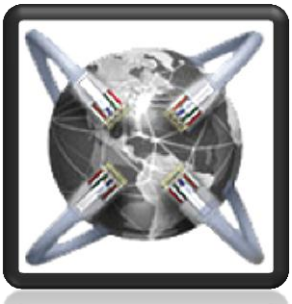
Chapter 3

TCP/UDP Synchronous Socket Programming

المواضيع الرئيسية فى هذا الفصل:

1- مدخل إلى برمجة الـ Socket بالدوت نت
2- معمارية الـ Synchronous TCP Socket Programming
3- معمارية الـ Synchronous UDP Socket Programming Including Unicast, Multicast and Broadcast Socket



3.1: مدخل إلى برمجة الـSocket بالدوت نت :

تعرف الـSocket على أنها الأداة المنطقية التي يتم من خلالها ربط كل من الـTransport Layer والـNetwork Layer في بقية طبقات الـ TCP/IP ومن خلالها يمكننا تعريف البروتوكول المستخدم في عملية النقل والـPort الخاص به وكذلك الـIP Address لطرف المقابل. وقد تم دعم الـ .NET Framework بمجموعة من الـClasses والتي تمكننا من التعامل مع الـSocket ضمن الـSystem.Net.Socket Namespace ويبين الجدول التالي أهم هذه الـClasses:

Class	Description
Socket	وهو الـClass الأساسي لتعامل مع الـSocket بالدوت نت ويحتوي على جميع التعريفات والإعدادات المحتمل التعامل معها إثناء عمل أي نوع من الاتصال باستخدام أي نوع من بروتوكولات النقل في الـTransport Layer و الـNetwork Layer حيث يتم تعريفها وتحديدها بشكل يدوي.
TcpClient TcpListener UdpClient	تعرف هذه الـClasses بالـHelper Socket Classes وهي بالأساس Simple Socket Classes معرفة وجاهزة لاستخدامها بشكل مباشر بالبرنامج بحيث تعتمد الإعدادات الافتراضية والهدف من وجودها تسهيل التعامل مع الـSocket بشكل عام وهي بديل عن استخدام الـSocket Class لمن يريد استخدام الـSocket لإرسال بيانات مباشرة بأحد البروتوكولين الـTCP أو الـUDP.

ويبين الشكل التالي التركيب العام لـSocket Class بالدوت نت:

Socket Class Declaration:

```
System.Net.Sockets.Socket MySocket = new Socket(
AddressFamily. // Network Layer Example: InterNetwork
,SocketType. //Socket Transport Type For Example: Stream
,ProtocolType. //Transport Protocol For Exmaple TCP );
```

حيث يتم في الباروميتر الأول تحديد الـNetwork Layer Protocol والذي سوف تتعامل معه في تحديد عنوان المرسل والمستقبل ويعطيك عدد من الخيارات ومنها IPX والمستخدم في شبكات الـNovel أو ATM أو NetBIOS Address وغيرها، ومن أهم هذه الخيارات الـInterNetwork ويوضع عند التعامل مع بروتوكول الـ IPv4 والـInterNetworkV6 والذي يوضع عند التعامل مع الـIPv6.

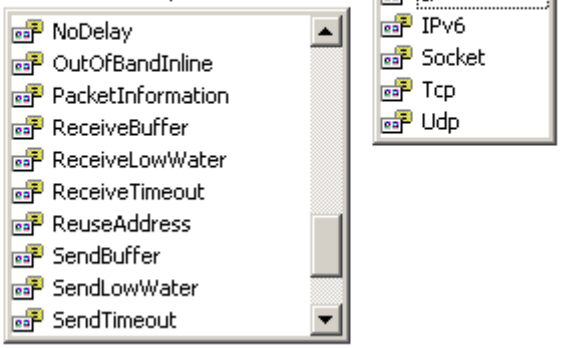
في الباروميتر الثاني يتم تحديد الـSocket Transport Type وينقسم أسلوب النقل في العادة إلى ثلاثة أقسام رئيسية وهي Stream أو Dgram (Data gram) أو Raw Socket وبشكل دائم يتم استخدام الـStream في حالة اعتماد الـ TCP في عملية النقل والـ Dgram في حالة التعامل مع الـUDP في عملية النقل و الـ Raw Socket مع البروتوكولات التي لا تطلب التعامل مع الـTransport Layer Protocols بحيث لا تحتاج في عملية التراسل إلى استخدام Port محدد ومن الأمثلة عليها الـ ICMP (لمزيد من المعلومات انظر الفصل السابع Raw Socket Programming)، وأخيرا نحدد نوع البروتوكول المستخدم للاتصال هل هو UDP أو TCP أو بروتوكولات أخرى مثل الـ ICMP Internet Control Message Protocol أو الـ IGMP Internet Group Management Protocol وغيرها، وهنا سوف نختار الـTCP أو UDP ومن المعروف أن برتوكول الـTCP هو برتوكول موجه وهذا يعني إجراء عملية التحقق من الوصول والتوصيل إلى شخص ما

محدد أما برتوكول الـUDP فهو برتوكول سريع نسبيا و لكنه لا يدعم عملية التحقق من الوصول السليم للبيانات المرسلة وهو مفيد جدا لإجراء عملية البث Broadcast وإنشاء مجموعات البث Multicast Group وهو ما سيتم شرحه لاحقا في هذا الفصل.

