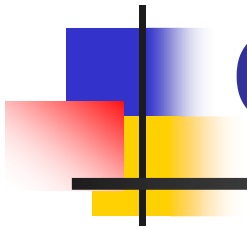


# EC 553
# Communication Networks

Mohamed Khedr

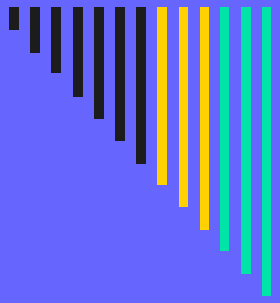
www.aast.edu/~khedr

http://www.aast.edu/~khedr/Courses/Undergraduate/Communication%20Networks%20EC553/

# Syllabus

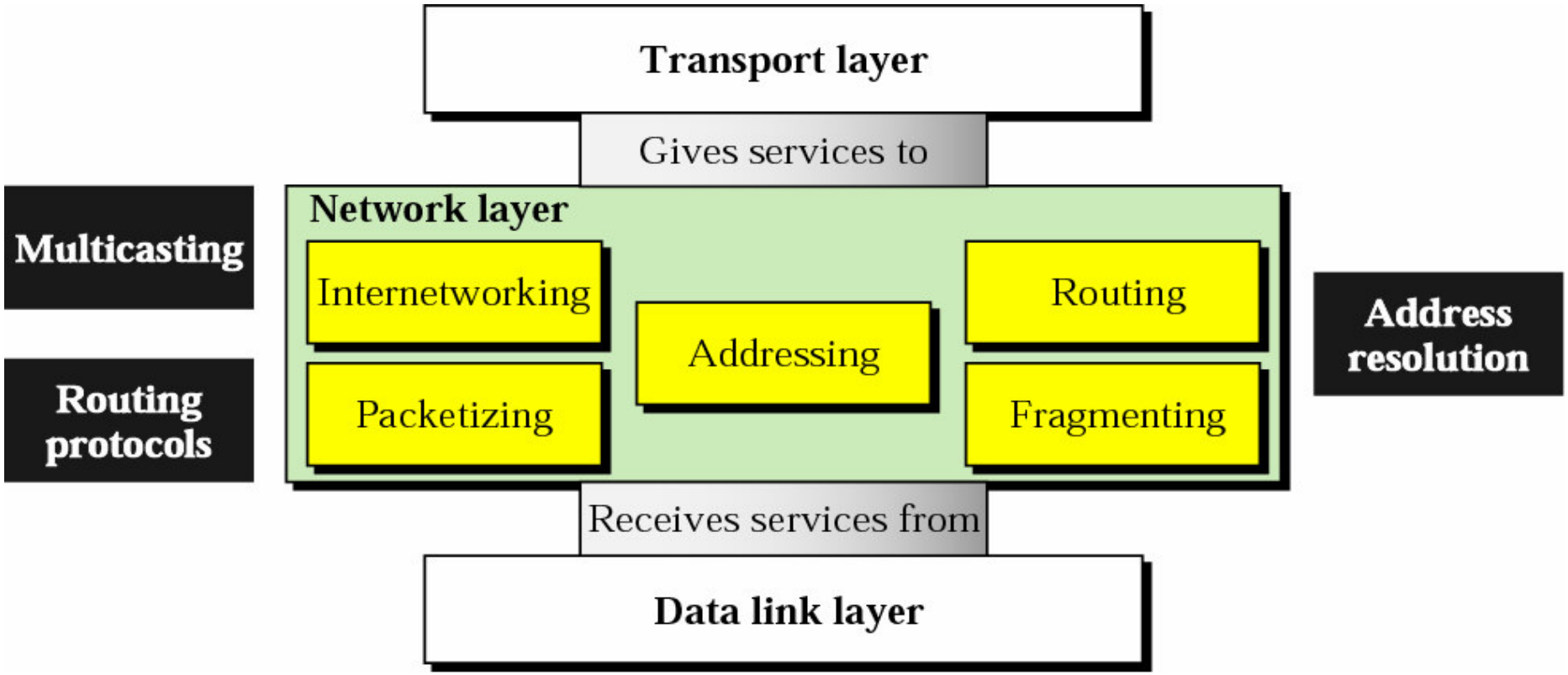
- Tentatively

| | |
|---|---|
| Week 1 | Overview |
| Week 2 | Packet Switching |
| Week 3 | **IP addressing and subnetting** |
| Week 4 | Introduction to Routing concept |
| Week 5 | Routing algorithms |
| Week 6 | Routing protocols |
| Week 7 | Multiple Access I |
| Week 8 | Multiple access II |
| Week 9 | LAN networks |
| Week 10 | Token ring networks |
| Week 11 | VOIP |
| Week 12 | WLAN |
| Week 13 | TCP |
| Week 14 | Congestion control |
| Week 15 | QOS |

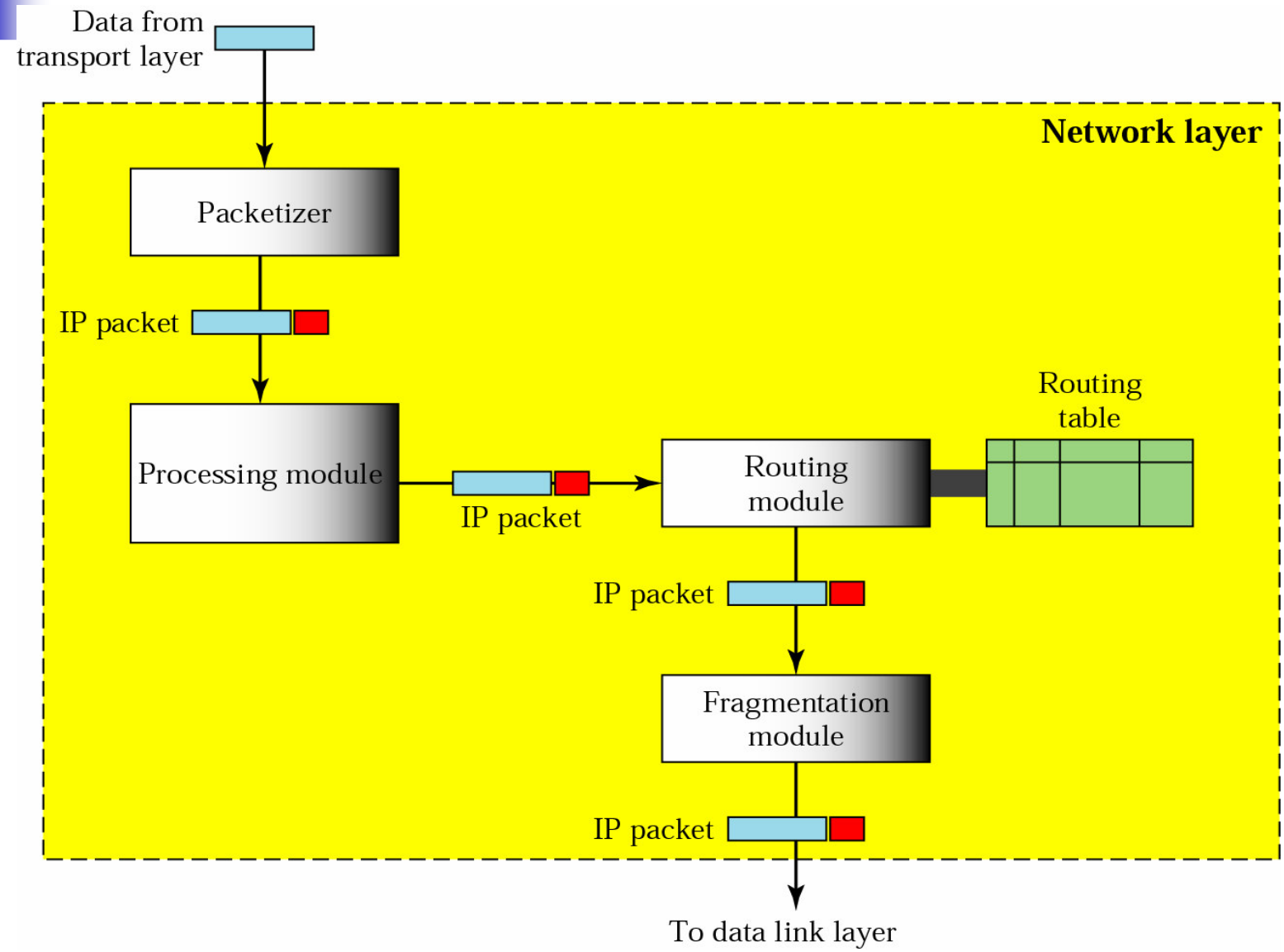


# Network Layer

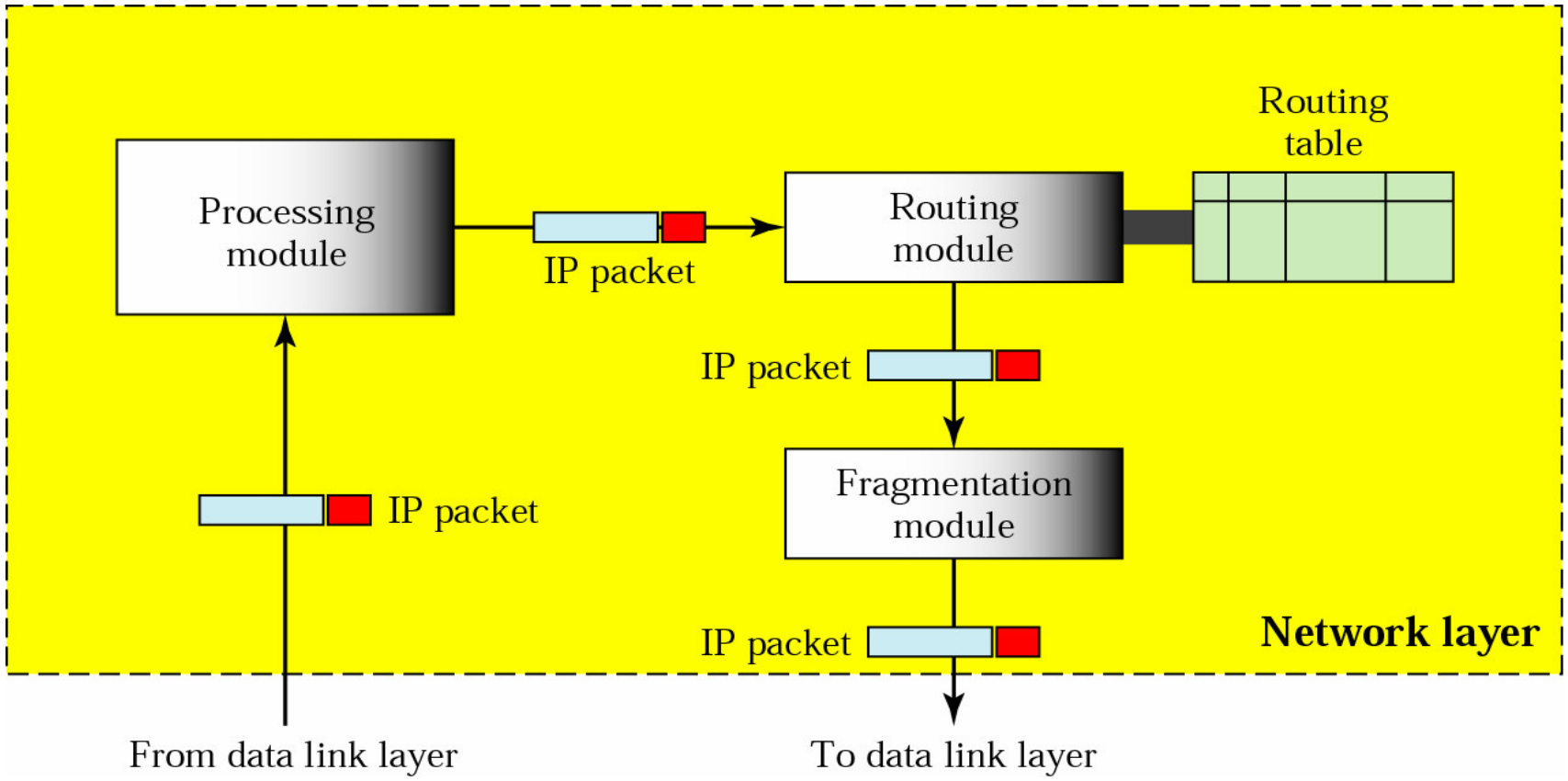
# Position of network layer



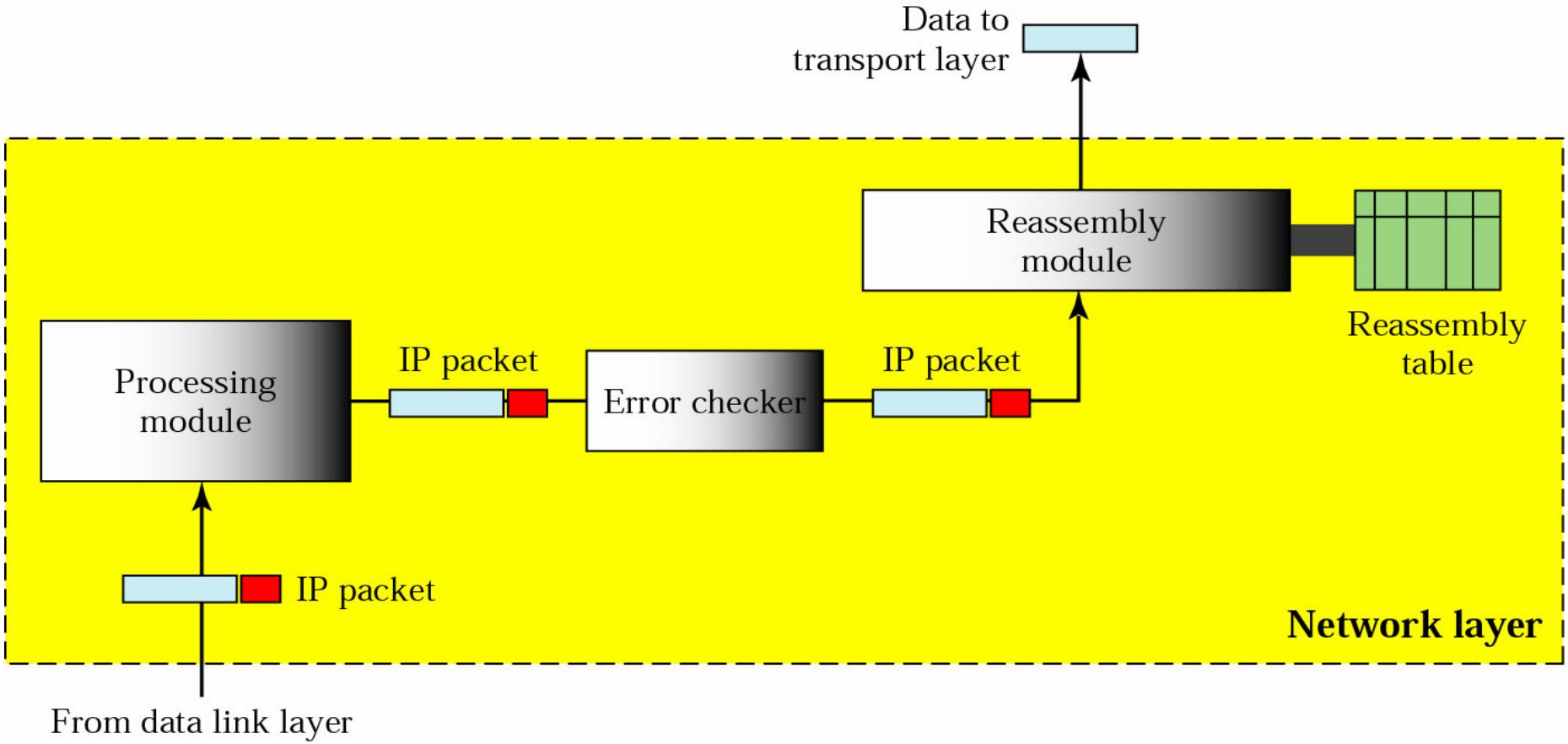
## Figure 19.4 Network layer at the source



