

2006

# أحترف برمجة الشبكات والنظم الموزعة (النسخة الورقية الكاملة – الإصدار 2006)

.Net Network, Distributed Systems and  
TCP/IP Programming Professional  
Using C# & VB.NET

احترف برمجة الشبكات والنظم الموزعة

Microsoft .NET Framework 2

الكتاب الأول والوحيد المتخصص بهذا المجال  
وباللغة العربية

فادي عبدالقادر



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 2009/11/1 تم أخضاع كتاب (احترف برمجة الشبكات والنظم الموزعة بالدوت نيت 2006) إلى رخصة SCPL وبذلك سيكون كل مافي الكتاب مجاني بالكامل بشكل إلكتروني 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)

---



**Fadi M. Abdelqader**  
Systems Engineer & Developer  
[www.SocketCoder.Com](http://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](http://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 والـ Transport Layer Protocols والـ Network Layer Programming وبناء أنظمة متقدمة باستخدام الـ Multicasting 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 المرسله باستخدام الـ Raw Programming والـ Packet Sniffing ، كما ويحتوي على شرح مفصل لأهم بروتوكولات 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 لا بد 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.

**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)

**Part 1 Networks & TCPIP Programming Overview:**

**Chapter 1:** TCP/IP Layers & Message Encapsulation Overview

- 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

**Chapter 2:** IPv4 & IPv6 Architecture Overview

- IPv4 Architecture
- Classful IP Address
  - Unicast IP
  - Broadcast IP
  - Multicast IP
- CIDR Nation Overview
- IPv6 Architecture Overview

**Chapter 3:** IP Multicasting Overview

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

**Part2 Streaming:**

**Chapter 4:** Streaming (Classes & Members)

- Stream Classes
- Stream Members
- Stream Manipulation

**Chapter 5:** Applied Streaming in Dot Net

- Create a Simple Remote Control Application Using Stream Reader & Writer Classes
- Create a Simple Remote Desktop Application
- 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

## **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
  - o ICMP – Internet Control Message Protocol Programming (Ping & Tracing)
  - o 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**

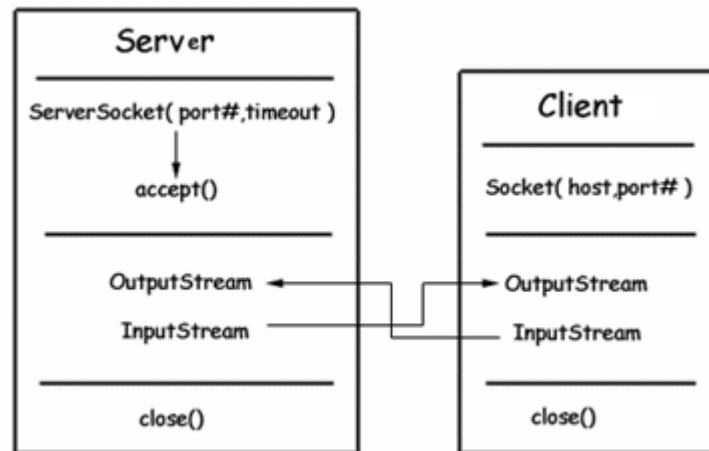
- TCP/IP Layers Encapsulation Overview
- TCP / UDP Connection Establishment
- TCP & UDP Header Encapsulation
- Using TCP Connection Oriented in Dot Net to Send Unicast Messages
- Introduction to Binary Streaming in Dot Net
- Using UDP Connectionless in Dot Net to Send Uni & Broadcast Messages

## 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 ك-Headers على البيانات المرسله. وتم اختصارها إلى خمسة طبقات في بروتوكول TCP/IP وتبين الصورة المرفقة هذه الطبقات:

OSI	TCP/IP
Application	Applications SMTP, POP, HTTP, FTP, Telnet, DNS ...
Presentation	
Session	
Transport	Transport TCP          UDP
Network	Internet IP, ICMP, IGMP, ARP, RARP
Data Link	Network Interface
Physical	Hardware

تبدأ عملية توليف الرسالة المرسله 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 الـ Transport Layer مع الـ Network Layer أو الـ Internet Layer ويوضح الشكل التالي طبيعة عمل الـ Socket :



حيث يربط رقم Port المحدد في Transport Layer مع Destination IP في Network Layer ويقوم Server بالطرف المقابل بالموافقة على طلب Client وتقتصر وظيفة Socket في Server على ربط رقم Port مع Socket Option التي يتم تحديدها ثم البدء بعملية التصنت على Port 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 المرسلّة 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 الـ ASCIIEncoding 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 الـ TCP بمجموعة من المراحل تسمى Three Way Hand Shake أما في الـ UDP 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 الـ Server ذلك
  - 3- قبول أو رفض الجلسة ويتم ذلك بإرسال الموافقة على فتح الجلسة أو رفضها
  - 4- بدأ الجلسة وقيام الـ Server بعمل Listening على الـ Port 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 الـ UDP Connectionless Protocol ففي البروتوكول الأول يتم تحديد طرفين وهما المرسل والمستقبل و Port 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 الـ Multicasting (الإرسال إلى مجموعة Group)، ويتم ذلك بوضع الـ Broadcast IP أو الـ Multicast IP مع رقم الـ Port في الـ Socket (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 mywrite = new StreamWriter (mynetsream);  
mywrite.WriteLine("Your Message");  
mywrite.Close ();  
mynetsream.Close ();  
tcpclnt.Close ();
```

#### **VB.NET:**

```
Dim mynetsream As NetworkStream = tcpclnt.GetStream  
Dim mywrite As StreamWriter = New StreamWriter(mynetsream)  
mywrite.WriteLine("Your Message")  
mywrite.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 الـ Network Layer على مجموعة من البروتوكولات منها IP,IPv6,ARB-Address Resolution Protocol ولكل منها وظيفة معينة سنأتي على شرحها لاحقاً...

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

**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:**

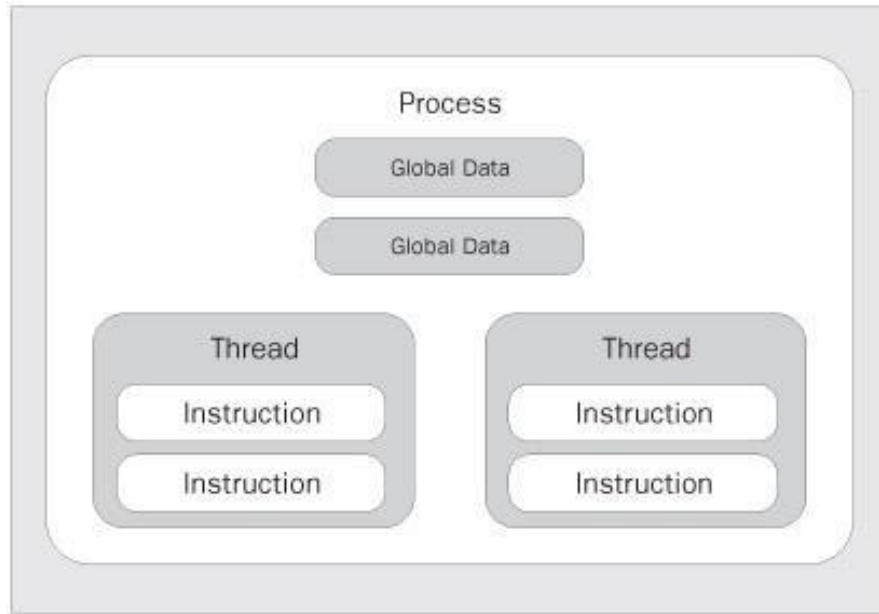
```
Imports System.IO
Imports System.Net.Sockets
Imports System.Threading

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 أي شيء إلا بعد انتهاء الـ Loop وهو ما لن يحدث أبداً إذ أنه 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 أداء 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 أولاً تعريف الـ System.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#:**

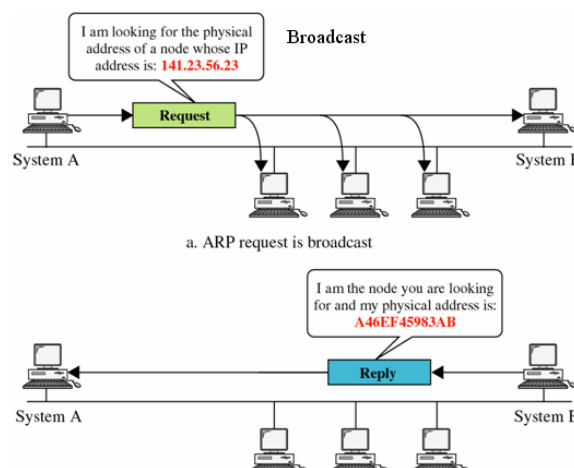
```
private void Form1_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
{
mylistener.Stop();
myth.Abort();
}
```

### **VB.NET:**

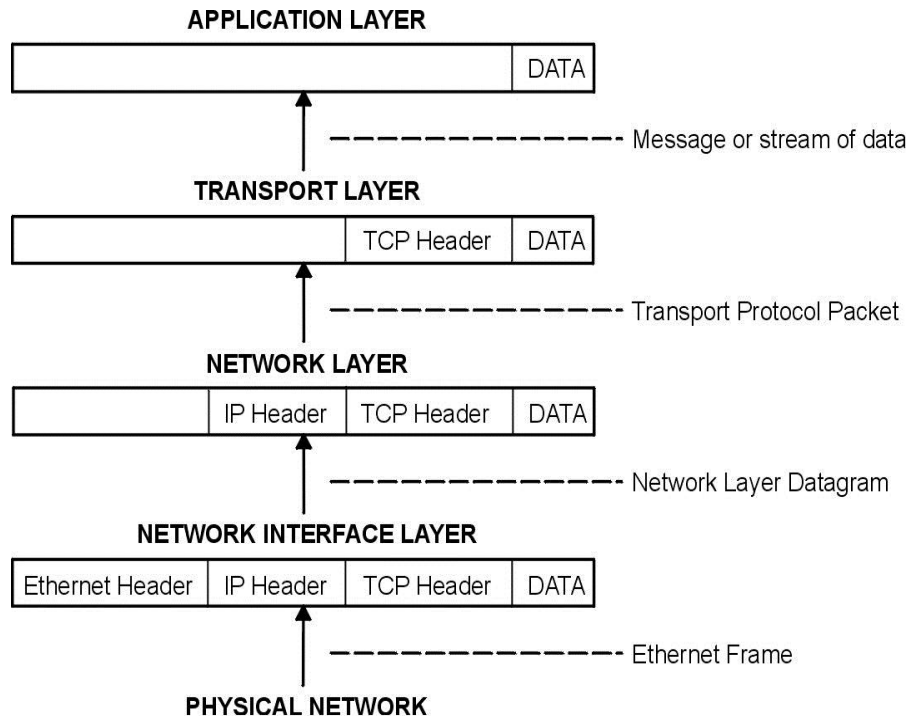
```
Private Sub Form1_Closing(ByVal sender As Object, ByVal e As
System.ComponentModel.CancelEventArgs)
Try
mylistener.Stop()
myth.Aport()
Catch ex As Exception
Msgbox(ex.Message)
End Try
End Sub
```

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...

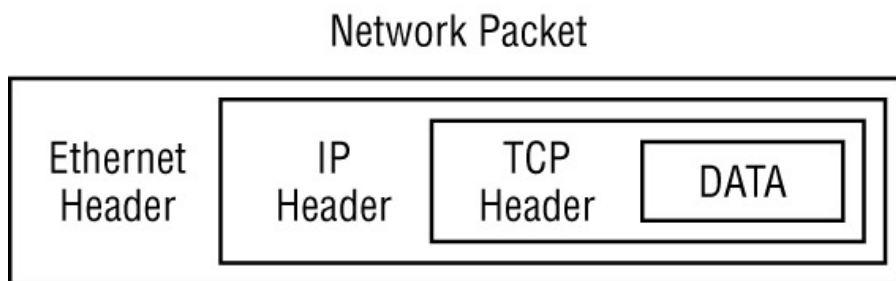
بعد إجراء عملية العنونة يقوم المرسل بسؤال عن عنوان الـ MAC Address الخاص بالـ Server وتتم هذه العملية عبر بروتوكول الـ ARP-Address Resolution Protocol ويقوم هذا البروتوكول بالتحقق من وجود الـ MAC Address في الـ MAC Table وفي حالة عدم وجوده يقوم بإرسال الـ ARB Broadcast Message Request إلى كل الأجهزة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



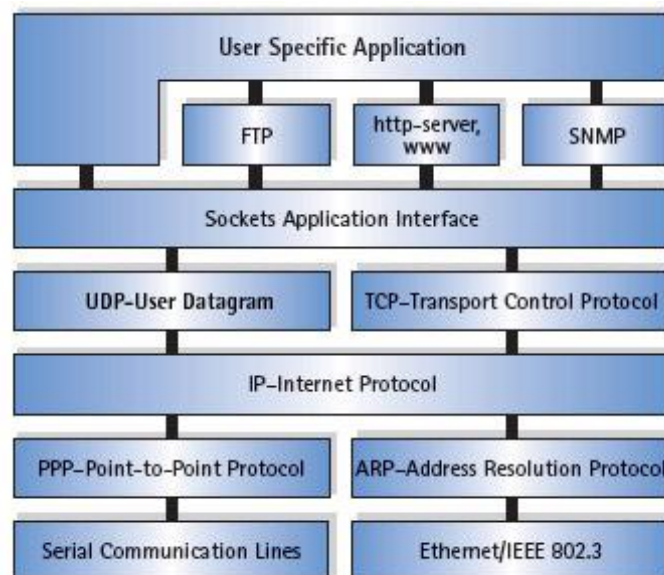
وبعد هذه المرحلة يتم تحديد نوع الـ Encapsulation هل سيكون كـ Internet Encapsulation أو Ethernet Encapsulation ويتم ذلك في الـ Data Link Layer والشكل التالي يوضح كل هذه العمليات:



وفي النهاية يكون الشكل العام للـ Packet كما يلي:



ويجب التفريق بين عملية الإرسال باستخدام الـ Ethernet وعملية الإرسال باستخدام الـ Internet أو الـ Serial Connection إذ أنه في الـ Ethernet تتم عملية الإرسال بعد معرفة الـ MAC Address لطرف الآخر باستخدام الـ ARP أما في الـ Internet فيتم الوصول وفق مبدأ الـ PPP – Point to Point Protocol لاحظ الشكل التالي:



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

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

### مثال على استخدام الـ ARB والـ DNS والـ HTTP :

لاحظ الشكل التالي في برنامج الـ Ethereal والذي يقوم بعمليات التصنت والتحليل للبيانات المارة من خلال الـ Network Interface Card ، حيث قمنا بطلب الدخول على موقع الـ Google باستخدام الـ Internet 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 الـ MAC Address الخاص بالـ Gateway وهو هنا 10.0.0.138 حيث أرسلت هذه الرسالة كـ Broadcast Message وفي العملية الثانية قام الـ Router بالرد على الـ ARP Request وأرسل عنوان الـ MAC Address الخاص به بعد هذه العملية سيقوم الـ Client بالاستفسار عن عنوان الـ Google بإرسال الـ DNS Request Query ، لاحظ أنه في المثال قد وجد الـ IP Address الخاص بالـ Google في الـ DNS

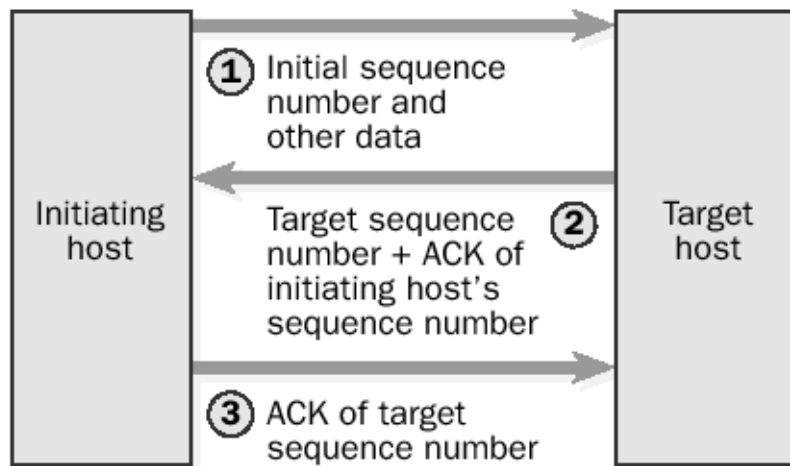
Table والموجودة ضمن نظام التشغيل وهذا يعني انه تم الدخول على الـ Google من قبل والدليل على ذلك أن الـ IP Address الخاص بالـ Google موجود في الـ DNS Table وبعد هذه العملية قام موقع الـ Google بالرد على الطلب يخبره بوجوده وضمن العنوان المحدد لاحظ أن العمليات من 8 إلى 20 هي عمليات طلب و إرسال لمحتويات الصفحة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

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
17	12.976418	64.233.183.103	10.0.0.10	TCP	[TCP window Update] http > 1717 [ACK] Seq=
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=65
21	14.013510	Planette_30:6e:c9	Spanning-tree-(for STP	RST.	Root = 32768/00:12:a9:67:23:20 Cost
22	16.015309	Planette_30:6e:c9	Spanning-tree-(for STP	RST.	Root = 32768/00:12:a9:67:23:20 Cost
23	16.229503	10.0.0.10	64.233.183.103	TCP	1717 > http [RST, ACK] Seq=346 Ack=1677 W

Frame 15 (399 bytes on wire, 399 bytes captured)  
 Ethernet II, Src: 10.0.0.10 (00:11:09:d0:13:21), Dst: 10.0.0.138 (00:0e:50:8d:f2:50)  
 Internet Protocol, Src: 10.0.0.10 (10.0.0.10), Dst: 64.233.183.103 (64.233.183.103)  
 Transmission Control Protocol, Src Port: 1717 (1717), Dst Port: http (80), Seq: 1, Ack: 1, Len: 345  
 Hypertext Transfer 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 الآخر 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 الـ Server ويكون هذا الرقم المولد نقطة البدء لعملية الإرسال بحيث يتم زيادته بمقدار واحد ( increment By One ) عند كل عملية إرسال.

2- يستلم الطرف المقابل الـ Sequence 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 الـ Acknowledgment الذي أرسل من قبل المستقبل.

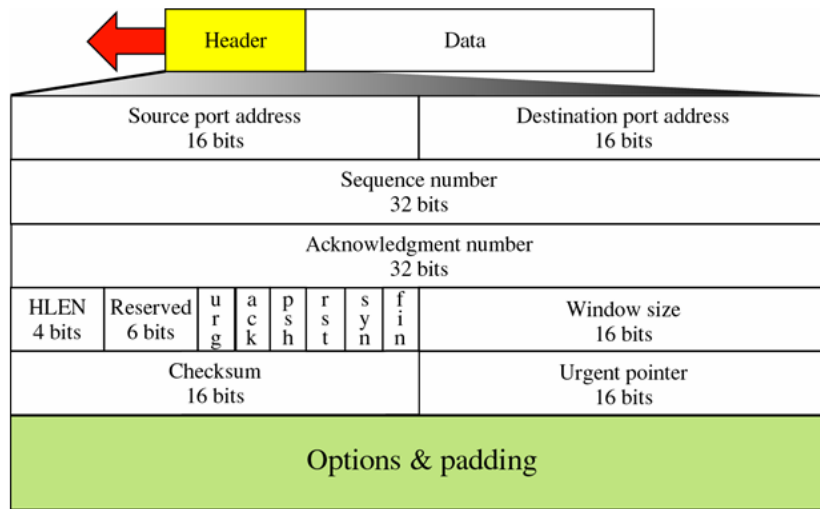
أما في بروتوكول الـ UDP فنتم بدون إرسال Acknowledgments ولا يتحقق الـ UDP من عمليات الوصول كما هو الحال في الـ TCP وهو ما سيتم توضيحه في الجزء التالي من هذا الفصل.

### 1.3 TCP & UDP Header Encapsulation

في طبقة الـ Transport Layer يتم التعامل مع إحدى البروتوكولين TCP أو الـ UDP حيث سنحدد فيها طبيعة الإرسال فإذا كان المطلوب هو الإرسال كـ Stream ونوع الإرسال هو Unicast فيتم اختيار الـ TCP أما في حالة كان المطلوب هو الإرسال كـ Broadcast أو Multicast فيتم اختيار الـ UDP لعملية الإرسال ، وسوف نبين في الجزء التالي من هذا الفصل مبدأ عملية الإرسال باستخدام بروتوكول الـ TCP وبروتوكول الـ UDP ...

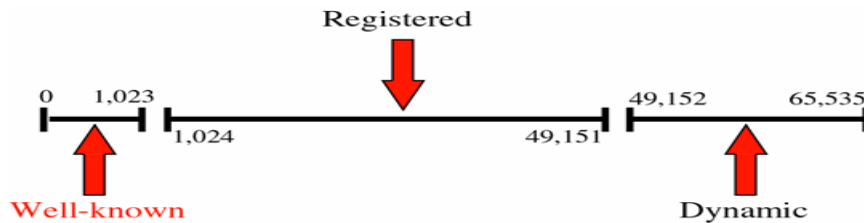
#### 1.3.1 TCP Encapsulation

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



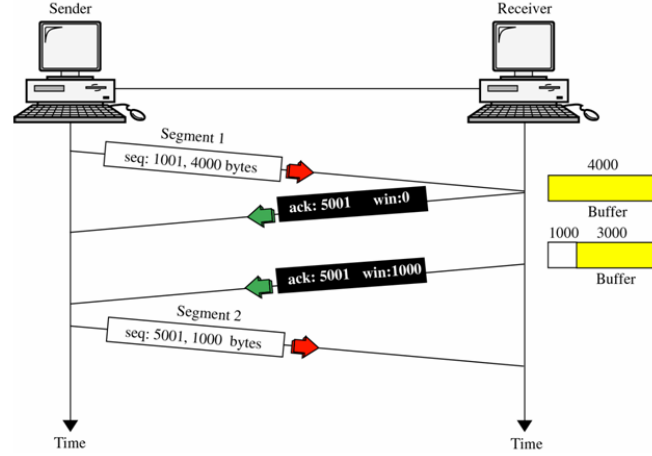
#### 1.3.1.1 Source & Destination Port

في المنطقة المخصصة بـ Port الإرسال والاستقبال ، يتم تحديد الـ Source Port والـ Destination Port الخاصة بالبرنامج الذي يجري عملية الاتصال ومن المعروف أنه لا يمكن لأكثر من برنامج استخدام نفس رقم الـ Port لكن يمكن للبرنامج الواحد استخدام أكثر من Port Address ، وبتأكيد فإن عملية اختيار الـ Port 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 الـ HTTP والذي يستخدم الـ Port 80 وبروتوكول الـ FTP والذي يستخدم الـ Port 21 وغيره، ويفضل عند اختيار الـ Port 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 للـ Packet 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 الـ Header الخاص بالـ TCP على 32 Bits مقسمة على الـ Windows Size 16 Bits وأخرى للـ Header Length + Validation Controls لاحظ الشكل التالي:

HLEN 4 bits	Reserved 6 bits	u r g	a c k	p s h	r e s t	s y n	f i n	Window size 16 bits
----------------	--------------------	-------------	-------------	-------------	------------------	-------------	-------------	------------------------

ويحتوي الـ Header Length على حجم الـ Header الخاص بالـ TCP مقسوم على 4 أي لمعرفة حجم الـ Header نضرب الـ HLEN (Header Length) بـ 4 ، 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	RST: Reset the connection
ACK: Acknowledgment is valid	SYN: Synchronize sequence numbers
PSH: Request for push	FIN: Terminate the connection

URG	ACK	PSH	RST	SYN	FIN
-----	-----	-----	-----	-----	-----

### 1.3.1.4 Window Size

ويعرف فيه حجم الـ Packet 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 بناء على أدائية الشبكة.

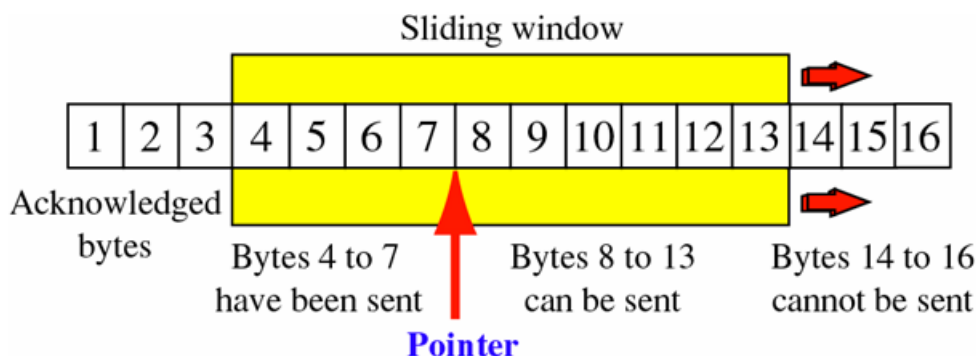
### : Check Sum 1.3.1.5

وهي 16 Bits وتستخدم لعملية التحقق من وصول الـ TCP Header بشكل السليم حيث يتم جمع كافة قيم الـ TCP Header (كل 16 Bits لوحدها) ثم قلبها ووضع الناتج في الـ Check Sum وفي الطرف المستقبل يقوم بتأكد من الـ Checksum بضرب قيمة الـ HLEN بـ 4 ثم مقارنة الناتج مع مقلوب الـ Checksum وكما هو واضح في الشكل التالي والذي يظهر الـ Checksum في الـ IP Protocol إذ انها تتم بنفس الطريقة بالـ TCP والـ UDP:

4	5	0	28	
1			0	0
4	17	0		
10.12.14.5				
12.6.7.9				
4, 5, and 0	→	01000101	00000000	
28	→	00000000	00011100	
1	→	00000000	00000001	
0 and 0	→	00000000	00000000	
4 and 17	→	00000100	00010001	
0	→	00000000	00000000	
10.12	→	00001010	00001100	
14.5	→	00001110	00000101	
12.6	→	00001100	00000110	
7.9	→	00000111	00001001	
<hr/>				
Sum	→	01110100	01001110	
Checksum	→	10001011	10110001	

### : Urgent Pointer 1.3.1.6

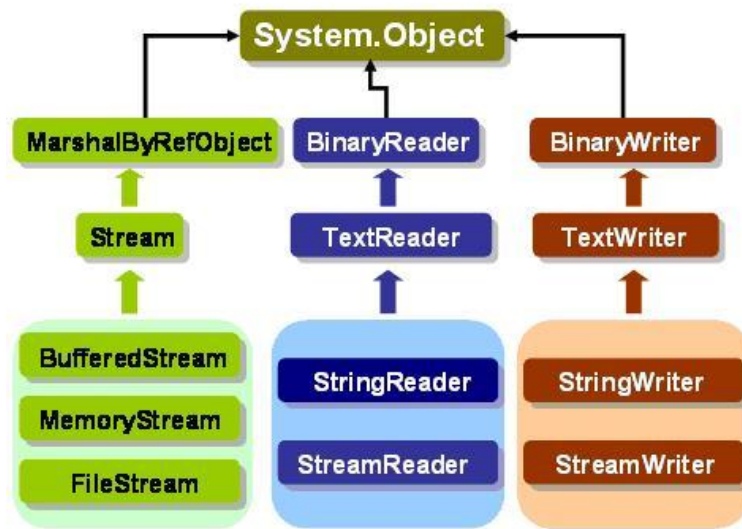
من المعروف أن الـ Data المرسلة عبر الـ TCP يتم تجميعها في الـ Buffer قبل أن يتم عرضها حيث يتم تحديد موقع الـ Data القادمة الجديدة في الـ Buffer ومن هنا نحن بحاجة إلى وجود Pointer يؤشر على موقع الـ Data في الـ Buffer وهو هنا الـ Urgent Pointer لاحظ الشكل التالي والذي يوضح عملية وضع الـ Data قادمة جديدة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



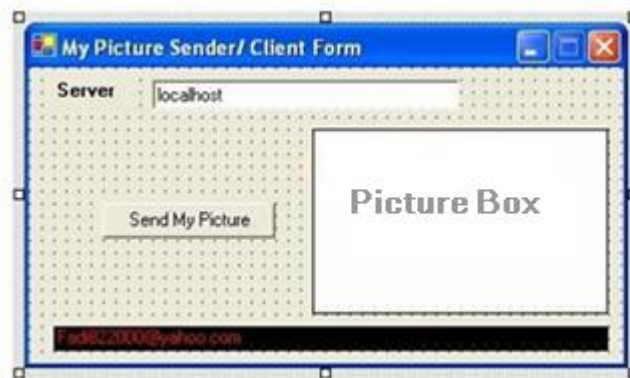


### : Dot Net Binary Streaming Overview : 1.3.2

يمكن من خلال الـ Stream Classes نقل Text أو Binary Data ، و بينا سابقا كيفية التعامل من الـ Socket لنقل Text باستخدام الـ Stream Reader والـ Stream Writer وفي هذا الجزء سنبين كيفية التعامل معه لنقل Object (أي نوع آخر 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 كـ Object Serialization (انظر الفصل الخامس عشر) أو باستخدام الـ Binary Reader و الـ Binary Writer والتي تمكنك من التعامل مع أي نوع من البيانات ، كما وتستخدم الـ Stream Reader والـ Stream Writer للتعامل 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 الـ Stream إلى مجموعة من الـ Bits والـ Bits إلى Stream مرة أخرى لاحظ الشكل التالي والذي يوضح تسلسل مكتبات الـ Stream في الدوت نيت:



وكمثال تطبيقي على هذا سوف نقوم ببناء برنامج يقوم بعملية نقل صورة Image من جهاز إلى آخر Client/Server وللبداء 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 أولاً من اشتقاق Instance من الكلاس MemoryStream والتي سوف نستخدمها لتخزين الصورة داخل الذاكرة بشكل مؤقت لكي نحولها لاحقاً إلى مصفوفة Binary ثم إرسالها باستخدام NetworkStream عبر الـ Socket إلى الـ Server:

**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 بال-Server تتم عملية التصنت على الـ Port واستقبال الـ Stream عبر الـ Socket 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 المرسله من خلال `Stream` لكي يتم تحويلها من `Stream` إلى صورة تعرض على الـ `PictureBox` 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 mytcp1; // Declare TCP Listener
Socket mysocket; // Declare an object from Socket Class
NetworkStream myns; //
StreamReader mysr;

```

```

void Image_Receiver()
{
mytcp1 = new TcpListener (5020); // Open The Port
mytcp1.Start (); // Start Listening
mysocket = mytcp1.AcceptSocket ();
myns = new NetworkStream (mysocket);
pictureBox1.Image = Image.FromStream(myns); // Show The Image
that Resaved as Binary Stream
mytcp1.Stop(); // Close TCP Session
if (mysocket.Connected == true) // if Connected Start Again
{
    while (true)
    {
        Image_Receiver(); // Back to First Method
    }
}
}

```

### **VB.NET:**

```

Private mytcp1 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()
    mytcp1 = New TcpListener(5000)
    mytcp1.Start()
    mysocket = mytcp1.AcceptSocket
    myns = New NetworkStream(mysocket)
    pictureBox1.Image = Image.FromStream(myns)
    mytcp1.Stop()
    If mysocket.Connected = True Then
        While True
            Image_Receiver()
        End While
    End If
End Sub

```

ولتنفيذ الـ Image\_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 المستقبلية:

**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:**

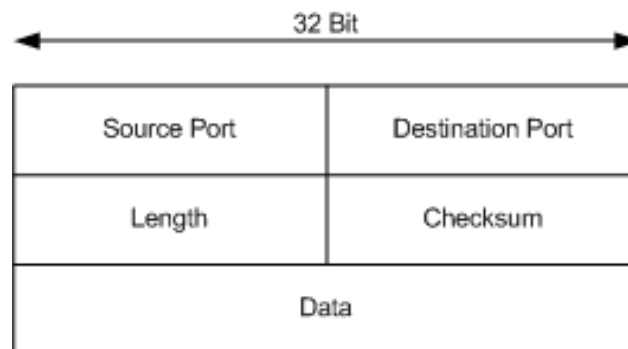
```

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

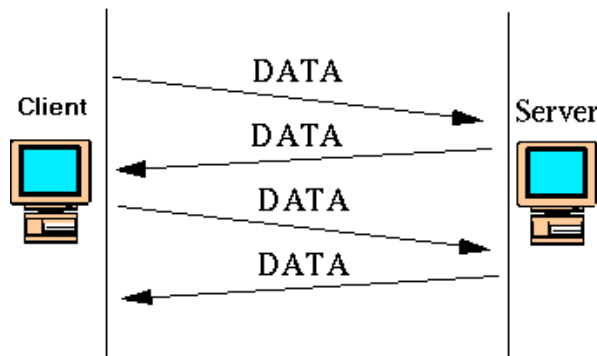
```

### **UDP-User Datagram Protocol- Encapsulation : 1.3.3**

سوف نبين في هذا الجزء طبيعة الاتصال باستخدام الـ UDP حيث يتميز هذا البروتوكول بإمكانية الإرسال كـ Multicast و Broadcast بعكس الـ TCP الذي يدعم الإرسال كـ Unicast فقط ، لكن مشكلة هذا البروتوكول هو عدم دعمه لعمليات التحكم على مستوى الـ Data Flow أو حجم الـ Buffer 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:



الـ Check Sum و الـ Length أو الـ Header Length هي نفسها في الـ TCP لكن لاحظ عدم وجود أي من الأمور الخاصة بالـ Buffer Management أو الـ Delivered On Sequence في الـ Header الخاص بالـ UDP ، والمشكلة هنا أننا لا نستطيع عمل الـ Fragmentation للـ Packets حيث أن إعادة تجميعها بالترتيب الصحيح أمر غير مضمون، كما أنه لا وجود لأي Acknowledgment للتحقق من وصول البيانات أو عدم وصولها ، لاحظ الشكل التالي والذي يوضح طبيعة التراسل باستخدام الـ UDP :



لاستخدام الـ UDP في الدوت نيت يلزم أولاً تعريف System.Net Namespace والـ System.Net.Socket لاحظ انه في الـ TCP كان يلزم تعريف رقم الـ Port والعنوان للجهاز المستقبل أما في الـ UDP فتستطيع تعريفه كما هو في TCP كما وتستطيع عمل Broadcast باستخدام IPAddress.Broadcast عند الإرسال بعد اشتقاق كائن من الكلاس IPAddress و IPEndPoint وتستطيع أيضاً عدم تحديد رقم الـ Port وذلك بتمرير الرقم صفر 0 إلى الـ Bind Method في الـ Socket المخصصة للاستقبال.

في المثال التالي يتم فتح الـ Port 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 الـ Socket وتحديد نوع البروتوكول 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 الـ Port بالـ Socket :

3)

**C#:**

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

**VB.NET:**

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

الآن تم استقبال الرسالة ونريد بثها إلى كل من يتصل مع الـ Server على الـ Port السابقة ولعمل ذلك يلزم أولاً 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 الـ Server 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 الـ Server 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 تحدده 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 الـ Server في حالة ورود نص معين وهكذا...

### **C#:**

```

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

class SimpleUdpSrvr
{
    public static void Main()
    {
        int recv;
        byte[] data = new byte[1024];
        IPEndPoint ipep = new IPEndPoint(IPAddress.Any, 5020);
        Socket newsock = new Socket(AddressFamily.InterNetwork,
            SocketType.Dgram, ProtocolType.Udp);
        newsock.Bind(ipep);
        Console.WriteLine("Waiting for a client...");
        IPEndPoint sender = new IPEndPoint(IPAddress.Any, 0);
        EndPoint Remote = (EndPoint)(sender);
        recv = newsock.ReceiveFrom(data, ref Remote);
        Console.WriteLine("Message received from {0}:",
Remote.ToString());
        Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));
        string welcome = " Welcome Customer ...";
        data = Encoding.ASCII.GetBytes(welcome);
        newsock.SendTo(data, data.Length, SocketFlags.None, Remote);
    }
}

```

```

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

```

### **VB.NET:**

```

Imports System
Imports System.Net
Imports System.Net.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

```

ثانياً الجزء الخاص بالـ Client ، يقتصر العمل هنا على قيام الـ Client بإنشاء جلسة مع الـ Server وذلك بعد تعريفه بالـ IPAddress ورقم الـ Port وكما تم في السابق.

```
C#:
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
class SimpleUdpClient
{
public static void Main()
{
byte[] data = new byte[1024]; string input, stringData;
EndPoint ipep = new EndPoint( 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);
EndPoint sender = new EndPoint(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.Sockets
Imports System.Text
Class SimpleUdpClient
    Public Shared Sub Main()
        Dim data(1024) As Byte
        Dim input As String
        Dim stringData As String
        Dim ipep As IPEndPoint = New
        IPEndPoint(IPAddress.Parse("127.0.0.1"), 5020)
        Dim server As Socket = New Socket(AddressFamily.InterNetwork,
        SocketType.Dgram, ProtocolType.Udp)
        Dim welcome As String = "Hello, are you there?"
        data = Encoding.ASCII.GetBytes(welcome)
        server.SendTo(data, data.Length, SocketFlags.None, ipep)
        Dim sender As IPEndPoint = New IPEndPoint(IPAddress.Any, 0)
        Dim Remote As EndPoint = CType(sender, EndPoint)
        data = New Byte(1024)
        Dim recv As Integer = server.ReceiveFrom(data, Remote)
        Console.WriteLine("Message received from {0}:",
        Remote.ToString())
        Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv))
        While True
            input = Console.ReadLine
            If input = "exit" Then
                ' break
            End If
            server.SendTo(Encoding.ASCII.GetBytes(input), Remote)
            data = New Byte(1024)
            recv = server.ReceiveFrom(data, Remote)
            stringData = Encoding.ASCII.GetString(data, 0, recv)
            Console.WriteLine(stringData)
        End While
        Console.WriteLine("Stopping client")
        server.Close()
    End Sub
End Class
```

ويمكن تطبيق الـ Broadcast Address على الـ UDP Socket وذلك بعد تفعيلها من الـ Socket 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 RecvBroadcast
{
    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 RecvBroadcast
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 الـ TCP/IP واستخدام الـ TCP والـ UDP في الدوت نيت، سيتم الحديث في الفصل التالي عن Network Layer Encapsulation ومعمارية بروتوكول الـ IPv4 والـ IPv6 ...

---

## **Chapter 2**

### **IPv4 & IPv6 Architecture Overview**

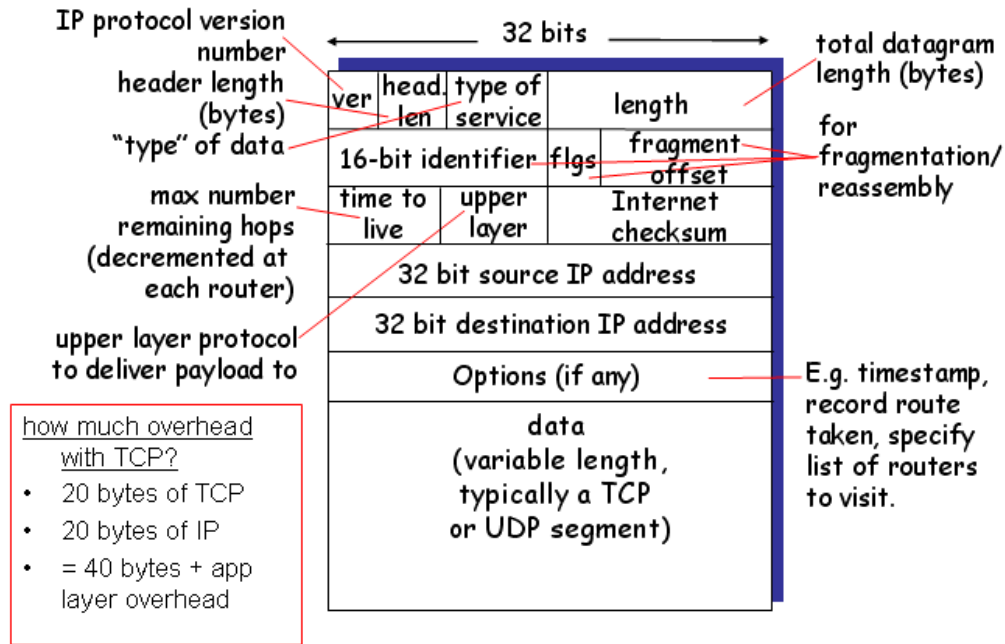
- IPv4 Architecture
- Classful IP Address
  - i. Unicast IP
  - ii. Broadcast IP
  - iii. Multicast IP
- CIDR Nation Overview
- IPv6 Architecture Overview

## :The Internet Protocol

في الـ Network Layer تتم عملية عنونة الـ Packet باستخدام بروتوكول الإنترنت IP، وتتم هذه المرحلة بناءاً على البروتوكول المستخدم في الـ Transport Layer فإذا تم استخدام الـ TCP عندها لا نستطيع إلا أن يكون العنوان المستخدم هو Unicast أما في حالة كان الـ UDP 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 الـ Unicast والـ Multicast والـ Broadcast فيه:

### IPv4 Architecture : 2.1

يوضح الشكل التالي التركيب العام لبروتوكول الإنترنت IPv4 :



يتراوح حجم الـ Header الخاص بالـ IPv4 من 20 إلى 60 Bytes بناءاً على الـ Options المستخدم فإذا لم يتم استخدام الـ Options عندها سيكون حجم الـ Header ثابت وهو 20 Bytes وفي هذه الحالة يمكننا معرفة حجم الـ Data المرسل عبر الـ TCP كما يلي:

$$\text{Data} = \text{total Length} - (\text{20 Bytes for IPv4 Header} + \text{20 Bytes For TCP Header})$$

وتقسم الـ 20 Bytes الخاصة بالـ IPv4 Header كما يلي:

**4 Bits** يوضع فيها الـ Version المستخدم وهو هنا IPv4 ويتم تمثيله 0100 في الـ Binary.  
**4 Bits** للـ Header Length ويوضع في حجم الـ Header مقسوم على 4  
**8 Bits** للـ Type of Services حيث يمثل فيها مدى جودة الخدمة المطلوبة للبروتوكول المستخدم في الـ Application Layer ، ومنها Minimum Delay أو الـ Maximum Services وغيرها...

