456: Network-Centric Programming

Yingying (Jennifer) Chen

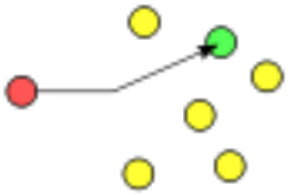
Web: https://www.winlab.rutgers.edu/~yychen/
Email: yingche@scarletmail.rutgers.edu

Department of Electrical and Computer Engineering
Rutgers University

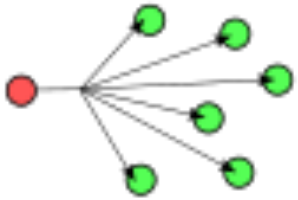
Broadcast and Multicast

Different forms of addressing

- Unicast: one source to one destination
  - Web, telnet, FTP, ssh



- Broadcast: one source to all destinations
  - LAN applications
  - Never used over the Internet



- Multicast: one source to many destinations
  - Several important applications

Different forms of addressing

Type	IPv4	IPv6	TCP	UDP	# IP interfaces identified	# IP interfaces delivered to
Unicast	•	•	•	•	One	One
Multicast	opt.	•		•	A set	All in set
Broadcast	•			•	All	All

• Provide * Not deployed

- Multicasting support is optional in IPv4, but mandatory in IPv6
- Broadcasting support is not provided in IPv6. Any IPv4 application that uses broadcasting must be recorded for IPv6 to use multicasting instead

Different forms of addressing

Type	IPv4	IPv6	TCP	UDP	# IP interfaces identified	# IP interfaces delivered to
Unicast	•	•	•	•	One	One
Multicast	opt.	•		•	A set	All in set
Broadcast	•			•	All	All

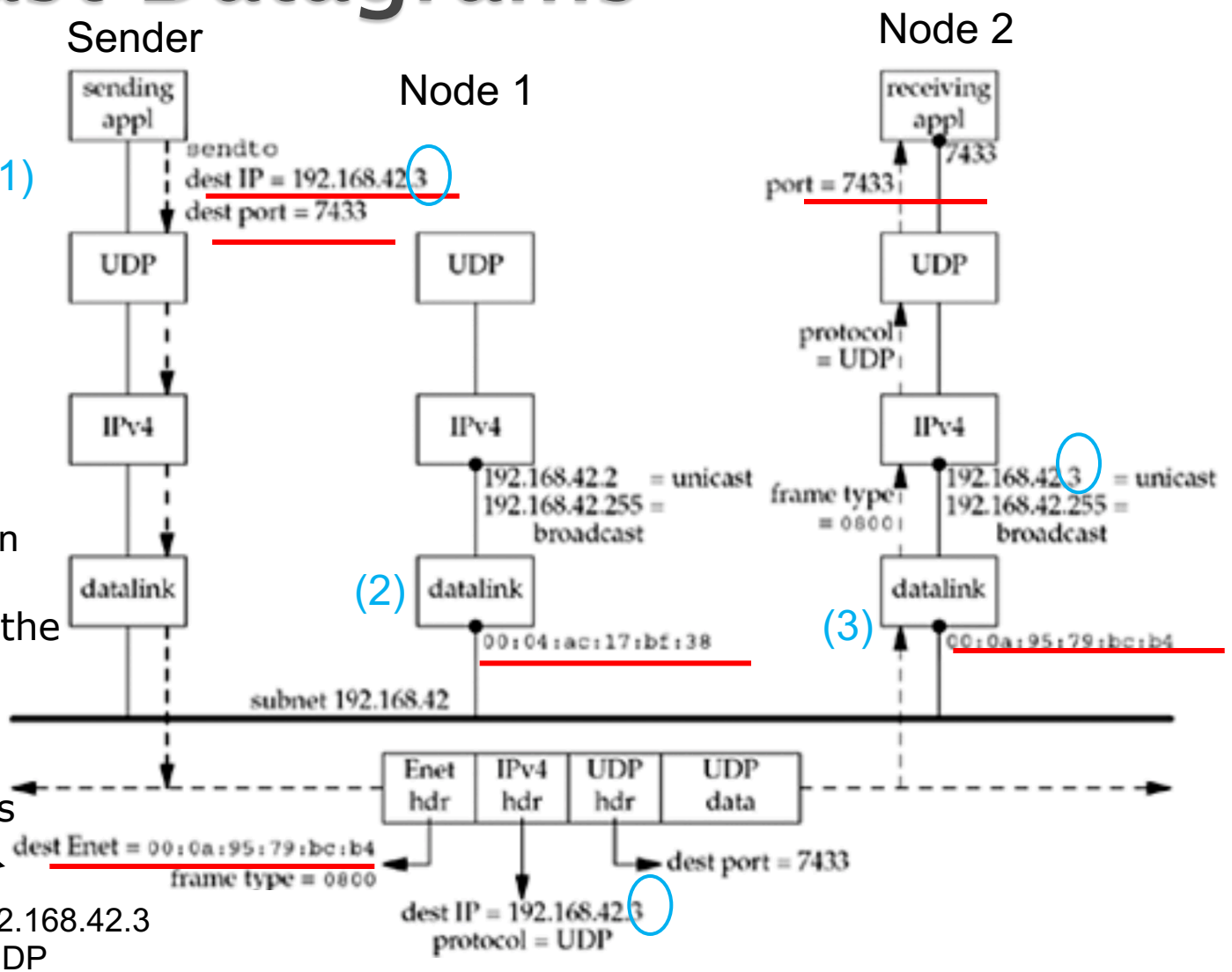
• Provide * Not deployed

- Broadcasting and multicasting only work across UDP
- TCP only works with unicast addresses and can not work with multicast and broadcast

