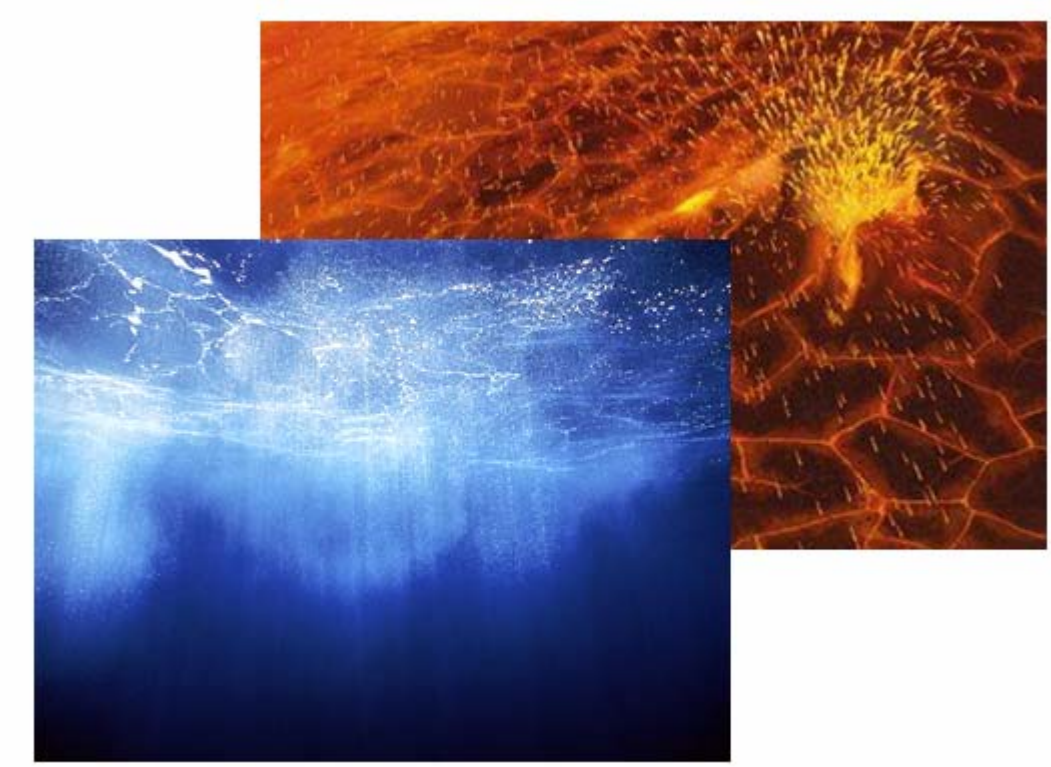
Dot Net Networks & TCP/IP Programming

Version 2

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



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

النسخة الورقية هي النسخة المعتمدة من الكتاب مع إضافة الكثير من الدروس التعليمية الجديدة ...

الكتاب الأول والمتخصص في هذا المجال فقط باللغة العربية — النسخة الإلكترونية—

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

لطلب أ والاستفسار أو التوزيع يرجى الاتصال على احد لعناوين التالية

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

تأليف فادي عبد القادر – الأردن

Version 2.0, 2006

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 للغات الدوت نيت كما ويحتوي على شرح مفصل عن Socket Programming والـTransport Layer Protocols و الـ 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 وبرمجيات الـ Remote Desktop وأنظمة التحكم عن بعد وغيرها ، كما ويحتوي على شرح مفصل لأهم برتوكولات الـ Application Layer واستخداماتها في برمجيات الشبكات ، وأخيرا شرح مفصل عن طرق الحماية ووضع الصلاحيات والسياسات في برمجيات الشبكات بالإضافة إلى طرق تحليل والتصنت على الـ Packets المرسلة باستخدام الـ Row Programming والـ Packet Sniffer ...

الإهداء:

كان الهدف من تأليف الكتاب وجود مرجع لكل الطلاب و المبرمجين العرب فيما يتعلق بالـ Network Programming في بيئة الدوت نيت لذلك اهدي هذا الكتاب إلى كل الطلاب والمبرمجين العرب في جميع أنحاء العالم ...

فادي عبدالقادر
April – 2006

الأمور التي سيتم إضافتها إلى الكتاب النسخة الورقية **(النسخة الغير المجانية):**

أولا : في **Voice Over IP Programming** سيتم التطرق إلى معايير الـ **VOIP** مثل الـ **SIP** والـ **H.323** بالإضافة إلى الـ **SCTP Transport Protocol** .

ثانيا : **Advanced Web Services Programming & Remotting Services.**

ثالثا : **ARP,RARP,ICMP & Advanced Row & Packet Sniffering Applications Programming.**

رابعا : **Remote Desktop Applications & Controls.**

خامسا : إنشاء أنظمة المراقبة عبر الكاميرا والاتصال عبر **TAPI Telephony** في حالة ورود تغيير في الصور الملتقطة باستخدام الـ **Image Processing.**

سادسا : شرح مفصل عن تكنولوجيا الـ **Multithreading** واستخدامها في برمجيات الشبكات.

سابعا : شرح مفصل عن **Network Security Programming** ومعاير التشفير **DES** والـ**RSA** والـ**DSA** والـ **SSL** واستخدامها في برمجيات الشبكات.

ثامنا : شرح عن طرق بناء برمجيات الـ **Keyboard Listening** والتصنت على حركة الـ**Mouse** ونقل المعلومات عبر الشبكة باستخدام الـ **Hooks Programming** والـ **Network Programming.**

تاسعا : شرح موسع عن الـ **Application Layer Protocols** واستخدامها في بيئة الدوت نيت.

عاشرا : مجموعة اكبر من المشاريع والأمثلة العملية **Real Applications** على كل فصول الكتاب.

ملخص الفصول:

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

Part2 Streaming:
Chapter 4: Streaming (Classes & Members)
Chapter 5: Applied Streaming in Dot Net **(+Advanced Applications in Full Version Book)**

Part3 Transport & Network Layer Programming:
Chapter 6: Transport TCP & UDP (Classes & Members)
Chapter 7: Synchronous Sockets Programming
Chapter 8: Asynchronous Sockets Programming
Chapter 9: Advanced Multicasting Systems (Architecture & Conference Systems)
Chapter 10 VOIP - Voice Over IP Programming **(+Advanced VOIP Programming in Full Version Book)**

Part4 Application Layer Programming:
Chapter 11 DNS Programming
Chapter 12 HTTP Programming
Chapter 13 Web Services & XML Programming **(+Advanced Remotting in Full Version Book)**
Chapter 14 SMTP & POP3 Programming **(+ Create a Program Similar Like Microsoft Outlook in Full Version Book)**

Chapter 15 FTP Programming

Part5 Network Security Programming:

Chapter 16 Cryptography **(+ More Advanced Topics Just in Full Version Book)**

Chapter 17 Socket Permissions
Chapter 18 Packet Sniffer & Row Programming (**Just in Full Version Book**)

Part6 Multithreading
Chapter 19 Multithreading (Using & Managing) (**Just in Full Version Book**)

Appendixes: Most Important Dot Net Namespaces For Network Applications

فهرس المحتويات في النسخة الإلكترونية:

Part1
Networks & TCP/IP Programming 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

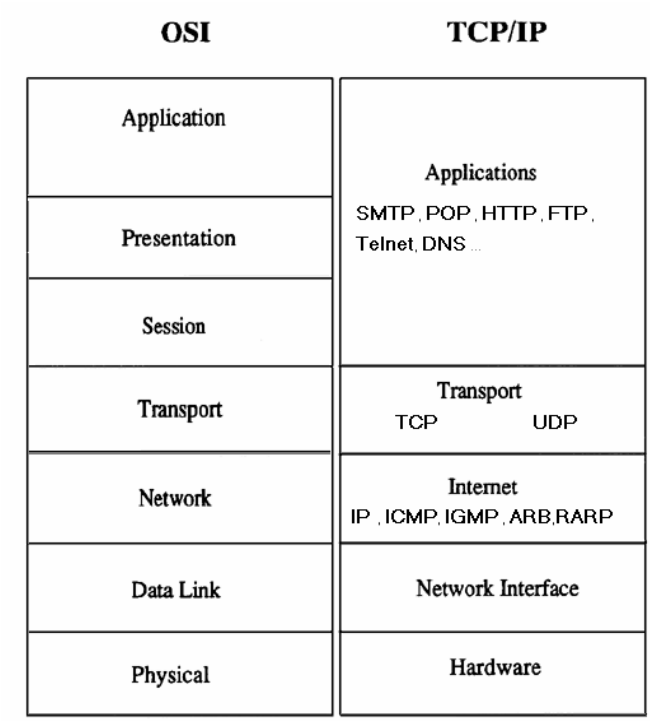
- Using UDP Connectionless in Dot Net to Send Broadcast Messages

- Streaming & Threading Overview & Using it to Send Images Throw Network

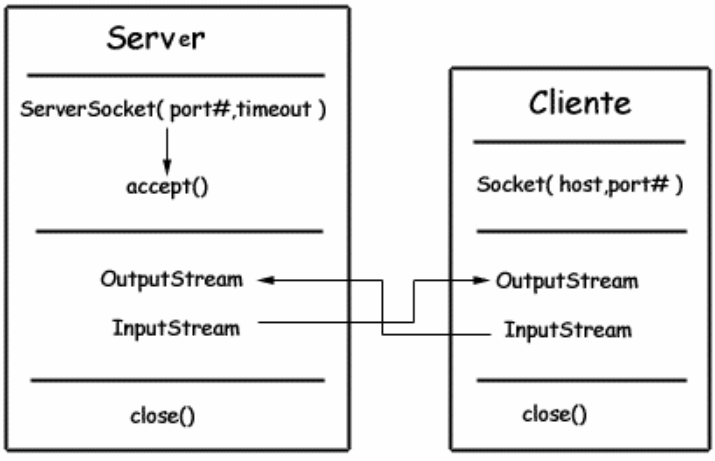
بسم الله الرحمن الرحيم

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 OSIلعملية الاتصال والتي تمر بسبعة طبقات لكل طبقة منها وظيفة معينة يتم إضافتها كـ Headers على البيانات المرسلة.وتم اختصارها إلى خمسة طبقات في برتوكول TCP/IP وتبين الصورة المرفقة هذه الطبقات:



تبدأ عملية توليف الرسالة المرسلة في الـ Application Layer ووظيفتها هنا التعامل مع الرسالة نفسها وتحويلها من صيغة نصية إلى Data يمكن إرسالها عبر الشبكة ، ففي برمجيات الدردشة Chat يتم تحويل النص المكتوب إلى ASCII Code ثم إلى مجموعة من Binary Code أو الـ Bits توضع في مصفوفة لتجهيزها وإرسالها عبر Socket والذي يربط طبقة الـ Transport Layer مع الـ Network Layer أو الـ Internet Layer و يوضح الشكل التالي طبيعة عمل الـSocket :



حيث يربط رقم الـPort المحدد في Transport Layer مع Destination IP في Network Layer ويقوم الـServer بالطرف المقابل بالموافقة على طلب الـClient وتقتصر وظيفة الـSocket في الـServer على ربط رقم الـPort مع الـ Socket Option التي يتم تحديدها ثم

البدء بعملية التصنت على الـPort الذي تم تحديده ، ويمكن في هذه المرحلة وضع شروط معينة لقبول الجلسة مثل عمليات التحقق الـ Authentication أو ما شابه أو الموافقة بشكل مباشر.

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

Application لتعامل مع البرنامج نفسه أو ما يسمى User Interface

Presentation تمثيل البيانات المرسلة وهي كما ظهرت سابقا بتحويل البيانات إلى الـASCII أو استخدام أساليب لضغط البيانات أو تشفيرها ، في الدوت نيت تتم عملية تمثيل الرسالة بالـBinary باستخدام الـASCIIEncoding Class كما يلي:

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 و التعريف ببعضهم البعض(فتح الجلسة) ...

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 و الـ Port المحدد وذلك بعد تحديد عملية الاتصال سواء عبر UDP أو عبر TCP.
2- التعريف بنفسه وعمل الـ Authentication إذا تطلب الـServer ذلك
3- قبول أو رفض الجلسة ويتم ذلك بإرسال الموافقة على فتح الجلسة أو رفضها
4- بدأ الجلسة وقيام الـServer بعمل Listening على الـPort الخاص بالبرنامج
وهنا مثال يوضح عمل هذه الطبقة باستخدام ال 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 Client بتحميل الرسالة إلى الطبقة الأخرى وهي هنا طبقة 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 وPort الاتصال أما الـ UDP فيمكن أن يكون الطرف المستقبل كل الأجهزة Broadcast وهذا يعني أن أي شخص يقوم بتصنت عبر هذا الـ Port يستطيع استقبال الرسالة ، كما يمكن من عمل الـMulticasting ، ويتم ذلك بوضع الـBroadcast IP أو الـMulticast IP مع رقم الـPort في الـSocket ...

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

C#:

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

VB.NET:

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

وبعد ذلك تسلم إلى Network Layer إذ تتم عنونة الرسالة ووضع عنوان المرسل والمستقبل عليها وتسلم إلى الطبقة الأدنى ليتم إرسالها عبر ال Physical Tunnel ومن الأمثلة عليها IP,IPv6,ARB-Address Resolution Protocol ...
أما بنسبة للجهاز المستقبل ال Server فيقوم بالمرور على نفس الطبقات ولكن بالعكس حيث يستلم كرت الشبكة ال Bits لتحول إلى Data link ثم Network ثم Transport ثم Application ومنها تحول من Binary إلى ASCII ومن ASCII إلى Text .. وهذا الكود يوضح مبدأ عمل ال Server :

C#:

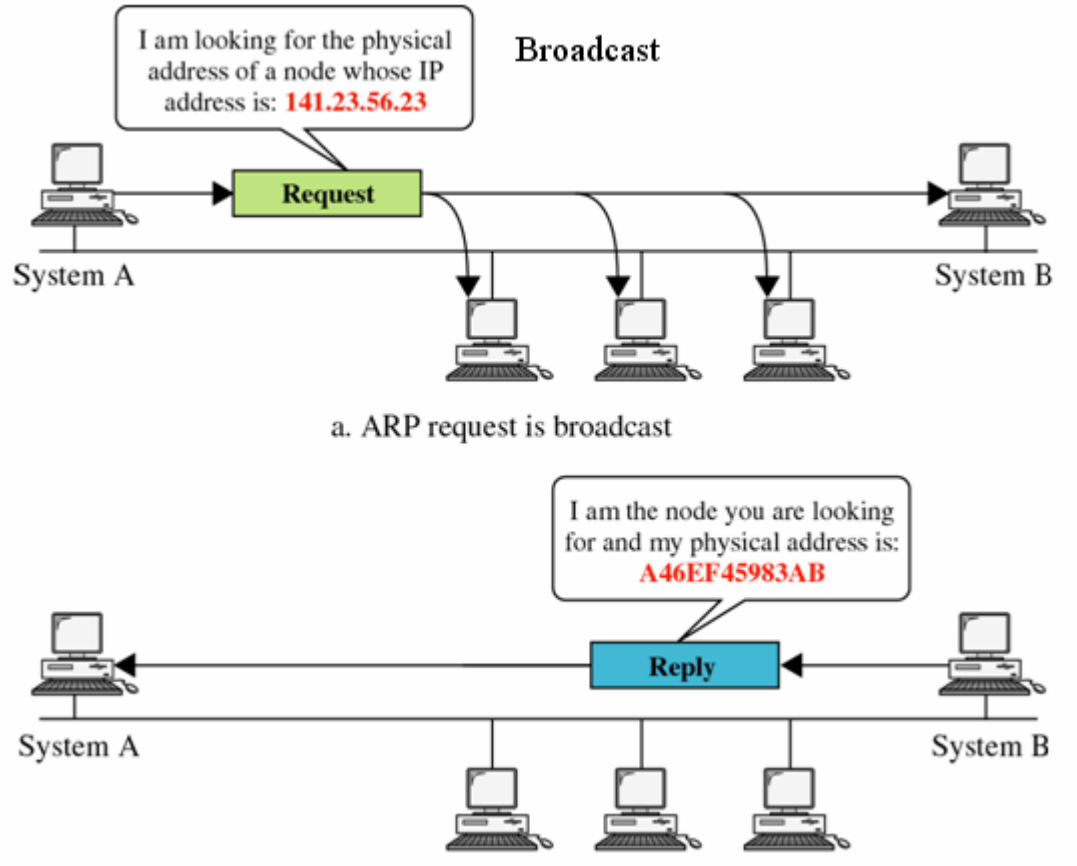
```
TcpListener myList=new TcpListener(5020);
myList.Start();
Socket s=myList.AcceptSocket();
NetworkStream myns = new NetworkStream (mysocket);
StreamReader mysr = new StreamReader (myns);
Console.Write (mysr.ReadLine());
s.Close();
```

VB.NET:

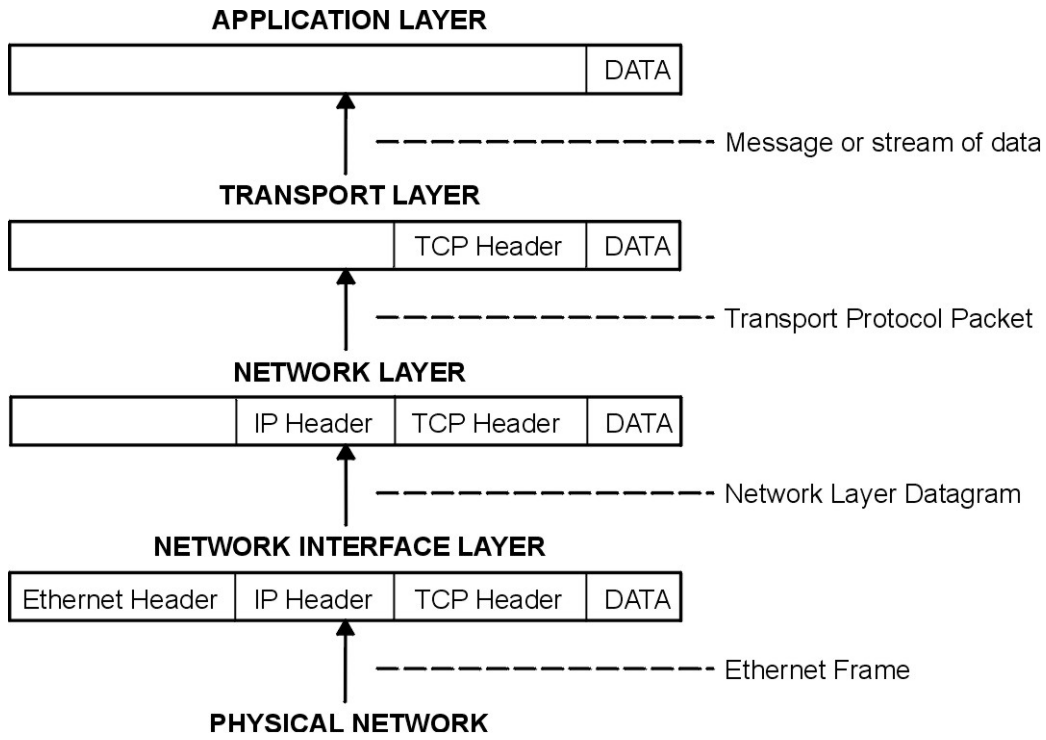
```
Dim myList As TcpListener = New TcpListener(5020)
myList.Start()
Dim s As Socket = myList.AcceptSocket()
Dim myns As NetworkStream = New NetworkStream(mysocket)
Dim mysr As StreamReader = New StreamReader(myns)
Console.Write (mysr.ReadLine())
s.Close()
```

بعد إجراء عملية العنونة يقوم المرسل بسؤال عن عنوان الـMAC Address الخاص بالـServer وتتم هذه العملية عبر بروتوكول الـARP-Address Resolution Protocol ويقوم هذا البروتوكول بالتحقق من وجود الـMAC Address في الـMAC Table وفي حالة عدم وجوده يقوم

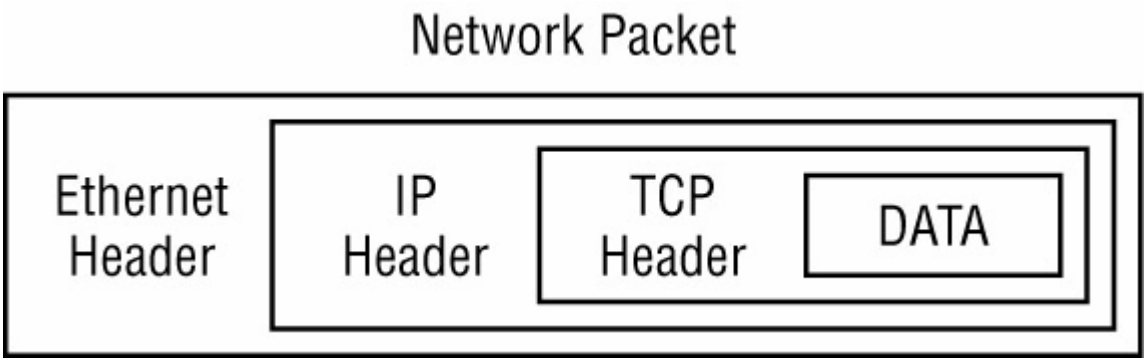
بإرسالARB Broadcast Message Request إلى كل الشبكة يسأل فيها عن صاحب الـ IP Address المراد الإرسال له فإذا وجده يرسل الجهاز المعني ICMP Message يخبره فيها بعنوان الـMAC Address الخاص به وبعد استلام الرسالة يقوم بتخزين العنوان المعني في MAC Table الخاصة به ولكي يتم استخدامه في المرات اللاحقة ، لاحظ الشكل التالي:



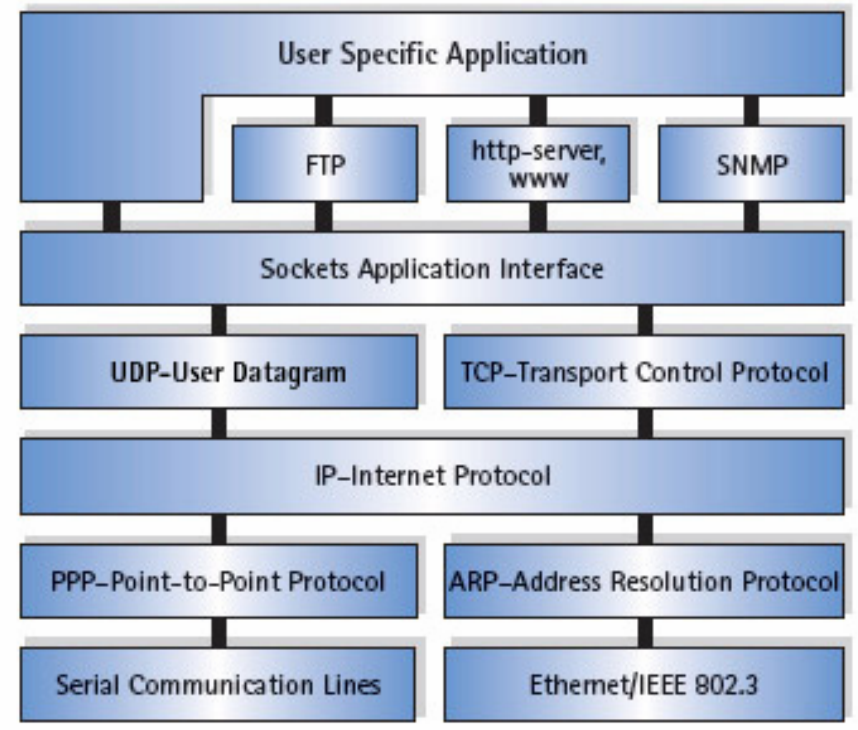
وبعد هذه المرحلة يتم تحديد نوع الـEncapsulation هل سيكون كـInternet Encapsulation أو Ethernet Encapsulation ويتم ذلك في الـData Link Layer والشكل التالي يوضح كل هذه العمليات:



وفي النهاية يكون الشكل العام للـPacket كما يلي:



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

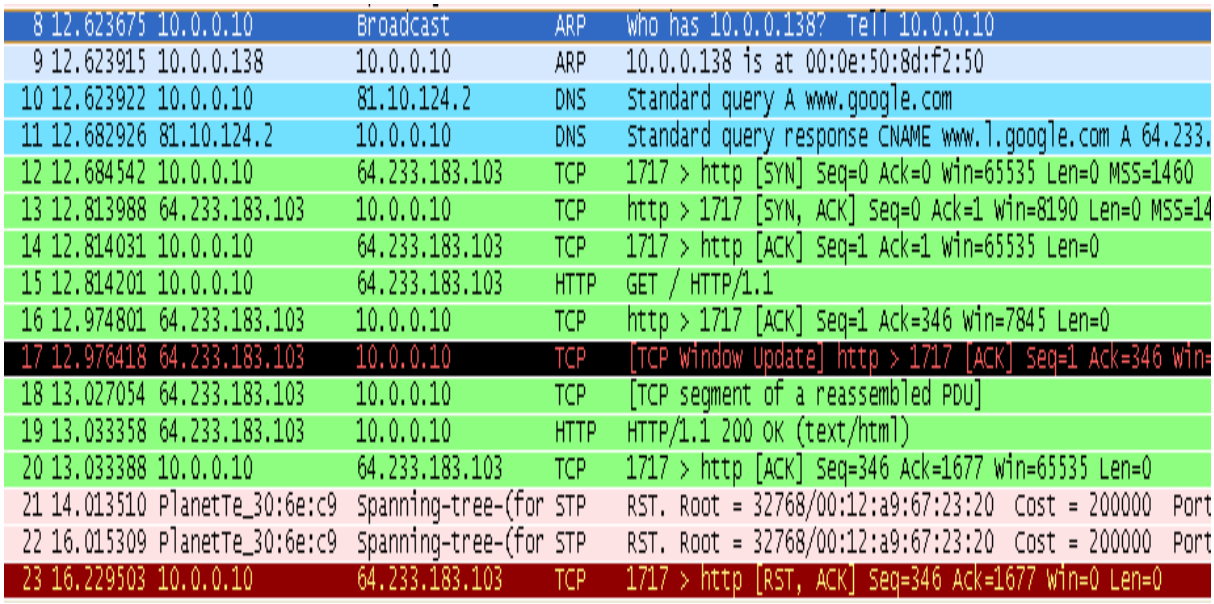


في كل طبقة يتم إضافة Header على الـData ويتراوح حجم الـHeader من 20 إلى 60 بايت حسب الـOptions التي يتم إضافتها إلى الـ Header.
في المثال التالي سنقوم باستخدام برنامج الـEthereal وهو برنامج يقوم بعملية الـ Packet Sniffer والذي يمكن تثبيته من الموقع الخاص به وهو:

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

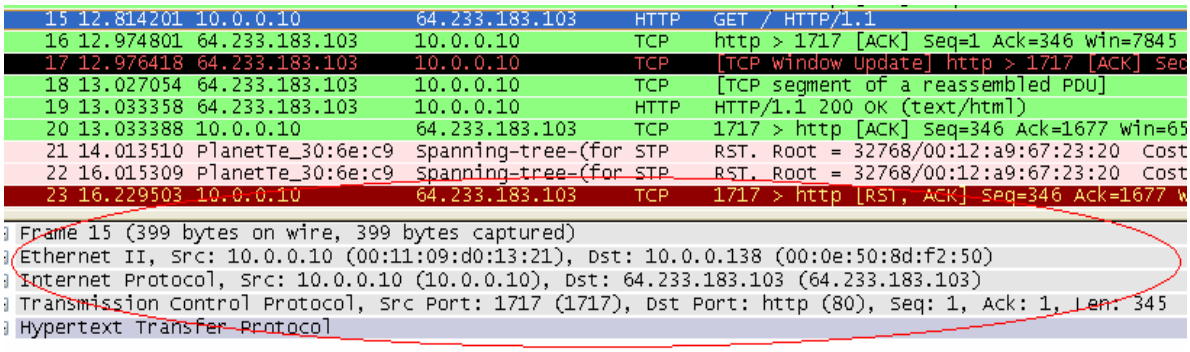
مثال على استخدام الـARB والـDNS والـHTTP :

لاحظ الشكل التالي في برنامج الـEthereal والذي يقوم بتصنت على الـInterface Card حيث قمنا بطلب الدخول على موقع الـGoogle باستخدام الـInternet Explorer :



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

لاحظ أن أول عملية كانت السؤال عن الـMAC Address الخاص بالـGateway وهو هنا 10.0.0.138 حيث أرسلت هذه الرسالة كـBroadcast Message وفي العملية الثانية قام الـRouter بالرد على الـARP Request وأرسل عنوان الـMAC Address الخاص به بعد هذه العملية سيقوم الـClient بلإستفسار عن عنوان الـGoogle بإرسال DNS Request Query لاحظ أنه في المثال قد وجد الـIP Address الخاص بالـGoogle في الـDNS Table والموجودة ضمن نظام التشغيل وهذا يعني انه تم الدخول على الـGoogle من قبل والدليل على ذلك أن الـIP Address الخاص بالـGoogle موجود في الـDNS Table وبعد هذه العملية قام موقع الـGoogle بالرد على الطلب يخبره بوجوده وضمن العنوان المحدد لاحظ أن العمليات من 8 إلى 20 هي عمليات طلب و إرسال لمحتويات الصفحة الخاصة بموقع الـGoogle وتتم باستخدام الـ HTTP Protocol.

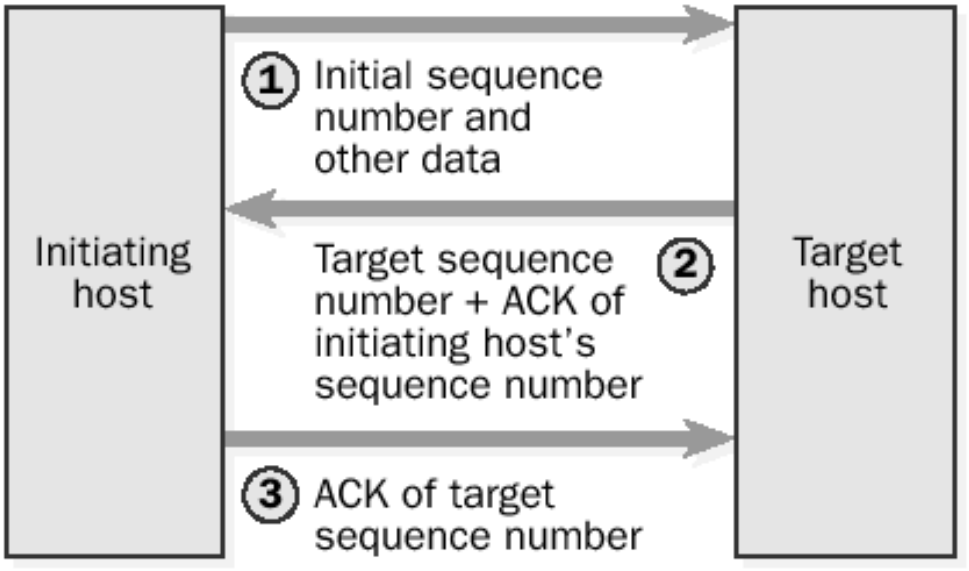


15 12.814201	10.0.0.10	64.233.183.103	HTTP	GET / HTTP/1.1
16 12.974801	64.233.183.103	10.0.0.10	TCP	http > 1717 [ACK] Seq=1 Ack=346 Win=7845
17 12.976418	64.233.183.103	10.0.0.10	TCP	[TCP Window Update] http > 1717 [ACK] Seq
18 13.027054	64.233.183.103	10.0.0.10	TCP	[TCP segment of a reassembled PDU]
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 الأخر وعند الموافقة على إجراء الاتصال يتم البدء بإرسال مجموعة من المعلومات إلى الطرف المستقبل، في بروتوكول الـ TCP تتم هذه العملية بثلاثة مراحل تسمى Three-way hand-shake و كما هو واضح في الشكل التالي:



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

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

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

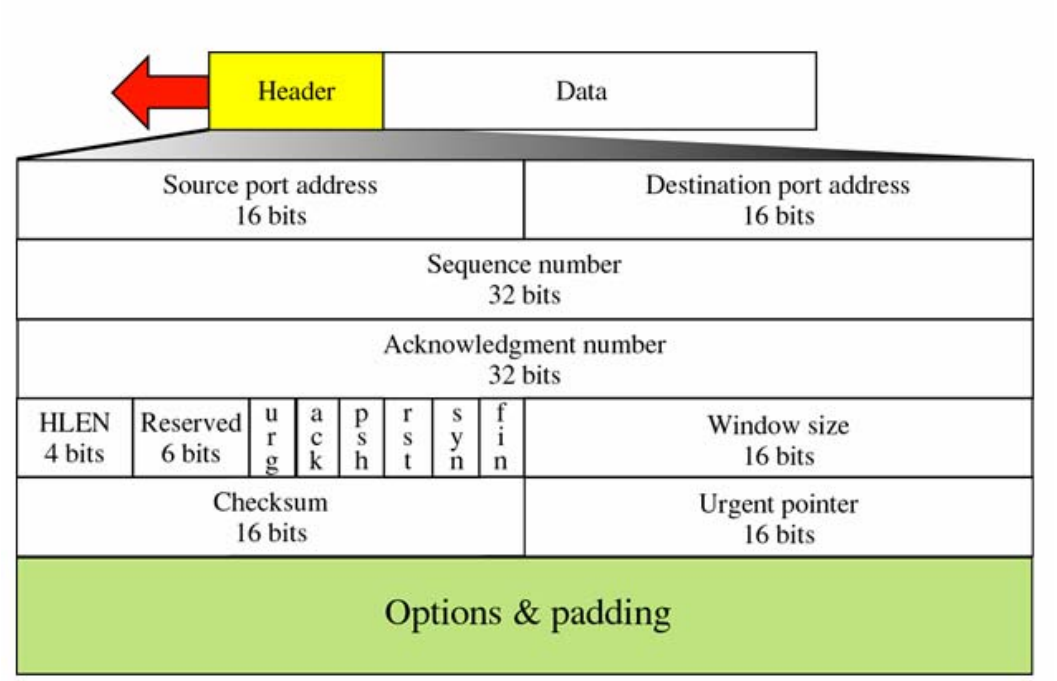
أما في بروتوكول الـ UDP فتتم بدون إرسال Acknowledgments ولا يتحقق الـUDP من عمليات الوصول كما هو الحال في الـTCP وهو ما سيتم توضيحه في الجزء التالي من هذا الفصل.

