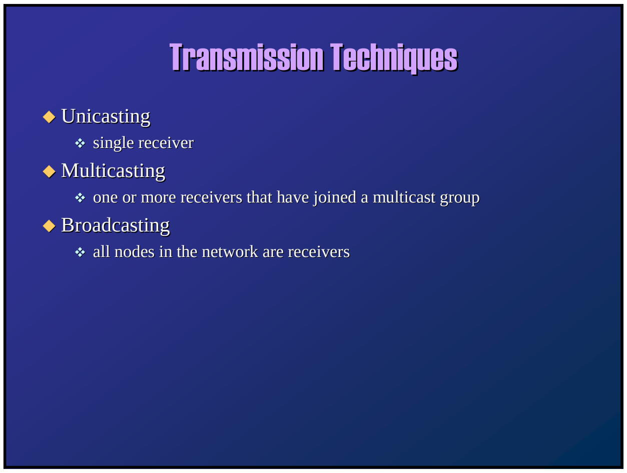
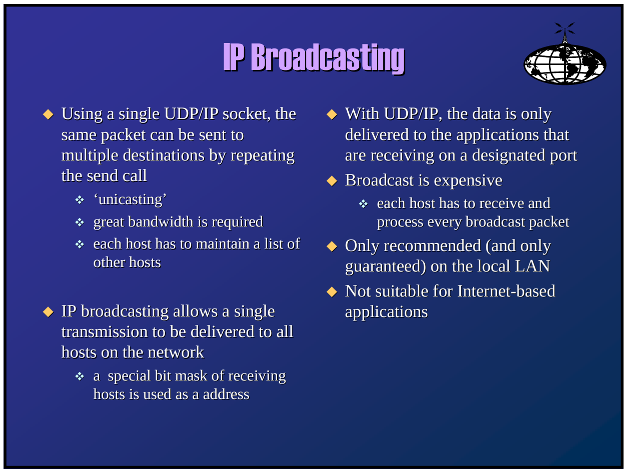


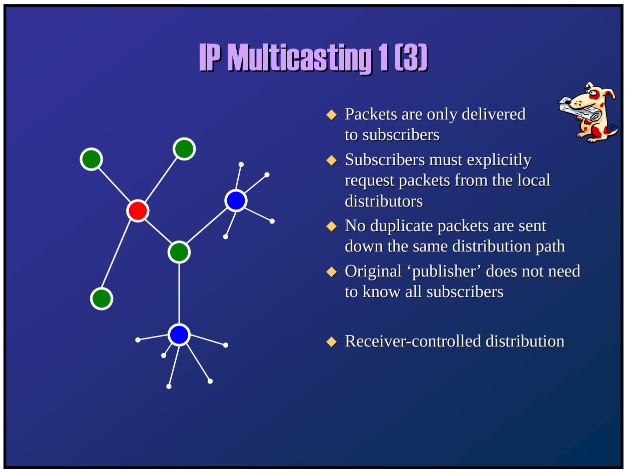
Transmission Techniques

- Unicasting
  - single receiver
- Multicasting
  - one or more receivers that have joined a multicast group
- Broadcasting
  - all nodes in the network are receivers



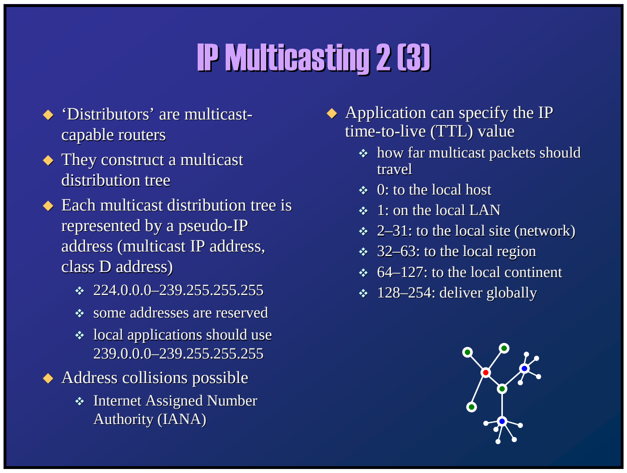
IP Broadcasting

- Using a single UDP/IP socket, the same packet can be sent to multiple destinations by repeating the send call
  - 'unicasting'
  - great bandwidth is required
  - each host has to maintain a list of other hosts
- IP broadcasting allows a single transmission to be delivered to all hosts on the network
  - a special bit mask of receiving hosts is used as a address
- With UDP/IP, the data is only delivered to the applications that are receiving on a designated port
- Broadcast is expensive
  - each host has to receive and process every broadcast packet
- Only recommended (and only guaranteed) on the local LAN
- Not suitable for Internet-based applications



