Network Protocols

Dr. Ayman A. Abdel-Hamid

College of Computing and Information Technology

Arab Academy for Science, Technology,

and Maritime Transport

Multicast

Outline

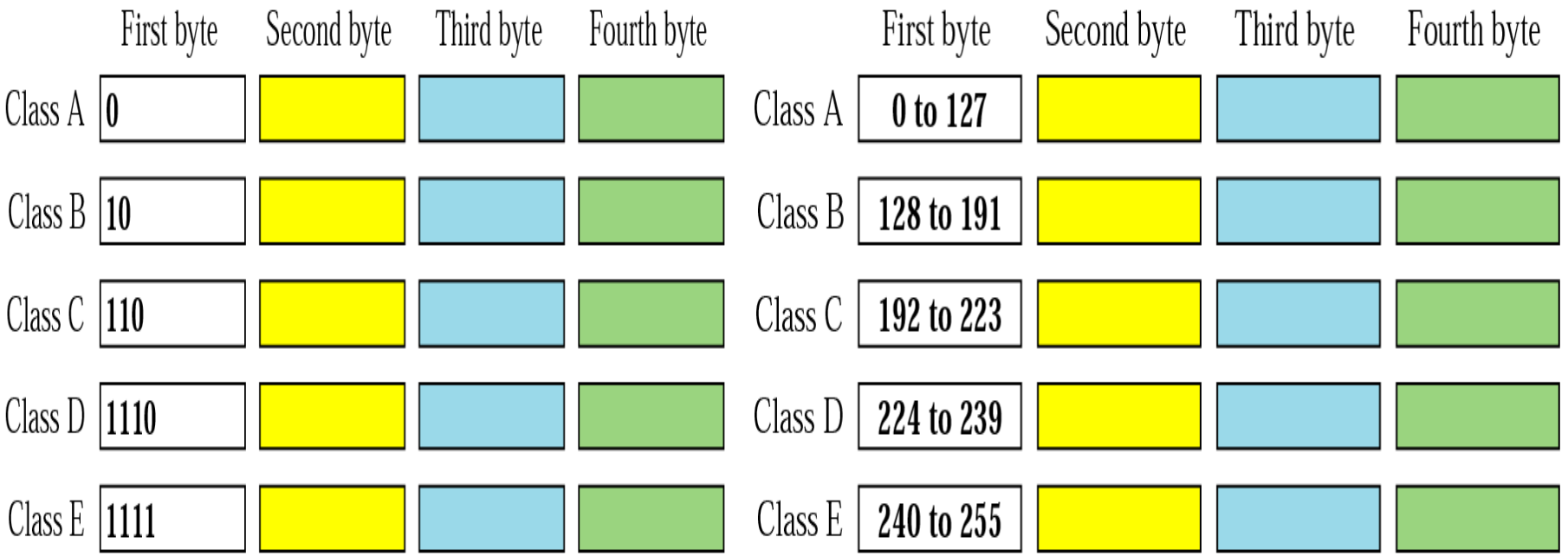
- Multicasting (Chapter 21)
  - Introduction
  - IPv4 Multicast addresses
  - Sending and Receiving Messages
  - Multicasting on a LAN
  - Multicasting on a WAN
  - Multicast Issues

Introduction

- A unicast address identifies a single IP interface
- A broadcast address identifies all IP interfaces on the subnet
- A multicast address identifies a set of IP interfaces
- A multicast datagram is received only by those interfaces interested in the datagram (applications wishing to participate in the multicast group)

IPv4 Multicast Addresses 1/4

- Class D addresses in the range 224.0.0.0 through 239.255.255.255
- Low order 28 bits of class D address (see appendix A) form the multicast group ID (32-bit address is the group address)



IPv4 Multicast Addresses 2/4

- Mapping of IPv4 multicast address to Ethernet address
  - High-order 24 bits of Ethernet address are always 01:00:5E
  - Next bit always 0
  - Low-order 23 bits are copied from low-order 23 bits of multicast group address
  - High-order 5 bits of group address are ignored in the mapping
  - Mapping not one-to-one

IPv4 Multicast Addresses 3/4

224.0.1.88 mapped into an Ethernet address?

- Remember an Ethernet address is 48 bits
- The address 224 is E0 in hex, 0 is 00 in hex, 1 is 01 in hex, and 88 is 58 in hex. However, only the low-order 23 bits are used
- Therefore, the IP address of 224.0.1.88 converted to a MAC address is 01-00-5E-00-01-58.

