Lecture # 22-32

Network Layer

By: Sanjeev Patel
Asst. Professor, CSE& IT Department
JIIT, Noida Sector-128

Outline

- IP Addressing: Sub-netting, Super-netting, CIDR
- IP Datagram (IPv4) and IPv6
- NAT
- Routing protocol: DV routing and LS routing
- Hierarchical Routing
- Internet Routing: RIP, OSPF and BGP
- Broadcast Routing and Multicast routing
- ICMP, ARP, RARP, and DHCP

Note

- ***An IP address is a 32-bit address.***
- ***The IP addresses are unique.***

Note

The address space of IPv4 is

$$2^{32}$$

or

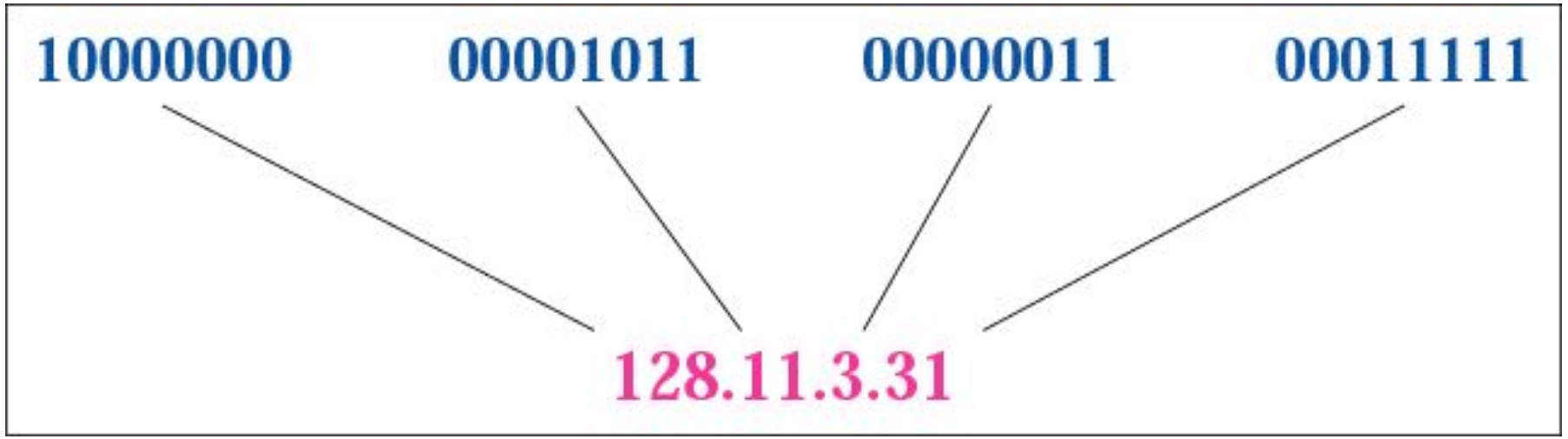
4,294,967,296.

Binary Notation

01110101 10010101 00011101 11101010

Figure 4-1

Dotted-decimal notation



Hexadecimal Notation

0111 0101 1001 0101 0001 1101 1110 1010

75 95 1D EA

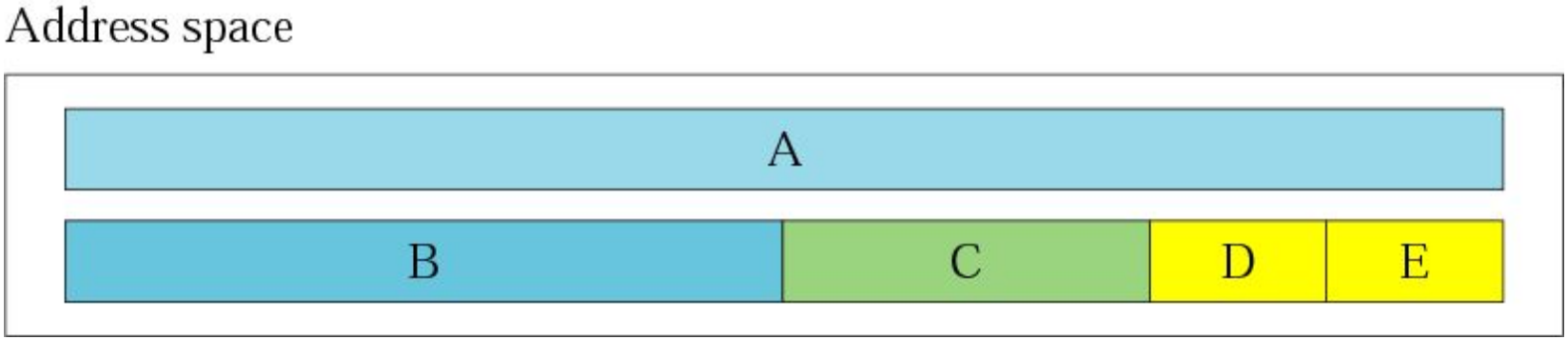
0x75951DEA

4.2

CLASSFUL ADDRESSING

Figure 4-2

Occupation of the address space



Note

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

Figure 4-3

Finding the class in binary notation

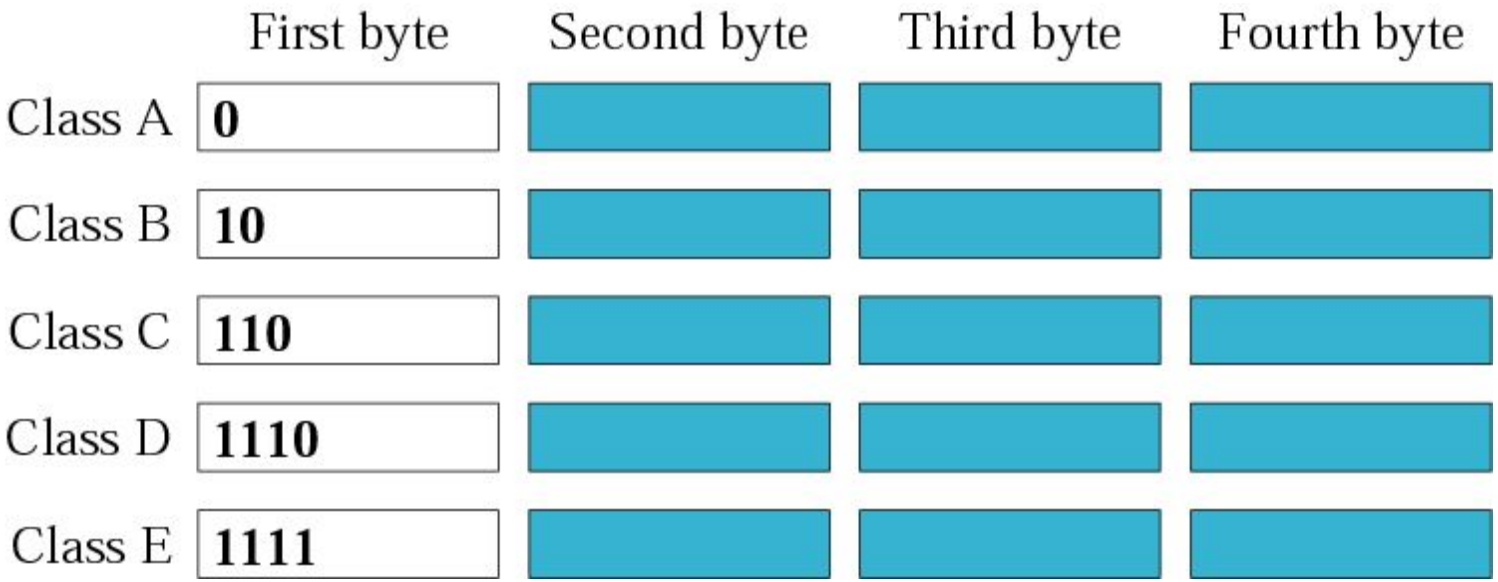
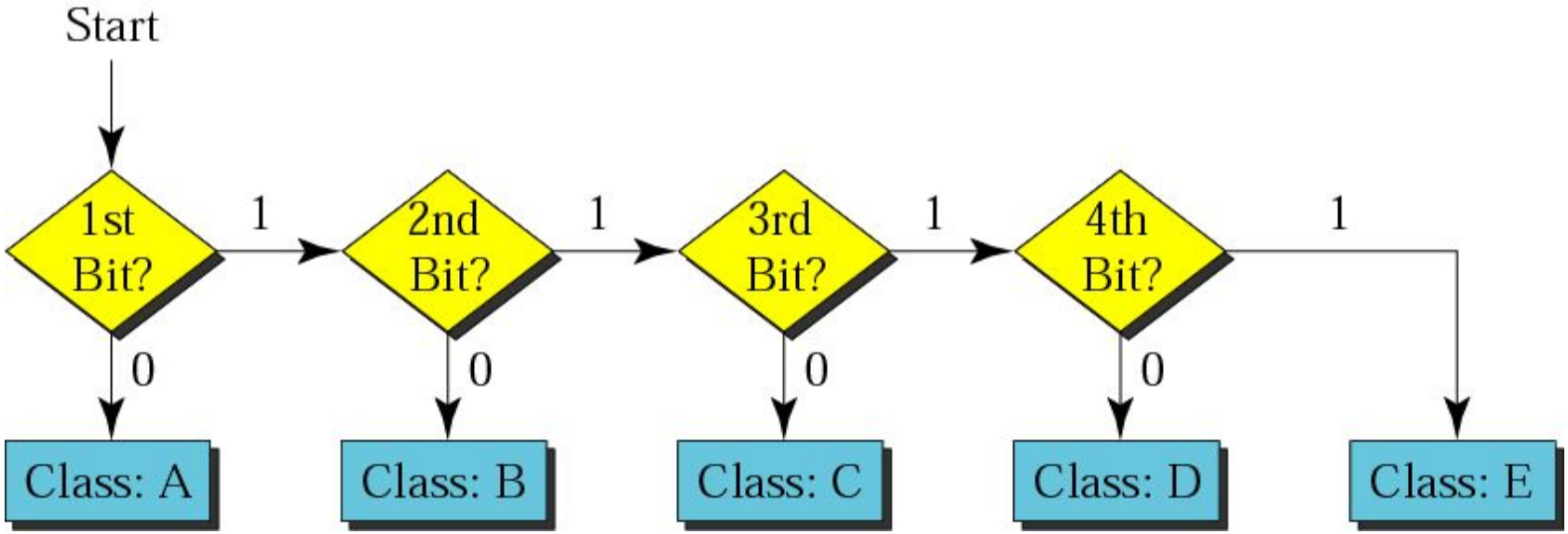


Figure 4-4

Finding the address class



Example 5

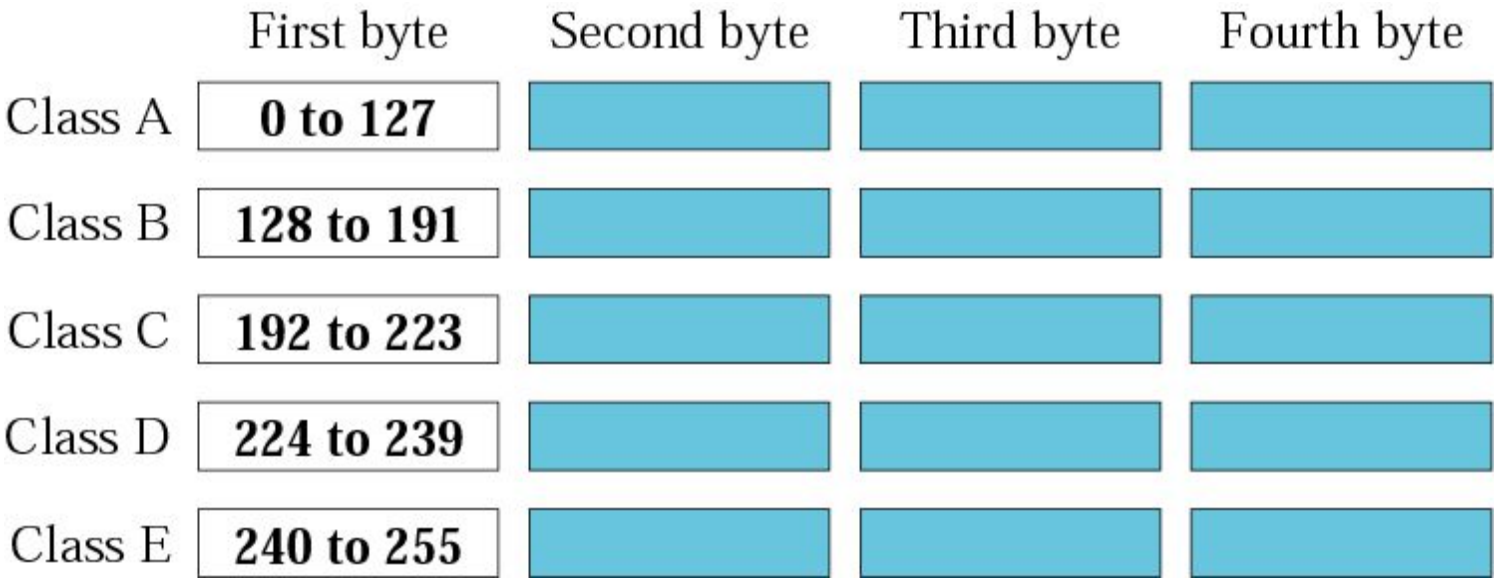
How can we prove that we have 2,147,483,648 addresses in class A?

Solution

In class A, only 1 bit defines the class. The remaining 31 bits are available for the address. With 31 bits, we can have 2^{31} or 2,147,483,648 addresses.

Figure 4-5

Finding the class in decimal notation



Example 7

Find the class of the address:

227.12.14.87

Solution

The first byte is 227 (between 224 and 239); the class is D.

Figure 4-6

Netid and hostid

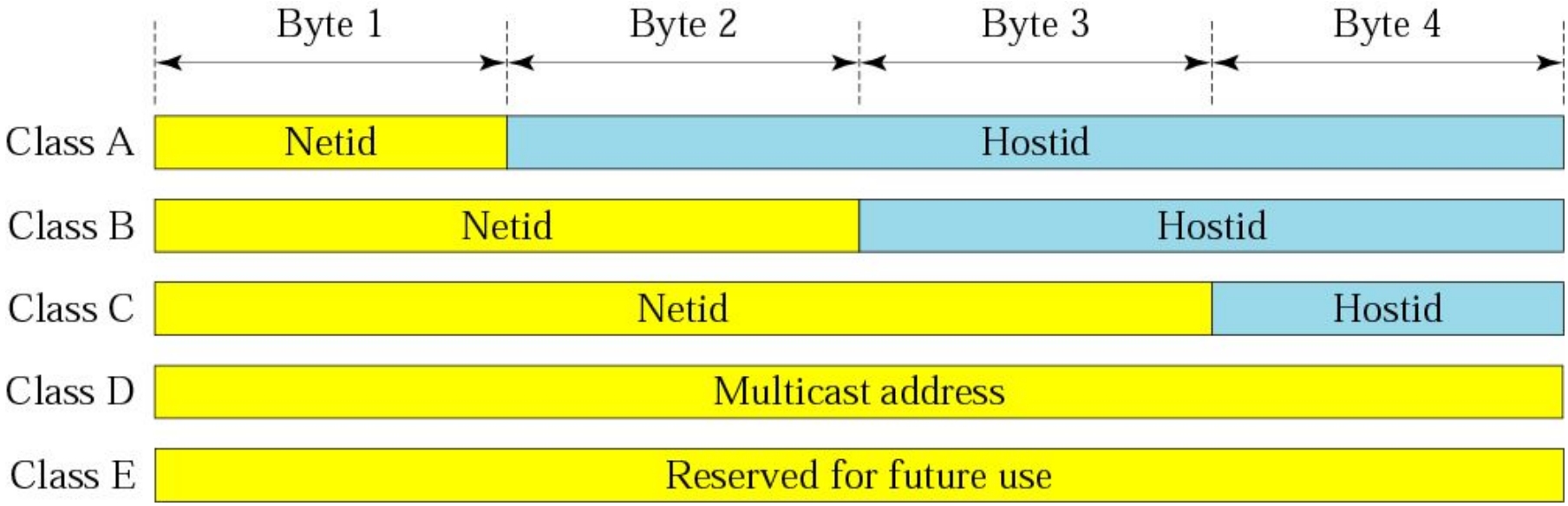
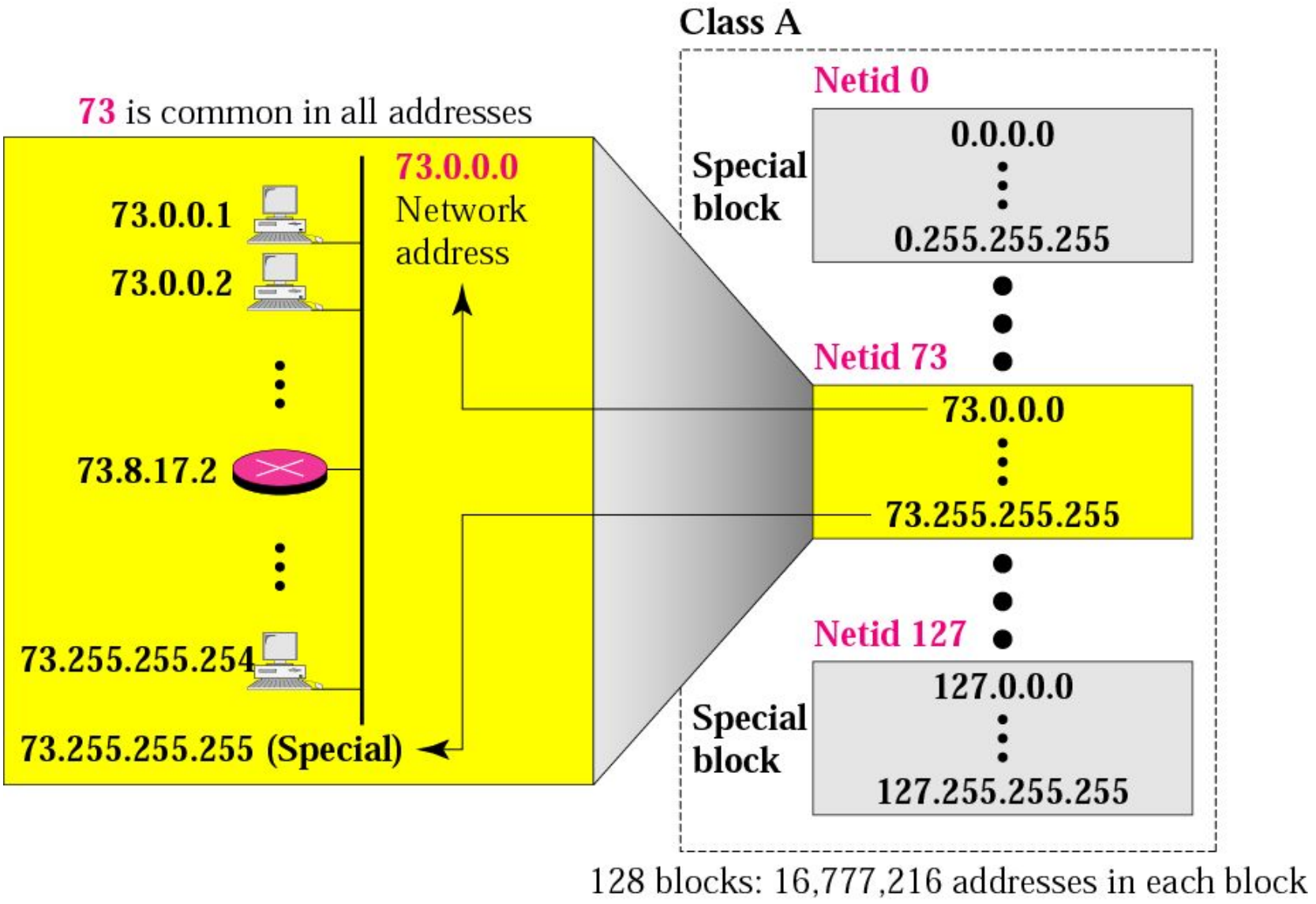


