Network Programming
[CACS355]
BCA 6th Sem

Er. Sital Prasad Mandal

(Email : info.sitalmandal@gmail.com)
Bhadrapur, Jhapa, Nepal

https://networkprogam-mmc.blogspot.com/

https://networkprogam-mmc.blogspot.com/

Unit-11
IP Multicast

1	Multicasting
2	Multicast Addresses and Groups
3	Clients and Servers
4	Routers and Routing
5	Working with Multicast Sockets
6	The Constructors
7	Communicating with a Multicast Group

Unit-11

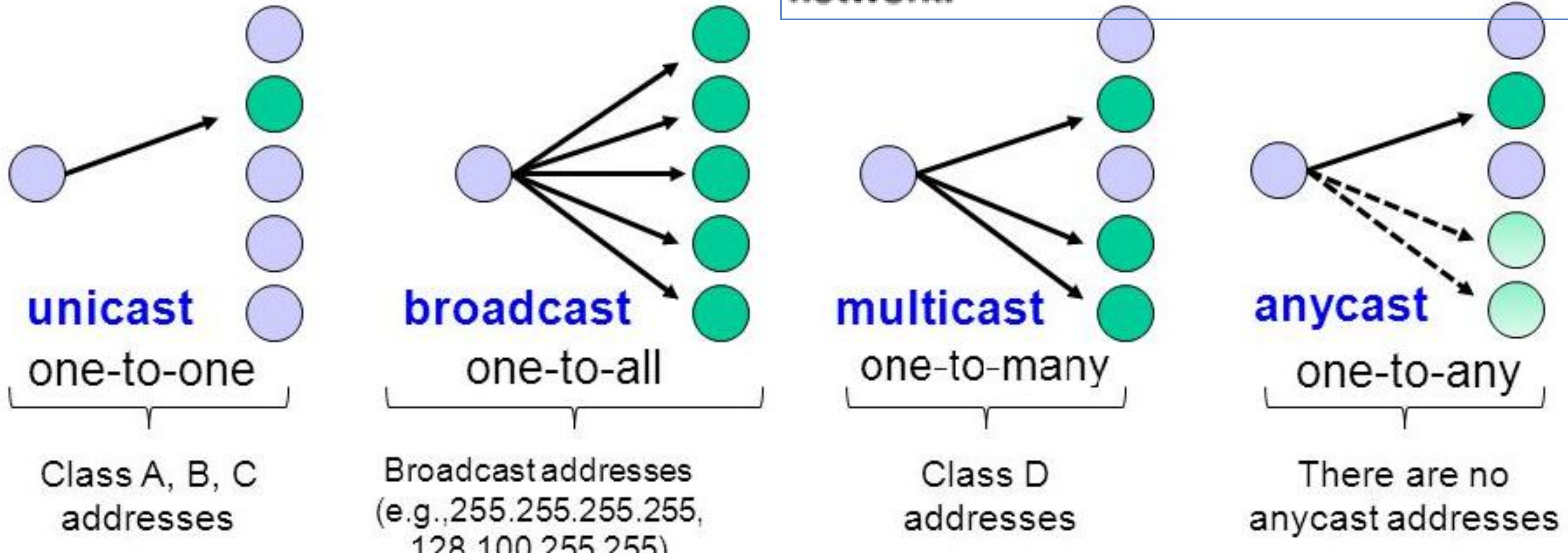
IP Multicast

- Supported by IPv4
  - one-to-one (unicast)
  - one-to-all (broadcast)
  - one-to-many (multicast)
- Not supported by IPv4:
  - one-to-any (anycast)

Multicast: 1 to many

Transfers packets from one source to many recipients. Multicast receivers only receive packets if they showed interest beforehand.

The stream from the camera is unique and sent no more than once on a physical line when on the network.