**32 Bits** وتستخدم لعمليات الـ Fragmentation وإعادة ترتيب الـ Fragments  
**8 Bits** وتستخدم لتحديد الـ TTL - Time to Live ويعبر عن عدد الـ Hops أو الـ Routers التي يسمح للـ Packet 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 الـ Upper Layer سواء TCP أو UDP ...  
**16 Bits** للـ Checksum ولا تختلف طريقة حسابه عن الطريقة المستخدمة في الـ TCP أو UDP 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 الـ Unicast والـ Broadcast  
 والـ Unicast باستخدام الـ Classful والـ CIDR - Classless InterDomain Routing :

### Classful IP Address : 2.1.1

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

	8 bits	8 bits	8 bits	8 bits
<b>Class A:</b>	Network	Host	Host	Host
<b>Class B:</b>	Network	Network	Host	Host
<b>Class C:</b>	Network	Network	Network	Host
<b>Class D:</b>	Multicast start 224.0.0.1			
<b>Broadcast:</b>	Network	255 11111111	255 11111111	255 11111111

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

	From	To
<b>Class A</b>	0.0.0.0 Netid Hostid 10000000 0 ..... 0	127.255.255.255 Netid Hostid 10111111 1 ..... 1
<b>Class B</b>	128.0.0.0 Netid Hostid 11000000 0 ..... 0	191.255.255.255 Netid Hostid 11011111 1 ..... 1
<b>Class C</b>	192.0.0.0 Netid Hostid	223.255.255.255 Netid Hostid
<b>Class D</b>	224.0.0.0 Multicast Address	239.255.255.255 Multicast Address
<b>Class E</b>	240.0.0.0 Reserved	255.255.255.255 Reserved

ولاستخراج عنوان الـ Broadcast من أي من التقسيمات السابقة يتم تعبئة الـ Bits الخاصة بالـ Host ID بواحد وكمثال لمعرفة الـ Broadcast Address للعنوان 10.0.0.1 يجب أولاً تحديد إلى أي Class ينتمي هذا العنوان، ومن الواضح أنه ينتمي إلى الـ Class A لأن الجزء الخاص بالـ Network ID يبدأ بـ 10 وهو بين 1 والـ 126، إذا الجزء الخاص بالـ Host ID

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

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

### CIDR - Classless InterDomain Routing IP Address : 2.1.2

وتتم هذه العملية باستئجار مجموعة من الـ Bits الخاصة بالـ Network Part وضمها إلى الـ Host Part وكمثال لتوليد 1024 عنوان جديد من العنوان 192.168.1.0/24 نقوم باستئجار 2 Bits من الـ 24 Bits الخاصة بالـ Network ID عندها يصبح الـ Subnet كما يلي : 192.168.1.0/22 ويصبح للـ Host ID ، 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 الـ Addresses Range ستبدأ بـ 192.168.252.0 وستنتهي بـ 192.168.255.254 إذا سيكون الـ Broadcast IP هو : 192.168.255.255 ...

لاحظ أن مشكلة المحدودية للعناوين قد حلت بشكل جزئي باستخدام الـ CIDR لكن مازالت الإمكانيات محدودة إذا أردنا توليد عناوين لملايين من الأجهزة وكان الحل في الـ IPv6 حيث زاد فيه حجم العناوين من 32 Bits إلى 128 Bits والذي سيتم الحديث عنه في الجزء التالي من هذا الفصل.

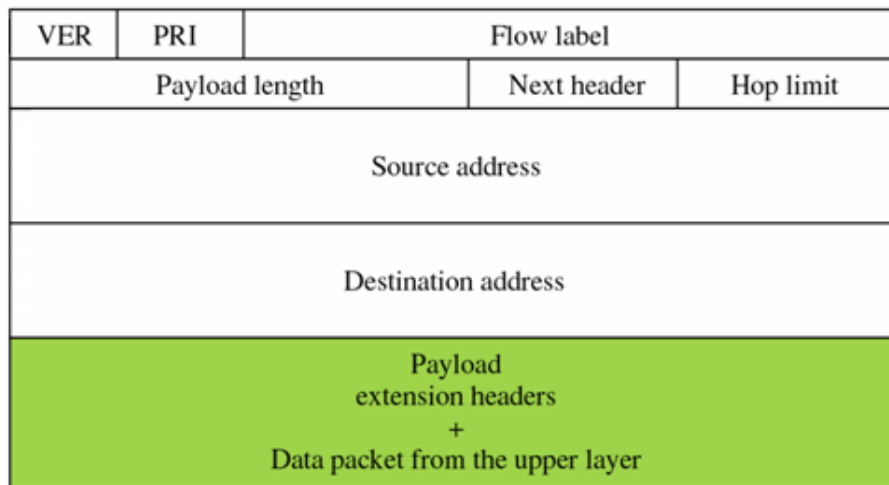
### IPv6 Architecture : 2.2

الـ IPv6 وهو الجيل التالي للـ IPv4 حل الـ IPv6 الكثير من المشاكل التي تواجه الـ IPv4 ونلخصها بمجموعة من النقاط:

- الحاجة إلى عناوين أكثر حيث أن القيمة العظمى للـ IPv4 Address تبقى محدودة مع زيادة الطلب على عناوين الـ IP في العالم.
- قد يؤدي الـ IPv4 إلى مجموعة من المشاكل وخاصة في الـ Routing Table والتي قد تم حلها في الـ IPv6
- مشكلة الـ Security في الـ IPv4 حيث لم يدعم أي من عمليات التشفير والتحقق الـ Authentication على مستوى الـ Network Layer وقد حلت هذه المشكلة في الـ IPv4

باستخدام بروتوكول الـ IPSec حيث يتم تشفير الـ IP والـ Port على مستوى الـ Socket لكن أصبحت الحاجة ملحة لجعل هذه الـ Security مدمجة على مستوى الـ Network Layer .  
- تطوير مبدأ الـ Broadcast حيث تم تطويره إلى الـ any cast إذ يتم إرسال رسالة واحدة إلى كل جهاز على الشبكة وفي حالة وجد الجهاز المعني يتوقف الـ Router عن الإرسال ، والهدف في هذه الطريقة إيجاد طريقة تخفض من الـ Bandwidth المستخدم عند البحث عن جهاز ما على الشبكة.

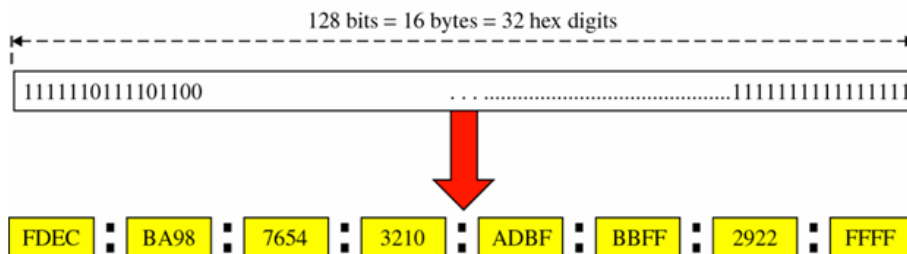
حل الـ IPv6 كل هذه المشاكل حيث دعم ما يعادل  $2^{128}$  Addresses وهو رقم كبير جدا كما قلل من حجم الـ Routing Table مما سرع عمليات التوجيه Routing كما دعم عمليات الـ Authentication والتشفير على مستوى الـ Network Layer كم تم حذف الـ Type of Services وحل محلها الـ Priority أي الأولويات، ويوضح الشكل التالي الـ Header الخاصة بالـ IPv6 :



يستخدم الـ IPv6 الـ hexadecimal بدلا من الـ Decimal في الـ IPv4 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 بالـ IPv4 ، في كل منزلة يوضع بها 16 Bits. ولتمثيله في الـ Hexadecimal نعطي 4 Digits لكل منزلة فيه وكما في الشكل التالي:



وتقسم العناوين في الـ IPv6 إلى نوعين Geographic- و Provider-Based Addresses based Addresses حيث يقوم الـ ISP بتوزيع العناوين على الـ Clients باستخدام الـ Standard الخاص الـ 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:

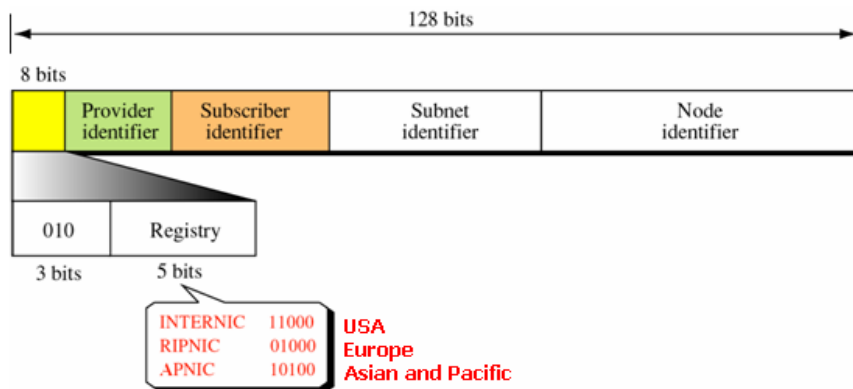
- Provider-Based Addresses

- 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 الـ IPv4 والـ IPv6 وكيفية توليد عناوين الـ Unicast والـ Broadcast باستخدام الـ Classful والـ CIDR Nation .

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

---

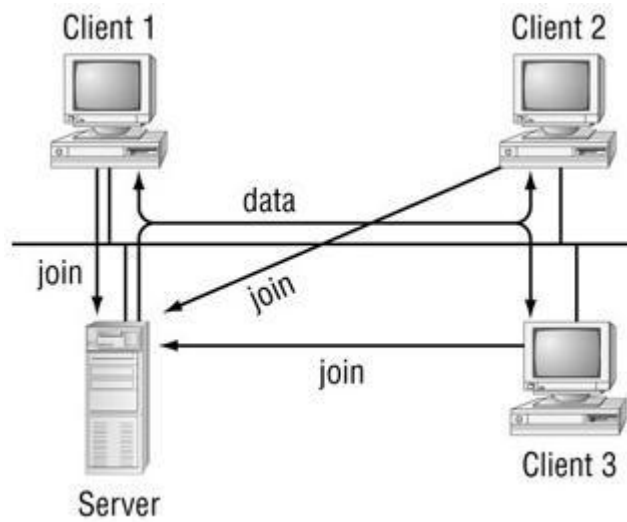
## **Chapter 3**

### **IP Multicasting Programming Overview**

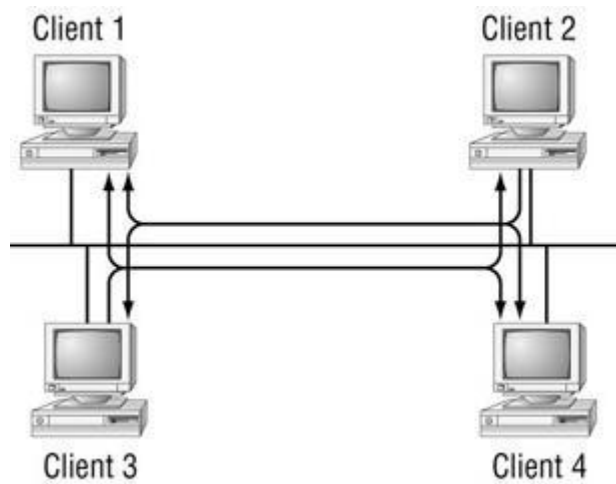
- IP Multicasting Overview
- Using IP Multicasting in Dot Net to Create a Multicast Groups

### : IP Multicasting Overview :3.1

تحدثنا في الفصل الأول عن بروتوكول الـ UDP وشرحنا كيفية استخدامه لعمل برود كاست حيث تستطيع عمل البرود كاست بطريقتين إما بوجود Server يقوم بعملية التصنت على الـ Port المحدد و يستقبل من خلاله أي رسالة ثم يقوم ببثها إلى كل الأجهزة أو باستخدام الـ IPAddress.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 خلاله أي رسالة إلى كل الأجهزة وتشبه عملية الـ Broadcast 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 أغلب برمجيات الـ Conferences تعتمد عليه بشكل كبير ويعرف Multicast على أنه 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 الـ Server على الطلب يقوم بضم عنوان الجهاز إلى الـ IP Address List الخاصة به وتشارك 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 Internetnetwork

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 وإلى المجموعة (multicast group join & leave) كما يستخدم لإضافة وإلغاء العضوية AddMembership و DropMembership بالإضافة إلى العمليات المتعلقة بالـ Multicast Routing ، ويمكن استخدام الـ UdpClient Object لتحديد رقم الـ Port 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 يتصل على هذا الـ Port ويستخدم نفس الـ IP Multicast استقبال هذه الرسالة ، يستخدم الكود التالي لإرسال رسالة إلى عدة جهات بحيث نستخدم رقم الـ Port 5020 و ضمن الـ Group 224.100.0.1 كمثال:

**C#:**

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
class MultiSend
{
    public static void Main()
    {
        Socket server = new Socket(AddressFamily.InterNetwork,
            SocketType.Dgram, ProtocolType.Udp);
        IPEndPoint iep = new IPEndPoint(IPAddress.Parse("224.100.0.1"),
            5020);
        byte[] data = Encoding.ASCII.GetBytes("This is a test message");
        server.SendTo(data, iep);
    }
}
```

```

        server.Close();
    }
}

```

### **VB.NET:**

```

Imports System
Imports System.Net
Imports System.Net.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.Parse("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 الـ Socket وتحديد الجهة التي سوف تستقبل الرسالة وهي (أي Socket يمتصت على الشبكة باستخدام الـ IP Multicast Group المحدد) ثم تحديد نوع الـ Socket والبروتوكول 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 الـ TTL للـ Multicast Packet ويعبر عن عدد الـ Routers التي يسمح للـ Packet 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 آخر 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 ، و يمكنك أيضا عمل برنامج لإرسال صورة عبر الكاميرا إلى جهات متعددة باستخدام نفس الخاصية والتي سأتي على شرحها في الفصل التاسع

... Advanced Multicast Systems

## **Part 2**

### **Streaming in Dot Net**

Chapter4 Streaming in Dot Net

Chapter5 Applied Streaming in Dot Net

---

## **Chapter 4**

### **Streaming in Dot Net**

#### **Managed I/O: Streams, Readers, and Writers**

- Stream Classes
- Stream Members
- Stream Manipulation

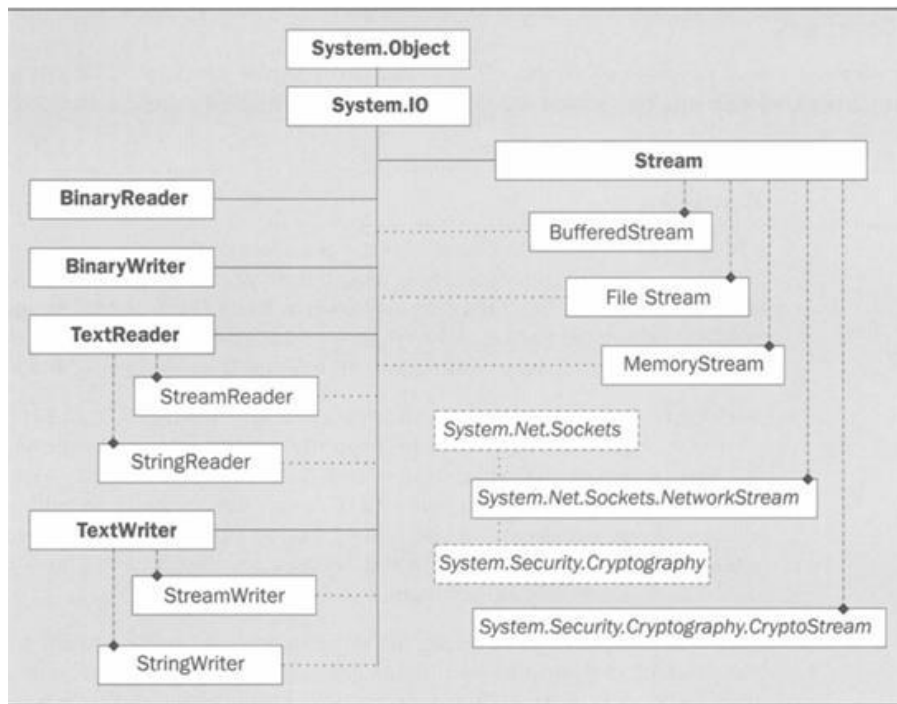
## **: Managed I/O: Streams, Readers, and Writers : 2.1**

إن الهدف من إنشاء مكتبات الـ Stream تسهيل عملية نقل البيانات من مكان إلى آخر سواء عبر الشبكة أو ضمن الـ Local 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 الـ Stream بأسلوبين المتزامن **Synchronous** والغير متزامن **Asynchronous** وبشكل افتراضي تعمل جميع الـ IO Streams بالأسلوب المتزامن لكن العيب فيه هو تأثيره الشديد على أدائية النظام إذ يقوم بإغلاق الـ Processing Unit في الـ Thread المخصصة للبرنامج بحيث لا يسمح بتنفيذ أي أمر آخر إلا بعد الانتهاء من العملية الجارية ولا ينصح ابدأ استخدام الأسلوب المتزامن في حالة إذا كنت تتعامل مع أجهزة قراءة وكتابة بطيئة نسبياً مثل الـ Floppy Disk أو الـ Magnetic Tape لكنها مهمة جداً بالبرمجيات التي تعتمد على أنظمة الزمن الحقيقي أو الـ Real Time Systems حيث أنها تعتمد الأسلوب المتزامن في عملية إرسال واستقبال البيانات وهو ما يمنع القيام بأي عملية أخرى إلى حين الانتهاء من تنفيذ الأمر ومن الأمثلة عليها أنظمة السحب أو الإيداع في الرصيد البنكي أو أنظمة حجز التذاكر أو شحن بطاقة الهاتف وغيرها .. طبعاً 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 تجري من خلال أجهزة بطيئة نسبياً 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 الـ **EndWrite** و الـ **EndRead** ..

### **أولاً: Stream Classes**

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



**1- BufferedStream Class :** ويستخدم بشكل أساسي لحجز مقدار معين من الذاكرة بشكل مؤقت لتنفيذ عملية معينة كما تستخدم بعض البرمجيات الـ Buffering لتحسين الأداء حيث 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 الـ Buffering في العمليات التي يكون فيها المعالج أسرع من عمليات الإدخال و الإخراج حيث يتم معالجة البيانات ووضعها في الـ Buffer في انتظار إرسالها وهو ما يساهم في تحسين الأداء بشكل كبير ، ويستخدم الـ BufferedStream عادةً في برمجيات الشبكات مع الـ NetworkStream لتخزين البيانات المراد إرسالها عبر الشبكة في الذاكرة حيث لا يستخدم هذا الكلاس Backing storage كما ذكرنا سابقاً .. بشكل افتراضي يتم حجز 4096 bytes عند استخدام الـ BufferedStream ويمكن زيادتها أو تقليلها حسب الحاجة.. ويستخدم الـ BufferedStream كما يلي كمثال :

```

C#
using System;
using System.Text;
using System.IO;

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

            MemoryStream ms = new MemoryStream(xx);
            readBufStream(ms);
        }
    }
}
  
```

```

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

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

```

### **VB.NET:**

```

Imports System
Imports System.Text
Imports System.IO
Namespace Network_Buffering
    Class Program
        Shared Sub Main(ByVal args As String())
            Dim asen As ASCIIEncoding = New ASCIIEncoding
            Dim xx As Byte() = asen.GetBytes("Hello Buffering")
            Dim ms As MemoryStream = New MemoryStream(xx)
            readBufStream(ms)
        End Sub

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

```

حيث قمنا بتحويل نص إلى Byte Array باستخدام الـ ASCIIEncoding وتحميله في عبر الـ MemoryStream ثم أرسلناه إلى الدالة readBufStream والتي أنشأناها حيث استقبلنا من خلالها الـ Stream وحملناه في ذاكرة مؤقتة باستخدام الكلاس الـ BufferedStream ثم قمنا بطباعة محتوياته بعد تحويله إلى نص مرة أخرى باستخدام الـ Encoding.ASCII وطباعته..

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

## C#

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

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

## VB.NET:

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

4- **FileStream** : يتم استدعائها باستخدام System.IO Namespaces وتستخدم بشكل أساسي في التعامل مع الملفات سواء للكتابة إلى ملف أو القراءة من ملف ويستخدم أيضا لتحويل الملف إلى Byte Array وحتى يسهل نقله عبر الشبكة ويعتبر هذا الكلاس Backing Storage Class حيث تستخدم ذاكرة Buffer لتخزين 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#

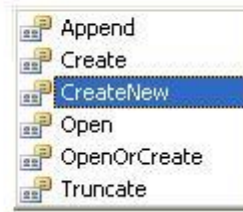
```
FileStream FS = new FileStream(@"C:\MyStream.txt",  
FileMode.CreateNew); // Any Action For Example CreateNew to Create  
Folder
```

## VB.NET:

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

يمكننا استخدام الـ Enumeration التالية مع الـ FileMode :

```
FileStream(@"C:\MyStream.txt", FileMode.);
```



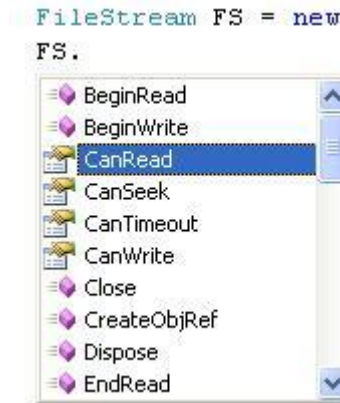
- 1- Append لإضافة نص ما إلى الملف الموجود اصلا
- 2- Create لإنشاء ملف جديد ويقوم بعمل overwriting في حالى إذا كان الملف موجود بشكل مسبق
- 3- CreateNew وهو كما في الـ Create إلا انه يعطي Exception في حالة وجود الملف بشكل مسبق
- 4- Open لقراءة ملف ما حيث يعطي Exception في حالة عدم وجود الملف المحدد
- 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 الـ **FileStream** انظر الفصل السابع عشر **FTP Programming**.



## ثانيا : Stream Members :

هنالك مجموعة من الخواص و الـ Methods (Members) التي تشترك بها مكتبات الـ Stream وهي كما يلي :



- 1- **CanRead** و **CanWrite** وتستخدم لمعرفة إذا كان الـ Stream المستخدم يقبل عملية القراءة أو الكتابة أم لا حيث ترجع قيمة True في حالة إذا كان يقبل و False في حالة أنه لا يقبل ويستخدم عادة قبل إجراء عملية القراءة أو الكتابة لفحص مدى الصلاحية قبل المحاولة ..
- 2- **CanSeek** حيث يستخدم الـ Seeking عادة لتحديد موقع الـ Current Stream وترجع هذه الخاصية قيمة True في حالة كان الـ Stream يدعم الـ Seeking و False في حالة أنه لا يدعم الـ Seeking وفي العادة تدعم الـ Classes التي تستخدم Backing Storage هذه العملية مثل الـ FileStream وعندها ترجع قيمة True .
- 3- **CanTimeout** وترجع قيمة True في حالة إذا كان الـ stream يحتوي على خاصية الـ Timeout والتي تعطي وقت محدد للعملية .
- 4- **Length** وتستخدم لمعرفة حجم الـ Stream بالـ Byte ويمكن الاستفادة منها لمعرفة نهاية الـ Stream أو لتحديد حجم المصفوفة بناء على حجم الـ Stream .
- 5- **Position** وتستخدم الـ Get و Set لمعرفة أو تحديد موقع الـ Stream

وتتشترك مكتبات الـ Stream بمجموعة من الـ Methods وهي كما يلي :

### 1- الدوال المتزامنة Synchronous Methods :

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

### 2- الدوال غير المتزامنة Asynchronous Methods :

- I. **BeginRead** و **BeginWrite** وتستخدم لعملية القراءة أو الكتابة باستخدام الـ Stream الغير المتزامن وتأخذ خمسة باروميترات كما في الشكل التالي :

```
FS.BeginRead(  
[AsyncResult FileStream.BeginRead (byte[] array, int offset, int numBytes, AsyncCallback userCallback, object stateObject)  
array: The buffer to read data into.
```

- 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 الـ Buffer.

- 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 الـ Requests .  
ترجع الـ Begin Methods الـ IAsyncResult والذي يمثل حالة الـ Stream الـ Operation .

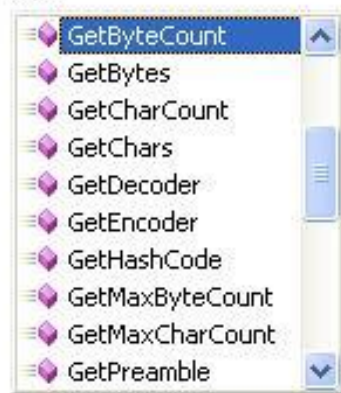
### وهناك بعض الدوال والتي تستخدم لإدارة الـ Stream وهي :

- 1- **Flush** وتستخدم لتفريغ محتويات الـ Buffer بعد إتمام العملية المحددة حيث يتم نقل محتويات الـ Buffer إلى الـ **Destination** الذي تم تحديده في الـ Stream Object.
- 2- **Close** وتستخدم لإغلاق الـ Stream وتحرير الـ Resources المحبوزة من قبل الـ Stream Object وينصح باستخدامها في الجزء الخاص بـ Finally block ولتأكد من أن الـ Stream سيتم إغلاقه وتحرير كافة الموارد في حالة حدوث أي Exception أثناء التنفيذ ولضمان عدم بقاء هذه الموارد في الذاكرة بعد إغلاق البرنامج.
- 3- **SetLength** وتستخدم لتحديد حجم الـ Stream والذي نريد إرساله أو استقباله لكن في حالة إذا كان الـ Stream أقل من المحدد في الـ SetLength سوف يؤدي ذلك إلى انقطاع الـ Stream وعدم وصوله بشكل سليم ، لن تستطيع استخدام هذه الخاصية إلا إذا تأكدت أنك تملك الصلاحية لذلك من خلال الخاصية CanWrite و CanSeek لذا ينصح بفحص الصلاحية أولاً قبل تحديد حجم الـ Stream .

### ثالثاً : Stream Manipulation :

يمكن استخدام مكتبات الـ Stream لنقل Binary Data أو Text وفي العادة يتم استخدام الـ BinaryReader و الـ BinaryWriter لتعامل مع الـ Binary Data ويتم استخدام الـ StreamReader والـ StreamWriter لتعامل مع الـ Text ، ويتم استخدام الـ ASCIIEncoding 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 الـ String إلى Byte Array حتى نستطيع إرسالها باستخدام الـ Stream .
- 3- **GetCharCount** حيث تأخذ Byte Array وترجع عدد الأحرف التي سوف تكون في الـ Character Array .
- 4- **GetChars** وتستخدم لتحويل من Byte Array إلى String وتستخدم عند استقبال البيانات من الـ Stream حيث نحولها إلى نص مرة أخرى .

ولتعامل مع الـ StreamReader و الـ StreamWriter لنقل Text يجب أولاً استدعائها من الـ Namespace 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 الـ BinaryReader والـ BinaryWriter لنقل Binary Data يتم استدعائها من الـ Namespace System.IO ، حيث تستخدم BinaryReader لقراءة Binary Data من الـ Stream والـ BinaryWriter لإرسال BinaryData إلى الـ Stream عبر الـ Socket 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 الـ **Stream** في الدوت نيت وطرق التعامل معها ، والفرق بين الـ **Streams** المتزامن والغير متزامن ، سوف نطبق في الفصل التالي مجموعة من الأمثلة العملية على الـ **Steaming** 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 الـ Binary Data في قواعد البيانات Access و SQL Server Database باستخدام مكتبات الـ Streams.

### 5.1 Remote Control Example باستخدام الـ Stream Reader & Writer:

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

أولا : إنشاء برنامج الإرسال Client : لا يختلف برنامج الإرسال عن برنامج الـ Client Chat الذي قمنا بإنشائه في الـ Chapter1 ويستخدم فيه كل من TCP Connection والـ StreamWriter و الـ NetworkStream لإجراء عملية الإرسال فباستخدام الدالة WriteLine الموجودة ضمن الـ StreamWriter Object تتم عملية تحويل النص المكتوب في الـ Textbox إلى مجموعة من الـ Bytes ليتم إرسالها باستخدام الـ NetworkStream عبر الـ TCP Socket Connection إلى برنامج الـ Server وللبداء 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 الـ 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 الـ Server والذي يعمل على استقبال الـ Stream 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 mytcp;
Socket mysocket;
NetworkStream myns;
StreamReader mysr;

```

**VB.NET:**

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

وإنشاء دالة الـ Server والتي ستتقبل الـ Commands من الـ Clients :

**C#:**

```
void our_Server ()
{
    mytcp = new TcpListener (5020);
    mytcp.Start ();
    mysocket = mytcp.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);

    mytcp.Stop();

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

**VB.NET:**

```
Sub our_Server()
    mytcp = New TcpListener(5020)
    mytcp.Start()
    mysocket = mytcp.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
mytcppl.Stop()
If mysocket.Connected = True Then
    While True
        our_Server()
    End While
End If
End Sub

```

حيث تقوم هذه الدالة بتصنت على الـ Socket في حالة ورود أي Request يقوم بالموافقة عليه وإنشاء Session جديدة معه وفي حالة ورود أي بيانات عبر الـ Socket يتسلمها باستخدام الـ StreamReader 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 Namespaces System.Diagnostics...

ولتشغيلها ضمن Thread جديد لا بد من وضع تعريف الـ Thread في حدث بدأ التشغيل للـ Form 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 الـ Form Closing وذلك لتأكد من إغلاق الـ Socket والـ Stream وإنهاء الـ Thread في البرنامج ..

**C#:**

```

private void Form1_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
{
    try
    {
        mytcppl.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)
```

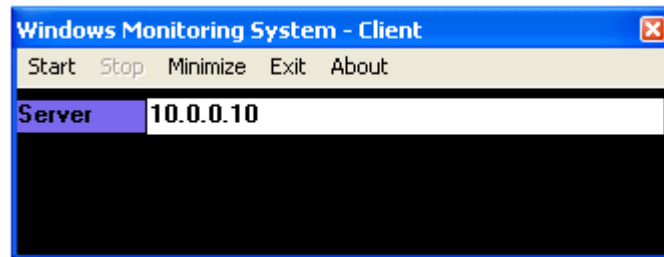
```
    Try  
        mytcp1.Stop()  
        myth.Aport();  
        Catch ex As Exception  
            MsgBox(ex.Message)  
        End Try  
End Sub
```

كما يمكننا بنفس الخطوات السابقة إنشاء برنامج يقوم بإطفاء الـ Clients Computer 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 لاحقاً ، ولالتقاط صورة سطح المكتب نستخدم الدالة CaptureScreen.GetDesktopImage() حيث ترجع صورة سطح المكتب كـ Image ، ولإرسالها لآبد من تحويل هذه الصورة إلى Stream ووضعها في الـ Buffer قبل عملية الإرسال ، وتتم هذه العملية باستخدام الـ MemoryStream Class والذي شرحناه سابقاً 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 لآبد 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)

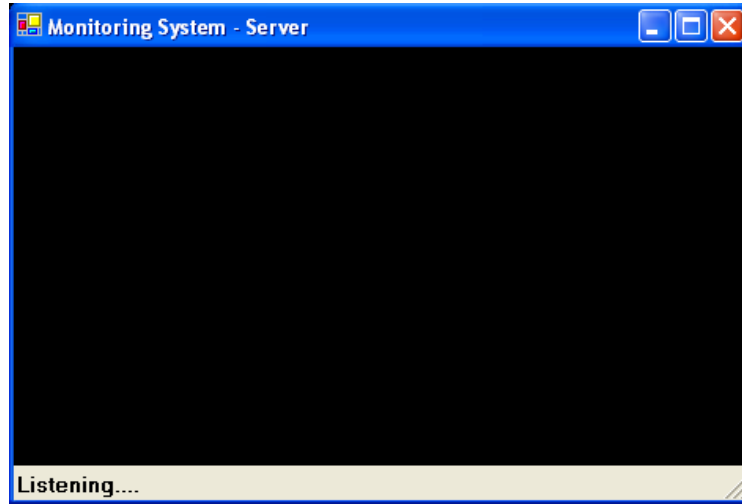
ms.Close()
mysw.Close ()
myns.Close ()
myclient.Close ()

```

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 mytcppl = new TcpListener (5020);
Socket mysocket;
NetworkStream myns;
```

### VB.NET:

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

ثم نقوم بتعريف دالة الاستقبال وكما يلي:

### C#:

```
void ServerMethod ()
{
    try
    {
        mytcppl.Start ();
```

```

mysocket = mytcp1.AcceptSocket ();
myns = new NetworkStream (mysocket);

pictureBox1.Image = Image.FromStream(myns);
mytcp1.Stop();

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

}
catch (Exception){ }
}

```

### **VB.NET:**

```

Private Sub ServerMethod()
Try

    mytcp1.Start()
    mysocket = mytcp1.AcceptSocket()
    myns = New NetworkStream(mysocket)

    pictureBox1.Image = Image.FromStream(myns)
    mytcp1.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 والـ Keyboard للـ RemoteServer وذلك باستخدام دوال الـ API.

### **5.3 مثال لتخزين صورة في قاعدة بيانات Access و SQL Server باستخدام الـ ADO.NET ومكتبات الـ Stream :**

سنستخدم في هذا المثال الـ Memory Stream Class لتمثيل صورة بالذاكرة على شكل Stream Data حيث يمكننا تحويلها إلى Bytes لاحقاً باستخدام الـ 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 الـ OleDbParameter لتمرير الـ Byte Array إلى الـ Database 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 الـ Binary Data والتي سيتم وضعها في الـ Dataset 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 الـ Dataset وتخزينها في Byte Array وبعد ذلك وبتمرير الـ Array إلى الـ MemoryStream Class ، سيتم تحويل الـ Byte Array إلى Stream مرة أخرى وعندها يمكننا عرضها على PictureBox باستخدام الـ FromStream ... Method

### **ثانياً استخدام Microsoft SQL Server :**

لا تختلف عملية التخزين بقاعدة بيانات SQL Server عن الـ Microsoft Access سوى انه يتم استخدام الـ SqlParameter بدلاً من الـ OleDbParameter ، كما أن صيغة الـ Connection String ستختلف بين الـ Access 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)
```

**Transport Layer & Network Layer** سيتم الحديث في الجزء التالي عن استخدام  
**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**

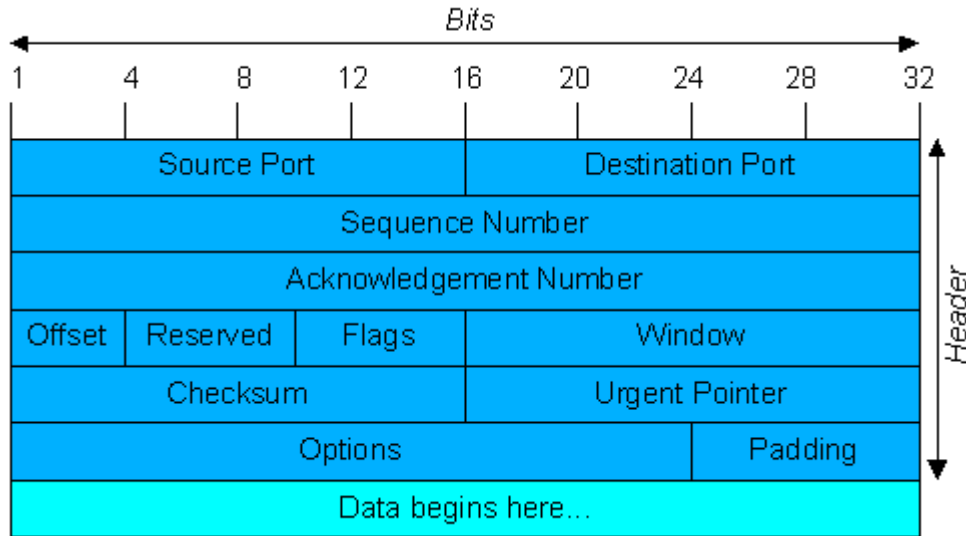
#### **(Classes & Members)**

- TCP Classes Members
- UDP Classes Members

## 6 : TCP & UDP Classes Members :

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

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

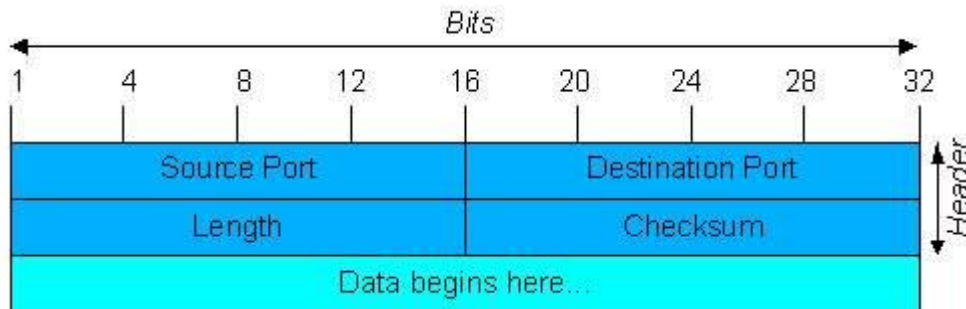


TCP Header

**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.

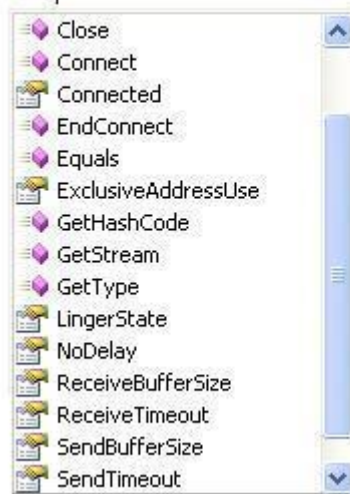
ثانيا الـ UDP Header ويتكون من 32 Bits من البيانات للـ Packet الواحد ويحتوي على عنوان المرسل 16 Bits والمستلم 16 Bits و الـ Checksum والـ Total Length:



UDP Header

## أولا الكلاسات الخاصة بال TCP Connection Oriented Protocol :

```
TcpClient tcp = new TcpClient();  
tcp.|
```



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

### **أولا: TCPClient Methods :**

**Connect**: وتستخدم لأجراء عملية الاتصال مع الـ server حيث نمرر فيها عنوان الـ IP الخاص بالـ Server ورقم الـ Port 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 الـ TCP Socket.  
**GetStream**: ويستخدم لقراءة الـ Stream من الـ Socket في عملية الإرسال و الاستقبال وتستخدم عادة لربط الـ Socket مع الـ NetworkStream Class.

### **ثانيا: أهم الخصائص TCPClient Properties :**

**LingerState**: وتأخذ get أو Set لتحديد أو معرفة الـ Linger Time  
**NoDelay**: وتأخذ get أو Set لتحديد أو معرفة إذا كان هناك وقت معين لتأخير أم لا  
**ExclusiveAddressUse**: وتأخذ get أو Set لتحديد أو معرفة إذا كان الـ Socket يسمح باستخدام الـ Client Port أم لا ، وتستخدم عادة مع الـ SocketOption Class (لمزيد من المعلومات أنظر الـ Appendixes (A))  
**ReceiveBufferSize** و **SendBufferSize**: وتأخذ get أو Set لتحديد أو معرفة حجم الـ Buffer المستخدم في الـ stream والمعرف في الـ TCP Client Object.

**SendTimeout و ReceiveTimeout:** وتأخذ get أو Set لتحديد أو معرفة الوقت المتاح لعملية الإرسال أو الإستقبال حيث يعطي Time Out في حالة أنه لم يجد الطرف الآخر خلال فترة زمنية معينة.