Unicast

- A type of communication where data is sent from one computer to another computer
- There is only one sender and one receiver
- Example:
  - Browsing a website
    - (Webserver is the sender and your computer is the receiver)
  - Downloading a file from a FTP server
    - (FTP server is the sender and your computer is the receiver
- Unicast works through both TCP and UDP connections
- TCP can only talk to unicast address

Unicast Datagrams



Address Resolution Protocol (ARP) is invoked to map the destination IP address to the corresponding Ethernet address

dest IP = 192.168.42.3
Protocol = UDP
dest Enet = 00:0a:95:79:bc:b4

Node 1: 00:04:ac:17:bf:38

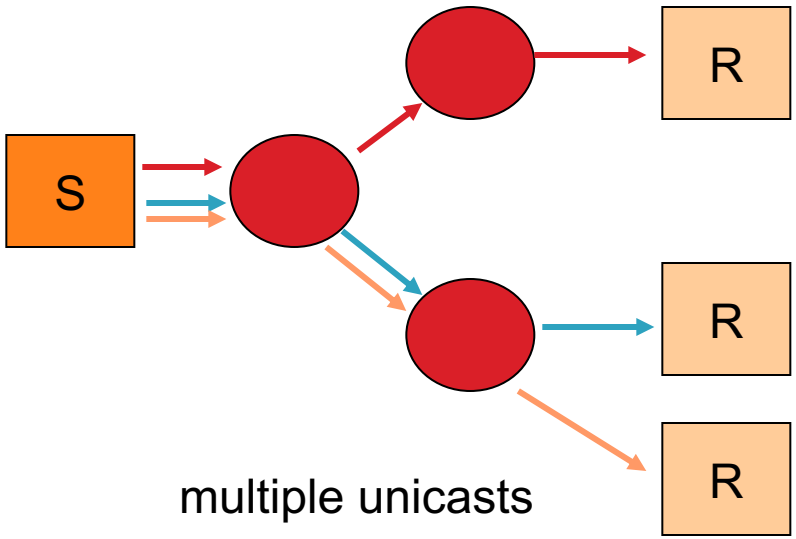
Not equal, ignore the frame

Node 2: 00:0a:95:79:bc:b4

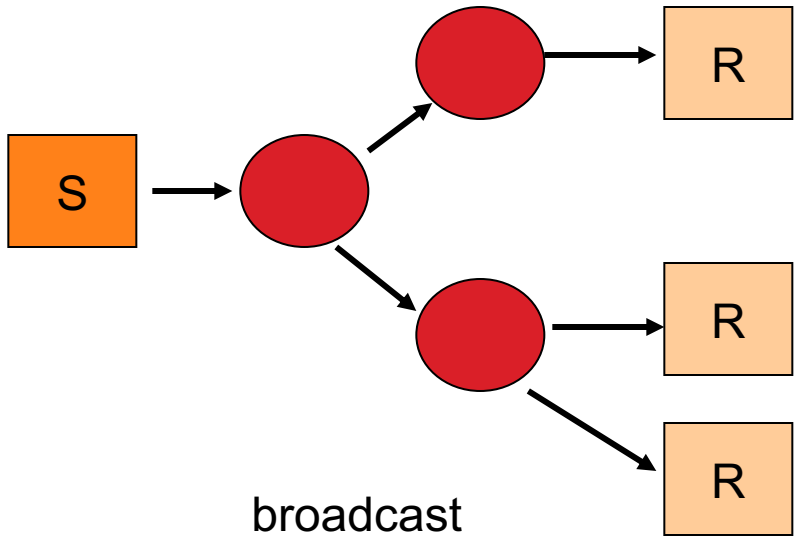
Equal, accept the frame

Broadcast: One to many communication

- Application level: one to many communication
- multiple unicasts

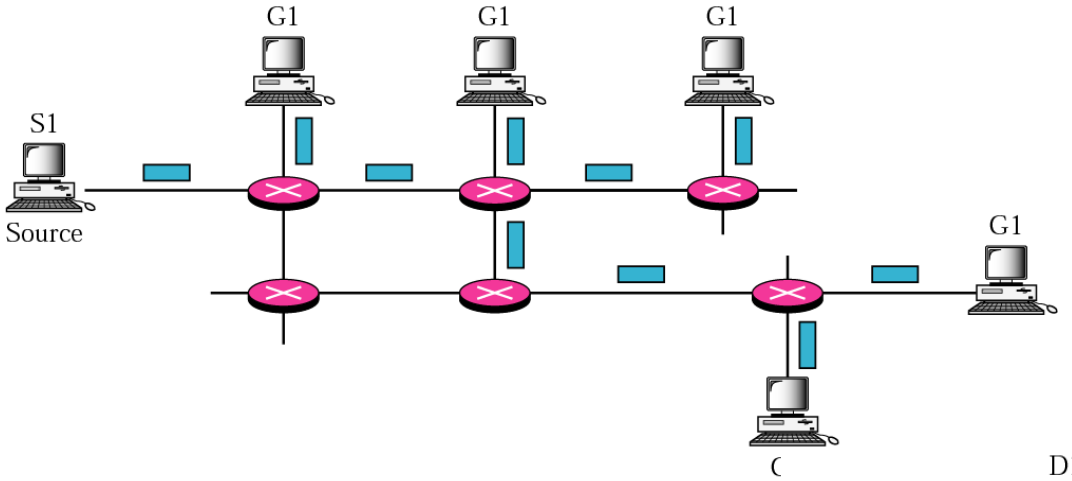


multiple unicasts

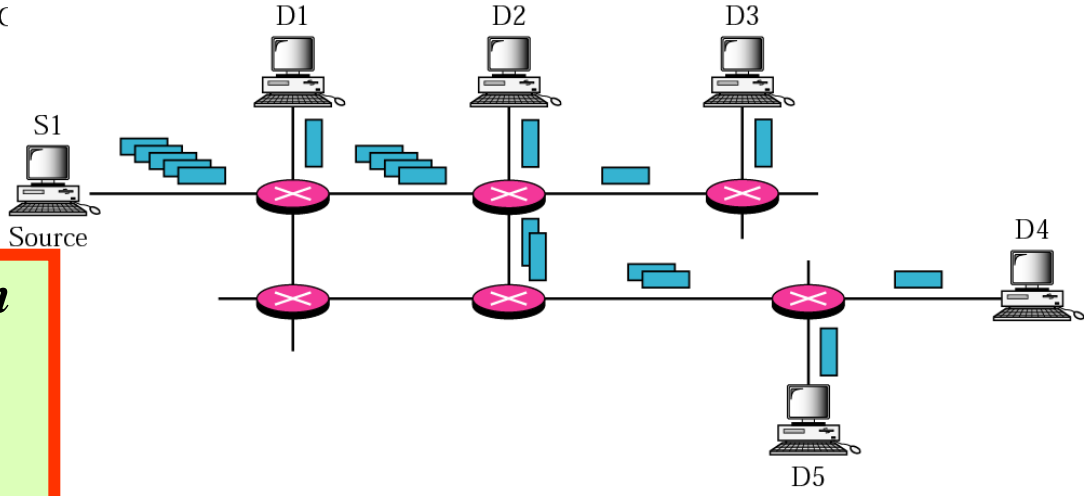


broadcast

Multicasting versus multiple unicasting



a. Multicasting



b. Multiple unicasting

Emulation of multicasting through multiple unicasting is not efficient and may create long delays, particularly with a large group.

Usage of Broadcasting

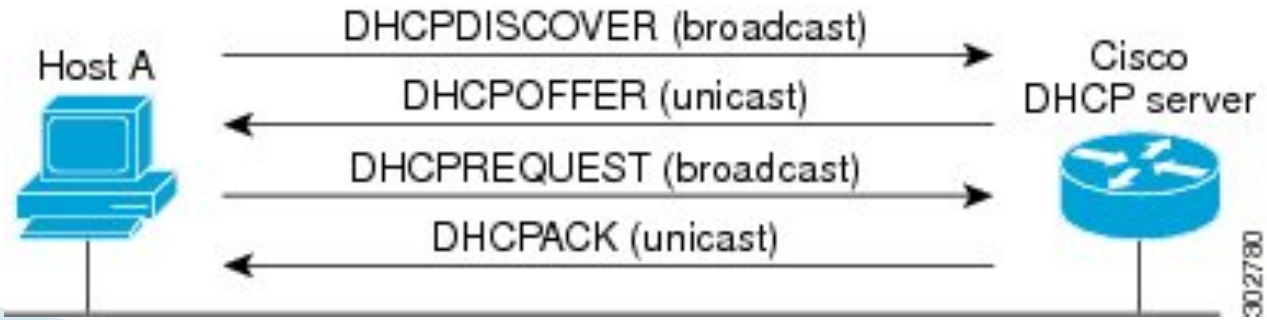
- Resource discovery
  - Locate a server on the local subnet when the serve's unicast IP address is not known
- Reducing network traffic
  - Minimize the network traffic on a LAN when there are multiple clients communicating with a single server