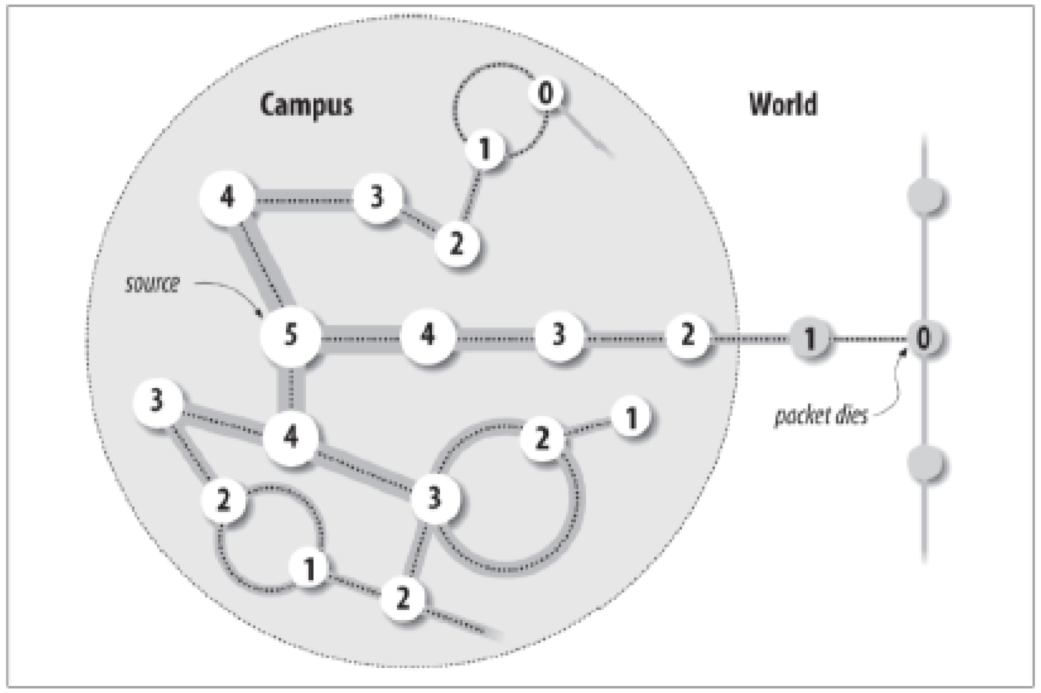
- IP multicast also supports a many-to-many service.
- IP multicast requires support of other protocols (IGMP, multicast routing)

Internet Group Management Protocol(IGMP)

Unit-11

1. Multicasting

- Multicasting: sends data from one host to many different hosts
  - Not to everyone: only clients expressing an interest by joining a multicast group
  - Examples:
    - Apple's Bonjour (a.k.a. Zeroconf) and Apache's River both use IP multicasting to dynamically discover services on the local network
- Most of the work is done by routers and transparent to programmers
  - The routers make sure the packet is delivered to all the hosts in the multicast group
    - The biggest problem is that not all routers support multicasting
  - Time-To-Live (TTL) of IP: maximum number of routers that the datagram is allowed



Destinations	TTL value
The local host	0
The local subnet	1
The local campus—that is, the same side of the nearest Internet router—but on possibly different LANs	16
High-bandwidth sites in the same country, generally those fairly close to the backbone	32
All sites in the same country	48
All sites on the same continent	64
High-bandwidth sites worldwide	128
All sites worldwide	255

