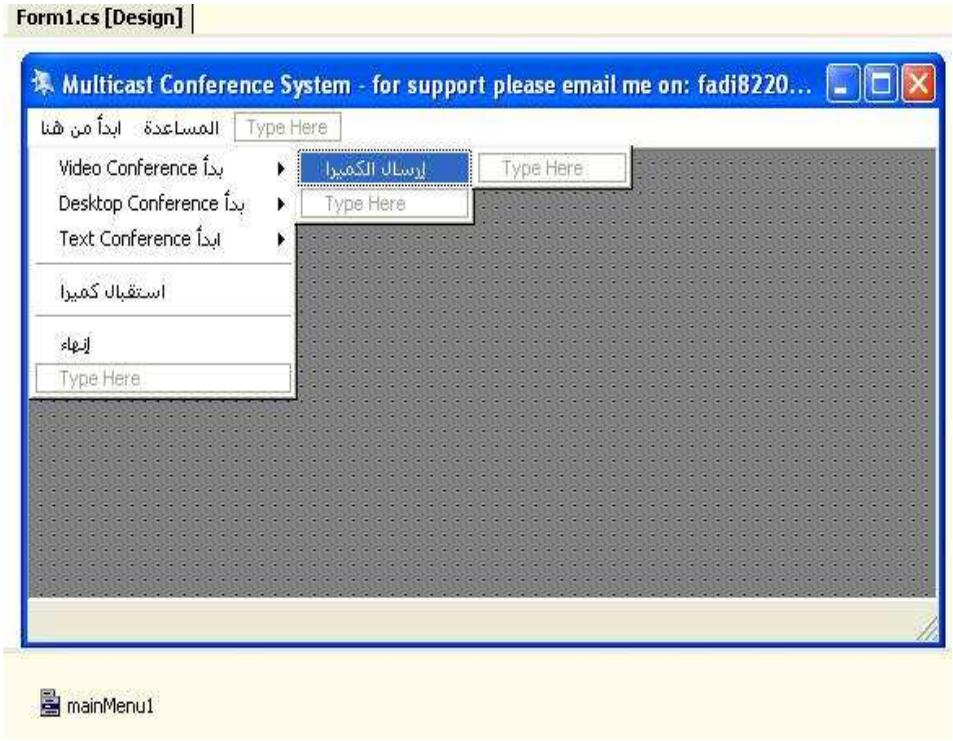
3- يمكن لكل جهاز أن ينضم إلى أكثر من مجموعة بحيث يستقبل من جهات متعددة، كذلك يستطيع الإرسال إلى عدة مجموعات.

4- في العادة تكون السعة المسموحه لإرسال الـMulticasting Data عبر الـ sendto Method محدودة والسبب الأساسي لهذه المشكلة هي استخدام الـ Multicasting بروتوكول الـ UDP والذي لا يمكن تجزئة الـ Data المرسلة في حالة استخدام حيث لا يدعم التوصيل وفـق الترتيـب ممـا يحصـر إمكانيـات الإرسـال بـالحجم الأقصـى للـ Ethernet Encapsoulation.

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 (Full Duplex System) بالإضافة إلى إمكانية عرض المحاضرة على الـ Power Point Slides (By Get Desktop Screen)كما يستطيع الطلاب التحدث مع الأستاذ باستخدام Text Chatting والـVoice Chatting...

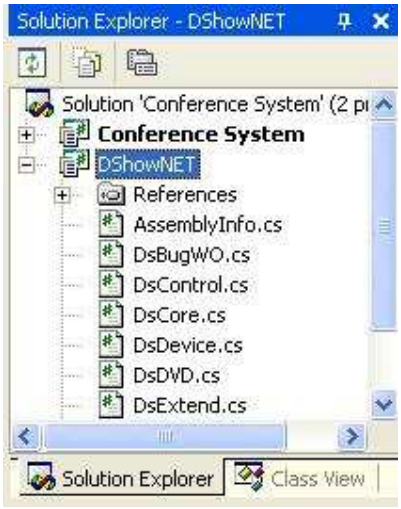
سوف نقوم هنا بتقسيم نظام المؤتمرات إلى ثلاثة أنظمة رئيسية وهي نظام مؤتمرات الفيديو ونظام مؤتمرات سطح المكتب ونظام المؤتمرات النصية وسنضيف عليه لاحقا نظام المحادثة الصوتية، في البداية سوف نقوم بإنشاء الشاشة الرئيسية للبرنامج و كما في الشكل التالي:



1- Full/Half Duplex Multicast Video Conferencing System :

وفرت لنا Microsoft مجموعة من الـClasses الخارجية والتي تتعامل مع الـ DirectX 9 مباشرة حيث نستطيع استخدامها لتعامل مع الكاميرا أو الـScanner أو الصوت أو أي طرفية أخرى وفي هذا التطبيق سوف نستخدم الـDirect Show Dot Net Classes لالتقاط صورة عبر الكاميرا وعرضها على الـPicture box حيث نستطيع إرسالها لاحقا إلى الـMulticast Group باستخدام الـmemory Stream والـSendto method وهو ما بيناه سابقا ..

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

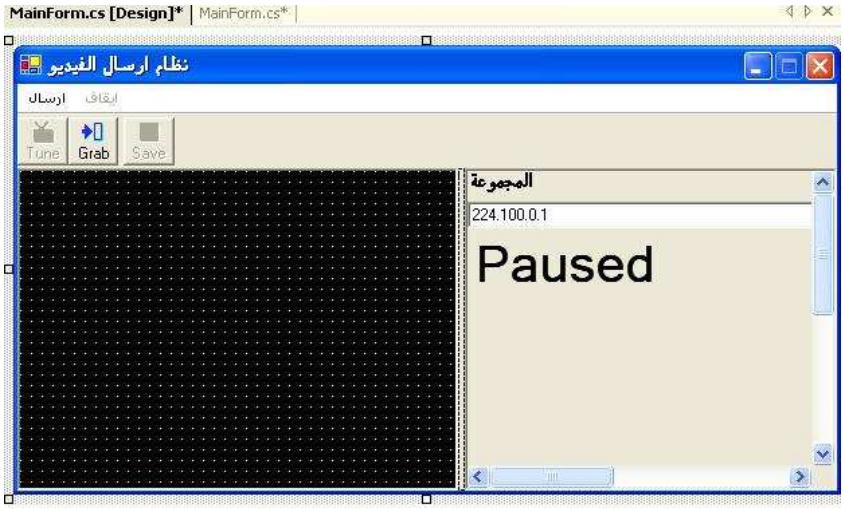


وحتى نتعامل معها سوف نستدعيها باستخدام :

C#:
```
using DShowNET;
using DShowNET.Device;
```

VB.NET:
```
imports DShowNET
imports DShowNET.Device
```

وسيكون شكل برنامج إرسال الكاميرا كما في الشكل التالي:



سوف نستخدم الـDeviceSelector Class لاختيار 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 لالتقاط 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 &H3) <> 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 نستدعي دالة إرسال الصورة والتي قمنا بإنشائها من خلال استخدام Timer ولهدف استمرار عملية الإرسال كل فترة معينة، وكما يلي:

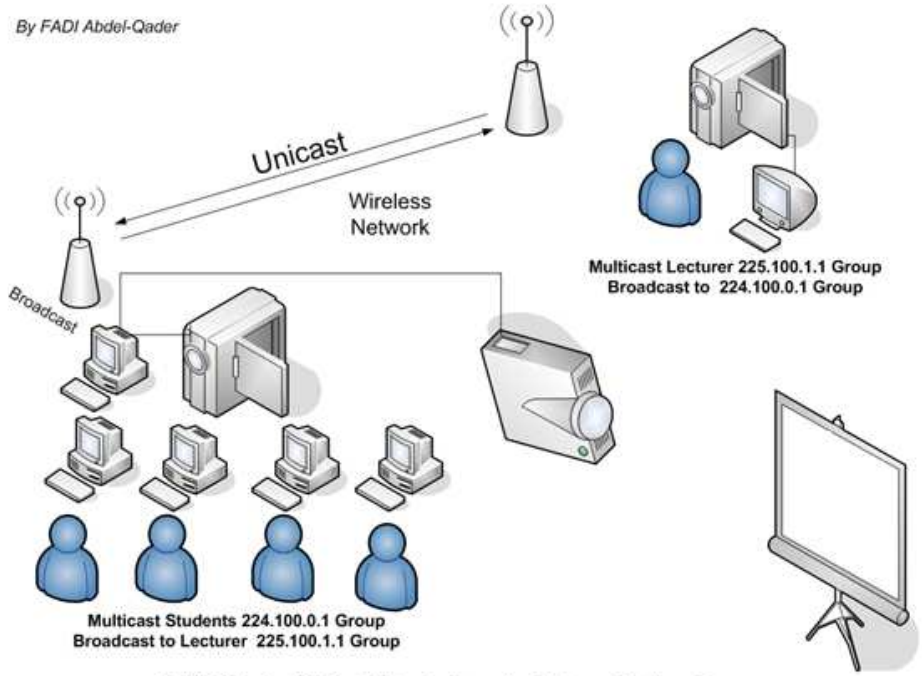
C#:

```
try
{
MemoryStream ms = new MemoryStream();
pictureBox.Image.Save(ms,System.Drawing.Imaging.ImageFormat.Jpe
g);
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 الـAccess Point بأسلوب الـUnicast وسوف يتولى الـAccess Point توزيع البيانات إلى جميع الأعضاء المنضمين إلى الـMulticast Group ويرسلها لهم كـBroadcast وكما في الشكل التالي:



Full/Half Duplex Multicast Video Conferencing System – For Class Room
That uses a Peer-to-Peer Wireless Network

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 أيضا فإن عملية الإرسال بين الـAccess Point1 والـAccess Point2 تتم باستخدام الـUnicast ...

وحتى يستطيع الطلاب رؤية أستاذهم والأستاذ رؤية طلابه ، لابد من إنشاء برنامج الاستقبال حيث سنستخدم نفس الـmethod التي شرحنها سابقا لاستقبال الصورة وللبدء قم بعمل New Form جديد كما في الشكل التالي:

سوف نستخدم الـ Namespaces التالية لاستقبال الصورة من الـMulticast Group :

C#:
```
using System.Net.Socket ;
using System.Net;
using System.IO;
using System.Threading;
```

VB.NET:
```
imports System.Net.Socket
imports System.Net
imports System.IO
imports System.Threading
```

والـ method التالية للاستقبال:

C#:
```
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 الـThreading ، وحتى نقوم بذلك قم بعمل Timer وضع فيه الكود التالي لاستخدام الـThreading :

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){}
```

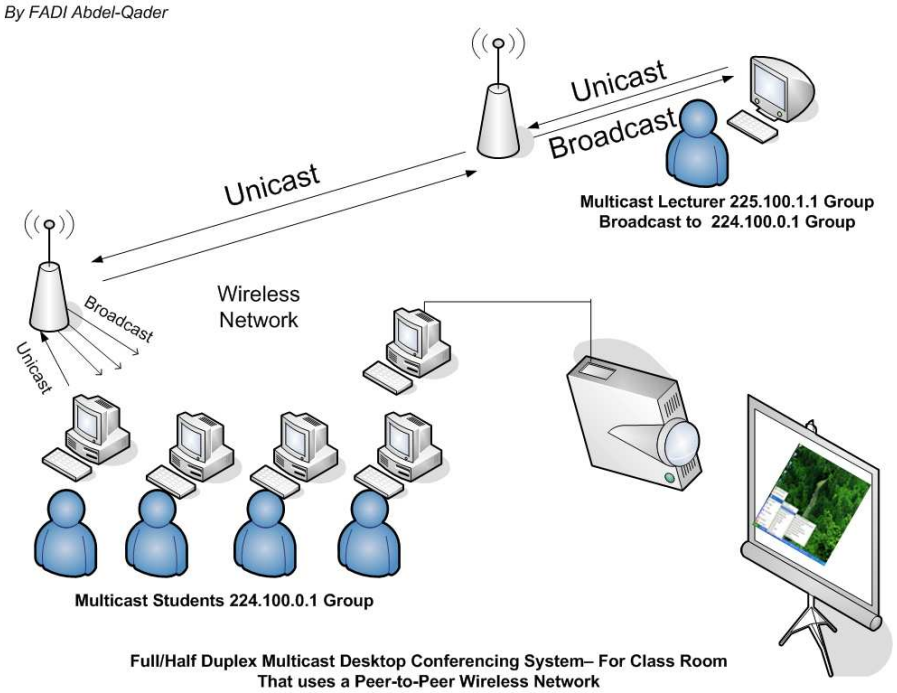
VB.NET:

```
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
```

وهكذا قد تم الانتهاء من المشروع الأول وهو الـVideo Conference System ، وحتى يستطيع المحاضر عرض المحاضرة باستخدام برنامج الـPower 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 الـ 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 الـMulticast Group ومن ثم استقبالها وعرضها على الطلاب باستخدام Data Show Projector .
ويوضح الشكل التالي مخطط عمل المشروع:



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 لكن 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:

ولإلتقاط صورة سطح المكتب سنستخدم دوال الـ API ومنها الـ GetDesktopWindow Method والموجودة ضمن الـuser32.dl والـBitBlt Method لرسم الصورة، ثم سنقوم بتحويل الصورة إلى Byte Array وحتى نستطيع إرسالها إلى الـ Clients ويتم ذلك كما يلي :

C#:

```
using System;
using System.Drawing;
using System.Drawing.Imaging;
using System.Runtime.InteropServices;
using System.IO;

public class ScreenCapture
{
[DllImport("user32.dll")]
private static extern IntPtr GetDesktopWindow();

[DllImport("gdi32.dll")]
private static extern bool BitBlt(
        IntPtr hdcDest, // handle to destination DC
        int nXDest, // x-coord of destination upper-left corner
        int nYDest, // y-coord of destination upper-left corner
        int nWidth, // width of destination rectangle
        int nHeight, // height of destination rectangle
        IntPtr hdcSrc, // handle to source DC
        int nXSrc, // x-coordinate of source upper-left corner
        int nYSrc, // y-coordinate of source upper-left corner
        System.Int32 dwRop // raster operation code
);
```

```
private const Int32 SRCCOPY = 0xCC0020;
[DllImport("user32.dll")]
private static extern int GetSystemMetrics(int nIndex);

private const int SM_CXSCREEN = 0;
private const int SM_CYSCREEN = 1;

public Size GetDesktopBitmapSize()
{
return new Size(GetSystemMetrics(SM_CXSCREEN),
GetSystemMetrics(SM_CYSCREEN));
}

public byte[] GetDesktopBitmapBytes()
        {
Size DesktopBitmapSize = GetDesktopBitmapSize();
Graphics Graphic = Graphics.FromHwnd(GetDesktopWindow());
Bitmap MemImage = new Bitmap(DesktopBitmapSize.Width,
DesktopBitmapSize.Height, Graphic);
Graphics MemGraphic = Graphics.FromImage(MemImage);
IntPtr dc1 = Graphic.GetHdc();
IntPtr dc2 = MemGraphic.GetHdc();
BitBlt(dc2, 0, 0, DesktopBitmapSize.Width,
DesktopBitmapSize.Height, dc1, 0, 0, SRCCOPY);
Graphic.ReleaseHdc(dc1);
MemGraphic.ReleaseHdc(dc2);
Graphic.Dispose();
MemGraphic.Dispose();
Graphics g = System.Drawing.Graphics.FromImage(MemImage);
System.Windows.Forms.Cursor cur =
System.Windows.Forms.Cursors.Arrow;
cur.Draw(g,new Rectangle(System.Windows.Forms.Cursor.Position.X-
10,System.Windows.Forms.Cursor.Position.Y-
10,cur.Size.Width,cur.Size.Height));
MemoryStream ms = new MemoryStream();
MemImage.Save(ms,System.Drawing.Imaging.ImageFormat.Jpeg);
return ms.GetBuffer();
}
}
```

VB.NET

```
Imports System
Imports System.Drawing
Imports System.Drawing.Imaging
Imports System.Runtime.InteropServices
Imports System.IO
```

```
Public Class ScreenCapture : Inherits System.MarshalByRefObject
   <DllImport("user32.dll")> _
   Private Shared Function GetDesktopWindow() As IntPtr
   End Function

   <DllImport("gdi32.dll")> _
   Private Shared Function BitBlt(ByVal hdcDest As IntPtr, ByVal
nXDest As Integer, ByVal nYDest As Integer, ByVal nWidth As
Integer, ByVal nHeight As Integer, ByVal hdcSrc As IntPtr, ByVal
nXSrc As Integer, ByVal nYSrc As Integer, ByVal dwRop As
System.Int32) As Boolean
   End Function

   Private Const SRCCOPY As Int32 = &HCC0020
   <DllImport("user32.dll")> _
Private Shared Function GetSystemMetrics(ByVal nIndex As Integer)
As Integer
End Function

 Private Const SM_CXSCREEN As Integer = 0
 Private Const SM_CYSCREEN As Integer = 1
 Public Function GetDesktopBitmapSize() As Size
 Return New Size(GetSystemMetrics(SM_CXSCREEN),
GetSystemMetrics(SM_CYSCREEN))
   End Function

   Public Function GetDesktopBitmapBytes() As Byte()
      Dim DesktopBitmapSize As Size = GetDesktopBitmapSize()
      Dim Graphic As Graphics =
Graphics.FromHwnd(GetDesktopWindow())
      Dim MemImage As Bitmap = New
Bitmap(DesktopBitmapSize.Width, DesktopBitmapSize.Height,
Graphic)
      Dim MemGraphic As Graphics =
Graphics.FromImage(MemImage)
      Dim dc1 As IntPtr = Graphic.GetHdc()
      Dim dc2 As IntPtr = MemGraphic.GetHdc()
      BitBlt(dc2, 0, 0, DesktopBitmapSize.Width,
DesktopBitmapSize.Height, dc1, 0, 0, SRCCOPY)
      Graphic.ReleaseHdc(dc1)
      MemGraphic.ReleaseHdc(dc2)
      Graphic.Dispose()
      MemGraphic.Dispose()

Dim g As Graphics =
System.Drawing.Graphics.FromImage(MemImage)
```

```
Dim cur As System.Windows.Forms.Cursor =
System.Windows.Forms.Cursors.Arrow
cur.Draw(g, New Rectangle(System.Windows.Forms.Cursor.Position.X
- 10, System.Windows.Forms.Cursor.Position.Y - 10, cur.Size.Width,
cur.Size.Height))
Dim ms As MemoryStream = New MemoryStream
MemImage.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg)
Return ms.GetBuffer()
   End Function
End Class
```

وحتى نستطيع التحكم في حجم الصورة سنستخدم الـ Bitmap Class والـ Graphics Class إذ سنمرر لها الـHeight والـ Width والصورة التي نريد تغيير حجمها وكما يلي:

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 الـNamespaces التالية في البرنامج لتعامل مع الـMulticasting Socket :

C#:

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

VB.NET:

```
imports System.Net
```

```
imports System.Net.Socket
imports System.IO
```

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

C#:

```
 Bitmap bt = new Bitmap(CaptureScreen.GetDesktopBitmapBytes ());
picScreen.Image = ResizeBitmap(bt, 600, 400 );
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.GetDesktopBitmapBytes)
picScreen.Image = ResizeBitmap(bt, 600, 400)
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
```

3- Full/Half Duplex Multicast Text Conferencing System:

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

ثم قم بإضافة الـNamespaces التالية:

C#:

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

VB.NET:

```
imports System.Net
imports System.Net.Socket
imports System.Text
imports System.Threading
```

سوف نستخدم الـmethod التالية لإجراء عملية الإرسال حيث سترسل الرسالة عند الضغط على الـEnter بعد كتابة الرسالة في الـTextbox 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 الدالة 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 الـThreading ، قم بعمل Timer واستدعي فيه الـmethod السابقة باستخدام الـThreadStart Delegate 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 الـTimer 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
```

تعرفنا في هذا الفصل على كيفية عمل Multicast Conference Systems في بيئة الدوت نيت وطرق إنشاء الـVideo Conference System والـ Desktop Conference System والـ Text Conference System ، و ستتمكن بعد قراءة الفصل التالي من إنشاء الـ Voice Conference System .

سيتم الحديث في الفصل التالي عن الـ Voice Over IP Programming واستخدامه لعمل Voice Chat وVoice Conference System باستخدام الـ API's والـ Direct Play 9 بالإضافة إلى استخدام الـTAPI Telephony في بيئة الدوت نيت.

Chapter 10
Voice Over IP Programming

- The Concept & Requirements of Voice Communication Systems

- How to Create a Voice Chat Through Dot Net Using Unmanaged API's Functions

- Testing UDP Multicasting, TCP and Thinking in SCTP to Transfer Voice Through Networks

- How to Create a Voice Conference System Using Microsoft Direct Play 9

- Design an Advanced Video/Voice Conferencing System By Using API & TAPI Telephony to Transfer Video/Voice Through Networks & Voice Through Analog Telephone Line

10 - Voice Over IP Programming :

تتلخص الفكرة الأساسية من نقل الصوت عبر بروتوكول الإنترنت IP بتحويل الصوت إلى مجموعة من الـ Bits تجمع في Byte Array ثم كبسلته ليتم نقله كـ Datagram Packets أو كـ Data Stream عبر الشبكة ، وللاستقبال الصوت في الطرف الأخر يتم تجميع الـ Packets مرة أخرى في مصفوفة Byte Array ، وتتم عملية القراءة وفق مبدأ الـ – FIFO First In First Out أي القادم أولا يعرض أولا ...

تكمن المشكلة الأساسية بنقل الصوت في مدى توفر الشروط اللازمة حتى يتم إيصال وعرض الصوت بالشكل السليم و وفق الترتب الذي أرسل عليه ، وتعتبر محدوديات ومشاكل بروتوكولات الـ Transport Layer من أهم ما دعا Microsoft من العزوف عن دعم الـ Dot Net لعملية نقل الصوت وخاصة في بيئة النظام الحالي واكتفت بجعلها آداه خارجية باستخدام الـ API أو Managed باستخدام الـDirectX، ومن المعروف أن نظام التشغيل Windows XP يدعم الاتصال باستخدام بروتوكول TCP أو UDP فقط وهذا يعني انك إذا كنت تعمل تحت منصة نظام التشغيل Windows XP فإن أي عملية اتصال لن تكون إلا باستخدام واحد من هذه البروتوكولات.

أولا : The Requirements of Voice Communication Systems :

سوف نناقش في هذا الجزء متطلبات نقل الصوت عبر الشبكة ومشاكل نقل الصوت باستخدام بروتوكول الـ TCP و الـ UDP ...

– متطلبات نقل الصوت المثلى :

1 – يفضل أن يكون أسلوب النقل كـ Stream ، ويدعم هذا الأسلوب في كلا البروتوكولين الـTCP والـSCTP أما الـUDP فقط في حالة استخدامه مع الـRTP

2 – البروتوكول المستخدم لنقل الصوت يجب أن يدعم Delivered on Sequence

3 – تعتمد سرعة النقل على مدى حجم الضغط المستخدم Voice Compression والجودة المطلوبة ويفضل في هذه الحالة أن لا تقل سرعة النقل عن 31 KB\S بمعدل لا يقل عن 8.000 KHz كحد أدنى لجودة الصوت ومن الممكن أعتماد اساليب ضغط أخرى مثل G.711 لمعرفة كيفي أستخدامه بالدوت نيت أنظر الرابط التالي

http://www.codeproject.com/KB/audio-video/VoiceChatApplicationInCS.aspx

- السؤال الذي يطرح نفسه الآن ، هل وفر بروتوكول الـ TCP والـ UDP هذه الأمور ؟

أولا بروتوكول الـ TCP : يدعم بروتوكول الـ TCP كل هذه الأمور وبكفاءة عالية لكن المشكلة في هذا البروتوكول هو أنه ليس Real Time كذلك عدم إمكانية استخدامه لعمل Multicast Conference System إذ انه من المعروف أن الـMulticasting والـ Broadcasting من الأمور الخاصة ببروتوكول الـ UDP ولا يدعم الـ TCP أي من هذه الأمور وهو ما بينته في الفصل السابق ويمكن أن يكون هذا البروتوكول Real Time Protocol في حالة قمنا بإلغاء الـAcknowledgment في الـTCP Header ، إذ يعتر الـTCP بروتوكول موجه Oriented Protocol لذلك لا يمكن الاعتماد عليه في حالة حاجتنا لعمل Multicast Conference System أو في حالة البث الإذاعي Broadcasting .

ثانيا بروتوكول الـ UDP : لا يعتبر هذا البروتوكول حل جيد لعملية نقل الصوت بالكفاءة العالية (فقط في حالة ما أردنا نقل صوت عالي الدقة) إذ أنه لا يدعم عملية Delivered on Sequence وهو ما سبب من استحالة عمل Fragmentation للـ Packets المرسل ومن المعروف أن حجم Ethernet Encapsulation لا يزيد عن 1500 Bytes للـ Packet

الواحد وهو الحجم الأقصى للـ Datagram Encapsulation الخاص بالـ Ethernet لذلك في حالة قمنا بعمل Fragmentation لصوت فإننا لن نضمن وصول الصوت وفق الترتيب المرسل وهو ما يسبب مشكلة كبيرة في عملية إعادة ترتيب الـ Fragments المرسل ومن هذه النقطة قدمت الكثير من الشركات والمنظمات العالمية حلول خاصة لعملية نقل الصوت عبر الـ UDP منها شركة أدوبي بتقديمها بروتوكول النقل Real Time Messaging Protocol – RTMP http://www.adobe.com/devnet/rtmp ومنظمة IETF Internet Engineering Task Force بتقديمها بروتوكول النقل RTP – Real Time Transport Protocol ، حيث أضاف هذا المعيار تحسينات على بروتوكول الـ UDP لعملية نقل الصوت في الزمن الحقيقي إذ يستخدم أسلوب الـ Stream المستخدم في TCP.

ثانيا :The Concept Of Voice Communication :

تمر عملية التقاط الصوت بمجموعة من المراحل تبدأ بالتقاط الصوت من المايكروفون وتمثيل الذبذبات الصوتية ثم تحويلها إلى مجموعة من الـ Bits وذلك بعمل Sampling لذبذبات الصوتية الملتقطة وبعد هذه العملية يمكننا نقل الصوت عبر الشبكة وتمر عملية نقل الصوت عبر الشبكة بمجموعة من المراحل وهي :

1- في الـ Application Layer ، طريقة التقاط الصوت وتحويله إلى Bits وهو ما ذكرته سابقا ، و استخدام تقنيات لضغط الصوت Audio Compression Teachings وحتى يمكن إرساله عبر الإمكانيات المحدودة لشبكة الاتصال.

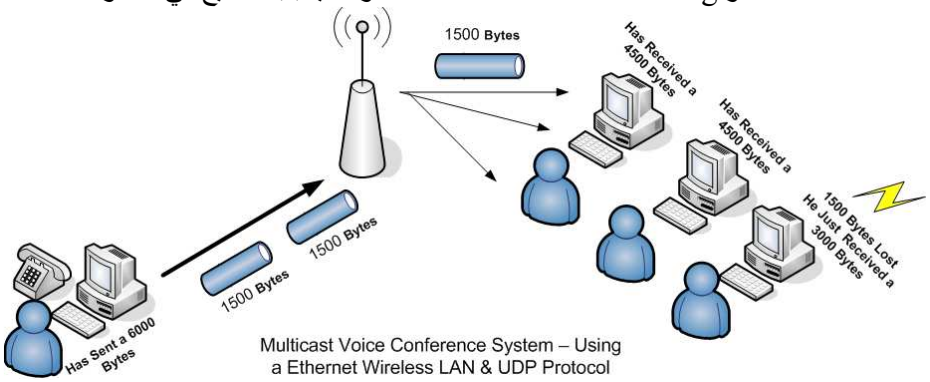
2- في الـ Transport Layer ، وهو من أهم الأمور التي يجب أخذها بعين الاعتبار إذ أن المفاضلة بين اختيار بروتوكول الـ UDP أو الـTCP تعتمد على مدى الحاجة التي نريدها ودقة الصوت من جهة أخرى إذ أن أفضل طريقة لنقل الصوت هي استخدام تقنيات الـStream لكن من المعروف أن بروتوكول الـUDP لا يدعم عملية النقل كـ Stream كونه لا يدعم التوصيل وفق الترتيب Delivered on Sequence إذ أننا في هذه الحالة لن نتمكن من عمل الـ Fragmentation للـ Buffer حيث لن نضمن وصول الـ Fragments وفق الترتيب الذي أرسل عليه وسوف نضطر إلى التقيد بمحدوديات الـ Ethernet للـ Packet وهي 1500 BYTES للـ Packet الواحد ولحل هذه المشكلة سوف نلجأ إلى تبني بعض التقنيات الجديدة والتي تعتمد على بروتوكول الـ UDP وحتى يتم نقل الصوت كـ Stream ومنها أسلوب النقل H.323 والذي ذكرته سابقا ، لكن لم تدعم الدوت نيت أي من هذه التقنيات ، لذلك عندما نريد نقل صوت من جهاز إلى آخرك Stream لابد لنا من استخدام بروتوكول الـ TCP لكن كما ذكرنا سابقا فإنه لا يدعم الـ IP Multicasting و الـ Broadcasting

3- في الـ Network Layer يتم عنونة الـ Packets وإذا ما قررنا اعتماد الـ UDP فإننا سوف نتمكن من عمل البث الإذاعي Broadcasting ومجموعات البث Multicasting

4- في الـ Data Link Layer سيتم تحديد طبيعة ولإرسال سواء باستخدام الـ Ethernet أو غيره وفي هذه الحالة سيتم الاعتماد على الـ Ethernet لكن مشكلته كما ذكرتها سابقا بمحدودية حجم الـ Frame إذ لا تتجاوز الـ 1500 BYTES

5- في الـ Physical Layer طبعا المشاكل التي قد تحدث إثناء عملية النقل كثيرة جدا وقد يحدث تأخير Delay لسبب أو لآخر أو قد تضيع بعض الـ Bits إثناء الإرسال لذلك لابد من وجود بروتوكولات تدعم التصحيح لكل هذه المشاكل والتي قد تحدث إثناء عملية الإرسال.
يستقبل الطرف المقابل الـ Bits من طبقة الـ Physical Layer وتمر عبر الـ Data Link Layer ومن ثم الـ Network Layer وفي إثناء هذه المرحلة فإن مستقبل الـ Packets قد يكون هو الشخص المعني في حالة كان أسلوب البث Unicast أو قد يكون جزء من مجموعة الاستقبال Multicast أو قد يكون من ضمن الشبكة التي تم الإرسال لها كـ Broadcast لذلك

في حالة كونه جزء من مجموعة فإن جهة الإرسال غير معنية بالجهة التي سوف تستقبل الـPackets وفي هذه الحالة فإنه غير معني سواء استقبلت جزء من الـ Packets أو كلها حيث لن يتم إرسال أي Acknowledgment إلى المرسل لذلك قد تحدث الكثير من المشاكل إثناء هذه المرحلة منها ضياع جزء من الـ Packets المرسل والذي سوف يسبب وصول الصوت بشكل متقطع ، وطبعا سوف يكون الاعتماد في هذه الحالة على بروتوكولات الطبقة الأعلى وهي هنا Transport Layer فإذا كان المرسل والمستقبل يستخدم الـ TCP فإن كل هذه المشاكل سوف تحل لكن المشكلة تكمن في كونه يستخدم الـ UDP حيث لا يوجد حل إلا بإتباع معيار مساند يضمن وصول كافة الـ Packets بترتيب الذي أرسل عليه وبدون ضياع أجزاء من الـ Packets المرسل.
الشكل التالي يوضح عملية ضياع بعض الـ Packets إثناء الإرسال باستخدام شبكة Ethernet Wireless LAN و UDP IP Multicasting مما سوف يسبب تقطيع في الصوت:



ثالثا: How to Create a Voice Chat Through Dot Net Using Unmanaged API's Functions :

كما بينا سابقا فإن الدوت نيت لم تدعم أي من عمليات التقاط وعرض الصوت ، لكن لإجراء هذه العمليات لابد من استخدام الـAPI's والمغلفة ضمن ملفات الـ DLL والتي تأتي مع نظام التشغيل ومنها ملف winmm.dll الشهير ، والخاص بالتعامل مع وسائل الـ Multimedia في نظام التشغيل ، حيث يدعم هذا الملف مجموعة من الـ Methods لالتقاط الصوت عبر المايكروفون وتخزينه في Byte Array Buffer ومن ثم عرضه مرة أخرى وهذه الـMethod هي :

waveInGetNumDevs والتي تستخدم لتحديد عدد أجهزة الإدخال والمربوطة مع الـSound Card ولا تأخذ أي بارومترات.
waveInAddBuffer وتستخدم لتخزين الـBits الواردة من جهاز الإدخال في Byte Array Buffer وتأخذ هذه الـMethod ثلاثة بارومترات وهي:

```
waveInAddBuffer(IntPtr hwi, ref WaveHdr pwh, int cbwh)
```

حيث يمرر للأول جهاز الإدخال والذي تم اختياره و يحدد في الثاني Reference لموقع تخزين الـ Buffer وفي الثالث يحدد حجم الـ Buffer المستلم

الدالة **waveInClose** و **waveInOpen** لفتح وإغلاق الاتصال مع جهاز الإدخال.
الدالة **waveInPrepareHeader** لتجهيز وحجز الـ Buffer وتأخذ نفس البارومترات الموجودة في waveInAddBuffer.
الدالة **waveInUnprepareHeader** ويتم استدعائها بعد تعبئة الـ Buffer حتى يتم إرسال الـ Buffer ومن ثم تفريغه للاستعداد لتعبئته مرة أخرى.
الدالة **waveInReset** لإرجاع مؤشر الـPointer الخاص بالـ Buffer إلى صفر
الدالة **waveInStart** و الدالة **waveInStop** بدأ وإغلاق عملية الإدخال من المايكروفون.

ولاستخدام هذه الدوال في الدوت نيت نقوم بتعريفها أولا باستخدام DllImport وكما يلي:

C#:

```
[DllImport(winmm.dll)]
public static extern int waveInGetNumDevs();
[DllImport(winmm.dll)]
public static extern int waveInAddBuffer(IntPtr hwi, ref WaveHdr pwh,
int cbwh);
[DllImport(winmm.dll)]
public static extern int waveInClose(IntPtr hwi);
[DllImport(winmm.dll)]
public static extern int waveInOpen(out IntPtr phwi, int uDeviceID,
WaveFormat lpFormat, WaveDelegate dwCallback, int dwInstance, int
dwFlags);
[DllImport(winmm.dll)]
public static extern int waveInPrepareHeader(IntPtr hWaveIn, ref
WaveHdr lpWaveInHdr, int uSize);
[DllImport(winmm.dll)]
public static extern int waveInUnprepareHeader(IntPtr hWaveIn, ref
WaveHdr lpWaveInHdr, int uSize);
[DllImport(winmm.dll)]
public static extern int waveInReset(IntPtr hwi);
[DllImport(winmm.dll)]
public static extern int waveInStart(IntPtr hwi);
[DllImport(winmm.dll)]
public static extern int waveInStop(IntPtr hwi);
```

VB.NET

```
<DllImport(winmm.dll)> _
Public Shared Function waveInGetNumDevs() As Integer
End Function
<DllImport(winmm.dll)> _
Public Shared Function waveInAddBuffer(ByVal hwi As IntPtr, ByRef
pwh As WaveHdr, ByVal cbwh As Integer) As Integer
End Function
<DllImport(winmm.dll)> _
Public Shared Function waveInClose(ByVal hwi As IntPtr) As Integer
End Function
<DllImport(winmm.dll)> _
Public Shared Function
waveInOpen(<System.Runtime.InteropServices.Out()> ByRef phwi As
IntPtr, ByVal uDeviceID As Integer, ByVal lpFormat As WaveFormat,
ByVal dwCallback As WaveDelegate, ByVal dwInstance As Integer,
ByVal dwFlags As Integer) As Integer
End Function
```

```
<DllImport(winmm.dll)> _
Public Shared Function waveInPrepareHeader(ByVal hWaveIn As
IntPtr, ByRef lpWaveInHdr As WaveHdr, ByVal uSize As Integer) As
Integer
End Function
<DllImport(winmm.dll)> _
Public Shared Function waveInUnprepareHeader(ByVal hWaveIn As
IntPtr, ByRef lpWaveInHdr As WaveHdr, ByVal uSize As Integer) As
Integer
End Function
<DllImport(winmm.dll)> _
Public Shared Function waveInReset(ByVal hwi As IntPtr) As Integer
End Function
<DllImport(winmm.dll)> _
Public Shared Function waveInStart(ByVal hwi As IntPtr) As Integer
End Function
<DllImport(winmm.dll)> _
Public Shared Function waveInStop(ByVal hwi As IntPtr) As Integer
End Function
```

وكما سوف نستخدم مجموعة الـMethods التالية لتحويل الـ Byte Array Buffer إلى صوت مرة أخرى وعرضه على جهاز الإخراج :

C#:

```
[DllImport(winmm.dll)]
public static extern int waveOutGetNumDevs();
[DllImport(winmm.dll)]
public static extern int waveOutPrepareHeader(IntPtr hWaveOut, ref
WaveHdr lpWaveOutHdr, int uSize);
[DllImport(winmm.dll)]
public static extern int waveOutUnprepareHeader(IntPtr hWaveOut, ref
WaveHdr lpWaveOutHdr, int uSize);
[DllImport(winmm.dll)]
public static extern int waveOutWrite(IntPtr hWaveOut, ref WaveHdr
lpWaveOutHdr, int uSize);
[DllImport(winmm.dll)]
public static extern int waveOutOpen(out IntPtr hWaveOut, int
uDeviceID, WaveFormat lpFormat, WaveDelegate dwCallback, int
dwInstance, int dwFlags);
[DllImport(winmm.dll)]
public static extern int waveOutReset(IntPtr hWaveOut);
[DllImport(mmdll)]
public static extern int waveOutClose(IntPtr hWaveOut);
[DllImport(mmdll)]
public static extern int waveOutPause(IntPtr hWaveOut);
[DllImport(mmdll)]
public static extern int waveOutRestart(IntPtr hWaveOut);
```

```
[DllImport(mmdll)]
public static extern int waveOutGetPosition(IntPtr hWaveOut, out int
lpInfo, int uSize);
[DllImport(mmdll)]
public static extern int waveOutSetVolume(IntPtr hWaveOut, int
dwVolume);
[DllImport(mmdll)]
public static extern int waveOutGetVolume(IntPtr hWaveOut, out int
dwVolume);
```

VB.NET

```
<DllImport(winmm.dll)> _
Public Shared Function waveOutGetNumDevs() As Integer
End Function
<DllImport(winmm.dll)> _
Public Shared Function waveOutPrepareHeader(ByVal hWaveOut As
IntPtr, ByRef lpWaveOutHdr As WaveHdr, ByVal uSize As Integer) As
Integer
End Function
<DllImport(winmm.dll)> _
Public Shared Function waveOutUnprepareHeader(ByVal hWaveOut
As IntPtr, ByRef lpWaveOutHdr As WaveHdr, ByVal uSize As Integer)
As Integer
End Function
<DllImport(winmm.dll)> _
Public Shared Function waveOutWrite(ByVal hWaveOut As IntPtr,
ByRef lpWaveOutHdr As WaveHdr, ByVal uSize As Integer) As
Integer
End Function
<DllImport(winmm.dll)> _
Public Shared Function
waveOutOpen(<System.Runtime.InteropServices.Out()> ByRef
hWaveOut As IntPtr, ByVal uDeviceID As Integer, ByVal lpFormat As
WaveFormat, ByVal dwCallback As WaveDelegate, ByVal dwInstance
As Integer, ByVal dwFlags As Integer) As Integer
End Function
<DllImport(winmm.dll)> _
Public Shared Function waveOutReset(ByVal hWaveOut As IntPtr) As
Integer
End Function
<DllImport(mmdll)> _
Public Shared Function waveOutClose(ByVal hWaveOut As IntPtr) As
Integer
End Function
<DllImport(mmdll)> _
Public Shared Function waveOutPause(ByVal hWaveOut As IntPtr) As
Integer
```

```
End Function
<DllImport(mmdll)> _
Public Shared Function waveOutRestart(ByVal hWaveOut As IntPtr)
As Integer
End Function
<DllImport(mmdll)> _
Public Shared Function waveOutGetPosition(ByVal hWaveOut As
IntPtr, <System.Runtime.InteropServices.Out()> ByRef lpInfo As
Integer, ByVal uSize As Integer) As Integer
End Function
<DllImport(mmdll)> _
Public Shared Function waveOutSetVolume(ByVal hWaveOut As
IntPtr, ByVal dwVolume As Integer) As Integer
End Function
<DllImport(mmdll)> _
Public Shared Function waveOutGetVolume(ByVal hWaveOut As
IntPtr, <System.Runtime.InteropServices.Out()> ByRef dwVolume As
Integer) As Integer
End Function
```

وحتى نتمكن من عرض محتويات الـ Buffer نستخدم التعريف التالي:

C#:

```
using System;
using System.Runtime.InteropServices;
using System.Resources;
using System.IO;

 public class Winmm
        {
                public const UInt32 SND_ASYNC = 1;
                public const UInt32 SND_MEMORY = 4;

                [DllImport("Winmm.dll")]
public static extern bool PlaySound(byte[] data, IntPtr hMod,   UInt32
dwFlags);

public static void PlayWavResource(byte[] buffer)
{
        PlaySound(buffer, IntPtr.Zero,  SND_ASYNC |
SND_MEMORY);
}
}
```

VB.NET

```
Imports System
Imports System.Runtime.InteropServices
```

```
Imports System.Resources
Imports System.IO

Public Class Winmm
   Public Const SND_ASYNC As UInt32 =
System.Convert.ToUInt32(1)
   Public Const SND_MEMORY As UInt32 =
System.Convert.ToUInt32(4)
   <DllImport("Winmm.dll")> _
   Public Shared Function PlaySound(ByVal data As Byte(), ByVal
hMod As IntPtr, ByVal dwFlags As UInt32) As Boolean
   End Function
   Public Sub New()
   End Sub
   Public Shared Sub PlayWavResource(ByVal buffer As Byte())
      PlaySound(buffer, IntPtr.Zero, SND_ASYNC Or
SND_MEMORY)
   End Sub
End Class
```

حيث نمرر للـ PlaySound Method الـ Byte Buffer والمستلم من Method الاستقبال الخاصة بالـSocket وكما يلي:

C#:

```
void Voice_Receiver()
        {
UdpClient sock = new UdpClient(5020);
sock.JoinMulticastGroup(IPAddress.Parse(multicast_IP.Text));
IPEndPoint iep = new IPEndPoint(IPAddress.Any, 0);
byte[] voice_Come = sock.Receive(ref iep);
Winmm.PlayWavResource(voice_Come);
sock.Close();
        }
```

VB.NET

```
Private Sub Voice_Receiver()
   Dim sock As UdpClient = New UdpClient(5020)
   sock.JoinMulticastGroup(IPAddress.Parse(multicast_IP.Text))
   Dim iep As IPEndPoint = New IPEndPoint(IPAddress.Any, 0)
   Dim voice_Come As Byte() = sock.Receive(iep)
   Winmm.PlayWavResource(voice_Come)
   sock.Close()
End Sub
```

البدء بإنشاء برنامج المحادثة الصوتية Voice Chat System :

سوف نجزئ عملية التقاط الصوت وتخزينه في الـ Buffer ثم عرضه مرة أخرى في مجموعة من الـClasses وهو تقسيم تم استخدامه في الكثير من البرمجيات الخاصة بـ Microsoft ومنها برنامج Windows Sound Recorder وسوف نجمع هذه الـ Classes في ملف واحد نسميه Voice Library وسوف تجد محتويات هذه الـ Classes ، في القرص المدمج والمرفق مع الكتاب ، وهذه الـ Classes هي:

WaveIn Class وسوف نستخدمه لوضع كافة الـ Methods الخاصة بالتقاط الصوت وتخزينه في Byte Array

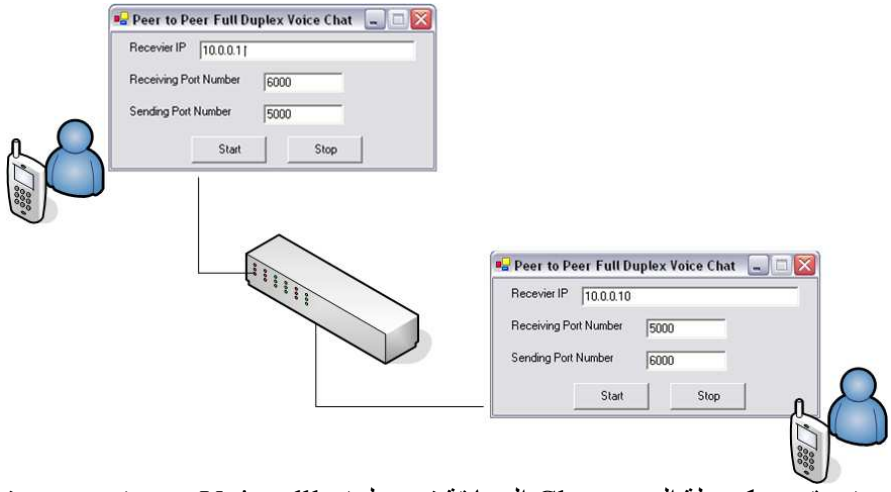
WaveOut Clasas وسوف نستخدمه لعرض الصوت الأتي من الـ Buffer ثم عرضه

WaveStream Class والذي سوف نستخدمه لتحويل الصوت إلى Stream حيث يسهل إرساله عبر الشبكة ويشبه عمله عمل MemoryStream المستخدمة في الدوت نيت

الدالة FifoStream لتنظيم الـ Stream بحيث يتم عرض الداخل أولا خارج أولا

الدالة WaveNative ويتم فيها وضع كافة التعريفات للـ Methods الخاصة بالملف winmm.dl والتي شرحناها سابقاً.

سوف نستخدم في هذا المثال بروتوكول الـ UDP لعملية النقل ومعتمدا على أسلوب البث Full Diplex Unicast Voice Chat System وسيكون الشكل العام لبرنامج الاتصال كما يلي :



وسوف نقوم بكبسلة الـ Classes السابقة في ملف Voice.dll وسوف نضعه في الـReferences الخاصة بالبرنامج وحتى نستطيع استخدام هذا الملف في جميع البرامج التي سوف تستخدم عملية الاتصال الصوتي بعد هذه العملية يمكن استخدام الملف باستدعائه كأي Managed Library Class في الدوت نيت وكما يلي:

C#:

```
using System.Net;
using System.Net.Socket;
using System.Threading;
using Voice;
```

VB.NET

```
Imports System.Net
Imports System.Net.Socket
Imports System.Threading
Imports Voice
```

ثم نقوم بتعريف الـSocket والـThread والذي سوف نستخدمه في البرنامج ويفضل وضع هذه التعريفات في بداية البرنامج أي بعد تعريف الـ Class الرئيسي والهدف من هذه العملية هي القدرة على إغلاق الـ Socket والـThread عند إطفاء البرنامج وحتى لا تبقى في الذاكرة عند إغلاق برنامج الاتصال ، ويتم ذلك كما يلي:

C#:
```
public class Form1 : System.Windows.Forms.Form
{
private Socket socket;
private Thread thread;
```

VB.NET
```
Public Class Form1 : Inherits System.Windows.Forms.Form
   Private socket As Socket
   Private thread As Thread
```

وسوف نعرف Object من الـClasses السابقة ونعرف الـBuffer الذي سيتم تسجيل الصوت المراد إرساله والـBuffer الذي سيتم عرض الصوت المستلم من الـSocket

C#:
```
private WaveOutPlayer m_Player;
private WaveInRecorder m_Recorder;
private FifoStream m_Fifo = new FifoStream();
private byte[] m_PlayBuffer;
private byte[] m_RecBuffer;
```

VB.NET
```
Private m_Player As WaveOutPlayer
Private m_Recorder As WaveInRecorder
Private m_Fifo As FifoStream = New FifoStream
Private m_PlayBuffer As Byte()
Private m_RecBuffer As Byte()
```

في الـ Constructor الخاص بالبرنامج أو في الـ Form Load Event قم بكتابة التعريف الخاص بالـ Socket والـ Thread :

C#:
```
public Form1()
{
InitializeComponent();
socket = new Socket(AddressFamily.InterNetwork, SocketType.Dgram,
ProtocolType.Udp);
thread = new Thread(new ThreadStart(Voice_In));
}
```

VB.NET
```
socket = New Socket(AddressFamily.InterNetwork,
SocketType.Dgram, ProtocolType.Udp)
thread = New Thread(AddressOf Voice_In)
```

سوف نضع في الـVoice_In Method الكود الخاص بعملية استقبال الصوت من الـSocket وكما يلي:

C#:
```
private void Voice_In()
{
        byte[] br;
        socket.Bind(new IPEndPoint(IPAddress.Any, 5020));

while (true)
        {
        br = new byte[16384];
        socket.Receive(br);
        m_Fifo.Write(br, 0, br.Length);
        }
}
```

VB.NET
```
Private Sub Voice_In()
   Dim br As Byte()
   socket.Bind(New IPEndPoint(IPAddress.Any, 5020))

   Do While True
      br = New Byte(16383)
      socket.Receive(br)
      m_Fifo.Write(br, 0, br.Length)
   Loop
End Sub
```

حيث يتم استقبال الصوت من الشبكة باستخدام الـReceive Method ثم نمرر الصوت المستقبل إلى الـm_Fifo.Write Method وحتى يتم تنفيذه وتحويله إلى صوت مرة أخرى.

أما الـMethod التي تقوم بتسجيل الصوت وإرساله إلى الجهاز الأخر فهي:

C#:
```
private void Voice_Out(IntPtr data, int size)
{
//for Recorder
if (m_RecBuffer == null || m_RecBuffer.Length < size)
m_RecBuffer = new byte[size];
System.Runtime.InteropServices.Marshal.Copy(data, m_RecBuffer, 0,
size);
```

```
//Microphone ==> data ==> m_RecBuffer ==> m_Fifo
socket.SendTo(m_RecBuffer, new
IPEndPoint(IPAddress.Parse(Peer_IP.Text),5030));
}
```

VB.NET

```
Private Sub Voice_Out(ByVal data As IntPtr, ByVal size As Integer)
   'for Recorder
   If m_RecBuffer Is Nothing OrElse m_RecBuffer.Length < size Then
      m_RecBuffer = New Byte(size - 1)
   End If
   System.Runtime.InteropServices.Marshal.Copy(Data, m_RecBuffer,
0, size)
   'Microphone ==> data ==> m_RecBuffer ==> m_Fifo
   socket.SendTo(m_RecBuffer, New
IPEndPoint(IPAddress.Parse(Peer_IP.Text), 5030))
End Sub
```

لاحظ أنه في حالة إذا ما أردنا عمل برنامج Full Duplex بحيث يرسل ويستقبل في نفس الوقت فإننا بحاجة إلى تعريف Tow Ports واحد للإرسال وأخرى للاستقبال وفي الطرف الأخر تكون Port الإرسال لديك هي Port الاستقبال لديه والعكس صحيح .