Figure 4-7

Blocks in class A

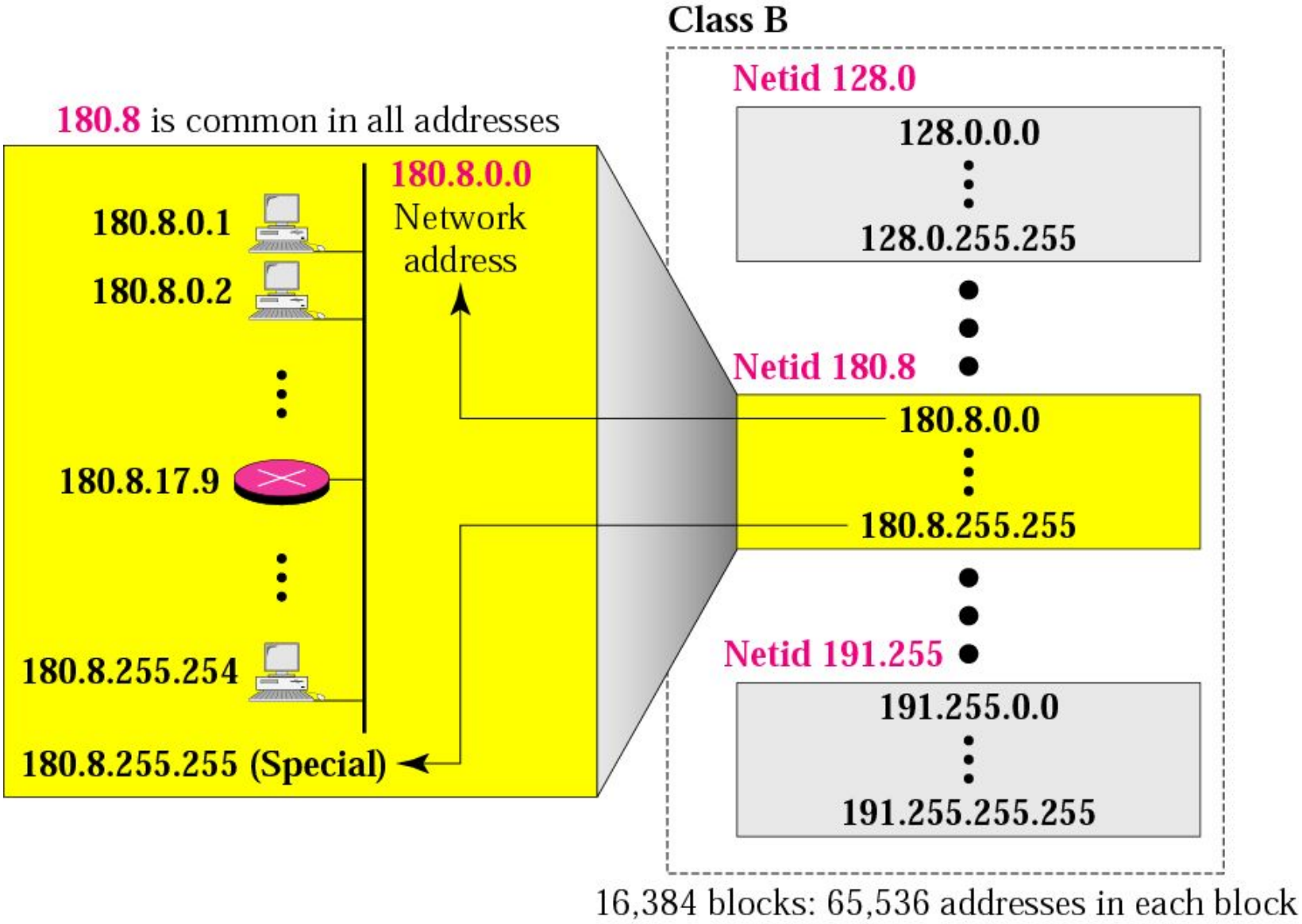


Note

Millions of class A addresses are wasted.

Figure 4-8

Blocks in class B

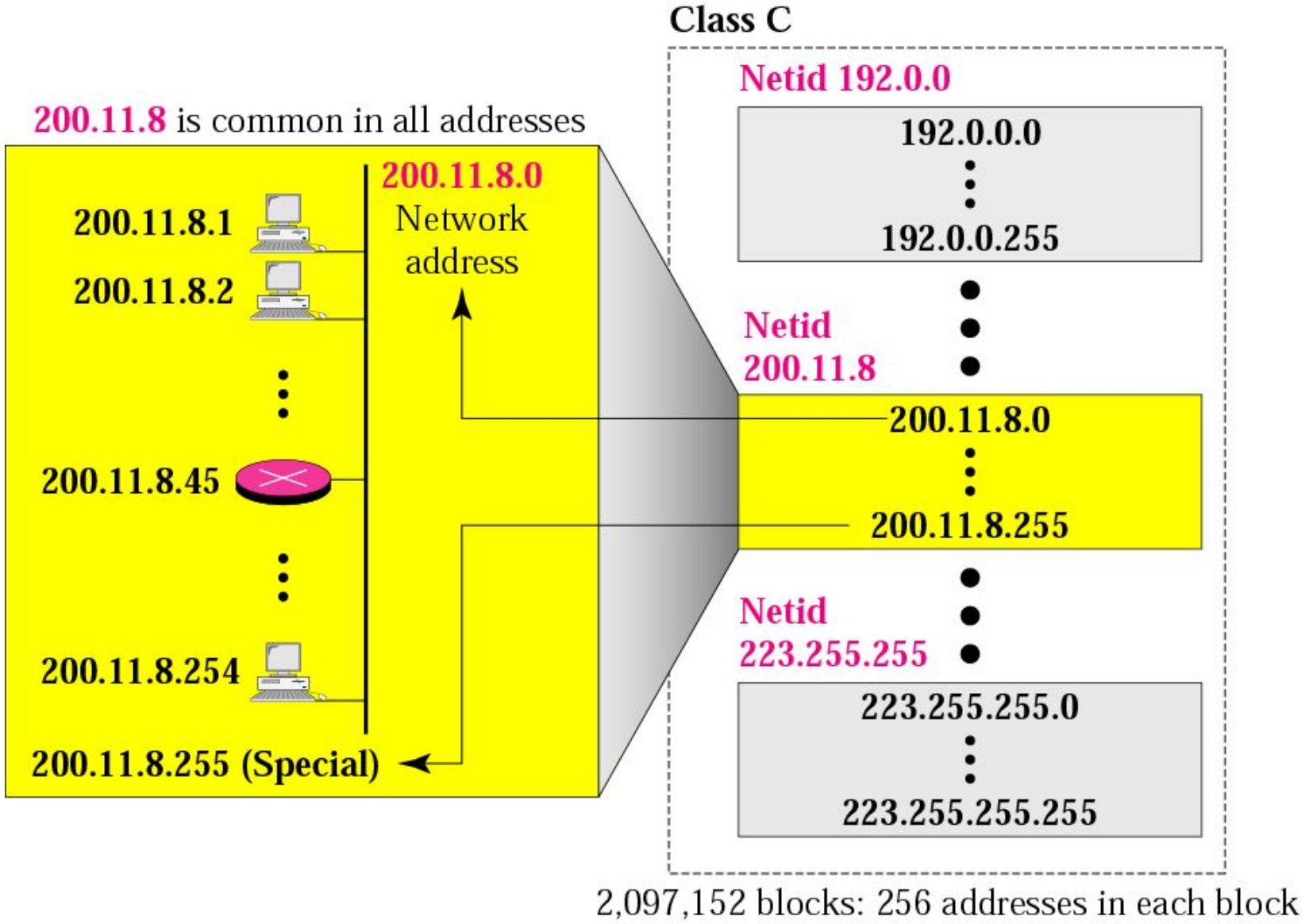


Note

Many class B addresses are wasted.

Figure 4-9

Blocks in class C



Note

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

Note

Class D addresses are used for multicasting; there is only one block in this class.

Note

Class E addresses are reserved for special purposes; most of the block is wasted.

Network Addresses

The network address is the first address.

The network address defines the network to the rest of the Internet.

Given the network address, we can find the class of the address, the block, and the range of the addresses in the block

Note

In classful addressing, the network address (the first address in the block) is the one that is assigned to the organization.

Example 9

Given the network address 17.0.0.0, find the class, the block, and the range of the addresses.

Solution

The class is A because the first byte is between 0 and 127. The block has a netid of 17. The addresses range from 17.0.0.0 to 17.255.255.255.

Example 10

Given the network address 132.21.0.0, find the class, the block, and the range of the addresses.

Solution

The class is B because the first byte is between 128 and 191. The block has a netid of 132.21. The addresses range from 132.21.0.0 to 132.21.255.255.

Example 11

Given the network address 220.34.76.0, find the class, the block, and the range of the addresses.

Solution

The class is C because the first byte is between 192 and 223. The block has a netid of 220.34.76. The addresses range from 220.34.76.0 to 220.34.76.255.

Mask

A mask is a 32-bit binary number that gives the first address in the block (the network address) when bitwise ANDed with an address in the block.

Figure 4-10

Masking concept

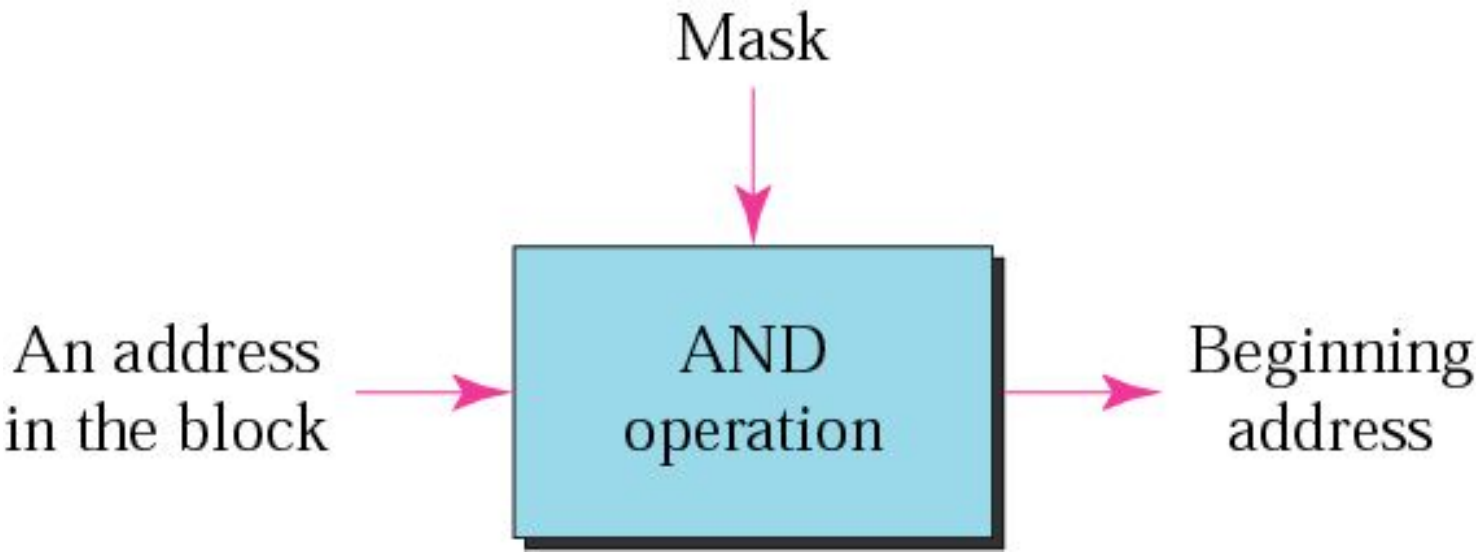
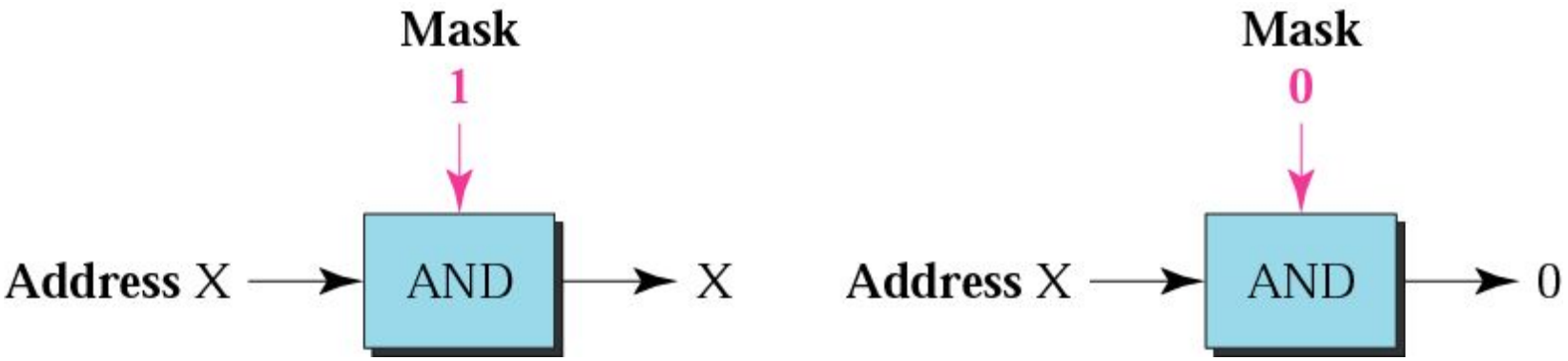


Figure 4-11

AND operation



Note

The network address is the beginning address of each block. It can be found by applying the default mask to any of the addresses in the block (including itself). It retains the netid of the block and sets the hostid to zero.

Example 12

Given the address 23.56.7.91 and the default class A mask, find the beginning address (network address).

Solution

The default mask is 255.0.0.0, which means that only the first byte is preserved and the other 3 bytes are set to 0s. The network address is 23.0.0.0.

Example 13

Given the address 132.6.17.85 and the default class B mask, find the beginning address (network address).

Solution

The default mask is 255.255.0.0, which means that the first 2 bytes are preserved and the other 2 bytes are set to 0s. The network address is 132.6.0.0.

Example 14

Given the address 201.180.56.5 and the class C default mask, find the beginning address (network address).

Solution

The default mask is 255.255.255.0, which means that the first 3 bytes are preserved and the last byte is set to 0. The network address is 201.180.56.0.

Note

We must not apply the default mask of one class to an address belonging to another class.

Private Addresses

A number of blocks in each class are assigned for private use. They are not recognized globally. These blocks are depicted as follows:

Class A: 10.0.0(Net id) and no of Blocks is 1

Class B: 172.16-172.31 and 16

Class C: 192.168.0-192.168.255 and 256 blocks

Unicast, Multicast, and Broadcast Addresses

Unicast communication is *one-to-one*.

Multicast communication is *one-to-many*.

Broadcast communication is *one-to-all*.

Note

IP addresses are designed with two levels of hierarchy.