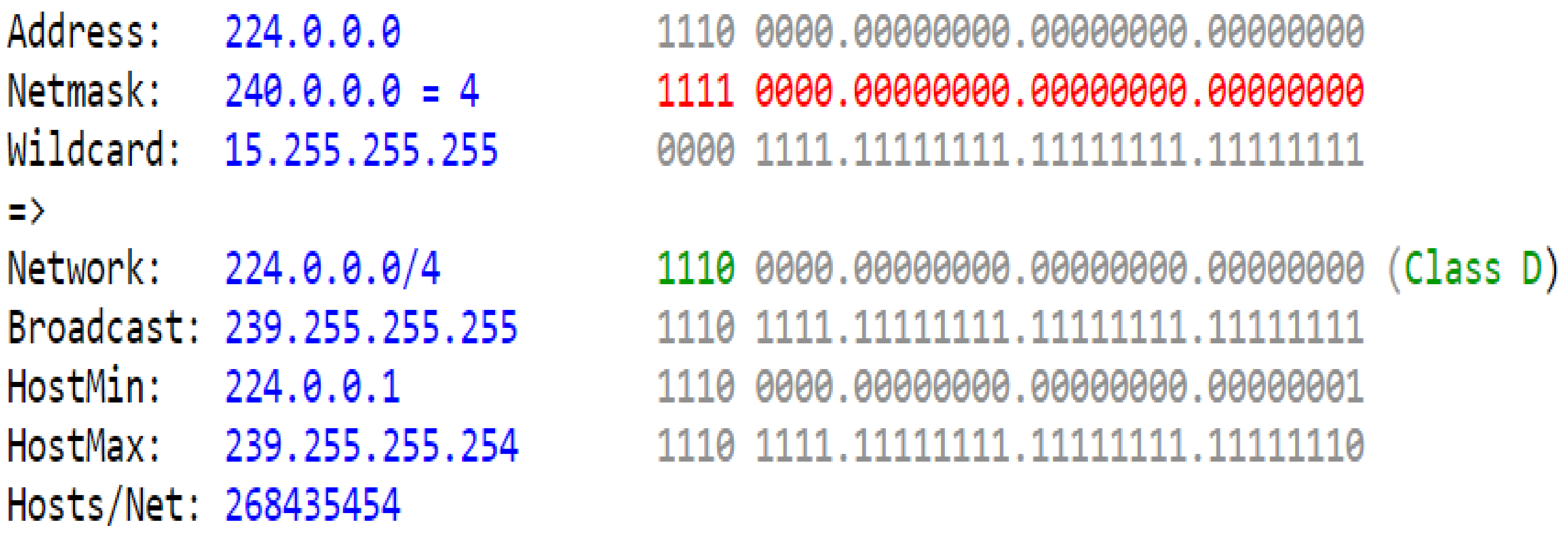
Unit-11

2. Multicast Addresses and Groups

- A multicast address is the shared address of a group of hosts called a multicast group
  - IPv4 CIDR group: 224.0.0.0/4 (224.0.0.0 to 239.255.255.255)
    - All addresses have the leading four binary digits 1110
  - IPv6 CIDR group: ff00::/8
    - All start with the byte 0xFF, or 1111 1111 in binary
- A multicast group
  - Open: enter or leave at any time
  - Either permanent or transient
    - Permanent: the assigned address remains constant
      - Assigned: ***224.1.*** or ***224.2.***
      - The complete list is available from iana.org
    - Transient: exist only as long as they have members
      - Create a new multicast group address from ***225***.0.0.0 to ***238***.255.255.255
- Link-local multicast addresses: from ***224.0.0***.0 to ***224.0.0***.255
  - Packets addressed to a multicast group from 224.0.0.0 to 224.0.0.255 are never forwarded beyond the local subnet

Unit-11

2. Multicast Addresses and Groups



```
Address:   224.0.0.0            1110 0000.00000000.00000000.00000000
Netmask:   240.0.0.0 = 4        1111 0000.00000000.00000000.00000000
Wildcard:  15.255.255.255       0000 1111.11111111.11111111.11111111
=>
Network:   224.0.0.0/4          1110 0000.00000000.00000000.00000000 (Class D)
Broadcast: 239.255.255.255      1110 1111.11111111.11111111.11111111
HostMin:   224.0.0.1            1110 0000.00000000.00000000.00000001
HostMax:   239.255.255.254      1110 1111.11111111.11111111.11111110
Hosts/Net: 268435454
```

