

# Key Points

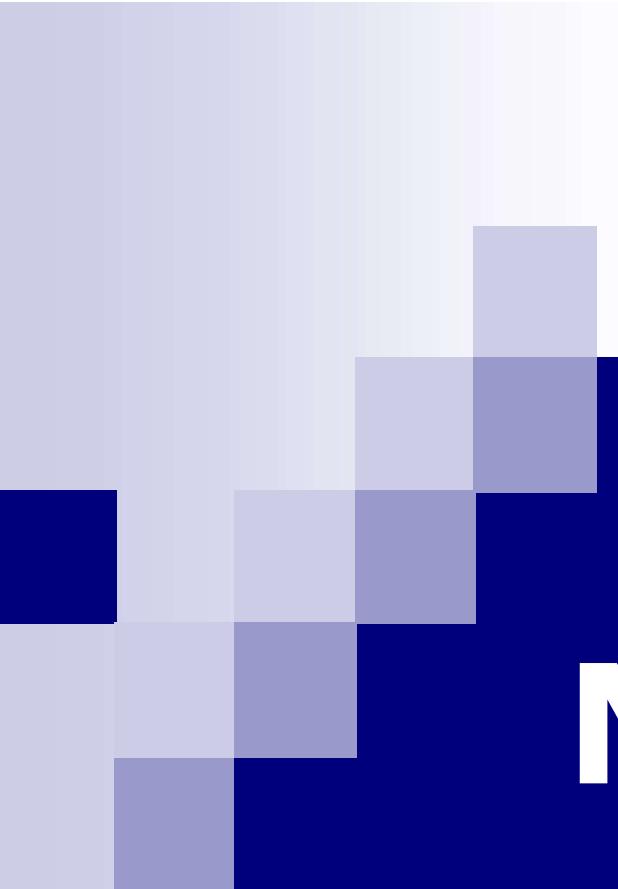
<b>Routing Algorithms</b>	Flooding, Dijkstra, Distance Vector, Link State	熟练掌握
	Hierarchical Routing	掌握
<b>Congestion Control</b>	Principles, Solutions	掌握
<b>IP</b>	IP Datagram format, Fragmentation, IP Addresses, Sub-netting, CIDR and Route Aggregation (Super-netting), IP routing and forwarding, ARP, ICMP, NAT, IPv6	熟练掌握
<b>Routing in the Internet</b>	RIP	熟练掌握
	OSPF, BGP	掌握
<b>IP Multicasting</b>	Concepts, principles	理解
<b>Mobile IP</b>	Concepts, principles	理解

# Chapter 5: Roadmap

- **Network Layer and Design Issues**
- **Routing Algorithms**
- **Congestion Control Algorithm**
- **IP**
- **IP Multicasting**
- **Mobile IP**

# Network Layer

- Concerned with getting data from source (**sending host**) to destination (**receiving host**).
- The network layer must know the topology of the subnet and choose appropriate paths through it.
- When source and destination are in ***different networks***, the network layer (e.g. IP) must deal with these differences.



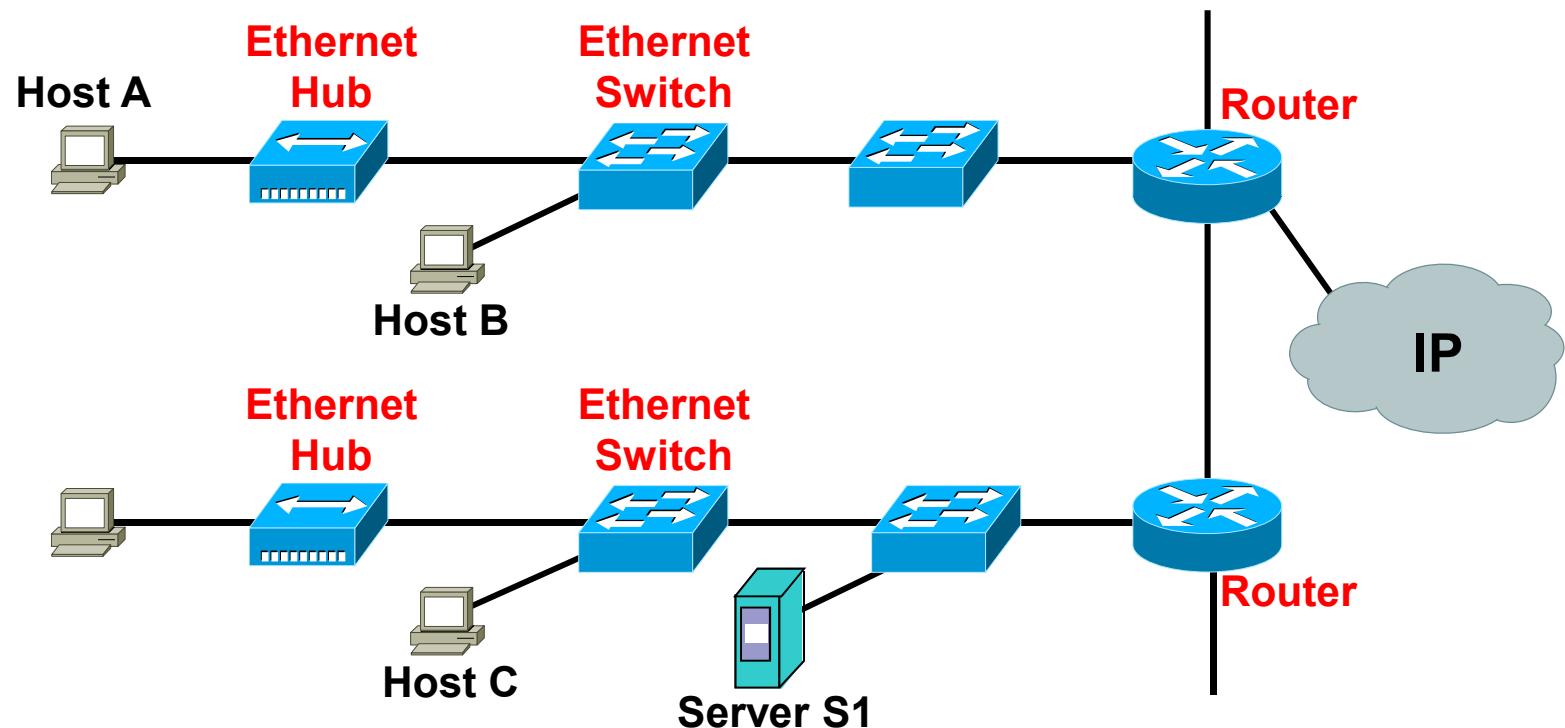
# **Chapter 05**

# **Network Layer**

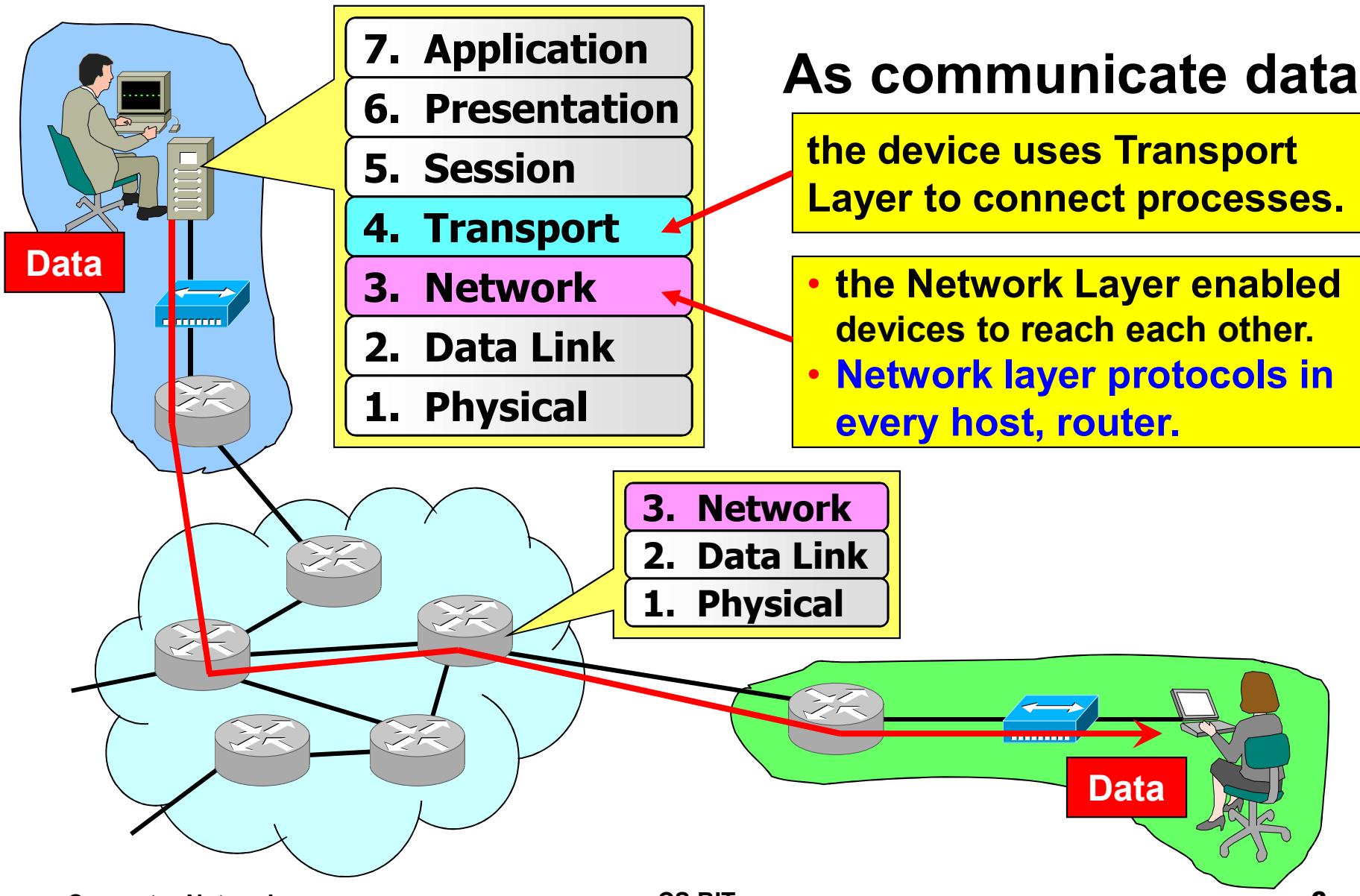
**Associate Prof. Zheng, Hong (郑宏)**  
**Computer School**  
**Beijing Institute of Technology**

# Network Layer

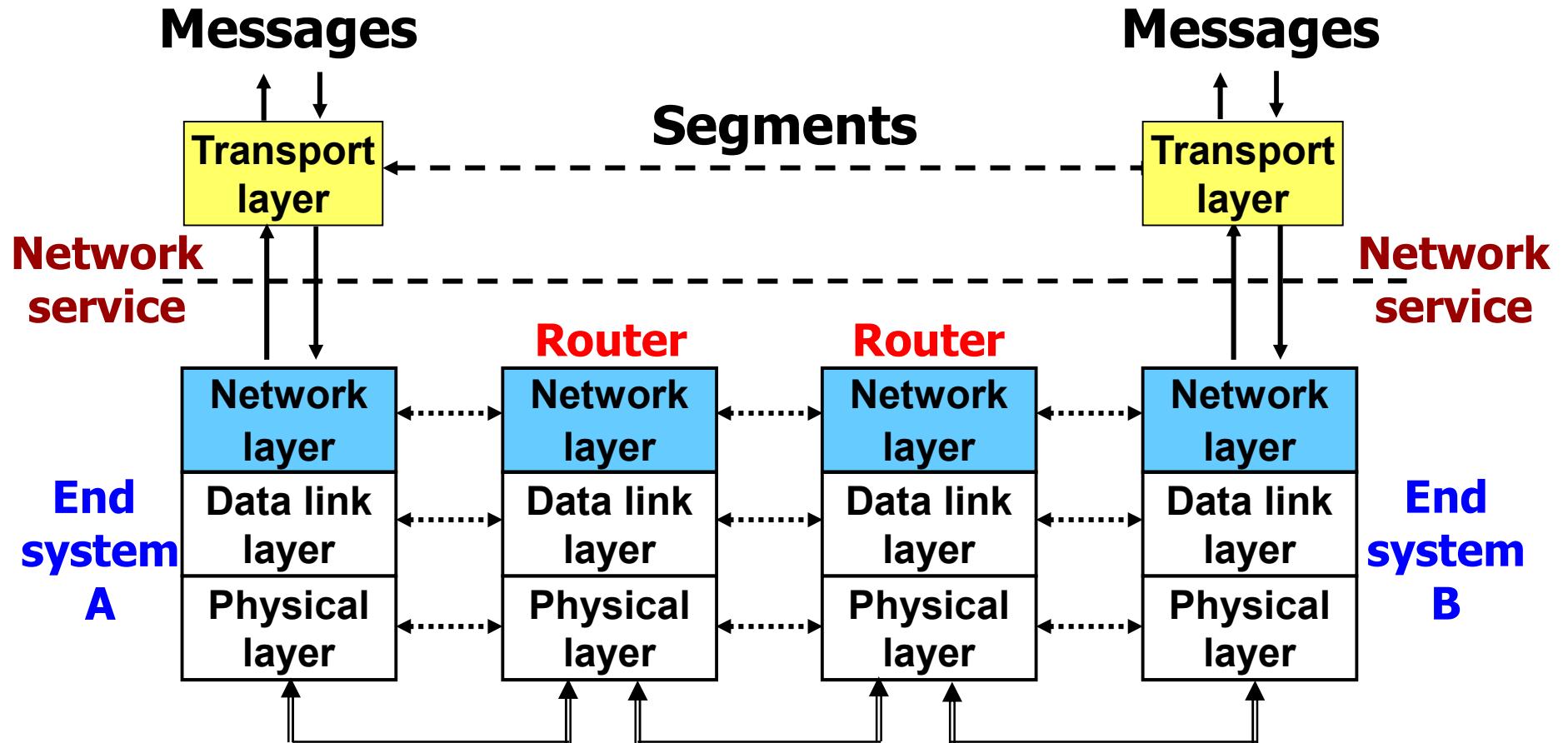
- What does the “*different networks*” means?
- How do you/hubs/switches/routers determine if the source and destination are in different/same networks?



# Network Layer

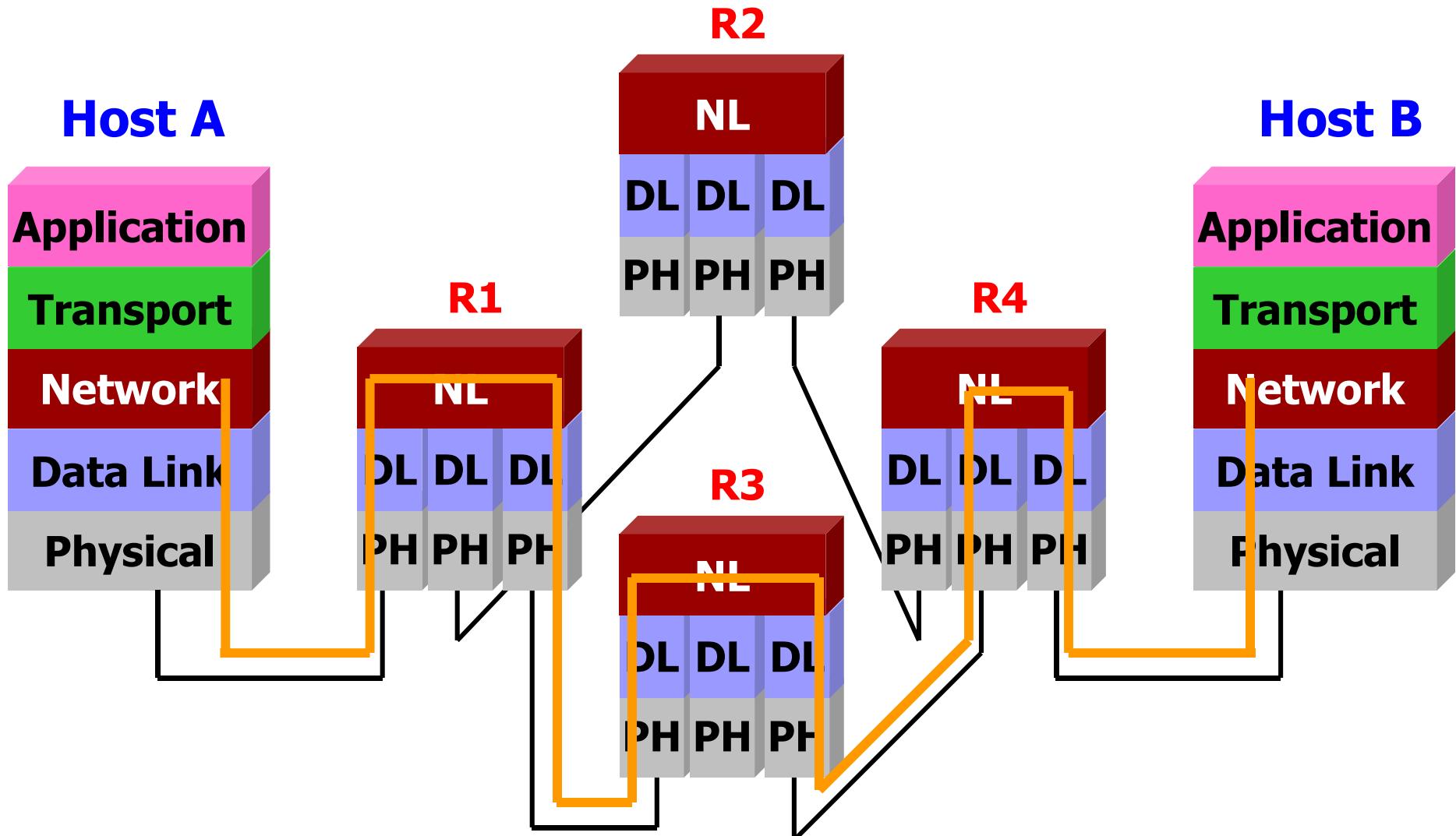


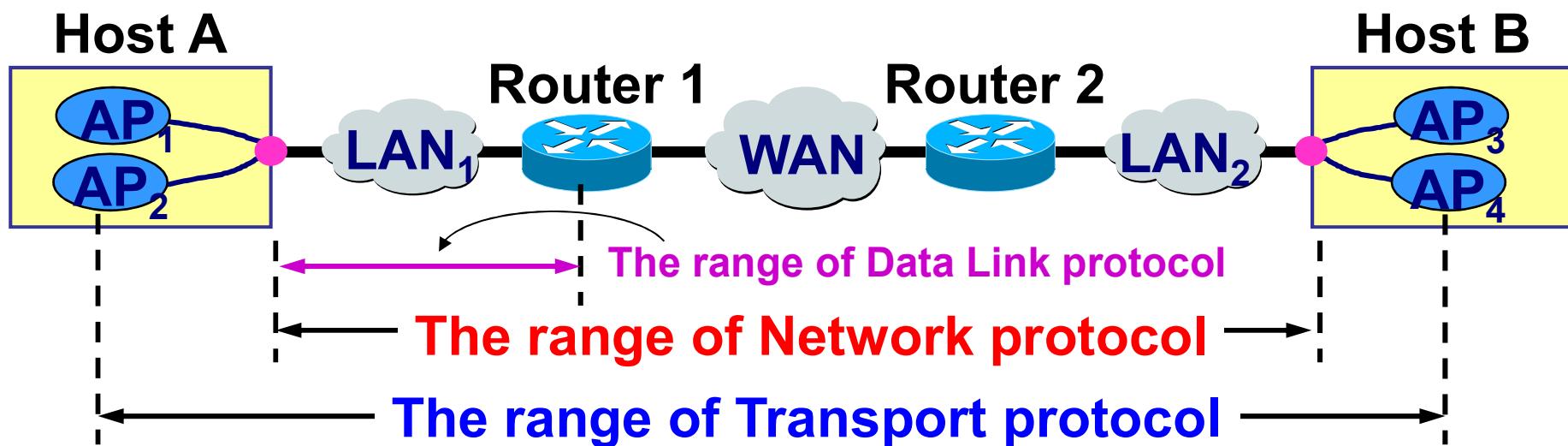
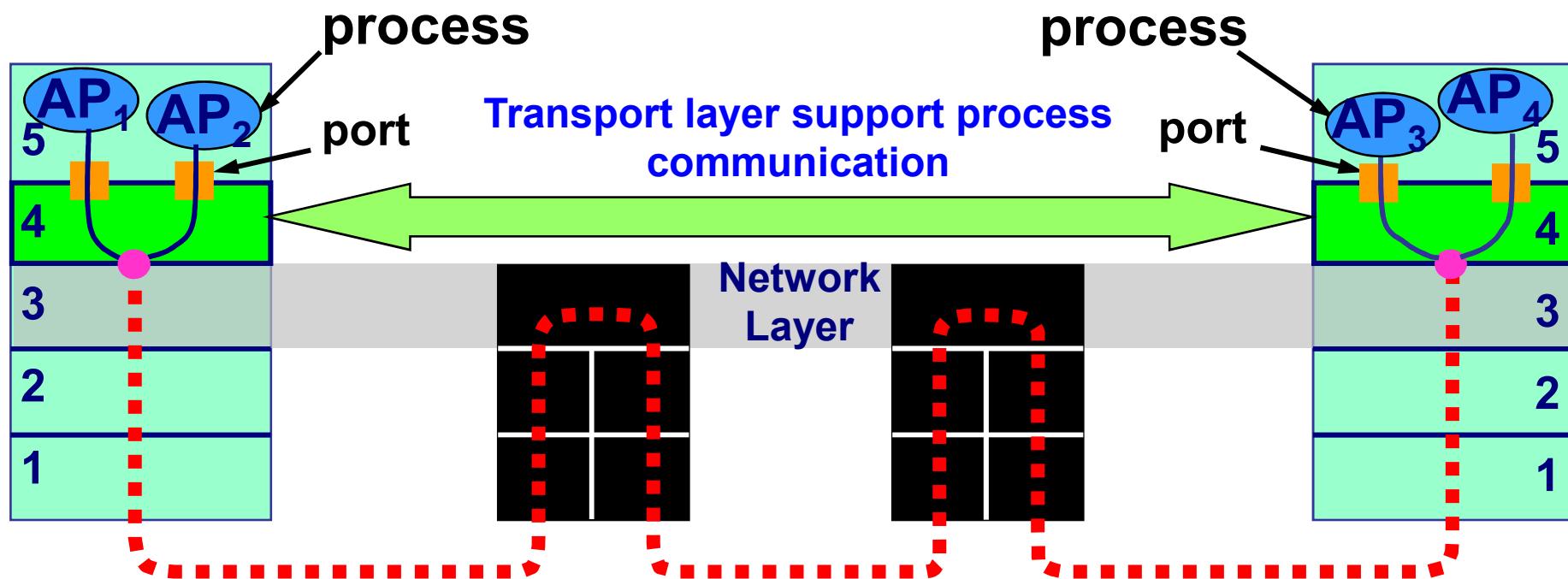
# Network Layer



Network Layer Model

# Network Layer





# Network Layer Design Issues

- Network layer is responsible for
  - host-to-host delivery
  - routing the packets through the routers or switches.
- **Issues:**
  - Services provided to the upper Layer
  - Packet Switching

# Services Provided to the Transport Layer

## Design Goals:

1. The services provided by the network layer should be **independent** of the subnet topology.
2. The Transport Layer should be **shielded** from the number, type and topology of the subnets present.
3. The network addresses available to the Transport Layer should use a **uniform** numbering plan (even across LANs and WANs).

# Services Provided to the Transport Layer

- **Qusetion:** What *service model* for “channel” transporting packets from sender to receiver?
  - guaranteed bandwidth?
  - preservation of inter-packet timing (no jitter)?
  - loss-free delivery?
  - in-order delivery?
  - congestion feedback to sender?

# Services Provided to the Transport Layer

## Two camps:

### Connection-oriented services

- Makes a connection.
- Sends packets one after another in same path.
- Terminates the connection.

### Connectionless services

- Treats each packet independently.
- Packets may or may not travel in the same path.