في زر البدء يتم تنفيذ الدالة التالية حيث استخدمنا الـ WaveFormat لتحديد جودة الصوت بـ2 Channels & 44100 KHz 16 Bits , وأسندت إلى الـWaveOutPlayer Class والذي سيستخدم لتعبئة الـBuffer وفي حال تعبئته يتم إرساله إلى الـ Voice_Out Method لإرساله عبر الشبكة إلى الجهاز الآخر :

C#:

```
private void Start()
{
        Stop();
try
{
WaveFormat fmt = new WaveFormat(44100, 16, 2);
m_Player = new WaveOutPlayer(-1, fmt, 16384, 3, new
BufferFillEventHandler(Filler));
m_Recorder = new WaveInRecorder(-1, fmt, 16384, 3, new
BufferDoneEventHandler(Voice_Out));
}
catch
{ Stop();throw;}
}
```

VB.NET

```
Private Sub Start()
        Stop()
   Try
```

```
      Dim fmt As WaveFormat = New WaveFormat(44100, 16, 2)
      m_Player = New WaveOutPlayer(-1, fmt, 16384, 3, New
BufferFillEventHandler(AddressOf Filler))
      m_Recorder = New WaveInRecorder(-1, fmt, 16384, 3, New
BufferDoneEventHandler(AddressOf Voice_Out))
   Catch
Stop()
      Throw
   End Try
End Sub
```

أما في زر الإيقاف فيتم تنفيذ الدالة التالية حيث سيتم تفريغ محتويات الـ Buffer وإيقاف تسجيل الصوت:

C#:

```
private void Stop()
{
if (m_Player != null)
        try
                {
                m_Player.Dispose();
                }
        finally
{m_Player = null;}
if (m_Recorder != null)
        try
        {m_Recorder.Dispose();}
        finally
        {m_Recorder = null;}
        m_Fifo.Flush(); // clear all pending data
}
```

VB.NET

```
Private Sub [Stop]()
   If Not m_Player Is Nothing Then
      Try
         m_Player.Dispose()
      Finally
         m_Player = Nothing
      End Try
   End If
   If Not m_Recorder Is Nothing Then
      Try
         m_Recorder.Dispose()
      Finally
         m_Recorder = Nothing
      End Try
```

```
    End If
    m_Fifo.Flush() ' clear all pending data
End Sub
```

وتستخدم الـ Filler Method لتعبئة الصوت في الـ Buffer ولإتمام عملية عرضه على السماعات(Output Device):

C#:

```
private void Filler(IntPtr data, int size)
{
if (m_PlayBuffer == null || m_PlayBuffer.Length < size)
m_PlayBuffer = new byte[size];
if (m_Fifo.Length >= size)
m_Fifo.Read(m_PlayBuffer, 0, size);
else
for (int i = 0; i < m_PlayBuffer.Length; i++)
m_PlayBuffer[i] = 0;
System.Runtime.InteropServices.Marshal.Copy(m_PlayBuffer, 0, data,
size);
// m_Fifo ==> m_PlayBuffer==> data ==> Speakers
}
```

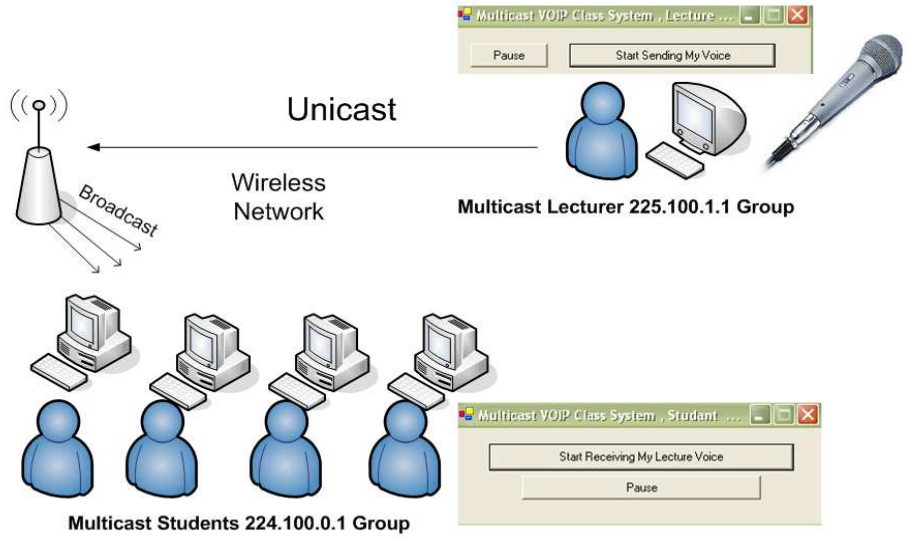
VB.NET

```
Private Sub Filler(ByVal data As IntPtr, ByVal size As Integer)
    If m_PlayBuffer Is Nothing OrElse m_PlayBuffer.Length < size Then
        m_PlayBuffer = New Byte(size - 1)
    End If
    If m_Fifo.Length >= size Then
        m_Fifo.Read(m_PlayBuffer, 0, size)
    Else
        Dim i As Integer = 0
        Do While i < m_PlayBuffer.Length
            m_PlayBuffer(i) = 0
            i += 1
        Loop
    End If
    System.Runtime.InteropServices.Marshal.Copy(m_PlayBuffer, 0,
data, size)
    ' m_Fifo ==> m_PlayBuffer==> data ==> Speakers
End Sub
```

رابعا: Testing TCP,UDP and Thinking in SCTP to Transfer Voice Through Networks :

لإنشاء برنامج Multicast Half Duplex Voice Chat System نقوم بإضافة التعريفات التالية والخاصة بالـMulticasting والتي شرحناها في الفصل السابق ، وسوف نعتمد على وجود برنامجين واحد للمحاضر يتم من خلاله تسجيل الصوت وآخر لطالب حيث يستمع فيه لمحاضرة المعلم وكما في الشكل التالي:

By FADI Abdel-Qader



Half Duplex Multicast Voice Conferencing System– For Class Room That uses a Peer-to-Peer Wireless Network

في برنامج الإرسال (برنامج المعلم) لا يختلف الكود بشيء فقط عند الإرسال يتم ذلك باستخدام الـ IP Multicasting وكما يلي:

C#:

```
private void Voice_Out(IntPtr data, int size)
{
//for Recorder
if (m_RecBuffer == null || m_RecBuffer.Length < size)
m_RecBuffer = new byte[size];
System.Runtime.InteropServices.Marshal.Copy(data, m_RecBuffer, 0,
size);
//Microphone ==> data ==> m_RecBuffer ==> m_Fifo
socket.SendTo(m_RecBuffer, new
IPEndPoint(IPAddress.Parse("224.0.1.7"), 5020));
}
```

VB.NET

```
Private Sub Voice_Out(ByVal data As IntPtr, ByVal size As Integer)
   'for Recorder
```

```
   If m_RecBuffer Is Nothing OrElse m_RecBuffer.Length < size Then
      m_RecBuffer = New Byte(size - 1)
   End If
   System.Runtime.InteropServices.Marshal.Copy(data, m_RecBuffer,
0, size)
   'Microphone ==> data ==> m_RecBuffer ==> m_Fifo
   socket.SendTo(m_RecBuffer, New
IPEndPoint(IPAddress.Parse("224.0.1.7"), 5020))
End Sub
```

في الطرف المستقبل نقوم بالانضمام إلى الـIP Multicast Group ومن ثم الاستقبال من خلاله وكما يلي:

C#:

```
private void Voice_In()
{
UdpClient sock = new UdpClient(5000);
sock.JoinMulticastGroup(IPAddress.Parse("224.0.1.7"));
IPEndPoint iep = new IPEndPoint(IPAddress.Any,0);
while (true)
        {
        m_Fifo.Write(sock.Receive(ref iep), 0,sock.Receive(ref
iep).Length);
        }
}
```

VB.NET

```
Private Sub Voice_In()
   Dim sock As UdpClient = New UdpClient(5000)
   sock.JoinMulticastGroup(IPAddress.Parse("224.0.1.7"))
   Dim iep As IPEndPoint = New IPEndPoint(IPAddress.Any, 0)
   Do While True
      m_Fifo.Write(sock.Receive(iep), 0, sock.Receive(iep).Length)
   Loop
End Sub
```

الآن نفذ البرنامج ... لاحظ أن الصوت قد يتقطع أحيانا وبتأكيد السبب واضح وهو أنه في حالة استخدام بروتوكول الـ UDP والـMulticasting فإن عملية الإرسال ستكون عشوائية وهذا قد يسبب ضياع واحد أو أكثر من الـPackets المرسلة عبر الشبكة كل فترة وبما أن بروتوكول الـ UDP لا يدعم أي من عمليات التحقق من الوصول وعمليات التوصيل على الترتيب فإن حدوث واحد أو أكثر من هذه المشاكل أمر محتمل .

دعنا الآن نختبر عملية الإرسال باستخدام بروتوكول TCP ، لاحظ أن هذا البروتوكول هو بروتوكول موجه Oriented Protocol كما يدعم جميع عمليات التحقق من الوصول بالإضافة إلى كونه يدعم الاتصال بين الطرفين باستخدام أسلوب الـ Stream وهو ما يميز هذا البروتوكول عن غيره إذ أننا في حالة استخدامه لن نضطر إلى الأخذ بحجم البيانات المرسلة حيث يتم عمل Fragmentation لها بكل سهولة بالإضافة إلى سهولة تجميعها مرة أخرى وبدون الخوف من مشاكل الـ Delivered Out on Sequence .

كل ما علينا تغييره هو تعريف الـ Socket السابق وجعله كما يلي :

C#:
```
socket = new Socket(AddressFamily.InterNetwork, SocketType.Stream
, ProtocolType.Tcp);
```

VB.NET
```
socket = New Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp)
```

أما في عملية الإرسال فيمكننا استخدام الـNetwork Stream Class وكما يلي:

C#:
```
private void Voice_Out(IntPtr data, int size)
{
//for Recorder
if (m_RecBuffer == null || m_RecBuffer.Length < size)
m_RecBuffer = new byte[size];
System.Runtime.InteropServices.Marshal.Copy(data, m_RecBuffer, 0,
size);
//Microphone ==> data ==> m_RecBuffer ==> m_Fifo
sock.Connect(new IPEndPoint(IPAddress.Parse("10.0.0.10"),5020));
NetworkStream ns = new NetworkStream (socket);
ns.Write (m_RecBuffer,0,m_RecBuffer.Length);
}
```

VB.NET
```
Private Sub Voice_Out(ByVal data As IntPtr, ByVal size As Integer)
   'for Recorder
   If m_RecBuffer Is Nothing OrElse m_RecBuffer.Length < size Then
      m_RecBuffer = New Byte(size - 1)
   End If
   System.Runtime.InteropServices.Marshal.Copy(data, m_RecBuffer,
0, size)
   'Microphone ==> data ==> m_RecBuffer ==> m_Fifo
   sock.Connect(New IPEndPoint(IPAddress.Parse("10.0.0.10"), 5020))
   Dim ns As NetworkStream = New NetworkStream(socket)
   ns.Write(m_RecBuffer, 0, m_RecBuffer.Length)
End Sub
```

في الطرف المستقبل نقوم باستخدام الـ Network Stream مرة أخرى لكن للاستقبال:

C#:
```
private void Voice_In()
{
Socket sock = new Socket(AddressFamily.InterNetwork,
SocketType.Stream , ProtocolType.Tcp);
```

```
        sock.Bind(new IPEndPoint(IPAddress.Parse("10.0.0.10"),5020));
        NetworkStream ns = new NetworkStream (sock);
        while (true){
        byte[] buffer = new byte [16384];
        ns.Read (buffer,0,16384);
        m_Fifo.Write(buffer, 0, buffer.Length); }
}
```

VB.NET

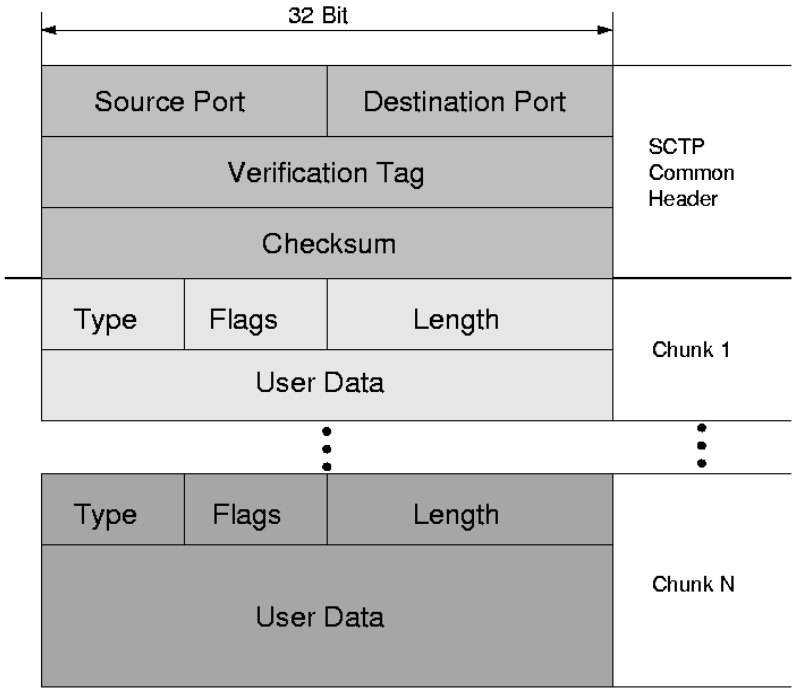
```
Private Sub Voice_In()
   Dim sock As Socket = New Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp)
   sock.Bind(New IPEndPoint(IPAddress.Parse("10.0.0.10"), 5020))
   Dim ns As NetworkStream = New NetworkStream(sock)
   Do While True
      Dim buffer As Byte() = New Byte(16383)
      ns.Read(buffer, 0, 16384)
      m_Fifo.Write(buffer, 0, buffer.Length)
   Loop
End Sub
```

لاحظ أن دقة الصوت أصبحت ممتازة كما أنه لا يوجد أي تقطيع في الصوت لكن المشكلة تكمن في أننا لن نستطيع الاستفادة من هذه الإمكانيات الرائعة في الـMulticasting أو الـBroadcasting ...

والحل الوحيد هو إما استخدام المعايير السابقة مع بروتوكول الـUDP وهو ما تم استخدامه الآن في معظم البرمجيات يمكنك الرجوع إلى القرص المدمج المثال OpenH323 لمعرفة كيفية استخدام الـUDP والـRTP لنقل الصوت وفق المعيار H323. أو استخدام بروتوكول الـTCP وهو جيد في حالة كان عامل التزامن غير مهم بين المرسل والمستقبل وكمثال على ذلك برنامج Windows Media Server والذي يستخدم الـTCP مع الـHTTP لنقل الصوت والصورة أو استخدام أسلوب النقل عبر الـ Direct Play ضمن الـ DirectX.

السؤال الذي يطرح نفسه الآن ما هو الجديد بهذا البروتوكول هل حل المشكلة؟؟

الجواب بكل بساطة نعم قد حل المشكلة حيث أن معمارية هذا البروتوكول الذي استفاد من ميزات الـTCP والدعم المقدم من قبل الـUDP لعمليات الـMulticasting إذ أصبح لدينا الآن منصة قوية يعتمد عليها في عمل الـFragmentation وإعادة ترتيبها بكل سهولة بالإضافة إلى دعمه عملية Delivered on Sequence وهذا واضح من بنية الـHeader الخاصة بهذا البروتوكول انظر إلى الشكل التالي:



خامسا: How to Create a Voice Conference System Using Microsoft Direct Play 9 :

دعمت Microsoft تقنيات رائعة جدا للنقل الصوت في الإصدار الخاص بـDirect Play9 ويسمى أيضا DirectPlay Transport Protocol وهو جزء من مجموعة الـDirectX وكان الهدف من إطلاقها وجود معيار موحد لمبرمجين الألعاب فيما يخص الـ Network Games، وتعتمد هذه المكتبة على مجموعة من المعايير الخاصة بتشبيك حيث كان الهدف منها هو جعل عملية الاتصال ممكنة تحت جميع البيئات المختلفة و سواء كان البروتوكول المستخدم هو الـTCP/IP أو IPX الخاص بـNovel فإن عملية الاتصال ممكنة وبدون أي اختلافات من النواحي البرمجية، ومن أهم ميزات الـ DirectPlay Transport Protocol:

- Reliable delivery of messages حيث يدعم عملية التحقق التوصيل للجهة المعنية
- Sequential a delivery of messages حيث يدعم التوصيل وفق الترتيب الصحيح
- Send prioritization حيث يدعم عملية وضع أولويات للإرسال بناء على الأهمية
- Streaming Session حيث تدعم عملية النقل كStream Data ،

قدمت Microsoft هذه الحلول كبدائل لدعم بروتوكول الـSCTP المنتظر ، وقد حلت جميع المشكلات التي كانت تواجه المبرمجين لنقل الصوت عبر بروتوكول الـTCP أو الـUDP وحل محله أسلوب آخر لربط ضمن مستوى طبقة الـ Transport Layer ، وتحتوي الـ Direct Play على مجموعة ضخمة من الـClasses ومن أهمها :

أولا : الـ Connect Classes والخاصة بعملية الربط:

الـ Address، Guido , Peer Classes حيث تستخدم عند إنشاء الاتصال مع الطرف الأخر ، وتستخدم DirectPlay طريقة لتمييز البرنامج عن الأخر بتوليد Hash Code خاص بكل برنامج ويتم ذلك باستخدام الـGuido Class ويتم تمرير الكود المولد وعنوان الجهاز المقابل إلى الـ Peer Class وهذه العملية شبيهة بشكل كبير لعملية الربط باستخدام الـ Socket في بروتوكول TCP/IP ، ويتم استخدامها كما يلي كمثال:

C#:

```
using Microsoft.DirectX.DirectPlay;
.
.
.
Address hostAddress = new Address();
hostAddress.ServiceProvider = Address.ServiceProviderTcpIp;
                        // Select TCP/IP service provider

ApplicationDescription dpApp = new ApplicationDescription();
appGuid = Guid.NewGuid();          // Create a GUID for the application
dpApp.GuidApplication = appGuid;   // Set the application GUID
dpApp.SessionName = "My Session";  // Optional Session Name

myPeer.Host(dpApp, hostAddress);   // Begin hosting
```

VB.NET

```
Imports Microsoft.DirectX.DirectPlay
.
.
.
Private Address hostAddress = New Address()
Private hostAddress.ServiceProvider = Address.ServiceProviderTcpIp
' Select TCP/IP service provider

Private dpApp As ApplicationDescription = New
ApplicationDescription
Private appGuid = Guid.NewGuid() ' Create a GUID for the application
Private dpApp.GuidApplication = appGuid ' Set the application GUID
Private dpApp.SessionName = "My Session" ' Optional Session Name
myPeer.Host(dpApp, hostAddress) ' Begin hosting
```

طبعا يجب الاختيار طبيعة البروتوكول المستخدم سواء كان TCP/IP أو IPX وحتى نستطيع وضع العنوان المقابل أو الـ IP Address ويتم ذلك كما يلي:

C#:

```
using Microsoft.DirectX.DirectPlay;
ServiceProviderInfo[] mySPInfo;
Peer myPeer;
System.Windows.Forms.ListBox listBox1;
ApplicationDescription myAppDesc;
.
.
myPeer = new Peer();
hostAddress = new Address();
peerAddress = new Address();
```

```
// Set the service provider to TCP/IP
peerAddress.ServiceProvider = Address.ServiceProviderTcpIp;
hostAddress.ServiceProvider = Address.ServiceProviderTcpIp;

// Attach FindHostResponseEventHandler to receive
FindHostResponseMessages
myPeer.FindHostResponse += new
FindHostResponseEventHandler(myEnumeratedHosts);

// Call FindHosts to start the enumeration
myPeer.FindHosts(myAppDesc, hostAddress, peerAddress, null, 10, 0,
0, FindHostsFlags.OkToQueryForAddressing);
```

VB.NET

```
Imports Microsoft.DirectX.DirectPlay
Private mySPInfo As ServiceProviderInfo()
Private myPeer As Peer
Private listBox1 As System.Windows.Forms.ListBox
Private myAppDesc As ApplicationDescription

Private myPeer = New Peer
Private hostAddress = New Address
Private peerAddress = New Address

' Set the service provider to TCP/IP
Private peerAddress.ServiceProvider = Address.ServiceProviderTcpIp
Private hostAddress.ServiceProvider = Address.ServiceProviderTcpIp

' Attach FindHostResponseEventHandler to receive
FindHostResponseMessages
Private myPeer.FindHostResponse += New
FindHostResponseEventHandler(myEnumeratedHosts)

' Call FindHosts to start the enumeration
myPeer.FindHosts(myAppDesc, hostAddress, peerAddress, Nothing,
10, 0, 0, FindHostsFlags.OkToQueryForAddressing)
```

حيث تم تعريف نوع البروتوكول المستخدم وهو TCP/IP ويتم البحث عن الطرف الأخر في الشبكة باستخدام الـFindHosts Method والموجودة ضمن الـPeer Class ، وتتم عملية الربط مباشرة باستخدام الـConnect Method والموجودة ضمن الـ Peer Class وكما يلي:

C#:

```
using Microsoft.DirectX.DirectPlay;
// Structure for FindHostResponseMessages
public struct HostInfo
{
        public ApplicationDescription appdesc;
```

```
        public Address deviceAddress;
        public Address senderAddress;

}
.
.
.
Peer myPeer = new Peer();
HostInfo hostinfo = new HostInfo();

// The FindHostResponseEventHandler
public void myEnumeratedHosts(object o, FindHostResponseEventArgs
args)
{
        hostinfo.appdesc = args.Message.ApplicationDescription;
        hostinfo.deviceAddress = args.Message.AddressDevice;
        hostinfo.senderAddress = args.Message.AddressSender;
}
// Attach the ConnectCompleteEventHandler to receive
ConnectCompleteMessages
myPeer.ConnectComplete += new
ConnectCompleteEventHandler(OnConnectComplete);

// Call connect passing the Host information returned in the
FindHostResponse event
myPeer.Connect(hostinfo.appdesc, hostinfo.deviceAddress,
hostinfo.senderAddress, null,
ConnectFlags.OkToQueryForAddressing);
```

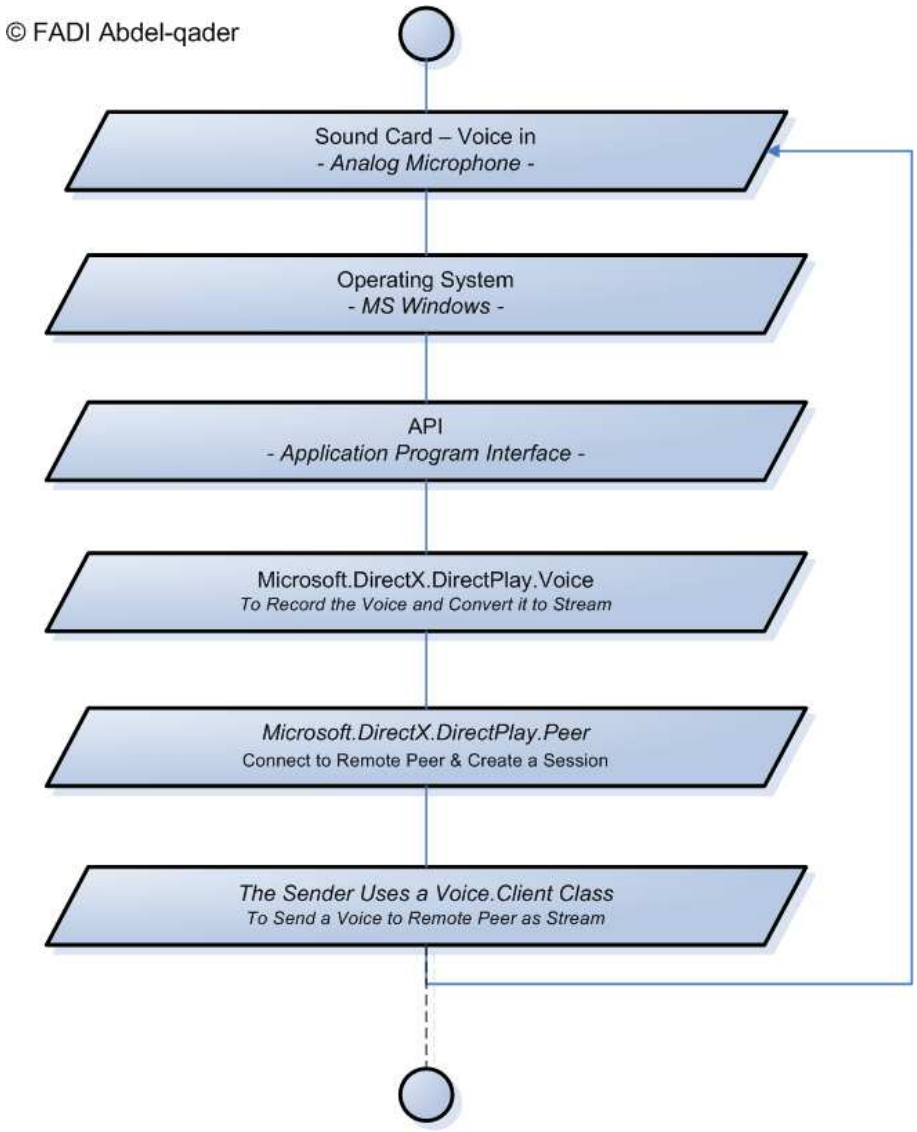
VB.NET

```
Imports Microsoft.DirectX.DirectPlay

' Structure for FindHostResponseMessages

Public Structure HostInfo
   Public appdesc As ApplicationDescription
   Public deviceAddress As Address
   Public senderAddress As Address

End Structure

Private myPeer As Peer = New Peer
Private hostinfo As HostInfo = New HostInfo

' The FindHostResponseEventHandler
```

```
Public Sub myEnumeratedHosts(ByVal o As Object, ByVal args As
FindHostResponseEventArgs)
    hostinfo.appdesc = args.Message.ApplicationDescription
    hostinfo.deviceAddress = args.Message.AddressDevice
    hostinfo.senderAddress = args.Message.AddressSender
End Sub

' Attach the ConnectCompleteEventHandler to receive
ConnectCompleteMessages

Private myPeer.ConnectComplete += New
ConnectCompleteEventHandler(OnConnectComplete)

' Call connect passing the Host information returned in the
FindHostResponse event

myPeer.Connect(hostinfo.appdesc, hostinfo.deviceAddress,
hostinfo.senderAddress, Nothing,
ConnectFlags.OkToQueryForAddressing)
```

ثانيا: Microsoft.DirectX.DirectPlay.Voice والخاصة بكل عمليات تسجيل ونقل وعرض الصوت:

تمر عملية تسجيل ونقل وعرض الصوت بمجموعة من المراحل ونلخصها بالشكل التالي:



وسوف نقسمها إلى مجموعة من الـ Classes حسب الوظيفة لكل منها.

1- الـ Classes الخاصة بطرف الـ Server لإنشاء وإدارة الـ Sessions :
Server Class ويستخدم كما يلي كمثال:

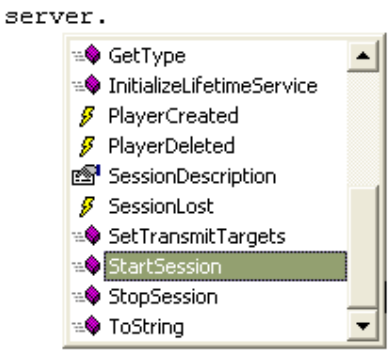
C#:

```
Server server = new Voice.Server(peerObject);
```

VB.NET

```
Dim server As Server = New Voice.Server(peerObject)
```

حيث نسند له الـ Peer Object والذي تم اشتقاقه من الـ Peer Class ، ويحتوي الـ Server Object على الـMethods الخاصة بعملية بدأ وإنهاء الجلسة بالإضافة إلى مجموعة من العمليات الأخرى:



ولبدأ الجلسة يجب أولا إسناد خصائص الجلسة إلى الـMethod StartSession حيث يتم تعريفها من خلال الـ SessionDescription و كما يلي:

C#:

```
//set up session description for the voice server
Voice.SessionDescription sessionDesc = new
Voice.SessionDescription();
sessionDesc.BufferAggressiveness =
Voice.BufferAggressiveness.Default;
sessionDesc.BufferQuality = Voice.BufferQuality.Default;
sessionDesc.Flags = 0;
sessionDesc.SessionType = type;
sessionDesc.GuidCompressionType = compressionType;
```

VB.NET

```
'set up session description for the voice server
Dim sessionDesc As Voice.SessionDescription = New
Voice.SessionDescription
sessionDesc.BufferAggressiveness =
Voice.BufferAggressiveness.Default
sessionDesc.BufferQuality = Voice.BufferQuality.Default
sessionDesc.Flags = 0
sessionDesc.SessionType = type
sessionDesc.GuidCompressionType = compressionType
```

ولإنشاء Voice Session نقوم بإنشاء Method نمرر لها الـ Peer Object والـ Voice Session Type ونوع الضغط المستخدم Compression Type ويتم ذلك كما يلي:

C#:

```
protected void CreateVoiceSession(Peer dpp, Voice.SessionType type,
Guid compressionType)
    {
try
```

```
    {
//set up session description for the voice server
Voice.SessionDescription sessionDesc = new
Voice.SessionDescription();
sessionDesc.BufferAggressiveness =
Voice.BufferAggressiveness.Default;
sessionDesc.BufferQuality = Voice.BufferQuality.Default;
sessionDesc.Flags = 0;
sessionDesc.SessionType = type;
sessionDesc.GuidCompressionType = compressionType;
//start the session
         try
          {
             server.StartSession(sessionDesc);
             mIsHost = true;
             mInSession = true;
          }
          catch(DirectXException dxe)
          {
             throw dxe;
          }
       }
       catch(Exception e)
       {
          throw e;
       }
   }
```

VB.NET

```
Protected Sub CreateVoiceSession(ByVal dpp As Peer, ByVal type As
Voice.SessionType, ByVal compressionType As Guid)
   Try
      'set up session description for the voice server
      Dim sessionDesc As Voice.SessionDescription = New
Voice.SessionDescription
      sessionDesc.BufferAggressiveness =
Voice.BufferAggressiveness.Default
      sessionDesc.BufferQuality = Voice.BufferQuality.Default
      sessionDesc.Flags = 0
      sessionDesc.SessionType = type
      sessionDesc.GuidCompressionType = compressionType
      'start the session
      Try
         server.StartSession(sessionDesc)
         mIsHost = True
         mInSession = True
      Catch dxe As DirectXException
```

```
        Throw dxe
      End Try
    Catch e As Exception
      Throw e
    End Try
End Sub
```

ويتم استدعائها كما يلي:

C#:

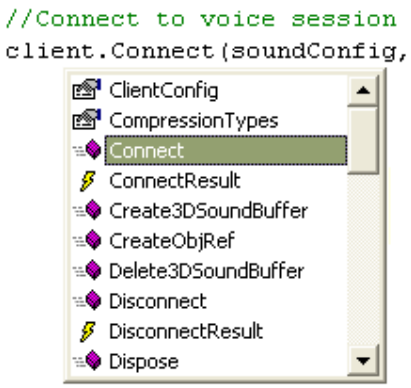
```
CreateVoiceSession(host, Voice.SessionType.Peer,
mConfigForm.CompressionGuid);
```

VB.NET

```
CreateVoiceSession(host, Voice.SessionType.Peer,
mConfigForm.CompressionGuid)
```

2- الـ Classes الخاصة بطرف الـ Client للاتصال مع الـ Sessions التي أنشئها الـServer:

الـ Client Class ومن أهم الـMethods الموجودة في الـ Client Class :



الدالـة Connect وتستخدم لـربط مـع الـSession Voice حيث يرسل الـ Server رقم الـSession للـ Client وعندها يتمكن الـ Client من الدخول إلى الجلسة ، ويتم ذلك كما يلي:

C#:

```
  protected void ConnectToVoiceSession(Peer dpp, Form wnd)
  {
    try
    {
Voice.SoundDeviceConfig soundConfig = new
Voice.SoundDeviceConfig();
//Set sound config to defaults
soundConfig.GuidPlaybackDevice  =
DSoundHelper.DefaultVoicePlaybackDevice;
soundConfig.GuidCaptureDevice =
DSoundHelper.DefaultVoiceCaptureDevice;
soundConfig.Window = wnd;
```

```
//TODO: add error message for specific failures?
//Connect to voice session
client.Connect(soundConfig, mClientConfig, Voice.VoiceFlags.Sync);
//set state
mInSession = true;
//set transmit targets to all players
int[] xmitTargets = new int[1];
xmitTargets[0] = (int) PlayerID.AllPlayers;
client.TransmitTargets = xmitTargets;
//get sound device config to check for half-duplex
soundConfig = client.SoundDeviceConfig;
mHalfDuplex =  ((soundConfig.Flags &
Voice.SoundConfigFlags.HalfDuplex) != 0);
}
catch(Exception e)
{
throw e;
}
   }
}
```

VB.NET

```
Protected Sub ConnectToVoiceSession(ByVal dpp As Peer, ByVal wnd
As Form)
   Try
Dim soundConfig As Voice.SoundDeviceConfig = New
Voice.SoundDeviceConfig
       'Set sound config to defaults
       soundConfig.GuidPlaybackDevice =
DSoundHelper.DefaultVoicePlaybackDevice
       soundConfig.GuidCaptureDevice =
DSoundHelper.DefaultVoiceCaptureDevice
       soundConfig.Window = wnd
       'TODO: add error message for specific failures?
       'Connect to voice session
       client.Connect(soundConfig, mClientConfig,
Voice.VoiceFlags.Sync)
       'set state
       mInSession = True
       'set transmit targets to all players
       Dim xmitTargets As Integer() = New Integer(0)
       xmitTargets(0) = CInt(PlayerID.AllPlayers)
       client.TransmitTargets = xmitTargets

       'get sound device config to check for half-duplex
       soundConfig = client.SoundDeviceConfig
```

```
mHalfDuplex = ((soundConfig.Flags And
Voice.SoundConfigFlags.HalfDuplex) <> 0)

   Catch e As Exception
      Throw e
   End Try
End Sub
```

لاحظ أن المشاكل في البروتوكولين الـ TCP والـUDP قد تم حلها في الـ DirectPlay لكن وكما هو معروف فإن الهدف من إنشاء الـ Direct Play لم يكن سوى لدعم برمجة الألعاب ومع المرونة الكبيرة التي تقدمها الـ Direct Play في عملية الاتصال الصوتي إلا أنها تفتقر لميزات الـ Voice Over IP والتي يقدمها بروتوكول الـ SCTP الخاص بالـ Linux...

وهكذا بينا ملخص عن أهم الطرق لاتصال الصوتي عبر الشبكة وطرق برمجة الـ Voice Chat تحت منصة الدوت نيت واهم ميزات وعيوب بروتوكولات الـTransport Layer ومدى إمكانياتها لنقل الصوت عبر الشبكة وأخيرا شرح لأهم الـ Classes والمستخدمة في الاتصال الصوتي باستخدام الـ DirectPlay Transport Protocol، سنقوم الآن بتطبيق برنامج Video Voice Conference System باستخدام الـ TCP لإرسال الصورة والـ UDP لإرسال الصوت .

ثالثـا: تصـميم برنـامج للمحادثـة الصـوتية والمرئيـة وطـرق التعامـل مـع الــ TAPI Telephony في بيئة الدوت نيت:

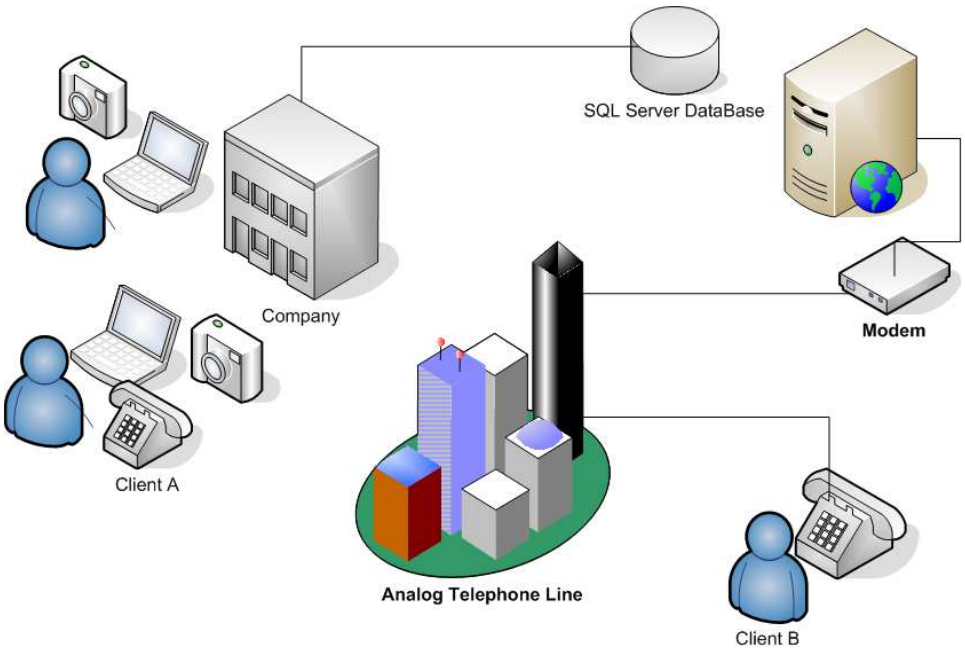
Design an Advanced Video /Voice Conferencing System By Using API & TAPI Telephony to Transfer Video/Voice Through Networks & Voice Through Analog Telephone Line:

ملاحظة هامة جدا:

يفترض بك قبل البدء بقراءة هذا الجزء من فصل الـ VOIP قراءة المواضيع التالية:

1- Streaming in Dot Net
2- TCP,UDP Socket Programming
3- Advanced Multicasting Systems
4- الجزء الأول من هذا الفصل VOIP

بينا في الجزء الأول من هذا الفصل كيفية استخدام الـ API unmanaged methods من الملف winmm.dll المخصص لتعامل مع الــ Multimedia في بيئة نظام التشغيل واستخدمنا مجموعة الـ Methods لتسجيل الصوت من المايكروفون وتحويله إلى Byte Array لإرساله ومن ثم تحويل الـ Byte Array إلى صوت مرة أخرى باستخدام أسلوب العرض FIFO – First in First Out ...، وسنحاول في هذا المثال الاستفادة من خدمات الـTAPI telephony وذلك لنقل الصوت من الحاسوب إلى طرف آخر يستخدم الهاتف النقال أو الهاتف الأرضي Analog Phone، ويمكن الاستفادة من الـ TAPI telephony بعدة طرق أسهلها الاستفادة من البرنامج الموجود مع نظام التشغيل TAPI V3.0 والمتاح مع نظام التشغيل Windows XP ... ، وسيكون المخطط العام للمشروع كما في الشكل التالي:

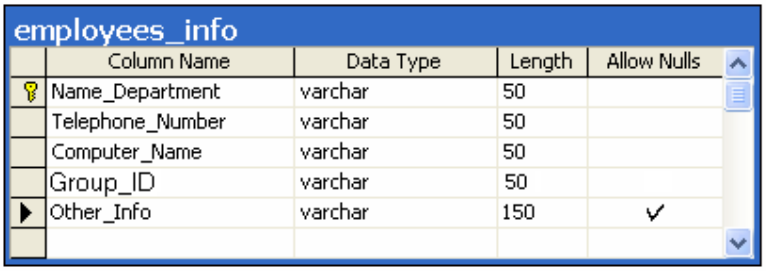


A Simple Example that shows a Company that has an Internal VOIP & Video Conference System that Connected With TAPI Telephony

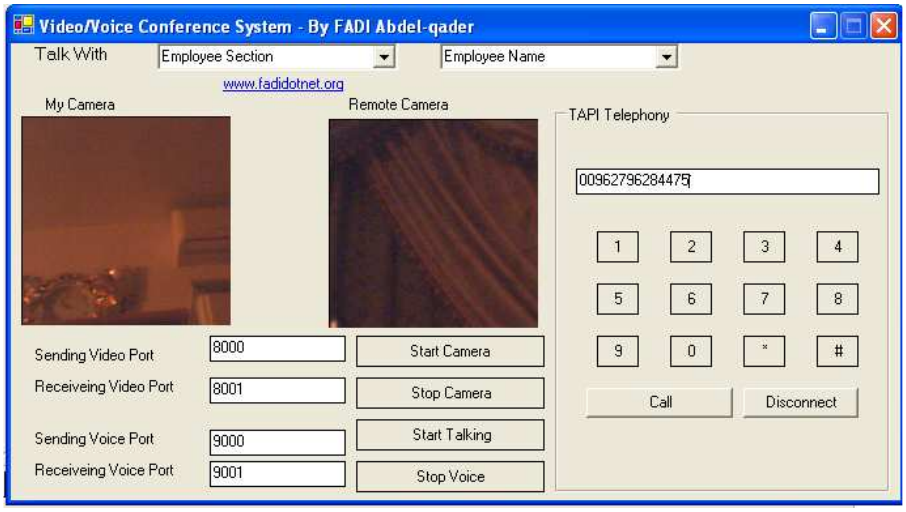
1- سيحتوي المشروع على Database Server ، وتقتصر وظيفته على تخزين معلومات الموظفين في الشركة كأرقام الهواتف و العناوين الـ Computers Names الخاصة بأجهزة الموظفين...

2- سيتمكن جميع الموظفين من الاتصال الصوتي والمرئي مع أي من أقسام الشركة، بالإضافة إلى استخدام الـTAPI Telephony للاتصال مع الـ Analog Phone وبالعكس... وللبدء بتطبيق المشروع سنقوم بتصميم قاعدة بيانات تحتوي على أسماء الموظفين والـComputer Name لكل منهم وأقسامهم وأرقام هواتفهم، كما يلي:

Name@Department	String (Primary Kay)
Computer Name (hidden for Clients)	String
Telephone Number	String
Group ID	String (foreign Kay)
Other Info (Email Address)	String



وجدول آخر يحتوي على الـ Group ID والـ Group Name حيث يضاف إليه أسماء الأقسام في الشركة ويرتبط مع جدول معلومات الموظفين بالـ Group ID ... ، بعد تصميم قاعدة البيانات السابقة ، سنبدأ بتصميم شاشة الـClient التي سيستخدمها الموظفين داخل الشركة وسيكون الشكل العام لها كما يلي:



حيث يحدد الموظف أو المدير اسم القسم الذي يريد الاتصال معه وعند اختيار اسم القسم سيظهر الـ Combo box المقابل جميع أسماء الموظفين الذين يعملون في ذلك القسم وفي حال اختيار إحداهم فإنه يقوم بعمل SELECT Query على الـ Telephone Number الخاص بالموظف بالإضافة إلى الـ Computer_Name الخاصة به ، حيث يمكنه إما إجراء مكالمة باستخدام الـ TAPI Telephony أو إجراء مكالمة داخلية باستخدام الـ VOIP ، وفي الحالة الأولى فإن الاتصال سيتم عبر الـ Modem وخطوط الاتصالات العادية ، أما في حالة اختيار الاتصال بالـ VOIP عندها يتم البدء بإرسال الصوت إلى الـ Computer_Name الذي يريد

الاتصال معه (يستخرج من قاعدة البيانات عند اختيار اسم الموظف)و يتم بإرسال تنبيه إلى الـClient لإخباره بوجود مكالمة صوتية منتظرة وفي حال الموافقة يحول الصوت إلى الـComputer الخاص بالموظف ، وتتم العملية في الاتصال المرئي بنفس الطريقة التي شرحناها في الاتصال الصوتي ...

تتم عملية استخدام الـ TAPI Telephony في الدوت نيت باستخدام دوال الـ API ويمكننا الاستفادة من الـMethods التي يوفرها الملف المخصص لتعامل مع الـ TAPI Telephony وهو tapi32.dll ويحتوي هذا الملف على عدد ضخم من الـ Methods والتي يمكن التعامل معها في بيئة الدوت نيت وكمثال عليها:

Method Name	Description
tapiRequestMakeCall(string stNumber, string stDummy1, string stDummy2, string stDummy3)	وتستخدم لإجراء عملية الاتصال باستخدام الـ Modem حيث يمرر لها رقم الهاتف الذي نريد إجراء اتصال معه
phoneShutdown (hPhoneApp app) [DllImport("tapi32.dll")] static extern long phoneShutdown(long hPhoneApp);	وتستخدم لإغلاق برنامج الـ TAPI Telephony
LONGWINAPIphoneSetVolume(HPHONE hPhone, DWORD dwHookSwitchDev, DWORD dwVolume)	وتستخدم هذه الدالة لتحديد درجة الصوت ويمرر لها الدرجة بالـ Hex حيث تعبر القيمة 0x00000000 على عدم وجود صوت أما القيمة 0x0000FFFF فتعبر عن الـ maximum volume ...

تنويه : للإطلاع على كافة الـ Methods الخاصة بالـ TAPI Telephony أنظر الملحقات (Part 5). Appendixes A

سنستخدم في هذا المشروع الـ tapiRequestMakeCall Method والتي نمرر لها رقم الهاتف لإجراء عملية الاتصال ، ويتم تعريفها كما يلي:

C#:

```
[System.Runtime.InteropServices.DllImport("tapi32.dll")]
static extern long tapiRequestMakeCall(string stNumber, string
stDummy1, string stDummy2, string stDummy3);
.
.
.

long RetVal = tapiRequestMakeCall(Telephone_number, "", "", "");
```

VB.NET:

```
<System.Runtime.InteropServices.DllImport("tapi32.dll")> _
```

```
Shared Function tapiRequestMakeCall(ByVal stNumber As String,
ByVal stDummy1 As String, ByVal stDummy2 As String, ByVal
stDummy3 As String) As Long
End Function
.
.
Dim RetVal As Long = tapiRequestMakeCall(Telephone_number, "",
"", "")
```

تتم عملية استقبال وإرسال الصوت كما يلي:

C#:

```
private void Voice_In() // استقبال الصوت من السيرفر
{
byte[] br;
r.Bind(new IPEndPoint(IPAddress.Any,
int.Parse(text_ReceiveingPORT.Text)));
        while (true)
                {
        br = new byte[16384];
        r.Receive(br);
        m_Fifo.Write(br, 0, br.Length);
                }
}

private void Voice_Out(IntPtr data, int size)//إرسال الصوت إلى السيرفر
{
if (m_RecBuffer == null || m_RecBuffer.Length < size)
m_RecBuffer = new byte[size];
System.Runtime.InteropServices.Marshal.Copy(data, m_RecBuffer, 0,
size);
r.SendTo(m_RecBuffer, new
IPEndPoint(IPAddress.Parse(Server_name),
int.Parse(text_SendingPort.Text)));
}
```

VB.NET:

```
Private Sub Voice_In() ' استقبال الصوت من السيرفر
   Dim br As Byte()
   r.Bind(New IPEndPoint(IPAddress.Any,
Integer.Parse(text_ReceiveingPORT.Text)))
   Do While True
      br = New Byte(16383)
      r.Receive(br)
      m_Fifo.Write(br, 0, br.Length)
   Loop
End Sub
```

```
Private Sub Voice_Out(ByVal data As IntPtr, ByVal size As Integer)
'إرسال الصوت إلى السيرفر
   If m_RecBuffer Is Nothing OrElse m_RecBuffer.Length < size Then
      m_RecBuffer = New Byte(size - 1)
   End If
   System.Runtime.InteropServices.Marshal.Copy(data, m_RecBuffer,
0, size)
   r.SendTo(m_RecBuffer, New
IPEndPoint(IPAddress.Parse(Server_name),
Integer.Parse(text_SendingPort.Text)))
End Sub
```

وتتم عملية إرسال واستقبال صورة الكاميرا من وإلى الـ Server كما يلي:

C#:

```
private void Start_Receiving_Video_Conference()
{
        try
        {

        // Open The Port
mytcpl = new TcpListener (int.Parse(text_Camera_rec_port.Text));
mytcpl.Start (); // Start Listening on That Port
        mysocket = mytcpl.AcceptSocket ();
ns = new NetworkStream (mysocket);
pictureBox2.Image = Image.FromStream(ns);
mytcpl.Stop();
        if (mysocket.Connected ==true)
                        {
        while (true)
                {
        Start_Receiving_Video_Conference ();
                        }
                }
myns.Flush();

                }
catch (Exception){}
}

private void Start_Sending_Video_Conference(string remote_IP,int
port_number)
        {
        try
        {
```

```
MemoryStream ms = new MemoryStream();// Store it in Binary Array
as Stream
pictureBox1.Image.Save(ms,System.Drawing.Imaging.ImageFormat.Jp
eg);
byte[] arrImage = ms.GetBuffer();
myclient = new TcpClient (server_IP,port_number);//Connecting with
server
myns = myclient.GetStream ();
                mysw = new BinaryWriter (myns);
                mysw.Write(arrImage);//send the stream to above address
                ms.Flush();
                mysw.Flush();
                myns.Flush();
                ms.Close();
                mysw.Close ();
                myns.Close ();
                myclient.Close ();
        }
catch (Exception ex)
{
MessageBox.Show(ex.Message,"Video Conference Error
Message",MessageBoxButtons.OK,MessageBoxIcon.Error);
}
}
```

VB.NET:

```
Private Sub Start_Receiving_Video_Conference()
   Try
      mytcpl = New
TcpListener(Integer.Parse(text_Camera_rec_port.Text))
      mytcpl.Start()
      mysocket = mytcpl.AcceptSocket()
      ns = New NetworkStream(mysocket)
      pictureBox2.Image = Image.FromStream(ns)
      mytcpl.Stop()

      If mysocket.Connected = True Then
         Do While True
            Start_Receiving_Video_Conference()
         Loop
      End If
      myns.Flush()
   Catch e1 As Exception
   End Try
End Sub
```

```
Private Sub Start_Sending_Video_Conference(ByVal remote_IP As
String, ByVal port_number As Integer)
   Try
      Dim ms As MemoryStream = New MemoryStream
      pictureBox1.Image.Save(ms,
System.Drawing.Imaging.ImageFormat.Jpeg)
      Dim arrImage As Byte() = ms.GetBuffer()
      myclient = New TcpClient(server_IP, port_number)
      myns = myclient.GetStream()
      mysw = New BinaryWriter(myns)
      mysw.Write(arrImage)
      ms.Flush()
      mysw.Flush()
      myns.Flush()
      ms.Close()
      mysw.Close()
      myns.Close()
      myclient.Close()
   End Try
```

في الطرف الأخر فإن الاختلاف يجب أن يكون في الـ Port وكمثال إذا أرسل الموظف من قسم المبيعات إلى الموظف في قسم الحاسوب على الـ Port 5000 فيجب في هذه الحالة استقبال الصوت في الطرف الآخر على نفس الـ Port التي تم الإرسال عليها أما في حالة استخدام Server عندها يمكن لكل الـClients الإرسال على Port معين إلى الـServer وكذلك قيام الـServer بإرسال الـMessages إلى كل الـClients على Port محدد.