http://www.jodies.de/ipcalc?host=224.0.0.0&mask1=4&mask2=

Unit-11

2. Multicast Addresses and Groups

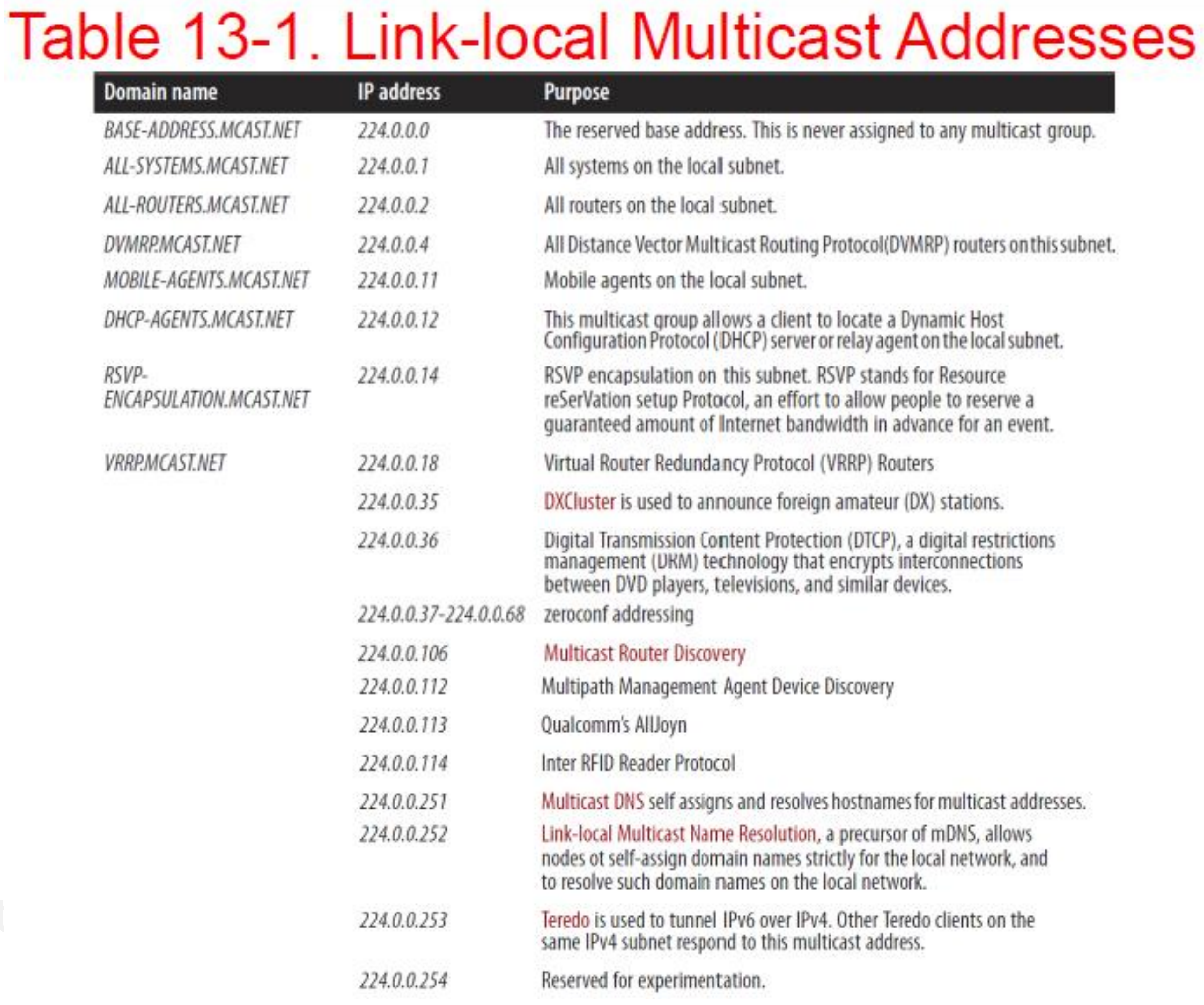
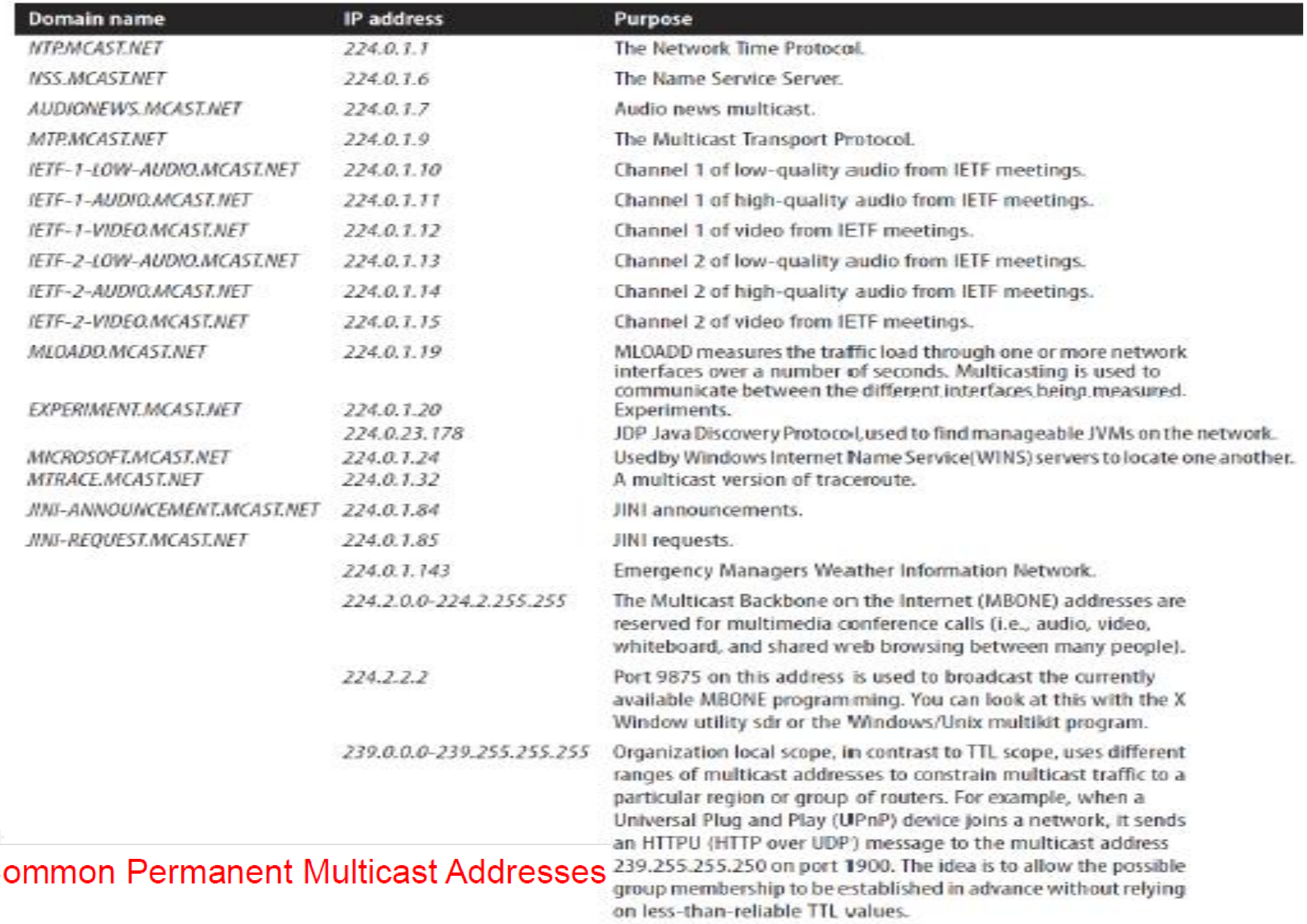