# Services Provided to the Transport Layer

## *Connection-Oriented*



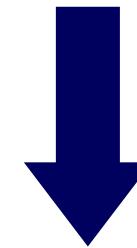
virtual circuit service

**Virtual Circuit Switching**

**Example:** ATM, X.25,  
Frame Relay

**Analogy:** Telephone

## *Connection-Less*



datagram service

**Datagram Switching**

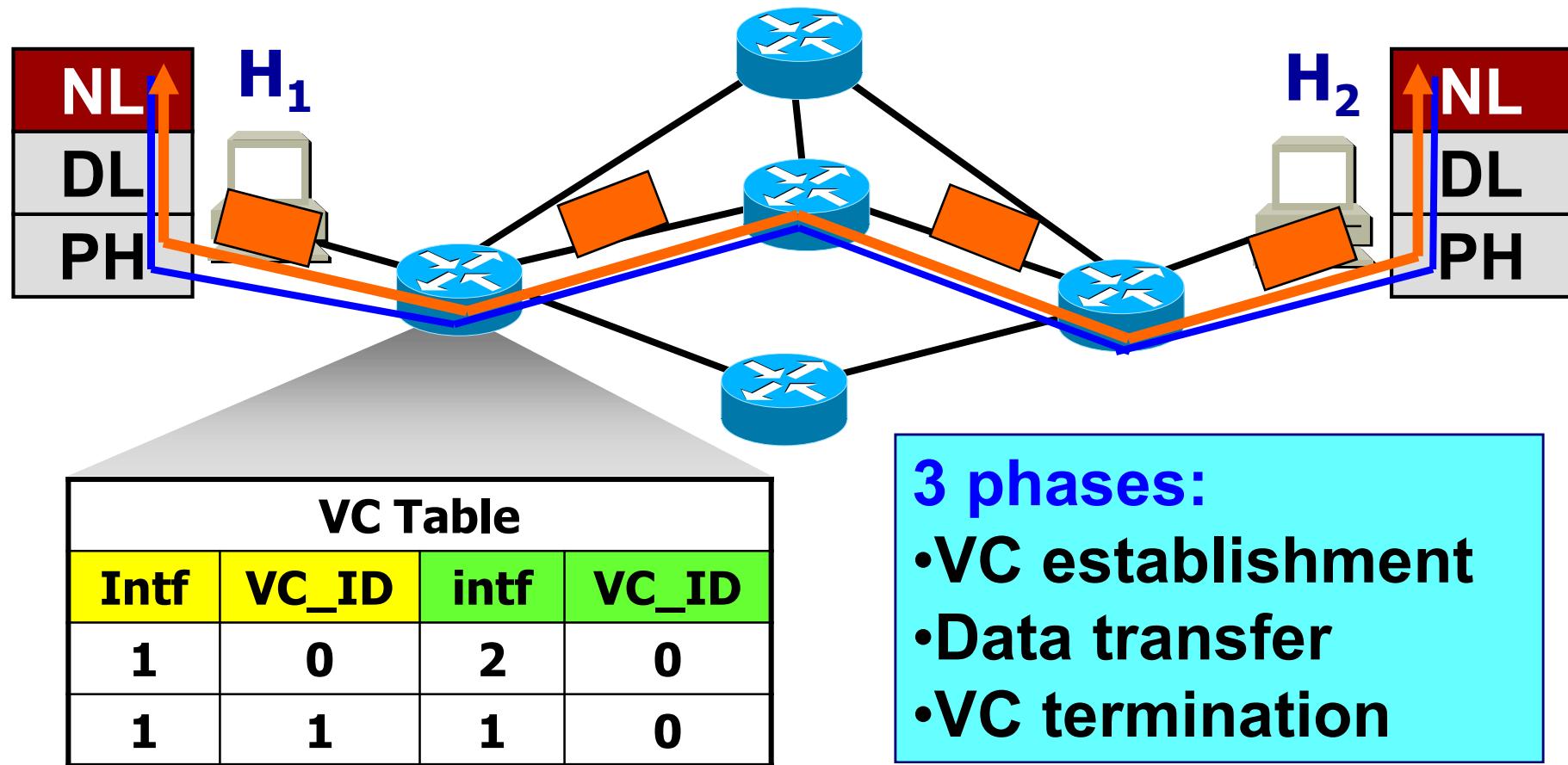
**Example:** IP networks

**Analogy:** Postal service

# Packet Switching

- **Question:** How can data be exchanged between networks?
- **Packet Switching**
  - **Virtual-Circuit Packet Switching**
  - **Datagram Packet Switching**

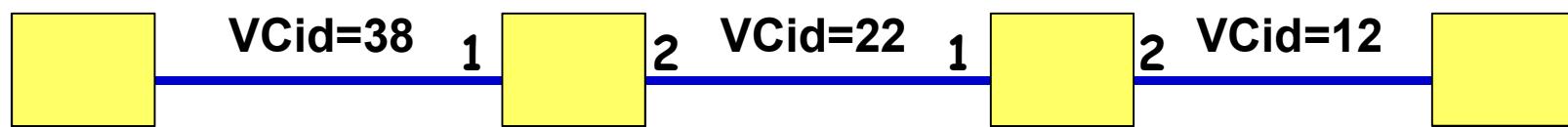
# Virtual Circuit



Each router maintains per-call state in its  
**Virtual Circuit (VC) Table (forwarding table)**

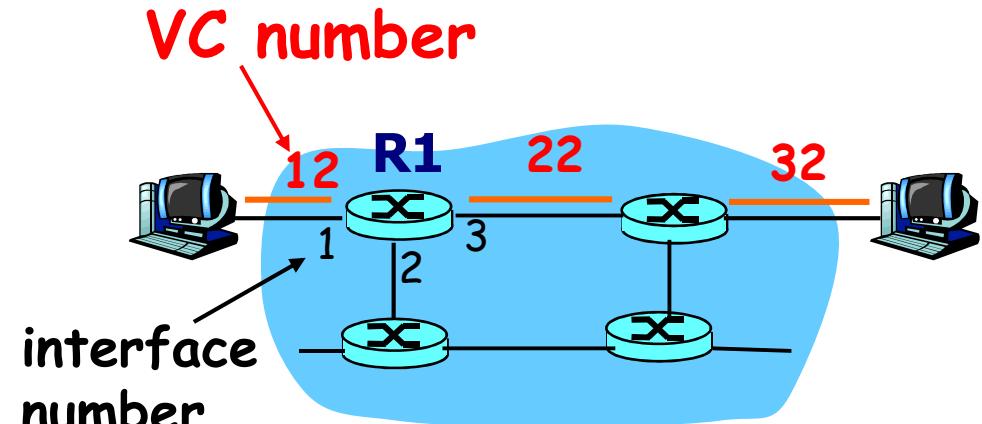
# Virtual Circuit

- A VC consists of:
  1. path from source to destination
  2. VC numbers (VCid), one number for each link along path
  3. entries in VC tables in routers along path
- Packet belonging to VC carries VC number (rather than destination address)
- How to assign VC number?
  - Assigned on each link.



# VC (Forwarding) table

Forwarding table in R1 router:

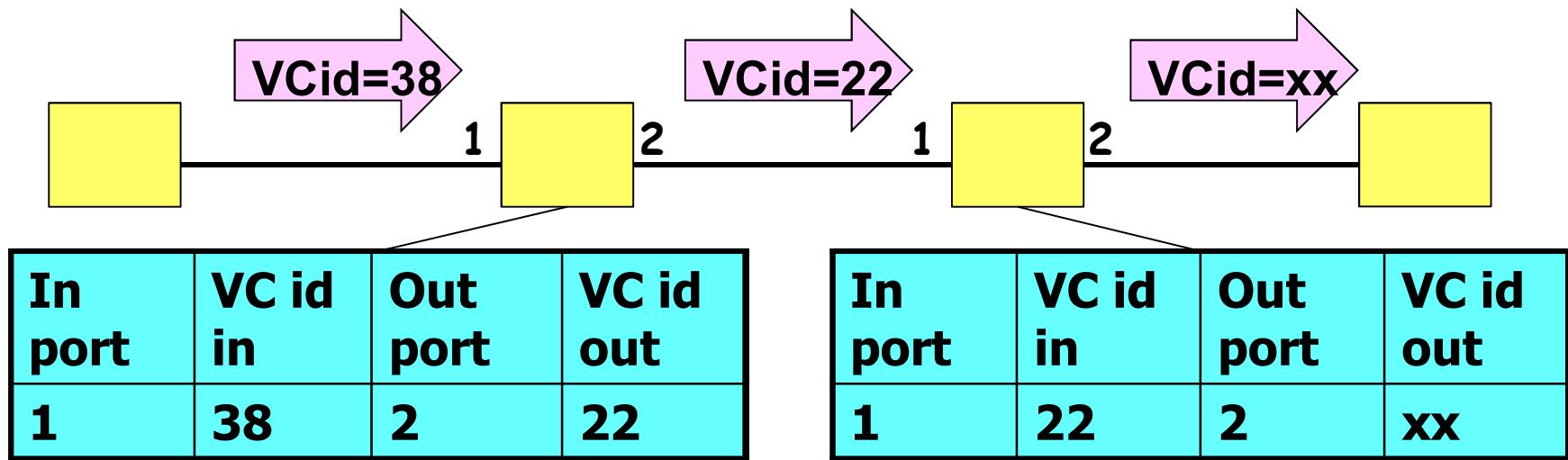


Incoming interface	Incoming VC #	Outgoing interface	Outgoing VC #
1	12	3	22
2	63	1	18
3	7	2	17
1	97	3	87
...	...	...	...

Routers maintain connection state information!

# VC (Forwarding) table

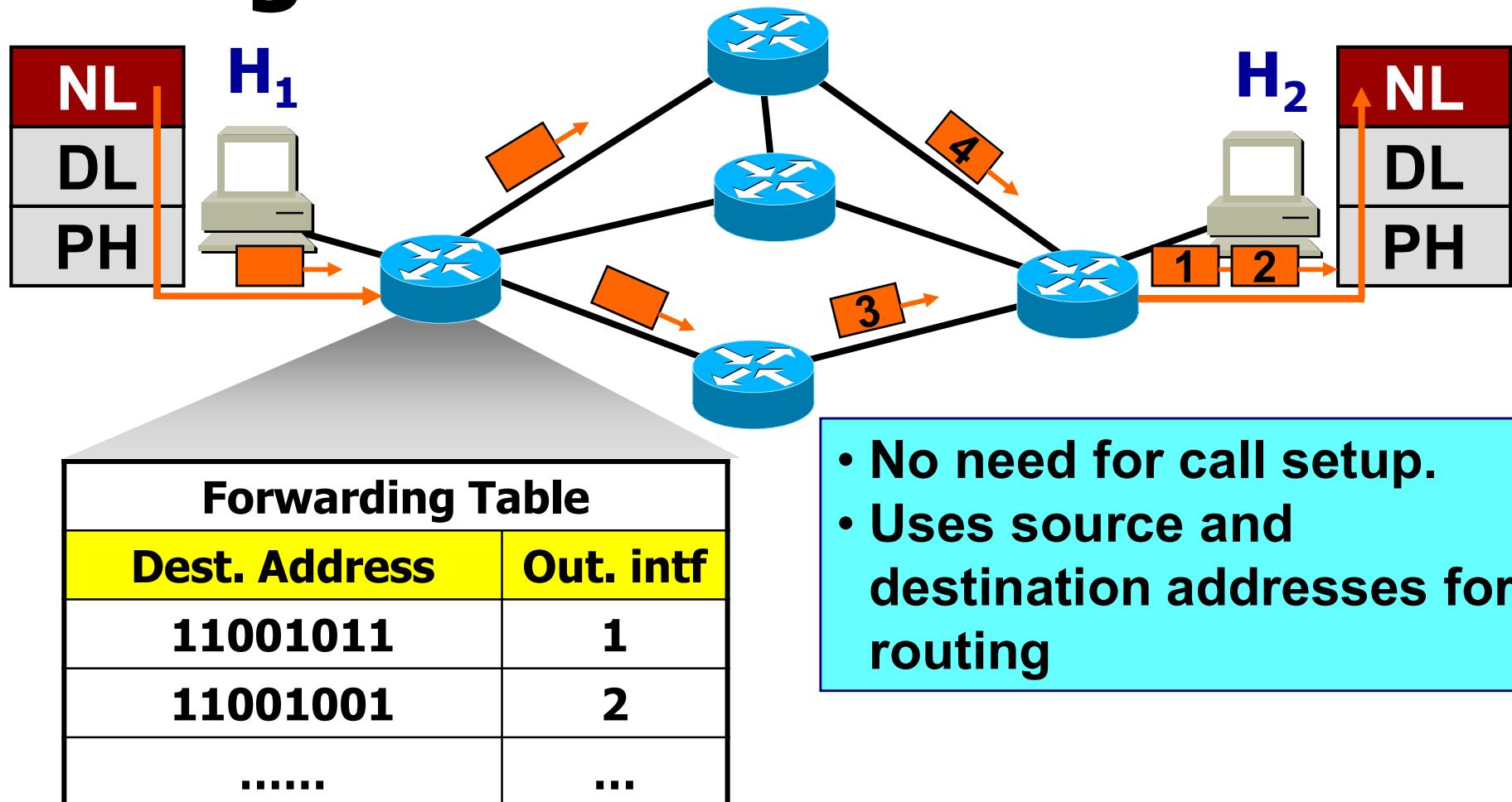
## ■ Example runtime



# Datagram

- Packets in this approach are referred to as **datagrams**.
- Each packet is treated **independently** of all others.
- Datagrams may arrive in **out of order**.
- No need for call setup and virtual circuit identifiers.
- **Uses source and destination addresses for routing.**

# Datagram



routers: no state about end-to-end connections

# Forwarding Table

Destination Address	Output Interface
11001000 00010111 00010	1
11001000 00010111 00011000	2
11001000 00010111 00011	3
<b>otherwise</b>	<b>5</b>

**Examples:**

DA: 11001000 00010111 00010110 10100001

**Which interface?**

DA: 11001000 00010111 00011000 10101010

**Which interface?**

**Solution: Longest prefix matching**

# Comparison of virtual circuit and datagram

Issue	Datagram subnet	Virtual-circuit subnet
Circuit setup	Not needed	Required
Addressing	Each packet contains the full source and destination address	Each packet contains a short VC number
State information	Routers do not hold state information about connections	Each VC requires router table space per connection
Routing	Each packet is routed independently	Route chosen when VC is set up; all packets follow it
Effect of router failures	None, except for packets lost during the crash	All VCs that passed through the failed router are terminated
Quality of service	Difficult	Easy if enough resources can be allocated in advance for each VC
Congestion control	Difficult	Easy if enough resources can be allocated in advance for each VC

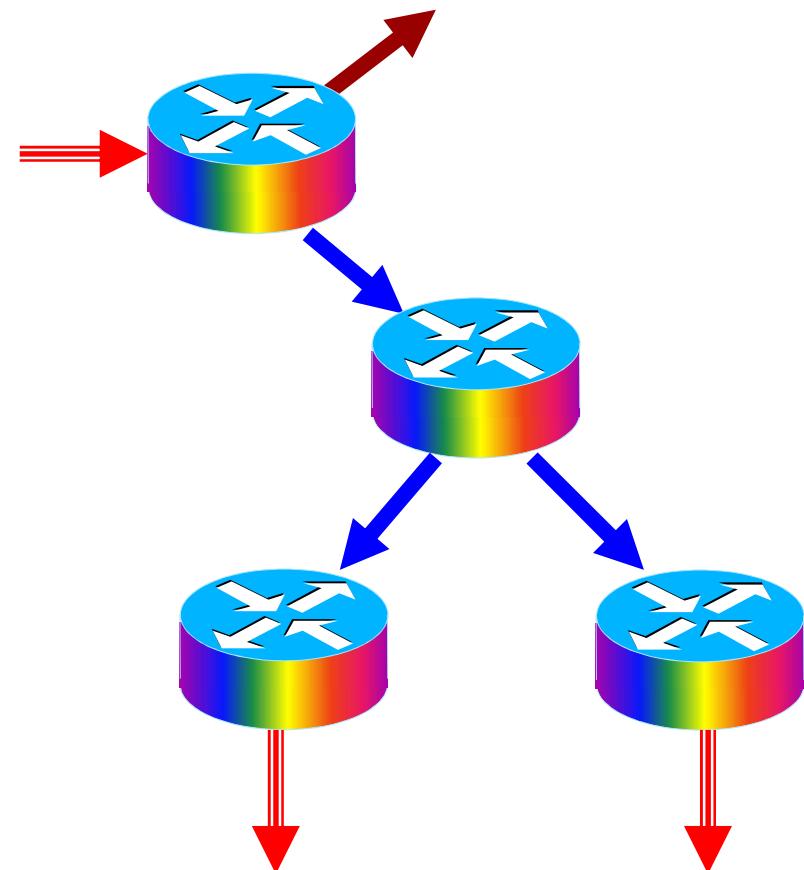
# Chapter 5: Roadmap

- Network Layer and Design Issues
- **Routing Algorithms**
- Congestion Control Algorithm
- IP
- IP Multicasting
- Mobile IP

# Key Network-Layer Functions

- ***Forwarding:***

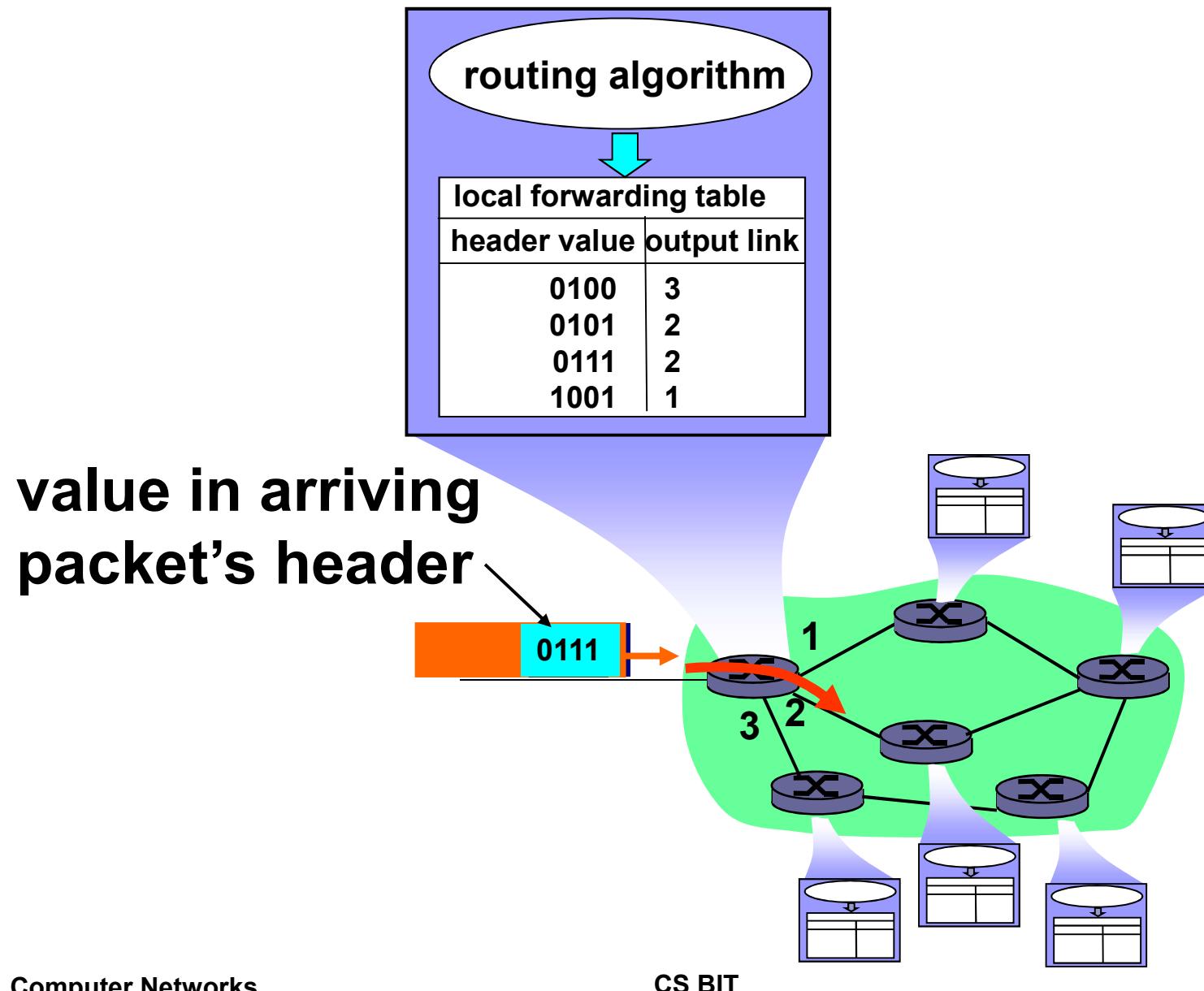
**move packets from router's input to appropriate router output**



- ***Routing:***

**determine route taken by packets from source to dest.**

# Interplay between routing and forwarding



# Routing Algorithms

- **Network Layer:**

**At a given node, it decides which output line an incoming packet should be sent.**

- **Routing algorithms:**

**determine the **route** and maintain the **routing table**.**

- **Goal:**

**determine “**good**” paths (sequences of routers) through network from sources to destination.**

# Routing Algorithms

- **Desired properties for a routing algorithm:**
  - **1. correctness**
  - **2. simplicity**
  - **3. robustness with respect to failures and changing conditions**
  - **4. stability of the routing decisions**
  - **5. fairness of the resource allocation**
  - **6. optimality of the packet travel times**

# Routing Classification

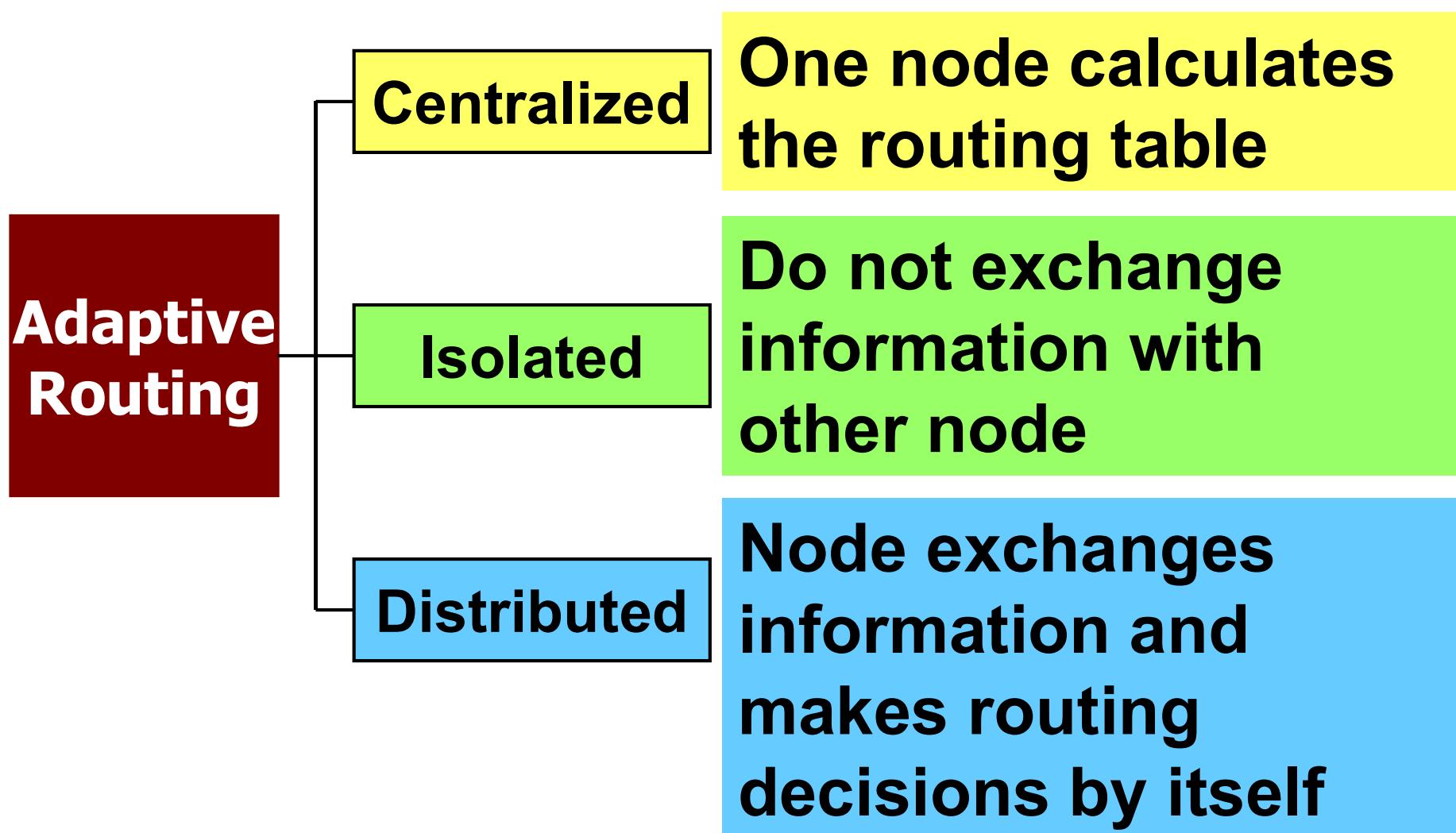
## Non-Adaptive Routing (Static Routing)

- Route is computed in advanced, off-line
- Not depends on current traffic or topology

## Adaptive Routing (Dynamic Routing)

- Route is computed dynamically, on-line
- Based on current traffic and/or topology

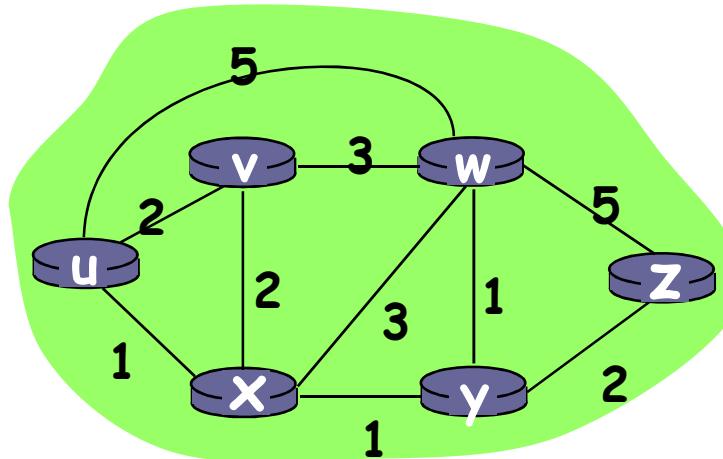
# Routing Classification



# Differences b/w Adaptive and Non-Adaptive Routing Algorithm

Basis Of Comparison	Adaptive Routing algorithm	Non-Adaptive Routing algorithm
Define	Constructs the routing table based on the network conditions.	Constructs the static table to determine which node to send the packet.
Usage	Used by dynamic routing.	Used by static routing.
Routing decision	Routing decisions are made based on topology and network traffic.	Routing decisions are the static tables.
Categorization	The types are Centralized, isolation and distributed algorithm.	The types are flooding and random walks.
Complexity	More complex.	Simple.

