Redes de Comunicação
2024/2025

TP06
IP Multicast addresses

Jorge Granjal
University of Coimbra

UNIVERSIDADE D COIMBRA

TP06: IP Multicast addresses

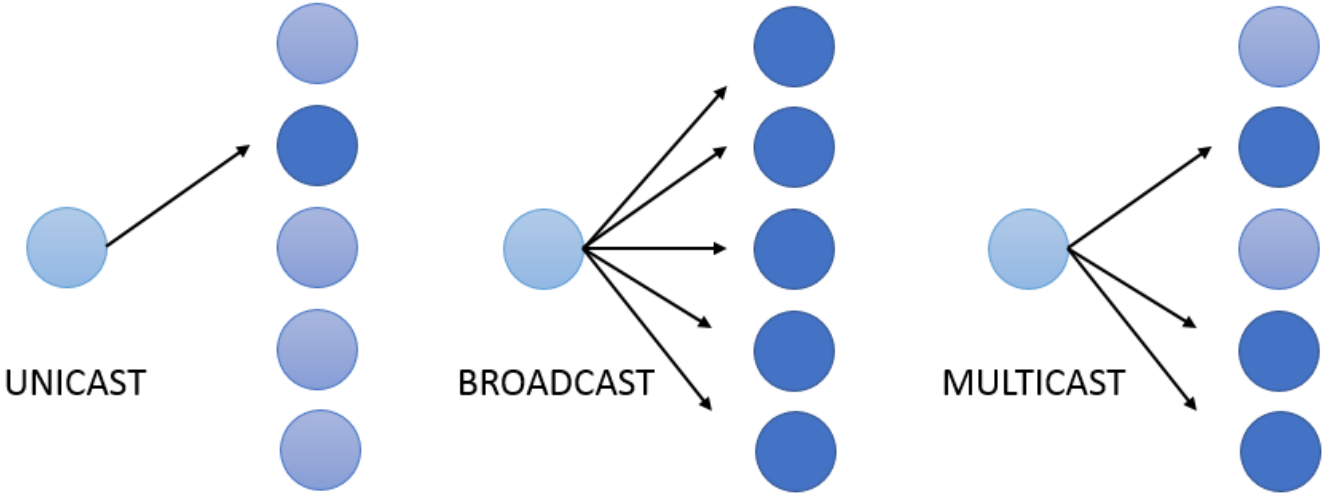
Overview:

- Communications with IP (multicast)
- IPv4 classful addressing
- Network programming (Linux)

Communications with IP (multicast)

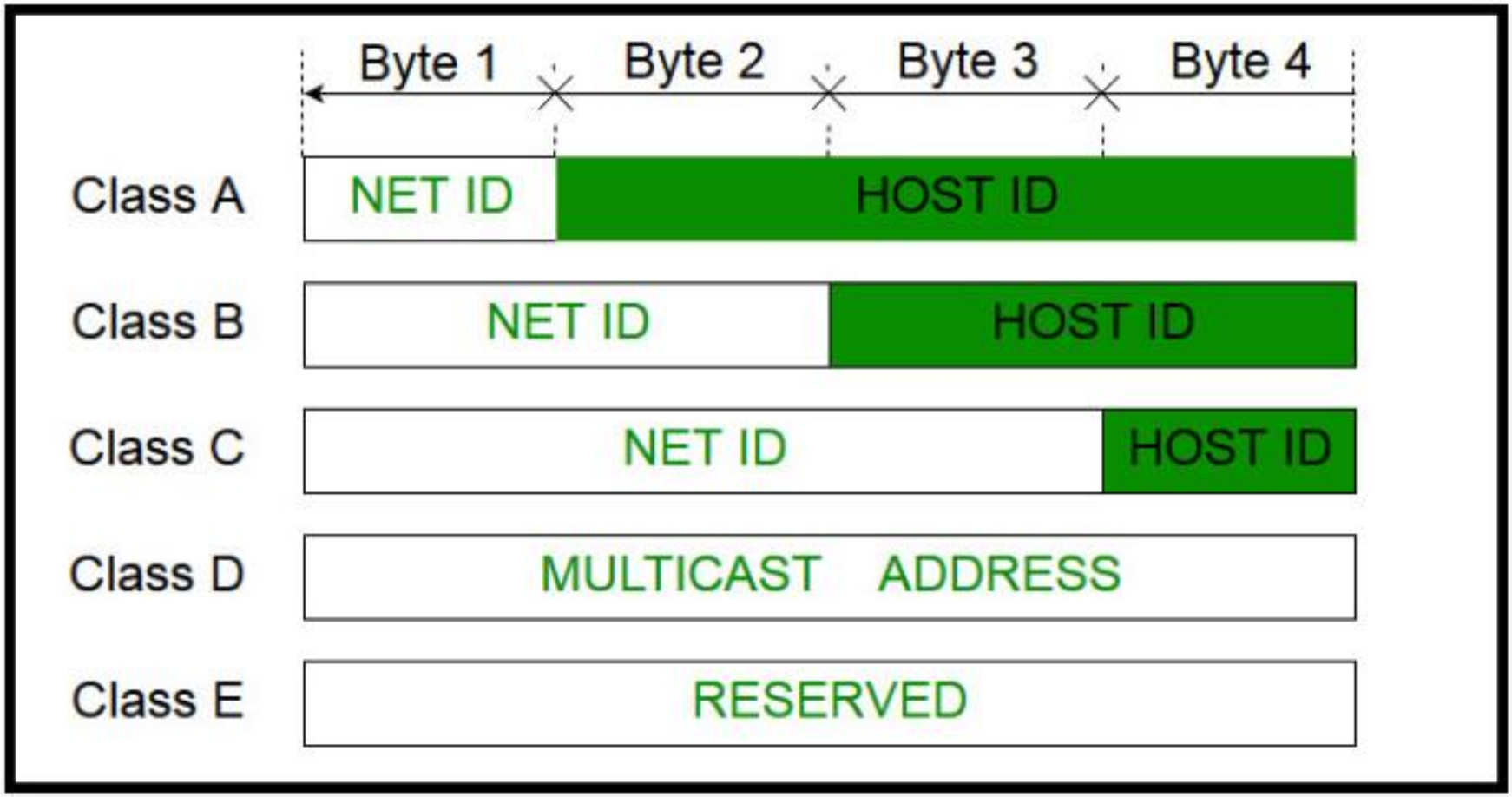
IP SUPPORTS THE FOLLOWING SERVICES:

- ONE TO ONE (UNICAST)
- ONE TO ALL (BROADCAST)
- ONE TO SEVERAL (MULTICAST)



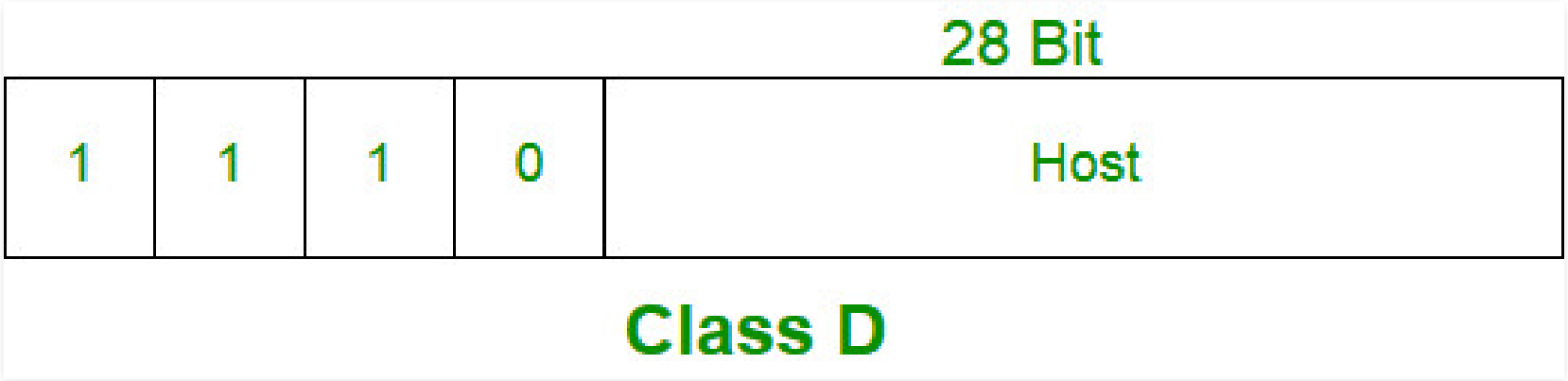
IP MULTICAST ALSO SUPPORTS A MANY TO MANY SERVICE
IP MULTCAST REQUIRES SUPPORT OF OTHER PROTOCOLS (IGMP, MULTICAST ROUTING)

IPv4 classful addressing



IPv4 classful addressing

- **Class D** IPv4 addresses:
  - Reserved for multicast communications
  - Higher order bits of the first byte is always 1110
  - From 224.0.0.0 to 239.255.255.255



Network Programming (Linux)

- **IP multicasting** provides the capability for an application to send a single IP datagram that a group of hosts in a network can receive. The hosts in the group may reside on a single subnet or may be on different subnets (connected by multicast capable routers).