IPv4 Multicast Addresses 4/4

Some special IPv4 multicast addresses

- 224.0.0.0 reserved
- 224.0.0.1 all-host group
- 224.0.0.2 all-routers group
- 224.0.0.1 through 224.0.0.255 reserved for routing-protocols
- Datagrams destined to any of theses addresses are never forwarded by a multicast router

Sending & Receiving Multicast Messages

Receiving Multicast Messages

- Create a UDP socket
- Bind it to a UDPport, e.g., 1234
  - All processes must bind to the same port in order to receive the multicast messages
- Join a multicast group address
- Use *recv* or *recvfrom* to read the messages

Sending Multicast Messages

- You may use the same socket (you used for receiving) for sending multicast messages or you can use any other UDP socket (it does not have to join any multicast group)

Multicast on a LAN 1/4

- Receiving application creates a UDP socket, binds to port 123 and joins multicast group 224.0.1.1
  - IPv4 layers saves the information internally and tells appropriate datalink to receive Ethernet frames destined to 01:00:5E:00:01:01
- Sending application creates a UDP socket and sends a datagram to 224.0.1.1, port 123
- Ethernet frame contains destination Ethernet address, destination IP address, and destination port
- *A host on the LAN that did not express interest in receiving multicast from that group* will ignore such datagram
  - Destination Ethernet address does not match the interface address
  - Destination Ethernet address is not the ethernet broadcast address
  - The interface has not been told to receive any group addresses

Multicast on a LAN 2/4

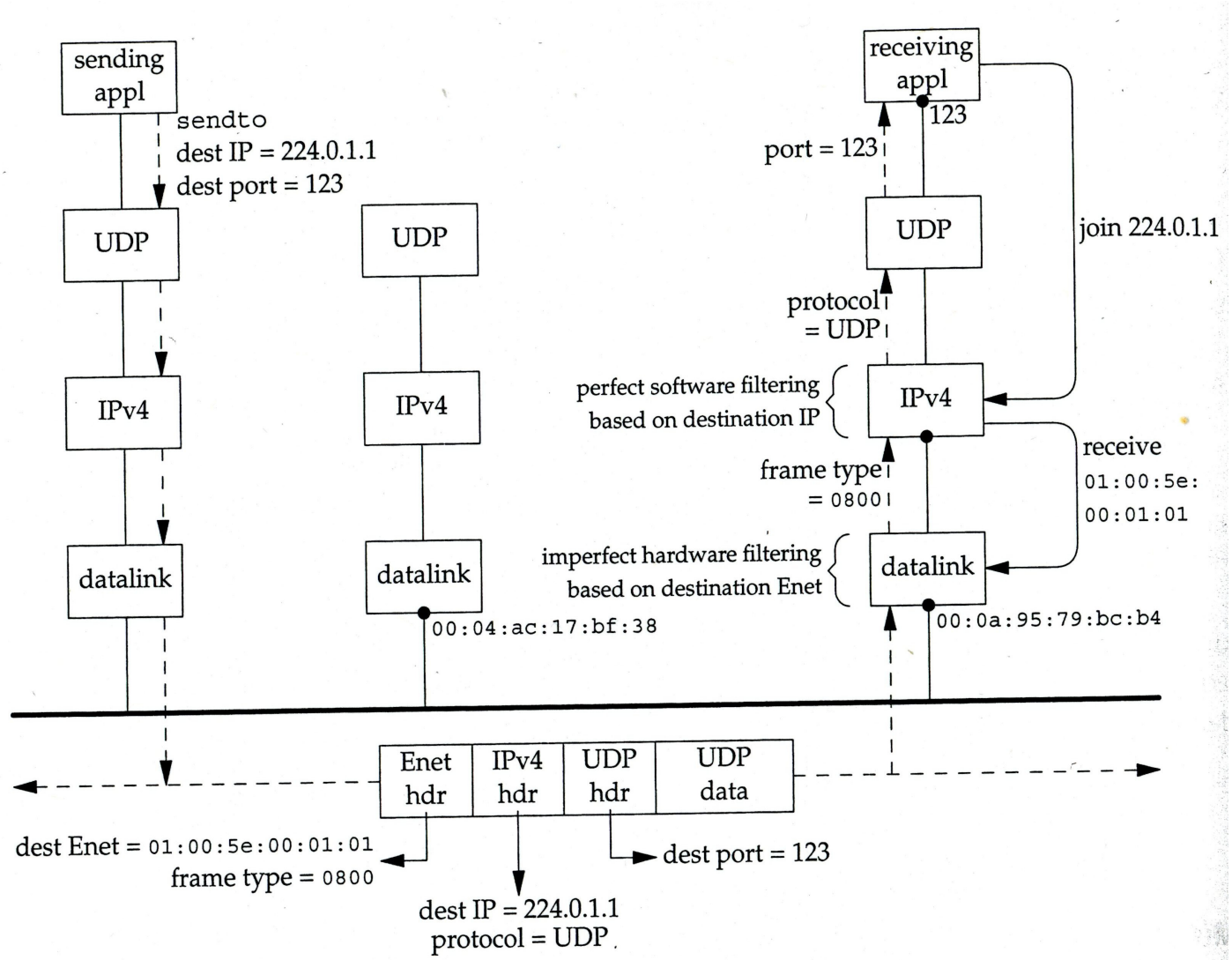
- Ethernet frame received by datalink of receiver based on *imperfect filtering* (When interface told to receive frames destined to one specific Ethernet multicast address, it can receive frames destined to other Ethernet multicast addresses)
  - Ethernet interface cards apply a hash function to group address, calculating a value between 0 and 511. This information turns on a bit in a 512-bit array
  - Small size bit-array implies receiving unwanted frames
  - Some network cards provide perfect filtering
  - Some network cards have no multicast filtering at all (multicast promiscuous mode)
- Packet passed to IP layer (IP layer compares group address against all multicast addresses that applications on this host have joined → *perfect filtering*)
- Packet passed to UDP layer, which passes it to socket bound to port 123