3.1.1: التعديل على خواص الـSocket باستخدام الدالة SetSocketOption:

يمكننا من خلال الخاصية SetSocketOption التعديل في أي أمر له علاقة بعملية الاتصال سواء على مستوى الـNetwork Layer أو الـTransport Layer أو حتى على مستوى الـ Socket نفسه كما وسنحتاج هذه الدالة في تفعيل الـMulticasting و الـBroadcasting على الـSocket أما الصيغة العامة لها فهي كالتالي:

```
.SetSocketOption(SocketOptionLevel. ,
SocketOptionName, value);
```



من خلال الـSocket Object نختار الدالة SetSocketOption حيث تحتوي على ثلاثة باروميترات الأول يحدد مستوى التعديل المطلوب Socket Option Level وتقسم إلى ثلاثة مستويات:

الأول: على مستوى طبقة الـNetwork Layer ويتبين بالـIP والـIPv6 وسنحتاجه عند الحاجة لتعديل شيء ما على مستوى الـIP مثل الـTTL أو إدارة الـMulticasting.

الثاني: على مستوى طبقة الـTransport ويحدد بالـTCP والـUDP Transport Protocols.

الثالث: على مستوى الـSocket ويحدد عند الحاجة لتعديل في كلا الطبقتين الـNetwork والـTransport.

كما يحدد في الباروميتر الثاني نوعية الـ Option التي نريد تفعيلها أو منعها أو تغيير في قيمها ويبين الجدول 3.1.1.1 ما يمكننا القيام به من خلال هذه الدالة.

سيتم التطرق إلى هذه الـOptions خلال موضوعات فصول الكتاب عند الحاجة إليها.

ويوضع في الباروميتر الأخير الـ Object Value فإذا كانت العملية هي السماح أو الرفض عندها تأخذ إما True أو False وإذا كانت تغيير أو تحديد قيم ما فتأخذ القيمة التي نريد تغييرها.

Member name	Description
AcceptConnection	Socket is listening.
AddMembership	Add an IP group membership.
AddSourceMembership	Join a source group.
BlockSource	Block data from a source.
Broadcast	Permit sending broadcast messages on the socket.
BsdUrgent	Use urgent data as defined in RFC-1222. This option can be set only once, and once set, cannot be turned off.
ChecksumCoverage	Set or get UDP checksum coverage.
DontFragment	Do not fragment IP datagrams.
DontLinger	Close socket gracefully without lingering.
DontRoute	Do not route; send directly to interface addresses.
DropMembership	Drop an IP group membership.
DropSourceMembership	Drop a source group.
ExclusiveAddressUse	Enables a socket to be bound for exclusive access.
HeaderIncluded	Indicates application is providing the IP header for

	outgoing datagrams.
IPOptions	Specifies IP options to be inserted into outgoing datagrams.
IpTimeToLive	Set the IP header time-to-live field.
KeepAlive	Send keep-alives.
MaxConnections	Maximum queue length that can be specified by Listen.
MulticastInterface	Set the interface for outgoing multicast packets.
MulticastLoopback	IP multicast loopback.
MulticastTimeToLive	IP multicast time to live.
NoChecksum	Send UDP datagrams with checksum set to zero.
NoDelay	Disables the Nagle algorithm for send coalescing.
OutOfBandInline	Receives out-of-band data in the normal data stream.
PacketInformation	Return information about received packets.
ReceiveBuffer	Send low water mark.