Table 13-1. Link-local Multicast Addresses

Domain name	IP address	Purpose
BASE-ADDRESS.MCAST.NET	*224.0.0.0*	The reserved base address. This is never assigned to any multicast group.
ALL-SYSTEMS.MCAST.NET	*224.0.0.1*	All systems on the local subnet.
ALL-ROUTERS.MCAST.NET	*224.0.0.2*	All routers on the local subnet.
DVMRP.MCAST.NET	*224.0.0.4*	All Distance Vector Multicast Routing Protocol(DVMRP) routers on this subnet.
MOBILE-AGENTS.MCAST.NET	*224.0.0.11*	Mobile agents on the local subnet.
DHCP-AGENTS.MCAST.NET	*224.0.0.12*	This multicast group allows a client to locate a Dynamic Host Configuration Protocol (DHCP) server or relay agent on the local subnet.
RSVP-ENCAPSULATION.MCAST.NET	*224.0.0.14*	RSVP encapsulation on this subnet. RSVP stands for Resource reSerVation setup Protocol, an effort to allow people to reserve a guaranteed amount of Internet bandwidth in advance for an event.
VRRP.MCAST.NET	*224.0.0.18*	Virtual Router Redundancy Protocol (VRRP) Routers
	224.0.0.35	DXCluster is used to announce foreign amateur (DX) stations.
	224.0.0.36	Digital Transmission Content Protection (DTCP), a digital restrictions management (DRM) technology that encrypts interconnections between DVD players, televisions, and similar devices.
	224.0.0.37-224.0.0.68	zeroconf addressing
	224.0.0.106	Multicast Router Discovery
	224.0.0.112	Multipath Management Agent Device Discovery
	224.0.0.113	Qualcomm's AllJoyn
	224.0.0.114	Inter RFID Reader Protocol
	224.0.0.251	Multicast DNS self assigns and resolves hostnames for multicast addresses.
	224.0.0.252	Link-local Multicast Name Resolution, a precursor of mDNS, allows nodes ot self-assign domain names strictly for the local network, and to resolve such domain names on the local network.
	224.0.0.253	Teredo is used to tunnel IPv6 over IPv4. Other Teredo clients on the same IPv4 subnet respond to this multicast address.
	224.0.0.254	Reserved for experimentation.