**TcpListener Class-2:** حيث تحتوي على مجموعة من الـ Methods والـ Properties وهي كما يلي:

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



**أولاً: أهم الدوال في الـ TcpListener:**  
**AcceptSocket:** وتستخدم لقبول عملية الاتصال مع الـ Client.  
**Start:** وهي Overloaded Method حيث انه في حالة تمرير رقم إليها يتم تحديد عدد الأجهزة التي تسمح بوجودها في الـ Queue ، وفي حالة عدم تمرير رقم معين تصبح الـ Queue غير محدد.  
**Stop:** وتستخدم لإغلاق عملية التصنت ويفضل وضعها في الـ Finally عند استخدام الـ Try و الـ Catch وحتى يتم إنهاء عملية التصنت حتى لو حدث أي Exception.

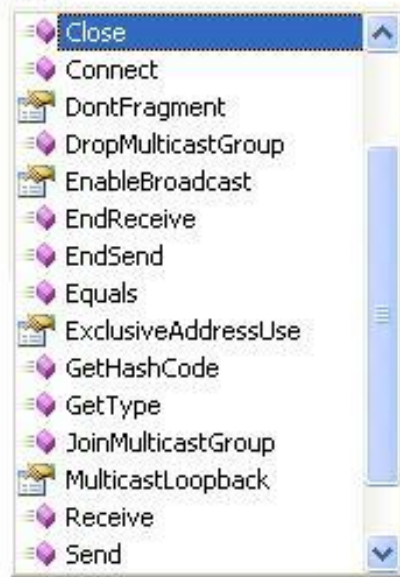
**ثانياً: أهم الخصائص في TcpListener:**  
**LocalEndpoint:** حيث يرجع الـ IP ورقم الـ Port المستخدم في الـ LocalEndpoint المحدد.  
**Server:** ومن خلالها نستطيع الوصول إلى كل الخصائص و الدوال في الـ TCP Server والتي شرحناها سابقاً مثل الـ Accept والـ Sendto والـ Receive و Listen وغيرها



## ثانياً الدوال الخاصة بالـ UDP Connectionless Protocol Classes :

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

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



ومن أهم الدوال والخصائص الخاصة بها :

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

**MulticastOption Class-2:** ويستخدم في Multicasting حيث يتم تخزين IP Address List لتعامل معها في Multicast Group لعمل Join و Drop لأي Multicast Group وتستخدم كما يلي كمثال لإضافة عضوية لاستقبال رسائل Multicast :  
أولاً نعرف الـ UDP Socket وكما يلي :

### C#:

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

### VB.NET:

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

ثانياً نقوم بتعريف Address List ثم نسند إليها الـ IP الذي نريد ضمه في الـ Group أو نجعل الـ User يدخل العنوان بنفسه بعد ذلك نربطها بالـ Socket 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)
```

ثالثاً نقوم بتعريف الـ Multicast Option ونسند لها العنوان المحدد كما يلي:

**C#:**

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

**VB.NET:**

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

ولتفعيلها يجب تعريفها في الـ `SocketOptions Method` و تأخذ هذه الدالة ثلاثة بارامترات 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 التغيير 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 عضوية AddMembership أو إلغاء عضوية DropMembership وأخيراً نسند إليه الـ MulticastOption Object والذي قمنا بإنشائه و كما يلي:

**C#:**

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

**VB.NET:**

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

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

---

## **Chapter 7**

### **Network Layer & Synchronous Socket Programming**

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

## **: Network Layer & Synchronous Socket Programming :7**

في هذا الجزء سوف نبين بشكل أكثر تفصيلا عن برمجة طبقة الـ Network Layer Socket وهي التي يتم التعامل معها لإرسال واستقبال البيانات بعد تحويلها من و إلى Stream عبر الشبكة، قمنا سابقا باستخدام الـ TCP والـ UDP للإرسال وللاستقبال وبيننا الفرق بينهما وفي هذا الجزء سوف نتحدث عن الـ Socket Programming والـ Classes والـ Members الخاصة بالـ Socket Class..

### **Introduction to Socket Programming :7.1**

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

#### **C#:**

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

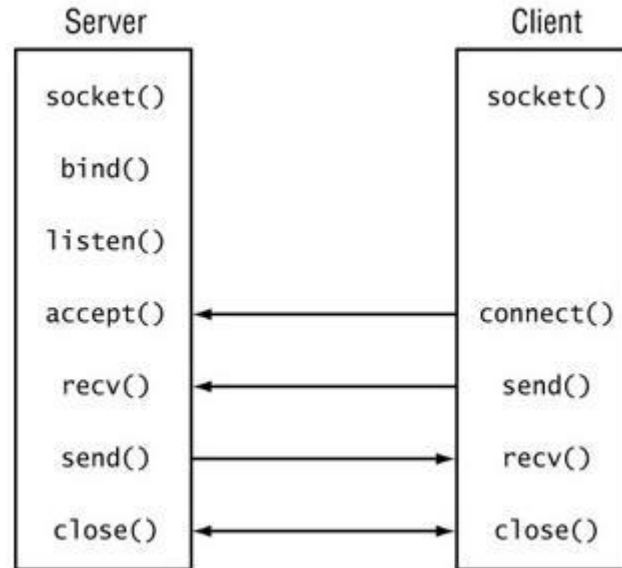
#### **VB.NET:**

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

حيث يتم في الباروميتر الأول تحديد نوعية الـ IP Address والذي سوف تتعامل معه ويعطيك عدد كبير من الخيارات ومنها IPX والمستخدم في شبكات الـ Novel أو الـ ATM والمستخدم في شبكات الـ ATM Networks أو الـ NetBIOS Address وغيرها ... ومن أهم هذه الخيارات الـ InterNetwork 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 الإستهناء عنه وليحل محله الـ IPv6 (لمزيد من المعلومات انظر الفصل الثاني)، في الباروميتر الثاني يتم تحديد نوع الـ Socket أي هل سوف نستخدم الـ Stream أو الـ Dgram لإرسال البيانات أو الـ 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 الـ Stream في حالة إستخدام الـ TCP والـ Dgram في حالة استخدام الـ UDP ، وأخيرا نحدد نوع البروتوكول المستخدم للإتصال هل هو من النوع UDP أو TCP أو بروتوكولات أخرى مثل الـ ICMP Internet Control Message Protocol أو الـ IGMP Internet Group Management Protocol أو اننا نريد مثلا إنشاء الـ Socket لتعريف الـ IP Security Header بإختيار الـ IPSecAuthenticationHeader وغيرها، وهنا سوف نختار الـ TCP أو UDP ومن المعروف أن بروتوكول الـ TCP هو بروتوكول موجه وهذا يعني إجراء عملية التحقق من الوصول والتوصيل إلى شخص ما محدد أما بروتوكول الـ UDP فهو بروتوكول سريع نسبيا ولكنه لا يدعم عملية التحقق من الوصول السليم للبيانات المرسله وهو مفيد جدا لإجراء عملية البث الإذاعي Broadcast وإنشاء مجموعات البث Multicast Group وهو ما شرحناه في الفصل الأول والثاني والثالث .

## 7.2 استخدام الـ Synchronous Socket Programming لإنشاء TCP Connection:

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



إذ تبدأ العملية في الـ Client و الـ server بإنشاء الـ Socket 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 الـ Socket 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 الـ Port المحدد باستخدام الدالة listener ويمكنك تحديد 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 الـ Port (5020 كمثال) حيث يتم ربطها بالـ Socket باستخدام الدالة Bind وقمنا بتعريف Listen لا نهائي العدد -1..

ولتعريف برنامج الإرسال TCP Client باستخدام الـ Socket لابد من تعريف الـ Socket وإسناد عنوان الـ Server ورقم الـ Port بالـ Object Instance الـ IPEndPoint ثم إرسال البيانات باستخدام الدالة Send وتتم عملية الإرسال بما تم تعريفه في الـ socket حيث سنستخدم الـ 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();
```

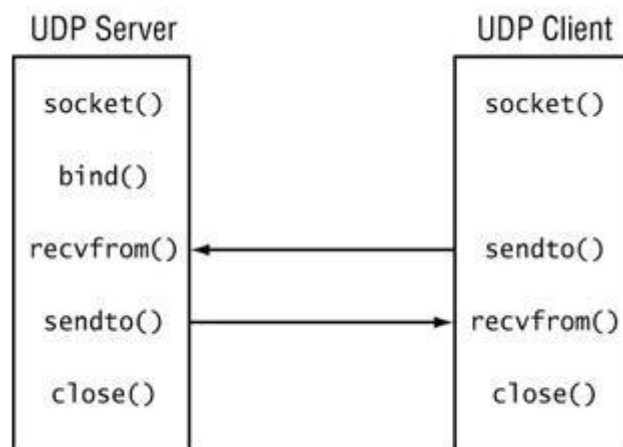
**VB.NET:**

```
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 الـ Synchronous Socket Programming لإنشاء UDP : Connectionless

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



وتشبه عملية استخدام الـ UDP الـ TCP إلا حد كبير إذ تبدأ العملية في الـ Client و الـ Server بإنشاء الـ Socket 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 الـ Socket 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 الـ EndPoint Reference بناء على ما تم تعريفه في السابق ونمرره كـ Reference Object مع مصفوفة

الدالة Byte 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 المرسله إذ يمكننا معرفته من خلال الخاصية Length ونوع الـ Flags وأخيرا نمرر له الـ EndPointObject والذي يعرف الـ IP Address والـ Port الـ Socket 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)
```

## **:Synchronous Socket Classes Members7.4**

**Class -1 IPAddress :** ويستخدم لتعريف الـ IPAddress حيث يمكن إسناده إلى الـ IPEndPoint 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 (حيث يقوم بعملية حساب الـ IP Broadcast دون تدخل المستخدم ، لمعرفة كيفية حساب الـ Broadcast 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 بالـ Network من العنوان

**NetworkToHostOrder** ويرجع الجزء الخاص بالـ Host من العنوان

**IPEndPoint Class** : حيث استخدمناه لتحديد العنوان والـ Port 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 الـ Socket Class وهي كما يلي:

**AddressFamily** ويرجع مجموعة العناوين المعرفة على الـ Socket

**Available** ويرجع حجم البيانات الجاهزة للقراءة من الـ Socket

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

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

**ProtocolType** ويستخدم لمعرفة البروتوكول الذي يستخدم في الـ Socket

**RemoteEndPoint** ويرجع معلومات عن الـ Socket الذي يستخدم مع الـ Remote Host

**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

## : Asynchronous Socket Programming :8

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 استخدام في حالة إذا كانت الجهة المستقبل للبيانات تستخدم Slow Connection كاعتماد الشبكة على الـ Dialup لربط الجهازين المرسل مع المستقبل أو في حالة إذا كان هنالك مجموعة كبيرة من المستخدمين تستخدم الـ Server حيث يمنع الأسلوب المتزامن بقية المستخدمين على الشبكة من إجراء عملية الإرسال في حالة كون الـ Server يستقبل بيانات من جهاز آخر ، وفي هذه الحالة ينصح باستخدام الاتصال الغير المتزامن إذ يعتبر مهم جدا في حالة إذا أردنا من البرنامج القيام بعدة مهام وعلى نفس الـ Thread وباستخدام نفس الـ Connection ، أو كما ذكرنا سابقا في حالة إذا كان الاتصال بطيء نسبيا أو انه يوجد عدد مستخدمين يستخدمون نفس الـ Server ..

### : Asynchronous Socket Class and its members

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

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



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

**C#:**

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

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

**VB.NET:**

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

حيث سيتم إضافة الـ Client Request في Callback Reference منفصل عن السابق وهنا لابد من إنشاء method لاستقبال الـ Client 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 الـ Begin Accept Method تأخذ عدة أشكال كما يلي:

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 يمكن فيه تحديد الـ Accepted Socket:

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

**-2 BeginConnect** وتستخدم لربط الـ Client مع الـ Server أو الـ Remote Machine كـ Asynchronous Connection حيث يسند لها الـ IPEndPoint Instance Object (يحدد فيه الـ IP Address ورقم الـ Port) ويسند لها أيضا الـ Asynchronous Callback والـ State Object وكما يلي:

```
MySocket.BeginConnect(EndPoint IP, Synccallback 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 الـ Connect\_him Method يتم تحديد الـ Socket Callback 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 الـ Client 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 الـ UDP حيث يضاف إلى التركيب السابق **IPEndPoint** Refrance Object .

**BeginSend -4** 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 الـ UDP حيث يضاف إلى التركيب السابق IPEndPoint Refranch Object .

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

التركيب التالي:

تأخذ الـ **:BeginSendFile**

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

والبـ **BeginReceiveMessageFrom** التركيب التالي:

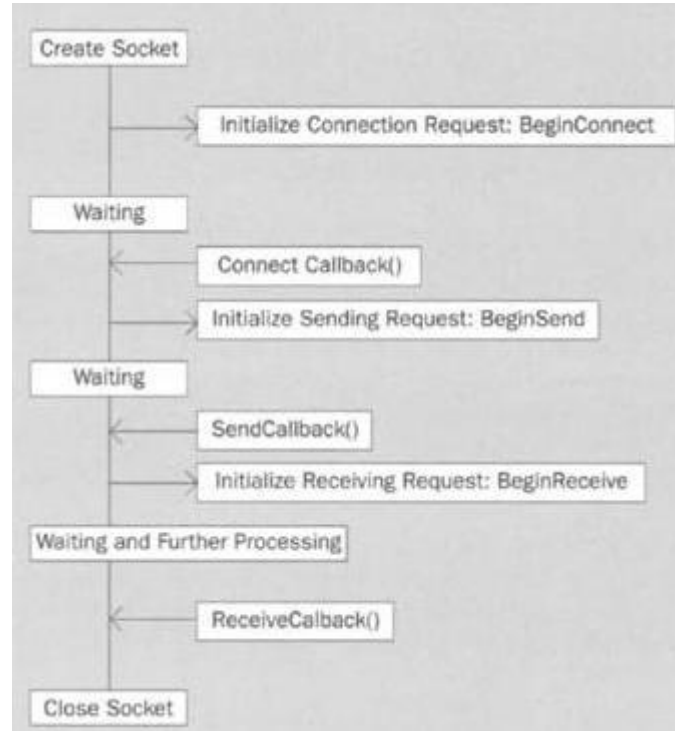
```
MySocket.BeginReceiveMessageFrom(byte Buffer ,int offset,int
size,SocketFlags sf,ref EndPoint,AsyncCallback ascb,object state)
```

والبـ **BegonDisconnect** التركيب التالي:

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

## ثانيا: تطبيقات Asynchronous Socket في الدوت نيت :

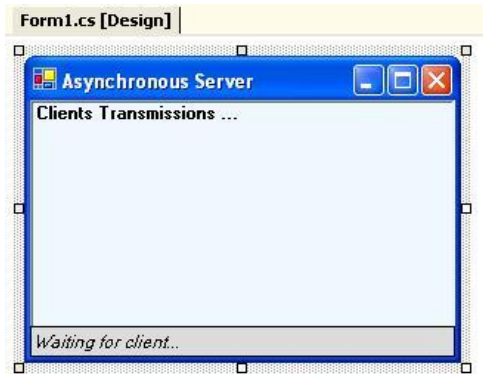
تمر عملية الاتصال الغير متزامن بمجموعة من المراحل تبدأ بإنشاء الـ Socket Object في الـ Server Side بعد ذلك يتم تعريف الـ BeginConnect لبدأ الـ Asynchronous Connection على الـ Socket حيث يتم إسناد الـ IPEndPoint Object والـ Asynchronous Callback الـ State Object لها وتبدأ في هذه الحالة عملية الاتصال بالـ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



نلاحظ في الشكل السابق بقاء الـ Server بوضعية الإنتظار إذ أنه وفي كل عملية طلب للـ Request ما يتم إنهائه باستخدام الـ Endxxx Method وتستدعى هذه الـ Method باستخدام الـ Delegate المخصص لإجراء عملية الإتصال الغير متزامن وهي الـ 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 الـ Asynchronous Socket لإرسال واستقبال البيانات .

وللبدء قم بإنشاء مشروع جديد كما في الشكل التالي:



سوف نستخدم الNamespaces 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 الGlobal Declaration (أي بعد تعريف الMain 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 الـ **Form Load** قم بإضافة الكود التالي حيث سنعرف **Connection** يعتمد على الـ **TCP** ويعمل على الـ **Port 5020** ثم تعريف عملية قبول الاتصال باستخدام الـ **BeginAccept**:

**C#:**

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

**VB.NET:**

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

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

**C#:**

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

**VB.NET:**

```
Sub AcceptConn(ByVal iar As IAsyncResult)
    Dim oldserver As Socket = CType(iar.AsyncState, Socket)
    Dim client As Socket = oldserver.EndAccept(iar)
    conStatus.Text = "Connected to: " + client.RemoteEndPoint.ToString()
    Dim stringData As String = "Welcome to my server"
```

```

Dim message1 As Byte() = Encoding.ASCII.GetBytes(stringData)
client.BeginSend(message1, 0, message1.Length, SocketFlags.None,
New AsyncCallback(SendData), client)
End Sub

```

ثم إنشاء **Send Callback method** لإنهاء الـ **BeginSend** وكما يلي:

**C#:**

```

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

```

**VB.NET:**

```

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

```

ثم إنشاء **Receive Callback method** لإنهاء الـ **BeginReceive** وكما يلي:

**C#:**

```

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

```

**VB.NET:**

```

Sub ReceiveData(ByVal iar As IAsyncResult)
    Dim client As Socket = CType(iar.AsyncState, Socket)
    Dim recv As Integer = client.EndReceive(iar)

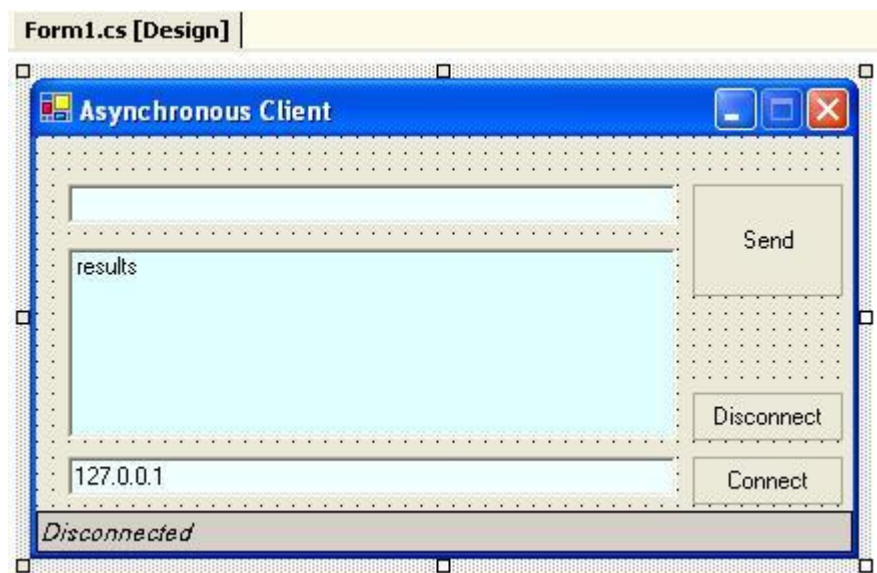
```

```

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

```

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



سوف نستخدم الـ Namespaces 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 الـ Global Declaration (أي بعد تعريف الـ Main Class) قم بإضافة التعاريف التالية:

**C#:**

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

في الـ **Connect 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 الـ **BeginReceive** وكما يلي:

#### **C#:**

```

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

```

#### **VB.NET:**

```

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

```

ثم إضافة الكود التالي في الـ **Send Button** :

#### **C#:**

```

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

```

### **VB.NET:**

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

ثم إنشاء **Send Callback method** لإنهاء الـ **BeginSend** وكما يلي:

### **C#:**

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

### **VB.NET:**

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

ثم إنشاء **Receive Callback method** لإنهاء الـ **BeginReceive** وكما يلي:

### **C#:**

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

```

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

```

### **VB.NET:**

```

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

```

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

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

## : Advanced Multicasting Systems :9

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

### : Architecture of Multicast Socket

من المعروف انه يتم التعامل مع الـ Multicasting عبر بروتوكول الـ UDP وباستخدام الـ Class D Subnet Mask وتتم عملية إدارة المجموعات باستخدام بروتوكول الـ IGMP – Internet Group Management Protocol والذي هو جزء من الـ Internet Protocol Model وكما يتضح من الشكل التالي فإن بروتوكول الـ IGMP يحتوي على عمليات التحقق من الوصول السليم للبيانات (حيث يتم إرسال حجم البيانات الكلي لرسالة وهي اختيارية إذ يمكن إلغاؤها بوضع الرقم صفر) ، و تحتوي أيضا على الـ TTL 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 الـ Range المحدد للـ Class D .

8-bit Type	8-bit Max Response Time	16-bit Checksum
32-bit Group Address		

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

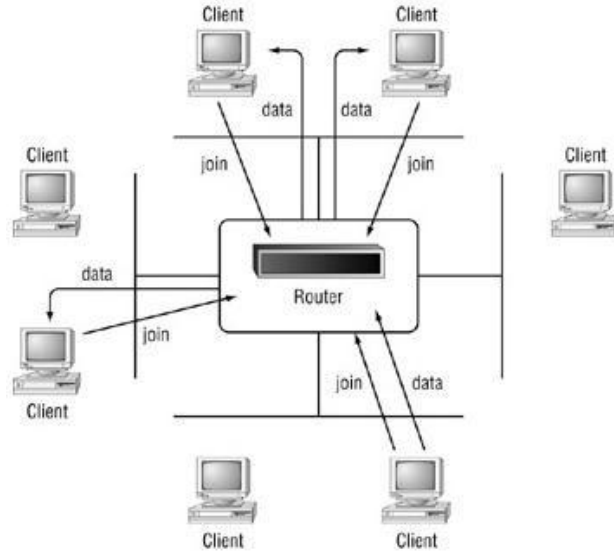
كمثال التخصيص 224.0.0.1 يستخدم في جميع الشبكات المحلية حيث لا يتم تمريره إلى شبكة أخرى عبر الـ Router. لمعرفة جميع التخصيصات للـ Multicasting انظر الرابط التالي الخاص بمنظمة الـ Iana:

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

يتم نقل الـ Multicast Packets بين الـ Backbone Tunnels كـ Unicast حيث يتم إرسالها من داخل الشبكة إلى الـ Router وترسل من Router إلى آخر عبر الـ Backbone Tunnel بأسلوب الـ Unicast وهو ما يوفر الكثير من الـ Bandwidth في الشبكة حيث ترسل نسخة واحدة إلى الـ Router ويقوم هو بتوزيعها على الأجهزة كـ Broadcast ، المشكلة الوحيدة في الـ Multicast هو انه يعتمد بشكل كامل على استخدام الـ UDP Connectionless Protocol.

ويمكننا استخدام الـ Multicasting في ثلاثة أنواع من الشبكات وهي شبكات الـ Peer to Peer حيث لا وجود لجهاز Server والكل يستقبل ويرسل من وإلى الـ Group الذي هو فيه،

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

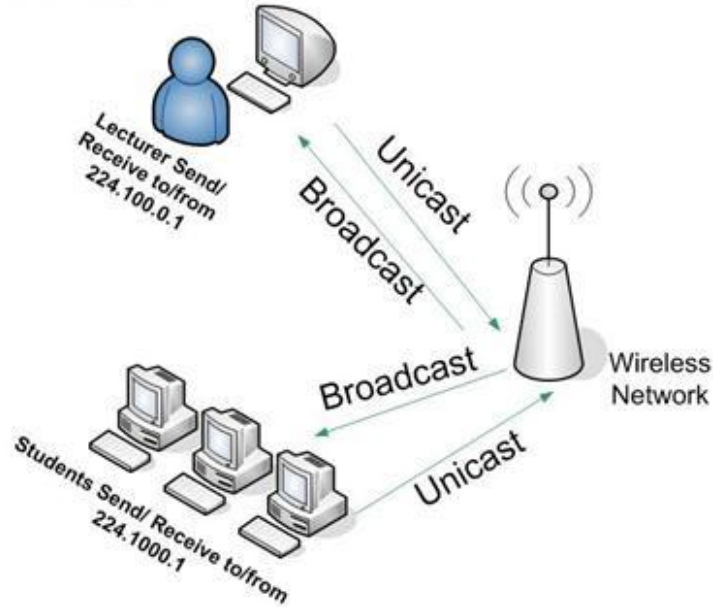


وكما كان الحال في الإرسال باستخدام الـ Broadcasting يتم الإرسال في Multicasting من جهاز محدد إلى مجموعة معينة وليس إلى الكل كما في الـ Broadcast ، حيث تكون كل مجموعة من الأجهزة Group خاص ويتم التخصيص كما ذكرنا سابقاً وفق الـ IP Multicasting حيث تمتلك كل مجموعة نفس الـ IP Multicast ويوجد عدة أشكال للـ Multicasting 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 الـ Sender User نفس الـ IP Multicasting الذي يملكه الـ Receiver Users ويتم الإرسال من داخل الـ Group إلى جميع أعضائه حيث ترسل كـ Unicast إلى الـ Access Point حيث يقوم بتوزيعها على كافة الأعضاء في المجموعة بأسلوب الـ Broadcast وكما في الشكل التالي:

By FADI Abdel-Qader

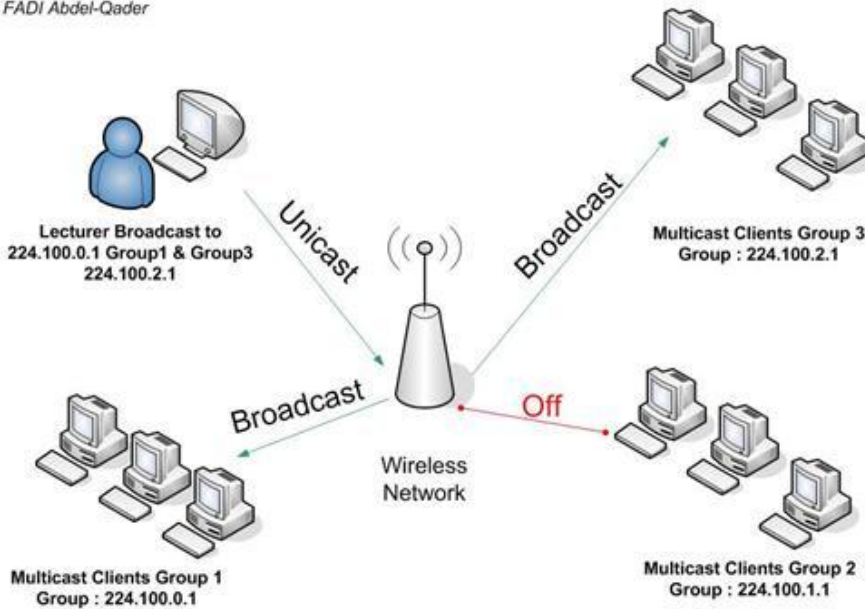


**Full Duplex Multicast System for one Group**

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

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

By FADI Abdel-Qader



**Half Duplex Multicast System for Multi-Groups**

### ثانيا :Using Multicast Socket with .NET:

شرحنا سابقا كيفية التعامل مع Multicasting 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 Multicasting .

من العمليات الأساسية في التعامل مع Multicasting :

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

لا تلزم عملية الانضمام إلى الـMulticast Group أي عمليات تحقق سوى التصنت على الـport والـIP Multicasting المحدد ، ويتم ذلك بعد تعريف الـudpClient Object وباستخدام الـJoinMulticastGroup Method يتم تعريف الـIP Multicasting الذي سوف ننضم إليه وكما يلي:

#### C#:

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

#### VB.NET:

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

وكما يلي لإلغاء عملية الانضمام من مجموعة:

#### C#:

```
sock.DropMulticastGroup(IPAddress.Parse("225.100.0.1"));
```

#### VB.NET:

```
sock.DropMulticastGroup(IPAddress.Parse("225.100.0.1"))
```

إذ تستخدم الـ **JoinMulticastGroup** و **DropMulticastGroup** Methods لضم أو إلغاء عنوان أو مجموعة من العناوين من الـMulticast Group ، وباستخدام الـMulticastOption Class يمكننا تخزين الـIP Address List لتعامل معها في الـMulticast Group لعمل الـJoin 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)
```

ثانياً نقوم بتعريف Address List ثم نسند إليها الـ IP الذي نريد إدخاله في الـ Group أو نجعل الـ User 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)
```

ثالثاً نقوم بتعريف الـ Multicast Option ونسند لها العنوان المحدد كما يلي:

**C#:**

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

**VB.NET:**

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

ومن ثم نضيف التغير على SetSocketOption حيث تأخذ هذه الدالة ثلاثة باروميترات الأول لتحديد مستوى التغير 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 التغير حيث نريد إضافة عضوية ويمكن الاختيار بين إضافة عضوية AddMembership أو إلغاء عضوية DropMembership وأخيراً نسند إليه الـ MulticastOption Object والذي قمنا بإنشائه و كما يلي:

**C#:**

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

**VB.NET:**

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

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

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

**C#:**

```

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

```

**VB.NET:**

```

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

```

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

**C#:**

```

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

```

**VB.NET:**

```

Dim ms As MemoryStream = New MemoryStream
PictureBox1.Image.Save(ms,
System.Drawing.Imaging.ImageFormat.Jpeg)
Dim arrImage As Byte() = ms.GetBuffer
ms.Close

```

```
Dim server As Socket = New Socket(AddressFamily.InterNetwork,
SocketType.Dgram, ProtocolType.Udp)
Dim iep As IPEndPoint = New
IPEndPoint(IPAddress.Parse("225.100.0.1"), 5020)
server.SendTo(arrImage, iep)
```

### 3- الاستقبال من مجموعة Receive Data From a Multicast Group

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

#### C#:

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

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

#### VB.NET:

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

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

#### C#:

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

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

### **VB.NET:**

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

### **ملاحظات هامة في استخدام Multicasting في برمجيات الشبكات :**

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

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

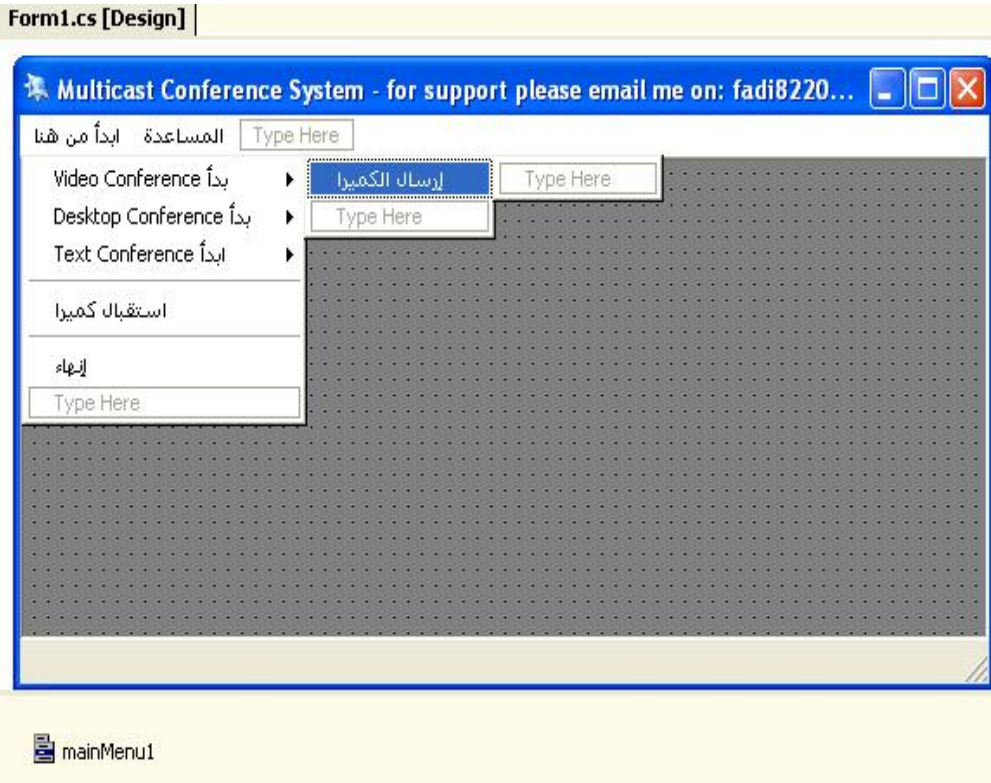
3- يمكن لكل جهاز أن ينضم إلى أكثر من مجموعة بحيث يستقبل من جهات متعددة، كذلك يستطيع الإرسال إلى عدة مجموعات.

4- في العادة تكون السعة المسموحة لإرسال الـ Multicasting Data عبر الـ sendto Method محدودة والسبب الأساسي لهذه المشكلة هي استخدام الـ Multicasting بروتوكول الـ UDP والذي لا يمكن تجزئة الـ Data المرسل في حالة استخدام حيث لا يدعم التوصيل وفق الترتيب مما يحصر إمكانيات الإرسال بالحجم الأقصى للـ Ethernet Encapsulation.

### 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 والـ Voice Chatting...

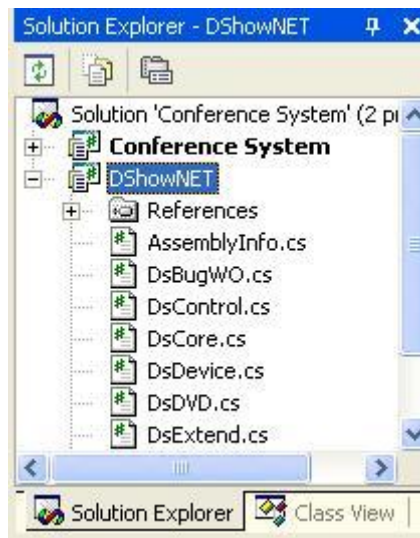
سوف نقوم هنا بتقسيم نظام المؤتمرات إلى ثلاثة أنظمة رئيسية وهي نظام مؤتمرات الفيديو ونظام مؤتمرات سطح المكتب ونظام المؤتمرات النصية وسنضيف عليه لاحقا نظام المحادثة الصوتية، في البداية سوف نقوم بإنشاء الشاشة الرئيسية للبرنامج و كما في الشكل التالي:



## **: Full/Half Duplex Multicast Video Conferencing System -1**

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

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



وحتى نتعامل معها سوف نستدعيها باستخدام :

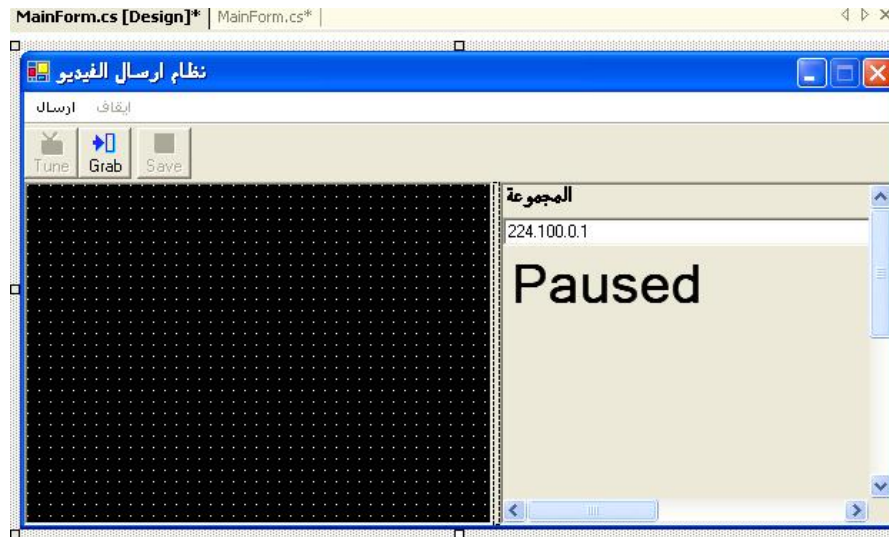
### **C#:**

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

### **VB.NET:**

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

وسيكون شكل برنامج إرسال الكاميرا كما في الشكل التالي:



سوف نستخدم الـ DeviceSelector 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.IsNothing 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.IsNothing 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 الآخر سوف نستدعي دالة إرسال الصورة والتي قمنا بإنشائها من خلال استخدام Timer ولهدف استمرار عملية الإرسال كل فترة معينة، وكما يلي:

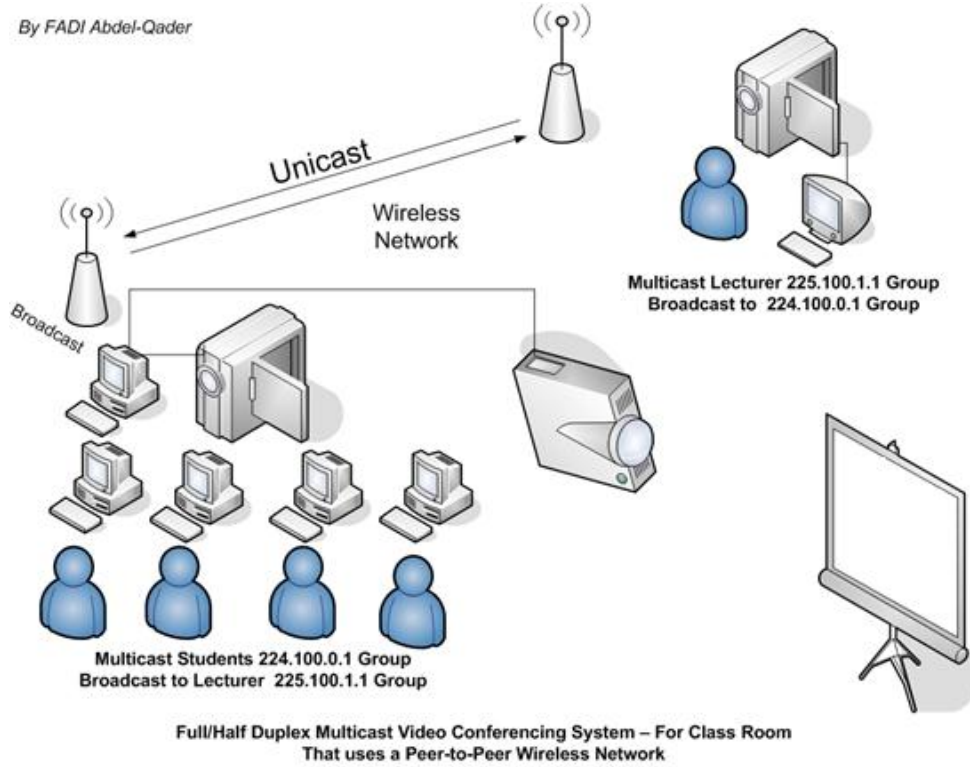
**C#:**

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

**VB.NET:**

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

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



وكما نلاحظ في الشكل السابق فإن المحاضر ينضم إلى مجموعتين مجموعة الأساتذة وهي 225.100.1.1 حيث سيستقبل صورة طلابه عليها، ومجموعة الطلاب 224.100.0.1 والتي سوف يرسل الصورة إليها .. وكما نلاحظ أيضا فإن عملية الإرسال بين الـ Access Point1 والـ Access Point2 تتم باستخدام الـ Unicast ...

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



سوف نستخدم الـ Namespaces التالية لاستقبال الصورة من الـ Multicast 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 الـ Threading ، وحتى نقوم بذلك قم بعمل Timer وضع فيه الكود التالي لاستخدام الـ Threading :

**C#:**

```

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

```

**VB.NET:**

```

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

```

وحتى تتمكن من تخزين الصورة الملتقطة عبر الكاميرا على هيئة JPEG Image File 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

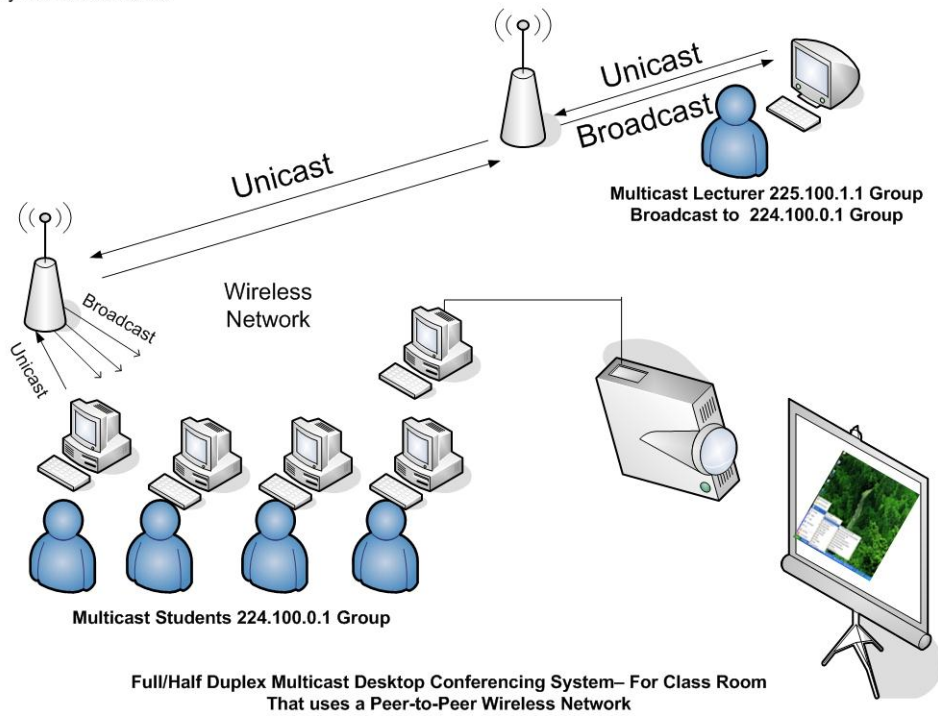
```