  - Numerous examples of Internet applications
    - Address Resolution Protocol (ARP)
    - Dynamic Host Configuration Protocol (DHCP)
    - Network Time Protocol (NTP)
    - Routing daemons

Internet Applications Using Broadcasting

- Address Resolution Protocol (ARP)
  - A protocol that lies underneath IPv4
  - Map IP network addresses to the hardware addresses used by a data link protocol (i.e., MAC address)
  - Uses link-layer broadcast to request the hardware address of a system given the system's IP address
  - Example: send the request "IP address of X.X.X.X please identify yourself with hardware address"

Internet Applications Using Broadcasting

- Dynamic Host Configuration Protocol (DHCP)
  - Network management protocol used on TCP/IP networks
  - Dynamically assigns an IP address and other network configuration parameters to each device in the network
  - DHCP client sends request to the broadcast address 255.255.255.255



Internet Applications Using Broadcasting

- Network Time Protocol (NTP)
  - The client needs to update time based on the time-of-day returned by the servers and the Round-Trip-Time (RTT) to the servers.
  - A server can broadcast the current time every 64 seconds for all the clients on a LAN
- Routing daemons
  - Routers broadcast their routing tables to other routers

Recall:
Network prefix and Host number

- IP address (v4) is 4 bytes long.
- The network prefix identifies a network and the host number identifies a specific host (actually, interface on the network).