Unit-11

2. Multicast Addresses and Groups



Domain name	IP address	Purpose
NTP.MCAST.NET	*224.0.1.1*	The Network Time Protocol.
NSS.MCAST.NET	*224.0.1.6*	The Name Service Server.
AUDIONEWS.MCAST.NET	*224.0.1.7*	Audio news multicast.
MTP.MCAST.NET	*224.0.1.9*	The Multicast Transport Protocol.
IETF-1-LOW-AUDIO.MCAST.NET	*224.0.1.10*	Channel 1 of low-quality audio from IETF meetings.
IETF-1-AUDIO.MCAST.NET	*224.0.1.11*	Channel 1 of high-quality audio from IETF meetings.
IETF-1-VIDEO.MCAST.NET	*224.0.1.12*	Channel 1 of video from IETF meetings.
IETF-2-LOW-AUDIO.MCAST.NET	*224.0.1.13*	Channel 2 of low-quality audio from IETF meetings.
IETF-2-AUDIO.MCAST.NET	*224.0.1.14*	Channel 2 of high-quality audio from IETF meetings.
IETF-2-VIDEO.MCAST.NET	*224.0.1.15*	Channel 2 of video from IETF meetings.
MLOADD.MCAST.NET	*224.0.1.19*	MLOADD measures the traffic load through one or more network interfaces over a number of seconds. Multicasting is used to communicate between the different interfaces being measured.
EXPERIMENT.MCAST.NET	*224.0.1.20*	Experiments.
	224.0.23.178	JDP Java Discovery Protocol, used to find manageable JVMs on the network.
MICROSOFT.MCAST.NET	*224.0.1.24*	Usedby Windows Internet Name Service(WINS) servers to locate one another.
MTRACE.MCAST.NET	*224.0.1.32*	A multicast version of traceroute.
JINI-ANNOUNCEMENT.MCAST.NET	*224.0.1.84*	JINI announcements.
JINI-REQUEST.MCAST.NET	*224.0.1.85*	JINI requests.
	224.0.1.143	Emergency Managers Weather Information Network.
	224.2.0.0-224.2.255.255	The Multicast Backbone on the Internet (MBONE) addresses are reserved for multimedia conference calls (i.e., audio, video, whiteboard, and shared web browsing between many people).
	224.2.2.2	Port 9875 on this address is used to broadcast the currently available MBONE programming. You can look at this with the X Window utility sdr or the Windows/Unix multikit program.
	239.0.0.0-239.255.255.255	Organization local scope, in contrast to TTL scope, uses different ranges of multicast addresses to constrain multicast traffic to a particular region or group of routers. For example, when a Universal Plug and Play (UPnP) device joins a network, it sends an HTTPU (HTTP over UDP) message to the multicast address 239.255.255.250 on port 1900. The idea is to allow the possible group membership to be established in advance without relying on less-than-reliable TTL values.