# Graph abstraction



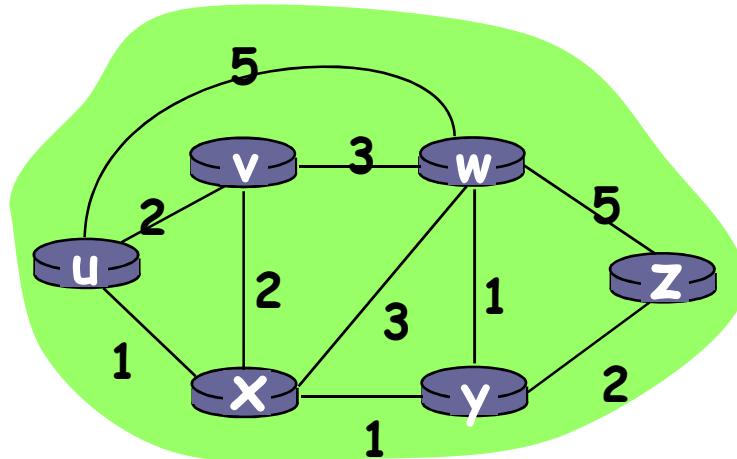
**Graph:  $G = (N, E)$**

**$N = \text{set of routers} = \{ u, v, w, x, y, z \}$**

**$E = \text{set of links} =$**

**$\{ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) \}$**

# Graph abstraction: costs



- $c(x,x') = \text{cost of link } (x,x')$   
- e.g.,  $c(w,z) = 5$
- cost could always be 1, or inversely related to bandwidth, or inversely related to congestion

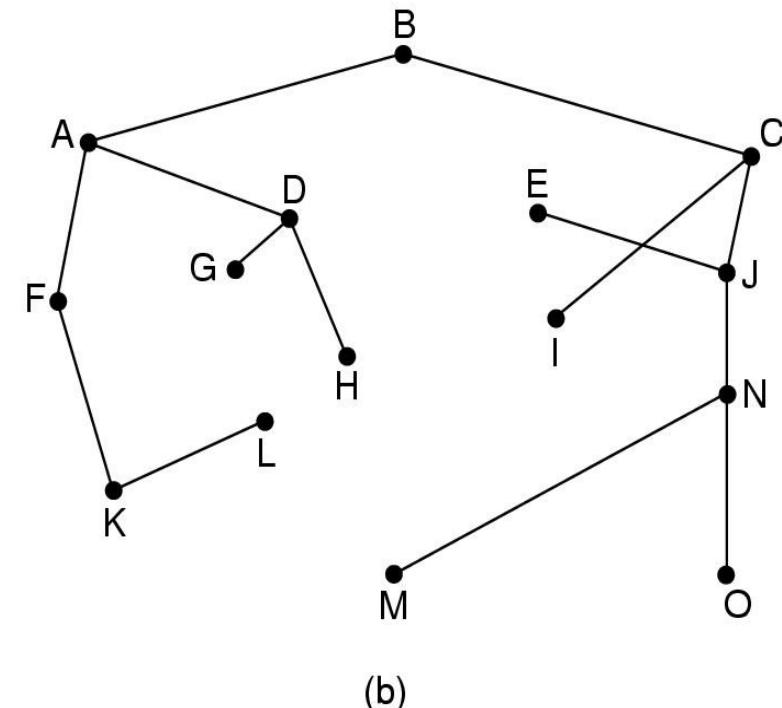
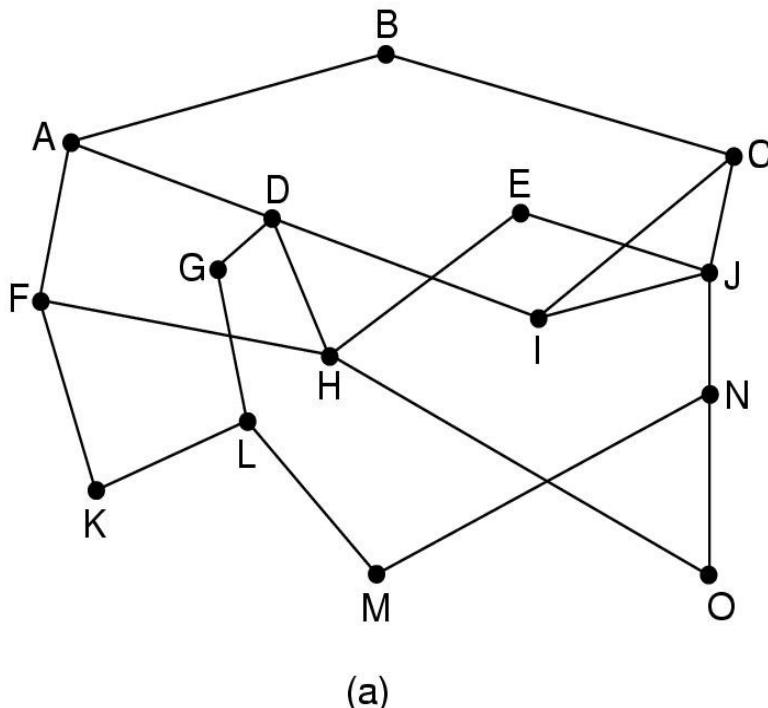
Cost of path  $(x_1, x_2, x_3, \dots, x_p) = c(x_1, x_2) + c(x_2, x_3) + \dots + c(x_{p-1}, x_p)$

**Question:** What's the **least-cost** path between u and z ?

**Routing algorithm:** algorithm that finds least-cost path

# Sink Tree

- The set of **optimal routes** from source to given destinations forms a tree: **sink tree**.



# Routing Algorithms

## ■ Static Algorithms

- Shortest Path Routing

- Flooding

- Flow based Routing

## ■ Dynamic Algorithms

- Distance Vector Routing

- Link State Routing

- Hierarchical Routing

# Shortest Path Routing

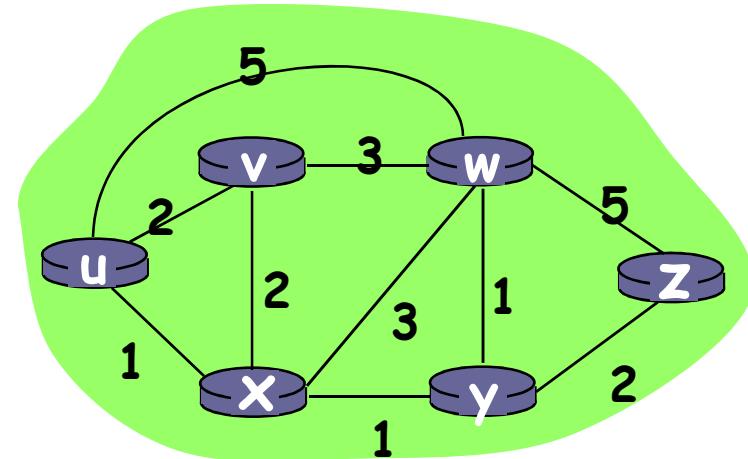
- A non-adaptive routing algorithm
- Given a network topology and a set of weights describing the cost to send data across each link in the network
- Find the shortest path from a specified source to all other destinations in the network.
- Shortest path algorithm first developed by E. W. Dijkstra

# Dijkstra's Algorithm

- Find shortest paths from given source node to all other nodes, by developing paths in order of increasing path length
- $N$  = set of nodes in the network
- $s$  = source node
- $T$  = set of nodes so far incorporated by the algorithm
- $w(i, j)$  = link cost from node  $i$  to node  $j$ 
  - $w(i, i) = 0$
  - $w(i, j) = \infty$  if the two nodes are not directly connected
  - $w(i, j) \geq 0$  if the two nodes are directly connected
- $L(n)$  = cost of least-cost path from node  $s$  to node  $n$  currently known
  - At termination,  $L(n)$  is cost of least-cost path from  $s$  to  $n$

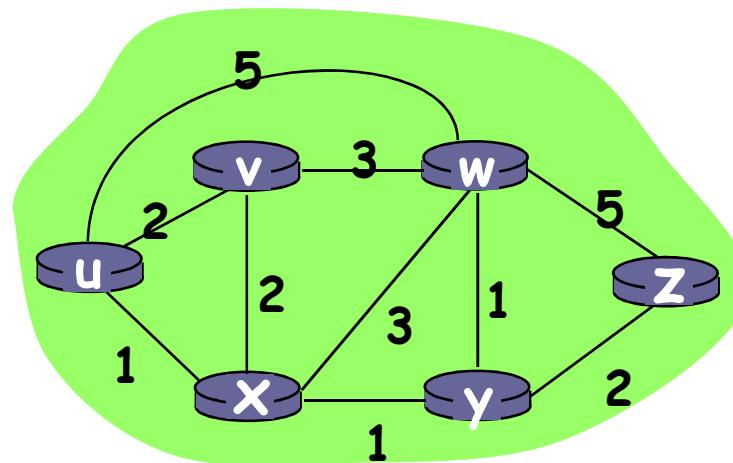
# Dijsktra's Algorithm

```
1 Initialization:
2   T = {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 T such that D(w) is a minimum
10  add w to T
11  update D(v) for all v adjacent to w and not in T :
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 T
```



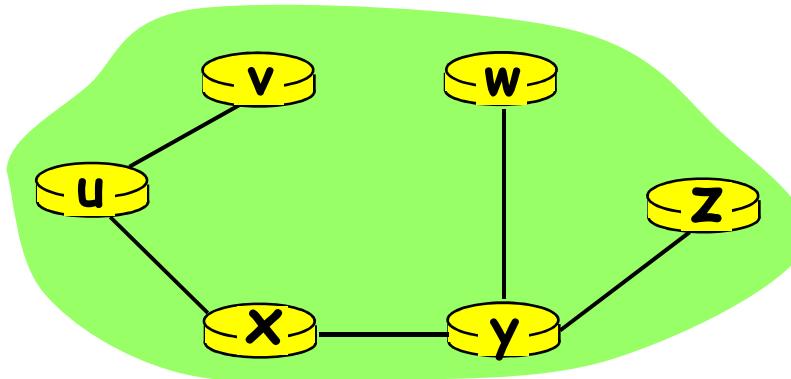
# Dijkstra's algorithm: example

Step	T	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0	u	2, u	5, u	1, u	$\infty$	$\infty$
1	ux	2, u	4, x		2, x	$\infty$
2	uxy	2, u	3, y			4, y
3	uxyv		3, y			4, y
4	uxywv					4, y
5	uxyvwz					



# Dijkstra's algorithm: example (2)

Resulting shortest-path tree from u:



Resulting routing table in u:

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

# Dijkstra's Algorithm

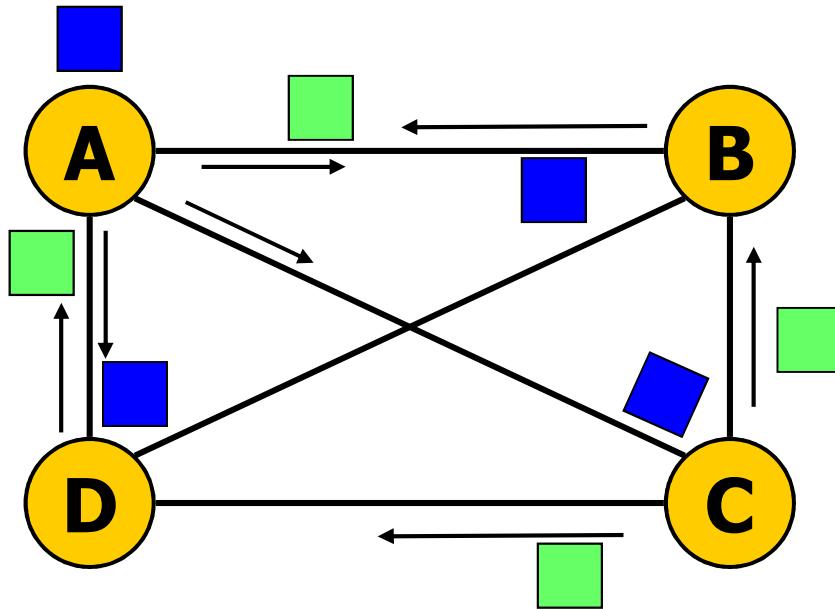
Illustration of  
Dijkstra's algorithm  
search for finding  
path from a start  
node (lower left, red)  
to a goal node  
(upper right, green)  
in a **robot motion  
planning** problem.



# Flooding

- A non-adaptive routing algorithm
- No network info required
- Packet sent by node to every neighbor
  - Every incoming packet is sent out on every outgoing link except the one it arrived on.
- Termination of flooding process
  - Hop counter
  - Record of packet which has been flooded
- Applications of flooding
  - Military application
  - Comparison

# Flooding



**Packet sent by node to every neighbor.**

**Problems:**

- Loops
- Broadcast storms

# Distance Vector Routing

- A distributed routing algorithm
- Basis of RIP, IGRP, EIGRP routing protocols
- Based on the Bellman-Ford algorithm
- Basic idea:  
each network node maintains a Distance Vector table containing the *distance* between itself and ALL possible destination nodes.

# Distance Vector Routing

## Distance Table data structure

Destination	Neighbor (next hop)	Distance
W	X	5

- each node has its own
- row for each possible destination
- column for each directly-attached neighbor to node

# Distance Vector Routing

- To find D, node S asks each neighbor X
  - How far X is from D
  - X asks its neighbors ... comes back and says  $C(X,D)$
  - Node S deduces  $C(S,D) = C(S,X) + C(X,D)$
  - S chooses neighbor  $X_i$  that provides min  $C(S,D)$
- Later,  $X_j$  may find better route to D
  - $X_j$  advertizes  $C(X_j,D)$
  - All nodes update their cost to D if new min found

# Distance Vector Routing

- Define

$d_x(y) := \text{cost of least-cost path from } x \text{ to } y$

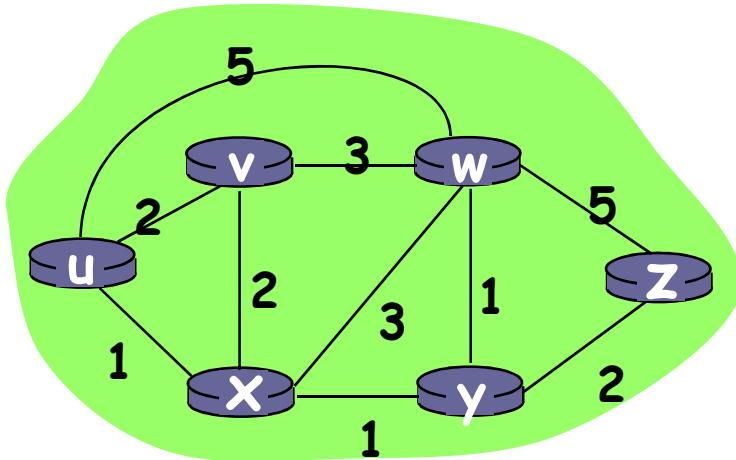
- Then

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

(Bellman-Ford Equation)

where  $\min_v$  is taken over all neighbors v of x.

# Example



Clearly,

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

B-F equation says:

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

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

# Distance Vector Algorithm

## Algorithm:

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

$$D_x(y) \leftarrow \min_v \{ c(x,v) + D_v(y) \}$$

*for each node  $y \in N$*

# Distance Vector Algorithm

**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:**

*wait* for (change in local link cost or msg from neighbor)

*recompute* estimates

if DV to any dest has changed, *notify* neighbors

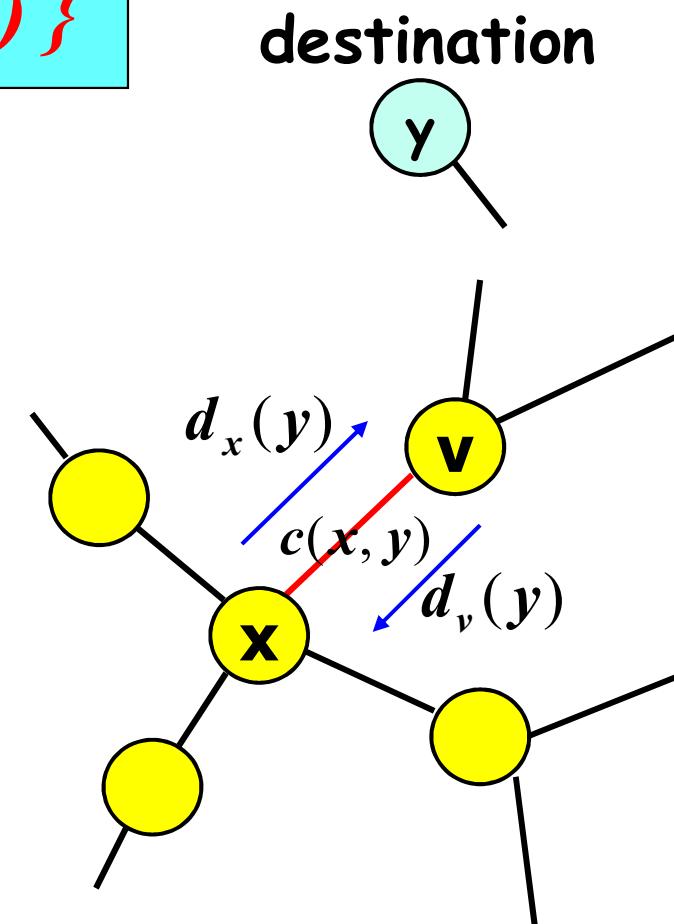
# Distance Vector Algorithm

- At node  $x$ , the basic **update rule**

$$D_x(y) \leftarrow \min_v \{ c(x,v) + D_v(y) \}$$

Where:

- $d_x(y)$  denotes the distance estimation from  $x$  to the destination.
- $\min_v$  is taken over all neighbors  $v$  of  $x$ .
- $c(x, v)$  is the distance of the direct link from  $x$  to  $v$ ; assume positive.



# Distance Vector Algorithm

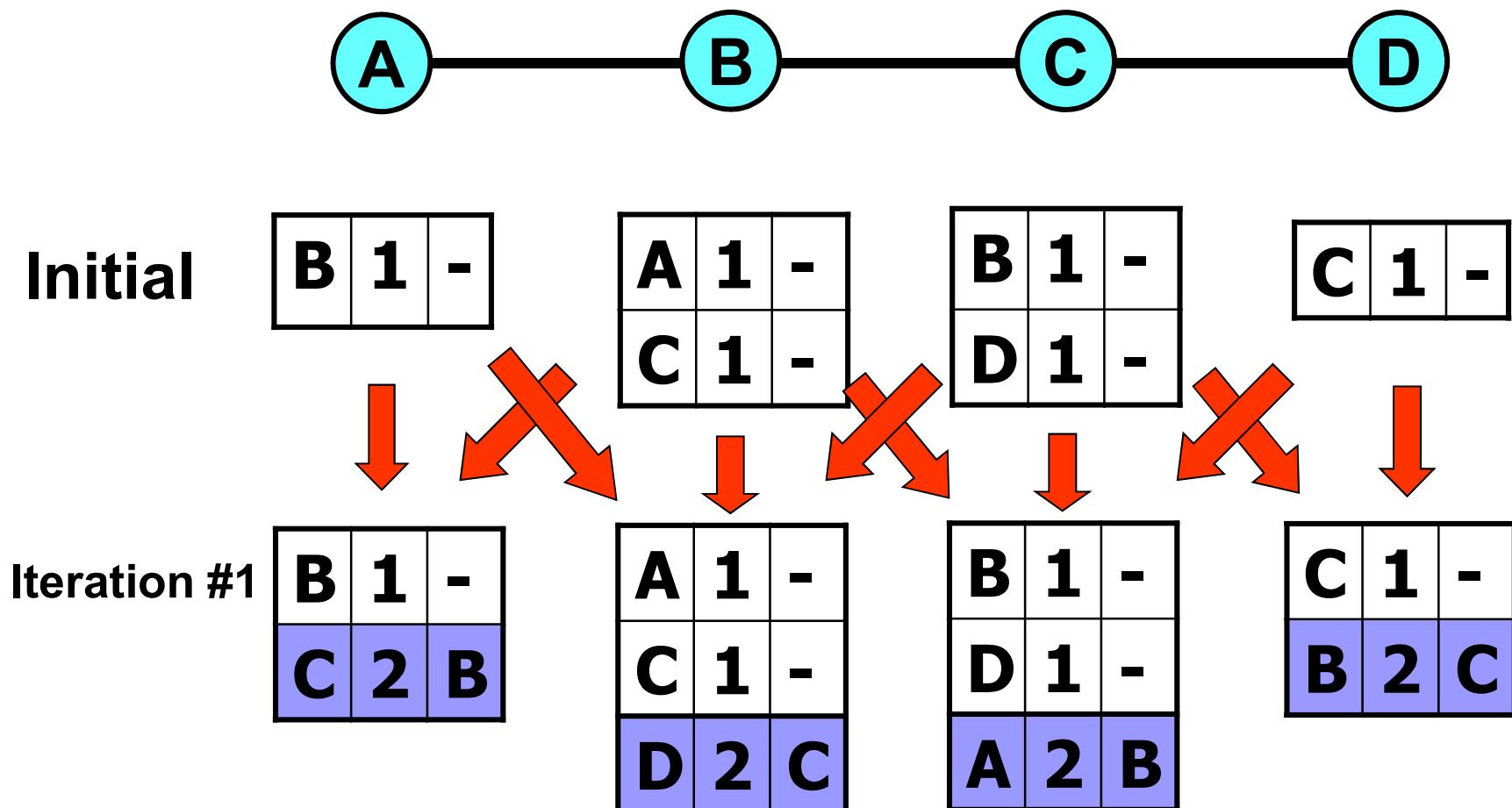
**At all nodes, X:**

- 1 **Initialization:**
- 2 **for all adjacent nodes v:**
- 3    $D_X(*,v) = \text{infty}$     /\* **the \* operator means "for all rows"** \*/
- 4    $D_X(v,v) = c(X,v)$
- 5 **for all destinations, y**
- 6   **send  $\min_w D_X(y,w)$  to each neighbor /\* w over all X's neighbors \*/**