- How do we know how long the network prefix is?
  - The network prefix used to be implicitly defined (**class-based addressing, A,B,C,D…**)

Class	Left-most Bit	Starting IP Address	Last IP Address
A	0xxx	0.0.0.0	127.255.255.255
B	10xx	128.0.0.0	191.255.255.255
C	110x	192.0.0.0	223.255.255.255
D	1110	224.0.0.0	239.255.255.255
E	1111	240.0.0.0	255.255.255.255

  - The network prefix now is flexible and is indicated by a **prefix**/**netmask (classless).**

Example

Example: www.example.com

- IP address is 128.143.137.144

- Using Prefix notation IP address is: **128.143.137.144/16**
  - Network prefix is 16 bits long

- Network mask is: **255.255.0.0**

  -----> Network id (IP address **AND** Netmask) is: **128.143.0.0**

  -----> Host number (IP address **AND** inverse of Netmask **0.0.255.255**) is: **137.144**

128.143	137.144
network prefix	host number

Broadcast addressing

- Broadcast addresses are special values in the host-identification part of an IP address
- Steps to obtain broadcast address of the subnet
  - Compute the bit complement of subnet mask
  - Perform a bitwise OR operation between the bit complement of subnet mask and the host's IP address.

Broadcast addressing

- Example: broadcasting a packet to an entire IPv4 subnet using IP address space 172.16.0.0/12

```
Address:    172.16.0.0          10101100.0001 0000.00000000.00000000
Netmask:    255.240.0.0         11111111.1111 0000.00000000.00000000
                                      12 bits
```

- The bit compliment of the mask 255.240.0.0 is 0.15.255.255 (00000000.00001111.11111111.11111111)

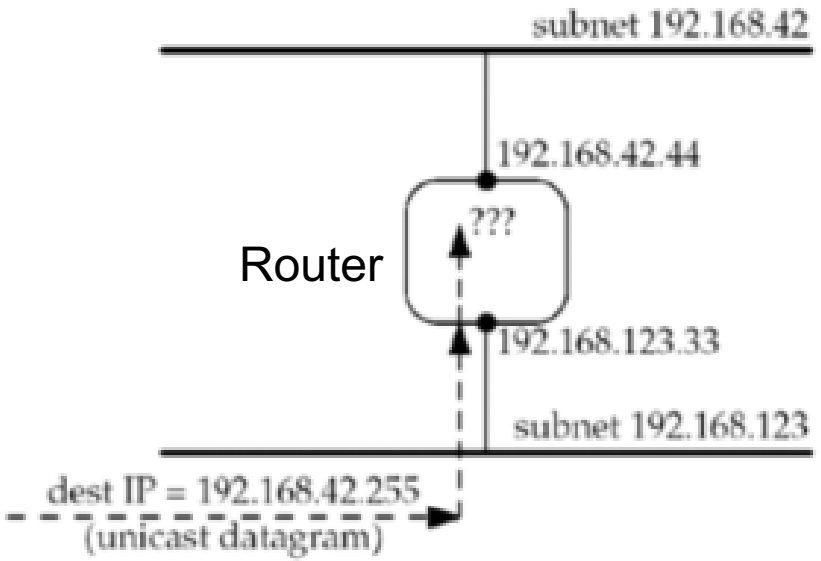
- The broadcast address is

```
10101100.00010000.00000000.00000000     172.16.0.0
                bitwise OR
00000000.00001111.11111111.11111111     0.15.255.255
= 10101100.00011111.11111111.11111111   172.31.255.255
```