Figure 5-1

A network with two levels of hierarchy (not subnetted)

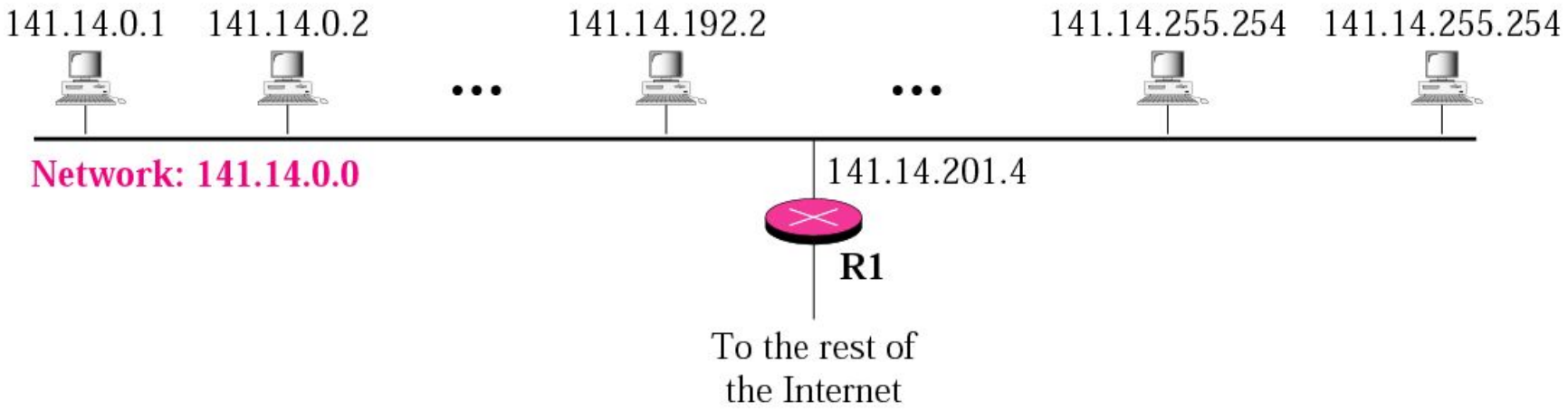


Figure 5-2

A network with three levels of hierarchy (subnetted)

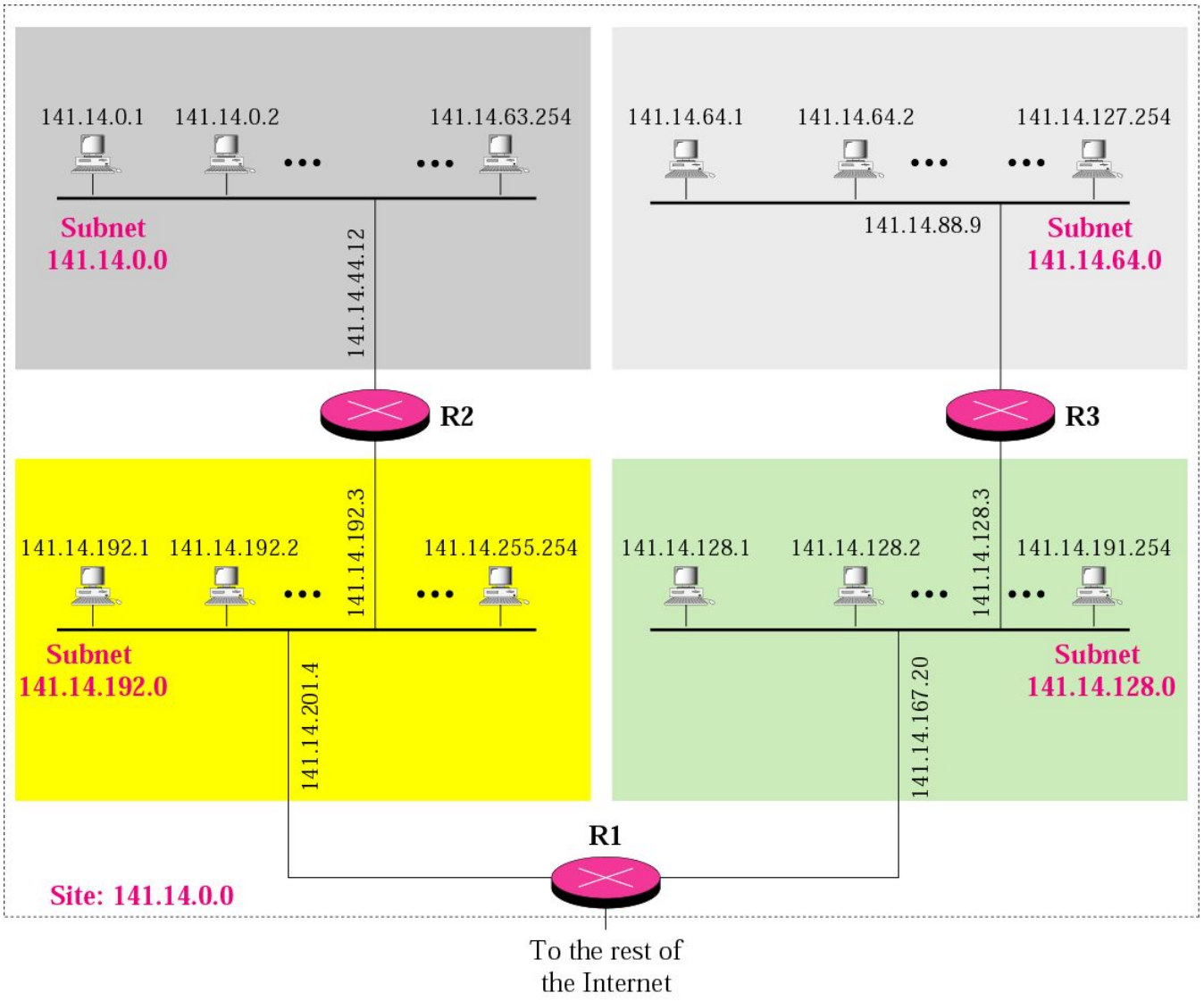


Figure 5-3

Addresses in a network with and without subnetting

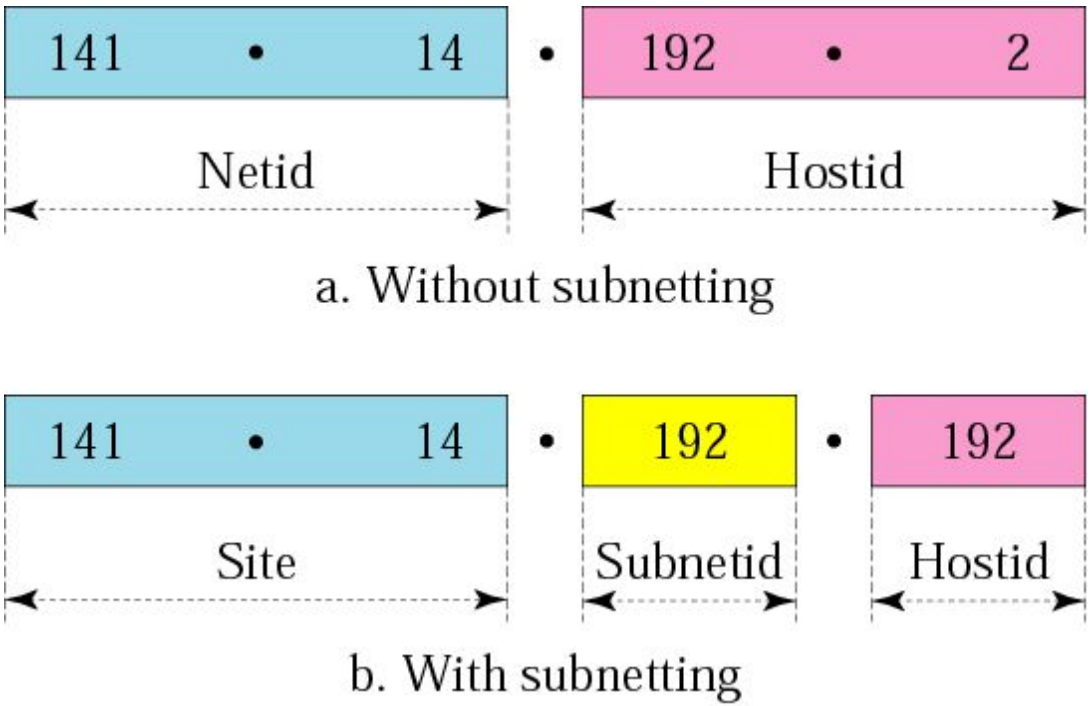
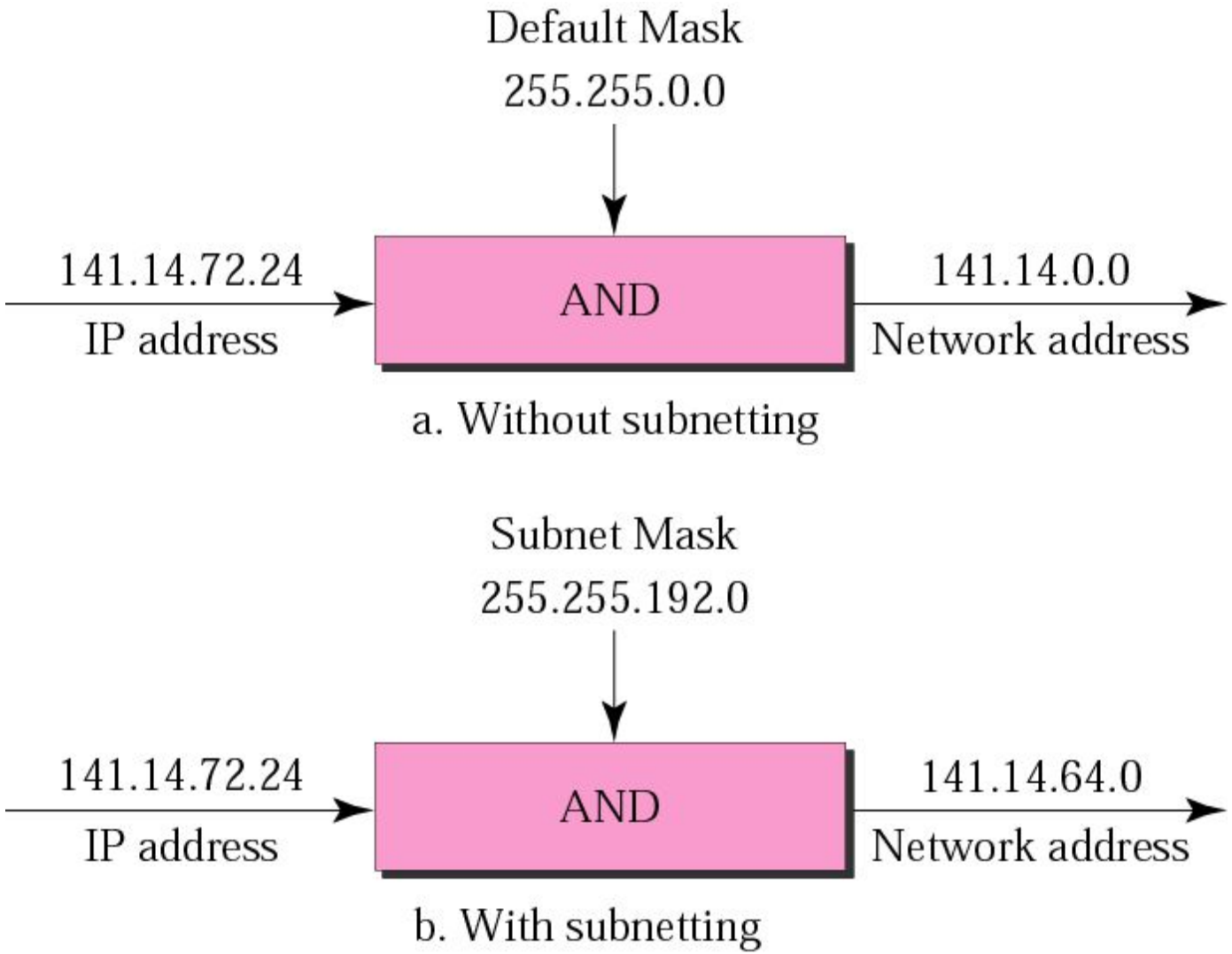


Figure 5-4

Hierarchy concept in a telephone number

Figure 5-5

Default mask and subnet mask



Finding the Subnet Address

Given an IP address, we can find the subnet address the same way we found the network address in the previous chapter. We apply the mask to the address. We can do this in two ways: straight or short-cut.

Straight Method

In the straight method, we use binary notation for both the address and the mask and then apply the AND operation to find the subnet address.

Example 1

What is the subnetwork address if the destination address is 200.45.34.56 and the subnet mask is 255.255.240.0?

Solution

11001000 00101101 00100010 00111000

11111111 11111111 1111<u>0000</u> <u>00000000</u>

11001000 00101101 0010**0000** **00000000**

The subnetwork address is 200.45.32.0.

Short-Cut Method

** If the byte in the mask is 255, copy the byte in the address.

** If the byte in the mask is 0, replace the byte in the address with 0.

** If the byte in the mask is neither 255 nor 0, we write the mask and the address in binary and apply the AND operation.

Example 2

What is the subnetwork address if the destination address is 19.30.84.5 and the mask is 255.255.192.0?

Solution

See Figure 5.6

Figure 5-6

Example 2

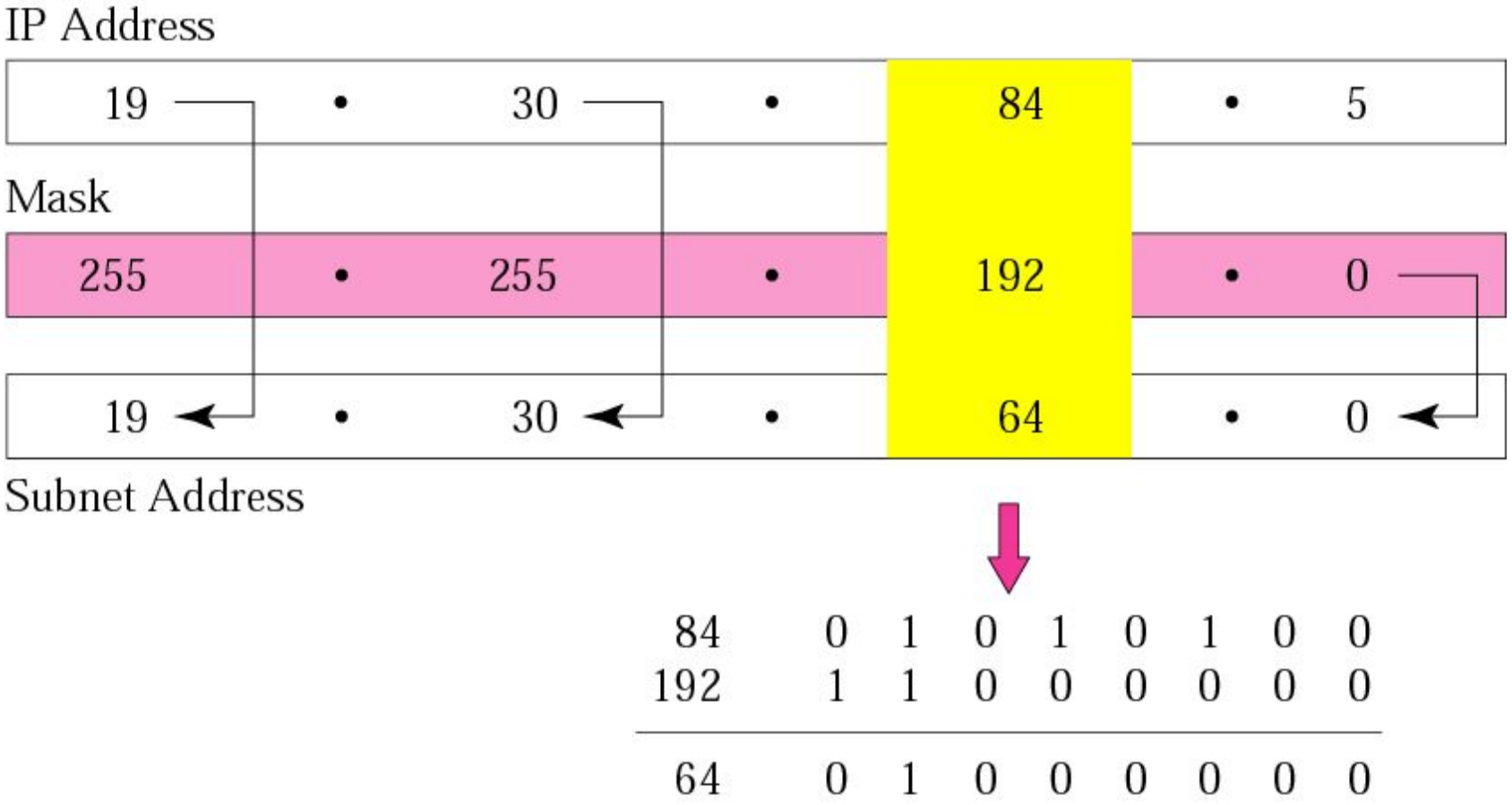
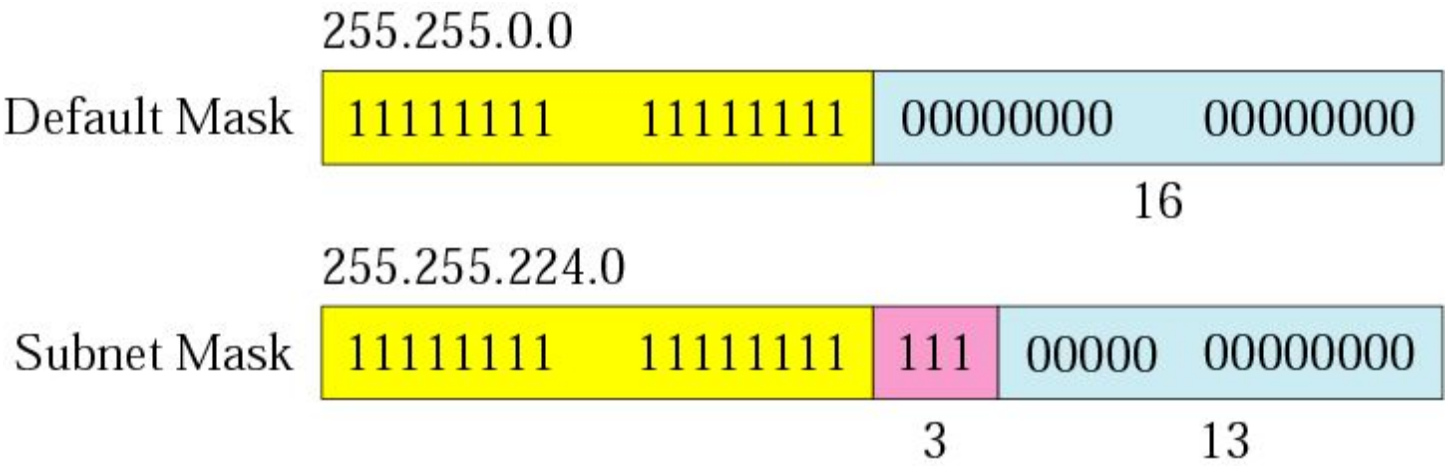


Figure 5-7

Comparison of a default mask and a subnet mask



Note

The number of subnets must be a power of 2.