Chapter 11
Raw Socket Programming

- Introduction to Raw Socket & Raw Protocols
- Raw Socket Programming
    - ICMP – Internet Control Message Protocol Programming (Ping & Tracing)
    - Using ARP Protocol to Get The MAC of a Remote Machine in Dot Net
- Introduction to Packet Sniffing Applications

أولا Introduction to Raw Socket & Raw Protocols

يعتبر موضوع الـ Raw Socket من المواضيع الهامة وخاصة للبرمجيات التي تعتمد على التعامل مع البروتوكولات في المستوى الأدنى من الـ Transport Layer ومنها بروتوكول الـARP والـ RARP والمستخدم بشكل أساسي لجلب الـ MAC أو الـ Physical Address للـ Ethernet Card (لمزيد من المعلومات انظر الفصل الأول) والـ IGMP المخصص لإدارة المجموعات في الـ Multicasting بالإضافة إلى بروتوكول الـ ICMP والذي يستخدم بشكل أساسي في عملية الـ Ping بالإضافة إلى الكثير من العمليات الأخرى لإرسال حالة الـMessage بعد إرسالها إلى الـ Destination ، ويأخذ الـ Header الخاص بهذا البروتوكول 64 Byte مقسمة كما هو واضح في الشكل التالي:

Type	Code	Checksum
Identifier		Sequence
Message		

ICMP Header

16 Bytes ويحدد فيها نوع الرسالة المرسلة والـ Code الخاص بهذه الرسالة وتقسم إلى نوعين Query استعلام أو Error Message تقرير لخطأ ما ويوضح الجدول 1-11 هذه الأنواع:

TYPE	CODE	Description	Query	Error
0	0	Echo Reply	x	
3	0	Network Unreachable		X
3	1	Host Unreachable		X
3	2	Protocol Unreachable		X
3	3	Port Unreachable		X
3	4	Fragmentation needed but no frag. bit set		X
3	5	Source routing failed		X
3	6	Destination network unknown		X
3	7	Destination host unknown		X
3	8	Source host isolated (obsolete)		X
3	9	Destination network administratively prohibited		X
3	10	Destination host administratively prohibited		X
3	11	Network unreachable for TOS		X
3	12	Host unreachable for TOS		X

TYPE	CODE	Description	Query	Error
3	13	Communication administratively prohibited by filtering		X
3	14	Host precedence violation		X
3	15	Precedence cutoff in effect		X
4	0	Source quench		
5	0	Redirect for network		
5	1	Redirect for host		
5	2	Redirect for TOS and network		
5	3	Redirect for TOS and host		
8	0	Echo request	x	
9	0	Router advertisement		
10	0	Route solicitation		
11	0	TTL equals 0 during transit		X
11	1	TTL equals 0 during reassembly		X
12	0	IP header bad (catchall error)		X
12	1	Required options missing		X
13	0	Timestamp request (obsolete)	x	
14		Timestamp reply (obsolete)	x	
15	0	Information request (obsolete)	x	
16	0	Information reply (obsolete)	x	
17	0	Address mask request	x	
18	0	Address mask reply	x	

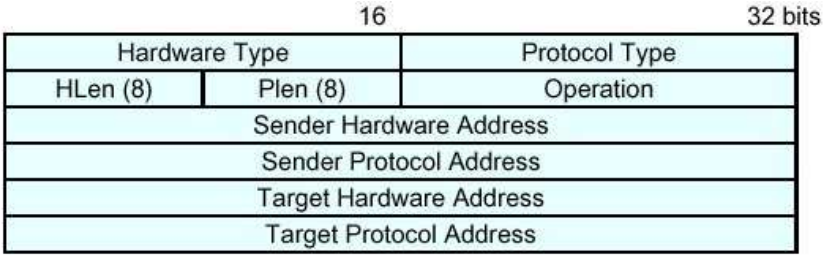
جدول 1-11 أنواع الـ ICMP Messages

16 Bytes أخرى ويوضع فيها الكود الخاص بالـ Checksum لتحقق من وصول الـ Header الخاص بالـICMP Message بشكل الصحيح (ارجع إلى الفصل الأول لمزيد من المعلومات حول الـ Checksum وطريقة حسابه) وهو اختياري.

32 Bytes مقسمة إلى 16 للـ ID Packet و 16 لرقم التسلسلي للـ ICMP Packet المرسل.

ومن البروتوكولات الهامة أيضا في الـ Network Layer بروتوكول الـ ARP – Address Resolution Protocol إذ يستخدم لجلب الـ MAC Address للجهة التي تريد الإرسال إليها ومن الاستخدامات الأخرى لهذا البروتوكول إجراء عملية الـ Filtering للـ MAC Address وتستخدم هذه العملية بشكل كبير لحماية الشبكات اللاسلكية من عملية الدخول غير المخول حيث تضاف الـ MAC Addresses المسموح لها باستخدام الشبكة ضمن الـ MAC List في الـ Router لمنع الدخول لغير الأجهزة التي تملك MAC Address مشابه للموجود ضمن الـMAC List ... ، ويحتوي الـ Header الخاص به على كل من الـ Hardware

Address والـ Protocol Address بالإضافة إلى Hardware & Protocol Type المستخدم لهذا البروتوكول ، ويوضح الشكل التالي التركيب العام له:



ومن المعروف أن الـ ICMP و الـ ARP تصنف من البروتوكولات التي تعمل على مستوى الـNetwork Layer ضمن الـ TCP/IP Model وهذا يعني أنها Connectionless Protocols، و لا تعتمد هذه البروتوكولات على استخدام أي من البروتوكولات في الـTransport Layer إذ لا حاجة لاستخدام الـ Ports عند التعامل معها...
سنبين في الجزء التالي من هذا الفصل كيفية برمجة الـ Raw Socket وبروتوكولاتها مثل الـICMP ، بالإضافة إلى طرق التعامل مع الـ ARP Protocol في الدوت نيت.

ثانيا : Raw Socket Programming :

لتعامل مع الـ Raw Socket في الدوت نيت يجب أولا تعريف الـ Raw Socket Type بعد ذلك يمكننا اختيار أي من البروتوكولات التي تعمل على الـ Network Layer وتدعم الـRaw Socket التعامل مع البروتوكولات التالية وكما هو واضح في الجدول 2-15 التالي:

Protocol Name	Description
GGP	Gateway-to-Gateway Protocol
ICMP	Internet Control Message Protocol
IGMP	Internet Group Management Protocol
IP	A raw IP packet
IPX	Novell IPX Protocol
ND	Net Disk Protocol
PUP	Xerox PARC Universal Protocol (PUP)
ROW	A raw IP packet
SPX Version 1 & 2	Novell SPX Protocol

Table 2-15 – و يبين البروتوكولات التي تدعمها الـ Raw Socket

وكمثال لبرمجة بروتوكول الـ ICMP يجب أولا تعريف الـ Raw Socket كما يلي:

C#:
```
Socket row_socket = new
Socket(AddressFamily.InterNetwork,SocketType.Raw,ProtocolType.Icmp);
```

VB.NET:
```
Dim row_socket As Socket = New
Socket(AddressFamily.InterNetwork, SocketType.Raw,
ProtocolType.Icmp)
```

ويعتبر بروتوكول الـ ICMP بروتوكول Connectionless وهذا يعني عدم حاجته أيضا لإجراء عملية الـConnect أو الـ Binding مع الـ Socket لذلك نحدد له فقط في الـIPEndPoint Object الـ IP Address للـ Destination ودون تحديد رقم الـ Port حيث تأخذ القيمة صفر وكما يلي:

C#:
```
IPEndPoint iep = new IPEndPoint(IPAddress.Parse("10.0.0.10"), 0);
```

VB.NET:
```
Dim iep As IPEndPoint = New
IPEndPoint(IPAddress.Parse("10.0.0.10"), 0)
```

وتتم عملية الإرسال مباشرة باستخدام الـ Sendto Method بعد تحديد الشكل العام للـ ICMP Header Format حيث ترسل محتوياته كـ Byte Array وتتم عملية الاستقبال باستخدام الـ ReceiveFrom Method وبنفس الطريقة التي أرسل بها ، ولإنشاء ICMP Header Class يجب التقيد بالشكل العام لهذا البروتوكول والـData Type لكل Field وكما هو واضح في الجدول 3-15 التالي:

Data Variable	Size	Type
Type	1 byte	Byte
Code	1 byte	Byte
Checksum	2 bytes	Unsigned 16-bit integer
Message	multibyte	Byte array

Table 15-3 , ICMP Message Format

ولإنشاء الـ ICMP Class يجب التقيد بالتركيب السابق للـ ICMP Header ويتم ذلك كما يلي:

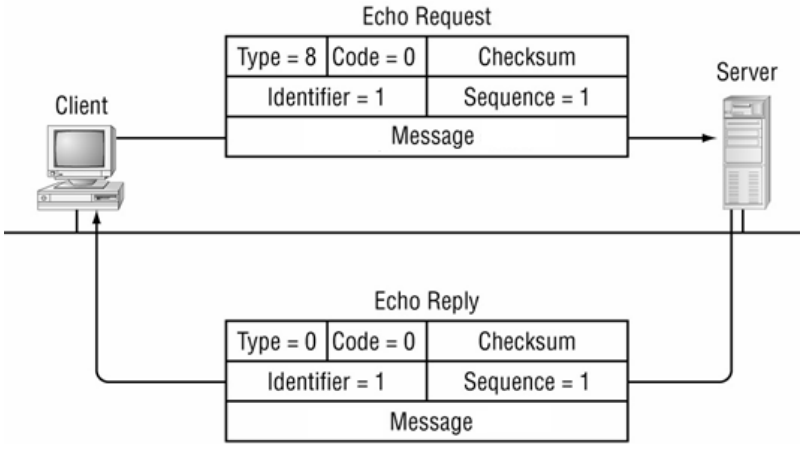
C#:
```
class ICMP
{
public byte Type;
public byte Code;
```

```
public UInt16 Checksum;
public int MessageSize;
public byte[] Message = new byte[1024];
}
```

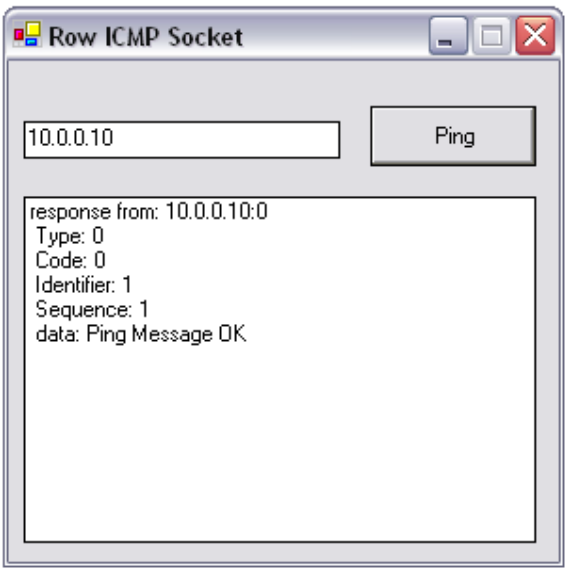
VB.NET:

```
Class ICMP
   Public Type As Byte
   Public Code As Byte
   Public Checksum As UInt16
   Public MessageSize As Integer
   Public Message As Byte() = New Byte(1024)
End Class
```

سنقوم الآن بتنفيذ برنامج بسيط لإجراء عملية الـ Ping ومن المعروف أن الـ Ping يستخدم الـ8 Type لإرسال والـ 0 Type عند الاستقبال حيث يقوم الطرف المقابل باستقبال الرسالة وإعادة إرسالها كما هي إلى الـSourceAddress ، وإذا تم استقبال الـ ICMP Message بالـ0 Type هذا يعني انه تم الوصول إلى الـDestination ويمكن التحقق من وصول الـICMP Header بالشكل الصحيح بعد حساب ناتج الـ Checksum فإذا تطابق مع مقلوب مجموع الـHeader Data عندها نستطيع معرفة أنه تم تسليم الـMessage لـ Destination بشكل الصحيح وكما هو واضح في الشكل التالي:



وللبدء بالتطبيق سيكون الشكل العام للبرنامج كما في الشكل التالي:



ولإنشاء الـ ICMP Class يجب أولا تعريف الـ ICMP Header وتحويله إلى Byte Array بعد إسناد الـPing Data التي نريدها إليه وكما يلي:

C#:

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

class ICMP
{
        public byte Type;
        public byte Code;
        public UInt16 Checksum;
        public int MessageSize;
        public byte[] Message = new byte[1024];
        public ICMP()
        {
        }
        public ICMP(byte[] data, int size)
        {
                Type = data[20];
                Code = data[21];
                Checksum = BitConverter.ToUInt16(data, 22);
                MessageSize = size - 24;
                Buffer.BlockCopy(data, 24, Message, 0, MessageSize);
        }

        public byte[] getBytes()
        {
byte[] data = new byte[MessageSize + 9];
```

```
Buffer.BlockCopy(BitConverter.GetBytes(Type), 0, data, 0, 1);
Buffer.BlockCopy(BitConverter.GetBytes(Code), 0, data, 1, 1);
Buffer.BlockCopy(BitConverter.GetBytes(Checksum), 0, data, 2, 2);
Buffer.BlockCopy(Message, 0, data, 4, MessageSize);
return data;
        }

        public UInt16 getChecksum()
        {
        UInt32 chcksm = 0;
        byte[] data = getBytes();
        int packetsize = MessageSize + 8;
        int index = 0;
                while ( index < packetsize)
                {
        chcksm += Convert.ToUInt32(BitConverter.ToUInt16(data,
index));
        index += 2;
                }
        chcksm = (chcksm >> 16) + (chcksm & 0xffff);
        chcksm += (chcksm >> 16);
        return (UInt16)(~chcksm);

        }
}
```

VB.NET:

```
Imports Microsoft.VisualBasic
Imports System
Imports System.Net
Imports System.Net.Sockets
Imports System.Text

Friend Class ICMP
        Public Type As Byte
        Public Code As Byte
        Public Checksum As UInt16
        Public MessageSize As Integer
        Public Message As Byte() = New Byte(1023)
        Public Sub New()
        End Sub

        Public Sub New(ByVal data As Byte(), ByVal size As Integer)
                Type = data(20)
                Code = data(21)
                Checksum = BitConverter.ToUInt16(data, 22)
                MessageSize = size - 24
```

```
            Buffer.BlockCopy(data, 24, Message, 0, MessageSize)
        End Sub

        Public Function getBytes() As Byte()
Dim data As Byte() = New Byte(MessageSize + 9 - 1)
Buffer.BlockCopy(BitConverter.GetBytes(Type), 0, data, 0, 1)
Buffer.BlockCopy(BitConverter.GetBytes(Code), 0, data, 1, 1)
Buffer.BlockCopy(BitConverter.GetBytes(Checksum), 0, data, 2, 2)
        Buffer.BlockCopy(Message, 0, data, 4, MessageSize)
            Return data
        End Function

        Public Function getChecksum() As UInt16
            Dim chcksm As UInt32 = System.Convert.ToUInt32(0)
            Dim data As Byte() = getBytes()
            Dim packetsize As Integer = MessageSize + 8
            Dim index As Integer = 0
            Do While index < packetsize
chcksm +=chcksm Convert.ToUInt32(BitConverter.ToUInt16(data,
index))
index += 2
        Loop
      chcksm = (chcksm >> 16) + (chcksm And &HFFFF)
      chcksm += (chcksm >> 16)
      Return System.Convert.ToUInt32((Not chcksm))
        End Function
End Class
```

حيث قمنا بإنشاء الـ getBytes Method لتحويل الـ ICMP Packet إلى Byte Array باستخدام الـBitConverter.GetBytes Method وأنشئنا الـ getChecksum Method لحساب الـ Checksum للـICMP Packet حيث سيسند إليه وسيستخدم لعملية التحقق من وصول الـ Packet Header للـ Destination بالشكل الصحيح .

ويمكن استخدام الـ Class السابق مباشرة بعد تعريف Raw Socket ثم تعريف IPEndpoint Object لإسناده إلى sendto Method ثم اشتقاق instance Object من الـICMP Class السابق و إسناد قيم ابتدائية للـ ICMP Header وإنشاء ICMP Message وتحويلها إلى Byte Array وفي النهاية يمكننا إرسالها إلى الـ Destination باستخدام الـ sendto method وبتأكيد يمكننا تغير مدة الـ Time Out للمدة التي سينتظرها الـSource إلى حين الدخول على الـ SocketException في حالة عدم الاستجابة ويتم كل ذلك كما يلي:

C#:

```
byte[] data = new byte[1024];
int recv;
Socket host = new Socket(AddressFamily.InterNetwork,
SocketType.Raw,ProtocolType.Icmp);
IPEndPoint iep = new IPEndPoint(IPAddress.Parse(textBox1.Text),0);
```

```
EndPoint ep = (EndPoint)iep;
ICMP packet = new ICMP();

packet.Type = 0x08;
packet.Code = 0x00;
packet.Checksum = 0;

Buffer.BlockCopy(BitConverter.GetBytes((short)1), 0, packet.Message,
0, 1);

data = Encoding.ASCII.GetBytes("Ping Message OK");

Buffer.BlockCopy(data, 0, packet.Message, 4, data.Length);
packet.MessageSize = data.Length + 4;
int packetsize = packet.MessageSize + 4;
UInt16 chcksum = packet.getChecksum();
packet.Checksum = chcksum;

host.SetSocketOption(SocketOptionLevel.Socket,SocketOptionName.R
eceiveTimeout, 1000);

host.SendTo(packet.getBytes(), packetsize, SocketFlags.None, iep);
```

VB.NET:

```
Dim data As Byte() = New Byte(1023)
Dim recv As Integer
Dim host As Socket = New Socket(AddressFamily.InterNetwork,
SocketType.Raw,ProtocolType.Icmp)
Dim iep As IPEndPoint = New
IPEndPoint(IPAddress.Parse(textBox1.Text),0)
Dim ep As EndPoint = CType(iep, EndPoint)
Dim packet As ICMP = New ICMP()

packet.Type = &H08
packet.Code = &H00
packet.Checksum = 0
Buffer.BlockCopy(BitConverter.GetBytes(CShort(1)), 0,
packet.Message, 0, 2)
Buffer.BlockCopy(BitConverter.GetBytes(CShort(1)), 0,
packet.Message, 2, 2)
data = Encoding.ASCII.GetBytes("Ping Message OK")

Buffer.BlockCopy(data, 0, packet.Message, 4, data.Length)
packet.MessageSize = data.Length + 4
Dim packetsize As Integer = packet.MessageSize + 4
Dim chcksum As UInt16 = packet.getChecksum()
packet.Checksum = chcksum
```

```
host.SetSocketOption(SocketOptionLevel.Socket,SocketOptionName.R
eceiveTimeout, 1000)
host.SendTo(packet.getBytes(), packetsize, SocketFlags.None, iep)
```

ولاستقبال الرد على الرسالة من الـ Destination سنستخدم نفس التعريف السابق للـ Socket حيـث سنسـتخدم الـــReceiveFrom Method ونسـند إليهـا الـــ Value Reference للـIPEndpoint الذي أرسلنا من خلاله الرسالة ، ونسند الرد الذي تم استقباله إلى الـ ICMP Instance Object الذي أنشئ من الـ ICMP Class ، ويتم كل ذلك كما يلي:

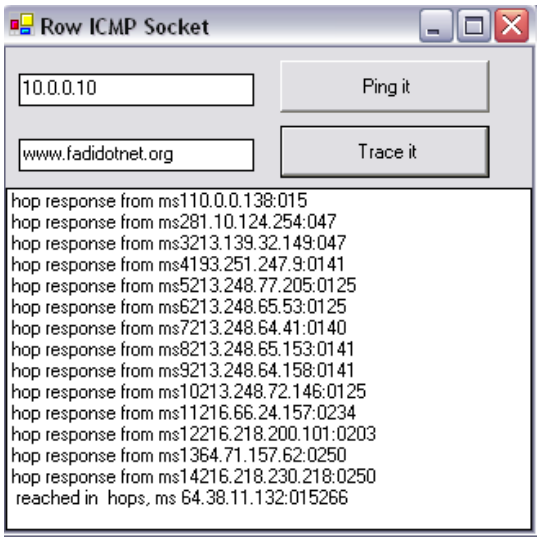
C#:

```
try
{
data = new byte[1024];
recv = host.ReceiveFrom(data, ref ep);
}
catch (SocketException)
        {
        listBox1.Items.Clear();
        listBox1.Items.Add("No response from remote host");
        return;
        }
ICMP response = new ICMP(data, recv);
        listBox1.Items.Clear();
        listBox1.Items.Add("response from: "+ ep.ToString());
        listBox1.Items.Add(" Type: "+ response.Type);
        listBox1.Items.Add(" Code: "+ response.Code);
        int Identifier = BitConverter.ToInt16(response.Message, 0);
        int Sequence = BitConverter.ToInt16(response.Message, 2);

        listBox1.Items.Add(" Identifier: "+ Identifier);
        listBox1.Items.Add(" Sequence: "+ Sequence);
string stringData = Encoding.ASCII.GetString(response.Message,4,
response.MessageSize - 4);
listBox1.Items.Add(" data: "+stringData);
host.Close();
```

VB.NET:

```
Try
data = New Byte(1023)
recv = host.ReceiveFrom(data, ep)
Catch e1 As SocketException
listBox1.Items.Clear()
listBox1.Items.Add("No response from remote host")
Return
End Try

Dim response As ICMP = New ICMP(data, recv)

listBox1.Items.Clear()
listBox1.Items.Add("response from: " & ep.ToString())
listBox1.Items.Add(" Type: " & response.Type)
listBox1.Items.Add(" Code: " & response.Code)

Dim Identifier As Integer = BitConverter.ToInt16(response.Message, 0)
Dim Sequence As Integer = BitConverter.ToInt16(response.Message, 2)

listBox1.Items.Add(" Identifier: " & Identifier)
listBox1.Items.Add(" Sequence: " & Sequence)

Dim stringData As String =
Encoding.ASCII.GetString(response.Message,4, response.MessageSize
- 4)
listBox1.Items.Add(" data: " & stringData)
host.Close()
```

ويمكننا أيضا استخدام الـ Class السابق لإنشاء برنامج لعمل الـ Tracing ومن المعروف بالـTracing Routing انه يقوم بتعقب الـ Hops أو الـ Routers التي يمر عبرها الـPacket حيث يرجع كل Router يمر الـ Packet من خلاله ICMP Replay Message تحتوي على الـ Message التي أرسلت بالإضافة إلى الـ IP للـ Router الذي تم تمرير الـMessage من خلاله والوقت الذي استغرق للوصول إلى كل Router حتى وصول الـPacket إلى الـ Destination ، وسيكون الشكل العام للبرنامج كما يلي:



وكما هو واضح من تعريف الـ Tracing فإن البرنامج سيعتمد على جلب الـ IP لكل Router يمر من خلاله إلى حين الوصول إلى الطرف المعني ويمكن إجراء هذه العملية باستخدام الـIpTimeToLive ضمن الـMethod SetSocketOption حيث نضعها داخل Loop وفي كل مرة نزيد الـ Loop بمقدار واحد ونسند قيمته إلى الـ TTL حيث يعبر كل TTL عن Hop أو Router تم المرور من خلاله ويتم إرسال Message Ping إلى ذلك الـ Hop وخلال ذلك نقيس الفترة الزمنية المستغرقة بين عملية إرسال الـ Ping واستقبال الرد من كل Hop إلى حين الوصول إلى آخر Hop مر الـ Packet من خلاله ويمكننا قياس ذلك باستخدام الـ TickCount ضمن الـClass Environment وترجع هذه الـ Property قيمة Integer بالـ Millisecond من بدأ نظام التشغيل بالعمل إلى الوقت الحالي حيث يمكننا اخذ هذه القيمة عند بداية عمل الـ Ping وأخذها مرة أخرى عند استقبال الرد ومن ثم طرح القيمة الأولى من الثانية...، ويتم كل ذلك كما يلي:

C#:

```
byte[] data = new byte[1024];
int recv, timestart, timestop;

Socket trace_socket = new
Socket(AddressFamily.InterNetwork,SocketType.Raw,
ProtocolType.Icmp);

IPHostEntry dest = Dns.Resolve(textBox2.Text);
IPEndPoint iep = new IPEndPoint(dest.AddressList[0], 0);
EndPoint ep = (EndPoint)iep;

ICMP packet = new ICMP();
packet.Type = 0x08;
packet.Code = 0x00;
packet.Checksum = 0;
Buffer.BlockCopy(BitConverter.GetBytes(1), 0, packet.Message, 0, 2);
Buffer.BlockCopy(BitConverter.GetBytes(1), 0, packet.Message, 2, 2);
data = Encoding.ASCII.GetBytes("Tracing is OK,FADI Abdel-qader");
Buffer.BlockCopy(data, 0, packet.Message, 4, data.Length);
```

```
packet.MessageSize = data.Length + 4;
int packetsize = packet.MessageSize + 4;
UInt16 chcksum = packet.getChecksum();
packet.Checksum = chcksum;

trace_socket.SetSocketOption(SocketOptionLevel.Socket,SocketOption
Name.ReceiveTimeout, 1000);
int badcount = 0;

for (int i = 1; i < 50; i++)
{
trace_socket.SetSocketOption(SocketOptionLevel.IP,SocketOptionNam
e.IpTimeToLive, i);
timestart = Environment.TickCount;
trace_socket.SendTo(packet.getBytes(), packetsize, SocketFlags.None,
iep);
        try
        {
        data = new byte[1024];
        recv = trace_socket.ReceiveFrom(data, ref ep);
        timestop = Environment.TickCount;
        ICMP response = new ICMP(data, recv);

        if (response.Type == 11)
listBox1.Items.Add("hop response from ms"+ i+ ep.ToString()+
(timestop-timestart));

        if (response.Type == 0)
        {
        listBox1.Items.Add(" reached in  hops, ms " + ep.ToString()+ i+
(timestop-timestart));
                break;
        }
        badcount = 0;
                }
catch (SocketException)
        {
        listBox1.Items.Add("hop No response from remote host "+ i);
        badcount++;
        if (badcount == 5)
                {
        listBox1.Items.Add("Unable to contact remote host");
                break;
                }
        }
        }
```

```
        trace_socket.Close();
```

VB.NET:

```
Dim data As Byte() = New Byte(1023)
Dim recv, timestart, timestop As Integer

Dim trace_socket As Socket = New
Socket(AddressFamily.InterNetwork, SocketType.Raw,
ProtocolType.Icmp)
Dim dest As IPHostEntry = Dns.Resolve(textBox2.Text)
Dim iep As IPEndPoint = New IPEndPoint(dest.AddressList(0), 0)
Dim ep As EndPoint = CType(iep, EndPoint)

Dim packet As ICMP = New ICMP
packet.Type = &H08
packet.Code = &H00
packet.Checksum = 0
Buffer.BlockCopy(BitConverter.GetBytes(1), 0, packet.Message, 0, 2)
Buffer.BlockCopy(BitConverter.GetBytes(1), 0, packet.Message, 2, 2)
data = Encoding.ASCII.GetBytes("Tracing is OK,FADI Abdel-qader")
Buffer.BlockCopy(data, 0, packet.Message, 4, data.Length)
packet.MessageSize = data.Length + 4

Dim packetsize As Integer = packet.MessageSize + 4
Dim chcksum As UInt16 = packet.getChecksum()
packet.Checksum = chcksum
trace_socket.SetSocketOption(SocketOptionLevel.Socket,SocketOption
Name.ReceiveTimeout, 1000)
Dim badcount As Integer = 0
For i As Integer = 1 To 49
trace_socket.SetSocketOption(SocketOptionLevel.IP,SocketOptionNam
e.IpTimeToLive, i)
timestart = Environment.TickCount
trace_socket.SendTo(packet.getBytes(), packetsize, SocketFlags.None,
iep)

Try
data = New Byte(1023)
recv = trace_socket.ReceiveFrom(data, ep)
timestop = Environment.TickCount
Dim response As ICMP = New ICMP(data, recv)
If response.Type = 11 Then
listBox1.Items.Add("hop response from ms" & i+ ep.ToString()+
(timestop-timestart))
End If
If response.Type = 0 Then
```

```
listBox1.Items.Add(" reached in  hops, ms " & ep.ToString()+ i+
(timestop-timestart))
        Exit Try
End If
badcount = 0
Catch e1 As SocketException
listBox1.Items.Add("hop No response from remote host " & i)
badcount += 1
If badcount = 5 Then
listBox1.Items.Add("Unable to contact remote host")
Exit For
End If
End Try
Next i
trace_socket.Close()
```

وهكذا بينا كيفية استخدام الـ ICMP في الدوت نيت وكيفية استخدامه لإنشاء برنامج لعمل الـPing وآخر لعمل الـRouting Tracing ، سوف نبين في الجزء التالي من هذا الفصل كيفية التعامل مع الـ ARP Protocol في الدوت نيت لجلب الـ MAC Address لأي Destination على الشبكة.

استخدام الـ ARP Protocol لجلب الـ MAC Address في برمجيات الشبكات :

يمكننا جلب الـ MAC Address لأي Host IP من خلال الأمر nbtstat من خلال الـCommand Prompt لاحظ الشكل التالي:

```
D:\Documents and Settings\fadi>nbtstat -a 10.0.0.10

Local Area Connection:
Node IpAddress: [10.0.0.10] Scope Id: []

           NetBIOS Remote Machine Name Table

       Name               Type         Status
    ---------------------------------------------
    FADI-CYBER     <00>  UNIQUE      Registered
    FMO            <00>  GROUP       Registered
    FADI-CYBER     <20>  UNIQUE      Registered
    FADI-CYBER     <03>  UNIQUE      Registered
    FMO            <1E>  GROUP       Registered

    MAC Address = 00-11-09-D0-13-21
```

ويمكننا الاستفادة من هذا الـ Command من داخل الدوت نيت باستخدام الـClass ProcessStartInfo والـClass Process الموجود ضمن الـSystem.Diagnostics Namespace ، حيث نمرر الـ Command والـ Arguments الذي نريد تنفيذه إلى الـProcessStartInfo Instance Object ثم ننفذ الأمر بتمرير الـinstance object السابق إلى الـ Process.Start Method ويتم ذلك كما يلي:

C#:

```
public string GetMac(string IP)
{
string str1=String.Empty;
try
```

```
        {
string str2=String.Empty;
ProcessStartInfo info1 = new ProcessStartInfo();
Process process1 = new Process();
info1.FileName = "nbtstat";
info1.RedirectStandardInput = false;
info1.RedirectStandardOutput = true;
info1.Arguments = "-A " + IP;
info1.UseShellExecute = false;
process1 = Process.Start(info1);
int num1 = -1;
        while (num1 <= -1)
                {
        num1 = str2.Trim().ToLower().IndexOf("mac address", 0);
        if (num1 > -1) { break; }
str2 = process1.StandardOutput.ReadLine();
                }
process1.WaitForExit();
str1 = str2.Trim();
                }
catch (Exception exception2)
                { throw exception2; }
                return str1;
                }
        }
```

VB.NET:

```
Public Function GetMac(ByVal IP As String) As String
   Dim str1 As String = String.Empty
   Try
      Dim str2 As String = String.Empty
      Dim info1 As ProcessStartInfo = New ProcessStartInfo
      Dim process1 As Process = New Process
      info1.FileName = "nbtstat"
      info1.RedirectStandardInput = False
      info1.RedirectStandardOutput = True
      info1.Arguments = "-A " & IP
      info1.UseShellExecute = False
      process1 = Process.Start(info1)
      Dim num1 As Integer = -1
      Do While num1 <= -1
         num1 = str2.Trim().ToLower().IndexOf("mac address", 0)
         If num1 > -1 Then
            Exit Do
         End If
         str2 = process1.StandardOutput.ReadLine()
      Loop
```

```
        process1.WaitForExit()
        str1 = str2.Trim()
    Catch exception2 As Exception
        Throw exception2
    End Try
    Return str1
End Function
```

ولتنفيذ الـ Method السابقة نمرر الـ IP Address (الخاص بالجهاز الذي نريد أن نجلب الـMAC Address له) إلى الـ Method السابقة حيث قمنا بفصل الـ MAC Address عن بقية المعلومات التي يخرجها لنا باستخدام الـ Trim Method ...

وهكذا بينا كيفية استخدام كل من البروتوكولين الـ ICMP والـ ARP في الدوت نيت ، سوف نبين في الجزء التالي من هذا الفصل مبدأ عمل برمجيات الـ Packet Sniffing وسنقوم من خلاله ببناء برنامج Sniffing بسيط.

ثالثا : Introduction to Packet Sniffing Applications

تكمن الفكرة الأساسية من الـ Packet Sniffing Applications بالتصنت على الـNetwork Layer أو الـRawSocket بعد اختيار الـNetwork Interface إذ أنها لا تعمل ضمن نطاق Port محدد ونستطيع من خلال الـ Raw Socket التقاط أي Packet يمر من خلال الـ Network Interface Card ولتحليل الـPacket لابد من معرفة الحجم الكلي لكل Header ثم إضافته ، وبشكل دائم يتم إضافة الـ Header Length إلى كل Header ضمن كل طبقة من طبقات الـ TCP/IP ، لاحظ موقع الـ Header Length في كل Header من بروتوكولات الـTCP/IP :

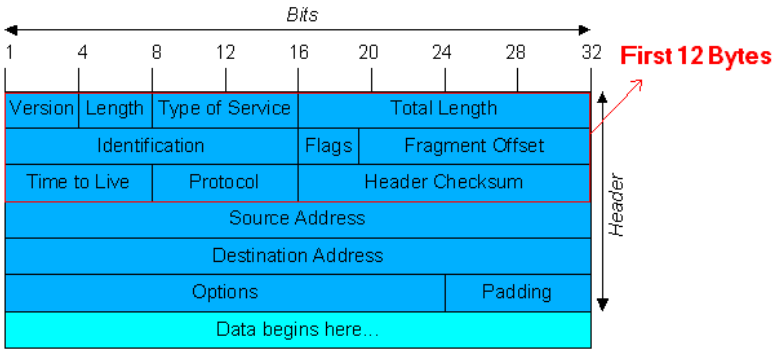
```
⊟ Frame 97 (214 bytes on wire, 214 bytes captured)
    Arrival Time: May 16, 2006 15:09:55.733567000
    [Time delta from previous packet: 0.000092000 seconds]
    [Time since reference or first frame: 4.790695000 seconds]
    Frame Number: 97
    Packet Length: 214 bytes
    Capture Length: 214 bytes
    [Protocols in frame: eth:ip:tcp:nbss:smb:dcerpc]
⊞ Ethernet II, Src: Micro-St_d0:13:21 (00:11:09:d0:13:21), Dst: CameoCom_a7:75:
⊟ Internet Protocol, Src: 10.0.0.10 (10.0.0.10), Dst: 10.0.0.111 (10.0.0.111)
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 200
    Identification: 0x3491 (13457)
  ⊞ Flags: 0x04 (Don't Fragment)
    Fragment offset: 0
    Time to live: 128
    Protocol: TCP (0x06)
  ⊞ Header checksum: 0xb126 [correct]
    Source: 10.0.0.10 (10.0.0.10)
    Destination: 10.0.0.111 (10.0.0.111)
⊟ Transmission Control Protocol, Src Port: 1170 (1170), Dst Port: microsoft-ds
    Source port: 1170 (1170)
    Destination port: microsoft-ds (445)
    Sequence number: 4896    (relative sequence number)
    [Next sequence number: 5056    (relative sequence number)]
    Acknowledgement number: 6062    (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x0018 (PSH, ACK)
    Window size: 65403
```

ويمكننا فصل كل Layer عن الأخرى بعد معرفة الـ Header Length لكل Layer ، لاحظ من الشكل السابق انه يمكننا معرفة الحجم الكلي للـ Packet من خلال الجزء الأول منه والمسمى بالـ Frame ، ويمكننا معرفة موقع الـData في الـ Packet من خلال المعادلة التالية:

Data in Application Layer =

$$\text{Total Length} - \sum \begin{matrix}(\text{IP Header Length} + \text{Options}) \\ (\text{Transport Protocol Header Length} + \text{Options})\end{matrix} \quad and$$

ويمكننا مثلا معرفة الـ IP Address لطرف المرسل مثلا من خلال تخزين الـ Packet الملتقط في Byte Array ثم تغيير الـ offset للـمكان الذي نريد القراءة منه ، إذ يأتي موقع الـIP Source Address في الـ IP Packet دائما بعد 12 Bytes لاحظ الشكل التالي:



ويجب دائما إنشاء Class أو Struct يحتوي على الـ Header Elements لكل Protocol يتم التعامل معه وحتى نستطيع تحليله لاحقا ، وتحديد سعة كل منها بناء على السعة الحقيقية لها وكمثال لإنشاء Struct خاص بالـIPHeader :

C#:

```
public struct IPHeader
        {
[FieldOffset(0)] public byte    ip_verlen; //IP version and IP Header
length
[FieldOffset(1)] public byte    ip_tos; //Type of Service
[FieldOffset(2)] public ushort  ip_totallength; //Total Packet Length
[FieldOffset(4)] public ushort  ip_id; //Unique ID
[FieldOffset(6)] public ushort  ip_offset; //Flags and Offset
[FieldOffset(8)] public byte    ip_ttl; //Time To Live
[FieldOffset(9)] public byte    ip_protocol; //Protocol (TCP, UDP,
ICMP, Etc.)
[FieldOffset(10)] public ushort ip_checksum; //IP Header Checksum
[FieldOffset(12)] public uint   ip_srcaddr; //Source IP Address
[FieldOffset(16)] public uint   ip_destaddr; //Destination IP Address
        }
```

VB.NET:

```
Public Structure IPHeader
   <FieldOffset(0)> _
   Public ip_verlen As Byte 'IP version and IP Header length
   <FieldOffset(1)> _
   Public ip_tos As Byte 'Type of Service
   <FieldOffset(2)> _
   Public ip_totallength As System.UInt16 'Total Packet Length
   <FieldOffset(4)> _
   Public ip_id As System.UInt16 'Unique ID
   <FieldOffset(6)> _
   Public ip_offset As System.UInt16 'Flags and Offset
   <FieldOffset(8)> _
   Public ip_ttl As Byte 'Time To Live
   <FieldOffset(9)> _
   Public ip_protocol As Byte 'Protocol (TCP, UDP, ICMP, Etc.)
   <FieldOffset(10)> _
   Public ip_checksum As System.UInt16 'IP Header Checksum
   <FieldOffset(12)> _
   Public ip_srcaddr As System.UInt32 'Source IP Address
   <FieldOffset(16)> _
   Public ip_destaddr As System.UInt32 'Destination IP Address
End Structure
```

ولتفعيل عملية التصنت على الـ IP Layer يجب تعريف Raw Socket ونحدد فيها الـ IP كـProtocol Type ثم نقوم بعمل الـ Binding مع الـ IP Address للـ Interface Card الذي نريد التصنت عليه ، وبتأكيد لا تحتاج الـ Raw Socket تحديد Port معين إذ أنها لا تتعامل مع أي من الـ Transport Layer Protocols ، ويتم ذلك كما يلي:

C#:

```
socket = new Socket(AddressFamily.InterNetwork, SocketType.Raw,
ProtocolType.IP);
socket.Bind(new IPEndPoint(IPAddress.Parse("10.0.0.10"), 0));
socket.BeginReceive(m_Buffer, 0, m_Buffer.Length,
SocketFlags.None, new AsyncCallback(this.OnReceive), null);
```

VB.NET:

```
socket = New Socket(AddressFamily.InterNetwork, SocketType.Raw,
ProtocolType.IP)
socket.Bind(New IPEndPoint(IPAddress.Parse("10.0.0.10"), 0))
socket.BeginReceive(m_Buffer, 0, m_Buffer.Length,
SocketFlags.None, New AsyncCallback(AddressOf Me.OnReceive),
Nothing)
```

ثم إضافة الـ AsyncCallback Method للـ BeginReceive Method السابقة ويتم ذلك كما يلي:

C#:

```
private void OnReceive(IAsyncResult ar)
```

```
{
int received = socket.EndReceive(ar);
if (socket != null)
        {
byte[] packet = new byte[received];
Array.Copy(Buffer, 0, packet, 0, received);
// Here The all Captured Packet in The Buffer
        }
socket.BeginReceive(Buffer, 0, Buffer.Length, SocketFlags.None, new
AsyncCallback(this.OnReceive), null);
}
```

VB.NET:

```
Private Sub OnReceive(ByVal ar As IasyncResult)
   Dim received As Integer = socket.EndReceive(ar)
   If Not socket Is Nothing Then
      Dim packet As Byte() = New Byte(received)
      Array.Copy(Buffer, 0, packet, 0, received)
      ‘ Here The all Captured Packet in The Buffer
   End If
   socket.BeginReceive(Buffer, 0, Buffer.Length, SocketFlags.None,
New AsyncCallback(AddressOf Me.OnReceive), Nothing)
End Sub
```

وفي هذه المرحلة فإن أي Packet يمر من خلال الـ Network Layer يتم التقاطه وتخزينه في الـ Buffer السابق، حيث يمكن تحليله لاحقا بحسب حجم الـ Header للـPacket الملتقط بالإضافة إلى معرفة الحجم لكل Element في الـ Header ...

وهكذا بينا كيفية التعامل مع الـ Raw Socket في الدوت نيت وكيفية استخدامها لبرمجة بروتوكولات الـICMP والـ Raw IP ...

سيتم الحديث في الجزء التالي عن برمجة بروتوكولات الـ Application Layer والـRemoting في بيئة الدوت نيت.

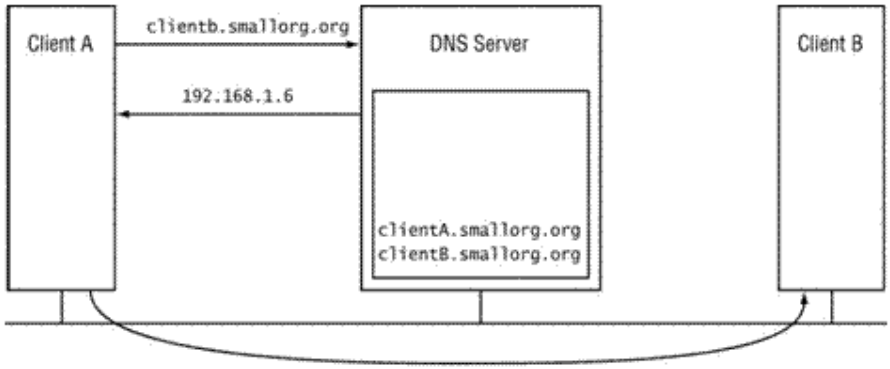
Part 4
Application Layer Programming

Chapter 12
DNS Programming

- Synchronous DNS Members
- Asynchronous DNS Members

12: DNS Programming:

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



في الدوت نيت يمكننا التعامل مع DNS باستخدام System.Net Namespace والتي تحتوي على جميع الـClasses و الـMethods الخاصة بـ DNS ، وتقسم دوال الـ DNS Class إلى قسمين متزامن Synchronous Methods و غير متزامن Asynchronous Methods وهي كما يلي:

أولا الدوال المتزامنة Synchronous Methods وهي :

GetHostName والتي تستخدم لجلب اسم الـ Host وترجع هذه الدالة قيمة String تحتوي على الـComputer Name ولا تأخذ هذه الدالة أي بارومترات ويمكن استخدامها كما يلي :

C#:
```
string hostname = Dns.GetHostName();
```

VB.NET:
```
Private hostname As String = Dns.GetHostName
```

الدالة GetHostByName و الدالة 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)
```

الدالة Resolve وهي Overloaded Method حيث ترجع Host Name إذا أرسلت لها IP Address وترجع Host Address إذا أرسلت لها Host Name في الـIPHostEntry ولا يختلف استخدامها عن استخدام الدوال 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");
Console.WriteLine(IPHost.HostName);

IPAddress[] addr = IPHost.AddressList;
                for(int i= 0; i < addr.Length ; i++)
                {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
```

ثانيا الدوال الغير المتزامنة Asynchronous Methods :

وتبدأ عادة بكلمة Begin أو End ومن الأمثلة عليها :

و EndGetHostByName و BeginResolve و BeginGetHostByName و EndResolve

طبيعة عملها كما هو الحال في الدوال المتزامنة لكنها تختلف بكون انه لا يشترط تنفيذها لإكمال عمل البرنامج في حين المتزامن لا تسمح الانتقال إلى الخطوة الثانية في البرنامج إلا في حالة انتهاء عملها وقد تسبب هذه السيئة بخفض الأدائية 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 الـ Host Name في البارومتر الأول و البارومتر الثاني يعرف فيه الـdelegate وتسمح لك بتمرير مدخلات إلى delegate Method ، ويستخدم End____ لإنهاء العملية في الـ Back Result كما يلي :

```
public static IPHostEntry EndResolve(IasyncResult ar)
```

وهنا مثال شامل و بسيط يقوم بجلب جميع الـIP's الموجودة على الشبكة حيث يعمل على جلب الـhost names من ProcessStartInfo من خلال الخاصية StandardOutput بتنفيذ الأمر Net View ، حيث يتم تحويله إلى host name من خلال الدالة GetMachineNamesFromProcessOutput ثم تخزينها في Collicaion ثم يتم تحويل الأسماء إلى عناوين من خلال الدالة Dns.Resolve .. طبعا يتم استخدام الـStreamReader لقراءة الـcollection الخاص بالـProcessStartInfo 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
```

وهكذا بينا في هذه الفصل أهمية الـDNS وطرق التعامل معه في بيئة الدوت نيت ، سيتم الحديث في الفصل التالي عن HTTP وطرق برمجته في بيئة الدوت نيت.

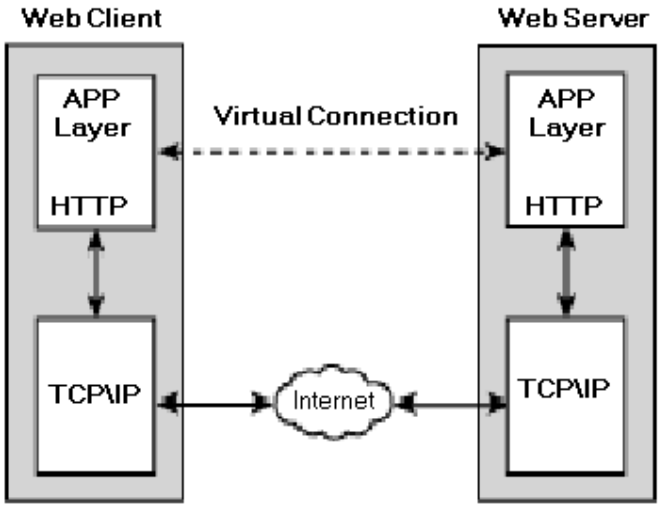
Chapter 13
HTTP Programming

- The Concept of HTTP Protocol
- Using HTTP in Dot Net
- Advanced HTTP Programming
- Using HttpWebRequest
- Using HttpWebResponse

HTTP – Hyper Text Transfer Protocol Programming :13

تتلخص وظيفة الـHTTP بشكل عام على انه البرتوكول المستخدم لتوصيل طلب المستخدم User Request إلى الويب Server ثم قيام الـweb server بالرد على الـRequest والذي يسمى بـ Server Response وبتأكيد تستطيع نقل جميع أشكال (Multimedia) من النص وصورة و صوت و فيديو وغيره .. من الـWeb Server إلى الـClient Application كـByte object.