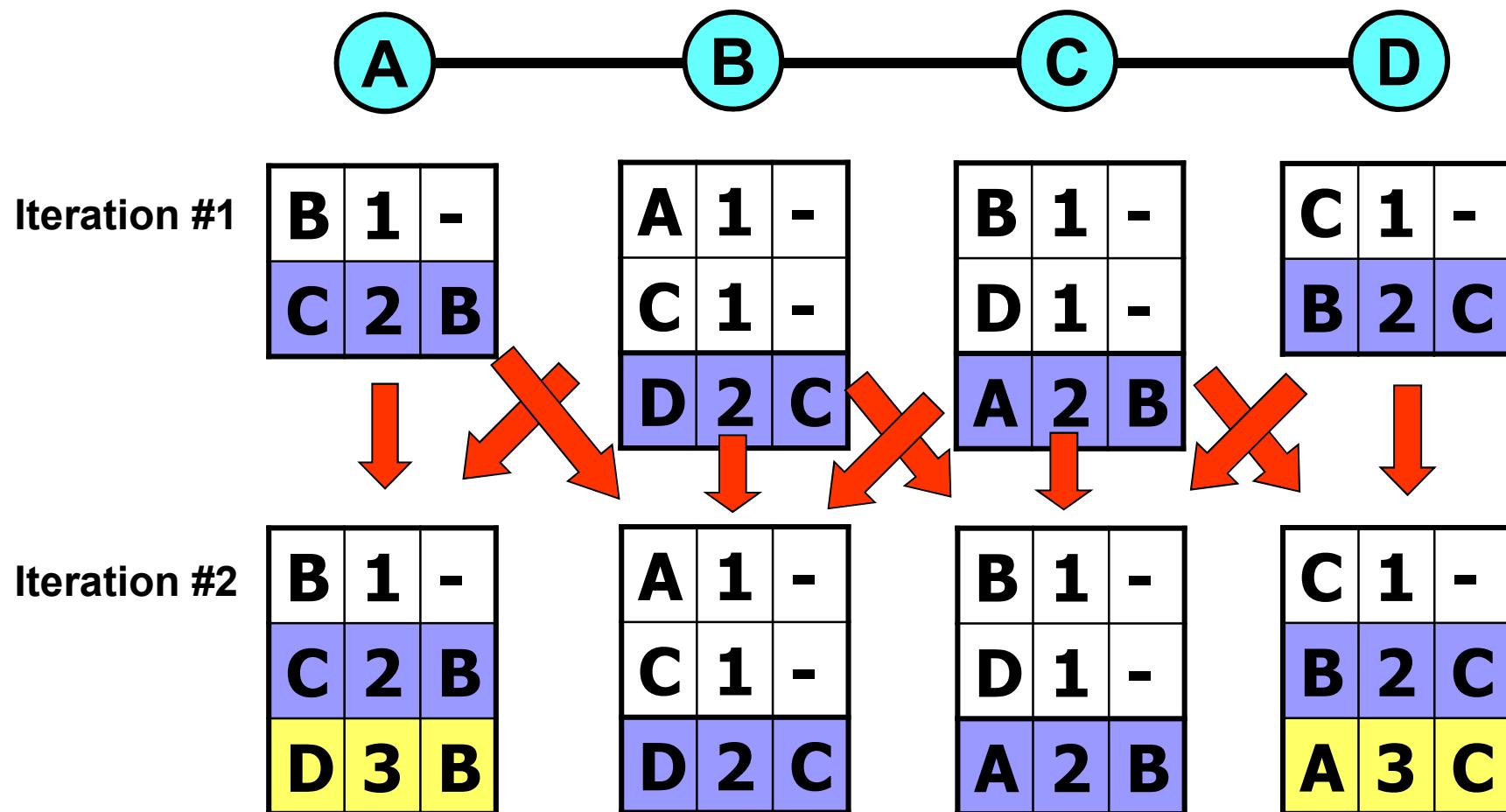
# Distance Vector Algorithm (cont.)

```
8 loop
9 wait (until a link cost change to neighbor V
10 or until receive update from neighbor V)
11
12 if (c(X,V) changes by d)
13 /* change cost to all dest's via neighbor v by d */
14 /* note: d could be positive or negative */
15 for all destinations y:  $D_X(y,V) = D_X(y,V) + d$ 
16
17 else if (update received from V wrt destination Y)
18 /* shortest path from V to some Y has changed */
19 /* V has sent a new value for its  $\min_w D^V(Y,w)$  */
20 /* call this received new value is "newval" */
21 for the single destination y:  $D_X(Y,V) = c(X,V) + newval$ 
22
23 if a new  $\min_w D_X(Y,w)$  for any destination Y
24 send new value of  $\min_w D^X(Y,w)$  to all neighbors
25
26 forever
```

# Example 1

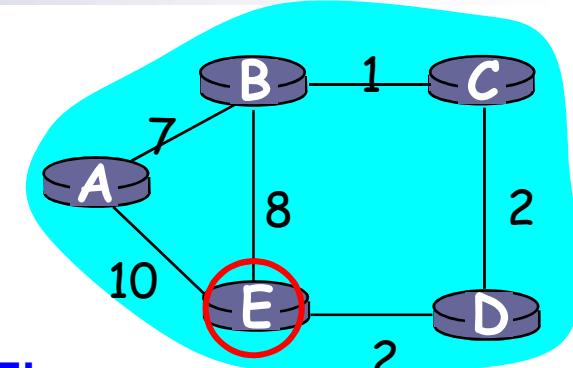


# Example 1



# Example 2

Below is just one step!  
The algorithm repeats forever!



distance tables from neighbors				computation			E's distance table	distance table E sends to its neighbors
d <sub>E()</sub>	A	B	D	A	B	D		
destinations	0	7	$\infty$	10	15	$\infty$	A: 10	A: 10
A	0	7	$\infty$	10	15	$\infty$	A: 10	A: 10
B	7	0	$\infty$	17	8	$\infty$	B: 8	B: 8
C	$\infty$	1	2	$\infty$	9	4	D: 4	C: 4
D	$\infty$	$\infty$	0	$\infty$	$\infty$	2	D: 2	D: 2
	10	8	2				E: 0	

# Example 3

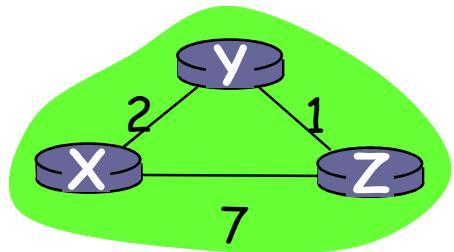
Dest.	Cost	Cost	Cost
A	2	3	1
B	0	3	2
C	3	0	2
D	2	2	0
E	3	1	1
F	5	3	3

B'  
DV    C'  
DV    D'  
DV

Dest.	Cost	Nex hop	Cost	Next hop
A	0	-	0	-
B	2	B	2	B
C	5	C	3	D
D	1	D	1	D
E			2	D
F	8	C	4	D

A' DV  
A' DV  
updated

# Example 4



	X	cost via Y	Z
D			
d			
e	2	∞	
s	∞		
t		7	

	Y	cost via X	Z
D			
d			
e	2	∞	
s	∞		
t		1	

	Z	cost via X	Y
D			
d			
e	7	∞	
s	∞		
t		1	

Initially

	X	cost via Y	Z
D			
d			
e	2	8	
s	3	7	
t			

$$D^X(Y, Z) = c(X, Z) + \min_w \{D^Z(Y, w)\}$$
  

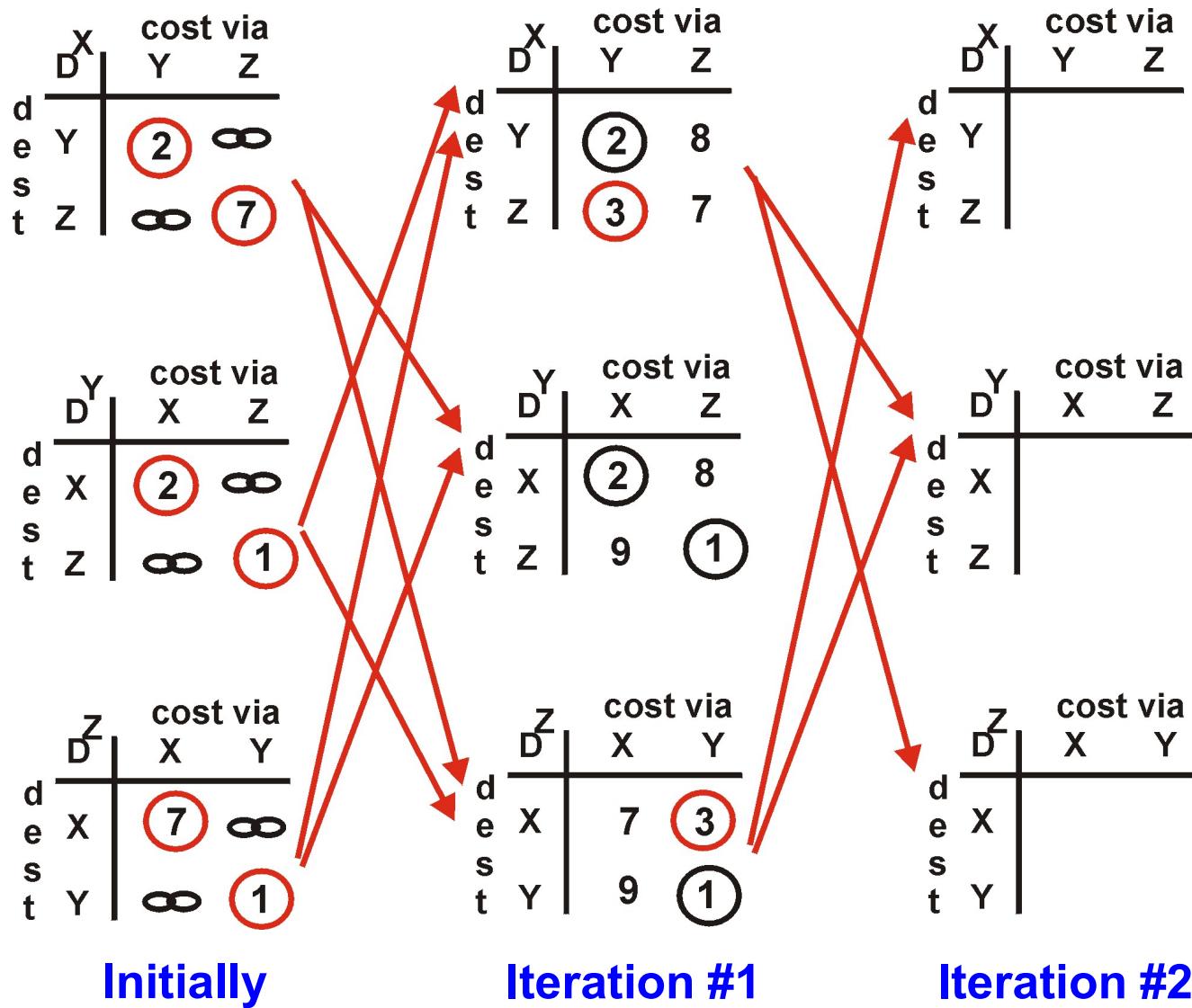
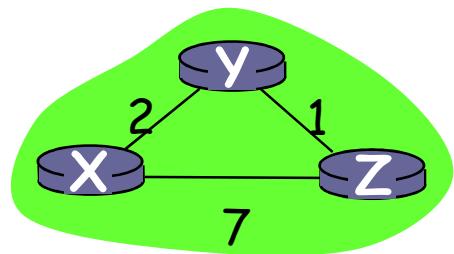
$$= 7 + 1 = 8$$