TCP & UDP Header Encapsulation : 1.3

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

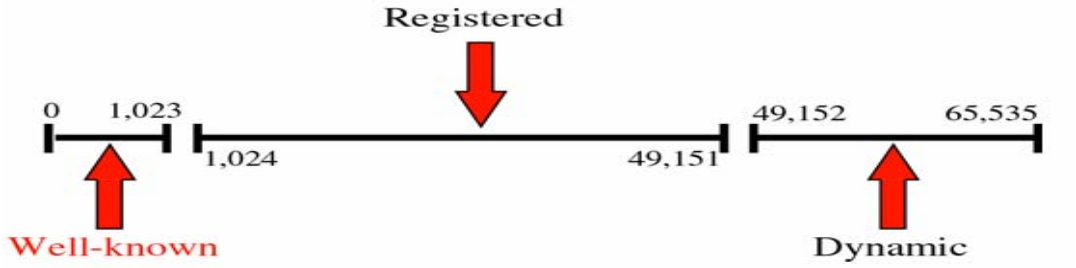
TCP Encapsulation : 1.3.1

سوف نبين في هذا الجزء طبيعة الاتصال باستخدام الـTCP حيث يتميز هذا البروتوكول بكل عمليات التحكم سواء على مستوى Data Flow أو حجم الـBuffer كما يدعم عمليات التحقق من الوصول وفق الترتيب السليم Delivered on Sequence وهذا واضح من تركيب Header الخاصة به أنظر إلى الشكل التالي:



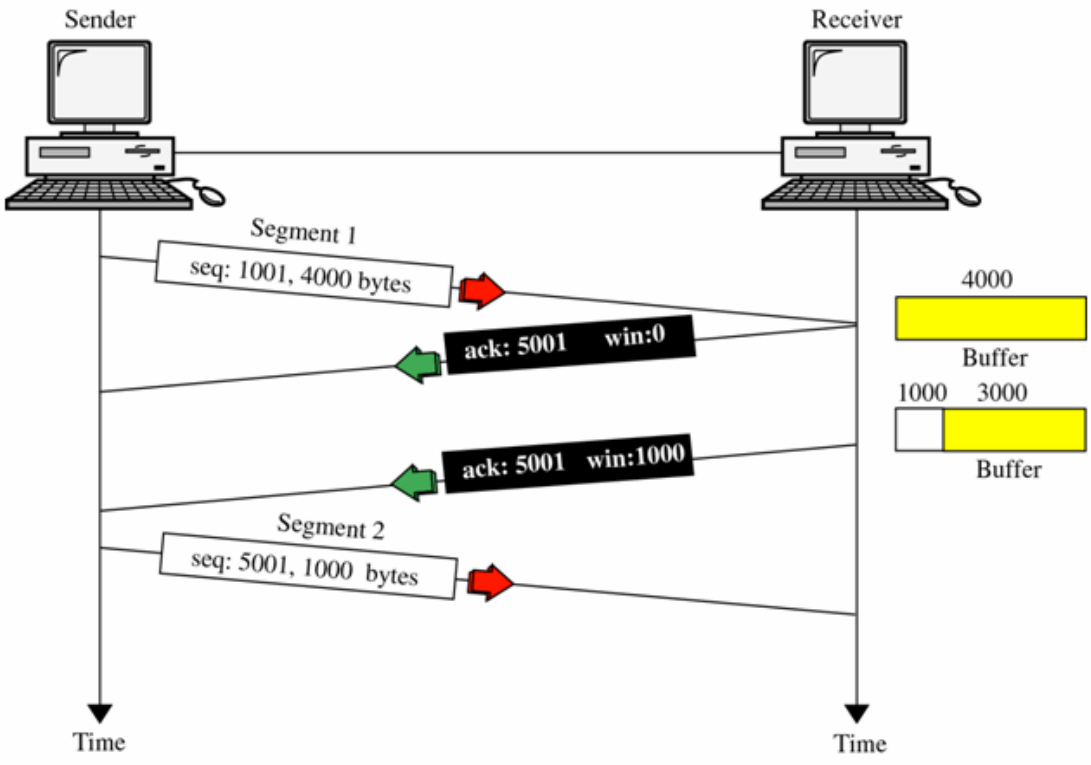
Source & Destination Port 1.3.1.1 :

حيث يتم تحديد الـSource Port والـDestination Port الخاصة بالبرنامج الذي يجري عملية الاتصال ومن المعروف أنه لا يمكن لأكثر من برنامج استخدام نفس رقم الـPort لاكن يمكن للبرنامج الواحد استخدام أكثر من Port Address ، وبتأكيد فإن عملية اختيار الـPort لليست عشوائية حيث انه يجب الابتعاد عن الأرقام التي تبدأ بـ 0 وتنتهي 1023 إذ أنها أرقام لبروتوكولات معروفة ويتم استخدامها في نظام التشغيل ومن الأمثلة عليها بروتوكول الـHTTP والذي يستخدم الـPort 80 وبروتوكول الـ FTP والذي يستخدم الـPort 21 وغيره، ويفضل عند اختيار الـPort أن لا يبدأ برقم يقل عن 49.151 انظر إلى الشكل التالي:



1.3.1.2 : Sequence & Acknowledgment Number

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



1.3.1.3 : Header Length & Validation Controls

ويحتوي الجزء الثاني من الـHeader الخاص بالـTCP على32 Bits مقسمة على الـ Windows Size 16 Bits والـValidation Controls + Header Length 16Bits
وكما هو في الشكل التالي:

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

ويحتوي الـHeader Length على حجم الـHeader الخاص بالـTCP مقسوم على 4 أي لمعرفة حجم الـHeader نضرب الـHLEN بـ4 ، أما الـValidation 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

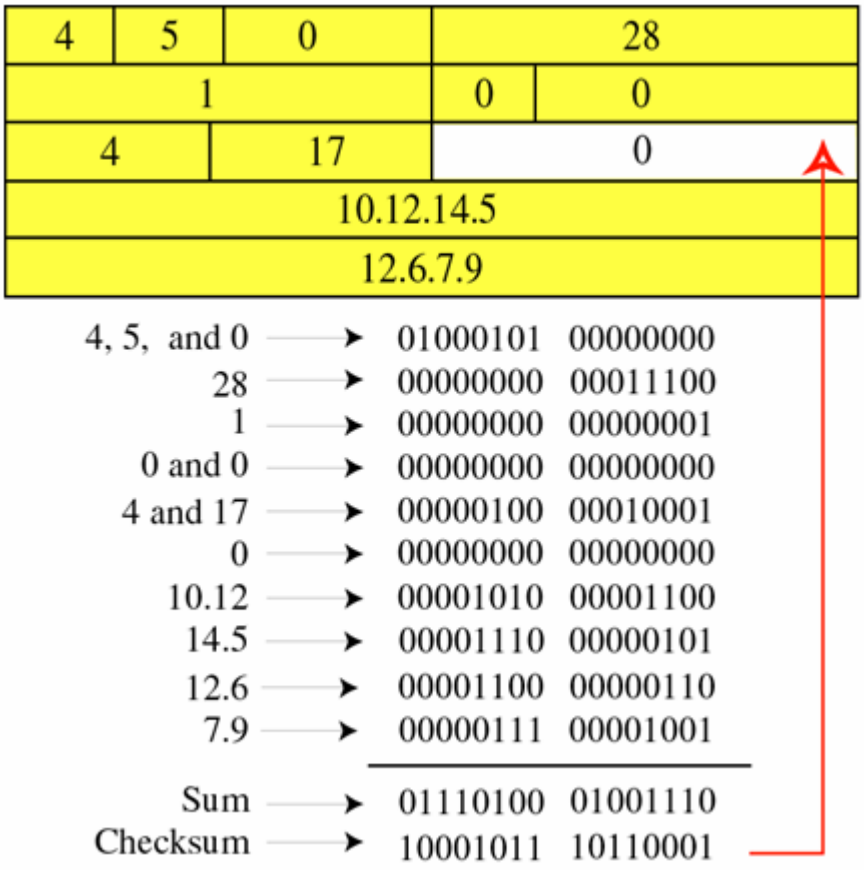
URG	ACK	PSH	RST	SYN	FIN

1.3.1.4 : Window Size

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

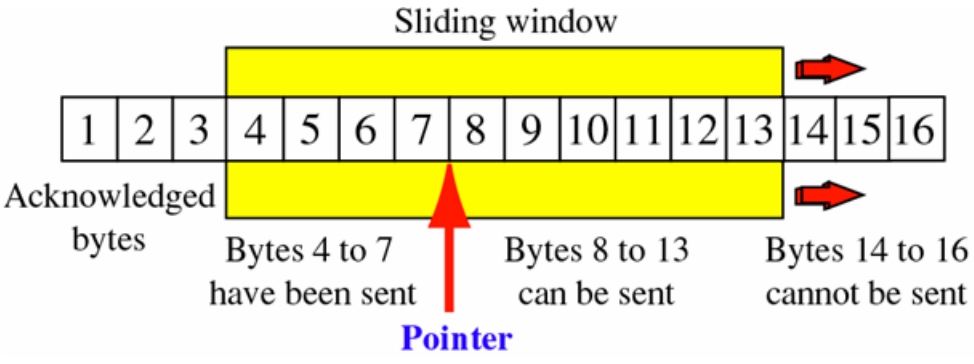
1.3.1.5 : Check Sum

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



1.3.1.6 : Urgent Pointer

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

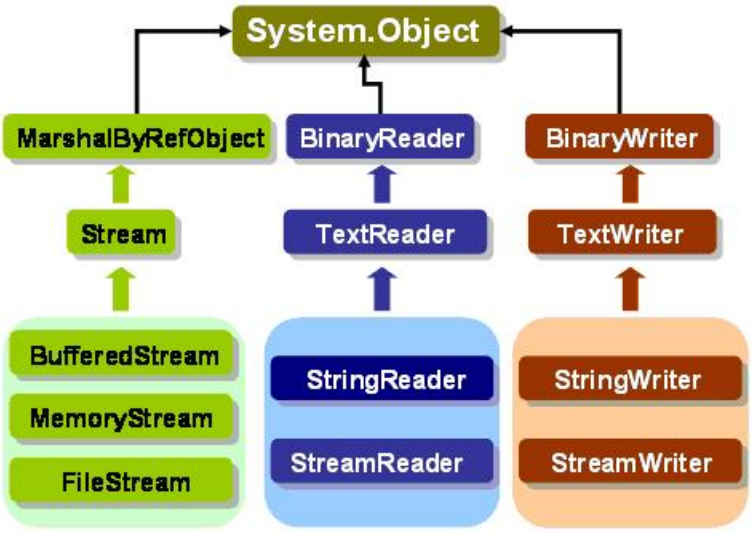


Dot Net Streaming & Threading Overview: 1.3.2

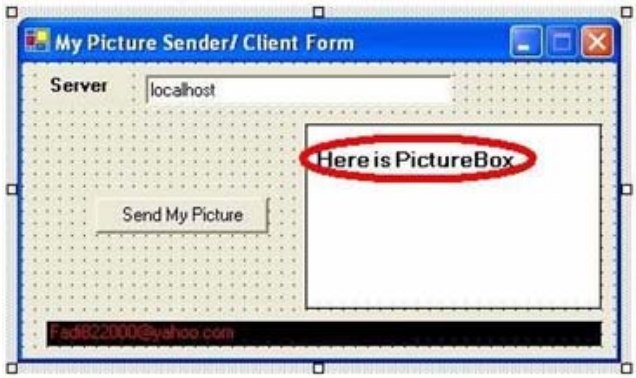
تعرفنا سابقا على أجزاء OSI و TCP/IP وبينا كيفية التعامل مع هذه الطبقات في برمجيات الشبكات ، وفي هذا الجزء سوف نبين كيفية التعامل مع الـ Stream Library لإرسال Binary Data بالإضافة إلى استخدام الـ Thread في برمجيات الشبكة...

أولا : الـ Socket عرفنا الـSocket سابقا على أنها الأداة التي يتم نقل البيانات من خلالها ، ولاستخدامها يلزم في البداية تعريف System.Net.Sockets حيث يحتوي هذا ال Namespaces على عدد كبير من ال Classes والتي يتم استخدامها في برمجيات الشبكة.

يمكن الـStream Classes من نقل Text أو Binary Data ،حيث بينا سابقا كيفية التعامل من الـSocket لنقل Text باستخدام Stream Reader وال Stream Writer وفي هذا الجزء سنبين كيفية التعامل معه لنقل Object (أي نوع أخر من البيانات ويمكن أن يكون صورة Image أو صوت Voice أو أي شيء آخر يمكن أن يحول إلى Binary Data ..)، وكما هو الحال في نقل ال Text كنا نحول Text إلى ASCII Code ثم إلى Binary أما في Object فيتم التعامل معه باستخدام الـBinary Reader و ال Binary Writer والتي تمكنك من التعامل مع أي Object ، كما وتستخدم الـ Stream Reader والـ Stream Writer لتعامل مع الـ Text والـ File Stream لتسهيل التعامل مع الملفات بالإضافة إلى الـ Memory Stream والتي تستخدم كـ Buffer لحفظ البيانات قبل إرسالها أو بعد استقبالها ، وتقوم أيضا بتحويل الـStream إلى مجموعة من الـBits والـBits إلى Stream مرة أخرى لاحظ الشكل التالي والذي يوضح مكتبات الـStream في الدوت نيت:



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



في البداية قم بإضافة ال Namespaces التالية:

C#:

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

VB.NET:

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

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

C#:

```
try
{
openFileDialog1.ShowDialog ();
string mypic_path =openFileDialog1.FileName ;
pictureBox1.Image = Image.FromFile(mypic_path);
MemoryStream ms = new MemoryStream();
pictureBox1.Image.Save(ms,pictureBox1.Image.RawFormat);
byte[] arrImage = ms.GetBuffer();
ms.Close();
TcpClient myclient = new TcpClient (txt_host.Text,5020);//Connecting with server
```

VB.NET:

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

C#:

```
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 );}
```

VB.NET:

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

في الجزء الخاص بالـServer والذي يقوم بعملية التصنت على الـ Port واستقبال الـ Stream عبر الـ Socket و قراءتها 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 ب الـ Picture Box والتي تسهل علينا إمكانية عرض الصورة المرسلة من خلال Stream لكي يتم تحويلها من Stream إلى صورة تعرض على الـ Picture box وكما في المثال التالي:

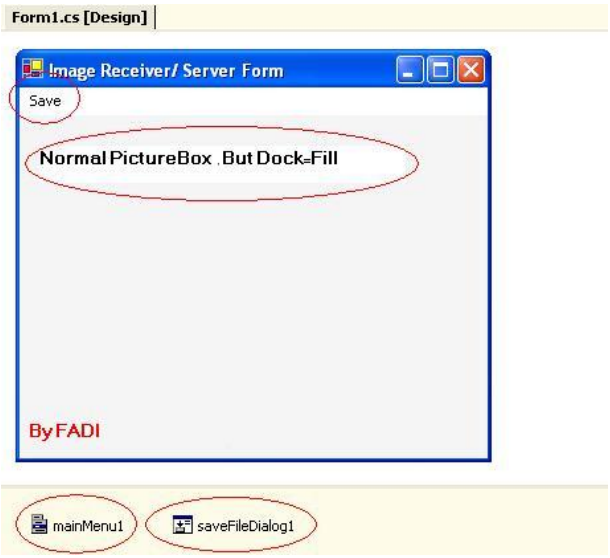
C#:

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

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

void Image_Receiver()
{
mytcpl = new TcpListener (5000);// Open The Port
mytcpl.Start ();// Start Listening on That Port
mysocket = mytcpl.AcceptSocket ();
myns = new NetworkStream (mysocket);
pictureBox1.Image = Image.FromStream(myns); // Show The Image that Resaved
as Binary Stream
mytcpl.Stop();// Close TCP Session

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

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

VB.NET:

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

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

ولتطبيق سنقوم بإنشاء مشروع جديد كما في الشكل التالي :



سنقوم بوضع الـImage_Receiver() Method إما في الـ Constructer الخاص بالبرنامج أو بحدث بدأ التشغيل الخاص بالـ Form ، و الميثود التالية في حدث الـ Closing الخاص بالـ Form وذلك لتأكد من إغلاق الـSocket عند إنهاء البرنامج:

C#:

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

VB.NET:

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

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

C#:

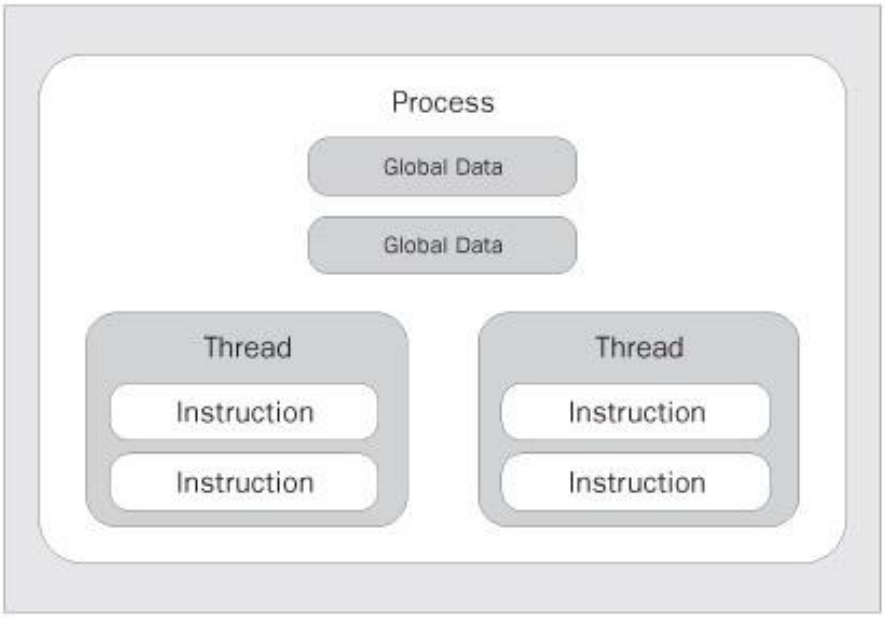
```
try
{
saveFileDialog1.Filter = "JPEG Image (*.jpg)|*.jpg" ;
if(saveFileDialog1.ShowDialog() == DialogResult.OK)
       {
string mypic_path = saveFileDialog1.FileName;
pictureBox1.Image.Save(mypic_path);
       }
}
catch (Exception)
       {

       }
```

VB.NET:

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

Threading Overview1.3.3 :

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



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

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

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

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

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

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

VB.NET:

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

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

C#:

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

}
```

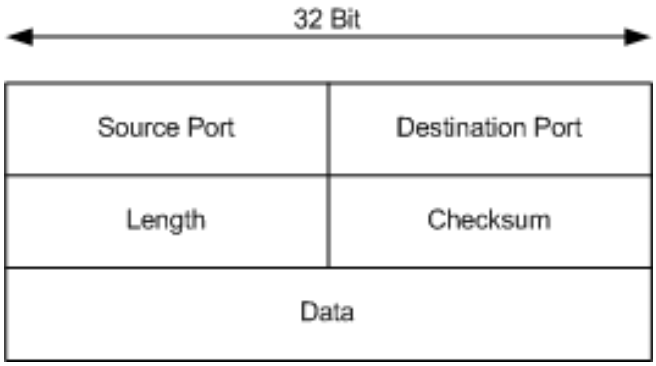
VB.NET:

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

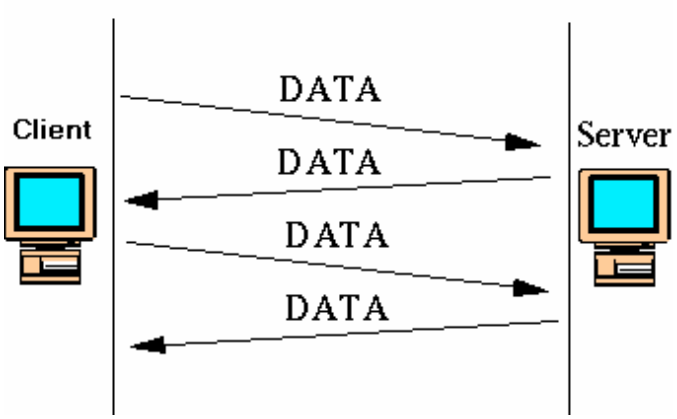
ميزة ال Thread رائعة جدا إذ تمكنك من تشغيل أكثر من Thread وفي نفس الوقت وفي نفس البرنامج وهو ما يسمى بالـ Multithreading ، سنبين في الجزء التالي من هذا الفصل طبيعة الاتصال باستخدام الـUDP .

UDP-User Datagram Protocol- Encapsulation : 1.3.4

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



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



لاستخدام ال UDP في الدوت نت يلزم أولا تعريف System.Net Name Space و الـ System.Net.Socket لاحظ انه في ال TCP كان يلزم تعريف رقم الـPort والعنوان للجهاز المستقبل أما في الـ UDP فتستطيع تعريفه كما هو في TCP كما وتستطيع عمل Broadcast باستخدام IPAddress.Any أو IPAddress.Broadcast بعد اشتقاق كائن من الكلاس IPEndPoint وتستطيع أيضا عدم تحديد رقم الـPort باستخدام ال Bind Method حيث يتم تعريفها بـ 0 ...

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

C#:

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

VB.NET

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

وتتم عملية إنشاء الـSocket وتحديد نوع البرتوكول المستخدم كما يلي:

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 الميثود Send ...

في الـ Bind Method والتي يتم وضعها في الطرف المستقبل فقط يتم استخدامها لربط الـ IP Address ورقم الـ Port بالـ Socket :

C#:

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

VB.NET:

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

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

C#:

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

VB.NET:

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

لاحظ أن العنوان المسند إلى IPEndPoint Object هو Any ورقم الـPort صفر وهذا يعني إرسال الرسالة المستلمة إلى الكل وبما فيهم الشخص مرسل الرسالة و الـServer :

C#:

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

VB.NET:

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

لطباعة عنوان مرسل الرسالة و الرسالة نفسها:

C#:

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

VB.NET:

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

نقوم هنا بإرسال رسالة ترحيبية لكل جهاز جديد يقوم بشبك مع الـServer لنخبره بها انه تم الموافقة على انضمامه:

C#:

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

VB.NET:

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

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

C#:

```
while(true)
 {
   data = new byte[1024];
   recv = newsock.ReceiveFrom(data, ref Remote);


   Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));
   newsock.SendTo(data, recv, SocketFlags.None, Remote);
 }
server.Close();
```

VB.NET:

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

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

وسوف اعرض المثال الكامل لعملية الإرسال عبر الـUDP ،

أولا الجزء الخاص بالServer :

C#:

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

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

      while (true)
      {
         data = new byte[1024];
         recv = newsock.ReceiveFrom(data, ref Remote);

         Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));
         newsock.SendTo(data, recv, SocketFlags.None, Remote);
      }
   }
}
```

VB.NET:

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

Class SimpleUdpSrvr

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

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

ثانيا الجزء الخاص بالـ Client ، يقتصر العمل على قيام الـ Client بإنشاء جلسة مع الـServer وذلك بعد تعريفه بالـ IPEndPoint ورقم الـPort وكما تم في السابق إلا أن الاختلاف هو في الوظيفة إذا يقتصر فقط على استقبال الرسالة من الـServer وإرسال أي رساله له عبر الـPort المخصص :

C#:

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

class SimpleUdpClient
{
```

```
public static void Main()

{

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

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

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

في حالة فقدان الاتصال مع الServer يظهر الرسالة التالية :

```
  string welcome = "Hello, are you there?";
 data = Encoding.ASCII.GetBytes(welcome);
 server.SendTo(data, data.Length, SocketFlags.None, ipep);
 IPEndPoint sender = new IPEndPoint(IPAddress.Any, 0);
 EndPoint Remote = (EndPoint)sender;
 data = new byte[1024];
 int recv = server.ReceiveFrom(data, ref Remote);
 Console.WriteLine("Message received from {0}:", Remote.ToString());
 Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));

while(true)
 {
  input = Console.ReadLine();
  Exit  في حالة إذا أردت إنهاء الجلسة اكتب
if (input == "exit")
   break;
  server.SendTo(Encoding.ASCII.GetBytes(input), Remote);
  data = new byte[1024];
  recv = server.ReceiveFrom(data, ref Remote);
  stringData = Encoding.ASCII.GetString(data, 0, recv);
  Console.WriteLine(stringData);
 }
 Console.WriteLine("Stopping client");
 server.Close();
}
}
```

VB.NET:

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

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

Chapter 2
IPv4 & IPv6 Architecture Overview

- IPv4 Architecture

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

- CIDR Nation Overview

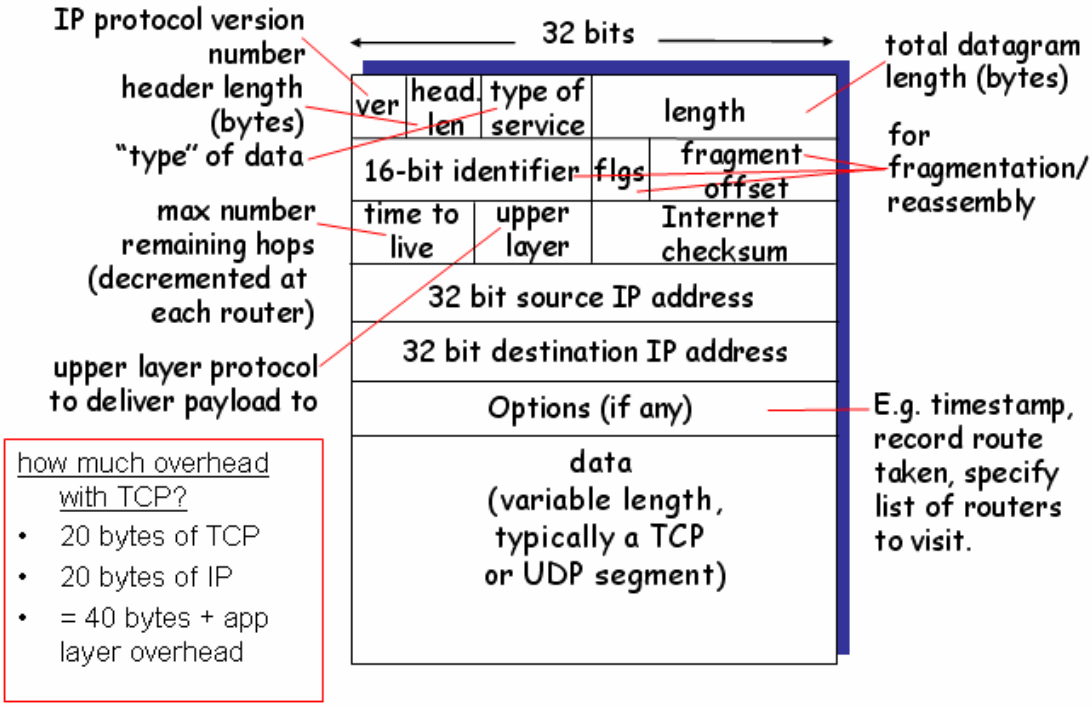
- IPv6 Architecture Overview

بسم الله الرحمن الرحيم

في الـNetwork Layer تتم عملية عنونة الـPacket باستخدام بروتوكول الإنترنت IP، وتتم هذه المرحلة بناءا على البروتوكول المستخدم في الـTransport Layer فإذا تم استخدام الـTCP عندها لا نستطيع إلا أن يكون العنوان المستخدم 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 الـUnicast والـMulticast والـBroadcast فيه:

IPv4 Architecture : 2.1

يوضح الشكل التالي التركيب العام لبروتوكول الإنترنت IPv4 :



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

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

وتقسم الـ20 Bytes الخاصة بالـIPv4 Header كما يلي:

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

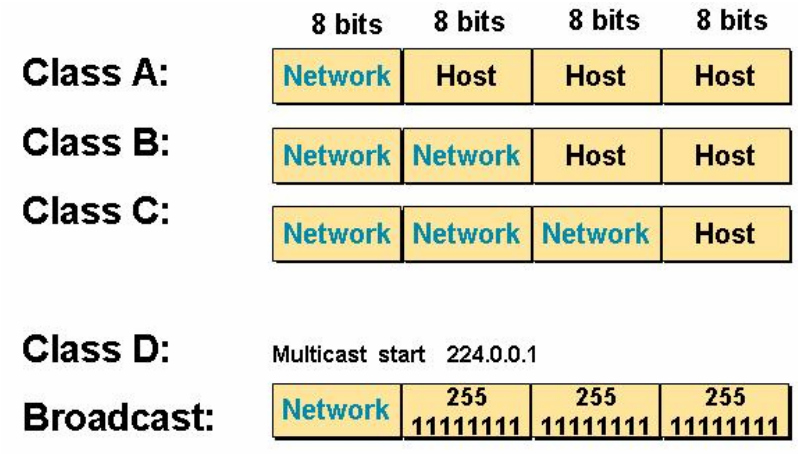
16 Bits للـ Checksum ولا تختلف طريقة حسابه عن الطريقة المستخدمة في الـTCP أو الـUDP والتي شرحناها في الفصل الأول.

32 Bits لتحديد عنوان الجهاز المرسل
32 Bits أخرى لتحديد عنوان الجهاز المستقبل

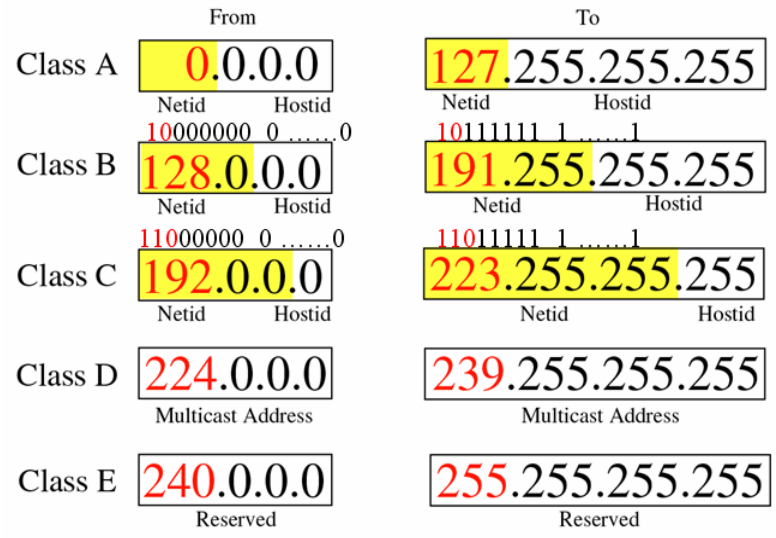
وسف نبين في الجزء التالي من هذا الفصل طريقة توليد الـUnicast والـBroadcast والـUnicast باستخدام الـ Classful والـ CIDR - Classless InterDomain Routing :

Classful IP Address : 2.1.1

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



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

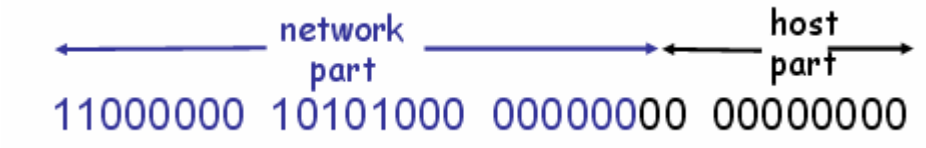


ولاستخراج عنوان الـ Broadcast من أي من التقسيمات السابقة يتم تعبئة الـBits الخاصة بالـHost ID بواحد وكمثال لمعرفة الـ Broadcast Address للعنوان 10.0.0.1 يجب أولا تحديد إلى أي Class ينتمي هذا العنوان ، ومن الواضح أنه ينتمي إلى Class A لأن الجزء الخاص بالـ Network ID يبدأ بـ10 وهو بين 1 والـ126 ، إذا الجزء الخاص بالـHost ID هو 0.0.1 ولتحويله إلى Broadcast Address نضع قيمة 255 في الجزء الخاص بالـHost ID ويصبح العنوان كما يلي: 10.255.255.255 وهو الـ Broadcast Address للـ Class A في الـ Classful Nation ...