This is the obtained broadcast address

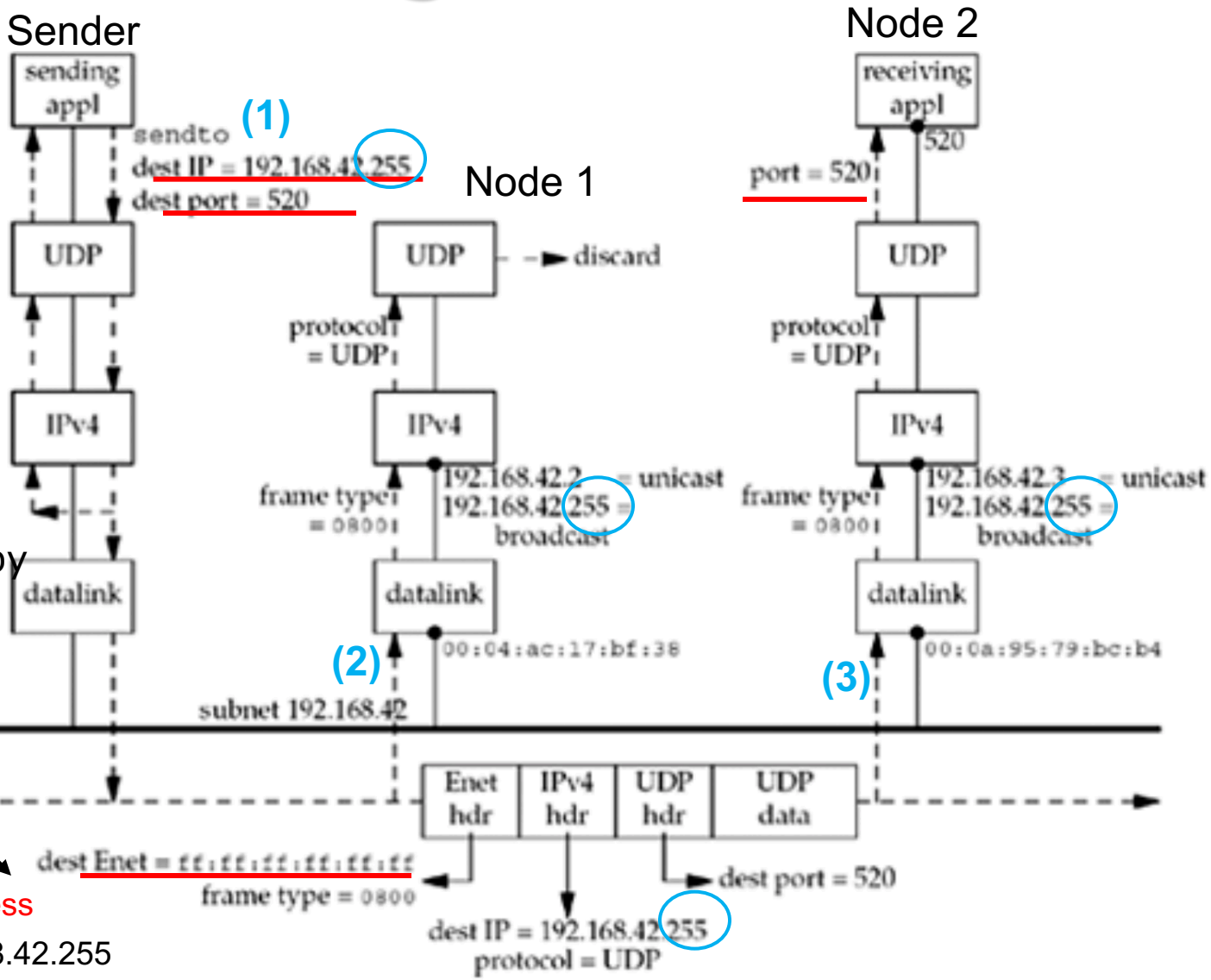
How to direct broadcast packet to the target subnet?

- Send broadcast packet using broadcast address (e.g., 192.168.42.255)
- Broadcast packet is forwarded as a unicast packet in the subnet that is not the target subnet
- Only the router connecting to the target subnet (e.g., 192.168.42) stops forwarding it to other subnets



Broadcast Datagrams

Note 1 and node 2 are in the target subnet



Obtained the corresponding Ethernet address by Address Resolution Protocol (ARP)

Broadcast address

dest IP = 192.168.42.255
Protocol = UDP
dest Enet = ff:ff:ff:ff:ff:ff

A broadcast way in local network

- A special broadcast address 255.255.255.255
- Stands for “all the nodes in this network” (the local network)
- Transmission to this address is limited within the local network and it is not forwarded by the routers to other networks
- Note that Internet Protocol version 6 (IPv6) does not implement broadcast

Using Broadcast

- First need to enable broadcast in socket option
  - setsockopt()

  int setsockopt (int socket, int level, int option_name, const void *option_value, socklen_t option_len);
    - level (protocol level such as socket level SOL_SOCKET or TCP level IPPROTO_TCP)
    - option_name (such as broadcast option SO_BROADCAST)
    - option_value ("1" to enable and "0" to disable the option)
  - Example

    ```
    const int on = 1;
    setsockopt(sockfd, SOL_SOCKET, SO_BROADCAST, &on, sizeof(on));
    ```

- Use standard system calls with broadcast address to send (e.g, the subset broadcast address and 255.255.255.255)

Example (Broadcast example): lecture_11_code/broadcast/broadcast_sender.c

```
#include <sys/socket.h>
#include <arpa/inet.h>
#include <strings.h>
#include <stdio.h>
#include <unistd.h>
int main() {
 /*create a udp socket*/
  int sockfd = socket(PF_INET, SOCK_DGRAM, 0); //create UDP socket

  char msg[] = "This is a broadcast message.\n";

  struct sockaddr_in dest;
  bzero(&dest, sizeof(dest));
  dest.sin_family = PF_INET;
//use the broadcast address 255.255.255.255
  inet_pton(PF_INET,"255.255.255.255",&(dest.sin_addr.s_addr));
  dest.sin_port = htons(10000);
  printf("The broadcast sender is on.");
```

In this program, the sender periodically sends "This is a broadcast message" to all the clients in the local network

Example (Broadcast example): lecture_11_code/broadcast/broadcast_sender.c

```
  int on=1; //Enable the socket option
//Set the socket option to be broadcast
  setsockopt(sockfd, SOL_SOCKET, SO_BROADCAST, &on, sizeof(on));


  //Broadcast a message every 5 seconds
  while(1) {
    printf("The broadcast sender sends: This is a broadcast message.\n");
    sendto(sockfd, msg, sizeof(msg), 0, (struct sockaddr*) &dest, sizeof(dest));
    sleep(5);

 }
}
```