$$D^X(Z, Y) = c(X, Y) + \min_w \{D^Y(Z, w)\}$$
  

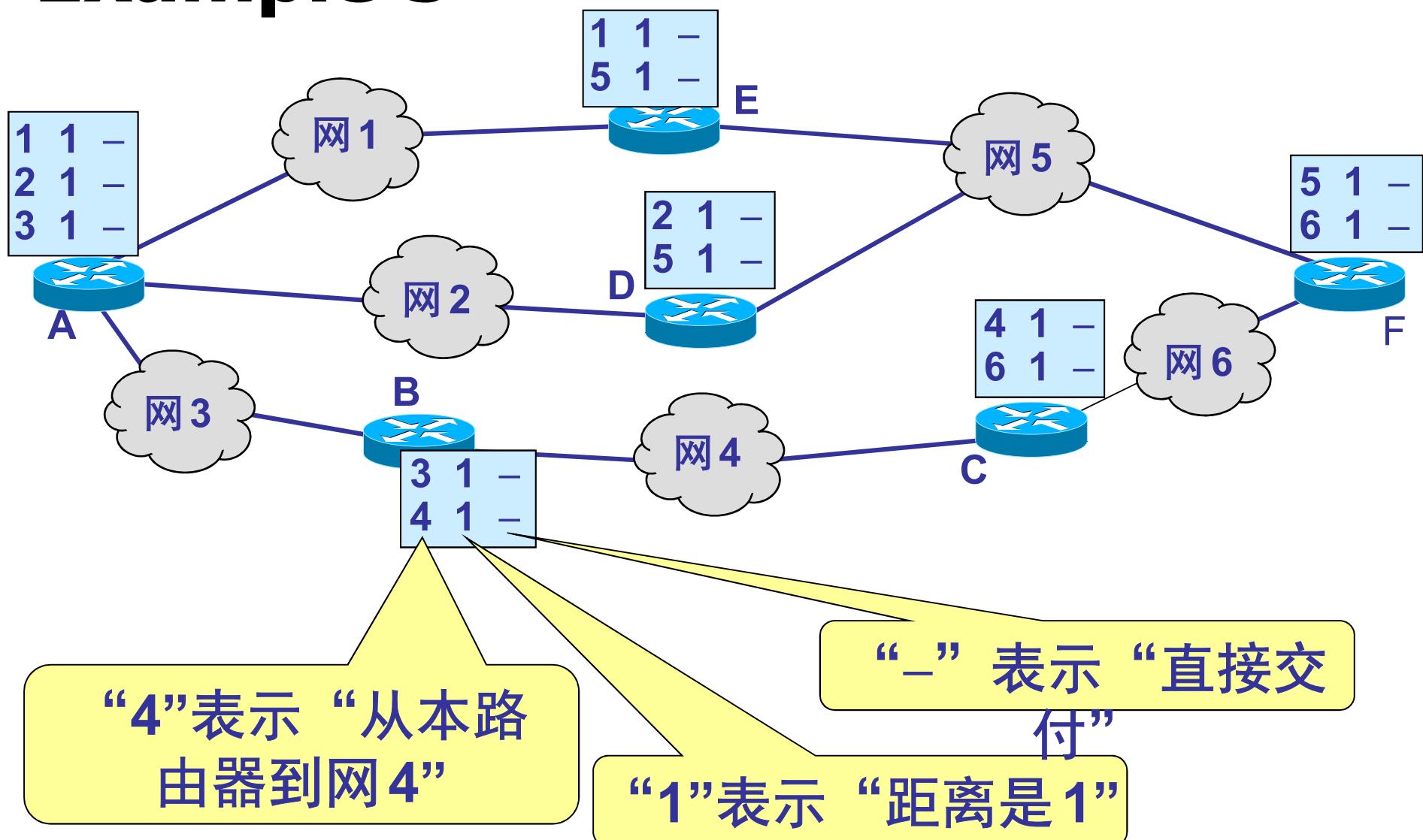
$$= 2 + 1 = 3$$

Iteration #1

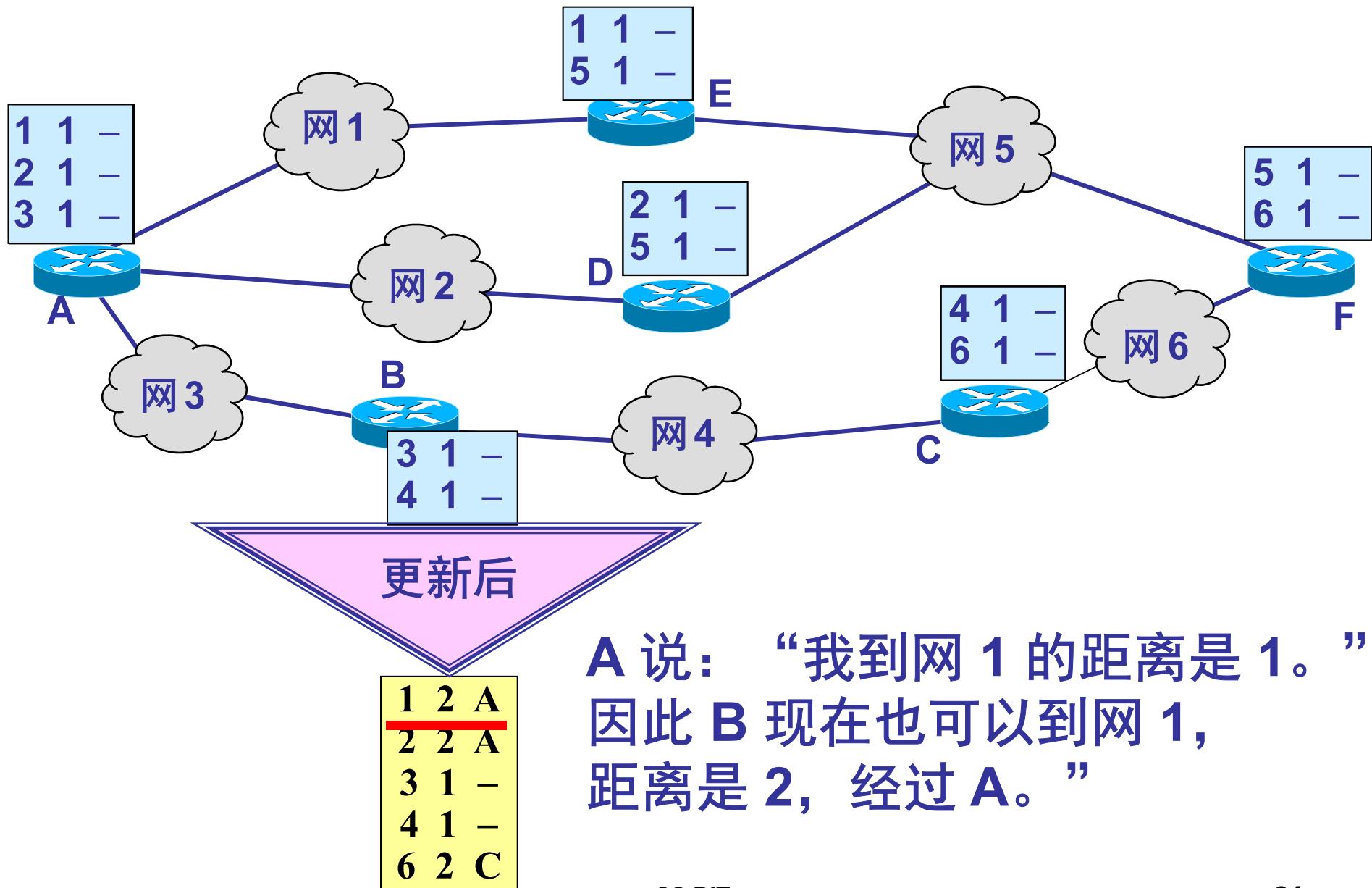
# Example 4



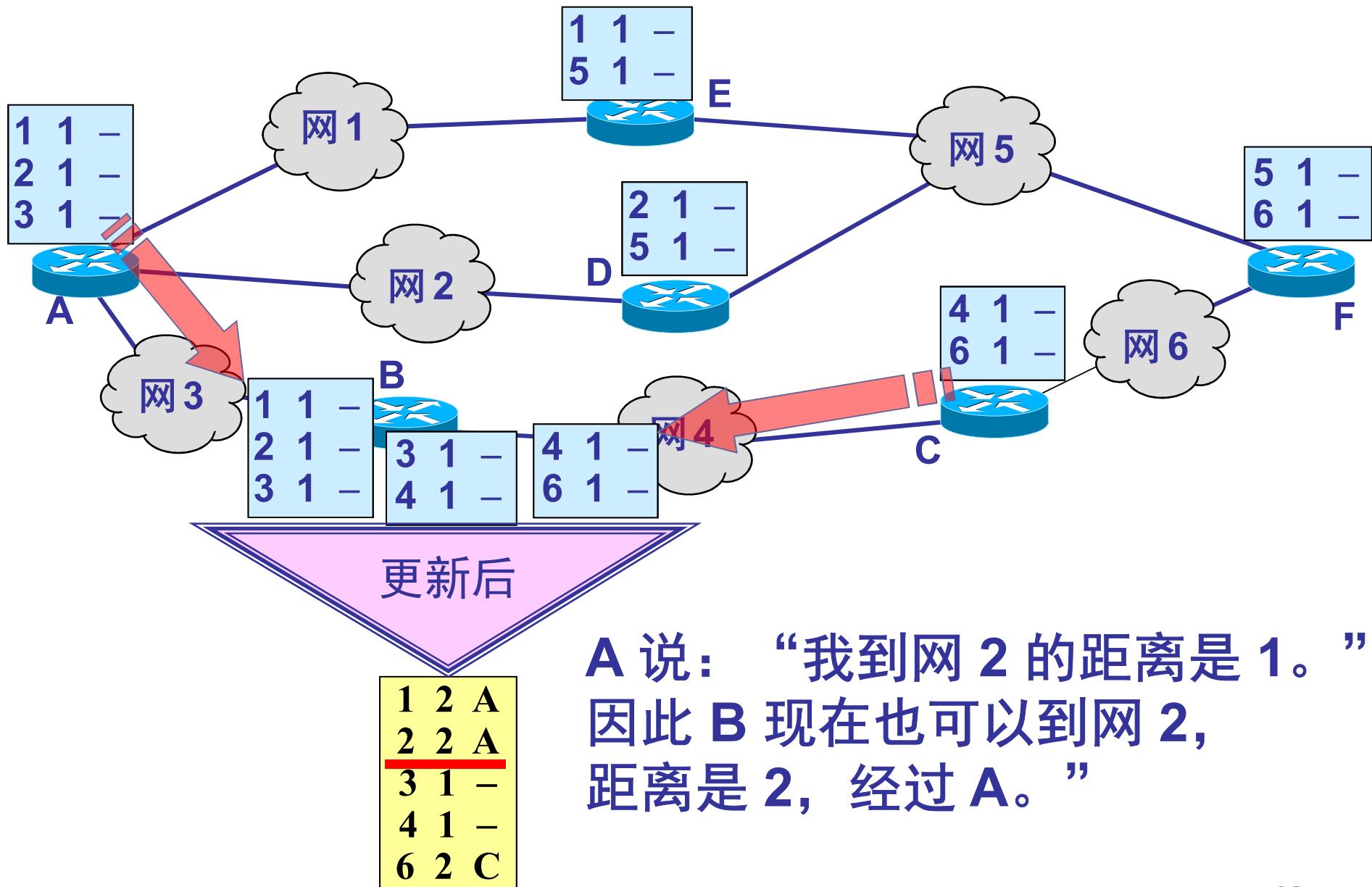
# Example 5



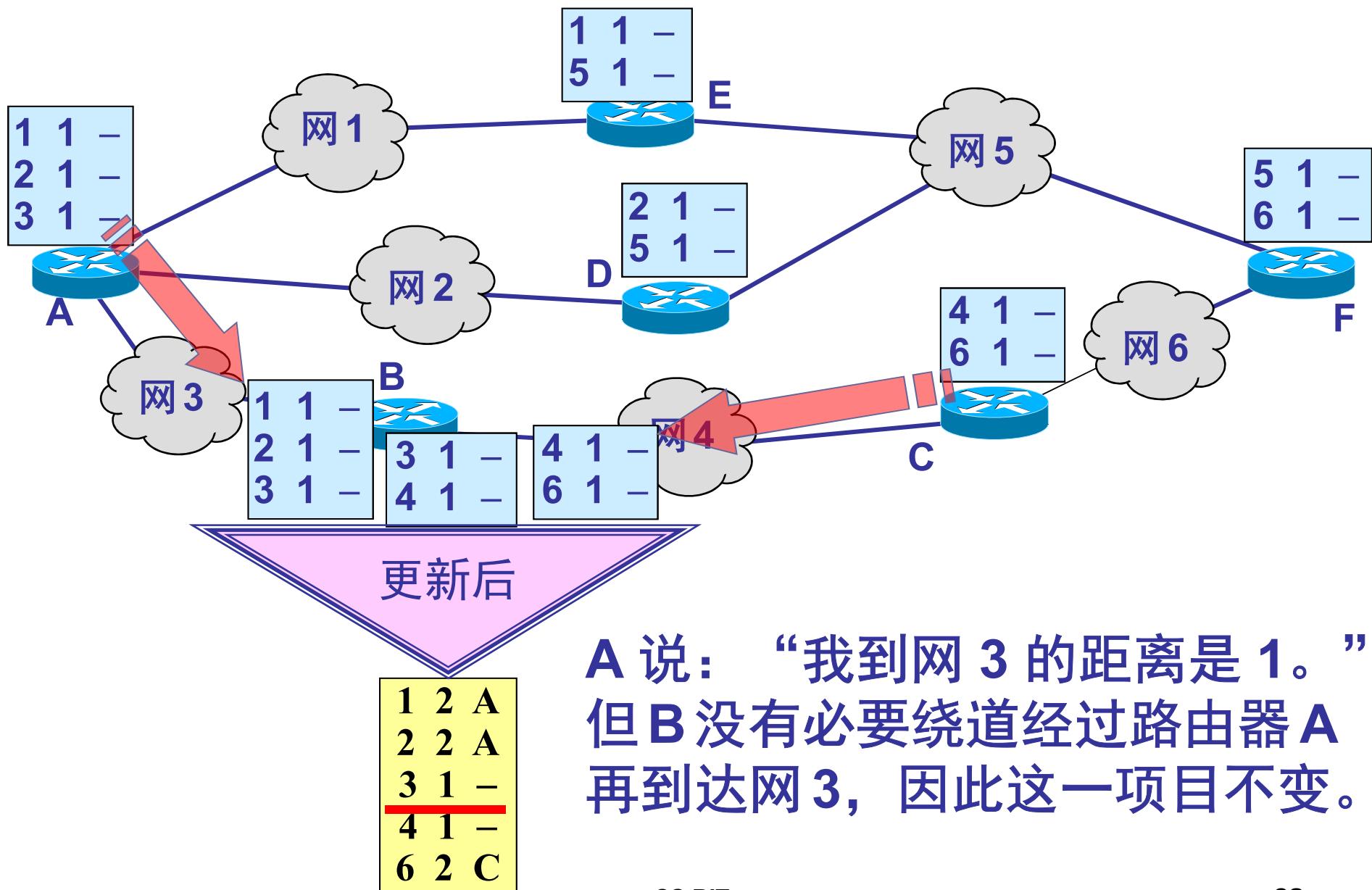
# 路由器 B 收到相邻路由器 A 和 C 的路由表



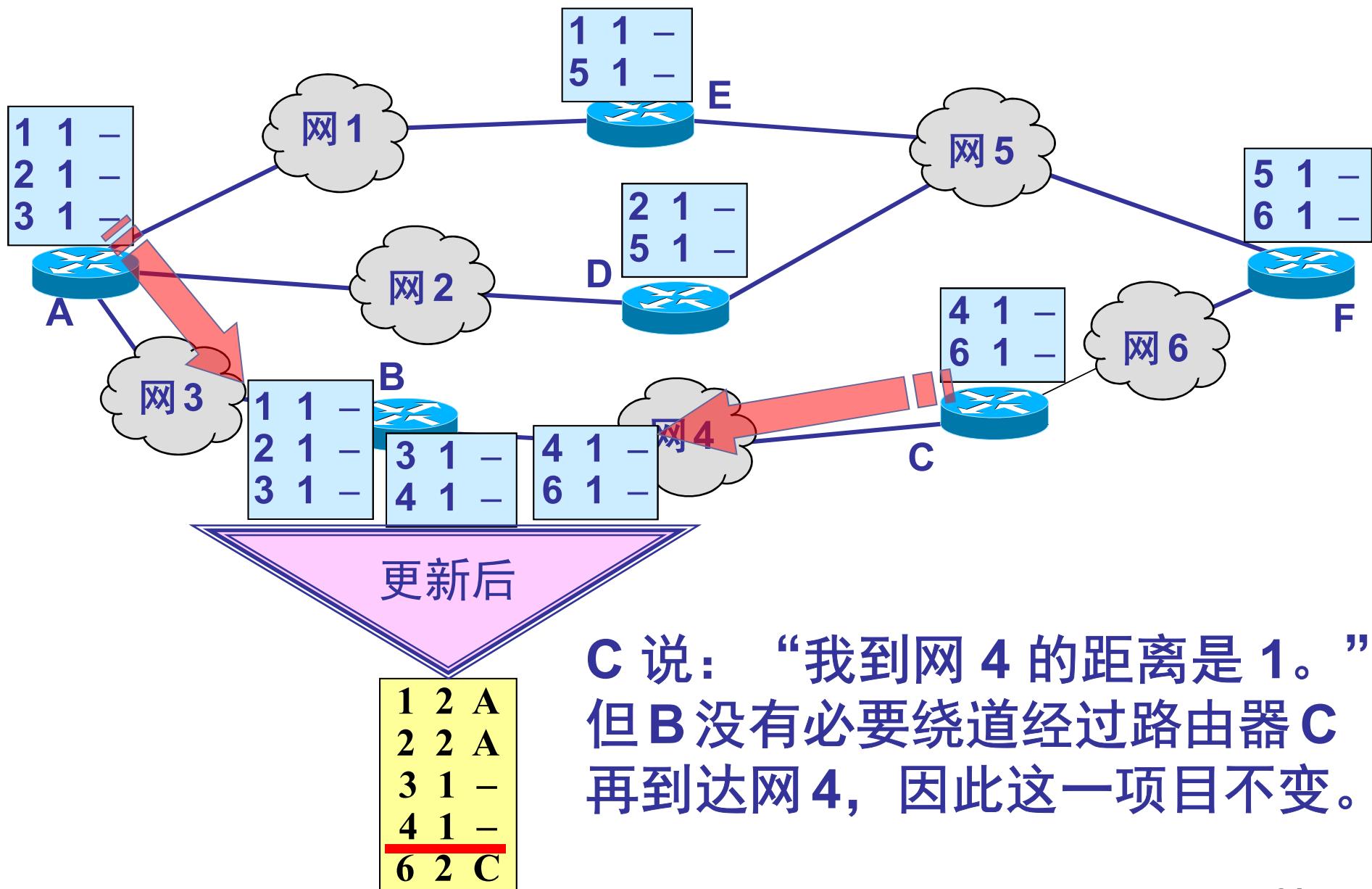
# 路由器 B 收到相邻路由器 A 和 C 的路由表



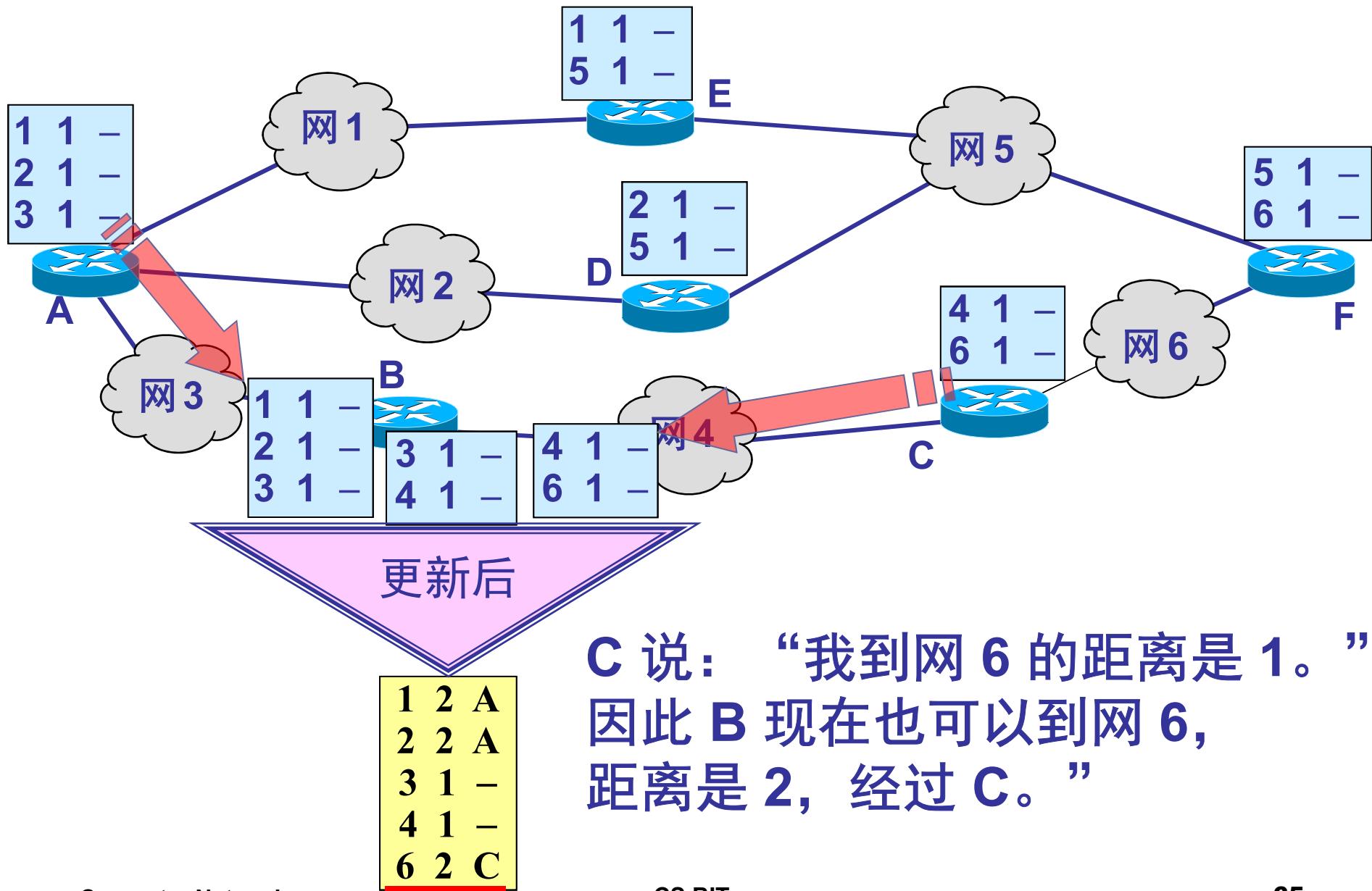
# 路由器 B 收到相邻路由器 A 和 C 的路由表



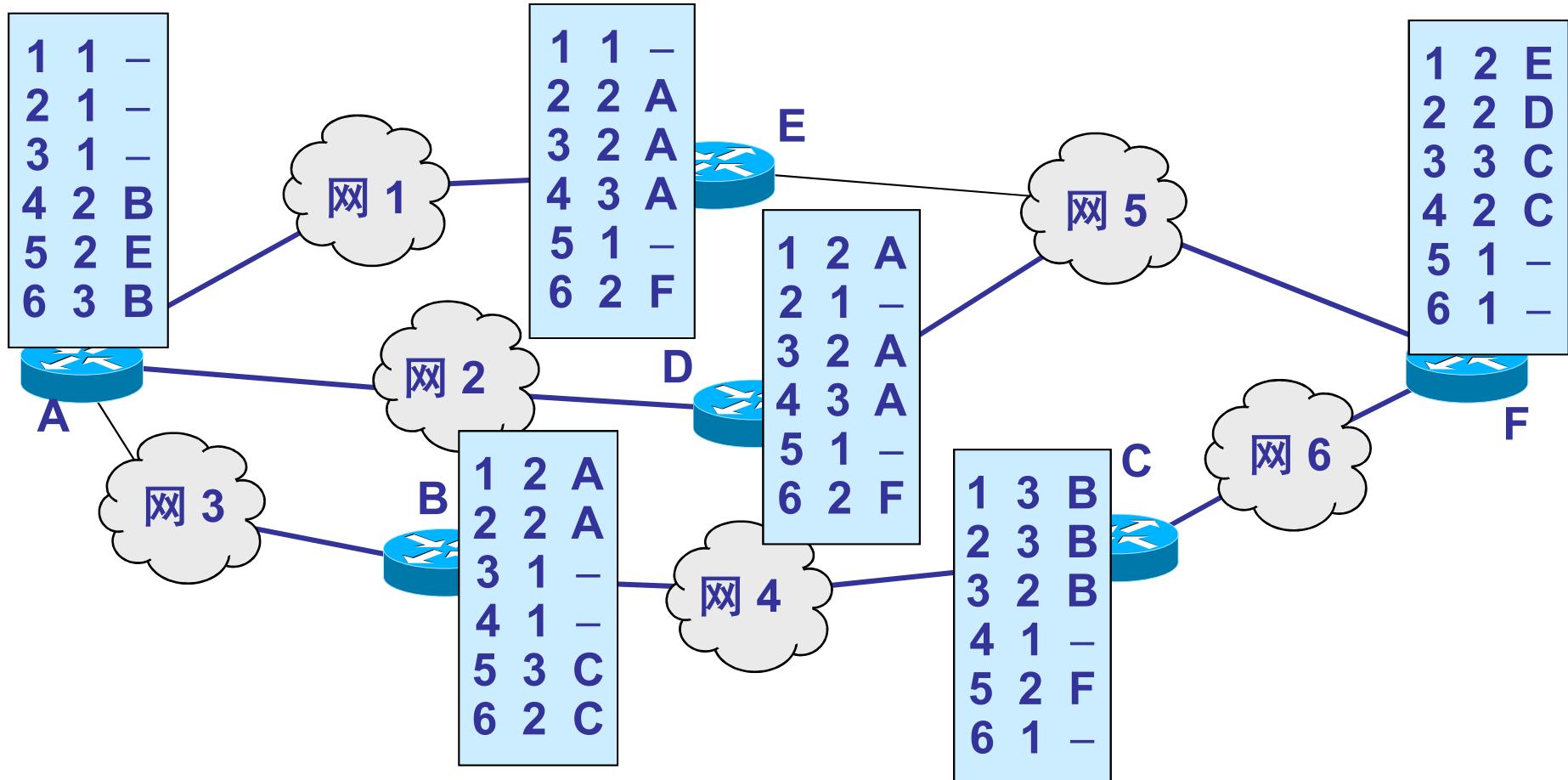
# 路由器 B 收到相邻路由器 A 和 C 的路由表



# 路由器 B 收到相邻路由器 A 和 C 的路由表



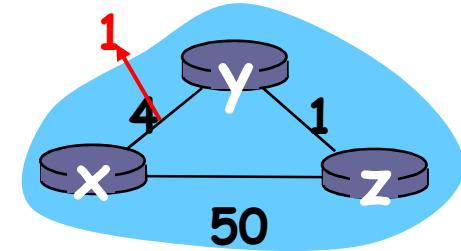
# 最终所有的路由器的路由表都更新了



# Distance Vector: link cost changes

## Link cost changes:

- if DV changes, notify neighbors



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.

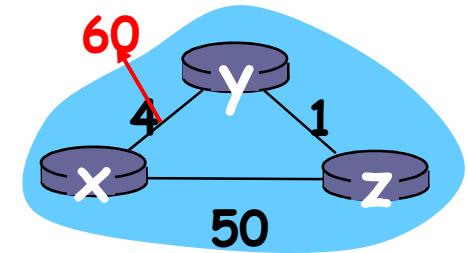
At time  $t_2$ , **y** receives **z**'s update and updates its distance table. **y**'s least costs do not change and hence **y** does not send any message to **z**.

**Q: When can it get complicated ?**

# Distance Vector: link cost changes

## Link cost changes:

- Y thinks Z's best cost is 5
- Thus  $C(y,x) = 5 + 1 = 6$
- Announces this cost
- Z thinks  $C(z,x) = 6 + 1 \dots$



**Will this converge ?**

**If so, after how many rounds ?**

**How can this be solved?**

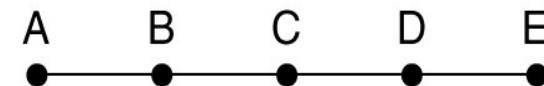
**Should Y announce change from 4 to 60?**

# Problems with distance vector



• Initially  
1 • • • After 1 exchange  
1 2 • • After 2 exchanges  
1 2 3 • After 3 exchanges  
1 2 3 4 After 4 exchanges

(a)



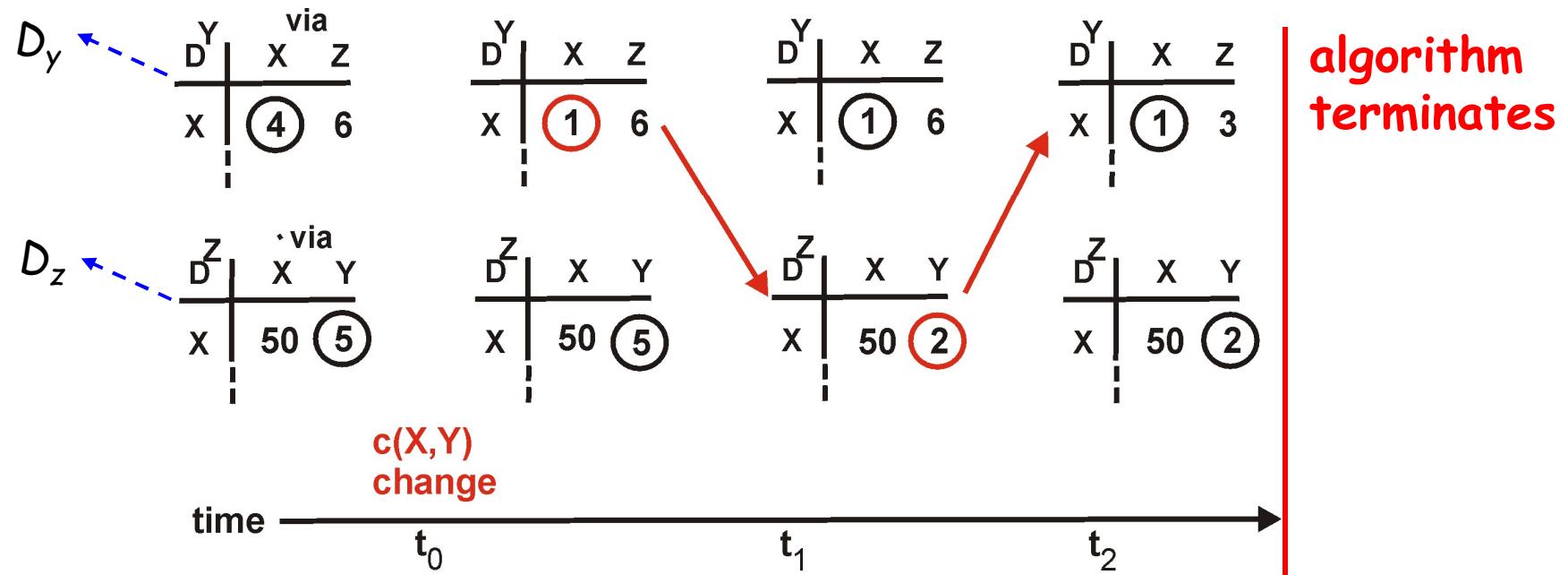
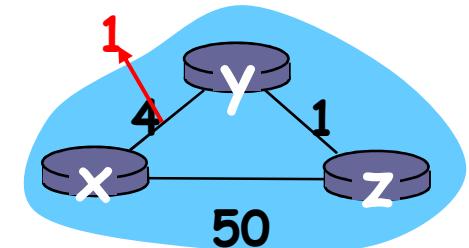
1 2 3 4 Initially  
3 2 3 4 After 1 exchange  
3 4 3 4 After 2 exchanges  
5 4 5 4 After 3 exchanges  
5 6 5 6 After 4 exchanges  
7 6 7 6 After 5 exchanges  
7 8 7 8 After 6 exchanges  
⋮  
• • • •

(b)

## The count-to-infinity problem

# Distance Vector: link cost changes

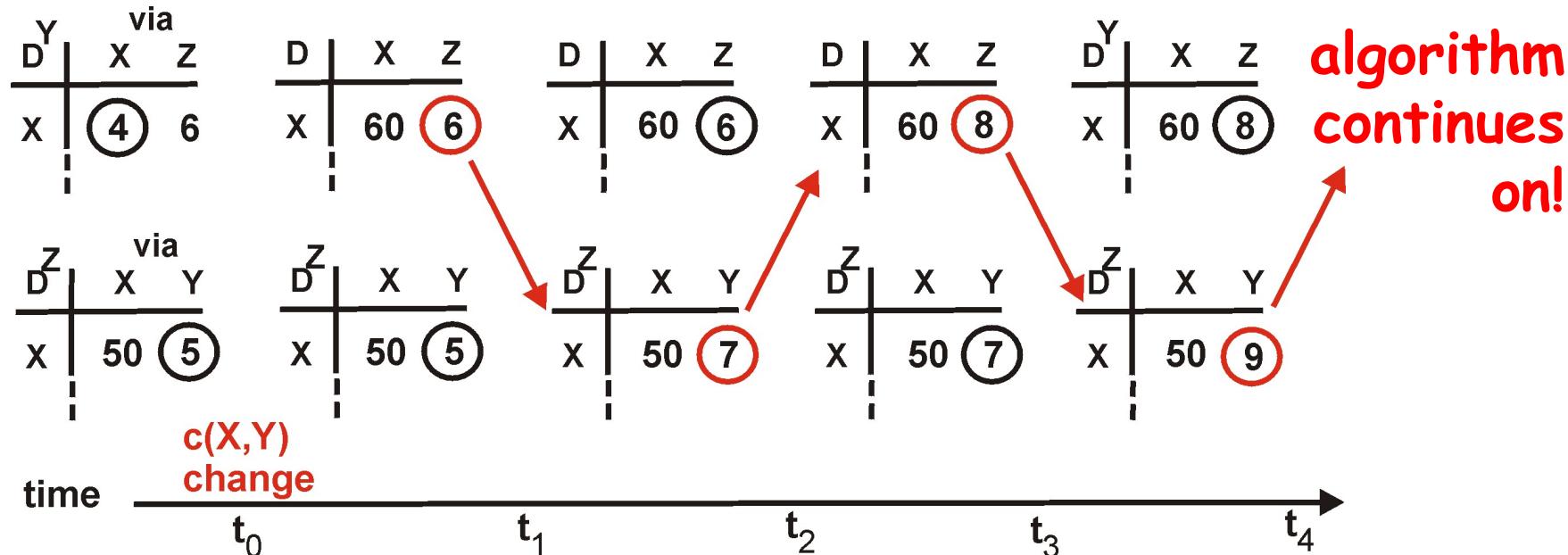
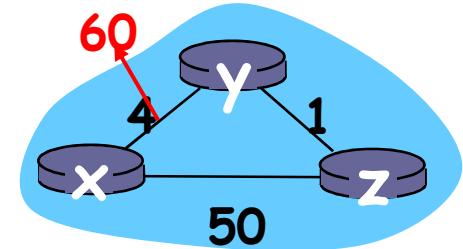
Good news travels fast.



# Distance Vector: link cost changes

**Bad news travels slow.**

- “count to infinity” problem!



- 44 iterations before algorithm stabilizes.
- During this unstable period, packets may wander in **loop**, without ever reaching to destination.

# 慢收敛的解决

- **有限计数**

- 用有限数如**16**代表无穷大（不可达）

- **Split horizon :**

- 如果 Z 通过 Y 到达 X, 则 Z 不将该路由通告给 Y。

- **Poisoned reverse:**

- 如果 Z 通过 Y 到达 X, 则 Z 将该路由通告给 Y,  
但该路由的距离为无穷大（不可达）

但不能完全消除路由回路!

# Link State Routing

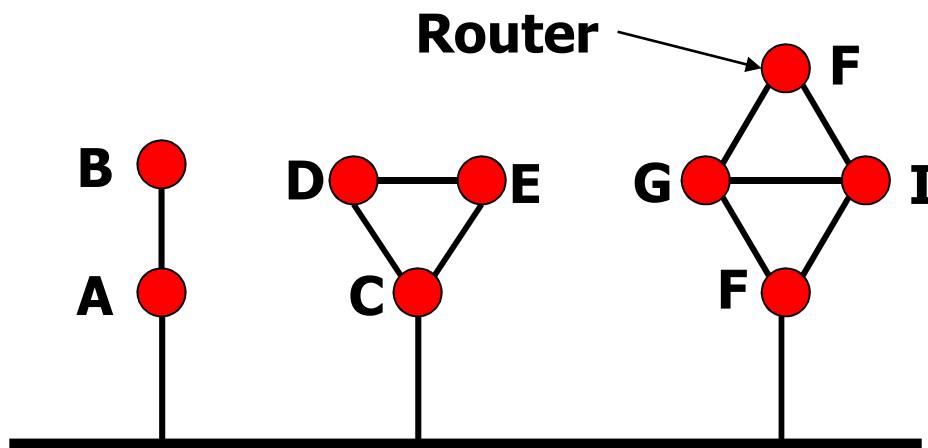
- Each router must do the following:
  1. Discover neighbors and learn their addresses.
  2. Measure the cost (delay) to each neighbor.
  3. Construct a packet containing all this information
  4. Send this packet to all other routers.
  5. Compute the shortest path to every other router.