Example 3

A company is granted the site address 201.70.64.0 (class C). The company needs six subnets. Design the subnets.

Solution

The number of 1s in the default mask is 24 (class C).

Solution (Continued)

The company needs six subnets. This number 6 is not a power of 2. The next number that is a power of 2 is 8 (2^3). We need 3 more 1s in the subnet mask. The total number of 1s in the subnet mask is 27 (24 + 3).

The total number of 0s is 5 (32 − 27). The mask is

Solution (Continued)

11111111 11111111 11111111 11100000

or

255.255.255.224

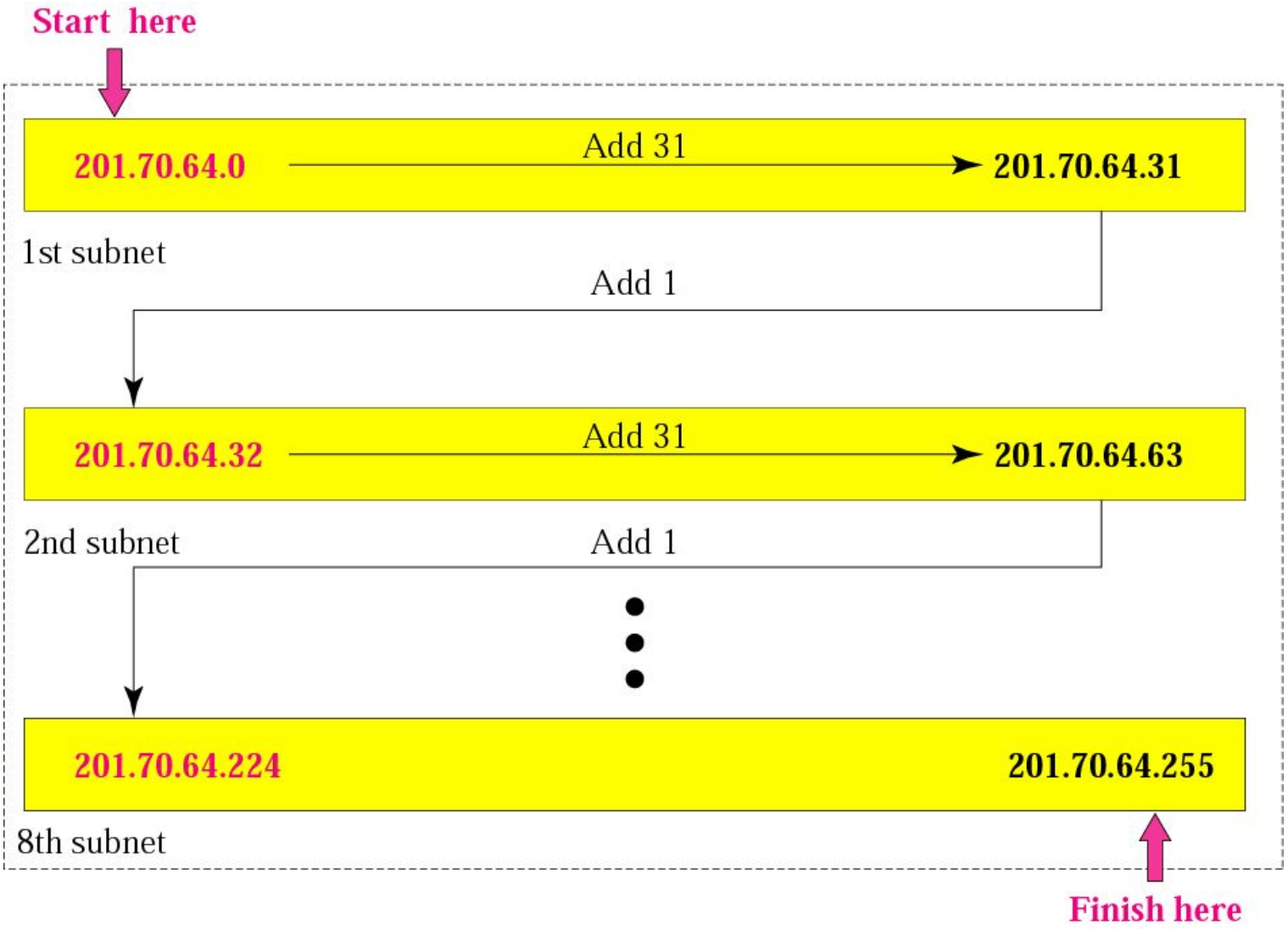
The number of subnets is 8.

The number of addresses in each subnet is 2^5 (5 is the number of 0s) or 32.

See Figure 5.8

Figure 5-8

Example 3



Example 4

A company is granted the site address 181.56.0.0 (class B). The company needs 1000 subnets. Design the subnets.

Solution

The number of 1s in the default mask is 16 (class B).

Solution (Continued)

The company needs 1000 subnets. This number is not a power of 2. The next number that is a power of 2 is 1024 (2^{10}). We need 10 more 1s in the subnet mask.

The total number of 1s in the subnet mask is 26 (16 + 10).

The total number of 0s is 6 (32 − 26).

Solution (Continued)

The mask is

11111111 11111111 11111111 11000000

or

255.255.255.192.

The number of subnets is 1024.

The number of addresses in each subnet is 2^6 (6 is the number of 0s) or 64.

See Figure 5.9

Figure 5-9

Example 4

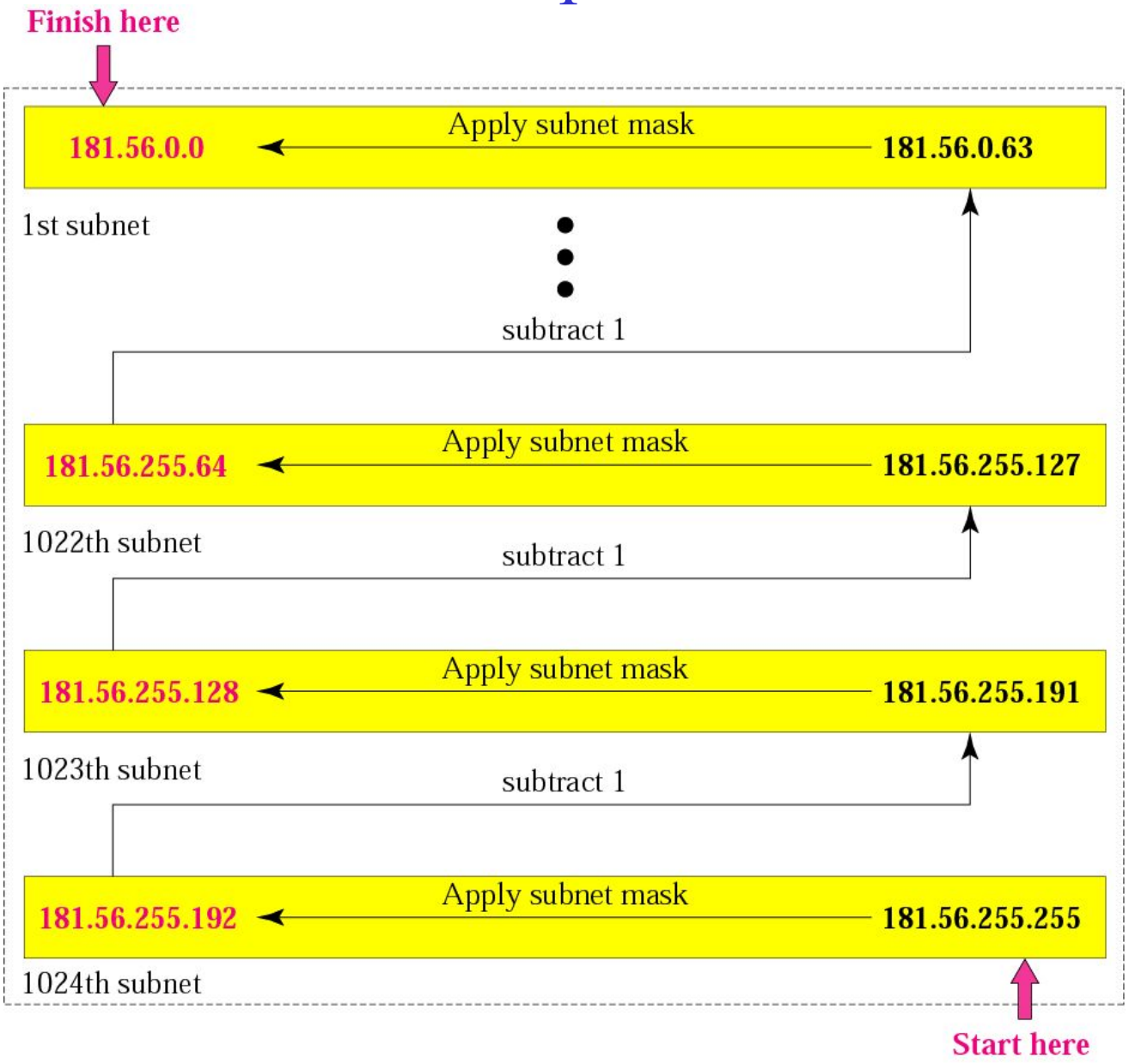
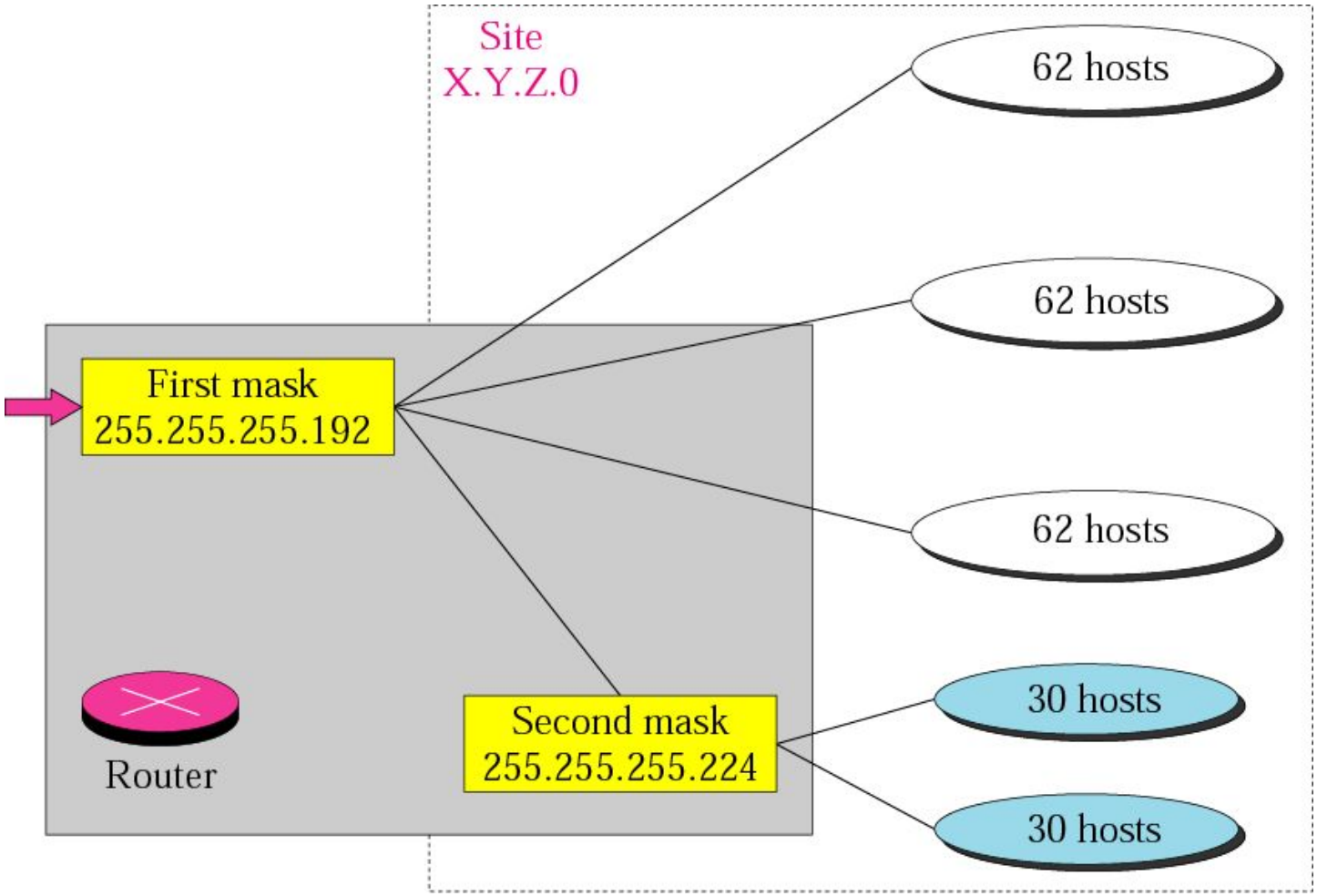


Figure 5-10

Variable-length subnetting

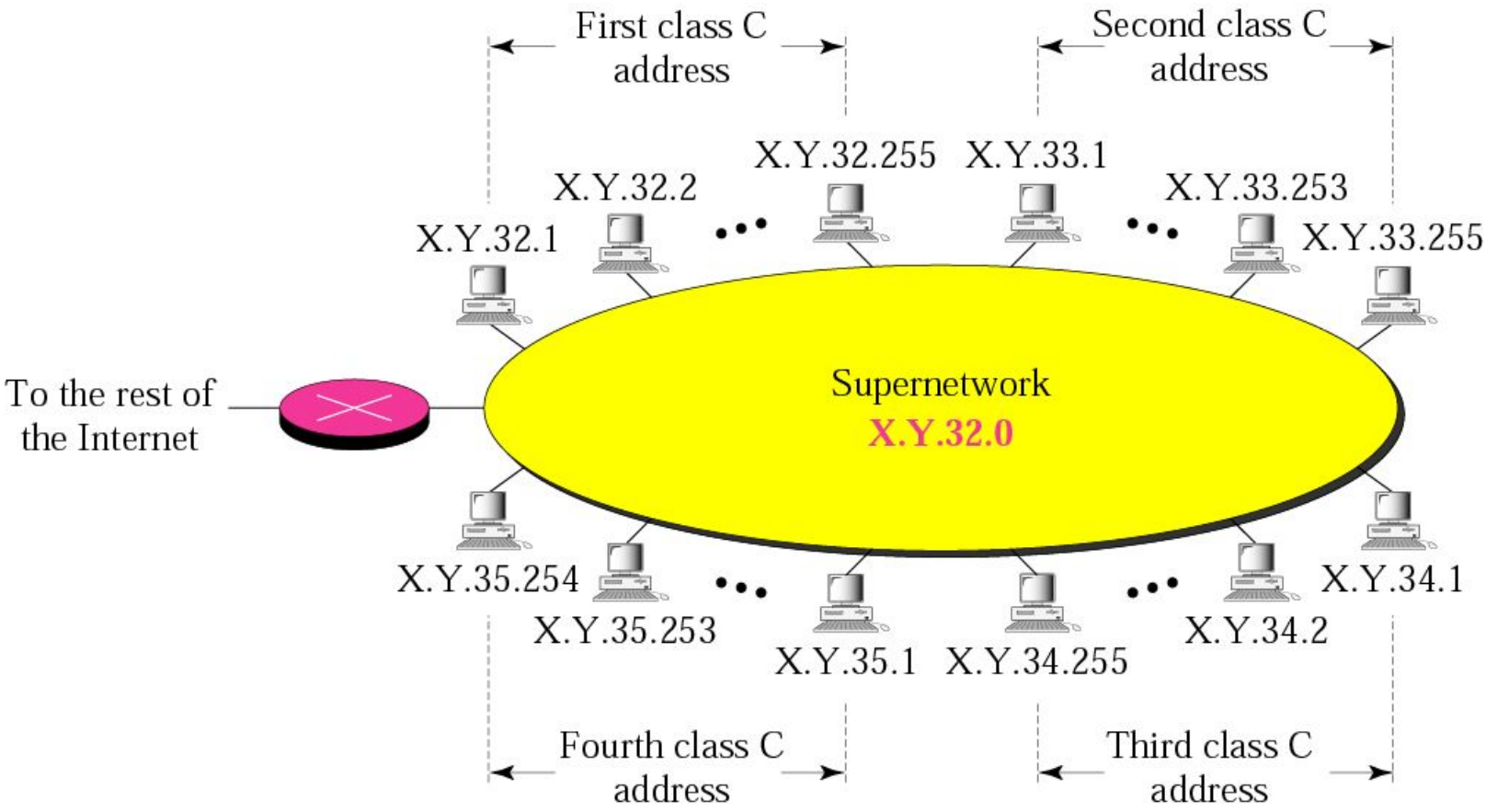


5.2

SUPERNETTING

Figure 5-11

A supernetwork



Rules:

** The number of blocks must be a power of 2 (1, 2, 4, 8, 16, . . .).

** The blocks must be contiguous in the address space (no gaps between the blocks).

** The third byte of the first address in the superblock must be evenly divisible by the number of blocks. In other words, if the number of blocks is *N*, the third byte must be divisible by *N*.

Example 5

A company needs 600 addresses. Which of the following set of class C blocks can be used to form a supernet for this company?