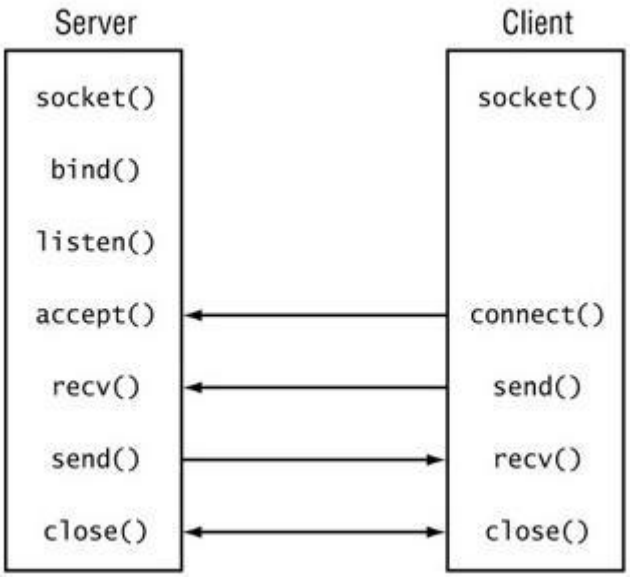
ReceiveLowWater	Receive low water mark.
ReceiveTimeout	Receive time out.
ReuseAddress	Allows the socket to be bound to an address that is already in use.
SendBuffer	Specifies the total per-socket buffer space reserved for sends. This is unrelated to the maximum message size or the size of a TCP window.
SendLowWater	Specifies the total per-socket buffer space reserved for receives. This is unrelated to the maximum message size or the size of a TCP window.
SendTimeout	Send timeout.
TypeOfService	Change the IP header type of service field.
UnblockSource	Unblock a previously blocked source.
UseLoopback	Enable Or Disable a Loop Back Connection

الجدول 3.1.1.1 ويبين إعدادات الـSocket التي يمكننا القيام بها من خلال الدالة SetSocketOption
(المصدر Microsoft MSDN)

3.2: معمارية الـSynchronous TCP Socket Programming:

لإجراء أي عملية اتصال من خلال الـTCP لابد في البداية من وجود طرف المستقبل وفي العادة يكون ذلك الطرف أما الـServer في حالة الـClient/Server Architecture أو الـPeer Client في حالة Peer-to-Peer Architecture لمزيد من المعلومات أنظر الفصل الحادي عشر.

ويمكننا استخدام الـTCP فقط في حالة التراسل كـUnicast وبالتالي هنالك طرف واحد فقط يستطيع استقبال البيانات بحيث يخصص Connection لكل متصل، ويمر الـTCP بمجموعة من المراحل ويوضح الشكل 3.1 هذه المراحل لكلا الطرفين المتصل والمشار إليه بالـClient والمستضيف والمشار إليه بالـServer.



الشكل 3.1 ويبين مراحل الاتصال من خلال الـTCP بين طرفين المتصل والمستضيف

ولإنشاء برنامج اتصال بسيط باستخدام الـTCP يجب تعريف الـSocket Class عند كلا الطرفين المتصل والمستضيف بالشكل التالي:

TCP Socket Class Declaration:

```
System.Net.Sockets.Socket MySocket = new Socket(
    AddressFamily.InterNetwork  //InterNetwork = IPv4
  , SocketType.Stream // Streaming Session
  , ProtocolType.Tcp);
```

في طرف المتصل يتم من خلال الدالة Connect بدأ محاولة الاتصال بالطرف المقابل ويمرر إليها الـ IP Address والـPort Number لطرف المستضيف مباشرة أو من خلال الـIPEndPoint Class.

TCP Socket - Using The Connect Method:

```
// Pass The IP Address and The Port Number Directly
MySocket.Connect("127.0.0.1", 5000);

// Or Pass it By Using The IPEndPoint Class
System.Net.IPEndPoint ServerAddress = new
System.Net.IPEndPoint(System.Net.IPAddress.Parse("127.0.0.1"), 5000);
MySocket.Connect(ServerAddress);
```

بعد النجاح في عملية الاتصال يتم عمل Stream Session منطقي بين المتصل والمستضيف وتستطيع من خلاله البدء في عملية الإرسال والاستقبال ويتم من خلاله أيضا التأكد من أن عملية الاتصال مستمرة بين الطرفين. عند الإرسال لابد في البداية من عملية تحويل البيانات المراد إرسالها إلى Array of Bytes وذلك من خلال عمل Encoding لها ويعتمد ذلك على نوع البيانات المراد إرسالها فمثلا في حالة إرسال نص يتم تحويل ذلك النص من string إلى Bytes بحسب طريقة الترميز الخاصة به إذا كانت Unicode أو ASCII Code كمثال.

TCP Socket - Start Sending & Receiving in The Client Socket:

```
// For Sending:
byte[] TheStringAsByteArray = UnicodeEncoding.Unicode.GetBytes("Hello
World!");
MySocket.Send(TheStringAsByteArray);

// For Receiving:
//The initial declaration of the array indexes
byte[] ReceivedData = new byte[1024];
MySocket.Receive(ReceivedData);
string ReceivedMessage = UnicodeEncoding.Unicode
                  .GetString(ReceivedData);
// To Close The Conncetion:
MySocket.Close();
```

أما في طرف المستضيف فيجب في البداية تعريف الـSocket ثم عمل الـ Bind وهو ربط الـNetwork IP الخاص بالمستضيف بالـTransport Port التي سيتم من خلالها استقبال طلب الاتصال بعد ذلك ومن خلال الدالة Listen تبدأ عملية التصنت على الـTCP Port الذي تم تحديده مسبقا ثم عند استقبال طلب اتصال يتم تحويله مباشرة إلى الدالة Accept والتي تقوم بدورها بإنشاء الـSession بين المتصل والمستقبل ويتم وضعه على Socket Class Object خاص لكي يتم التعامل معه بشكل منفصل في عملية التراسل.