# Link State Routing

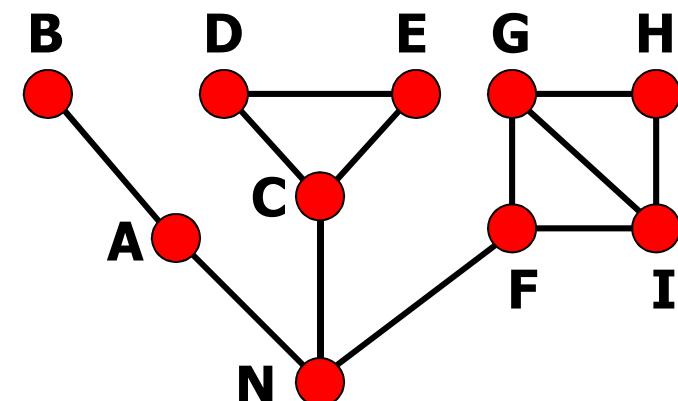
- Each router knows the id of every other router in the network.
- Each router maintains a topology map of the whole network.
- Each router, periodically floods its link state updates (with its direct connectivity information).
- Upon receiving a vector, a router updates the local topology map and recalculates shortest paths.

# 1. Discovering Neighbors

- Send “Hello” packet on each point-to-point line. Destination node replies with its address.



9 routers connected to LAN



Graph Model

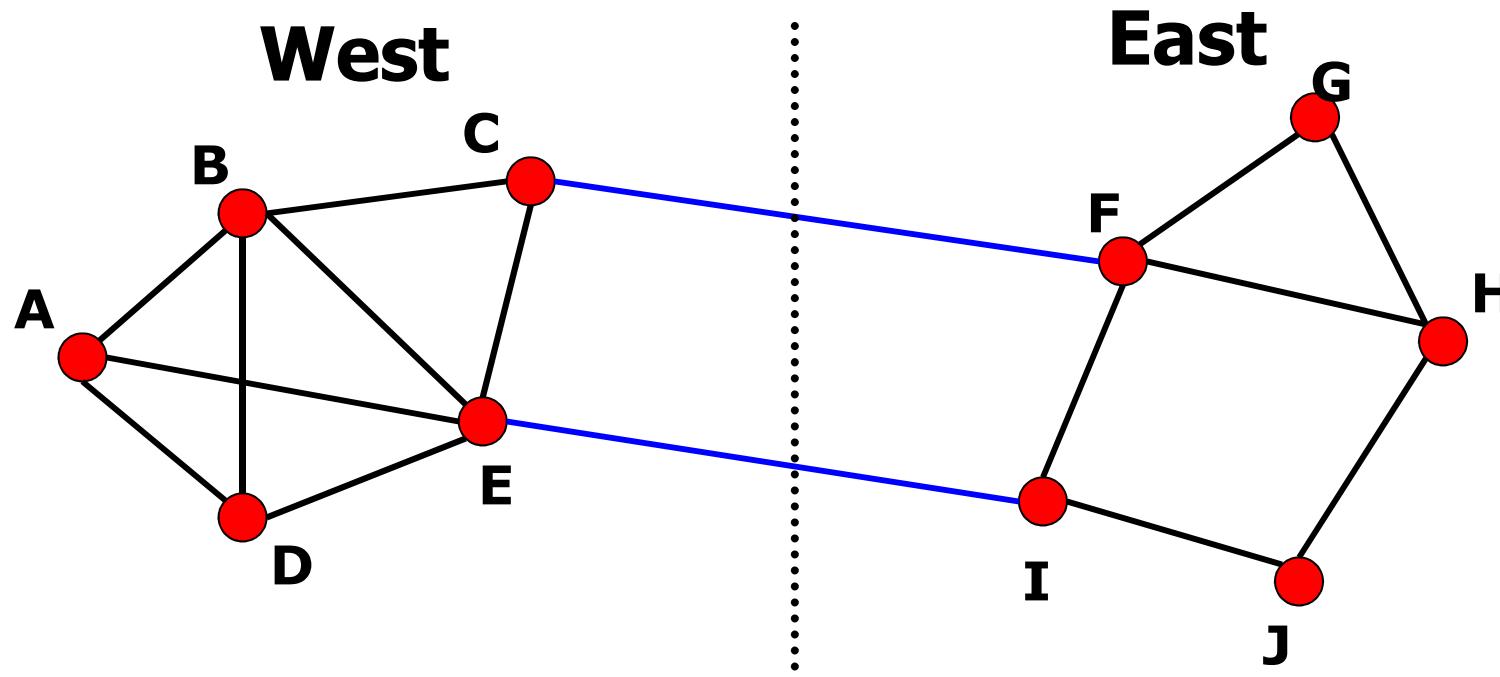
One way to model LAN is to consider it as node (Graph model).

## 2. Measuring Line Cost

- Send an “**ECHO**” packet over the line.
- Destination is required to respond to “**ECHO**” packet immediately.
- Measure the cost required for this operation:
  - Round-trip time
  - Traffic load
  - Line bandwidth

## 2. Measuring Line Cost

- **Question:** Should we measure just the time it takes to transmit the packet, or should we include the time that the packet waits in the queue?



## **2 Arguments :**

- Should include the time that the packet spends in the queue, as this provides a more accurate picture of the real delays.**
- Should only include the transmission times, otherwise the network is likely to oscillate between preferred paths.**

# 3. Building Link State Packets

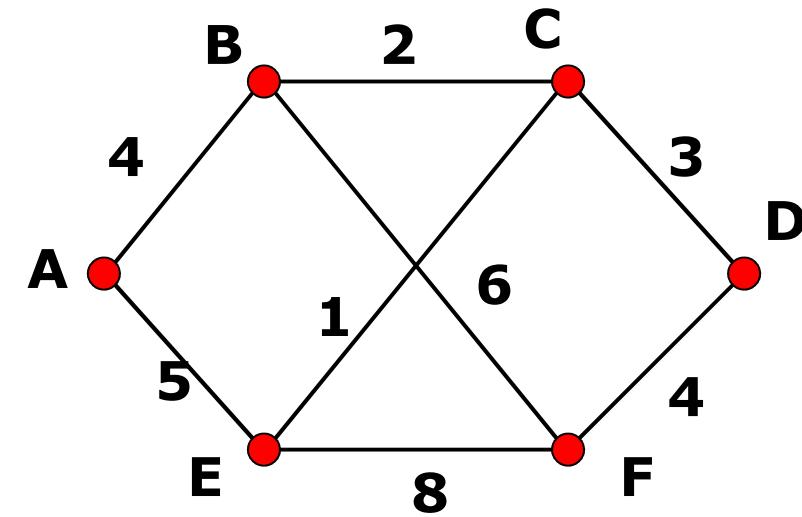
## Packet Format:

- Identity of Sender
- Sequence Number
- Age
- List of Neighbors
- Corresponding Delay

A
Seq.
Age
B 4
E 5

B
Seq.
Age
A 4
C 2
F 6

C
Seq.
Age
B 2
D 3
E 1



D
Seq.
Age
C 3
F 4

E
Seq.
Age
A 5
C 1
F 8

F
Seq.
Age
B 6
D 4
E 8

Packets easily built, **problem** with knowing when to build them

# 4. Distributing the Link State Packets

- Use **reliable flooding**
- all routers maintain a track list of  
*(source, seq, age)*
- New link state packets is checked :
  - If **new/unseen** (based on the sequence number) then it is broadcasted to all neighboring routers with exception of the sender.
  - If **duplicate**, it is disregarded.
  - If sequence number is **lower**, it is rejected.
  - The age is decremented once a second, and every time it is forwarded by a router. When the age hits **zero**, the LSP is discarded

# 4. Distributing the Link State Packets

## ■ Reliable flooding algorithm

### □ Acknowledgments and retransmissions

			Sent Flags			Acknowledged Flags			
Source	Seq.	Age	A	C	F	A	C	F	Data
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

Example of packed buffer for router B of subnet in previous figure

# When to Initiate Flooding

- **Topology change**

- Link or node failure
  - Link or node recovery

- **Configuration change**

- Link cost change

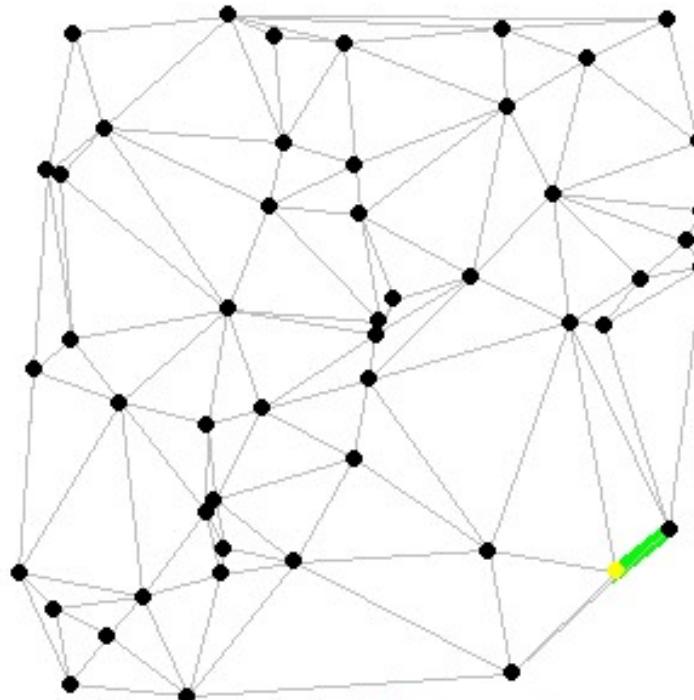
- **Periodically**

- Refresh the link-state information
  - Typically (say) 30 minutes
  - Corrects for possible corruption of the data

# 5. Computing the New Routes

- Use Dijkstra's algorithm locally to construct the shortest path to all possible destinations.

Dijkstra's algorithm



# Comparison of LS and DV algorithms

## Message complexity

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

## Speed of Convergence

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

## Robustness: what happens if router malfunctions?

### 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 thru  
network

# Hierarchical Routing

- network “**flat**”: all routers identical

**Flat routing problem: Not scale.**

- Proportionally large routing tables
- More memory capacity and CPU time
- More bandwidth is needed to send status reports, .....

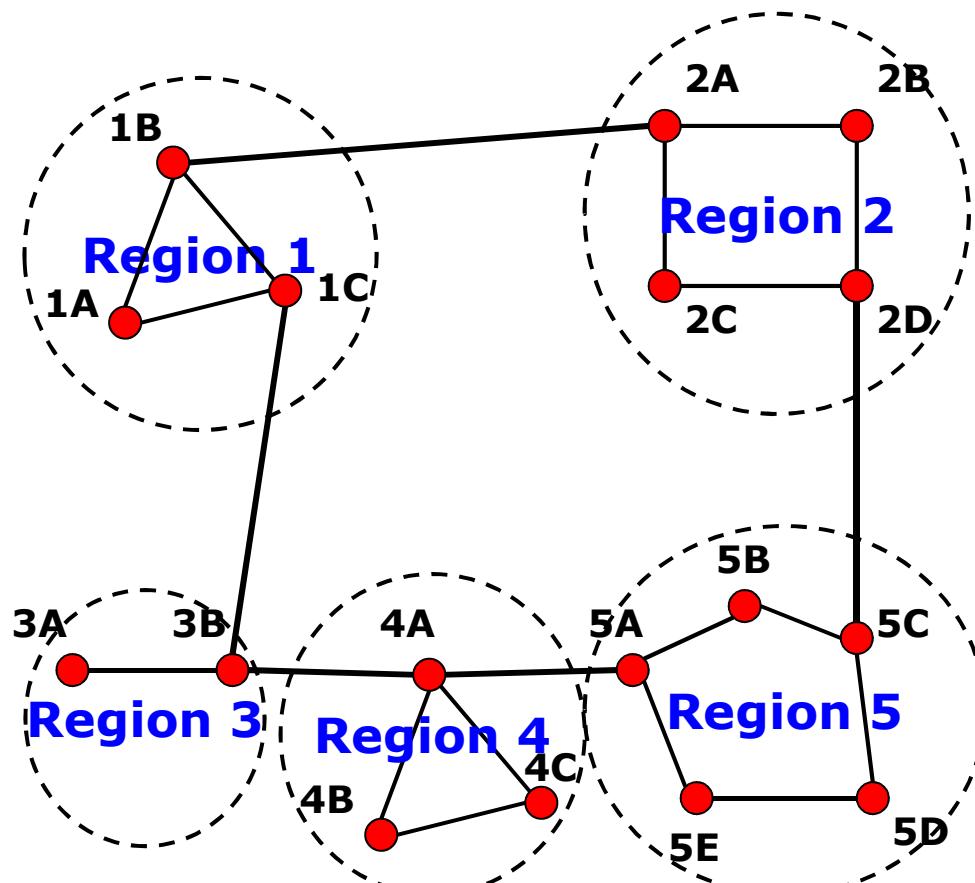
**Solution:**

- Routing has to be done hierarchically.

# Hierarchical Routing

- Routers divided in *Regions*:
  - Each router knows how to route packets to destinations **within its own region**.
  - However, router **does not** have any information regarding the topology of the network of other regions.
- **Huge networks will require more than two-level hierarchy.**

# Two Level Hierarchical Routing Example



Full table for 1A		
Dest.	Line	Hops
1A	-	-
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

Hierarchical table for 1A		
Dest.	Line	Hops
1A	-	-
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

# Penalty for Hierarchical Routing

- **sub-optimal routes**
  - Path length may increase. But this increase is sufficiently small and usually acceptable.



# Others

- **Broadcast Routing**
- **Multicast Routing**
- **Routing for Mobile Hosts**
- **Routing in Ad Hoc Networks**
- **Node Lookup in Peer-to-Peer Networks**
- .....

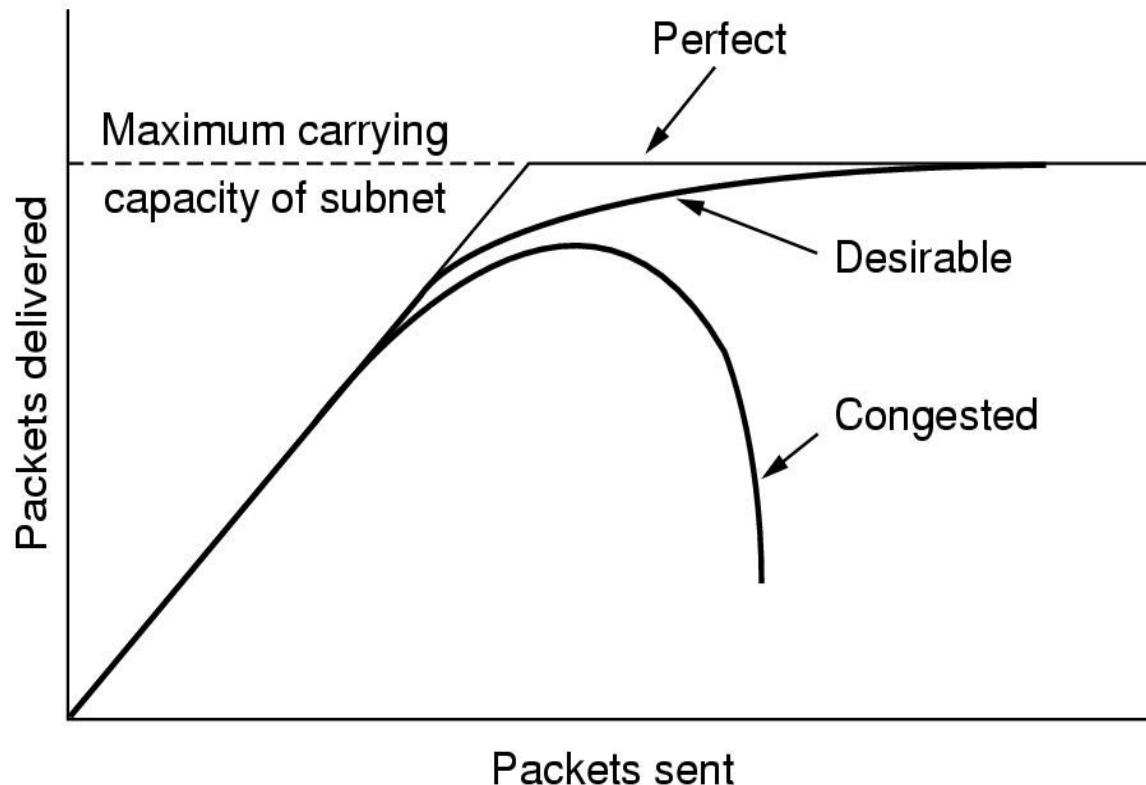
# Chapter 5: Roadmap

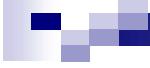
- Network Layer and Design Issues
- Routing Algorithms
- **Congestion Control Algorithm**
- IP
- IP Multicasting
- Mobile IP

# Congestion Control

- **Problem:** when too many packets have to be transmitted through the network, we can get into a serious performance problem:

When too much traffic is offered, congestion sets in and performance degrades sharply.

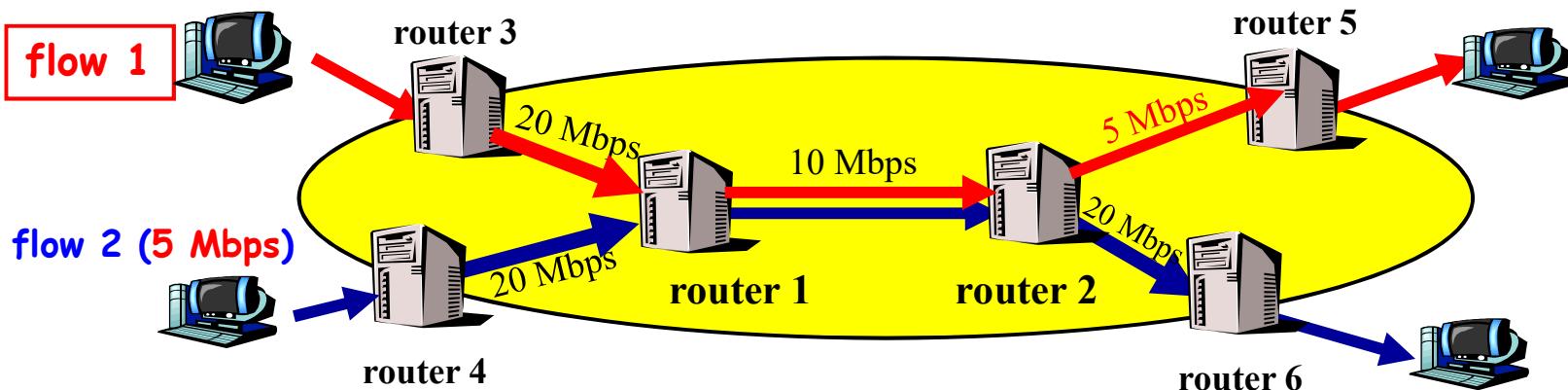




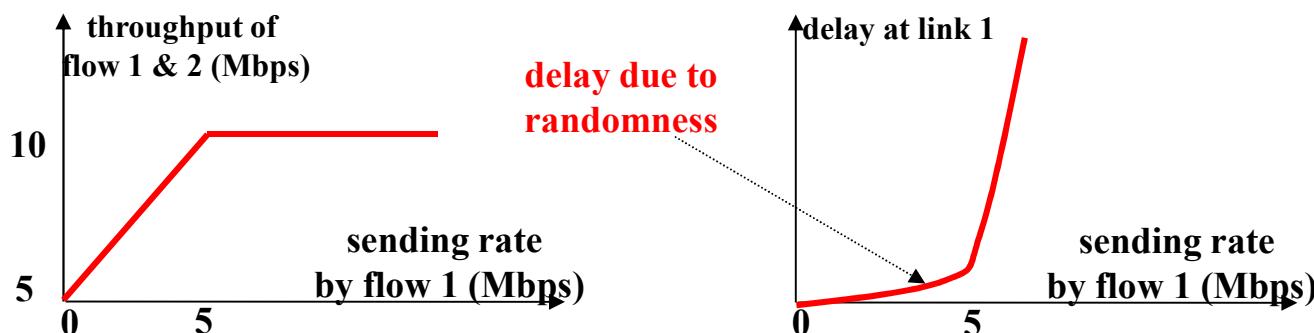
# **Some General Questions**

- **How can congestion happen?**
- **What is congestion control?**
- **Why is congestion control difficult?**

# Cause/Cost of Congestion: Single Bottleneck

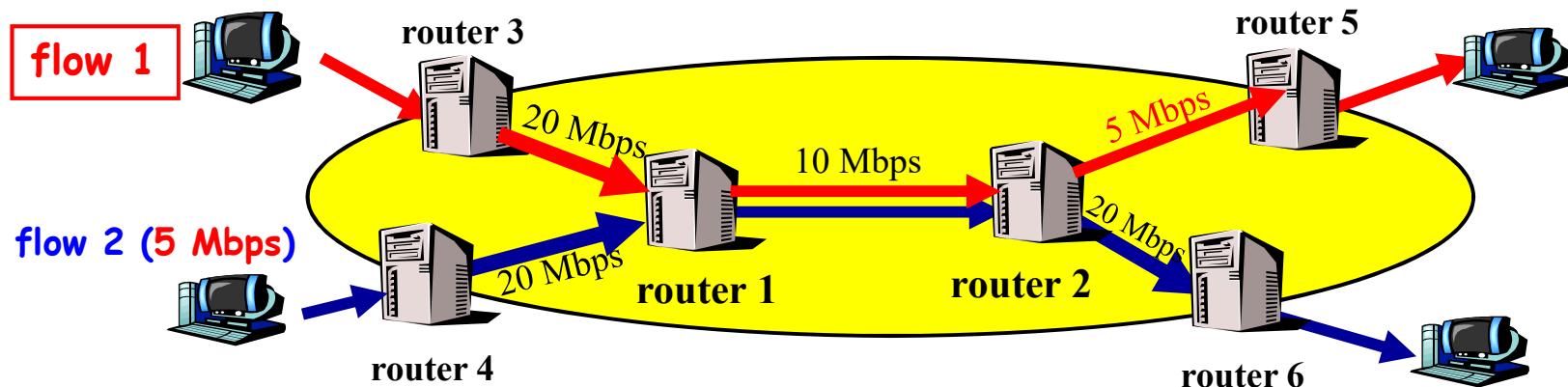


- Flow 2 has a fixed sending rate of 5 Mbps
- We vary the sending rate of flow 1 from 0 to 20 Mbps
- Assume
  - no retransmission
  - the link from router 1 to router 2 has infinite buffer
  - throughput: e2e packets delivered in unit time



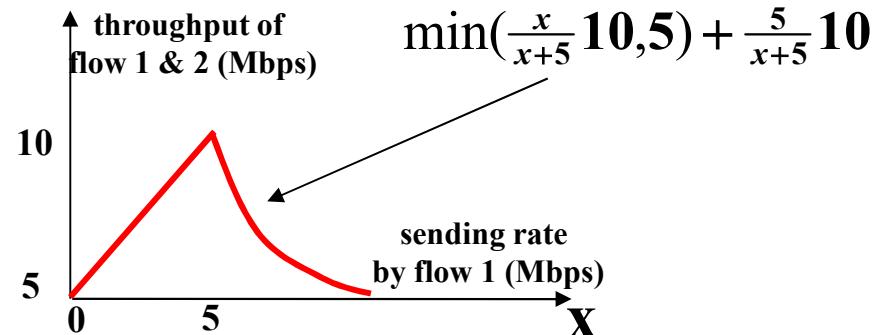
large delays  
when  
congested

# Cause/Cost of Congestion: Single Bottleneck



❑ Assume

- no retransmission
- the link from router 1 to router 2 has finite buffer
- throughput: e2e packets delivered in unit time



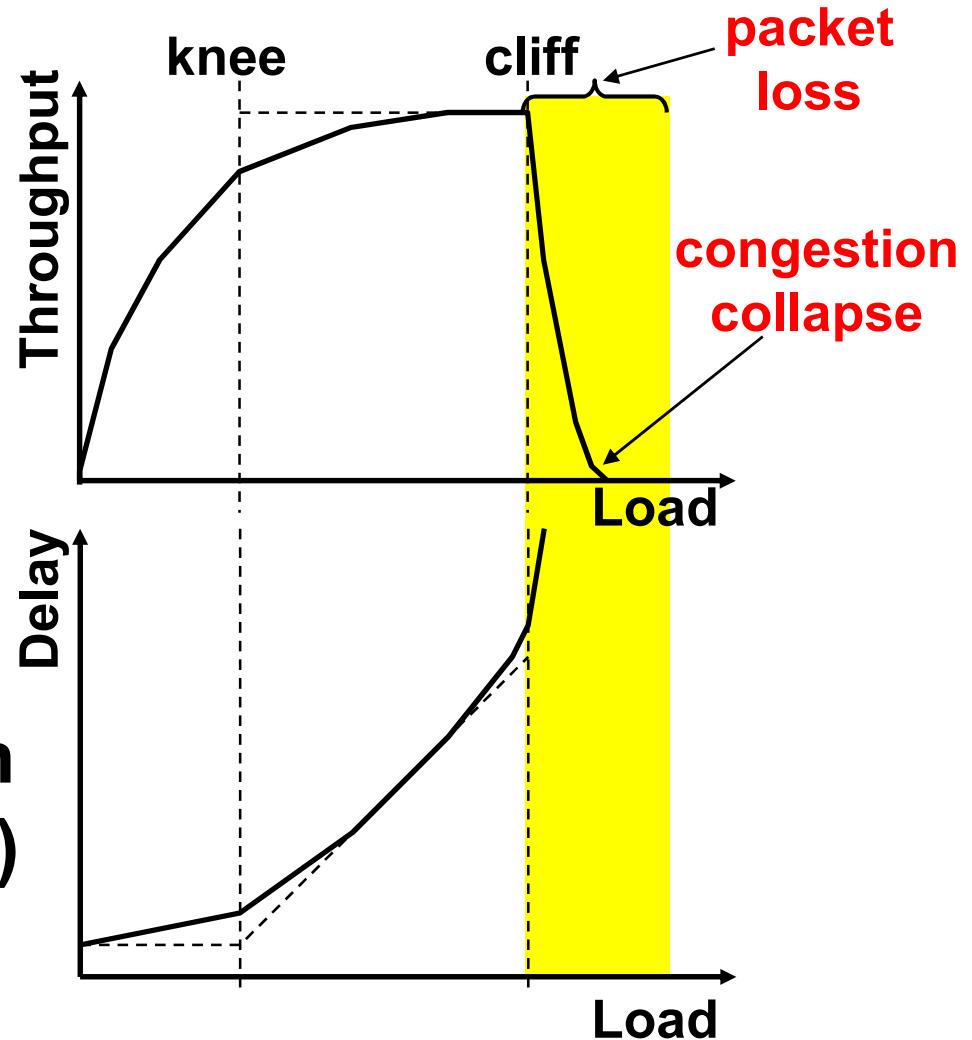
What if retransmission?

when packet dropped  
at the link from router  
2 to router 5, the  
upstream transmission  
from router 1 to  
router 2 used for that  
packet was wasted!

