Chapter - 11
IP Multicast

Contents

- **Multicasting**: Multicast Address and Groups, Clients and Servers, Routers and Routing
- **Working with MulticastSocket**: The Constructor, Communicating with a Group

Multicast Addresses and Groups

- A multicast address is the shared address of a group of hosts called a multicast group. We'll talk about the address first.
- IPv4 multicast addresses are IP addresses in the CIDR group 224.0.0.0/4 (i.e., they range from 224.0.0.0 to 239.255.255.255).
- IPv6 multicast addresses are in the CIDR group ff00::/8 (i.e., they all start with the byte 0xFF, or 11111111 in binary).

Common permanent multicast addresses

- **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

Clients and Servers

- When a host wants to send data to a multicast group, it puts that data in multicast datagrams, which are nothing more than UDP datagrams
- addressed to a multicast group.
- Multicast data is sent via UDP, which, though unreliable, can be as much as three times faster than data sent via connection-oriented TCP.
- In IP multicasting, the TTL limits the multicast geographically. For example, a TTL value of 16 limits the packet to the local area, generally one organization or perhaps an organization and its immediate upstream and downstream neighbors

Routers and Routing

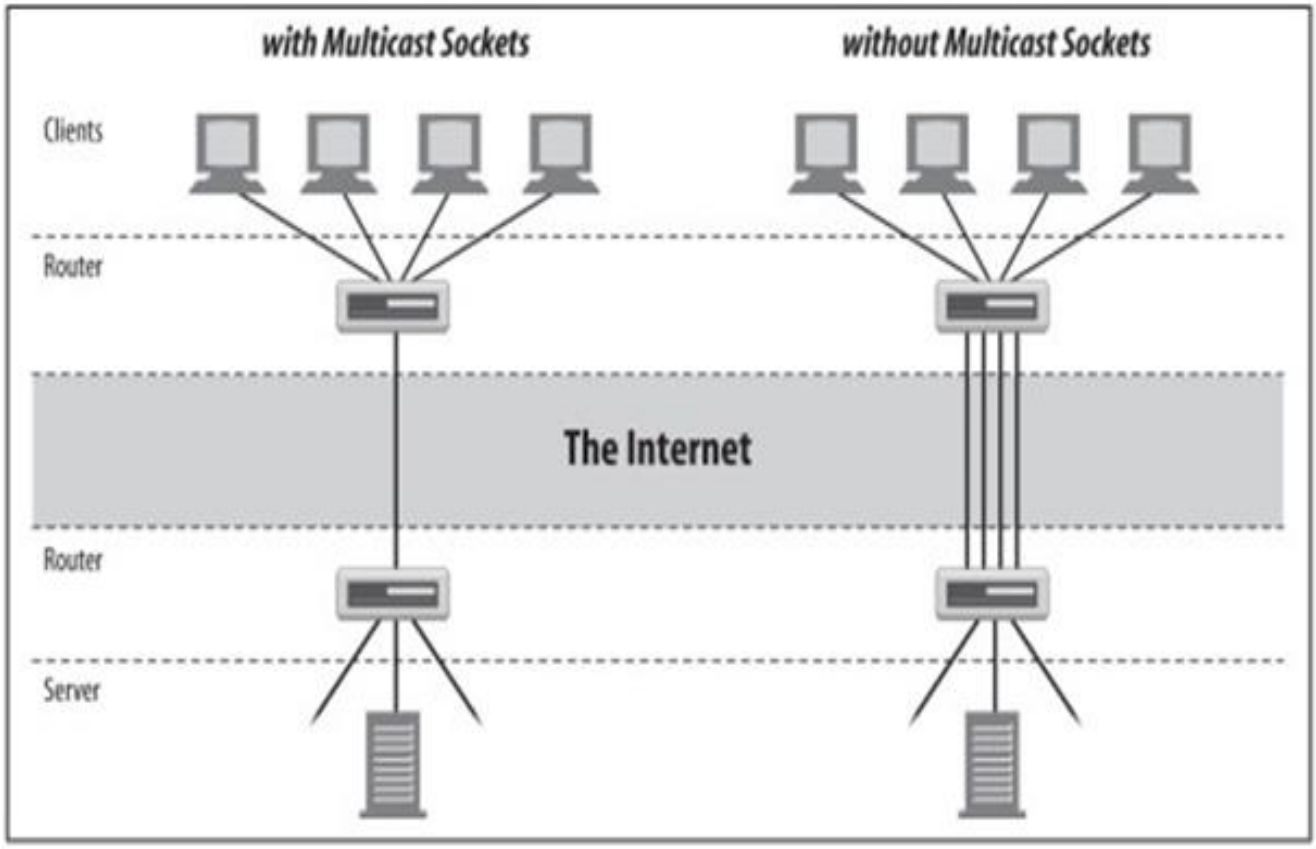

Figure: with and without multicast sockets

Working with Multicast Sockets

- In Java, you multicast data using the **java.net.MulticastSocket** class, a subclass of **java.net.DatagramSocke**t:

Syntax

public class MulticastSocket extends DatagramSocket implements Closeable, AutoCloseable

Working with Multicast Sockets

- To receive data that is being multicast from a remote site, first create a MulticastSocket with the MulticastSocket() constructor. As with other kinds of sockets, you need to know the port to listen on. This code fragment opens a MulticastSocket that listens on port 2300:
- MulticastSocket ms = new MulticastSocket(2300);
- Next, join a multicast group using the MulticastSocket'sjoinGroup() method:

```
InetAddress multicastAddress = InetAddress.getByName("224.0.0.1");
MulticastSocket socket = new MulticastSocket(multicastPort);
socket.joinGroup(multicastAddress);
```

- Once you've joined the multicast group, you receive UDP data just as you would with a DatagramSocket
- Create a DatagramPacket with a byte array that serves as a buffer for data and enter a loop in which you receive the data by calling the receive() method inherited from the DatagramSocket class:

```
byte[] buffer = new byte[1024];
DatagramPacket packet = new DatagramPacket(buffer, buffer.length);
socket.receive(packet);
```

- When you no longer want to receive data, leave the multicast group by invoking the socket's leaveGroup() method. You can then close the socket with the close() method inherited from DatagramSocket:

```
socket.leaveGroup(multicastAddress);
socket.close();
```

The Constructors

- The constructors are simple. You can either pick a port to listen on or let Java assign an anonymous port for you:
  public MulticastSocket()

  public MulticastSocket(int port)

  public MulticastSocket(SocketAddress bindAddress)

- **For example:**

  MulticastSocket ms1 = new MulticastSocket(); MulticastSocket ms2 = new MulticastSocket(4000);

  SocketAddress address = new InetSocketAddress("192.168.254.32", 4000); MulticastSocket ms3 = new MulticastSocket(address);

MulticastSocketClient.java

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.InetAddress;
import java.net.MulticastSocket;
import java.net.UnknownHostException;
public class MulticastSocketClient {
final static String INET_ADDR = "224.0.0.3";
final static int PORT = 8888;

public static void main(String[] args) throws UnknownHostException {
// Get the address that we are going to connect to. InetAddress address = InetAddress.getByName(INET_ADDR);

// Create a buffer of bytes, which will be used to store the incoming bytes containing the information from the server. Since the message is small here, 256 bytes should be enough.
byte[] buf = new byte[256];

// Create a new Multicast socket (that will allow other sockets/programs to join it as well.
```

```
try (MulticastSocket clientSocket = new
MulticastSocket(PORT)){
//Joint the Multicast group. clientSocket.joinGroup(address);

while (true) {
// Receive the information and print it.
DatagramPacket msgPacket = new DatagramPacket(buf,
buf.length);
clientSocket.receive(msgPacket);

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

MulticastSocketServer.java

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.UnknownHostException;

public class MulticastSocketServer {

final static String INET_ADDR = "224.0.0.3";
final static int PORT = 8888;

public static void main(String[] args) throws UnknownHostException, InterruptedException {
// Get the address that we are going to connect to. InetAddress addr = InetAddress.getByName(INET_ADDR);

// Open a new DatagramSocket, which will be used to send the data.
try (DatagramSocket serverSocket = new DatagramSocket())
{
```

```
for (int i = 0; i < 5; i++)
{
String msg = "Sent message no " + i;

// Create a packet that will contain the data (in the form of bytes) and send it.
DatagramPacket msgPacket = new DatagramPacket(msg.getBytes(),
msg.getBytes().length, addr, PORT);
serverSocket.send(msgPacket);

System.out.println("Server sent packet with msg: " + msg);
Thread.sleep(500);
}
} catch (IOException ex) { ex.printStackTrace();
}
}
}
```

Communicating with a Multicast Group

- Once a MulticastSocket has been created, it can perform four key operations:

1) Join a multicast group.
2) Send data to the members of the group.
3) Receive data from the group.
4) Leave the multicast group