TCP Socket - The Server Part:

```
// The Server Socket Declaration
Socket ServerSocket = new Socket(
AddressFamily.InterNetwork, SocketType.Stream , ProtocolType.Tcp);

// Any = Use All IP Addresses of the Server
IPEndPoint ipend = new IPEndPoint(IPAddress.Any, 5000);

// Bind Both The Network IP and The Transport Port
ServerSocket.Bind(ipend);

// -1= Allow for Unlimited number of Connections
ServerSocket.Listen(-1);

// Make a Special Socket For Each Client
Socket New_Client = ServerSocket.Accept();

// Sending:
byte[] SendingBuffer =
UnicodeEncoding.Unicode.GetBytes("Welcome to Our Server");
New_Client.Send(SendingBuffer);

// Receiving:
byte[] ReceivedBuffer = new byte[1024];
New_Client.Receive(ReceivedBuffer);
string msg = UnicodeEncoding.Unicode.GetString(ReceivedBuffer);
```

ولتكرار عملية الإرسال والاستقبال لا بد من وضع جملة تكرار مثل الـWhile بعد عملية الـListen وبالتالي يبقى المستضيف يعمل إلى أن ينفذ شرط ما كتمرير جملة معينة أو انتهاء عدد الدورات أو جعله يعمل إلى الأبد بوضع True مع جملة التكرار While وفي هذه الحالة فلا بد من فصل دالة الإرسال والاستقبال بوضعها على Thread خاص لكي لا يؤثر ذلك على عمل البرنامج بشكل عام وهو ما تم شرحه سابقا في الفصل الثاني وكمثال على ذلك:

TCP Socket - The Server Part With Using Threading:

```
void StartTheServerOnNewThread()
{
  Thread ServerThread = new Thread(new
                ThreadStart(ServerMethod));
ServerThread.IsBackground = true;
ServerThread.Start();
}
```

```
void ServerMethod()
{
Socket ServerSocket = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);

// Socket Binding
IPEndPoint ipend = new IPEndPoint(IPAddress.Any, 5000);
ServerSocket.Bind(ipend);

// Lestening
ServerSocket.Listen(-1);

while (true)
       {
Socket New_Client = ServerSocket.Accept();
byte[] buffer = new byte[1024];
New_Client.Receive(buffer);
string ReceivedMessage = UnicodeEncoding.Unicode.GetString(buffer);
        }
}
```

3.2.1: الـCross Threading Exception والطريقة الأنسب لحلها:

دائما ما نحتاج إلى عرض البيانات التي تم استقبالها من خلال دالة الـThread على Windows Control موجود على الـForm كمثال الـListbox Windows Form Control وفي العادة ما نقوم به هو التالي:

Cross Thread Exception:

```
void ServerMethod()
{
Socket ServerSocket = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);

// Socket Binding
IPEndPoint ipend = new IPEndPoint(IPAddress.Any, 5000);
ServerSocket.Bind(ipend);

// Lestening
ServerSocket.Listen(-1);

while (true)
       {
Socket New_Client = ServerSocket.Accept();
byte[] buffer = new byte[1024];
New_Client.Receive(buffer);
string ReceivedMessage = UnicodeEncoding.Unicode.GetString(buffer);
listBox1.Items.Add(ReceivedMessage);
        }
}
```

عند تنفيذ البرنامج سيتم إرجاع Cross-Thread operation not allowed exception عند استقبال أي بيانات ومحاولة عرضها على Listbox والسبب في ذلك يعود إلى كون أن Listbox Control يعمل على Thread آخر غير المعرف عليه ذلك الـControl ولتجاوز هذا الأمر ينصح بتمرير البيانات التي تم استقبالها من خلال وسيط وهو الـDelegate حيث يتم عمل Invoke للبيانات المستقبلة من خلاله إلى الـWindows Control ليصبح بشكل التالي:

Solve The Cross Thread Exception:

```
void ServerMethod()
{
Socket ServerSocket = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);
// Socket Binding
IPEndPoint ipend = new IPEndPoint(IPAddress.Any, 5000);
ServerSocket.Bind(ipend);
// Lestening
ServerSocket.Listen(-1);
while (true)
       {
Socket New_Client = ServerSocket.Accept();
byte[] buffer = new byte[1024];
New_Client.Receive(buffer);
string ReceivedMessage = UnicodeEncoding.Unicode.GetString(buffer);

// Create a delegate to handle the cross-thread calls
this.Invoke(new mydelegate(ViewText), ReceivedMessage);
        }
}
// to pass the received text using this delegate
private delegate void mydelegate(string PassedData);
private void ViewText(string PassedData)
{
listbox1.Item.Add(PassedData);
}
```