198.47.32.0 198.47.33.0 198.47.34.0

198.47.32.0 198.47.42.0 198.47.52.0 198.47.62.0

198.47.31.0 198.47.32.0 198.47.33.0 198.47.52.0

198.47.32.0 198.47.33.0 198.47.34.0 198.47.35.0

Solution

1: No, there are only three blocks.

2: No, the blocks are not contiguous.

3: No, 31 in the first block is not divisible by 4.

4: Yes, all three requirements are fulfilled.

Note

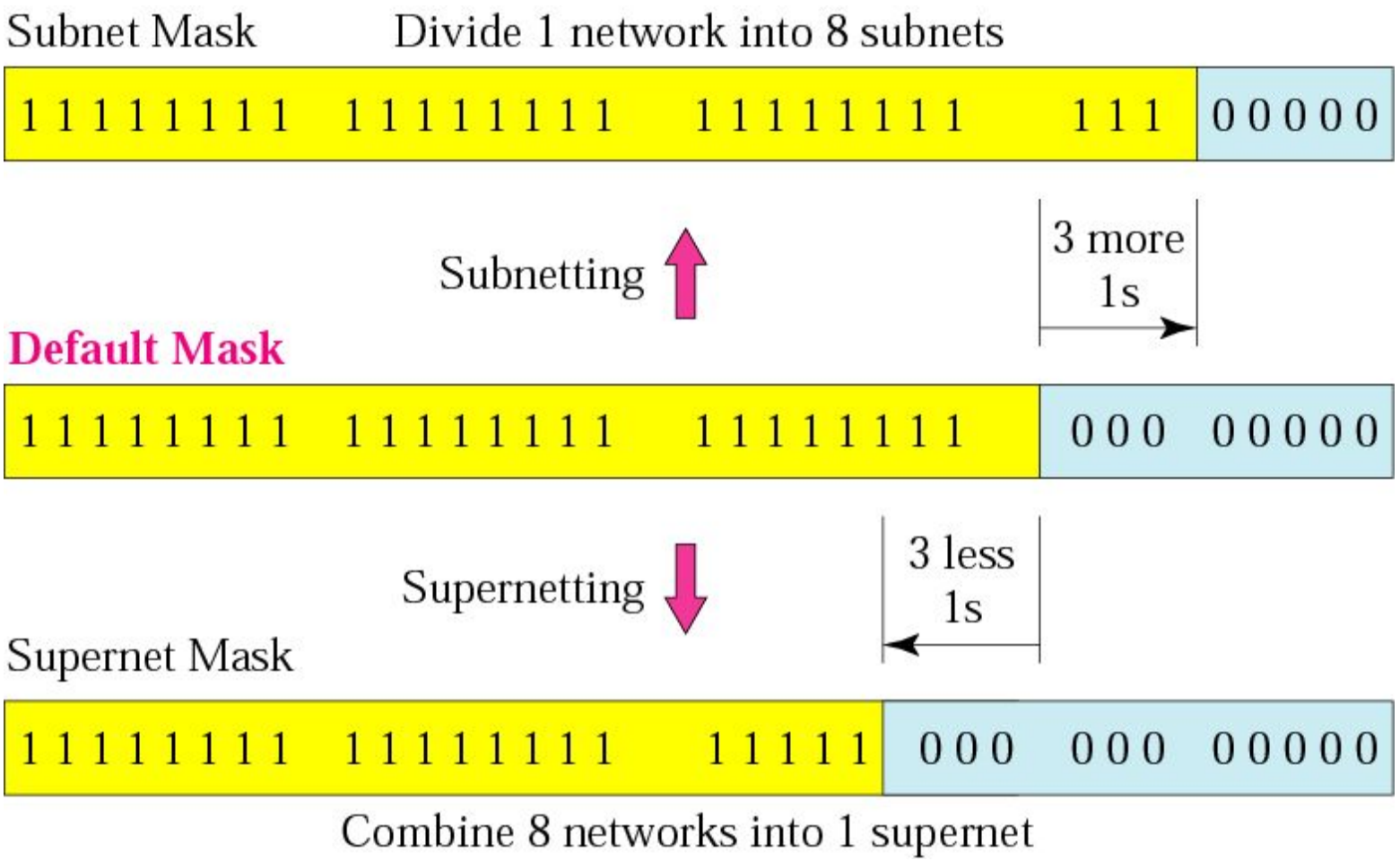
In subnetting, we need the first address of the subnet and the subnet mask to define the range of addresses.

Note

In supernetting,
we need the first address of
the supernet
and the supernet mask to
define the range of addresses.

Figure 5-12

Comparison of subnet, default, and supernet masks



Example 6

We need to make a supernetwork out of 16 class C blocks. What is the supernet mask?

Solution

We need 16 blocks. For 16 blocks we need to change four 1s to 0s in the default mask. So the mask is

11111111 11111111 11110000 00000000

or

255.255.240.0

Example 7

A supernet has a first address of 205.16.32.0 and a supernet mask of 255.255.248.0. A router receives three packets with the following destination addresses:

205.16.37.44
205.16.42.56
205.17.33.76

Which packet belongs to the supernet?

Solution

We apply the supernet mask to see if we can find the beginning address.

205.16.37.44 AND 255.255.248.0 → 205.16.32.0

205.16.42.56 AND 255.255.248.0 → 205.16.40.0

205.17.33.76 AND 255.255.248.0 → 205.17.32.0

Only the first address belongs to this supernet.

Example 8

A supernet has a first address of 205.16.32.0 and a supernet mask of 255.255.248.0. How many blocks are in this supernet and what is the range of addresses?

Solution

The supernet has 21 1s. The default mask has 24 1s. Since the difference is 3, there are 2^3 or 8 blocks in this supernet. The blocks are 205.16.32.0 to 205.16.39.0. The first address is 205.16.32.0. The last address is 205.16.39.255.

5.3

CLASSLESS ADDRESSING

Figure 5-13

Variable-length blocks

Figure 5-14

Slash notation

Note

Slash notation is also called CIDR (Classless Inter Domain Routing) notation.

Example 11

A small organization is given a block with the beginning address and the prefix length 205.16.37.24/29 (in slash notation). What is the range of the block?

olution

The beginning address is 205.16.37.24. To find the last address we keep the first 29 bits and change the last 3 bits to 1s.

Beginning: 11001111 00010000 00100101 00011000

Ending : 11001111 00010000 00100101 00011111

There are only 8 addresses in this block.

Example 12

We can find the range of addresses in Example 11 by another method. We can argue that the length of the suffix is 32 − 29 or 3. So there are $2^3 = 8$ addresses in this block. If the first address is 205.16.37.24, the last address is 205.16.37.31 (24 + 7 = 31).

Note

A block in classes A, B, and C can easily be represented in slash notation as

A.B.C.D/ *n*

where *n* is either 8 (class A), 16 (class B), or 24 (class C).

Example 13

What is the network address if one of the addresses is 167.199.170.82/27?

Solution

The prefix length is 27, which means that we must keep the first 27 bits as is and change the remaining bits (5) to 0s. The 5 bits affect only the last byte. The last byte is 01010010. Changing the last 5 bits to 0s, we get 01000000 or 64. The network address is 167.199.170.64/27.

Example 14

An organization is granted the block 130.34.12.64/26. The organization needs to have four subnets. What are the subnet addresses and the range of addresses for each subnet?

Solution

The suffix length is 6. This means the total number of addresses in the block is 64 (2^6). If we create four subnets, each subnet will have 16 addresses.

Solution (Continued)

Let us first find the subnet prefix (subnet mask). We need four subnets, which means we need to add two more 1s to the site prefix. The subnet prefix is then /28.

Subnet 1: 130.34.12.64/28 to 130.34.12.79/28.

Subnet 2 : 130.34.12.80/28 to 130.34.12.95/28.

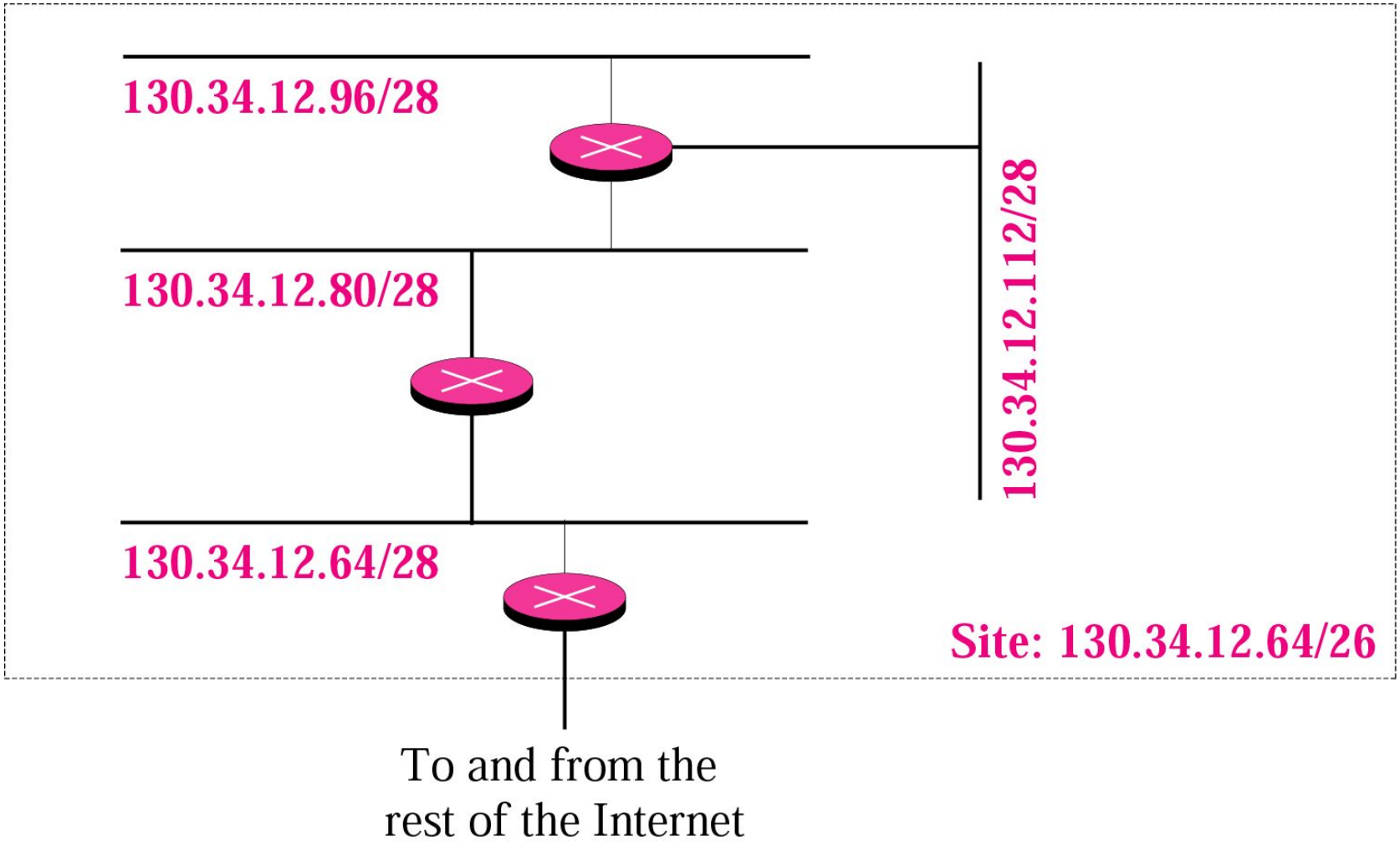
Subnet 3: 130.34.12.96/28 to 130.34.12.111/28.

Subnet 4: 130.34.12.112/28 to 130.34.12.127/28.

See Figure 5.15 on next slide

Figure 5-15

Example 14



Example 15

An ISP is granted a block of addresses starting with 190.100.0.0/16. The ISP needs to distribute these addresses to three groups of customers as follows:

1. The first group has 64 customers; each needs 256 addresses.
2. The second group has 128 customers; each needs 128 addresses.
3. The third group has 128 customers; each needs 64 addresses.

Design the subblocks and give the slash notation for each subblock. Find out how many addresses are still available after these allocations.

Solution

Group 1

For this group, each customer needs 256 addresses. This means the suffix length is 8 ($2^8 = 256$). The prefix length is then $32 - 8 = 24$.

01: 190.100.0.0/24 □ 190.100.0.255/24

02: 190.100.1.0/24 □ 190.100.1.255/24

……

64: 190.100.63.0/24 □ 190.100.63.255/24

Total = $64 \times 256 = 16{,}384$

Solution (Continued)

Group 2

For this group, each customer needs 128 addresses. This means the suffix length is 7 ($2^7 = 128$). The prefix length is then $32 - 7 = 25$. The addresses are:

001: 190.100.64.0/25 □ 190.100.64.127/25

002: 190.100.64.128/25 □ 190.100.64.255/25

003: 190.100.127.128/25 □ 190.100.127.255/25

Total = $128 \times 128 = 16{,}384$

Solution (Continued)

Group 3

For this group, each customer needs 64 addresses. This means the suffix length is 6 ($2^6 = 64$). The prefix length is then $32 - 6 = 26$. The addresses are:

001: 190.100.128.0/26 □ 190.100.128.63/26

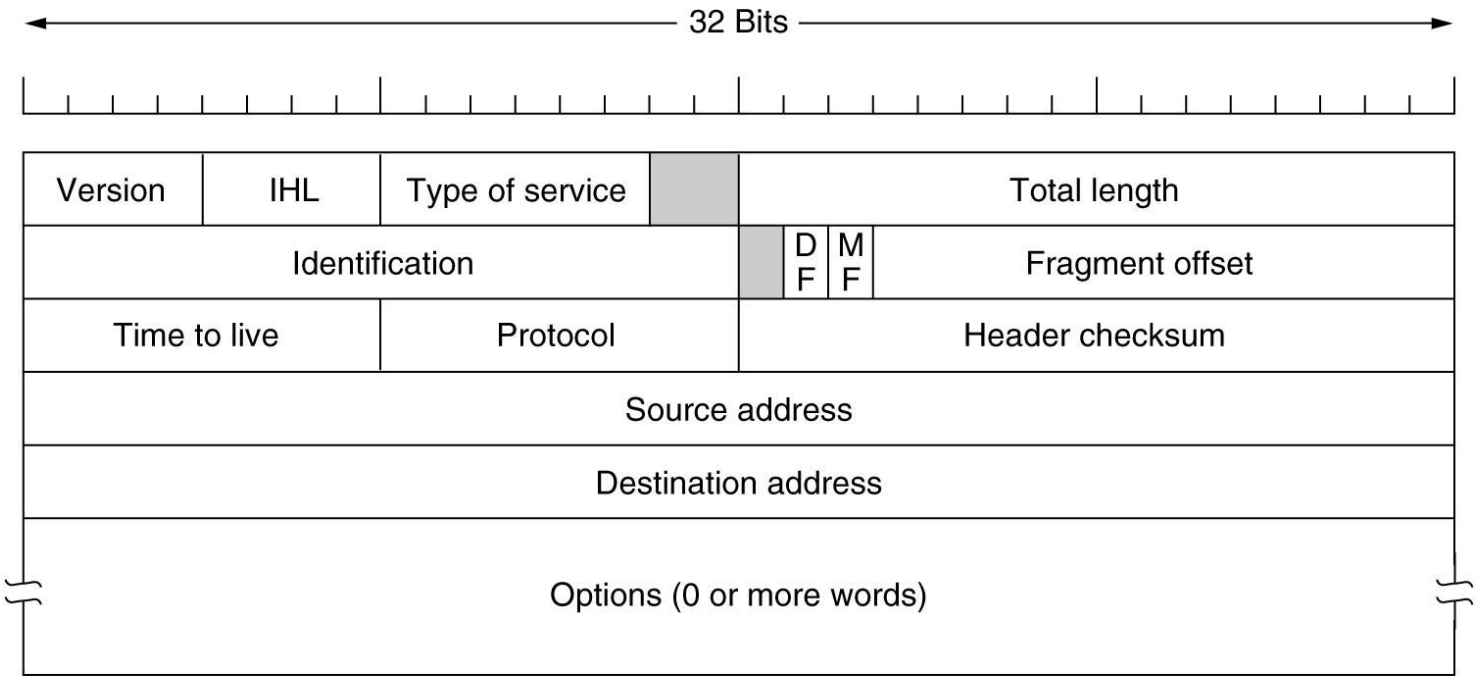
002: 190.100.128.64/26 □ 190.100.128.127/26

128: 190.100.159.192/26 □ 190.100.159.255/26

Total = $128 \times 64 = 8{,}192$

IP Protocol

- Internet Protocol (IP) is the glue
  - It facilitates packets to be transported across different types of networks, from source host to destination host



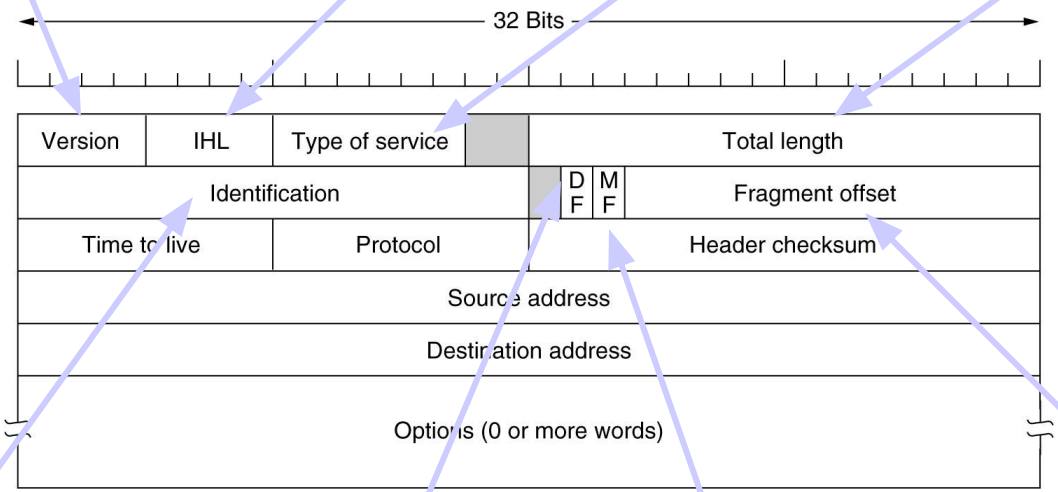
IP packet format

version of the IP protocol

IP header length in 32 bit words

used for DiffServ

length of header + payload



Unique packet id

"do not fragment"

"more fragments"

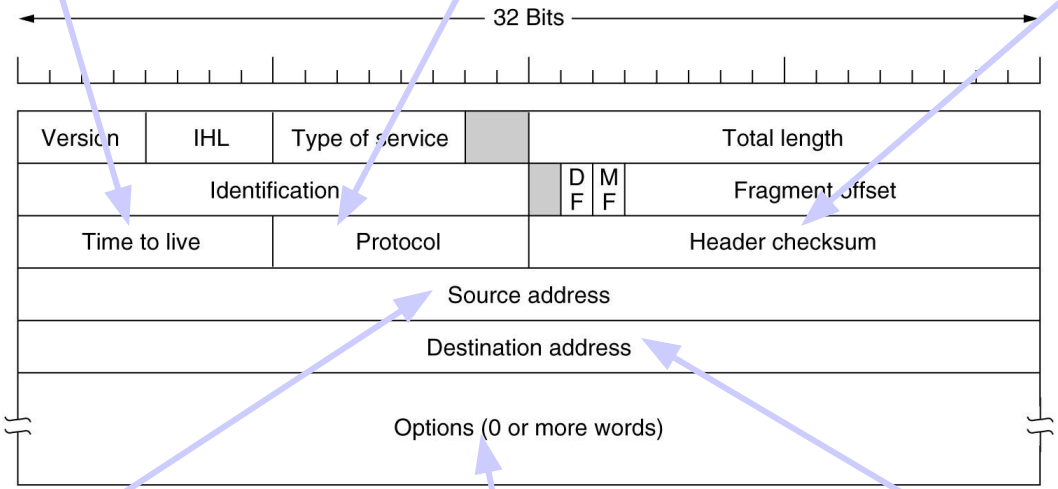
Specified in terms of "no of 8 bytes"

IP packet format

Helps to limit the no. of hops or time spent in the network

Protocol used to generate the payload (TCP, UDP etc.)