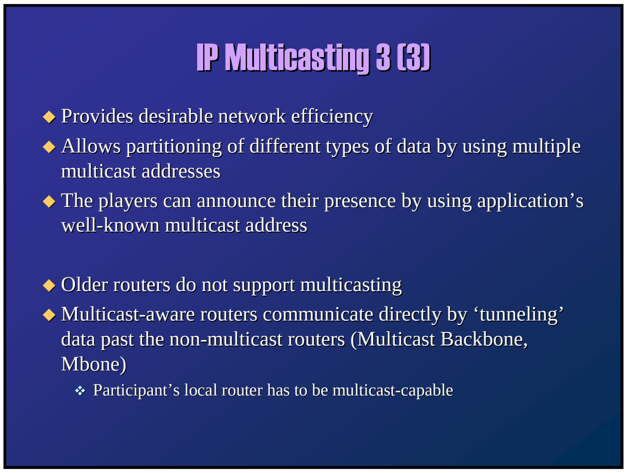
IP Multicasting 1 (3)

- Packets are only delivered to subscribers
- Subscribers must explicitly request packets from the local distributors
- No duplicate packets are sent down the same distribution path
- Original 'publisher' does not need to know all subscribers
- Receiver-controlled distribution



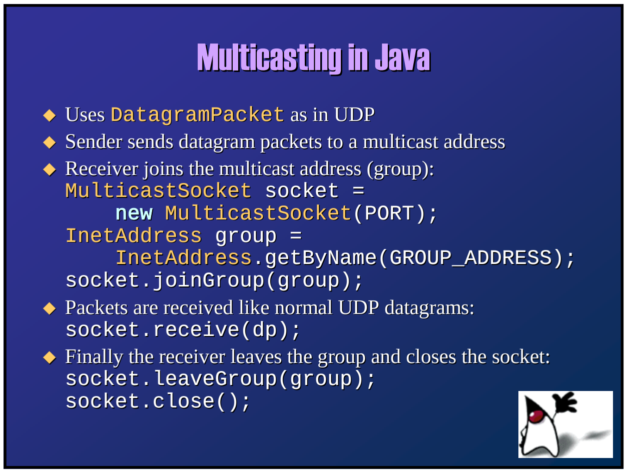
IP Multicasting 2 (3)

- 'Distributors' are multicast-capable routers
- They construct a multicast distribution tree
- Each multicast distribution tree is represented by a pseudo-IP address (multicast IP address, class D address)
  - 224.0.0.0–239.255.255.255
  - some addresses are reserved
  - local applications should use 239.0.0.0–239.255.255.255
- Address collisions possible
  - Internet Assigned Number Authority (IANA)
- Application can specify the IP time-to-live (TTL) value
  - how far multicast packets should travel
  - 0: to the local host
  - 1: on the local LAN
  - 2–31: to the local site (network)
  - 32–63: to the local region
  - 64–127: to the local continent
  - 128–254: deliver globally



IP Multicasting 3 (3)

- Provides desirable network efficiency
- Allows partitioning of different types of data by using multiple multicast addresses
- The players can announce their presence by using application's well-known multicast address
- Older routers do not support multicasting
- Multicast-aware routers communicate directly by 'tunneling' data past the non-multicast routers (Multicast Backbone, Mbone)
  - Participant's local router has to be multicast-capable



Multicasting in Java

- Uses `DatagramPacket` as in UDP
- Sender sends datagram packets to a multicast address
- Receiver joins the multicast address (group):