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

CIDR - Classless InterDomain Routing IP Address : 2.1.2

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



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

```
8              8         6
192          168             1           0
11111111 11111111  11111100    00000000
```

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

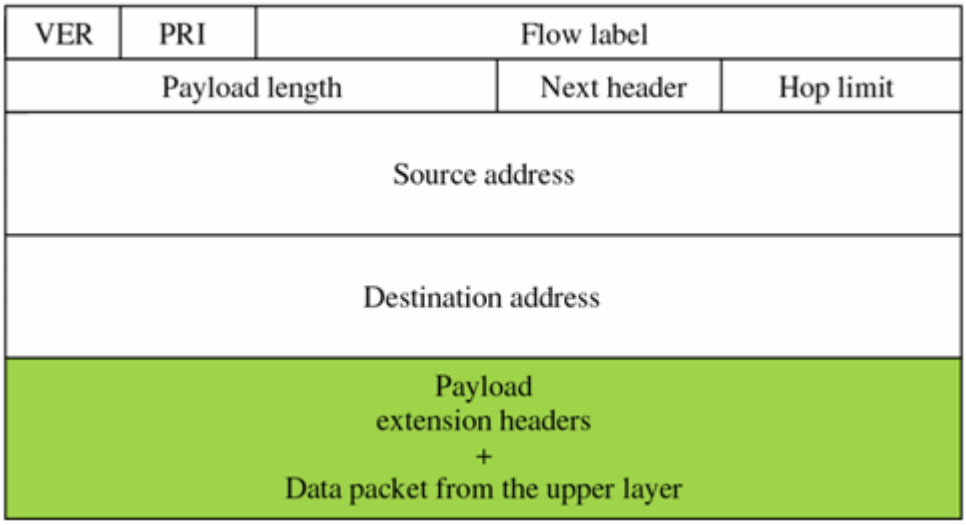
لاحظ أن مشكلة المحدودية للعناوين قد حلت بشكل جزئي باستخدام الـCIDR لاكن مازالت الإمكانيات محدودة إذا أردنا توليد عناوين لملاين من الأجهزة وكان الحل في الـIPv6 حيث زاد فيه حجم العناوين من 32 Bits إلى 128 Bits والذي سيتم الحديث عنه في الجزء التالي من هذا الفصل.

IPv6 Architecture : 2.2

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

- الحاجة إلى عناوين أكثر حيث أن القيمة العظمى للـIPv4 Address تبقى محدودة مع زيادة الطلب على عناوين الـIP's في العالم.
- قد يؤدي الـIPv4 إلى مجموعة من المشاكل وخاصة في Routing والتي قد تم حلها في الـIPv6
- مشكلة الـSecurity في الـIPv4 حيث لم يدعم أي من عمليات التشفير والتحقق Authentication على مستوى الـNetwork Layer وقد حلت هذه المشكلة في الـIPv4 باستخدام بروتوكول الـIPSec حيث يتم تشفير الـIP والـPort على مستوى الـSocket لاكن أصبحت الحاجة ملحة لجعل هذه الـSecurity مدمجة على مستوى الـNetwork Layer .
- تطوير مبدأ الـBroadcast حيث تم تطويره إلى الـany cast إذ يتم إرسال رسالة واحدة إلى كل جهاز على الشبكة وفي حالة وجد الجهاز المعني يتوقف الـRouter عن الإرسال ، والهدف في هذه الطريقة إيجاد طريقة تخفض من الـBandwidth المستخدم عند البحث عن جهاز ما على الشبكة.

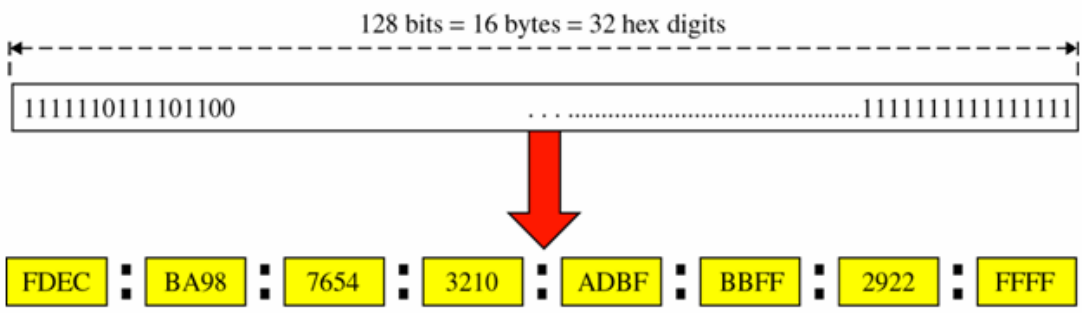
حل الـIPv6 كل هذه المشاكل حيث دعم ما يعادل 2^{128} Addresses وهو رقم كبير جدا كما قلل من الحجم الـRouting Table مما سرع من عمليات التوجيه Routing كما دعم عمليات الـAuthentication والتشفير على مستوى الـNetwork Layer كم تم حذف الـ Type of Services وحل محلها الـPriority أي الأولويات، ويوضح الشكل التالي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



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

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

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

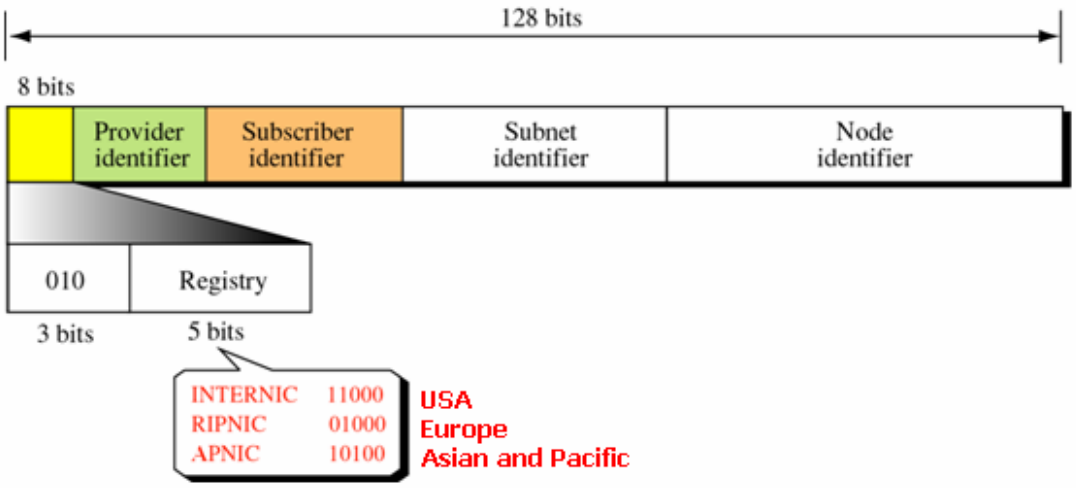


وتقسم العناوين في الـIPv6 إلى نوعين Provider-Based Addresses و Geographic-based Addresses حيث يقوم الـISP's بتوزيع العناوين على الـClients باستخدام الـStandard الخاص Provider-Based Addresses وأما الـ Geographic-based Addresses فهو مخصص لإعطاء العناوين الدولية ، أي انه سيكون لكل دولة رمز خاص يكون في بداية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

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

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

لاحظ الشكل التالي:



وقد بينا في هذا الفصل مبدأ عمل الـIPv4 والـIPv6 وكيفية توليد عناوين الـUnicast والـBroadcast باستخدام الـ Classful والـ CIDR Nation .
سيتم الحديث في الفصل التالي عن IP Multicasting واستخدامها لعمل الـ Multicast Group ...

Chapter 3
IP Multicasting Programming Overview

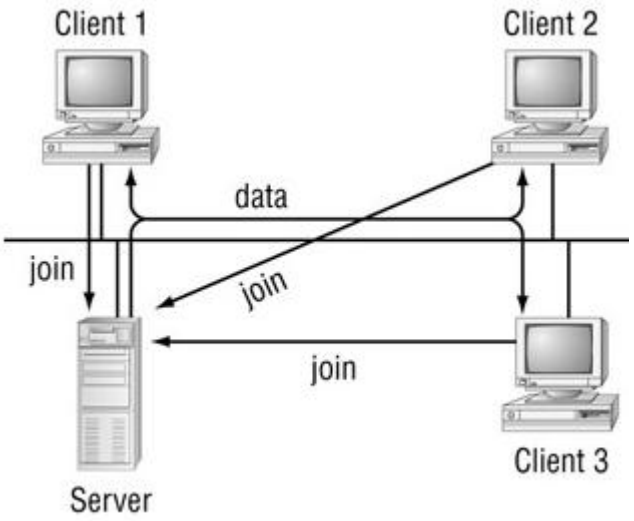
- IP Multicasting Overview

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

بسم الله الرحمن الرحيم

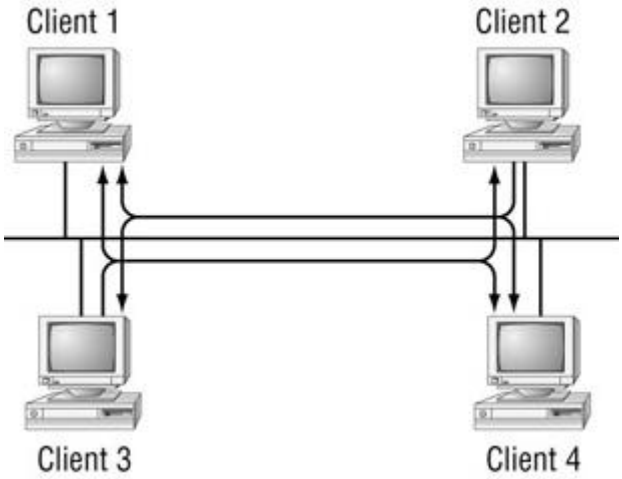
3.1: IP Multicasting Overview :

تحدثنا سابقا عن برتوكول ال UDP وشرحنا كيفية استخدامه لعمل برود كاست حيث تستطيع عمل البرود كاست بطريقتين إما باستخدام IPAddress.Any والذي يلزمه وجود Server يقوم بعملية التصنت على الPort المحدد حيث يستقبل من خلاله أي رسالة ثم يقوم ببثها إلى كل الأجهزة أو باستخدام IPAddress.Broadcast والذي من خلاله يمكن عمل بث إلى كل الأجهزة حيث لا ضرورة لوجود جهاز Server بحيث أن الكل يمكنه التصنت على الـ Port المحدد و يستقبل ويرسل من خلالها 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 اغلب برمجيات الـ Conferences تعتمد عليه بشكل كبير ويعرف Multicast على انه الإرسال إلى مجموعة من المستخدمين سواء كان Managed باستخدام Client/Server حيث يكون هنالك جهاز Server في الشبكة وظيفته استقبال الرسائل من الـ Clients Group ثم إرسالها إلى كامل المجموعة مرة أخرى انظر إلى الشكل التالي :



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

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



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

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

3.2: IP Multicasting واستخدامها لعمل Multicasting Group

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

C#:

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

VB.NET:

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

Class MultiSend

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

في البداية قمنا بتعريف الـSocket وتحديد الجهة التي سوف تستقبل الرسالة وهي (أي شخص يتنصت على الشبكة باستخدام الـ IP Multicast Group المحدد) ثم تحديد نوع الـSocket والبرتوكول المستخدم ...

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

C#:

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

Class UdpClientMultiRecv

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

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

Part 2
Streaming in Dot Net

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

- Stream Classes
- Stream Members
- Stream Manipulation

بسم الله الرحمن الرحيم

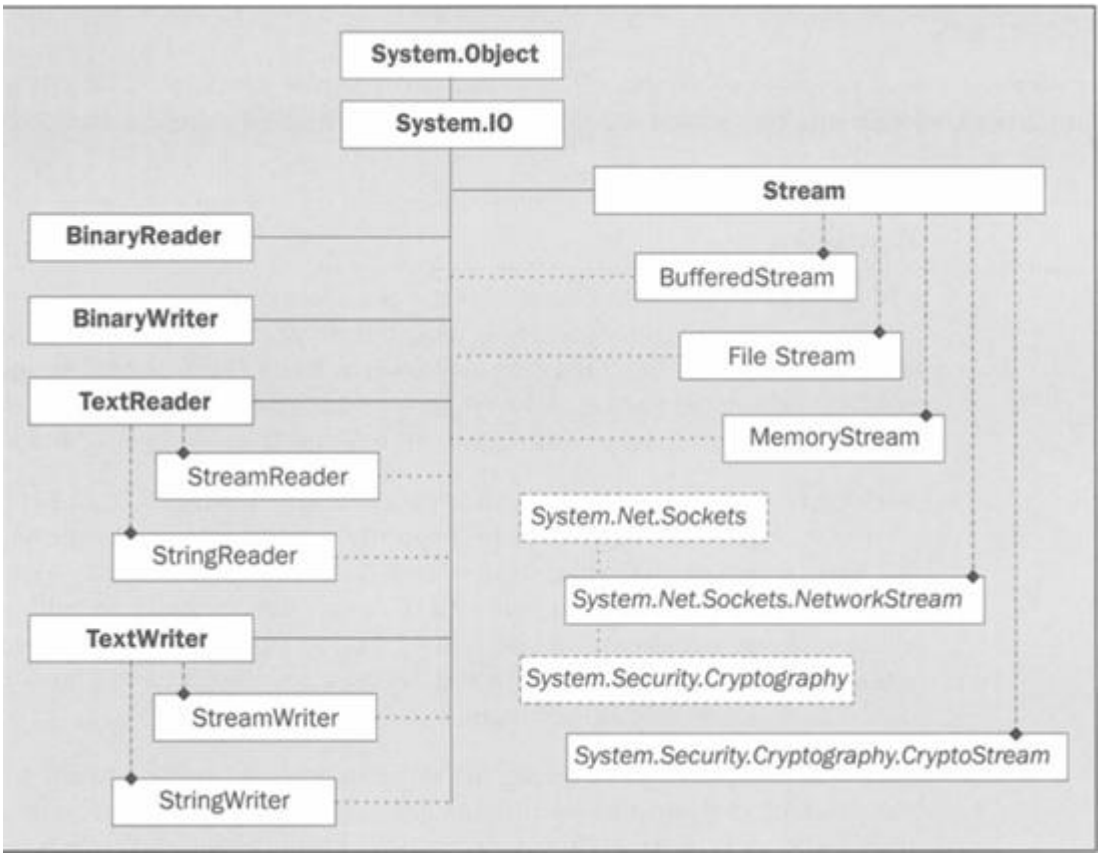
2.1 : Managed I/O: Streams, Readers, and Writers :

تحدثنا سابقا في الجزء الأول بشكل عام عن استخدامات الـ Streams Library واستخدامها لإرسال Binary Data و Text Data من جهاز إلى آخر وكمثال قمنا بإرسال صورة من الـ Client إلى الـ Server باستخدام الـ Binary Reader & Binary Writer ..
إن الهدف من إنشاء مكتبات الـ Stream هو تسهيل عملية نقل البيانات من مكان إلى أخر سواء عبر الشبكة أو داخل نفس الجهاز كما هو الحال بتعامل مع الملفات أو التعامل مع الطابعة أو أي طرفية أو جهاز آخر موصول بالكمبيوتر حيث تسهل علينا عملية تحويلها إلى Byte Array وإرسالها وهو ما حل الكثير من المشاكل التي كانت تواجه المبرمجين في التعامل مع Binary Data ..

يمكن التعامل مع الـStream بأسلوبين المتزامن **Synchronous** والغير متزامن **Asynchronous** وبشكل افتراضي تعمل جميع الـIO Streams بالأسلوب المتزامن لاكن العيب فيه هو تأثيره الشديد على أدائية النظام إذ يقوم بإغلاق الـProcessing Unit في الـThread المخصصة للبرنامج بحيث لا يسمح بتنفيذ أي أمر آخر إلا بعد الانتهاء من العملية الجارية ولا ينصح ابدآ استخدام الأسلوب المتزامن في حالة إذا كنت تتعامل مع أجهزة قراءة وكتابة بطيئة نسبيا مثل الـFloppy Disk أو الـMagnetic Tape لكنها مهمة جدا بالبرمجيات التي تعتمد على أنظمة الزمن الحقيقي أو الـReal Time Systems حيث أنها تعتمد الأسلوب المتزامن في عملية إرسال واستقبال البيانات وهو ما يمنع القيام بأي عملية أخرى إلى حين الانتهاء من تنفيذ الأمر ومن الأمثلة عليها أنظمة السحب أو الإيداع في الرصيد البنكي أو أنظمة حجز التذاكر أو شحن بطاقة الهاتف وغيرها .. طبعا في حالة إذا كان برنامجك لا يحتاج إلى وجود الخواص السابقة عندها ينصح باستخدام الأسلوب الغير متزامن Asynchronous حيث تستطيع من خلاله تنفيذ عمليات أخرى في وحدة المعالجة وبدون الحاجة لانتظار إنهاء العملية الجارية إذ يتم إنشاء Separate thread لكل عملية طلب إدخال أو إخراج مما لا يؤثر على أدائية النظام وينصح باستخدامه إذا كانت عملية القراءة أو الكتابة تجرى من خلال أجهزة بطيئة نسبيا ويمكن تميز الميثود المتزامن عن الغير متزامن في الدوت نيت بوجود كلمة Begin أو End في بداية اسم الميثود الغير متزامن وكمثال عليها BeginWrite و BeginRead و الـEndWrite و الـEndRead ..

أولا: Stream Classes

تدعم الدوت نيت عمليات الـStreams بمجموعة من الـClasses والمندرجة تحت Name System.IO Space والتي تستخدم لعمليات الإدخال و الإخراج لنقل البيانات . تستخدم بعض الـStream Classes ، Backing storage ، ومن الأمثلة عليها FileStream و الـBufferedStream والـMemoryStream وكذلك فإن بعضها لا يستخدم أي Back Storage ومن الأمثلة عليها الـNetworkStream والتي تستخدم لنقل الـStream عبر الشبكة وبدون استخدام Backing Storage ، و تقسم الـStream Classes في الدوت نيت كما في الشكل التالي :



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

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

C#

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

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

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

VB.NET:

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

    Class Program

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

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

حيث قمنا بتحويل نص إلى Byte Array باستخدام الـASCIIEncoding وتحميله في عبر الـMemoryStream ثم ارسلناه إلى الميثود readBufStream والتي انشأناها حيث استقبلنا من خلالها الـStream وحملناه في ذاكرة مؤقتة بستخدام الكلاس الـBufferedStream ثم قمنا بطباعة محتوياته بعد تحويله إلى نص مرة اخرى بستخدام الـEncoding.ASCII وطباعته ..

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

C#

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

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

VB.NET:

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

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

C#

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

NetworkStream myns = myclient.GetStream ();

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

VB.NET:

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

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

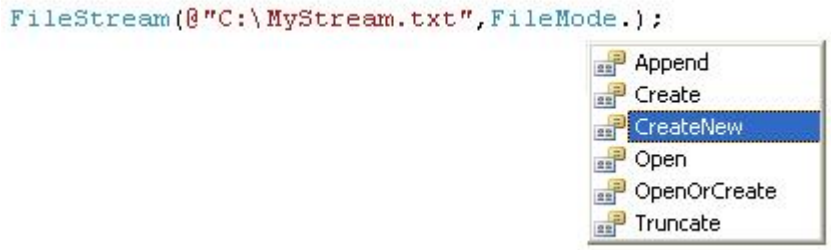
C#

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

VB.NET:

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

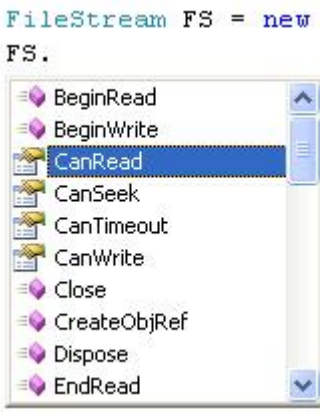
يمكننا استخدام الـEnumeration التالية مع الـFileMode :



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

ثانيا : Stream Members :

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



1- CanRead و CanWrite وتستخدم لمعرفة إذا كان الـStream المستخدم يقبل عملية القراءة أو الكتابة أم لا حيث ترجع قيمة True في حالة إذا كان يقبل و False في حالة أنه لا يقبل ويستخدم عادة قبل إجراء عملية القراءة أو الكتابة لفحص مدى الصلاحية قبل المحاولة ..
2- CanSeek حيث يستخدم الـSeeking عادة لتحديد موقع الـCurrent Stream والعادة تدعم الكلاسات التي تستخدم Backing Storage هذه العملية مثل الـFileStream وعندها ترجع قيمة True وترجع قيمة false في حالة إذا كان الـStream Class لا يحتوي على Backing Storage .

3- CanTimeout وترجع قيمة True في حالة إذا كان الـstream يحتوي على خاصية الـ Timeout والتي تعطي وقت محدد للعملية .
4- Length وتستخدم لمعرفة حجم الـStream بالـByte ويمكن الاستفادة منها لمعرفة نهاية الـStream أو لتحديد حجم المصفوفة بناء على حجم الـStream .
5- Position وتستخدم الـGet و Set لمعرفة أو تحديد الموقع لـ Stream
وتشترك مكتبات الـStream بمجموعة من الميثودس وهي كما يلي :

1- الميثودس المتزامنة Synchronous Methods :

I. Read و ReadByte وتستخدم لقراءة Stream Data وتخزينه في الـBuffer ويمكن تحديد عدد البايتات التي سيتم قراءتها باستخدام الـReadByte كما نستطيع من خلالها معرفة نهاية الـStream حيث ترجع الـRead قيمة 0 والـReadByte قيمة -1 في حالة انتهاء الـStream.
II. Write والـWriteByte وتستخدم لعملية الإرسال عبر الـStream ويمكن تحديد عدد البايتات التي سيتم كتابتها في كل مرة باستخدام الـWriteByte.

2- الميثودس غير المتزامنة Asynchronous Methods :

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

```
FS.BeginRead(
```

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

1- الـByte Buffer والتي سوف تستخدم لعملية القراءة منه أو الكتابة عليه
2- الـoffset والذي سوف يحدد فيه موقع القراءة أو الكتابة
3- الـnumByte والذي سوف يتم فيه تحديد الحد الأقصى من البايتات التي سيتم كتابتها أو قراءتها
4- الـAsyncCallback وهو Optional Delegate حيث يتم استدعائه عند الانتهاء من عملية القراءة أو الكتابة
5- الـStateobject وهي User Provided Object وتستخدم لتميز الـ Read & Write Request عن غيره الـRequests .
ترجع الـBegin Methods الـIAsyncResult والذي يمثل حالة الـStream Operation .

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

وهناك بعض الميثود والتي تستخدم لإدارة الـStream وهي :

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

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

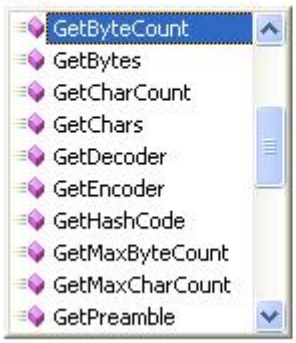
3- SetLength وتستخدم لتحديد حجم الـStream والذي نريد إرساله أو استقباله لاكن في حالة إذا كان الـStream أقل من المحدد في الـSetLength سوف يؤدي ذالك إلى انقطاع الـStream وعدم وصوله بشكل سليم ، لن تستطيع استخدام هذه الخاصية إلا إذا تأكدت انك

تملك الصلاحية لذلك من خلال الخاصية CanWrite و CanSeek لذا ينصح بفحص الصلاحية أولا قبل تحديد حجم الـStream .

ثالثا : Stream Manipulation:

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

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



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

ولتعامل مع الـStreamReader و الـStreamWriter لنقل Text يجب أولا استدعائها من الـSystem.IO نيم سبيسس وتستخدم كما يلي:

StreamReader للقراءة من ملف:

C#

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

VB.NET:

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

StreamWriter للكتابة إلى ملف:

C#

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

VB.NET:

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

و لتعامل مع الـBinaryReader والـBinaryWriter لنقل Binary Data يتم استدعائها من الـSystem.IO نيم سبيسس وتستخدم كما يلي:

BinaryReader لقراءة Binary Data من الـStream:

C#

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

BinaryWriterr لإرسال BinaryData إلى الـStream عبر الـSocket:

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

VB.NET:

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

وهكذا بينا أهم مكتبات الـStream في الدوت نيت وطرق التعامل معها ، والفرق بين الـStreams المتزامن والغير متزامن في بيئة الدوت نيت ، سوف نطبق في الفصل التالي مثالين على الـSteaming في بيئة الدوت نيت الأول لإرسال رسائل تحكم والثاني لتخزين صورة في قاعدة بيانات Access و SQL.

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 + (Control in Full Version Book)

- Create an Advanced Remote Web Camera Monitoring System By Using TCP Streaming Connection & Image Processing.

- 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

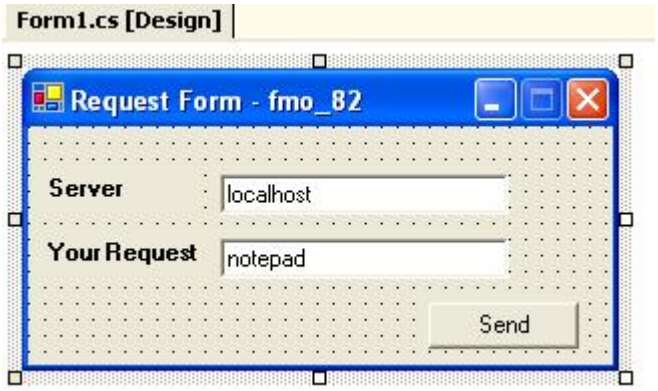
بسم الله الرحمن الرحيم

سوف نناقش في هذا الفصل مجموعة من التطبيقات باستخدام الـ TCP Streaming حيث سنبني نظام لتحكم عن بعد بإطفاء وتشغيل البرامج وإطفاء وتشغيل الأجهزة ونظام آخر لمراقبة سطح المكتب عن بعد ونظام لمراقبة المكان عبر الكاميرا عن بعد حيث يقوم بعملية مقارنة لصور الملتقطة وفي حالة وجد أي اختلاف يقوم النظام بالاتصال بشخص صاحب النظام باستخدام الـ TAPI IP Telephony ، وأخيرا طريقة تخزين الـ Binary Data في قواعد البيانات Access و SQL Server Database.

Remote Control Example 5.1 باستخدام الـStream Reader & Writer:

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

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



ثم قم بإضافة Name Spaces التالية :

C#

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

في Send Button قم بكتابة الكود التالي:

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

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

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

VB.NET:

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

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

قم بإضافة Name Spaces التالية :

C#

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

ثم إضافة التعاريف التالية :

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

ثم نقوم بإنشاء ميثود جديدة كما يلي :

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

// you can add any order and Response Here
if (order=="notepad") System.Diagnostics.Process.Start("notepad");
```

```
else if (order=="calc") System.Diagnostics.Process.Start("calc");
else MessageBox.Show("Sorry Sir Your Request is not in my hand",order);

mytcpl.Stop();// Close TCP Session

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

VB.NET:

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

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

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

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

C#

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

ثم قم بإضافة التالي في حدث الـForm Closing وذلك لتأكد من إغلاق الـSocket والـStream في البرنامج ..

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

VB.NET:

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

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

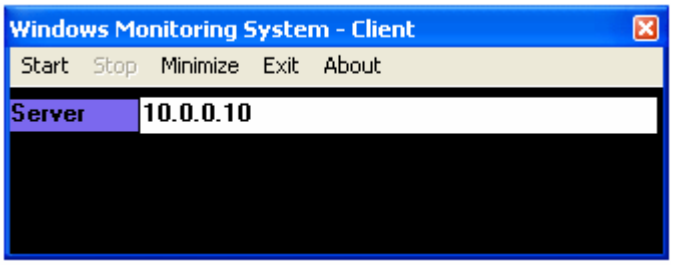
كما يمكننا بنفس الخطوات السابقة إنشاء برنامج يقوم بإطفاء الـClients Computer عن بعد وذلك بتنفيذ الملف Shutdown.EXE (والمرفق مع الكتاب) عند ورود كلمة Shutdown أو تنفيذ أمر الـ RPC – Remote Procedure Call والذي يأتي مع نظام التشغيل Windows 2000/XP ...، في المثال التالي سنستخدم الـStream Library لتخزين صورة في قاعدة بيانات Access وقاعدة بيانات Microsoft SQL Server ...

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 اخر باستخدام الـ TCP والـ Stream Library :

اولا إنشاء برنامج الـ Client :

لتطبيق سنقوم بإنشاء مشروع جديد كما في الشكل التالي:



سوف نستخدم الـ Namespaces التالية:

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

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

C#:

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

VB.NET:

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

ولإرسال الصور الملتقطة إلى الجهاز الآخر لابد من تعريف Socket وكما يلي:

في حالة استخدام TCP Socket :

C#:

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

myclient = new TcpClient (Server_IP.Text,5020);//Connecting with server
myns = myclient.GetStream ();
mysw = new BinaryWriter (myns);
mysw.Write(arrImage);//send the stream to above address

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

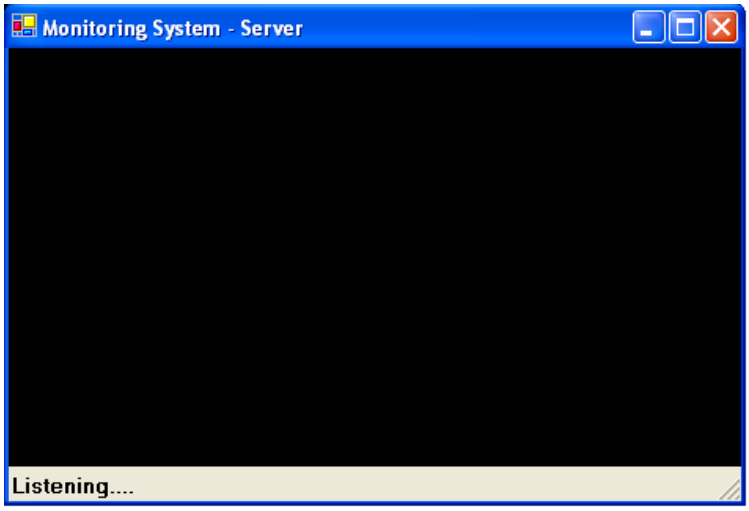
VB.NET:

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

myclient = New TcpClient (Server_IP.Text,5020) 'Connecting with server
myns = myclient.GetStream ()
mysw = New BinaryWriter (myns)
mysw.Write(arrImage) 'send the stream to above address

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

وبتأكيد نضع هذا الكود في Timer ويحدد الـ Interval فيه بناء على سرعة الإرسال التي نحتاجها من جهة ومقدرة الشبكة من جهة اخرى ، 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.Sockets
System.IO
```

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

C#:

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

VB.NET:

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

ثم نقوم بتعريف Method الإستقبال كما يلي:

C#:

```
void ServerMethod ()
{
        try
        {

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

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

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

}
catch (Exception){}
}
```

VB.NET:

```
Private Sub ServerMethod()
    Try

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

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

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

    Catch e1 As Exception
    End Try
End Sub
```

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

C#:

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

VB.NET:

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

وهكذا بينا طريقة إرسال صورة سطح المكتب من جهاز إلى آخر ، سوف نبين في النسخة الورقية طريقة عمل برنامج Remote Desktop مع إضافة خاصية التحكم عبر الـ Mouse والـ Kayborad وذلك باستخدام دوال الـ API .

5.3 مثال نظام المراقبة عبر الكاميرا والاتصال عبر TAPI Telephony في حالة ورود تغير في الصور الملتقطة :

ستجد كافة تفاصيل هذا الموضوع في النسخة الورقية من الكتاب
لطلب أ والاستفسار أو التوزيع يرجى الاتصال على احد العناوين التالية

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

الموقع الرسمي للكتاب
www.fadidotnet.org

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

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

C#:

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

VB.NET

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

أولا استخدام Microsoft Access :

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

C#:

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

try
{

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

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

```
        }

finally {oleDbConnection1.Close();}

 }
```

VB.NET

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

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

C#:

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

VB.NET

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

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

ثانيا استخدام Microsoft SQL Server :

لاتختلف عملية التخزين بقاعدة بيانات SQL Server عن الـMicrosoft Access سوى انه يتم استخدام الـ SqlParameter بدلا من OleDbParameter ، كما أن صيغة الـ Connection String ستختلف بين الـAccess Database والـSQL 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 Streing :

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
(?)",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 (?)",
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)
```

وهكذا بينا في المثال الأول كيفية استخدام الـStream لإرسال أوامر إلى الـ Server وفي المثال الثاني كيفية استخدام الـ MemoryStream لتخزين Binary Data في الـ SQL Server والـ Microsoft Access Database ...