## Figure 19.5 Network layer at a router



## Figure 19.6 Network layer at the destination



# 19.2 Addressing

***Internet Address***

***Classful Addressing***

***Subnetting***

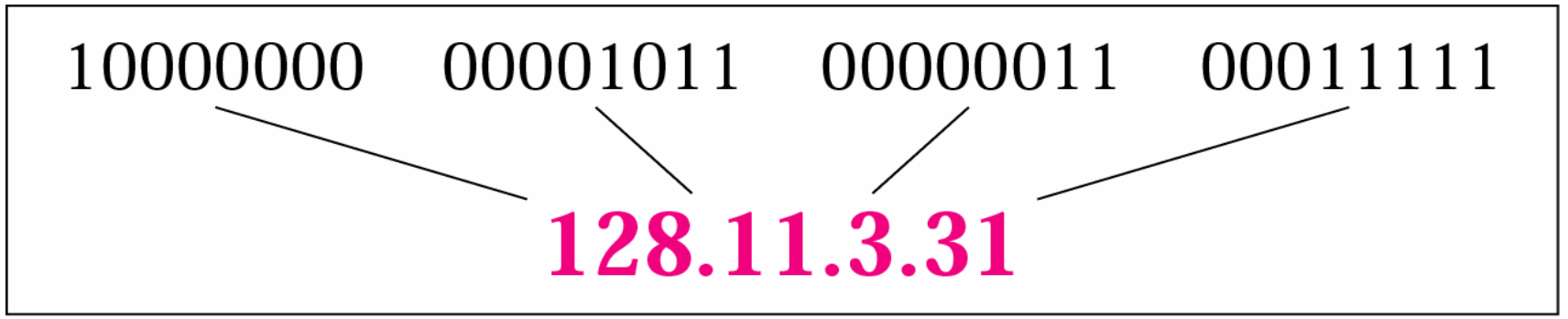
***Network Address Translation***

**Note:**

***An IP address is a 32-bit address.***

***The IP addresses are unique and universal.***

## Figure 19.9 Dotted-decimal notation



## *Example 1*

Change the following IP addresses from binary notation to dotted-decimal notation.

a. 10000001 00001011 00001011 11101111

b. 11111001 10011011 11111011 00001111

## *Solution*

**We replace each group of 8 bits with its equivalent decimal number (see Appendix B) and add dots for separation:**

**a. 129.11.11.239**

**b. 249.155.251.15**

## *Example 2*

Change the following IP addresses from dotted-decimal notation to binary notation.

a. 111.56.45.78

b. 75.45.34.78

## *Solution*

**We replace each decimal number with its binary equivalent**

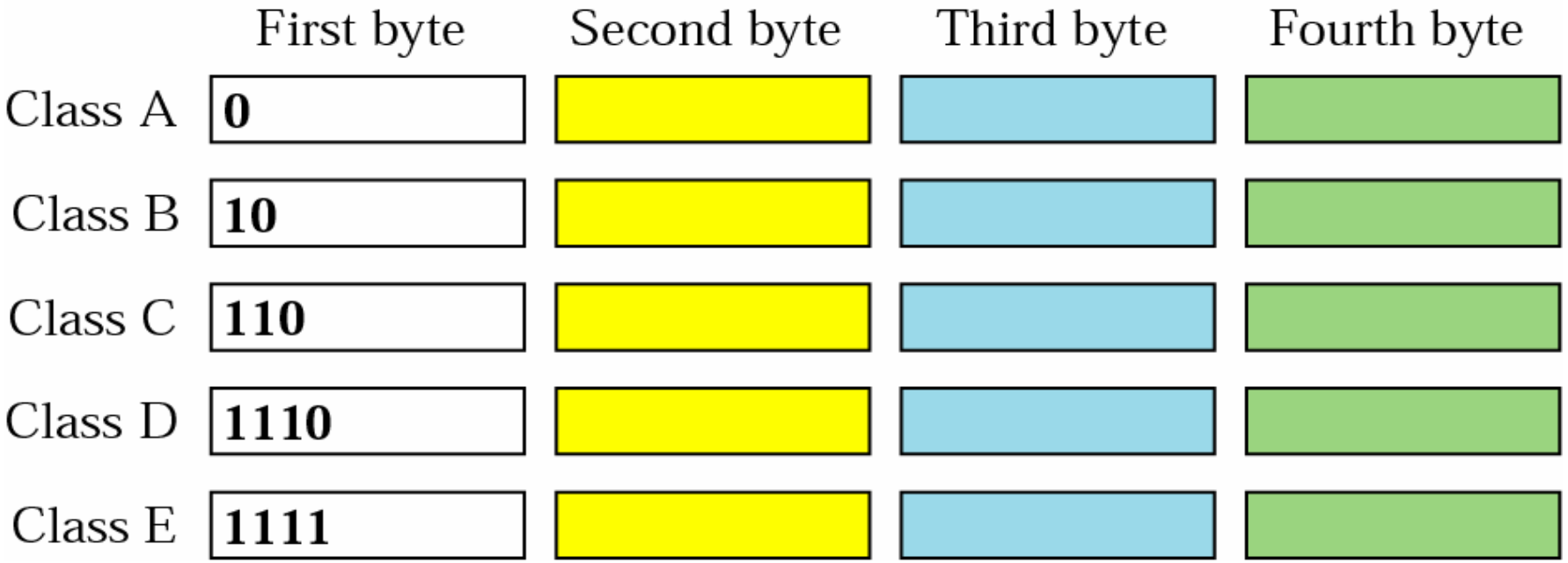
**a. 01101111 00111000 00101101 01001110**

**b. 01001011 00101101 00100010 01001110**

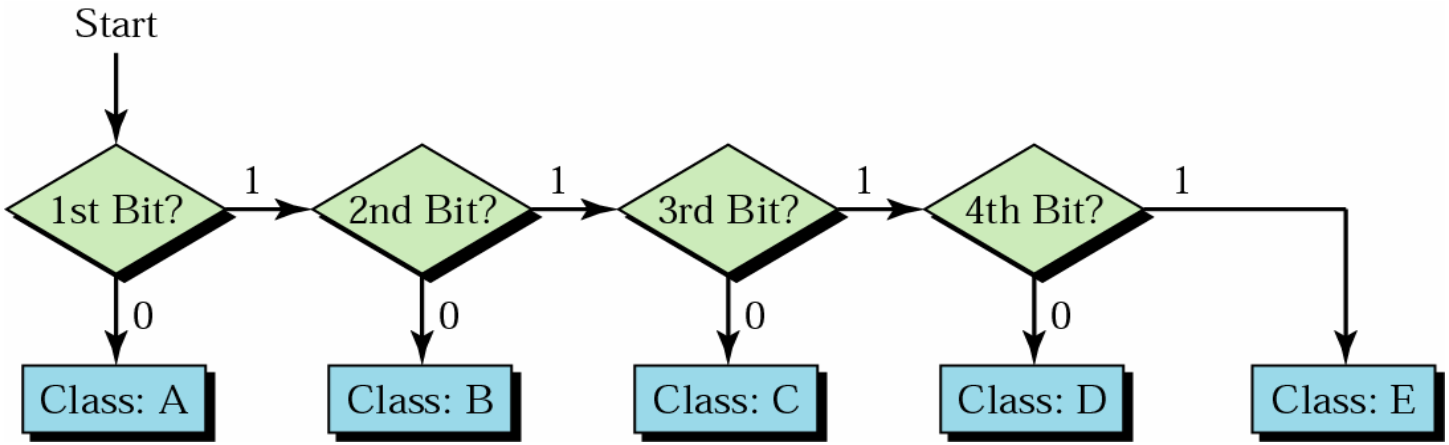
**Note:**

***In classful addressing, the address space is divided into five classes: A, B, C, D, and E.***

## Figure 19.10 Finding the class in binary notation



## Figure 19.11 Finding the address class



## *Example 3*

Find the class of each address:

a. **0**0000001 00001011 00001011 11101111

b. **1111**0011 10011011 11111011 00001111

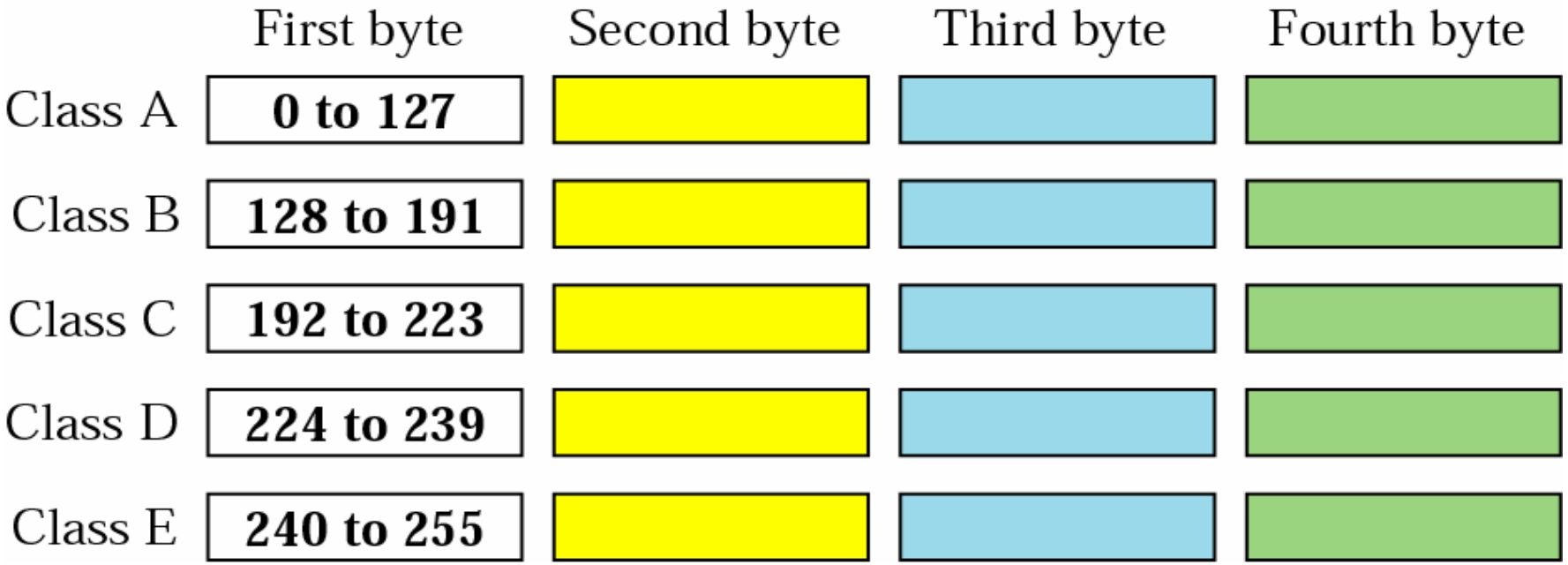
## *Solution*

**See the procedure in Figure 19.11.**

**a. The first bit is 0; this is a class A address.**

**b. The first 4 bits are 1s; this is a class E address.**

## Figure 19.12 Finding the class in decimal notation



## *Example 4*

Find the class of each address:

a. **227**.12.14.87

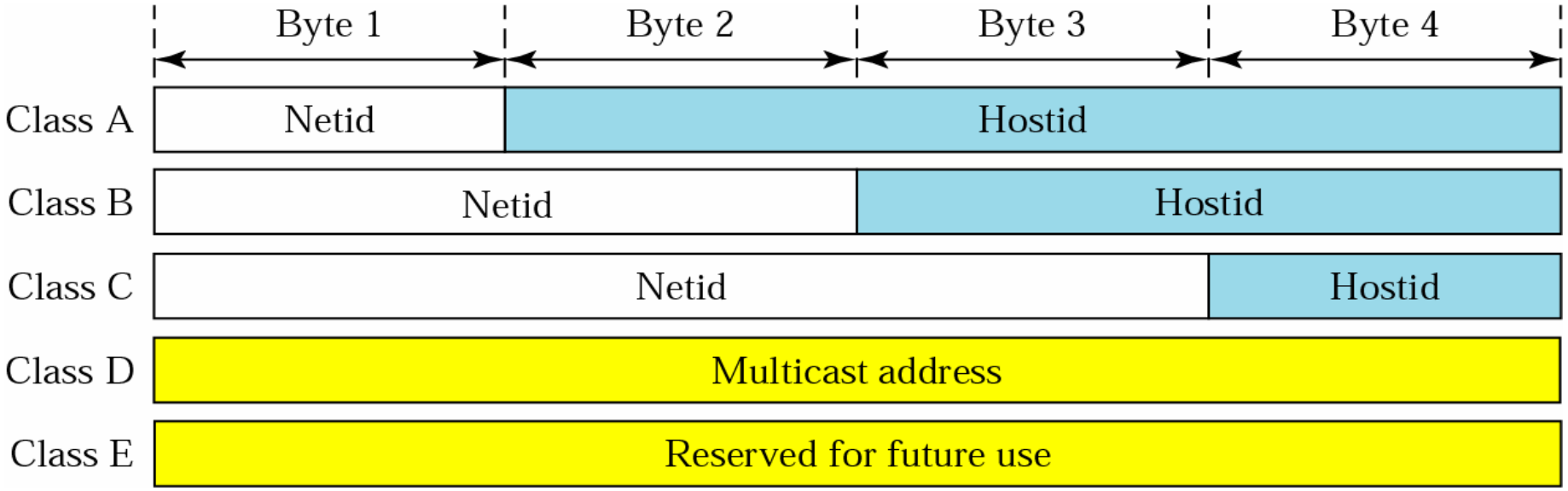
b. **252**.5.15.111

c. **134**.11.78.56

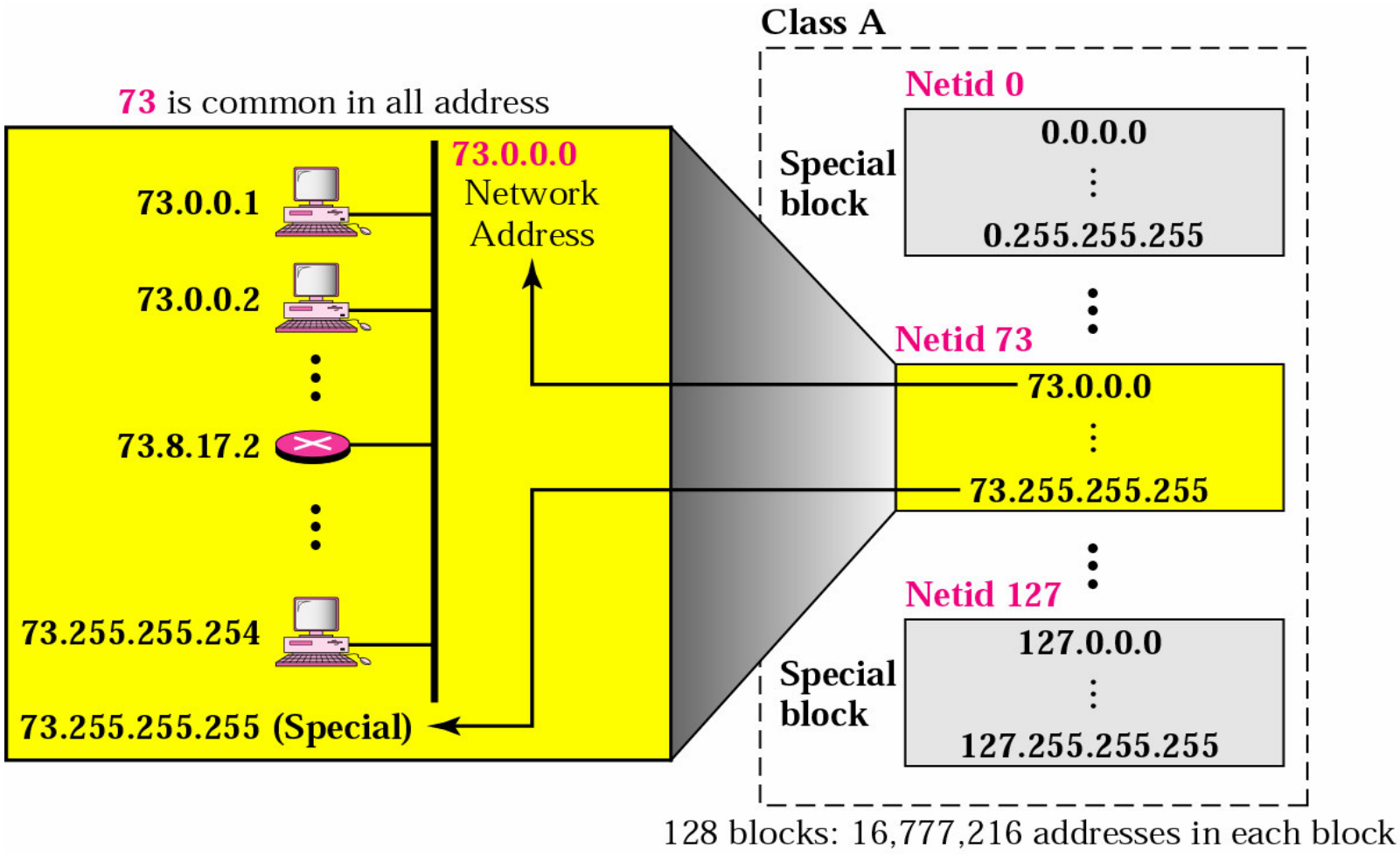
## *Solution*

**a. The first byte is 227 (between 224 and 239); the class is D.**
**b. The first byte is 252 (between 240 and 255); the class is E.**
**c. The first byte is 134 (between 128 and 191); the class is B.**