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

## :Full/Half Duplex Multicast Desktop Conferencing System -2

الهدف من هذا المشروع هو تمكين الأستاذ من عرض المحاضرة باستخدام برنامج 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 الـ Multicast Group ومن ثم استقبالها وعرضها على الطلاب باستخدام Data Show Projector .  
ويوضح الشكل التالي مخطط عمل المشروع:

By FADI Abdel-Qader



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 محاضراته 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 الـ `user32.dll` والـ `BitBlt 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 الـ Height 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 الـ Namespaces التالية في البرنامج لتعامل مع الـ Multicasting 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 Multicast Group المحدد:

#### **C#:**

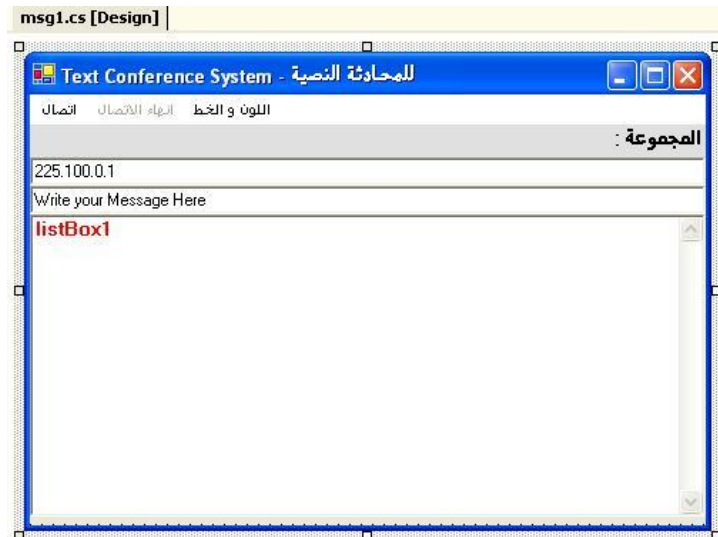
```
Bitmap bt = new Bitmap(CaptureScreen.GetDesktopBitmapBytes ());
picScreen.Image = ResizeBitmap(bt, 600, 400 );
MemoryStream ms = new MemoryStream();
picScreen.Image.Save(ms,System.Drawing.Imaging.ImageFormat.Jpeg)
;
byte[] arrImage = ms.GetBuffer();
ms.Close();
Socket server = new Socket(AddressFamily.InterNetwork,
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 Text Chat Multicast Conference System سوف نقوم بإنشاء New Form جديد وكما في الشكل التالي:



ثم قم بإضافة Namespaces 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 الـ method التالية لإجراء عملية الإرسال حيث سترسل الرسالة عند الضغط على الـ Enter بعد كتابة الرسالة في الـ Textbox المخصص :

#### C#:

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

```

server.Close();
msg.Clear();
msg.Focus();
}
catch(Exception){ }
}
}

```

### **VB.NET:**

```

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

```

وسوف نستخدم الدالة التالية لعملية الاستقبال حيث ستعرض الرسالة المستقبلية 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 الـ Threading ، قم بعمل Timer واستدعي فيه الـ method السابقة باستخدام الـ ThreadStart 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 الـ Timer عند الضغط على زر الاتصال باستخدام timer1.Enabled = true وفي زر إنهاء الاتصال قم بإضافة الكود التالي:

### **C#:**

```
timer1.Enabled = false;
txt_host.ReadOnly = false;
msg.Enabled=false;
    try
    {
        Socket server = new Socket(AddressFamily.InterNetwork,
        SocketType.Dgram, ProtocolType.Udp);
        IPEndPoint iep = new IPEndPoint(IPAddress.Parse(txt_host.Text),
        5020);
```

```

byte[] data = Encoding.ASCII.GetBytes("has Left the Room");

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

```

### **VB.NET:**

```

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

```

تعرفنا في هذا الفصل على كيفية عمل Multicast Conference Systems في بيئة الدوت نيت وطرق إنشاء الـ Video Conference System والـ Desktop Conference System والـ Text Conference System ، و سنتمكن 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 و Voice Conference System باستخدام الـ API's والـ Direct Play 9 بالإضافة إلى استخدام الـ TAPI 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

## **: Voice Over IP Programming - 10**

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 Stream Data عبر الشبكة ، وللاستقبال الصوت في الطرف الآخر يتم تجميع الـ Packets مرة أخرى في مصفوفة Byte Array ، وتتم عملية القراءة وفق مبدأ الـ FIFO – First In First Out أي القادم أولاً يعرض أولاً ...

تكمن المشكلة الأساسية بنقل الصوت في مدى توفر الشروط اللازمة حتى يتم إيصال وعرض الصوت بالشكل السليم و وفق الترتيب 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 أداة خارجية باستخدام الـ API أو Managed باستخدام الـ DirectX، ومن المعروف أن نظام التشغيل Windows XP يدعم الاتصال باستخدام بروتوكول TCP أو UDP فقط وهذا يعني أنك إذا كنت تعمل تحت منصة نظام التشغيل Windows XP فإن أي عملية اتصال لن تكون إلا باستخدام واحد من هذه البروتوكولات.

## **: The Requirements of Voice Communication Systems : أولاً**

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 الـ TCP والـ SCTP أما الـ UDP فقط في حالة استخدامه مع الـ RTP
- 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 اعتماد أساليب ضغط أخرى مثل G.711 لمعرفة كيفية استخدامه بالدوت نيت أنظر الرابط التالي

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

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

أولاً 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 أنه من المعروف أن الـ Multicasting 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 الـ Acknowledgment في الـ TCP Header ، إذ يعتبر الـ TCP بروتوكول موجه Oriented Protocol لذلك لا يمكن الاعتماد عليه في حالة حاجتنا لعمل Multicast Conference System أو في حالة البث الإذاعي Broadcasting .

ثانياً 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 الـ Fragmentation للـ Packets المرسل ومن المعروف أن حجم الـ 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> IETF Internet ومنظمة RTP – Real Time Transport Engineering Task Force بتقديمها بروتوكول النقل 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 الإمكانات 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 الـ TCP تعتمد على مدى الحاجة التي نريدها ودقة الصوت من جهة أخرى إذ أن أفضل طريقة لنقل الصوت هي استخدام تقنيات الـ Stream لكن من المعروف أن بروتوكول الـ UDP 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 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 آخر كـ Stream لابد لنا من استخدام بروتوكول الـ TCP لكن كما ذكرنا سابقا فإنه لا يدعم الـ IP Multicasting و الـ Broadcasting

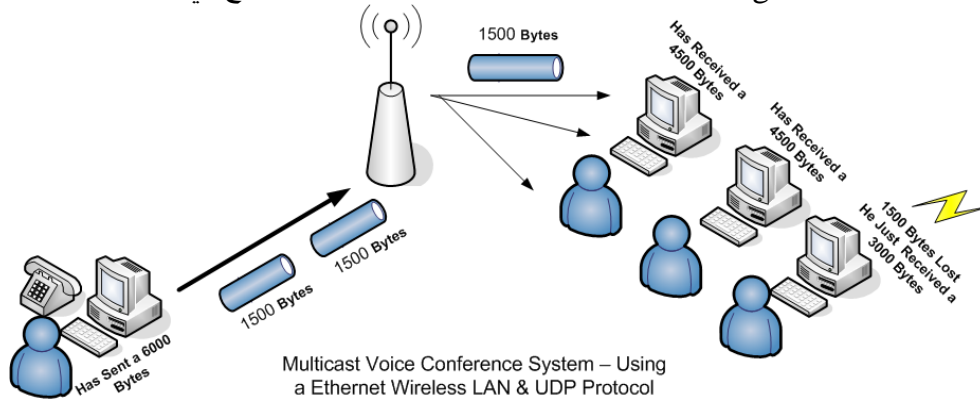
3- في الـ Network Layer يتم عنوانة الـ Packets وإذا ما قررنا اعتماد الـ UDP فإننا سوف نتمكن من عمل البث الإذاعي Broadcasting ومجموعات البث Multicasting

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 أثناء عملية النقل كثيرة جدا وقد يحدث تأخير Delay لسبب أو لآخر أو قد تضيق بعض الـ Bits أثناء الإرسال لذلك لابد من وجود بروتوكولات تدعم التصحيح لكل هذه المشاكل والتي قد تحدث أثناء عملية الإرسال. يستقبل الطرف المقابل الـ Bits من طبقة الـ Physical Layer وتمر عبر الـ Data Link Layer ومن ثم الـ Network Layer وفي أثناء 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 Packets 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 أثناء هذه المرحلة منها ضياع جزء من الـ Packets المرسل والذي سوف يسبب وصول الصوت بشكل متقطع ، وطبعاً 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 أثناء الإرسال باستخدام شبكة Ethernet Wireless LAN و UDP IP Multicasting مما سوف يسبب تقطيع في الصوت:



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

كما بينا سابقاً فإن الدوت نيت لم تدعم أي من عمليات التقاط وعرض الصوت ، لكن لإجراء هذه العمليات لابد من استخدام الـ API 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 الـ Method هي :

**waveInGetNumDevs** والتي تستخدم لتحديد عدد أجهزة الإدخال والمربوطة مع الـ Sound Card ولا تأخذ أي باروميترات.

**waveInAddBuffer** وتستخدم لتخزين الـ Bits الواردة من جهاز الإدخال في Byte Array Buffer وتأخذ هذه الـ Method ثلاثة باروميترات وهي:

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

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

الدالة **waveInOpen** و **waveInClose** لفتح وإغلاق الاتصال مع جهاز الإدخال. الدالة **waveInPrepareHeader** لتجهيز وحجز الـ Buffer وتأخذ نفس الباروميترات الموجودة في **waveInAddBuffer**.

الدالة **waveInUnprepareHeader** ويتم استدعاؤها بعد تعبئة الـ Buffer حتى يتم إرسال الـ Buffer ومن ثم تفرغه للاستعداد لتعبئته مرة أخرى.

الدالة **waveInReset** لإرجاع مؤشر الـ Pointer الخاص بالـ Buffer إلى صفر. الدالة **waveInStart** و **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 الـ Methods 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 بالـ Socket وكما يلي:

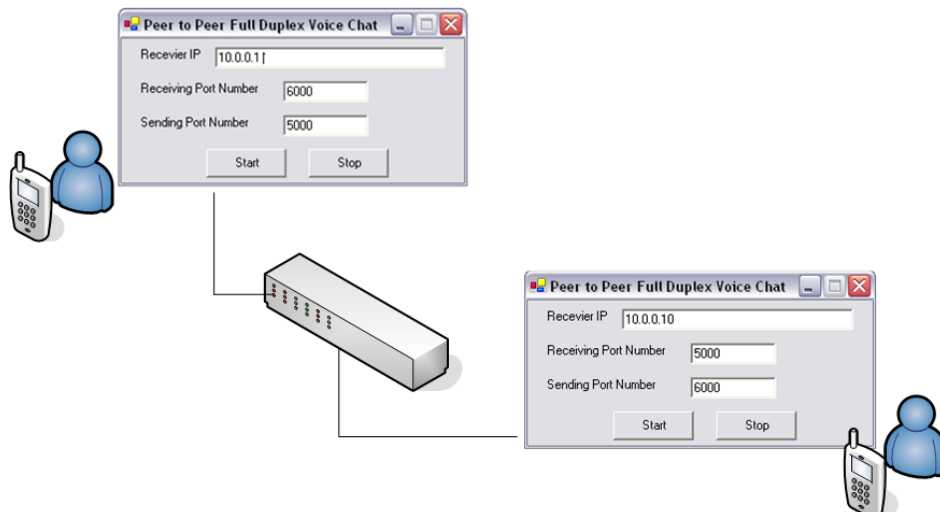
```
C#:
void Voice_Receiver()
{
    UdpClient sock = new UdpClient(5020);
    sock.JoinMulticastGroup(IPAddress.Parse(multicast_IP.Text));
    IPEndPoint iep = new IPEndPoint(IPAddress.Any, 0);
    byte[] voice_Come = sock.Receive(ref iep);
    Winmm.PlayWavResource(voice_Come);
    sock.Close();
}
```

```
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
```

### البداية بإنشاء برنامج المحادثة الصوتية Voice Chat System :

سوف نجري عملية التقاط الصوت وتخزينه في الـ Buffer ثم عرضه مرة أخرى في مجموعة من الـ Classes 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 Class وسوف نستخدمه لعرض الصوت الآتي 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 أولاً خارج أولاً  
الدالة WaveNative ويتم فيها وضع كافة التعريفات للـ Methods الخاصة بالملف winmm.dll والتي شرحناها سابقاً.  
سوف نستخدم في هذا المثال بروتوكول الـ UDP لعملية النقل ومعتمداً على أسلوب البث Full Duplex Unicast Voice Chat System وسيكون الشكل العام لبرنامج الاتصال كما يلي :



وسوف نقوم بكبسلة الـ Classes السابقة في ملف Voice.dll وسوف نضعه في الـ References الخاصة بالبرنامج وحتى نستطيع استخدام هذا الملف في جميع البرامج التي سوف تستخدم عملية الاتصال الصوتي بعد هذه العملية يمكن استخدام الملف باستدعائه كأبي 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 الـ Socket والـ Thread 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 والـ Thread 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 الـ Classes السابقة ونعرف الـ Buffer الذي سيتم تسجيل الصوت المراد إرساله والـ Buffer الذي سيتم عرض الصوت المستلم من الـ Socket

#### **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 الـ Voice\_In Method الكود الخاص بعملية استقبال الصوت من الـ Socket 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 الـ Receive Method ثم نمرر الصوت المستقبل إلى الـ m\_Fifo.Write Method وحتى يتم تنفيذه وتحويله إلى صوت مرة أخرى.

أما الـ Method التي تقوم بتسجيل الصوت وإرساله إلى الجهاز الآخر 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 Two Ports واحد للإرسال وأخرى للاستقبال وفي الطرف الآخر تكون Port الإرسال لديك هي Port الاستقبال لديه والعكس صحيح .

في زر البدء يتم تنفيذ الدالة التالية حيث استخدمنا الـ WaveFormat لتحديد جودة الصوت بـ 2 Channels & 44100 KHz 16 Bits , وأسندت إلى الـ WaveOutPlayer Class والذي سيستخدم لتعبئة الـ Buffer 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**

```

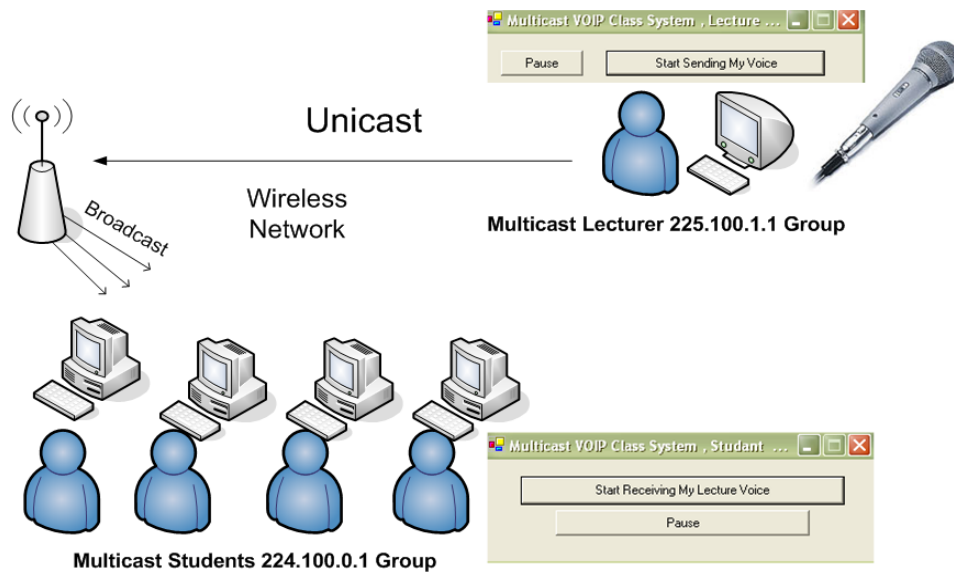
Private Sub Filler(ByVal data As IntPtr, ByVal size As Integer)
If m_PlayBuffer Is Nothing OrElse m_PlayBuffer.Length < size Then
m_PlayBuffer = New Byte(size - 1)
End If
If m_Fifo.Length >= size Then
m_Fifo.Read(m_PlayBuffer, 0, size)
Else
Dim i As Integer = 0
Do While i < m_PlayBuffer.Length
m_PlayBuffer(i)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

```

## رابعاً: 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 بالـ Multicasting 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 IP 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 والـ Multicasting فإن عملية الإرسال ستكون عشوائية وهذا قد يسبب ضياع واحد أو أكثر من الـ Packets المرسل 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 المرسله 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 الـ Network 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**

```

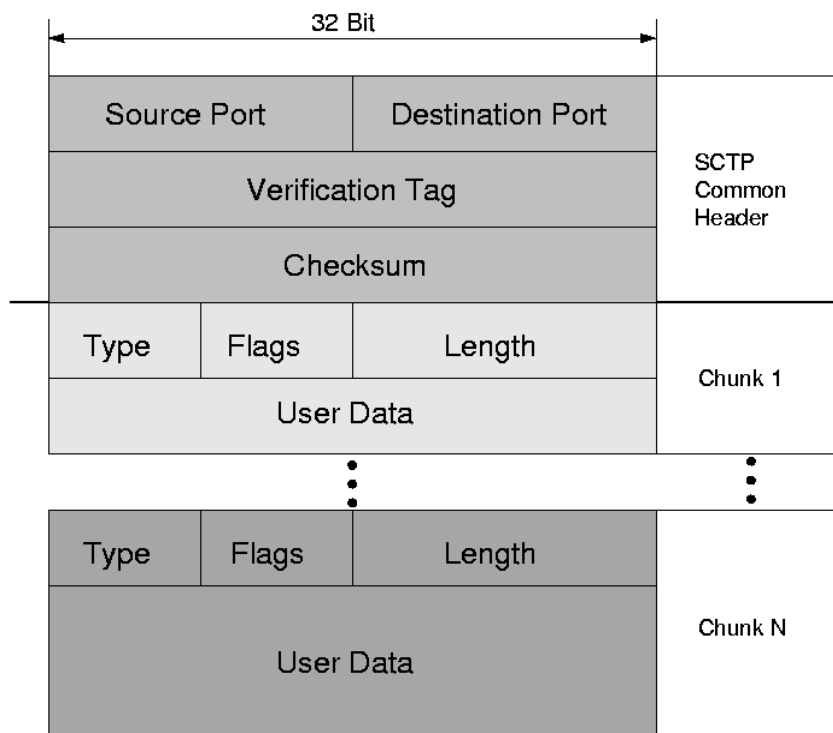
Private Sub Voice_In()
    Dim sock As Socket = New Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp)
    sock.Bind(New IPEndPoint(IPAddress.Parse("10.0.0.10"), 5020))
    Dim ns As NetworkStream = New NetworkStream(sock)
    Do While True
        Dim buffer As Byte() = New Byte(16383)
        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 Multicasting أو Broadcasting...

والحل الوحيد هو إما استخدام المعايير السابقة مع بروتوكول الـ UDP وهو ما تم استخدامه الآن في معظم البرمجيات يمكنك الرجوع إلى القرص المدمج المثال OpenH323 لمعرفة كيفية استخدام الـ UDP والـ RTP لنقل الصوت وفق المعيار H.323. أو استخدام بروتوكول الـ TCP وهو جيد في حالة كان عامل التزامن غير مهم بين المرسل والمستقبل وكمثال على ذلك برنامج Windows Media Server والذي يستخدم الـ TCP مع الـ HTTP 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 الـ TCP والدعم المقدم من قبل الـ UDP لعمليات الـ Multicasting إذ أصبح لدينا الآن منصة قوية يعتمد عليها في عمل الـ Fragmentation وإعادة ترتيبها بكل سهولة بالإضافة إلى دعمه عملية Delivered on Sequence وهذا واضح من بنية الـ Header 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 بـ Direct Play9 ويسمى أيضا DirectPlay Transport Protocol وهو جزء من مجموعة الـ DirectX 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 الـ TCP/IP أو الـ IPX الخاص بـ Novel 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 كـ Stream Data ،

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

#### أولا : الـ Connect Classes والخاصة بعملية الربط:

الـ Address ، Peer Classes ، Guido حيث تستخدم عند إنشاء الاتصال مع الطرف الآخر ، وتستخدم الـ DirectPlay طريقة لتمييز البرنامج عن الآخر بتوليد Hash Code خاص بكل برنامج ويتم ذلك باستخدام الـ Guido 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
```

طبعاً 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 الآخر في الشبكة باستخدام الـ FindHosts Method والموجودة ضمن الـ Peer Class ، وتتم عملية الربط مباشرة باستخدام الـ Connect 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)

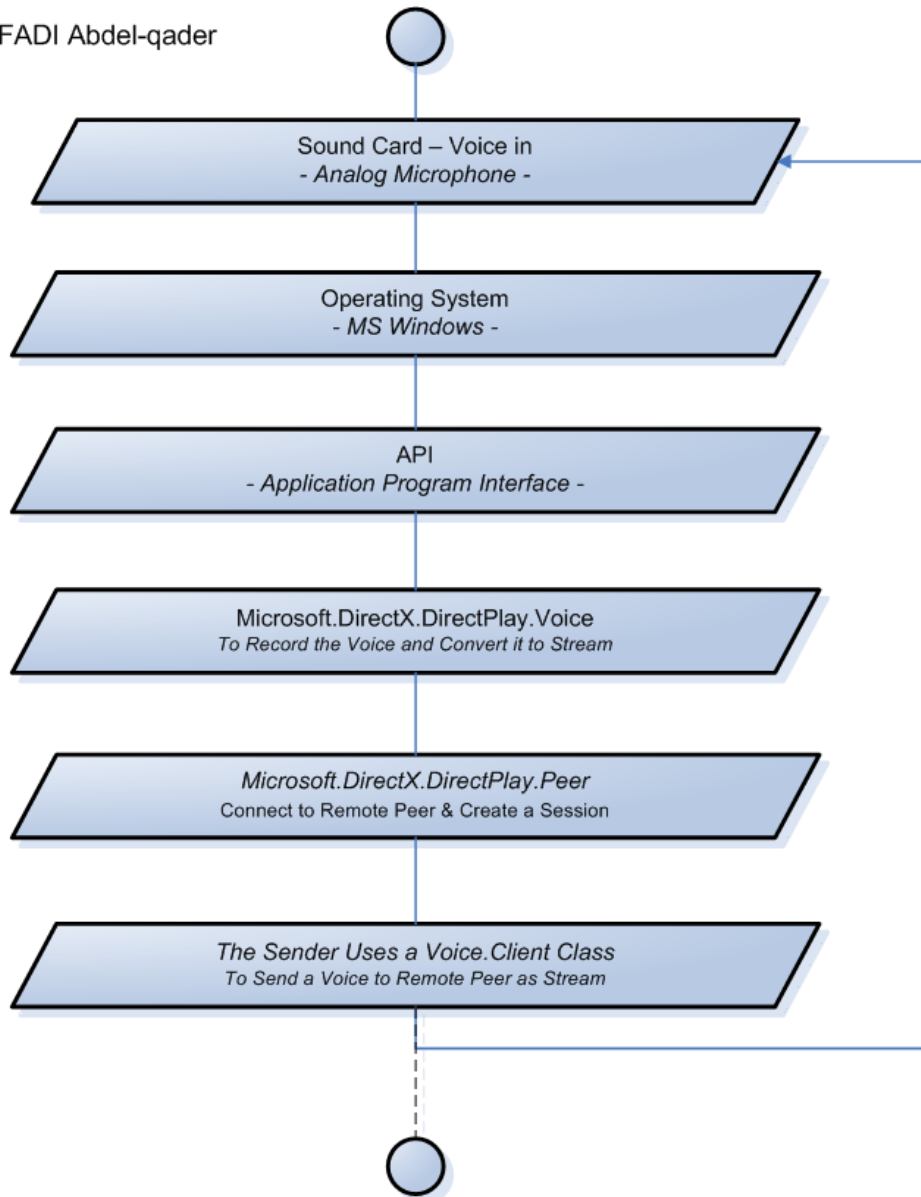
' Call connect passing the Host information returned in the
FindHostResponse event

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

```

ثانياً: Microsoft.DirectX.DirectPlay.Voice والخاصة بكل عمليات تسجيل ونقل وعرض الصوت:  
تمر عملية تسجيل ونقل وعرض الصوت بمجموعة من المراحل ونلخصها بالشكل التالي:

© FADI Abdel-qader



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:

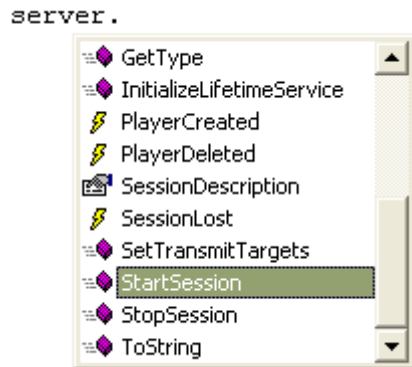
**C#:**

```
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 الـ Methods الخاصة بعملية بدأ وإنهاء الجلسة بالإضافة إلى مجموعة من العمليات الأخرى:



ولبدأ الجلسة يجب أولاً إسناد خصائص الجلسة إلى الـ StartSession Method 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 dx)
    {
        throw dx;
    }
}
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 dx 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);

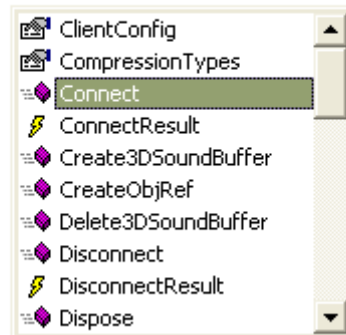
**VB.NET**  
 CreateVoiceSession(host, Voice.SessionType.Peer,  
 mConfigForm.CompressionGuid)

2- الـ Classes الخاصة بطرف الـ Client للاتصال مع الـ Sessions التي أنشئها الـ Server:  
 الـ Client Class ومن أهم الـ Methods الموجودة في الـ Client Class :

```

//Connect to voice session
client.Connect(soundConfig,

```



الدالة Connect وتستخدم لربط مع الـ Voice Session حيث يرسل الـ Server رقم الـ Session 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 والـ UDP 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 وأهم ميزات وعيوب بروتوكولات الـ Transport Layer ومدى إمكانياتها لنقل الصوت عبر الشبكة وأخيرا شرح لأهم الـ Classes والمستخدمات 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 .

**ثالثاً: تصميم برنامج للمحادثة الصوتية والمرئية وطرق التعامل مع الـ 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 جداً:**

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

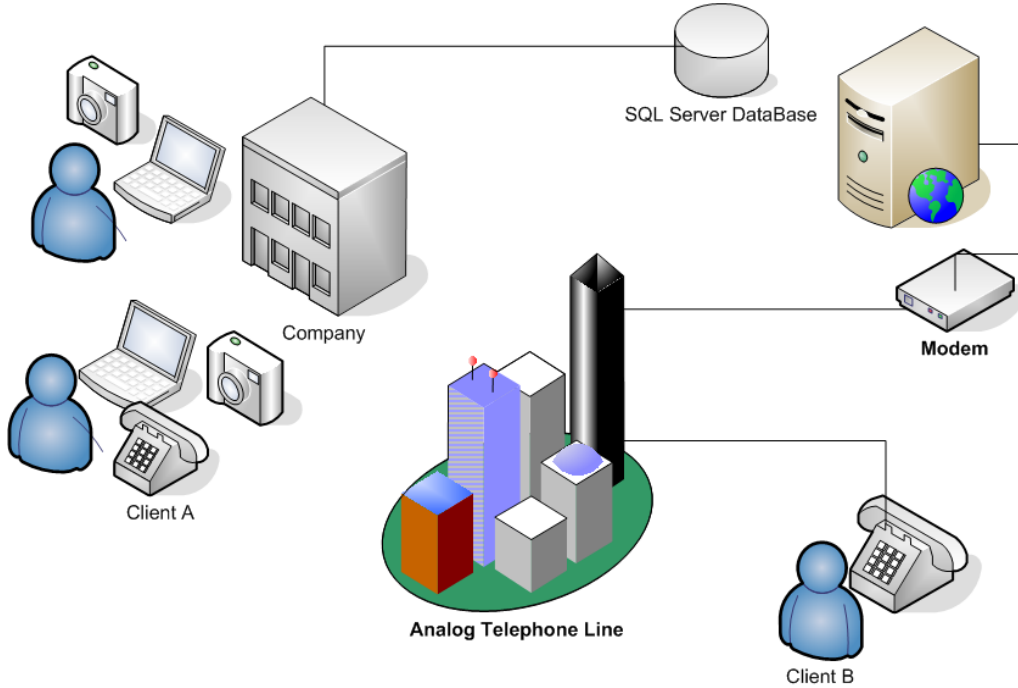
**Streaming in Dot Net -1**

**TCP,UDP Socket Programming -2**

**Advanced Multicasting Systems -3**

**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 الـ Multimedia في بيئة نظام التشغيل واستخدامنا 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 الـ TAPI 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 الـ TAPI Telephony للاتصال مع الـ Analog Phone وبالعكس... وللبدا بتطبيق المشروع سنقوم بتصميم قاعدة بيانات تحتوي على أسماء الموظفين والـ Computer 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

employees_info				
	Column Name	Data Type	Length	Allow Nulls
🔑	Name_Department	varchar	50	
	Telephone_Number	varchar	50	
	Computer_Name	varchar	50	
	Group_ID	varchar	50	
▶	Other_Info	varchar	150	✓

وجداول 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 الـ Client التي سيستخدمها الموظفون 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 لجميع 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) ويتم بإرسال تنبيه إلى Client لإخباره بوجود مكالمة صوتية منتظرة وفي حال الموافقة يحول الصوت إلى الـ Computer 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 الـ Methods 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
<code>tapiRequestMakeCall(string stNumber, string stDummy1, string stDummy2, string stDummy3)</code>	وتستخدم لإجراء عملية الاتصال باستخدام الـ Modem حيث يمرر لها رقم الهاتف الذي نريد إجراء اتصال معه
<code>phoneShutdown (hPhoneApp app) [DllImport("tapi32.dll")] static extern long phoneShutdown(long hPhoneApp);</code>	وتستخدم لإغلاق برنامج الـ TAPI Telephony
<code>LONGWINAPIphoneSetVolume( HPHONE hPhone, DWORD dwHookSwitchDev, DWORD dwVolume)</code>	وتستخدم هذه الدالة لتحديد درجة الصوت ويمرر لها الدرجة بالـ Hex حيث تعبر القيمة <code>0x00000000</code> على عدم وجود صوت أما القيمة <code>0x0000FFFF</code> 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_ReceivingPORT.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_ReceivingPORT.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
        mytcp1 = new TcpListener (int.Parse(text_Camera_rec_port.Text));
        mytcp1.Start (); // Start Listening on That Port
        mysocket = mytcp1.AcceptSocket ();
        ns = new NetworkStream (mysocket);
        pictureBox2.Image = Image.FromStream(ns);
        mytcp1.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
        mytcppl = New
        TcpListener(Integer.Parse(text_Camera_rec_port.Text))
        mytcppl.Start()
        mysocket = mytcppl.AcceptSocket()
        ns = New NetworkStream(mysocket)
        pictureBox2.Image = Image.FromStream(ns)
        mytcppl.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 الآخر 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 الـ Clients الإرسال على Port معين إلى الـ Server وكذلك قيام الـ Server بإرسال الـ Messages إلى كل الـ Clients 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 الـ ARP 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 الـ Message بعد إرسالها إلى الـ Destination ، يأخذ 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 المرسلية والـ Code الخاص بهذه الرسالة وتقسّم 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 11-1 أنواع الـ ICMP Messages

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

Bytes 32 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 الـ MAC Address مشابه للموجود ضمن الـ MAC 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:

16		32 bits
Hardware Type		Protocol Type
HLen (8)	Plen (8)	Operation
Sender Hardware Address		
Sender Protocol Address		
Target Hardware Address		
Target Protocol Address		

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

### **ثانيا : Raw Socket Programming :**

لتعامل مع الـ Raw Socket في الدوت نيت يجب أولاً تعريف الـ Raw Socket Type بعد ذلك يمكننا اختيار أي من البروتوكولات التي تعمل على الـ Network Layer وتدعم الـ Raw Socket التعامل مع البروتوكولات التالية وكما هو واضح في الجدول 15-2 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 15-2 – و يبين البروتوكولات التي تدعمها الـ Raw Socket

وكمثال لبرمجة بروتوكول الـ ICMP يجب أولاً 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 أيضاً لإجراء عملية الـ Connect أو الـ Binding مع الـ Socket لذلك نحدد له فقط في الـ IP Endpoint 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 الـ ICMP Header Class يجب التقيد بالشكل العام لهذا البروتوكول والـ Data Type لكل Field وكما هو واضح في الجدول 15-3 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

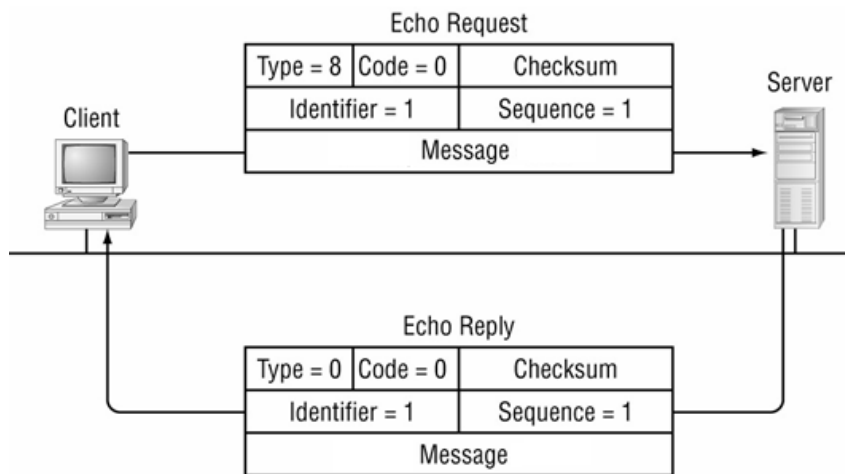
Public Checksum As UInt16

Public MessageSize As Integer

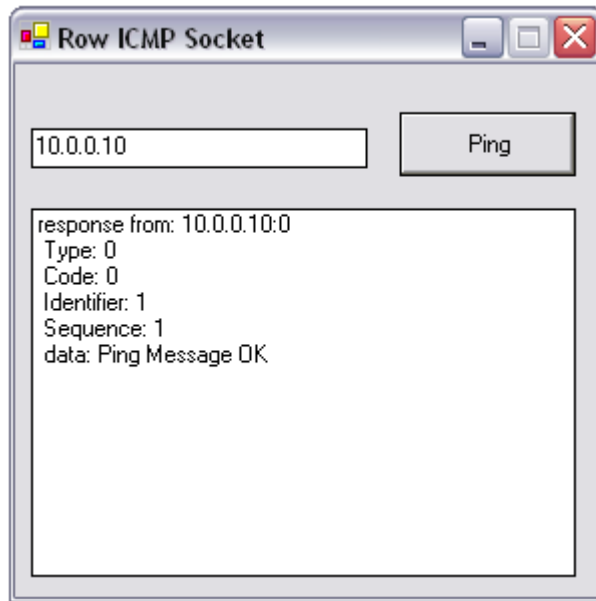
Public Message As Byte() = New Byte(1024)

End Class

سنقوم الآن بتنفيذ برنامج بسيط لإجراء عملية الـ Ping ومن المعروف أن الـ Ping يستخدم الـ Type 8 لإرسال والـ Type 0 عند الاستقبال حيث يقوم الطرف المقابل باستقبال الرسالة وإعادة إرسالها كما هي إلى الـ SourceAddress ، وإذا تم استقبال الـ ICMP Message بالـ Type 0 هذا يعني أنه تم الوصول إلى الـ Destination ويمكن التحقق من وصول الـ ICMP Header بالشكل الصحيح بعد حساب ناتج الـ Checksum فإذا تطابق مع مقلوب مجموع الـ Header Data عندها نستطيع معرفة أنه تم تسليم الـ Message لـ Destination بشكل الصحيح وكما هو واضح في الشكل التالي:



ولبدء بالتطبيق سيكون الشكل العام للبرنامج كما في الشكل التالي:



ولإنشاء الـ ICMP Class يجب أولاً تعريف الـ ICMP Header وتحويله إلى Byte Array بعد إسناد الـ Ping 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
    End Sub

```

```

        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 الـ `BitConverter.GetBytes Method` وأنشئنا الـ `getChecksum Method` لحساب الـ `Checksum` للـ `ICMP Packet` حيث سيسند إليه وسيستخدم لعملية التحقق من وصول الـ `Packet Header` للـ `Destination` بالشكل الصحيح .

ويمكن استخدام الـ `Class` السابق مباشرة بعد تعريف `Socket Raw` ثم تعريف `Object IPendpoint` لإسناده إلى `sendto Method` ثم اشتقاق `Object instance` من الـ `ICMP 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 تغيير مدة الـ `Time Out` للمدة التي سينتظرها الـ `Source` إلى حين الدخول على الـ `SocketException` في حالة عدم الاستجابة ويتم كل ذلك كما يلي:

```

C#:
byte[] data = new byte[1024];
int rcv;
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.ReceiveTimeout, 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.ReceiveTimeout, 1000)
host.SendTo(packet.getBytes(), packet.size, SocketFlags.None, iep)
```

ولاستقبال الرد على الرسالة من الـ Destination سنستخدم نفس التعريف السابق للـ Socket حيث سنستخدم الـ `ReceiveFrom` Method ونسند إليها الـ `Reference Value` للـ `IPEndpoint` 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)

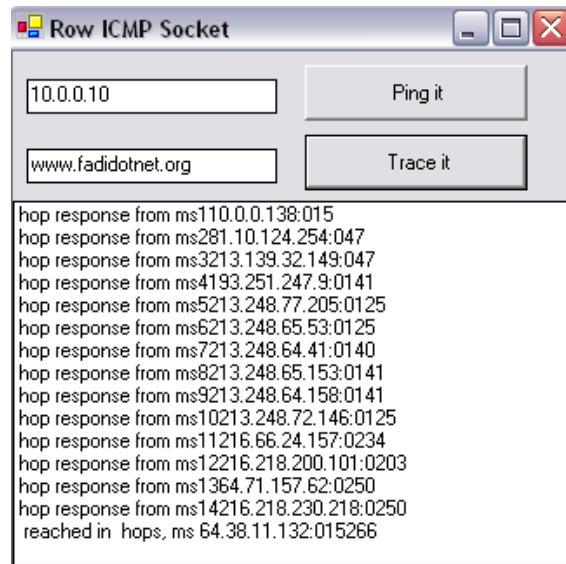
listBox1.Items.Clear()
listBox1.Items.Add("response from: " & ep.ToString())
listBox1.Items.Add(" Type: " & response.Type)
listBox1.Items.Add(" Code: " & response.Code)

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 بالـ Tracing Routing انه يقوم بتعقب الـ Hops أو الـ Routers التي يمر عبرها الـ Packet حيث يرجع كل Router يمر الـ Packet من خلاله الـ ICMP Replay Message تحتوي على الـ Message التي أرسلت بالإضافة إلى الـ IP للـ Router الذي تم تمرير الـ Message من خلاله والوقت الذي استغرق للوصول إلى كل Router حتى وصول الـ Packet 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 الـ IpTimeToLive ضمن الـ SetSocketOption Method 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 الـ Environment Class 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 checksum = packet.getChecksum();
packet.Checksum = checksum;

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

for (int i = 1; i < 50; i++)
{
    trace_socket.SetSocketOption(SocketOptionLevel.IP, SocketOptionName.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 IPEndPoint = 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 checksum As UInt16 = packet.getChecksum()
```

```
packet.Checksum = checksum
```

```
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 Ping وآخر لعمل الـ Routing 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 الـ Command Prompt بالشكل 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 الـ Class ProcessStartInfo والـ Class Process الموجود ضمن الـ System.Diagnostics Namespace ، حيث نمرر الـ Command والـ Arguments الذي نريد تنفيذه إلى الـ ProcessStartInfo Instance Object الـ 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 الـ MAC 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

تكمّن الفكرة الأساسية من الـ Packet Sniffing Applications بالتصنّت على الـ Network Layer أو الـ RawSocket بعد اختيار الـ Network 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 وتحليل الـ Packet 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 الـ TCP/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

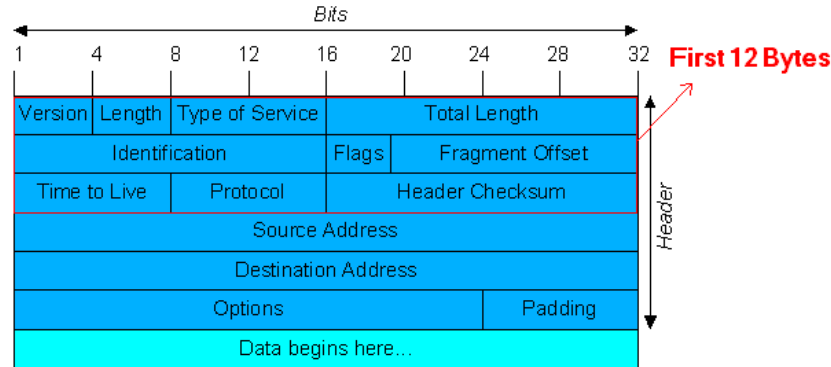
```

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 الـ Data في الـ Packet من خلال المعادلة التالية:

Data in Application Layer =

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

ويمكننا مثلا معرفة الـ IP Address لطرف المرسل مثلا من خلال تخزين الـ Packet الملتقط في Byte Array ثم تغيير الـ offset للمكان الذي نريد القراءة منه ، إذ يأتي موقع الـ IP 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 بالـ IPHeader :

**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

ولتنفيذ عملية التصنت على الـ IP Layer يجب تعريف Raw Socket ونحدد فيها الـ Interface Card ثم نقوم بعمل الـ Binding مع الـ IP Address 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 لاحقاً بحسب حجم الـ Header للـ Packet الملتقط بالإضافة إلى معرفة الحجم لكل Element في الـ Header ...