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

Part 3
Transport & Network Layer Programming

Chapter 6
Transport TCP & UDP (Classes & Members)

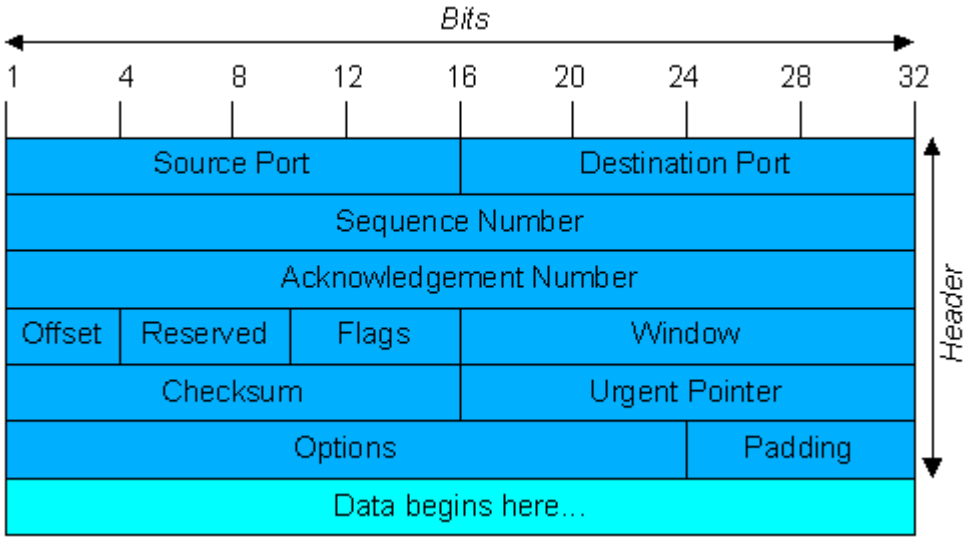
- TCP Classes Members
- UDP Classes Members

بسم الله الرحمن الرحيم

: TCP & UDP Classes Members : 6

سوف نتحدث في هذا الفصل عن الـ TCP Classes والـ UDP Classes واهم الـ Members لكل منهما والتي يتم التعامل معها في الـ Transport Layer ، ومن المعروف أن في هذه الطبقة يمكن التحكم بخصائص الإرسال ومنها الـ Offset والـ Flags بالإضافة إلى وضع وتحديد الـ Source و الـ Destenation Port لدى الطرف المرسل والمستقبل لاحظ الشكل التالي والذي يوضح الـ Header الخاصة بالـ TCP والـ Header الخاص بالـ UDP.

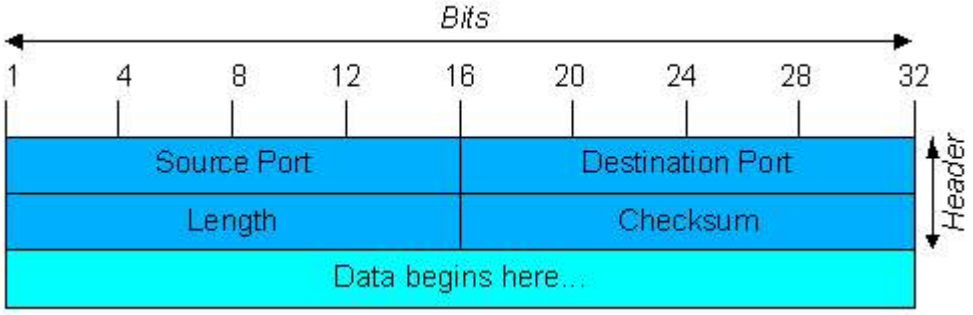
أولا الـ TCP Header ويتكون من 32 Bits للبكت الواحد حيث يتم فيه تخزين عنوان البورت المرسل في 16 Bits والمستقبل في 16 Bits والرقم التسلسلي في 32 Bits ورقم التحقق بالإضافة إلى الـ Checksum وفي النهاية يتم وضع الجزء الخاص بالبيانات:



TCP Header

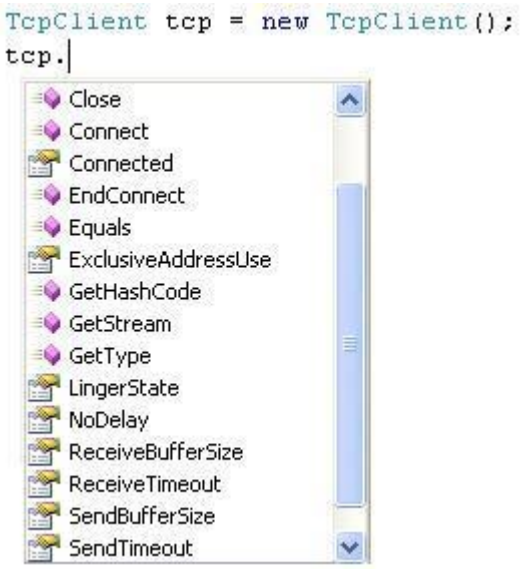
Data Offset: 4 bits the number of 32 bit words in the TCP Header. This indicates where the data begins. The TCP header (even one including options) is an integral number of 32 bits long.
Window: The number of data octets beginning with the one indicated in the acknowledgment field which the sender of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

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



UDP Header

أولا الـClasses الخاصة بالـ TCP Connection Oriented Protocol :



1-TcpClient Class: حيث تحتوي على مجموعة من الـ Methods والـ Properties وهي كما يلي:

أولا: أهم المثود الخاصة بها TCPClient Methods :

Connect: وتستخدم لأجراء عملية الاتصال مع الـserver حيث نمرر فيها عنوان الـIP الخاص بالـServer و رقم الـPort وكما يلي:

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

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

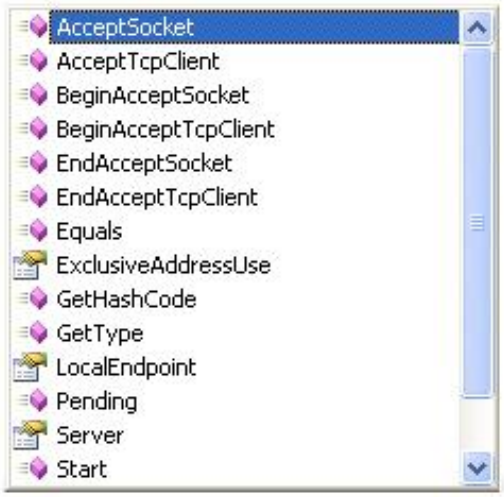
Close: لإنهاء الاتصال مع الـTCP Socket.
EndConnect: لإنهاء Asynchronies Connection حيث ترجع Asynchronies Result.
GetStream: ويستخدم لقراءة الـStream من الـSocket في عملية الإرسال و الاستقبال.

ثانيا: أهم الخصائص TCPClient Properties :

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

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

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



أولا: أهم الميثود الخاصة بها TcpListener Methods :

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

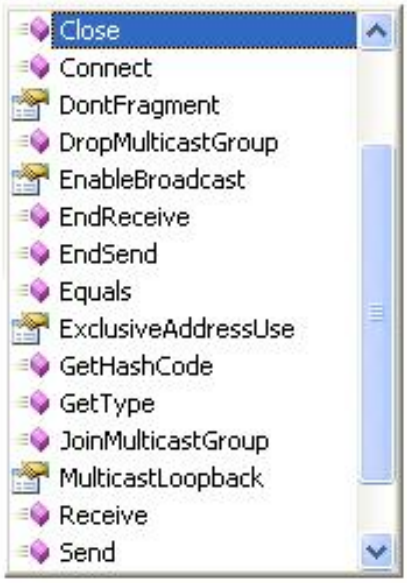
ثانيا: أهم الخصائص في TcpListener:

LocalEndpoint : حيث يرجع الـIP ورقم الـPort المستخدم في الـLocalEndpoint المحدد.
Server: ومن خلالها نستطيع الوصول إلى كل الخصائص و الميثود في الـTCP Server والتي شرحناها سابقا مثل الـAccept والـSendto والـReceive و Listen وغيرها

ثانيا الـClasses الخاصة بالـ UDP Connectionless Protocol :

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

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



ومن أهم الميثود والخصائص الخاصة بها :

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

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

أولا نعرف الـUDP Socket وكما يلي :

C#:

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

VB.NET:

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

ثانيا نقوم بتعريف Address List ثم نسند إليها الـIP الذي نريد إدخاله في الـGroup أو نجعل الـUser يدخل العنوان بنفسه نربطها بالسكوت باستخدام الميثود Bind وكما يلي:

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

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

ثالثا نقوم بتعريف الـMulticast Option ونسند لها العنوان المحدد كما يلي:

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

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

ومن ثم نضيف التغير على الـحيث تأخذ هذه الميثود ثلاثة باروميترات الأول لتحديد مستوى التغيير على IP أو على IPv6 أو على Socket أو TCP أو UDP وفي حالتنا هذه سوف نستخدم التغير على IP إذ ما نريده هو ضم IP إلى Multicast Group وفي الباروميتر الثاني نحدد نوع التغيير حيث نريد إضافة عضوية ويمكن الاختيار بين إضافة عضويه AddMembership أو إلغاء عضوية DropMembership وأخيرا نسند إليه 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 اهم الخصائص والتي يمكن التحكم بها في بيئة الدوت نيت والخاصة بالـ Transport Layer و الـ Network Layer، في الفصل التالي سيتم الحديث بشكل مفصل عن بقية الـ Members والتي يتم التعامل معها في الـ Network Layer وطرق التعامل مع الـ Synchronous Sockets والـ Asynchronous Sockets.

Chapter 7
Network Layer & Synchronous Sockets Programming

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

بسم الله الرحمن الرحيم

7: Network Layer & Synchronous Sockets Programming :

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

7.1: Introduction to Socket Programming

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

C#:

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

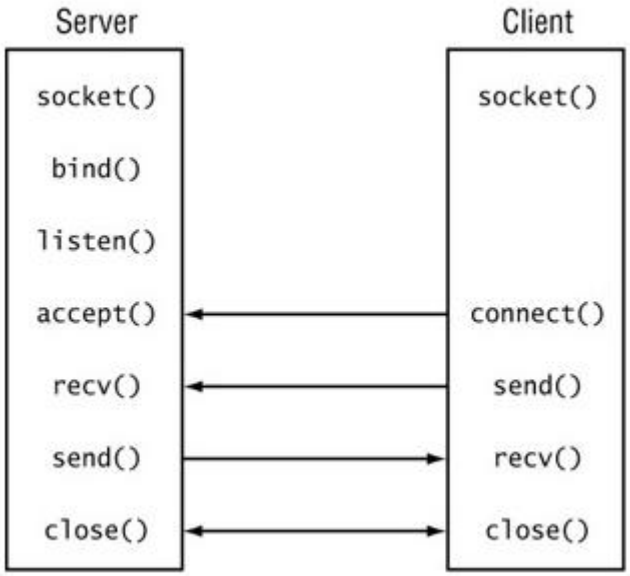
VB.NET:

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

حيث يتم في الباروميتر الأول تحديد نوعية الـIP Address والذي سوف تتعامل معه ويعطيك عدد كبير من الخيارات ومنها IPX والمستخدم في شبكات الـNovel أو ATM والمستخدم في شبكات الـATM Networks أو NetBIOS Address وغيرها ... ومن أهم هذه الخيارات الـInterNetwork وهو ما نستخدمه بشكل دائم مع البرمجيات الخاصة بالشبكات ويعرف على أن نوع IP هو من النوع IPv4 وهو المعتاد مع نظام مايكروسوفت وأغلب أنظمة التشغيل المعروفة حاليا وفي المستقبل القريب جدا سيتم الإستغناء عنه وليحل محله الـIPv6، في الباروميتر الثاني يتم تحديد نوع الـSocket أي هل سوف نستخدم Stream لإرسال البيانات أو شيء اخر وعادة ما يتم استخدام الـStream لهذه المهمة حيث اننا سنعتمد نمطية التراسل من النوع Stream ، وأخيرا نحدد نوع البروتوكول المستخدم للإتصال فهل هو من النوع UDP أو TCP أو بروتوكولات أخرى مثل ICMP Internet Control Message Protocol أو IGMP Internet Group Management Protocol أو اننا نريد مثلا إنشاء الـSocket لتعريف IP Security Header بإختيار IPSecAuthenticationHeader وغيرها وسوف نأتي على شرح مثل هذه الأمور لاحقا إنشاء الله، وهنا سوف نختار الـTCP أو UDP ومن المعروف أن برتوكول الـTCP هو برتوكول موجه وهذا يعني إجراء عملية التحقق من الوصول والتوصيل إلى شخص ما محدد أما برتوكول الـUDP فهو برتوكول سريع نسبيا و لاكنه لا يدعم عملية التحقق من الوصول السليم للبيانات المرسلة وهو مفيد جدا لإجراء عملية البث الإذاعي Broadcast وإنشاء مجموعات البث Multicast Group وهو ما شرحناه في الفصل الأول والثاني والثالث .

7.2 استخدام الـSynchronous Socket Programming لإنشاء TCP Connection:

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



إذ تبدأ العملية في الـClient و الـserver بإنشاء الـSocket كما يلي :

C#:

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

VB.NET:

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

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

C#:

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

VB.NET:

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

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

C#:

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

VB.NET:

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

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

C#:

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

VB.NET:

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

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

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

C#:

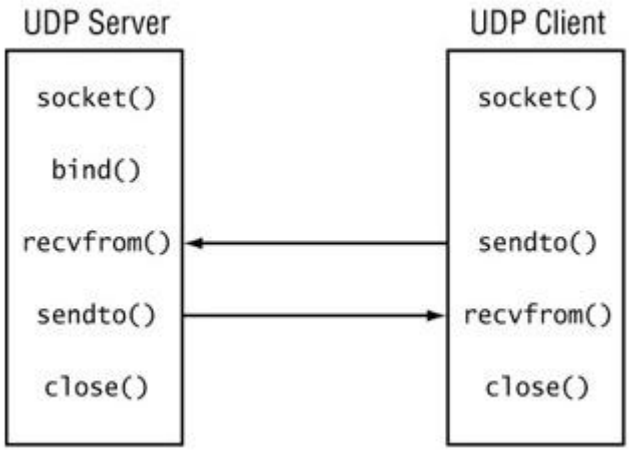
```
String str = Console.ReadLine();
ASCIIEncoding asen = new ASCIIEncoding();
byte[] msg = asen.GetBytes(str);

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

VB.NET:

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

7.3 استخدام الـSynchronous Socket Programming لإنشاء UDP Connectionless :

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



وتتشابه عملية الاتصال كما في الـTCP إذ تبدأ العملية في الـClient و الـserver بإنشاء الـSocket كما يلي :

C#:

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

VB.NET:

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

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

C#:

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

VB.NET:

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

ولاستقبال البيانات نستخدم الميثود ReceiveFrom حيث نعرف في البداية End Point Reference بناء على ما تم تعريفه في السابقة ونمرره ك reference مع مصفوفة الـByte إلى الميثود ReceiveFrom ومن ثم نستطيع تحويل المصفوفة إلى String من خلال الميثود 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 الميثود SendTo حيث نمرر لها البيانات بعد تحويلها من String إلى Byte Array وحجم البيانات المرسلة إذ يمكننا معرفته من خلال الميثود Length ونوع الـFlags حيث نريد عمل Broadcast لرسالة المرسلة واخيرا نمرر له الـEndPoint Object وكما يلي:

C#:

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

VB.NET:

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

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

Synchronous Socket Classes Members7.4:

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

C#:

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

VB.NET:

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

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

Any ويستخدم لتمثيل أي عنوان متاح على الشبكة

Broadcast ويستخدم لتمثيل البث الإذاعي لجميع الأجهزة على الشبكة

Loopback ويستخدم لتمثيل العنوان المعروف لل loopback وهو 127.0.0.1

None ويستخدم في حالة عدم وجود Network Interfase في النظام

كما يدعم مجموعة من الميثود وأهمها :

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

GetHashCode وتستخدم لإرجاع العنوان إلى صيغة Hash Code

HostToNetworkOrder ويرجع الجزء الخاص بالـNetwork من العنوان

NetworkToHostOrder ويرجع الجزء الخاص بالـHost من العنوان

1- **IPEndPoint Class :** حيث استخدمناه لتحديد العنوان والـPort للـ Host والذي نريد الاتصال به والصيغة العامة له كما يلي :

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

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

مجموعة الخواص التي تدعم في الـSocket Class وهي كما يلي:

AddressFamily ويرجع مجموعة العناوين المعرفة على الـSocket

Available ويرجع حجم البيانات الجاهزة للقراءة من الـSocket

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

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

Handle ويستخدم لمعرفة نظام التشغيل الذي يتعامل مع الـSocket

ProtocolType ويستخدم لمعرفة البروتوكول الذي يستخدم في الـSocket

RemoteEndPoint ويرجع معلومات عن الـSocket الذي يستخدم مع الـRemote Host

وكمثال لاستخداماتها:

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

VB.NET:

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

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

End Sub
```

حيث سترجع المعلومات التالية:

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

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 Sockets

- Asynchronous Socket Class and its members

- Applied Asynchronous Socket in Dot Net

بسم الله الرحمن الرحيم

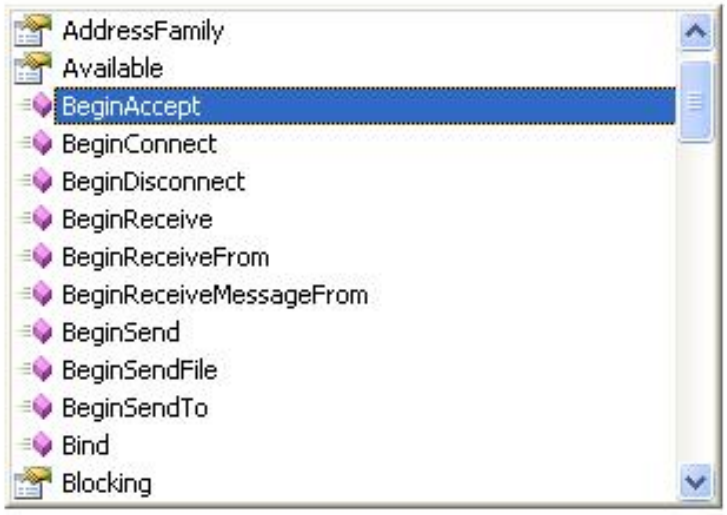
8: Asynchronous Sockets Programming :

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

أولا Asynchronous Socket Class and its members :

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

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



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

C#:

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

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

VB.NET:

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

حيث سيتم إضافة الـClient Request في Callback Reference منفصل عن السابق وهنا لابد من إنشاء method لاستقبال الـClient Request وإنهاء الـClient Accepted Object باستخدام الميثود 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 الـBeginAccept Method تأخذ عدة أشكال كما يلي:

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

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

الشكل الثاني في الدوت نيت 2005 حيث يمكنك فيه تحديد حجم البيانات المستلمة

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

الشكل الثالث في الدوت نيت 2005 حيث بمكن فيه تحديد ال Accepted Socket

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

2- BeginConnect وتستخدم لبدأ Asynchronous Connection على الـSocket ورقم الـPort المحدد حيث يسند لها الـIPEndPoint والـAsynchronous Callback والـState Object وكما يلي:

MySocket.BeginConnect(EndPoint IP,Syncallback Result,object state)

وتستخدم كما يلي كمثال:

C#:

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

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

VB.NET:

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

في الـConnected Method يتم تحديد الـCallBack Socket كما يلي:

C#:

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

VB.NET:

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

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

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

ويستخدم كما يلي كمثال:

C#:

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

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

VB.NET:

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

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

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

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

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

وتستخدم كما يلي كمثال:

C#:

```
private static void SendData(IAsyncResult iar)
 {
        Socket server = (Socket)iar.AsyncState;
        int sent = server.EndSend(iar);
  }
byte[] data = Encoding.ASCII.GetBytes("Hello Word");
MySocket.BeginSend(data, 0, data.Length, SocketFlags.None,
new AsyncCallback(SendData), MySocket);
```

VB.NET:

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

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

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

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

تأخذ الـ**BeginSendFile** التركيب التالي:

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

والـ**BeginReceiveMessageFrom** التركيب التالي:

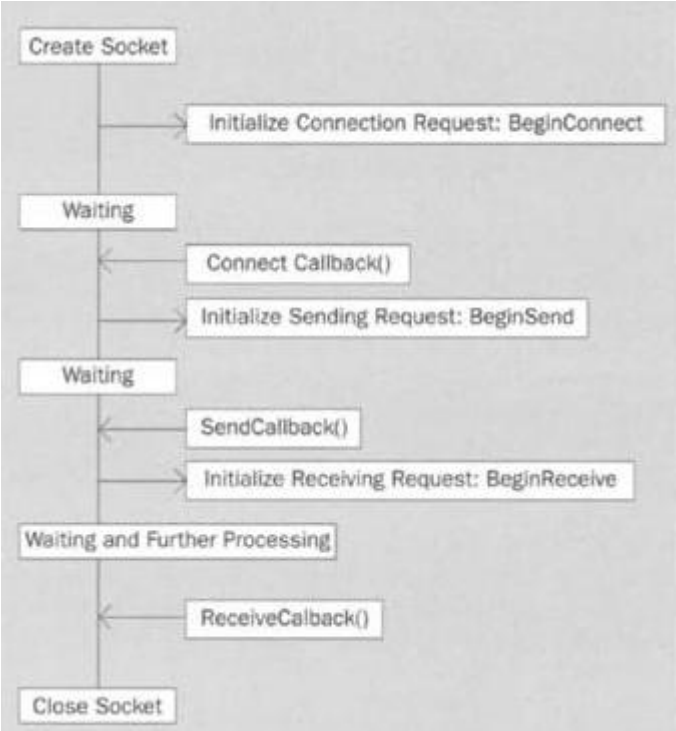
```
MySocket.BeginReceiveMessageFrom(byte Buffer ,int offset,int size,SocketFlags
sf,ref EndPoint,AsyncCallback ascb,object state)
```

والـ**BegonDiconnect** التركيب التالي:

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

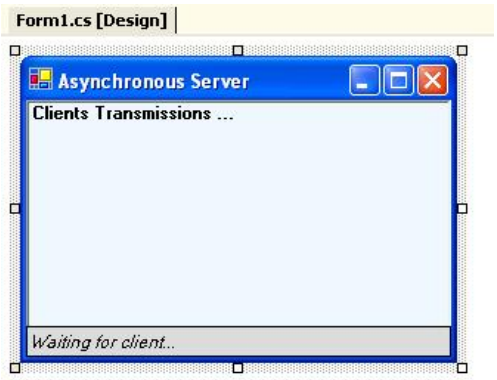
ثانيا: تطبيقات الـAsynchronous Socket في الدوت نت :

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



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

<u>وللبدء قم بإنشاء مشروع جديد كما في الشكل التالي:</u>



سوف نستخدم الـNamespaces التالية:

C#:

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

VB.NET:

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

قم بإضافة التعاريف التالية في الـGlobal Declaration (أي بعد تعريف الـMain Class):

C#:

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

VB.NET:

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

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

في الـForm Load قم بإضافة الكود التالي حيث سنعرف Connection يعتمد على الـTCP ويعمل على الـPort 5020 ثم تعريف عملية قبول الاتصال باستخدام الـBeginAccept :

C#:

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

VB.NET:

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

ثم إنشاء Accept Callback Method والذي سيتم فيه إنهاء الـAccepted Request باستخدام الـEndAccept Method وبعد ذلك إرسال Acknowledgement إلى الـClient تخبره فيها بقبول الطلب وترسل باستخدام الـBeginSend Method كما يلي:

C#:

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

VB.NET:

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

ثم إنشاء Send Callback method لإنهاء الـBeginSend وكما يلي:

C#:

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

VB.NET:

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

ثم إنشاء Receive Callback method لإنهاء الـBeginReceive وكما يلي:

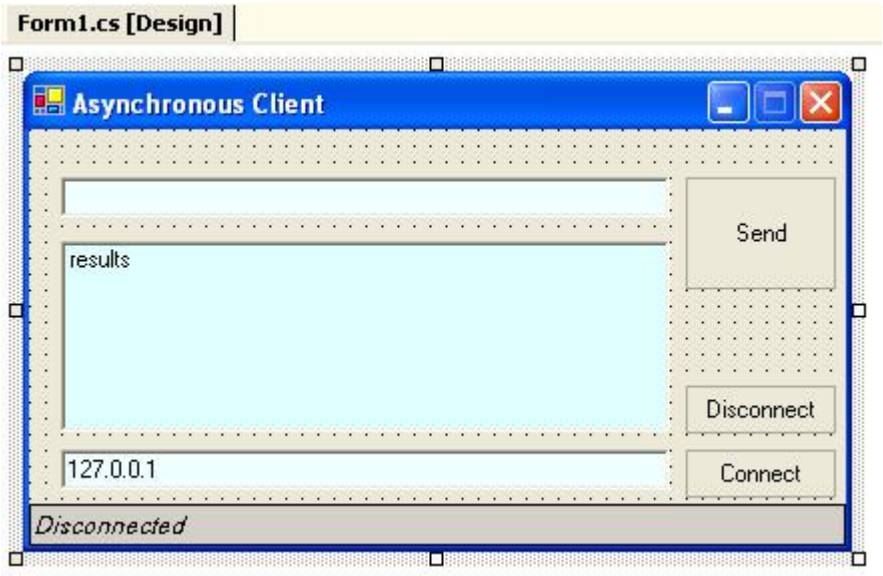
C#:

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

VB.NET:

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

وهنا قد تم الانتهاء من برنامج الـServer والآن سوف نقوم بإنشاء برنامج الـClient وللبدء قم بإنشاء مشروع جديد كما في الشكل التالي:



سوف نستخدم الـNamespaces التالية:

C#:

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

VB.NET:

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

في الـGlobal Declaration (أي بعد تعريف الـMain Class)قم بإضافة التعاريف التالية:

C#:

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

في الـConnect Button قم بكتابة الكود التالي:

C#:

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

VB.NET:

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

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

ثم قم بإنشاء Callback Connect method كما يلي:

C#:

```
void Connected(IAsyncResult iar)
{
client = (Socket)iar.AsyncState;
        try
        {
client.EndConnect(iar);
conStatus.Text = "Connected to: " + client.RemoteEndPoint.ToString();
client.BeginReceive(data, 0, size, SocketFlags.None, new
AsyncCallback(ReceiveData), client);
        }
catch (SocketException)
                {
        conStatus.Text = "Error connecting";
                }
}
```

VB.NET:

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

ثم إنشاء Receive Callback method لإنهاء الـBeginReceive وكما يلي:

C#:

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

VB.NET:

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

ثم إضافة الكود التالي في الـSend Button :

C#:

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

VB.NET:

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

ثم إنشاء Send Callback method لإنهاء الـBeginSend وكما يلي:

C#:

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

}
```

VB.NET:

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

ثم إنشاء Receive Callback method لإنهاء الـBeginReceive وكما يلي:

C#:

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

VB.NET:

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

وكما لاحظنا فإن برنامج الـClient لا يختلف كثيرا عن برنامج الـServer حيث نعرف في الـServer الـSocket Connection والـBeginAccept Method أما في الـClient فنعرف الـSocket Connection و الـBeginConnect Method وتبقى عملية الإرسال والاستقبال هي نفسها في Server والـClient ...

Chapter 9
Advanced Multicasting Systems

- Architecture of Multicast Sockets

- Using Multicast Sockets 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

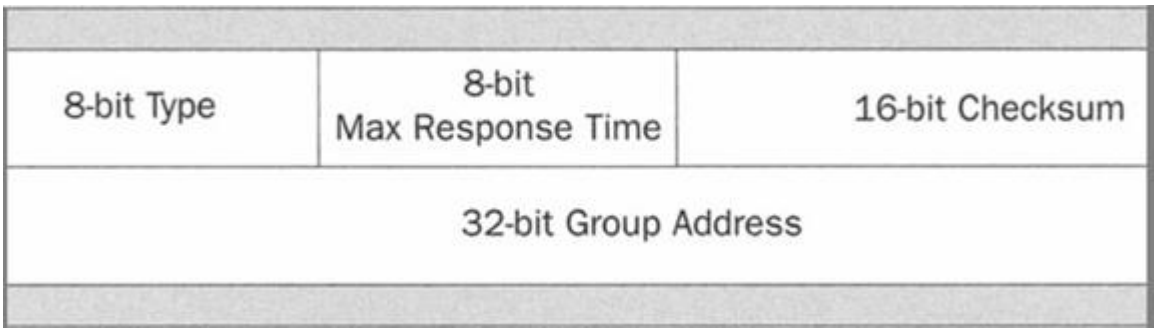
بسم الله الرحمن الرحيم

9: Advanced Multicasting Systems :

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

أولا : Architecture of Multicast Sockets :

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



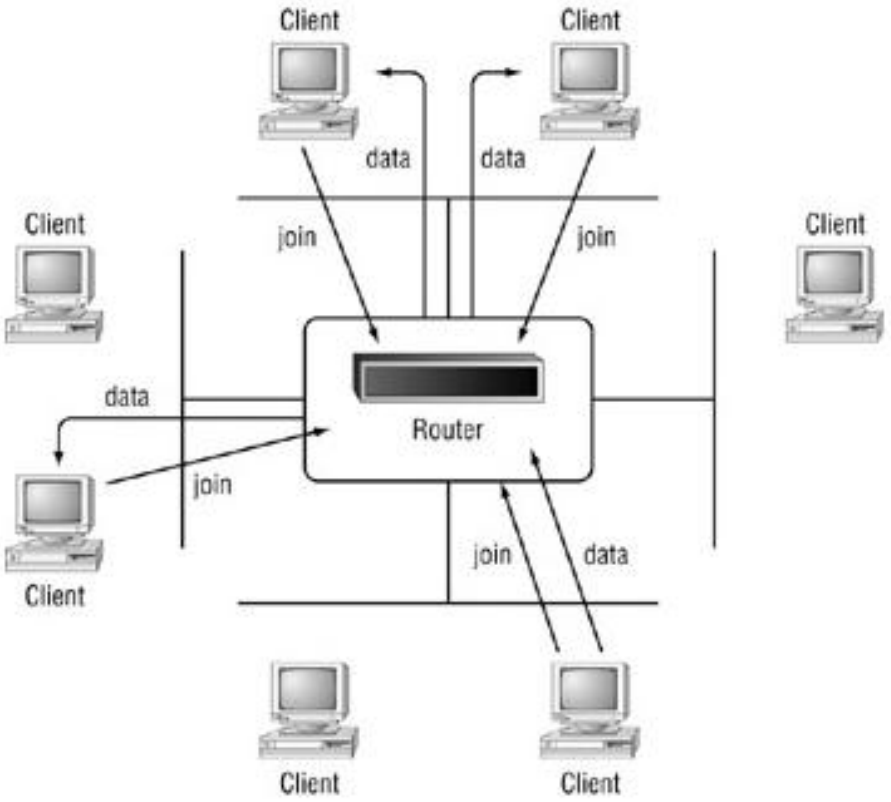
وتم تخصيص الـRange في الـMulticasting من 224.0.0.0 إلى 239.255.255.255 ونستطيع تحديده بثلاثة طرق فإما بشكل يدوي Static أو Dynamic أو على أساس الـScope-Relative وبشكل عام تستخدم هذه التوزيعات كما يلي كمثال:
التخصيص 224.0.0.1 ويستخدم في جميع الشبكات المحلية فقط حيث لا يتم تمريره إلى شبكة أخرى عبر الـRouter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

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

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

ويمكننا استخدام الـMulticasting في ثلاثة أنواع من الشبكات وهي شبكات الـPeer to Peer حيث لا وجود لجهاز Server والكل يستقبل و يرسل من و إلى الـGroup الذي هو فيه، والنوع الثاني Server Based Network حيث يتم إرسال رسالة واحدة إلى الـServer ويقوم

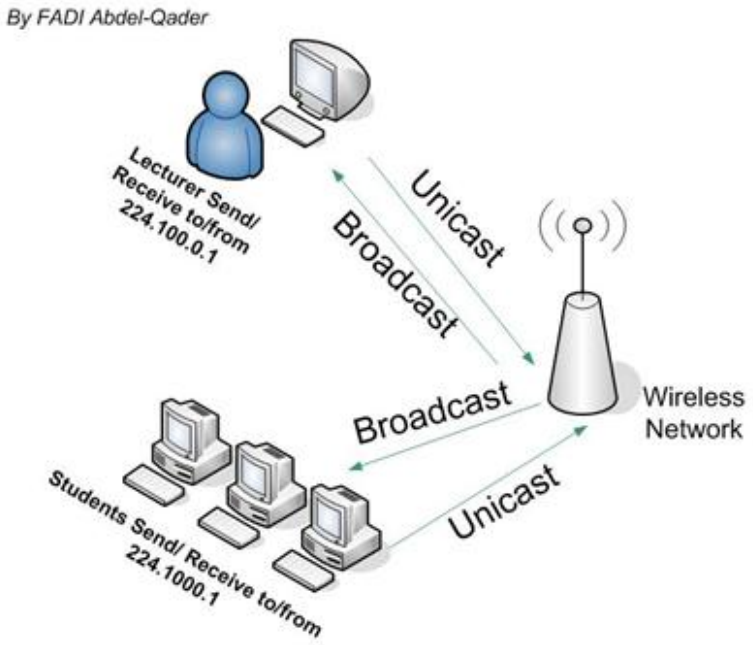
الـServer بتوزيعها على بقية الأجهزة في الشبكة ، أما النوع الثالث فيتم من خلال الـRouter ، وكما يتضح من الشكل التالي فإن عملية الإرسال تتم بعد انضمام الـClient إلى المجموعة التي تملك الـIP Multicast ويرسل الـClient رسالة واحدة إلى الـRouter حيث يقوم الـRouter بتوزيعها على الأجهزة في المجموعة مستخدما الـRouting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



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

1 – الإرسال من واحد إلى مجموعة One to Group:

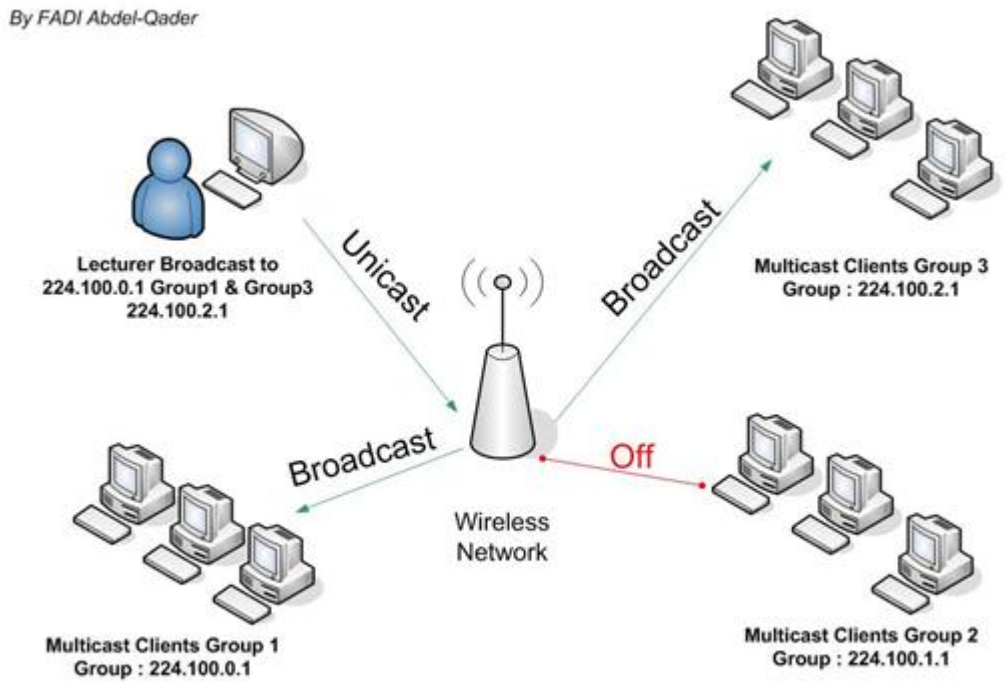
وفيه يملك الـSender User نفس الـIP Multicasting الذي يملكه الـReceiver Users ويتم الإرسال من داخل الـGroup إلى جميع أعضائه حيث ترسل ك Unicast إلى الـAccess Point حيث يقوم بتوزيعها على كافة الأعضاء في المجموعة بأسلوب الـBroadcast وكما في الشكل التالي:



Full Duplex Multicast System for one Group

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

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



Half Duplex Multicast System for Multi-Groups

ثانيا :Using Multicast Sockets with .NET:

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

من العمليات الأساسية في التعامل مع الـMulticasting :

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

لا تلزم عملية الانضمام إلى الـMulticast Group أي عمليات تحقق سوى التصنت على الـport والـIP Multicasting المحدد ، ويتم ذلك بعد تعريف udpClient Object وباستخدام الـ JoinMulticastGroup 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 Methods **JoinMulticastGroup و DropMulticastGroup** لضم أو إلغاء عنوان أو مجموعة من العناوين من الـMulticast Group ، وباستخدام Class MulticastOption: يمكننا تخزين IP Address List لتعامل معها في Multicast Group لعمل Join و Drop لأي Multicast Group وتستخدم كما يلي كمثال لإضافة عضوية لاستقبال رسائل Multicast :

أولا نعرف الـUDP Socket وكما يلي :

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

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

ثانيا نقوم بتعريف Address List ثم نسند إليها الـIP الذي نريد إدخاله في الـGroup أو نجعل الـUser يدخل العنوان بنفسه نربطها بالسكوت باستخدام الميثود Bind وكما يلي :

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

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

ثالثا نقوم بتعريف الـMulticast Option ونسند لها العنوان المحدد كما يلي:

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

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

ومن ثم نضيف التغير على SetSocketOption حيث تأخذ هذه الميثود ثلاثة باروميترات الأول لتحديد مستوى التغيير على IP أو على IPv6 أو على Socket أو TCP أو UDP وفي حالتنا هذه سوف نستخدم التغير على IP إذ ما نريده هو ضم IP إلى Multicast Group وفي الباروميتر الثاني نحدد نوع التغيير حيث نريد إضافة عضوية ويمكن الاختيار بين إضافة عضويه AddMembership أو إلغاء عضوية DropMembership وأخيرا نسند إليه 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 الـIP Multicasting لابد أولا من تعريف الـSocket Object باستخدام الـUDP Connection وإسناد الـIP Multicasting ورقم الـPort إلى الـ Object IPEndPoint ... ونستطيع الإرسال باستخدام الـsendto method حيث نسند لها الـ data as Bytes Array والـIPEndPoint Object وكما يلي لإرسال رسالة نصية:

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

VB.NET:

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

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

C#:

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

VB.NET:

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

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

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

C#:

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

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

VB.NET:

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