## Figure 19.13 Netid and hostid



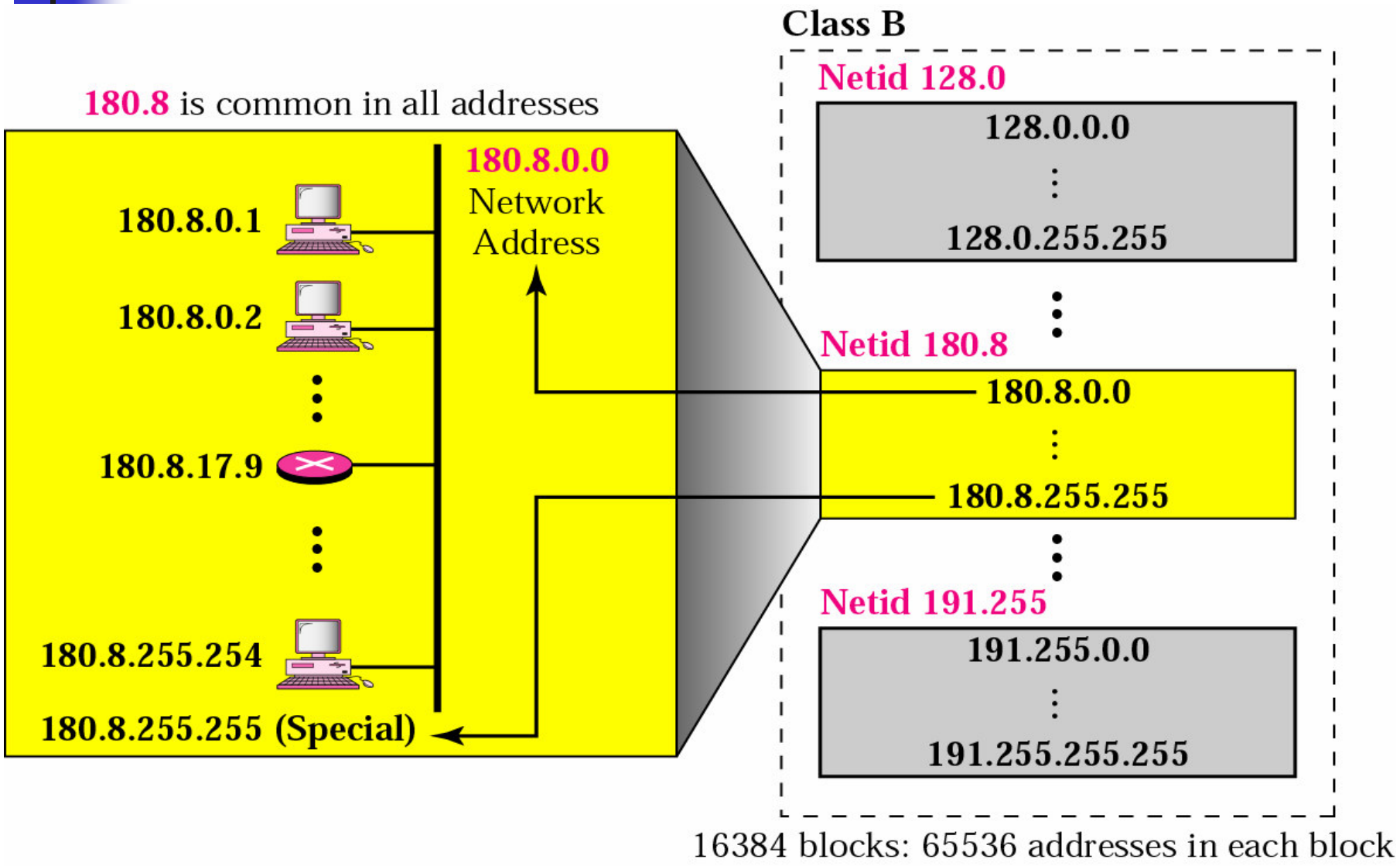
## Figure 19.14 Blocks in class A



**Note:**

***Millions of class A addresses are wasted.***

## Figure 19.15 Blocks in class B



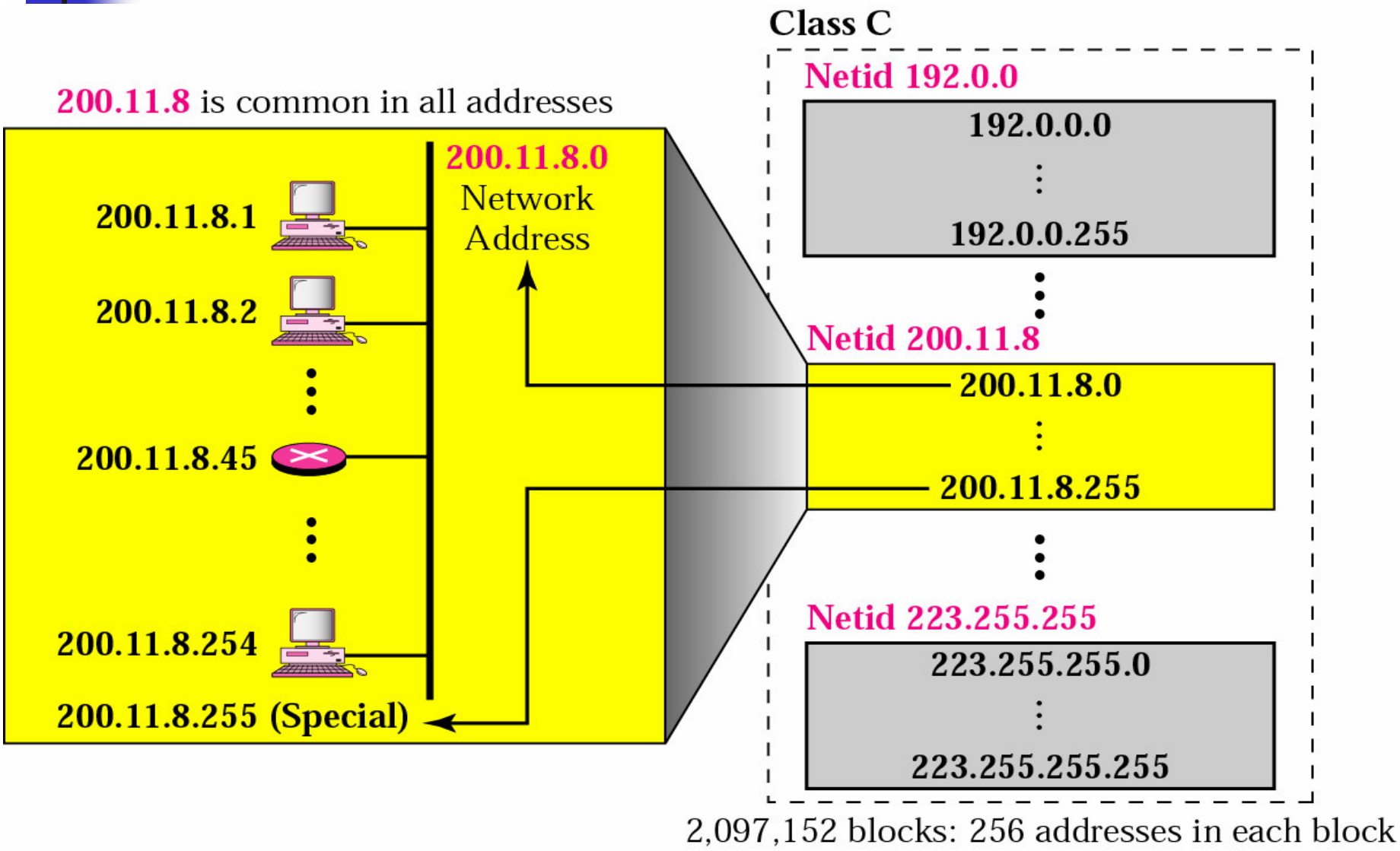
**Note:**

***Many class B addresses are wasted.***

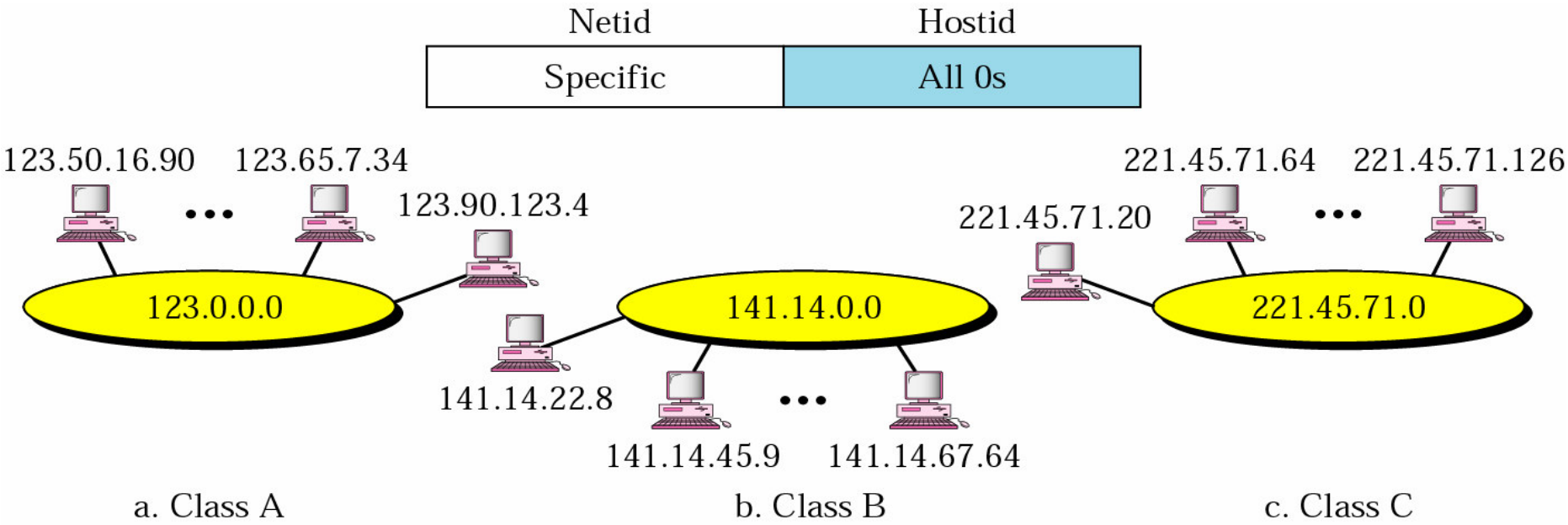
**Note:**

***The number of addresses in class C is smaller than the needs of most organizations.***

## Figure 19.16 Blocks in class C



## Figure 19.17 Network address



**Note:**

***In classful addressing, the network address is the one that is assigned to the organization.***

## *Example 5*

Given the address 23.56.7.91, find the network address.

## *Solution*

**The class is A. Only the first byte defines the netid. We can find the network address by replacing the hostid bytes (56.7.91) with 0s. Therefore, the network address is 23.0.0.0.**

## *Example 6*

Given the address 132.6.17.85, find the network address.

## *Solution*

**The class is B. The first 2 bytes defines the netid. We can find the network address by replacing the hostid bytes (17.85) with 0s. Therefore, the network address is 132.6.0.0.**

## *Example 7*

Given the network address 17.0.0.0, find the class.