قمنا في المثال السابق بتعريف delegate يمرر بارومتر من نوع String ثم قمنا بإنشاء دالة ضمن الـMain Thread الرئيسي حيث مررنا من خلالها النص الذي سوف يتم تمريره من خلال الـDelegate باستخدام دالة الـInvoke إلى الـListbox Control ويعتبر هذا الحل هو الحل الأمثل لتجاوز هذا الاستثناء كما يوجد طريقة لإيقاف ذلك الاستثناء وذلك بتمرير قيمة False إلى الخاصية CheckForIllegalCrossThreadCalls ضمن الـControl Class ووضعها على حدث بداية تشغيل البرنامج كما يلي كمثال:

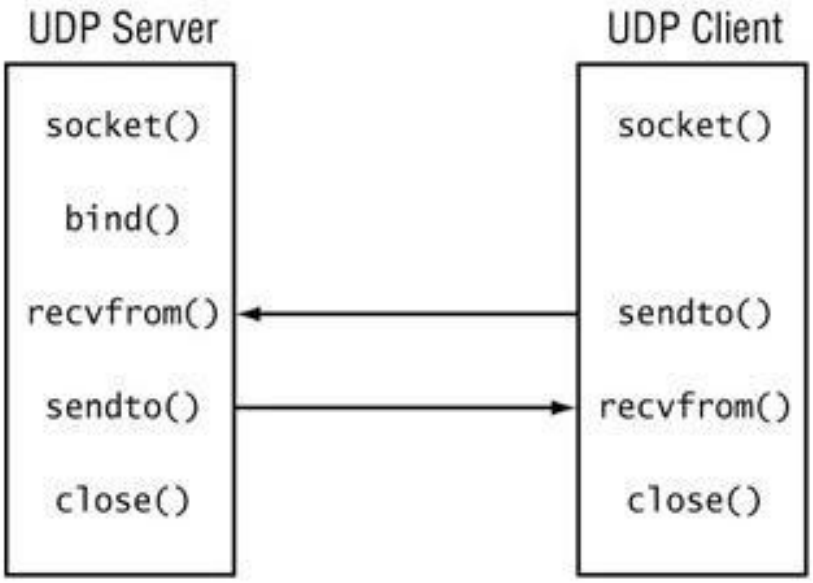
Disable The Cross Thread Exception Warning:

```
public Form1()
{
InitializeComponent();
Control.CheckForIllegalCrossThreadCalls = false;
}
```

لكن يبقى استخدام الطريقة الأولى أفضل لضمان استقرار النظام.

3.3: معمارية الـ Synchronous UDP Socket Programming Including Unicast, Multicast and Broadcast Socket:

إن معمارية الـUDP بشكل عام هي ابسط من الـTCP ففي الـUDP سنحتاج فقط إلى تعريف الـSocket في طرف المتصل ونبدأ الإرسال والاستقبال والسبب في ذلك أن الـUDP لا يقوم بعملية إنشاء Session قبل عملية الإرسال كما هو الحال في الـTCP ويكتفي الـUDP بالإرسال إلى العنوان المحدد فقط والاستقبال كذلك وتعتبر هذه المعمارية أساسية في عملية الإرسال والاستقبال من خلال الـUDP إذ لا يقوم الـUDP بأي عملية تحقق من وصول الرسالة كذلك لا يقوم بعملية التحقق إذا كان الطرف المرسل إليه متصل بالشبكة أم لا ويبين الشكل 3.2 معمارية الاتصال من خلال الـUDP.



الشكل 3.2 ويبين معمارية التراسل من خلال الـUDP

في طرف المستضيف يتم تعريف الـSocket ثم عمل Bind للـNetwork IP والـ Transport Port Number الخاص بالمستضيف ثم تبدأ عملية الإرسال والاستقبال.

ما يميز الـUDP Protocol غير السرعة هو إمكانية استخدامه لإرسال بيانات كـUnicast و Multicast و Broadcast بعكس الـTCP الذي يمكن استخدامه لتراسل كـUnicast فقط والسبب في ذلك معمارية الـUDP التي لا تحتوي على أي عمليات إنشاء اتصال مباشر بين المتصل والمستضيف. سنبين في هذا الفصل كيفية الإرسال والاستقبال من خلال الـUDP بثلاثة طرق الإرسال كـUnicast و Multicast و Broadcast.