Table 13-2. Common Permanent Multicast Addresses

Unit-11

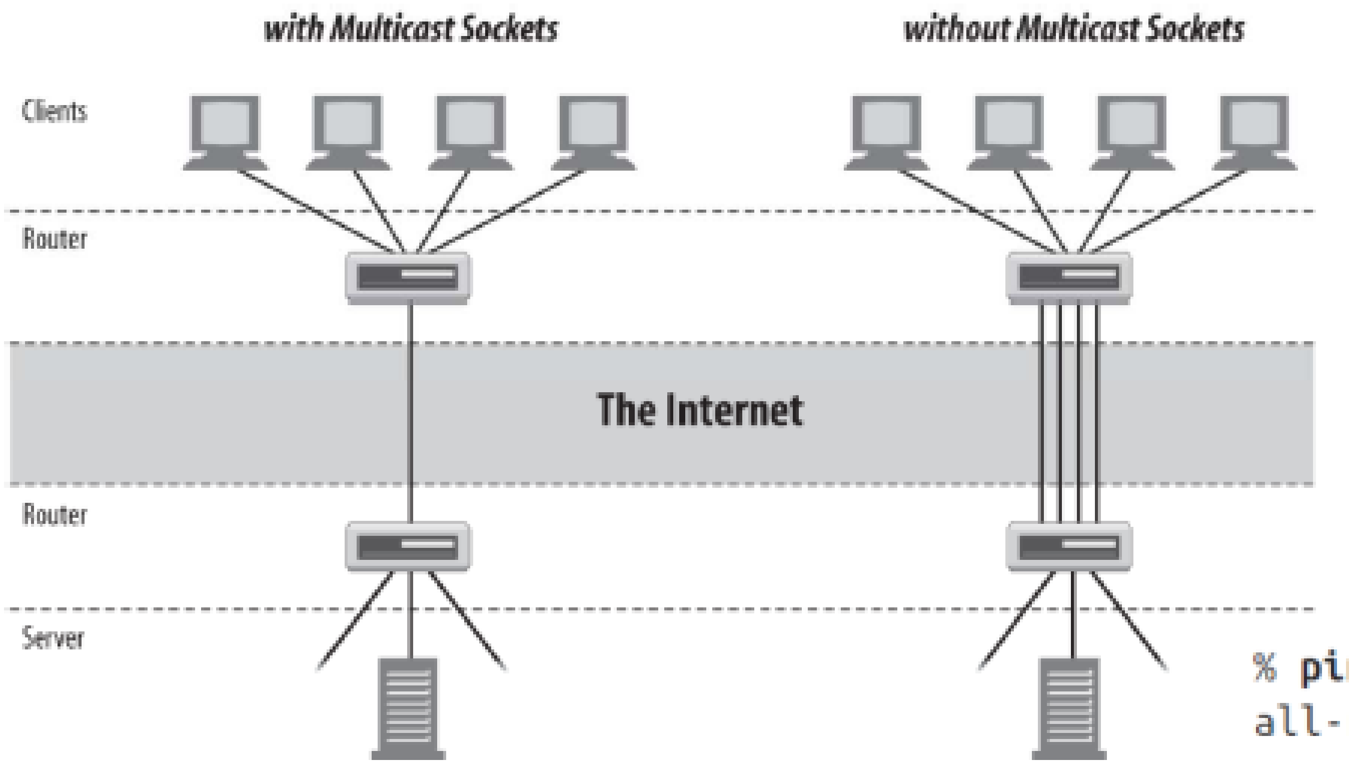
3. Clients and Servers

- Send
  - Once the data has been stuffed into one or more datagrams, the sending host launches the datagrams onto the Internet like sending regular UDP data
- Receive
  - When data arrives at a host in a multicast group, the host receives it as it receives any other UDP datagram

Unit-11

4. Routers and Routing

- Mrouters: the biggest restriction on multicasting is the availability of special multicast routers
  - For packets to reach any given host, there must be a path of multicast capable routers between your host and the remote host



```
% ping all-routers.mcast.net
all-routers.mcast.net is alive
```

http://www.jodies.de/ipcalc?host=224.0.0.0&mask1=4&mask2=

Unit-11

5. Working with Multicast Sockets

Java.net.MulticastSocket

```
public class MulticastSocket extends DatagramSocket
    implements Closeable, AutoCloseable
```

- Steps to receive multicast data
  1. MulticastSocket(): create a MulticastSocket with its constructor
     ```
     MulticastSocket ms = new MulticastSocket(2300);
     ```
  2. joinGroup(): join a multicast group
     ```
     InetAddress group = InetAddress.getByName("224.2.2.2");
     ms.joinGroup(group);
     ```
  3. Receive UDP data just as with a DatagramSocket and DatagramPacket
     ```
     byte[] buffer = new byte[8192];
     DatagramPacket dp = new DatagramPacket(buffer, buffer.length);
     ms.receive(dp);
     ```
  4. leaveGroup(): leave the multicast group
  5. Close socket with close()
     ```
     ms.leaveGroup(group);
     ms.close();
     ```
- Steps to send multicast data: similar to sending UDP data
  - Create a MulticastSocket and directly send data to it
    - Do not need to join a multicast group
      ```
      InetAddress ia = InetAddress.getByName("experiment.mcast.net");
      byte[] data = "Here's some multicast data\r\n".getBytes("UTF-8");
      int port = 4000;
      DatagramPacket dp = new DatagramPacket(data, data.length, ia, port);
      MulticastSocket ms = new MulticastSocket();
      ms.send(dp);
      ```

https://networkprogam-mmc.blogspot.com/

Unit-11

7. Communicating with a Multicast Group

- 3 constructors

```
public MulticastSocket() throws SocketException
public MulticastSocket(int port) throws SocketException
public MulticastSocket(SocketAddress bindAddress) throws IOException
```

  – bind()

```
MulticastSocket ms = new MulticastSocket(null);
ms.setReuseAddress(false);
SocketAddress address = new InetSocketAddress(4000);
ms.bind(address);
```