يعمل برتوكول الـHTTP على الـApplication 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 الـWeb Server في الدوت نيت باستخدام الـWebClient Class الموجود في System.Net Namespace إذ تقدم لنا جميع الإمكانيات لتوصيل طلب الزبون و الرد عليهUser Request & Server Response وتدعم الـWebClient Class ثلاثة Methods لتحميل البيانات من الـWeb Server وهي:

1- **DownloadData** ووظيفتها جلب البيانات من الـWeb 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
```

DownloadFile-2 ووظيفتها نقل ملف ما من الـWeb 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 والـServer لجلب بيانات من URL محدد وتخزينه في Stream Object بعد تمرير الـURL للموقع الذي تريد عرضه وباستخدام الدالة ReadLine نستطيع عرض البيانات المخزنة في الـ Stream Object على شكل HTML Code .

ملاحظة : تستخدم الدالة Peek لمعرفة نهاية الـStream 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 الـ**WebClient Class** على مجموعة من الـ**Properties** والتي تستخدم لجلب معلومات عن الـWeb Host مثل ResponseHeaders property والذي يستخدم لجلب معلومات هامة عن الـweb host مثل عدد الـHeaders ونوع الـcontrol cash واسم الـServer و نوع الـEncoding 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 الـWebClient أربعة Methods لتحميل البيانات إلى الـWeb Server وهي :

1- **OpenWrite** ويستخدم لإرسالـStream Data إلى الـWeb Server وذلك بعد تمرير عنوان الـURL للملف والنص الذي نريد كتابته على الـWeb 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 الـWeb Server وهذا يعني انك تستطيع من خلالها رفع أي نوع من البيانات مثل النص الصور الفيديو وغيره إلى الـweb 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 الدالة لرفع ملف من الـLocal Computer إلى الـ 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 والـvalues الخاصة بها إلى الويب Server وذلك بعد تحويل الـ**Collection** إلى Byte Array ولتعريف **Collection** نستخدم الكلاس NameValueCollection الموجود في Namespace System.Collections.Specialized وبعد تعريفه نستخدم الدالة add لإضافة الـCollection 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 الـHTTP Programming:

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

1- استخدام HttpWebRequest Class :

يحتوي هذا الـClass على مجموعة من الـProperties والتي تستخدم بشكل أساسي في تطبيقات الـWeb Client Applications لإنشاء مثل :

1- استخدام خاصية الـWeb Proxy : والتي نمرر فيها عنوان الـProxy Server ورقم الـPort حتى نستطيع التعامل مع الـHTTP Web Requestsمن خلف Proxy Server أو Firewall ويتم تعريف الـProxy 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 الـHttpWebRequest Object ثم نعرف WebProxy Object من الـwebProxy Class ونسند له عنوان الـProxy Server ورقم الـ Port وبعد ذلك نستطيع إسناده إلى أي Object باستخدام الخاصية Proxy التي تكون موجودة عادة في جميع HttpWebRequest Objects .

2- استخدام الـHttpWebrequest لإرسال بيانات إلى الويب Server باستخدام الـStreams 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 الـObject HttpWebRequest نقوم بتعريف Object Stream ونسند له الـRequest Stream من خلال الدالة GetRequestStream .

2 - استخدام HttpWebResponse Class:

تستخدم الـ**HttpWebResponse Object** لإرجاع بيانات من الويب Server إلى الـClient حيث نستخدم الدالة **GetResponse** و الدالة **BeginGetResponse** لهذه العملية ولا يوجد فرق في وظيفة هذه الـMethod سوى أن **BeginGetResponse** تعتبر **asynchronous Method** .

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

1- **CharacterSet** : وتستخدم لتحديد نوع الـCharacter Set
2- **ContentEncoding** : وتستخدم لعملية الـencoding
3- **ContentLength** : وتستخدم لمعرفة حجم الرد
4- **ContentType** : لتحديد نوع الـResponse
5- **Cookies** : لتعامل مع الـCookies واستخدامها 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 الـHTTP 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 الـHTTP Headers

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

8- **Method** : لمعرفة الدالة والتي تستخدم في الـHTTP Response

9 – **ProtocolVersion** : لمعرفة الـHTTP Version

10 – **ResponseUri** : الـURL الخاص بـServer

11 – **Server** : لمعرفة اسم الـServer

12 – **StatusCode** : لمعرفة نوع الـCoding المستخدم

13 – **StatusDescription** : لإرجاع Text يحتوي على حالة الـHTTP

بينا في هذا الفصل كيفية برمجة الـ HTTP في بيئة الدوت نيت وطرق التعامل مع الـ HttpWebRequest والـHttpWebResponse في بيئة الدوت نيت ، سوف نتحدث في الفصل التالي عن Web Services والـXML وطرق التعامل معه في بيئة الدوت نيت.

Chapter 14
Web Services & XML Programming

-Introduction to Web services & XML

-Create A Simple Web Service Application

-Using ADO.NET With Web Services

- Create a Simple GIS System to Send Information About Countries Via SOAP & XML to Clients.

Web Services Programming :14

تحدثنا في الفصل السابق عن برمجة الـHTTP وبينا فيه كيفية التفاعل بين الـweb server والـclient ويعتبر هذا الفصل مكمل لما تحدثنا عنه في الفصل السابق.
تتلخص وظيفة الـweb services بإمكانية الاستفادة من الـMethods الموجودة بالـ web server داخل برنامج الزبون، وباستخدام برتوكول الـ SOAP وهو اختصار لـ Simple Object Access Protocol يتم نقل الـResult من الـweb Services server إلى الـClient بعد تحويلها إلى الـXML - extensible Markup Language حيث تنقل عبر برتوكول الـHTTP إلى جهاز الزبون والهدف من استخدامه هو تسهيل وصول الـData من الـweb server إلى الـClient من خلال الـ firewalls والبيئات المختلفة إذ أن جميع بيئات الشبكات تدعم برتوكول الـHTTP والذي يعمل على الـPort 80 . ولا تختلف لغة الـXML عن الـHTML إذ تستخدم نفس القواعد في الـHTML وهي مجموعة من الـ Elements والـAttributes مثل الـ</> <> لكن تتميز بمرونة اكبر وكمثال عليها :

```
<MyStuff>
<myName>FADI Abdel-qader</myName>
<myTelephone>+962796...</myTelephone>
<myEmail>fadi822000@yahoo.com</myEmail>
<myAge>25</myAge>
<myGender>M</myGender>
</myStuff>
```

ويمكن نقل أي شيء نريده باستخدام الـXML سواء Text,Binary Data, Serialization Object أو أي شيء اخر وما ميز الـ XML عن غيرها هو قدرتها العالية على نقل جميع انواع البيانات بجميع البيئات...

ويمكننا التعامل مع الـ XML في الدوت نيت باستخدام الـ System.xml Namespaces وهذا مثال يوضح كيفية قراءة ملف الـ XML السابق باستخدام الـXmlDocument :

C#:

```
using System.Xml;
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 Gender = xDoc.GetElementsByTagName("myGender");
MessageBox.Show(
"Name: " + name[0].InnerText +"\n"+
"Telephone: " + telephone[0].InnerText +"\n"+
"Email: "+ email[0].InnerText +"\n"+
"Age: "+ age[0].InnerText +"\n"+
"Gender: "+ Gender[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 Gender As XmlNodeList =
xDoc.GetElementsByTagName("myGender")
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)
& "" + "Gender: " + Gender(0).InnerText + "" &
Microsoft.VisualBasic.Chr(10) & "")
```

ويمكن استخدام الـ XmlTextWriter Class لخزين بيانات على هيئة XML Data وكما في المثال التالي:

C#:

```
private void btnSave_Click(object sender, System.EventArgs e)
                {
        try
                        {
XmlTextWriter XMLWriter = new XmlTextWriter
("AddressBook.XML", null);
XMLWriter.Formatting = Formatting.Indented;
```

```
XMLWriter.WriteComment("العناوين ملف");
XMLWriter.WriteStartElement("AddressBook");

        XMLWriter.WriteStartElement("Name");
        XMLWriter.WriteString(tx_Name.Text);
        XMLWriter.WriteEndElement();

        XMLWriter.WriteStartElement("Address");
        XMLWriter.WriteString(tx_Address.Text);
        XMLWriter.WriteEndElement();

XMLWriter.WriteEndElement();
XMLWriter.Flush();
XMLWriter.Close();
MessageBox.Show("تم");
}
catch(Exception ex)
        {
MessageBox.Show(ex.Message);
        }
}

private void btnViewXML_Click(object sender, System.EventArgs e)
{
try
{
XmlDocument doc = new XmlDocument();

doc.PreserveWhitespace = true;
doc.Load("AddressBook.xml");
MessageBox.Show(doc.InnerXml,"محتويات الملف");
}
catch(Exception ex)
        {
MessageBox.Show(ex.Message);
        }
}
```

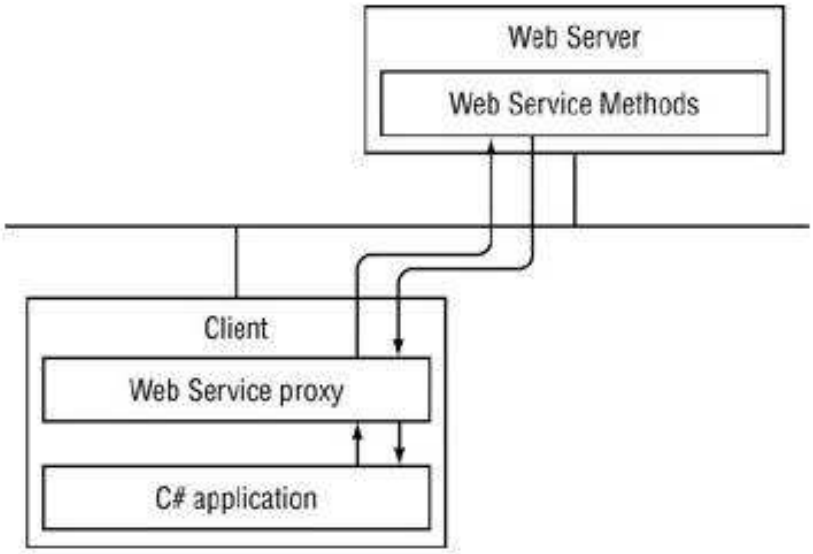
VB.NET:

```
Private Sub btnSave_Click(ByVal sender As Object, ByVal e As
System.EventArgs)
   Try
  Dim XMLWriter As XmlTextWriter = New
XmlTextWriter("AddressBook.XML", Nothing)
  XMLWriter.Formatting = Formatting.Indented
```

```
XMLWriter.WriteComment("العناوين ملف")
XMLWriter.WriteStartElement("AddressBook")

XMLWriter.WriteStartElement("Name")
XMLWriter.WriteString(tx_Name.Text)
XMLWriter.WriteEndElement()

XMLWriter.WriteStartElement("Address")
XMLWriter.WriteString(tx_Address.Text)
XMLWriter.WriteEndElement()



XMLWriter.WriteEndElement()
XMLWriter.Flush()
XMLWriter.Close()
MessageBox.Show("تم")
   Catch ex As Exception
      MessageBox.Show(ex.Message)
   End Try
End Sub

Private Sub btnViewXML_Click(ByVal sender As Object, ByVal e As
System.EventArgs)
   Try
      Dim doc As XmlDocument = New XmlDocument
      doc.PreserveWhitespace = True
      doc.Load("AddressBook.xml")
      MessageBox.Show(doc.InnerXml, "الملف محتويات")
   Catch ex As Exception
      MessageBox.Show(ex.Message)
   End Try
End Sub
```

تمر عملية استخدام الـweb services بثلاثة مراحل وهي :

1- The web service server : والذي يتم من خلاله إرسال واستقبال البيانات عبر برتوكول الـSOAP باستخدام الـIIS والـASP.NET .
2- The proxy object : والذي يسمح للـ Client بإرسال و استقبال البيانات من وإلى الـweb Services Server حيث يتم تعريفه في الـHttpWebRequest من خلال الكلاس WebProxy وهو ما بينته في الجزء السابق.
3- The client application : وهو الواجهة الخاصة بزبون والتي يتم ربطها بالـ Web Services Server كما في الشكل التالي :



: Create a Simple Web Services Application :13.2

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

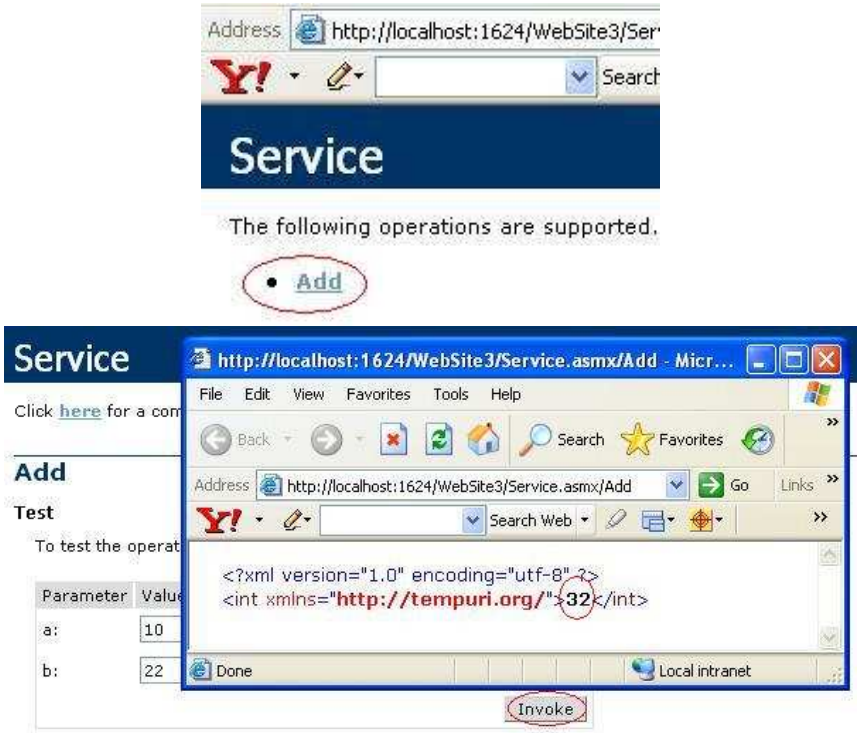
C#:

```
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 الـClient على شكل XML باستخدام برتوكول الـSOAP وكما يظهر في الشكل التالي :



ولإنشاء برنامج الـClient يجب أولا تحويل الكلاس السابق إلى File Dll و إرفاقه بالـClient Resources ويتم استخدامه كما يلي :

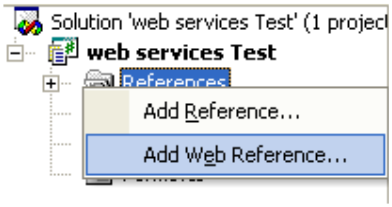
C#:

```
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);
   }
}
}
```

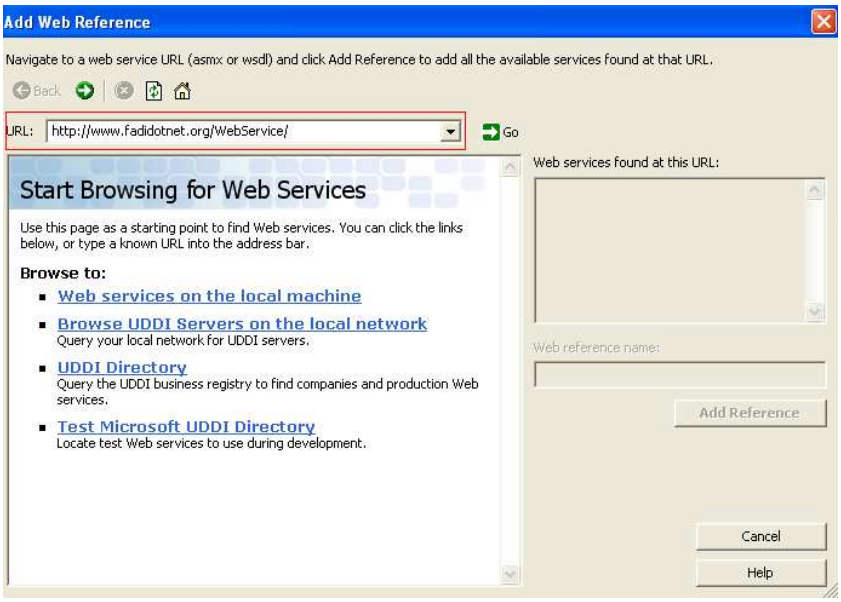
VB.NET:.

```
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
```

كما يمكن إنشاء Connection Object بين الـ Client والـ Web Services Server بشكل مباشر ويتم ذلك بإضافة Web Services Reference إلى المشروع وكما في الشكل التالي:



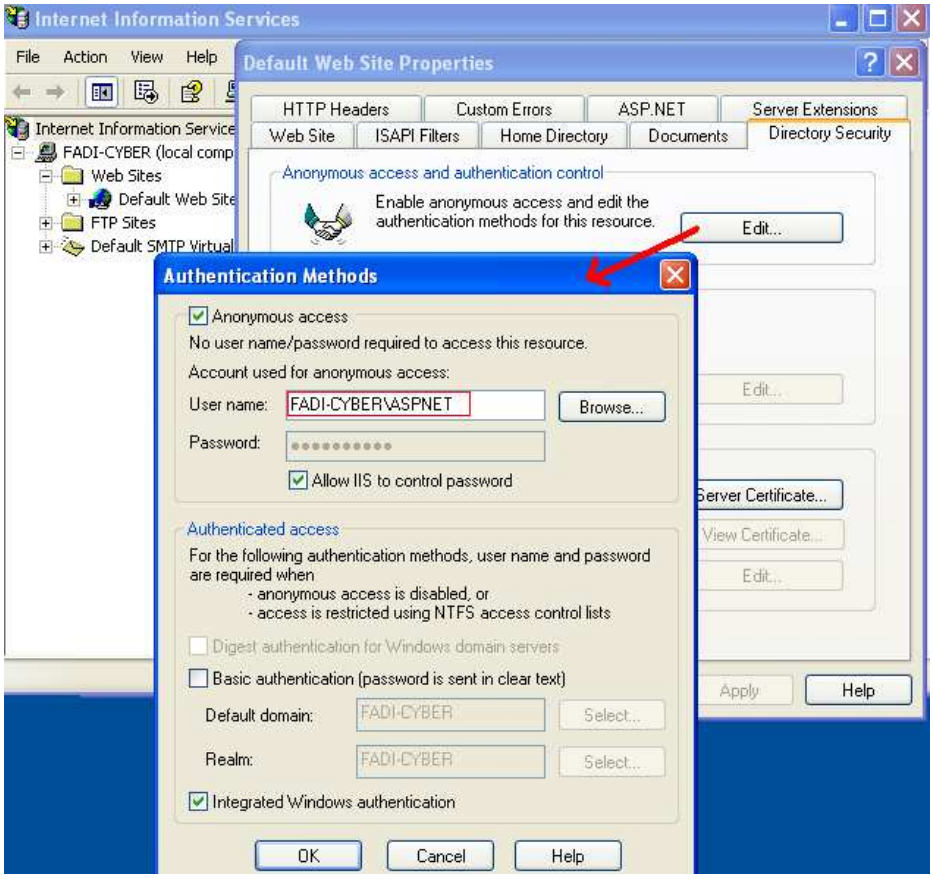
حيث يتم الاختيار سواء كانت الـ Web Services على نفس الجهاز أو على جهاز بعيد وذلك بوضع الـ URL للموقع وكما في الشكل التالي:



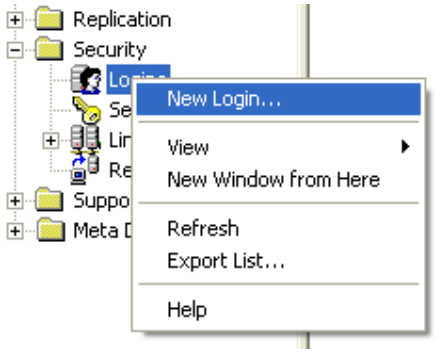
عندها نستطيع استدعاء الـ Methods السابقة في البرنامج كما وإنها على نفس الجهاز ، سنبين في الجزء التالي من هذا الفصل كيفية ربط الـ ADO.NET مع الـ Web Services لإرسال الـ Result كـ XML إلى الـClient.

Using ADO.NET With Web Services :13.3

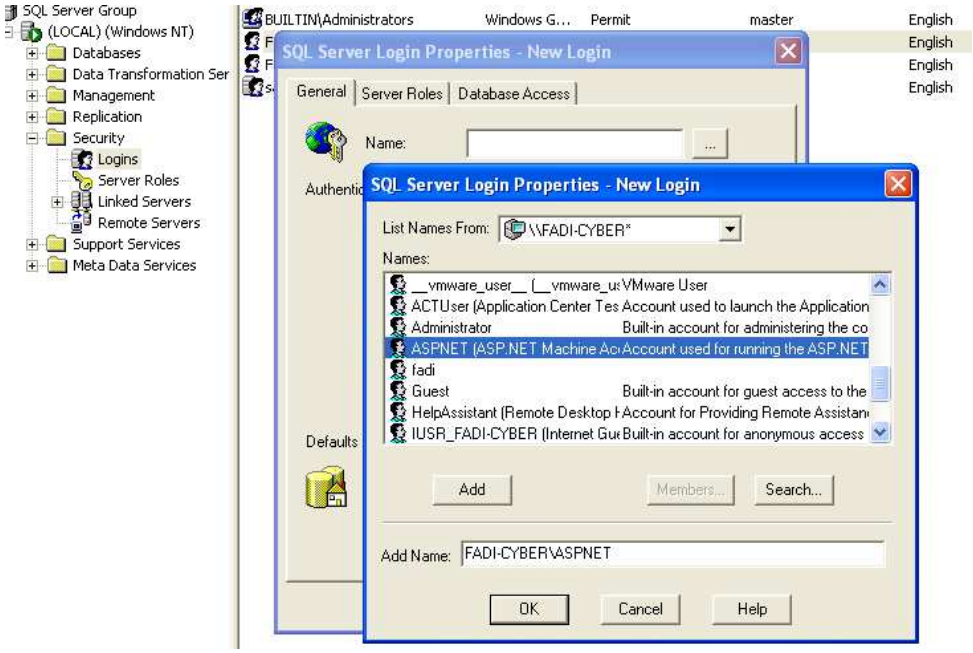
استخدمنا في المثال السابق الـ Web Services لإرسال بيانات إلى الـ Client مستخدما بروتوكول الـ SOAP ، وفي هذا المثال سنقوم بإنشاء Web Services لنقل بيانات من الـ SQL Server Database إلى الـ Client ويمكن أن تكون هذه البيانات عبارة عن Text أو Image أو غيره ... وللبدء سنقوم أولا بتأكد من إعدادات الـAuthentication في الـ IIS والـ SQL Server ، ويجب أن تكون إعدادات الـ Authentication في الـIIS كما في الشكل التالي:



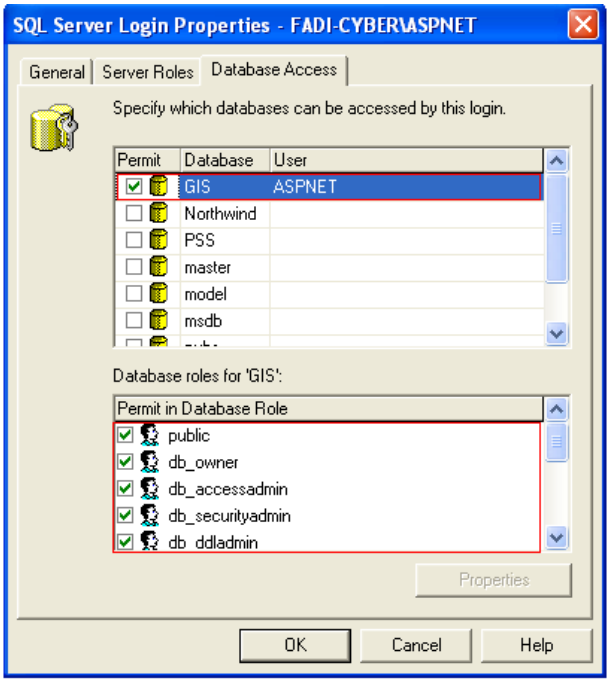
حيث نحدد User معين ونعطي له صلاحيات معينة لدخول على قاعدة البيانات والموجودة على الـ SQL Server ، ونعطي له هذه الصلاحيات من الـ Security ضمن الـEnterprise Manager الخاص بالـ Microsoft SQL Server وكما في الشكل التالي:



ثم ندخل اسم المستخدم الذي حددناه في الـ IIS ضمن الـ Users المسموح لهم بالدخول على قاعدة البيانات وكما في الشكل التالي:



ولتحديد الصلاحيات نختار قاعدة البيانات التي نريد استخدامها ونحدد صلاحيات الـ Database Access والـServer Roles من الـ SQL Server Login Properties وكما في الشكل التالي:



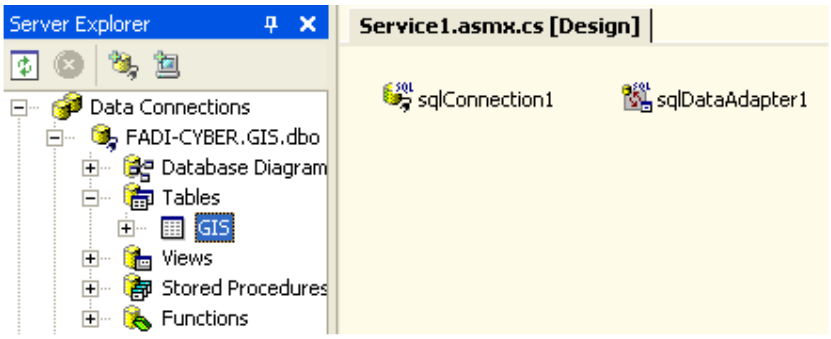
وبتأكيد يمكنك تحديد الصلاحيات التي تريدها والتي تتناسب مع البرنامج الذي تريد إنشائه.

وللبدء سنقوم بعمل فكرة بسيطة لإنشاء نظام GIS بسيط يرسل الـ Client اسم البلد الذي يختاره من الخارطة إلى الـWeb Services Server ويرجع الـ Web Services معلومات

عن هذا البلد من قاعدة البيانات إلى الـ Client وسوف نقوم في البداية بعمل Table في قاعدة البيانات وتحتوي على أسم البلد و معلومات عن هذا البلد وكما في الشكل التالي:

	Column Name	Data Type	Length	Allow Nulls
	Country	varchar	50	✓
	Info	varchar	50	✓

ثم سنقوم بربط قاعدة البيانات مع الـ Web Services بالسحب والإفلات من الـ Server Explorer وكما في الشكل التالي:



سيرسل الـ Client اسم البلد الذي يقوم بضغط عليه من الخريطة إلى الـ Web Services وفي هذه الحالة نحن بحاجة إلى إنشاء Method تأخذ باروميتر واحد وترجع String Return Value تحتوي على معلومات عن هذا البلد وكما يلي:

C#:

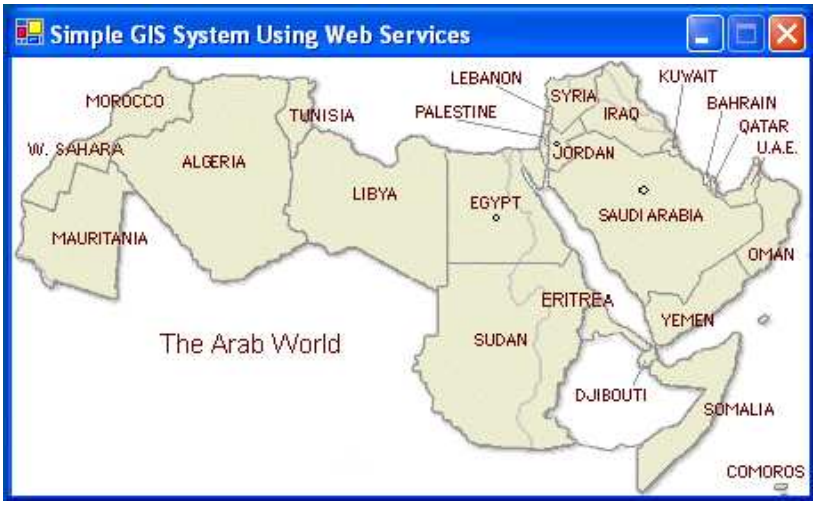
```
[WebMethod]
public string GIS_Country(string country)
{
try
{
sqlDataAdapter1.SelectCommand.CommandText = "select * from GIS
where Country='"+ country+"'";
DataSet ds = new DataSet ();
sqlDataAdapter1.Fill (ds);
return ds.Tables[0].Rows[0][1].ToString ();
}

catch (Exception){return "I Can not Find Your Country! Please Select
Another One";}
}
```

VB.NET:

```
<WebMethod()> _
Public Function GIS_Country(ByVal country As String) As String
    Try
        sqlDataAdapter1.SelectCommand.CommandText = "select * from
GIS where Country='" & country & "'"
```

```
        Dim ds As DataSet = New DataSet
        sqlDataAdapter1.Fill(ds)
        Return ds.Tables(0).Rows(0)(1).ToString()
    Catch e1 As Exception
        Return "I Can not Find Your Country! Please Select Another One"
    End Try
End Function
```

في طرف الـ Client سنقوم بربطه مع الـ Web Service التي تم إنشائها وكما في المثال السابق وسيمرر الـClient اسم الدولة إلى الـ Method Web Services ويرجع لنا معلومات عن الدولة، وسيكون الشكل العام للـClient كما يلي:



نستطيع تحديد حدود كل بلد بعمل Areas حيث عند الضغط عليها يرسل اسم البلد إلى الـ Web Services ، ولكي لا ندخل بتعقيدات الـ GDI سنحل الموضوع بإدراج Panel على كل بلد ونجعل لون الخلفية له من النوع الشفاف الـTransparency ، وسيكون الكود التالي عند حدث الـ Mouse Down على الـ Panel الموضوعة على كل بلد وكما يلي:

C#:

```
private void panel1_MouseDown(object sender,
System.Windows.Forms.MouseEventArgs e)
{
localhost.Service1 sr = new GIS.localhost.Service1 ();
this.Text =  sr.GIS_Country ("Jordan");
}
```

VB.NET

```
Private Sub panel1_MouseDown(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs)
   Dim sr As localhost.Service1 = New GIS.localhost.Service1
   Me.Text = sr.GIS_Country("Jordan")
End Sub
```

ويمكن استخدام الـ Web Services لإرجاع Binary Data كصورة مثلا بعد إحضارها من قاعدة البيانات وهو ما تم شرحه في الفصل الخاص بالـ Streaming ، ويمكن تطبيق كل هذه المفاهيم على الـ ASP.NET وبنفس الطريقة، سيتم الحديث في الفصل التالي عن الـ Remoting وتطبيقاتها في الدوت نيت.

Chapter 15
Remoting & Distributed Systems Programming & Design

- The Distributed Systems Concept & Design
    - Design a Distributed eCommerce System by Using ASP.NET and Web Services.

- Serialization Programming
    - The Serialization (Classes & Members)
    - Using BinaryFormatter & SoapFormatter to Serialize Objects & Images Through Network

- Remoting Programming
    - Remoting (Classes & Members)
    - Using Remoting Applications in Dot Net
    - Create an Advanced Distributed eLearning System (Remote Class Room)
    - Create an Advanced Remote Desktop Application With Remote (Mouse/Keyboard) Control Features

15 ، أولا - The Distributed Systems Concept & Design :

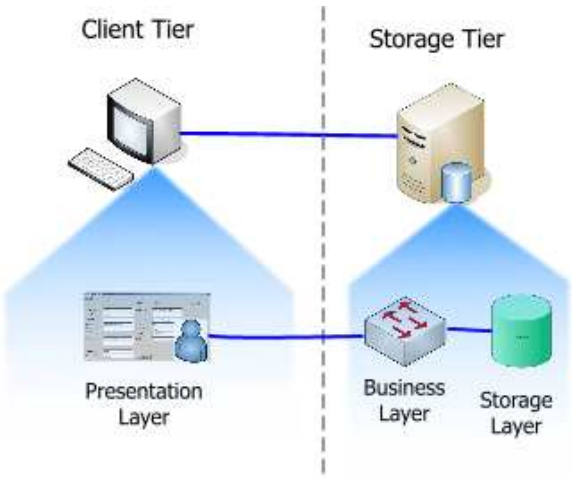
يعرف النظام الموزع على انه مجموعة من الأجهزة التي تعمل مع بعضها كنظام واحد بمعنى توزيع مهام البرنامج على مجموعة من الأجهزة بحيث تعمل بشكل متوازي وهو ما يسمى بالـLoad Balance حيث تكون هنالك نسخة من البرنامج على كل Server وتعمل مع بعضها على أساس توزيع الحمل أو بشكل متسلسل بحيث تعمل مع بعضها على أساس توزيع المهام ، ويفضل في برمجيات الأعمال Business Applications تقسيم المهام التالية كل منها على Server منفصل:

1- واجهة المستخدم أو الـ Presentation وهي الواجهة التي يتعامل معها الزبون بشكل مباشر.

2- الجزء الذي له علاقة بطبيعة برنامج الأعمالBusiness Tier وجميع القواعد التي لها علاقة بالعمل مثل طريقة حساب الخصومات Discounts أو طريقة الدفع أو طريقة توصيل السلعة لزبون أو ما شابه.

3- الجزء المخصص لحفظ البيانات Data Source وهو الجزء الذي يحتوي على قواعد البيانات والتي تحفظ عليها معلومات الزبائن أو حساباتهم وكل ما له علاقة بذلك.

ويمكن أن يتم ربط كل من هذه الأجزاء مع بعضها البعض مستخدما الـ(Web Services) أو استخدامها كـ DLL Reference وهو ما بينته في الفصل السابق Web Services Programming ، وفي كلا الحالتين يجب استخدام أسلوب الـ Remoting لربط بين الـServers المتعددة ، وسوف نبين في الجزء التالي من هذا الفصل كيفية الربط باستخدام الـRemoting وكيفية الاستفادة من الـ Serialization لاستخدام مكتبات الـ Dll والتي قد تكون موجودة على جهاز بعيد لاحظ الشكل التالي:



تعتبر الـ Web Services من الأمور الهامة وخاصة ببرمجة النظم الموزعة حيث نقوم بربط أجزاء البرنامج مع بعضها البعض وكمثال سنقوم بتطبيق نظام لشركة تصنيع العاب الحاسوب تستخدم أسلوب البيع الإلكتروني (التجارة الإلكترونية) ، حيث سيقوم الزبون بالدخول على الـWeb Site الخاص بالشركة والمبني على الـ ASP.NET ويحتوي هذا الموقع على المنتجات الخاصة بالشركة والأسعار لكل منتج وتعتمد الشركة بنك الـ ABC لدفع الإلكتروني، و الـ DHL لتوصيل المنتج إلى البلد الذي يقيم فيه الزبون، كما وتقدم الشركة خصومات بنسبة 20 % لزبائن المقيمين في الأردن و فلسطين والعراق وسوريا وخصومات تصل إلى 50 % لزبائن الذين يشترون 3 منتجات أو أكثر من منتجات الشركة.

و قبل البدء ببناء النظام سنقوم بتقسيم هذه المهام على ثلاثة أجزاء وكما يلي:
1- واجهة المستخدم الـ Presentation ويمكن أن تكون موقع الكتروني مبني على الـASP.NET يحتوي على صور المنتجات وشرح مختصر عن كل منتج بالإضافة إلى سلة المشتريات الـ Shopping Cart والتي يجمع فيها الزبون مشترياته من الموقع والقيمة الإجمالية لها.
2- الـ Business Part وهو مقسم هنا على ثلاثة أجزاء (البنك،نظام الخصومات أو الضرائب ،نظام التوصيل DHL كمثال)
3- الـ Data Source Part وهو هنا قاعدة البيانات الخاصة بالشركة والتي تقسم إلى جزأين (جزء معلومات الزبائن ، جزء العروض والأسعار).

تنويه: جميع المعلومات المتعلقة بالشركات المذكورة هي معلومات وهمية وأسماء الشركات والبنوك الموضوعة في المثال هي أسماء مسجلة لأصحابها ولا تعبر المعلومات الموجودة عن الطريقة المتبعة وإنما تم تبسيطها قدر الإمكان لغايات التعلم.

سيعتمد أسلوب الربط بين هذه الأنظمة على الـ Web Services حيث سيوفر بنك الـ ABC الـ Web Service الخاصة بأسلوب الدفع الإلكتروني وستوفر الـ DHL الـ Web Service الخاصة بتوصيل المنتج وسيتم ربط هذه الـ Web Services بالموقع الإلكتروني وسنفترض أن البنك سيحتاج المعلومات التالية لإتمام عملية الدفع:

Data Type	Field Name
String	اسم الزبون (Full Name)
String	العنوان (Full Address) + رقم الهاتف والفاكس ...
Integer	رمز الدولة و المدينة (Zip Code)
String	رقم بطاقة الائتمان سواء كان MasterCard, VISA, Discover , American Express أو غيره
Float	المبلغ المودع
String	رقم الحساب الخاص بالشركة

وستكون الـ Method بالـ Web Services الخاصة بالبنك كما يلي كمثال:

C#

```
[WebMethod]
public bool Bank_Tier(string Customer_Name,string Address,int
zip_code,string payment_method,string ecard_number,float
depositor,string acount_number)
{
// Check if the eCard number is correct
// Check if the Account is Available
// Do The Transaction
// Return if the Transaction Done Successfully or not
}
```

VB.NET

```
<WebMethod()> _
Public Function Bank_Tier(ByVal Customer_Name As String, ByVal
Address As String, ByVal zip_code As Integer, ByVal payment_method
As String, ByVal ecard_number As String, ByVal depositor As Single,
ByVal acount_number As String) As Boolean
   ' Check if the eCard number is correct
   ' Check if the Account is Available
   ' Do The Transaction
   ' Return if the Transaction Done Successfully or not
End Function
```

سنفترض أن الشركة قد تم تسجيلها في الـ DHL Customers Table حيث يملك الـ DHL المعلومات الكاملة عن الشركة (مثل رقم الحساب وعنوانها وما شابه) ...الآن سيطلب الـDHL هذه المعلومات لتوصيل السلعة إلى الزبون:

Data Type	Field Name
String	اسم الزبون (Full Name)
String	العنوان (Full Address) + رقم الهاتف والفاكس ...
Integer	رمز الدولة و المدينة (Zip Code)
String	اسم الشركة التي تم الشراء منها
String	البريد الإلكتروني Email (Primary Kay)
Float	صافي المبلغ المطلوب (المحول) *(سيتم معرفته من method أخرى تحسب القيمة المطلوبة لتوصيل السلعة إلى البلد الذي يقيم فيه الزبون)*

وستكون الـ Method بالـ Web Services الخاصة بالـ DHL كما يلي كمثال:

C#:

```
[WebMethod]
public string DHL_Tier(string Customer_Name,string Address,int
zip_code,string company_name,string email,float cost, int quantity)
        {
// Check the Company Account
// Store all information about the Customer
// Do The Transaction
// if the Transaction Done Successfully
// it Return The Number of Days that Need to Deliver the Product to
Customer
        }
```

VB.NET

```
<WebMethod()> _
Public Function DHL_Tier(ByVal Customer_Name As String, ByVal
Address As String, ByVal zip_code As Integer, ByVal company_name
As String, ByVal email As String, ByVal cost As Single, ByVal
quantity As Integer) As String
   ' Check the Company Account
   ' Store all information about the Customer
   ' Do The Transaction
   ' if the Transaction Done Successfully
   ' it Return The Number of Days that Need to Deliver the Product to
Customer
End Function
```

لذلك عندما يريد أي زبون التسجيل في الموقع سيتطلب منه إدخال المعلومات التالية:

Data Type	Field Name
String	اسم الزبون (Full Name)
String	كلمة المرور (Password)
String	البريد الإلكتروني Email (Primary Kay)
Integer	العنوان (Full Address) + رقم الهاتف والفاكس ...
String	رمز الدولة و المدينة (Zip Code)

وتسجل جميع عمليات الشراء (السلع التي تم شرائها) وغير المشترى بعد (في عربة التسوق shopping Cart) على قاعدة بيانات موجودة في الـ Storage Tier وستحتوي على الحقول التالية:

Data Type	Field Name
String	البريد الإلكتروني Email (Foreign Kay)
String أو Barcode	البار كود لسلعة (Foreign Kay)
Integer	العدد المطلوب
Date/Time	تاريخ ووقت الطلب
Date/Time	تاريخ ووقت التسليم
Float	صافي المبلغ المطلوب
String	طريقة الدفع (Check or Bank Transfer..)
String	طريقة التسليم (DHL For Example)
Boolean (True/False)	هل تم الدفع أم لا (Statues)

وسنفترض وجود Table آخر يحتوي على السلع المتوفرة والكميات بالإضافة إلى صورة السلعة:

Data Type	Field Name
String أو Barcode	البار كود لسلعة (Primary Kay)
String	اسم السلعة
Integer	العدد المتوفر
Binary	صورة السلعة

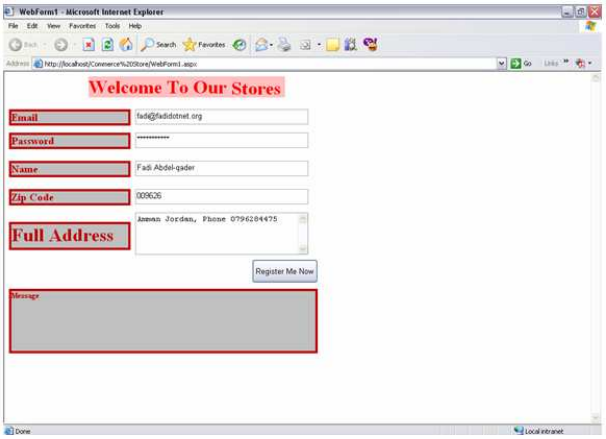
وقاعدة بيانات أخرى لحساب الضريبة Tax حيث تحتوي على رمز الدولة والضريبة لكل دولة:

Data Type	Field Name
Integer	رمز الدولة و المدينة (Zip Code)
Float	الضريبة Tax

وجدول آخر يحتوي على اسم الدولة والخصم Discount الممنوح لكل دولة:

Data Type	Field Name
Integer	رمز الدولة و المدينة (Zip Code)
Float	الخصم discount

والآن سنحتاج إلى بناء قاعدة البيانات التي تحتوي على معلومات العملاء وربطها بصفحة ASP.NET وكما في الشكل التالي:



بعد الضغط على الـ Register Me Now يجب أن يتم أولا التحقق من صحة البيانات المدخلة ثم إدخال المعلومات إلى قاعدة البيانات :

C#:

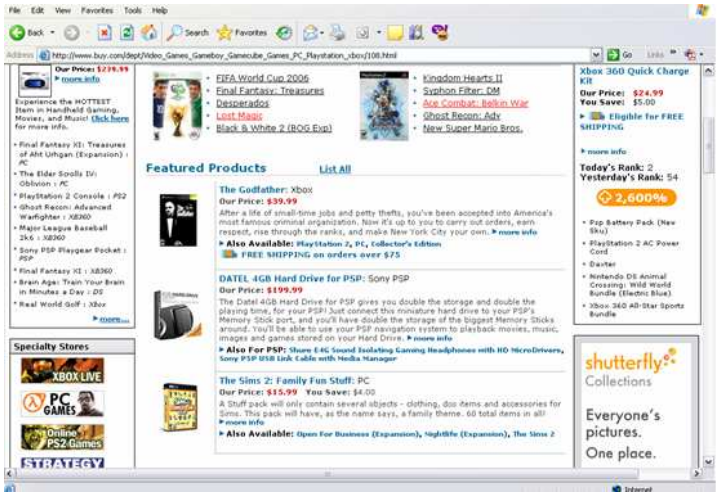
```
try
{
// Check if All Information is Correct if Yes do:
sqlConnection1.Open();
sqlCommand1.CommandText = "insert into Clients_info
values('"+Email_TextBox.Text+"','"+Pass_TextBox.Text+"','"+Name_T
extBox.Text+"',"+Zipcode_TextBox.Text +
",'"+address_TextBox.Text+"')";
```

```
sqlCommand1.ExecuteNonQuery();
Response.Redirect("http://www.fadidotnet.org/welcome.aspx");
}
catch(Exception ex){msg_txt.Text = ex.Message;}
finally{sqlConnection1.Close ();
}
```

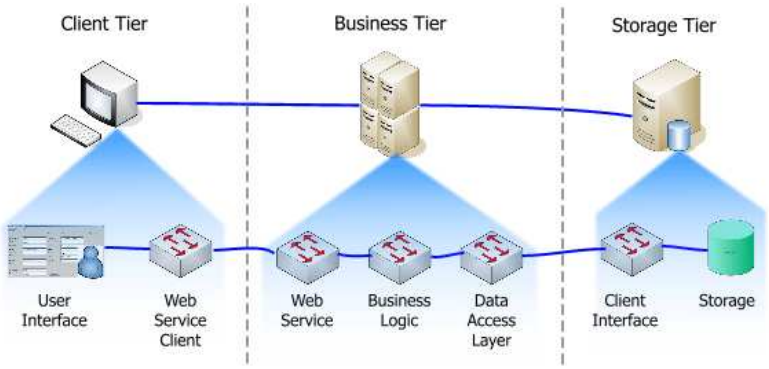
VB.NET

```
Try
‘ Check if All Information is Correct if Yes do:
sqlConnection1.Open()
sqlCommand1.CommandText = "insert into Clients_info values('" &
Email_TextBox.Text & "','" & Pass_TextBox.Text & "','" &
Name_TextBox.Text & "'," & Zipcode_TextBox.Text & ",'" &
address_TextBox.Text & "')"
sqlCommand1.ExecuteNonQuery()
Response.Redirect("http://www.fadidotnet.org/welcome.aspx")
Catch ex As Exception
msg_txt.Text = ex.Message
Finally sqlConnection1.Close ()
End Try
```

وبعد ذلك سينتقل إلى الصفحة التي تحتوي على المنتجات الخاصة بالشركة حيث يمكنك اختيار السلعة التي تريد شرائها ويتم حفظها في سلة المشتريات...



يتم في الأنظمة الموزعة فصل الـ Client Information Table بوصفها ضمن الـPresentation Tier عن بقية الأمور المتعلقة بالشراء وما شابه Business Tire ويتم ربطها جميعا باستخدام Web Services يربط مع مشروع الـ ASP.NET الخاص بالشركة وترسل المعلومات كـ Serialization Object من نظام إلى آخر ، ويفترض الفصل التام بين كل طبقة من طبقات النظام الموزع حيث سترجع كل Remote Method قيمة صافية دون إظهار كيفية استخراج هذه القيمة، لاحظ الشكل التالي:



في الـ Business Tier Web Services :

ترجع هذه الـ Web Services والتي تحتوي على (نظام البيع و الخصومات و الضرائب) إلى الـ Presentation Tier القيمة الصافية المطلوبة من الزبون بعد اختياره شراء سلعة من الموقع حيث ستأخذ أربعة باروميترات الأول سيحتوي على رمز الدولة والثاني سيحتوي على الـ Email الخاص بالزبون والثالث على الـ Barcode الخاص بالسلعة والرابع سيحتوي على طريقة الدفع وترجع هذه الـ Method القيمة الصافية المطلوبة من الزبون لشراء السلعة ويتم تصميمها كما يلي:

C#

```
[WebMethod]
public float Business_Tier_Get_Net_Cost (int zip_code,string
email,string barcode,string payment_method,int quantity)
{
float tax =  Calculate_Tax(email,zip_code,barcode,quantity);
float dicount =  Calculate_Discount(email,zip_code,barcode,quantity);
float DHL_Cost =
DHL_Cost(Products_barcode,Product_weighing,Company_ID,quantity;
float Product_Cost = Product(barcode,quantity);
return  (Product_Cost + tax + DHL_Cost) - dicount;
}
```

VB.NET

```
<WebMethod()> _
Public Function Business_Tier_Get_Net_Cost(ByVal zip_code As
Integer, ByVal email As String, ByVal barcode As String, ByVal
payment_method As String, ByVal quantity As Integer) As Single
   Dim tax As Single = Calculate_Tax(email, zip_code, barcode,
quantity)
   Dim dicount As Single = Calculate_Discount(email, zip_code,
barcode, quantity)
   Dim DHL_Cost As Single = DHL_Cost(Products_barcode,
Product_weighing, Company_ID, quantity)
   Dim Product_Cost As Single = Product(barcode, quantity)
   Return (Product_Cost + tax + DHL_Cost) - dicount
End Function
```

وفي حالة موافقة الزبون على شراء السلعة سينتقل به الموقع إلى الصفحة التي تحتوي على معلومات البنك ، حيث يتم تمرير القيم إلى الـ Server عبر الـ Web Service الخاصة بالبنك وكما يلي:

C#:

```
[WebMethod]
public bool Payment (string Customer_Name,string Address,int
zip_code,string payment_method,string ecard_number,string
acount_number)
{
Return Bank_Tier( Customer_Name, Address, zip_code,
payment_method, ecard_number, Net_Cost, acount_number);
}
```

VB.NET:

```
<WebMethod()> _
Public Function Payment(ByVal Customer_Name As String, ByVal
Address As String, ByVal zip_code As Integer, ByVal payment_method
As String, ByVal ecard_number As String, ByVal acount_number As
String) As Boolean
   Return Bank_Tier(Customer_Name, Address, zip_code,
payment_method, ecard_number, Net_Cost, acount_number)
End Function
```

وكما لاحظنا في النظم الموزعة فإن كل خدمة من الخدمات تعمل بجهة منفصلة عن الجهة أو الواجهة التي يتعامل معها الزبون، وتعمل جميعا وكأنها في نظام واحد... سنتحدث في الجزء التالي عن الـ Serialization والـRemoting وتطبيقاتها في برمجيات الشبكات والنظم الموزعة.

ثانيا - The Serialization Programming :

The Serialization Classes & Members : يمكننا من خلال الـSerialization عمل Serializing لـObject من جهاز إلى آخر سواء عبر الشبكة المحلية باستخدام الـ Stream Socket أو عبر الإنترنت باستخدام الـWebServices والـRemoting ، ويمكننا استخدام الـ Serialization في الـدوت نيت من خلال الـ System.Runtime.Serialization Namespace حيث يحتوي هذا الـ Namespace على عدد ضخم من الـClasses والتي تساعد في عمل Serialization لأي Object عبر الإنترنت أو الشبكة المحلية، ويبين الجدول التالي أهم هذه الـ Classes والتي يمكننا الاستفادة منها في بناء النظم الموزعة:

Formatters , Formatter & FormatterServices	ويحتوي هذا الـ Class على كل الوظائف الرئيسية لعمل Serialization لأي Object ومنها يستخدم الـ Serialize Method لإرسال Object Serializing عبر الـ Stream والـ Deserialize والتي تستخدم للاستقبال الـ Serializing Object من الـStream و تحويله إلى Object مرة أخرى. وتأخذ عملية الإرسال نوعين هما الإرسال عبر بروتوكول الـ Soap ويأتي ضمن الـ Namespace *System.Runtime.Serialization.Formatters.Soap* حيث يجب إضافته إلى الـ Reference للمشروع وقد بينا في الفصل السابق كيفية عمل بروتوكول الـ SOAP حيث يحول الـ Object إلى XML Serial Stream ، والطريقة الثانية هي بتحويل الـObject إلى Binary Serial Stream ويأتي ضمن الـNamespace : *System.Runtime.Serialization.Formatters.Binary* والذي سيتم شرحه بالتفصيل في الأمثلة القادمة في هذا الفصل.
FormatterConverter	حيث يحتوي على الـ IFormatter Interface والـ IConvertible Interface لأستخدمهما في عمليات تمثيل الـ Object حتى يمكن إرساله عبر الـ Serializing Stream ...
ObjectIDGenerator	ويستخدم لتوليد رقم ID عشوائي لأي Object حيث يحتوي هذا الـ Class على Tow Methods الأول يقوم بتوليد رقم تسلسلي للـ Object يسمى GetID والثاني يرجع قيمة الـ ID الخاص بالـ Object ويسمى HasId ويأخذ كل منهما اسم الـ Object الذي نريد توليد رقم تسلسلي له وقيمة True أو False لتحديد فيما إذا كانت هذه هي المرة الأولى التي تم فيها توليد رقم تسلسلي للـ Object أم لا... GetId (object_name,out bool) →to Set an ID HasId(object_name,out bool) → to Return the ID
SerializationInfo	ويحتوي على كل المعلومات التي يمكن أن نحتاج لها لعملية الـ Serialize و الـ Deserialize لأي Object مثلا AssemblyName و الـ FullTypeName والـ MemberCount وتحويلات الـ Data Types للـ Objects وغيرها...

لإنشاء أي Serializable Class يجب أولا أن يعرف بالـ Attribute [Serializable] حيث يعرف أن هذا الـClass من الـ Classes التي يمكن أن يتم عمل Serializing عليها وكما يلي كمثال لعمل Serializable Class للـBank_Tier الخاص بالمثال السابق:

C#:

```
[Serializable]
public class Bank_Tier
{
public string Customer_Name;
public string Address;
public int zip_code;
public string payment_method;
public string ecard_number;
public float depositor;
public string acount_number;
public Bank_Tier(){// Get_Session_ID();}
}
```

VB.NET:

```
<Serializable()> _
Public Class Bank_Tier
   Public Customer_Name As String
   Public Address As String
   Public zip_code As Integer
   Public payment_method As String
   Public ecard_number As String
   Public depositor As Single
   Public acount_number As String
   Public Sub New()
      ' Get_Session_ID()
   End Sub
End Class
```

ولإنشاء البرنامج الخاص بعملية الإرسال لابد من تحديد طريقة الإرسال سواء كـ Binary Stream Data أو XML Data ففي الأول سنستخدم الـ BinaryFormatter والموجود ضمن الـNamespace System.Runtime.Serialization.Formatters.Binary أما الثاني فسيستخدم مع الـ SOAP Protocol والمعرف ضمن الـ SOAPFormatter Class والموجود ضمن الـ System.Runtime.Serialization.Formatters.Soap Namespace.