3.3.1: معمارية التراسل باستخدام الـUDP كـ Unicast:

في عملية الإرسال أو الاستقبال نكتفي بتعريف الـSocket وتحديد عنوان المستضيف وأخير إجراء عملية الإرسال باستخدام الدالة SendTo و الاستقبال باستخدام الدالة ReceiveFrom والتي يتم فيها إسناد عنوان المرسل إليه أو المستقبل منه.

UDP Socket – Unicast Client:

```
void ClientSender()
{
// Create The UDP Socket
Socket client_socket = new Socket(AddressFamily.InterNetwork,
SocketType.Dgram, ProtocolType.Udp);
// The Unicast IP End Point
IPEndPoint ipend = new IPEndPoint(IPAddress.Parse("127.0.0.1"),
5000);
// Encoding
byte[] buffer = UnicodeEncoding.Unicode.GetBytes("Hello Unicasting
UDP World!");

// Send the Text
client_socket.SendTo(buffer,ipend);

// Close The Socket
client_socket.Close();
}

void ClientReceiver()
{
// Create The UDP Socket
Socket client_socket = new Socket(AddressFamily.InterNetwork,
SocketType.Dgram, ProtocolType.Udp);
// The Unicast IP End Point
IPEndPoint ipend = new IPEndPoint(IPAddress.Parse("127.0.0.1"),
5000);
byte[] buffer = new byte[1024];
EndPoint ip = (EndPoint)ipend;
   while (true)
     {
client_socket.ReceiveFrom(buffer, ref ip);
string ReceivedMessage = UnicodeEncoding.Unicode.GetString(buffer);
     }
}
```

في طرف المستضيف يتم تعريف الـSocket ثم عمل Bind للـIP Network والـ Transport Port Number الخاص بالمستضيف ثم تبدأ عملية الإرسال أو الاستقبال ويتم ذلك كما يلي:

UDP Socket – Unicast Server:

```
void UDP_Server()
{
// Creation Socket
Socket sock = new Socket(AddressFamily.InterNetwork,
SocketType.Dgram, ProtocolType.Udp);

// Socket Binding
IPEndPoint ipend = new IPEndPoint(IPAddress.Any, 5000);
sock.Bind(ipend);

   while (true)
       {
// Receiving
byte[] buffer = new byte[1024];
EndPoint ip = (EndPoint)ipend;
sock.ReceiveFrom(buffer, ref ip);
// Encoding
string ReceivedMessage = UnicodeEncoding.Unicode.GetString(buffer);
        }
}
```

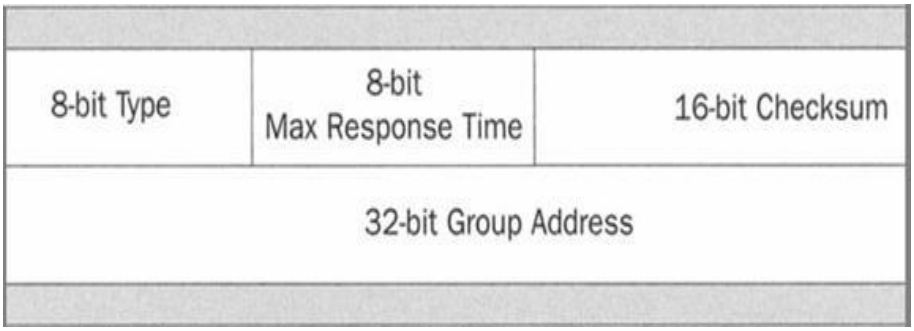
في العادة ما يتم استخدام الـ ReceiveFrom و الـ SendTo مع الـUDP والسبب لأنه لا يوجد إعداد مسبق لعملية الاتصال كما هو الحال في الـTCP حيث يتم فقط تعريف الـSocket والإرسال أو الاستقبال مباشرة لذلك لابد من إرسال عنوان المرسل إليه مع دالة الـSend كما توفر لنا الدالة ReceiveFrom إمكانية معرفة مرسل البيانات من خلال إسناد قيمة الـ EndPoint object كـReference Value إليها ويمكننا عرض عنوان صاحب الرسالة ورقم الـPort الذي تم الإرسال من خلاله بتحويل قيمة الـEndPoint إلى string كما يلي: ip.ToString(). كما يمكن أيضا استخدام ذلك العنوان للإرسال رد إلى المرسل وذلك بإسناد قيمة الـEndPoint التي تم إسنادها من خلال الـReceiveFrom إلى الـSendTo كمثال:

UDP Socket – Send a Replay By Using The EndPoint Object:

```
EndPoint ip = (EndPoint)ipend;
sock.ReceiveFrom(buffer, ref ip);
// To Send a Replay, For Example:
byte[] buffer = UnicodeEncoding.Unicode.GetBytes("Received,
Thanks.");
sock.SendTo(buffer, ip);
```

3.3.2: معمارية التراسل باستخدام الـUDP كـ Multicast:

يعرف الـMulticast على أنه الإرسال من شخص إلى مجموعة One-to-many باستخدام الـ Class D Subnet Mask في الـIP Protocol وتتم عملية نقل البيانات من خلال الـ UDP Transport Protocol ويستخدم بروتوكول خاص لعملية إدارة المجموعات هو الـ IGMP – Internet Group Management Protocol والذي هو جزء من الـInternet Protocol Model وكما يتضح من الشكل 3.3 فإن برتوكول الـIGMP يحتوي على عمليات التحقق من الوصول السليم للبيانات حيث يتم إرسال حجم البيانات الكلي لرسالة، و تحتوي أيضا على الـMax response Time والذي يحدد فيه الفترة الزمنية للـ corresponding report، وكذلك نوع العملية (ضم إلى مجموعة Join to Group، الخروج من مجموعة Drop From Group، أو إرجاع معلومات عن المجموعة Membership Query) وأخيرا عنوان الـ IP للمجموعة.



الشكل 3.3 ويوضح تركيب بروتوكول الـIGMP

تم تخصيص الـRange في الـIP Multicasting من 224.0.0.0 إلى 239.255.255.255 ونستطيع تحديده بثلاثة طرق إما بشكل يدوي Static أو Dynamic أو على أساس الـScope-Relative (لمزيد من المعلومات حول تخصيصات الـMulticast Ranges انظر الرابط التالي:

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

يتم نقل الـMulticast Packets بين الـBackbone Tunnels كـUnicast حيث يتم إرسالها من داخل الشبكة إلى الـRouter و ترسل من Router إلى أخر عبر الـBackbone Tunnel بأسلوب الـUnicast وهو ما يوفر الكثير من الـBandwidth في الشبكة حيث ترسل نسخة واحدة إلى الـRouter ويقوم هو بتوزيعها على الأجهزة كـBroadcast.

سنقوم في البداية بشرح كيفية التعامل مع العمليات الأساسية بالدوت نت والتي سنحتاجها للإرسال أو الاستقبال من الـMulticast Group كالانضمام أو الخروج من مجموعة و الإرسال والاستقبال من مجموعة وشرح بعض الإعدادات الضرورية في عملية الإرسال كـMulticasting.

3.3.2.1: الانضمام أو الخروج من مجموعة Join || Drop a multicast Group :

تتم عملية الانضمام إلى مجموعة Join a Group بالشكل التالي بعد تعريف الـSocket:

UDP Socket – Join to Multicast Group:

```
// Socket Declaration
Socket MySocket = new Socket(AddressFamily.InterNetwork,
                  SocketType.Dgram, ProtocolType.Udp);

// IP Declaration
IPAddress Multicast_IP = IPAddress.Parse("224.0.0.1");

// Join to The Multicast Group Using The SocketOption
MySocket.SetSocketOption(SocketOptionLevel.IP,
SocketOptionName.AddMembership, new MulticastOption(Multicast_IP));
```

حيث أن الـMulticasting هو جزء من الـIP Infrastructure فإن التغيير في الـSocket Option سيتم على مستوى الـIP لذلك تم اختيار SocketOptionLevel.IP ، في الباروميتر الثاني حددنا نوع الإجراء الذي سيتم على مستوى الـIP وقد قمنا بتعريف نوع الإجراء بـAddMembership في الباروميتر الثالث يتم وضع القيمة الجديدة لتغيير المطلوب وقد قمنا بتمرير الـIP Multicast وبنفس الطريقة يمكن حذف العضوية وذلك باختيار SocketOptionName.DropMembership في الـSocketOption.

3.3.2.2: الإرسال و الاستقبال Sending And Receiving From a multicast Group :

قبل عملية الإرسال يفضل تحديد عدد الـTTL – Time To Live والمقصود بها عدد الـRouters التي يمكن أن تمر الـMulticast Packets من خلالها حيث إذا تم وضع قيمة 1 فهذا يعني أن الـ Multicast Packets لن تخرج عن الـLocal Network والقيمة التي يفضل أن يتم وضعها هنا تعتمد على طبيعة الـInfrastructure لشبكة التي سيعمل عليها البرنامج إذا كان من الضروري توسيع مجال الإرسال إلى شبكات أخرى ضمن الشبكة المحلية أم لا ويتم ذلك كما يلي كمثال:

UDP Socket – Set The Number Of TTL In a Multicasting Socket:

```
// Set The Number Of TTL in the SetSocketOption
MySocket.SetSocketOption(SocketOptionLevel.IP,
SocketOptionName.MulticastTimeToLive, 1);
```