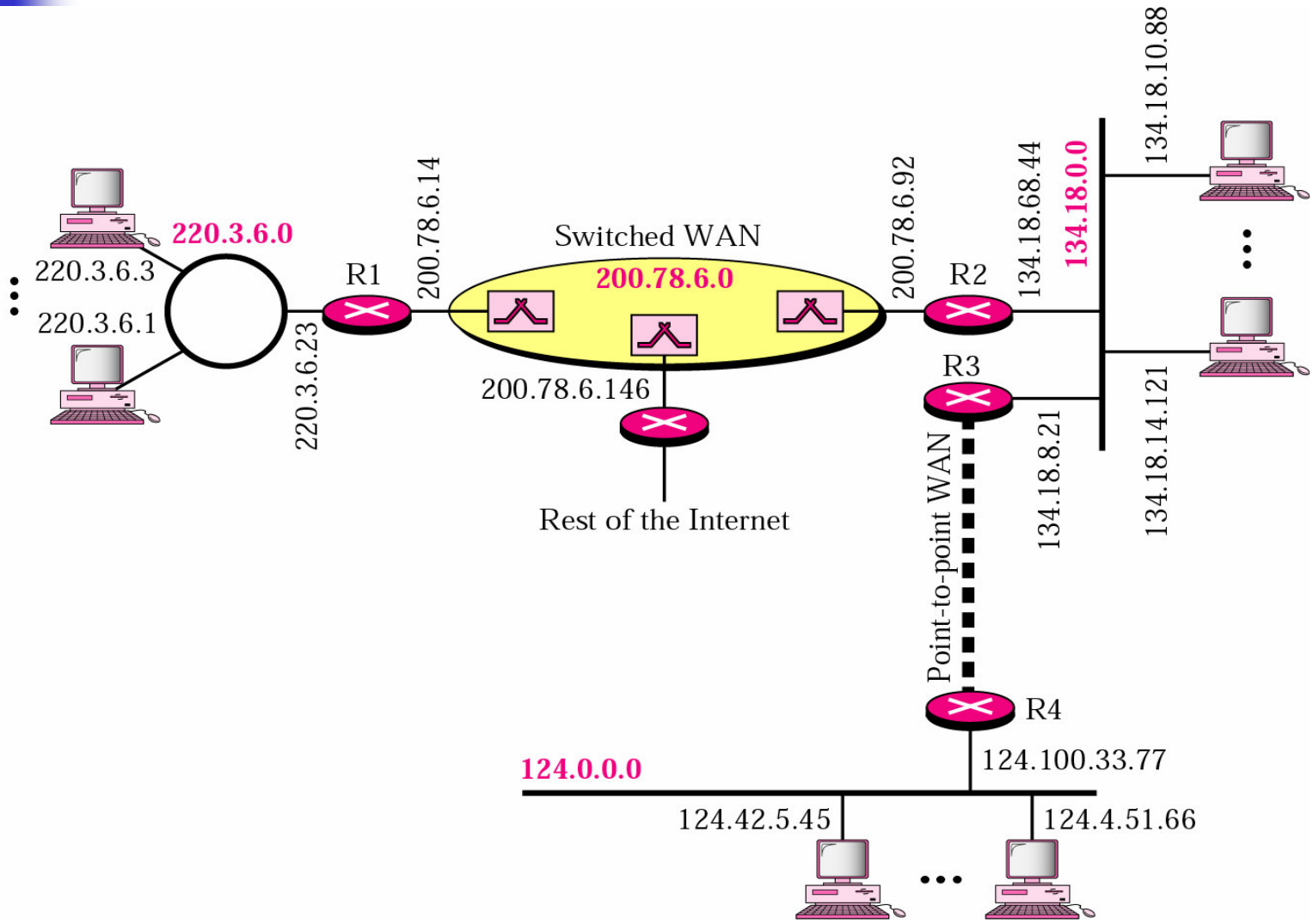
## *Solution*

**The class is A because the netid is only 1 byte.**

**Note:**

***A network address is different from a netid. A network address has both netid and hostid, with 0s for the hostid.***

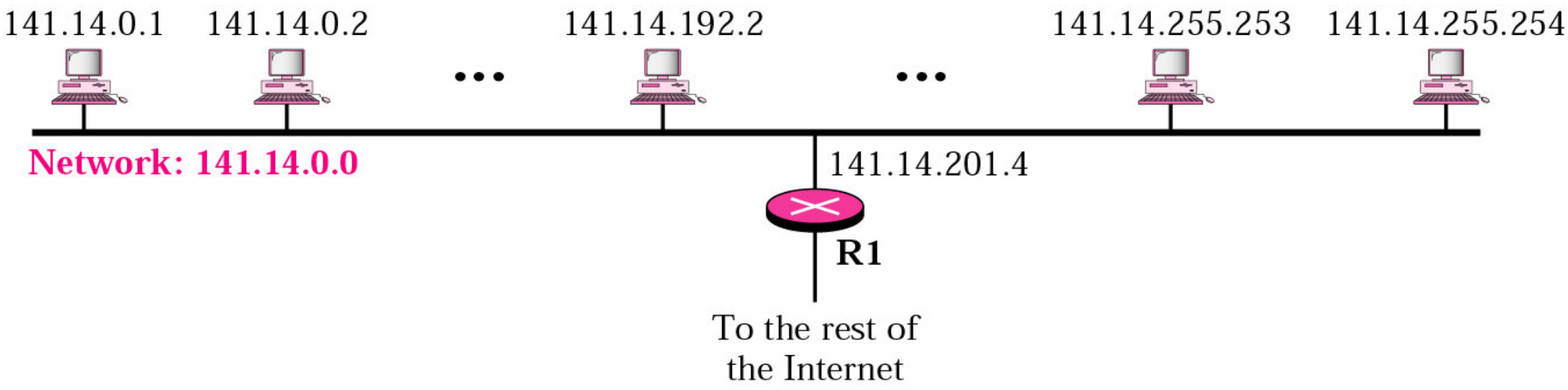
## Figure 19.18 Sample internet



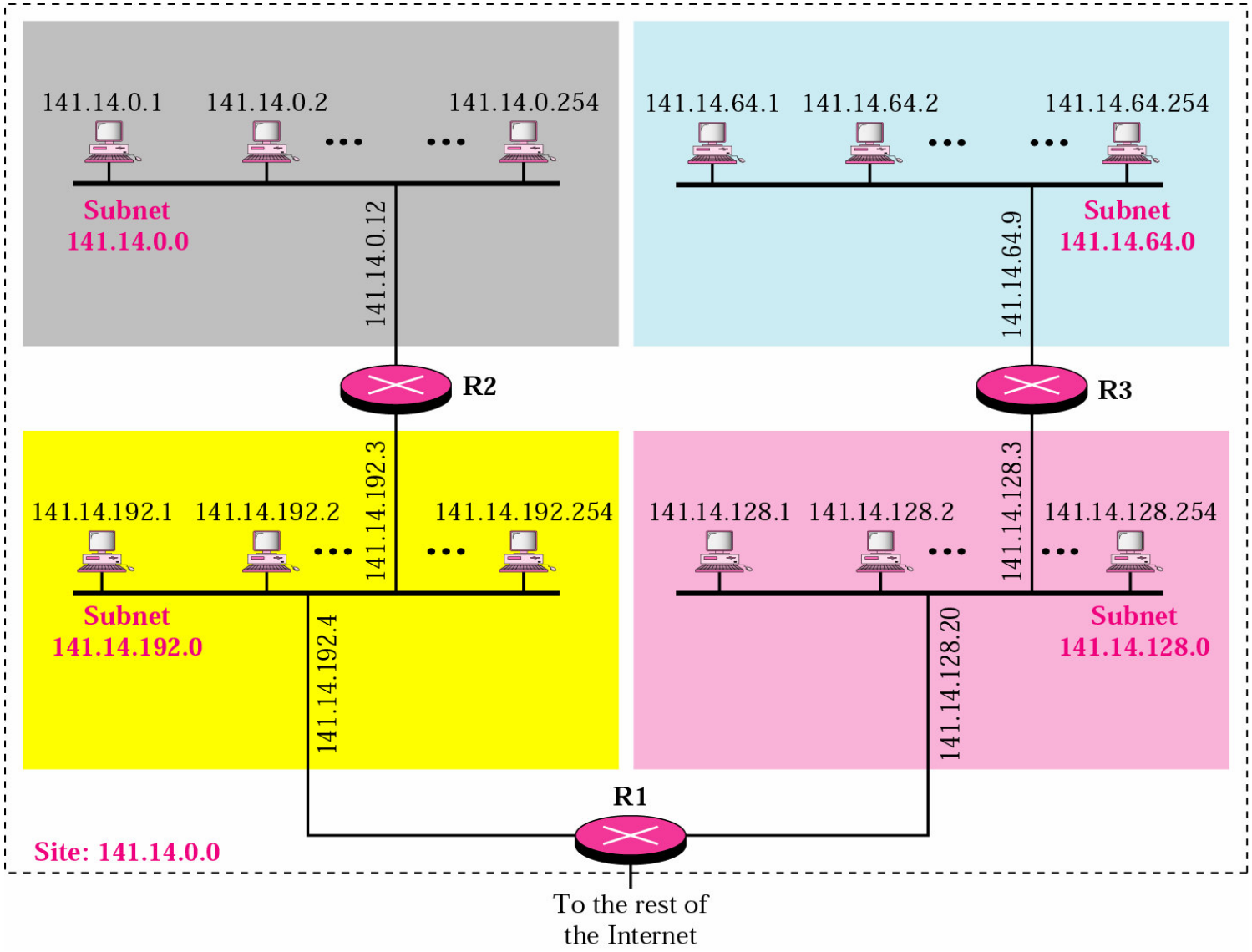
**Note:**

***IP addresses are designed with two levels of hierarchy.***

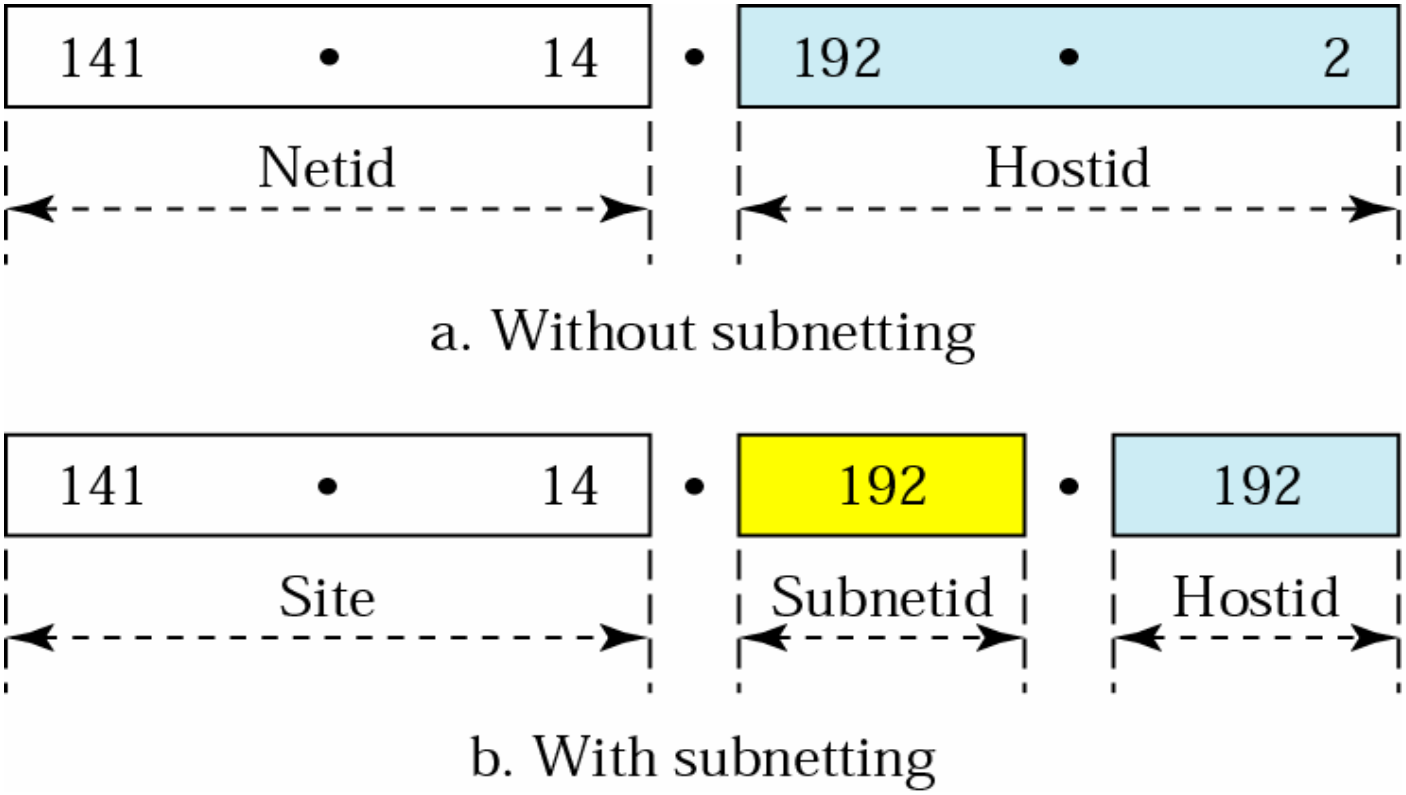
## Figure 19.19 A network with two levels of hierarchy



## Figure 19.20 A network with three levels of hierarchy (subnetted)



## Figure 19.21 Addresses in a network with and without subnetting



## *Table 19.1 Default masks*

| Class | In Binary | | In Dotted-Decimal | Using Slash |
|---|---|---|---|---|
| A | 11111111 00000000 | 00000000 00000000 | 255.0.0.0 | /8 |
| B | 11111111 11111111 | 00000000 00000000 | 255.255.0.0 | /16 |
| C | 11111111 111111111 | 11111111 00000000 | 255.255.255.0 | /24 |

**Note:**

***The network address can be found by applying the default mask to any address in the block (including itself). It retains the netid of the block and sets the hostid to 0s.***

## *Example 8*

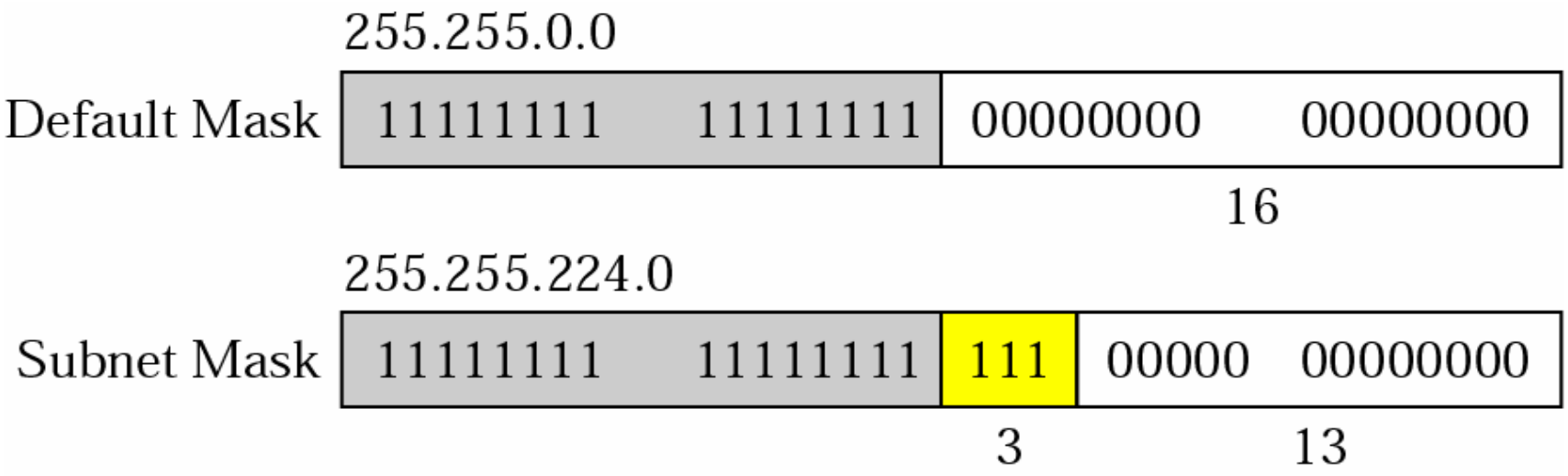
A router outside the organization receives a packet with destination address 190.240.7.91. Show how it finds the network address to route the packet.

## *Solution*

The router follows three steps:

1. The router looks at the first byte of the address to find the class. It is class B.
2. The default mask for class B is 255.255.0.0. The router ANDs this mask with the address to get 190.240.0.0.
3. The router looks in its routing table to find out how to route the packet to this destination. Later, we will see what happens if this destination does not exist.

## Figure 19.23 Subnet mask



## *Example 9*

A router inside the organization receives the same packet with destination address 190.240.33.91. Show how it finds the subnetwork address to route the packet.

## *Solution*

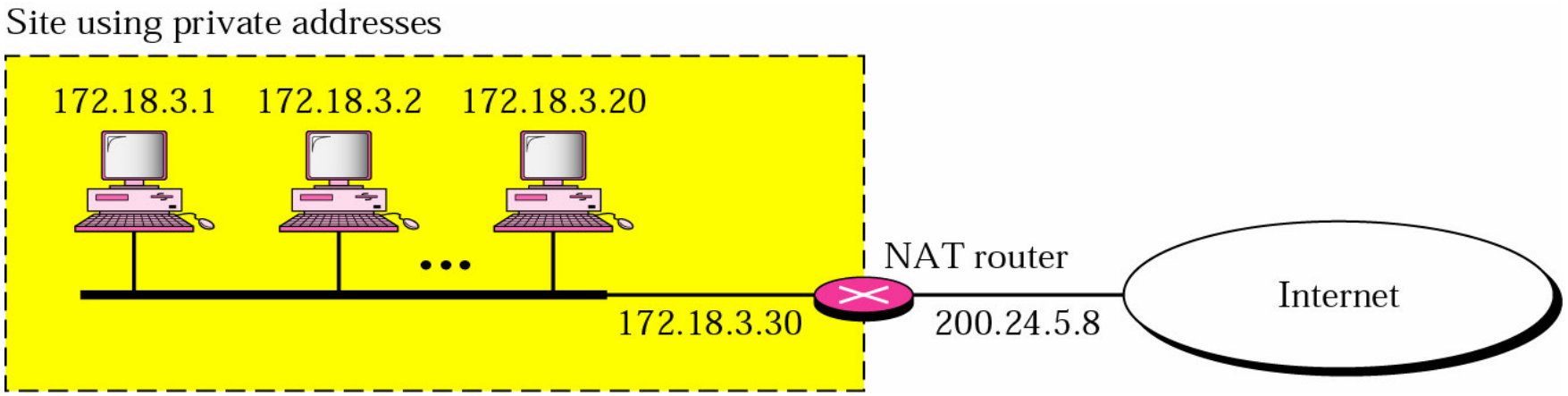
**The router follows three steps:**

1. **The router must know the mask. We assume it is /19, as shown in Figure 19.23.**
2. **The router applies the mask to the address, 190.240.33.91. The subnet address is 190.240.32.0.**
3. **The router looks in its routing table to find how to route the packet to this destination. Later, we will see what happens if this destination does not exist.**