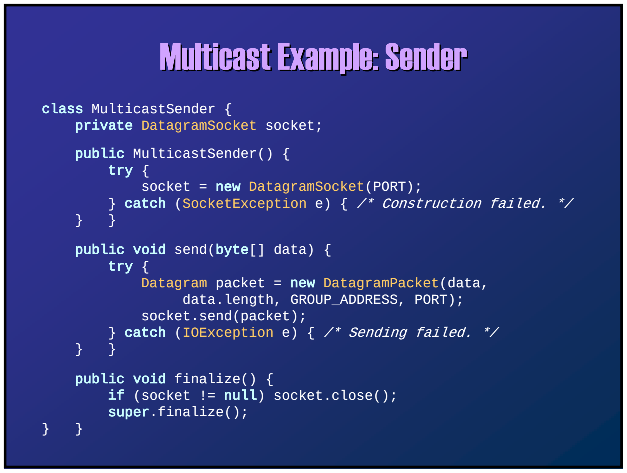
```
MulticastSocket socket =
    new MulticastSocket(PORT);
InetAddress group =
    InetAddress.getByName(GROUP_ADDRESS);
socket.joinGroup(group);
```

- Packets are received like normal UDP datagrams:

```
socket.receive(dp);
```

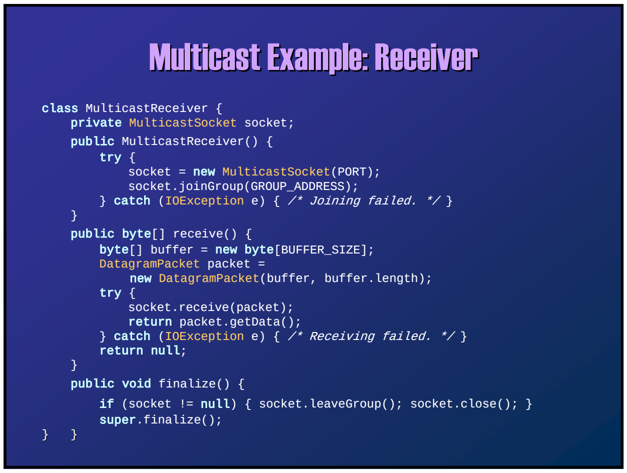
- Finally the receiver leaves the group and closes the socket:

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



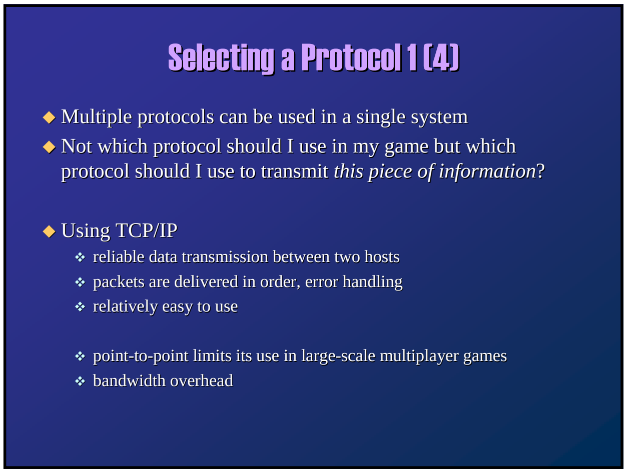
Multicast Example: Sender

```
class MulticastSender {
    private DatagramSocket socket;

    public MulticastSender() {
        try {
            socket = new DatagramSocket(PORT);
        } catch (SocketException e) { /* Construction failed. */
    }   }

    public void send(byte[] data) {
        try {
            Datagram packet = new DatagramPacket(data,
                 data.length, GROUP_ADDRESS, PORT);
            socket.send(packet);
        } catch (IOException e) { /* Sending failed. */
    }   }

    public void finalize() {
        if (socket != null) socket.close();
        super.finalize();
}   }
```



Multicast Example: Receiver

```
class MulticastReceiver {
    private MulticastSocket socket;
    public MulticastReceiver() {
        try {
            socket = new MulticastSocket(PORT);
            socket.joinGroup(GROUP_ADDRESS);
        } catch (IOException e) { /* Joining failed. */ }
    }
    public byte[] receive() {
        byte[] buffer = new byte[BUFFER_SIZE];
        DatagramPacket packet =
            new DatagramPacket(buffer, buffer.length);
        try {
            socket.receive(packet);
            return packet.getData();
        } catch (IOException e) { /* Receiving failed. */ }
        return null;
    }
    public void finalize() {
        if (socket != null) { socket.leaveGroup(); socket.close(); }
        super.finalize();
}   }
```