```
Dim data As Byte() = sock.Receive(iep)
Dim stringData As String = Encoding.ASCII.GetString(data, 0, data.Length)
listBox1.Items.Add(iep.Address.ToString + " :_ " + stringData)
```

ولاستقبال صورة نستخدم الـmemory Stream لاستقبال البيانات من الـReceive Method وتخزينها في الذاكرة على هيئة Stream Data ثم تحويلها إلى صورة مرة أخرى باستخدام الـimage.FromStream Method وكما يلي:

C#:

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

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

VB.NET:

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

ملاحظات هامة في استخدام الـMulticasting في برمجيات الشبكات :

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

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

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

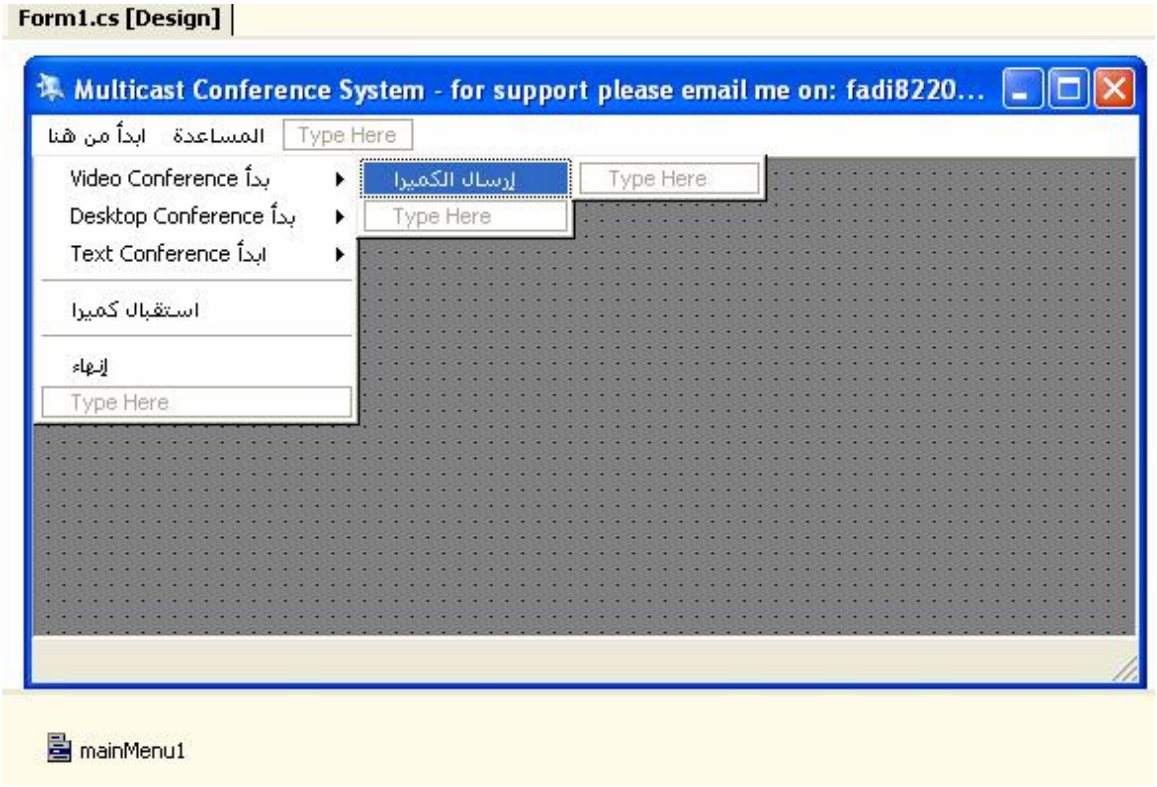
4- في العادة تكون السعة المسموحة لإرسال الـMulticasting Data عبر الـsendto Method محدودة لذلك يمكنك استخدام الـBinary Reader & Writer والـStream Reader & Writer لإرسال والاستقبال بدلا منها ...

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

ثالثا تطبيق مشروع نظام المؤتمرات Multicasting Conferencing Systems:

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

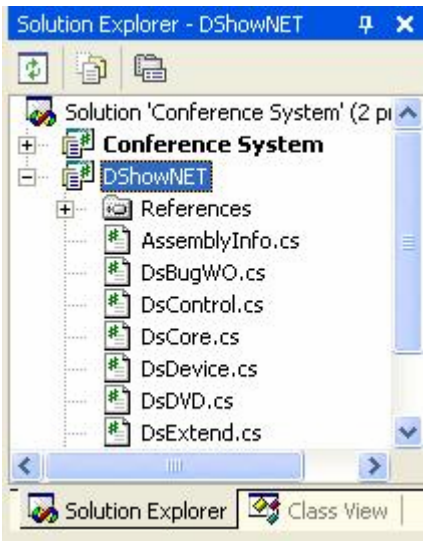
سوف نقوم هنا بتقسيم نظام المؤتمرات إلى ثلاثة أنظمة رئيسية وهي نظام مؤتمرات الفيديو ونظام مؤتمرات سطح المكتب ونظام المؤتمرات النصية، في البداية سوف نقوم بعمل الشاشة الرئيسية للبرنامج و كما في الشكل التالي:



1- Full/Half Duplex Multicast Video Conferencing System :

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

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



وحتى نتعامل معها سوف نستدعيها باستخدام :

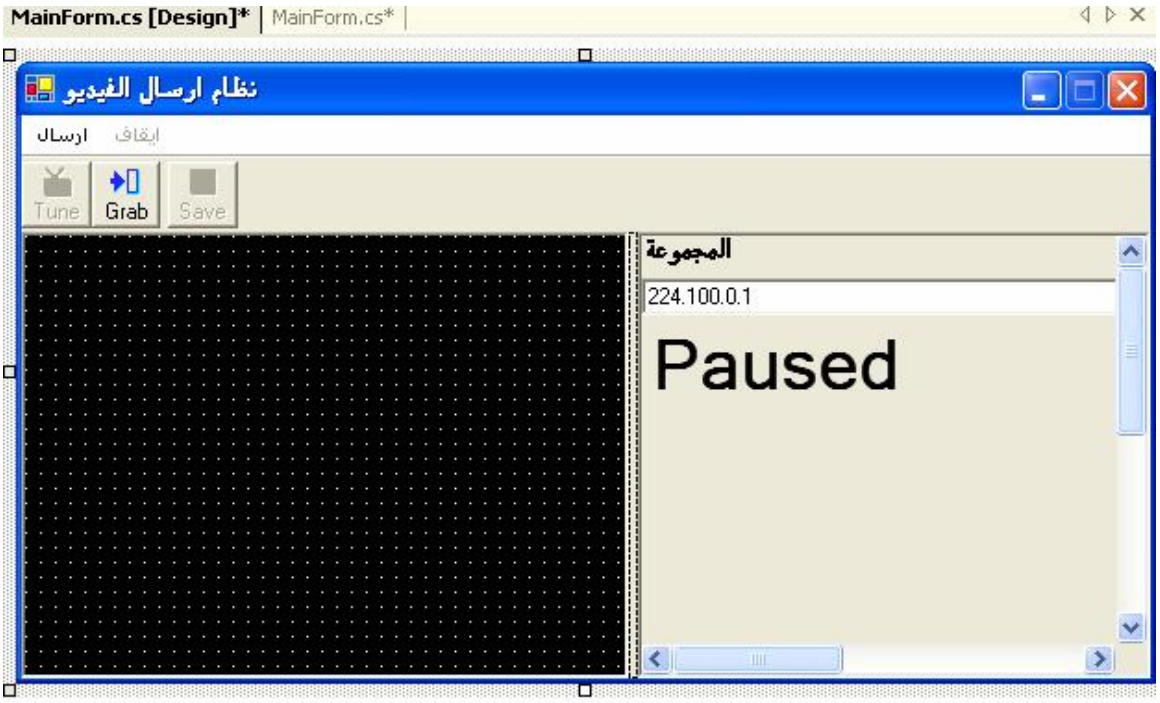
C#:

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

VB.NET:

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

وسيكون شكل برنامج الإرسال عبر الكاميرا كما في الشكل التالي:



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

C#:

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

VB.NET:

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

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

C#:

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

VB.NET:

```
    Private Sub OnCaptureDone()
        Try
Trace.WriteLine("!!DLG: OnCaptureDone")
toolBarBtnGrab.Enabled = True
Dim hr As Integer
If sampGrabber Is Nothing Then
Return
End If
hr = sampGrabber.SetCallback(Nothing, 0)
Dim w As Integer = videoInfoHeader.BmiHeader.Width
Dim h As Integer = videoInfoHeader.BmiHeader.Height
```

```
If ((w And &H3) <> 0) OrElse (w < 32) OrElse (w > 4096) OrElse (h < 32) OrElse
(h > 4096) Then
Return
End If
Dim stride As Integer = w * 3
Dim handle As GCHandle = GCHandle.Alloc(savedArray, GCHandleType.Pinned)
Dim scan0 As Integer = CInt(handle.AddrOfPinnedObject())
scan0 += (h - 1) * stride
Dim b As Bitmap = New Bitmap(w, h, -stride, PixelFormat.Format24bppRgb, New
IntPtr(scan0))
handle.Free()
savedArray = Nothing
Dim old As Image = pictureBox.Image
pictureBox.Image = b
If Not old Is Nothing Then
old.Dispose()
End If
toolBarBtnSave.Enabled = True
Catch e1 As Exception
End Try
End Sub
```

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

C#:

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

VB.NET:

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

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

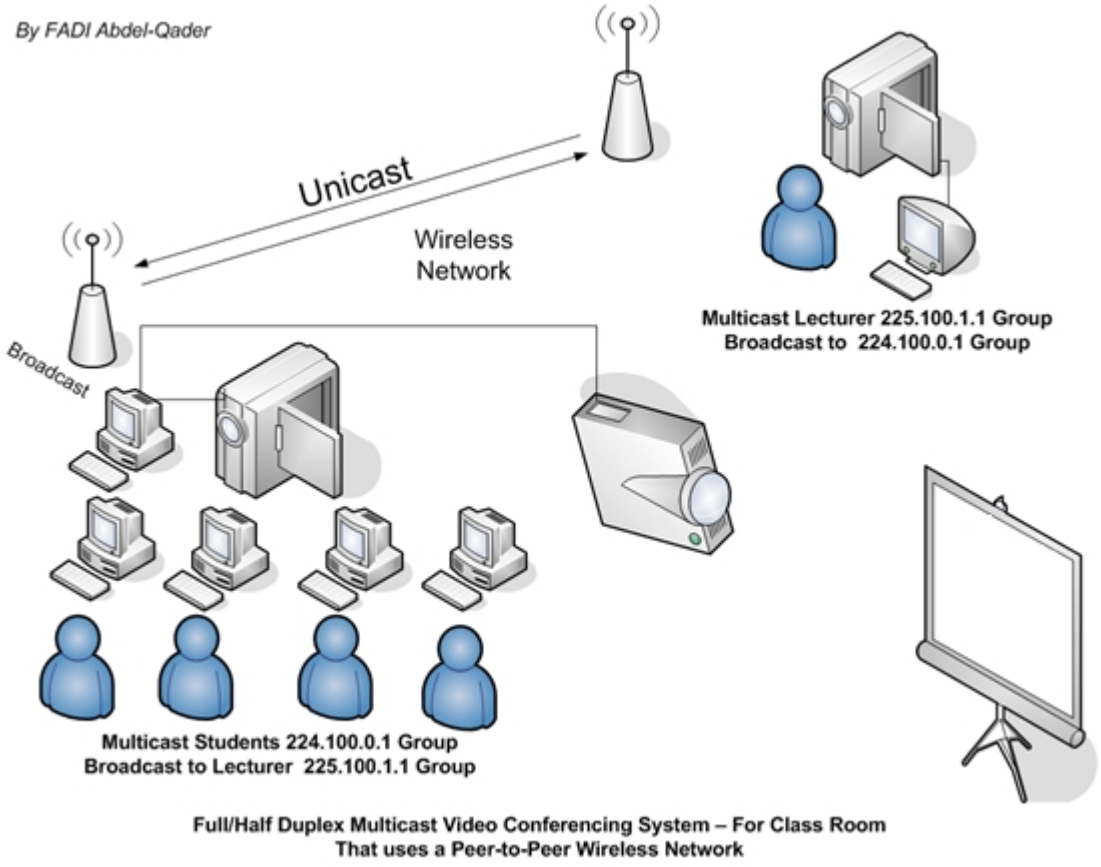
C#:

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

VB.NET:

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

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



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

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

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

سوف نستخدم الـ Namespaces التالية لاستقبال الصورة من الـMulticast Group :

C#:

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

VB.NET:

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

ثم قم بكتابة method الاستقبال كما يلي:

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

        byte[] data = sock.Receive(ref iep);
```

```
        MemoryStream ms = new MemoryStream(data);
        pictureBox1.Image = Image.FromStream(ms);
        sock.Close();
}
```

VB.NET:

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

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

C#:

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

VB.NET:

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

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

C#:

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

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

VB.NET:

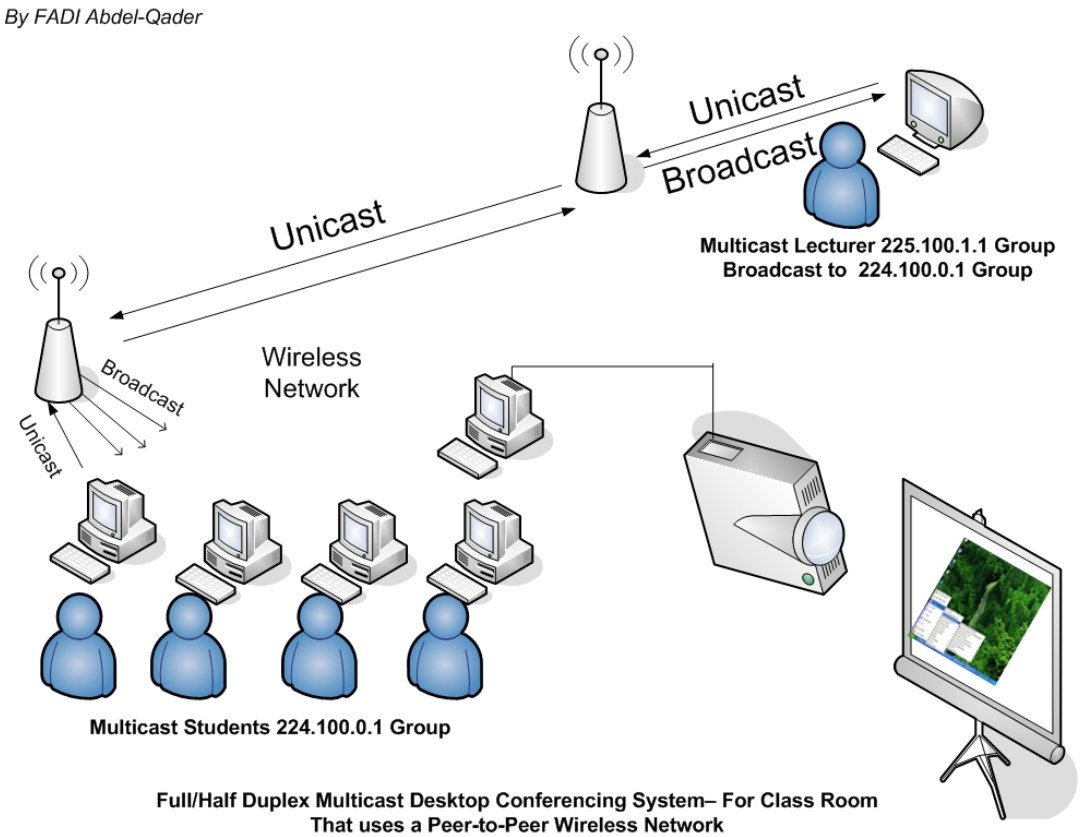
```
Try
 saveFileDialog1.Filter = "JPEG Image (*.jpg)|*.jpg"
 If saveFileDialog1.ShowDialog = DialogResult.OK Then
Dim mypic_path As String = saveFileDialog1.FileName
   pictureBox1.Image.Save(mypic_path)
 End If
Catch generatedExceptionVariable0 As Exception
End Try
```

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

:Full/Half Duplex Multicast Desktop Conferencing System -2

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

وهنا مخطط عمل البرنامج :



وكما نلاحظ من الشكل التالي فإن الأستاذ يقوم بشرح المحاضرة على جهازه الشخصي ويرسل الصورة إلى الطلاب وكما نلاحظ أيضا فإن هذه العملية هي أحادية الاتجاه وكما يمكن جعلها باتجاهين Half Duplex || Full لاكن لابد من إنشاء مجموعة جديدة لعملية الإرسال

من الطالب إلى الأستاذ حيث يعرض الأستاذ محاضرته ويرسلها إلى مجموعة الطلاب ويستطيع أحد الطلاب عرض جهازه على الأستاذ إذ يرسل الصورة إلى مجموعة الأستاذ ...

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

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

أولا PlatFormInvokeGDI32.cs لالتقاط صورة سطح المكتب باستخدام الـ+GDI والـAPI:

C#:

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

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

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

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

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

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

```
public static extern IntPtr CreateCompatibleBitmap(IntPtr hdc,
int nWidth, int nHeight);
[DllImport ("gdi32.dll",EntryPoint="CreateCompatibleDC")]
public static extern IntPtr CreateCompatibleDC(IntPtr hdc);
[DllImport ("gdi32.dll",EntryPoint="SelectObject")]
public static extern IntPtr SelectObject(IntPtr hdc,IntPtr bmp);
#endregion

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

VB.NET:

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

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

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

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

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

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

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

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

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

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

ثانيا PlatformInvokeUSER32.cs إذ سوف نستخدمها مع الـClass السابق لالتقاط صورة سطح المكتب باستخدام الـuser32 API :

C#:

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

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

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

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

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

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

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

                #endregion

                #region Public Constructor
                public PlatformInvokeUSER32()
```

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

VB.NET:

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

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

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

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

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

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

        <DllImport("user32.dll", EntryPoint:="ReleaseDC")> _
        Public Shared Function ReleaseDC(ByVal hWnd As IntPtr, ByVal hDc As
IntPtr) As IntPtr
        End Function
#End Region
#Region "Public Constructor"
        Public Sub New()
        End Sub
#End Region
    End Class
```

```
    'This structure shall be used to keep the size of the screen.
    Public Structure SIZE
        Public cx As Integer
        Public cy As Integer
    End Structure
End Namespace
```

ثالثا: CaptureScreen.cs والتي سوف نستخدمها بشكل مباشر في البرنامج حيث يتعامل مع ال PlatFormInvokeGDI32 Class والـPlatformInvokeUSER32 Class :

C#:

```
using System;
using System.Drawing;

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

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

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

VB.NET:

```
Imports System
Imports System.Drawing

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

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

```
            'We create a compatible bitmap of screen size using screen device
context.
            hBitmap = PlatformInvokeGDI32.CreateCompatibleBitmap(hDC, size.cx,
size.cy)
            'As hBitmap is IntPtr we can not check it against null. For this purspose
IntPtr.Zero is used.
            If Not hBitmap.Equals(IntPtr.Zero) Then
                'Here we select the compatible bitmap in memory device context and
keeps the reference to Old bitmap.
                Dim hOld As IntPtr =
CType(PlatformInvokeGDI32.SelectObject(hMemDC, hBitmap), IntPtr)
                'We copy the Bitmap to the memory device context.
                PlatformInvokeGDI32.BitBlt(hMemDC, 0, 0, size.cx, size.cy, hDC, 0, 0,
PlatformInvokeGDI32.SRCCOPY)
                'We select the old bitmap back to the memory device context.
                PlatformInvokeGDI32.SelectObject(hMemDC, hOld)
                'We delete the memory device context.
                PlatformInvokeGDI32.DeleteDC(hMemDC)
                'We release the screen device context.

PlatformInvokeUSER32.ReleaseDC(PlatformInvokeUSER32.GetDesktopWindow(),
hDC) 'Image is created by Image bitmap handle and stored in local variable.
                Dim bmp As Bitmap = System.Drawing.Image.FromHbitmap(hBitmap)
                'Release the memory to avoid memory leaks.
                PlatformInvokeGDI32.DeleteObject(hBitmap)
                'This statement runs the garbage collector manually.
                GC.Collect() 'Return the bitmap
                Return bmp
            End If 'If hBitmap is null return null.
            Return Nothing
        End Function
#End Region
    End Class
End Namespace
```

وحتى نستطيع التحكم في حجم الصورة سوف نكتب الـmethod التالية:

C#:

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

VB.NET:

```
Public Function ResizeBitmap(ByVal b As Bitmap, ByVal nWidth As Integer, ByVal
nHeight As Integer) As Bitmap
    Dim result As Bitmap = New Bitmap(nWidth, nHeight)
    ' Using
```

```
    Dim g As Graphics = Graphics.FromImage(CType(result, Image))
    Try
        g.DrawImage(b, 0, 0, nWidth, nHeight)
    Finally
        CType(g, IDisposable).Dispose()
    End Try
    Return result
End Function
```

سوف نستخدم الـNamespaces التالية في البرنامج لتعامل مع الـMulticasting :

C#:

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

VB.NET:

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

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

C#:

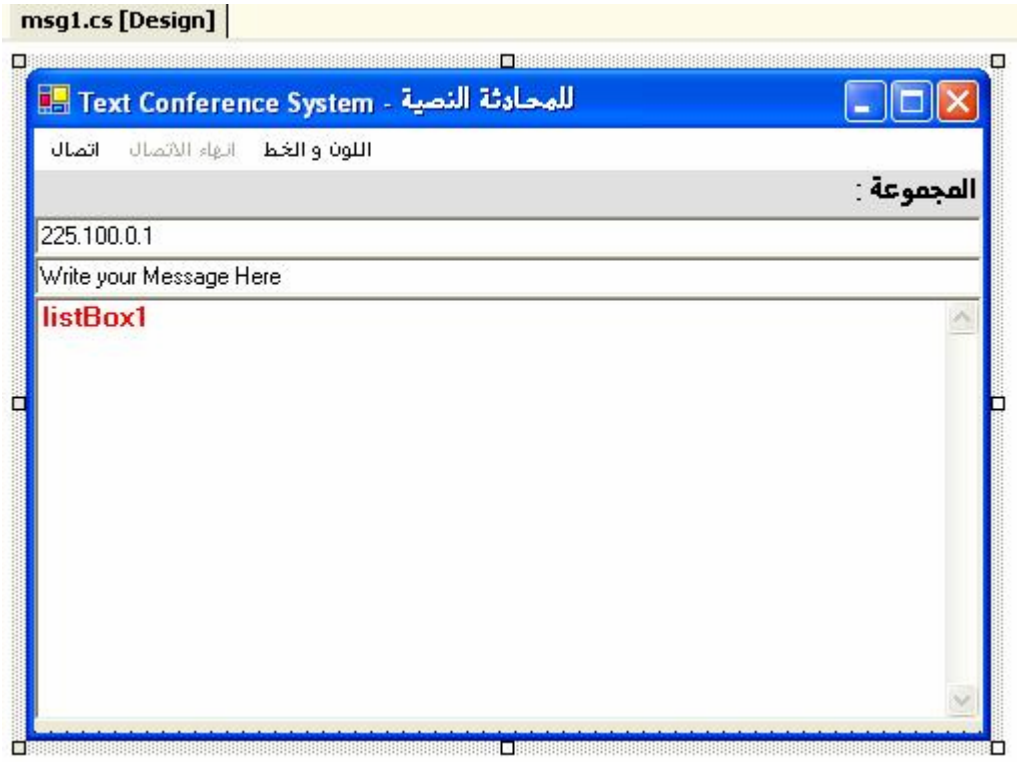
```
 Bitmap bt = new Bitmap(CaptureScreen.GetDesktopImage());
picScreen.Image = ResizeBitmap(bt, 600, 400 );
MemoryStream ms = new MemoryStream();
picScreen.Image.Save(ms,System.Drawing.Imaging.ImageFormat.Jpeg);
byte[] arrImage = ms.GetBuffer();
ms.Close();
Socket server = new Socket(AddressFamily.InterNetwork,
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.GetDesktopImage)
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 الـNamespaces التالية:

C#:

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

VB.NET:

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

سوف نستخدم الـmethod التالية لإجراء عملية الإرسال حيث سترسل الرسالة عند الضغط على الـEnter بعد كتابة الرسالة في الـTextbox المخصص :

C#:

```
private void msg_KeyPress(object sender,
System.Windows.Forms.KeyPressEventArgs e)
{if(e.KeyChar == '\r'){
try{
```

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

VB.NET:

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

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

C#:

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

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

VB.NET:

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

ولاستدعائها لابد من استخدام الـThreading ، قم بعمل Timer واستدعي فيه الـmethod السابقة باستخدام الـThread وكما يلي:

C#:

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

VB.NET:

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

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

C#:

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

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

VB.NET:

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

قمنا بهذا الفصل بتعرف على كيفية عمل Multicast Conference Systems في بيئة الدوت نيت وطرق إنشاء الـ Video Conference System وغيره .

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 في بيئة الدوت نيت.

Chapter 10
Voice Over IP Programming

- The Concept & Requirements of Voice Communication Systems

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

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

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

بسم الله الرحمن الرحيم

10 Voice Over IP Programming :

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

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

أولا : The Requirements of Voice Communication Systems

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

– متطلبات نقل الصوت المثلى :

1 - أسلوب النقل Stream

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.

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

أولا بروتوكول ال TCP : يدعم بروتوكول ال TCP كل هذه الأمور وبكفاءة عالية لاكن المشكلة الوحيدة في هذا البروتوكول هو عدم إمكانية استخدامه لعمل Multicast Conference System إذ انه من المعروف أن ال Multicasting وال Broadcasting من الأمور الخاصة ببروتوكول ال UDP ولا يدعم ال TCP أي من هذه الأمور وهو ما بينته في الفصل السابق، إذ يعتر الـ TCP بروتوكول موجه Oriented Protocol لذلك لا يمكن الاعتماد عليه في حالة حاجتنا لعمل Multicast Conference System أو في حالة البث الإذاعي Broadcasting .. إذا الحل الأخر والوحيد هو بروتوكول الـ UDP في حالة حاجتنا لهذه الأمور.

ثانيا بروتوكول ال UDP : لا يعتبر هذا البروتوكول حل جيد لعملية نقل الصوت بالكفاءة العالية إذ أنه لا يدعم عمليةDelivered on Sequence وهو ما سبب من استحالة عمل Fragmented للـ Packets المرسل ومن المعروف أن حجم Ethernet Encapsulation لا يزيد عن 1500 KB للـ Packet الواحد وهو الحجم الأقصى للـ Datagram Encapsulation الخاص بالـ Ethernet لذلك في حالة قمنا بعمل Fragmented 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 ال UDP منها منظمة الـ ITU- International Telecommunications Union بتقديمها أسلوب النقل H.323 ومنظمة IETF Internet Engineering Task Force بتقديمها أسلوب النقل RTP – Real Time Transport Protocol ، حيث أضاف هذا المعيار تحسينات على بروتوكول ال UDP لعملية نقل الصوت في الزمن الحقيقي إذ يستخدم أسلوب ال Stream المستخدم في TCP لاكن تحت منصة ال UDP وقد حل هذا المعيار بعض

هذه المشاكل لاكن ليس جميعها ، إذ أن الحاجة أصبحت ملحة لوجود بروتوكول يدعم عملية النقل وفق الترتيب الصحيح Delivered on Sequence بالإضافة إلى دعم الاتصال ك Stream ودعم للـ IP Multicasting وال Broadcasting ، وكان الحل بإنشاء بروتوكول أخر وهو الـ SCTP Stream Control Transmission Protocol – لاكن المشكلة أن منصة Windows لا تدعم هذا البروتوكول وقد تم دعمه بشكل كامل في نظام التشغيل Linux ، كما وعدت Microsoft بدعم هذا البروتوكول في الإصدار التالي من نظام التشغيل Windows والذي سآتي على شرحه في الجزء التالي من هذا الفصل.