ولعمل Serialization للـ Class السابق باستخدام الـ SOAP Formatter لابد أولا من تعريف الـNamespaces التالية:

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

ثم نقوم بعمل Instance من الـ Bank Web Services حيث نستطيع فيما بعد إسناد المعلومات الآتية من الـPresentation Tier إليه ثم إرسالها إلى الـ Bank Server باستخدام الـ Remoting أو الـNetworkStream أو حفظها على الجهاز كـ Binary Data أو XML File وكما يلي كمثال لتخزين المخرجات بـ XML File باستخدام بروتوكول الـ SOAP :

C#:

```
Bank_Tier bt = new Bank_Tier();
bt.acount_number = acount_number;
bt.Address = Address;
bt.depositor=depositor;
bt.ecard_number = ecard_number;
bt.payment_method = payment_method;
bt.zip_code = zip_code;
Stream str = new FileStream("bank_result.xml", FileMode.Create,
FileAccess.ReadWrite);
IFormatter formatter = new SoapFormatter();
formatter.Serialize(str, bt);
str.Close();
```

VB.NET:

```
Dim bt As Bank_Tier = New Bank_Tier
bt.acount_number = acount_number
bt.Address = Address
bt.depositor=depositor
bt.ecard_number = ecard_number
bt.payment_method = payment_method
bt.zip_code = zip_code

Dim str As Stream = New FileStream("bank_result.xml",
FileMode.Create, FileAccess.ReadWrite)

Dim formatter As IFormatter = New SoapFormatter
formatter.Serialize(str, bt)
str.Close()
```

وسكون الـ Output ملف XML يحتوي على البيانات التي تم إدخالها وكما يلي:

```
- <SOAP-ENV:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:clr="http://schemas.microsoft.com/soap/encoding/clr/1.0"
SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/
">
- <SOAP-ENV:Body>
- <a1:Form1_x002B_Bank_Tier id="ref-1"
xmlns:a1="http://schemas.microsoft.com/clr/nsassem/Serializatio
n/Serialization%2C%20Version%3D1.0.2309.39316%2C%20C
ulture%3Dneutral%2C%20PublicKeyToken%3Dnull">
```

```
  <Customer_Name xsi:null="1" />
  <Address id="ref-3">Amman Jordan</Address>
  <zip_code>962</zip_code>
  <payment_method id="ref-4">Master Card</payment_method>
  <ecard_number id="ref-5">66-6655-444433-
55655</ecard_number>
  <depositor>600</depositor>
  <acount_number id="ref-6">3399948845</acount_number>
  </a1:Form1_x002B_Bank_Tier>
  </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

وبتأكيد نستطيع استخدام هذا المثال لإرسال البيانات عبر الـ Network Stream أو باستخدام الـ Remoting ، كما يمكننا الاستفادة من الـ Serialization لتمثيل أي Object على شكل XML أو Binary Data لاستخدامها في أمور أخرى...
سنبين في المثال التالي كيفية الاستفادة من الـ BinaryFormatter Class لإرسال Object أو Binary Data من جهاز إلى آخر عبر الشبكة كما سنبين كيفية الاستفادة من الـ SOAP Protocol لإرسال Binary Data كـXML Stream وكمثال نريد إرسال صورة من جهاز إلى آخر بعد تحويلها إلى Binary Stream باستخدام الـMemoryStream وكما يلي:

أولا : باستخدام الـ BinaryFormatter في برنامج الإرسال:

C#:

```
using System.Runtime.Serialization.Formatters.Binary;
using System.Runtime.Serialization;
.
.
private void Send_Image_Using_BinaryFormatter
{
MemoryStream ms = new MemoryStream ();
pictureBox1.Image.Save
(ms,System.Drawing.Imaging.ImageFormat.Jpeg);
object op = (object) ms;

BinaryFormatter br = new BinaryFormatter ();

TcpClient myclient = new TcpClient ("localhost",5000);
NetworkStream myns = myclient.GetStream ();
br.Serialize (myns,op);

myns.Close ();
myclient.Close ();
}
```

VB.NET

```
Imports System.Runtime.Serialization.Formatters.Binary
Imports System.Runtime.Serialization
.
.
Private Property Send_Image_Using_BinaryFormatter() As void
Dim ms As MemoryStream = New MemoryStream ()
pictureBox1.Image.Save
(ms,System.Drawing.Imaging.ImageFormat.Jpeg)
Dim op As Object = CObj(ms)

Dim br As BinaryFormatter = New BinaryFormatter

Dim myclient As TcpClient = New TcpClient("localhost", 5000)
Dim myns As NetworkStream = myclient.GetStream()
br.Serialize (myns,op)

myns.Close ()
myclient.Close ()
End Property
```

استخدام الـ BinaryFormatter في برنامج الاستقبال:

ولاستخدام الـ BinarFormatter في برنامج الاستقبال سنستخدم الدالة المعاكسة للـSerialize وهي Deserialize ثم نقوم بعمل Casting للـ Object المستقبل ونحوله إلى MemoryStream Object مرة أخرى ، عندها نستطيع عرضه على الـ Picture box باستخدام الـ FromStream Method والموجودة ضمن الـ Image Class :

C#:

```
void Image_Receiver()
{

NetworkStream myns;
TcpListener mytcpl;
Socket mysocket;
Thread myth;

mytcpl = new TcpListener (5000);
mytcpl.Start ();
mysocket = mytcpl.AcceptSocket ();
myns = new NetworkStream (mysocket);
BinaryFormatter br = new BinaryFormatter ();
object op;

op= br.Deserialize (myns); // Deserialize the Object from Stream
MemoryStream mm = (MemoryStream) op; // Casting The Object to
MemoryStream Object
```

```
pictureBox1.Image = Image.FromStream(mm);
mytcpl.Stop();
if (mysocket.Connected ==true)
          {while (true){Image_Receiver();}}
}
.
.
private void server_Load(object sender, System.EventArgs e)
{

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

VB.NET

```
Imports System.Runtime.Serialization.Formatters.Binary
Imports System.Runtime.Serialization
.
.
Private Sub Image_Receiver()

   Dim myns As NetworkStream
   Dim mytcpl As TcpListener
   Dim mysocket As Socket
   Dim myth As Thread

   mytcpl = New TcpListener(5000)
   mytcpl.Start()
   mysocket = mytcpl.AcceptSocket()
   myns = New NetworkStream(mysocket)
   Dim br As BinaryFormatter = New BinaryFormatter
   Dim op As Object
   op = br.Deserialize(myns) ' Deserialize the Object from Stream
   Dim mm As MemoryStream = CType(op, MemoryStream) ' Casting
The Object to MemoryStream Object

   pictureBox1.Image = Image.FromStream(mm)
   mytcpl.Stop()

   If mysocket.Connected = True Then
      Do While True
         Image_Receiver()
      Loop
   End If
End Sub
```

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

ثانيا : استخدام الـ SOAP في برنامج الإرسال:

حيث سترسل الصورة كـ XML Data ثم تحول في الطرف المقابل إلى صورة مرة أخرى وتتم هذه العملية باستخدام بروتوكول الـ SOAP وكما يلي في برنامج الإرسال:

C#:

```
using System.Runtime.Serialization.Formatters.Soap;
using System.Runtime.Serialization;
.
.
private void Send_Image_Using_ SoapFormatter
{
server fr = new server ();
fr.Show();

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

TcpClient myclient = new TcpClient ("localhost",5000);
NetworkStream myns = myclient.GetStream ();

IFormatter formatter = new SoapFormatter();
formatter.Serialize(myns, op);
myns.Close();
}
```

VB.NET:

```
Imports System.Runtime.Serialization.Formatters.Soap
Imports System.Runtime.Serialization
.
.
Private void Property SoapFormatter() As Send_Image_Using_
Dim fr As server = New server
fr.Show()

Dim ms As MemoryStream = New MemoryStream
```

```
pictureBox1.Image.Save
(ms,System.Drawing.Imaging.ImageFormat.Jpeg)
Dim op As Object = CObj(ms)

Dim myclient As TcpClient = New TcpClient("localhost", 5000)
Dim myns As NetworkStream = myclient.GetStream()

Dim formatter As IFormatter = New SoapFormatter
formatter.Serialize(myns, op)
myns.Close()
End Property
```

استخدام الـ SoapFormatter في برنامج الاستقبال:

ولاستخدام بروتوكول الـ SOAP في برنامج الاستقبال سنستخدم الدالة المعاكسة للـSerialize وهي Deserialize ثم نقوم بعمل Casting للـ Object المستقبل ونحوله إلى MemoryStream Object مرة أخرى ، عندها نستطيع عرضه على الـ Picture box باستخدام الـ FromStream Method والموجودة ضمن الـ Image Class :

C#:

```
using System.Runtime.Serialization.Formatters.Soap;
using System.Runtime.Serialization;
.
.
void Image_Receiver()
{
NetworkStream myns;
TcpListener mytcpl;
Socket mysocket;
mytcpl = new TcpListener (5000);
mytcpl.Start ();
mysocket = mytcpl.AcceptSocket ();
myns = new NetworkStream (mysocket);
object op;
IFormatter formatter = new SoapFormatter();
op= formatter.Deserialize(myns);
myns.Close();
MemoryStream mm = (MemoryStream) op;
pictureBox1.Image = Image.FromStream(mm);
mytcpl.Stop();

        if (mysocket.Connected ==true)
        {
        while (true)
                {Image_Receiver();}
        }
}
.
```

```
.
private void server_Load(object sender, System.EventArgs e)
{
Thread myth;
myth= new Thread  (new System.Threading
.ThreadStart(Image_Receiver)); // Start Thread Session
myth.Start ();
}
```

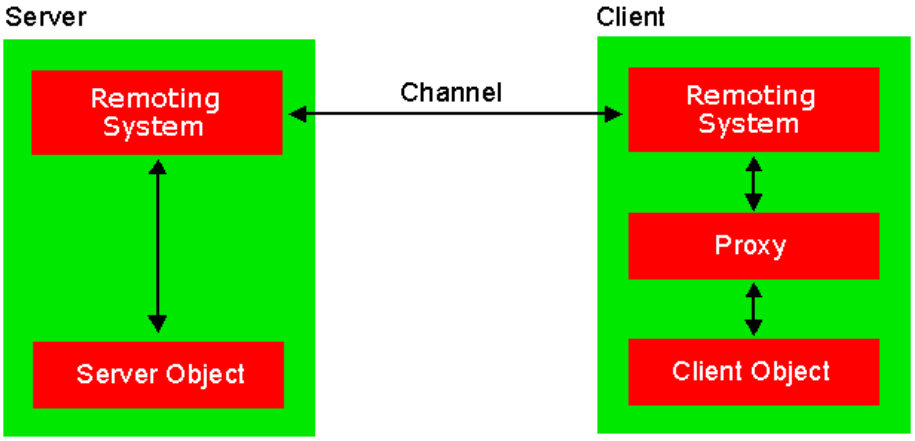
VB.NET:

```
Imports System.Runtime.Serialization.Formatters.Soap
Imports System.Runtime.Serialization
.
.
Private Sub Image_Receiver()
   Dim myns As NetworkStream
   Dim mytcpl As TcpListener
   Dim mysocket As Socket
   mytcpl = New TcpListener(5000)
   mytcpl.Start()
   mysocket = mytcpl.AcceptSocket()
   myns = New NetworkStream(mysocket)
   Dim op As Object
   Dim formatter As IFormatter = New SoapFormatter
   op = formatter.Deserialize(myns)
   myns.Close()
   Dim mm As MemoryStream = CType(op, MemoryStream)
   pictureBox1.Image = Image.FromStream(mm)
   mytcpl.Stop()
   If mysocket.Connected = True Then
      Do While True
         Image_Receiver()
      Loop
   End If
End Sub
.
.
Private Sub server_Load(ByVal sender As Object, ByVal e As
System.EventArgs)
   Dim myth As Thread
   myth = New Thread(New System.Threading.ThreadStart(AddressOf
Image_Receiver)) ' Start Thread Session
   myth.Start()
End Sub
```

وهكذا بينا كيفية استخدام الـ Serialization في بيئة الدوت نيت بنوعيها الـ Binary والـXML Data ، سيتم الحديث في الجزء التالي من هذا الفصل عن الـ Remoting واستخدامها في برمجيات الشبكات والنظم الموزعة...

ثالثا: Remoting Programming :

يعتبر موضوع الـ Remoting من المواضيع الهامة جدا في برمجة الشبكات ، إذ يقدم لنا إمكانيات رائعة في مشاركة و استخدام الـ Remote Methods والموجودة على الـ Remote Server وتشبه عملية التعامل مع الـRemoting عملية الـ Web Services إلى حد كبير حيث يتطلب استخدامها عمل Proxy Object في طرف الـClient، لكن مع إمكانيات اكبر حيث تقوم بإنشاء الـServer بنفسك ولست مضطرا لتثبيت
الـ IIS أو غيره من التطبيقات الوسيطة للاستفادة من خدمات الـ Remote Server ، وتستطيع من خلاله تنفيذ أي عملية معالجة على الـ Server مما يعزز من مبدأ الـDistributed Systems والذي شرحناه في الجزء الأول من هذا الفصل حيث تبقى الـmethods و عملية المعالجة الخاصة بها على الـ Server في حين يتم إرسال الـResult إلى الـ Clients كـ XML Data أو Binary Data حسب قناة الاتصال المستخدمة سواء HttpChannel أو TcpChannel ، وتمرر إلى الـ Client عبر الـ Proxy Object والذي ربط الـ Remote Class الموجود على الـ Server مع الـ Client لاحظ الشكل التالي:



1- Using Remoting in Dot Net :

يمكن استخدام الـ Remoting بطريقتين الأولى عبر بروتوكول الـ HTTP حيث تتم عملية التراسل كـ XML Data بين الـ Server والـ Clients وتتم باستخدام بروتوكول
الـ SOAP أما الطريقة الثانية فتتم باستخدام الـ TCP Protocol حيث تكون عملية التراسل كـ Binary Data بين الـ Server وبقية الـ Clients ، ويفضل دائما استخدام الـ TCP Channel في الحالتين التاليتين:

الأولى: إذا كانت قناة الاتصال سريعة نسبيا
الثانية: توافقية البيئات حيث تستخدم نظام التشغيل Microsoft Windows في جميع الـclients ، ويعتبر هذه الأمر من أهم عيوب استخدام الـ TCP Channel وخاصة في حالة الـWeb حيث لا تضمن البيئة التي سيعمل عليها الـClient ويفضل في هذه الحالة استخدام الـHTTP Channel ، لكن تبقى أدائية الـ TCP Channel أفضل من الـ HTTP ...
وما يميز الـ HTTP Channel هو استخدامه بروتوكول الـ SOAP والذي يتعامل مع الـXML في عملية التراسل مما يضمن توافقيته مع جميع البيئات.

يلزم استخدام الـ Remoting في الدوت نيت ثلاثة أمور :
أولا، تحديد طبيعة القناة المستخدمة سواء عبر الـ HTTP كـXML أو عبر الـ TCP كـBinary Formatter ودعمت الدوت نيت كلا منهما في الـHttpChannel Class والموجود ضمن System.Runtime.Remoting.Channels.Http Namespace والـTcpChannel Class والموجود ضمن
الـSystem.Runtime.Remoting.Channels.Tcp Namespace (يجب إضافته إلى

الـReferences في المشروع)، حيث يجب تعريفها في كلا الطرفين الـ Client والـ Server وكما يلي:

C#:
```
HttpChannel http_Channel = new HttpChannel(Port_Number);
```

VB.NET:
```
Dim http_Channel As HttpChannel = New HttpChannel(Port_Number)
```

وفي حالة استخدام الـ TcpChannel كما يلي:

C#:
```
TcpChannel Tcp_Channel = new TcpChannel(Port_Number);
```

VB.NET:
```
Dim Tcp_Channel As TcpChannel = New TcpChannel(Port_Number)
```

بعد ذلك يجب تسجيل القناة المستخدمة باستخدام الدالة RegisterChannel والموجودة ضمن الـChannelServices Class ضمن الـ System.Runtime.Remoting.Channels Namespace ، ويتم ذلك في كلا الطرفين أيضا الـClient والـ Server وكما يلي:

C#:
```
ChannelServices.RegisterChannel(chan);
```

VB.NET:
```
ChannelServices.RegisterChannel(chan)
```

بعد ذلك يجب تعريف الـ Class الذي سيتم مشاركته على الـ Server ويتم ذلك باستخدام الدالة RegisterWellKnownServiceType والموجودة ضمن
الـRemotingConfigurationClass والموجود ضمن
الـSystem.Runtime.Remoting Namespace ويتم ذلك كما يلي في طرف الـ Server:

C#:
```
RemotingConfiguration.RegisterWellKnownServiceType
(Type.GetType ("Class_File_Name, Distributed_Class_Name"),
"unique_Service_Name", WellKnownObjectMode.SingleCall);
```

VB.NET:
```
RemotingConfiguration.RegisterWellKnownServiceType
(Type.GetType ("Class_File_Name, Distributed_Class_Name"),
"unique_Service_Name", WellKnownObjectMode.SingleCall)
```

وتأخذ الـ RegisterWellKnownServiceType Method ثلاثة باراميترات الأول الـType الخاص باسم الـRemote Class مع اسم الـ Dll File ومساره والذي نريد توزيعه ويأخذ الباراميتر الثاني اسم الـ URI الخاص بالـ Object الذي نريد توزيعه ويجب أن يكون الـ Object URI اسم فريد unique على مستوى الـPort Application ، ويأخذ الباراميتر الثالث نوع الـObject Remote Instance فإذا تم اختيار SingleCall فهذا يعني إنشاء New Instance جديد لكل Client يقوم بإنشاء Instance من الـ Remote Class، وإذا تم اختيار Singleton فهذا يعني استخدام نفس الـ Remote Instance

Object لكـل الـ Clients، وينصـح اسـتخدام SingleCall فـي حالـة كـان النظـام المـوزع سيستخدم Session لكل User وينصح باستخدام Singleton في حالة كان الـ Server سيبث نفس المعلومـات إلـى كل الـ Clients علـى الشبكة... أمـا فـي الـ Client فيجـب تعريـف نـوع الـChannel سـواء TCP أو HTTP Channel حسـب نـوع القنـاة التـي تـم اسـتخدمها فـي الـServer ولا يتم وضع رقم الـ Port في الـ Client Channel ونكتفي بوضعه في الـ URI Object ويتم تعريف القناة في الـ Client كما يلي:

C#:
```
HttpChannel http_Channel = new HttpChannel
```

VB.NET:
```
Dim http_Channel As HttpChannel = New HttpChannel()
```

وفي حالة استخدام الـ TcpChannel كما يلي:

C#:
```
TcpChannel Tcp_Channel = new TcpChannel();
```

VB.NET:
```
Dim Tcp_Channel As TcpChannel = New TcpChannel()
```

ويتم تسجيل القناةRegisterChannel في الـ Client كما هو الحال في الـ Server وكما يلي:

C#:
```
ChannelServices.RegisterChannel(chan);
```
VB.NET:
```
ChannelServices.RegisterChannel(chan)
```

بعد ذلك نقوم بتعريف New Instance Object من الـ Remote Client حيث يجب أن يحتوي الـClient Project على الـ Dll File الخـاص بالـ Distributed Class حيث نقوم في البداية بتعريف الـ URI والذي سيحتوي على الـ Protocol المستخدم ويتبع بعنوان الجهاز أو الـ DNS الخاص بالـ Server ومن ثم رقم الـPort المستخدم و اسم الخدمة التي تم تعريفها في الـ Server ويتم ذلك كما يلي:

في حالة استخدام TCP Channel :

C#:
```
string URI = "Tcp://"+Remote_IP.Text+":Port/ unique_Service_Name
";
```

VB.NET:
```
Dim URI As String = "Tcp://" & Remote_IP.Text & ": Port/
unique_Service_Name "
```

في حالة استخدام HTTP Channel :

C#:
```
string URI = "Http://"+Remote_IP.Text+": Port /
unique_Service_Name";
```

VB.NET:
```
Dim URI As String = " Http://" & Remote_IP.Text & ": Port /
unique_Service_Name "
```

وبعد ذلك نقوم بإنشاء New Instance Object من الـ Remote Class ويتم ذلك باستخدام الـGetObject Method والموجود ضمن الـ Activator Class ونمرر لها الـDistributed Class Type و الـ URI الذي عرفناه ، وكما يلي:

C#:
```
Distributed_Class_Name obj = (Distributed_Class_Name)
Activator.GetObject
(typeof (Distributed_Class_Name), URI);
```

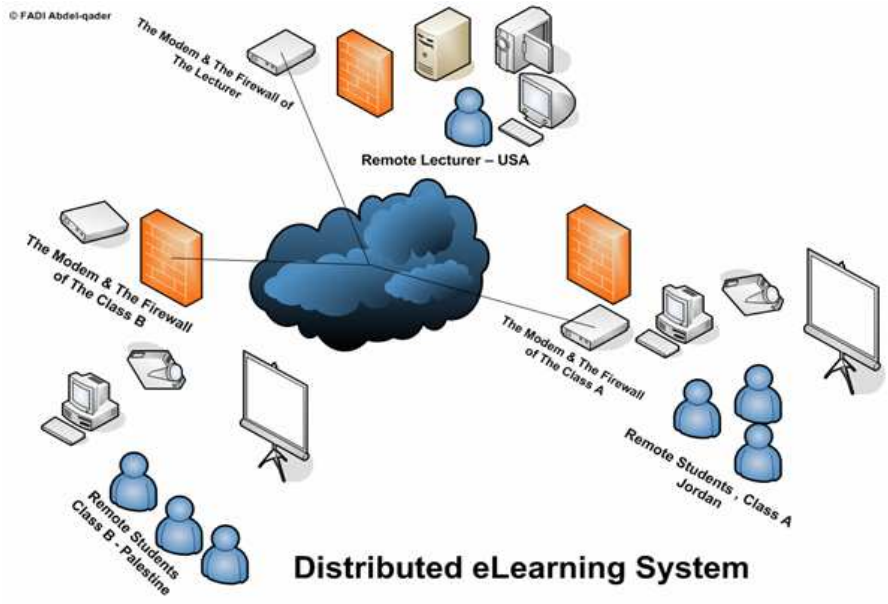
VB.NET:
```
Dim obj As Distributed_Class_Name =
CType(Activator.GetObject(GetType(Distributed_Class_Name), URI),
Distributed_Class_Name)
```

ملاحظة هامة:
يجب أن يتم توريث الـ MarshalByRefObject Class إلى الـ Distributed Class الذي نريد توزيعه والموجود ضمن System Namespace وعند توريث الـMarshalByRefObject إلى الـ Distributed Class سيتم تنفيذ كافة العمليات أو الـProcess على الـ Server وسيحصل الـ Client على الـ Result النهائي في حين إذا تم استخدام الـMarshalByValueObject عندها سيتم استخدام الـ Remote Class و لكن ستتم عملية المعالجة على الـ Client Side ...

وهكذا بينا كيفية استخدام الـ Remoting في الدوت نيت سنقوم الآن بتطبيق هذه المفاهيم في مشروعين الأول سيكون نظام تعليمي موزع حيث سيبث فيها صورة الكاميرا للمحاضر وسطح المكتب الخاص به إلى الطلبة والموجودين في دول متعددة.

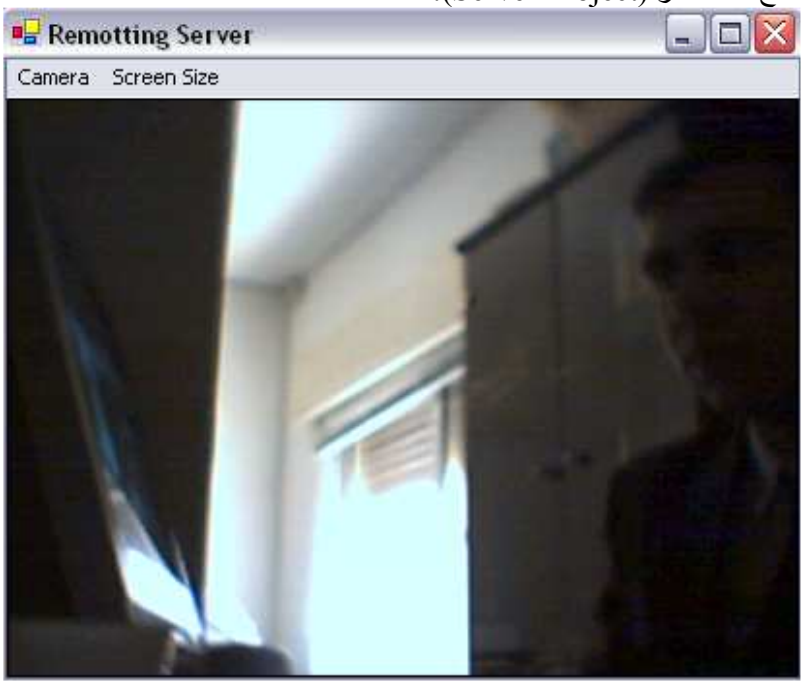
أولا: إنشاء نظام Distributed eLearning System :

لإنشاء هذا النظام لابد أولا من بإنشاء Distributed Class يقوم بعملية التقاط لصورة سطح المكتب الخاص بالـServer وبثها إلى الـ Client مستخدما الـ TcpChannel بين الـ Server والـ Client وسنفترض في هذا المثال إنشاء Remote Classroom حيث يقوم المعلم بإلقاء محاضرة إلى طلابه ويتم بث صورة الكاميرا وسطح المكتب الخاص بالمعلم عبر الإنترنت إلى الطلاب في الـ Classroom والموجودين في دول متعددة وكما في الشكل التالي:

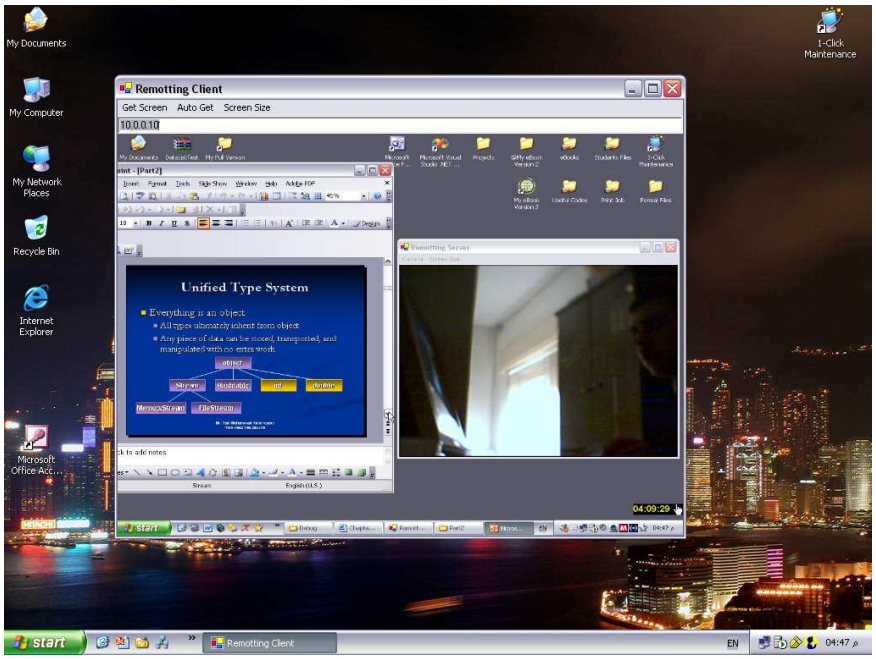


سنستخدم في هذا المثال الـ TCP Channel والـ HttpChannel وسنقارن الأداء في كل منهما ، سيكون الشكل العام للبرنامج شبيه بالشكل التالي:

أولا برنامج المحاضر (Server Project):



برنامج الـ Client ويمكن أن يكون من خلال الإنترنت في حالة كان لديك Real IP أو في الشبكة المحلية وكما هو مبين في الشكل التالي:



ولإنشاء الـ Remote Class والذي سيقوم بجلب صورة سطح المكتب ، سنستخدم دوال الـAPI ومنها الـMethod GetDesktopWindow والموجودة ضمن الـuser32.dl والـMethod BitBlt لرسم الصورة، ثم سنقوم بتحويل الصورة إلى Byte Array وحتى نستطيع إرسالها إلى الـ Clients ، وبتأكيد سنستخدم طريقة MarshalByRefObject وحتى تتم عملية التقاط ومعالجة الصورة على الـ Server Side ، ويتم ذلك كما يلي:

C#:

```
using System;
using System.Drawing;
using System.Drawing.Imaging;
using System.Runtime.InteropServices;
using System.IO;

public class ScreenCapture : System.MarshalByRefObject
{
[DllImport("user32.dll")]
private static extern IntPtr GetDesktopWindow();

[DllImport("gdi32.dll")]
private static extern bool BitBlt(
        IntPtr hdcDest, // handle to destination DC
        int nXDest, // x-coord of destination upper-left corner
        int nYDest, // y-coord of destination upper-left corner
        int nWidth, // width of destination rectangle
        int nHeight, // height of destination rectangle
        IntPtr hdcSrc, // handle to source DC
        int nXSrc, // x-coordinate of source upper-left corner
        int nYSrc, // y-coordinate of source upper-left corner
        System.Int32 dwRop // raster operation code
);

private const Int32 SRCCOPY = 0xCC0020;
[DllImport("user32.dll")]
private static extern int GetSystemMetrics(int nIndex);

private const int SM_CXSCREEN = 0;
private const int SM_CYSCREEN = 1;

public Size GetDesktopBitmapSize()
{
return new Size(GetSystemMetrics(SM_CXSCREEN),
GetSystemMetrics(SM_CYSCREEN));
}

public byte[] GetDesktopBitmapBytes()
        {
Size DesktopBitmapSize = GetDesktopBitmapSize();
Graphics Graphic = Graphics.FromHwnd(GetDesktopWindow());
Bitmap MemImage = new Bitmap(DesktopBitmapSize.Width,
DesktopBitmapSize.Height, Graphic);
Graphics MemGraphic = Graphics.FromImage(MemImage);
IntPtr dc1 = Graphic.GetHdc();
```

```
IntPtr dc2 = MemGraphic.GetHdc();
BitBlt(dc2, 0, 0, DesktopBitmapSize.Width,
DesktopBitmapSize.Height, dc1, 0, 0, SRCCOPY);
Graphic.ReleaseHdc(dc1);
MemGraphic.ReleaseHdc(dc2);
Graphic.Dispose();
MemGraphic.Dispose();

Graphics g = System.Drawing.Graphics.FromImage(MemImage);
System.Windows.Forms.Cursor cur =
System.Windows.Forms.Cursors.Arrow;
cur.Draw(g,new Rectangle(System.Windows.Forms.Cursor.Position.X-
10,System.Windows.Forms.Cursor.Position.Y-
10,cur.Size.Width,cur.Size.Height));

MemoryStream ms = new MemoryStream();
MemImage.Save(ms,System.Drawing.Imaging.ImageFormat.Jpeg);
return ms.GetBuffer();
}
}
```

VB.NET

```
Imports System
Imports System.Drawing
Imports System.Drawing.Imaging
Imports System.Runtime.InteropServices
Imports System.IO

Public Class ScreenCapture : Inherits System.MarshalByRefObject
   <DllImport("user32.dll")> _
   Private Shared Function GetDesktopWindow() As IntPtr
   End Function

   <DllImport("gdi32.dll")> _
   Private Shared Function BitBlt(ByVal hdcDest As IntPtr, ByVal
nXDest As Integer, ByVal nYDest As Integer, ByVal nWidth As
Integer, ByVal nHeight As Integer, ByVal hdcSrc As IntPtr, ByVal
nXSrc As Integer, ByVal nYSrc As Integer, ByVal dwRop As
System.Int32) As Boolean
   End Function

   Private Const SRCCOPY As Int32 = &HCC0020
   <DllImport("user32.dll")> _
Private Shared Function GetSystemMetrics(ByVal nIndex As Integer)
As Integer
End Function
```

```
 Private Const SM_CXSCREEN As Integer = 0
 Private Const SM_CYSCREEN As Integer = 1
 Public Function GetDesktopBitmapSize() As Size
 Return New Size(GetSystemMetrics(SM_CXSCREEN),
GetSystemMetrics(SM_CYSCREEN))
   End Function

   Public Function GetDesktopBitmapBytes() As Byte()
      Dim DesktopBitmapSize As Size = GetDesktopBitmapSize()
      Dim Graphic As Graphics =
Graphics.FromHwnd(GetDesktopWindow())
      Dim MemImage As Bitmap = New
Bitmap(DesktopBitmapSize.Width, DesktopBitmapSize.Height,
Graphic)
      Dim MemGraphic As Graphics =
Graphics.FromImage(MemImage)
      Dim dc1 As IntPtr = Graphic.GetHdc()
      Dim dc2 As IntPtr = MemGraphic.GetHdc()
      BitBlt(dc2, 0, 0, DesktopBitmapSize.Width,
DesktopBitmapSize.Height, dc1, 0, 0, SRCCOPY)
      Graphic.ReleaseHdc(dc1)
      MemGraphic.ReleaseHdc(dc2)
      Graphic.Dispose()
      MemGraphic.Dispose()

Dim g As Graphics =
System.Drawing.Graphics.FromImage(MemImage)
Dim cur As System.Windows.Forms.Cursor =
System.Windows.Forms.Cursors.Arrow
cur.Draw(g, New Rectangle(System.Windows.Forms.Cursor.Position.X
- 10, System.Windows.Forms.Cursor.Position.Y - 10, cur.Size.Width,
cur.Size.Height))

Dim ms As MemoryStream = New MemoryStream
MemImage.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg)
Return ms.GetBuffer()
   End Function
End Class
```

وبتأكيد يجب تحويل الـ Class السابق إلى Dll File ثم وضعه بجانب الملف التنفيذي للمشروع ، وأيضا إضافته ضمن الـ References في مشروع الـ Client وحتى نستطيع عمل الـ Casting على الـ Object القادم من الـSever .

أولا : إنشاء الـ Remoting Server :
سيحتوي برنامج الـ Server على الأمور التالية:

Class	Object Name
pictureBox	pictureBox1
mainMenu	mainMenu1
WebCamCapture – External Dll File	WebCamCapture1

سنقوم الآن بإنشاء وتسجيل TCP Remote Channel ثم إنشاء الـ Server Object ويتم ذلك كما يلي:

C#:
```
TcpChannel chan = new TcpChannel(6600);
ChannelServices.RegisterChannel(chan);
RemotingConfiguration.RegisterWellKnownServiceType(Type.GetTyp
e("ScreenCapture, ScreenCapture"),
"MyCaptureScreenServer",WellKnownObjectMode.Singleton);
```

VB.NET
```
Dim chan As TcpChannel = New TcpChannel(6600)
ChannelServices.RegisterChannel(chan)
RemotingConfiguration.RegisterWellKnownServiceType(Type.GetTyp
e("ScreenCapture, ScreenCapture"),
"MyCaptureScreenServer",WellKnownObjectMode.Singleton)
```

قمنا بتسمية الـ Server Object بـ MyCaptureScreenServer و طبيعة الـ Instance سيكون Singleton وهذا يعني استخدام نفس الـ Instance Object لكل الـ Clients و في حالة إذا أردنا إنشاء New Instance Object لكل Client Request فيجب استخدام الـ SingleCall وفي حالتنا هذه ينصح باستخدام الأول وحتى لا يزيد الـ Load على الـServer في حالة ربطه على مجموعة كبيرة من الـ Clients إذ أن المعلومات التي يتم بثها إلى الـ Clients هي نفسها ...

ولربط الـ WebCamCapture Class في الـ Picturebox نضع الكود التالي في حدث ImageCaptured الخاص بالـ WebCamCapture Class وكما يلي:

C#:
```
private void webCamCapture1_ImageCaptured(object source,
WebCam_Capture.WebcamEventArgs e)
        {
pictureBox1.Image = e.WebCamImage;
        }
```

VB.NET:

```
Private Sub webCamCapture1_ImageCaptured(ByVal source As
Object, ByVal e As WebCam_Capture.WebcamEventArgs)
   pictureBox1.Image = e.WebCamImage
End Sub
```

والآن نستطيع تشغيل الكاميرا عند الضغط على الـ Start Button الموجود في الـ main Menu ونضع في الحدث الخاص بها كود التشغيل ونمرر له الـ Period Time لكل عملية التقاط وهو مثلا 1 Milliseconds ويتم ذلك كما يلي:

C#:

```
this.webCamCapture1.TimeToCapture_milliseconds = 1;
this.webCamCapture1.Start(0);
```

VB.NET:

```
Me.webCamCapture1.TimeToCapture_milliseconds = 1
Me.webCamCapture1.Start(0)
```

ثانيا : إنشاء الـ Remoting Client :
ويحتوي أيضا على الأمور التالية:

Class	Object Name
pictureBox	pictureBox1
mainMenu	mainMenu1
Timer	timer1
TextBox	textBox1
ScreenCapture Class – External Class in References	ScreenCapture

وسنقوم بتعريف المتغيرات والـ Objects التالية في الـ Global Declaration للبرنامج:

C#:

```
ScreenCapture obj;
TcpChannel chan;
string URI;
```

VB.NET:

```
Dim obj As ScreenCapture
Dim chan As TcpChannel
Dim URI As String
```

سنستخدم الـScreenCapture Object لعمل الـ Casting على الـ Incoming Remote Instance Object ، وسنستخدم الـ TcpChannel Object لإنشاء الـ Remoting Channel وسنضع في الـ URI String Variable الـ URI الخاص بالـ Server Object والذي تم تعريفه في الـ Server .

ثم سنقوم بوضع التعريفات الأساسية للـ TCP Channel والـ Remote Object في الـConstructer الخاص بالمشروع وكما يلي:

C#:
```
public Form1()
{

URI = "Tcp://"+textBox1.Text+":6600/MyCaptureScreenServer";
chan = new TcpChannel();
ChannelServices.RegisterChannel(chan);
obj = (ScreenCapture)Activator.GetObject(typeof(ScreenCapture),
URI);
 }
```

VB.NET:
```
URI = "Tcp://" & textBox1.Text & ":6600/MyCaptureScreenServer"
chan = New TcpChannel()
ChannelServices.RegisterChannel(chan)
obj = CType(Activator.GetObject(GetType(ScreenCapture), URI),
ScreenCapture)
```

حيث سيتم تمرير الـ Remote IP أو الـ DNS الخاص بالـ Server من الـ TextBox ويتبعه رقم الـ Port والـ Remote Object الذي تم تعريفه في الـ Client...

ولجلب الصورة سنضع الكود التالي في Timer حيث يجلب الصورة من الـ Server كل فترة محددة:

C#:
```
private void timer1_Tick(object sender, System.EventArgs e)
{
try
{
URI = "Tcp://"+textBox1.Text+":6600/MyCaptureScreenServer";
byte[] buffer = obj.GetDesktopBitmapBytes();
MemoryStream ms = new MemoryStream(buffer);
pictureBox1.Image = Image.FromStream(ms);
}
catch(Exception ex){timer1.Enabled=false;
MessageBox.Show(ex.Message);}
}
```

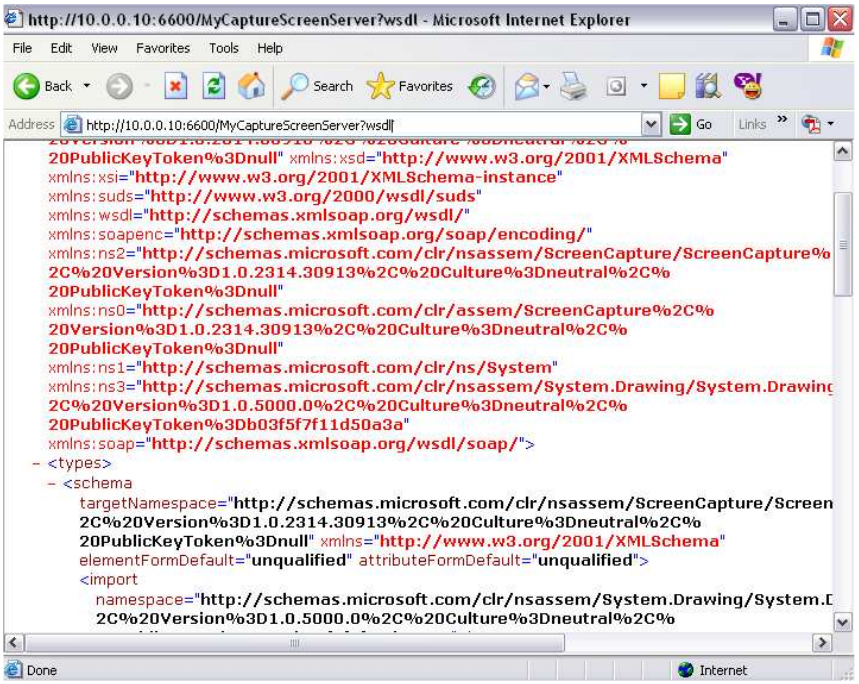
VB.NET:
```
 Private Sub timer1_Tick(ByVal sender As Object, ByVal e As
System.EventArgs)
   Try
      URI = "Tcp://" & textBox1.Text &
":6600/MyCaptureScreenServer"
      Dim buffer As Byte() = obj.GetDesktopBitmapBytes()
      Dim ms As MemoryStream = New MemoryStream(buffer)
```

```
      pictureBox1.Image = Image.FromStream(ms)
   Catch ex As Exception
      timer1.Enabled = False
      MessageBox.Show(ex.Message)
   End Try
End Sub
```

ولتحويل قناة الاتصال إلى Http Channel فقط قم بتغيير الـ TcpChannel إلى HttpChannel وعندها ستتمكن من مشاهدة الـ Remote Methods عبر الـ Internet Browser ولمشاهدتها نضيف الـ WSDL – Web Services Definition Language Query التالي إلى الـ URI الخاص بالـ Remote Object:

http://10.0.0.10:6600/MyCaptureScreenServer?wsdl

وستظهر لنا الـ Remote Interface Class بهيئة XML لاحظ الشكل التالي:



وتستطيع الاستفادة من التركيب السابق للـ Class في حالة لم تتمكن من الحصول على الـ Dll الخاص بالـ File Remote Class ، حيث تستطيع تحويل الـ XML السابق إلى ملف Dll لوضعه مع الـ References في برنامج الـ Client ويتم ذلك باستخدام الـ Metadata ومنها الـ Standard w3cxsd2001 حيث يوضع الجزء الأول الخاص بالـ Namespace XML Namespace في الـ Serializable Attribute وكما يلي كمثال:

C#:

```
using System;
using System.Runtime.Remoting.Messaging;
using System.Runtime.Remoting.Metadata;
using System.Runtime.Remoting.Metadata.W3cXsd2001;
```

```
[Serializable,SoapType(XmlNamespace=@"http://http://schemas.micro
soft.com/clr/nsassem/ScreenCapture/ScreenCapture%2C%20Version%3
D1.0.2314.30913%2C%20Culture%3Dneutral%2C%20PublicKeyToke
n%3Dnull")]
public class ScreenCapture : System.MarshalByRefObject
{// Your Methods with Return Values Only}
```

VB.NET:

```
Imports System
Imports System.Runtime.Remoting.Messaging
Imports System.Runtime.Remoting.Metadata
Imports System.Runtime.Remoting.Metadata.W3cXsd2001

<Serializable(),
SoapTypeAttribute(XmlNamespace:="http://http://schemas.microsoft.co
m/clr/nsassem/ScreenCapture/ScreenCapture%2C%20Version%3D1.0.
2314.30913%2C%20Culture%3Dneutral%2C%20PublicKeyToken%3
Dnull"), Serializable(),
SoapTypeAttribute(XmlNamespace:="http://http://schemas.microsoft.co
m/clr/nsassem/ScreenCapture/ScreenCapture%2C%20Version%3D1.0.
2314.30913%2C%20Culture%3Dneutral%2C%20PublicKeyToken%3
Dnull")> _
Public Class ScreenCapture : Inherits System.MarshalByRefObject
‘Your Methods with Return Values Only
End Class
```

وللاختيار بين الـ TCP أو الـ HTTP Channel فيفضل دائما في حالة كان الـ Remote Object موجود على جهاز تتصل معه عبر الإنترنت استخدام الـ HTTP Channel بسبب محدودية سرعة الاتصال بالإضافة إلى كونه يتوافق مع أي نظام سواء Windows أو غيره لكن يبقى أداء وسرعة الـ TCP Channel أفضل فقط في الـLocal Network ...

وهكذا بينا كيفية استخدام الـ Remoting في الدوت نيت وطبقنا عملية إنشاء Distributed eLearning System ، وبتأكيد تستطيع الآن إضافة ميزة نقل الصوت إلى جانب نقل صورة الشاشة والكاميرا الخاصة بالمحاضر (لمزيد من المعلومات ارجع إلى الفصل الخاص بالـVOIP)، وسنبين في الجزء التالي من هذا الفصل كيفية إنشاء برنامج Remote Desktop Application مع خاصية التحكم.

ثانيا : إنشاء برنامج التحكم عن بعد Create an Advanced Remote Desktop Application With Remote (Mouse/Keyboard) Control Features :

تعتمد الفكرة الأساسية في مثل هذه البرامج على إرسال إحداثيات الـMouse (X,Y) والـKay Press Event إلى الـ Server وبتم كل ذلك باستخدام دوال الـ API والموجودة في ملف المستخدم الشهير user32.dll حيث يحتوي هذا الملف على عدد ضخم من الدوال ومنها الدالة SendInput والتي ترسل من خلالها الـ input Button Type الذي تم الضغط عليه بالـ Keyboard أو الـ Mouse والدالة SetCursorPos وتأخذ إحداثيات
الـ X والـ Y للـ Mouse حيث يتم وضع مؤشر الـMouse على الإحداثيات المرسلة ، ويمكن استخدام الدالة GetSystemMetrics لمعرفة حجم الـ Screen Desktop للـServer (Height & Width) ، ويمكن استخدام هذه الدوال لتحكم عن بعد بطريقتين:

1- إرسال الإحداثيات إلى الـ Server كـ Serialization Object عبر الـ Socket ثم استقبالها وتنفيذها بتمرير الإحداثيات المستقبلة إلى الدوال ، و لكن المشكلة في هذه الطريقة هي البطء في عملية إرسال الإحداثيات حيث يمكن أن يحدث تأخير إثناء الإرسال ناهيك عن الـBandwidth الذي ستحاجه في عملية إرسال الـ Control Object ...

2- استخدام الـ Remoting في عملية التحكم حيث يستدعي الـ Client الـ Methods الموجودة على الـServer ومن ثم وتجرى عملية معالجتها على الـ Server Side ويمرر الـClient الإحداثيات الخاصة بالتحكم إلى الـRemote Methods الموجودة على الـServer ، وتعتبر هذه الطريقة من أسرع الطرق لإرسال معلومات التحكم إلى الـServer ، حيث يتم استدعاء الـ Methods الموجودة على الـ Server وكأنها على الـClient و تنفذ بعد ذلك على الـ Server Side .
وللبدء سنقوم بإنشاء الـ Distributed Class والذي سيحتوي على كل الدوال الخاصة بالتحكم ، وسنستخدم System.MarshalByRefObject ولكي يتم تنفيذ كل الـMethods على الـServer Side ، في البداية سنعرف مجموعة من الـ Unsigned Variables ولكي نستخدمها لاحقا لتمرير طبيعة العملية التي نريد التحكم بها ونمرر لها مجموعة القيم بالـ Hex Decimal والتي تخص كل عملية :

C#:

```
const uint MOUSEEVENTF_MOVE      = 0x0001; //mouse move
const uint MOUSEEVENTF_LEFTDOWN  = 0x0002; // left button down
const uint MOUSEEVENTF_LEFTUP    = 0x0004; // left button up
const uint MOUSEEVENTF_RIGHTDOWN = 0x0008; // right button down
const uint MOUSEEVENTF_RIGHTUP   = 0x0010; // right button up
const uint MOUSEEVENTF_MIDDLEDOWN = 0x0020; // middle button down
const uint MOUSEEVENTF_MIDDLEUP = 0x0040; // middle button up
const uint MOUSEEVENTF_WHEEL  = 0x0800; // wheel button rolled
const uint MOUSEEVENTF_ABSOLUTE   = 0x8000; // absolute move
const uint KEYEVENTF_EXTENDEDKEY = 0x0001;
const uint KEYEVENTF_KEYUP       = 0x0002;
const uint INPUT_MOUSE    = 0;
const uint INPUT_KEYBOARD = 1;
```

VB.NET:

```
Const MOUSEEVENTF_MOVE As Integer = &H1    'mouse move
Const MOUSEEVENTF_LEFTDOWN As Integer = &H2    ' left button
down
Const MOUSEEVENTF_LEFTUP As Integer = &H4    ' left button up
Const MOUSEEVENTF_RIGHTDOWN As Integer = &H8    ' right
button down
Const MOUSEEVENTF_RIGHTUP As Integer = &H10   ' right button
up
Const MOUSEEVENTF_MIDDLEDOWN As Integer= &H20 'middle
button down
Const MOUSEEVENTF_MIDDLEUP As Integer = &H40   ' middle
button up
Const MOUSEEVENTF_WHEEL As Integer = &H800  ' wheel button
rolled
Const MOUSEEVENTF_ABSOLUTE As Integer = &H8000 ' absolute
move
Const KEYEVENTF_EXTENDEDKEY As Integer = &H1
Const KEYEVENTF_KEYUP As Integer = &H2
Const INPUT_MOUSE As Integer = 0
Const INPUT_KEYBOARD As Integer = 1
```

ثم سنعرف مجموعة من الـ Structs لاستخدامها في عملية إدخال الـ Kay Press Value من قبل الـ Client إلى الـ Server وهي MOUSE_INPUT لإدخال Mouse Buttons Press Values والـKEYBD_INPUT لإدخال Keyboard Button Press Value وتستدعى كل منها باستخدام الـ INPUT Struct وكما يلي:

C#:

```
struct MOUSE_INPUT
{
public uint dx;
public uint dy;
public uint mouseData;
public uint dwFlags;
public uint time;
public uint dwExtraInfo;
}

struct KEYBD_INPUT
{
public ushort wVk;
public ushort wScan;
public uint   dwFlags;
public uint   time;
public uint   dwExtraInfo;
}
```

```
[StructLayout(LayoutKind.Explicit)]
struct INPUT
{
[FieldOffset(0)]
public uint type;
// union
[FieldOffset(4)]
public MOUSE_INPUT mi;
[FieldOffset(4)]
public KEYBD_INPUT ki;
}
```

VB.NET:

```
Friend Structure MOUSE_INPUT
   Public dx As System.UInt32
   Public dy As System.UInt32
   Public mouseData As System.UInt32
   Public dwFlags As System.UInt32
   Public time As System.UInt32
   Public dwExtraInfo As System.UInt32
End Structure

Friend Structure KEYBD_INPUT
   Public wVk As System.UInt16
   Public wScan As System.UInt16
   Public dwFlags As System.UInt32
   Public time As System.UInt32
   Public dwExtraInfo As System.UInt32
End Structure

<StructLayout(LayoutKind.Explicit)> _
Friend Structure INPUT
   <FieldOffset(0)> _
   Public type As System.UInt32

   ' union
   <FieldOffset(4)> _
   Public mi As MOUSE_INPUT

   <FieldOffset(4)> _
   Public ki As KEYBD_INPUT
End Structure
```

وسوف نستخدم الـ Methods التالية لتنفيذ عملية التحكم عن بعد:

- الدالة SetCursorPos وتأخذ إحداثيات الـ X والـ Y للـ Mouse حيث يتم وضع مؤشر الـMouse بناء على الإحداثيات المرسلة من الـ Client ، حيث ستستخدم لتحريك مؤشر الـ Mouse في الـ Server .