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

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



# Part 4

## Application Layer Programming

Chapter12 DNS Programming

Chapter13 HTTP Programming

Chapter14 Web Services & XML Programming

Chapter15 Remoting & Distributed Systems  
Programming & Design

Chapter16 SMTP & POP3 Programming

Chapter17 FTP Programming

---

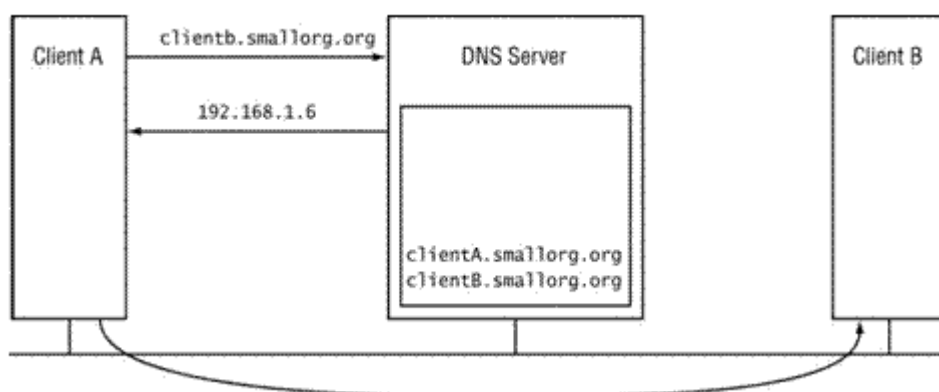
## **Chapter 12**

### **DNS Programming**

- Synchronous DNS Members
- Asynchronous DNS Members

## :DNS Programming :12

تعتبر خدمة DNS واحدة من أهم الخدمات التي تستخدم في الإنترنت والشبكات بشكل عام، وتختصر وظيفة DNS بالقيام بعملية ترجمة الـ Domain 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 الـ Client A بطلب الـ Domain الخاص بالـ Client B وذلك بإدخال الـ Domain Name الخاص به -حيث تم مسبقا قيام الـ Client B بتعريف نفسه في قاعدة البيانات الخاصة بـ DNS Server - كما يحتوي كل Client على قاعدة بيانات تحتوي على عناوين الـ Domains وتسمى بالـ local DNS حيث يقوم بالبحث بداخلها على عنوان الـ Domain من خلال الـ Domain Name فإذا لم يجده يقوم بطلب عنوان الدومين من الـ DNS Server وبعد إيجاده يقوم الـ DNS Server بإرسال العنوان إلى الـ Client ويقوم بدوره بتخزين العنوان في الـ Local DNS الخاص به ، انظر إلى الشكل التالي:



في الدوت نيت يمكننا التعامل مع DNS باستخدام System.Net Namespace والتي تحتوي على جميع الـ Classes و الـ Methods 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 الـ Computer Name ولا تأخذ هذه الدالة أي باراميترات ويمكن استخدامها كما يلي :

#### C#:

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

#### VB.NET:

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

الدالة GetHostName 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 الـ IPHostEntry 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 :

BeginGetHostByName و BeginResolve و EndGetHostByName 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 الأداء 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 الباروميتر الأول و الباروميتر الثاني يعرف فيه delegate 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 ب جلب جميع الـ IP's الموجودة على الشبكة حيث يعمل على جلب الـ host names من ProcessStartInfo من خلال الخاصية StandardOutput بتنفيذ الأمر Net View ، حيث يتم تحويله إلى الـ host name من خلال الدالة GetMachineNamesFromProcessOutput ثم تخزينها في Collicaion ثم يتم تحويل الأسماء إلى عناوين من خلال الدالة Dns.Resolve .. طبعا يتم استخدام الـ StreamReader لقراءة الـ collection الخاص بالـ ProcessStartInfo وهذا هو المثال :

**C#:**

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

namespace NetworkIPs
{
    public class Names
    {
        public StringCollection GetNames()
        {
            ProcessStartInfo _startInfo = new ProcessStartInfo("net", "view");
            _startInfo.CreateNoWindow = true;
            _startInfo.UseShellExecute = false;
            _startInfo.RedirectStandardOutput = true;
            Process _process = Process.Start(_startInfo);
            StreamReader _reader = _process.StandardOutput;
            StringCollection _machineNames =
                GetMachineNamesFromProcessOutput(_reader.ReadToEnd());
        }
    }
}
```

```

StringCollection _machineIPs = new StringCollection();
    foreach(string machine in _machineNames)
    {
        _machineIPs.Add(IPAddresses(machine));
    }
    return _machineIPs;
}
private static string IPAddresses(string server)
{
    try
    {
System.Text.ASCIIEncoding ASCII = new
System.Text.ASCIIEncoding();
        // Get server related information.
        IPEndPoint 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 الـ DNS وطرق التعامل معه في بيئة الدوت نيت ، سيتم الحديث في الفصل التالي عن HTTP وطرق برمجته في بيئة الدوت نيت.



---

## **Chapter 13**

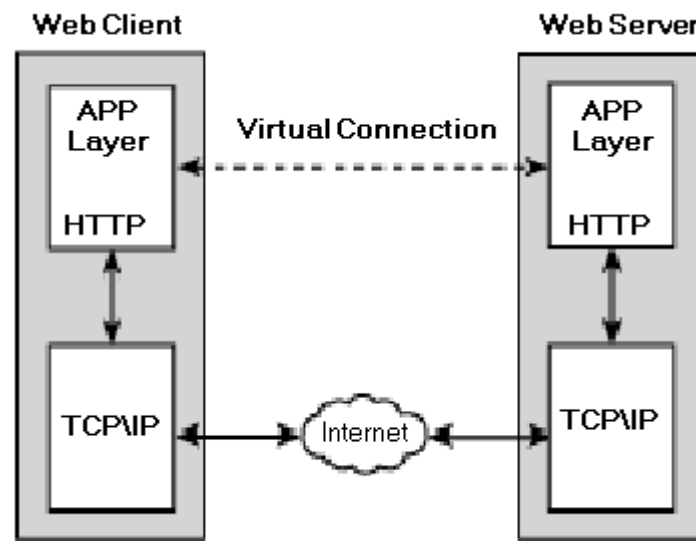
### **HTTP Programming**

- 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 الـ HTTP بشكل عام على انه البرتوكول المستخدم لتوصيل طلب المستخدم User Request إلى الويب Server ثم قيام الـ web server بالرد على الـ Request والذي يسمى بـ Server Response وتأكيد تستطيع نقل جميع أشكال (Multimedia) من النص وصورة و صوت و فيديو وغيره .. من الـ Web Server إلى الـ Client Application كـ Byte object.

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



#### أولا : Downloading From Web Server

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

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

**C#:**

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

```

Console.WriteLine(Encoding.ASCII.GetString(response));
}
}

```

### **VB.NET:**

```

Imports System
Imports System.Net
Imports System.Text

```

Class DownloadData\_Method

```

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

```

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

### **C#:**

```

using System;
using System.Net;

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

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

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

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

```

### **VB.NET:**

```

Imports System
Imports System.Net

```

```
Imports System.Text
```

```
Class DownloadData_Method
```

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

3- **OpenRead** ووظيفتها إنشاء Read Only Stream بين الزبون والـ Server لجلب بيانات من URL محدد وتخزينه في Stream Object بعد تمرير الـ URL 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 الـ Stream Object .

**C#:**

```
using System;  
using System.IO;  
using System.Net;  
  
class OpenRead_Method  
{  
public static void Main ()  
{  
WebClient wc = new WebClient();  
string response;  
  
Stream strm = wc.OpenRead("http://www.google.com");  
StreamReader sr = new StreamReader(strm);  
  
while(sr.Peek() > -1)  
{  
response = sr.ReadLine();  
Console.WriteLine(response);  
}  
sr.Close();  
}  
}
```

**VB.NET:**

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

## Class OpenRead\_Method

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

ويحتوي الـ **WebClient Class** على مجموعة من الـ **Properties** والتي تستخدم لجلب معلومات عن الـ Web Host مثل الـ **ResponseHeaders** property والذي يستخدم لجلب معلومات هامة عن الـ web host مثل عدد الـ **Headers** ونوع الـ **cash control** واسم الـ **Server** و نوع الـ **Encoding** المستخدم وغيرها من المعلومات الهامة، ويستخدم كما يلي كمثال:

### C#:

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

### VB.NET:

```
Imports System  
Imports System.Net  
  
Class ResponseHeaders_property
```

```

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

```

#### //Output:

```

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

```

#### ثانيا : Uploading to Web Server

يدعم الـ WebClient أربعة Methods لتحميل البيانات إلى الـ Web Server وهي :  
**OpenWrite** -1 يستخدم لإرسال الـ Stream Data إلى الـ Web Server وذلك بعد تمرير عنوان الـ URL والملف والنص الذي نريد كتابته على الـ Web Page طبعا يجب أن تملك الصلاحيات لذلك ويستخدم كما يلي كمثال:

#### C#:

```

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

```

```
}
```

### **VB.NET:**

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

```
Class OpenWrite_method
```

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

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

### **C#:**

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

```
Method_class UploadData
```

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

### **VB.NET:**

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

```
Class UploadData_Method
```

```

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

```

**UploadFile -3** وتستخدم هذه الدالة لرفع ملف من الـ Local Computer إلى الـ Web Host وهي بسيطة 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

```

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

**C#:**

```

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

class UploadValues_Method
{

```



```

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

    Console.WriteLine(Encoding.ASCII.GetString(response));
}
}

```

### **VB.NET:**

```

Imports System
Imports System.Collections.Specialized
Imports System.Net
Imports System.Text
Class UploadValues_Method

```

```

    Public Shared Sub Main()
        Dim wc As WebClient = New WebClient
        Dim nvc As NameValueCollection = New NameValueCollection
        nvc.Add("firstname", "Fadi")
        nvc.Add("lastname", "Abdel-qader")
        Dim response As Byte() =
        wc.UploadValues("http://localhost/mypage.aspx", nvc)
        Console.WriteLine(Encoding.ASCII.GetString(response))
    End Sub
End Class

```

### **ثالثا: المواضيع الأكثر تقدما في الـ HTTP Programming:**

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

#### **1- استخدام HttpWebRequest Class :**

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

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

### **C#:**

```

using System;
using System.Net;

```

```

class ProxyServer_Property
{

public static void Main ()
    {
HttpRequest 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 الـ `HttpWebRequest Object` ثم نعرف `WebProxy Object` من الـ `webProxy Class` ونسند له عنوان الـ `Proxy Server` ورقم الـ `Port` وبعد ذلك نستطيع إسناده إلى أي `Object` باستخدام الخاصية `Proxy` التي تكون موجودة عادة في جميع `HttpWebRequest Objects`.

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

**C#:**

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

**VB.NET:**

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

بعد تعريف الـ **HttpWebRequest** Object نقوم بتعريف الـ **Stream** Object ونسند له الـ **Request Stream** من خلال الدالة **GetRequestStream**.

## 2 - استخدام **HttpWebResponse** Class:

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

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

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

### C#:

```
HttpWebRequest hwr =  
(HttpWebRequest)WebRequest.Create(http://www.amazon.com);  
hwr.CookieContainer = new CookieContainer();  
وذلك قبل الـ HTTP Request ثم نسنده إليه كما يلي :  
HttpWebResponse hwrsp = (HttpWebResponse)hwr.GetResponse();  
hwrsp.Cookies = hwr.CookieContainer.GetCookies(hwr.RequestUri);
```

### VB.NET:

```
Dim hwr As HttpWebRequest =  
CType(WebRequest.Create("http://www.amazon.com"),  
HttpWebRequest)  
hwr.CookieContainer = New CookieContainer  
Dim hwrsp As HttpWebResponse = CType(hwr.GetResponse,  
HttpWebResponse)  
hwrsp.Cookies = hwr.CookieContainer.GetCookies(hwr.RequestUri)
```

6- **Headers** : لمعرفة الـ HTTP Headers

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

8- **Method** : لمعرفة الدالة والتي تستخدم في الـ HTTP Response

9- **ProtocolVersion** : لمعرفة الـ HTTP Version

10- **ResponseUri** : الـ URL الخاص بـ Server

11- **Server** : لمعرفة اسم الـ Server

12- **StatusCode** : لمعرفة نوع الـ Coding المستخدم

13- **StatusDescription** : لإرجاع Text يحتوي على حالة الـ HTTP

بينما في هذا الفصل كيفية برمجة الـ HTTP في بيئة الدوت نيت وطرق التعامل مع الـ **HttpWebRequest** والـ **HttpWebResponse** في بيئة الدوت نيت ، سوف نتحدث في الفصل التالي عن **Web Services** والـ **XML** 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 الـ HTTP وبيننا فيه كيفية التفاعل بين الـ web server والـ client ويعتبر هذا الفصل مكمل لما تحدثنا عنه في الفصل السابق.

تتلخص وظيفة الـ web services بإمكانية الاستفادة من الـ Methods الموجودة بالـ web server داخل برنامج الزبون، وباستخدام بروتوكول الـ SOAP وهو اختصار لـ Simple Object Access Protocol يتم نقل الـ Result من الـ web Services server إلى الـ Client بعد تحويلها إلى الـ XML - extensible Markup Language حيث تنتقل عبر بروتوكول الـ HTTP إلى جهاز الزبون والهدف من استخدامه هو تسهيل وصول الـ Data من الـ web server إلى الـ Client من خلال الـ firewalls والبيئات المختلفة إذ أن جميع بيانات الشبكات تدعم بروتوكول الـ HTTP والذي يعمل على الـ Port 80 . ولا تختلف لغة الـ XML عن الـ HTML إذ تستخدم نفس القواعد في الـ HTML وهي مجموعة من الـ Elements والـ Attributes 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 الـ XML سواء Text, Binary Data, Serialization Object أو أي شيء آخر 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 الـ XmlDocument :

```
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 لحزين 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()

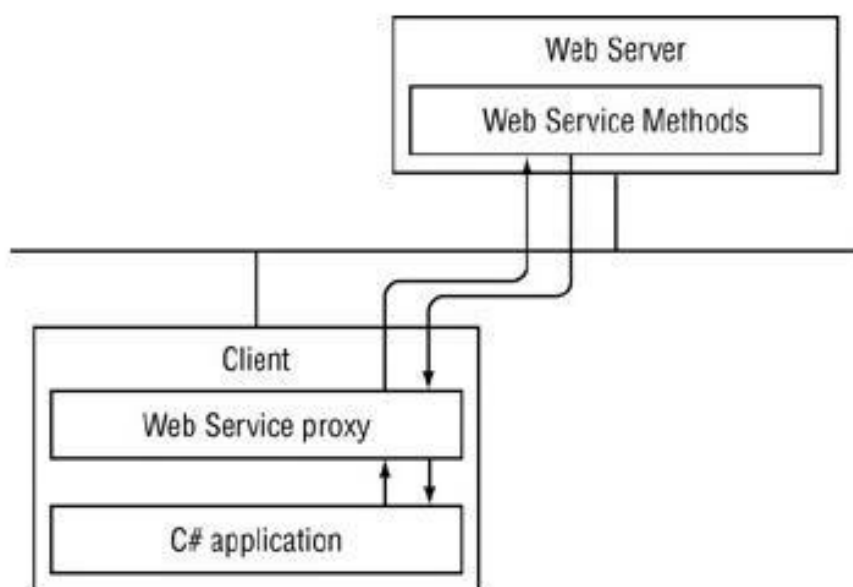
XMLWriter.WriteEndElement()
XMLWriter.Flush()
XMLWriter.Close()
MessageBox.Show("تم")
    Catch ex As Exception
        MessageBox.Show(ex.Message)
    End Try
End Sub

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 الـ web services بثلاثة مراحل وهي :

- 1- The web service server : والذي يتم من خلاله إرسال واستقبال البيانات عبر بروتوكول الـ SOAP باستخدام الـ IIS والـ ASP.NET .
- 2- The proxy object : والذي يسمح للـ Client بإرسال و استقبال البيانات من وإلى الـ web Services Server حيث يتم تعريفه في الـ HttpRequest من خلال الكلاس الـ WebProxy وهو ما بينته في الجزء السابق.
- 3- The client application : وهو الواجهة الخاصة بـ Web 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 الـ WebService للـ Class 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

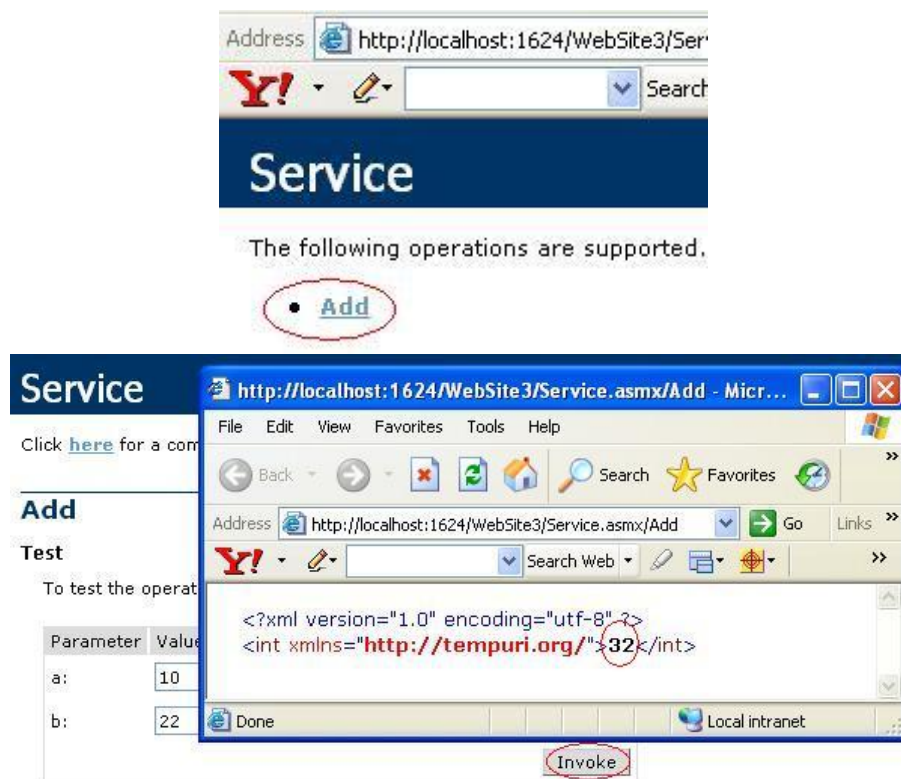
Integer

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 Client على شكل XML باستخدام بروتوكول SOAP وكما يظهر في الشكل التالي :



ولإنشاء برنامج Client يجب أولاً تحويل الكلاس السابق إلى File Dll وإرفاقه بالـ Client Resources ويتم استخدامه كما يلي :

**C#:**

```

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

```

**VB.NET:**

```

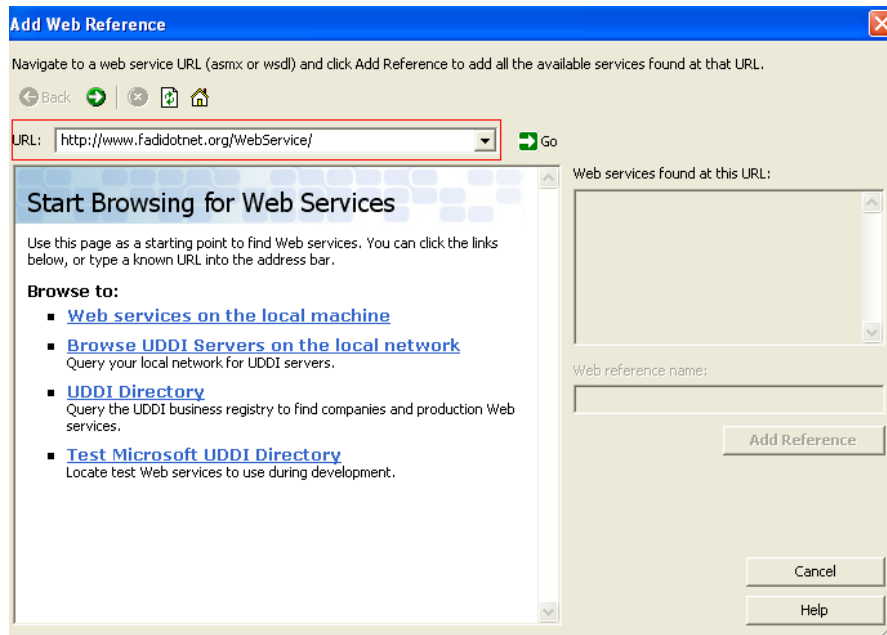
Class Client_side
    Public Shared Sub Main(ByVal argv As String())
        Dim mm As My_main_class = New My_main_class
        Dim x As Integer = Convert.ToInt16(argv(0))
        Dim y As Integer = Convert.ToInt16(argv(1))
        Dim sum As Integer = mm.Add(x, y)
        Console.WriteLine(sum)
  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

```

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



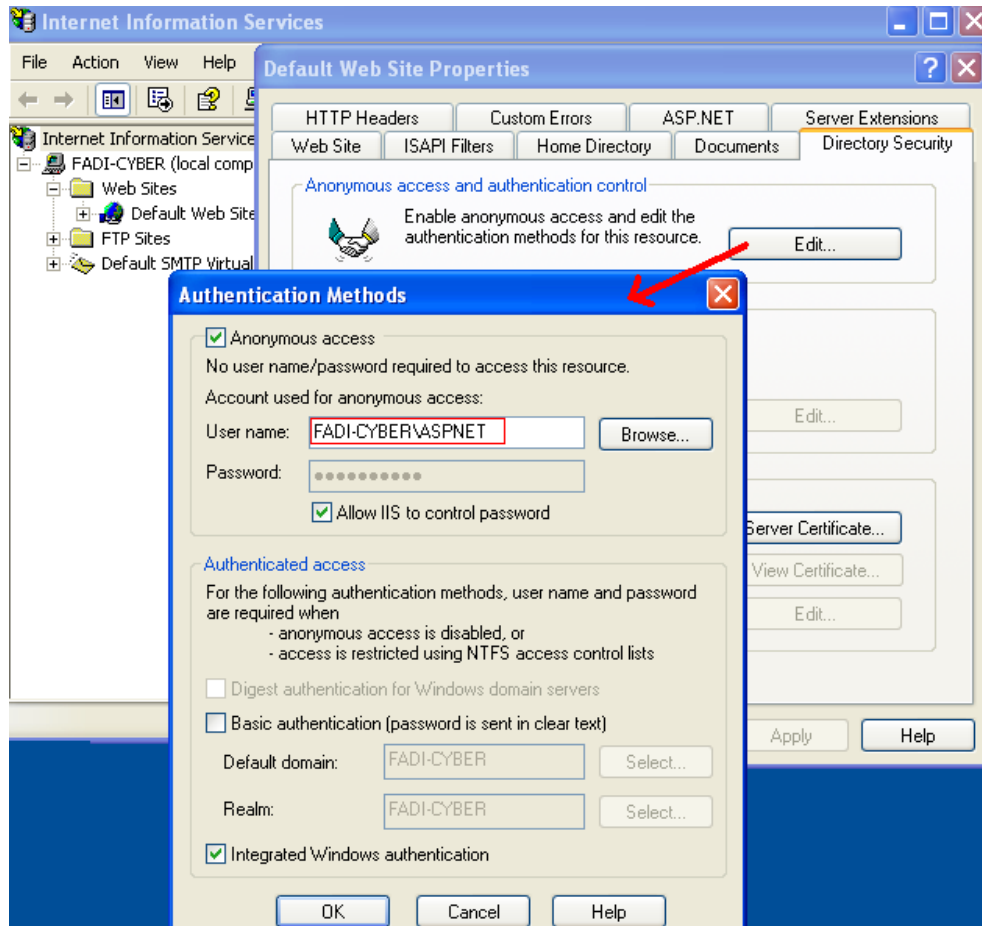
حيث يتم الاختيار سواء كانت الـ Web Services على نفس الجهاز أو على جهاز بعيد وذلك بوضع الـ URL للموقع وكما في الشكل التالي:



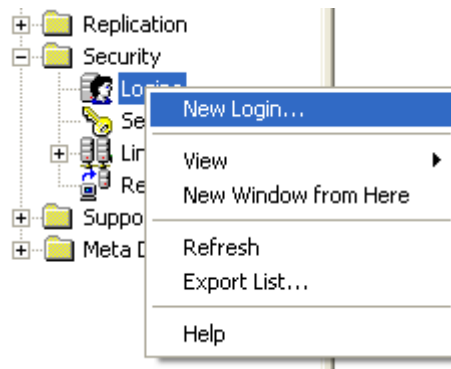
عندها نستطيع استدعاء الـ Methods السابقة في البرنامج كما وإنها على نفس الجهاز ، سنبين في الجزء التالي من هذا الفصل كيفية ربط الـ ADO.NET مع الـ Web Services لإرسال الـ Result كـ XML إلى الـ Client.

### 13.3: Using ADO.NET With Web Services

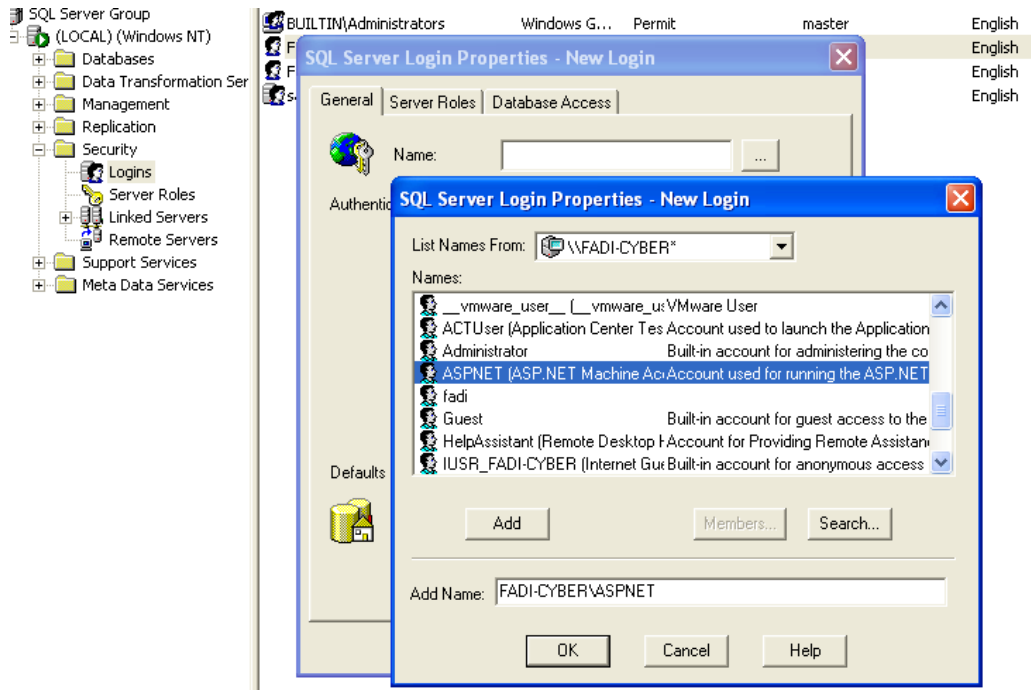
استخدمنا في المثال السابق الـ Web Services لإرسال بيانات إلى الـ Client مستخدماً بروتوكول الـ SOAP ، وفي هذا المثال سنقوم بإنشاء Web Services لنقل بيانات من الـ SQL Server Database إلى الـ Client ويمكن أن تكون هذه البيانات عبارة عن Text أو Image أو غيره ... وللبدء سنقوم أولاً بتأكد من إعدادات الـ Authentication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



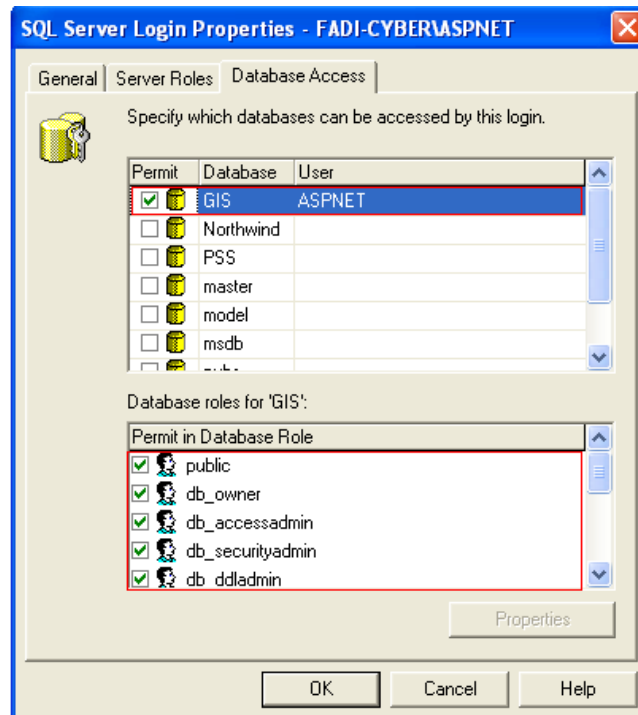
حيث نحدد User معين ونعطي له صلاحيات معينة لدخول على قاعدة البيانات والموجودة على الـ SQL Server ، ونعطي له هذه الصلاحيات من الـ Security ضمن الـ Microsoft SQL Server Enterprise Manager الخاص بالـ Microsoft SQL Server وكما في الشكل التالي:



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 والـ Server Roles من الـ SQL Server Login Properties وكما في الشكل التالي:



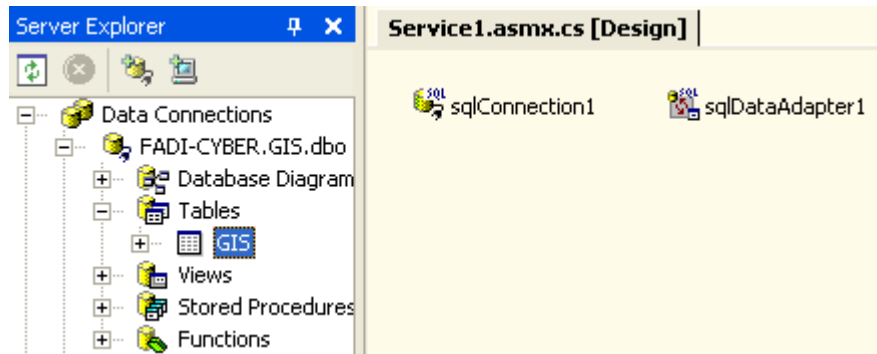
وبتأكيد يمكنك تحديد الصلاحيات التي تريدها والتي تتناسب مع البرنامج الذي تريد إنشائه.

وللبداء سنقوم بعمل فكرة بسيطة لإنشاء نظام GIS بسيط يرسل الـ Client اسم البلد الذي يختاره من الخارطة إلى الـ Web 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 & "'"
```

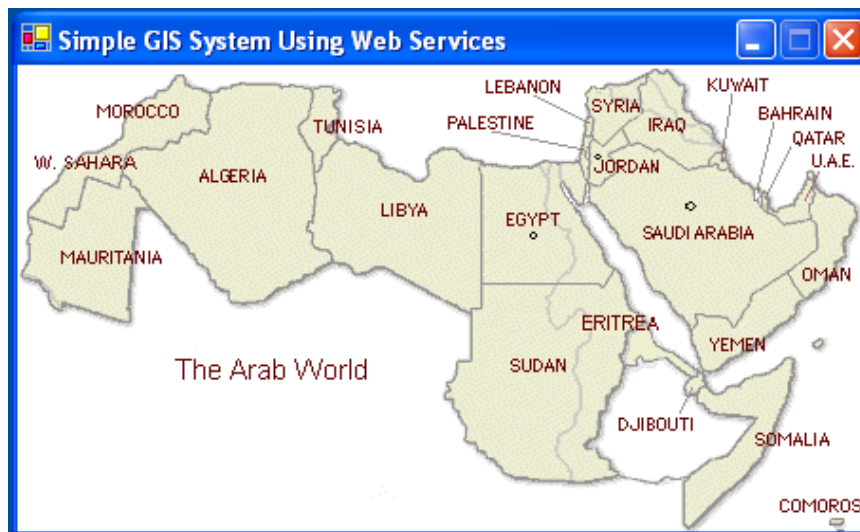


```

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 ال Client سنقوم بربطه مع ال Web Service التي تم إنشائها وكما في المثال السابق وسيمرر ال Client اسم الدولة إلى ال Web Services Method ويرجع لنا معلومات عن الدولة، وسيكون الشكل العام لل Client كما يلي:



نستطيع تحديد حدود كل بلد بعمل Areas حيث عند الضغط عليها يرسل اسم البلد إلى ال Web Services ، ولكي لا ندخل بتعقيدات ال GDI سنحل الموضوع بإدراج Panel على كل بلد ونجعل لون الخلفية له من النوع الشفاف ال Transparency ، وسيكون الكود التالي عند حدث ال Mouse Down على ال Panel الموضوع 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 مثلاً 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 :

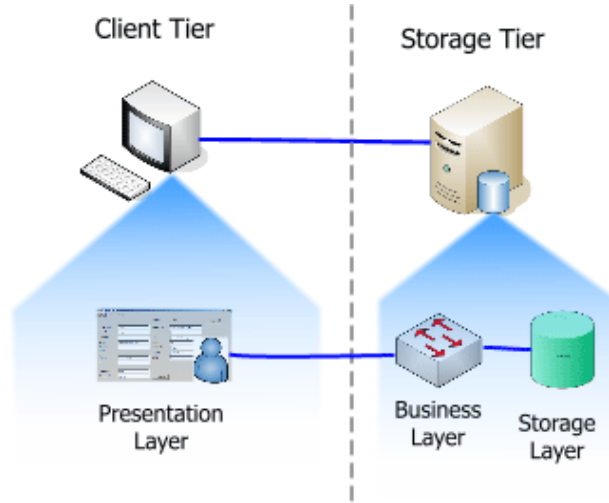
يعرف النظام الموزع على انه مجموعة من الأجهزة التي تعمل مع بعضها كنظام واحد بمعنى توزيع مهام البرنامج على مجموعة من الأجهزة بحيث تعمل بشكل متوازي وهو ما يسمى بالـ Load Balance حيث تكون هنالك نسخة من البرنامج على كل Server وتعمل مع بعضها على أساس توزيع الحمل أو بشكل متسلسل بحيث تعمل مع بعضها على أساس توزيع 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 الأعمال Business Tier وجميع القواعد التي لها علاقة بالعمل مثل طريقة حساب الخصومات Discounts أو طريقة الدفع أو طريقة توصيل السلعة لـ زبون 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 مستخدماً 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 Servers المتعددة ، وسوف نبين في الجزء التالي من هذا الفصل كيفية الربط باستخدام الـ Remoting 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 ألعاب الحاسوب تستخدم أسلوب البيع الإلكتروني (التجارة الإلكترونية) ، حيث سيقوم الزبون بالدخول على الـ Web Site الخاص بالشركة والمبنى 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 % لـ زبائن المقيمين في الأردن و فلسطين والعراق وسوريا وخصومات تصل إلى 50 % لـ زبائن الذين يشترون 3 منتجات أو أكثر من منتجات الشركة.

وقبل البدء ببناء النظام سنقوم بتقسيم هذه المهام على ثلاثة أجزاء وكما يلي:

- 1- واجهة المستخدم الـ Presentation ويمكن أن تكون موقع الكتروني مبني على الـ ASP.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 الموضوعية 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 account_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 account_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 الـ DHL 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 المشتري 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

وجداول 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 أولاً 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:

```

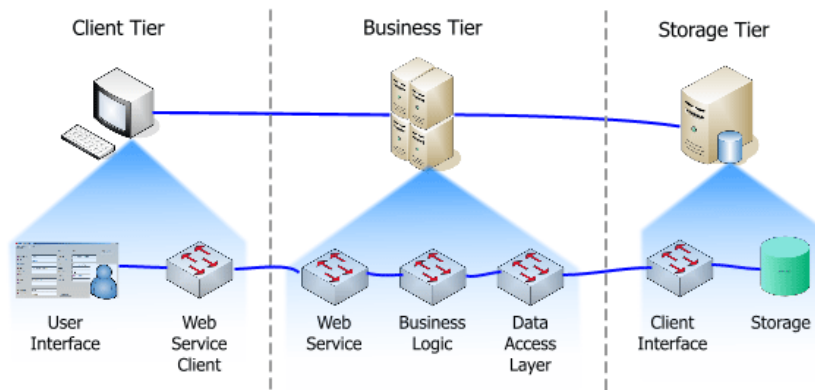
SqlConnection1.Open()
SqlCommand1.CommandText = "insert into Clients_info values('" &
Email_TextBox.Text & "','" & Pass_TextBox.Text & "','" &
Name_TextBox.Text & "','" & Zipcode_TextBox.Text & "','" &
address_TextBox.Text & "')"
SqlCommand1.ExecuteNonQuery()
Response.Redirect("http://www.fadidotnet.org/welcome.aspx")
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 ال Client Information Table بوصفها ضمن ال Presentation Tier عن بقية الأمور المتعلقة بالشراء وما شابه Business Tire ويتم ربطها جميعا باستخدام Web Services يربط مع مشروع ال ASP.NET الخاص بالشركة وترسل المعلومات ك 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
account_number)
{
Return Bank_Tier( Customer_Name, Address, zip_code,
payment_method, ecard_number, Net_Cost, account_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 account_number As
String) As Boolean
Return Bank_Tier(Customer_Name, Address, zip_code,
payment_method, ecard_number, Net_Cost, account_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 والـ Remoting وتطبيقاتها في برمجيات الشبكات والنظم الموزعة.

## ثانيا - The Serialization Programming :

The Serialization Classes & Members : يمكننا من خلال الـ Serialization عمل Serializing لـ Object من جهاز إلى آخر سواء عبر الشبكة المحلية باستخدام الـ Stream Socket أو عبر الإنترنت باستخدام الـ WebServices والـ Remoting ، ويمكننا استخدام الـ Serialization في الدوت نيت من خلال الـ System.Runtime.Serialization Namespace حيث يحتوي هذا الـ Namespace على عدد ضخم من الـ Classes 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:

ويحتوي هذا الـ Class على كل الوظائف الرئيسية لعمل Serialization لأي Object ومنها يستخدم الـ Serialize Method لإرسال Object عبر الـ Stream والـ Deserialize والتي تستخدم للاستقبال الـ Serializing Object من الـ Stream و تحويله إلى Object مرة أخرى. وتأخذ عملية الإرسال نوعين هما الإرسال عبر بروتوكول الـ Soap ويأتي ضمن الـ Namespace <i>System.Runtime.Serialization.FormatterServices</i> 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 الـ Object إلى Binary Serial Stream ويأتي ضمن الـ Namespace : <i>System.Runtime.Serialization.FormatterServices</i> والذي سيتم شرحه بالتفصيل في الأمثلة القادمة في هذا الفصل.	Formatters , Formatter & FormatterServices
حيث يحتوي على الـ IFormatter Interface والـ IConvertible Interface لأستخدمهما في عمليات تمثيل الـ Object حتى يمكن إرساله عبر الـ Serializing Stream ...	FormatterConverter
ويستخدم لتوليد رقم ID عشوائي لأي Object حيث يحتوي هذا الـ Class على Two 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	ObjectIDGenerator
ويحتوي على كل المعلومات التي يمكن أن نحتاج لها لعملية الـ Serialize و الـ Deserialize لأي Object مثلا الـ AssemblyName و الـ FullTypeName والـ MemberCount و تحويلات الـ Data Types للـ Objects وغيرها...	SerializationInfo

لإنشاء أي Class Serializable يجب أولا أن يعرف بالـ [Serializable] Attribute حيث يعرف أن هذا الـ Class من الـ Classes التي يمكن أن يتم عمل Serializing عليها وكما يلي كمثال لعمل Serializable Class للـ Bank\_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 account_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 account_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 الـ Namespace System.Runtime.Serialization.Formatters.Binary أما الثاني فسيستخدم مع الـ SOAP Protocol والمعرف ضمن الـ SOAPFormatter Class والموجود ضمن الـ Namespace System.Runtime.Serialization.Formatters.Soap.