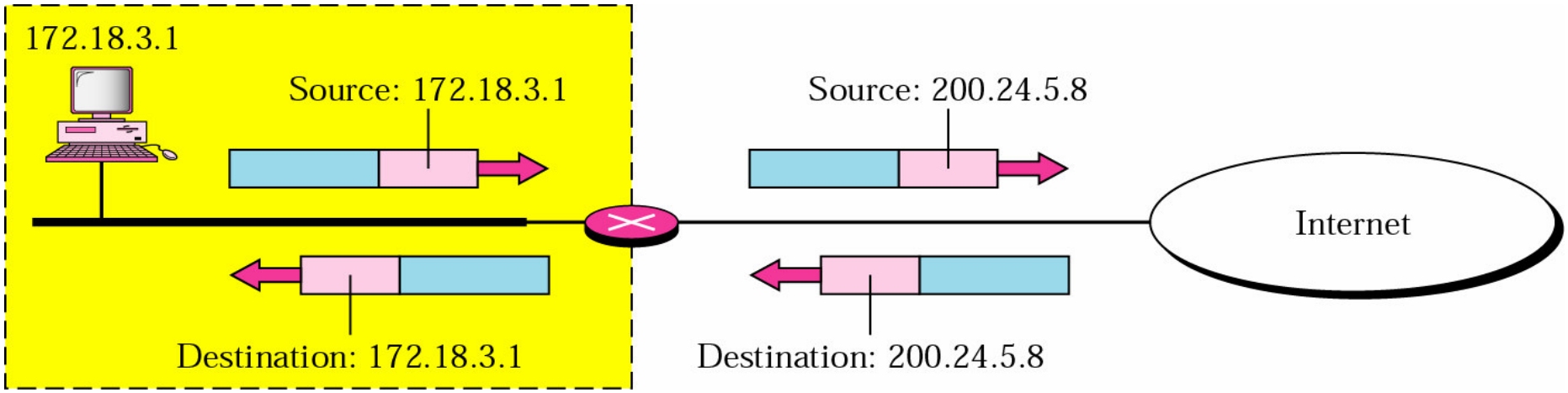
# Subnet Mask Calculation

| | Binary Representation | Dotted Decimal |
|---|---|---|
| **IP address** | 11000000.11100100.00010001 .00111001 | 192.228.17 .57 |
| **Subnet mask** | 11111111.11111111.11111111 .11100000 | 255.255.255 .224 |
| **Bitwise AND of address and mask (resultant network/subnet number)** | 11000000.11100100.00010001 .00100000 | 192.228.17 .32 |
| **Subnet number** | 11000000.11100100.00010001 .001 | 1 |
| **Host number** | 00000000.00000000.00000000 .00011001 | 25 |

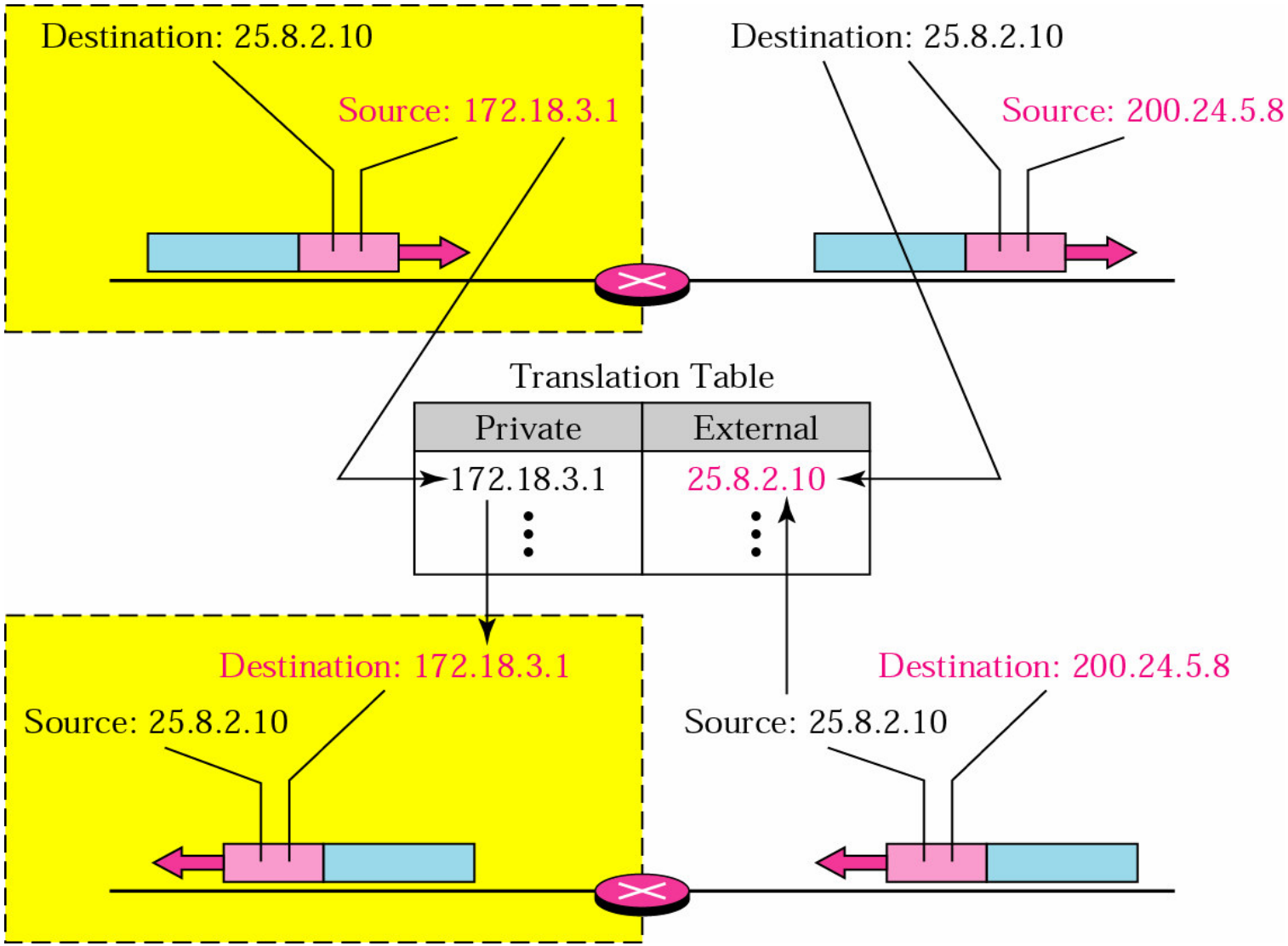
## Figure 19.25 NAT Network Address Translation




## Figure 19.26 Address translation



## Figure 19.27 Translation



## *Table 19.3 Five-column translation table*

| *Private Address* | *Private Port* | *External Address* | *External Port* | *Transport Protocol* |
|---|---|---|---|---|
| 172.18.3.1 | 1400 | 25.8.3.2 | 80 | TCP |
| 172.18.3.2 | 1401 | 25.8.3.2 | 80 | TCP |
| ... | ... | ... | ... | ... |

# Network Address Translation

- Each organization-single *IP* address
- Within organization – each host with IP unique to the orgn., from reserved set of IP addresses

| 3 Reserved ranges |
| --- |
| 10.0.0.0 – 10.255.255.255 (16,777,216 hosts) |
| 172.16.0.0 – 172.31.255.255/12 (1,048,576 hosts) |
| 192.168.0.0 – 192.168.255.255/16 (65,536 hosts) |

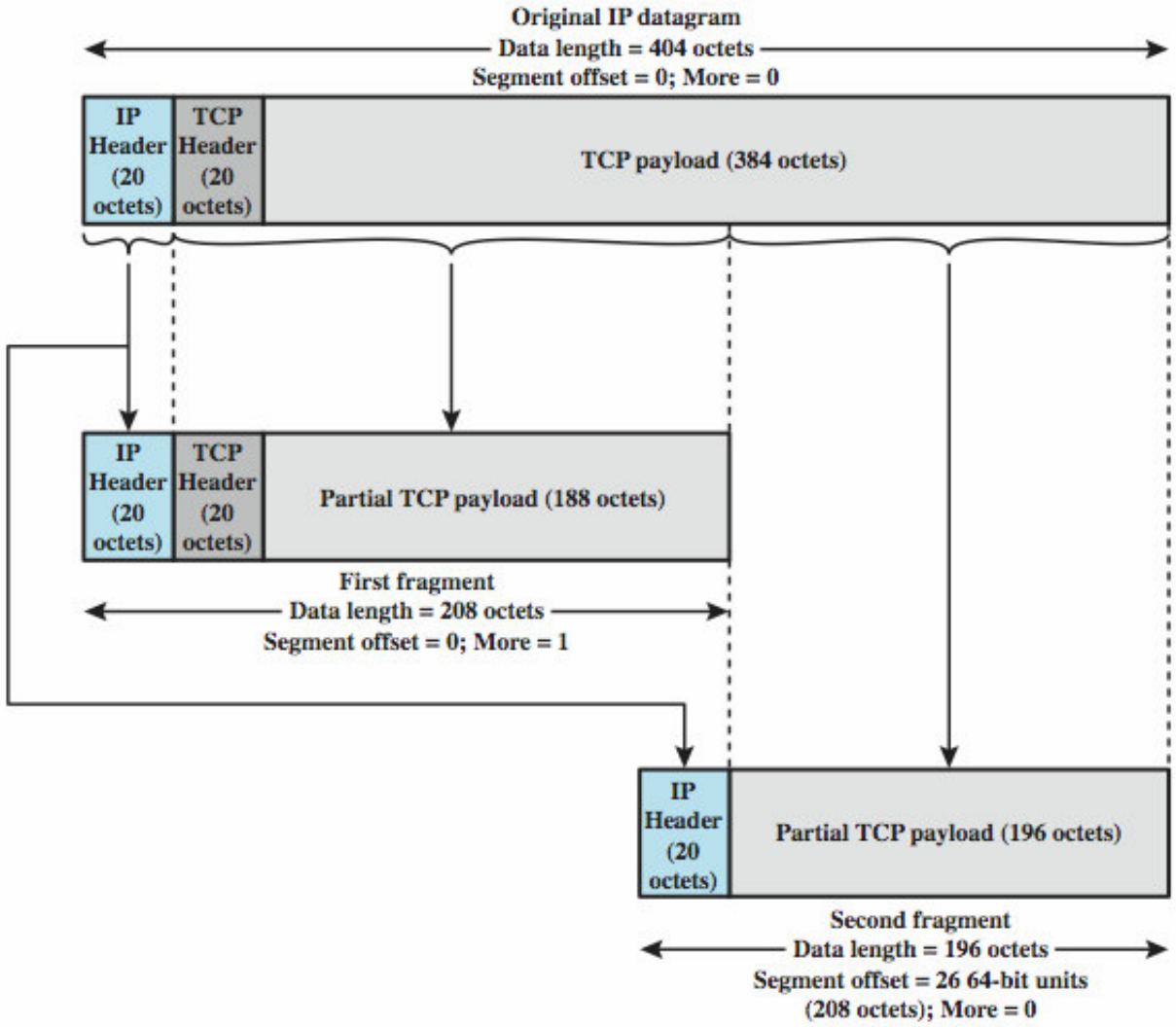
# Fragmentation and Re-assembly

- may have different packet sizes
  - on networks along path used by datagram
- issue of when to re-assemble
  - at destination
    - packets get smaller as data traverses internet
  - intermediate re-assembly
    - need large buffers at routers
    - buffers may fill with fragments
    - all fragments must go through same router

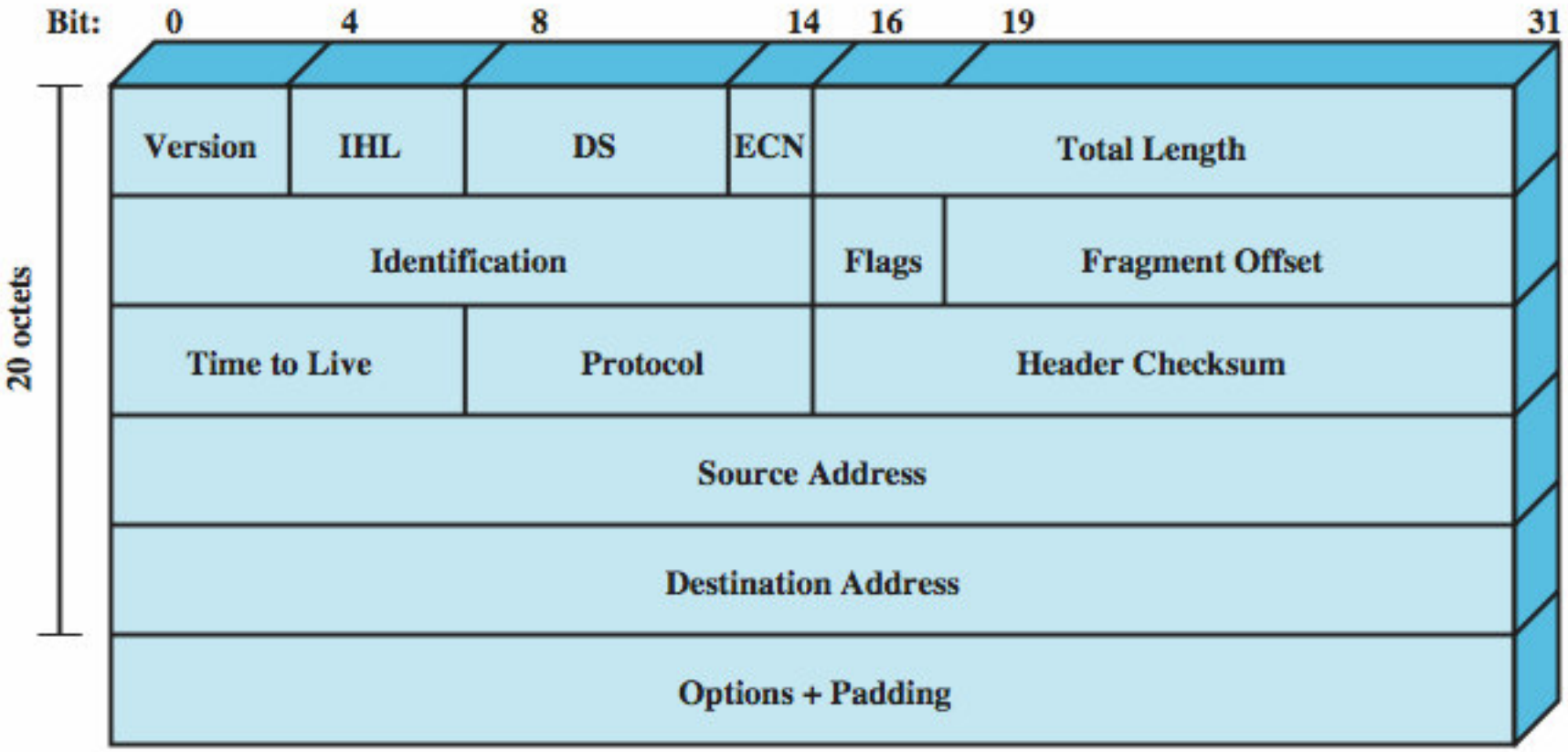
# IP Fragmentation

- IP re-assembles at destination only
- uses fields in header
  - Data Unit Identifier (ID)
    - identifies end system originated datagram
  - Data length
    - length of user data in octets
  - Offset
    - position of fragment of user data in original datagram
    - in multiples of 64 bits (8 octets)
  - *More* flag
    - indicates that this is not the last fragment