- الدالة SendInput وسنرسل فيها ثلاثة بارومترات الأول عدد الإدخالات في كل عملية و Reference Value للعملية المدخلة مثلا في حالة الـ Keyboard فإنها تأخذ الـ Virtual Kay Code للـ Button الذي تم الضغط عليه، ولاستخدام الـ SendInput Method سننشئ PressOrReleaseMouseButton Method والـSendKeystroke Method وسنمرر لكل منها مجموعة من المتغيرات حيث سنعرف Instance من الـInput Struct والذي عرفناه في بداية البرنامج وسنمرر له الـ Input Type سواء من الـ Mouse أو الـKeyboard فإذا كان من الـ Mouse نمرر له إحداثيات المؤشر والـKay Button سواء Left أو Right ، أما إذا كان المدخل من الـ Keyboard فسنمرر له الـVirtualKeyCode الخاص بالـ Button الذي تم الضغط عليه ...

C#:

```
[DllImport("user32.dll")]
private static extern uint SendInput(
uint nInputs,     // count of input events
ref INPUT input,
int cbSize        // size of structure
);

public void PressOrReleaseMouseButton(bool Press, bool Left, int X,
int Y)
{
INPUT input = new INPUT();

input.type          = INPUT_MOUSE;
input.mi.dx         = (uint) X;
input.mi.dy         = (uint) Y;
input.mi.mouseData  = 0;
input.mi.dwFlags    = 0;
input.mi.time       = 0;
input.mi.dwExtraInfo = 0;

if (Left)
                {
input.mi.dwFlags = Press ? MOUSEEVENTF_LEFTDOWN :
MOUSEEVENTF_LEFTUP;
                }
else
        {
input.mi.dwFlags = Press ? MOUSEEVENTF_RIGHTDOWN :
MOUSEEVENTF_RIGHTUP;
        }

SendInput(1, ref input, Marshal.SizeOf(input));
        }

[DllImport("user32.dll")]
private static extern void SetCursorPos(int x, int y);
```

```
public void MoveMouse(int x, int y)
{SetCursorPos(x, y);}

public void SendKeystroke(byte VirtualKeyCode, byte ScanCode, bool
KeyDown, bool ExtendedKey)
        {
INPUT input = new INPUT();

input.type          = INPUT_KEYBOARD;
input.ki.wVk        = VirtualKeyCode;
input.ki.wScan      = ScanCode;
input.ki.dwExtraInfo = 0;
input.ki.time       = 0;

if (!KeyDown)
        {
input.ki.dwFlags |= KEYEVENTF_KEYUP;
        }

if (ExtendedKey)
        {
        input.ki.dwFlags |= KEYEVENTF_EXTENDEDKEY;
        }

SendInput(1, ref input, Marshal.SizeOf(input));
        }
```

VB.NET:

```
<DllImport("user32.dll")> _
Private Shared Function SendInput(ByVal nInputs As System.UInt32,
ByRef input As Input, ByVal cbSize As Integer) As System.UInt32
End Function

Public Sub PressOrReleaseMouseButton(ByVal Press As Boolean,
ByVal Left As Boolean, ByVal X As Integer, ByVal Y As Integer)
   Dim input As Input = New Input

   input.type = INPUT_MOUSE
   input.mi.dx = System.Convert.ToUInt32(X)
   input.mi.dy = System.Convert.ToUInt32(Y)
   input.mi.mouseData = 0
   input.mi.dwFlags = 0
   input.mi.time = 0
   input.mi.dwExtraInfo = 0

   If Left Then
      If Press Then
```

```
            input.mi.dwFlags = MOUSEEVENTF_LEFTDOWN
        Else
            input.mi.dwFlags = MOUSEEVENTF_LEFTUP
        End If
    Else
        If Press Then
            input.mi.dwFlags = MOUSEEVENTF_RIGHTDOWN
        Else
            input.mi.dwFlags = MOUSEEVENTF_RIGHTUP
        End If
    End If

    SendInput(1, input, Marshal.SizeOf(input))
End Sub

<DllImport("user32.dll")> _
Private Shared Sub SetCursorPos(ByVal x As Integer, ByVal y As
Integer)
End Sub

Public Sub MoveMouse(ByVal x As Integer, ByVal y As Integer)
    SetCursorPos(x, y)
End Sub

Public Sub SendKeystroke(ByVal VirtualKeyCode As Byte, ByVal
ScanCode As Byte, ByVal KeyDown As Boolean, ByVal ExtendedKey
As Boolean)
    Dim input As Input = New Input

    input.type = INPUT_KEYBOARD
    input.ki.wVk = VirtualKeyCode
    input.ki.wScan = ScanCode
    input.ki.dwExtraInfo = 0
    input.ki.time = 0

    If (Not KeyDown) Then
        input.ki.dwFlags = input.ki.dwFlags Or KEYEVENTF_KEYUP
    End If

    If ExtendedKey Then
        input.ki.dwFlags = input.ki.dwFlags Or
KEYEVENTF_EXTENDEDKEY
    End If

    SendInput(1, input, Marshal.SizeOf(input))
End Sub
```

وأما فيما يتعلق بجلب الـ Bitmap لصورة سطح المكتب الخاص بالـ Server فسنستخدم نفس الخطوات التي عرفناها في المثال السابق ، والآن سنقوم بتصميم شاشات البرنامج وسنحول الـClass السابق إلى ملف Dll حيث سيرفق بالـ Client كـ Reference وسيوضع بجانب الملف التنفيذي الخاص بطرف الـServer :

أولا الـ Server :

سيحتوي برنامج الـServer على الـ Remoting Registration Channel والذي سيقوم بتعريف قناة الاتصال وتوزيع الـ Control Class والذي تم إنشائه حيث سيمكننا من الاتصال مع الـ Remote Class عبر Port محدد ويتم ذلك كما يلي:

C#:

```
TcpChannel chan = new TcpChannel(6600);
ChannelServices.RegisterChannel(chan);
RemotingConfiguration.RegisterWellKnownServiceType(Type.GetTyp
e("ScreenCapture, ScreenCapture"),
"MyCaptureScreenServer",WellKnownObjectMode.Singleton);
```

VB.NET

```
Dim chan As TcpChannel = New TcpChannel(6600)
ChannelServices.RegisterChannel(chan)
RemotingConfiguration.RegisterWellKnownServiceType(Type.GetTyp
e("ScreenCapture, ScreenCapture"),
"MyCaptureScreenServer",WellKnownObjectMode.Singleton)
```

ثانيا الـ Client :

يشبه برنامج الـ Client البرنامج الذي قمنا بإنشائه في المثال السابق لكن سنقوم من خلاله بتمرير قيم الإحداثيات الخاصة بالـ Mouse و الـ Virtual Kay Code الخاص بالـKeyboard Buttons ، وللبدء قم بإنشاء مشروع جديد وكما في الشكل التالي:

سنحتاج إلى استخدام الـ Namespaces التالية:

```
System.Runtime.Remoting
System.Runtime.Remoting.Channels
System.Runtime.Remoting.Channels.Tcp
System.IO
System.Runtime.InteropServices
System.Threading
```

بعد ذلك قم بإضافة التعاريف التالية إلى منطقة الـ Global Declaration في البرنامج:

C#:

```
string URI;
ScreenCapture obj;
TcpChannel chan;
bool connected = false;
```

VB.NET:

```
Dim URI As String
Dim obj As ScreenCapture
Dim chan As TcpChannel
Dim connected As Boolean = False
```

وسوف نستخدم MapVirtualKey Method والتي سنستدعيها من الـ user32.dll لجلب الـ Virtual Kay Code الخاص بالـ Button الذي سيتم الضغط عليه حيث سنستخدمه لمعرفة أزرار التحكم التي تم الضغط عليها إثناء كتابة الأحرف مثل الـShift أو الـ Control أو غيره:

C#:

```
[DllImport("")]
private static extern uint MapVirtualKey(
uint uCode,     // virtual-key code or scan code
uint uMapType   // translation to perform
);
```

VB.NET:

```
<DllImport("user32.dll")> _
Private Shared Function MapVirtualKey(ByVal uCode As
System.UInt32, ByVal uMapType As System.UInt32) As
System.UInt32
End Function
```

ثم سنقوم بكتابة الكود الخاص بتعريف قناة الاتصال في Start Method جديدة وكما يلي:

C#:

```
void start()
{
try
{

URI = "Tcp://"+textBox1.Text+":6600/MyCaptureScreenServer";
chan = new TcpChannel();
ChannelServices.RegisterChannel(chan);
obj = (ScreenCapture)Activator.GetObject(typeof(ScreenCapture),
URI);

connected = true;
timer1.Enabled = true;
textBox1.ReadOnly = true;

this.FormBorderStyle = FormBorderStyle.None;// Full Size Mode
this.WindowState = FormWindowState.Maximized;
textBox1.Visible = false;
menuItem5.Enabled = true;
}
catch (Exception){stop();};
}
```

VB.NET:

```
Private Sub start()
   Try
      URI = "Tcp://" & textBox1.Text &
":6600/MyCaptureScreenServer"
      chan = New TcpChannel
      ChannelServices.RegisterChannel(chan)
obj = CType(Activator.GetObject(GetType(ScreenCapture), URI),
ScreenCapture)
connected = True
timer1.Enabled = True
textBox1.ReadOnly = True
Me.FormBorderStyle = FormBorderStyle.None ' Full Size Mode
Me.WindowState = FormWindowState.Maximized
textBox1.Visible = False
menuItem5.Enabled = True
End Try
```

وسنستدعي الـ Start Method السابقة عند زر الـ Start monitoring في حين سنقوم بإنشاء Method أخرى لإغلاق الاتصال وتدمير الـ Remote Object الذي تم إنشائه و لإيقاف الـ Timer الذي يقوم بجلب صورة سطح المكتب الخاص بالـServer:

C#:

```
void stop()
{
        try
        {
        timer1.Enabled = false;
        textBox1.ReadOnly = false;
        connected = false;

        this.FormBorderStyle = FormBorderStyle.Sizable; // Normal
Size Mode
        this.WindowState = FormWindowState.Normal;
        textBox1.Visible = true;
        this.Width = 584;
        this.Height = 440;

ChannelServices.UnregisterChannel(chan);//to Un Register chan
Channel
        }
catch(Exception){}
}
```

VB.NET:

```
Private Sub Stop()
 Try
timer1.Enabled = False
textBox1.ReadOnly = False
connected = False
Me.FormBorderStyle = FormBorderStyle.Sizable ' Normal Size Mode
Me.WindowState = FormWindowState.Normal
textBox1.Visible = True
Me.Width = 584
Me.Height = 440
ChannelServices.UnregisterChannel(chan) 'to Un Register chan Channel
End Try
```

وتتم عملية جلب صورة سطح المكتب من الـ Remote Object كما يلي حيث سنضع هذا الكود في Timer لتكرير عملية الجلب وعرض الصورة على Picture box:

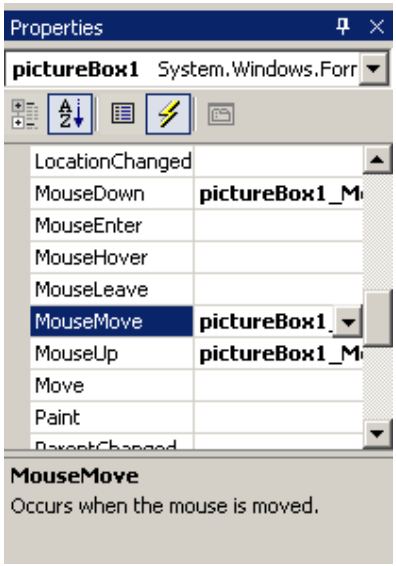
C#:

```
try
{
URI = "Tcp://"+textBox1.Text+":6600/MyCaptureScreenServer";
byte[] buffer = obj.GetDesktopBitmapBytes();
MemoryStream ms = new MemoryStream(buffer);
pictureBox1.Image = Image.FromStream(ms);}
catch (Exception){stop();}
```

VB.NET

```
Try
URI = "Tcp://" & textBox1.Text & ":6600/MyCaptureScreenServer"
Dim buffer As Byte() = obj.GetDesktopBitmapBytes()
Dim ms As MemoryStream = New MemoryStream(buffer)
pictureBox1.Image = Image.FromStream(ms)
Catch e1 As Exception
stop()
End Try
```

ولتمرير إحداثيات الـ Mouse إلى الـ Server سنقوم باستدعاء الـ MoveMouse Method والتـي عرفناهـا فـي الـ remote Class السـابق ويوضـع فـي Mouse Move Event للـPicture box وكما يلي:



C#:

```
private void pictureBox1_MouseMove(object sender,
System.Windows.Forms.MouseEventArgs e)
{
if (connected == true) {obj.MoveMouse(e.X, e.Y);}
}
```

VB.NET:

```
Private Sub pictureBox1_MouseMove(ByVal sender As Object, ByVal
e As System.Windows.Forms.MouseEventArgs)
   If connected = True Then
      obj.MoveMouse(e.X, e.Y)
   End If
End Sub
```

وسـنمرر الـ Mouse Button Event إلـى الـ PressOrReleaseMouseButton Method باسـتدعائها مـن الـup والـ Mouse Down Events وتأخـذ هـذه

الـMethod أربعة باروميترات الأول يمرر له قيمة True أو False فإذا كان الـ Event هو Mouse Down عندها يجب أن تكون قيمتها True والعكس في حالة الـ Mouse up Event ويأخذ الباروميتر الثاني الـ Button الذي تم الضغط عليه بالـ Mouse سواء Left أو Right ويأخذ قيمة True أو False حيث تقارن الـ e.Button مع الـ MouseButtons.Left فإذا كانت True فإن ذلك يدل على أن الـ Button Mouse الذي تم الضغط عليه هو Left والعكس في حالة الـ Right ويأخذ الباروميتر الثالث والرابع إحداثيات الـ X والـ Y للمؤشر:

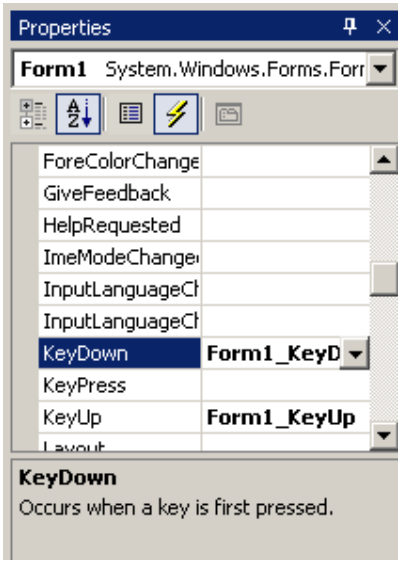
C#:

```
private void pictureBox1_MouseDown(object sender,
System.Windows.Forms.MouseEventArgs e)
        {
if (connected == true)
        {
obj.PressOrReleaseMouseButton(true, e.Button == MouseButtons.Left,
e.X, e.Y);
        }
}

private void pictureBox1_MouseUp(object sender,
System.Windows.Forms.MouseEventArgs e)
        {
if (connected == true)
        {
obj.PressOrReleaseMouseButton(false, e.Button == MouseButtons.Left,
e.X, e.Y);
        }
}
```

VB.NET:

```
Private Sub pictureBox1_MouseDown(ByVal sender As Object, ByVal
e As System.Windows.Forms.MouseEventArgs)
   If connected = True Then
      obj.PressOrReleaseMouseButton(True, e.Button =
MouseButtons.Left, e.X, e.Y)
   End If
End Sub
Private Sub pictureBox1_MouseUp(ByVal sender As Object, ByVal e
As System.Windows.Forms.MouseEventArgs)
   If connected = True Then
      obj.PressOrReleaseMouseButton(False, e.Button =
MouseButtons.Left, e.X, e.Y)
   End If
End Sub
```

ولتمرير الـ Kay Code للـ Buttons الخاص بالـ Keyboard الذي تم الضغط عليه سنستخدم الأحداث Kay Down والـKay up الخاصة بالـ Form وكما يلي:



C#:

```
private void Form1_KeyDown(object sender,
System.Windows.Forms.KeyEventArgs e)
{
if (connected == true)
        {
        e.Handled = true;
obj.SendKeystroke((byte) e.KeyCode, (byte) MapVirtualKey((uint)
e.KeyCode, 0), true, false);
        }
}

private void Form1_KeyUp(object sender,
System.Windows.Forms.KeyEventArgs e)
{
if (connected == true)
        {
e.Handled = true;
obj.SendKeystroke((byte) e.KeyCode, (byte) MapVirtualKey((uint)
e.KeyCode, 0), false, false);
        }
}
```

VB.NET:

```
Private Sub Form1_KeyDown(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyEventArgs)
   If connected = True Then
      e.Handled = True
      obj.SendKeystroke(CByte(e.KeyCode),
CByte(MapVirtualKey(System.Convert.ToUInt32(e.KeyCode), 0)),
True, False)
```

```
    End If
End Sub

Private Sub Form1_KeyUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyEventArgs)
    If connected = True Then
        e.Handled = True
        obj.SendKeystroke(CByte(e.KeyCode),
CByte(MapVirtualKey(System.Convert.ToUInt32(e.KeyCode), 0)),
False, False)
    End If
End Sub
```

قمنا بالبداية بإيقاف فاعلية الـ Buttons باستخدام الـe.Handled = True وبعد ذلك مررنا الـ Kay code الذي سيتم الضغط عليه إلى الـ SendKeystroke Method وتأخذ هذه الـMethod أربعة باروميترات على النحو التالي، الأول ويأخذ الـ Byte للـ Virtual Kay code الذي تم الضغط عليه ويأخذ الثاني الـ Scan Code لحالة الـKeys سواء Shift Kay أو Control Kay أو إذا كنت تحت الـ Caps look Mode أو غيره من أزرار التحكم الخاصة بالـ Keyboard ويتم ذلك بعد تمرير قيم الـ unassigned Integer Code الراجع من الـe.KayCode إلى الـ MapVirtualKey التي عرفناها في بداية البرنامج ...

أما الباروميتر الثالث فيأخذ قيمة True في حالة كان الحدث هو KayDown و False في حالة كان الحدث هو KayUp ويأخذ الباروميتر الرابع قيمة false حيث لن تستخدم الـExtended Keys أي الضغط المطول...

وهكذا بينا كيفية بناء نظام لتحكم عن بعد باستخدام الـ Remoting وطرق استخدام الـSerialization في الدوت نيت وطرق بناء وتصميم الأنظمة الموزعة، سنتحدث في الفصل التالي عن طرق استخدام الـ SMTP لإرسال Emil's والـ POP3 لقراءة الـ Mail Box في الدوت نيت...

Chapter 16
SMTP & POP3 Programming

- SMTP Protocol
    1. SMTP Concept
    2. Using SMTP in Dot Net
    3. Advanced SMTP Programming

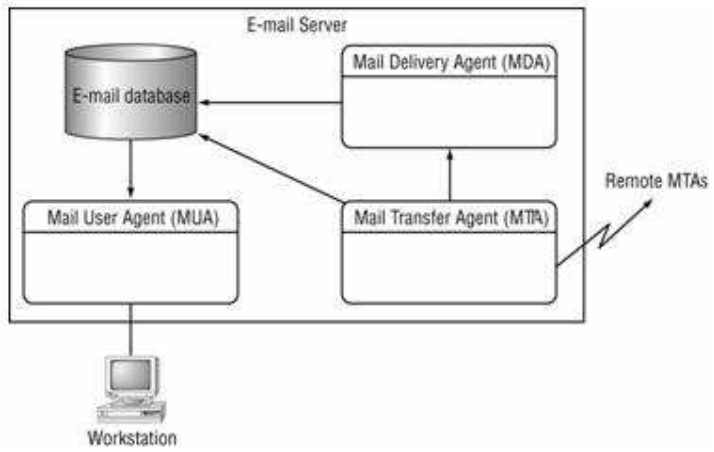
- POP3 Protocol
    1. POP3 Concept
    2. Using POP3 in Dot Net

16: SMTP & POP3 Programming :

سوف نتحدث في هذا الجزء عن برمجة الـSMTP والمسئول عن إرسال الرسائل عبر البريد الإلكتروني إلى الـMail Server و الـPOP3 والمسئول عن عملية الـ Download لرسالة من الـMail Server إلى جهاز الزبون ...

أولا: SMTP – Simple Mail Transfer Protocol Programming

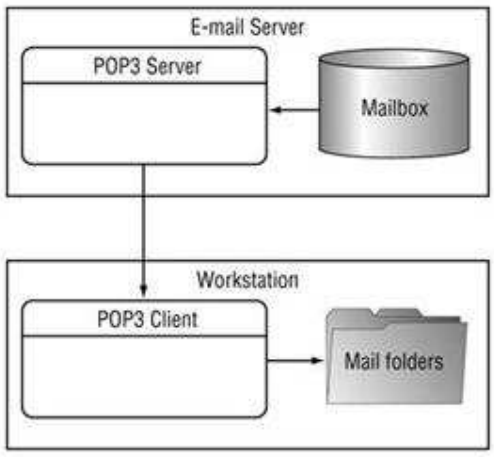
من المعروف أن الـMail Server يقوم بتجزئة عمليات إرسال و استقبال البريد الإلكتروني عبر الإنترنت إلى ثلاثة أجزاء وهي كما في الشكل التالي :



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

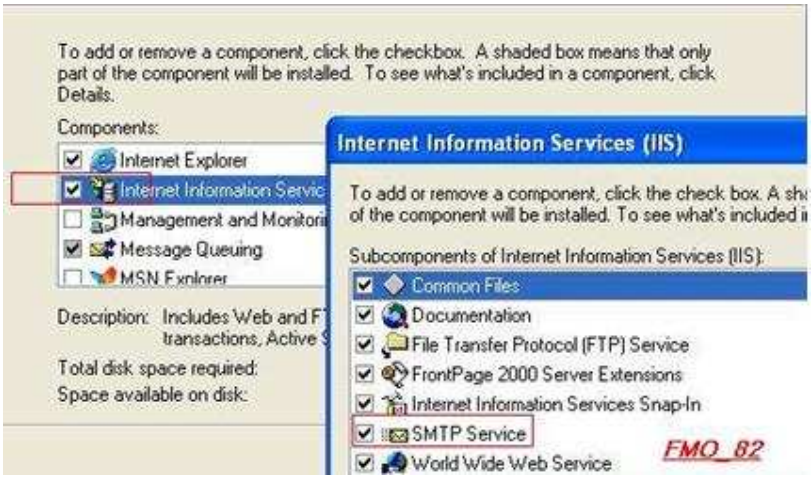
Message Delivery Agent -MDA و المسئول عن عمليات الـ filtering والتأكد من وصول الرسالة

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 الـSMTP Simple Mail Transfer Protocol بشكل أساسي في الـMTA أي عمليات إرسال Outgoing وتوصيل Incoming الرسائل .

لتطبيق يجب أولا التأكد من أنك تملك حساب SMTP من الـInternet Provider الخاص بك تستطيع تجربة الـAccount الخاص بك من خلال برنامج الـOutlook Express الموجود مع الـWindows إذا كنت لا تملك حساب SMTP تستطيع تجربة البرنامج من خلال إنشاء Virtual SMTP Server عن طريق الـIIS وذلك بتثبيتها من : Control Panel >> Add/Remove Programs تأكد من تفعيل كل من الـIIS والـSMTP كما في الشكل التالي:



ثم إعداد الـServer من الـIIS كما في الشكل التالي :

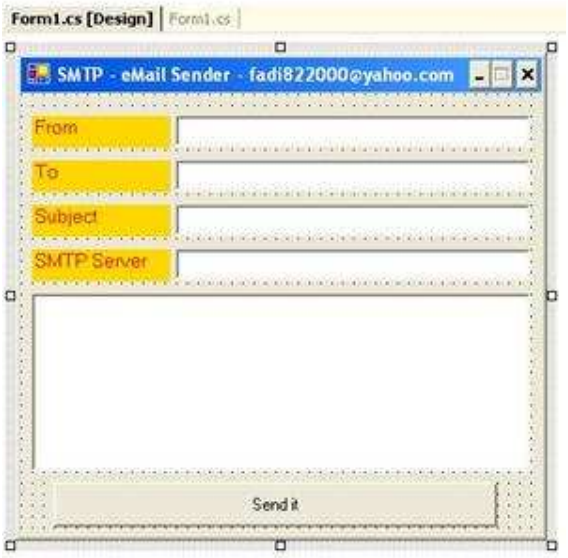
تـــدعم الـــدوت نيـــت اســـتخدام بروتوكـــول الـــSMTP مـــن خـــلال Namespace System.Web.Mail و تحتوي على الكلاس SmtpMail والتي من خلالها نستخدم الدالة Sendوالتي تستخدم لإرسال الرسالة عبر الـ Port 25 وهو الـPort المخصص لبروتوكول SMTP و تعتبر الدالة Send " overloaded Method " حيث تأخذ عدة أشكال إذ بإمكانك

استخدامها مع باروميتر واحد إلى أربعة باروميترات ، وبشكل افتراضي نستخدم الباروميترات التالية :

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

الباروميتر الأول يوضع فيه عنوان المرسل والثاني يوضع فيه عنوان المرسل إليه و الباروميتر 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 Namespace ، (إذا لم تظهر لديك Mail. قم بإدراج System.Web Namespace إلى الـReferences) ثم قم بكتابة الكود التالي :

C#:
```
using System.Web.Mail;
```

VB.NET:
```
imports System.Web.Mail;
```

وتتم عملية إرسال الرسالة كما يلي:

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، لكن يجب التأكد من تفعيل الـSMTP من الـIIS كما ذكر في السابق والـ IP الخاص بالـSMTP (والذي تم تعريفه مسبقا في SMTP Virtual Server) بالـ SMTP Server Textbox ، يجب التأكد أيضا من الـSMTP 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 الـSMTP Virtual Server أو مع Microsoft Exchange Server2003 أما إذا كنت تستخدم Exchange Version 5 أو 5.5 فسوف تحصل على الـException السابق الذكر .

الجزء الأكثر تقدم: SMTP Advanced Programming

يعتبر المثال السابق مثال بسيط لإرسال رسائل عبر الـ SMTP باستخدام CDO2 ، وفي العادة عند إنشاء برامج مثل برنامج الـOutlook يتم استخدام الـHTML Format بالإضافة إلا إمكانية إرسال ملحقات وطبعا يعطيك عدة خيارات لإرسال و استقبال البريد الإلكتروني هل باستخدام الـHTTP أو الـPOP3 ...

سوف نقوم بإنشاء برنامج بسيط يقوم بإرسال واستقبال البريد الإلكتروني باستخدام الـSMTP والـ POP3 بنسبة لاستخدام الـPOP3 فيجب أن يتوفر لديك حساب POP3 أو أن تقوم بتثبيت الـService POP3 والتي تأتي مع Windows Server أو أن تستخدم Microsoft Exchange Server على جهازك وإعداده بحيث يستخدم الـPOP3 إذ عندها سوف تحتاج لوجود Domain Controller مثبت على الجهاز و Windows Server بالإضافة إلى تثبيت الـActive Directory عليه.

قدمت الدوت نيت دعم ممتاز لاستخدام هذه الخواص وذلك من خلال Namespace System.Web.Mail وباستخدام الكلاس MailMessage لدعم الـHTML Format و الكلاس MailAttachment لدعم إمكانية إرسال ملحقات مع الرسالة ولكن لبرمجة الـPOP3 يلزم استخدام System.Net.Socket Namespace و System.Net و System.IO حيث يتم عمل Session خاص مع الـServer للقيام بعملية تفحص وجود رسائل جديدة وفي حالة وجودها يقوم بتعبئتها في List Box أو Treelist حسب الحاجة وعند الضغط على إحداها يقوم الـClient بعمل Download لرسالة من الـMail Server ولعمل Advanced SMTP eMail Sender قم بأخذ Object من الكلاس MailMessage كما يلي :

C#:

```
using System.Web.Mail;

try
{
 MailMessage mm = new MailMessage();
  mm.From = textBox1.Text;
    mm.To = textBox2.Text;
 // mm.Cc =
 // mm.Bcc =
  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 (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 الـHTML وهذا يمكنك من وضع أي لون أو حجم أو أي شيء يمكن عمله باستخدام الـHTML ، 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);
```

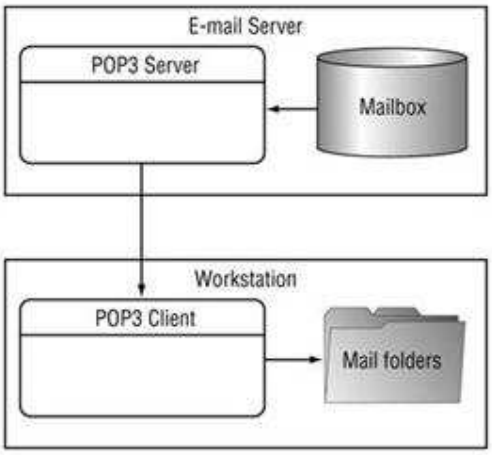
VB.NET:

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

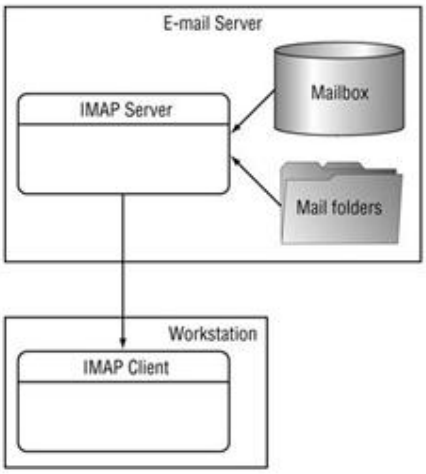
وهكذا قد انتهينا من برنامج الـSMTP بشكل كامل ، طبعا عملية ال Design وغيرها تعتمد على حسب ذوق وذكاء وخبرة المبرمج.

ثانيا : POP3- Post Office Protocol Version 3 Programming

كما تحدثنا سابقا فإن وظيفة برتوكول الـPOP3 والذي يعمل في جزء الـ MUA - Mail User Agent على Port 110 ضمن برتوكول الـTCP تكمن في كونه المسئول عن عملية توصيل الرسالة إلى الزبون Client من خلال عمل Download لها من الـMail Server حيث تحفظ الرسائل في الـMail Folder والموجود أساسا في جهاز الـClient أنظر إلى الشكل التالي:



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



لاحظ أن الـMail Folder يقع ضمن Mail Server ويتم قراءته بعد التحقق Authentication من اسم المستخدم وكلمة المرور لكن كما قلنا فإن مشكلته تكمن في محدودية سعة الـMail Folder لذا ينصح لشركات الكبيرة استخدام الـPOP3 كونه غير محدود السعة فالذي يتحكم في السعة هو الـClient ولا دخل لـ Mail Server Administrator بها.

وبما أننا قررنا اعتماد الـPOP3 لعملية قراءة الرسائل سوف نبدأ ببرمجته إذ يلزم الأمر استخدام System.Net.Socket Namespace و System.Net و System.IO حيث يتم عمل Session خاص مع الـServer باستخدام الـSocket 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 الـClient بعمل Download لرسالة من الـ Mail Server إلى الـMail Folder ثم عرضها في Textbox.
ولتطبيق قم بإنشاء New Form جديد كما يظهر في الشكل التالي :

ثم قم بإضافة Namespaces التالية :

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

VB.NET:
```
imports System.Net
imports System.Net.Socket
imports System.IO
```

لاحظ انه يتم التعامل مع الـSocket والـStream لإنشاء Session مع الـServer باستخدام برتوكول الـTCP وقراءة الرسالة من الـPOP3 Server، ويستخدم الـ Stream Socket السابق لتمرير الـ Commands إلى الـServer وهي:

USER (USER USER_NAME) PASS (PASS PASSWORD)	إدخال الـ User Name والـ Password لإجراء عمليات التحقق الـ Authentication
RETR (RETR Message_Number)	وتستخدم لجلب الرسالة من الـ POP3 وترجع كـ Byte Array ...
STAT	لمعرفة وجود رسائل جديدة وعدد الرسائل الجديدة في الـ inbox
QUIT	إغلاق الـ Session مع الـ POP3 Server

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

C#:

```
public TcpClient Server;// اشتقاق كائن من برتوكول التي سي بي وذلك بهدف إنشاء الجلسة
public NetworkStream NetStrm;// سوف نستخدمه لإرسال معلومات المستخدم
public StreamReader  RdStrm; //  لقراءة المعلومات الواردة من البوب 3
public string Data; // لاستخدامها في تمرير الأوامر إلى السيرفر
public byte[] szData; //  لتخزين البيانات الواردة من البوب 3
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 الـConnect Button قم بإضافة الكود التالي :

C#:

```
// create server POP3 with port 110
Server = new TcpClient(POPServ.Text,110);

try
{
// initialization
NetStrm = Server.GetStream();
RdStrm= new StreamReader(Server.GetStream());
Status.Items.Add(RdStrm.ReadLine());

// Login Process
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 الـDisconnect Button :

C#:

```
// 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 الـRead 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
```

وهكذا بينا كيفية عمل الـPOP3 وبرمجته في الدوت نت وهذا مثال بسيط تستطيع البدء منه لعمل مشروع كامل شبيه بالـOutlook الخاص بميكروسوفت حيث تستطيع استخدام ملف الـDLL 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 ،

سيتم الحديث في الفصل التالي عن الـ FTP وطرق التعامل معه في بيئة الدوت نيت.

Chapter 17
FTP Programming

- Introduction to FTP – File Transfer Protocol

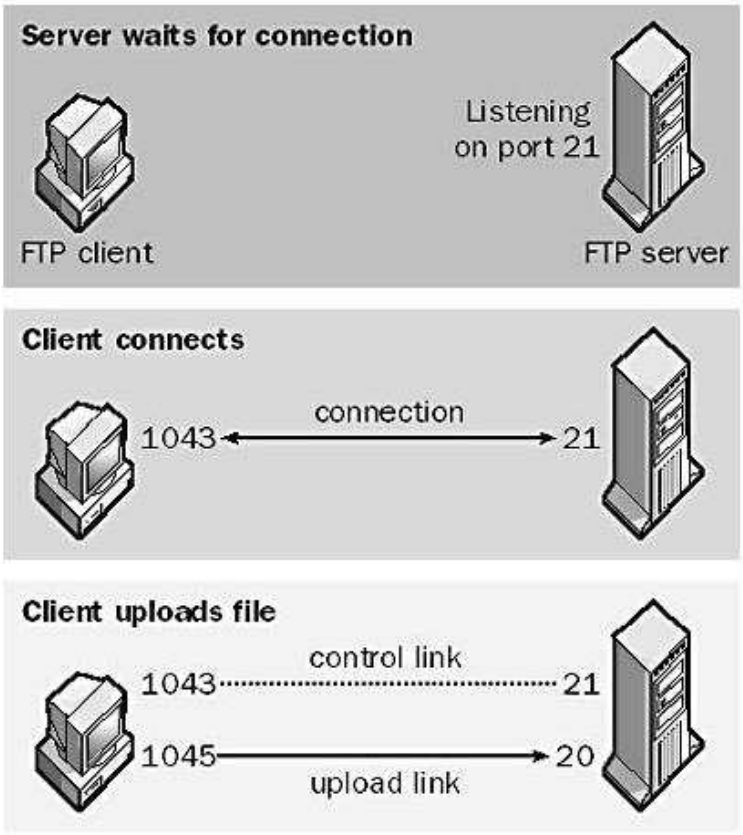
- Create a Simple Application to Transfer Files By Using COM Components

- Create a Simple Application to Transfer Files By Using Web Classes Components

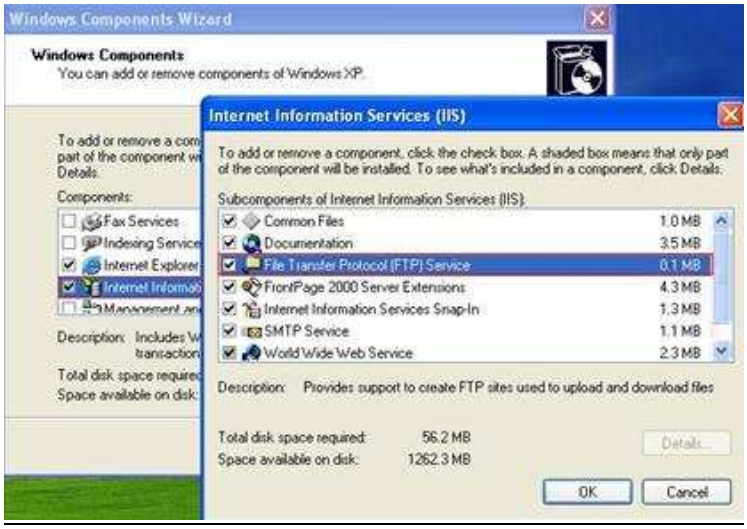
- Create a Simple Application to Transfer Files By Using Socket Programming & Streaming Libraries

17 FTP – File Transfer Protocol Programming :

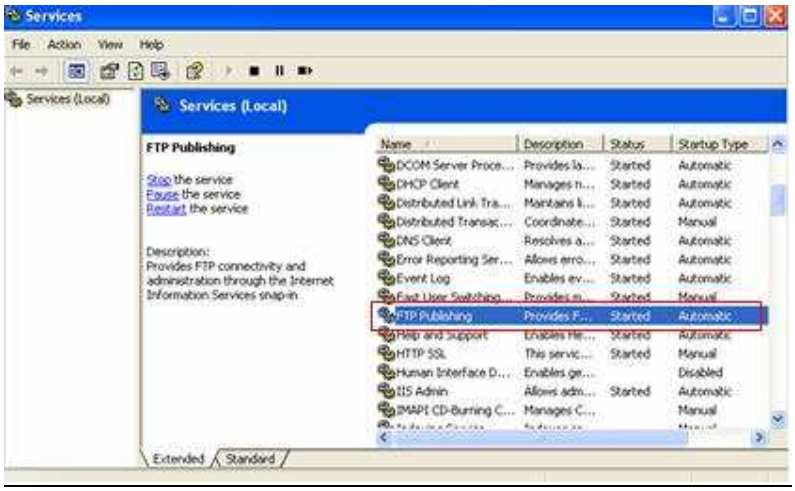
سوف نبدأ هنا بشرح برتوكول أخر من برتوكولات الـApplication Layer وهو برتوكول الـFTP والذي يستخدم بشكل أساسي في عملية الـ downloading و الـ uploading الملفات من وإلى الـFTP Server وكالعادة في اغلب برمجيات الشبكات و التي تعتمد على وجود Client/Server حيث يقوم الـServer بتصنت على الـPort المخصص للـ FTP وهو الـPort 21 باستخدام الـTCP Connection Oriented Protocol حيث يبقى الـServer بوضع الانتظار لورود طلب من الـClient بإنشاء Session معه وبعد إجراء عمليات التحقق Authentication والتأكد من الصلاحيات يتم الموافقة على البدء بالجلسة حيث يتم تحديد رقم الـPort والذي سوف يتم استقبال البيانات من خلاله ويتم الإرسال إلى جهاز الزبون عبر الـPort 20 في الـServer وتتضح هذه العملية كما في الشكل التالي :



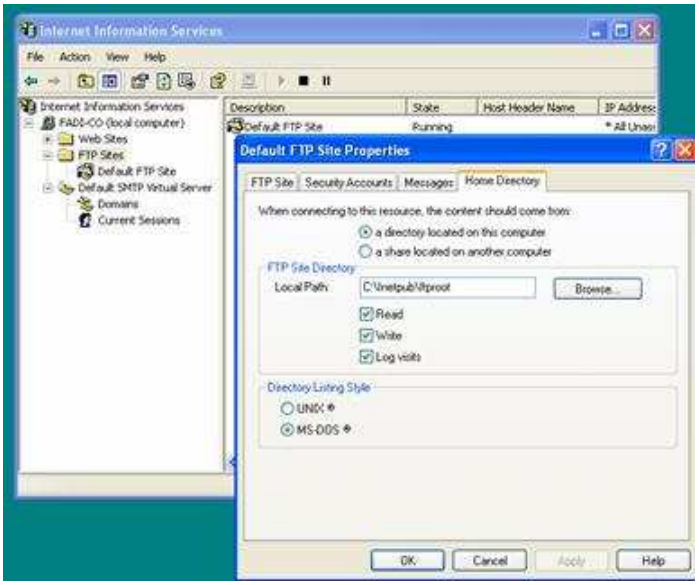
ملاحظة: لتفعيل خدمة الـFTP لديك بحيث يعمل جهازك كـ FTP Server يجب أولا التأكد من أن الـFTP Services مثبتة لديك مع الـIIS و كما يظهر في الشكل التالي :



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

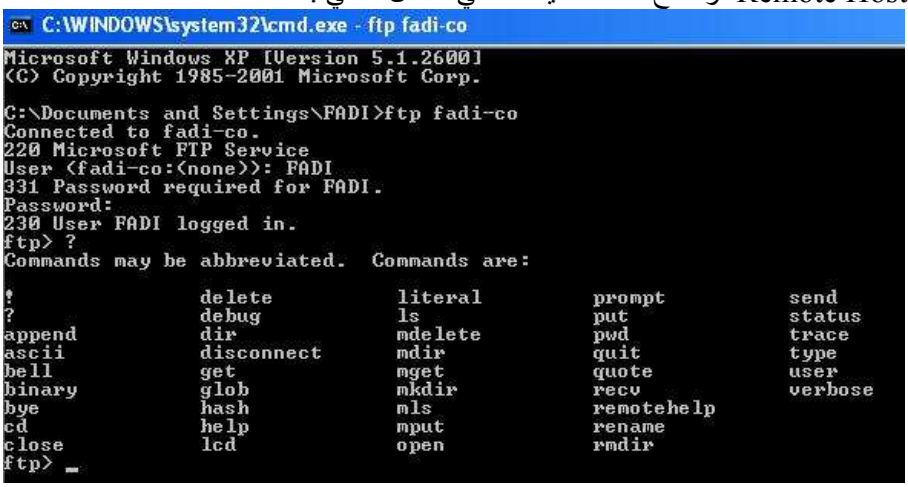


ثم التأكد منه في الـIIS بحيث يظهر كما في الشكل التالي :



أولا : FTP Commands

تشبه عملية الاتصال و الاستخدام للـ FTP عملية الـTelnet إلى حد كبير حيث يدعم برتوكول الـFTP مجموعة من الأوامر والتي يتم من خلالها عملية التخاطب مع الـServer أو مع الـRemote Host وتضح هذه العملية كما في الشكل التالي :



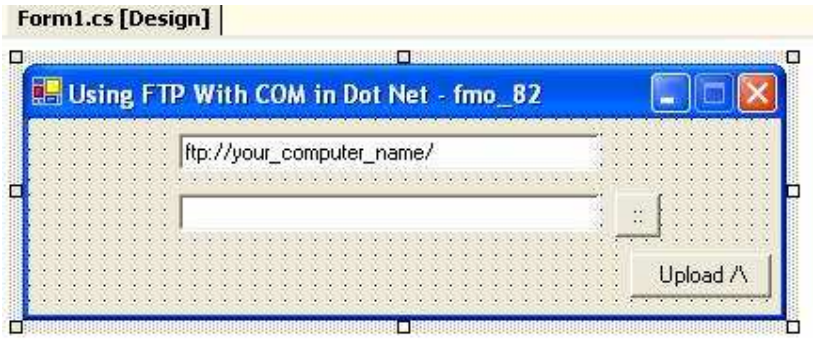
وهنا شرح لأهم الـFTP Commands:

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

	Active إذ انه في حالة الـPassive يتم تفعيل الاتصال فقط في حالة ورود أو رفع أي ملف من و إلى الـServer .
STAT أو Status	لفحص حالة الاتصال و uploading & Downloading
Delete , cd , mkdir , rename …	وهي كما هو متعارف عليه في التعامل مع الملفات و المجلدات في نظام الـDOS
QUIT أو Close	لإنهاء الجلسة مع الـRemote Host

ثانيا : التعامل مع الـFTP في الدوت نيت باستخدام COM Components :

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



ثم قم بإضافة Namespaces التالية :

C#:
```
using System.IO;
using System.Reflection;
```

VB.NET:
```
imports System.IO
imports System.Reflection
```

ثم إضافة الكود التالي إلى الـUpload 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 الـITC من خلال الـ Type Class والموجود ضمن Namespace System.Reflection ثم عرفنا Array من النوع Object وذلك لاستخدامها في تمرير اسم الملف و الـFTP Server إلى الدالة InvokeMember والموجودة ضمن الـ ITC Control Object ...

سوف تجد الملف الذي سيتم رفعه في المجلد :
C:\Inetpub\ftproot

ثالثا : التعامل مع الـFTP فى الدوت نيت باستخدام الـWeb Class :

يمكــن برمجــة الــFTP باســتخدام Class web والموجــودة ضــمن Namespaces System.Net وتشـبه عمليـة التعامـل معـه كمـا فـي التعامـل مـع الـ WebRequest و الــwebResponse Classes و التــي تعاملنــا معهــا فــي برمجــة الــHTTP حيـث يمكننــا الاستفادة منها لتعامل مع الـFTP Protocol وهي كما يلي :

- **WebClient** إذ تم دعم dot net Framework 2 استخدام الكلاس WebClient والذي يدعم التعامل مع الـFTP والذي يتم استدعائه من System.Net Namespaces 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 الـFTP 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 الـServer ويتم تعريفها كما يلي:

C#:

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

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.Socket
  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
```

- **FtpWebStream** ويستخدم لتعريف الـStream والذي سوف يستخدم لعملية النقل ويعرف بشكل مبدئي كما يلي:

C#:

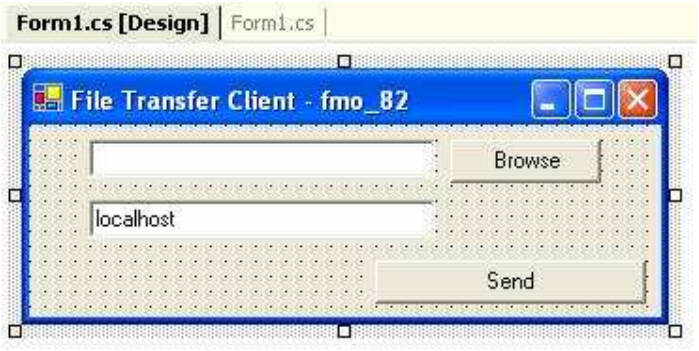
```
using System;
using System.IO;
using System.Net.Socket;

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.Socket
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 الـStream والـSocket:

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



سوف نستخدم Namespaces التالية :

C#:
```
using System.IO;
using System.Net;
using System.Net.Socket;
using System.Text;
```

VB.NET:
```
imports System.IO
imports System.Net
imports System.Net.Socket
imports System.Text
```

في الـSend 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 بـ Stream Object وحتى نستطيع إرساله عبر الـSocket لابد من تحويله إلى مصفوفة من النوع Byte وقمنا بتسميته بـ FileBuffer ثم تعبئته باستخدام الدالة Read والموجودة ضمن FileStream وبعد ذلك قمنا بإنشاء اتصال مع الـServer باستخدام الـTCP Connection حيث تم إرسال محتويات الـFileBuffer إلى الـServer باستخدام NetworkStream Class ...