ولعمل Serialization للـ Class السابق باستخدام الـ SOAP Formatter لابد أولاً من تعريف الـ Namespaces 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 الـ Presentation Tier إليه ثم إرسالها إلى الـ Bank Server باستخدام الـ Remoting أو الـ NetworkStream 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>
<account_number id="ref-6">3399948845</account_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 كـ XML Stream وكمثال نريد إرسال صورة من جهاز إلى آخر بعد تحويلها إلى Binary Stream باستخدام الـ MemoryStream وكما يلي:

أولاً : 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 **BinaryFormatter** في برنامج الاستقبال سنستخدم الدالة المعاكسة  
للـ **Serialize** وهي **Deserialize** ثم نقوم بعمل Casting للـ **Object** المستقبل ونحوه إلى  
**Object** **MemoryStream** 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 mytcp;
Socket mysocket;
Thread myth;

mytcp = new TcpListener (5000);
mytcp.Start ();
mysocket = mytcp.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);
mytcp1.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 mytcp1 As TcpListener

Dim mysocket As Socket

Dim myth As Thread

mytcp1 = New TcpListener(5000)

mytcp1.Start()

mysocket = mytcp1.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)

mytcp1.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 ال SoapFormatter في برنامج الاستقبال:  
 ولإستخدام بروتوكول ال SOAP في برنامج الإستقبال سنستخدم الدالة المعاكسة للـ Serialize وهي Deserialize ثم نقوم بعمل Casting للـ Object المستقبل ونحوّله إلى Object MemoryStream مرة أخرى ، عندها نستطيع عرضه على الـ Picture box باستخدام الـ FromStream Method الموجودة ضمن الـ Image Class :

```

C#:
using System.Runtime.Serialization.Formatters.Soap;
using System.Runtime.Serialization;
.
.
void Image_Receiver()
{
NetworkStream myns;
TcpListener mytcp;
Socket mysocket;
mytcp = new TcpListener (5000);
mytcp.Start ();
mysocket = mytcp.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);
mytcp.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

```

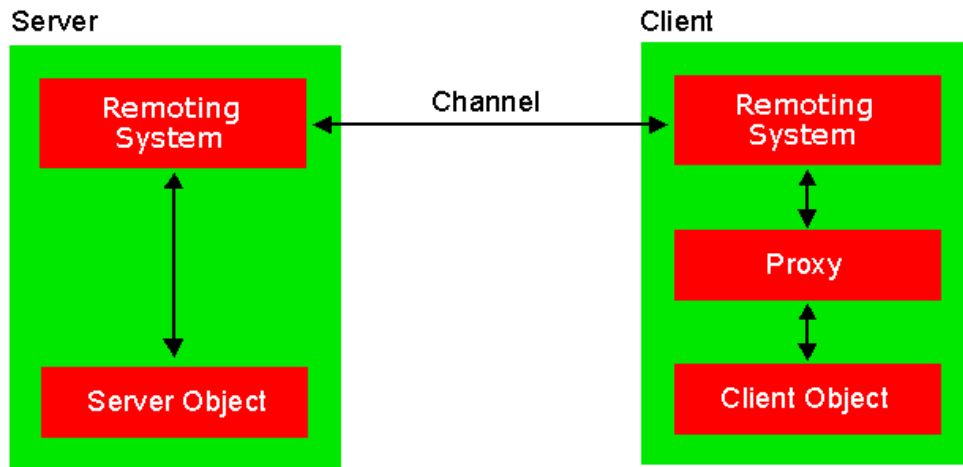
.
.
Private Sub Image_Receiver()
Dim myns As NetworkStream
Dim mytcppl As TcpListener
Dim mysocket As Socket
mytcppl = New TcpListener(5000)
mytcppl.Start()
mysocket = mytcppl.AcceptSocket()
myns = New NetworkStream(mysocket)
Dim op As Object
Dim formatter As IFormatter = New SoapFormatter
op = formatter.Deserialize(myns)
myns.Close()
Dim mm As MemoryStream = CType(op, MemoryStream)
pictureBox1.Image = Image.FromStream(mm)
mytcppl.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 بنوعها الـ Binary والـ XML 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 الـ Remoting عملية الـ Web Services إلى حد كبير حيث يتطلب استخدامها عمل الـ Proxy Object في طرف الـ Client، لكن مع إمكانيات اكبر حيث تقوم بإنشاء الـ Server بنفسك ولست مضطرا لتنصيب 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 الـ Distributed Systems والذي شرحناه في الجزء الأول من هذا الفصل حيث تبقى الـ methods و عملية المعالجة الخاصة بها على الـ Server في حين يتم إرسال الـ Result إلى الـ Clients كـ XML Data أو Binary Data حسب قناة الاتصال المستخدمة سواء الـ 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:



### : Using Remoting in Dot Net -1

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 الـ clients ، ويعتبر هذه الأمر من أهم عيوب استخدام الـ TCP Channel وخاصة في حالة الـ Web حيث لا تضمن البيئة التي سيعمل عليها الـ Client ويفضل في هذه الحالة استخدام الـ HTTP Channel ، لكن تبقى أدائية الـ TCP Channel أفضل من الـ HTTP ...  
وما يميز الـ HTTP Channel هو استخدامه بروتوكول الـ SOAP والذي يتعامل مع الـ XML 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 كـ XML أو عبر الـ TCP كـ Binary Formatter ودعمت الدوت نيت كلا منهما في الـ HttpChannel Class والموجود ضمن الـ System.Runtime.Remoting.Channels.Http Namespace والـ TcpChannel Class والموجود ضمن الـ System.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 الـ ChannelServices Class ضمن الـ System.Runtime.Remoting.Channels Namespace ، ويتم ذلك في كلا الطرفين أيضا الـ Client 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 الـ RemotingConfiguration Class والموجود ضمن الـ System.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 بارامترات الأول الـ Type الخاص باسم الـ Remote Class مع اسم الـ Dll File ومساره والذي نريد توزيعه ويأخذ الباروميتر الثاني اسم الـ URI الخاص بالـ Object الذي نريد توزيعه ويجب أن يكون الـ Object URI اسم فريد unique على مستوى الـ Application Port ، ويأخذ الباروميتر الثالث نوع الـ Remote Instance Object فإذا تم اختيار الـ SingleCall فهذا يعني إنشاء New Instance جديد لكل Client يقوم بإنشاء Instance من الـ Remote Class، وإذا تم اختيار الـ Singleton فهذا يعني استخدام نفس الـ Remote Instance

Object لكل الـ Clients، وينصح استخدام Singleton في حالة كان النظام الموزع سيستخدم Session لكل User وينصح باستخدام Singleton في حالة كان الـ Server سيبث نفس المعلومات إلى كل الـ Clients على الشبكة... أما في الـ Client فيجب تعريف نوع الـ Channel سواء TCP أو HTTP Channel حسب نوع القناة التي تم استخدامها في الـ Server 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 القناة RegisterChannel في الـ Client كما هو الحال في الـ Server وكما يلي:

**C#:**

```
ChannelServices.RegisterChannel(chan);
```

**VB.NET:**

```
ChannelServices.RegisterChannel(chan)
```

بعد ذلك نقوم بتعريف Object New Instance من الـ Remote Client حيث يجب أن يحتوي الـ Client Project على الـ Dll File الخاص 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 الـ Port 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 الـ GetObject Method والموجود ضمن الـ Activator Class ونمرر لها الـ Distributed 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:**

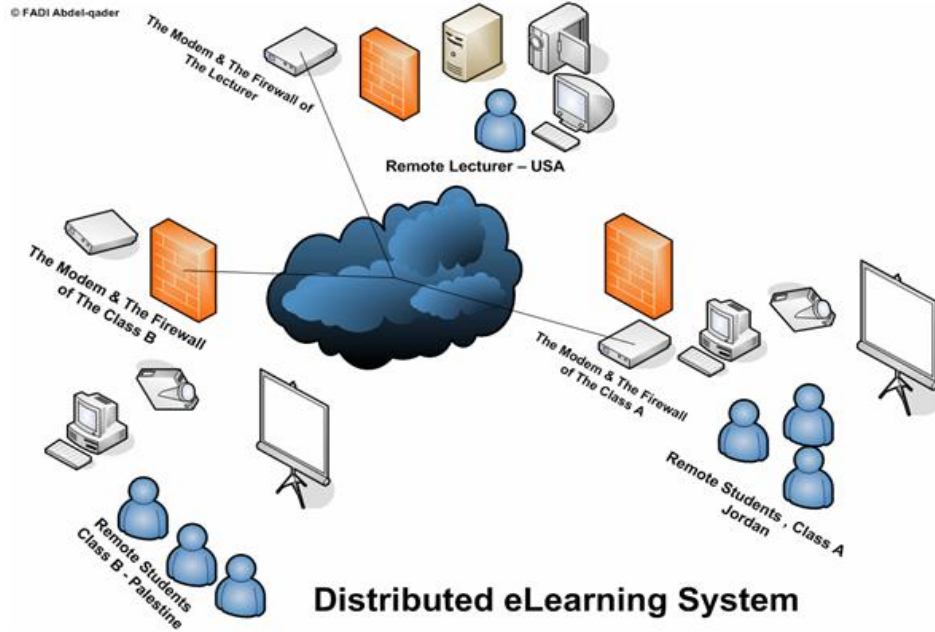
يجب أن يتم توريث الـ MarshalByRefObject Class إلى الـ Distributed Class الذي نريد توزيعه والموجود ضمن System Namespace وعند توريث الـ MarshalByRefObject إلى الـ Distributed Class سيتم تنفيذ كافة العمليات أو الـ Process على الـ Server وسيحصل الـ Client على الـ Result النهائي في حين إذا تم استخدام الـ MarshalByValueObject 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 و سطح المكتب الخاص به إلى الطلبة والموجودين في دول متعددة.



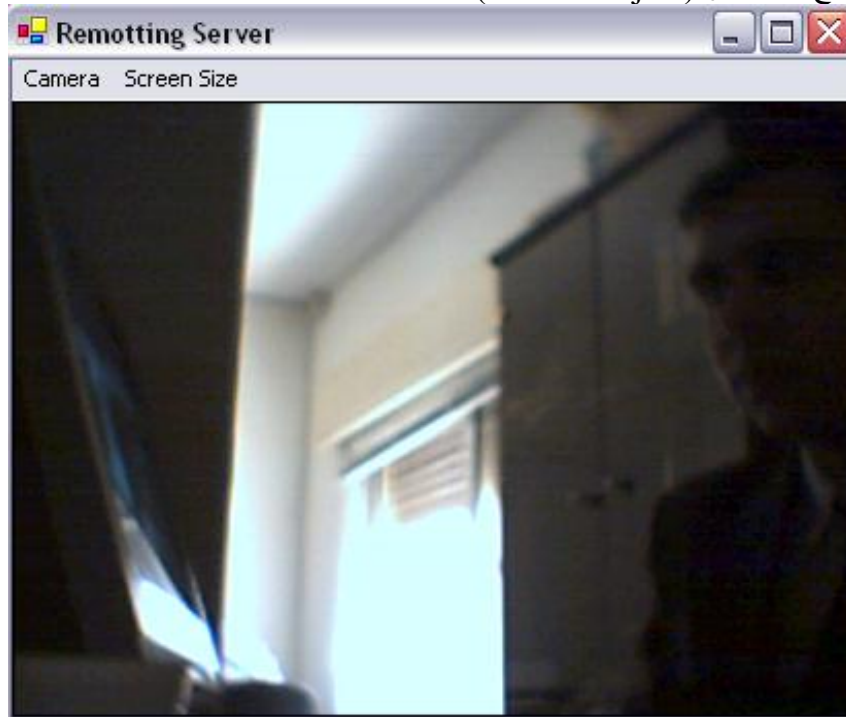
### أولاً: إنشاء نظام Distributed eLearning System :

لإنشاء هذا النظام لابد أولاً من بإنشاء Distributed Class يقوم بعملية التقاط لصورة سطح المكتب الخاص بالـ Server وبنها إلى الـ Client مستخدماً 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:

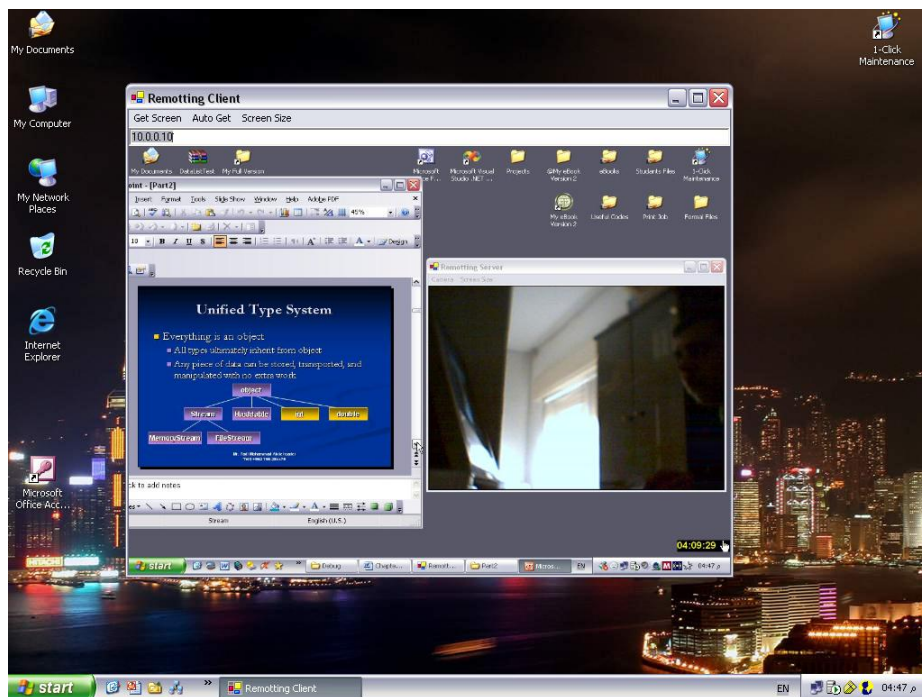


سنستخدم في هذا المثال الـ TCP Channel والـ HttpChannel وسنفارن الأداء في كل منهما ، سيكون الشكل العام للبرنامج شبيه بالشكل التالي:

أولا برنامج المحاضر (Server Project):



برنامج ال Client ويمكن أن يكون من خلال الإنترنت في حالة كان لديك Real IP أو في الشبكة المحلية وكما هو مبين في الشكل التالي:



ولإنشاء ال Remote Class والذي سيقوم بجلب صورة سطح المكتب ، سنستخدم دوال ال API ومنها ال GetDesktopWindow Method والموجودة ضمن ال user32.dll وال BitBlt Method لرسم الصورة، ثم سنقوم بتحويل الصورة إلى Byte Array وحتى نستطيع إرسالها إلى ال Clients ، وبتأكيد سنستخدم طريقة ال MarshalByRefObject وحتى تتم عملية التقاط ومعالجة الصورة على ال 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 الـ Sever .

أولاً : 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.GetType(
"ScreenCapture, ScreenCapture"),
"MyCaptureScreenServer", WellKnownObjectMode.Singleton);
```

#### **VB.NET**

```
Dim chan As TcpChannel = New TcpChannel(6600)
ChannelServices.RegisterChannel(chan)
RemotingConfiguration.RegisterWellKnownServiceType(Type.GetType(
"ScreenCapture, ScreenCapture"),
"MyCaptureScreenServer", WellKnownObjectMode.Singleton)
```

قمنا بتسمية الـ Server Object بـ MyCaptureScreenServer وطبيعة الـ Instance سيكون Singleton وهذا يعني استخدام نفس الـ Instance Object لكل الـ Clients وفي 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 الـ Server 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 الـ 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 الـ ScreenCapture Object لعمل الـ Casting على الـ Incoming Remote Object Instance ، 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 الـ Constructor 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 IP Remote 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)
    
```



```

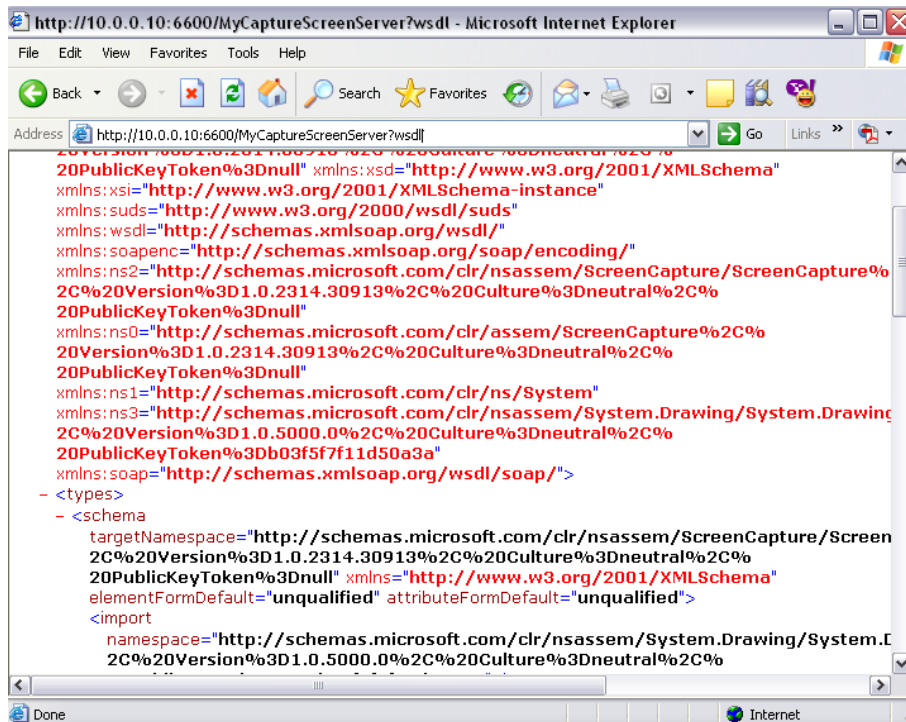
        pictureBox1.Image = Image.FromStream(ms)
    Catch ex As Exception
        timer1.Enabled = False
        MessageBox.Show(ex.Message)
    End Try
End Sub

```

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 Remote Class ، حيث تستطيع تحويل الـ XML السابق إلى ملف Dll لوضعه مع الـ References في برنامج الـ Client ويتم ذلك باستخدام الـ Metadata Namespace ومنها الـ Standard W3CXSD2001 حيث يوضع الجزء الأول الخاص بالـ 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.microsoft.com/clr/nsassem/ScreenCapture/ScreenCapture%2C%20Version%3D1.0.2314.30913%2C%20Culture%3Dneutral%2C%20PublicKeyToken%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.com/clr/nsassem/ScreenCapture/ScreenCapture%2C%20Version%3D1.0.2314.30913%2C%20Culture%3Dneutral%2C%20PublicKeyToken%3Dnull"), Serializable(),
```

```
SoapTypeAttribute(XmlNamespace:="http://http://schemas.microsoft.com/clr/nsassem/ScreenCapture/ScreenCapture%2C%20Version%3D1.0.2314.30913%2C%20Culture%3Dneutral%2C%20PublicKeyToken%3Dnull")> _
```

```
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 الـ Local 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 بالـ VOIP)، 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 الـ Mouse (X,Y) والـ Kay Press Event إلى الـ Server ويتم 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 الـ Mouse على الإحداثيات المرسله ، ويمكن استخدام الدالة GetSystemMetrics لمعرفة حجم الـ Screen للـ Server (Height & Width) ، ويمكن استخدام هذه الدوال للتحكم عن بعد بطريقتين:

1- إرسال الإحداثيات إلى الـ Server كـ Object Serialization عبر الـ Socket ثم استقبالها وتنفيذها بتمرير الإحداثيات المستقبلية إلى الدوال ، و لكن المشكلة في هذه الطريقة هي البطء في عملية إرسال الإحداثيات حيث يمكن أن يحدث تأخير أثناء الإرسال ناهيك عن الـ Bandwidth الذي ستحاجه في عملية إرسال الـ Control Object ...

2- استخدام الـ Remoting في عملية التحكم حيث يستدعي الـ Client الـ Methods الموجودة على الـ Server ومن ثم وتجرى عملية معالجتها على الـ Server Side ويمرر الـ Client الإحداثيات الخاصة بالتحكم إلى الـ Remote Methods الموجودة على الـ Server ، وتعتبر هذه الطريقة من أسرع الطرق لإرسال معلومات التحكم إلى الـ Server ، حيث يتم استدعاء الـ Methods الموجودة على الـ Server وكأنها على الـ Client و تنفذ بعد ذلك على الـ Server Side .

وللبدء سنقوم بإنشاء الـ Distributed Class والذي سيحتوي على كل الدوال الخاصة بالتحكم ، وسنستخدم الـ System.MarshalByRefObject ولكي يتم تنفيذ كل الـ Methods على الـ Server 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 Key Press Value قبل الـ Client إلى الـ Server وهي MOUSE\_INPUT لإدخال Mouse Buttons والـ KEYBD\_INPUT لإدخال Keyboard Button Press Values وتستخدم 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 الـ Mouse بناء على الإحداثيات المرسلة من الـ Client ، حيث ستستخدم لتحريك مؤشر الـ Mouse في الـ Server .

- الدالة SendInput وسنرسل فيها ثلاثة باروميترات الأول عدد الإدخالات في كل عملية و Reference Value للعملية المدخلة مثلاً في حالة الـ Keyboard فإنها تأخذ الـ Virtual Key Code للـ Button الذي تم الضغط عليه، ولاستخدام الـ SendInput Method سننشئ الـ PressOrReleaseMouseButton Method والـ SendKeystroke Method وسنمرر لكل منها مجموعة من المتغيرات حيث سنعرف الـ Instance من الـ Input Struct والذي عرفناه في بداية البرنامج وسنمرر له الـ Input Type سواء من الـ Mouse أو الـ Keyboard فإذا كان من الـ Mouse نمرر له إحداثيات المؤشر والـ Key Button سواء الـ Left أو الـ Right ، أما إذا كان المدخل من الـ Keyboard فسنمرر له الـ VirtualKeyCode 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 الـ Class السابق إلى ملف Dll حيث سيرفق بالـ Client كـ Reference وسيوضع بجانب الملف التنفيذي الخاص بطرف الـ Server :

### أولا الـ Server :

سيحتوي برنامج الـ Server 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.GetType(
"ScreenCapture, ScreenCapture"),
"MyCaptureScreenServer", WellKnownObjectMode.Singleton);
```

#### VB.NET

```
Dim chan As TcpChannel = New TcpChannel(6600)
ChannelServices.RegisterChannel(chan)
RemotingConfiguration.RegisterWellKnownServiceType(Type.GetType(
"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 الأحداث الخاصة بالـ Mouse و الـ Virtual Key Code الخاص بالـ Keyboard Buttons ، وللبداء 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 Key Code الخاص بالـ Button الذي سيتم الضغط عليه حيث سنستخدمه لمعرفة أزرار التحكم التي تم الضغط عليها أثناء كتابة الأحرف مثل الـ Shift 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 الـ 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 بالـ Server:

**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:

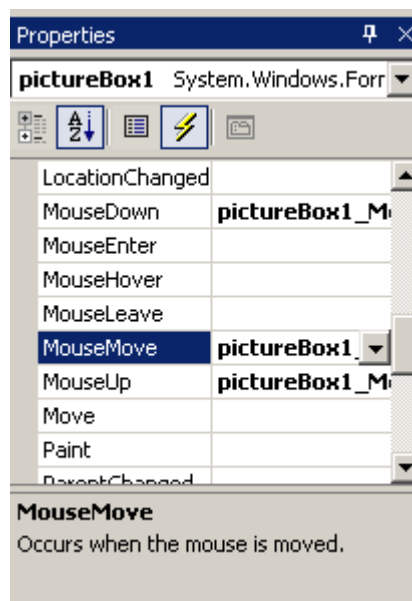
**C#:**

```
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 والتي عرفناها في الـ Class remote السابق ويوضع في الـ Mouse Move Event للـ PictureBox 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
```

وسنمرر الـ Mouse Button Event إلى الـ PressOrReleaseMouseButton Method باستدعائها من الـ Mouse up والـ Mouse Down Events وتأخذ هذه

الـ Method 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 Mouse Button 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);
}
}

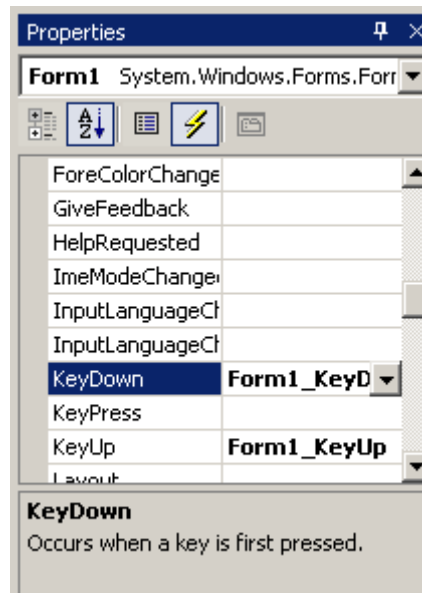
private void pictureBox1_MouseUp(object sender,
System.Windows.Forms.MouseEventArgs e)
{
if (connected == true)
{
obj.PressOrReleaseMouseButton(false, e.Button == MouseButtons.Left,
e.X, e.Y);
}
}
```

#### **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 Up والـ Kay Down 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.ToInt32(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(CType(e.KeyCode),
CByte(MapVirtualKey(System.Convert.ToInt32(e.KeyCode), 0)),
False, False)
    End If
End Sub

```

قمنا بالبداية بإيقاف فاعلية الـ Buttons باستخدام الـ `e.Handled = True` وبعد ذلك مررنا الـ `Kay code` الذي سيتم الضغط عليه إلى الـ `SendKeystroke Method` وتأخذ هذه الـ `Method` أربعة باروميترات على النحو التالي، الأول ويأخذ الـ `Byte` للـ `Virtual Kay code` الذي تم الضغط عليه ويأخذ الثاني الـ `Scan Code` لحالة الـ `Keys` 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 الـ `e.KeyCode` 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 الـ `Extended Keys` أي الضغط المطول ...

وهكذا بينا كيفية بناء نظام لتحكم عن بعد باستخدام الـ **Remoting** وطرق استخدام الـ **Serialization** 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**

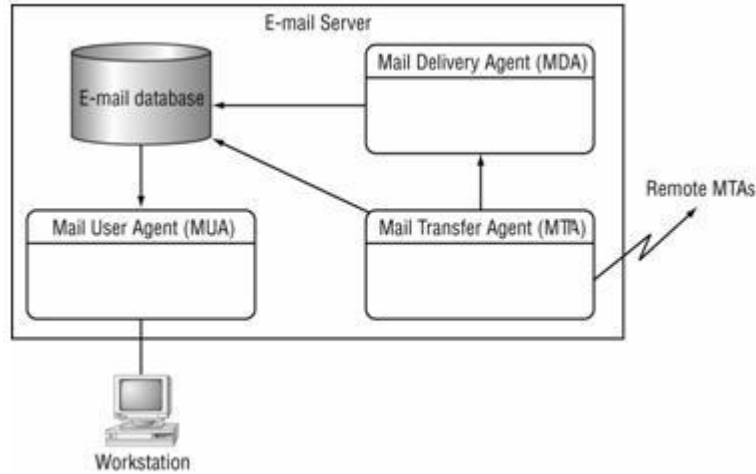
- SMTP Protocol
  1. SMTP Concept
  2. Using SMTP in Dot Net
  3. Advanced SMTP Programming
- POP3 Protocol
  1. POP3 Concept
  2. Using POP3 in Dot Net

## : SMTP & POP3 Programming :16

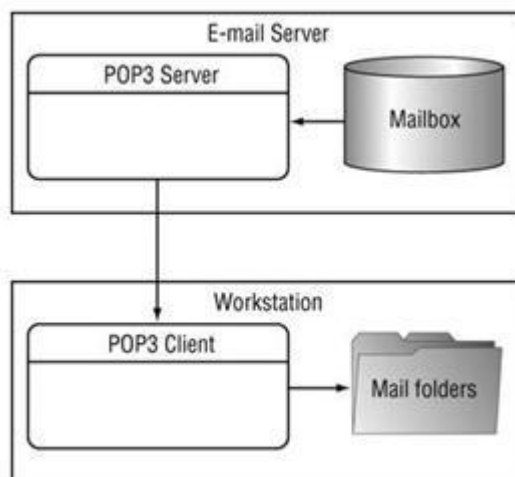
سوف نتحدث في هذا الجزء عن برمجة الـ SMTP والمسئول عن إرسال الرسائل عبر البريد الإلكتروني إلى Mail Server و الـ POP3 والمسئول عن عملية الـ Download لرسالة من الـ Mail Server إلى جهاز الزبون ...

### SMTP – Simple Mail Transfer Protocol Programming أولاً:

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

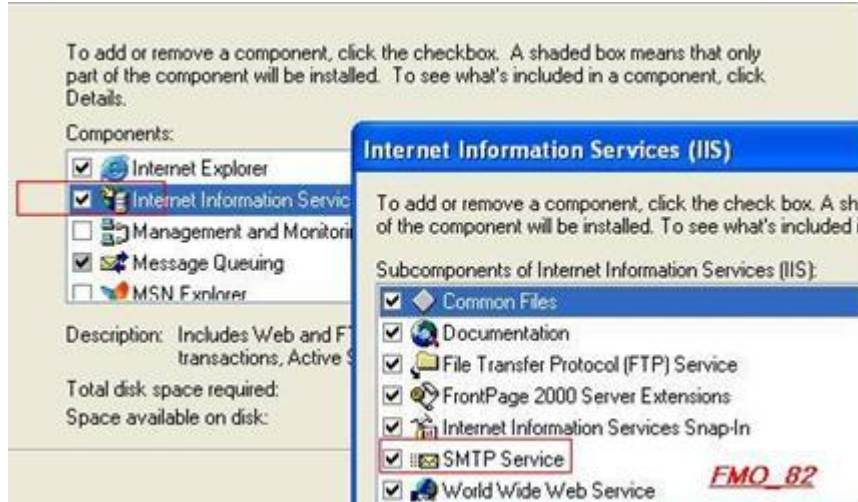


Message Transfer Agent – MTA والمسئول عن الإرسال Outgoing والتوصيل Incoming للرسائل  
Message Delivery Agent -MDA والمسئول عن عمليات الـ filtering والتأكد من وصول الرسالة  
Message User Agent -MUA والمسئول عن عملية قراءة و تخزين الرسالة في Database لدى المستقبل Client وتتم هذه العملية باستخدام بروتوكول POP - Post Office Protocol انظر إلى الشكل التالي :

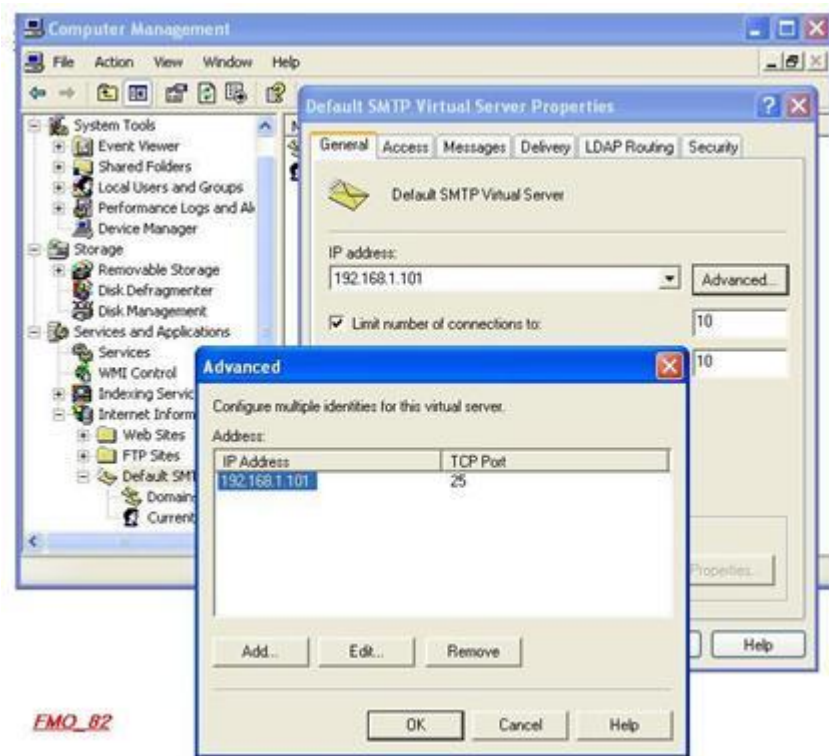


و يستخدم بروتوكول الـ SMTP Simple Mail Transfer Protocol بشكل أساسي في الـ MTA أي عمليات إرسال Outgoing وتوصيل Incoming الرسائل .

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



ثم إعداد Server من IIS كما في الشكل التالي :



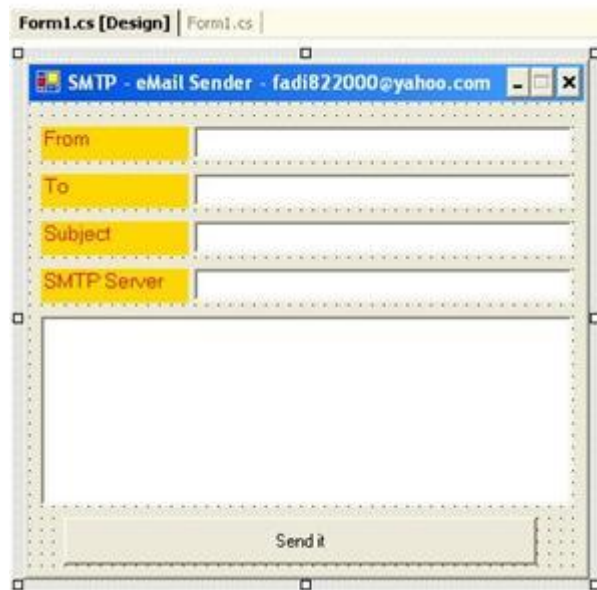
تدعم الدوت نيت استخدام بروتوكول SMTP من خلال Namespace System.Web.Mail و تحتوي على الكلاس SmtMail والتي من خلالها نستخدم الدالة Send والتي تستخدم لإرسال الرسالة عبر Port 25 وهو Port المخصص لبروتوكول SMTP و تعتبر الدالة " Send overloaded Method " حيث تأخذ عدة أشكال إذ بإمكانك

استخدامها مع باروميتر واحد إلى أربعة باروميترات ، وبشكل افتراضي نستخدم الباروميترات التالية :

`SmtMail.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 `References` ) 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;
    SmtMail.SmtpServer = textBox5.Text;
    SmtMail.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 الـ SMTP من الـ IIS كما ذكر في السابق والـ IP الخاص بالـ SMTP (والذي تم تعريفه مسبقا في SMTP Virtual Server) بالـ SMTP Server Textbox ، يجب التأكد أيضا من الـ SMTP Server لديك يدعم استخدام المكتبة CDO2 - Microsoft Collaboration Data Objects Version 2 وإلا سوف تحصل على Exception يخبرك بأنه لا يستطيع الوصول إلى CDO2 Object ، في العادة يتم استخدامها مع Windows XP و Windows 2000 وتعمل بشكل افتراضي عند تثبيت الـ SMTP Virtual Server أو مع Microsoft Exchange Server 2003 أما إذا كنت تستخدم Exchange Version 5 أو 5.5 فسوف تحصل على الـ Exception السابق الذكر .

## الجزء الأكثر تقدم: SMTP Advanced Programming

يعتبر المثال السابق مثال بسيط لإرسال رسائل عبر الـ SMTP باستخدام الـ CDO2 ، وفي العادة عند إنشاء برامج مثل برنامج الـ Outlook يتم استخدام الـ HTML Format بالإضافة إلا إمكانية إرسال ملحقات وطبعاً يعطيك عدة خيارات لإرسال و استقبال البريد الإلكتروني هل باستخدام الـ HTTP أو الـ POP3 ...

سوف نقوم بإنشاء برنامج بسيط يقوم بإرسال واستقبال البريد الإلكتروني باستخدام الـ SMTP والـ POP3 بنسبة لاستخدام الـ POP3 فيجب أن يتوفر لديك حساب الـ POP3 أو أن تقوم بتنصيب الـ POP3 Service والتي تأتي مع الـ Windows Server أو أن تستخدم الـ Microsoft Exchange Server على جهازك وإعداده بحيث يستخدم الـ POP3 إذ عندها سوف تحتاج لوجود الـ Domain Controller مثبت على الجهاز و الـ Windows Server بالإضافة إلى تثبيت الـ Active Directory عليه.

قدمت الدوت نيت دعم ممتاز لاستخدام هذه الخواص وذلك من خلال الـ Namespace System.Web.Mail وباستخدام الكلاس الـ MailMessage لدعم الـ HTML Format و الكلاس الـ MailAttachment لدعم إمكانية إرسال ملحقات مع الرسالة ولكن لبرمجة الـ POP3 يلزم استخدام الـ System.Net.Socket Namespace و الـ System.Net و الـ System.IO حيث يتم عمل الـ Session خاص مع الـ Server للقيام بعملية تفحص وجود رسائل جديدة وفي حالة وجودها يقوم بتعبئتها في الـ List Box أو الـ Treelist حسب الحاجة وعند الضغط على إحداها يقوم الـ Client بعمل الـ Download لرسالة من الـ Mail Server ولعمل الـ Advanced SMTP eMail Sender قم بأخذ الـ Object من الكلاس الـ 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>";
    Smtplib.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 الـ HTML وهذا يمكنك من وضع أي لون أو حجم أو أي شيء يمكن عمله باستخدام الـ HTML ، 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:**

```

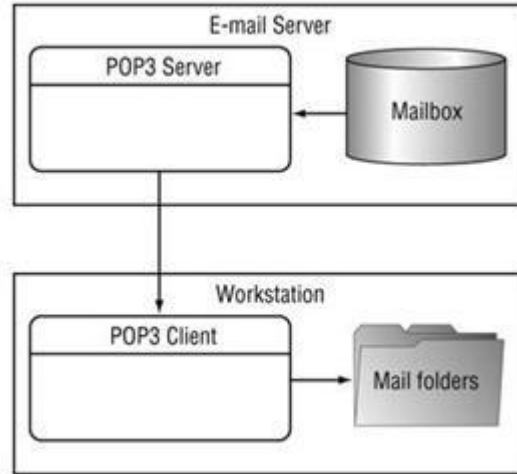
Dim myattach As MailAttachment = New
MailAttachment("Your_Attached_File_path.extension",
MailEncoding.Base64)

```

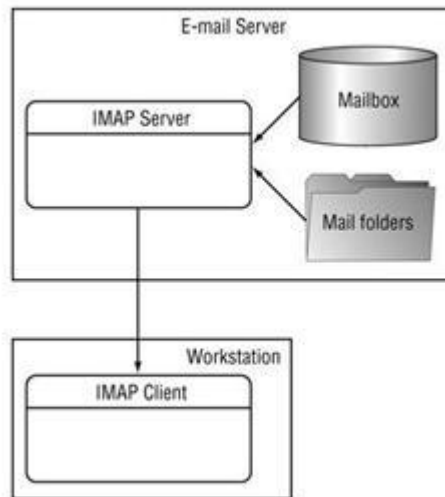
وهكذا قد انتهينا من برنامج الـ SMTP بشكل كامل ، طبعا عملية الـ Design وغيرها تعتمد على حسب ذوق وذكاء وخبرة المبرمج.

