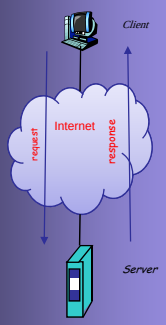


INTERNET PROTOCOLS AND CLIENT-SERVER PROGRAMMING SWE344

Fall Semester 2008-2009 (081)

Module 12: Broadcasting & Multicasting

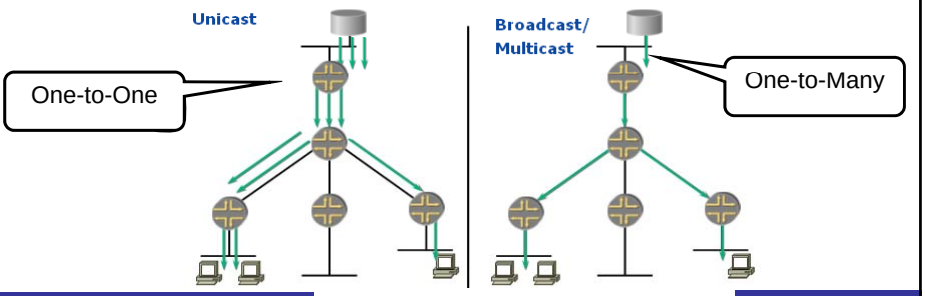
Dr. El-Sayed El-Alfy
Computer Science Department
King Fahd University of Petroleum and Minerals
alfy@kfupm.edu.sa

Objectives

- Learn about broadcasting and its advantages over unicasting
- Learn how to write broadcasting applications in C#
- Learn about Multicasting and how it compares with Broadcasting
- Learn about the IP Addresses used for Multicasting
- Learn about the Internet Group Management Protocol (IGMP) used for multicasting
- Learn how to write multicast applications in C#

Introduction

- Unicasting is a communication model in which a separate instance of a message is sent repeatedly to each client.
  - Example: Our UDP chat system in the UDP Lab
- Multicasting and Broadcasting models reduce network traffic by simultaneously delivering a single message to many clients.
  - Applications that take advantage of these models include video conferencing, distance learning, stock quotes, news, etc.



What is Broadcasting?

- Broadcasting involves the use of a special IP address (the broadcast address) to send the same packet of information to all clients that share the same network or subnet.
- Broadcasting requires the use of UDP since TCP requires connections between communicating devices.
- IP address format allows two types of broadcast addresses: local broadcast addresses and global broadcast addresses.

What is Broadcasting?...

Local Broadcast Address:

- The local broadcast address is used to send broadcast address to all devices in a particular subnet.
- Recall that an IP address consists of two components, the network part and the host part.
- The local broadcast IP address consists of the network address of a subnet together with all ones (255 in decimal) in the host part.
  - Example, for a class B network: 192.168.0.0, using the default subnet mask of 255.255.0.0, the local broadcast address is 192.168.255.255.
- If the network is subdivided using subnet mask 255.255.255.0, then each subnet will have its own local broadcast address.
  - Example, the subnet 192.168.200.0 will have the local broadcast address of 192.168.200.255.

What is Broadcasting?...

Global Broadcast Address:

- The global broadcast address was originally intended to allow a device to send packets to all devices on an inter-network.
  - It is the special IP address consisting of all ones: 255.255.255.255.
- The reality of the Internet (its popularity and security issues) dictated that global broadcasting is not feasible due to the possibility of using it to crash the Internet.
- Routers do not send global IP broadcast to other networks unless specifically configured to do so, which is practically never.
- Instead, routers silently ignore global broadcast messages, effectively making it a local broadcast.

Writing Broadcast Applications in C#

- By default, sockets are not allowed to send broadcast messages – doing so will cause a Socket Exception.
- To send broadcast packets, the broadcast socket option must be set on the socket using the ***SetSocketOption*** method

```
Socket socket = new  Socket(AddressFamily.InterNetwork,
                                SocketType.Dgram, ProtocolType.Udp);
socket.SetSocketOption(SocketOptionLevel.Socket,
                       SocketOptionName.Broadcast, 1);  //1 indicates true
```

OR `socket. EnableBroadcast = true; //.NET 2.0`

- After the socket option is set, broadcast address and a port number are then used to create an EndPoint for the broadcast

```
IPEndPoint endPoint=new IPEndPoint(IPAddress.Broadcast, 9090);
byte[] data = Encoding.ASCII.GetBytes("test message");
socket.Send(data, endPoint);
```