أما عملية الإرسال فيمكن أن تتم بطريقتين الأولى من خلال الدالة Send حيث يلزم عمل Conncet مع الـMulticast Group والثانية الإرسال مباشرة من خلال الدالة SendTo حيث يتم تمرير البيانات المراد إرسالها وعنوان الـMulticast Group الذي نريد أن نرسل إليه كما يلي كمثال:

UDP Socket – Sending In Multicasting:

```
Byte[] buffer = Encoding.Unicode.GetBytes("Hello My Group");

IPEndPoint IPEnd = new IPEndPoint(Multicast_IP, Port_Number);

// The First Way (Send After Joining)
MySocket.Connect(IPEnd);
MySocket.Send(buffer);

// The Seond Way (Send Directly)
MySocket.SendTo(buffer, IPEnd);
```

أما الاستقبال فيمكن أن يتم بطريقتين أيضا أما من خلال الـReceive أو الـReceiveFrom ونستفيد من الأخيرة بإمكانية معرفة عنوان مرسل الرسالة.

UDP Socket – Receiving In Multicasting:

```
// Binding
IPEndPoint ipend = new IPEndPoint(IPAddress.Any, Port_Number);
MySocket.Bind(ipend);

// Join To The Multicast Group
IPAddress Multicast_Group = IPAddress.Parse("224.0.0.1");
MySocket.SetSocketOption(SocketOptionLevel.IP,
SocketOptionName.AddMembership, Multicast_Group);

byte[] buffer = new byte[1024];
EndPoint ip = (EndPoint)ipend;

// The First Way if we need to know the sender address:
MySocket.ReceiveFrom(buffer, ref ip);
string ReceivedFrom = ip.ToString();

// The Second Way:
MySocket.Receive(buffer);

string ReceivedMessage = UnicodeEncoding.Unicode.GetString(buffer);
```

3.3.3: معمارية التراسل باستخدام الـUDP كـ Broadcasting:

تعني عملية الإرسال كـBroadcasting الإرسال إلى كل من هو على الشبكة بغض النظر عن الـIP الذي يملكه المستقبل ولا تختلف عملية الإرسال كـBroadcasting كثيرا عن الـUnicasting حيث أن الاختلاف فقط بالسماح بتفعيل الـBroadcasting على الـSocket وتمرير عنوان الـBroadcast إلى الـIPEndPoint عند الإرسال أما الاستقبال فهو تماما كالاستقبال في الـUncasting.

UDP Socket – Sending as Broadcasting:

```
// Normal UDP Socket declaration
Socket MySocket = new Socket(AddressFamily.InterNetwork,
SocketType.Dgram, ProtocolType.Udp);

// Declare the EndPoint destination as Broadcast
IPEndPoint iep = new IPEndPoint(IPAddress.Broadcast, Port_Number);

// Enable The Broadcasting at The Socket Level
MySocket.SetSocketOption(SocketOptionLevel.Socket,
SocketOptionName.Broadcast, true);

// Send The Data Normally
byte[] data = Encoding.ASCII.GetBytes("Hello All");
MySocket.SendTo(data, iep);

// Finally Close The Socket
MySocket.Close();
```

في الـIPEndPoint يمكن تمرير عنوان الـBroadcast الخاص بشبكتك بشكل يدوي كوضع العنوان "255.255.255.255" كمثال (لمزيد من المعلومات حول الـBroadcast IP وكيفية حسابه أنظر الفصل الأول) أو يمرر بشكل آلي من خلال تمرير الـ IPAddress.Broadcast.

أما الدالة SetSocketOption فقد قمنا بتمرير قيمة الـSocket Level في الباروميتر الأول وفي الثاني حددنا نوع العملية المطلوبة وهي تفعيل الـBroadcast على الـSocket و في الباروميتر الثالث مررنا قيمة true لسماح بالإرسال كـBroadcast.

UDP Socket – Receiving In Broadcasting:

```
// Normal UDP Socket declaration
Socket MySocket = new Socket(AddressFamily.InterNetwork,
SocketType.Dgram, ProtocolType.Udp);

IPEndPoint iep = new IPEndPoint(IPAddress.Any, Port_Number);
MySocket.Bind(iep);

EndPoint ep = (EndPoint) iep;

byte[] buffer = new byte[1024];
MySocket.ReceiveFrom(buffer, ref ep);

string ReceivedMessage =
       Encoding.ASCII.GetString(buffer);

string ReceivedFrom = ep.ToString();
```

الخلاصة:

بينا في هذا الفصل طريقة التعامل مع الـSynchronous Socket بجميع الطرق في الـTCP والـUDP ومعمارية الإرسال بالـSocket كـUnicast و Multicast وأخيرا كـBroadcast كما وبينا كيفية التحكم بخصائص الـSocket من خلال الدالة SetSocketOption. سيتم الحديث في الفصل التالي عن الـAsynchronous Socket وبرمجتها بجميع الطرق في الـTCP والـUDP.