### ثانيا : POP3- Post Office Protocol Version 3 Programming

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



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



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



وبما أننا قررنا اعتماد الـ POP3 لعملية قراءة الرسائل سوف نبدأ ببرمجته إذ يلزم الأمر استخدام System.Net و System.Net.Socket و System.IO حيث يتم عمل Session خاص مع الـ Server باستخدام الـ Socket للقيام بعملية تفحص وجود رسائل جديدة وفي حالة وجودها يقوم بتعبئة عناوينها في List Box أو Treelist خاص حسب الحاجة وعند الضغط على إحداها يقوم الـ Client بعمل Download لرسالة من الـ Mail Server إلى الـ Mail 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 الـ Socket والـ Stream لإنشاء Session مع الـ Server باستخدام بروتوكول الـ TCP وقراءة الرسالة من الـ POP3 Server، ويستخدم الـ Stream Socket السابق لتمرير الـ Commands إلى الـ Server 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 الـ Global Declaration):

#### **C#:**

```
public TcpClient Server;// 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 الـ Connect 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 الـ Disconnect 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 الـ Read Last Come Email Button :

```

C#:
string szTemp;
Message.Clear();
try
{
    // retrieve mail with number mail parameter
    Data = "RETR 1"+CRLF; // لتحديد رقم الرسالة المراد قراءتها

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

```

```

VB.NET:
Dim szTemp As String
Message.Clear
Try
    Data = "RETR 1" + CRLF
    szData = System.Text.Encoding.ASCII.GetBytes(Data.ToCharArray)
    NetStrm.Write(szData, 0, szData.Length)
    szTemp = RdStrm.ReadLine
    If Not (szTemp(0) = "-"C) Then
        While Not (szTemp = ".")
            Message.Text += szTemp + CRLF
            szTemp = RdStrm.ReadLine
        End While
    End If
}

```

```
Else  
    Status.Items.Add(szTemp)  
End If  
Catch err As InvalidOperationException  
    Status.Items.Add("Error: " + err.ToString)  
End Try
```

وهكذا بينا كيفية عمل الـ POP3 وبرمجته في الدوت نت وهذا مثال بسيط تستطيع البدء منه لعمل مشروع كامل شبيه بالـ Outlook الخاص بميكروسوفت حيث تستطيع استخدام ملف الـ DLL الخاص بالإنترنت إكسبلورر لعرض الرسائل الواردة بدلاً 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.

---

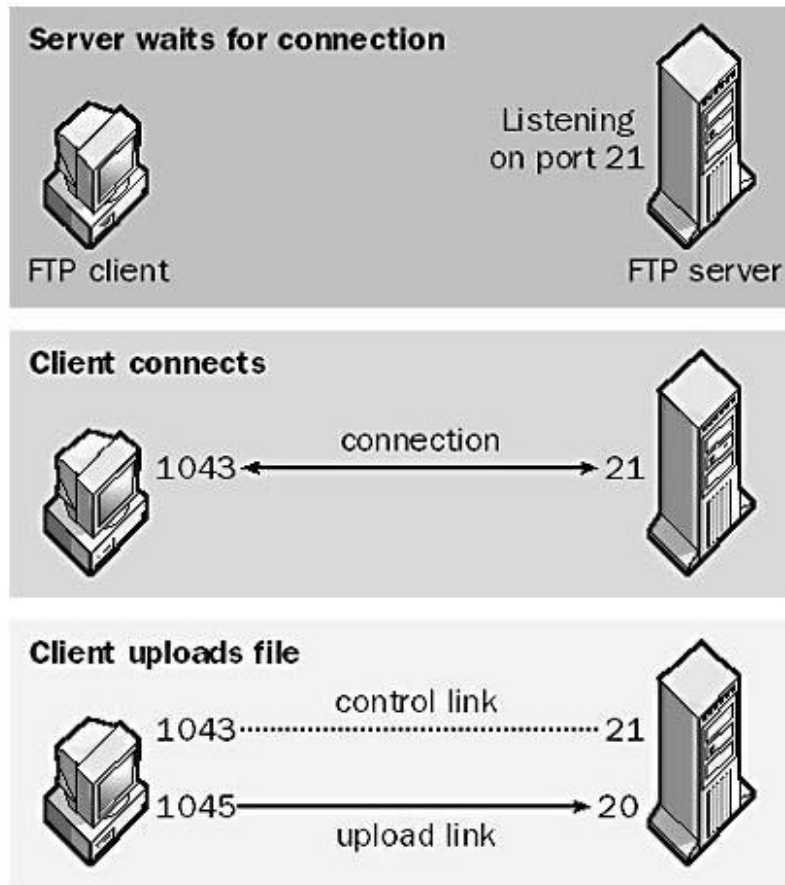
## **Chapter 17**

### **FTP Programming**

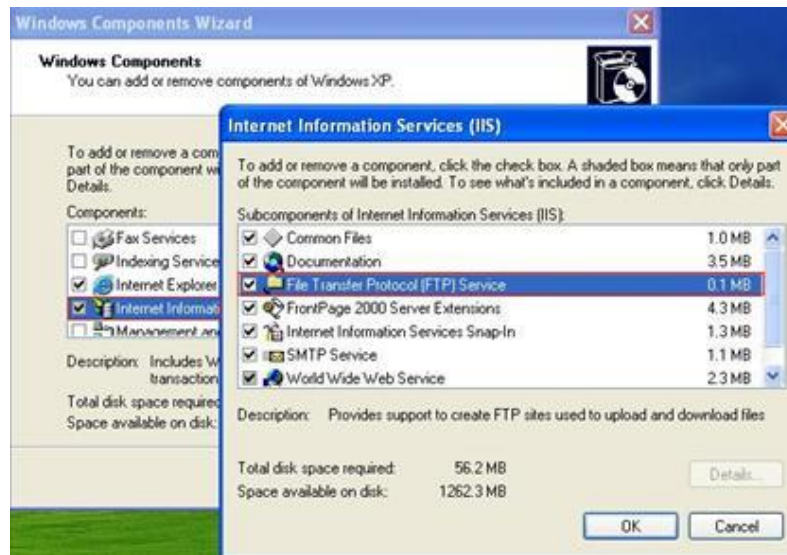
- Introduction to FTP – File Transfer Protocol
- Create a Simple Application to Transfer Files By Using COM Components
- Create a Simple Application to Transfer Files By Using Web Classes Components
- Create a Simple Application to Transfer Files By Using Socket Programming & Streaming Libraries

## : FTP – File Transfer Protocol Programming 17

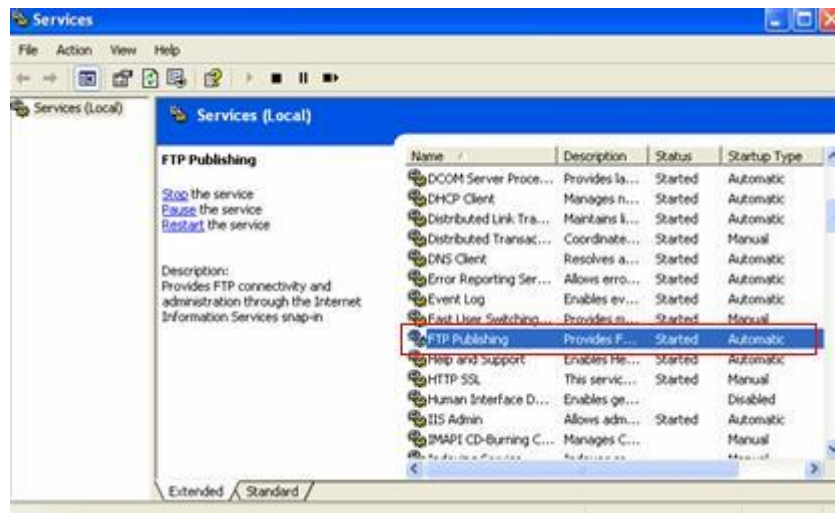
سوف نبدأ هنا بشرح بروتوكول آخر من بروتوكولات الـ Application Layer وهو بروتوكول الـ FTP والذي يستخدم بشكل أساسي في عملية الـ downloading و الـ uploading الملفات من وإلى الـ FTP Server وكالعادة في أغلب برمجيات الشبكات و التي تعتمد على وجود Client/Server حيث يقوم الـ Server بتصنت على الـ Port المخصص للـ FTP وهو الـ Port 21 باستخدام الـ TCP Connection Oriented Protocol حيث يبقى الـ Server بوضع الانتظار لورود طلب من الـ Client بإنشاء Session معه وبعد إجراء عمليات التحقق Authentication والتأكد من الصلاحيات يتم الموافقة على البدء بالجلسة حيث يتم تحديد رقم الـ Port والذي سوف يتم استقبال البيانات من خلاله ويتم الإرسال إلى جهاز الزبون عبر الـ Port 20 في الـ Server وتتضح هذه العملية كما في الشكل التالي :



**ملاحظة:** لتفعيل خدمة الـ FTP لديك بحيث يعمل جهازك كـ FTP Server يجب أولاً التأكد من أن الـ FTP Services مثبتة لديك مع الـ IIS و كما يظهر في الشكل التالي :

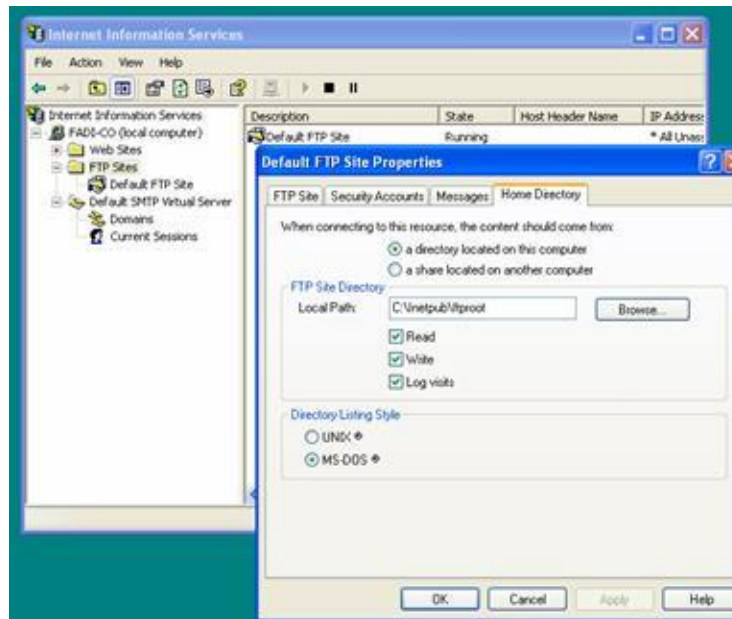


ومن ثم التأكد من تفعيلها بـ Services من Control Panel ثم Administrative Tools  
ثم Services وكما يظهر في الشكل التالي :



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





### أولا : FTP Commands

تشبه عملية الاتصال و الاستخدام لل FTP عملية الـ Telnet إلى حد كبير حيث يدعم بروتوكول الـ FTP مجموعة من الأوامر والتي يتم من خلالها عملية التخاطب مع الـ Server أو مع الـ Remote Host وتوضح هذه العملية كما في الشكل التالي :

```
C:\WINDOWS\system32\cmd.exe - ftp fadi-co
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\FADI>ftp fadi-co
Connected to fadi-co.
220 Microsoft FTP Service
User (fadi-co:(none)): FADI
331 Password required for FADI.
Password:
230 User FADI logged in.
ftp> ?
Commands may be abbreviated.  Commands are:

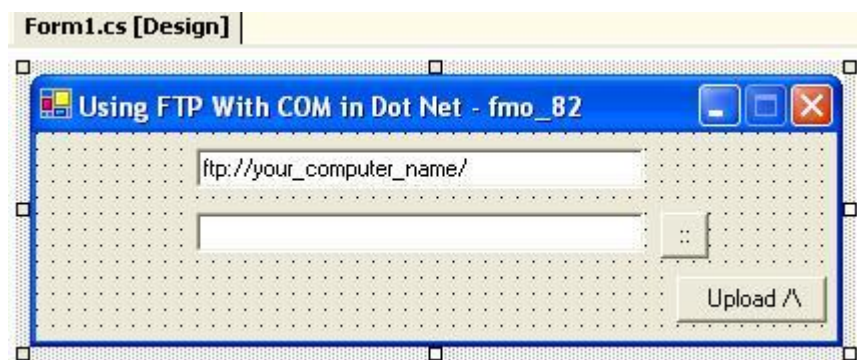
!          delete          literal          prompt          send
?          debug           ls              put             status
append     dir                     mdelete        pwd             trace
ascii      disconnect        mdir           quit            type
bell       get                mget           quote           user
binary     glob               mkdir          recu            verbose
bye        hash              mls            remotehelp
cd         help              mput           rename
close      lcd              open           rmdir
ftp> _
```

وهنا شرح لأهم الـ FTP Commands:

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

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

**ثانيا : التعامل مع الـ FTP في الدوت نيت باستخدام COM Components :**  
تدعم الدوت نيت استخدام الـ FTP عبر ITC – Internet Transfer Control وهو جزء من الـ COM Components Controls وللبداء 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 الـ Upload Button :

#### **C#:**

```
private void button1_Click(object sender, System.EventArgs e)
{
    FileInfo thisFile = new FileInfo(tbFile.Text);
    Type ITC;
    object[] parameter= new object[2];
    object ITCObject;
    ITC = Type.GetTypeFromProgID("InetCtls.Inet");
    ITCObject = Activator.CreateInstance(ITC);
    parameter[0] = (string)tbServer.Text;
    parameter[1] = (string)"PUT " + thisFile.FullName + " /" +
    thisFile.Name;
```

```
ITC.InvokeMember("execute", BindingFlags.InvokeMethod, null,
ITCObject, parameter);
}
```

### **VB.NET:**

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

تم في البداية تعريف الـ ITC من خلال الـ Class Type والموجود ضمن Namespace System.Reflection ثم عرفنا Array من النوع Object وذلك لاستخدامها في تمرير اسم الملف و الـ FTP Server إلى الدالة InvokeMember والموجودة ضمن الـ ITC Control ... Object

سوف تجد الملف الذي سيتم رفعه في المجلد :  
C:\Inetpub\ftproot

### ثالثا : التعامل مع الـ FTP في الدوت نيت باستخدام الـ Web Class :

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

**WebClient** - إذ تم دعم 2 dot net Framework استخدام الكلاس WebClient والذي يدعم التعامل مع الـ FTP 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 الـ FTP ويعرف كما يلي :

**C#:**

```
using System;
using System.Net;

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

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

**VB.NET:**

```
Imports System
Imports System.Net
Namespace FTP

    Public Class FtpRequestCreator
        Implements IWebRequestCreate

        Public Sub New()
        End Sub

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

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

**C#:**

```
using System;
using System.Net;
namespace FTP
{
```

```

public class FtpWebRequest : WebRequest
{
    private string username = "Fadi";
    internal string password = "fff";
    private Uri uri;
    private bool binaryMode = true;
    private string method = "GET";

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

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

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

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

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

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

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

```

**VB.NET:**

Imports System

Imports System.Net

Namespace FTP

Public Class FtpWebRequest

Inherits WebRequest

Private username As String = "Fadi"

Friend password As String = "fff"

Private uri As Uri

Private binaryMode As Boolean = True

Private method As String = "GET"

Friend Sub New(ByVal uri As Uri)

Me.uri = uri

End Sub

Public Property Username() As String

Get

Return username

End Get

Set(ByVal value As String)

username = value

End Set

End Property

Public WriteOnly Property Password() As String

Set(ByVal value As String)

password = value

End Set

End Property

Public Property BinaryMode() As Boolean

Get

Return binaryMode

End Get

Set(ByVal value As Boolean)

binaryMode = value

End Set

End Property

Public Overloads Overrides ReadOnly Property RequestUri() As  
System.Uri

Get

Return uri

End Get

End Property

```

Public Overloads Overrides Property Method() As String
    Get
        Return method
    End Get
    Set(ByVal value As String)
        method = value
    End Set
End Property

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

```

- **FtpWebResponse** يستخدم لعملية الرد من قبل الـ Server 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 الـ Stream 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:**

```

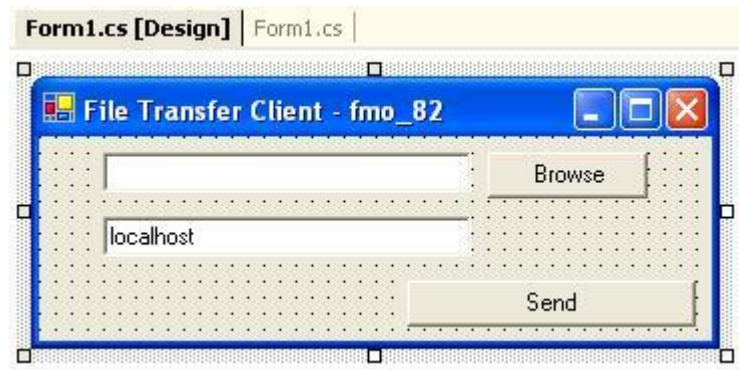
Imports System
Imports System.IO
Imports System.Net.Socket
Namespace FTP

    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

```

#### رابعاً : مثال تطبيقي لرفع ملف من جهاز Client إلى جهاز Server باستخدام الـ Stream والـ Socket:

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



سوف نستخدم Namespaces التالية :

#### C#:

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

#### VB.NET:

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

في الـ Send Button قم بكتابة الكود التالي :

#### C#:

```
try
{
    Stream fileStream = File.OpenRead(textBox1.Text);
    // Allocate 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 الـ Socket لابد من تحويله إلى مصفوفة من النوع Byte و قمنا بتسميته بـ FileBuffer ثم تعبئته باستخدام الدالة Read والموجودة ضمن FileStream وبعد ذلك قمنا بإنشاء اتصال مع الـ Server باستخدام الـ TCP Connection حيث تم إرسال محتويات الـ FileBuffer إلى الـ Server باستخدام الـ 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.Sockets;
using System.Text;
using System.IO;
```

### **VB.NET:**

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

ثم إضافة الـ Method 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
    End SyncLock
End Sub
```

```

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.ToString() + " 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)
{
    IPEndPoint 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:**

```

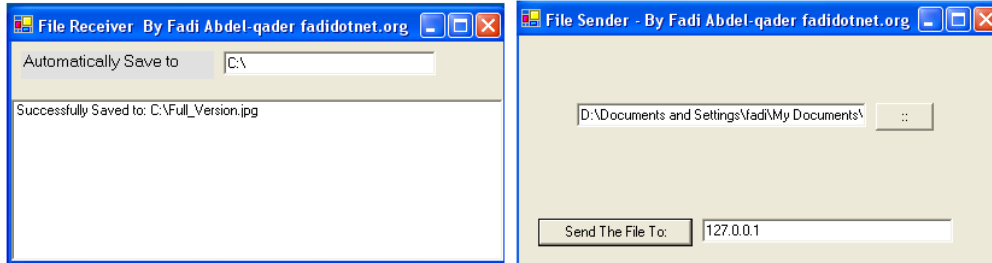
Private Sub Form1_Load(ByVal sender As Object, ByVal e As
System.EventArgs)
    Dim IPHost As IPEndPoint =
    Dns.GetHostByName(Dns.GetHostName)
    lbConnections.Text = "My IP address is " +
    IPHost.AddressList(0).ToString
    alSocket = New ArrayList
    Dim thdListener As Thread = New Thread(New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

```

باستخدام الـ Thread تم تنفيذ الـ Method listenerThread والتي قمنا فيها بتعريف الـ TcpListener وتفعيله على الـ Port 8880 حيث سيتم قبول أي طلب يأتي من الـ Client على هذا الـ Port وبعد ذلك استدعاء الدالة handlerThread والتي سيتم فيها استقبال الـ Data Stream وتخزينها في Byte Array ثم قراءتها وتخزينها في المكان المحدد وباستخدام الـ FileStream.Write حيث مررنا له الـ Stream والذي يحتوي على اسم الملف . thisRead والـ DataByte Array .

## خامسا استخدام الـ Serialization والـ FileStream Class لإرسال ملف من جهاز إلى آخر:

يمكننا الاستفادة من ميزة الـ Serialization 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 الـ FileInfo Class لجلب اسم الملف وحتى يتم إرساله كـ Serialization Object باستخدام الـ BinaryFormatter 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 mytcp;
Socket mysocket;
Thread myth;
BinaryReader bb;

void File_Receiver()
{
mytcp = new TcpListener (7000);
mytcp.Start ();
mysocket = mytcp.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();

```



```

mytcp1.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 mytcp1 As TcpListener
Private mysocket As Socket
Private myth As Thread
Private bb As BinaryReader
Private Sub File_Receiver()
mytcp1 = New TcpListener (7000)
mytcp1.Start ()
mysocket = mytcp1.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()
mytcp1.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

Chapter18 Cryptography & Network  
Security Programming

Chapter19 Socket Permissions

---

## **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



### : Cryptography Namespace Overview : أولا

**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 الـ Classes 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 وتجري عليها عمليات لمحاولة فك الشفرة ناهيك عن الحجم الهائل التي قد تحجزه من الـ Network Bandwidth. وتم حل هذه المشكلة بجعل عملية التشفير تتم على مستوى الـ Stream نفسه ويستخدم لهذه العملية الـ **CryptoStream** Class حيث يتم استخدام مفتاحين لتشفير مفتاح التشفير Encryption Key ومفتاح لفك التشفير IV Installation Victor Decryption ويشتراط استخدام نفس المفتاحين في عملية التشفير وفك التشفير ويستخدم الكلاس السابق مع الـ **FileStream** أو **MemoryStream** حيث نمرر له الـ Stream 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 الـ TripleDES 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:

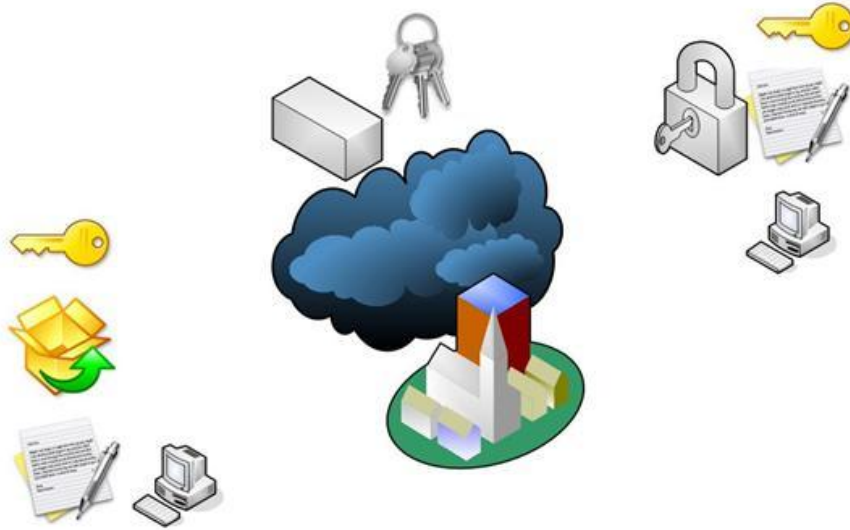
```
Dim Keyy As Byte() = {&H1, &H2, &H3, &H4, &H5, &H6, &H7,
&H8, &H9, &H10, &H11, &H12, &H13, &H14, &H15, &H16}
Dim IVv As Byte() = {&H1, &H2, &H3, &H4, &H5, &H6, &H7, &H8,
&H9, &H10, &H11, &H12, &H13, &H14, &H15, &H16}
ms.Position = 0
Dim data As Byte() = New Byte(1023)
Dim csr As CryptoStream = New 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
```

**Asymmetric algorithms -B**: الأسلوب الغير متماثل وهو أكثر أمانا من الأسلوب المتماثل إذ تشفر البيانات باستخدام مفتاح عام Public Key ولفك التشفير يستخدم مفتاح خاص Private Key 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 Key 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 الـ RSACryptoServiceProvider وذلك لتوليد المفتاح العام والخاص ونرفق المفتاح العام مع الرسالة ومن ثم يقوم المستلم بفك الرسالة باستخدام المفتاح الخاص لاحظ الشكل التالي:



تشفير الرسالة باستخدام مفتاح عام وخاص ( العام يرسل مع الرسالة ) والخاص يتفق عليه الطرفان المرسل والمستقبل

سنقوم الآن بإنشاء مشروع تشفير وفك تشفير يستخدم طريقة الـ RSA Algorithm ، سيكون الشكل العام للبرنامج كما يلي:

سنقوم بالبداية بإنشاء Tow Methods الأولى نمرر لها الـ String Plain Message والـ PrivateKey وذلك لتشفير الرسالة بناءا عليه ، وأخرى لفك التشفير ونمرر لها الرسالة المشفرة كـ Byte Array والمفتاح الخاص كـ String ، وسنستخدم الـ RSACryptoServiceProvider Class لتشفير الرسالة باستخدام الـ RSAAlgorithms ، ونمرر له الـ Private Kay كـ CspParameters Object وباسـتخدام الـ EncryptMethod والـ DecryptMethod ضمن الـ RSACryptoServiceProvider Class نتم عملية التشفير وفك التشفير ونمرر لها الرسالة المراد تشفيرها أو فك تشفيرها بالإضافة إلى قيمة True أو False لتفعيل طريقة التشفير كـ FOAEP 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

```

ولا استخدام الـ Methods السابقة يجب أولا إنشاء Object Instance من  
 الـ Class CspParameters قم بتمرير المفتاح العام إلى الـ KeyContainerName  
 Proparity ضمن الـ Object Instance الخاص بالـ Class CspParameters وكما يلي:

**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

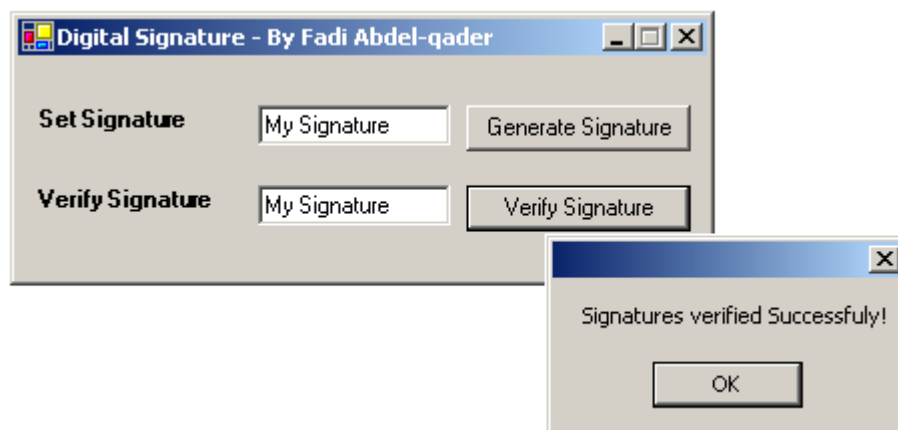
```

لاحظ أن من الفروق بين استخدام أسلوب التشفير الـ RSA والـ DES هي إمكانية الـ RSA 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 الـ Digital Signature وهو ما سنبينه في الجزء التالي من هذا الفصل.

### ثانياً: DSACryptoServiceProvider for Digital Signature Algorithm

التواقيع الرقمية: والهدف منها التحقق من هوية الشخص مرسل الرسالة وكمثال يقوم المرسل بتوليد ملخص لرسالة باستخدام Hash 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 DSACryptoServiceProvider لتوليد المفتاح العام والخاص ثم نكون الـ Hash sign Value ونخزنه في Byte Array ولفحصه نولد hash sign value جديد ونقارنه بالسابق فإذا تشابها عندها نقرر أن الشخص هو نفسه صاحب الرسالة المرسلة، سنقوم الآن بإنشاء مشروع نستخدم فيه الـ DSACryptoServiceProvider لتوليد توقيعات رقمية ثم إجراء عملية التحقق عليه، وسيكون الشكل العام للبرنامج كما في الشكل التالي:



سنقوم بإنشاء Four Methods ، الـ GetHashCode لتوليد الـ Hash Code لتوقيع الرقمي وسياخذ String Message وسيرجع الـ Hash Code كـ Byte Array وسيستخدم الـ ComputeHash Method من الـ MD5CryptoServiceProvider Instance الـ Object لتوليد الـ Hash Code السابق، الـ VerifyHash method ونستطيع من خلالها المقارنة بين الـ Message المرسلة والـ Hash Code المولد من الرسالة لتأكد من صحة تطابق النص مع الـ Hash المولد ، الـ CreateSignature Method وستستخدم لتوليد توقيع رقمي من الـ Hash Code الممرر حيث سنستخدم الـ RSAPKCS1SignatureFormatter Class ونمرر له الـ RSACryptoServiceProvider Instance Object والـ Hash الـ gorithm وباستخدام الـ CreateSignature Method سنولد التوقيع الرقمي ، الـ VerifySignature 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[] GetHashCode(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 RSAPParameters = New RSAPParameters
    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()

```

```

.
.
.

```

توليد التوقيع الرقمي 'Try'

```

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

```

```

.
.
.

```

لتأكد من صحة التوقيع 'Try'

```

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 الـ RSA، 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 نستطيع فك تشفير الرسالة وإرجاعها إلى حالتها السابقة ويستخدم بشكل أساسي لتوليد الـ Passwords وفي توليد التواقيع الرقمية الـ Digital Signature وفي اغلب الحالات تستخدم لتخزين كلمة المرور Password في الـ Database بشكل امن.

ويستخدم الـ SHA1Managed والـ SHA256Managed والـ SHA384Managed والـ SHA512Managed لتعريف Hash Object ومنه نستخدم الـ ComputeHash Method لتوليد الـ hash code وتخزينه في byte Array وكما يلي كمثال:

```
C#:
SHA1Managed shaM1 = new SHA1Managed ();
byte[] my_kay1= ASCIIEncoding.ASCII.GetBytes("convert this text to
hash code");
byte[] hashed_kay1 = shaM1.ComputeHash(my_kay1);
MessageBox.Show(ASCIIEncoding.ASCII.GetString(hashed_kay1));

SHA256Managed shaM2 = new SHA256Managed();
byte[] my_kay2= ASCIIEncoding.ASCII.GetBytes("convert this text to
hash code");
byte[] hashed_kay2 = shaM2.ComputeHash(my_kay2);
MessageBox.Show(ASCIIEncoding.ASCII.GetString(hashed_kay2));

SHA384Managed shaM3 = new SHA384Managed ();
byte[] my_kay3= ASCIIEncoding.ASCII.GetBytes("convert this text to
hash code");
byte[] hashed_kay3 = shaM3.ComputeHash(my_kay3);
MessageBox.Show(ASCIIEncoding.ASCII.GetString(hashed_kay3));
SHA512Managed shaM4 = new SHA512Managed ();
byte[] my_kay4= ASCIIEncoding.ASCII.GetBytes("convert this text to
hash code");
byte[] hashed_kay4 = shaM4.ComputeHash(my_kay4);
MessageBox.Show(ASCIIEncoding.ASCII.GetString(hashed_kay4));
```

### **VB.NET:**

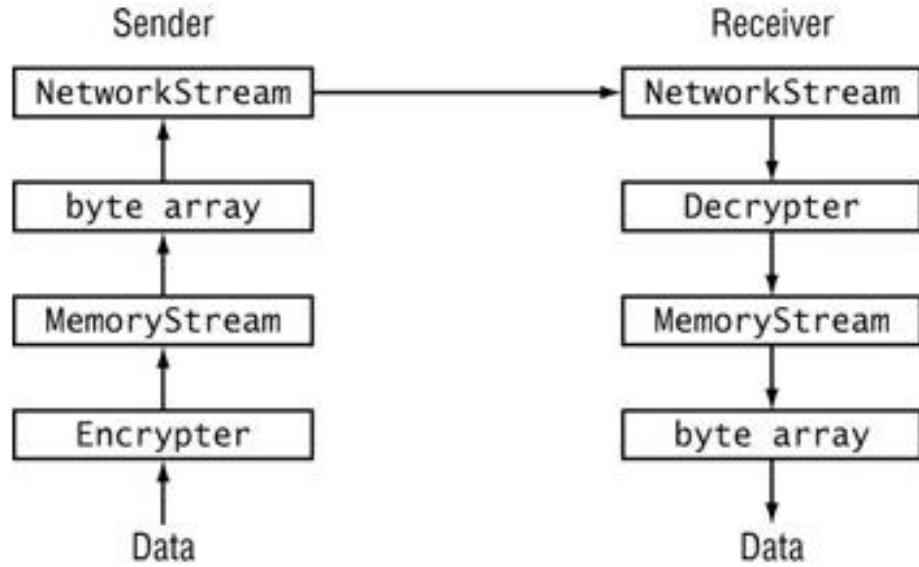
```
Dim shaM1 As SHA1Managed = New SHA1Managed
Dim my_kay1 As Byte() = ASCIIEncoding.ASCII.GetBytes("convert
this text to hash code")
Dim hashed_kay1 As Byte() = shaM1.ComputeHash(my_kay1)
Msgbox(ASCIIEncoding.ASCII.GetString(hashed_kay1))
Dim shaM2 As SHA256Managed = New SHA256Managed
Dim my_kay2 As Byte() = ASCIIEncoding.ASCII.GetBytes("convert
this text to hash code")
Dim hashed_kay2 As Byte() = shaM2.ComputeHash(my_kay2)
Msgbox(ASCIIEncoding.ASCII.GetString(hashed_kay2))
Dim shaM3 As SHA384Managed = New SHA384Managed
Dim my_kay3 As Byte() = ASCIIEncoding.ASCII.GetBytes("convert
this text to hash code")
Dim hashed_kay3 As Byte() = shaM3.ComputeHash(my_kay3)
Msgbox(ASCIIEncoding.ASCII.GetString(hashed_kay3))
Dim shaM4 As SHA512Managed = New SHA512Managed
Dim my_kay4 As Byte() = ASCIIEncoding.ASCII.GetBytes("convert
this text to hash code")
Dim hashed_kay4 As Byte() = shaM4.ComputeHash(my_kay4)
Msgbox(ASCIIEncoding.ASCII.GetString(hashed_kay4))
```

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 المرسلّة باستخدام أي من الأساليب السابقة لتشفير ثم نحول البيانات المشفرة إلى Stream لإرسالها عبر الـ Socket باستخدام الـ Stream Network كمثال، ثم يقوم الطرف المستقبل باستقبال الرسالة باستخدام الـ Stream Network عبر الـ Socket، ويجب أن تكون عملية فك التشفير كما هي الخوارزمية المستخدمة إنشاء التشفير سواء RSA أو DES أو التوقيعات الرقمية، ثم نستطيع تحمل الرسالة إلى الـ Memory stream حيث ستستقبل الرسالة كـ Stream 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 وسنستخدم فيه الـ RSA لتشفير النص المرسل حيث سيتفق الطرفان على الـ Private Key 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(msg);

    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(msg)

Dim op As Object = CObj(msg)
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
```

ثانياً المستقبل أو الـ Server :

وفي طرف المستقبل الـ Server يتم استقبال الـ Encrypted Message as an Object ثم تحويل 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 mytcp;
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
{
    mytcp = new TcpListener (7000);
    mytcp.Start ();
    mysocket = mytcp.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);

    mytcp.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 mytcp1 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
    mytcp1 = New TcpListener(7000)

```

```

mytcppl.Start()
mysocket = mytcppl.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)

mytcppl.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 الـ **Cryptography** لإجراء عمليات التشفير على الـ **Data** المرسلة عبر الشبكة ، كما بينا طرق استخدام معايير التشفير مثل الـ **DES** و الـ **Hashing** والـ **RSA** 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 الـ Permission Namespace في الدوت نيت ثلاثة أنواع من الصلاحيات وهي الـ Role- based permissions والـ Identity Permissions والـ Code access permissions...permissions

تطبق الـ Code access permissions على كل من الـ Classes 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 الـ Identity 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 PrincipalPermission Class فقط

**Socket Permission:** وتمكنك من تحديد صلاحيات استخدام الـ Socket في برمجيات الشبكات باستخدام الـ SocketPermission و الـ SocketPermissionAttribute ضمن الـ System.Net والـ System.Security.Permissions Namespaces وكمثال نستطيع منع Client Host Address معين من الاتصال مع الـ Listener Application ، ويتم ذلك بتعريف الـ SocketPermission Attribute نحدد فيها نوع العملية والـ Access Kind و عنوان الـ Host الذي سيطبق عليه الـ Permission ورقم الـ Port ونوع الـ Transport سواء موجه أو غير موجه TCP أو UDP.

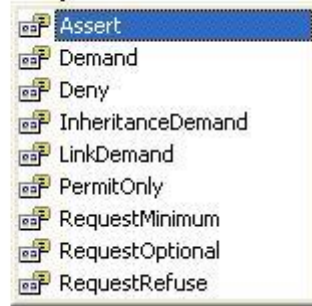
نريد في هذا المثال منع اتصال الـ Address 127.0.0.1 بالـ Socket عبر جميع الـ Ports وبغض النظر عن نوع الـ Socket المستخدم.

```
[SocketPermission(SecurityAction.Deny, Access="Connect",  
Host="127.0.0.1",Port="All", Transport="All")]
```



يمكننا الـ SecurityAction object من تحديد نوع العملية التي نريدها وكما يلي:

```
[SocketPermission(SecurityAction.Assert, Access="Connect",
```



**Assert**: وتعني السماح Client Host معين من إجراء عملية محددة  
**Demand**: وتعني تطبيق الصلاحيات على جميع الـ Classes التي تقع في منطقة الـ Stack أعلى الـ Defined Abstract  
**Deny**: وتعني منع الـ Client Host من إجراء عملية معينة.  
**InheritanceDemand**: وفيها تطبق الصلاحيات على الـ Class الذي سيرث الـ Class الحالي.  
**PermitOnly**: وفيه يمنع جميع الـ Access عدا الـ Client User المحدد ....

وفي الـ Access property نحدد نوع عملية المنع أو السماح وتأخذ خيارين هما:

**Accept** لمنع أو السماح لـ Client Socket من عمل Binding مع الـ IP Address و الـ Port المحدد.  
**Connect** لمنع أو السماح لـ Client Socket من عمل connect مع الـ Remote Host المحدد.

في الـ Host والـ Port نحدد عنوان الـ Host الذي سيطبق عليه الـ Permission و رقم الـ Port التي يتصل بها ( في الـ Port property نستطيع تمرير كلمة all لدلالة على تطبيق الصلاحية على جميع الـ Ports )

وأخيرا نحدد الـ Transport property والتي سنعرف فيها نوع الـ Socket المستخدم وتأخذ الخيارات التالية:

**All** بدون تحديد نوع الـ Socket إذ تطبق هذه الـ Permission على جميع الـ Socket Types.

**Connectionless** إذا كانت الـ Socket تستخدم الـ Datagram Protocols وكمثال بروتوكول UDP.

**ConnectionOriented** إذا كانت الـ Socket تستخدم الـ Oriented Protocols وكمثال بروتوكول TCP.

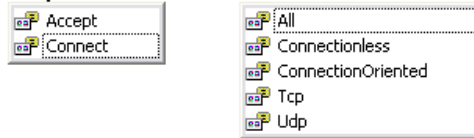
**TCP** إذ نستطيع تحديده مباشرة.

**UDP** إذ نستطيع تحديده مباشرة.

ويمكن أيضا استخدام الـ Socket Permission من خلال اشتقاق الـ Instance Object من الـ SocketPermission Class ثم تحديد نوع الـ Permission المراد وضعه ويمكنك أن تمرر له إما الـ Permission State وذلك لسماع أو منع الـ Permission المحدد أو تمرير الـ Permission الذي تريد من خلال تحديد الـ Network Access سواء اتصال Connect

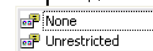
Outgoing أو Incoming Accept ثم تحديد نوع البروتوكول المستخدم سواء TCP أو UDP أو الاثنين معا وبعد ذلك نمرر الـ IP Address للـ Remote Machine في حالة السماح أو المنع للـ Access incoming أو الـ IP للـ Local Machine في حالة الـ Connect Outgoing ، ويتم ذلك كما يلي:

```
SocketPermission mySocketPermission =  
new SocketPermission(NetworkAccess.Accept, TransportType.Tcp, IP Address, Port);
```



كما ويمكن إسناد الـ Socket State للـ SocketPermission Class إذ نحدد له الـ State سواء بسماع 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(mySocketPermissionUnion);
```

**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 الـ Permission 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();
```

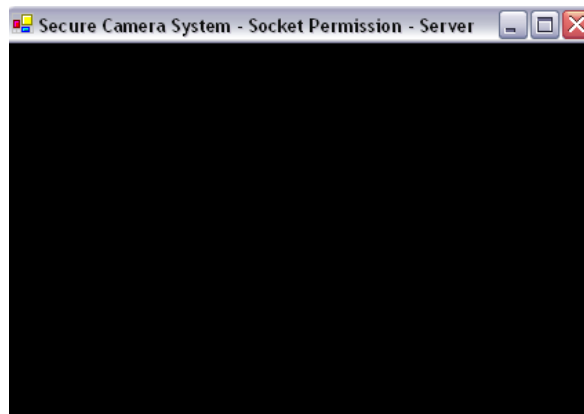
**VB.NET:**

```
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 الـ Server 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 mytcppl;
    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
    //      sp.Assert; // السماح
    sp.PermitOnly();

    mytcppl = new TcpListener (5020);
    mytcppl.Start ();
    mysocket = mytcppl.AcceptSocket ();
    myns = new NetworkStream (mysocket);

    pictureBox1.Image = Image.FromStream(myns);
    mytcppl.Stop();
```

```

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

```

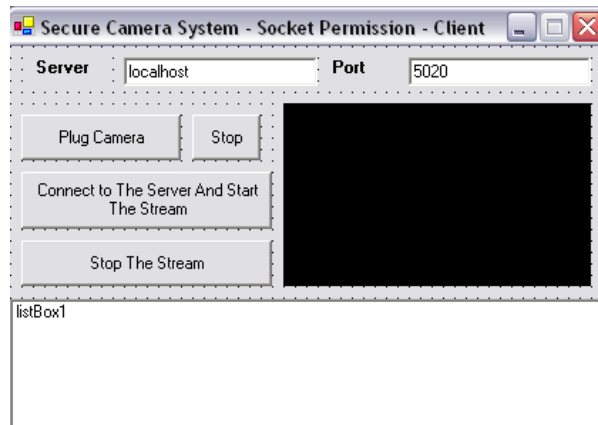
### **VB.NET:**

```

Private Sub StartMethod()
    Dim mytcp 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
    '    sp.Assert; // السماح
    sp.PermitOnly()
    mytcp = New TcpListener(5020)
    mytcp.Start()
    mysocket = mytcp.AcceptSocket()
    myns = New NetworkStream(mysocket)
    pictureBox1.Image = Image.FromStream(myns)
    mytcp.Stop()

    If mysocket.Connected = True Then
        Do While True
            StartMethod()
        Loop
    End If
End Sub

```



### 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.Jp
        eg);
        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** أو الـ **SocketPermission** ويبقى الفرق بين استخدام أي منها في حالة كنت تريد وضع الصلاحيات أثناء التشغيل وتريد إمكانية تغيير هذه الصلاحيات أثناء **Runtime** فيفضل استخدام الـ **SocketPermission** instance Object أما إذا أردت وضع الصلاحيات بشكل دائم وبدون إمكانية تغييرها وعلى مستوى الـ **Scope** للبرنامج كاملاً ، عندها يفضل استخدام الـ **Attribute** ...