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

سوف نستخدم الـ Namespaces التالية :

C#:

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

VB.NET:

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

ثم إضافة الـMethod التالية وليكن اسمها handlerThread وكما يلي :

C#:

```
public void handlerThread()
{
Socket handlerSocket = (Socket)alSocket[alSocket.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(alSocket(alSocket.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 دالة أخرى جديدة وذلك لفتح TCP Connection على الـ Port 8880 وهو افتراضي والتصنت عليها وليكن اسمها listenerThread وكما يلي :

C#:

```
public void listenerThread()
{
TcpListener tcpListener = new TcpListener(8880);
        tcpListener.Start();
        while(true)
        {
        Socket handlerSocket = tcpListener.AcceptSocket();
        if (handlerSocket.Connected)
                {
        lbConnections.Items.Add(handlerSocket.RemoteEndPoint.ToStri
ng() + " connected.");
        lock (this)
                {
                alSocket.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
                alSocket.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();

alSocket = new ArrayList();

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

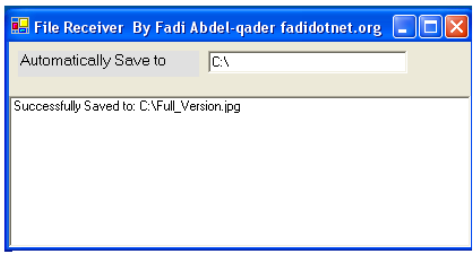
VB.NET:

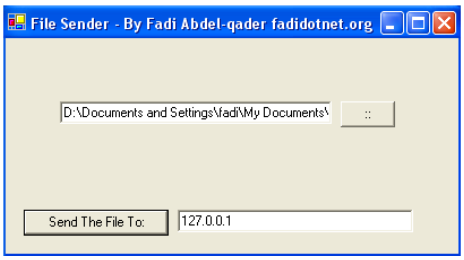
```
Private Sub Form1_Load(ByVal sender As Object, ByVal e As
System.EventArgs)
   Dim IPHost As IPHostEntry =
Dns.GetHostByName(Dns.GetHostName)
   lbConnections.Text = "My IP address is " +
IPHost.AddressList(0).ToString
   alSocket = New ArrayList
   Dim thdListener As Thread = New Thread(New
ThreadStart(listenerThread))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

باستخدام الـThread تم تنفيذ الـ listenerThread Method والتي قمنا فيها بتعريف الـTcpListener وتفعيله على الـ Port 8880 حيث سيتم قبول أي طلب يأتي من الـClient على هذا الـ Port وبعد ذلك استدعاء الدالة handlerThread والتي سيتم فيها استقبال الـData Stream وتخزينها في Byte Array ثم قراءتها وتخزينها في المكان المحدد وباستخدام الـfileStream.Write حيث مررنا له الـStream والذي يحتوي على اسم الملف thisRead والـDataByte Array .

خامسا استخدام الـ Serialization والـ FileStream Class لإرسال ملف من جهاز إلى آخر:

يمكننا الاستفادة من ميزة الـSerialization لإرسال اسم الملف مع محتوياته إذ يتم تخزين الملف على الطرف الآخر بناءا على اسم الملف الأصلي ، وسيكون الشكل العام للبرنامج كما يلي كمثال:





أولا برنامج الإرسال:

وسنستخدم فيه الـ FileStream Class لتحويل الملف المراد إرساله إلى Byte Array ، كما وسنستخدم الـFileInfo Class لجلب اسم الملف وحتى يتم إرساله كـ Serialization Object باستخدام الـBinaryFormatter Class وتتم هذه العملية كما يلي:

C#

```
string FileName;
.
.
openFileDialog1.ShowDialog();
textBox1.Text = openFileDialog1.FileName;
FileInfo TheFile = new FileInfo(textBox1.Text); // Get The File Name
FileName  = TheFile.Name;
.
.
FileStream fs = new FileStream(textBox1.Text,FileMode.Open);
byte[] buffer = new byte[fs.Length];
int len = (int)fs.Length;
fs.Read(buffer,0,len);
fs.Close();
BinaryFormatter br = new BinaryFormatter ();
TcpClient myclient = new TcpClient (text_IP.Text,7000);
NetworkStream myns = myclient.GetStream ();
br.Serialize (myns,FileName);
BinaryWriter  mysw = new BinaryWriter (myns);
mysw.Write(buffer);
mysw.Close ();
myns.Close ();
myclient.Close ();
```

VB.NET:

```
Private FileName As String
.
.
```

```
openFileDialog1.ShowDialog()
textBox1.Text = openFileDialog1.FileName
Dim TheFile As FileInfo = New FileInfo(textBox1.Text) ' Get The File
Name
FileName = TheFile.Name
.
.
Dim fs As FileStream = New
FileStream(textBox1.Text,FileMode.Open)
Dim buffer As Byte() = New Byte(fs.Length - 1)
Dim len As Integer = CInt(fs.Length)
fs.Read(buffer,0,len)
fs.Close()
Dim br As BinaryFormatter = New BinaryFormatter ()
Dim myclient As TcpClient = New TcpClient (text_IP.Text,7000)
Dim myns As NetworkStream = myclient.GetStream ()
br.Serialize (myns,FileName)
Dim mysw As BinaryWriter = New BinaryWriter (myns)
mysw.Write(buffer)
mysw.Close ()
myns.Close ()
myclient.Close ()
```

ثانيا برنامج الاستقبال:

C#:

```
NetworkStream myns;
TcpListener mytcpl;
Socket mysocket;
Thread myth;
BinaryReader bb;

void File_Receiver()
{
mytcpl = new TcpListener (7000);
mytcpl.Start ();
mysocket = mytcpl.AcceptSocket ();
myns = new NetworkStream (mysocket);
BinaryFormatter br = new BinaryFormatter ();
Object op= br.Deserialize (myns); // Deserialize the Object from Stream

bb = new BinaryReader (myns);
byte[] buffer = bb.ReadBytes(Buffer_Size);
FileStream fss = new FileStream(@textBox1.Text + (string) op,
FileMode.CreateNew, FileAccess.Write);
fss.Write(buffer,0,buffer.Length);
fss.Close();
```

```
mytcpl.Stop();
listBox1.Items.Add ("Successfully Saved to: " + textBox1.Text +
(string) op);
if (mysocket.Connected ==true)
        {
        while (true)
        {
        File_Receiver();
        }
        }
}
```

VB.NET:

```
Private myns As NetworkStream
Private mytcpl As TcpListener
Private mysocket As Socket
Private myth As Thread
Private bb As BinaryReader
Private Sub File_Receiver()
mytcpl = New TcpListener (7000)
mytcpl.Start ()
mysocket = mytcpl.AcceptSocket ()
myns = New NetworkStream (mysocket)
Dim br As BinaryFormatter = New BinaryFormatter ()
Dim op As Object
op= br.Deserialize (myns) ' Deserialize the Object from Stream
bb = New BinaryReader (myns)
Dim buffer As Byte() = bb.ReadBytes(Buffer_Size)
Dim fss As FileStream = New FileStream(textBox1.Text + CStr(op),
FileMode.CreateNew, FileAccess.Write)
fss.Write(buffer,0,buffer.Length)
fss.Close()
mytcpl.Stop()
listBox1.Items.Add ("Successfully Saved to: " & textBox1.Text +
CStr(op))

If mysocket.Connected =True Then
        Do While True
File_Receiver()
        Loop
End If
End Sub
```

وهكذا بينا طريقة عمل بروتوكول الـ FTP وطرق برمجته في بيئة الدوت نيت ، سيتم الحديث في الجزء التالي عن طرق وأنواع التشفير واستخدامها في الـ Network Security Programming .

Part 5
Network Security Programming

Chapter 18
Cryptography & Network Security Programming

- Cryptography in Dot Net:

  - Symmetric Encryption
    - DES (Data Encryption Standard)

  - Asymmetric Encryption
    - RSA (Rivest Shamir Adleman)
    - Digital Signature Algorithms
    - Hashing Algorithms

  - Using Security Encryption Standards With Network Applications
    - Create an Advanced RSA Client Server Chat System.

: Network Security Programming : 18

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 الـ 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 لكنها ليست آمنة كطريقة الغير المتماثلة ودعمت الدوت نيت التشفير المتماثل بمجموعة من الـClasses Algorithms وهي:

- الكلاس الذي يدعم التشفير باستخدام الـ 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 لكن 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 الشفرة ناهيك عن الحجم الهائل التي قد تحجزه من الـNetwork Bandwidth ..وتم حل هذه المشكلة بجعل عملية التشفير تتم على مستوى الـStream نفسه ويستخدم لهذه العملية الـ **CryptoStream Class** حيث يتم استخدام مفتاحين لتشفير مفتاح التشفير Encryption Kay ومفتاح لفك التشفير Installation Victor Decryption **IV** ويشترط استخدام نفس المفتاحين في عملية التشفير وفك التشفير ويستخدم الكلاس السابق مع الـ**MemoryStream** أو **FileStream** حيث نمرر له الـStream Data ونوع التشفير سواء DES أو TripleDES أو RC2 ومن الحلول الأخرى استخدام التشفير على مستوى الـ Socket حيث يتم تشفير القناة نفسها ويسمى

هذا النوع الـ SSL – Socket Secure Layer وهو من أشهر أنواع التشفير الذي يستخدم في الإنترنت بشكل خاص وخاصة في إجراء العمليات البنكية ، وكمثال سوف نستخدم الـTripleDES إذ يتكون كلا المفتاحين من 16 Bits وكما في المثال التالي:

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
```

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:

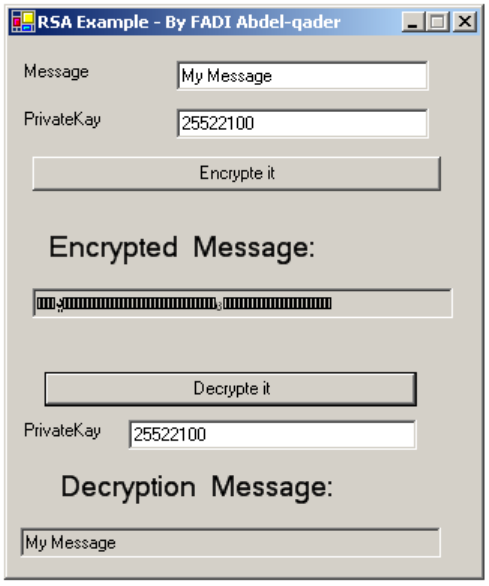
1- RSACryptoServiceProvider
2- DSACryptoServiceProvider for Digital Signature Algorithm

أولا: RSACryptoServiceProvider:

ويستخدم في إجراء التشفير وفك التشفير الغير متماثل وهو non inherited Class في البداية سوف ننشئ instance جديد من الـRSACryptoServiceProvider وذلك لتوليد المفتاح العام والخاص ونرفق المفتاح العام مع الرسالة ومن ثم يقوم المستلم بفك الرسالة باستخدام المفتاح الخاص لاحظ الشكل التالي:

تشفر الرسالة باستخدام مفتاح عام وخاص (العام يرسل مع الرسالة) والخاص يتفق عليه الطرفان المرسل والمستقبل

سنقوم الآن بإنشاء مشروع تشفير وفك تشفير يستخدم طريقة الـRSA Algorithm ، سيكون الشكل العام للبرنامج كما يلي:



سنقوم بالبداية بإنشاء Tow Methods الأولى نمرر لها الـString Plain Message والـPrivateKay وذلك لتشفير الرسالة بناءا عليه ، وأخرى لفك التشفير ونمرر لها الرسالة المشفرة كـ Array Byte والمفتاح الخاص كـ String ، وسنستخدم الـClass RSACryptoServiceProvider لتشفير الرسالة باستخدام الـRSAAlgorithms ، ونمرر له الـ Private Kay كـ Object CspParameters وباستخدام الـ EncryptMethod والـ DecryptMethod ضمن الـClass RSACryptoServiceProvider نتم عملية التشفير وفك التشفير ونمرر لها الرسالة المراد تشفيرها أو فك تشفيرها بالإضافة إلى قيمة True أو False لتفعيل طريقة التشفير كـFOAEP ويدعم هذا الأسلوب إصدارات الـ Windows XP وما بعدها ، لذلك يفضل اختيار False دائما ولضمان التوافقية مع الإصدارات الأقدم من نظام التشغيل ...

C#:

```
public byte[] Encrypt(string strData,CspParameters PrivateKey)
{
RSACryptoServiceProvider rsa = new
RSACryptoServiceProvider(PrivateKey);
byte[] data = rsa.Encrypt(Encoding.Unicode.GetBytes(strData), false);
return data;
}

public byte[] Decrypt(byte[] en_data,CspParameters PrivateKey)
{
RSACryptoServiceProvider rsa = new
RSACryptoServiceProvider(PrivateKey);
byte[] data = rsa.Decrypt(en_data, false);
return data;
}
```

VB.NET:

```
Public Function Encrypt(ByVal strData As String, ByVal PrivateKey
As CspParameters) As Byte()
   Dim rsa As RSACryptoServiceProvider = New
RSACryptoServiceProvider(PrivateKey)
   Dim data As Byte() =
rsa.Encrypt(Encoding.Unicode.GetBytes(strData), False)
   Return data
End Function

Public Function Decrypt(ByVal en_data As Byte(), ByVal PrivateKey
As CspParameters) As Byte()
   Dim rsa As RSACryptoServiceProvider = New
RSACryptoServiceProvider(PrivateKey)
   Dim data As Byte() = rsa.Decrypt(en_data, False)
   Return data
End Function
```

ولاستخدام الـ Methods السابقة يجب أولا إنشاء Object من Instance الـ Class CspParameters قم نمرر المفتاح العام إلى الـ KeyContainerName الخاص بالـ Instance Object ضمن الـ Proparity وكما يلي: CspParameters Class

C#:

```
public byte[] mymsg;

Try //لتشفير
{
CspParameters cp = new CspParameters();
cp.KeyContainerName = text_private.Text;
```

```
mymsg = Encrypt(text_msg_to_send.Text,cp);
text_encrypted.Text = Encoding.Unicode.GetString(mymsg);
}
catch (Exception ex){MessageBox.Show(ex.Message);}


try//لفك التشفير
{
CspParameters cp = new CspParameters();
cp.KeyContainerName = text_private1.Text;

byte[] my_msg = Decrypt(mymsg,cp);
text_decrypted.Text = Encoding.Unicode.GetString(my_msg);
}
catch (Exception ex){MessageBox.Show(ex.Message);}
```

VB.NET:

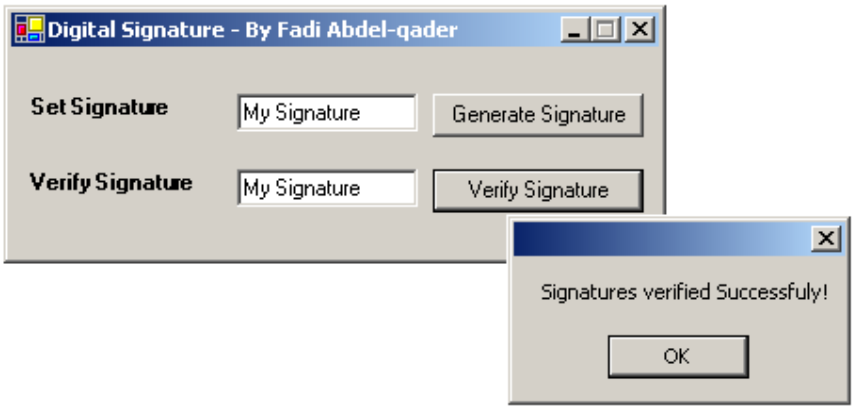
```
Public mymsg As Byte()

Try 'لتشفير
Dim cp As CspParameters = New CspParameters
cp.KeyContainerName = text_private.Text

mymsg = Encrypt(text_msg_to_send.Text,cp)
text_encrypted.Text = Encoding.Unicode.GetString(mymsg)
Catch ex As Exception
        MessageBox.Show(ex.Message)
End Try

Try 'لفك التشفير
Dim cp As CspParameters = New CspParameters
cp.KeyContainerName = text_private1.Text

Dim my_msg As Byte() = Decrypt(mymsg, cp)
text_decrypted.Text = Encoding.Unicode.GetString(my_msg)
Catch ex As Exception
        MessageBox.Show(ex.Message)
End Try
```

لاحظ أن من الفروق بين استخدام أسلوب التشفير الـ RSA والـDES هي إمكانية الـRSA من استخدام مفتاحين كما يختلف مفتاح التشفير عن مفتاح فك التشفير ، وحتى لو استخدمنا نفس المفتاح لعملية التشفير فإن ناتج التشفير سيختلف عند كل مرة نستخدم فيها نفس المفتاح، في حين يبقى الناتج في الـ DES كما هو ، ومن الاستخدامات الأخرى للـ RSA Algorithms الـDigital Signature وهو ما سنبينه في الجزء التالي من هذا الفصل.

ثانيا: DSACryptoServiceProvider for Digital Signature Algorithm

التواقيع الرقمية :والهدف منها التحقق من هوية الشخص مرسل الرسالة وكمثال يقوم المرسل بتوليد ملخص لرسالة باستخدام الـHash 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 الـDSACryptoServiceProvider لتوليد المفتاح العام والخاص ثم نكون الـHash 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 ، سنقوم الآن بإنشاء مشروع نستخدم فيه الـDSACryptoServiceProvider لتوليد تواقيع رقمية ثم إجراء عملية التحقق عليه ، وسيكون الشكل العام للبرنامج كما في الشكل التالي:



سنقوم بإنشاء Four Methods ، الـ GetHash لتوليد الـ Hash Code لتوقيع الرقمي وسيأخذ String Message وسيرجع الـ Hash Code كـ Byte Array وسيستخدم الـMD5CryptoServiceProvider Instance من الـ ComputeHash Method لتوليد الـ Hash Code السابق ،الـVerifyHash method ونستطيع من خلالها المقارنة بين الـMessage المرسلة والـ Hash Code المولد من الرسالة لتأكد من صحة تطابق النص مع الـ Hash المولد ، الـ CreateSignature Method وستستخدم لتوليد توقيع رقمي من الـHash Code الممرر حيث سنستخدم الـ RSAPKCS1SignatureFormatter Class ونمرر له الـRSACryptoServiceProvider Instance Object والـ Hash gorithm وباستخدام الـCreateSignature Method سنولد التوقيع الرقمي ، الـVerifySignature Method وستستخدم للمقارنة بين الـ Hash Code والتوقيع الرقمي المولد فإذا تطابق يرجع قيمة True والعكس يرجع False ، ويتم كل ذلك كما يلي:

C#:

```
using System;
using System.Security.Cryptography;
using System.Text;
public class MD5HashHelper
{

        public byte[] GetHash(string message)
```

```
        {
                byte[] data;
data = UTF8Encoding.ASCII.GetBytes(message);
MD5CryptoServiceProvider md5 = new MD5CryptoServiceProvider();
        return md5.ComputeHash(data, 0, data.Length);
        }

        public bool VerifyHash(string message, byte[] hash)
        {
        byte[] data;
        data = UTF8Encoding.ASCII.GetBytes(message);
        MD5CryptoServiceProvider md5 = new
MD5CryptoServiceProvider();
        byte[] hashTemp = md5.ComputeHash(data, 0, data.Length);

        for (Int32 counter = 0; counter <= hash.Length - 1; counter += 1)
                {
                if (hash[counter] != hashTemp[counter])
                        {
                        return false;
                        }
                }
                return true;
        }
}

public class DigitalSignatureHelper
{
        private RSAParameters m_public;

        public byte[] CreateSignature(byte[] hash)
        {
        RSACryptoServiceProvider RSA = new
RSACryptoServiceProvider();
RSAPKCS1SignatureFormatter RSAFormatter = new
RSAPKCS1SignatureFormatter(RSA);
RSAFormatter.SetHashAlgorithm("MD5");
m_public = RSA.ExportParameters(false);
return RSAFormatter.CreateSignature(hash);
        }

        public bool VerifySignature(byte[] hash, byte[] signedhash)
        {
RSACryptoServiceProvider RSA = new RSACryptoServiceProvider();
RSAParameters RSAKeyInfo = new RSAParameters();
RSAKeyInfo.Modulus = m_public.Modulus;
RSAKeyInfo.Exponent = m_public.Exponent;
```

```
RSA.ImportParameters(RSAKeyInfo);
RSAPKCS1SignatureDeformatter RSADeformatter = new
RSAPKCS1SignatureDeformatter(RSA);
RSADeformatter.SetHashAlgorithm("MD5");
return RSADeformatter.VerifySignature(hash, signedhash);
        }
}
```

VB.NET:

```
Imports System
Imports System.Security.Cryptography
Imports System.Text

Public Class MD5HashHelper

   Public Function GetHash(ByVal message As String) As Byte()
      Dim data As Byte()
      data = UTF8Encoding.ASCII.GetBytes(message)
      Dim md5 As MD5CryptoServiceProvider = New
MD5CryptoServiceProvider
      Return md5.ComputeHash(data, 0, data.Length)
   End Function
   Public Function VerifyHash(ByVal message As String, ByVal hash
As Byte()) As Boolean
      Dim data As Byte()
      data = UTF8Encoding.ASCII.GetBytes(message)
      Dim md5 As MD5CryptoServiceProvider = New
MD5CryptoServiceProvider
      Dim hashTemp As Byte() = md5.ComputeHash(data, 0,
data.Length)

      Dim counter As Int32 = 0
      Do While counter <= hash.Length - 1
         If hash(counter) <> hashTemp(counter) Then
            Return False
         End If
         counter += 1
      Loop
      Return True
   End Function
End Class

Public Class DigitalSignatureHelper
   Private m_public As RSAParameters
   Public Function CreateSignature(ByVal hash As Byte()) As Byte()
      Dim RSA As RSACryptoServiceProvider = New
RSACryptoServiceProvider
```

```
        Dim RSAFormatter As RSAPKCS1SignatureFormatter = New
RSAPKCS1SignatureFormatter(RSA)
        RSAFormatter.SetHashAlgorithm("MD5")
        m_public = RSA.ExportParameters(False)
        Return RSAFormatter.CreateSignature(hash)
    End Function

 Public Function VerifySignature(ByVal hash As Byte(), ByVal
signedhash As Byte()) As Boolean
        Dim RSA As RSACryptoServiceProvider = New
RSACryptoServiceProvider
        Dim RSAKeyInfo As RSAParameters = New RSAParameters
        RSAKeyInfo.Modulus = m_public.Modulus
        RSAKeyInfo.Exponent = m_public.Exponent
        RSA.ImportParameters(RSAKeyInfo)
        Dim RSADeformatter As RSAPKCS1SignatureDeformatter =
New RSAPKCS1SignatureDeformatter(RSA)
        RSADeformatter.SetHashAlgorithm("MD5")
        Return RSADeformatter.VerifySignature(hash, signedhash)
    End Function
End Class
```

ولإستخدام الـ Methods السابقة:

C#:

```
DigitalSignatureHelper ds = new DigitalSignatureHelper();
byte[] hash1;
byte[] hash2;
byte[] signedhash;
.
.
.

try //توليد التوقيع الرقمي
        {
MD5HashHelper md5 = new MD5HashHelper();
hash1 = md5.GetHash(textBox1.Text);
signedhash = ds.CreateSignature(hash1);
MessageBox.Show("Signature Created Successfully!");
        }
catch(Exception ex){MessageBox.Show(ex.Message);}
.
.
.

try //لتأكد من صحة التوقيع
        {
MD5HashHelper md5 = new MD5HashHelper();
```

```
hash2 = md5.GetHash(textBox2.Text);

if (ds.VerifySignature(hash2, signedhash))
        {
        MessageBox.Show("Signatures verified Successfully!");
        }
        else
        {
        MessageBox.Show("Signatures verification failed!");
        }
        }
catch(Exception ex){MessageBox.Show(ex.Message);}
```

VB.NET:

```
Dim ds As DigitalSignatureHelper = New DigitalSignatureHelper()
Dim hash1 As Byte()
Dim hash2 As Byte()
Dim signedhash As Byte()
.
.
.

Try'توليد التوقيع الرقمي
Dim md5 As MD5HashHelper = New MD5HashHelper()
hash1 = md5.GetHash(textBox1.Text)
signedhash = ds.CreateSignature(hash1)
MessageBox.Show("Signature Created Successfully!")
Catch ex As Exception
        MessageBox.Show(ex.Message)
End Try
.
.
.
Try 'لتأكد من صحة التوقيع
Dim md5 As MD5HashHelper = New MD5HashHelper()
hash2 = md5.GetHash(textBox2.Text)

If ds.VerifySignature(hash2, signedhash) Then
        MessageBox.Show("Signatures verified Successfully!")
        Else
        MessageBox.Show("Signatures verification failed!")
        End If
Catch ex As Exception
        MessageBox.Show(ex.Message)
End Try
```

وهكذا بينا كيفية إنشاء برنامج لتوليد الـ Digital Signature Algorithms باستخدام الـRSA، سنبين في الجزء التالي من هذا الفصل انواع الـ Hash Algorithms واستخدامها في بيئة الدوت نيت.

ثالثا- Hashing algorithms: ويعتبر من الأساليب القوية جدا لتشفير البيانات إذ يمكننا من استخدام algorithm تصل إلى 512 bits بدلا من 128 bits وذلك باستخدام الـ Message MAC Digest Algorithms وهنا لن تستطيع فك تشفير الرسالة وإرجاعها إلى حالتها السابقة ويستخدم بشكل أساسي لتوليد الـPasswords وفي توليد التواقيع الرقمية Digital Signature وفي اغلب الحالات تستخدم لتخزين كلمة المرور Password في الـDatabase بشكل امن.

ويستخدم الـSHA1Managed والـSHA256ManagedوالـSHA384Managed والـSHA512Managed لتعريف Hash Object ومنه نستخدم الـ ComputeHash Method لتوليد الـhash 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));
```

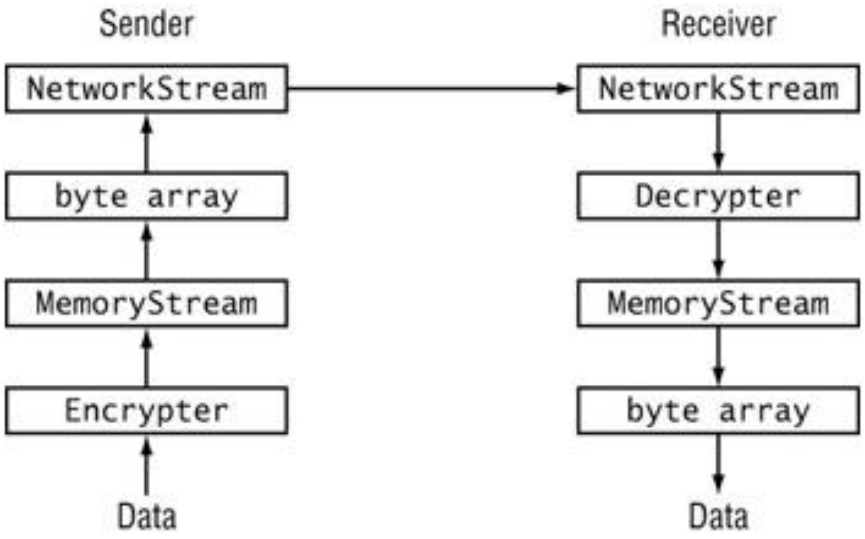
VB.NET:

```
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))
```

ويمكن استخدام الطريقة السابقة للمقارنة بين الـ Hash Codes ...، سنبين في الجزء التالي من هذا الفصل كيفية استخدام معايير التشفير في برمجيات الشبكات...

Using Security Encryption Standards With Network - C Applications:

في برمجيات الشبكات نقوم في البداية بتشفير البيانات المرسلة باستخدام أي من الأساليب السابقة لتشفير ثم نحول البيانات المشفرة إلى Stream لإرسالها عبر الـSocket باستخدام الـNetwork Stream كمثال، ثم يقوم الطرف المستقبل باستقبال الرسالة باستخدام الـNetwork Stream عبر الـSocket ،ويجب أن تكون عملية فك التشفير كما هي الخوارزمية المستخدمة إثناء التشفير سواء RSA أو DES أو التواقيع الرقمية ، ثم نستطيع تحمل الرسالة إلى الـMemory stream حيث ستستقبل الرسالة كـStream Data ، وبعد ذلك يمكن تحويلها إلى Byte Array لفك تشفيرها لتحويلها إلى رسالة مرة أخرى، لاحظ الشكل التالي:



سنقوم الآن بإنشاء برنامج بسيط (Client / Server) لإرسال نص من جهاز إلى آخر وسنخدم فيه الـ RSA لتشفير النص المرسل حيث سيتفق الطرفان على الـ Private Kay والذي سيستخدم لتشفير و فك تشفير الرسالة...

و سيكون الشكل العام للبرنامج كما في الشكل التالي ويبين طرفان مرسل ومستقبل ويقوم المرسل بتشفير الرسالة باستخدام المفتاح العام ويتفق الطرفان على المفتاح الخاص والذي سيتم فك الرسالة من خلاله:

أولا المرسل أو الـ Client :

وسنستخدم فيه الطريقة المعروفة لتشفير الرسائل باستخدام الـ RSA algorithms حيث سيتم تحويل الرسالة المشفرة إلى Byte Array وسنستخدم الـ Serialization لإرسالها حيث سيتم إرسالها كـ Serialization Object ويتم كل ذلك كما يلي:

C#:

```
using System.Text;
using System.Security.Cryptography;
using System.Net.Sockets;
using System.Net;
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;
.
.
.
public byte[] mymsg;
public byte[] Encrypt(string strData,CspParameters PrivateKey)
{
RSACryptoServiceProvider rsa = new
RSACryptoServiceProvider(PrivateKey);
byte[] data = rsa.Encrypt(Encoding.Unicode.GetBytes(strData), false);
return data;
}
.
.
.
try
{
CspParameters cp = new CspParameters();
cp.KeyContainerName = text_private.Text;
mymsg = Encrypt(text_msg_to_send.Text,cp);
text_encrypted.Text = Encoding.Unicode.GetString(mymsg);

object op = (object) mymsg;

BinaryFormatter br = new BinaryFormatter ();

TcpClient myclient = new TcpClient (text_IP.Text,7000);
NetworkStream myns = myclient.GetStream ();
br.Serialize (myns,op);

myns.Close ();
myclient.Close ();

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

VB.NET:

```
Imports System.Text
Imports System.Security.Cryptography
Imports System.Net.Sockets
Imports System.Net
Imports System.IO
Imports System.Runtime.Serialization.Formatters.Binary
.
.
.
Public mymsg As Byte()
Public Function Encrypt(ByVal strData As String, ByVal PrivateKey
As CspParameters) As Byte()
   Dim rsa As RSACryptoServiceProvider = New
RSACryptoServiceProvider(PrivateKey)
   Dim data As Byte() =
rsa.Encrypt(Encoding.Unicode.GetBytes(strData), False)
   Return data
End Function
.
.
.
Try
Dim cp As CspParameters = New CspParameters
cp.KeyContainerName = text_private.Text
mymsg = Encrypt(text_msg_to_send.Text,cp)
text_encrypted.Text = Encoding.Unicode.GetString(mymsg)

Dim op As Object = CObj(mymsg)
Dim br As BinaryFormatter = New BinaryFormatter
Dim myclient As TcpClient = New TcpClient(text_IP.Text, 7000)
Dim myns As NetworkStream = myclient.GetStream()
br.Serialize (myns,op)

myns.Close ()
myclient.Close ()

Catch ex As Exception
        MessageBox.Show(ex.Message)
End Try
```

ثانيا المستقبل أو الـ Server :

وفي طرف المستقبل الـ Server يتم استقبال الـ Encrypted Message as an Object ثم تحول إلى Byte Array بعد عمل الـ Casting عليها ومن ثم نمرر الـ Byte Array المستلم إلى الـ RSA Decryption Method ليتم فك تشفيرها باستخدام المفتاح الخاص والذي تم الاتفاق عليه .

C#:

```
using System.Text;
using System.Security.Cryptography;
using System.Net.Sockets ;
using System.IO;
using System.Threading;
using System.Runtime.Serialization.Formatters.Binary;
using System.Runtime.Serialization.Formatters;
.
.
.
public byte[] mymsg;
NetworkStream myns;
TcpListener mytcpl;
Socket mysocket;
Thread myth;

public byte[] Decrypt(byte[] en_data,CspParameters PrivateKey)
{
RSACryptoServiceProvider rsa = new
RSACryptoServiceProvider(PrivateKey);
byte[] data = rsa.Decrypt(en_data, false);
return data;
}
.
.
.
void RSA_Server ()//استقبال الرسالة بشكل مشفر
{
mytcpl = new TcpListener (7000);
mytcpl.Start ();
mysocket = mytcpl.AcceptSocket ();
myns = new NetworkStream (mysocket);
BinaryFormatter br = new BinaryFormatter ();
object op;
op= br.Deserialize (myns); // Deserialize the Object from Stream

mymsg = (byte[]) op;
text_encrypted.Text = Encoding.Unicode.GetString(mymsg);


mytcpl.Stop();

        while (true)
                {
        RSA_Server();
```

```
                }
}
.
.
.

try // فك تشفير الرسالة
{

CspParameters cp = new CspParameters();
cp.KeyContainerName = text_private1.Text;

byte[] my_msg = Decrypt(mymsg,cp);
text_decrypted.Text = Encoding.Unicode.GetString(my_msg);
}
catch (Exception ex){MessageBox.Show(ex.Message);}
```

VB.NET:

```
Imports System.Text
Imports System.Security.Cryptography
Imports System.Net.Sockets
Imports System.IO
Imports System.Threading
Imports System.Runtime.Serialization.Formatters.Binary
Imports System.Runtime.Serialization.Formatters
.
.
.
Public mymsg As Byte()
Private myns As NetworkStream
Private mytcpl As TcpListener
Private mysocket As Socket
Private myth As Thread

Public Function Decrypt(ByVal en_data As Byte(), ByVal PrivateKey
As CspParameters) As Byte()
   Dim rsa As RSACryptoServiceProvider = New
RSACryptoServiceProvider(PrivateKey)
   Dim data As Byte() = rsa.Decrypt(en_data, False)
   Return data
End Function
.
.
.

Private Sub RSA_Server() 'استقبال الرسالة بشكل مشفر
   mytcpl = New TcpListener(7000)
```

```
    mytcpl.Start()
    mysocket = mytcpl.AcceptSocket()
    myns = New NetworkStream(mysocket)
    Dim br As BinaryFormatter = New BinaryFormatter
    Dim op As Object
    op = br.Deserialize(myns) ' Deserialize the Object from Stream

    mymsg = CType(op, Byte())
    text_encrypted.Text = Encoding.Unicode.GetString(mymsg)

    mytcpl.Stop()

    Do While True
       RSA_Server()
    Loop
End Sub
.
.
.
Try ' فك تشفير الرسالة

Dim cp As CspParameters = New CspParameters
cp.KeyContainerName = text_private1.Text

Dim my_msg As Byte() = Decrypt(mymsg, cp)
text_decrypted.Text = Encoding.Unicode.GetString(my_msg)
Catch ex As Exception
        MessageBox.Show(ex.Message)
End Try
```

وهكذا بينا في هذا الفصل كيفية التعامل مع الـCryptography لإجراء عمليات التشفير على الـ Data المرسلة عبر الشبكة ، كما بينا طرق استخدام معاير التشفير مثل الـ DES و الـHashing والـRSA والـ Digital Signature في بيئة الدوت نيت ، سيتم الحديث في الفصل التالي عن الـ Socket Permission واستخدامها في بيئة الدوت نيت وبرمجيات الشبكات.

Chapter 19
Socket Permissions

- Permission Namespace Overview
- Security Action
- Socket Access property
- Socket Permissions

: Permission Namespace Overview : 19

وتدعم الـPermission Namespace في الدوت نيت ثلاثة أنواع من الصلاحيات وهي الـ Code access permissions والـIdentity Permissions والـ Role- based permissions...

تطبق الـ Code access permissions على كل من الـClasses التالية:

WebPermission: The ability to make/accept connection to/from the web.
OleDbPermission: The ability to access databases with OLEDB.
SQLClientPermission: The ability to access SQL databases.
UIPermission: The ability to use User Interface.
PrintingPermission: The ability to print.
FileIOPermission: The ability to work with files.
SocketPermission: The ability to make/accept TCP/IP connections on a transport address.
SecurityPermission: The ability to execute, assert permissions, call into unmanaged code, skip verification and other rights.

وتطبق الـIdentity permissions على الـ Classes التالية:

PublisherIdentityPermission: Software publisher's digital signature
StrongNameIdentityPermission: Assembly's strong Name
URLIdentityPermission: URL from which the code is originated
ZoneIdentityPermission: Zone from which the assembly is originated
SiteIdentityPermission: Location of web site from which the code is originated

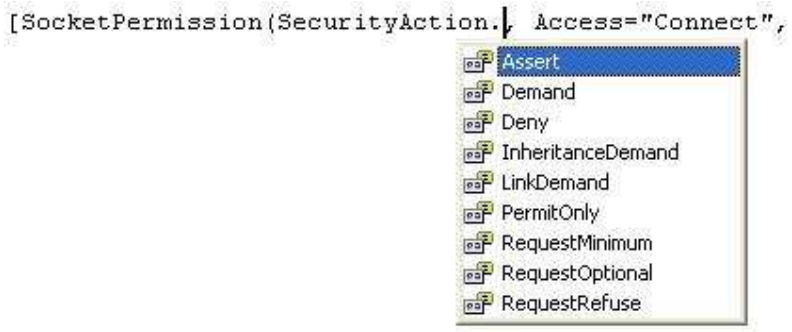
وتطبق الـ Role- based permissions على الـ Class **PrincipalPermission** فقط.

Socket_Permission: وتمكنك من تحديد صلاحيات استخدام الـSocket في برمجيات الشبكات باستخدام SocketPermission و SocketPermissionAttribute ضمن الـSystem.Net والـSystem.Security.Permissions Namespaces وكمثال نستطيع منع Client Host Address معين من الاتصال مع الـListener Application ، ويتم ذلك بتعريف SocketPermission Attribute نحدد فيها نوع العملية والـAccess Kind و عنوان الـHost الذي سيطبق عليه الـPermission ورقم الـPort ونوع الـTransport سواء موجه أو غير موجه TCP أو UDP.

نريد في هذا المثال منع اتصال الـloopback 127.0.0.1 Address بالـSocket عبر جميع الـPorts وبغض النظر عن نوع الـSocket المستخدم.

```
[SocketPermission(SecurityAction.Deny, Access="Connect",
Host="127.0.0.1",Port="All", Transport="All")]
```

يمكننا الـSecurityAction 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 الـClasses التي تقع في منطقة الـStack أعلى الـDefined Abstract
Deny: وتعني منع الـClient Host من إجراء عملية معينة.
InheritanceDemand : وفيها تطبق الصلاحيات على الـClass الذي سيرث الـClass الحالي.
PermitOnly: وفيه يمنع جميع الـAccess عدا الـClient User المحدد ....

وفي الAccess property نحدد نوع عملية المنع أو السماح وتأخذ خيارين هما :

Accept لمنع أو السماح لـ Client Socket من عمل Binding مع الـIP Address و الـPort المحدد.
Connect لمنع أو السماح لـ Client Socket من عمل connect مع الـRemote Host المحدد.

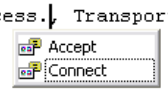
في الـHost والـPort نحدد عنوان الـHost الذي سيطبق عليه الـPermission و رقم الـPort التي يتصل بها (في الـPort property نستطيع تمرير كلمة all لدلالة على تطبيق الصلاحية على جميع الـPorts)

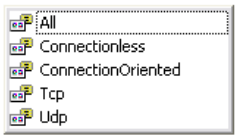
وأخيرا نحدد الـTransport property والتي سنعرف فيها نوع الـSocket المستخدم وتأخذ الخيارات التالية:
All بدون تحديد نوع الـSocket إذ تطبق هذه الـPermission على جميع الـ Socket Types.
Connectionless إذا كانت الـSocket تستخدم Protocols Datagram وكمثال بروتوكول UDP.
ConnectionOriented إذا كانت الـSocket تستخدم Protocols Oriented وكمثال بروتوكول TCP.
TCP إذ تستطيع تحديده مباشرة.
UDP إذ تستطيع تحديده مباشرة.

ويمكن أيضا استخدام الـ Socket Permission من خلال اشتقاق Instance Object من الـSocketPermission Class ثم تحديد نوع الـ Permission المراد وضعه ويمكنك أن تمرر له إما الـ Permission State وذلك لسماح أو منع الـ Permission المحدد أو تمرير الـ Permission الذي تريد من خلال تحديد الـ Network Access سواء اتصال Connect

Outgoing أو Accept Incoming ثم تحديد نوع البروتوكول المستخدم سواء TCP أو UDP أو الاثنين معا وبعد ذلك نمرر الـ IP Address للـ Remote Machine في حالة السماح أو المنع للـ incoming Access أو الـ IP للـ Local Machine في حالة الـ Connect Outgoing ، ويتم ذلك كما يلي:

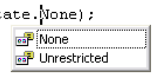
```
SocketPermission mySocketPermission =
new SocketPermission(NetworkAccess.| TransportType.Tcp, IP Address, Port);
```





كما ويمكن إسناد الـ Socket State للـSocketPermission Class إذ نحدد له الـ State سواء بسماح أو رفض الاتصال وكما يلي:

```
SocketPermission mySocketPermission= new SocketPermission(PermissionState.None);
```



كما يمكن دمج أكثر من Socket Permission وذلك باستخدام الـ Union Method ، وبعد ذلك تسند إلى SocketPermission Object جديد وكما يلي كمثال:

C#:

```
SocketPermission mySocketPermission1= new
SocketPermission(PermissionState.None);
mySocketPermission1.AddPermission(NetworkAccess.Accept,
TransportType.Tcp, IP, Port);

SocketPermission mySocketPermission2 =new
SocketPermission(NetworkAccess.Connect, TransportType.Tcp, IP,
Port);

SocketPermission mySocketPermissionUnion =
(SocketPermission)mySocketPermission1.Union(mySocketPermission2
);
```

VB.NET:

```
Dim mySocketPermission1 As SocketPermission = New
SocketPermission(PermissionState.None)
mySocketPermission1.AddPermission(NetworkAccess.Accept,
TransportType.Tcp, IP, Port)

Dim mySocketPermission2 As SocketPermission = New
SocketPermission(NetworkAccess.Connect, TransportType.Tcp, IP,
Port)
```

```
Dim mySocketPermissionUnion As SocketPermission =
CType(mySocketPermission1.Union(mySocketPermission2),
SocketPermission)
```

كما يمكن أيضا دمج أكثر من Union Permission List وذلك باستخدام الـ Intersect Method وكما يلي:

C#:
```
SocketPermission mySocketPermissionIntersect =
(SocketPermission)mySocketPermission1.Intersect(mySocketPermissio
nUnion);
```

VB.NET:
```
Dim mySocketPermissionIntersect As SocketPermission =
CType(mySocketPermission1.Intersect(mySocketPermissionUnion),
SocketPermission)
```

ونستطيع أيضا نقل الـ Permission List السابقة إلى XML Data ويمكن تحويلها من XML إلى Permission List مرة أخرى وذلك باستخدام الـ FromXml Method للقراءة من XML والـ ToXml Method لتحويل الـPermission إلى XML وكما يلي كمثال:

C#:
```
mySocketPermission1.FromXml(mySocketPermission1.ToXml());
```

VB.NET:
```
mySocketPermission1.FromXml(mySocketPermission1.ToXml());
```

ونستطيع تطبيق الـ Security Permissions باختيار واحدة من (Deny ، Assert,PermitOnly):

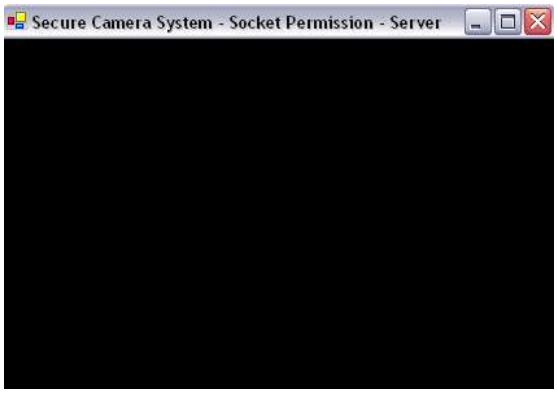
C#:
```
SocketPermission sp = new SocketPermission(PermissionState.None);
sp.AddPermission(NetworkAccess.Accept,TransportType.Tcp,"0.0.0.0"
,0);
sp.AddPermission(NetworkAccess.Connect,TransportType.Tcp,"0.0.0.0
",0)
//        sp.Deny; // المنع
//        sp.Assert;//السماح
sp.PermitOnly();
```

VB.NET:
```
Dim sp As SocketPermission = New
SocketPermission(PermissionState.None)
sp.AddPermission(NetworkAccess.Accept,TransportType.Tcp,"0.0.0.0"
,0)
sp.AddPermission(NetworkAccess.Connect,TransportType.Tcp,"0.0.0.0
",0)
```

```
'       sp.Deny; // المنع
'       sp.Assert;//السماح
sp.PermitOnly()
```

سنقوم الآن بتطبيق مثال بسيط نستخدم فيه الـ Socket Permissions لمنع الاتصال من قبل بروتوكول ما أو السماح للـ Client بالاتصال أو منعه ، فكرة البرنامج تكمن في نقل صورة الكاميرا من جهاز الـ Client إلى الـServer باستخدام الـ Socket ومن ثم وضع صلاحيات للاتصال مع الـ Server ...

أولا برنامج الـ Server :
ويحتوي على Picturebox كما في الشكل التالي:



C#:

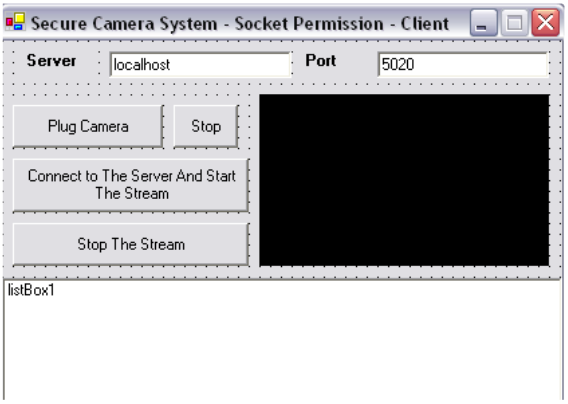
```
void StartMethod ()
{
TcpListener mytcpl;
Socket mysocket;
NetworkStream myns;

SocketPermission sp = new SocketPermission(PermissionState.None);
sp.AddPermission(NetworkAccess.Accept,TransportType.Tcp,"0.0.0.0"
,0);
sp.AddPermission(NetworkAccess.Connect,TransportType.Tcp,"0.0.0.0
",0);

//      sp.Deny; // الرفض
//      spAssert;//السماح
sp.PermitOnly();

mytcpl = new TcpListener (5020);
mytcpl.Start ();
mysocket = mytcpl.AcceptSocket ();
myns = new NetworkStream (mysocket);

pictureBox1.Image = Image.FromStream(myns);
mytcpl.Stop();
```

```
if (mysocket.Connected ==true)
        {
        while (true)
                {
        StartMethod ();
                }
        }
}
```

VB.NET:

```
Private Sub StartMethod()
   Dim mytcpl As TcpListener
   Dim mysocket As Socket
   Dim myns As NetworkStream
   Dim sp As SocketPermission = New
SocketPermission(PermissionState.None)
   sp.AddPermission(NetworkAccess.Accept, TransportType.Tcp,
"0.0.0.0", 5020)
   sp.AddPermission(NetworkAccess.Connect, TransportType.Tcp,
"0.0.0.0", 5020)
   '      sp.Deny; // الرفض
   '      spAssert;//السماح
   sp.PermitOnly()
   mytcpl = New TcpListener(5020)
   mytcpl.Start()
   mysocket = mytcpl.AcceptSocket()
   myns = New NetworkStream(mysocket)
   pictureBox1.Image = Image.FromStream(myns)
   mytcpl.Stop()

   If mysocket.Connected = True Then
      Do While True
         StartMethod()
      Loop
   End If
End Sub
```

ثانيا برنامج الـ Client :



C#:

```
private void timer1_Tick(object sender, System.EventArgs e)
{
try{
MemoryStream ms = new MemoryStream();// Store it in Binary Array as Stream
pictureBox1.Image.Save(ms,System.Drawing.Imaging.ImageFormat.Jpeg);
byte[] arrImage = ms.GetBuffer();
ms.Close();
TcpClient myclient = new TcpClient (txt_host.Text,int.Parse(text_Port.Text));//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 ();}
catch (Exception ex){timer1.Enabled=false;
listBox1.Items.Add(ex.Message );
}
```

VB.NET:

```
Private Sub timer1_Tick(ByVal sender As Object, ByVal e As System.EventArgs)
    Try
        Dim ms As MemoryStream = New MemoryStream   ' Store it in Binary Array as Stream
        pictureBox1.Image.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg)
        Dim arrImage As Byte() = ms.GetBuffer()
        ms.Close()
        Dim myclient As TcpClient = New TcpClient(txt_host.Text, Integer.Parse(text_Port.Text)) 'Connecting with server
        Dim myns As NetworkStream = myclient.GetStream()
```

```
    Dim mysw As BinaryWriter = New BinaryWriter(myns)
    mysw.Write(arrImage) 'send the stream to above address
    mysw.Close()
    myns.Close()
    myclient.Close()
  Catch ex As Exception
    timer1.Enabled = False
    listBox1.Items.Add(ex.Message)
  End Try
```

لاحظ أنه يمكنك الاختيار بين وضع الـ Permission Attribute أوالـSocketPermission ويبقى الفرق بين استخدام أي منها في حالة كنت تريد وضع الصلاحيات إثناء التشغيل وتريد إمكانية تغيير هذه الصلاحيات إثناء Runtime فيفضل استخدام الـ SocketPermission instance Object أما إذا أردت وضع الصلاحيات بشكل دائم وبدون إمكانية تغيرها وعلى مستوى الـ Scope للبرنامج كاملا ، عندها يفضل استخدام الـAttribute ...

وهكذا بينا طرق التعامل مع الـ Socket Permission في بيئة الدوت نيت ، سيتم الحديث في الجزء التالي عن الـ Multithreading واستخدامها في بيئة الدوت نيت.