Example (Broadcast example): lecture_11_code/broadcast/broadcast_receiver.c

```
#include <sys/socket.h>
#include <arpa/inet.h>
#include <strings.h>
#include <stdio.h>
#include <unistd.h>

int main() {

  int sockfd = socket(PF_INET, SOCK_DGRAM, 0); //create a UDP socket

  struct sockaddr_in dest;
  bzero(&dest, sizeof(dest));
  dest.sin_family = PF_INET;
  dest.sin_addr.s_addr=INADDR_ANY;    //set the client's address to be INADDR_ANY
  dest.sin_port = htons(10000);
```

Example (Broadcast example): lecture_11_code/broadcast/broadcast_receiver.c

```
  // bind the socket address to the receiver's socket
if( bind(sockfd, (struct sockaddr*) &dest, sizeof(dest)) < 0) {
    perror("bind");
    return 1;
  }
  int on=1;//enable the socket option
//set socket option to be broadcast
  setsockopt(sockfd, SOL_SOCKET, SO_BROADCAST, &on, sizeof(on));

  printf("The broadcast receiver is started. ");
  while(1) {
    char recvmsg[100];
    struct sockaddr_in src;
    socklen_t size=sizeof(src);
    recvfrom(sockfd, recvmsg, sizeof(recvmsg), 0, (struct sockaddr*) &dest, &size);
    printf("Received: %s",recvmsg);
 }
}
```

Sender: Terminal 1:
$ ./server

Receiver: Terminal 2 on this or other hosts:
$ ./client

Disadvantage of Broadcasting

- Broadcasting sends a datagram that all hosts on the designated network need to receive and process the broadcast message
  - The host not participating in the application need to process
  - Place an excessive processing load on these hosts especially for high data rate applications, such as audio or video

Multicasting solves the problem by only involving the hosts in interest to receive the datagram

Why Multicast

- When sending same data to multiple receivers
  - Better bandwidth utilization
  - Less host/router processing
- Application
  - Video/Audio multicast (One sender)
  - Video conferencing (Many senders)
  - Real time news distribution
  - Interactive gaming

Multicast Addresses

- Identify a set of IP interfaces
- Can be used on a LAN or across a WAN, while broadcast is limited to a LAN
- IPv4 Class D addresses are used for multicast
  - 224.0.0.0 through 239.255.255.255
- Each multicast address represents a group of arbitrary size, called a “host group”

- Some special IPv4 multicast addresses
  - 224.0.0.1 (all hosts group)
    - all multicast-capable nodes (hosts, routers, printers etc.) on a subnet must join this group.
  - 224.0.0.2 (all routers group)
    - All multicast-capable routers on a subnet must join this group

Multicast groups

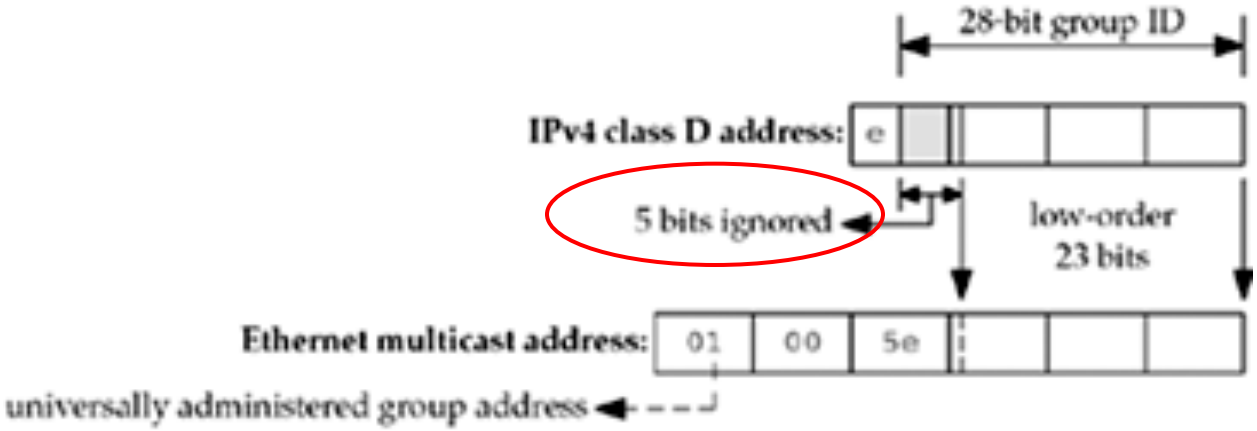
- class D Internet addresses reserved for multicast:

- host group semantics: ← 28 bits →
  - anyone can "join" (receive) multicast group
  - anyone can send to multicast group
  - no network-layer identification to hosts of members
- *needed:* infrastructure to deliver multicast-addressed datagrams to all hosts that have joined that multicast group