Multicast on a LAN 3/4

Some Other scenarios

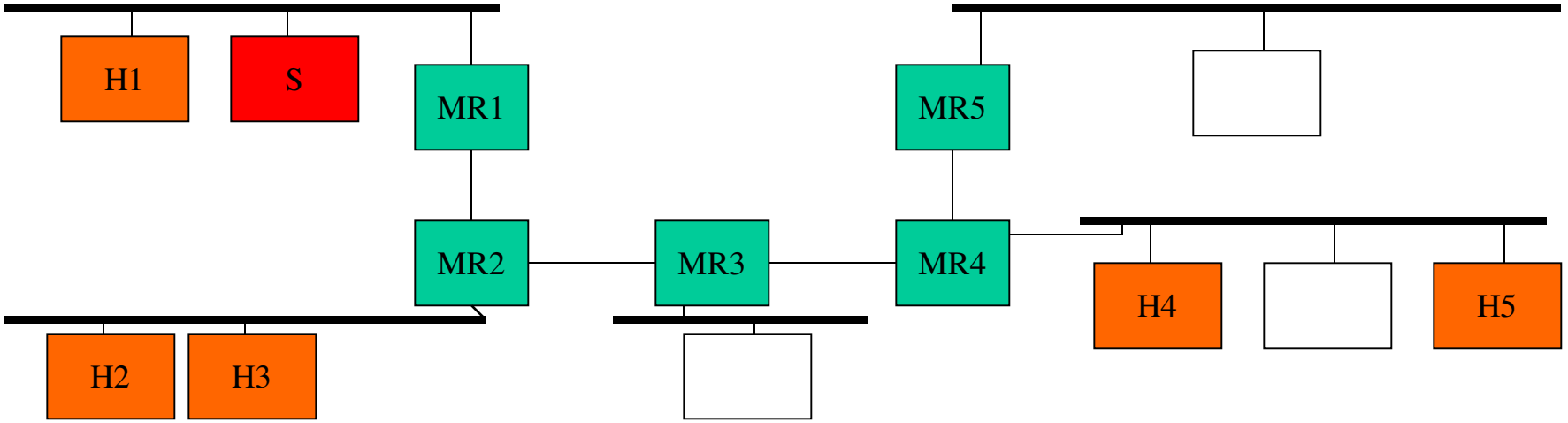
- A host running an application that has joined 225.0.1.1 → Ethernet address 01:00:5E:00:01:01. Packet will be discarded by perfect filtering in IP layer
- A host running an application that has joined some multicast group which the Ethernet address produces the same hash value as 01:00:5E:00:01:01. Packet will be discarded by datalink layer or by IP layer
- A packet destined to the same group, but a different port. Accepted by IP layer, but discarded by UDP layer (no socket has bound the different port)

Multicast on a LAN 4/4



Multicast on a WAN

- A program started on five hosts belonging to different LANs
- Multicast routers communicate with neighbor routers using a *multicast routing protocol* (MRP)
- When a process on a host joins a multicast group, that host sends an IGMP message to any attached multicast routers, which in turn exchange this information using MRP with neighbor routers
- When a sender sends a multicast message, multicast routing information is used to direct the message



Some Multicast Issues

Time To Live

Set TTL for outgoing multicast datagrams (default is 1 → local subnet)

Loopback mode

- Enable or disable local loopback of multicast datagrams
- By default loopback is enabled
- A copy of each multicast datagram sent by a process on the host will also be looped back and processed as a received datagram by that host