Example 1: Broadcast Echo System

```
1.  using System;
2.  using System.Net;
3.  using System.Net.Sockets;
4.  using System.Text;
5.  public class SimpleBroadcastSender {
6.     public static void Main() {
7.       Socket sock = new Socket(AddressFamily.InterNetwork,
8.                         SocketType.Dgram, ProtocolType.Udp);
9.       sock.SetSocketOption(SocketOptionLevel.Socket,
10.                           SocketOptionName.Broadcast, 1);
11.      IPEndPoint endPoint=new IPEndPoint(IPAddress.Broadcast,9090);
12.      while (true) {
13.        Console.Write("Enter message to broadcast: ");
14.        string message = Console.ReadLine();
15.        if (message=="")
16.             break;
17.        byte[] data = Encoding.ASCII.GetBytes(message);
18.        sock.SendTo(data, endPoint);
19.      }
20.      sock.Close();
21.    }
22. }
```

Example 1: Broadcast Echo System …

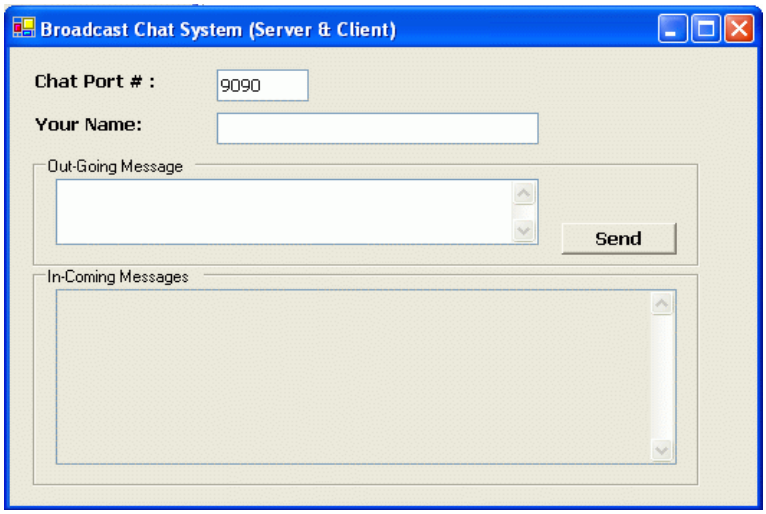
```
1.  using System;
2.  using System.Net;
3.  using System.Net.Sockets;
4.  using System.Text;
5.  public class SimpleBroadcastReceiver {
6.     public static void Main(){
7.        Socket sock = new Socket(AddressFamily.InterNetwork,
8.                        SocketType.Dgram, ProtocolType.Udp);
9.        IPEndPoint localEP = new IPEndPoint(IPAddress.Any, 9090);
10.       sock.Bind(localEP);
11.       EndPoint remoteEP = new IPEndPoint(IPAddress.Any, 0); //dummy
12.       byte[] data;  int size;
13.       string message;
14.       while (true) {
15.         data = new byte[1024];
16.         size = sock.ReceiveFrom(data, ref remoteEP);
17.         message = Encoding.ASCII.GetString(data, 0, size);
18.         Console.WriteLine("received: {0}  from: {1}",
19.                                 message, remoteEP.ToString());
20.       }
21.    }
22. }
```

No special socket option is required to receive broadcast packets. Just listen to the broadcast port.

Example 2: Broadcast Chat System

- In the UDP Lab, we developed a chat system where clients send their messages to a server, which then broadcast the message to all known clients.
- There were a number of limitations with that chat system:
  - The client must send a message to the server before it can be recognized as a known client.
  - The same message is repeatedly sent individually to each client – thus, wasting bandwidth.
  - For each message sent, the server has to search its database to check if the client is known – an expensive process as the number of clients increases.
- The next example shows a broadcast chat system that does not have the above limitations.
  - In the example, one program is used both to send and to receive broadcast messages.