Multicast Ethernet Addresses

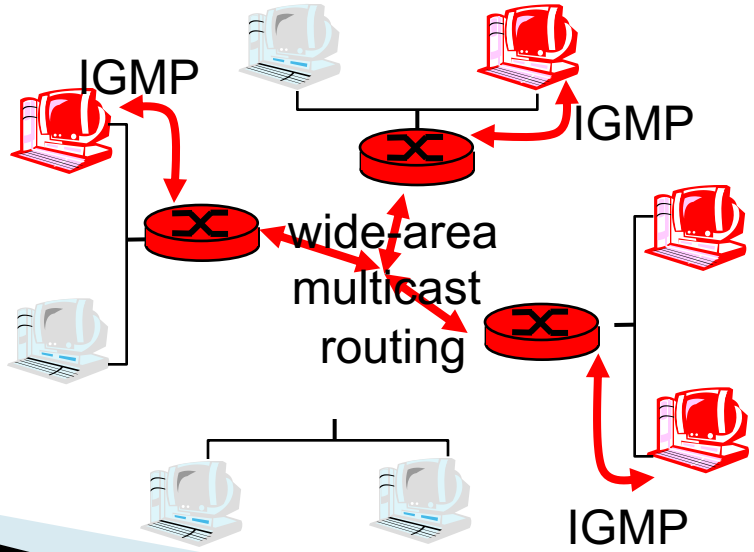
- Mapping of IPv4 multicast address (32bits) to Ethernet Addresses (48bits)
  - The higher 24 bits are 01:00:5e
  - The next bit is always 0
  - The low-order 23 bits are copied from the low-order 23 bits of the multicast group address
  - Not one-to-one mapping

01:00:5e + 0 + Low-order 23 bit of group ID

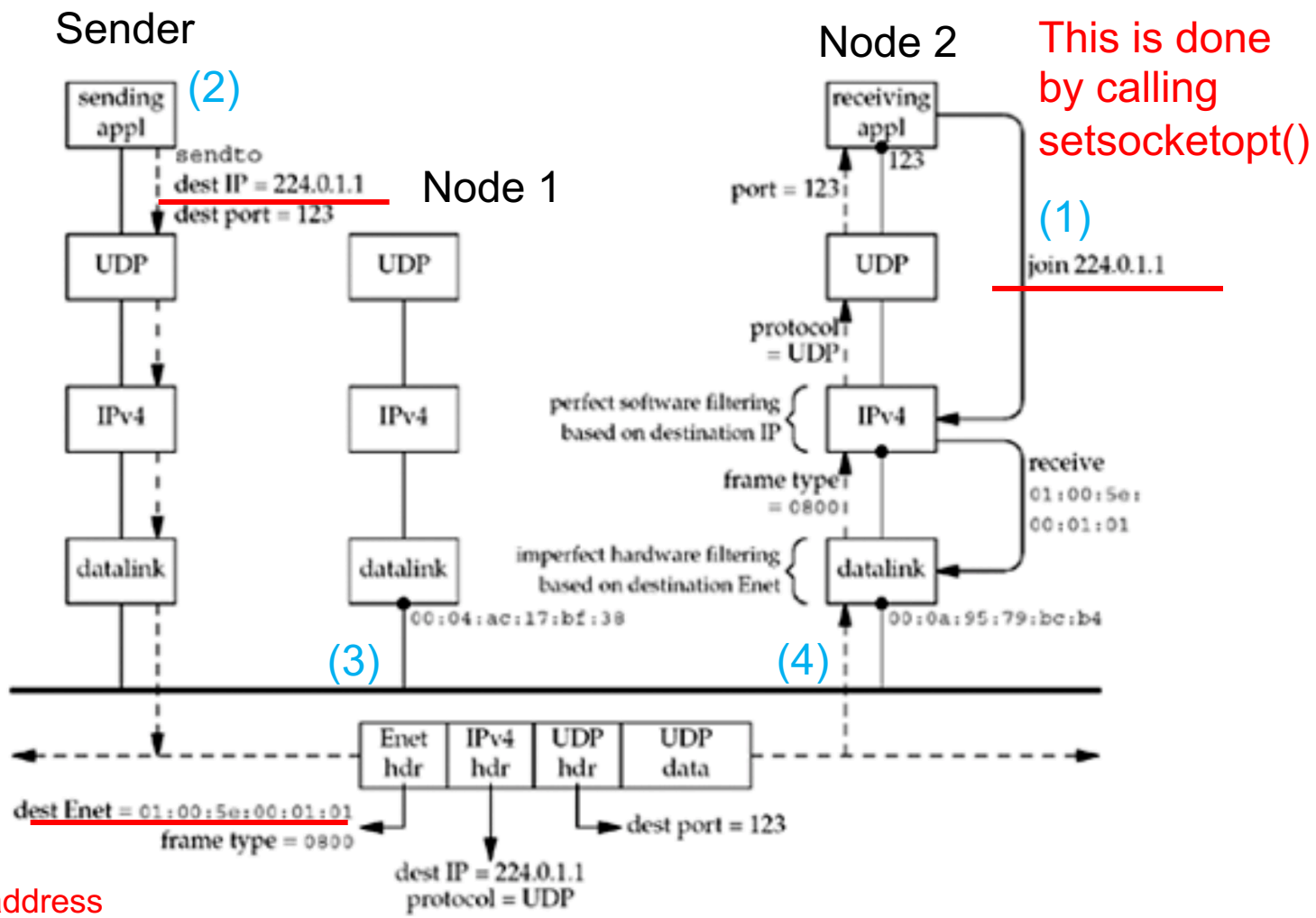


Joining a mcast group: two-step process

- *local:* host informs local multicast router of desire to join group: IGMP (Internet Group Management Protocol)
- *wide area:* local router interacts with other routers to receive mcast datagram flow
  - many protocols (e.g., DVMRP, MOSPF, PIM)