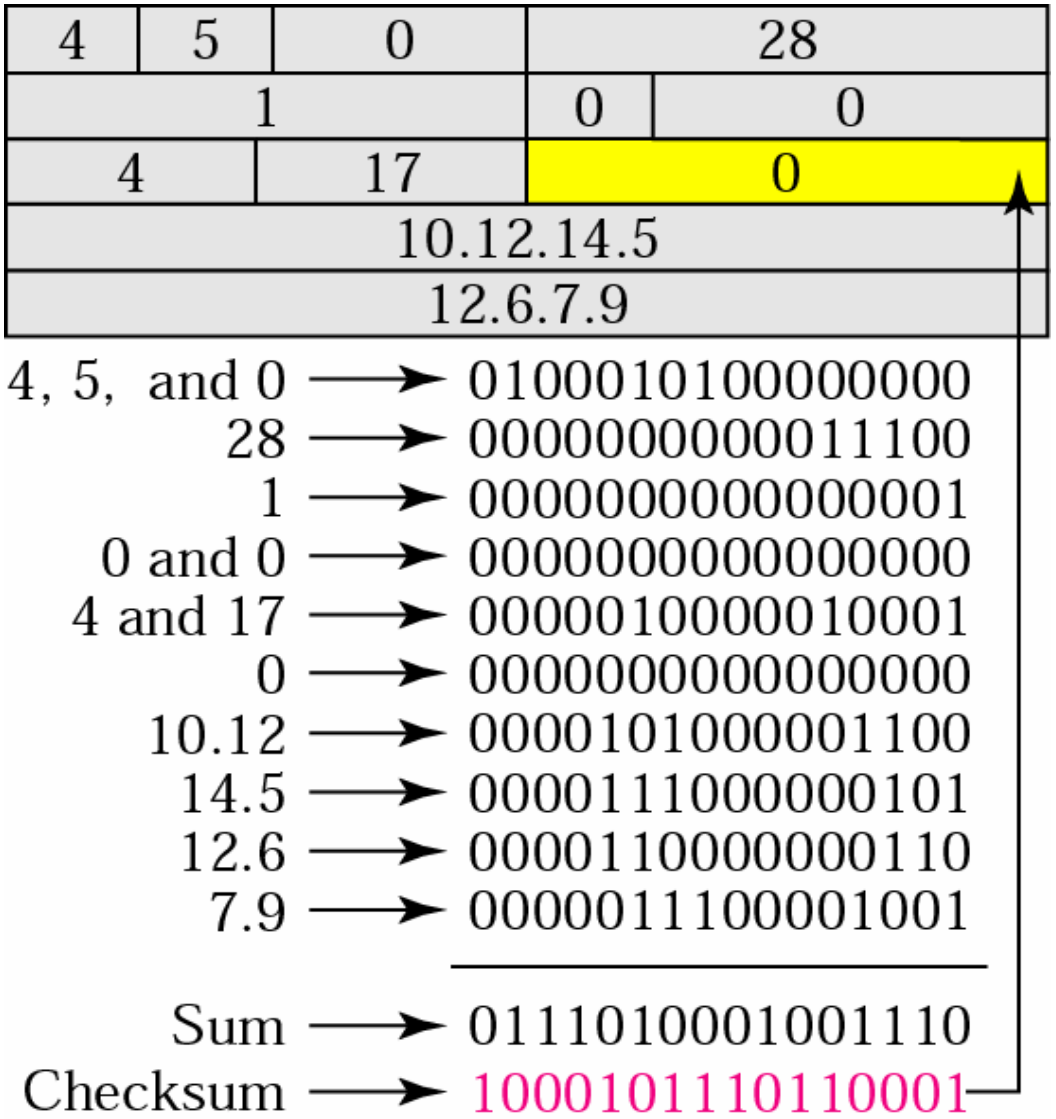
# Fragmentation Example



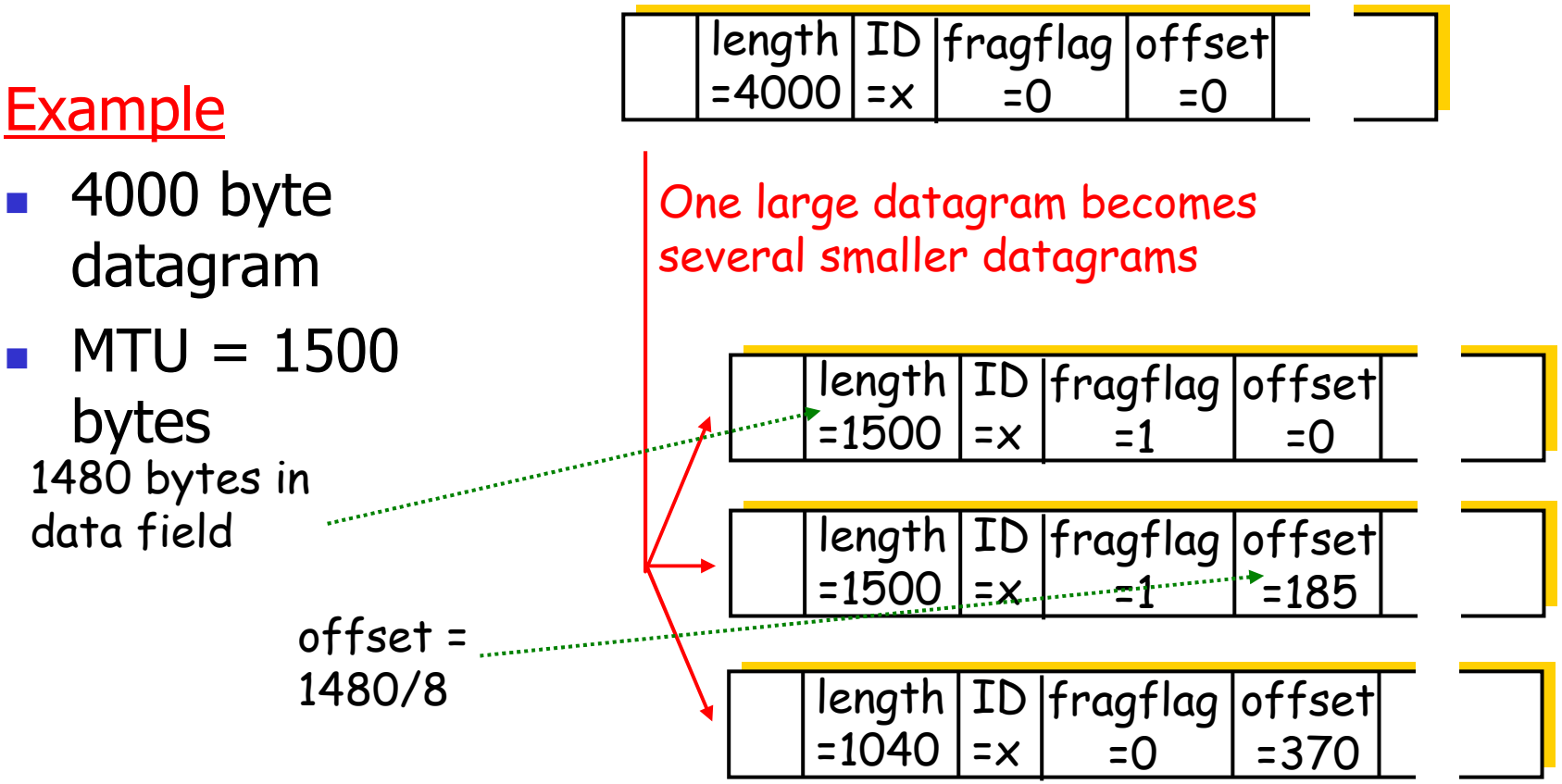
# IPv4 Header



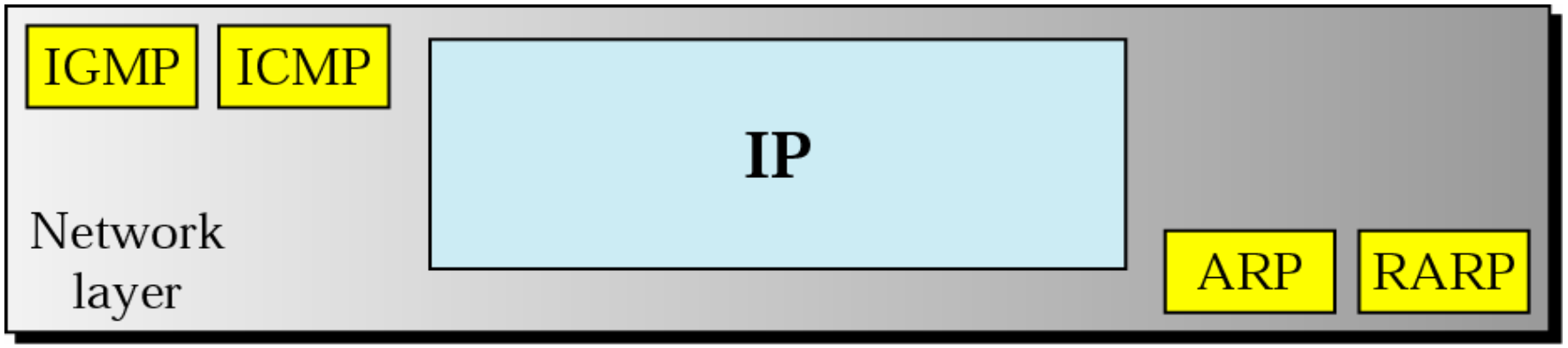
## Figure 20.9 Example of checksum calculation