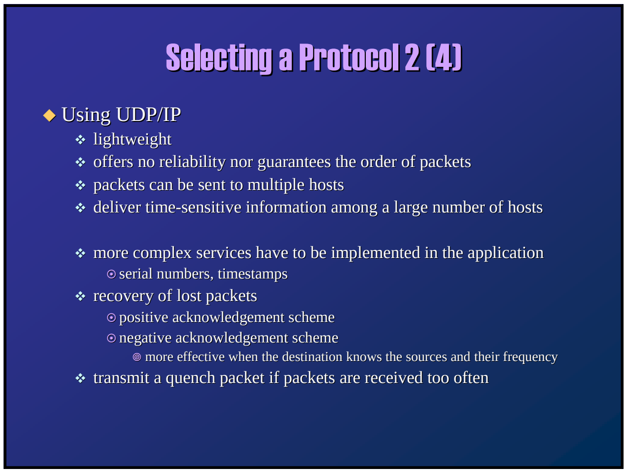


Selecting a Protocol 1 (4)

- Multiple protocols can be used in a single system
- Not which protocol should I use in my game but which protocol should I use to transmit *this piece of information*?

- Using TCP/IP
  - reliable data transmission between two hosts
  - packets are delivered in order, error handling
  - relatively easy to use

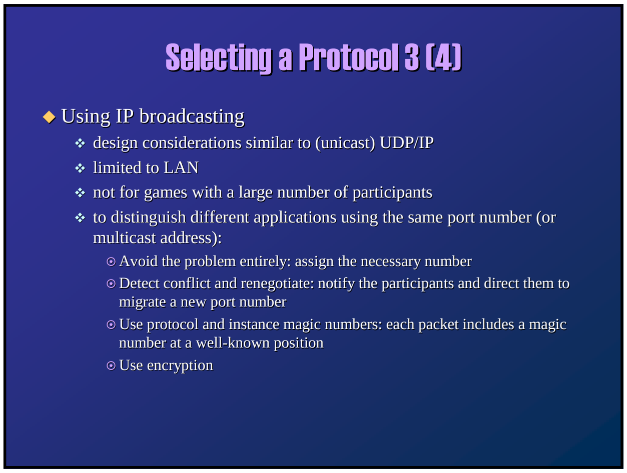
  - point-to-point limits its use in large-scale multiplayer games
  - bandwidth overhead



Selecting a Protocol 2 (4)

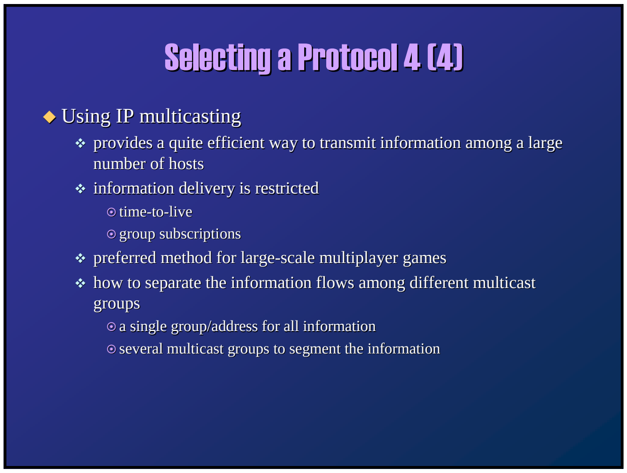
- Using UDP/IP
  - lightweight
  - offers no reliability nor guarantees the order of packets
  - packets can be sent to multiple hosts
  - deliver time-sensitive information among a large number of hosts

  - more complex services have to be implemented in the application
    - serial numbers, timestamps
  - recovery of lost packets
    - positive acknowledgement scheme
    - negative acknowledgement scheme
      - more effective when the destination knows the sources and their frequency
  - transmit a quench packet if packets are received too often



Selecting a Protocol 3 (4)

- Using IP broadcasting
  - design considerations similar to (unicast) UDP/IP
  - limited to LAN
  - not for games with a large number of participants
  - to distinguish different applications using the same port number (or multicast address):
    - Avoid the problem entirely: assign the necessary number
    - Detect conflict and renegotiate: notify the participants and direct them to migrate a new port number
    - Use protocol and instance magic numbers: each packet includes a magic number at a well-known position
    - Use encryption



Selecting a Protocol 4 (4)

- Using IP multicasting
  - provides a quite efficient way to transmit information among a large number of hosts
  - information delivery is restricted
    - time-to-live
    - group subscriptions
  - preferred method for large-scale multiplayer games
  - how to separate the information flows among different multicast groups
    - a single group/address for all information
    - several multicast groups to segment the information