Example 2: Broadcast Chat System …



Example 2: Broadcast Chat System …

Since this program is bound to a local end point, two instances of it cannot be run on the same machine.

```
1.   public partial class Form1 : Form {
2.       private Socket sendSocket, receiveSocket;
3.       private EndPoint remoteEP, localEP, broadcastEP;

4.       public Form1()   {
5.           InitializeComponent();
6.           Control.CheckForIllegalCrossThreadCalls = false;
7.           remoteEP = new IPEndPoint(IPAddress.Any, 0); //dummy
8.           localEP = new IPEndPoint(IPAddress.Any,
9.                                    int.Parse(txtPort.Text));
10.          receiveSocket = new Socket(AddressFamily.InterNetwork,
11.                                     SocketType.Dgram, ProtocolType.Udp);
12.          receiveSocket.Bind(localEP);
13.          broadcastEP = new IPEndPoint(IPAddress.Broadcast,
14.                                       int.Parse(txtPort.Text));
15.          sendSocket = new Socket(AddressFamily.InterNetwork,
16.                                  SocketType.Dgram, ProtocolType.Udp);
17.          sendSocket.SetSocketOption(SocketOptionLevel.Socket,
18.                                     SocketOptionName.Broadcast, 1);
19.          ThreadPool.QueueUserWorkItem(new WaitCallback(ReceiveData));
20.      }
```

Example 2: Broadcast Chat System …

```
21.     void ReceiveData(Object obj)
22.     {
23.         while (true)
24.         {
25.             byte[] data = new byte[2048];
26.             int recv = receiveSocket.ReceiveFrom(data,
27.                                 SocketFlags.None, ref remoteEP);
28.             txtIn.Text += Encoding.ASCII.GetString(data,0,recv)+"";
29.         }
30.     }

31.     private void btnSend_Click(object sender, EventArgs e)
32.     {
33.         byte[] data = Encoding.ASCII.GetBytes(txtName.Text +
34.                                                 ": " + txtOut.Text);
35.         sendSocket.SendTo(data, data.Length, SocketFlags.None,
36.                broadcastEP);
37.         txtOut.Text = "";
38.     }
```


What is Multicasting?

- Broadcasting is an excellent way to send information to all devices on a subnet. However, it has one serious drawback:
  - The broadcast packets are restricted to the local network or subnet.
  - Multicasting was designed to address this drawback.
- Multicasting allows applications to send a single packet across networks to a select subset of devices called a **multicast group**.
- Each multicast group is identified by a single special IP address (to be discussed next).
- A packet sent with the particular IP address as the destination address will be received by each member of the group.
- Multicast group is a dynamic group. That is, members can join and leave the group at any time.

Multicast IP Addresses

- IP multicasting uses a particular range of IP addresses to designate different multicast groups.
- The class D IP addresses in the range: 224.0.0.0 through 239.255.255.255 are used to represent multicast groups.
  - However, some of these addresses are reserved for special purposes as discussed below:
- The class D addresses are further divided into different blocks:

Local Control Block:

- Addresses in the range: 224.0.0.0 through 224.0.0.255 are reserved for used by network protocols on a local network.
  - For example, 224.0.0.1 represents all systems on this subnet. 224.0.0.2 represents all routers on this subnet.

Multicast IP Addresses …

Global Scope:

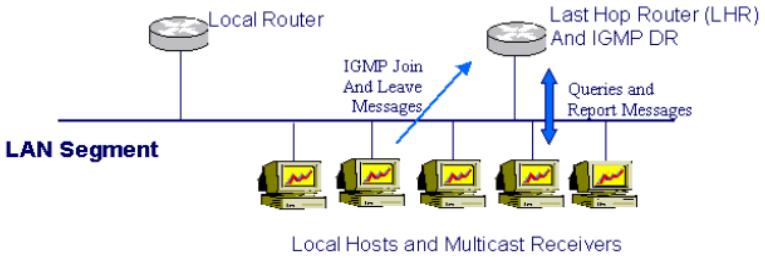
- Addresses in the range: 224.0.1.0 through 238.255.255.255 are called globally scoped addresses. That is, they can be used to multicast data across the Internet.
- Global Scope addresses are allocated by Internet Assigned Numbers Authority (IANA).

Limited Scope:

- Addresses in the range 239.0.0.0 through 239.255.255.255 are called limited scope addresses.
- Routers are normally configured to prevent multicast traffic with these addresses from crossing over the local network.
- More detailed information about multicast addresses can be found at: http://www.iana.org/assignments/multicast-addresses

Internet Group Management Protocol (IGMP)

- Since Multicast packets can cross over networks, routers must be involved in forwarding such packets.
- IGMP is the protocol used by hosts to register their dynamic multicast group membership with their local router. It is also used by routers to discover multicast group members.
- A router only forward packets for a particular multicast group to the local network if members exists for that group.

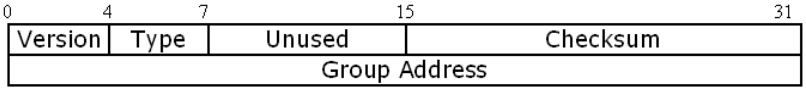


Internet Group Management Protocol (IGMP) …

- Two main versions of the IGMP protocol are in common used:

IGMP Version 1 (RFC 1112)

- The following figure shows the format of IGMP version 1 packet.