16 bit checksum, covers header only

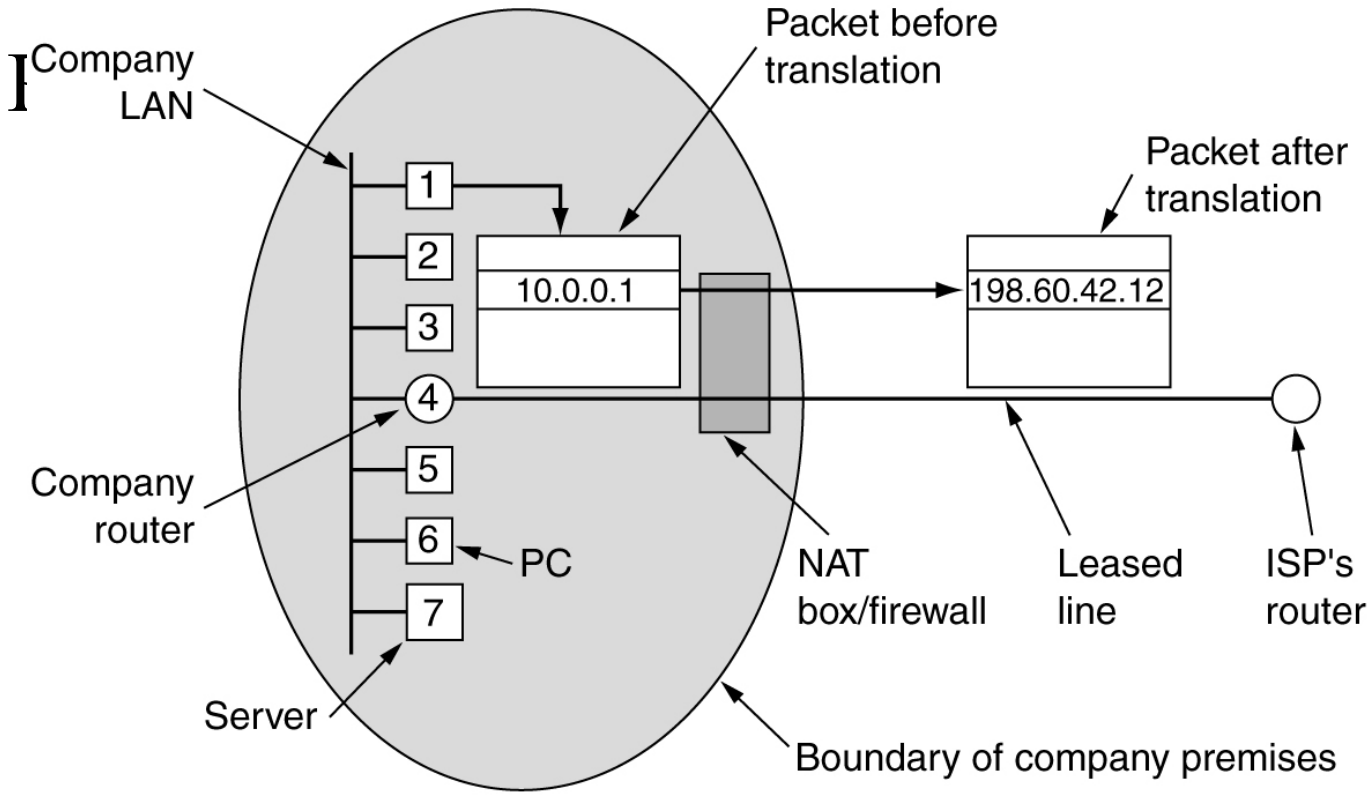


Source IP address

Optional information, such as source route

Destination IP address

NAT – Network Address Translation

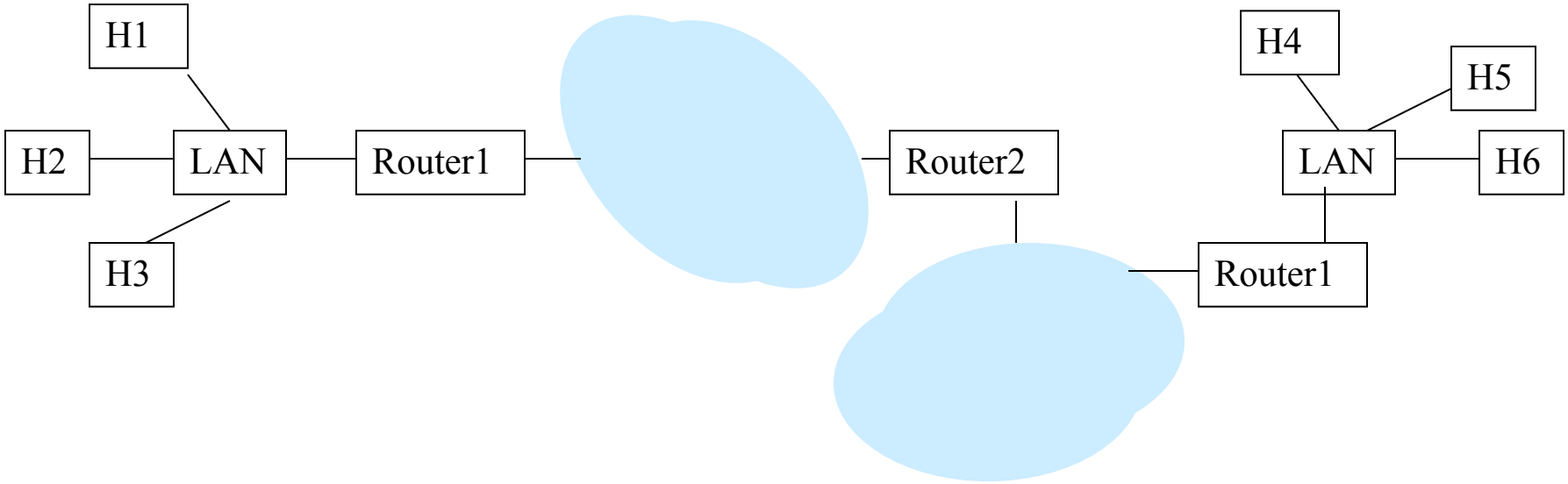


IP Fragmentation

- The *Identification* field, and *Fragment* offset field along with *Don't Fragment* and *More Fragment* flags in the IP protocol header are used for fragmentation and reassembly of IP datagram's.
- In a case where a routerIn a case where a router receives a protocol data unit (PDU) larger than the next hop's MTU(Maximum Transfer Unit).
- It has two options if the transport is IPv4It has two options if the transport is IPv4. Drop the PDU and send an Internet Control Message Protocol(ICMP) message which indicates the condition *Packet too Big*.
- or to fragment the IP packet and send it over the link with a smaller MTU.

Routing: the problem

- Largely concerned with routing datagram's through a subnet
- Between a pair of source-destination devices, packets may have to traverse several “subnets”
- Routing tables are updated every T seconds



Routing

- Performance metric :
  - Number of hops
  - Measured delay
  - Mean queue length
  - Avg traffic
  - Bandwidth
  - Transmission delay

Routing protocols: classification

- Static routes
  - Computed off-line
  - based on certain topology, traffic, performance metric
  - Not change, unless there is a major network overhaul
- Adaptive routing
  - Routes adapt to changes in topology, traffic
  - On-line based on current measurements
  - Based on complete or partial knowledge
  - Distributed computation vs. centralized computation
- Other algorithms
  - Flooding
  - Broadcasting

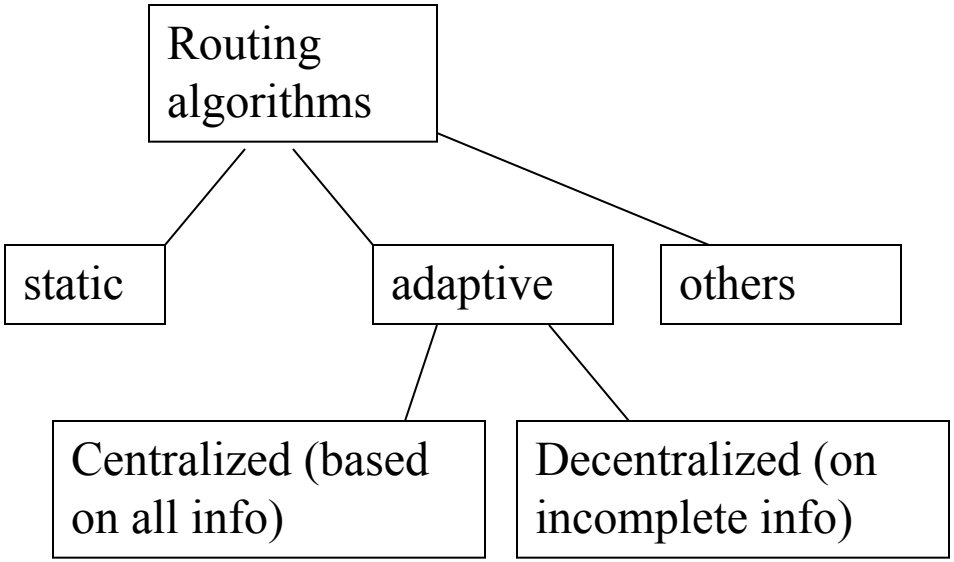
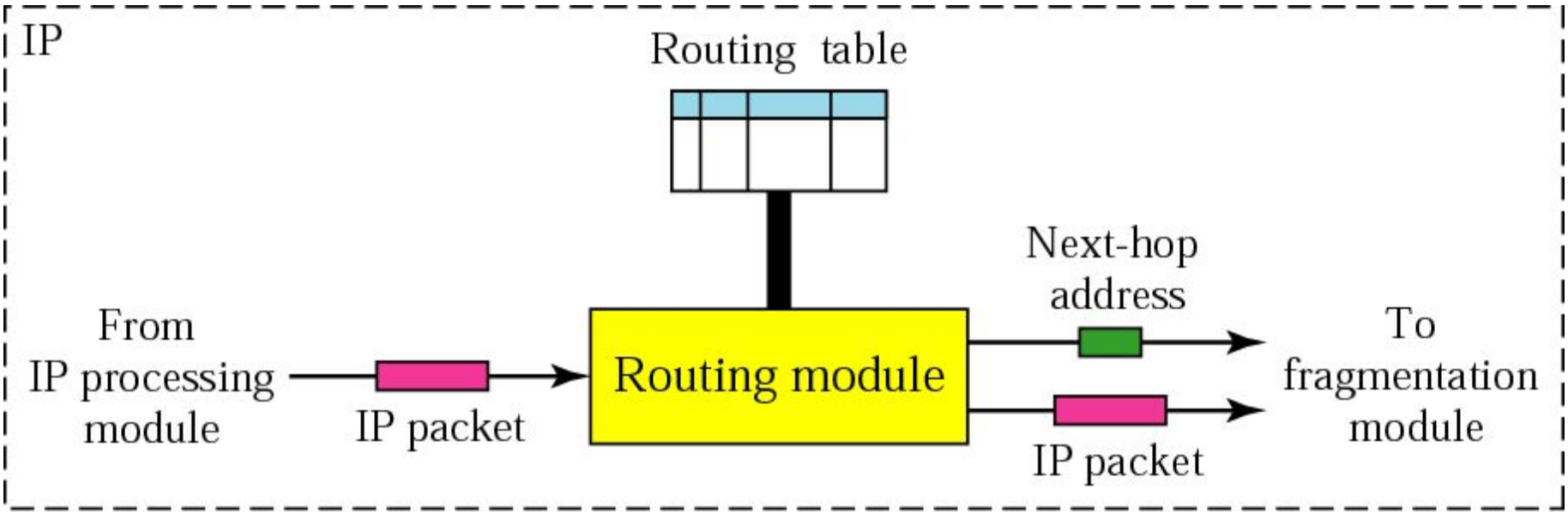


Figure 6-7

Routing module and routing table



Shortest Path Routing Algorithm

Dijkstra's algorithm

- net topology, link costs known to all nodes
  - accomplished via "link state broadcast"
  - all nodes have same info
- computes least cost paths from one node ('source") to all other nodes
  - gives forwarding table for that node
- iterative: after k iterations, know least cost path to k dest.'s

Notation:

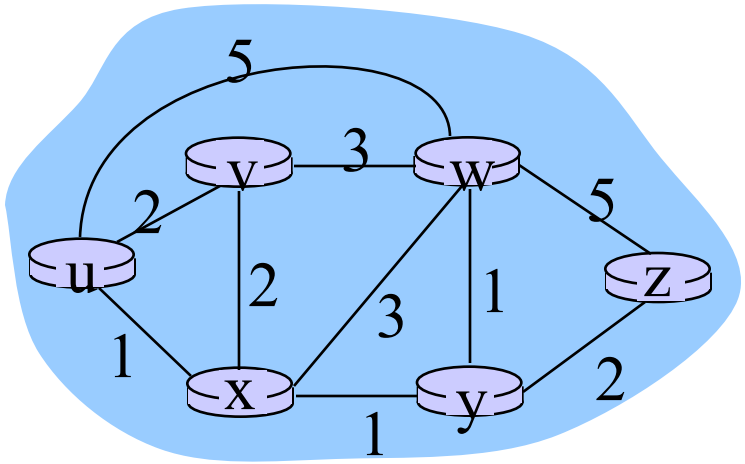
- c(x,y): link cost from node x to y; $= \infty$ if not direct neighbors
- D(v): current value of cost of path from source to dest. v
- p(v): predecessor node along path from source to v
- N': set of nodes whose least cost path definitively known

Dijsktra's Algorithm

```
1  Initialization:
2    N' = {u}
3    for all nodes v
4      if v adjacent to u
5          then D(v) = c(u,v)
6      else D(v) = ∞
7
8  Loop
9     find w not in N' such that D(w) is a minimum
10    add w to N'
11    update D(v) for all v adjacent to w and not in N' :
12       D(v) = min( D(v), D(w) + c(w,v) )
13    /* new cost to v is either old cost to v or known
14     shortest path cost to w plus cost from w to v */
15 until all nodes in N'
```

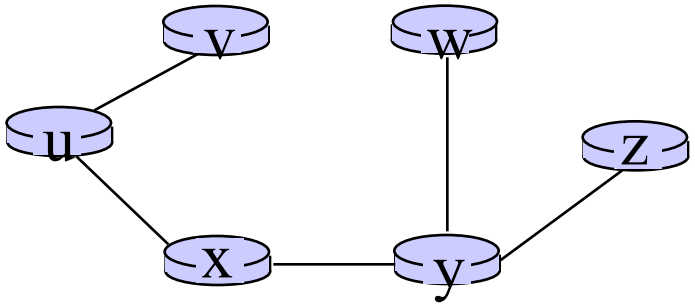
Shortest path Routing: (Dijkstra's algorithm)

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



Dijkstra's algorithm: example (2)

Resulting shortest-path tree from u:



Resulting forwarding table in u:

destination	link
v	(u, v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

Distance Vector Algorithm (1)

Bellman-Ford Equation (dynamic programming)

Define

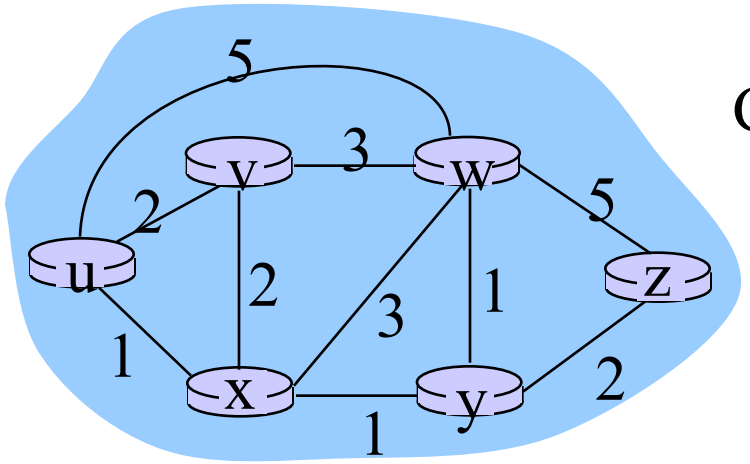
$d_x(y)$:= cost of least-cost path from x to y