# The Cost of Congestion

## ■ Packet loss

- ❑ wasted upstream bandwidth when a packet is discarded at downstream
- ❑ wasted bandwidth due to retransmission (a packet goes through a link multiple times)



## ■ High delay

# Approaches towards congestion control

## Two broad approaches :

### End-end congestion control:

- no explicit feedback from network
- congestion inferred from end-system observed loss, delay
- approach taken by TCP

### Network-assisted congestion control:

- routers provide feedback to end systems
- single bit indicating congestion (SNA, DECbit, TCP/IP ECN, ATM)
- explicit rate sender should send at

# Rate-based vs. Window-based

## Rate-based:

- Congestion control by explicitly controlling the sending rate of a flow, e.g. set sending rate to 128Kbps
- Example: ATM

## Window-based:

- Congestion control by controlling the window size of a transport scheme, e.g. set window size to 64KBytes
- Example: TCP

# Implicit vs. Explicit feedback

## Explicit:

- routers provide feedback to end systems
  - explicit rate sender should send at
  - single bit indicating congestion (SNA, DECbit, **TCP ECN**, ATM)

## Implicit:

- congestion inferred by end systems through observed loss, delay

# The Desired Properties of a Congestion Control Scheme

- **Efficiency:** close to full utilization but low delay
- **Fairness (resource sharing)**
- **Distributedness (no central knowledge for scalability)**

# Reducing Congestion

## ■ Increase resources

- Get additional bandwidth
  - Use faster lines
  - Obtain additional lines
  - Utilize alternate pathways
  - Utilize “spare” routers

## ■ Decrease Traffic

- Send messages to senders telling them to slow down
- Deny service to some users
- Degrade service to some or all users
- Schedule usage to achieve better load balance

# Two Categories of Congestion Control

- **Open loop solutions**

- Attempt to **prevent** problems rather than correct them
- Does not utilize runtime feedback from the system

- **Closed loop solutions**

- Uses feedback (measurements of system performance) to make corrections at runtime.

# Congestion Prevention Policies

## Policies that affect congestion

Layer	Policies
Transport	<ul style="list-style-type: none"><li>• Retransmission policy</li><li>• Out-of-order caching policy</li><li>• Acknowledgement policy</li><li>• Flow control policy</li><li>• Timeout determination</li></ul>
Network	<ul style="list-style-type: none"><li>• Virtual circuits versus datagram inside the subnet</li><li>• Packet queueing and service policy</li><li>• Packet discard policy</li><li>• Routing algorithm</li><li>• Packet lifetime management</li></ul>
Data link	<ul style="list-style-type: none"><li>• Retransmission policy</li><li>• Out-of-order caching policy</li><li>• Acknowledgement policy</li><li>• Flow control policy</li></ul>

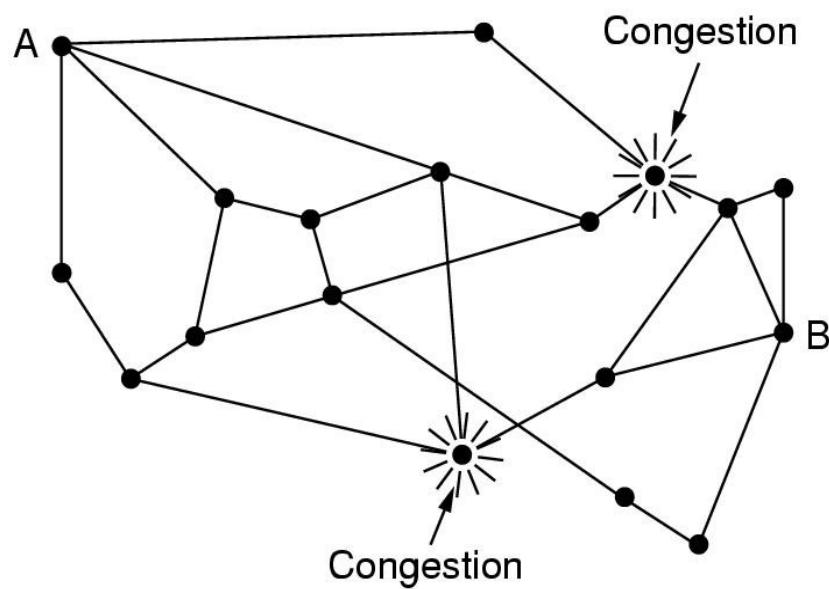
# General Principles of Closed Loop Congestion Control

- **Monitor the system to detect when and where congestion occurs**
  - Percentage of all packets discarded for lack of buffer space
  - The average queue lengths
  - The number of packets that time out and are retransmitted
  - The average packet delay
- **Pass information to places where action can be taken**
  - Transfer the information about the congestion from the point where it is detected to the point where something can be done
- **Adjust system operation to correct the problem**
  - Increase the resources or decrease the load

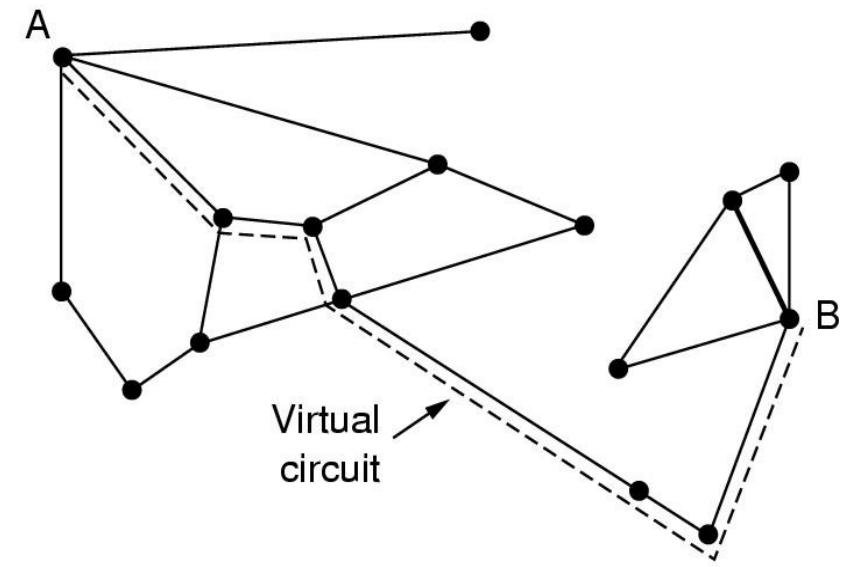
# Congestion Control in Virtual Circuits

- **Principle:** when you set up a circuit, be sure that congestion can be avoided.
  1. **Admission control:** if it's too busy, just refuse to set up a virtual circuit. This is the same as refusing new users at an FTP site.
  2. **Select alternative routes** when a part in the network is getting overloaded (i.e., temporarily rebuild your view of the network)
  3. **Negotiate the quality of the circuit in advance,** so that the network provider can reserve buffers and the like. Resources are guaranteed to be there.

# Congestion Control in Virtual Circuits



(a)



(b)

# Congestion Control in Datagram

## ■ Choke Packets

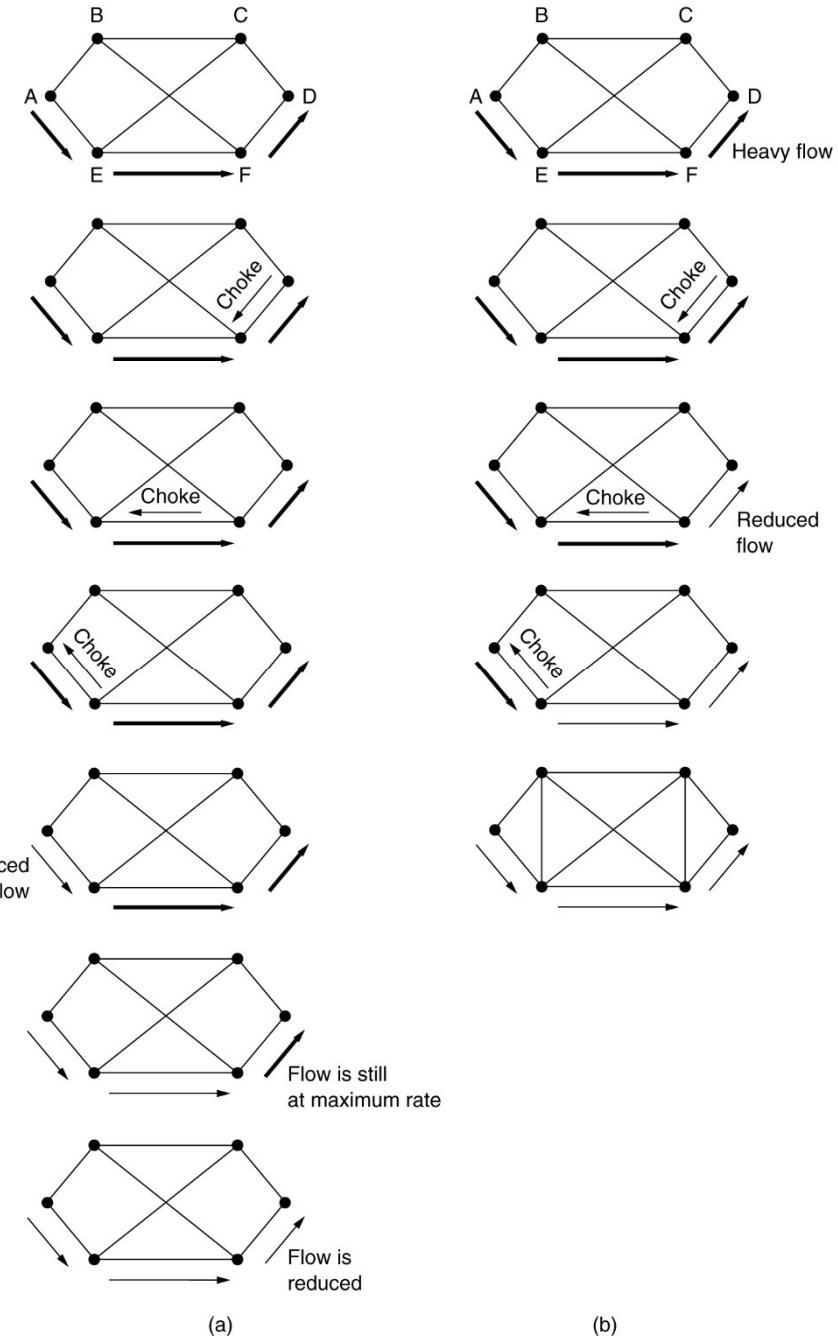
□ **Send info back to the source when the networks starts to perform bad.**

■ A router checks the status of an output line: if it's too occupied, it sends a choke packet to the source. The host is assumed to be cooperative, and that it will slow down (not always a good assumption).

# Hop-by-Hop Choke Packets

**(a) A choke packet that affects only the source.**

**(b) A choke packet that affects each hop it passes through.**

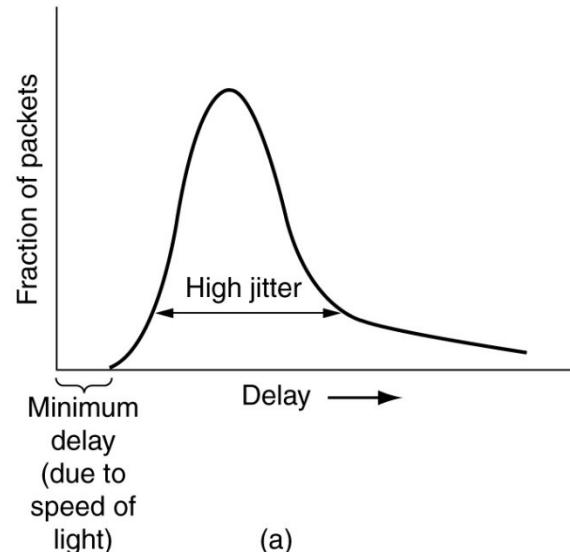


# Load Shedding

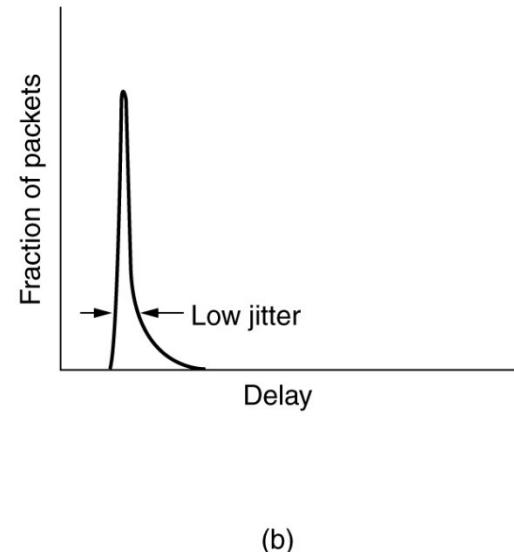
- Load shedding is a fancy way of saying that when routers are being inundated by packets that they cannot handle, they just **throw them away**.
- A router drowning in packets can just pick packets at random to drop, but usually it can do better than that.
- Which packet to discard may depend on the applications running
  - For file transfer, **wine policy** (old is better than new)
  - For multimedia, **milk policy** (new is better than old)
- **Random Early Detection**
  - The idea of discarding packets before all the buffer space is really exhausted.

# Jitter Control

- The variation in the packet arrival times is called jitter.
- Considering the average amount of congestion, jitter can be bounded by computing the expected transmit time for each hop along the path



(a)



(b)

**(a) High jitter.**

**(b) Low jitter.**

# Chapter 5: Roadmap

- Network Layer and Design Issues
- Routing Algorithms
- Congestion Control Algorithm
- IP
- IP Multicasting
- Mobile IP

# The Network Layer in the Internet

## ■ IPv4

- Datagram format;
- IP Addresses: subnetting, CIDR;
- NAT;
- IP Packet Routing and Forwarding;
- ICMP
- ARP

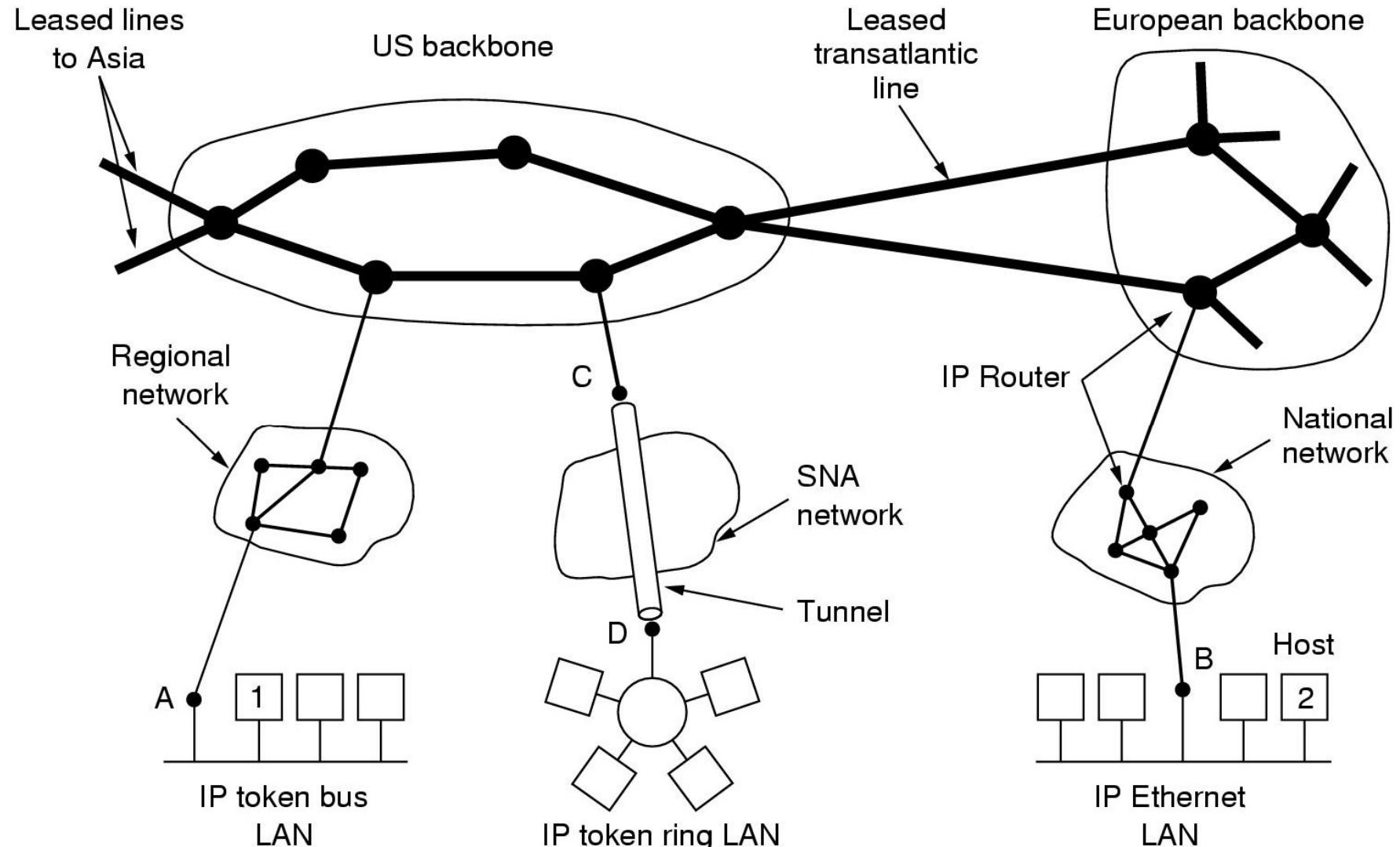
# The Network Layer in the Internet

- Routing in the Internet
  - Intra-AS and Inter-AS Routing
  - RIP
  - OSPF
  - BGP
- IPv6
- IP Multicasting
- Mobile IP

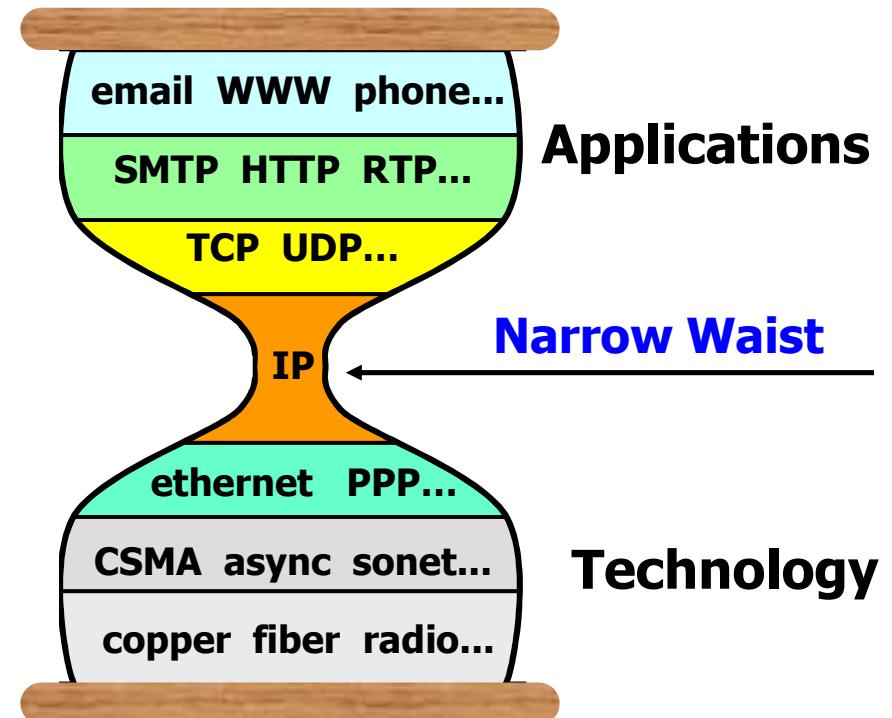
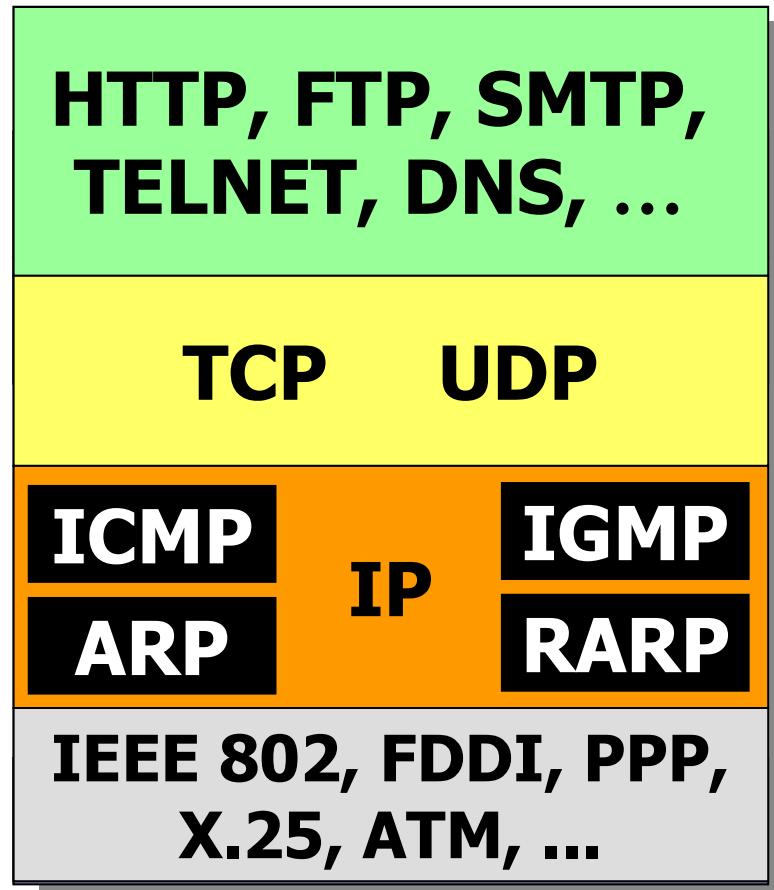
# The Internet

- **The Internet:** view it as a collection of **Autonomous Systems (ASes)** connected together by a bunch of backbones
- The glue that holds the whole Internet together is the network layer protocol, **IP (Internet Protocol)**