Port Reuse

- Allow the same multicast application to have several instances running on the same host
- In Java, Port reuse is enabled by default

Socket Options

•Various attributes that are used to determine the behavior of sockets (see chapter 7)

#include <sys/socket.h>

int getsockopt (int sockfd, int level, int optname, void * optval, socklen_t *optlen);

int setsockopt (int sockfd, int level, int optname, const void * optval, socklen_t optlen);

Both return 0 if OK, -1 on error

•*sockfd*: an open socket descriptor

•*level*: code in the system that interprets the option (general socket code, or protocol-specific code) (SOL_SOCKET, IPPROTO_IP, IPPROTO_IPv6, IPPROTO_TCP are examples)

•*optname*: see page 193

Socket Options

Some socket options examples (see table on page 193 and 194)

For multicast socket options see section 21.6 on page 559

For multicast group membership socket options, see page 560

- Socket Level
  - SO_SNDBUF, SO_RCVBUF, SO_KEEPALIVE, SO_BROADCAST, SO_REUSEADDR, SO_RESUEPORT
- IP Level
  - IP_TTL, IPMULTICAST_IF, IPMUTLICAST_TTL, IP_MULTICAST_LOOP, IP_ADD_MEMBERSHIP, IP_DROP_MEMBERSHIP
- TCP Level
  - TCP_KEEPALIVE, TCP_MAXSEG, TCP_NODELAY

Reusing Port Numbers 1/2

- What if you want to have multiple sockets on the same host listen to the same multicast group?
  - Need to bind the same port number to all sockets
  - This will cause an error when bind is called for the second and later sockets … unless socket has been set to reuse address
- Set SO_REUSEADDR socket option → allows completely duplicate bindings
  - A bind on an IP address and a port, when that same IP address and port are already bound to another socket (only for UDP sockets for multicast)

```
OptValue = 1;
setsockopt(sock, SOL_SOCKET, SO_REUSEADDR, (char *)
&OptValue, sizeof(OptValue));
```

Reusing Port Numbers 2/2

- SO_REUSEPORT
  - Can use SO_REUSEPORT socket option which was introduced in 4.4 BSD
  - Allows completely duplicate bindings, only if each socket that wants to bind the same IP address and port specify this socket option
  - Not supported by all systems
  - SO_REUSEADDR considered equivalent to SO_REUSEPORT if the IP address being bound is a multicast address
  - *Conclusion → When writing a multicast application that can run multiple times on the same host at the same time, set SO_REUSEADDR option and bind group's multicast address as the local IP address*

Receiving Multicast Data

- Create a standard SOCK_DGRAM socket
- Set SOL_REUSEADDR option for socket
- Bind address to socket
  - Specify IP address as multicast address
  - Specify port
- Set IP_ADD_MEMBERSHIP option for socket
  - Specify host group address
- After these steps complete successfully, receive multicast data for specified group address and port using **recvfrom()**
- Drop group membership when finished using IP_DROP_MEMBERSHIP option

Sending Multicast Data

- Use standard SOCK_DGRAM socket
- Sending alone does not require group membership
- To send multicast datagrams:
  - Use **sendto()** to send to appropriate group address and port number, or
  - Use **connect()** to set group address and port and then use **send()**
- Concerns (controlled with socket options)
  - Interface used to send: IP_MULTICAST_IF (relevant to hosts with multiple interfaces)
  - Extent of multicast: IP_MULTICAST_TTL
  - Receiving own data: IP_MULTICAST_LOOP

Textbook Multicast Helper Functions

- See section 21.7 on page 565

int mcast_join (int *sockfd*, const struct sockaddrgrp*, socklen_t *grplen*, const char* *ifname*, u_int *ifindex*);**

int mcast_leave(int *sockfd*, const struct sockaddr **grp*, socklen_t *grplen*);

int mcast_set_loop (int *sockfd*, int *flag*)

int mcast_set_ttl (int *sockfd*, int *ttl*);

//All Above return 0 if OK, -1 on error

int mcast get_loop (int *sockfd*);

int mcast_get_ttl (int *sockfd*);

//return value is OK, -1 on error

Example for Sending and Receiving

- Section 21.10 page 575
- Function ***udp_client*** introduced in 11.14 on page 334
  - Creates an unconnected UDP socket
  - Return value is the socket descriptor
- A program to send and receive multicast datagrams
  - Send datagram to a specific group every five seconds (datagram contains sender's hostname and process ID)
  - An infinite loop that joins the multicast group to which the sending part is sending and prints every received datagram
- Create a UDP socket then set multicast socket options for address reuse, joining the group, and setting loopback
- See **mcast/main.c**, **mcast/send.c**, and **mcast/recv.c**