Multicast Datagrams



Multicast address

dest IP = 224.0.1.1
Protocol = UDP

dest Enet = 01:00:5e:00:01:01

Node 1: 00:04:ac:17:bf:38

Didn't join the group, ignore the frame

Node 2: 00:0a:95:79:bc:b4

Joined the group, accept the frame

Using Multicast

- At the sender: send message using a multicast address / group
- At the receiver: join the multicast group and receive the message
  - Set socket option to IP_ADD_MEMBERSHIP
  - Use the ip_mreq structure when setting the socket option

Ip_mreq Structure

- ip_mreq
  - A structure provides multicast group information for IPv4 addresses

    typedef struct ip_mreq {
      struct in_addr imr_multiaddr;
      struct in_addr imr_interface;
    } IP_MREQ, *PIP_MREQ;

  - imr_multiaddr
    - The address of the IPv4 multicast group
  - imr_interface
    - The local IPv4 address of the client

Set Socket Option for Multicast

- Set socket option at the receiver
- setsockopt(fd,IPPROTO_IP, IP_ADD_MEMBERSHIP,&mreq,sizeof(mreq))
  - IPPROTO_IP
    - The socket option level for IP
  - IP_ADD_MEMBERSHIP
    - Join the socket to the supplied multicast group on the specified interface.

Example (Multicast example): lecture_11_code/multicast/multicast_sender.c

```
#include <sys/socket.h>
#include <arpa/inet.h>
#include <strings.h>
#include <stdio.h>
#include <unistd.h>
int main() {

  int sockfd = socket(PF_INET, SOCK_DGRAM, 0); //create UDP socket

  char msg[] = "This is a multicast message.\n";  //The message to multicast
 printf("The multicast sender is on.\n ");
  struct sockaddr_in dest;
  bzero(&dest, sizeof(dest));
  dest.sin_family = PF_INET;

  //Use the multicast address 225.0.0.37 to multicast a message
  inet_pton(PF_INET,"225.0.0.37",&(dest.sin_addr.s_addr));
  dest.sin_port = htons(10000);
```

Example (Multicast example): lecture_11_code/multicast/multicast_sender.c

```
  //int on=1;
  //setsockopt(sockfd, SOL_SOCKET, SO_BROADCAST, &on, sizeof(on));
  printf("The multicast sender sends the message: This is a multicast message.\n ");

//Send the multicast every 5 seconds
  while(1) {
    sendto(sockfd, msg, sizeof(msg), 0, (struct sockaddr*) &dest, sizeof(dest));
    sleep(5);
 }
}
```

Example (Multicast example): lecture_11_code/multicast/multicast_receiver.c

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <time.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

void main(int argc, char *argv[])
{
    struct sockaddr_in addr;
    int fd, nbytes;
    socklen_t addrlen;
    struct ip_mreq mreq; //structure provides multicast group information for IPv4 addresses
    char msgbuf[256]; //message buffer
```

Example (Multicast example): lecture_11_code/multicast/multicast_receiver.c

```
/* create an ordinary UDP socket */
    if ((fd=socket(AF_INET,SOCK_DGRAM,0)) < 0) {
            perror("socket");
            exit(1);
    }

    /* set up destination address */
    memset(&addr,0,sizeof(addr));
    addr.sin_family=AF_INET;
    addr.sin_addr.s_addr=htonl(INADDR_ANY); /* set the client's address to be INADDR_ANY*/
    addr.sin_port=htons(10000);

    /* bind to client's address */
    if (bind(fd,(struct sockaddr *) &addr,sizeof(addr)) < 0) {
            perror("bind");
            exit(1);
    }
```

Example (Multicast example): lecture_11_code/multicast/multicast_receiver.c

```
/* use setsockopt() to request joining a 225.0.0.37 group */
    mreq.imr_multiaddr.s_addr=inet_addr("225.0.0.37");
    mreq.imr_interface.s_addr=htonl(INADDR_ANY);
    if (setsockopt(fd,IPPROTO_IP, IP_ADD_MEMBERSHIP,&mreq,sizeof(mreq)) < 0) {
            perror("setsockopt");
            exit(1);
    }
    printf("The multicast receiver is on.");
    /* now enter the listening loop and print the received multicast message */
    while (1) {
            addrlen=sizeof(addr);
            if ((nbytes=recvfrom(fd,msgbuf,256,0,
                                    (struct sockaddr *) &addr,&addrlen)) < 0) {
                perror("recvfrom");
                exit(1);
            }
            puts(msgbuf);
    }
}
```

Sender: Terminal 1

$ ./server

Receiver: Terminal 2 on this or other hosts:

$ ./client

Homework Readings

- UNP Unix Network Programming
  - Chapter 20 and 21

Assignment 4 (Final Exam)

Security Assignment
3 weeks

Problem

- This assignment will help you gain firsthand experience with one of the methods commonly used to exploit security weaknesses in operating systems and network servers.

- It involves applying a series of *buffer overflow attacks* on an executable file bufbomb in the lab directory.

Detailed Information

- You can obtain your buffer bomb by pointing your Web browser at:
- http://vlsi3.engr.rutgers.edu:18213/
- The server will return a **tar** file called **buflab-handout.tar** to your browser.

- Upon receiving your solution, the server will validate your string and update the Buffer Lab scoreboard Web page, which you can view by pointing your Web browser at
- http://vlsi3.engr.rutgers.edu:18213/scoreboard

- **IMPORTANT NOTE:** You can work on your buffer bomb on any Linux machine. But in order to submit your solution, you will need to be **running on the computers in EE-103.**