- joinGroup(): join a group

```
public void joinGroup(InetAddress address) throws IOException
public void joinGroup(SocketAddress address, NetworkInterface interface)
    throws IOException
```

  – A single MulticastSocket can join multiple multicast groups
  – Multiple multicast sockets (on same or different JVMs) can all join the same group

- leaveGroup(): leave a group

```
public void leaveGroup(InetAddress address) throws IOException
public void leaveGroup(SocketAddress multicastAddress,
NetworkInterface interface)
    throws IOException
```

- close(): close the socket

Java 7+

```
try (MulticastSocket socket = new MulticastSocket()) {
  // connect to the server...
} catch (IOException ex) {
  ex.printStackTrace();
}
```

Java 6-

```
MulticastSocket socket = null;
try {
  socket = new MulticastSocket();
  // connect to the server...
} catch (IOException ex) {
  ex.printStackTrace();
} finally {
  if (socket != null) {
    try {
      socket.close();
    } catch(IOException ex){//ignore
} } }
```

Unit-11

7. Communicating with a Multicast Group

- Sending: multicast sockets uses a TTL of 1 by default
  - setTimeToLive(): set the default TTL

```
public void setTimeToLive(int ttl) throws IOException
public int getTimeToLive() throws IOException
```

```
try {
  InetAddress ia = InetAddress.getByName("www.ibiblio.org");
  MulticastSocket ms = new MulticastSocket(2048);
  ms.setInterface(ia);
  // send and receive data...
} catch (UnknownHostException ue) {
  System.err.println(ue);
} catch (SocketException se) {System.err.println(se);}
```

  - setLoopBack(): Set not to receive packets you sends (default is platform dependent)

```
public void setLoopbackMode(boolean disable) throws SocketException
public boolean getLoopbackMode() throws SocketException
```

- setInterface(), setNetworkInterface(): specify a network interface for multicast

```
public void setInterface(InetAddress address) throws SocketException
public InetAddress getInterface() throws SocketException
public void setNetworkInterface(NetworkInterface interface) throws SocketException
public NetworkInterface getNetworkInterface() throws SocketException
```

```
try {
  InetAddress ia = InetAddress.getByName("www.ibiblio.org");
  MulticastSocket ms = new MulticastSocket(2048);
  ms.setInterface(ia);
  // send and receive data...
} catch (UnknownHostException ue) {
  System.err.println(ue);
} catch (SocketException se) {
  System.err.println(se);
}
```

Sending Multicast Data

Unit-11

MulticastClient

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.InetAddress;
import java.net.MulticastSocket;
import java.net.UnknownHostException;
public class MulticastClient {
    final static  String INET_ADDR = "224.0.0.3";
    final static  int PORT = 8888;
    public static void main(String[] args) throws UnknownHostException {
        InetAddress address = InetAddress.getByName(INET_ADDR);
        byte[] buf = new byte[256];
        try(MulticastSocket cs = new MulticastSocket(PORT)){
            cs.joinGroup(address);
            while (true){
                DatagramPacket msgPacket = new DatagramPacket(buf,buf.length);
                cs.receive(msgPacket);

                String msg = new String(buf,0,buf.length);
                System.out.println("Socket 1 recived msg:" + msg);
            }
        }catch (IOException ex){
            ex.printStackTrace();
        }
    }
}
```

https://networkprogam-mmc.blogspot.com/

Unit-11

MulticastServer

```
import java.io.IOException;
import java.net.*;
public class MulticastServer {
    final static String INET_ADDR = "224.0.0.3";
    final static int PORT = 8888;
    public static void main(String[] args) throws UnknownHostException, InterruptedException {
        InetAddress addr = InetAddress.getByName(INET_ADDR);

        try (DatagramSocket ss = new DatagramSocket()) {
          for(int i=0;i<5;i++){
              String msg = "Sent Message server no" + i;
                DatagramPacket msgPacket = new
DatagramPacket(msg.getBytes(),msg.getBytes().length,addr,PORT);
                ss.send(msgPacket);
                System.out.println("Socket 1 recived msg:" + msg);
                Thread.sleep(500);
            }
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}
```

Unit-11

Summary

13.1 Multicasting

- Multicast group and addresses

13.2 Working with Multicast Sockets

- Java.net.MulticastSocket
- joinGroup(), leaveGroup(), bind(), close()

13.3 Two Simple Examples

- MulticastSniffer (Example 13-1), MulticastSender (Example 13-2)