ثانيا: The Concept Of Voice Communication :

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

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

2- في الـ Transport Layer ، وهو من أهم الأمور التي يجب أخذها بعين الاعتبار إذ أن المفاضلة بين اختيار بروتوكول الـ UDP أو الـTCP تعتمد على مدى الحاجة التي نريدها ودقة الصوت من جهة أخرى إذ أن أفضل طريقة لنقل الصوت هي استخدام تقنيات الـStream لاكن من المعروف أن بروتوكول الـUDP لا يدعم عملية النقل ك 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 KB للـ Packet الواحد ولحل هذه المشكلة سوف نلجأ إلى تبني بعض التقنيات الجديدة والتي تعتمد على بروتوكول الـ UDP وحتى يتم نقل الصوت ك Stream ومنها أسلوب النقل H.323 والذي ذكرته سابقا ، لاكن لم تدعم الدوت نيت أي من هذه التقنيات ، لذلك عندما نريد نقل صوت من جهاز إلى آخر ك Stream لابد لنا من استخدام بروتوكول الـ TCP لاكن كما ذكرنا سابقا فإنه لا يدعم الـ IP Multicasting و الـ Broadcasting

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

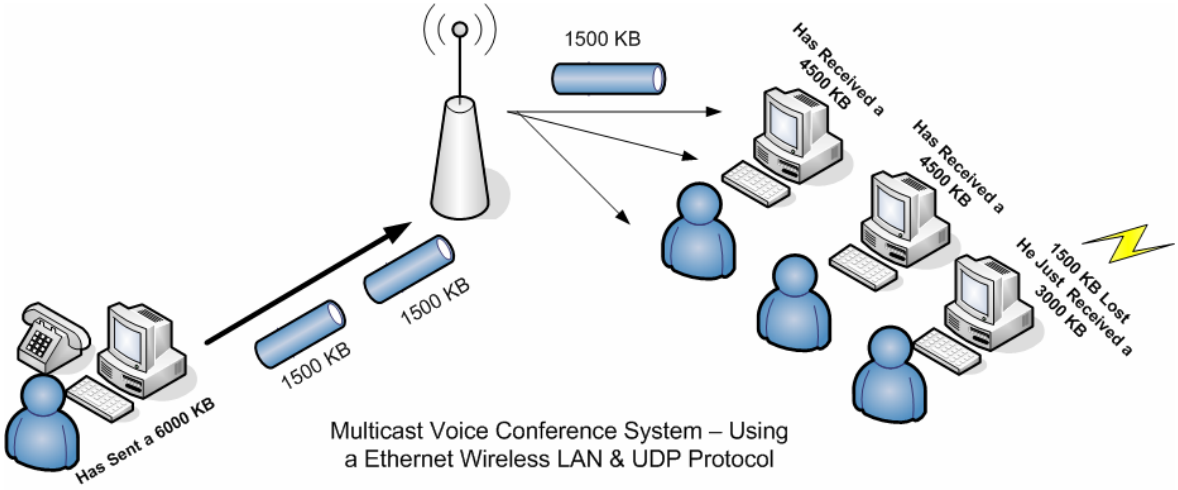
4- في الـ Data Link Layer سيتم تحديد طبيعة ولإرسال سواء باستخدام الـ Ethernet أو غيره وفي هذه الحالة سيتم الاعتماد على الـ Ethernet لاكن مشكلته كما ذكرتها سابقا بمحدودية حجم الـ Frame إذ لا تتجاوز الـ 1500 KB

5- في الـ Physical Layer طبعا المشاكل التي قد تحدث إثناء عملية النقل كثيرة جدا وقد يحدث تأخير Delay لسبب أو لآخر أو قد تضيع بعض الـ Bits إثناء الإرسال لذلك لابد من وجود بروتوكولات تدعم التصحيح لكل هذه المشاكل والتي قد تحدث إثناء عملية الإرسال.

يستقبل الطرف المقابل الـ Bits من طبقة الـ Physical Layer وتمر عبر الـ Data Link Layer ومن ثم الـ Network Layer وفي إثناء هذه المرحلة فإن مستقبل الـ Packets قد يكون هو الشخص المعني في حالة كان أسلوب البث Unicast أو قد يكون جزء من مجموعة الاستقبال Multicast أو قد يكون من ضمن الشبكة التي تم الإرسال لها ك Broadcast لذلك في حالة كونه جزء من مجموعة فإن جهة الإرسال غير معنية بالجهة التي سوف تستقبل الـ 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 إثناء هذه المرحلة منها ضياع جزء من الـ Packets المرسل والذي سوف يسبب وصول الصوت

بشكل متقطع ، وطبعا سوف يكون الاعتماد في هذه الحالة على بروتوكولات الطبقة الأعلى وهي هنا Transport Layer فإذا كان المرسل والمستقبل يستخدم الـ TCP فإن كل هذه المشاكل سوف تحل لاكن المشكلة تكمن في كونه يستخدم الـ UDP حيث لا يوجد حل إلا بإتباع معيار مساند يضمن وصول كافة الـ Packets بترتيب الذي أرسل عليه وبدون ضياع أجزاء من الـ Packets المرسل.

الشكل التالي يوضح عملية ضياع بعض الـ Packets إثناء الإرسال باستخدام شبكة Ethernet Wireless LAN و UDP IP Multicasting مما سوف يسبب تقطيع في الصوت:



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

كما قلنا سابقا فإن الدوت نيت لم تدعم أي من عمليات التقاط وعرض الصوت ، لاكن لإجراء هذه العمليات لابد من استخدام مجموعة 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 بارومترات.

waveInAddBuffer وتستخدم لتخزين الـBits الواردة من جهاز الإدخال في Byte Array Buffer وتأخذ هذه الـMethod ثلاثة بارومترات وهي:

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

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

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

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

C#:

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

VB.NET

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

```
End Function
<DllImport(winmm.dll)> _
Public Shared Function waveInStart(ByVal hwi As IntPtr) As Integer
End Function
<DllImport(winmm.dll)> _
Public Shared Function waveInStop(ByVal hwi As IntPtr) As Integer
End Function
```

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

C#:

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

VB.NET

```
<DllImport(winmm.dll)> _
Public Shared Function waveOutGetNumDevs() As Integer
End Function
<DllImport(winmm.dll)> _
Public Shared Function waveOutPrepareHeader(ByVal hWaveOut As IntPtr, ByRef
lpWaveOutHdr As WaveHdr, ByVal uSize As Integer) As Integer
End Function
<DllImport(winmm.dll)> _
```

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

وحتى نتمكن من عرض محتويات الـ Buffer نستخدم التعريف التالي:

C#:

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

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

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

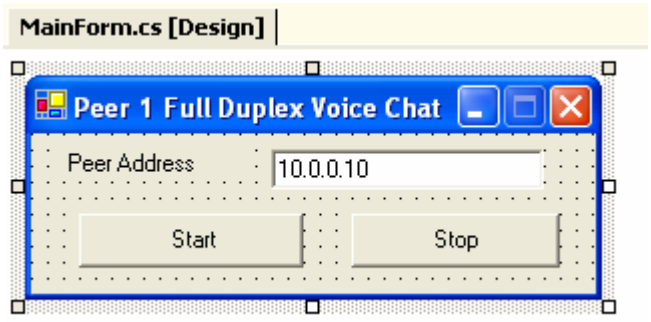
```
    Dim iep As IPEndPoint = New IPEndPoint(IPAddress.Any, 0)
    Dim voice_Come As Byte() = sock.Receive(iep)
    Winmm.PlayWavResource(voice_Come)
    sock.Close()
End Sub
```

البدء بإنشاء برنامج المحادثة الصوتية Voice Chat System :

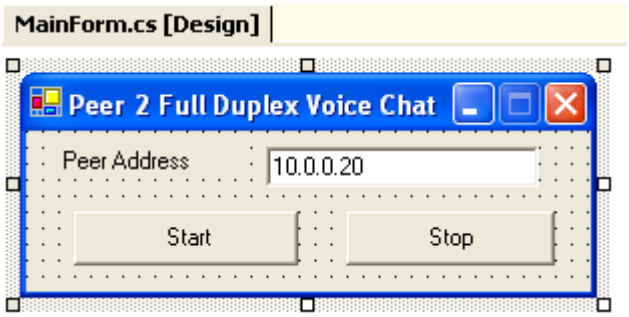
سوف نجزئ عملية التقاط الصوت وتخزينه في الـ Buffer ثم عرضه مرة أخرى في مجموعة من الـClasses وهو تقسيم تم استخدامه في الكثير من البرمجيات الخاصة ب Microsoft ومنها برنامج Windows Sound Recorder وسوف نجمع هذه الـ Classes في ملف واحد نسميه Voice Library وسوف أرفق محتويات هذه الـ Classes في ملحقات هذا الفصل ، وهذه الـ Classes هي:

WaveIn Class وسوف نستخدمه لوضع كافة الـ Methods الخاصة بالتقاط الصوت وتخزينه في Byte Array
WaveOut Clasas وسوف نستخدمه لعرض الصوت الأتي من الـ Buffer ثم عرضه
WaveStream Class والذي سوف نستخدمه لتحويل الصوت إلى Stream حيث يسهل إرساله عبر الشبكة ويشبه عمله عمل MemoryStream المستخدمة في الدوت نيت
الميثود FifoStream لتنظيم الـ Stream بحيث يتم عرض الداخل أولا خارج أولا
الميثود WaveNative ويتم فيها وضع كافة التعريفات للـ Methods الخاصة بالملف winmm.dl والتي شرحناها سابقاً.

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



وهنا صورة برنامج الطرف المقابل :



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 سنقوم الملف باستخدام الـ Using وكما يلي:

C#:

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

VB.NET

```
Imports System.Net
Imports System.Net.Sockets
Imports System.Threading
Imports Voice
```

ثم نقوم بتعريف الـSocket والـThread والذي سوف نستخدمه في البرنامج ويفضل وضع هذه التعريفات في بداية البرنامج أي بعد تعريف الـ Class الرئيسي والهدف من هذه العملية هي القدرة على إغلاق الـ Socket 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 الـClasses السابقة ونعرف الـBuffer الذي سيتم تسجيل الصوت المراد إرساله والـBuffer الذي سيتم عرض الصوت المستلم من الـSocket

C#:

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

VB.NET

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

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

C#:

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

VB.NET

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

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

C#:

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

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

VB.NET

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

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

حيث يتم استقبال الصوت من الشبكة باستخدام الـReceive Method ثم نمرر الصوت المستقبل إلى الـ m_Fifo.Write Method وحتى يتم تنفيذه وتحويله إلى صوت مرة أخرى.
أما الـMethod التي تقوم بتسجيل الصوت وإرساله إلى الجهاز الأخر فهي:

C#:

```
private void Voice_Out(IntPtr data, int size)
{
//for Recorder
if (m_RecBuffer == null || m_RecBuffer.Length < size)
m_RecBuffer = new byte[size];
System.Runtime.InteropServices.Marshal.Copy(data, m_RecBuffer, 0, size);
//Microphone ==> data ==> m_RecBuffer ==> m_Fifo
socket.SendTo(m_RecBuffer, new
IPEndPoint(IPAddress.Parse(Peer_IP.Text),5030));
}
```

VB.NET

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

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

في زر البدء يتم تنفيذ الميثود التالية:

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 الميثود التالية:

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
```

الميثود التي تقوم بعرض الـ Voice Buffer والمستلم من Socket على السماعة:

C#:

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

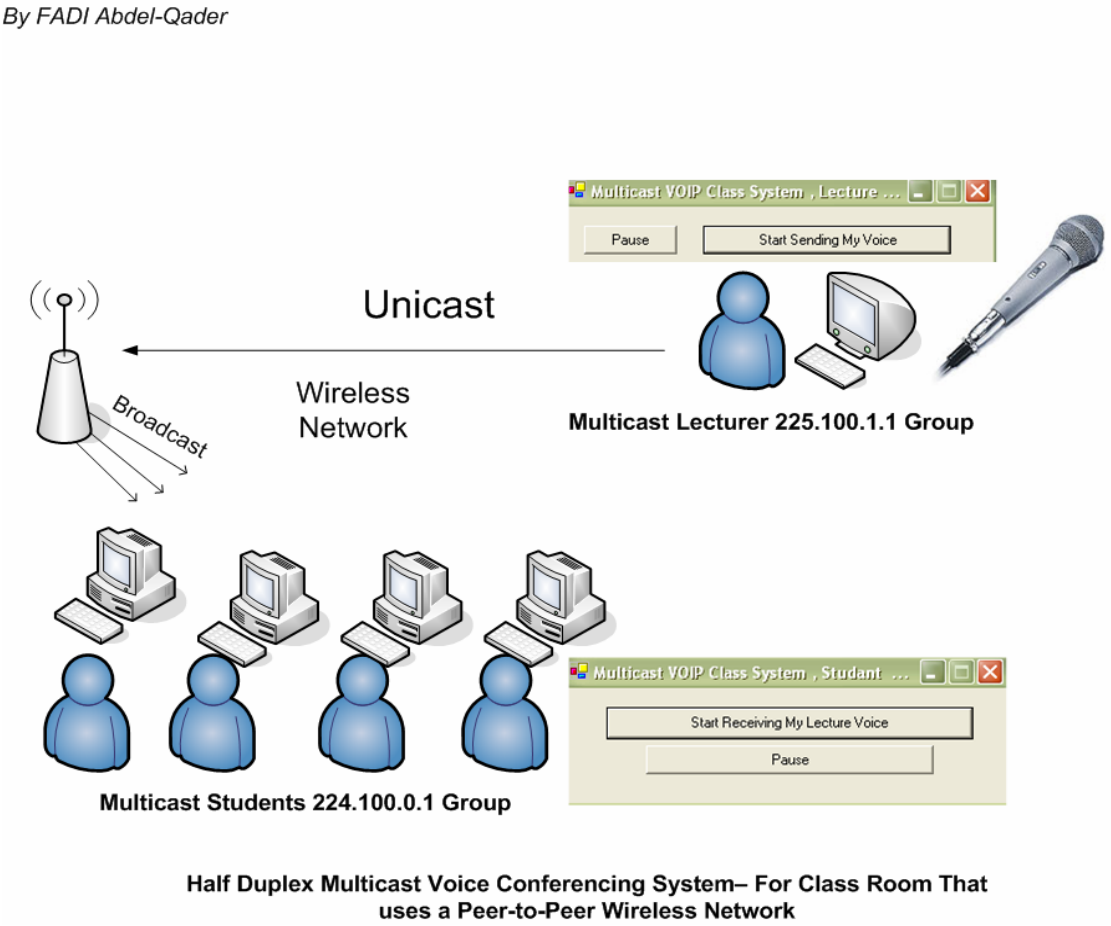
VB.NET

```
Private Sub Filler(ByVal data As IntPtr, ByVal size As Integer)
    If m_PlayBuffer Is Nothing OrElse m_PlayBuffer.Length < size Then
        m_PlayBuffer = New Byte(size - 1) {}
    End If
    If m_Fifo.Length >= size Then
        m_Fifo.Read(m_PlayBuffer, 0, size)
    Else
        Dim i As Integer = 0
        Do While i < m_PlayBuffer.Length
            m_PlayBuffer(i) = 0

            i += 1
        Loop
    End If
    System.Runtime.InteropServices.Marshal.Copy(m_PlayBuffer, 0, data, size)
    ' m_Fifo ==> m_PlayBuffer==> data ==> Speakers
End Sub
```

رابعا: Testing TCP,UDP and Thinking in SCTP to Transfer Voice Throw 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 الـIP Multicast Group ومن ثم الاستقبال من خلاله وكما يلي:

C#:

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

VB.NET

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

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

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

C#:

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

VB.NET

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

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

C#:

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

VB.NET

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

في الطرف المستقبل نقوم باستخدام الـ Network Stream مرة أخرى لاكن 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

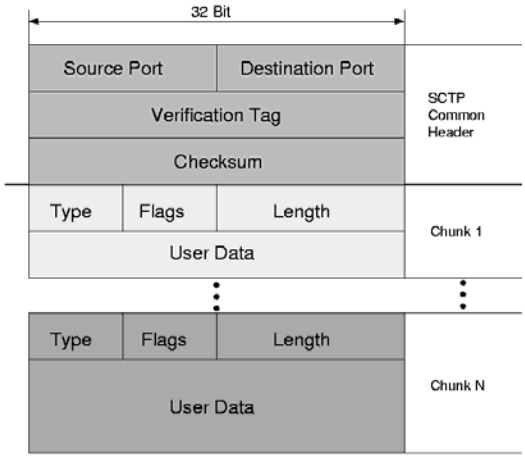
```
Private Sub Voice_In()
    Dim sock As Socket = New Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp)
    sock.Bind(New IPEndPoint(IPAddress.Parse("10.0.0.10"), 5020))
    Dim ns As NetworkStream = New NetworkStream(sock)
    Do While True
        Dim buffer As Byte() = New 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 لاكن المشكلة تكمن في أننا لن نستطيع الاستفادة من هذه الإمكانيات الرائعة في الـMulticasting أو الـBroadcasting ...

والحل الوحيد هو إما استخدام المعايير السابقة مع بروتوكول الـUDP أو استخدام بروتوكول الـTCP أو الانتظار لحين الانتهاء من مشروع Long Horn حتى تدعم Microsoft بروتوكول الـSCTP ...

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

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



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

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

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

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

أولا : الـ Connect Classes والخاصة بعملية الربط:

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

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

C#:

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

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

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

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

VB.NET

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

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

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

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

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

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

C#:

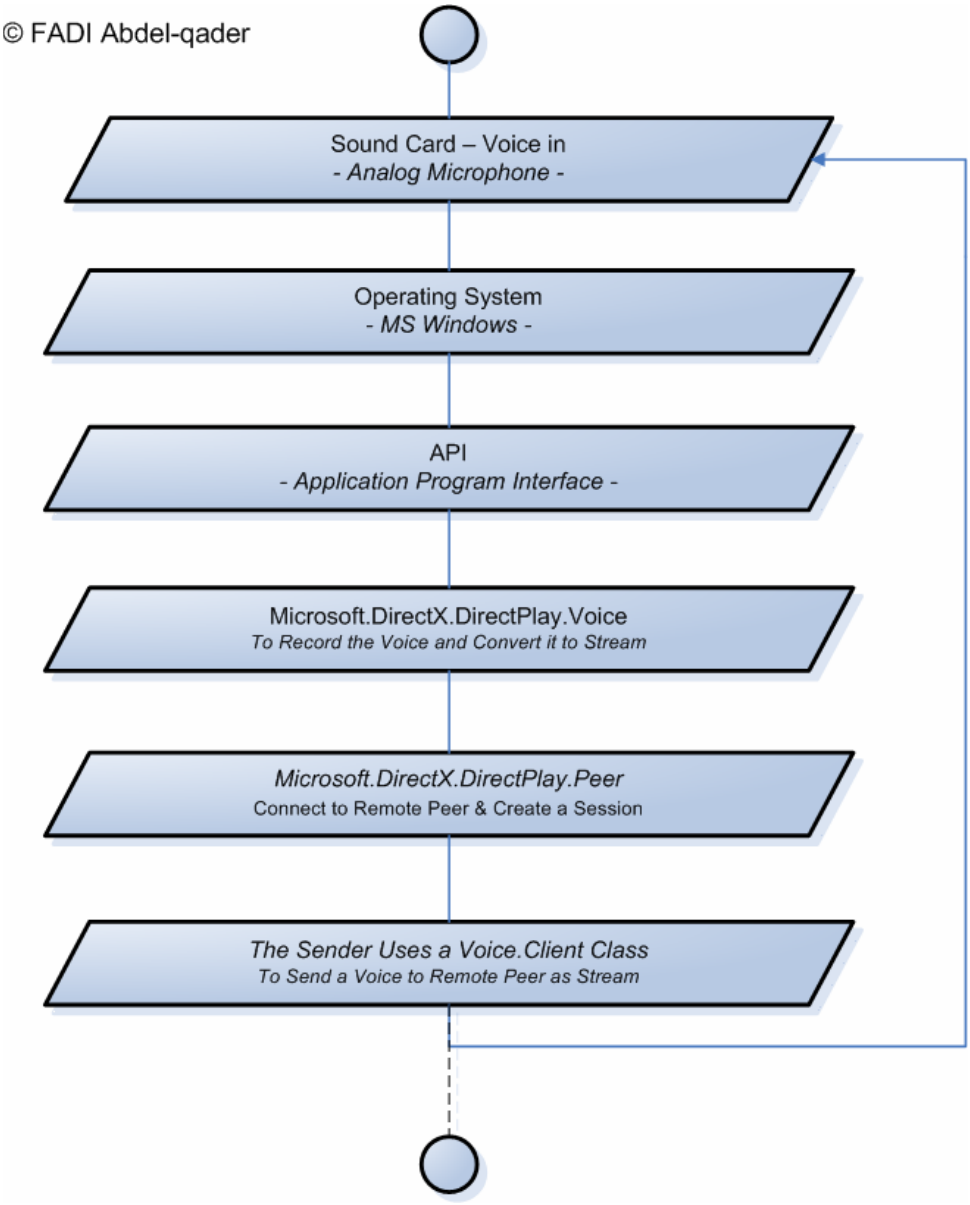
```
using Microsoft.DirectX.DirectPlay;
// Structure for FindHostResponseMessages
public struct HostInfo
{
        public ApplicationDescription appdesc;
        public Address deviceAddress;
        public Address senderAddress;

}
.
.
.
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)
```

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

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



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

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

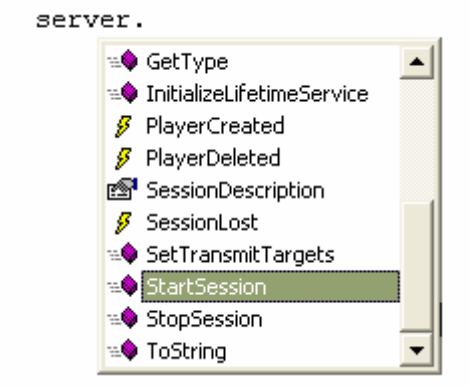
C#:

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

VB.NET

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

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



ولبدأ الجلسة يجب أولا إسناد خصائص الجلسة إلى الـStartSession 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 dxe)
          {
             throw dxe;
          }
       }
       catch(Exception e)
       {
          throw e;
       }
   }
```

VB.NET

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

      'start the session
      Try
         server.StartSession(sessionDesc)
         mIsHost = True
         mInSession = True
      Catch dxe As DirectXException
         Throw dxe
      End Try
   Catch e As Exception
      Throw e
   End Try
End Sub
```

ويتم استدعائها كما يلي:

C#:

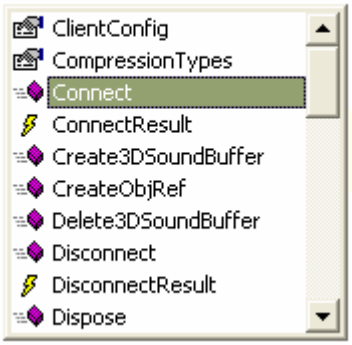
```
CreateVoiceSession(host, Voice.SessionType.Peer,
mConfigForm.CompressionGuid);
```

VB.NET

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

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

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



الميثود Connect وتستخدم لربط مع الـVoice Session حيث يرسل الـ Server رقم الـSession للـ Client وعندها يتمكن الـ Client من الدخول إلى الجلسة ، ويتم ذلك كما يلي:

C#:

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

VB.NET

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

        'get sound device config to check for half-duplex
        soundConfig = client.SoundDeviceConfig
mHalfDuplex = ((soundConfig.Flags And Voice.SoundConfigFlags.HalfDuplex) <>
0)

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

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

وهكذا بينا ملخص عن أهم الطرق لاتصال الصوتي عبر الشبكة وطرق برمجة الـ Voice Chat تحت منصة الدوت نيت واهم ميزات وعيوب بروتوكولات الـ Transport Layer ومدى إمكانياتها لنقل الصوت عبر الشبكة وأخيرا شرح لأهم الـ Classes والمستخدمة في الاتصال الصوتي باستخدام الـ DirectPlay Transport Protocol.

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

ستجد كافة تفاصيل هذا الموضوع في النسخة الورقية من الكتاب
لطلب أ والاستفسار أو التوزيع يرجى الاتصال على احد العناوين التالية

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

الموقع الرسمي للكتاب
www.fadidotnet.org

Part 4
Application Layer Programming

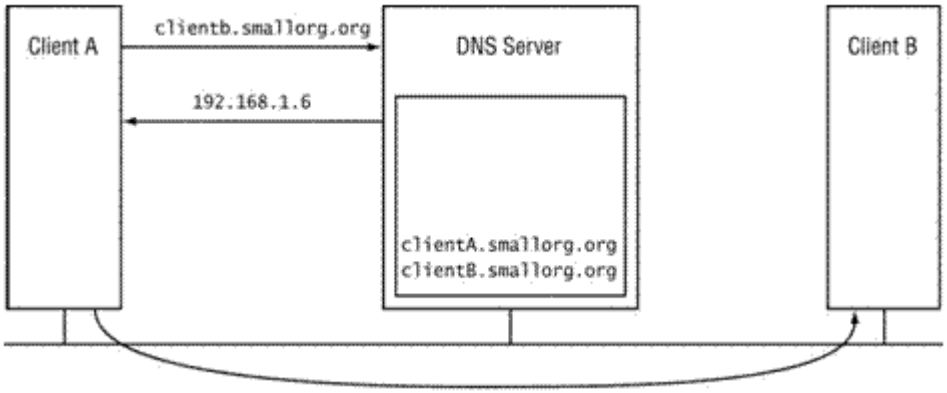
Chapter 11
DNS Programming

- Synchronous DNS Members
- Asynchronous DNS Members

بسم الله الرحمن الرحيم

11: DNS Programming :

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



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

أولا الميثودس المتزامنة Synchronous Methods وهي :

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

C#:

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

VB.NET:

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

الميثود GetHostByName و الميثود GetHostByAddress وتستخدم كل منها كما يلي:

C#:

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

VB.NET:

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

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

C#:

```
using System;
using System.Net;

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

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

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

VB.NET:

```
Imports System
Imports System.Net

Class FMO_DNS

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

ثانيا الميثودس غير المتزامنة Asynchronous Methods :

وتبدأ عادة بكلمة Begin أو End ومن الأمثلة عليها :
BeginGetHostByName و BeginResolve و EndGetHostByName و EndResolve
طبيعة عملها كما هو الحال في الميثودس المتزامنة لكنها تختلف بكون انه لا يشترط تنفيذها لإكمال عمل البرنامج في حين المتزامن لا تسمح بالإنتقال إلى الخطوة الثانية في البرنامج إلا في حالة انتهاء عملها وقد تسبب هذه السيئة بخفض البريفورمانس بشكل عام في البرنامج لذلك ينصح باستخدام الطريقة الغير متزامنة وتستخدم كما يلي : Begin___

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

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

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

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

C#:

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

namespace NetworkIPs
{
        public class Names

        {
                public StringCollection GetNames()

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

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

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

VB.NET:

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


Public Class Names

    Public Function GetNames() As StringCollection

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

```
        Return _machines
    End Function
End Class

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

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

Chapter 12
HTTP Programming

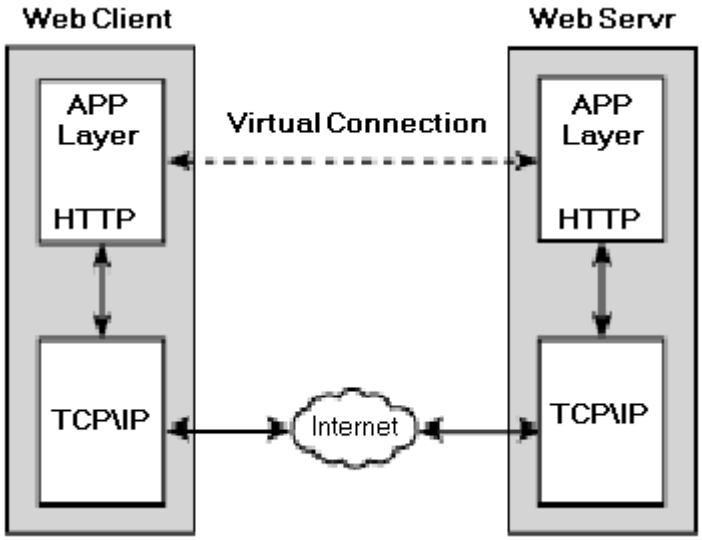
- The Concept of HTTP Protocol
- Using HTTP in Dot Net
- Advanced HTTP Programming
- Using HttpWebRequest
- Using HttpWebResponse

بسم الله الرحمن الرحيم

HTTP – Hyper Text Transfer Protocol Programming :12

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

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



أولا : Downloading From Web Server

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

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

C#:

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

VB.NET:

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

Class DownloadData_Method

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

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

C#:

```
using System;
using System.Net;

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

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

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

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

VB.NET:

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

Class DownloadData_Method

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

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

C#:

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

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

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

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

VB.NET:

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

Class OpenRead_Method

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

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

معلومات هامة عن الـweb host مثل عدد الـHeaders ونوع الـcash control واسم الـServer
و نوع الـEncoding المستخدم وغيرها من المعلومات الهامة، ويستخدم كما يلي كمثال:

C#:

```
using System;
using System.Net;

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

VB.NET:

```
Imports System
Imports System.Net

Class ResponseHeaders_property

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

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

ثانيا : Uploading to Web Server

يدعم الـWebClient أربعة Methods لتحميل البيانات إلى الـWeb Server وهي :

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

C#:

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

VB.NET:

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

Class OpenWrite_method

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

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

C#:

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

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

VB.NET:

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

Class UploadData_Method

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

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

C#:

```
using System;
using System.Net;

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

VB.NET:

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

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

C#:

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

class UploadValues_Method
{

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

VB.NET:

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

Class UploadValues_Method

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

ثالثا :المواضيع الأكثر تقدما في الـHTTP Programming:

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

1- استخدام HttpWebRequest Class :

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

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

C#:

```
using System;
using System.Net;

class ProxyServer_Property
{

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

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

VB.NET:

```
Imports System
Imports System.Net

Class ProxyServer_Property

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

نعرف في البداية الـHttpWebRequest Object ثم نعرف WebProxy Object من الـClass webProxy ونسند له عنوان الـProxy Server ورقم الـ Port وبعد ذلك نستطيع إسناده إلى أي Object باستخدام الخاصية Proxy التي تكون موجودة عادة في جميع HttpWebRequest Objects ..

2- استخدام الـHttpWebrequest لإرسال بيانات إلى الويب Server باستخدام الـStreams وتستخدم كما يلي كمثال:

C#:

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

VB.NET:

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

بعد تعريف الـHttpWebRequest Object نقوم بتعريف Stream Object ونسند له الـ Request من خلال الميثود GetRequestStream .

2 - استخدام HttpWebResponse Class:

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

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

1- **CharacterSet** : وتستخدم لتحديد نوع الـCharacter Set

2- **ContentEncoding** : وتستخدم لعملية الـencoding

3- **ContentLength** : وتستخدم لمعرفة حجم الرد

4- **ContentType** : لتحديد نوع الـRespone

5- **Cookies** : لتعامل مع الـCookies ولستخدامها يجب أولا إنشاء ملف Cookie فارغ وتعريفه كما يلي كمثال:

C#:

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

وذلك قبل الـHTTP Request ثم نسنده إليه كما يلي :

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

VB.NET:

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

6- **Headers** : لمعرفة الـHTTP Headers

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

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

9 – **ProtocolVersion** : لمعرفة الـHTTP Version

10 – **ResponseUri** : الـURL الخاص بـServer

11 – **Server** : لمعرفة اسم الـServer

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

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

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

Chapter 13
Web Services & XML Programming

- Introduction to Web services & XML
- Create A Simple Web Service Application
- Advanced Remotting & Web Services Programming

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

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

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

C#:

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

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

VB.NET:

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

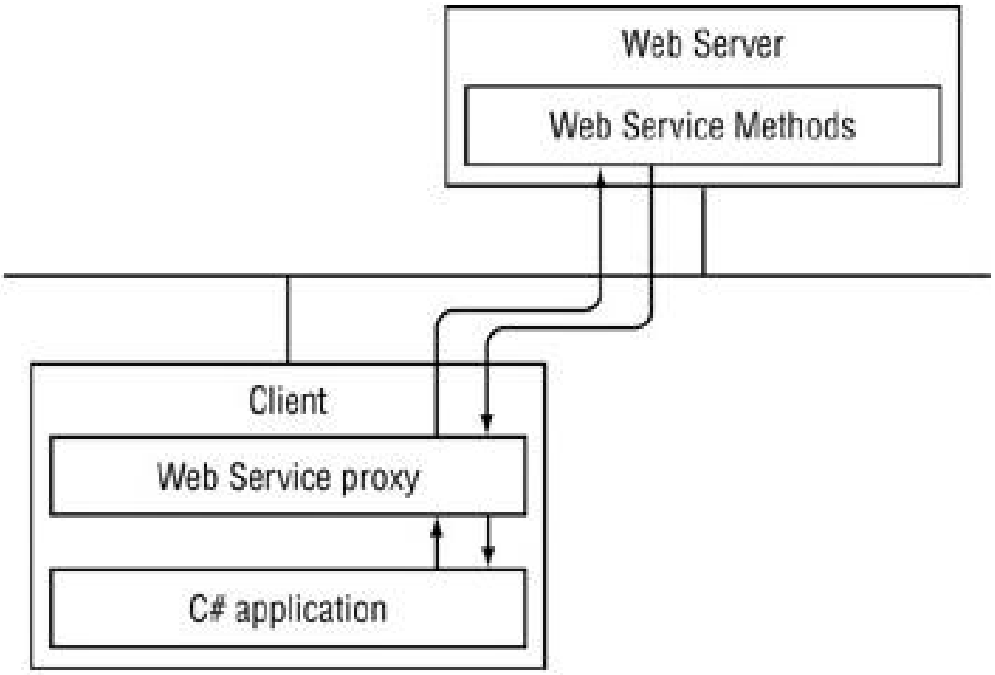
تمر عملية استخدام الـweb services بثلاثة مراحل وهي :

1- The web service server : والذي يتم من خلاله إرسال واستقبال البيانات عبر برتوكول الـSOAP باستخدام الـIIS والـASP.NET .

2- The proxy object : والذي يسمح للـ Client بإرسال و استقبال البيانات من وإلى الـweb Services Server حيث يتم تعريفه في الـHttpWebRequest من خلال الكلاس WebProxy وهو ما بينته في الجزء السابق.

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

كما في الشكل التالي :



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

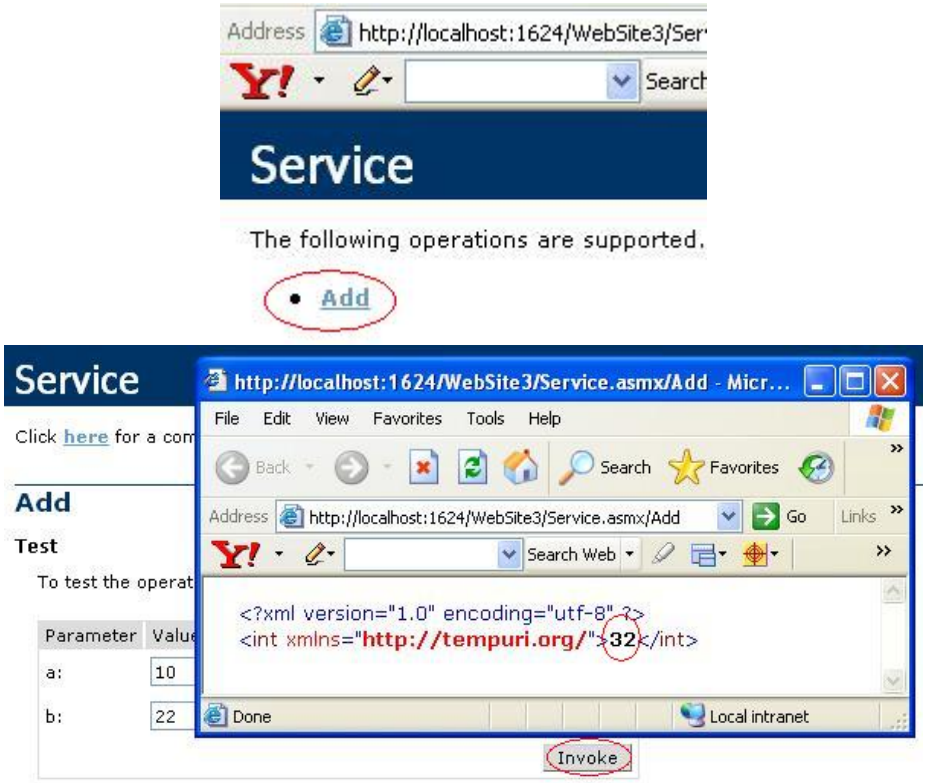
C#:

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

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

    }
```