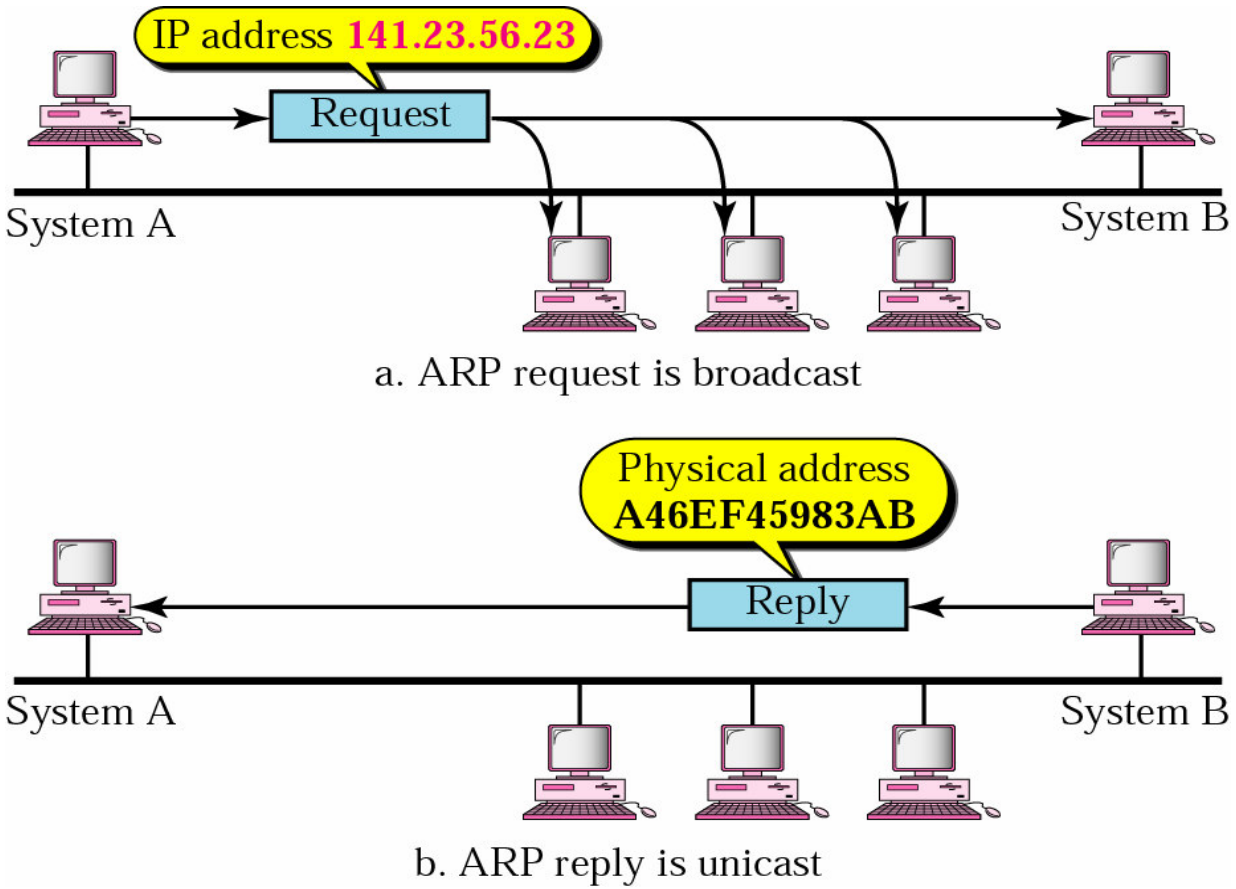
# IP Fragmentation and Reassembly



## Figure 20.1 Protocols at network layer

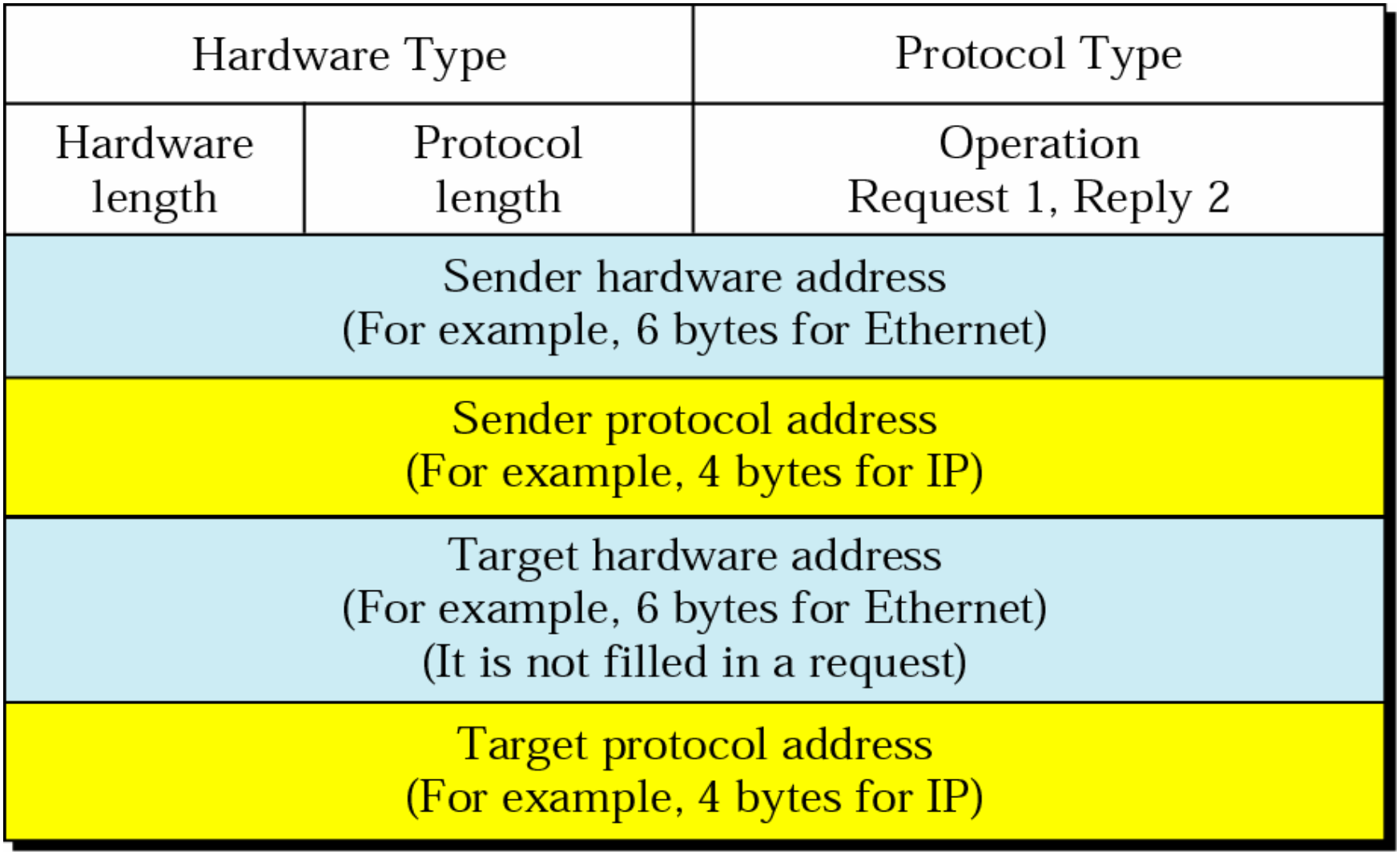


## Figure 20.2 ARP operation

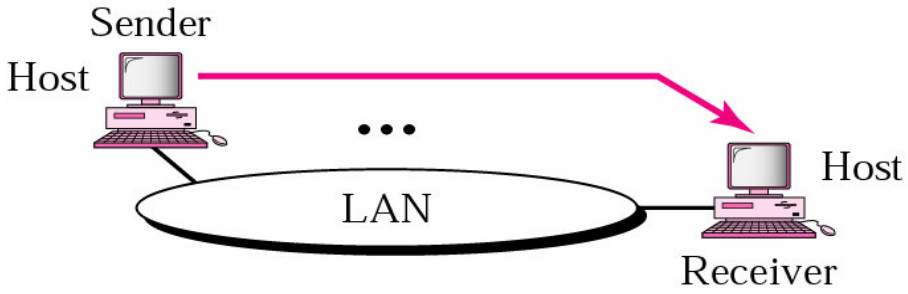



***An ARP request is broadcast; an ARP reply is unicast.***

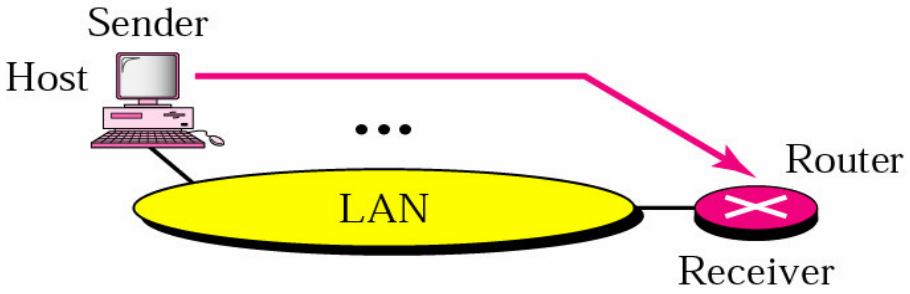
## Figure 20.3 ARP packet



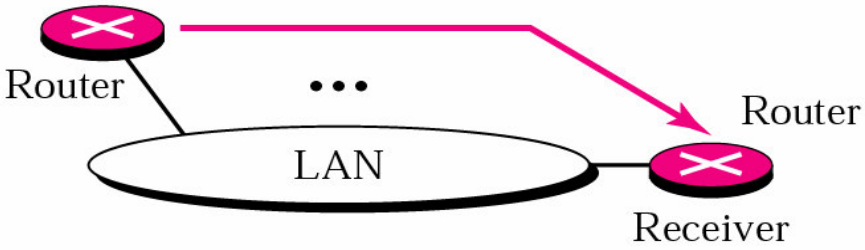
## Figure 20.5 Four cases using ARP



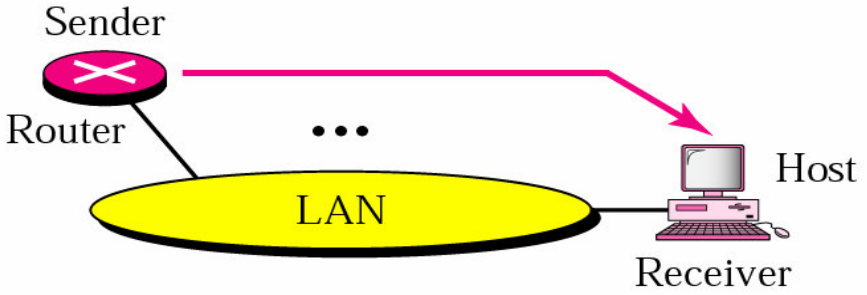

Case 1. A host has a packet to send to another host on the same network.




Case 2. A host wants to send a packet to another host on another network.
It must first be delivered to the appropriate router.




Case 3. A router receives a packet to be sent to a host on another network.
It must first be delivered to the appropriate router.



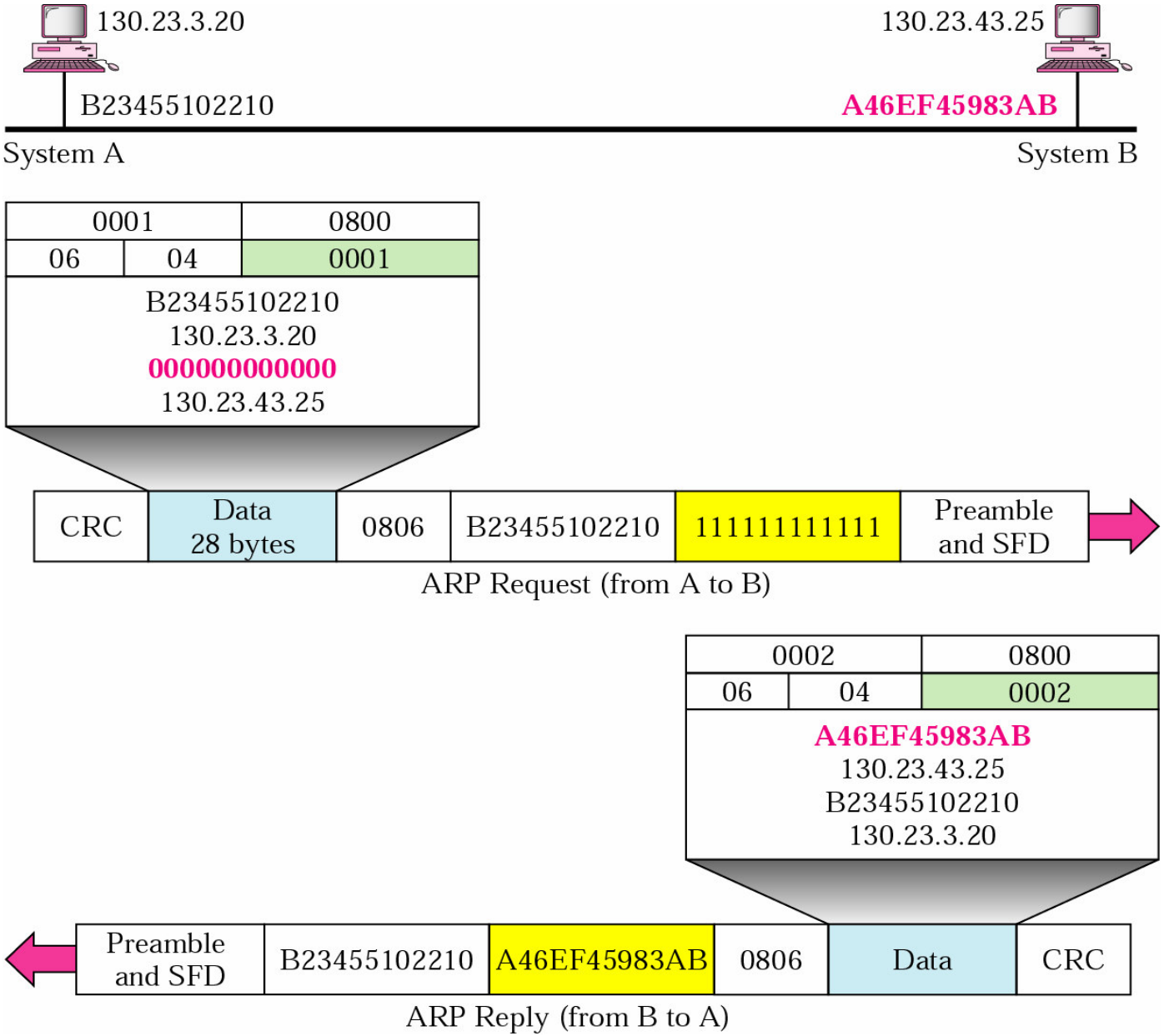

Case 4. A router receives a packet to be sent to a host on the same network.

## *Example 1*

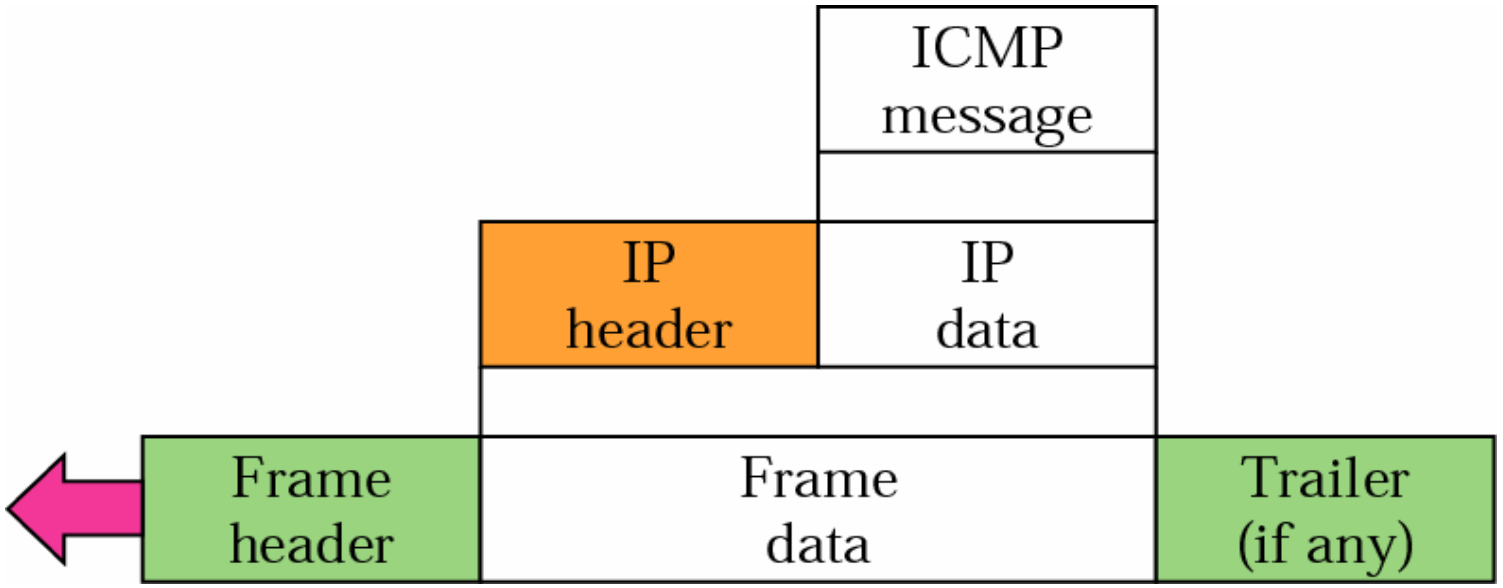
A host with IP address 130.23.3.20 and physical address B23455102210 has a packet to send to another host with IP address 130.23.43.25 and physical address A46EF45983AB. The two hosts are on the same Ethernet network. Show the ARP request and reply packets encapsulated in Ethernet frames.