Then

$d_x(y) = \min \{c(x,v) + d_v(y) \}$

where min is taken over all neighbors of x and v belongs to neighbors of x

Bellman-Ford example (2)



Clearly, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

B-F equation says:

$$
\begin{aligned}
d_u(z) = \min \{ & c(u,v) + d_v(z), \\
& c(u,x) + d_x(z), \\
& c(u,w) + d_w(z) \} \\
= \min \{ & 2 + 5, \\
& 1 + 3, \\
& 5 + 3 \} = 4
\end{aligned}
$$

Node that achieves minimum is next hop in shortest path ➜ forwarding table

Distance Vector Algorithm (3)

- $D_x(y)$ = estimate of least cost from x to y
- Distance vector: $\mathbf{D}_x = [D_x(y): y \in N]$
- Node x knows cost to each neighbor v: $c(x,v)$
- Node x maintains $\mathbf{D}_x = [D_x(y): y \in N]$
- Node x also maintains its neighbors' distance vectors
  - For each neighbor v, x maintains $\mathbf{D}_v = [D_v(y): y \in N]$

Distance vector algorithm (4)

Basic idea:

- Each node periodically sends its own distance vector estimate to neighbors v
- When a node x receives new DV estimate from neighbor, it updates its own DV using B-F equation:

$$D_x(y) \leftarrow \min_v \{c(x,v) + D_v(y)\} \quad \text{for each node } y \in N$$

r Under minor, natural conditions, the estimate $D_x(y)$ converge the actual least cost $d_x(y)$

Distance Vector Algorithm (5)

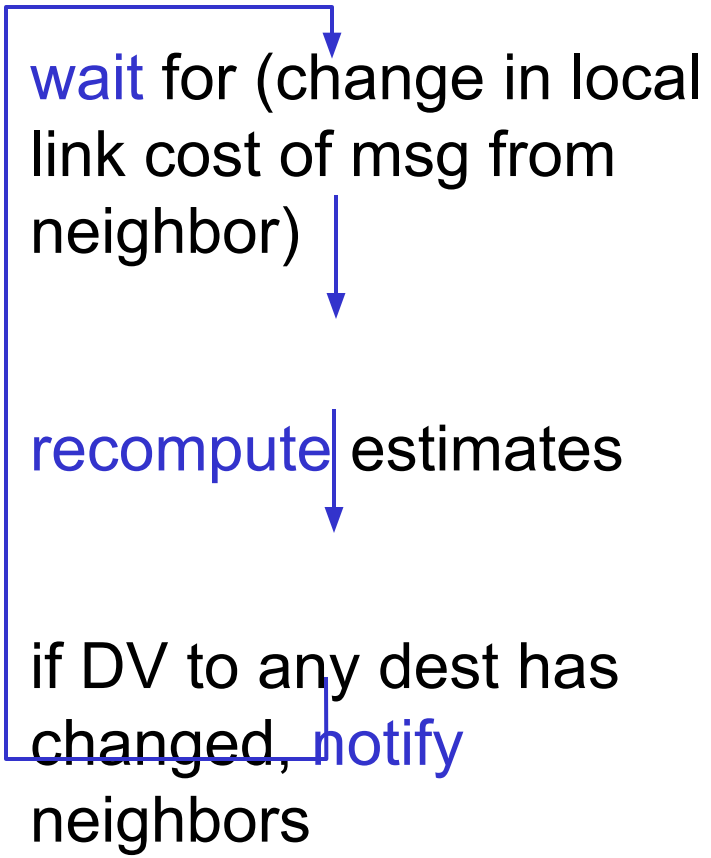
Iterative, asynchronous: each local iteration caused by:

- local link cost change
- DV update message from neighbor

Distributed:

- each node notifies neighbors *only* when its DV changes
  - neighbors then notify their neighbors if necessary

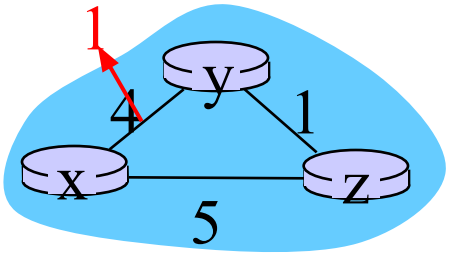
Each node:



Distance Vector: link cost changes

Link cost changes:

- node detects local link cost change
- updates routing info, recalculates distance vector
- if DV changes, notify neighbors



0

"good news travels fast"

•At time t_0, y detects the link-cost change, updates its DV, and informs its neighbors.

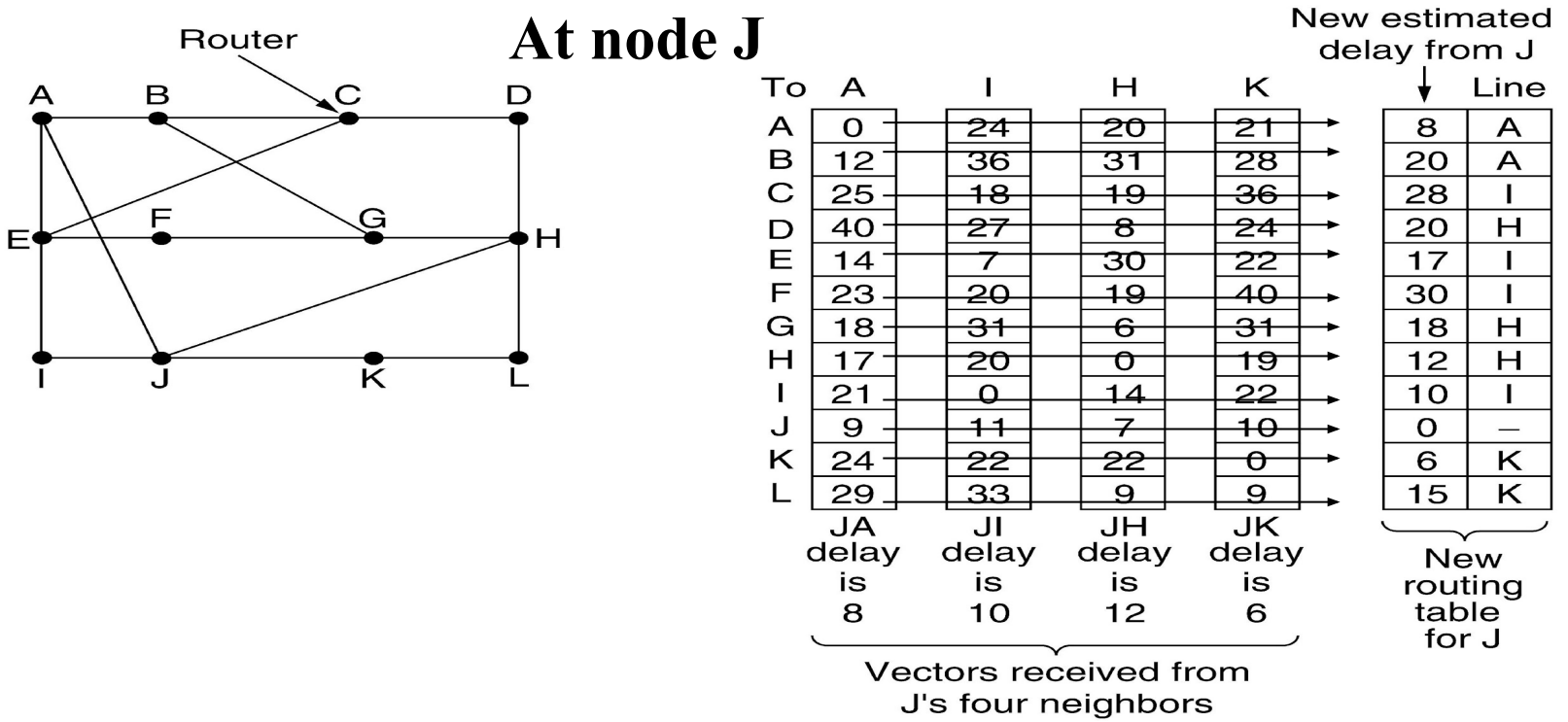
At time t_1, z receives the update from y and updates its table.

•It computes a new least cost to x and sends its neighbors its DV.

•At time t_2, y receives z's update and updates its distance table. y's least costs do not change and hence y does not send any message to z.

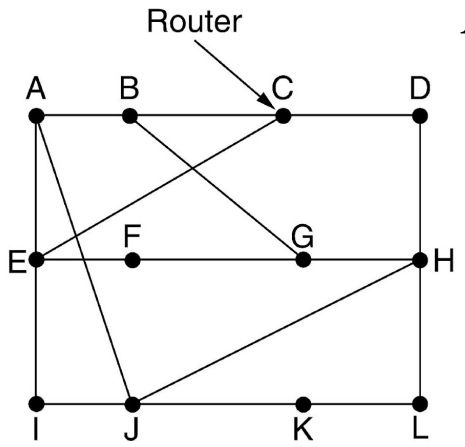
Distance-vector routing: Example

- Each router maintains a routing table, with estimated “distance” to each destination (and updates it periodically)
- Each router periodically exchanges this table with its neighbors



Distance-vector routing: Example

- Each router measures “distance” on each outgoing link
  - Using e.g. queue length, round-trip delay
- It re-computes the routes as follows:



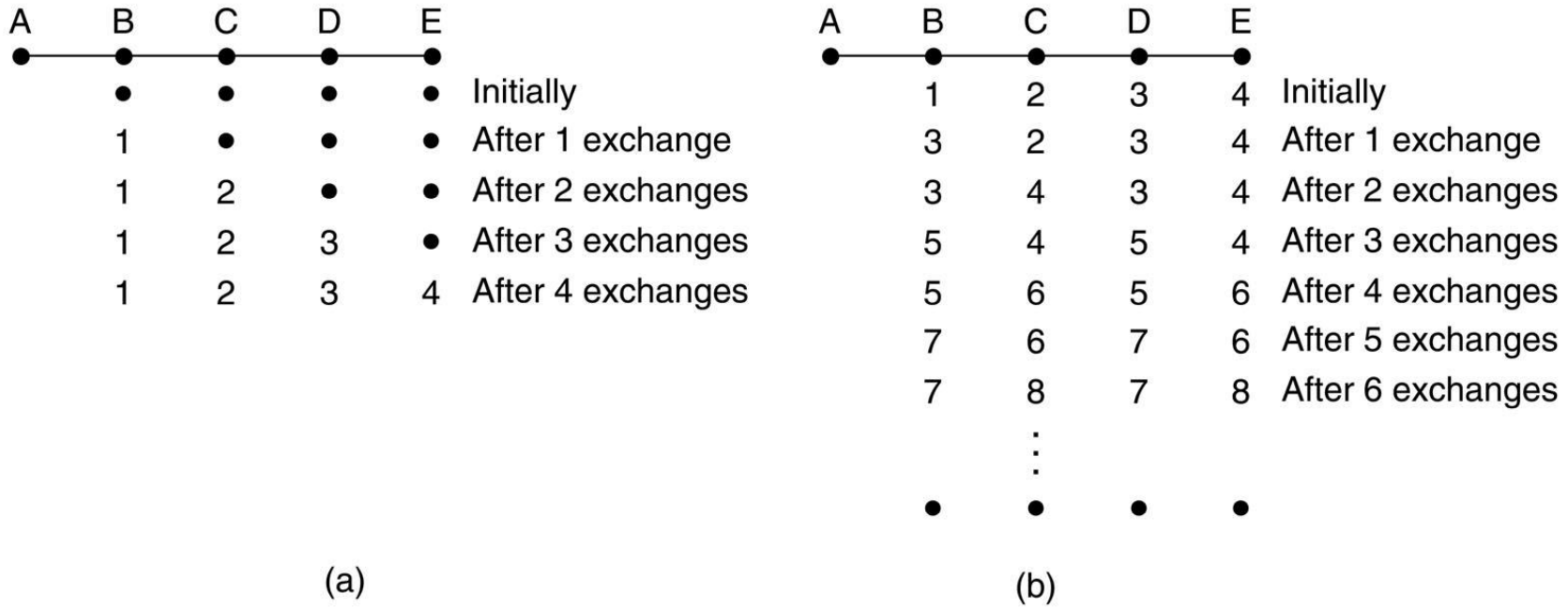
At node J

To	A	I	H	K	New estimated delay from J	Line
A	0	24	20	21	8	A
B	12	36	31	28	20	A
C	25	18	19	36	28	I
D	40	27	8	24	20	H
E	14	7	30	22	17	I
F	23	20	19	40	30	I
G	18	31	6	31	18	H
H	17	20	0	19	12	H
I	21	0	14	22	10	I
J	9	11	7	10	0	–
K	24	22	22	0	6	K
L	29	33	9	9	15	K
	JA delay is 8	JI delay is 10	JH delay is 12	JK delay is 6	New routing table for J	

Vectors received from J's four neighbors

Distance-vector routing

- Several problems with Distance Vector routing:
  - Poor estimate of delays along each link
  - Count-to-infinity problem:
    - Good news spreads fast
    - Bad news travels slow, very slow



Comparison of LS and DV algorithms

Message complexity

- LS: with n nodes, E links, O(nE) msgs sent
- DV: exchange between neighbors only

Speed of Convergence

- LS: $O(n^2)$ algorithm requires O(nE) msgs
- DV: convergence time varies
  - may be routing loops
  - count-to-infinity problem

LS:

- node can advertise incorrect *link* cost
- each node computes only its *own* table

DV:

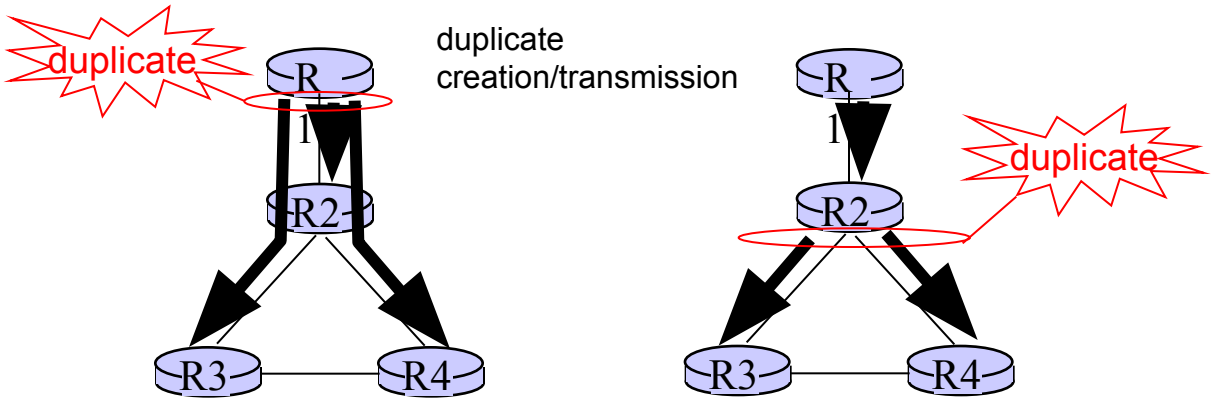
- DV node can advertise incorrect *path* cost
- each node's table used by others
  - error propagate through network

DVR vs Link State Routing

- Distance Vector Routing:
- - Tell neighbors about distance of all the destination
- - Node's computation depends on neighbors
- - Each router maintains distance vector, (dist , cost) tuple per
- destination
- - Periodically send copy of distance vector to all neighbors
- Link State Routing :
- - Tell about distance to each neighbor to all routers
- - Each router computes its best paths

Broadcast Routing

- Deliver packets from source to all other nodes
- Source duplication is inefficient:



source duplication

in-network duplication

- Source duplication: how does source determine recipient addresses?