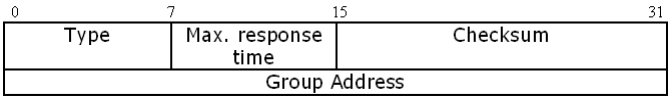


- IGMP Version 1 has just two types of IGMP messages:
  - Membership query
  - Membership report
- Hosts send IGMP membership report to their router to indicate their interest in joining a particular multicast group.
- Membership query is sent by routers periodically to verify that at least one host on the subnet is interested in receiving traffic directed to a particular multicast group.
- If there is no reply to three consecutive IGMP membership queries, the router times-out the group and stops forwarding traffic for the group.

Internet Group Management Protocol (IGMP) …

IGMP Version 2 (RFC 2236)

- The following shows the format of IGMP version 2 packet.



- IGMP Version 2 has four types of IGMP messages:
  - Membership query
  - Version 1 membership report
  - Version 2 membership report
  - Leave group
- The main difference with version 1 is that there is a leave group message.
- A Host can now inform the local router its intention to leave a group.
  - This triggers the router to send membership query to determine if there are any remaining hosts interested in the group (earlier than the normal query period).
  - If there are no replies, the router times-out the group.
  - Thus, version 2 can greatly reduce the leave latency compared to version 1.

Writing Multicast Applications in C#

- The .NET supports IP multicasting by using Socket options.
- There are two socket options that are used to join and to leave a multicast group respectively.
- These options are defined as IP-Level Socket options names, AddMembership and DropMembership.
- The value for each of these options is an instance of the MulticastOption class.
- The MulticastOption class has two constructors as follows:

MulticastOption(IPAddress multicastGroup)	IPAddress specifies the multicast group address. If the machine has more than one interface, all of them are affected by the socket option
MulticastOption(IPAddress mulicatGroup, IPAddress local)	The second address specifies the interface to be used for the multicast.

Example 3: Multicast Receiver

```
1.   using System;
2.   using System.Net;
3.   using System.Net.Sockets;
4.   public class SimpleMulticastReceiver {
5.      public static void Main() {
6.        Socket sock = new Socket(AddressFamily.InterNetwork,
7.                          SocketType.Dgram, ProtocolType.Udp);
8.        Console.WriteLine("Ready to receive...");
9.        IPEndPoint localEP = new IPEndPoint(IPAddress.Any, 9090);
10.       sock.Bind(localEP);
11.       sock.SetSocketOption(SocketOptionLevel.IP,
12.               SocketOptionName.AddMembership,
13.               new MulticastOption(IPAddress.Parse("224.100.0.1")));
14.       EndPoint remoteEP = new IPEndPoint(IPAddress.Any, 0); //dummy
15.       while (true) {
16.          byte[] data = new byte[1024];
17.          int recv = sock.ReceiveFrom(data, ref remoteEP);
18.          string message = Encoding.ASCII.GetString(data, 0, recv);
19.          Console.WriteLine("received: {0}  from: {1}", message,
20.                                              remoteEP.ToString());
21.       }
22.    }}
```

SetSocketOption Must be done after binding

Example 3: Multicast Receiver …

Notes:

- The SetSocketOption method must be called after the call to Bind method.
  - This enables the multicast group to be set for a specific IPEndPoint.
- Once a socket has been added to a specific multicast group, the ReceiveFrom method will accept packets destined for each of the following:
  - The IPEndPoint specified in the call to the Bind method.
  - The multicast group IP address specified in the MulticastOption constructor
  - Broadcast packets for the specified IPEndPoint.
- Thus, applications are not guaranteed to receive only packets destined for the multicast group and there is no easy way of distinguishing these packets.

Example 3: Local Multicast Sender

```
1.  using System;
2.  using System.Net;
3.  using System.Net.Sockets;
4.  using System.Text;
5.  public class SimpleMulticastSender {
6.     public static void Main()    {
7.       Socket sock = new Socket(AddressFamily.InterNetwork,
8.                           SocketType.Dgram, ProtocolType.Udp);
9.       IPEndPoint endPoint = new IPEndPoint(
10.                         IPAddress.Parse("224.100.0.1"), 9090);
11.      String message;
12.      while (true) {
13.        Console.Write("Enter message to Multicast:
14.        message = Console.ReadLine();
15.        if (message=="")
16.             break;
17.        byte[] data = Encoding.ASCII.GetBytes(message);
18.        sock.SendTo(data, endPoint);
19.      }
20.      sock.Close();
21.    }
22. }
```

Nothing special is required to send local multicast messages. Just indicate group address as target.