- **Hosts may join and leave groups at any time**. There are no restrictions on the location or number of members in a host group. A class D Internet address in the range 224.0.0.1 to 239.255.255.255 **identifies a host group**.

- An application program can send or receive multicast datagrams by using the socket() API and connectionless SOCK_DGRAM type sockets.

- When a socket of type SOCK_DGRAM is created, an application can use the **setsockopt()** function to control the multicast characteristics associated with that socket.

Network Programming (Linux)

The **setsockopt()** function accepts the following IPPROTO_IP level flags:

- IP_ADD_MEMBERSHIP: Joins the multicast group specified
- IP_DROP_MEMBERSHIP: Leaves the multicast group specified
- IP_MULTICAST_IF: Sets the interface over which outgoing multicast datagrams are sent
- IP_MULTICAST_TTL: Sets the Time To Live (TTL) in the IP header for outgoing multicast datagrams. By default it is set to 1. TTL of 0 are not transmitted on any sub-network. Multicast datagrams with a TTL of greater than 1 may be delivered to more than one sub-network, if there are one or more multicast routers attached to the first sub-network.
- IP_MULTICAST_LOOP: Specifies whether or not a copy of an outgoing multicast datagram is delivered to the sending host as long as it is a member of the multicast group.

Example: sending a multicast datagram

1. Create an AF_INET, SOCK_DGRAM type socket

2. Initialize a *sockaddr_in* structure with the destination group IP address and port number

3. Set the IP_MULTICAST_LOOP socket option according to whether the sending system should receive a copy of the multicast datagrams that are transmitted

4. Set the IP_MULTICAST_IF socket option to define the local interface over which you want to send the multicast datagrams

5. Send the datagram

Example: sending a multicast datagram

```
…
// create a UDP socket
    if ((sock = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
        perror("socket");
        exit(1);
    }

    // set up the multicast address structure
    memset(&addr, 0, sizeof(addr));
    addr.sin_family = AF_INET;
    addr.sin_addr.s_addr = inet_addr(“239.0.0.1”);
    addr.sin_port = htons(5000);

    // enable multicast on the socket
    int enable = 1;
    if (setsockopt(sock, IPPROTO_IP, IP_MULTICAST_TTL, &enable, sizeof(enable)) < 0) {
        perror("setsockopt");
        exit(1);
    }

    // send the multicast message
    if (sendto(sock, msg, msglen, 0, (struct sockaddr *)&addr, sizeof(addr)) < 0) {
        perror("sendto");
        exit(1);
    }
…
```

Example: receiving a multicast datagram

1. Create an AF_INET, SOCK_DGRAM type socket

2. Set the SO_REUSEADDR option to allow multiple applications to receive datagrams that are destined to the same local port number

3. Use bind() to specify the local port number. Specify the IP address as INADDR_ANY in order to receive datagrams that are addressed to a multicast group

4. Use the IP_ADD_MEMBERSHIP socket option to join the multicast group that receives the datagrams. When joining a group, specify the class D group address along with the IP address of a local interface.

5. Receive the datagram

Example: receiving a multicast datagram (1)

```
…
// create a UDP socket
    if ((sock = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
        perror("socket");
        exit(1);
    }

    // set up the multicast address structure
    memset(&addr, 0, sizeof(addr));
    addr.sin_family = AF_INET;
    addr.sin_addr.s_addr = _ANY;
    addr.sin_port = htons(5000);

    // bind the socket to the port
    if (bind(sock, (struct sockaddr *)&addr, sizeof(addr)) < 0) {
        perror("bind");
        exit(1);
    }

    // join the multicast group
    struct ip_mreq mreq;
    mreq.imr_multiaddr.s_addr = inet_addr(“239.0.0.1”);
    mreq.imr_interface.s_addr = INADDR_ANY;
    if (setsockopt(sock, IPPROTO_IP, IP_ADD_MEMBERSHIP, &mreq, sizeof(mreq)) < 0) {
        perror("setsockopt");
        exit(1);
    }
```

Example: receiving a multicast datagram (2)

```
// receive the multicast message
   int nbytes;
   if ((nbytes = recvfrom(sock, msg, sizeof(msg), 0, (struct sockaddr *)&addr, &addrlen)) < 0) {
       perror("recvfrom");
       exit(1);
   }
   printf("Received multicast message: %s\n", msg);

   // leave the multicast group
   if (setsockopt(sock, IPPROTO_IP, IP_DROP_MEMBERSHIP, &mreq, sizeof(mreq)) < 0) {
       perror("setsockopt");
       exit(1);
   }

   // close the socket
   close(sock);

…
```

TP08: Summary

What have we covered here?:

- Communications with IP (multicast)
- IPv4 classful addressing
- Network programming (Linux)