In-network duplication

- Flooding: when node receives brdcst pckt, sends copy to all neighbors
  - Problems: cycles & broadcast storm
- Controlled flooding: node only brdcsts pkt if it hasn't brdcst same packet before
  - Node keeps track of pckt ids already brdcsted
  - Or reverse path forwarding (RPF): only forward pckt if it arrived on shortest path between node and source
- Spanning tree
  - No redundant packets received by any node

Reverse Path Forwarding

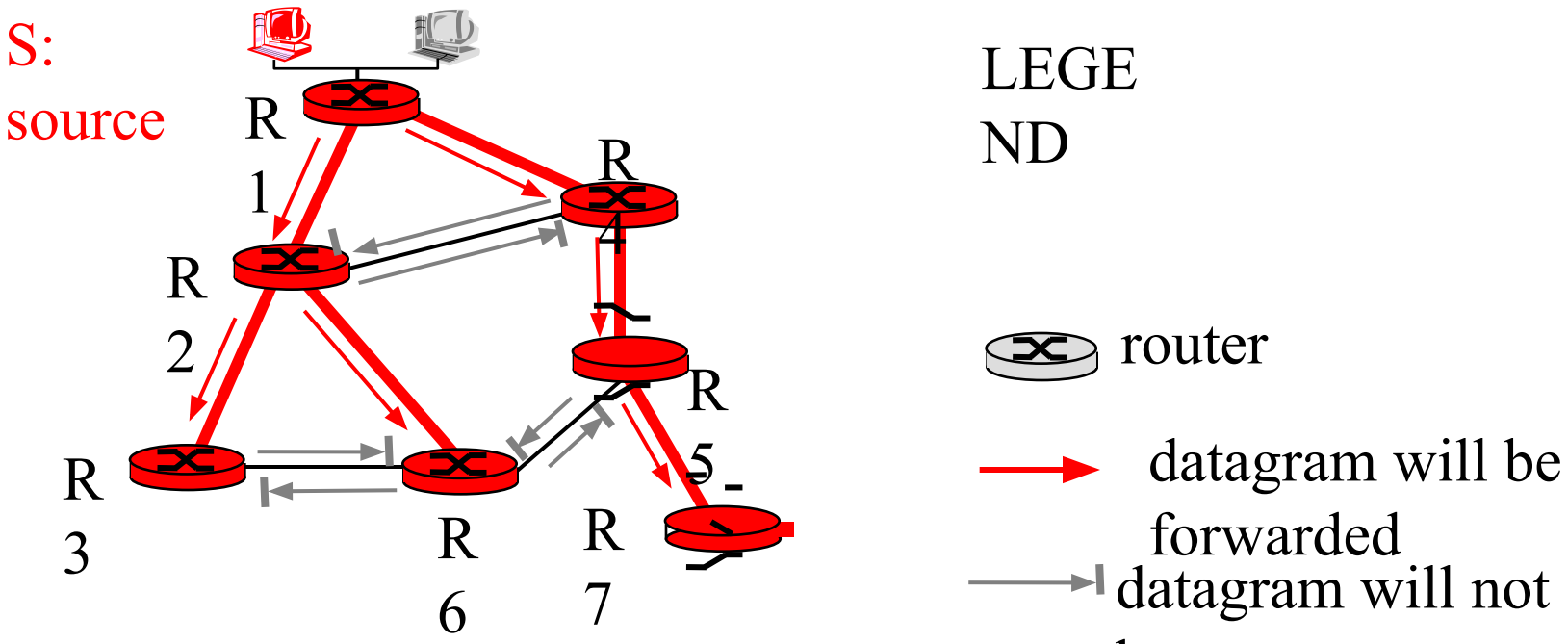
- rely on router's knowledge of unicast shortest path from it to sender
- each router has simple forwarding behavior:

if (broadcast datagram received on incoming link on shortest path back to center)

then flood datagram onto all outgoing links

else ignore datagram

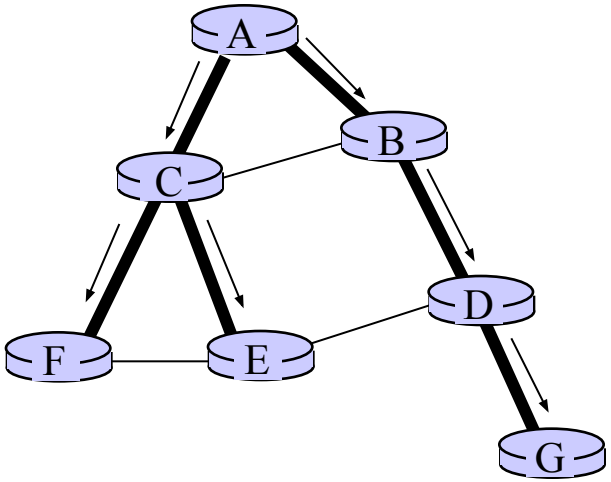
Reverse Path Forwarding: example



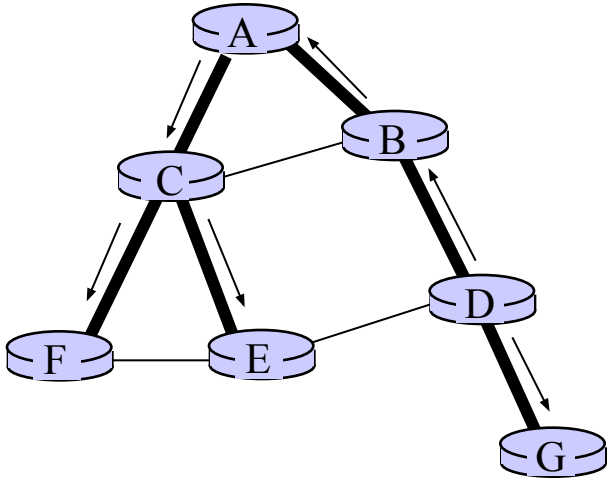
- result is a source-specific *reverse* SPT
  - may be a bad choice with asymmetric links

Spanning Tree

- First construct a spanning tree
- Nodes forward copies only along spanning tree



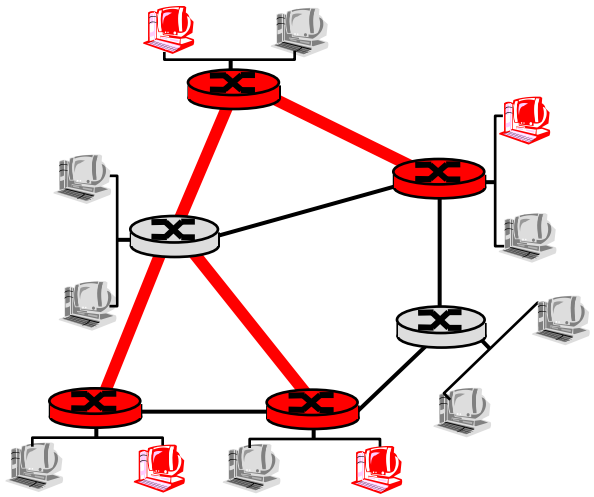
(a) Broadcast initiated at A



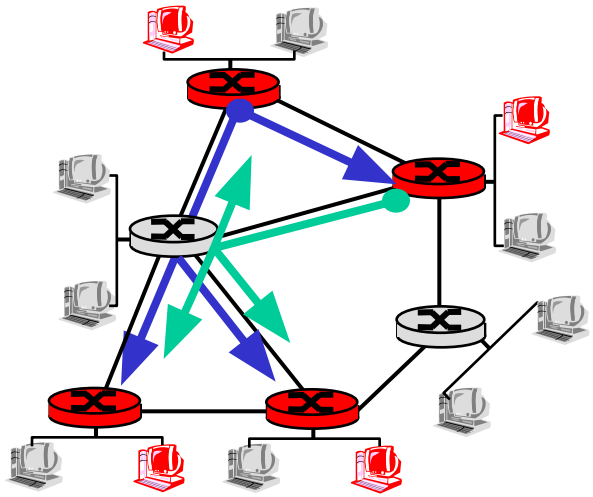
(b) Broadcast initiated at D

Multicast Routing: Problem Statement

- ***Goal:*** find a tree (or trees) connecting routers having local mcast group members
  - *tree:* not all paths between routers used
  - *source-based:* different tree from each sender to rcvrs
  - *shared-tree:* same tree used by all group members



Shared tree



Source-based trees

Approaches for building mcast trees

Approaches:

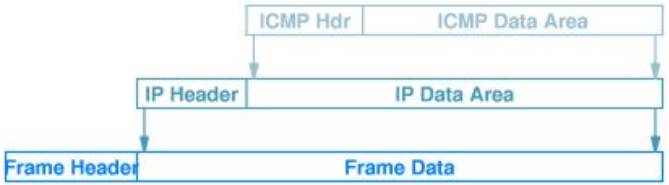
- source-based tree: one tree per source
  - shortest path trees
  - reverse path forwarding
- group-shared tree: group uses one tree
  - minimal spanning (Work out this only)
  - center-based trees

…

Internet control protocols

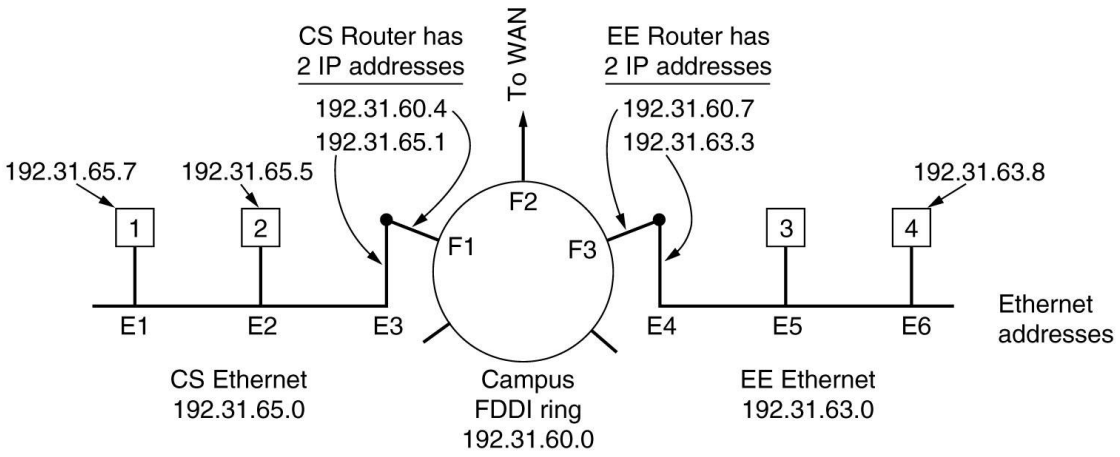
- Several protocols:
  - ARP, RARP (these are discussed later)
  - ICMP
    - Several messages, including “echo” and “echo-reply” used to “ping” hosts
    - These are encapsulated inside an IP packet

Message type	Description
Destination unreachable	Packet could not be delivered
Time exceeded	Time to live field hit 0
Parameter problem	Invalid header field
Source quench	Choke packet
Redirect	Teach a router about geography
Echo	Ask a machine if it is alive
Echo reply	Yes, I am alive
Timestamp request	Same as Echo request, but with timestamp
Timestamp reply	Same as Echo reply, but with timestamp



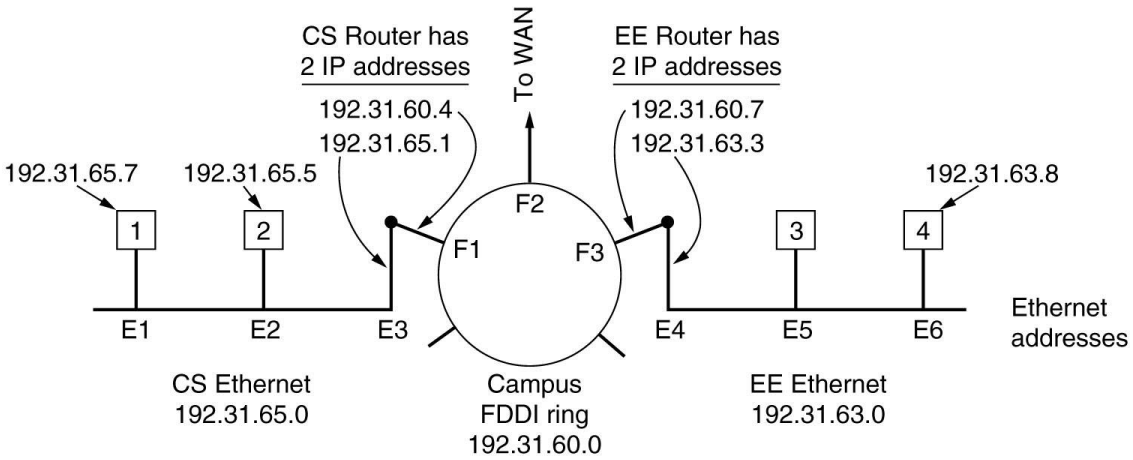
ARP protocol

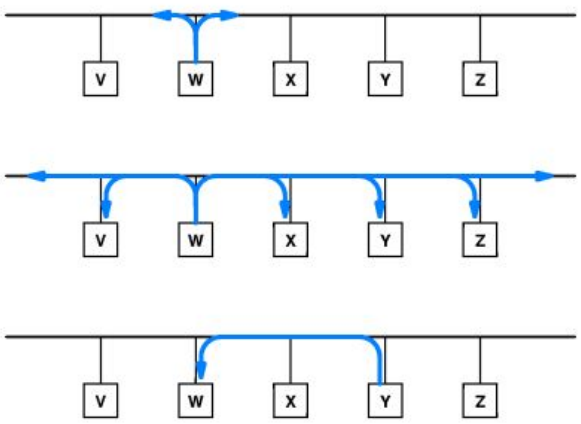
- ARP protocol: “address resolution protocol”
  - IP address □□ Data-link (or physical) address
  - This is distinct from”domain-name” □□ IP address problem



ARP protocol

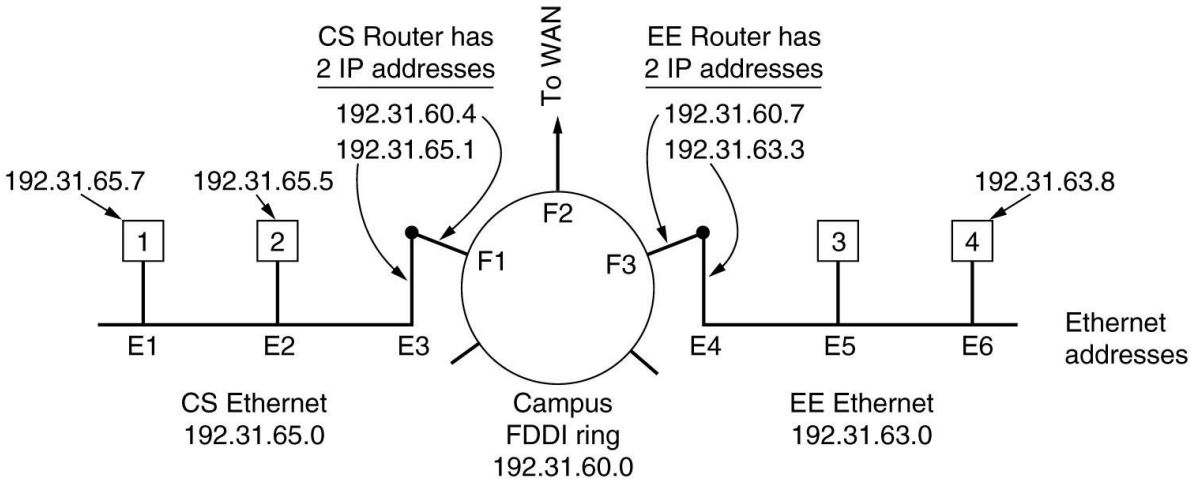
- ARP protocol:
  - ARP-REQ ARP-REPLY packets
    - ARP-REQ is broadcast over local subnet only
  - Destination IP address □□ Ethernet address is cached by source, once a reply is received
  - The destination also caches similar info about the source





ARP protocol

- Consider H1 to H4 communication
  - H1 issues an ARP-REQ, to which CS router responds with its E3 address
  - CS router issues an ARP-REQ on FDDI ring, to which EE router responds with its F3 address
  - EE router issues an ARP-REQ on the Ethernet, to which H4 responds with its E6 address

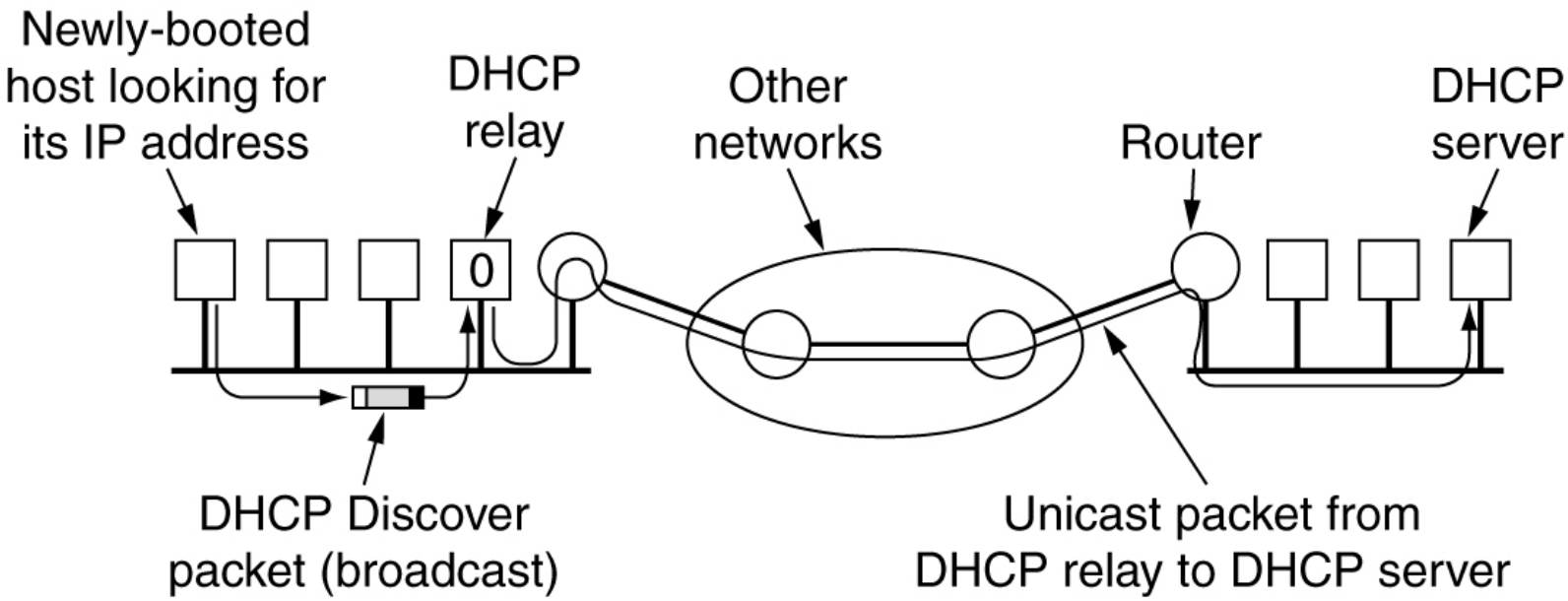


RARP protocol

- ARP gives IP-addr □ Physical-addr
- RARP solves the problem of “what is my IP address”?
  - A problem that occurs in disk-less workstations, that have no disk resident OS
- RARP-REQ issued by client, while RARP-REPLY is sent by RARP server
- Need a RARP server for each network separated by a router
- Need to have entries for each IP-addr □□ IP address
- Both problems solved using DHCP protocol

Dynamic Host Configuration Protocol

Operation of DHCP.



Thanks