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)



---

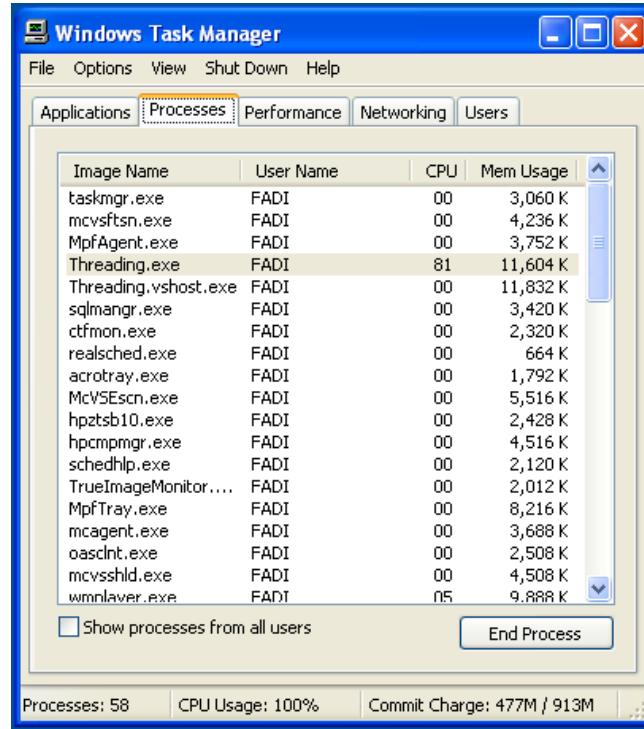
## **Chapter 20**

### **Multithreading Using & Managing**

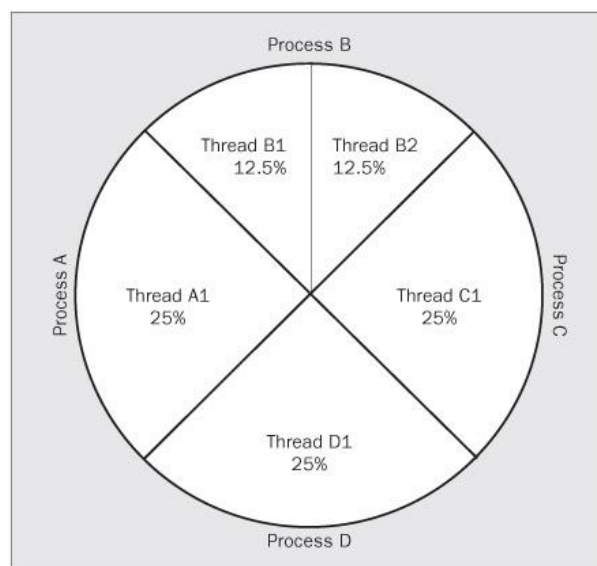
- Introduction to Threading in Dot Net
- Threading Classes & Members
- Multithreading & Network Applications

## : The Concept of Threading : 20.1

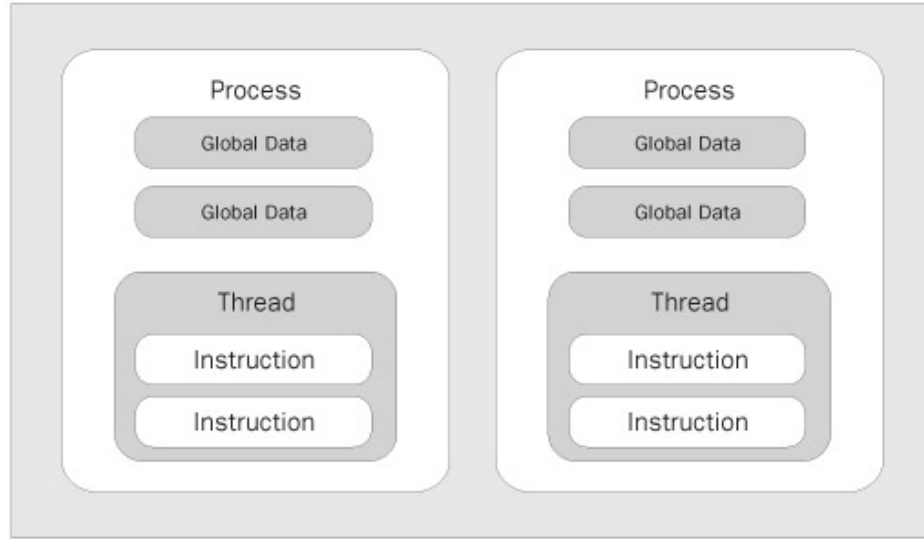
عزز مفهوم الـ Multithreading مفهوم الـ Multitasking في أنظمة التشغيل بحيث أن أي برنامج يمكنه أن يعمل على Thread منفصل عن الآخر بالإضافة إلا إمكانية أن يعمل نفس البرنامج على أكثر من Thread، أتى هذا المفهوم بعد تطوير فكرة الـ Multitasking في أنظمة التشغيل ، ويتم إدارة هذه العمليات من قبل نظام التشغيل الذي 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:

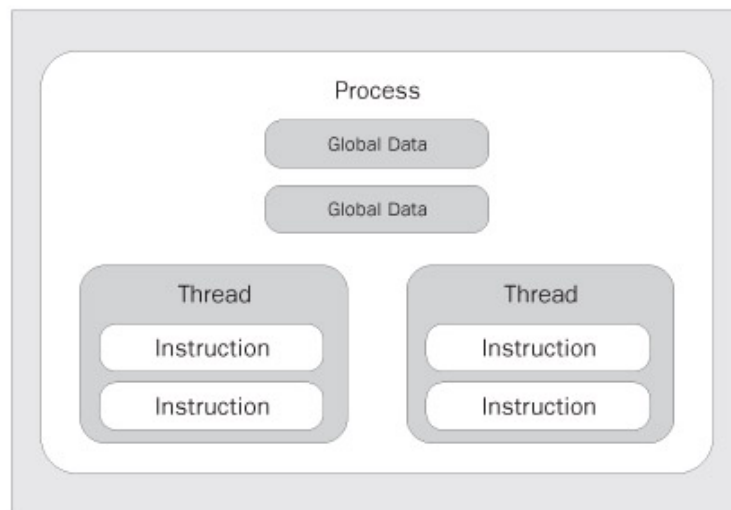


وبتأكيد يمكن التحكم بأي Thread في نظام التشغيل إذ يمكننا من عمل Interrupt له كما يمكننا إيقافه بشكل مؤقت، وأيضا إلغائه بعمل Abort له ، كل هذه العمليات متاحة لدى الـ User للقيام بها وأيضا متاح استخدامها برمجيا.



## **: Threading in Dot Net : 20.2**

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 Programming وهو 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 أولاً تعريف الـ System.Threading.Thread 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:

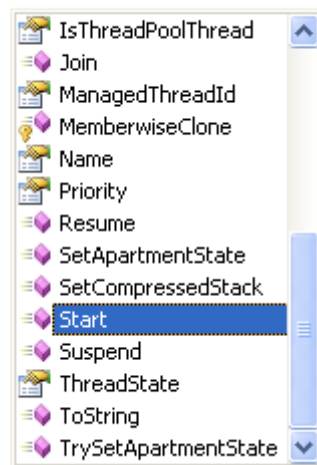
#### C#:

```
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 نهائياً من المعالجة.

#### Thread Class Methods أهم الـ

**Start** وتستخدم لبدأ الـ Thread في كل مرة يتم استدعاؤها 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 الـ Thread في وضع التوقف.

**Abort** وتستخدم لإيقاف الـ Thread بشكل نهائي وإلغائه من الذاكرة، ويفضل وضع هذه الـ Method في الـ Closing 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 لفترة زمنية معينة.

#### Thread Class Properties أهم الـ

**Name** وتأخذ Get أو Set لتحديد أو معرفة اسم الـ Thread الذي تم انشائه والهدف منها تسهيل التعامل مع الـ Multithreads في البرنامج.

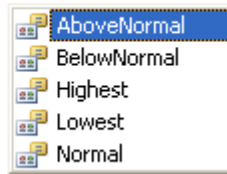
**CurrentCulture** وتأخذ Get أو Set لتحديد أو معرفة اللغة المعرفة لهذا الـ Thread.

**CurrentUICulture** وتأخذ Get لمعرفة اللغة المستخدمة في نظام التشغيل.

**IsAlive** وتأخذ Get لمعرفة إذا كان الـ Thread الحالي قيد التشغيل أم لا.

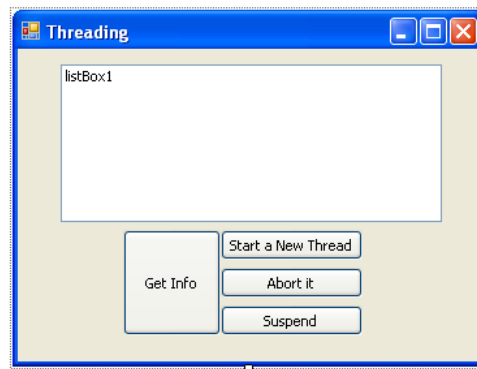
**IsBackground** وتأخذ Get أو Set لتحديد أو معرفة إذا كان الـ Thread يعمل أو سيعمل في الـ Background Process.  
**IsThreadPoolThread** وتأخذ Get لمعرفة إذا كان الـ Thread يستخدم الـ Pool queue لمجموعة من الـ Thread أم لا.  
**ManagedThreadId** وتأخذ Get لمعرفة الـ Unique ID الخاص بالـ Thread الحالي.  
**Priority** وتأخذ Get أو Set لتحديد أو معرفة الأولوية للـ Thread الحالي وفي الوضع الطبيعي تكون Normal ويمكن تغييرها كما يلي:

```
th.Priority = ThreadPriority.
```



حيث يحدد فيها مدى أهميتها 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 ويطلع 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(); }
}

```

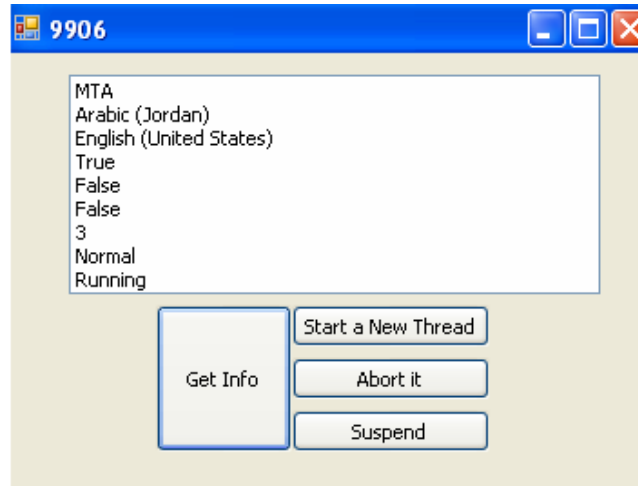
**VB.NET:**

```

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 الـ Server 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 الـ Server حيث سيكون منشغل باستقبال التعليمات التي تم 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 الـ Thread لأن عدم استخدام الـ Thread سيؤدي إلى توقف البرنامج عن الاستجابة.

## 2- استخدام الـ Threading في برمجيات الخادم Server :

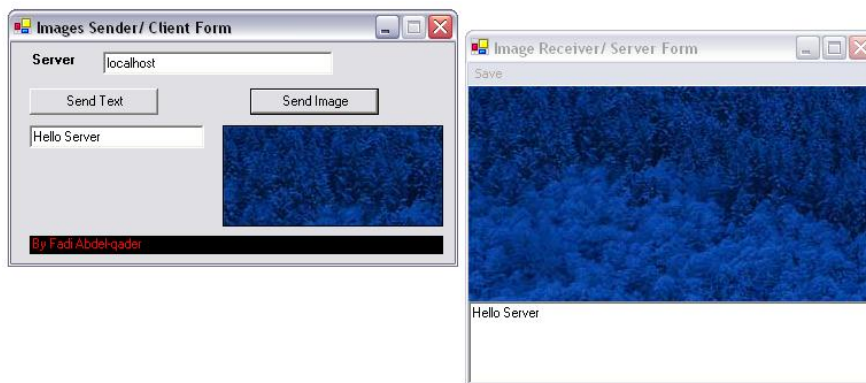
- استخدام دالة القبول AcceptSocket ، لا بد من استخدام Thread منفصل في حالة استخدام دالة القبول AcceptSocket في البرمجة المتزامنة إذ انه سيتوقف البرنامج عن الإستجابة في حالة عدم استخدام الـ Thread مع الـ AcceptSocket حيث توضع عادة في Infinity Loop.

### حالة خاصة:

قد تضطر إلى اللجوء إلى البرمجة غير المتزامنة في حالة إذا كان الـ Server سيستقبل أكثر من Client Request 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 الـ Server 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 mytcp;
Socket mysocket;
void Image_Receiver()
{
    mytcp = new TcpListener (5000);// Open The Port
    mytcp.Start ();// Start Listening on That Port
    mysocket = mytcp.AcceptSocket ();
    myns = new NetworkStream (mysocket);
    pictureBox1.Image = Image.FromStream(myns);

    mytcp.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 mytcp As TcpListener
Private mysocket As Socket
Private Sub Image_Receiver()
    mytcp = New TcpListener(5000) ' Open The Port

```

```

mytcp1.Start() ' Start Listening on That Port
mysocket = mytcp1.AcceptSocket()
myns = New NetworkStream(mysocket)
pictureBox1.Image = Image.FromStream(myns)
mytcp1.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 الـ Threading Aport وهو إغلاق كل ما له علاقة بالـ Socket مثل الـ TcpListener قبل إغلاق الـ Thread في حدث الـ Closing للـ Form ...

# **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 HttpRequest and HttpResponse classes.
HttpRequest	HTTP-specific implementation of the WebRequest class.
HttpResponse	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

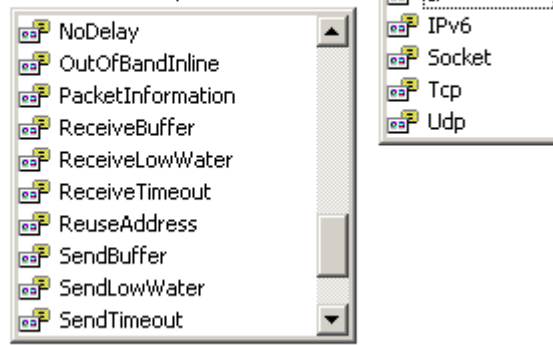
وتركز بشكل اساسي على برمجة الـ Transport Layer وخاصة TCP & UDP Socket Programming ومن اهم الـ Classes التي تدعمها:

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 الـ Transport 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
<b>AcceptConnection</b>	Socket is listening.
<b>AddMembership</b>	Add an IP group membership.
<b>AddSourceMembership</b>	Join a source group.
<b>BlockSource</b>	Block data from a source.
<b>Broadcast</b>	Permit sending broadcast messages on the socket.
<b>BsdUrgent</b>	Use urgent data as defined in RFC-1222. This option can be set only once, and once set, cannot be turned off.
<b>ChecksumCoverage</b>	Set or get UDP checksum coverage.
<b>Debug</b>	Record debugging information.
<b>DontFragment</b>	Do not fragment IP datagrams.
<b>DontLinger</b>	Close socket gracefully without lingering.
<b>DontRoute</b>	Do not route; send directly to interface addresses.
<b>DropMembership</b>	Drop an IP group membership.
<b>DropSourceMembership</b>	Drop a source group.
<b>Error</b>	Get error status and clear.
<b>ExclusiveAddressUse</b>	Enables a socket to be bound for exclusive access.
<b>Expedited</b>	Use expedited data as defined in



	RFC-1222. This option can be set only once, and once set, cannot be turned off.
<b>HeaderIncluded</b>	Indicates application is providing the IP header for outgoing datagrams.
<b>IPOptions</b>	Specifies IP options to be inserted into outgoing datagrams.
<b>IpTimeToLive</b>	Set the IP header time-to-live field.
<b>KeepAlive</b>	Send keep-alives.
<b>Linger</b>	Linger on close if unsent data is present.
<b>MaxConnections</b>	Maximum queue length that can be specified by Listen.
<b>MulticastInterface</b>	Set the interface for outgoing multicast packets.
<b>MulticastLoopback</b>	IP multicast loopback.
<b>MulticastTimeToLive</b>	IP multicast time to live.
<b>NoChecksum</b>	Send UDP datagrams with checksum set to zero.
<b>NoDelay</b>	Disables the Nagle algorithm for send coalescing.
<b>OutOfBandInline</b>	Receives out-of-band data in the normal data stream.
<b>PacketInformation</b>	Return information about received packets.
<b>ReceiveBuffer</b>	Send low water mark.
<b>ReceiveLowWater</b>	Receive low water mark.
<b>ReceiveTimeout</b>	Receive time out.
<b>ReuseAddress</b>	Allows the socket to be bound to an address that is already in use.
<b>SendBuffer</b>	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.
<b>SendLowWater</b>	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.

<b>SendTimeout</b>	Send timeout.
<b>Type</b>	Get socket type.
<b>TypeOfService</b>	Change the IP header type of service field.
<b>UnblockSource</b>	Unblock a previously blocked source.
<b>UseLoopback</b>	Enable Or Disable a Loop Back Connection

#### 4- System.Threading Namespace Classes

يستخدم الـ **Threading** لعمل **Session** منفصلة عن الـ **session** المستخدمة في البرنامج وهو أسلوب آخر للـ **Asynchronous** وتدعم **System.Threading** مجموعة من الـ **Classes** والتي تستخدم في ادارة عمليات الـ **Threading** وهي كما يلي:

<b>Class</b>	<b>Description</b>
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 الـ **Methods** التي تستخدم في ادارة الـ **Threading** 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 الـ Properties الخاصة بالـ Threading :

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.

<b>Public Property Name</b>	<b>Description</b>
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**

<b>Function</b>	<b>Description</b>
<b>lineInitializeEx</b>	Initializes the TAPI line abstraction for use by the invoking application. Synchronous. <b>API Example:</b> public static extern int lineInitialize(ref int hTAPI, int hInst, LineCallBackDelegate fnPtr, ref int szAppName, ref int dwNumLines);
<b>lineShutdown</b>	Shuts down the application's use of TAPI's line abstraction. Synchronous. <b>API Example:</b> [DllImport("Tapi32.dll", SetLastError = true)] public static extern int lineShutdown(int hLineApp);

### **Line Version Negotiation**

Function	Description
<b>lineNegotiateAPIVersion</b>	<p>Allows an application to negotiate a TAPI version to use. Synchronous.</p> <p><b>API Example:</b>  [DllImport("Tapi32.dll", SetLastError = true)]public static extern int  lineNegotiateAPIVersion(int hTAPI, int  dwDeviceID, int dwAPILowVersion, int  dwAPIHighVersion, ref int  lpdwAPIVersion, ref lineextensionid  lpExtensionID);</p>

### Line Status and Capabilities

Function	Description
<b>lineGetDevCaps</b>	<p>Returns the capabilities of a given line device. Synchronous.</p> <p><b>API Example:</b>  [DllImport("Tapi32.dll", SetLastError = true)]public static extern int  lineGetDevCaps(int hLineApp, int  dwDeviceID, int dwAPIVersion, int  dwExtVersion,ref linedevcaps  lpLineDevCaps);</p>
<b>lineGetDevConfig</b>	<p>Returns configuration of a media stream device. Synchronous.</p> <p><b>API Example:</b>  [DllImport("tapi32.dll")] static extern  long lineGetDevConfig(long  dwDeviceID, ref VARSTRING  lpDeviceConfig, string  lpzDeviceClass);</p>
<b>lineGetLineDevStatus</b>	<p>Returns current status of the specified open line device. Synchronous.</p> <p><b>API Example:</b>  [DllImport("tapi32.dll")] static extern  long lineGetLineDevStatus(long hLine,  ref LINEDEVSTATUS  lpLineDevStatus);</p>
<b>lineSetDevConfig</b>	<p>Sets the configuration of the specified media stream device. Synchronous.</p> <p><b>API Example:</b>  [DllImport("tapi32.dll")] static extern  long lineSetDevConfig(long  dwDeviceID, Any lpDeviceConfig,</p>

	long dwSize, string lpszDeviceClass);
<b>lineSetStatusMessages</b>	<p>Specifies the status changes for which the application needs to be notified. Synchronous.</p> <p><b>API Example:</b>  [DllImport("Tapi32.dll", SetLastError = true)]  public static extern int  lineSetStatusMessages(int hLine,int  dwLineStates, int dwAddressStates);</p>
<b>lineGetStatusMessages</b>	<p>Returns the application's current line and address status message settings. Synchronous.</p> <p><b>API Example:</b>  [DllImport("tapi32.dll")]  static extern long  lineGetStatusMessages(long hLine, ref  long lpdwLineStates, ref long  lpdwAddressStates);</p>
<b>lineGetID</b>	<p>Retrieves a device ID associated with the specified open line, address, or call. Synchronous.</p> <p><b>API Example:</b>  [DllImport("tapi32.dll")]  static extern long lineGetID(long hLine,  long dwAddressID, long hCall, long  dwSelect, ref VARSTRING  lpDeviceID, string lpszDeviceClass);</p>
<b>lineGetIcon</b>	<p>Allows an application to retrieve an icon for display to the user. Synchronous</p> <p><b>API Example:</b>  [DllImport("tapi32.dll")]  static extern long lineGetIcon(long  dwDeviceID, string lpszDeviceClass,  ref long lphIcon);.</p>
<b>lineConfigDialog</b>	<p>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.</p> <p><b>API Example:</b>  [DllImport("tapi32.dll")] static extern  long lineConfigDialog(long  dwDeviceID, long hwndOwner, string</p>

	lpszDeviceClass);
<b>lineConfigDialogEdit</b>	<p>Displays a dialog box allowing the user to change configuration information for a line device. Synchronous.</p> <p><b>API Example:</b>  [DllImport("tapi32.dll")] static extern long lineConfigDialogEdit(long dwDeviceID, long hwndOwner, string lpszDeviceClass, Any lpDeviceConfigIn, long dwSize, ref VARSTRING lpDeviceConfigOut);</p>

### Addresses

Function	Description
<b>lineGetAddressCaps</b>	<p>Returns the telephony capabilities of an address. Synchronous.</p> <p><b>API Example:</b>  [DllImport("tapi32.dll")] static extern long lineGetAddressCaps(long hLineApp, long dwDeviceID, long dwAddressID, long dwAPIVersion, long dwExtVersion, ref LINEADDRESSCAPS lpAddressCaps);</p>
<b>lineGetAddressStatus</b>	<p>Returns current status of a specified address. Synchronous.</p> <p><b>API Example:</b>  [DllImport("tapi32.dll")] static extern long lineGetAddressStatus(long hLine, long dwAddressID, ref LINEADDRESSSTATUS lpAddressStatus);</p>
<b>lineGetAddressID</b>	<p>Retrieves the address ID of an address specified using an alternate format. Synchronous.</p> <p><b>API Example:</b>  [DllImport("tapi32.dll")] static extern long lineGetAddressID(long hLine, ref long lpdwAddressID, long dwAddressMode, string lpsAddress, long dwSize);</p>

### Opening and Closing Line Devices

Function	Description
<b>lineOpen</b>	Opens a specified line device for providing subsequent monitoring and/or control of the line.



	<p>Synchronous.</p> <p><b>API Example:</b></p> <pre>[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);</pre>
<b>lineClose</b>	<p>Closes a specified opened line device. Synchronous.</p> <p><b>API Example:</b></p> <pre>[DllImport("tapi32.dll")] static extern long lineClose(long hLine);</pre>

### Address Formats

Function	Description
<b>lineTranslateAddress</b>	<p>Translates between an address in canonical format and an address in dialable format. Synchronous.</p> <p><b>API Example:</b></p> <pre>[DllImport("tapi32.dll")] static extern long lineTranslateAddress(long hLineApp, long dwDeviceID, long dwAPIVersion, string lpszAddressIn, long dwCard, long dwTranslateOptions, ref LINETRANSLATEOUTPUT lpTranslateOutput);</pre>
<b>lineSetCurrentLocation</b>	<p>Sets the location used as the context for address translation. Synchronous.</p> <p><b>API Example:</b></p> <pre>[DllImport("tapi32.dll")] static extern long lineSetCurrentLocation(long hLineApp, long dwLocation);</pre>
<b>lineSetTollList</b>	<p>Manipulates the toll list. Synchronous.</p> <p><b>API Example:</b></p> <pre>[DllImport("tapi32.dll")] static extern long lineSetTollList(long hLineApp, long dwDeviceID, string</pre>

	lpszAddressIn, long dwTollListOption);
<b>lineGetTranslateCaps</b>	Returns address translation capabilities. Synchronous. <b>API Example:</b> [DllImport("tapi32.dll")] static extern long lineGetTranslateCaps(long hLineApp, long dwAPIVersion, ref LINETRANSLATECAPS lpTranslateCaps);

### Call States and Events

Function	Description
<b>lineGetCallInfo</b>	Returns fixed information about a call. Synchronous. <b>API Example:</b> [DllImport("tapi32.dll")] static extern long lineGetCallInfo(long hCall, ref LINECALLINFO lpCallInfo);
<b>lineGetCallStatus</b>	Returns complete call status information for the specified call. Synchronous. <b>API Example:</b> [DllImport("tapi32.dll")] static extern long lineGetCallStatus(long hCall, ref LINECALLSTATUS lpCallStatus);
<b>lineSetAppSpecific</b>	Sets the application-specific field of a call's information structure. Synchronous. <b>API Example:</b> [DllImport("tapi32.dll")] static extern long lineSetAppSpecific(long hCall, long dwAppSpecific);

### Making Calls

Function	Description
<b>lineMakeCall</b>	Makes an outbound call and returns a call handle for it. Asynchronous. <b>API Example:</b> [DllImport("tapi32.dll")] static extern long lineMakeCall(long hLine, ref

	long lphCall, string lpszDestAddress, long dwCountryCode, ref LINECALLPARAMS lpCallParams);
<b>lineDial</b>	Dials (parts of one or more) dialable addresses. Asynchronous. <b>API Example:</b> [DllImport("tapi32.dll")] static extern long lineDial(long hCall, string lpszDestAddress, long dwCountryCode);

### Answering Incoming Calls

Function	Description
<b>lineAnswer</b>	Answers an incoming call. Asynchronous. <b>API Example:</b> [DllImport("Tapi32.dll", SetLastError = true)] public static extern int lineAnswer(int hCall, ref string lpsUserUserInfo, int dwSize);

### Toll Saver Support

Function	Description
<b>lineSetNumRings</b>	Indicates the number of rings after which incoming calls are to be answered. Synchronous. <b>API Example:</b> [DllImport("tapi32.dll")] static extern long lineSetNumRings(long hLine, long dwAddressID, long dwNumRings);
<b>lineGetNumRings</b>	Returns the minimum number of rings requested with lineSetNumRings. Synchronous. <b>API Example:</b> [DllImport("tapi32.dll")] static extern long lineGetNumRings(long hLine, long dwAddressID, ref long lpdwNumRings);

### Call Privilege Control

Function	Description
<b>lineSetCallPrivilege</b>	Sets the application's privilege to the privilege specified. Synchronous. <b>API Example:</b> [DllImport("tapi32.dll")] static extern long

	<code>lineSetCallPrivilege(long hCall, long dwCallPrivilege);</code>
--	--

## Call Drop Functions

Function	Description
<b>lineDrop</b>	Disconnects a call, or abandons a call attempt in progress. Asynchronous. <b>API Example:</b> [DllImport("Tapi32.dll", SetLastError = true)] public static extern int lineDrop(int hCall, string lpsUserUserInfo, int dwSize);
<b>lineDeallocateCall</b>	Deallocates the specified call handle. Synchronous. <b>API Example:</b> [DllImport("tapi32.dll")] static extern long lineDeallocateCall(long hCall);

## Call Handle Manipulation

Function	Description
<b>lineHandoff</b>	Hands off call ownership and/or changes an application's privileges to a call. Synchronous. <b>API Example:</b> [DllImport("tapi32.dll")] static extern long lineHandoff(long hCall, string lpzFileName, long dwMediaMode);
<b>lineGetNewCalls</b>	Returns call handles to calls on a specified line or address for which the application does not yet have handles. Synchronous. <b>API Example:</b> [DllImport("tapi32.dll")] static extern long lineGetNewCalls(long hLine, long dwAddressID, long dwSelect, ref LINECALLLIST lpCallList);
<b>lineGetConfRelated Calls</b>	Returns a list of call handles that are part of the same conference call as the call specified as a parameter. Synchronous. <b>API Example:</b> [DllImport("tapi32.dll")] static extern long lineGetConfRelatedCalls(long hCall, ref LINECALLLIST lpCallList);

### Location and Country/Region Information

Function	Description
<b>lineTranslateDialog</b>	Displays a dialog box allowing the user to change location and calling card information. Synchronous. <b>API Example:</b> [DllImport("tapi32.dll")] static extern long lineTranslateDialog(long hLineApp, long dwDeviceID, long dwAPIVersion, long hwndOwner, string lpszAddressIn);
<b>lineGetCountry</b>	Retrieves dialing rules and other information about a given country/region. Synchronous. <b>API Example:</b> [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
<b>lineRegisterRequestRecipient</b>	Registers or deregisters the application as a request recipient for the specified request mode. Synchronous. <b>API Example:</b> [DllImport("tapi32.dll")] static extern long lineRegisterRequestRecipient(long hLineApp, long dwRegistrationInstance, long dwRequestMode, long bEnable);
<b>lineGetRequest</b>	Gets the next request from the Telephony dynamic link library. Synchronous. <b>API Example:</b> [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
<b>allowAutoRedirect</b>	Client	A Boolean value that determines whether the client will handle HTTP redirects.
<b>bindTo</b>	Server	A string representing an IP address for the server to bind to.
<b>clientConnectionLimit</b>	Client	An integer defining how many concurrent connections can be opened to the server.
<b>connectionGroupName</b>	Client	A string representing a group name if the <code>unsafeAuthenticatedConnectionSharing</code> property is set.
<b>credentials</b>	Client	An <i>ICredentials</i> instance that allows a client to pass credentials for password- based authentication schemes such as Kerberos authentication.
<b>exclusiveAddressUse</b>	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.
<b>listen</b>	Server	A Boolean value that determines whether to allow activation to hook into the outside listener service.
<b>name</b>	Client and server	A string representing the name of the channel.
<b>port</b>	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.
<b>priority</b>	Client and server	An integer value that defines the priority of the channel when multiple channels are available.
<b>proxyName</b>	Client	A string identifying a proxy if the client communication must pass through a proxy.
<b>proxyPort</b>	Client	An integer identifying the port on

Member Name	Availability	Description
		which a proxy is listening for communication from a client.
<b>suppressChannelData</b>	Server	A Boolean value that prevents the server channel from returning channel properties.
<b>timeout</b>	Client	An integer that specifies how long the client channel will wait for a server response.
<b>unsafeAuthenticatedConnectionSharing</b>	Client	A Boolean value that indicates the client will supply credentials and a group name for the connection.
<b>useAuthenticatedConnectionSharing</b>	Client	A Boolean value that tells the server to reuse a connection from an authenticated user.
<b>useIpAddress</b>	Server	A Boolean value that forces the channel to use a specific IP address instead of using the <i>machineName</i> property.
<b>machineName</b>	Client and server	A string representing the machine name of the listening server.

#### ثانياً الـ TCP Channel ومن أهم الـ Members الخاصة بها:

Member Name	Availability	Description
<b>bindTo</b>	Server	A string representing an IP address for the server to bind to.
<b>exclusiveAddressUse</b>	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.
<b>machineName</b>	Client and server	A string representing the machine name of the listening server.
<b>Name</b>	Client and server	A string representing the name
<b>Port</b>	Server	An integer value that specifies what TCP port the server will listen on or what port the client will send TCP packets from.
<b>Priority</b>	Client and server	An integer value that defines the priority of the channel when multiple channels are available. A larger value indicates greater priority.
<b>rejectRemoteR</b>	Server	A Boolean value that determines



Member Name	Availability	Description
<b>equests</b>		whether the server will only accept requests from the local host.
<b>suppressChannelData</b>	Server	A Boolean value that prevents the server channel from returning channel properties.
<b>useIpAddress</b>	Server	A Boolean value that forces the channel to use a specific IP address rather than the <i>machineName</i> 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	<b>NUL</b> (null)	32	20	040	&#32;	<b>Space</b>
1	1	001	<b>SOH</b> (start of heading)	33	21	041	&#33;	<b>!</b>
2	2	002	<b>STX</b> (start of text)	34	22	042	&#34;	<b>"</b>
3	3	003	<b>ETX</b> (end of text)	35	23	043	&#35;	<b>#</b>
4	4	004	<b>EOT</b> (end of transmission)	36	24	044	&#36;	<b>\$</b>
5	5	005	<b>ENQ</b> (enquiry)	37	25	045	&#37;	<b>%</b>
6	6	006	<b>ACK</b> (acknowledge)	38	26	046	&#38;	<b>&amp;</b>
7	7	007	<b>BEL</b> (bell)	39	27	047	&#39;	<b>'</b>
8	8	010	<b>BS</b> (backspace)	40	28	050	&#40;	<b>(</b>
9	9	011	<b>TAB</b> (horizontal tab)	41	29	051	&#41;	<b>)</b>
10	A	012	<b>LF</b> (NL line feed, new line)	42	2A	052	&#42;	<b>*</b>
11	B	013	<b>VT</b> (vertical tab)	43	2B	053	&#43;	<b>+</b>
12	C	014	<b>FF</b> (NP form feed, new page)	44	2C	054	&#44;	<b>,</b>
13	D	015	<b>CR</b> (carriage return)	45	2D	055	&#45;	<b>-</b>
14	E	016	<b>SO</b> (shift out)	46	2E	056	&#46;	<b>.</b>
15	F	017	<b>SI</b> (shift in)	47	2F	057	&#47;	<b>/</b>
16	10	020	<b>DLE</b> (data link escape)	48	30	060	&#48;	<b>0</b>
17	11	021	<b>DC1</b> (device control 1)	49	31	061	&#49;	<b>1</b>
18	12	022	<b>DC2</b> (device control 2)	50	32	062	&#50;	<b>2</b>
19	13	023	<b>DC3</b> (device control 3)	51	33	063	&#51;	<b>3</b>
20	14	024	<b>DC4</b> (device control 4)	52	34	064	&#52;	<b>4</b>
21	15	025	<b>NAK</b> (negative acknowledge)	53	35	065	&#53;	<b>5</b>
22	16	026	<b>SYN</b> (synchronous idle)	54	36	066	&#54;	<b>6</b>
23	17	027	<b>ETB</b> (end of trans. block)	55	37	067	&#55;	<b>7</b>
24	18	030	<b>CAN</b> (cancel)	56	38	070	&#56;	<b>8</b>
25	19	031	<b>EM</b> (end of medium)	57	39	071	&#57;	<b>9</b>
26	1A	032	<b>SUB</b> (substitute)	58	3A	072	&#58;	<b>:</b>
27	1B	033	<b>ESC</b> (escape)	59	3B	073	&#59;	<b>;</b>
28	1C	034	<b>FS</b> (file separator)	60	3C	074	&#60;	<b>&lt;</b>
29	1D	035	<b>GS</b> (group separator)	61	3D	075	&#61;	<b>=</b>
30	1E	036	<b>RS</b> (record separator)	62	3E	076	&#62;	<b>&gt;</b>
31	1F	037	<b>US</b> (unit separator)	63	3F	077	&#63;	<b>?</b>

**Table 2:**

Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
64	40	100	&#64;	<b>Q</b>	96	60	140	&#96;	`
65	41	101	&#65;	<b>A</b>	97	61	141	&#97;	<b>a</b>
66	42	102	&#66;	<b>B</b>	98	62	142	&#98;	<b>b</b>
67	43	103	&#67;	<b>C</b>	99	63	143	&#99;	<b>c</b>
68	44	104	&#68;	<b>D</b>	100	64	144	&#100;	<b>d</b>
69	45	105	&#69;	<b>E</b>	101	65	145	&#101;	<b>e</b>
70	46	106	&#70;	<b>F</b>	102	66	146	&#102;	<b>f</b>
71	47	107	&#71;	<b>G</b>	103	67	147	&#103;	<b>g</b>
72	48	110	&#72;	<b>H</b>	104	68	150	&#104;	<b>h</b>
73	49	111	&#73;	<b>I</b>	105	69	151	&#105;	<b>i</b>
74	4A	112	&#74;	<b>J</b>	106	6A	152	&#106;	<b>j</b>
75	4B	113	&#75;	<b>K</b>	107	6B	153	&#107;	<b>k</b>
76	4C	114	&#76;	<b>L</b>	108	6C	154	&#108;	<b>l</b>
77	4D	115	&#77;	<b>M</b>	109	6D	155	&#109;	<b>m</b>
78	4E	116	&#78;	<b>N</b>	110	6E	156	&#110;	<b>n</b>
79	4F	117	&#79;	<b>O</b>	111	6F	157	&#111;	<b>o</b>
80	50	120	&#80;	<b>P</b>	112	70	160	&#112;	<b>p</b>
81	51	121	&#81;	<b>Q</b>	113	71	161	&#113;	<b>q</b>
82	52	122	&#82;	<b>R</b>	114	72	162	&#114;	<b>r</b>
83	53	123	&#83;	<b>S</b>	115	73	163	&#115;	<b>s</b>
84	54	124	&#84;	<b>T</b>	116	74	164	&#116;	<b>t</b>
85	55	125	&#85;	<b>U</b>	117	75	165	&#117;	<b>u</b>
86	56	126	&#86;	<b>V</b>	118	76	166	&#118;	<b>v</b>
87	57	127	&#87;	<b>W</b>	119	77	167	&#119;	<b>w</b>
88	58	130	&#88;	<b>X</b>	120	78	170	&#120;	<b>x</b>
89	59	131	&#89;	<b>Y</b>	121	79	171	&#121;	<b>y</b>
90	5A	132	&#90;	<b>Z</b>	122	7A	172	&#122;	<b>z</b>
91	5B	133	&#91;	<b>[</b>	123	7B	173	&#123;	<b>{</b>
92	5C	134	&#92;	<b>\</b>	124	7C	174	&#124;	<b> </b>
93	5D	135	&#93;	<b>]</b>	125	7D	175	&#125;	<b>}</b>
94	5E	136	&#94;	<b>^</b>	126	7E	176	&#126;	<b>~</b>
95	5F	137	&#95;	<b>_</b>	127	7F	177	&#127;	<b>DEL</b>

## 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](http://www.SocketCoder.Com)
- All API Functions <http://www.pinvoke.net>
- Microsoft MSDN, <http://msdn.microsoft.com>
- C# Corner, [www.c-sharpcorner.com](http://www.c-sharpcorner.com)
- Code Project [www.codeproject.com](http://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 الـ RTP بالإضافة إلى برمجة أنظمة المؤتمرات باستخدام Silverlight. رابط الإصدار الجديد على الموقع الرسمي:

<http://www.socketcoder.com/ArticleFile.aspx?index=2&ArticleID=67>

\*\*\*\*\*

تم بحمد الله

\*\*\*\*\*