VB.NET:

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

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

    Public Sub New()
    End Sub

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

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



ولإنشاء برنامج الـClient يجب أولا تحويل الكلاس السابق إلى Dll File و إرفاقه 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
```

وهكذا بينا الأساسيات وطرق التعامل مع الـ Web services في بيئة الدوت نيت ، سيتم الحديث في الفصل التالي عن SMTP & POP3 وطرق التعامل معه في بيئة الدوت نيت.

Chapter 14
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

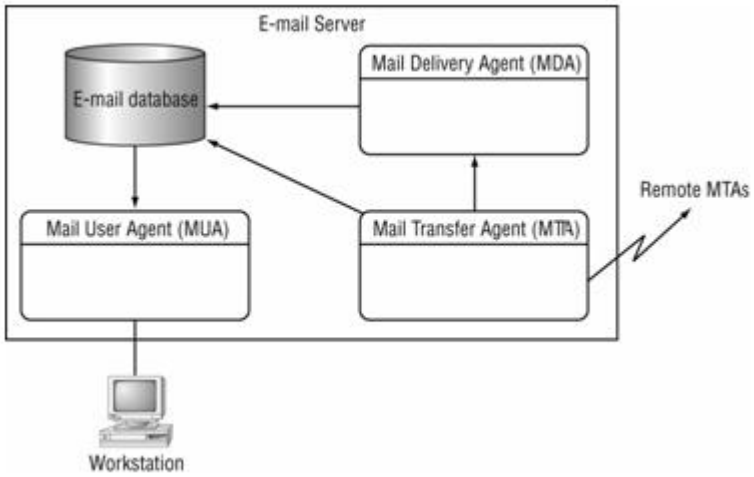
بسم الله الرحمن الرحيم

SMTP & POP3 Programming :14

تحدثنا في الجزء السابق عن برمجة برتوكول DNS والمسئول عن عملية ترجمة Domain من اسم نطاق إلى IP وبالعكس وبينا كيفية القيام بهذه العملية في سي شارب ، في هذا الجزء سوف نتحدث عن برمجة بعض البروتوكولات الأخرى لطبقة الـApplication Layer وهما هنا الـSMTP والمسئول عن إرسال الرسائل عبر البريد الإلكتروني و الـPOP3 والمسئول عن عملية توصيل الرسالة إلى الزبون من خلال عمل Download لها من الـMail Server وفي الجزء اللاحق سوف نتحدث عن الـ HTTP Programming والذي يستخدم بشكل أساسي في تصفح الـWeb ، مع العلم انه يوجد بروتوكولات كثيرة سوف آتي على شرحها عند الحاجة ..

أولا: SMTP – Simple Mail Transfer Protocol Programming

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

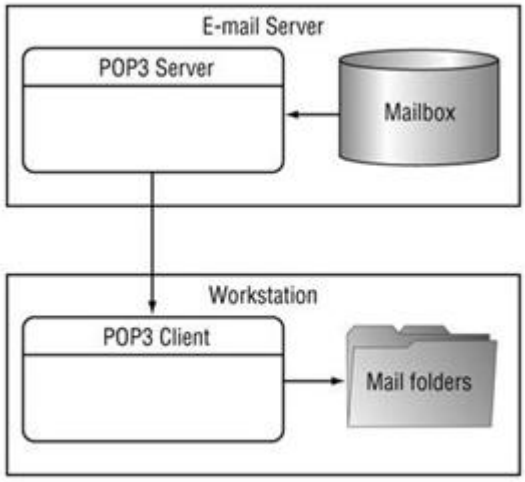


Message Transfer Agent – MTA والمسئول عن الإرسال Outgoing والتوصيل Incoming للرسائل

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

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

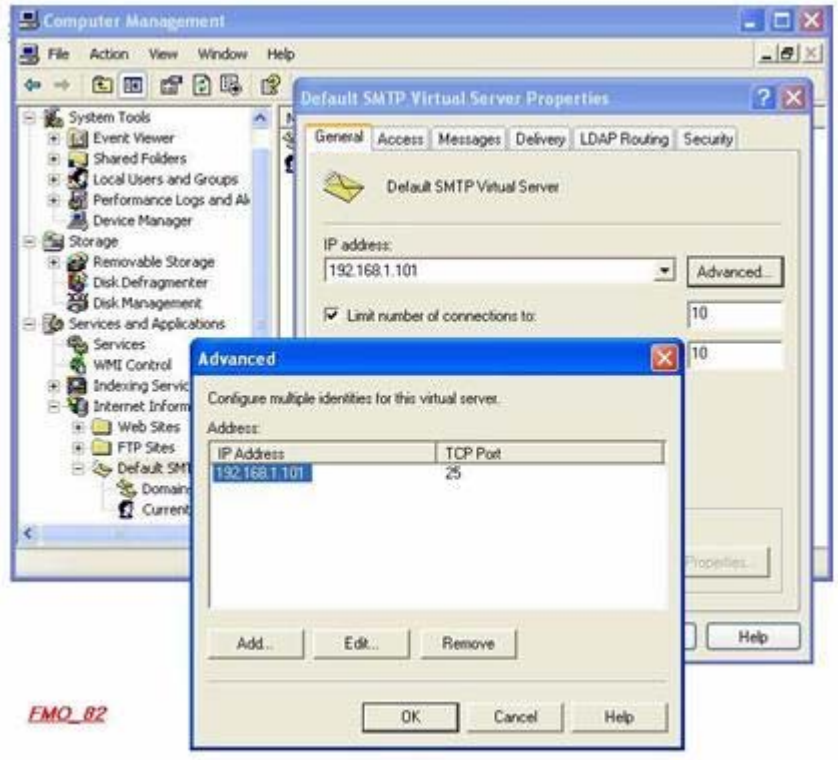
انظر إلى الشكل التالي :



و يستخدم برتوكول الـSMTP Simple Mail Transfer Protocol بشكل أساسي في الـMTA أي عمليات إرسال Outgoing وتوصيل Incoming الرسائل .

لتطبيق يجب أولا التأكد من أنك تملك حساب SMTP من الـInternet Provider الخاص بك تستطيع تجربة الـAccount الخاص بك من خلال برنامج الـOutlook Express الموجود مع الـWindows إذا كنت لا تملك حساب SMTP تستطيع تجربة البرنامج من خلال إنشاء Virtual SMTP Server عن طريق الـIIS وذلك بتثبيتها من : Control Panel >> Add/Remove Programs تأكد من تفعيل كل من الـIIS والـSMTP كما في الشكل التالي :

ثم إعداد الـServer من الـIIS كما في الشكل التالي :

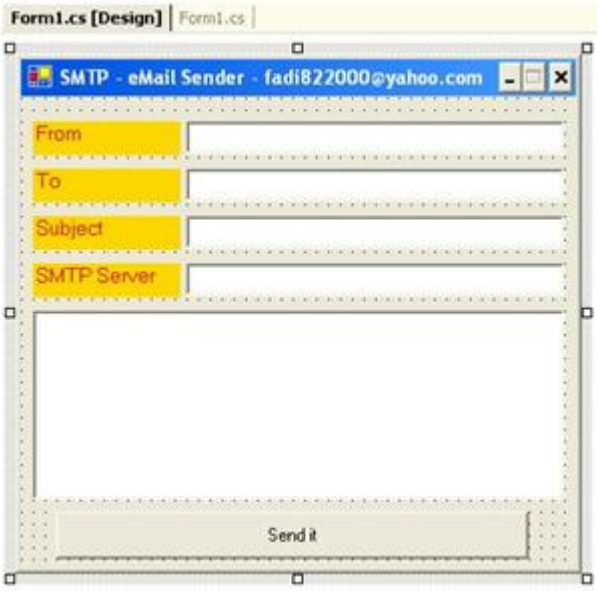


تدعم الدوت نيت استخدام بروتوكول الـSMTP من خلال System.Web.Mail Name Space و تحتوي على الكلاس SmtpMail والتي من خلالها نستخدم الميثود Send والتي تستخدم لإرسال الرسالة عبر الـPort 25 وهو الـPort المخصص لبروتوكول SMTP و تعتبر الميثود Send " overloaded Method " حيث تأخذ عدة أشكال إذ بإمكانك استخدامها مع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

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

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

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



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

لا تنسى إضافة Name Space هذا في بداية البرنامج

C#:

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

VB.NET:

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

ثم كتابة الكود هذا في زر الإرسال

C#:

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

VB.NET:

```
Try
Dim from As String = textBox1.Text
 Dim to As String = textBox2.Text
Dim subject As String = textBox3.Text
Dim body As String = textBox4.Text
 SmtpMail.SmtpServer = textBox5.Text
 SmtpMail.Send(from, to, subject, body)
Catch ex As Exception
Msgbox(ex.Message)
End Try
```

ملاحظة هامة جدا :

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

الجزء الأكثر تقدم: SMTP Advanced Programming

يعتبر المثال السابق مثال بسيط لإرسال رسائل عبر SMTP باستخدام CDO2 ، وفي العادة عند إنشاء برامج مثل برنامج الـOutlook يتم استخدام الـHTML Format بالإضافة إلا إمكانية إرسال ملحقات وطبعا يعطيك عدة خيارات لإرسال و استقبال البريد الإلكتروني هل باستخدام الـHTTP أو الـPOP3 ... وهنا سوف نقوم بإنشاء برنامج بسيط يقوم بإرسال واستقبال البريد الإلكتروني باستخدام الـSMTP و POP3 بنسبة لاستخدام الـPOP3 فيجب أن يتوفر لديك حساب POP3 من الـISP الخاص بك أو أن تقوم بتثبيت Microsoft Exchange Server2003 على جهازك وإعداده بحيث يستخدم الـPOP3 إذ عندها سوف تحتاج لوجود Domain Controller مثبت على الجهاز و Windows 2003 Server بالإضافة إلى تثبيت الـActive Directory عليه.

قدمت الدوت نيت دعم ممتاز لاستخدام هذه الخواص وذلك من خلال Name Space System.Web.Mail وباستخدام الكلاس MailMessage لدعم الـHTML Format و الكلاس MailAttachment لدعم إمكانية إرسال ملحقات مع الرسالة ولاكن لبرمجة الـPOP3 يلزم استخدام Name Space System.Net.Sockets و System.Net و System.IO حيث يتم عمل Session خاص مع الـServer للقيام بعملية تفحص وجود رسائل جديدة وفي حالة وجودها يقوم بتعبئتها في List Box أو Treelist حسب الحاجة وعند الضغط على إحداها يقوم الـClient بعمل Download لرسالة من الـMail Server ولعمل Advanced SMTP eMail قم بأخذ Object من الكلاس MailMessage كما يلي :

C#:

```
using System.Web.Mail;

try
{
 MailMessage mm = new MailMessage();

   mm.From = textBox1.Text;
     mm.To = textBox2.Text;
  // mm.Cc = لاحظ انه يمكنك من الإرسال لأكثر من شخص هذه حسب الحاجة
  // mm.Bcc =
   mm.Subject = textBox3.Text;
```

```
   mm.Headers.Add("Reply-To", "fadi822000@yahoo.com"); // لوضع أي إضافات تريدها مع الرسالة
   mm.Headers.Add("Comments", "This is a test HTML message");
   mm.Priority = MailPriority.High; // يمكنك وضع خيارات أهمية الرسالة
   mm.BodyFormat = MailFormat.Html; // نوع الفورمات المستخدم
   mm.Body = "<html><body><h1>" + textBox4.Text + "</h1></html>";
   SmtpMail.Send(mm);
}
catch (Exception ex) {MessageBox.Show(ex.Message);}
```

VB.NET:

```
imports System.Web.Mail;

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

لاحظ أن جسم الرسالة يستخدم كود الـHTML وهذا يمكنك من وضع أي لون أو حجم أو أي شيء يمكن عمله باستخدام الـHTML ، ولجعل البرنامج قادر على إرسال ملحقات يجب استخدام الكلاس MailAttachment وإدراج اسم الملف فيه وكما يلي بالكود :

C#:

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

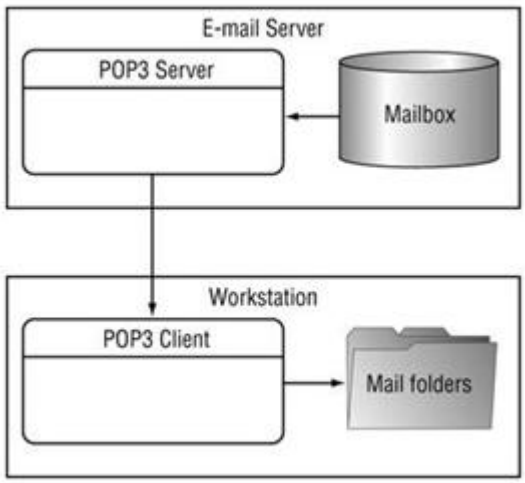
وهنا قد انتهينا من عمل برنامج الـSMTP بشكل كامل ، طبعا عملية ال Design وغيرها تعتمد على حسب ذوق وذكاء وخبرة المبرمج.

VB.NET:

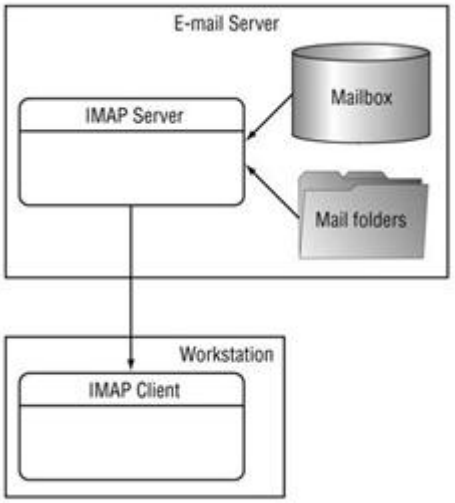
```
Dim myattach As MailAttachment = New
MailAttachment("Your_Attached_File_path.extension", MailEncoding.Base64)
```

ثانيا : POP3- Post Office Protocol Version 3 Programming

كما تحدثنا سابقا فإن وظيفة برتوكول الـPOP3 والذي يعمل في جزء الـ MUA - Mail User Agent على Port 110 ضمن برتوكول الـTCP تكمن في كونه المسئول عن عملية توصيل الرسالة إلى الزبون Client من خلال عمل Download لها من الـMail Server حيث تحفظ الرسائل في الـMail Folder والموجود أساسا في جهاز الـClient أنظر إلى الشكل التالي:



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



لاحظ أن الـMail Folder يقع ضمن Mail Server ويتم قراءته بعد التحقق Authentication من اسم المستخدم وكلمة المرور لاكن كما قلنا فإن مشكلته تكمن في محدودية سعة الـ Mail Folder لذا ينصح لشركات الكبيرة استخدام الـPOP3 كونه غير محدود السعة فالذي يتحكم في السعة هو الـClient ولا دخل ل Mail Server Administrator بها.

وبما أننا قررنا اعتماد الـPOP3 لعملية قراءة الرسائل سوف نبدأ ببرمجته إذ يلزم الأمر استخدام System.Net.Sockets Name Space و System.Net و System.IO حيث يتم عمل Session خاص مع الـServer باستخدام الـSocket للقيام بعملية تفحص وجود رسائل جديدة وفي حالة وجودها يقوم بتعبئة عناوينها في List Box أو Treelist خاص حسب الحاجة وعند

الضغط على إحداها يقوم الـClient بعمل Download لرسالة من الـMail Server إلى الـ Mail Folder ثم عرضها في Textbox.

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

ثم قم بإضافة Name Spaces التالية :

C#:

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

VB.NET:

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

لاحظ انه يتم التعامل مع الـSocket والـStream لإنشاء Session مع الـServer باستخدام برتوكول الـTCP وقراءة الرسالة من الـPOP3 Server .

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

C#:

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

VB.NET:

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

في الـConnect Button قم بإضافة الكود التالي :

C#:

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

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

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

قم بإضافة الكود التالي إلى الـDisconnect Button :

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

VB.NET:

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

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

C#:

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

// retrieve mail with number mail parameter
Data = "RETR 1"+CRLF; // لتحديد رقم الرسالة المراد قراءتها
szData = System.Text.Encoding.ASCII.GetBytes(Data.ToCharArray());
NetStrm.Write(szData,0,szData.Length);
szTemp = RdStrm.ReadLine(); // تخزين الرسالة بشكل مؤقت حتى يتم طباعتها
        if(szTemp[0]!='-')
        {
        while(szTemp!=".")
                {
        Message.Text += szTemp+CRLF;
        szTemp = RdStrm.ReadLine();
        }
                }
        else
        {Status.Items.Add(szTemp);}
        }
catch(InvalidOperationException err){Status.Items.Add("Error: "+err.ToString());}
```

VB.NET:

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

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

سيتم الحديث في الفصل التالي عن الـ FTP وطرق التعامل معه في بيئة الدوت نيت.

Chapter 15
FTP Programming

- Introduction to FTP – File Transfer Protocol

- Create a Simple Application to Transfer Files By Using COM Components

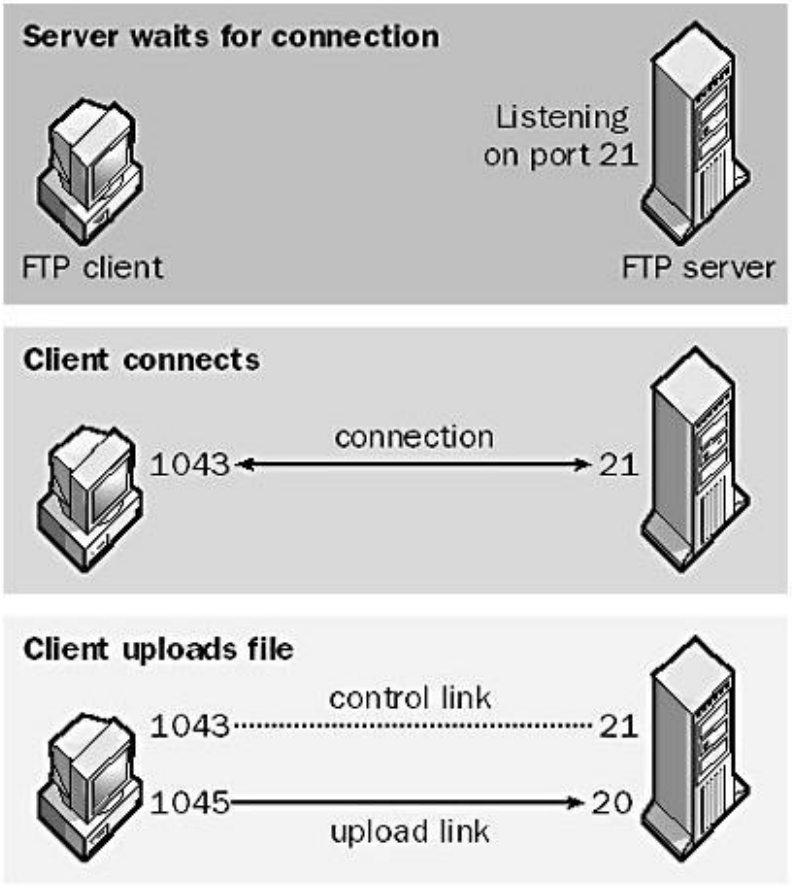
- Create a Simple Application to Transfer Files By Using Web Classes Components

- Create a Simple Application to Transfer Files By Using Socket Programming & Streaming Libraries

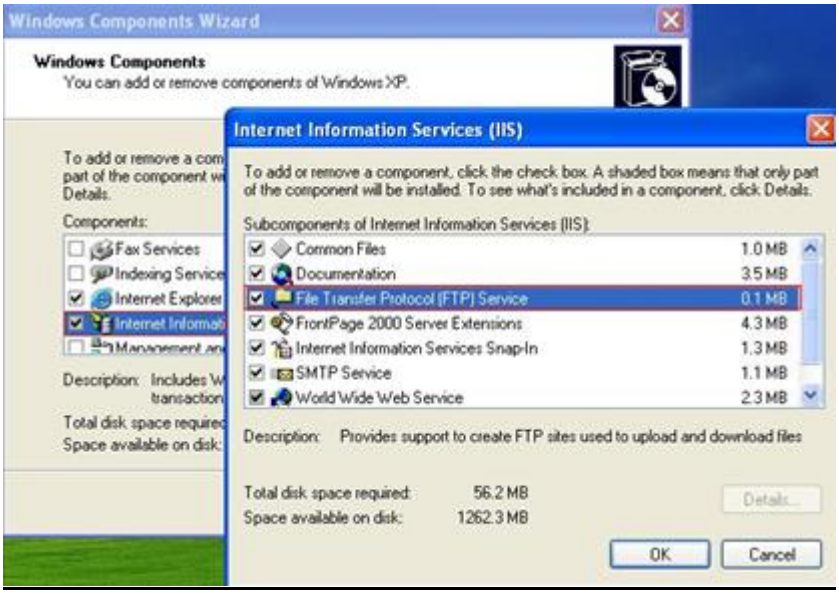
بسم الله الرحمن الرحيم

15 Programming – File Transfer Protocol – FTP :

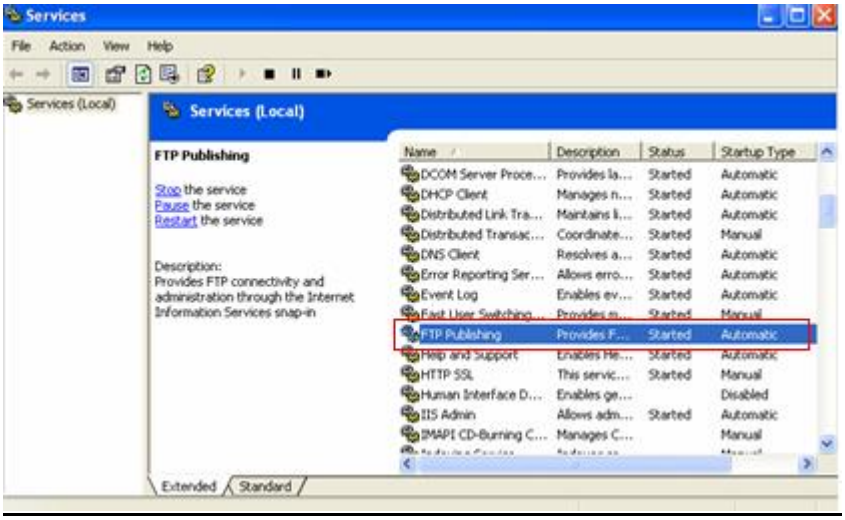
سوف نبدأ هنا بشرح برتوكول أخر من برتوكولات الـApplication Layer وهو برتوكول الـFTP والذي يستخدم بشكل أساسي في عملية تنزيل downloading و رفع uploadingالملفات من و إلى الـFTP Server وكالعادة في اغلب برمجيات الشبكات و التي تعتمد على وجود Client/Server حيث يقوم الـServer بتصنت على الـPort المخصص لل FTP وهو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



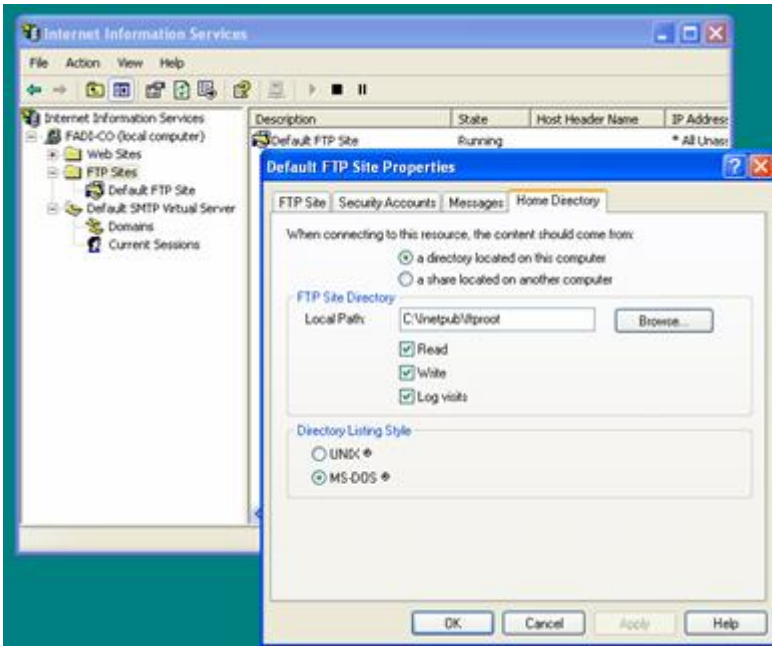
ملاحظة: لتفعيل خدمة الـFTP لديك بحيث يعمل جهازك ك FTP Server يجب أولا التأكد من أن الـFTP Services مثبتة لديك مع الـIIS و كما يظهر في الشكل التالي :



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

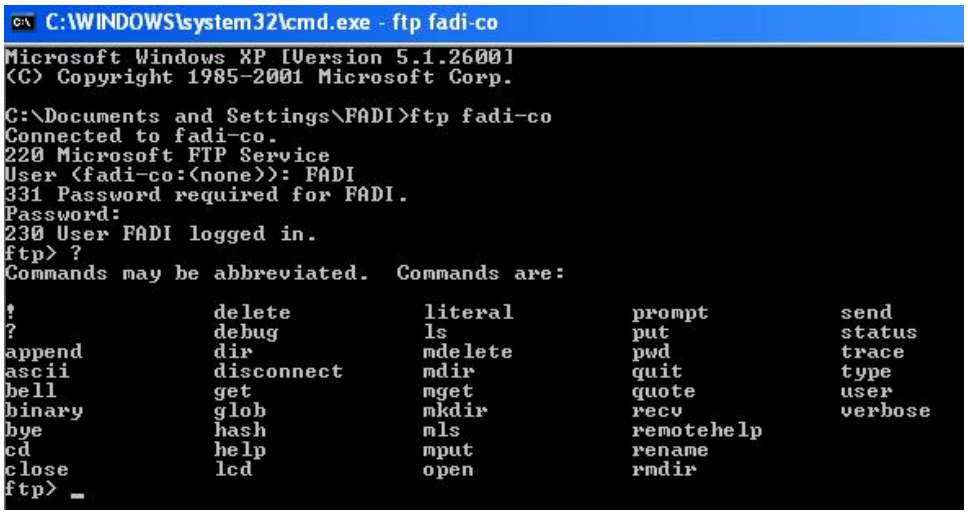


ثم التأكد منه في الـIIS بحيث يظهر كما في الشكل التالي :



أولا : FTP Commands

تشبه عملية الاتصال و الاستخدام لل FTP عملية الـTelnet إلى حد كبير حيث يدعم برتوكول الـFTP مجموعة من الأوامر والتي يتم من خلالها عملية التخاطب مع الـServer أو مع الـRemote Host وتضح هذه العملية كما في الشكل التالي :



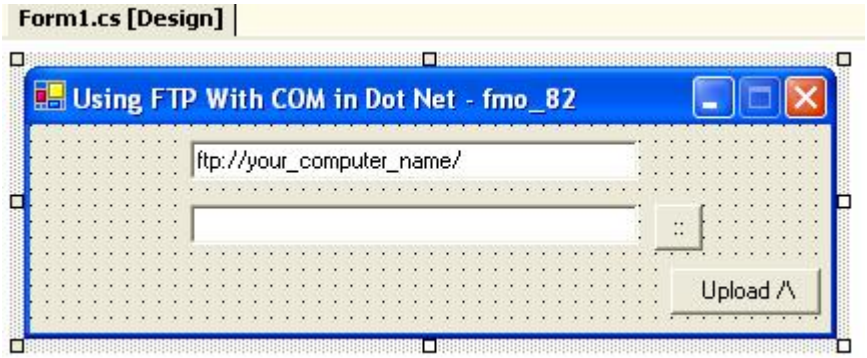
وهنا شرح لأهم الـFTP Commands:

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

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

ثانيا : التعامل مع الـFTP في الدوت نت باستخدام COM Components :

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



ثم قم بإضافة Name Spaces التالية :

C#:

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

VB.NET:

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

ثم إضافة الكود التالي إلى الـUpload Button :

C#:

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

VB.NET:

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

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

ثالثا : التعامل مع الـFTP في الدوت نت باستخدام الـWeb Class :

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

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

C#:

```
using System;
using System.Net;

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

      }

   }
}
```

VB.NET:

```
Imports System
Imports System.Net
Namespace Web_Client

   Class Program

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

- FtpRequestCreator ويستخدم لتسجيل وبدأ العمل مع الـFTP ويعرف كما يلي :

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

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

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

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

   Public Class FtpRequestCreator
      Implements IWebRequestCreate

      Public Sub New()
      End Sub

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

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

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

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

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

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

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

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

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

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

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

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

VB.NET:

```
Imports System
Imports System.Net
Namespace FTP

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

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

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

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

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

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

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

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

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

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

C#:

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

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

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

VB.NET:

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

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

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

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

C#:

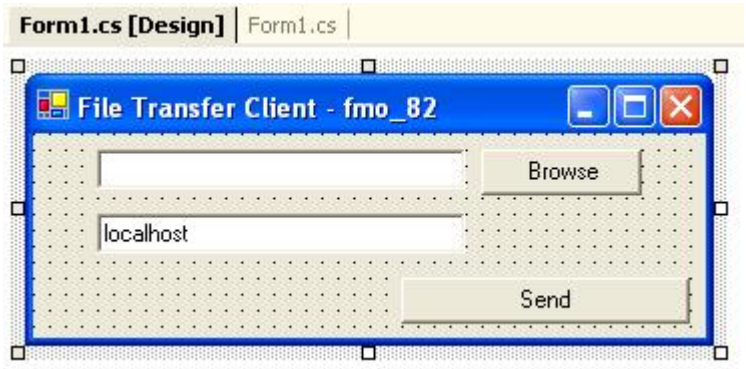
```
using System;
using System.IO;
using System.Net.Sockets;

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

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

VB.NET:

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

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

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

رابعا : مثال تطبيقي لرفع ملف من جهاز Client إلى جهاز Server باستخدام الـStream والـSocket:

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



سوف نستخدم Name Spaces التالية :

C#:

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

VB.NET:

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

في الـSend Button قم بكتابة الكود التالي :

C#:

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

VB.NET:

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

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

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

سوف نستخدم Name Spaces التالية :

C#:

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

VB.NET:

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

ثم إضافة الـMethod التالية وليكن اسمها handlerThread وكما يلي :

C#:

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

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

VB.NET:

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

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

C#:

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

VB.NET:

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

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

C#:

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

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

alSockets = new ArrayList();

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

VB.NET:

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

باستخدام الـThread تم تنفيذ الـ listenerThread Method والتي قمنا فيها بتعريف الـtcpListener وتفعيله على الـPort 8880 حيث سيتم قبول أي طلب يأتي من الـClient على هذا الـPort وبعد ذلك استدعاء الميثود handlerThread والتي سيتم فيها استقبال الـStream Data وتخزينها في Byte Array ثم قراءتها وتخزينها في المكان المحدد وباستخدام الـfileStream.Write حيث مررنا له الـStream والذي يحتوي على اسم الملف thisRead والـ dataByte Array...