## *Solution*

## Figure 20.6 Example 1

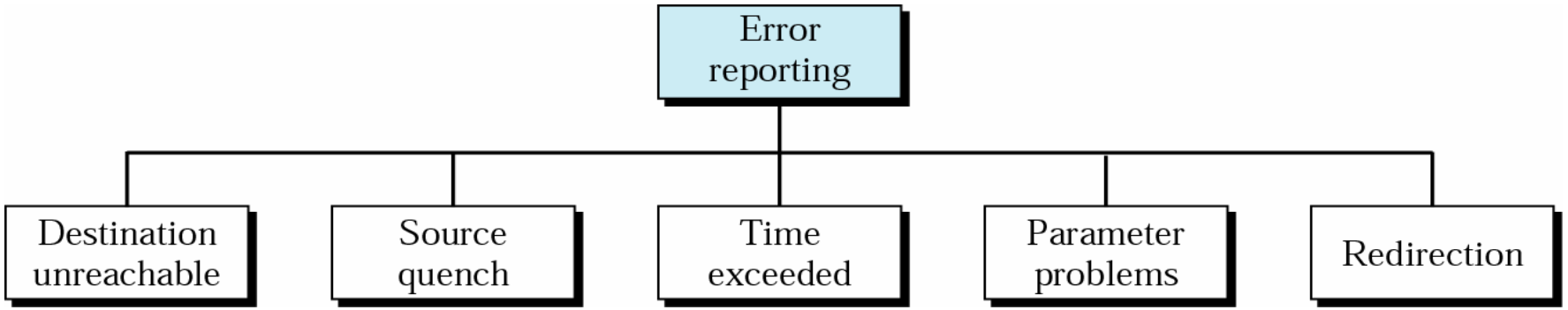


# 20.3 ICMP Internet Control Message Protocol

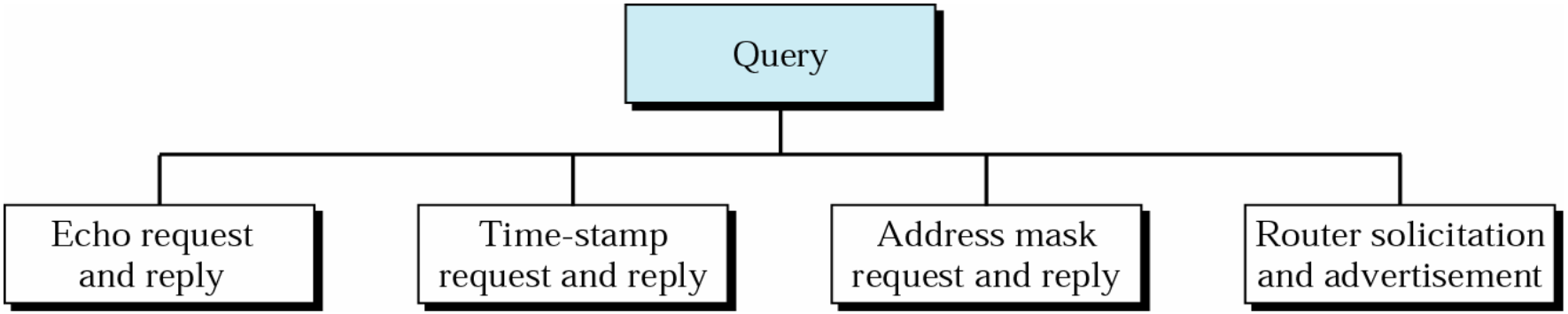



***ICMP always reports error messages to the original source.***

## Figure 20.13 Error-reporting messages



## Figure 20.14 Query messages



# 20.4 IPv6

*IPv6 Addresses*

*Categories of Addresses*

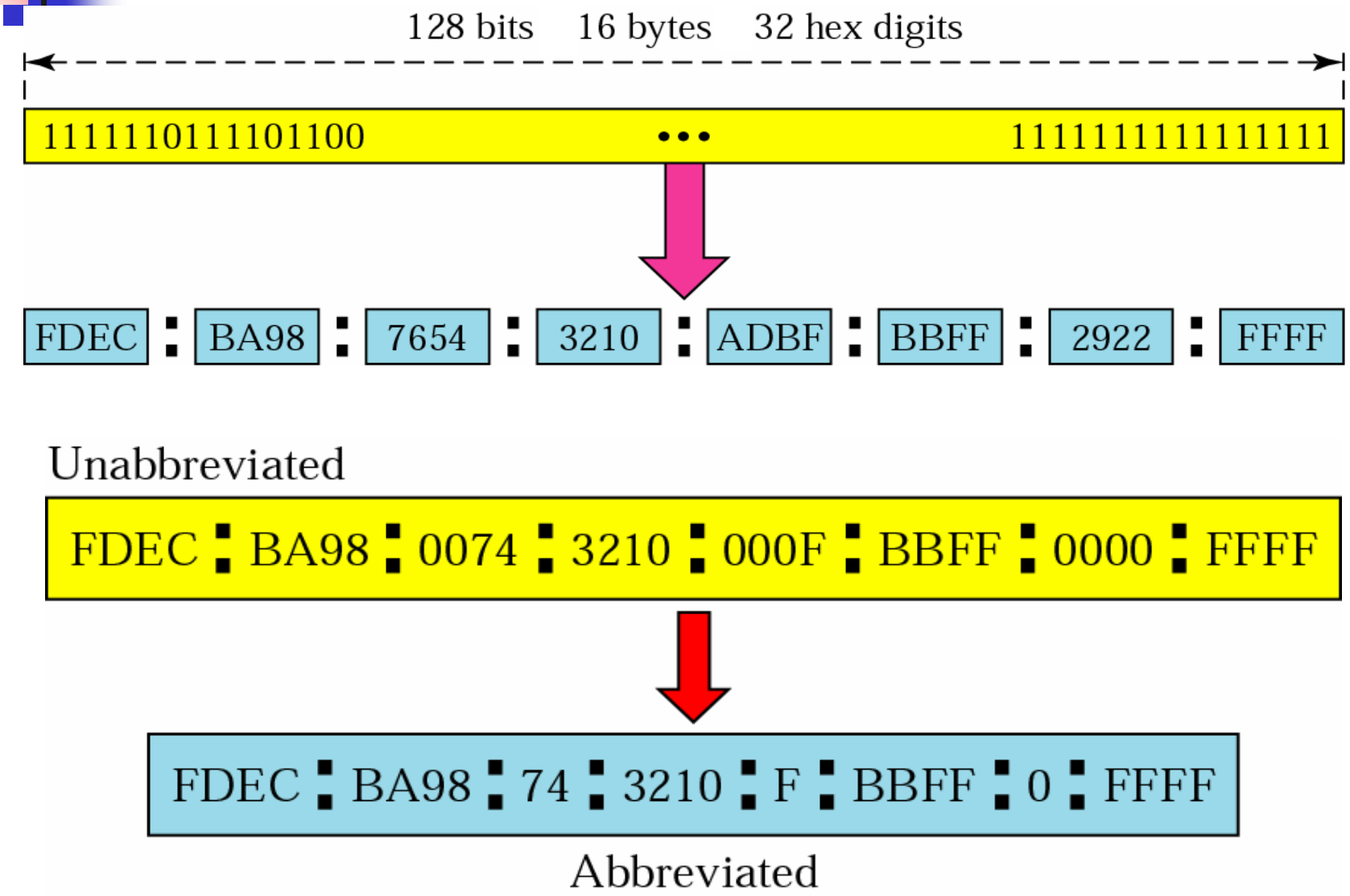
*IPv6 Packet Format*

*Fragmentation*

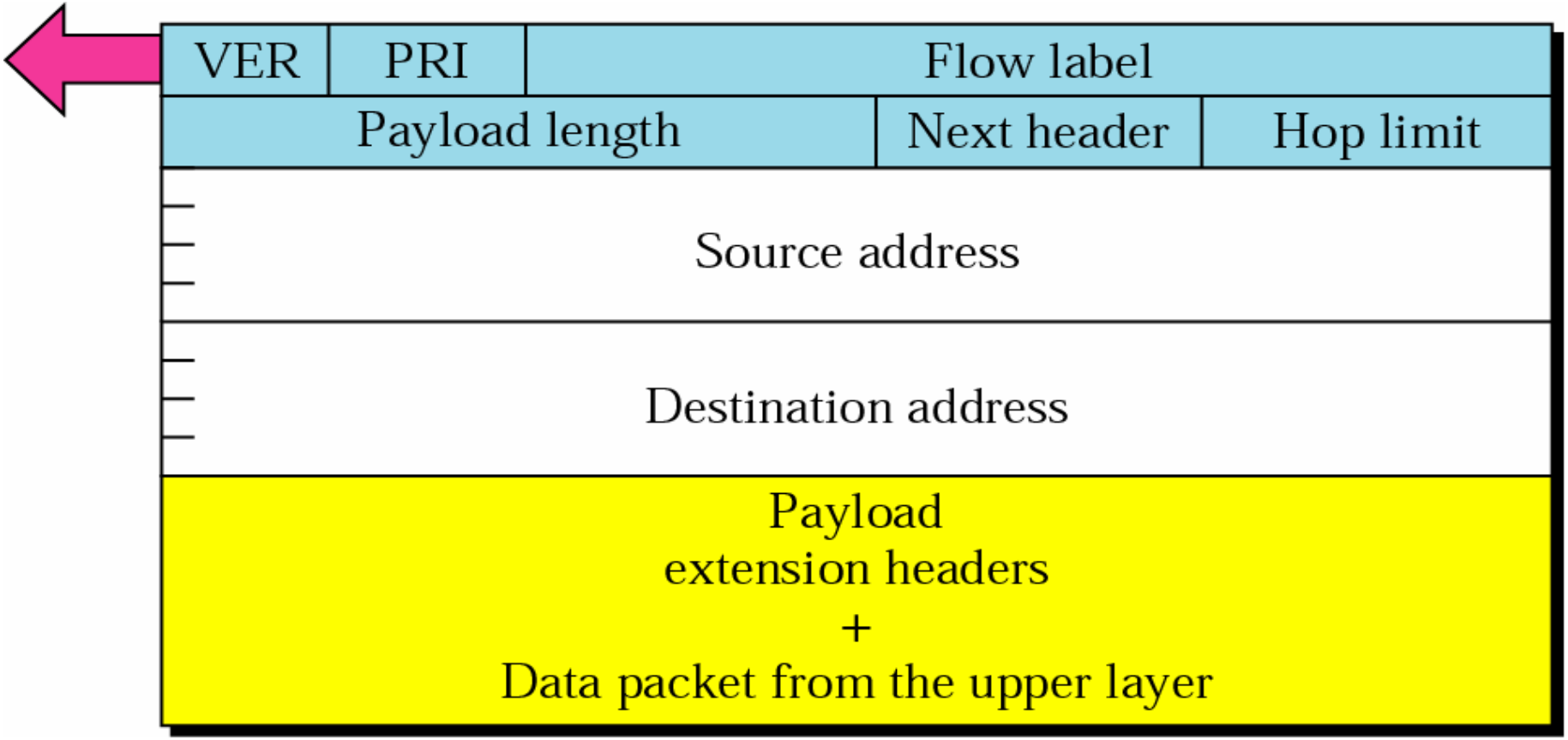
*ICMPv6*

*Transition*

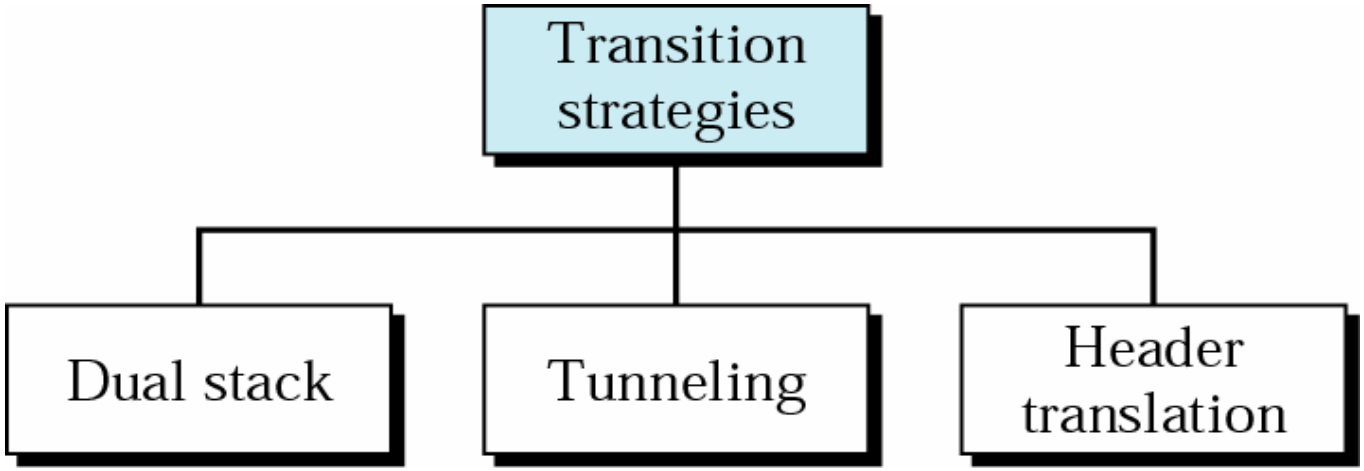
## Figure 20.15 IPv6 address



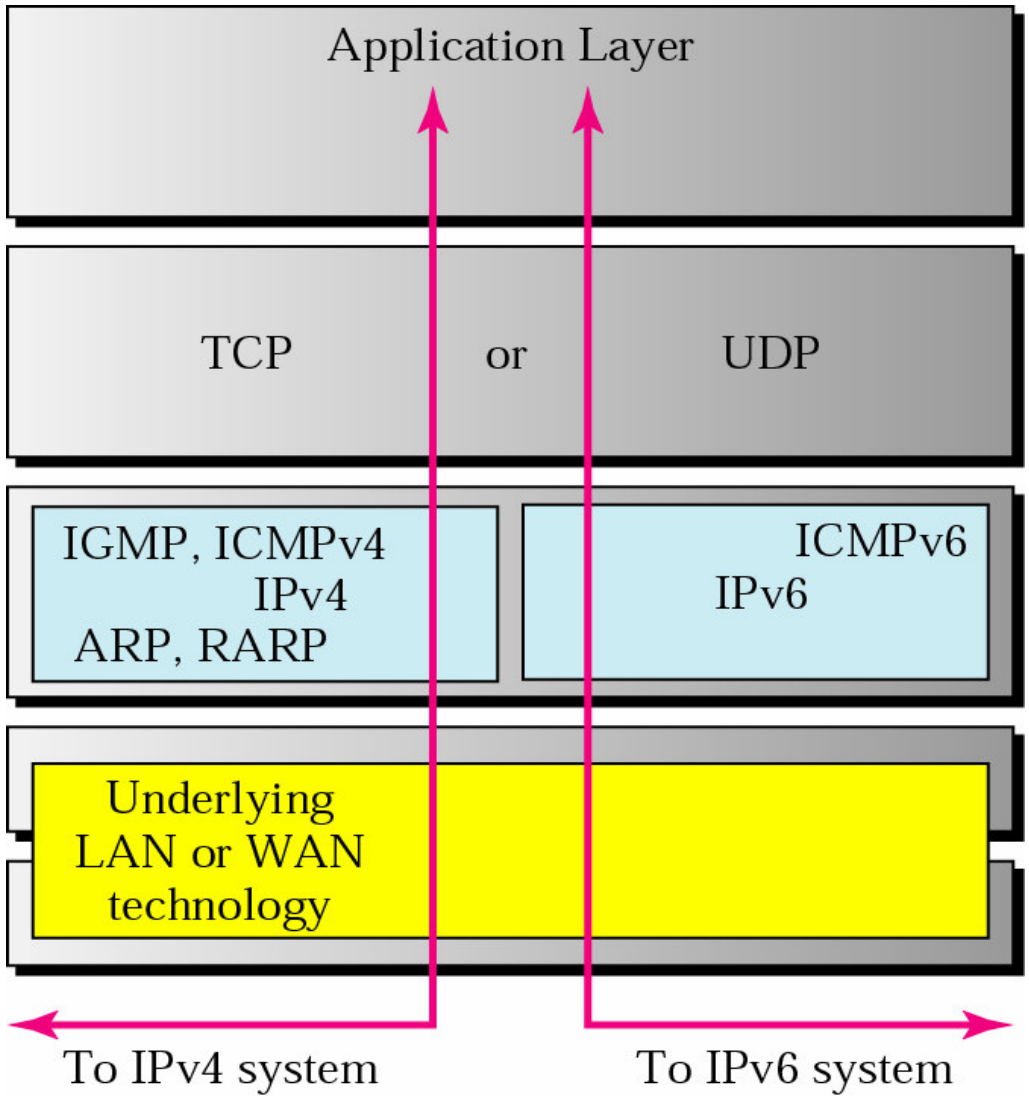
## Figure 20.19 Format of an IPv6 datagram



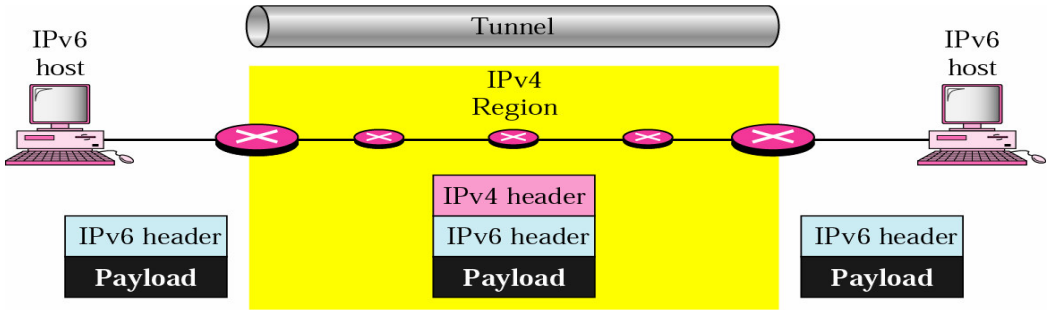
## Figure 20.21 Three transition strategies



## Figure 20.22 Three transition strategies

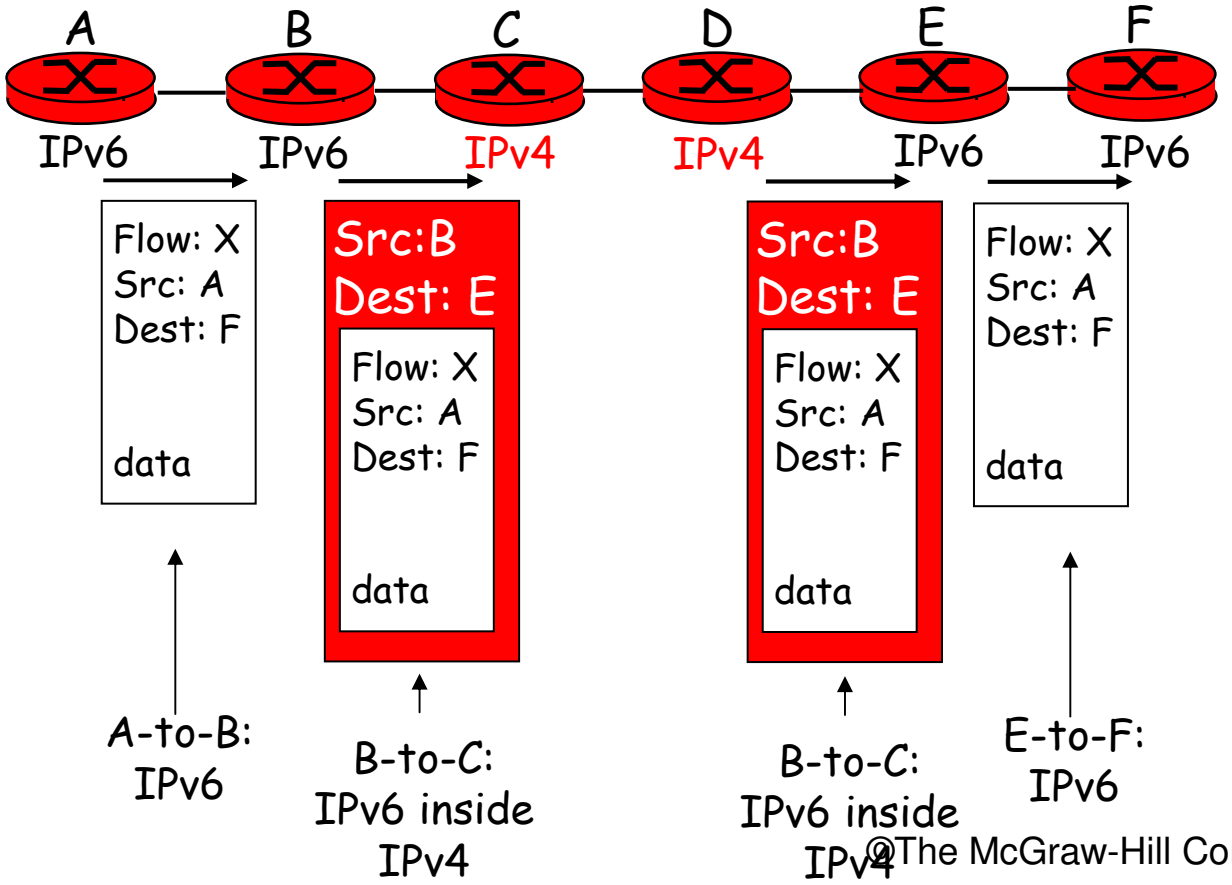


# Tunneling




Logical view:


Physical view:



## Figure 20.24 Header translation