Part 6
Multithreading

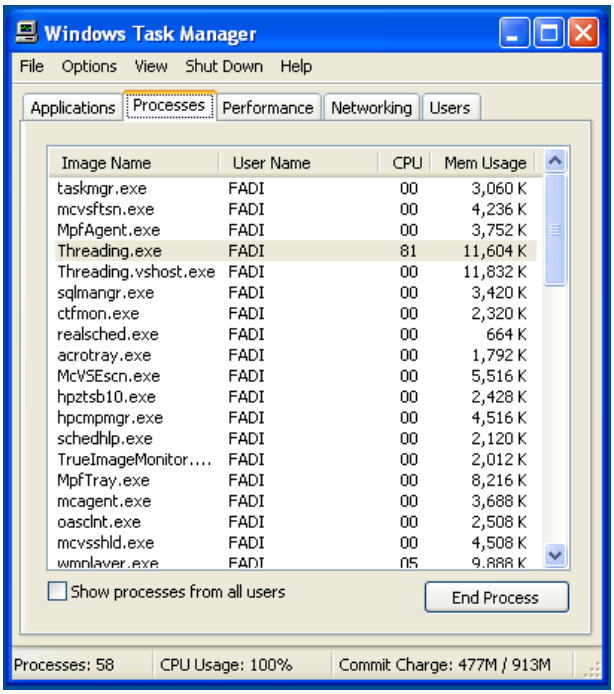
- Chapter 20 Multithreading (Using & Managing)

Chapter 20
Multithreading Using & Managing

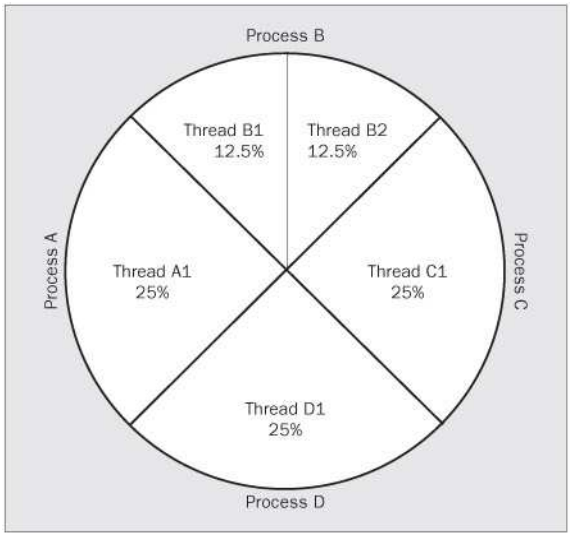
- Introduction to Threading in Dot Net
- Threading Classes & Members
- Multithreading & Network Applications

: The Concept of Threading : 20.1

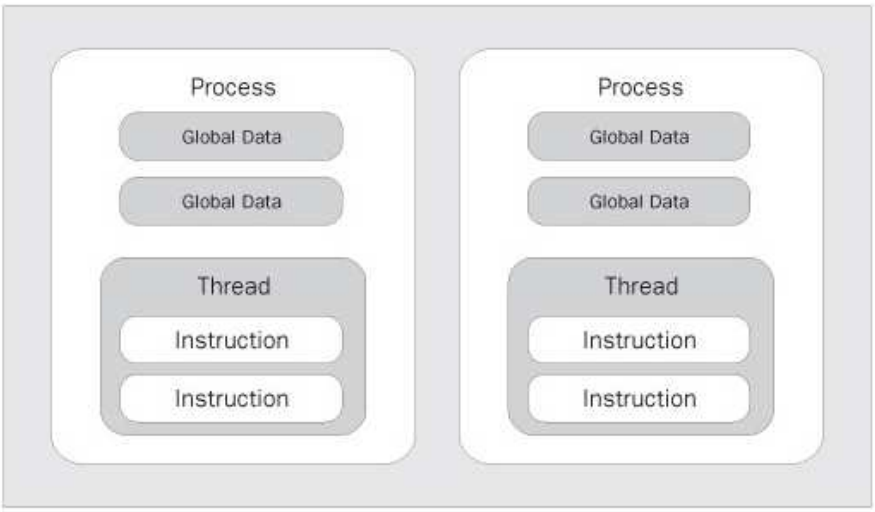
عزز مفهوم الـ Multithreading مفهوم الـ Multitasking في أنظمة التشغيل بحيث أن أي برنامج يمكنه أن يعمل على Thread منفصل عن الأخر بالإضافة إلا إمكانية أن يعمل نفس البرنامج على أكثر من Thread، أتى هذا المفهوم بعد تطوير فكرة الـ Multitasking في أنظمة التشغيل ، ويتم إدارة هذه العمليات من قبل نظام التشغيل الذي يقوم بتقسيم المهام على المعالج وفق الأولويات لكل Thread وعلى فترات زمنية ، ويمكن مشاهدة هذه العمليات في نظام التشغيل من خلال الـ Task Manager وكما في الشكل التالي:



لاحظ أن كل برنامج يحجز مقدار معين من المعالج بناء على حاجته ويقوم نظام التشغيل بتقسيم المهام على المعالج وفق الحاجة و الأولويات لاحظ الشكل التالي:

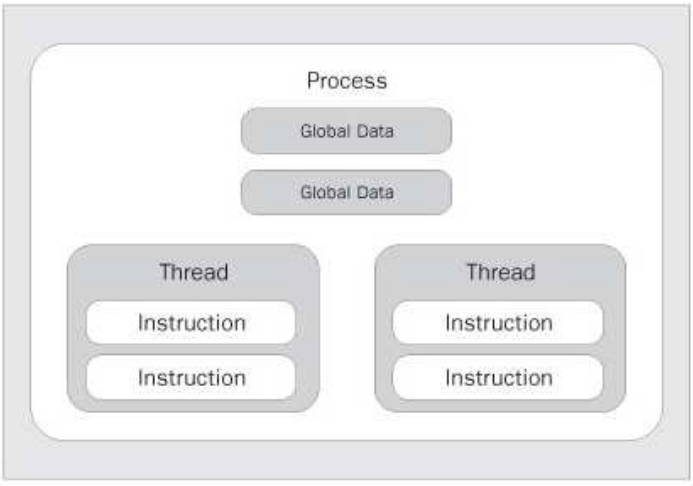


وبتأكيد يمكن التحكم بأي Thread في نظام التشغيل إذ يمكننا من عمل Interrupt له كما يمكننا إيقافه بشكل مؤقت، وأيضا إلغائه بعمل Abort له ، كل هذه العمليات متاحة لدى الـUser للقيام بها وأيضا متاح استخدامها برمجيا.



20.2 : Threading in Dot Net :

قمنا سابقا باستخدام الـ Threading في برمجيات الشبكات ، ولاحظنا انه لا يمكن أن يستقبل البرنامج أي شيء في حالة كونه يعمل على نفس الـ Thread الخاص بالـ Form مما يمنع تنفيذ أي عملية أخرى إلا بعد انتهاء العملية الجارية وهو ما لن يحدث أبدا ، إذ أننا استخدمنا Infinity Loop لبرنامج الاستقبال مما سيؤدي إلى عدم تنفيذ أي أمر آخر وسوف يتوقف البرنامج عن الاستجابة ، و الحل لهذه المشكلة الاختيار بين أمرين :
الأول، الابتعاد عن استخدام البرمجة المتزامنة Synchronous Programming و التوجه نحو البرمجة غير المتزامنة Asynchronous Programmingوهو ما تم شرحه في الجزء الخاص بالـ Streaming والـ Asynchronous Socket.
الثاني ، فصل الـ Infinity Loop عن الـ Thread الخاص بالبرنامج وجعله يعمل على Thread آخر منفصل ، وهذا ما يسمى بالـ Multithreading حيث سيتم تقسيم المعالجة على الـ Thread الخاص بالبرنامج والـ Thread الذي تم إنشائه ، لاحظ الشكل التالي:



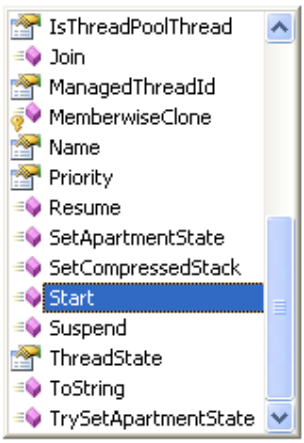
وسيعمل في هذه الحالة بشكل متوازي مع البرنامج وليس بشكل تسلسلي كما في السابق.

ولاستخدام الـ Threading في الدوت نيت يجب أولا تعريف الـ System.Threading Namespaces، ولتعريف Thread جديد نقوم باشتقاق Instance Object من الـ Thread Class ثم نمرر اسم الـ Method التي نريد عمل Thread لها إلى الـ Thread Start Delegate يمكن عمل كل ذلك مباشرة وبخطوة واحدة كما يلي:

C#:
```
Thread my_thread = new (new ThreadStart (my_method));
```

VB.NET:
```
Dim my_thread As Thread = New (AddressOf my_method)
```

```
my_thread.
```



لاحظ انه يمكنك التحكم بجميع العمليات التي تخص الـ Threading من خلال الـ Thread Object الذي تم اشتقاقه ومن أهم هذه العمليات الـ Start والـ Stop والتي تستخدم لتفعيل وإيقاف الـ Thread وكما نستخدم الـ Abort لإلغاء الـ Thread نهائيا من المعالجة.

أولا : أهم الـ Thread Class Methods

Start وتستخدم لبدأ الـ Thread في كل مرة يتم استدعائها يتم تنفيذ Thread آخر لنفس الـ Method التي تم تمريرها.

Join وتستخدم لتحكم بالفترة التي سيتم فيها إنهاء الـ Thread عند عمل Abort له.

Suspend وتستخدم لإيقاف الـ Thread الحالي بشكل مؤقت ولا نحصل على أي Exception في حالة كان الـThread في وضع التوقف.

Abort وتستخدم لإيقاف الـ Thread بشكل نهائي وإلغائه من الذاكرة، ويفضل وضع هذه الـ Method في الـClosing Event الخاص بالبرنامج لكي لا يبقى الـ Thread في الذاكرة.

Resume وتستخدم لإرجاع تنفيذ الـ Thread بعد إيقافه بشكل مؤقت باستخدام الـ Suspend .

Sleep وتستخدم لإيقاف الـ Thread فترة زمنية معينة.

ثانيا : أهم الـ Thread Class Properties

Name وتأخذ Get أو Set لتحديد أو معرفة اسم الـ Thread الذي تم انشائهة والهدف منها تسهيل التعامل مع الـMultithreads في البرنامج.

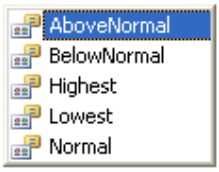
CurrentCulture وتأخذ Get أو Set لتحديد أو معرفة اللغة المعرفة لهذا الـ Thread.

CurrentUICulture وتأخذ Get لمعرفة اللغة المستخدمة في نظام التشغيل.

IsAlive وتأخذ Get لمعرفة إذا كان الـ Thread الحالي قيد التشغيل أم لا.

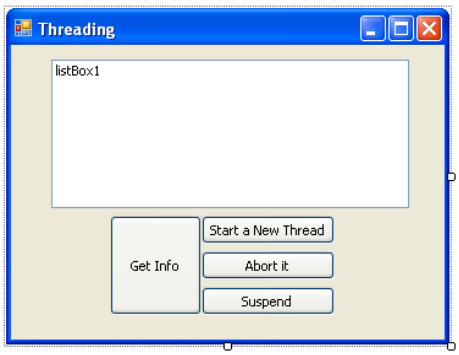
IsBackground وتأخذ Get أو Set لتحديد أو معرفة إذا كان الـ Thread يعمل أو سيعمل في الـBackgrount Proccess.
IsThreadPoolThread وتأخذ Get لمعرفة إذا كان الـ Thread يستخدم Pool queue لمجموعة من الـThread أم لا.
ManagedThreadId وتأخذ Get لمعرفة الـ Unique ID الخاص بالـ Thread الحالي.
Priority وتأخذ Get أو Set لتحديد أو معرفة الأولوية للـ Thread الحالي وفي الوضع الطبيعي تكون Normal ويمكن تغيرها كما يلي:

```
th.Priority = ThreadPriority.
```



حيث يحدد فيها مدى اهميتها بالنسبة لأولوية المعالجة.
ThreadState وتأخذ Get لمعرفة حالة الـ Thread الحالي هل هو يعمل أم لا وترجع True أو False .

مثال بسيط لـ Thread في الدوت نيت First Thread Program in Dot Net:
سنقوم في البداية بتصميم New Form كما في الشكل التالي:



لإنشاء Thread جديد :

C#:

```
using System.Threading;
.
.
Thread th;
.
.
private void Start_New_Thread_Click(object sender, EventArgs e)
{
th = new Thread(new ThreadStart(xx));
th.Start();
}
```

VB.NET:

```
Imports System.Threading
..
Private th As Thread
```

```
..
Private Sub Start_New_Thread_Click(ByVal sender As Object, ByVal e
As EventArgs)
   th = New Thread(AddressOf xx)
   th.Start()
End Sub
```

لإيقاف الـ Thread بشكل مؤقت:

```
th.Suspend()
```

لإيقاف الـ Thread بشكل نهائي من الذاكرة:

```
th.Abort()
```

ويفضل دائما وضع الـ Abort Method عن الـ Closing Event للبرنامج بحيث يقوم بإغلاق الـ Thread عند إطفاء البرنامج.

لمعرفة كافة المعلومات عن الـ Thread الحالي:

```
listBox1.Items.Add(th.ApartmentState)
listBox1.Items.Add(th.CurrentCulture)
listBox1.Items.Add(th.CurrentUICulture)
listBox1.Items.Add(th.IsAlive)
listBox1.Items.Add(th.IsBackground)
listBox1.Items.Add(th.IsThreadPoolThread)
listBox1.Items.Add(th.ManagedThreadId)
listBox1.Items.Add(th.Priority)
listBox1.Items.Add(th.ThreadState)
```

وتحتوي الـ Method على Infinity Loop يزيد كل مرة الـ I بمقدار واحد ويطبع النتيجة على الـ Form Text ، ويعتبر استخدام الـ Infinity Loop من أهم ما دعانا لإستخدام الـ Thread في هذه الحالة:

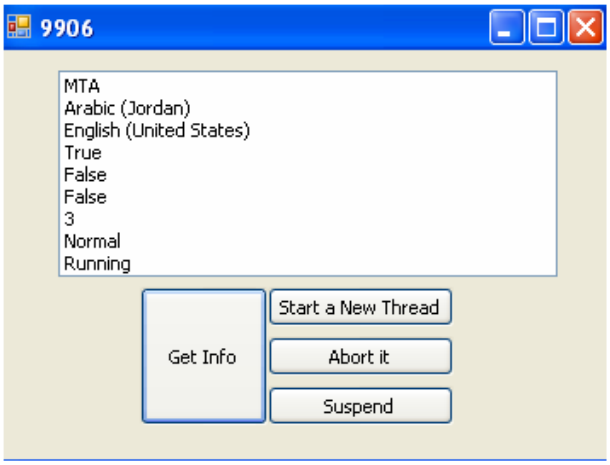
C#:

```
void xx()
{
int i = 0;
while (true) { i++; this.Text = i.ToString(); }
}
```

VB.NET:

```
Private Sub xx()
   Dim i As Integer = 0
   Do While True
      i += 1
      Me.Text = i.ToString()
   Loop
End Sub
```

وسيكون الـ Output للبرنامج كما في الشكل التالي:



استخدام الـ Multithreading في برمجيات الشبكات:

ينصح باستخدام الـ Multithreading في برمجيات الشبكات التي تعتمد البرمجة المتزامنة Synchronous Programming إذ أن طبيعة التنفيذ في هذه البرمجيات متسلسلة وهو ما سوف يمنع من تنفيذ عمليات أخرى لحين الانتهاء من العملية الجارية ، إذ أنه ومن المفضل استخدام Thread منفصل لكل عملية قد تطرنا إلى الانتظار فترة طويلة أو أن العملية ستبقى في وضع التنفيذ طيلة فترة تشغيل البرنامج.

1- استخدام الـ Threading في برمجيات الزبون Clients :

- استخدام دالة الربط Connect في الاتصال المتزامن: في حالة استخدام الـ Connect Method وفي حالة كان الـServer بطيء فإن عملية الموافقة ستأخذ بعض الوقت مما يؤدي إلى توقف البرنامج عن الاستجابة لحين إتمام عملية القبول ويفضل في هذه الحالة استخدام Thread منفصل للقيام بعملية الاتصال.

- استخدام الـ Client في عملية الإرسال و الاستقبال المتزامن ، يفضل في حالة الإرسال المتزامن عدم استخدام Thread منفصل لكل عملية وخاصة إذا كان الهدف من البرنامج إرسال مجموعة من التعليمات المختلفة إلى نفس الـServer حيث سيكون منشغل باستقبال التعليمات التي تم يتم إرسالها ، ويفضل في هذه الحالة استخدام البرمجة غير المتزامنة إذا كان الإرسال لأكثر من تعليمة ولنفس الجهة، أما في حالة الاستقبال المتزامن فلا بد من استخدام الـThread لأن عدم استخدام الـ Thread سيؤدي إلى توقف البرنامج عن الاستجابة.

2- استخدام الـ Threading في برمجيات الخادم Server :

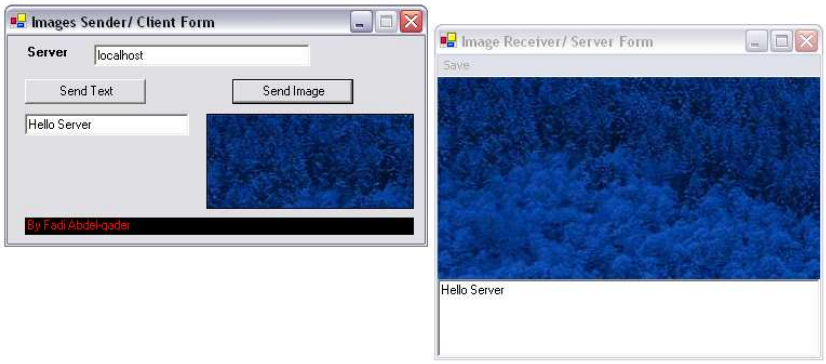
- استخدام دالة القبول AcceptSocket ، لابد من استخدام Thread منفصل في حالة استخدام دالة القبول AcceptSocket في البرمجة المتزامنة إذ انه سيتوقف البرنامج عن الإستجابة في حالة عدم استخدام الـ Thread مع الـ AcceptSocket حيث توضع عادة في Infinity Loop.

حالة خاصة:

قد تضطر إلى اللجوء إلى البرمجة غير المتزامنة في حالة إذا كان الـ Server سيستقبل أكثر من Client Reqouest في نفس الوقت مما سيؤدي إلى قبول أول طلب ورفض البقية وهذا امر غير فعال وخاصة إذا كان البرنامج سيعمل على الإنترنت وأن عدد الـ Clients وأوقات دخولهم غير معروفة ، وفي هذه الحالة لابد من اللجوء إلى البرمجة غير المتزامنة asynchronous Programming لإتمام عملية القبول.

مثال على استخدام الـ Multithreading في برمجيات الشبكات المتزامنة:

سوف ننشئ في هذا المثال برنامجين Client/Server حيث يمكن للـ Client إرسال Text و Images إلى الـServer وسيستقبل الـ Server الصور والـ Text باستخدام Tow Threads يعملان بشكل متوازي :



أولا برنامج الإرسال الـ Client :

C#:

```
void send(byte[] arrImage)
        {
try
{
TcpClient myclient = new TcpClient (txt_host.Text,5000);
NetworkStream myns = myclient.GetStream ();
BinaryWriter  mysw = new BinaryWriter (myns);
mysw.Write(arrImage);//send the stream to above address
mysw.Close ();
myns.Close ();
myclient.Close ();
}
catch (Exception ex){MessageBox.Show(ex.Message );}
}

void send(string msg) // overloading method
```

```
        {
try
        {
TcpClient myclient = new TcpClient (txt_host.Text,7000);
NetworkStream myns = myclient.GetStream ();
StreamWriter mysw = new StreamWriter (myns);
mysw.Write(msg);
mysw.Close ();
myns.Close ();
myclient.Close ();
        }
catch (Exception ex){MessageBox.Show(ex.Message );}
}
```

VB.NET:

```
Private Sub send(ByVal arrImage As Byte())
Try
Dim myclient As TcpClient = New TcpClient (txt_host.Text,5000)
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
MessageBox.Show(ex.Message)
End Try
End Sub

Private Sub send(ByVal msg As String) ‘ overloading method
        Try
Dim myclient As TcpClient = New TcpClient (txt_host.Text,7000)
Dim myns As NetworkStream = myclient.GetStream ()
Dim mysw As StreamWriter = New StreamWriter (myns)
        mysw.Write(msg)
        mysw.Close ()
        myns.Close ()
        myclient.Close ()
Catch ex As Exception
MessageBox.Show(ex.Message)
End Try
End Sub
```

ثانيا برنامج الإستقبال الـ Server :

C#:

```
Thread myth1;
```

```
Thread myth2;
private void Form1_Load(object sender, System.EventArgs e)
{
myth1= new Thread  (new System.Threading
.ThreadStart(Image_Receiver));
myth1.Start (); // Start Thread 1

myth2= new Thread  (new System.Threading
.ThreadStart(text_Receiver));
myth2.Start ();// Start Thread 2
}

NetworkStream ns;
TcpListener tcp;
 Socket sock;
StreamReader sr;

void text_Receiver()
                {
tcp = new TcpListener (7000);// Open The Port
tcp.Start ();// Start Listening on That Port
sock = tcp.AcceptSocket ();// Accept Any Request From Client and
Start a Session
ns = new NetworkStream (sock);// Receives The Binary Data From Port
sr = new StreamReader (ns);// Convert Received Data to String
textBox1.Text = sr.ReadLine();
tcp.Stop();
        while (true)
                {
                text_Receiver();// Back to First Method
                }
                }

NetworkStream myns;
TcpListener mytcpl;
Socket mysocket;
void Image_Receiver()
        {
mytcpl = new TcpListener (5000);// Open The Port
mytcpl.Start ();// Start Listening on That Port
mysocket = mytcpl.AcceptSocket ();
myns = new NetworkStream (mysocket);
pictureBox1.Image = Image.FromStream(myns);

mytcpl.Stop();

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

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

}
```

VB.NET:

```
Private myth1 As Thread
Private myth2 As Thread
Private Sub Form1_Load(ByVal sender As Object, ByVal e As
System.EventArgs)
   myth1 = New Thread(New System.Threading.ThreadStart(AddressOf
Image_Receiver))
   myth1.Start()  ' Start Thread 1

   myth2 = New Thread(New System.Threading.ThreadStart(AddressOf
text_Receiver))
   myth2.Start()  ' Start Thread 2
End Sub

Private ns As NetworkStream
Private tcp As TcpListener
Private sock As Socket
Private sr As StreamReader
Private Sub text_Receiver()
   tcp = New TcpListener(7000)  ' Open The Port
   tcp.Start()  ' Start Listening on That Port
   sock = tcp.AcceptSocket()  ' Accept Any Request From Client and
Start a Session
   ns = New NetworkStream(sock)  ' Receives The Binary Data From
Port
   sr = New StreamReader(ns)  ' Convert Received Data to String
   textBox1.Text = sr.ReadLine()
   tcp.Stop()
   Do While True
      text_Receiver() ' Back to First Method
   Loop
End Sub

Private myns As NetworkStream
Private mytcpl As TcpListener
Private mysocket As Socket
Private Sub Image_Receiver()
   mytcpl = New TcpListener(5000)  ' Open The Port
```

```
    mytcpl.Start()  ' Start Listening on That Port
    mysocket = mytcpl.AcceptSocket()
    myns = New NetworkStream(mysocket)
    pictureBox1.Image = Image.FromStream(myns)
    mytcpl.Stop()
    If mysocket.Connected = True Then 'if Connected Start Again
        Do While True
            Image_Receiver() ' Back to First Method
        Loop
    End If
End Sub
```

ويجب الإنتباه إلى أمر هام في الـThreading Aport وهو إغلاق كل ما له علاقة بالـSocket مثل الـTcplistener قبل إغلاق الـThread في حدث الـClosing للـForm ...

Appendixes (A)

- System.Net Namespace
- System.Net.Socket Namespace
- Socket Option Members
- System.Threading Namespace
- TAPI 2 Telephony Functions
- Remoting TCP / HTTP Channels Members

1- System.Net Namespace Classes

وتدعم هذه الـ Namespace برمجة الـ Application Layer و الـ Transport Layer والـ Web Protocols ومن اهم هذه الـ Classes واستخداماتها:

Class	Description
Authorization	Provides authentication messaging for a web server.
Cookie	Provides a set of properties and methods used to manage cookies. This class cannot be inherited.
Dns	Simple domain name resolution functionality.
EndPoint	Identifies a network address. This is an abstract class.
GlobalProxySelection	Global default proxy instance for all HTTP requests.
HttpVersion	Defines the HTTP version numbers supported by the HttpWebRequest and HttpWebResponse classes.
HttpWebRequest	HTTP-specific implementation of the WebRequest class.
HttpWebResponse	HTTP-specific implementation of the WebResponse class.
IPAddress	Internet Protocol (IP) address.
IPEndPoint	A network endpoint consisting of an IP address and a port number.
IPHostEntry	Container class for Internet host address information.
NetworkCredential	Provides credentials for password-based authentication schemes such as basic, digest, NTLM, and Kerberos authentication.
SocketAddress	Stores serialized information from EndPoint-derived classes.
SocketPermission	Controls rights to make or accept socket connections.
WebClient	Provides common methods for sending data to and receiving data from a resource identified by a URI.
WebException	The exception that is thrown when an error occurs while accessing resources via the HTTP protocol.
WebPermission	Controls rights to access HTTP Internet

Class	Description
	resources.
WebPermissionAttribute	Specifies permission to access Internet resources.
WebProxy	Contains HTTP proxy settings for the WebRequest class.
WebRequest	Makes a request to a Uniform Resource Identifier (URI). This class is abstract.
WebResponse	Provides a response from a Uniform Resource Identifier (URI). This class is abstract.

2- System.Net.Socket Namespace Classes

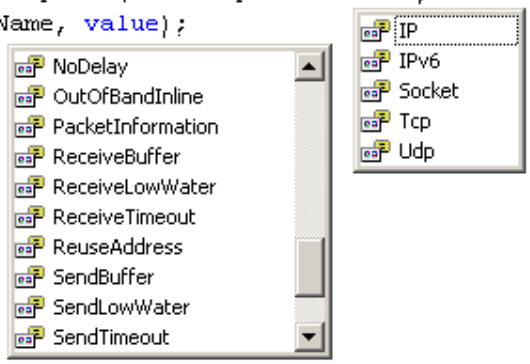
وتركز بشكل اساسي على برمجة الـTransport Layer وخاصة TCP & UDP Socket Programming ومن اهم الـClasses التي تدعمها:

Class	Description
LingerOption	Contains information about the amount of time it will remain available after closing with the presence of pending data (the socket's linger time).
MulticastOption	Contains IP address values for IP multicast packets.
NetworkStream	Provides the underlying stream of data for network access.
Socket	Implements the Berkeley Socket interface.
SocketException	The exception that is thrown when a socket error occurs.
TcpClient	Provides client connections for TCP network services.
TcpListener	Listens for connections from TCP network clients. This is essentially the TCP server class.
UdpClient	Provides User Datagram Protocol (UDP) network services.

3- Socket Options Class:

ويعتبر هذا الكلاس من الـ Classes المهمة في برمجيات الشبكات وخاصة أنه يسمح لك بتعديل أي أمر له علاقة بعملية الاتصال سواء على مستوى الـ Network Layer أو الـTransport Layer أو حتى على مستوى الـ Socket نفسه والصيغة العامة له كما يلي:

```
Socket host;
host.SetSocketOption(SocketOptionLevel. ,
SocketOptionName, value);
```



ويأخذ ثلاثة باروميترات يحدد في الأول مستوى الـ Socket Option التي نريد إسناد الـ Option إليها، ويحدد في الباروميتر الثاني نوعية الـ Option التي نريد تفعيلها أو منعها أو تغيير في قيمها ، ويأخذ الباروميتر الأخير Object Value فإذا كانت العملية هي السماح أو الرفض فتأخذ إما True أو False وإذا كانت تغيير أو تحديد قيم ما فتأخذ القيمة التي نريد تغييرها ، ومن أهم الـ Socket Option Name التالي:

Member name	Description
AcceptConnection	Socket is listening.
AddMembership	Add an IP group membership.
AddSourceMembership	Join a source group.
BlockSource	Block data from a source.
Broadcast	Permit sending broadcast messages on the socket.
BsdUrgent	Use urgent data as defined in RFC-1222. This option can be set only once, and once set, cannot be turned off.
ChecksumCoverage	Set or get UDP checksum coverage.
Debug	Record debugging information.
DontFragment	Do not fragment IP datagrams.
DontLinger	Close socket gracefully without lingering.
DontRoute	Do not route; send directly to interface addresses.
DropMembership	Drop an IP group membership.
DropSourceMembership	Drop a source group.
Error	Get error status and clear.
ExclusiveAddressUse	Enables a socket to be bound for exclusive access.
Expedited	Use expedited data as defined in

	RFC-1222. This option can be set only once, and once set, cannot be turned off.
HeaderIncluded	Indicates application is providing the IP header for outgoing datagrams.
IPOptions	Specifies IP options to be inserted into outgoing datagrams.
IpTimeToLive	Set the IP header time-to-live field.
KeepAlive	Send keep-alives.
Linger	Linger on close if unsent data is present.
MaxConnections	Maximum queue length that can be specified by Listen.
MulticastInterface	Set the interface for outgoing multicast packets.
MulticastLoopback	IP multicast loopback.
MulticastTimeToLive	IP multicast time to live.
NoChecksum	Send UDP datagrams with checksum set to zero.
NoDelay	Disables the Nagle algorithm for send coalescing.
OutOfBandInline	Receives out-of-band data in the normal data stream.
PacketInformation	Return information about received packets.
ReceiveBuffer	Send low water mark.
ReceiveLowWater	Receive low water mark.
ReceiveTimeout	Receive time out.
ReuseAddress	Allows the socket to be bound to an address that is already in use.
SendBuffer	Specifies the total per-socket buffer space reserved for sends. This is unrelated to the maximum message size or the size of a TCP window.
SendLowWater	Specifies the total per-socket buffer space reserved for receives. This is unrelated to the maximum message size or the size of a TCP window.

SendTimeout	Send timeout.
Type	Get socket type.
TypeOfService	Change the IP header type of service field.
UnblockSource	Unblock a previously blocked source.
UseLoopback	Enable Or Disable a Loop Back Connection

4- System.Threading Namespace Classes

يستخدم الـThreading لعمل Session منفصلة عن الـsession المستخدمة في البرنامج وهو أسلوب اخر للـAsynchronous وتدعم System.Threading مجموعة من الـClasses والتي تستخدم في ادارة عمليات الـThreading وهي كما يلي:

Class	Description
AutoResetEvent	This event notifies one or more waiting threads that an event has occurred.
Interlocked	This class protects against errors by providing atomic operations for variables that are shared by multiple threads.
ManualResetEvent	This event occurs when notifying one or more waiting threads that an event has occurred.
Monitor	This class provides a mechanism that synchronizes access to objects.
Mutex	A synchronization primitive that grants exclusive access to a shared resource to only one thread. It can also be used for inter-process synchronization.
ReaderWriterLock	This class defines a lock that allows single-writer and multiple-reader semantics.
RegisteredWaitHandle	This class represents a handle that has been registered when calling the RegisterWaitForSingleObject() method.
SynchronizationLockException	This exception is thrown when a synchronized method is invoked from an unsynchronized block of code.
Thread	This class creates and controls a thread, sets its priority, and gets its status.
ThreadAbortException	This exception is thrown when a call is

Class	Description
	made to the Abort() method.
ThreadExceptionEventArgs	This class provides data for the ThreadException event.
ThreadInterruptedException	This exception is thrown when a thread is interrupted while it is in a waiting state.
ThreadPool	This class provides a pool of threads that can be used to post work items, process asynchronous I/O, wait on behalf of other threads, and process timers.
ThreadStateException	This is the exception that is thrown when a thread is in an invalid state for the method call.
Timeout	This class simply contains a constant integer used when we want to specify an infinite amount of time.
Timer	This class provides a mechanism for executing methods at specified intervals.
WaitHandle	This class encapsulates operating system-specific objects that wait for exclusive access to shared resources.

اهم الـMethods التي تستخدم في ادارة الـThreading في الدوت نيت :

Public Method Name	Description
Abort()	This overloaded method raises a ThreadAbortException in the thread on which it is invoked, to begin the process of terminating the thread. Calling this method usually terminates the thread.
AllocateDataSlot()	This static method allocates an unnamed data slot on all the threads.
AllocateNamedDataSlot()	This static method allocates a named data slot on all threads.
FreeNamedDataSlot()	This static method frees a previously allocated named data slot.
GetData()	This static method retrieves the value from the specified slot on the current thread, within the current thread's current domain.
GetDomain()	This static method returns the current domain

Public Method Name	Description
	in which the current thread is running.
GetDomainID()	This static method returns a unique application domain identifier.
GetHashCode()	This method serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.
GetNamedDataSlot()	This static method looks up a named data slot.
Interrupt()	This method interrupts a thread that is in the WaitSleepJoin thread state.
Join()	This overloaded method blocks the calling thread until a thread terminates.
ResetAbort()	This static method cancels an Abort() requested for the current thread.
Resume()	This method resumes a thread that has been suspended.
SetData()	This static method sets the data in the specified slot on the currently running thread, for that thread's current domain.
Sleep()	This static and overloaded method blocks the current thread for the specified number of milliseconds.
SpinWait()	This static method causes a thread to wait the number of times defined by the iterations parameter.
Start()	This method causes the operating system to change the state of the current instance to ThreadState.Running.
Suspend()	This method will either suspend the thread, or if the thread is already suspended, has no effect.

ومن اهم الـProperties الخاصة بالـThreading :

Public Property Name	Description
ApartmentState	Sets or gets the apartment state of this thread.
CurrentContext	This static property gets the current context in which the thread is executing.
CurrentCulture	Sets or gets the culture for the current thread.

Public Property Name	Description
CurrentPrincipal	This static property sets or gets the thread's current principal. It is used for role-based security.
CurrentThread	This static property gets the currently running thread.
CurrentUICulture	Used at run time, this property sets or gets the current culture used by the Resource Manager to look up culture-specific resources.
IsAlive	Gets a value that indicates the execution status of the current thread.
IsBackground	Sets or gets a value that indicates whether a thread is a background thread or not.
IsThreadPoolThread	Gets a value indicating whether a thread is part of a thread pool.
Name	Sets or gets the name of the thread.
Priority	Sets or gets a value that indicates the scheduling priority of a thread.
ThreadState	Gets a value that contains the states of the current thread.

5- TAPI 2 Telephony Functions:

TAPI Initialization and Shutdown

Function	Description
lineInitializeEx	Initializes the TAPI line abstraction for use by the invoking application. Synchronous. **API Example:** public static extern int lineInitialize(ref int hTAPI, int hInst, LineCallBackDelegate fnPtr,ref int szAppName, ref int dwNumLines);
lineShutdown	Shuts down the application's use of TAPI's line abstraction. Synchronous. **API Example:** [DllImport("Tapi32.dll", SetLastError = true)] public static extern int lineShutdown(int hLineApp);

Line Version Negotiation

Function	Description

lineNegotiateAPIVersion	Allows an application to negotiate a TAPI version to use. Synchronous. **API Example:** [DllImport("Tapi32.dll", SetLastError = true)]public static extern int lineNegotiateAPIVersion(int hTAPI, int dwDeviceID, int dwAPILowVersion, int dwAPIHighVersion, ref int lpdwAPIVersion, ref lineextensionid lpExtensionID);

Line Status and Capabilities

Function	Description
lineGetDevCaps	Returns the capabilities of a given line device. Synchronous. **API Example:** [DllImport("Tapi32.dll", SetLastError = true)]public static extern int lineGetDevCaps(int hLineApp, int dwDeviceID, int dwAPIVersion, int dwExtVersion,ref linedevcaps lpLineDevCaps);
lineGetDevConfig	Returns configuration of a media stream device. Synchronous. **API Example:** [DllImport("tapi32.dll")] static extern long lineGetDevConfig(long dwDeviceID, ref VARSTRING lpDeviceConfig, string lpszDeviceClass);
lineGetLineDevStatus	Returns current status of the specified open line device. Synchronous. **API Example:** [DllImport("tapi32.dll")] static extern long lineGetLineDevStatus(long hLine, ref LINEDEVSTATUS lpLineDevStatus);
lineSetDevConfig	Sets the configuration of the specified media stream device. Synchronous. **API Example:** [DllImport("tapi32.dll")] static extern long lineSetDevConfig(long dwDeviceID, Any lpDeviceConfig, long dwSize, string lpszDeviceClass);

lineSetStatusMessages	Specifies the status changes for which the application needs to be notified. Synchronous. **API Example:** [DllImport("Tapi32.dll", SetLastError = true)] public static extern int lineSetStatusMessages(int hLine,int dwLineStates, int dwAddressStates);
lineGetStatusMessages	Returns the application's current line and address status message settings. Synchronous. **API Example:** [DllImport("tapi32.dll")] static extern long lineGetStatusMessages(long hLine, ref long lpdwLineStates, ref long lpdwAddressStates);
lineGetID	Retrieves a device ID associated with the specified open line, address, or call. Synchronous. **API Example:** [DllImport("tapi32.dll")] static extern long lineGetID(long hLine, long dwAddressID, long hCall, long dwSelect, ref VARSTRING lpDeviceID, string lpszDeviceClass);
lineGetIcon	Allows an application to retrieve an icon for display to the user. Synchronous **API Example:** [DllImport("tapi32.dll")] static extern long lineGetIcon(long dwDeviceID, string lpszDeviceClass, ref long lphIcon);.
lineConfigDialog	Causes the provider of the specified line device to display a dialog box that allows the user to configure parameters related to the line device. Synchronous. **API Example:** [DllImport("tapi32.dll")] static extern long lineConfigDialog(long dwDeviceID, long hwndOwner, string lpszDeviceClass);

lineConfigDialogEdit	Displays a dialog box allowing the user to change configuration information for a line device. Synchronous. **API Example:** [DllImport("tapi32.dll")] static extern long lineConfigDialogEdit(long dwDeviceID, long hwndOwner, string lpszDeviceClass, Any lpDeviceConfigIn, long dwSize, ref VARSTRING lpDeviceConfigOut);

Addresses

Function	Description
lineGetAddressCaps	Returns the telephony capabilities of an address. Synchronous. **API Example:** [DllImport("tapi32.dll")] static extern long lineGetAddressCaps(long hLineApp, long dwDeviceID, long dwAddressID, long dwAPIVersion, long dwExtVersion, ref LINEADDRESSCAPS lpAddressCaps);
lineGetAddressStatus	Returns current status of a specified address. Synchronous. **API Example:** [DllImport("tapi32.dll")] static extern long lineGetAddressStatus(long hLine, long dwAddressID, ref LINEADDRESSSTATUS lpAddressStatus);
lineGetAddressID	Retrieves the address ID of an address specified using an alternate format. Synchronous. **API Example:** [DllImport("tapi32.dll")] static extern long lineGetAddressID(long hLine, ref long lpdwAddressID, long dwAddressMode, string lpsAddress, long dwSize);

Opening and Closing Line Devices

Function	Description
lineOpen	Opens a specified line device for providing subsequent monitoring and/or control of the line. Synchronous.

	API Example: [DllImport("Tapi32.dll", SetLastError = true)] public static extern int lineOpen(int hLineApp, int dwDeviceID, ref int lphLine, int dwAPIVersion,int dwExtVersion, ref int dwCallbackInstance,int dwPrivileges, int dwMediaModes,ref int lpCallParams);
lineClose	Closes a specified opened line device. Synchronous. **API Example:** [DllImport("tapi32.dll")] static extern long lineClose(long hLine);

Address Formats

Function	Description
lineTranslateAddress	Translates between an address in canonical format and an address in dialable format. Synchronous. **API Example:** [DllImport("tapi32.dll")] static extern long lineTranslateAddress(long hLineApp, long dwDeviceID, long dwAPIVersion, string lpszAddressIn, long dwCard, long dwTranslateOptions, ref LINETRANSLATEOUTPUT lpTranslateOutput);
lineSetCurrentLocation	Sets the location used as the context for address translation. Synchronous. **API Example:** [DllImport("tapi32.dll")] static extern long lineSetCurrentLocation (long hLineApp, long dwLocation);
lineSetTollList	Manipulates the toll list. Synchronous. **API Example:** [DllImport("tapi32.dll")] static extern long lineSetTollList(long hLineApp, long dwDeviceID, string lpszAddressIn, long

	dwTollListOption);
lineGetTranslateCaps	Returns address translation capabilities. Synchronous. **API Example:** [DllImport("tapi32.dll")] static extern long lineGetTranslateCaps(long hLineApp, long dwAPIVersion, ref LINETRANSLATECAPS lpTranslateCaps);

Call States and Events

Function	Description
lineGetCallInfo	Returns fixed information about a call. Synchronous. **API Example:** [DllImport("tapi32.dll")] static extern long lineGetCallInfo(long hCall, ref LINECALLINFO lpCallInfo);
lineGetCallStatus	Returns complete call status information for the specified call. Synchronous. **API Example:** [DllImport("tapi32.dll")] static extern long lineGetCallStatus(long hCall, ref LINECALLSTATUS lpCallStatus);
lineSetAppSpecific	Sets the application-specific field of a call's information structure. Synchronous. **API Example:** [DllImport("tapi32.dll")] static extern long lineSetAppSpecific(long hCall, long dwAppSpecific);

Making Calls

Function	Description
lineMakeCall	Makes an outbound call and returns a call handle for it. Asynchronous. **API Example:** [DllImport("tapi32.dll")] static extern long lineMakeCall(long hLine, ref long lphCall, string lpszDestAddress, long dwCountryCode, ref LINECALLPARAMS lpCallParams);

lineDial	Dials (parts of one or more) dialable addresses. Asynchronous. **API Example:** [DllImport("tapi32.dll")] static extern long lineDial(long hCall, string lpszDestAddress, long dwCountryCode);

Answering Incoming Calls

Function	Description
lineAnswer	Answers an incoming call. Asynchronous. **API Example:** [DllImport("Tapi32.dll", SetLastError = true)] public static extern int lineAnswer(int hCall, ref string lpsUserUserInfo, int dwSize);

Toll Saver Support

Function	Description
lineSetNumRings	Indicates the number of rings after which incoming calls are to be answered. Synchronous. **API Example:** [DllImport("tapi32.dll")] static extern long lineSetNumRings(long hLine, long dwAddressID, long dwNumRings);
lineGetNumRings	Returns the minimum number of rings requested with lineSetNumRings. Synchronous. **API Example:** [DllImport("tapi32.dll")] static extern long lineGetNumRings(long hLine, long dwAddressID, ref long lpdwNumRings);

Call Privilege Control

Function	Description
lineSetCallPrivilege	Sets the application's privilege to the privilege specified. Synchronous. **API Example:** [DllImport("tapi32.dll")] static extern long lineSetCallPrivilege(long hCall, long dwCallPrivilege);

Call Drop Functions

Function	Description
lineDrop	Disconnects a call, or abandons a call attempt in progress. Asynchronous. **API Example:** [DllImport("Tapi32.dll", SetLastError = true)] public static extern int lineDrop(int hCall, string lpsUserUserInfo, int dwSize);
lineDeallocateCall	Deallocates the specified call handle. Synchronous. **API Example:** [DllImport("tapi32.dll")] static extern long lineDeallocateCall(long hCall);

Call Handle Manipulation

Function	Description
lineHandoff	Hands off call ownership and/or changes an application's privileges to a call. Synchronous. **API Example:** [DllImport("tapi32.dll")] static extern long lineHandoff(long hCall, string lpszFileName, long dwMediaMode);
lineGetNewCalls	Returns call handles to calls on a specified line or address for which the application does not yet have handles. Synchronous. **API Example:** [DllImport("tapi32.dll")] static extern long lineGetNewCalls(long hLine, long dwAddressID, long dwSelect, ref LINECALLLIST lpCallList);
lineGetConfRelated Calls	Returns a list of call handles that are part of the same conference call as the call specified as a parameter. Synchronous. **API Example:** [DllImport("tapi32.dll")] static extern long lineGetConfRelatedCalls(long hCall, ref LINECALLLIST lpCallList);

Location and Country/Region Information

Function	Description
lineTranslateDialog	Displays a dialog box allowing the user to change location and calling card information. Synchronous. **API Example:** [DllImport("tapi32.dll")] static extern long lineTranslateDialog(long hLineApp, long dwDeviceID, long dwAPIVersion, long hwndOwner, string lpszAddressIn);
lineGetCountry	Retrieves dialing rules and other information about a given country/region. Synchronous. **API Example:** [DllImport("tapi32.dll")] static extern long lineGetCountry(long dwCountryID, long dwAPIVersion, ref LINECOUNTRYLIST lpLineCountryList);

Request Recipient Services

The following two functions are used only in support of Assisted Telephony.

Function	Description
lineRegisterRequestRecipient	Registers or deregisters the application as a request recipient for the specified request mode. Synchronous. **API Example:** [DllImport("tapi32.dll")] static extern long lineRegisterRequestRecipient(long hLineApp, long dwRegistrationInstance, long dwRequestMode, long bEnable);
lineGetRequest	Gets the next request from the Telephony dynamic link library. Synchronous. **API Example:** [DllImport("tapi32.dll")] static extern long lineGetRequest(long hLineApp, long dwRequestMode, Any lpRequestBuffer);

6- Remoting TCP / HTTP Channels Members

أولا الـ HTTP Channel ومن أهم الـ Members الخاصة بها:

Member Name	Availability	Description
allowAutoRedirect	Client	A Boolean value that determines whether the client will handle HTTP redirects.
bindTo	Server	A string representing an IP address for the server to bind to.
clientConnectionLimit	Client	An integer defining how many concurrent connections can be opened to the server.
connectionGroupName	Client	A string representing a group name if the unsafeAuthenticatedConnectionSharing property is set.
credentials	Client	An *ICredentials* instance that allows a client to pass credentials for password- based authentication schemes such as Kerberos authentication.
exclusiveAddressUse	Server	A Boolean value that forces the server to use the listening IP address and port exclusively so that another application cannot steal the listening TCP port.
listen	Server	A Boolean value that determines whether to allow activation to hook into the outside listener service.
name	Client and server	A string representing the name of the channel.
port	Server	An integer value that specifies what TCP port the server will listen on or what port the client will send TCP packets from. If the value 0 is selected, the infrastructure will choose a port automatically.
priority	Client and server	An integer value that defines the priority of the channel when multiple channels are available.
proxyName	Client	A string identifying a proxy if the client communication must pass through a proxy.
proxyPort	Client	An integer identifying the port on which a proxy is listening for

Member Name	Availability	Description
		communication from a client.
suppressChannelData	Server	A Boolean value that prevents the server channel from returning channel properties.
timeout	Client	An integer that specifies how long the client channel will wait for a server response.
unsafeAuthenticatedConnectionSharing	Client	A Boolean value that indicates the client will supply credentials and a group name for the connection.
useAuthenticatedConnectionSharing	Client	A Boolean value that tells the server to reuse a connection from an authenticated user.
useIpAddress	Server	A Boolean value that forces the channel to use a specific IP address instead of using the *machineName* property.
machineName	Client and server	A string representing the machine name of the listening server.

ثانيا الـ TCP Channel ومن أهم الـ Members الخاصة بها:

Member Name	Availability	Description
bindTo	Server	A string representing an IP address for the server to bind to.
exclusiveAddressUse	Server	A Boolean value that forces the server to use the listening IP address and port exclusively so that another application cannot steal the listening TCP port.
machineName	Client and server	A string representing the machine name of the listening server.
Name	Client and server	A string representing the name
Port	Server	An integer value that specifies what TCP port the server will listen on or what port the client will send TCP packets from.
Priority	Client and server	An integer value that defines the priority of the channel when multiple channels are available. A larger value indicates greater priority.
rejectRemoteRequests	Server	A Boolean value that determines whether the server will only accept requests from the local host.

Member Name	Availability	Description
suppressChannelData	Server	A Boolean value that prevents the server channel from returning channel properties.
useIpAddress	Server	A Boolean value that forces the channel to use a specific IP address rather than the *machineName* property.

Appendixes (B)

- **ASCII Code Table**
- **References**
- **Useful Sites**

ASCII Code Table:

Table 1:

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space
1	1	001	SOH (start of heading)	33	21	041	!	!
2	2	002	STX (start of text)	34	22	042	"	"
3	3	003	ETX (end of text)	35	23	043	#	#
4	4	004	EOT (end of transmission)	36	24	044	$	$
5	5	005	ENQ (enquiry)	37	25	045	%	%
6	6	006	ACK (acknowledge)	38	26	046	&	&
7	7	007	BEL (bell)	39	27	047	'	'
8	8	010	BS (backspace)	40	28	050	(	(
9	9	011	TAB (horizontal tab)	41	29	051	)	)
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*
11	B	013	VT (vertical tab)	43	2B	053	+	+
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,
13	D	015	CR (carriage return)	45	2D	055	-	-
14	E	016	SO (shift out)	46	2E	056	.	.
15	F	017	SI (shift in)	47	2F	057	/	/
16	10	020	DLE (data link escape)	48	30	060	0	0
17	11	021	DC1 (device control 1)	49	31	061	1	1
18	12	022	DC2 (device control 2)	50	32	062	2	2
19	13	023	DC3 (device control 3)	51	33	063	3	3
20	14	024	DC4 (device control 4)	52	34	064	4	4
21	15	025	NAK (negative acknowledge)	53	35	065	5	5
22	16	026	SYN (synchronous idle)	54	36	066	6	6
23	17	027	ETB (end of trans. block)	55	37	067	7	7
24	18	030	CAN (cancel)	56	38	070	8	8
25	19	031	EM (end of medium)	57	39	071	9	9
26	1A	032	SUB (substitute)	58	3A	072	:	:
27	1B	033	ESC (escape)	59	3B	073	;	;
28	1C	034	FS (file separator)	60	3C	074	<	<
29	1D	035	GS (group separator)	61	3D	075	=	=
30	1E	036	RS (record separator)	62	3E	076	>	>
31	1F	037	US (unit separator)	63	3F	077	?	?

Table 2:

Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
64	40	100	@	@	96	60	140	`	`
65	41	101	A	A	97	61	141	a	a
66	42	102	B	B	98	62	142	b	b
67	43	103	C	C	99	63	143	c	c
68	44	104	D	D	100	64	144	d	d
69	45	105	E	E	101	65	145	e	e
70	46	106	F	F	102	66	146	f	f
71	47	107	G	G	103	67	147	g	g
72	48	110	H	H	104	68	150	h	h
73	49	111	I	I	105	69	151	i	i
74	4A	112	J	J	106	6A	152	j	j
75	4B	113	K	K	107	6B	153	k	k
76	4C	114	L	L	108	6C	154	l	l
77	4D	115	M	M	109	6D	155	m	m
78	4E	116	N	N	110	6E	156	n	n
79	4F	117	O	O	111	6F	157	o	o
80	50	120	P	P	112	70	160	p	p
81	51	121	Q	Q	113	71	161	q	q
82	52	122	R	R	114	72	162	r	r
83	53	123	S	S	115	73	163	s	s
84	54	124	T	T	116	74	164	t	t
85	55	125	U	U	117	75	165	u	u
86	56	126	V	V	118	76	166	v	v
87	57	127	W	W	119	77	167	w	w
88	58	130	X	X	120	78	170	x	x
89	59	131	Y	Y	121	79	171	y	y
90	5A	132	Z	Z	122	7A	172	z	z
91	5B	133	[	[	123	7B	173	{	{
92	5C	134	\	\	124	7C	174			|
93	5D	135	]	]	125	7D	175	}	}
94	5E	136	^	^	126	7E	176	~	~
95	5F	137	_	_	127	7F	177	&#127;	DEL

References:

- Microsoft MSDN
- Bechrouz A. Forouzan, TCP/IP Protocol Suite Third Edition.
- Microsoft, Network Programming for the Microsoft.NET.
- Richard Blum, C# Network Programming, Sybex.
- Fiach Reid, Network Programming in .NET , ELSEVIER.
- Wrox, Professional .NET Network Programming
- O'Reilly Programming.NET Security.
- Tobin Titus , A Press, C-Sharp Threading Handbook

Useful Sites:

- Fadi Abdel-qader, Free Online Courses www.SocketCoder.Com
- All API Functions http://www.pinvoke.net
- Microsoft MSDN, http://msdn.microsoft.com
- C# Corner, www.c-sharpcorner.com
- Code Project www.codeproject.com
- The Definitive .NET Companion http://www.classdotnet.com/
- Ethereal http://www.ethereal.com

ISBN 9957-09-255-3

الإصدار القادم:

سيحتوي الإصدار القادم بمشيئة الله على تحديث لأهم التقنيات الجديدة في برمجة الشبكات والنظم الموزعة كذلك التوسع في شرح برمجة بروتوكولات الـ VOIP مثل الـRTP و الـRTMP والـH.323 وكذلك SIP، رابط الإصدار الجديد على الموقع الرسمي:

http://www.socketcoder.com/ArticleFile.aspx?index=2&ArticleID=67

****

تـــم بحـــــــمد الـــلـــه

****