Sending Global Multicast Packets

- By default, packets sent by a socket have a TTL value of 1.
  - Thus, they cannot be forwarded by the router to another network.
- To send multicast packets that can traverse multiple routers, socket options must be set to:
  - Join the multicast group
  - Increase the TTL value.
- TTL value is increased by setting the MulticastTimeToLive SocketOption, which is an IP-level option.
  - The value is a positive integer indicating the number of hops.
- As with receiving, MulticastSocketOption must be set after binding the socket to a local end point.

Example 3: Global Multicast Sender

```
1.   public class BetterMulticastSender {
2.      public static void Main()    {
3.        Socket sock = new Socket(AddressFamily.InterNetwork,
4.                                 SocketType.Dgram, ProtocolType.Udp);
5.        IPEndPoint localEP = new IPEndPoint(IPAddress.Any, 0);
6.        sock.Bind(localEP);
7.        sock.SetSocketOption(SocketOptionLevel.IP,
8.                     SocketOptionName.AddMembership,
9.                     new MulticastOption(IPAddress.Parse("224.100.0.1")));
10.       sock.SetSocketOption(SocketOptionLevel.IP,
11.                            SocketOptionName.MulticastTimeToLive, 50);
12.       IPEndPoint multicastEP = new IPEndPoint(
13.                         IPAddress.Parse("224.100.0.1"), 9090);
14.       while (true) {
15.         Console.Write("Enter message to Multicast: ");
16.         string message = Console.ReadLine();
17.         if (message=="")
18.              break;
19.         byte[] data = Encoding.ASCII.GetBytes(message);
20.         sock.SendTo(data, multicastEP);
21.       }
22.       sock.Close(); } }
```

Time To Live (TTL)

Example 4: Using UdpClient class

- The UdpClient class supports multicasting by providing the following methods:

JoinMulticastGroup(IPAddress groupIP)	Join a multicast group identified by groupIP with the TLL value of 1
JoinMulticastGroup(IPAddress groupIP, int ttl)	Same as above but allows TTL value to be specified.
DropMulticastGroup(IPAddress groupIP)	Removes the socket from the multicast group identified by groupIP

Example 4: Using UdpClient class …

```
1.  using System;
2.  using System.Net;
3.  using System.Net.Sockets;
4.  using System.Text;
5.  class UdpClientMulticastReceiver {
6.    public static void Main()      {
7.      UdpClient sock = new UdpClient(9050);
8.      Console.WriteLine("Ready to receive...");
9.      sock.JoinMulticastGroup(IPAddress.Parse("224.100.0.1"), 50);
10.     IPEndPoint remoteEP = new IPEndPoint(IPAddress.Any, 0);
11.     byte[] data;
12.     string message;
13.     while (true) {
14.       data = sock.Receive(ref remoteEP);
15.       message = Encoding.ASCII.GetString(data, 0, data.Length);
16.       Console.WriteLine("received: {0}  from: {1}", message,
17.                         remoteEP.ToString());
18.     }
19.   }
20. }
```


Example 4: Using UdpClient class …

```
1.  using System;
2.  using System.Net;
3.  using System.Net.Sockets;
4.  using System.Text;
5.  class UdpClientMulticastSender {
6.    public static void Main()      {
7.      UdpClient sock = new UdpClient(9090);
8.      IPEndPoint remoteEP = new
9.                IPEndPoint(IPAddress.Parse("224.100.0.1"), 9090);
10.     while (true) {
11.       Console.Write("Enter Message to Multicast: ");
12.        string message = Console.ReadLine();
13.       if (message == "")
14.           break;
15.        byte[] data = Encoding.ASCII.GetBytes(message);
16.       sock.Send(data, data.Length, remoteEP);
17.     }
18.     sock.Close();
19.   }
20. }
```

Note that this sender program can only send to a group within the local subnet. To send to external groups, the socket has to join the group and increase the TTL value.

Resources

- MSDN Library
  - http://msdn.microsoft.com/en-us/default.aspx
- Books
  - Richard Blum, C# Network Programming. Sybex 2002.
- Lecture notes of previous offerings of SWE344 and ICS343
- Some other web sites and books; check the course website at
  - http://faculty.kfupm.edu.sa/ics/alfy/files/teaching/swe344/index.htm