وهكذا بينا طريقة عمل بروتوكول الـ FTP وطرق برمجته في بيئة الدوت نيت ، سيتم الحديث في الجزء التالي عن الـ Network Security Programming .

Part 5
Network Security Programming

Chapter 16
Network Security Programming & Cryptography

- Cryptography in Dot Net
- Hashing In Dot Net
- Digital Signature Algorithms

بسم الله الرحمن الرحيم

: Network Security Programming : 16

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

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

دعمت في الدوت نيت جميع أساليب الحماية التي ذكرناها سابقا باستخدام الـ Security Namespaces والتي تحتوي على مجموعة ضخمة من المكتبات الفرعية وهي كما في الشكل التالي

أولا : Cryptography Namespace Overview :

Cryptography in .NET: وهي المكتبة التي تهتم بكل ما يخص عمليات تشفير وفك تشفير البيانات من Clear Text إلى Cipher Text وبالعكس وتستخدم بشكل أساسي لتشفير البيانات قبل عملية الإرسال وفك تشفيرها عند الاستلام ، ونستطيع تقسيم طرق التشفير فيها إلى ثلاثة أقسام رئيسية هي:

A- Symmetric algorithms: الأسلوب المتماثل وفيه يستخدم المفتاح السري ذاته لعملية التشفير وفك التشفير وهي طريقة سريعة لإجراء عملية التشفير وفك التشفير لا كنها ليست آمنة كطريقة الغير المتماثلة ودعمت الدوت نيت التشفير المتماثل بمجموعة من الـAlgorithms Classes وهي:

- الكلاس الذي يدعم التشفير باستخدام ال DES-Data Encryption Standard : DESCryptoServiceProvider
- الكلاس الذي يدعم RC2 Algorithms : RC2CryptoServiceProvider
- الكلاس الذي يدعم Rijndael Managed Algorithms : RijndaelManaged

الطريقة المعتادة في التشفير بالأسلوب المتماثل هي تشفير الرسالة وإرسالها عبر الشبكة لاكن باستخدام هذه الطريقة فإن نسبة الخطأ التي قد تكون عالية جدا وقد نفقد بعض هذه البيانات مما يؤدي إلى فقد الرسالة أو قد تسرق وتجرى عليها عمليات لمحاولة فك الشيفرة ناهيك عن الحجم الهائل التي قد تحجزه من الـNetwork Bandwidth ..وتم حل هذه المشكلة بجعل عملية التشفير تتم على مستوى الـStream نفسه ويستخدم لهذه العملية الـ**CryptoStream Class** حيث يتم استخدام مفتاحين لتشفير مفتاح التشفير Encryption Kay ومفتاح لفك التشفير Installation Victor Decryption **IV** ويشترط استخدام نفس المفتاحين في عملية التشفير وفك التشفير ويستخدم الكلاس السابق مع الـ**MemoryStream** أو **FileStream** حيث نمرر له الـStream Data ونوع التشفير سواء DES أو TripleDES أو RC2 ، وكمثال سوف نستخدم الـTripleDES إذيجب أن يتكون كلا المفتاحين من 16 Bits ...

Symmetric Stream Encryption Example:

C#:

```
byte[] Key = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09,0x10, 0x11,
0x12, 0x13, 0x14, 0x15, 0x16};
byte[] IV =  {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09,0x10, 0x11,
0x12, 0x13, 0x14, 0x15, 0x16};

string phrase = msg.Text;
MemoryStream ms = new MemoryStream();

TripleDESCryptoServiceProvider tdes = new TripleDESCryptoServiceProvider();
CryptoStream csw = new CryptoStream(ms,tdes.CreateEncryptor(Key, IV),
CryptoStreamMode.Write);

csw.Write(Encoding.ASCII.GetBytes(phrase), 0, phrase.Length);
csw.FlushFinalBlock();

byte[] cryptdata = ms.GetBuffer();

textBox1.Text=Encoding.ASCII.GetString(cryptdata, 0, (int)ms.Length);
```

VB.NET:

```
Dim Key As Byte() = {&H1, &H2, &H3, &H4, &H5, &H6, &H7, &H8, &H9, &H10,
&H11, &H12, &H13, &H14, &H15, &H16}
Dim IV As Byte() = {&H1, &H2, &H3, &H4, &H5, &H6, &H7, &H8, &H9, &H10,
&H11, &H12, &H13, &H14, &H15, &H16}

Dim phrase As String = msg.Text
Dim ms As MemoryStream = New MemoryStream()

Dim tdes As TripleDESCryptoServiceProvider = New
TripleDESCryptoServiceProvider()
Dim csw As CryptoStream = New CryptoStream(ms, tdes.CreateEncryptor(Key,
IV), CryptoStreamMode.Write)

csw.Write(Encoding.ASCII.GetBytes(phrase), 0, phrase.Length)
csw.FlushFinalBlock()

Dim cryptdata As Byte() = ms.GetBuffer()

textBox1.Text=Encoding.ASCII.GetString(cryptdata, 0, CInt(ms.Length))
```

Symmetric Stream Decryption Example:

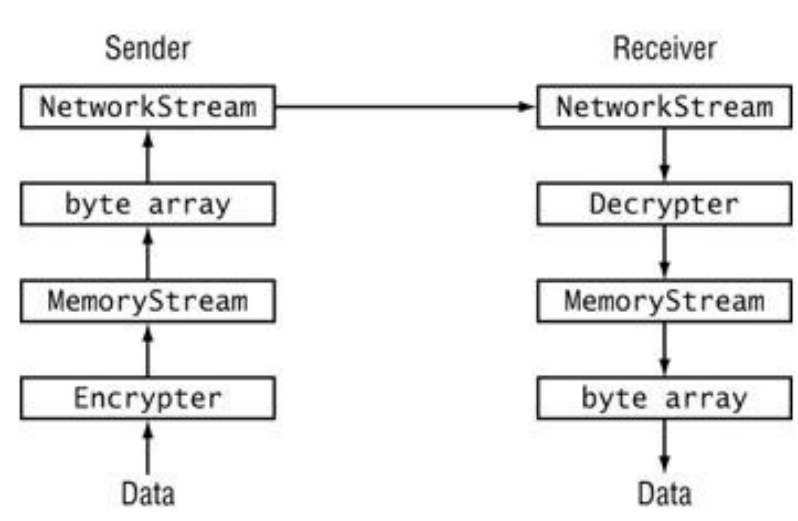
C#:

```
byte[] Keyy = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09,0x10, 0x11,
0x12, 0x13, 0x14, 0x15, 0x16};
byte[] IVv =  {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09,0x10, 0x11,
0x12, 0x13, 0x14, 0x15, 0x16};
ms.Position = 0;
byte[] data = new byte[1024];
CryptoStream csr = new CryptoStream(ms,tdes.CreateDecryptor(Keyy,
IVv),CryptoStreamMode.Read);
int recv = csr.Read(data, 0, data.Length);
string newphrase = Encoding.ASCII.GetString(data, 0, recv);
textBox1.Text=newphrase;
```

VB.NET:

```
Dim Keyy As Byte() = {&H1, &H2, &H3, &H4, &H5, &H6, &H7, &H8, &H9, &H10,
&H11, &H12, &H13, &H14, &H15, &H16}
Dim IVv As Byte() = {&H1, &H2, &H3, &H4, &H5, &H6, &H7, &H8, &H9, &H10,
&H11, &H12, &H13, &H14, &H15, &H16}
ms.Position = 0
Dim data As Byte() = New Byte(1023) {}
Dim csr As CryptoStream = New CryptoStream(ms, tdes.CreateDecryptor(Keyy,
IVv), CryptoStreamMode.Read)
Dim recv As Integer = csr.Read(data, 0, data.Length)
Dim newphrase As String = Encoding.ASCII.GetString(data, 0, recv)
textBox1.Text=newphrase
```

في برمجيات الشبكات نقوم في البداية بتشفير البيانات المرسلة باستخدام أي من الأساليب السابقة لتشفير ثم نحول البيانات المشفرة إلى Stream لإرسالها عبر الـSocket باستخدام الـNetwork Stream ، ثم يقوم الطرف المستقبل باستقبال الرسالة باستخدام الـNetwork Stream عبر الـSocket ، عملية فك التشفير تكون كما هي الخوارزمية المستخدمة ثم تحمل الرسالة إلى الـmemory stream وتخزن في Byte Array عندها يمكن أن تحول إلى رسالة مرة أخرى وكما في الشكل التالي:



B- Asymmetric algorithms: الأسلوب الغير متماثل وهو أكثر أمانا من الأسلوب المتماثل إذ تشفر البيانات باستخدام مفتاح عام Public Kay ولفك التشفير يستخدم مفتاح خاص Private Kay ويكون هناك علاقة بين المفتاحين ويستخدم 128 Bits لتشفير وهو أفضل أساليب التشفير للبيانات ودعمت الدوت نيت التشفير الغير متماثل والذي يدعم تشفير المفتاح الخاص Private Kay باستخدام Tow Algorithms Classes وهي:

1- DSACryptoServiceProvider for Digital Signature Algorithm

التواقيع الرقمية :والهدف منها التحقق من هوية الشخص مرسل الرسالة وكمثال يقوم المرسل بتوليد ملخص لرسالة باستخدام الHash Function وبعد ذلك يقوم بتشفير ملخص الرسالة الذي تم توليده لتكوين المفتاح الخاص والذي سيستخدم كتوقيع رقمي للمرسل ثم يرسل المفتاح العام مع الرسالة، أما بما يتعلق بالمستلم فيقوم بفك تشفير الملخص باستخدام المفتاح العام ويجب أن يتم ذلك باستخدام نفس الخوارزمية التي اتبعها المرسل في تشفير الملخص، فإذا كان ملخص الرسالة التي ولدها المستلم هي نفسها التي ولدها المرسل عندها يتحقق من أن الشخص مرسل الرسالة هو نفسه .
في البداية سوف ننشئ instance من الـDSACryptoServiceProvider لتوليد المفتاح العام والخاص ثم نكون الـHash sign Value ونخزنه في Byte Array ولفحصه نولد hash sign value جديد ونقارنه بالسابق فإذا تشابها عندها نقرر أن الشخص هو نفسه صاحب الرسالة المرسلة وكما يلي:

C#:

```
using System;
using System.Security.Cryptography;

class DSACSPSample
{

        static void Main()
        {
                try
                {
//Create a new instance of DSACryptoServiceProvider to generate
//a new key pair.
DSACryptoServiceProvider DSA = new DSACryptoServiceProvider();

//The hash value to sign.
byte[] HashValue =
{59,4,248,102,77,97,142,201,210,12,224,93,25,41,100,197,213,134,130,135};
//The value to hold the signed value.
byte[] SignedHashValue = DSASignHash(HashValue, DSA.ExportParameters(true),
"SHA1");

//Verify the hash and display the results.
if(DSAVerifyHash(HashValue, SignedHashValue, DSA.ExportParameters(false),
"SHA1"))
{Console.WriteLine("The hash value was verified.");}
else
{Console.WriteLine("The hash value was not verified.");}}
catch(ArgumentNullException e)
{Console.WriteLine(e.Message);}
}
public static byte[] DSASignHash(byte[] HashToSign, DSAParameters DSAKeyInfo,
string HashAlg)
```

```
{
        try
        {
//Create a new instance of DSACryptoServiceProvider.
DSACryptoServiceProvider DSA = new DSACryptoServiceProvider();

//Import the key information.
DSA.ImportParameters(DSAKeyInfo);

//Create an DSASignatureFormatter object and pass it the
//DSACryptoServiceProvider to transfer the private key.
DSASignatureFormatter DSAFormatter = new DSASignatureFormatter(DSA);

//Set the hash algorithm to the passed value.
DSAFormatter.SetHashAlgorithm(HashAlg);

//Create a signature for HashValue and return it.
return DSAFormatter.CreateSignature(HashToSign);
}
catch(CryptographicException e)
{Console.WriteLine(e.Message);return null;}
}
```

VB.NET:

```
Imports System
Imports System.Security.Cryptography

Friend Class DSACSPSample

    Shared Sub Main()
        Try
            'Create a new instance of DSACryptoServiceProvider to generate
            'a new key pair.
            Dim DSA As DSACryptoServiceProvider = New DSACryptoServiceProvider()

            'The hash value to sign.
            Dim HashValue As Byte() = {59, 4, 248, 102, 77, 97, 142, 201, 210, 12,
224, 93, 25, 41, 100, 197, 213, 134, 130, 135}
            'The value to hold the signed value.
            Dim SignedHashValue As Byte() = DSASignHash(HashValue,
DSA.ExportParameters(True), "SHA1")

            'Verify the hash and display the results.
            If DSAVerifyHash(HashValue, SignedHashValue,
DSA.ExportParameters(False), "SHA1") Then
                Console.WriteLine("The hash value was verified.")
            Else
                Console.WriteLine("The hash value was not verified.")
            End If
        Catch e As ArgumentNullException
            Console.WriteLine(e.Message)
```

```
        End Try
    End Sub
    Public Shared Function DSASignHash(ByVal HashToSign As Byte(), ByVal
DSAKeyInfo As DSAParameters, ByVal HashAlg As String) As Byte()
        Try
            'Create a new instance of DSACryptoServiceProvider.
            Dim DSA As DSACryptoServiceProvider = New DSACryptoServiceProvider()

            'Import the key information.
            DSA.ImportParameters(DSAKeyInfo)

            'Create an DSASignatureFormatter object and pass it the
            'DSACryptoServiceProvider to transfer the private key.
            Dim DSAFormatter As DSASignatureFormatter = New
DSASignatureFormatter(DSA)

            'Set the hash algorithm to the passed value.
            DSAFormatter.SetHashAlgorithm(HashAlg)

            'Create a signature for HashValue and return it.
            Return DSAFormatter.CreateSignature(HashToSign)
        Catch e As CryptographicException
            Console.WriteLine(e.Message)
            Return Nothing
        End Try
    End Function
```

C#:

```
public static bool DSAVerifyHash(byte[] HashValue, byte[] SignedHashValue,
DSAParameters DSAKeyInfo, string HashAlg)
{
        try
        {
//Create a new instance of DSACryptoServiceProvider.
DSACryptoServiceProvider DSA = new DSACryptoServiceProvider();

//Import the key information.
DSA.ImportParameters(DSAKeyInfo);

//Create an DSASignatureDeformatter object and pass it the
//DSACryptoServiceProvider to transfer the private key.
DSASignatureDeformatter DSADeformatter = new DSASignatureDeformatter(DSA);

//Set the hash algorithm to the passed value.
DSADeformatter.SetHashAlgorithm(HashAlg);

//Verify signature and return the result.
return DSADeformatter.VerifySignature(HashValue, SignedHashValue);
        }
catch(CryptographicException e){Console.WriteLine(e.Message);return false;}}}
```

VB.NET:

```
Public Shared Function DSAVerifyHash)ByVal HashValue As Byte ،()ByVal
SignedHashValue As Byte ،()ByVal DSAKeyInfo As DSAParameters ،ByVal HashAlg
As String (As Boolean
Try
'Create a new instance of DSACryptoServiceProvider.
Dim DSA As DSACryptoServiceProvider = New DSACryptoServiceProvider()

'Import the key information .
DSA.ImportParameters(DSAKeyInfo(

'Create an DSASignatureDeformatter object and pass it the
'DSACryptoServiceProvider to transfer the private key.
Dim DSADeformatter As DSASignatureDeformatter = New
DSASignatureDeformatter(DSA(

'Set the hash algorithm to the passed value.
DSADeformatter.SetHashAlgorithm(HashAlg(

'Verify signature and return the result .
Return DSADeformatter.VerifySignature(HashValue, SignedHashValue(
Catch e As CryptographicException
        Console.WriteLine(e.Message(
        Return False
End Try
End Function
```

2- RSACryptoServiceProvider

ويستخدم في إجراء التشفير وفك التشفير الغير متماثل وهو non inherited Class في البداية سوف ننشئ instance جديد من الـRSACryptoServiceProvider وذلك لتوليد المفتاح العام والخاص ونرفق المفتاح العام مع الرسالة ومن ثم يقوم المستلم بفك الرسالة باستخدام المفتاح الخاص وتتم كما في الشكل التالي:

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

وهنا مثال توضيحي لطريقة التشفير وفك التشفير باستخدام الـRSA Algorithm :

C#:

```
using System;
using System.Security.Cryptography;
using System.Text;

class RSACSPSample
{

        static void Main()
        {
                try
                {
//Create a UnicodeEncoder to convert between byte array and string.
UnicodeEncoding ByteConverter = new UnicodeEncoding();

//Create byte arrays to hold original, encrypted, and decrypted data.
        byte[] dataToEncrypt = ByteConverter.GetBytes("Data to Encrypt");
        byte[] encryptedData;
        byte[] decryptedData;

//Create a new instance of RSACryptoServiceProvider to generate
//public and private key data.
        RSACryptoServiceProvider RSA = new RSACryptoServiceProvider();

//Pass the data to ENCRYPT, the public key information
//(using RSACryptoServiceProvider.ExportParameters(false),
//and a boolean flag specifying no OAEP padding.
encryptedData = RSAEncrypt(dataToEncrypt,RSA.ExportParameters(false), false);

//Pass the data to DECRYPT, the private key information
//(using RSACryptoServiceProvider.ExportParameters(true),
//and a boolean flag specifying no OAEP padding.
decryptedData = RSADecrypt(encryptedData,RSA.ExportParameters(true), false);

//Display the decrypted plaintext to the console.
Console.WriteLine("Decrypted plaintext: {0}",
ByteConverter.GetString(decryptedData));
        }
catch(ArgumentNullException) {Console.WriteLine("Encryption failed.");}
        }
```

VB.NET:

```
Imports System
Imports System.Security.Cryptography
Imports System.Text

Friend Class RSACSPSample

    Shared Sub Main()
        Try
            'Create a UnicodeEncoder to convert between byte array and string.
            Dim ByteConverter As UnicodeEncoding = New UnicodeEncoding()

            'Create byte arrays to hold original, encrypted, and decrypted data.
            Dim dataToEncrypt As Byte() = ByteConverter.GetBytes("Data to
Encrypt")
            Dim encryptedData As Byte()
            Dim decryptedData As Byte()

            'Create a new instance of RSACryptoServiceProvider to generate
            'public and private key data.
            Dim RSA As RSACryptoServiceProvider = New RSACryptoServiceProvider()

            'Pass the data to ENCRYPT, the public key information
            '(using RSACryptoServiceProvider.ExportParameters(false),
            'and a boolean flag specifying no OAEP padding.
            encryptedData = RSAEncrypt(dataToEncrypt,
RSA.ExportParameters(False), False)

            'Pass the data to DECRYPT, the private key information
            '(using RSACryptoServiceProvider.ExportParameters(true),
            'and a boolean flag specifying no OAEP padding.
            decryptedData = RSADecrypt(encryptedData,
RSA.ExportParameters(True), False)

            'Display the decrypted plaintext to the console.
            Console.WriteLine("Decrypted plaintext: {0}",
ByteConverter.GetString(decryptedData))
        Catch e1 As ArgumentNullException
            Console.WriteLine("Encryption failed.")
        End Try
    End Sub
```

ننشئ الميثود التي ستقوم بتشفير الرسالة:

C#:

```
static public byte[] RSAEncrypt(byte[] DataToEncrypt, RSAParameters RSAKeyInfo,
bool DoOAEPPadding)
{
        try{
//Create a new instance of RSACryptoServiceProvider.
RSACryptoServiceProvider RSA = new RSACryptoServiceProvider();

//Import the RSA Key information. This only needs
//to include the public key information.
RSA.ImportParameters(RSAKeyInfo);

//Encrypt the passed byte array and specify OAEP padding.
//OAEP padding is only available on Microsoft Windows XP or
//later.
return RSA.Encrypt(DataToEncrypt, DoOAEPPadding);
                }
//Catch and display a CryptographicException
//to the console.
catch(CryptographicException e){Console.WriteLine(e.Message);return null;}
}
```

VB.NET:

```
Shared Public Function RSAEncrypt)ByVal DataToEncrypt As Byte ،()ByVal
RSAKeyInfo As RSAParameters ،ByVal DoOAEPPadding As Boolean (As Byte()
        Try
'Create a new instance of RSACryptoServiceProvider.
Dim RSA As RSACryptoServiceProvider = New RSACryptoServiceProvider()
'Import the RSA Key information. This only needs
'to include the public key information.
RSA.ImportParameters(RSAKeyInfo(
'Encrypt the passed byte array and specify OAEP padding  .
'OAEP padding is only available on Microsoft Windows XP or
'later  .
Return RSA.Encrypt(DataToEncrypt, DoOAEPPadding(
'Catch and display a CryptographicException
'to the console.
Catch e As CryptographicException
        Console.WriteLine(e.Message(
        Return Nothing
End Try
End Function
```

ننشئ الميثود التي ستقوم بفك تشفير الرسالة:

C#:

```
static public byte[] RSADecrypt(byte[] DataToDecrypt, RSAParameters
RSAKeyInfo,bool DoOAEPPadding)
{
try
{
//Create a new instance of RSACryptoServiceProvider.
RSACryptoServiceProvider RSA = new RSACryptoServiceProvider();
//Import the RSA Key information. This needs
//to include the private key information.
RSA.ImportParameters(RSAKeyInfo);

//Decrypt the passed byte array and specify OAEP padding.
//OAEP padding is only available on Microsoft Windows XP or
//later.
return RSA.Decrypt(DataToDecrypt, DoOAEPPadding);
                }
//Catch and display a CryptographicException
//to the console.
catch(CryptographicException e){Console.WriteLine(e.ToString());return null;}
}}
```

VB.NET:

```
Shared Public Function RSADecrypt)ByVal DataToDecrypt As Byte ،()ByVal
RSAKeyInfo As RSAParameters ،ByVal DoOAEPPadding As Boolean (As Byte()
Try
'Create a new instance of RSACryptoServiceProvider.
Dim RSA As RSACryptoServiceProvider = New RSACryptoServiceProvider()
'Import the RSA Key information. This needs
'to include the private key information.
RSA.ImportParameters(RSAKeyInfo(

'Decrypt the passed byte array and specify OAEP padding  .
'OAEP padding is only available on Microsoft Windows XP or
'later  .
Return RSA.Decrypt(DataToDecrypt, DoOAEPPadding(
'Catch and display a CryptographicException
'to the console.
Catch e As CryptographicException
        Console.WriteLine(e.ToString(()
        Return Nothing
End Try
End Function
```

C- Hashing algorithms: وهو أقوى الأساليب البرمجية لتشفير البيانات إذ يستخدم فيه 512 bits algorithm كحد أقصى بدلا من 128 bits باستخدام Message Digest Algorithms وهنا لن تستطيع فك تشفير الرسالة وإرجاعها إلى حالتها السابقة ويستخدم بشكل أساسي لتوليد الـPasswords وفي توليد التواقيع الرقمية Digital Signature وفي اغلب الحالات تستخدم لتخزين كلمة المرور Password في الـDatabase بشكل امن.
ويستخدم الـSHA1Managed و الـSHA256Managed والـ SHA384Managed والـSHA512Managed لتعريف Hash Object ومنه نستخدم الـComputeHash Method لتوليد الـhash code وتخزينه في byte Array وكما يلي كمثال:

C#:

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

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

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

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

VB.NET:

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

وهكذا بينا في هذا الفصل كيفية التعامل مع الـCryptography لإجراء عمليات التشفير على الـ Data المرسلة عبر الشبكة ، كما بينا طرق استخدام الـ Hashing والـ Digital Signature في بيئة الدوت نيت ، سيتم الحديث في الفصل التالي عن الـ Socket Permission واستخدامها في بيئة الدوت نيت.

ستجد كافة تفاصيل هذا الموضوع في النسخة الورقية من الكتاب
لطلب أ والاستفسار أو التوزيع يرجى الاتصال على احد العناوين التالية

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

Chapter 17
Socket Permissions

- Permission Namespace Overview
- Security Action
- Socket Access property

بسم الله الرحمن الرحيم

17 : Permission Namespace Overview :

وتدعم الـPermission Namespace في الدوت نيت ثلاثة أنواع من الصلاحيات وهي الـ Socket permissions والـIdentity Permissions والـRole- based permissions ...

Socket Permission: وتمكنك من تحديد صلاحيات استخدام الـSocket في برمجيات الشبكات باستخدام SocketPermission و SocketPermissionAttribute ضمن الـSystem.Net والـSystem.Security.Permissions Namespaces وكمثال نستطيع منع Client Host Address معين من الاتصال مع الـListener Application ، ويتم ذلك بتعريف SocketPermission Attribute نحدد فيها نوع العملية والـAccess Kind و عنوان الـHost الذي سيطبق عليه الـPermission ورقم الـPort ونوع الـTransport سواء موجه أو غير موجه TCP أو UDP.

نريد في هذا المثال منع اتصال الـAddress 127.0.0.1 loopback بالـSocket عبر جميع الـPorts وبغض النظر عن نوع الـSocket المستخدم.

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

يمكننا الـSecurityAction object من تحديد نوع العملية التي نريدها وكما يلي:

Assert: وتعني السماح Client Host معين من إجراء عملية محددة
Demand: وتعني تطبيق الصلاحيات على جميع الـClasses التي تقع في منطقة الـStack أعلى الـDefined Abstract
Deny: وتعني منع الـClient Host من إجراء عملية معينة.
InheritanceDemand : وفيها تطبق الصلاحيات على الـClass الذي سيرث الـClass الحالي.
PermitOnly: وفيه يمنع جميع الـAccess عدا الـClient User المحدد.
...

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

Accept لمنع أو السماح ل Client Socket من عمل Binding مع الـIP Address و الـPort المحدد.
Connect لمنع أو السماح ل Client Socket من عمل connect مع الـRemote Host المحدد.

في الـHost والـPort نحدد عنوان الـHost الذي سيطبق عليه الـPermission و رقم الـPort التي يتصل بها (في الـPort property نستطيع تمرير كلمة all لدلالة على تطبيق الصلاحية على جميع الـPorts)

وأخيرا نحدد الـTransport property والتي سنعرف فيها نوع الـSocket المستخدم وتأخذ الخيارات التالية:
All بدون تحديد نوع الـSocket إذ تطبق هذه الـPermission على جميع الـSocket Types.
Connectionless إذا كانت الـSocket تستخدم Datagram Protocols وكمثال بروتوكول UDP.
ConnectionOriented إذا كانت الـSocket تستخدم Oriented Protocols وكمثال بروتوكول TCP.
TCP إذ تستطيع تحديده مباشرة.
UDP إذ تستطيع تحديده مباشرة.

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.

Chapter 18
Packet Sniffer & Row Programming

- Introduction to Row Programming
- Create a Packet Sniffer Application
- Using ARP,RARP in Security Programming.

بسم الله الرحمن الرحيم

هذا الفصل فقط في النسخة الورقية
لطلب أ والاستفسار أو التوزيع يرجى الاتصال على احد العناوين التالية

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

الموقع الرسمي للكتاب
www.fadidotnet.org

Part 6
Multithreading

Chapter 19: Multithreading Using & Managing

Chapter 19
Multithreading
Using & Managing

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

بسم الله الرحمن الرحيم

هذا الفصل فقط في النسخة الورقية
لطلب أ والاستفسار أو التوزيع يرجى الاتصال على احد العناوين التالية

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

الموقع الرسمي للكتاب
www.fadidotnet.org

Appendixes

- System.Net Namespace
- System.Net.Socket Namespace
- System.Threading Namespace
- System.Runtime.Remoting
- System.Runtime.Serialization

1- System.Net Namespace Classes

وتدعم برمجة Application Layer و Transport Layer و Web Protocols ومن اهم هذه Classes واستخداماتها:

Class	Description
Authorization	Provides authentication messaging for a web server.
Cookie	Provides a set of properties and methods used to manage cookies. This class cannot be inherited.
Dns	Simple domain name resolution functionality.
EndPoint	Identifies a network address. This is an abstract class.
GlobalProxySelection	Global default proxy instance for all HTTP requests.
HttpVersion	Defines the HTTP version numbers supported by the HttpWebRequest and HttpWebResponse classes.
HttpWebRequest	HTTP-specific implementation of the WebRequest class.
HttpWebResponse	HTTP-specific implementation of the WebResponse class.
IPAddress	Internet Protocol (IP) address.
IPEndPoint	A network endpoint consisting of an IP address and a port number.
IPHostEntry	Container class for Internet host address information.
NetworkCredential	Provides credentials for password-based authentication schemes such as basic, digest, NTLM, and Kerberos authentication.
SocketAddress	Stores serialized information from EndPoint-derived classes.
SocketPermission	Controls rights to make or accept socket connections.
WebClient	Provides common methods for sending data to and receiving data from a resource identified by a URI.
WebException	The exception that is thrown when an error occurs while accessing resources via the HTTP protocol.
WebPermission	Controls rights to access HTTP Internet resources.
WebPermissionAttribute	Specifies permission to access Internet resources.
WebProxy	Contains HTTP proxy settings for the WebRequest class.
WebRequest	Makes a request to a Uniform Resource Identifier (URI). This class is abstract.
WebResponse	Provides a response from a Uniform Resource Identifier (URI). This class is abstract.

2- System.Net.Socket Namespace Classes

وتركز بشكل اساسي على برمجة الـTransport Layer وخاصة TCP & UDP Socket Ptogramming ومن اهم الـClasses التي تدعمها:

Class	Description
LingerOption	Contains information about the amount of time it will remain available after closing with the presence of pending data (the socket's linger time).
MulticastOption	Contains IP address values for IP multicast packets.
NetworkStream	Provides the underlying stream of data for network access.
Socket	Implements the Berkeley sockets 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- System.Threading Namespace Classes

يستخدم الـThreading لعمل Session منفصلة عن الـsession المستخدمة في البرنامج وهو اسلوب اخر للـ Asynchronous وتدعم System.Threading مجموعة من الـClasses والتي تستخدم في ادارة عمليات الـThreading وهي كما يلي:

Class	Description
AutoResetEvent	This event notifies one or more waiting threads that an event has occurred.
Interlocked	This class protects against errors by providing atomic operations for variables that are shared by multiple threads.
ManualResetEvent	This event occurs when notifying one or more waiting threads that an event has occurred.
Monitor	This class provides a mechanism that synchronizes access to objects.
Mutex	A synchronization primitive that grants exclusive access to a shared resource to only one thread. It can also be used for inter-process synchronization.
ReaderWriterLock	This class defines a lock that allows single-writer and multiple-reader semantics.
RegisteredWaitHandle	This class represents a handle that has been registered when calling the RegisterWaitForSingleObject() method.
SynchronizationLockException	This exception is thrown when a synchronized method is invoked from an unsynchronized block of code.
Thread	This class creates and controls a thread, sets its priority, and gets its status.
ThreadAbortException	This exception is thrown when a call is made to the Abort() method.
ThreadExceptionEventArgs	This class provides data for the ThreadException event.
ThreadInterruptedException	This exception is thrown when a thread is interrupted while it is in a waiting state.
ThreadPool	This class provides a pool of threads that can be used to post work items, process asynchronous I/O, wait on behalf of other threads, and process timers.
ThreadStateException	This is the exception that is thrown when a thread is in an invalid state for the method call.
Timeout	This class simply contains a constant integer used when we want to specify an infinite amount of time.
Timer	This class provides a mechanism for executing methods at specified intervals.
WaitHandle	This class encapsulates operating system-specific objects that wait for exclusive access to shared resources.

اهم الـMethods التي تستخدم في ادارة الـThreading في الدوت نيت :

Public Method Name	Description
Abort()	This overloaded method raises a ThreadAbortException in the thread on which it is invoked, to begin the process of terminating the thread. Calling this method usually terminates the thread.
AllocateDataSlot()	This static method allocates an unnamed data slot on all the threads.
AllocateNamedDataSlot()	This static method allocates a named data slot on all threads.
FreeNamedDataSlot()	This static method frees a previously allocated named data slot.
GetData()	This static method retrieves the value from the specified slot on the current thread, within the current thread's current domain.
GetDomain()	This static method returns the current domain 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 الـProparites الخاصة بالـThreading :

Public Property Name	Description
ApartmentState	Sets or gets the apartment state of this thread.
CurrentContext	This static property gets the current context in which the thread is executing.
CurrentCulture	Sets or gets the culture for the current thread.
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.

تم بحد الله ...

مجموعة فصول من كتاب احترف برمجة الشبكات وبروتوكول TCP/IP باستخدام C# والـVB.NET ....

كافة الفصول فقط في النسخة الورقية من الكتاب

الموقع الرسمي للكتاب

www.fadidotnet.org

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

لطلب أ والاستفسار أو التوزيع يرجى الاتصال على احد العناوين التالية

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

With My Best Wishs

FADI Abdel-qader Abdel-qader, Jordan

Dot Net Networks & TCP/IP Programming