# The Internet is an interconnected collection of many networks



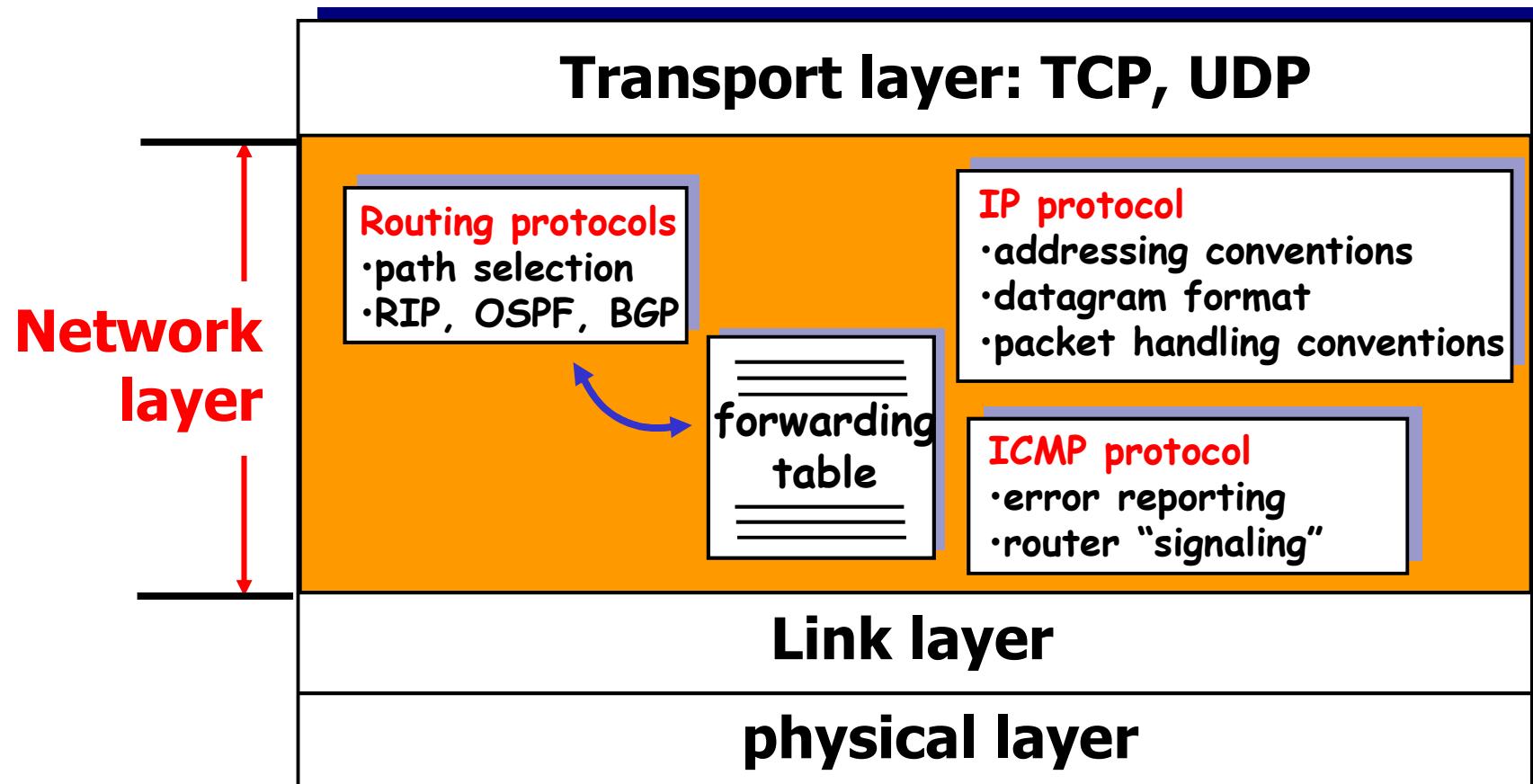
# Internet protocol stack



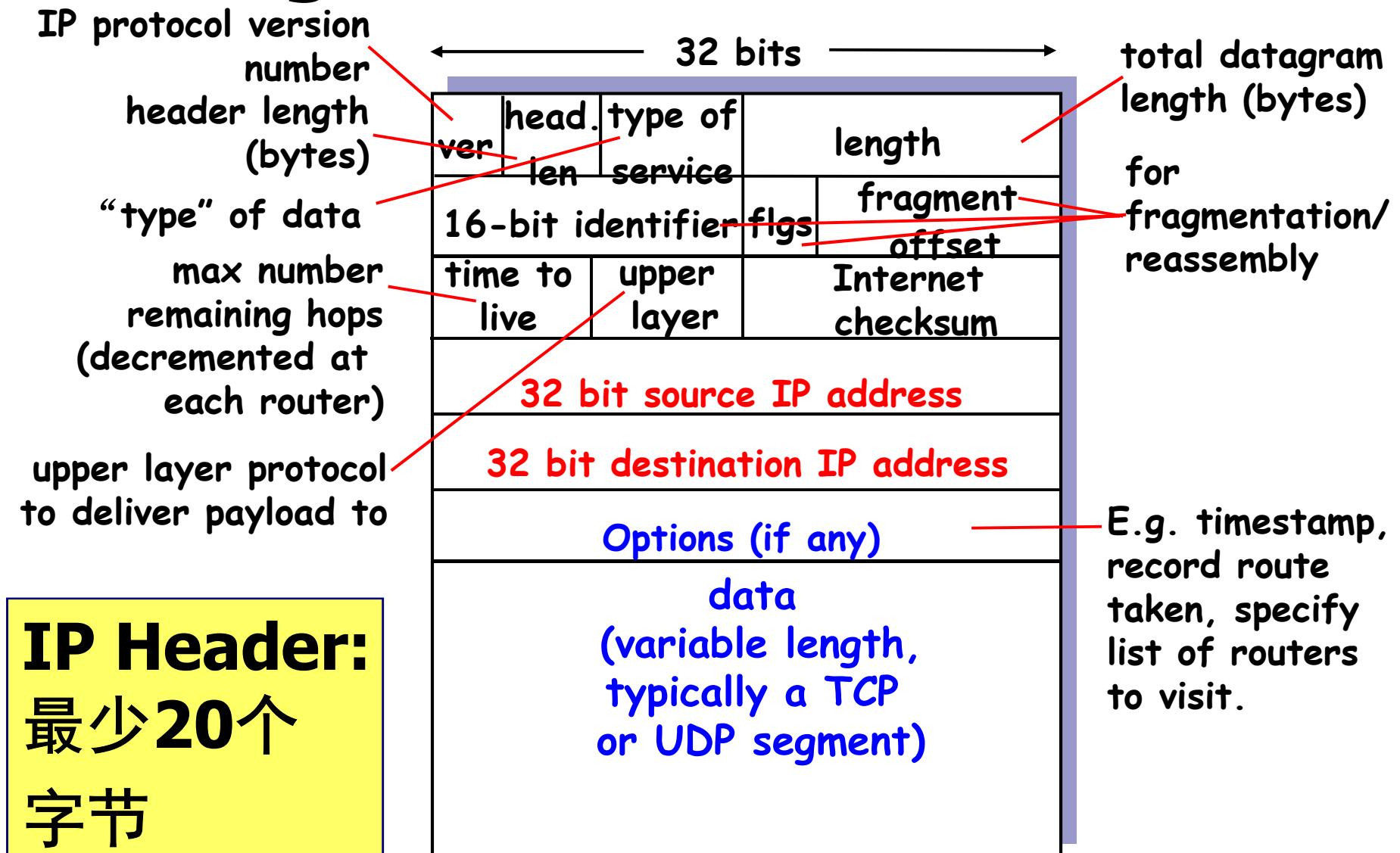
**Everything over IP, IP over everything**

# The Internet Network Layer

## Host, router network layer functions:



# IP datagram format



**IP Header:  
最少20个  
字节**

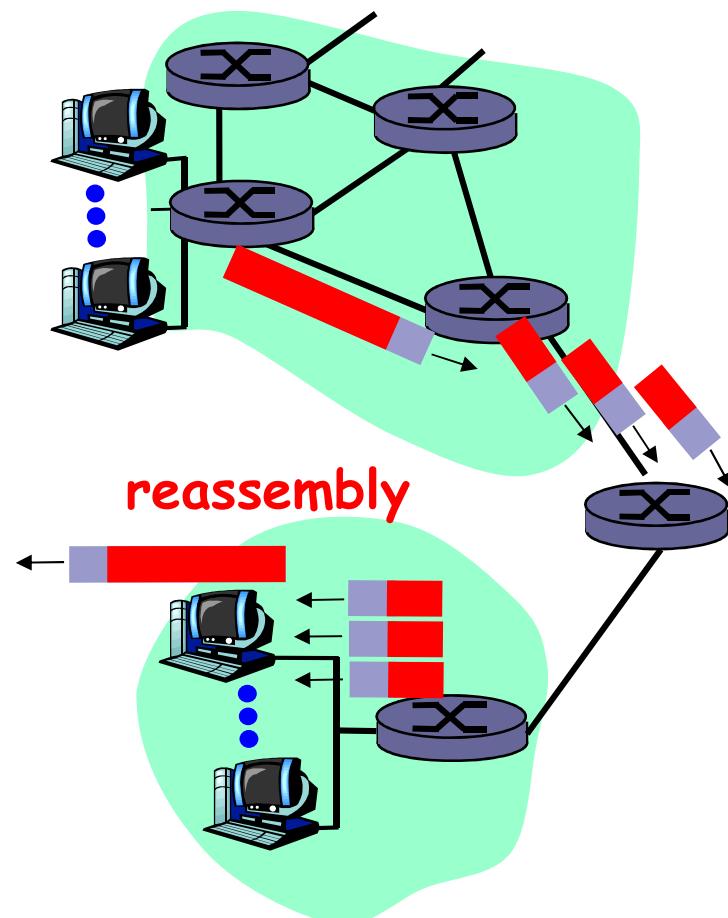
# IP Fragmentation & Reassembly

- network links have **MTU** (max.transfer size) -
  - different link types, different MTUs
- large IP datagram divided ("fragmented") within net
  - one datagram becomes several datagrams
  - "reassembled" only at final destination
  - IP header bits used to identify, order related fragments

**fragmentation:**

in: one large datagram

out: 3 smaller datagrams



# IP Fragmentation & Reassembly

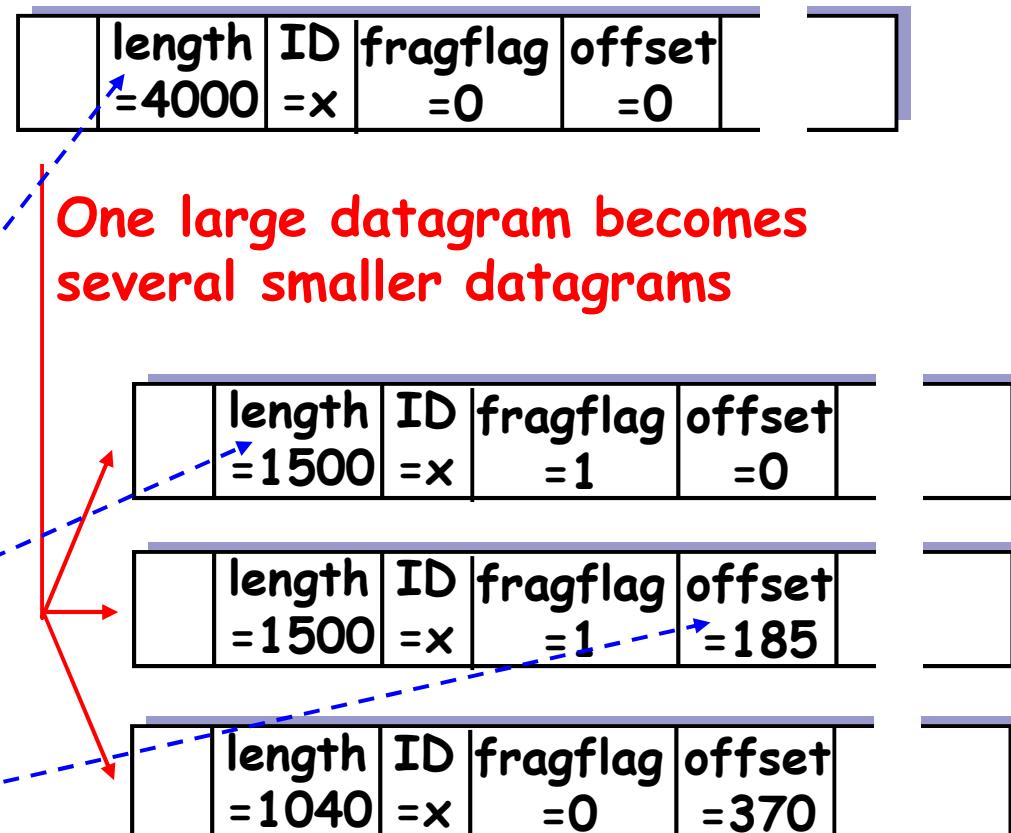
## Example

- 4000-byte datagram
- MTU = 1500B

$4000 - 20 = 3980$  bytes  
in data field

$1500 - 20 = 1480$  bytes  
in data field

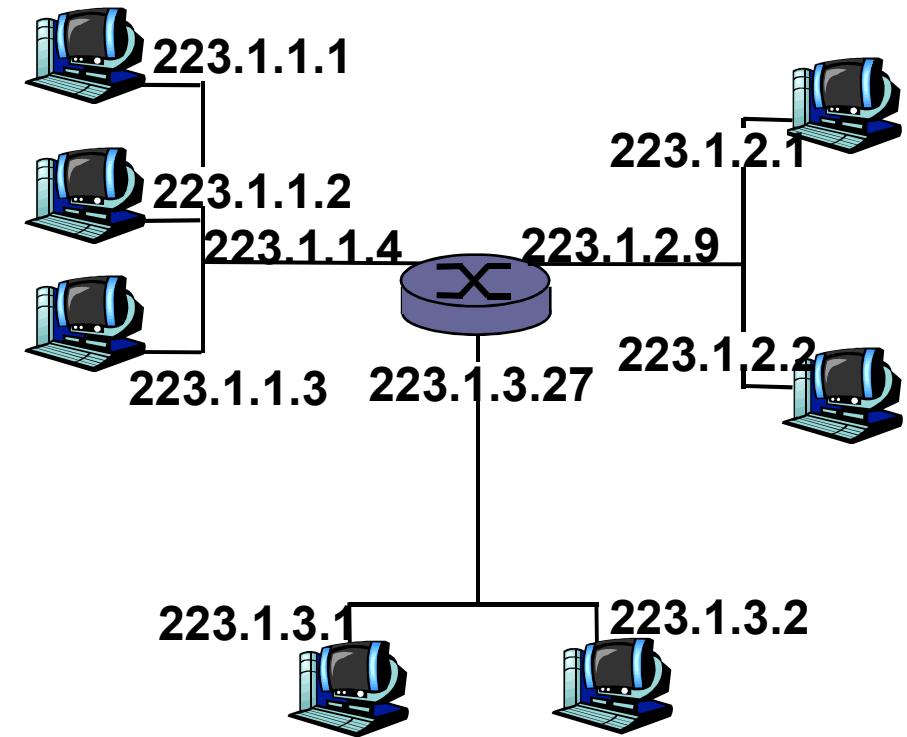
offset =  $1480 / 8$



将IP数据报中的数据部分分段！

# IP Addressing

- **IP address: 32-bit identifier for host, router *interface***
- ***interface:* connection between host/router and physical link**
  - router's typically have multiple interfaces
  - host typically has one interface
  - **IP addresses associated with each interface**



**In Internet, no two devices can have the same IP**

# IP Addressing

- **Binary Notation**
- **Dotted-Decimal Notation**
  - For readability, divide the IP address into 4 bytes.
  - Each byte is separated by dots.

10000000 00001011 00000011 00011111

128.11.3.31

# IP Addressing

## 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 and add dots for separation:

- a. 129.11.11.239
- b. 249.155.251.15

# IP Addressing

## 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

# Address Allocation

- **IANA: Internet Assigned Numbers Authority**
  - Domain name, IP address, protocol parameters, root server management
- **ICCAN: Internet Corporation for Assigned Names and Numbers**
  - allocates addresses
  - manages DNS
  - assigns domain names, resolves disputes
- **ASO: (Address Supporting Organization)**
  - Advisory on policy and structure

# Address Allocation

- **Q:** How does an ISP get block of addresses?
- **A: ICANN:**
  - allocates addresses
  - manages DNS
  - assigns domain names, resolves disputes

# IP Addressing Scheme

- We need an address to **uniquely** identify each destination
- Routing scalability needs flexibility in **aggregation** of destination addresses
  - we should be able to aggregate a set of destinations as a single routing unit

# IP Addressing Scheme

- **Classful Addressing** (有类寻址)

  - 5 classes: A, B, C, D, E

- **Subnetting** (子网)

  - Network Mask (网络掩码)

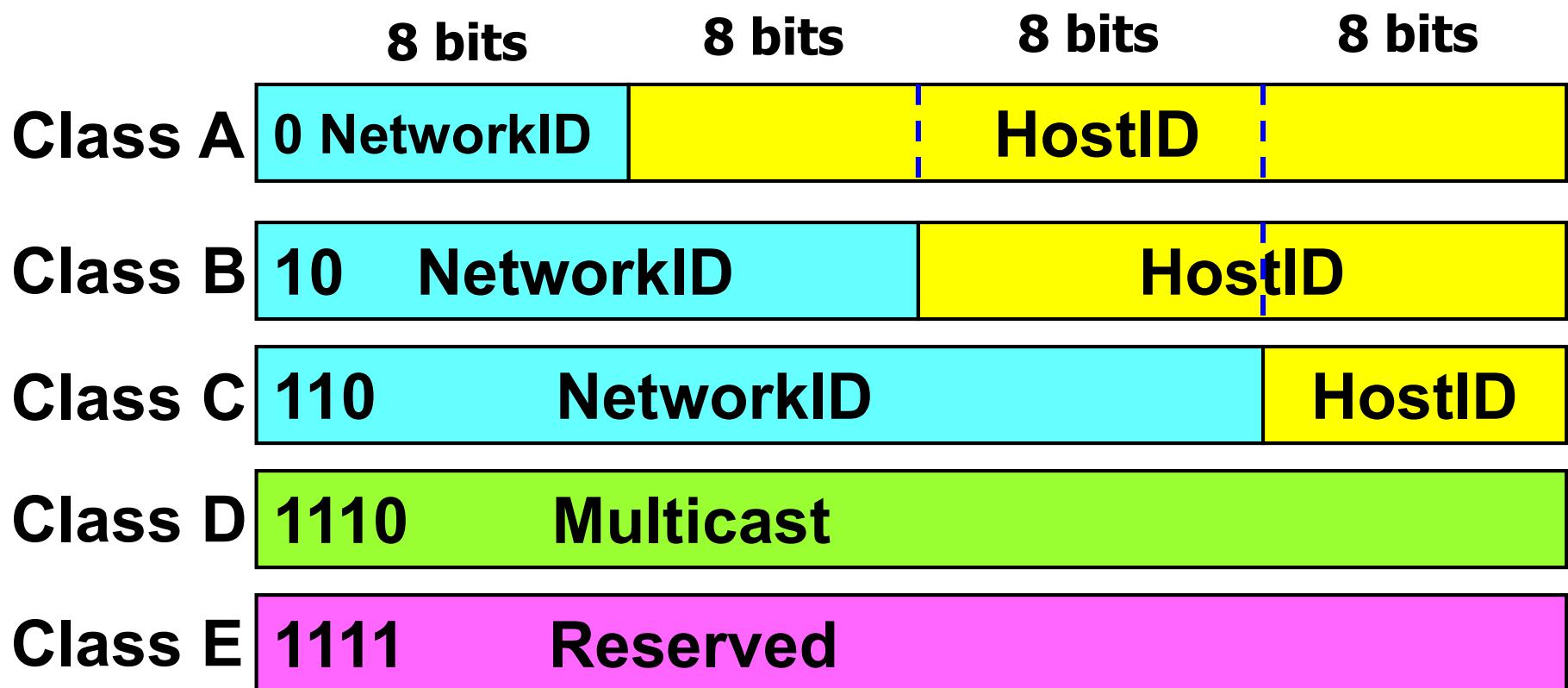
- **Classless Addressing** (无类寻址)

  - Subnet (子网) & Supernet (超网)

- **NAT (Network Address Translation)**

# Classful Addressing

- The address space is divided into five classes: A, B, C, D and E



**NetworkID = network address, HostID = host address**

# Classful Addressing

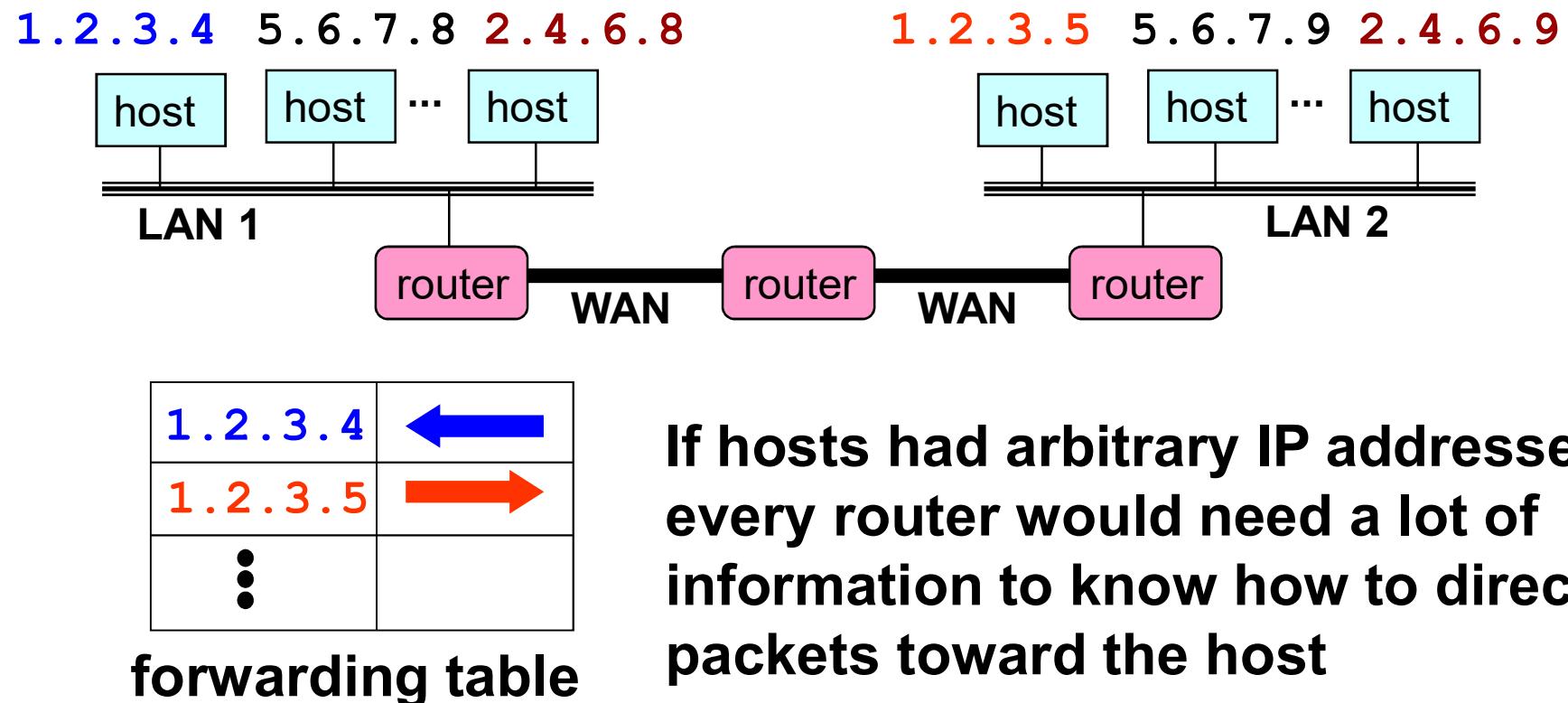
- **Unicast:** One source to one destination
- **Multicast:** One source to a group of destinations. Multicast address can be used only as a destination address, but never as a source address.

# Finding Class in Decimal Notation

	First byte	Second byte	Third byte	Fourth byte
Class A	<b>0 to 127</b>			
Class B	<b>128 to 191</b>			
Class C	<b>192 to 223</b>			
Class D	<b>224 to 239</b>			
Class E	<b>240 to 255</b>			
		<b>0 ~ 255</b>	<b>0 ~ 255</b>	<b>0 ~ 255</b>

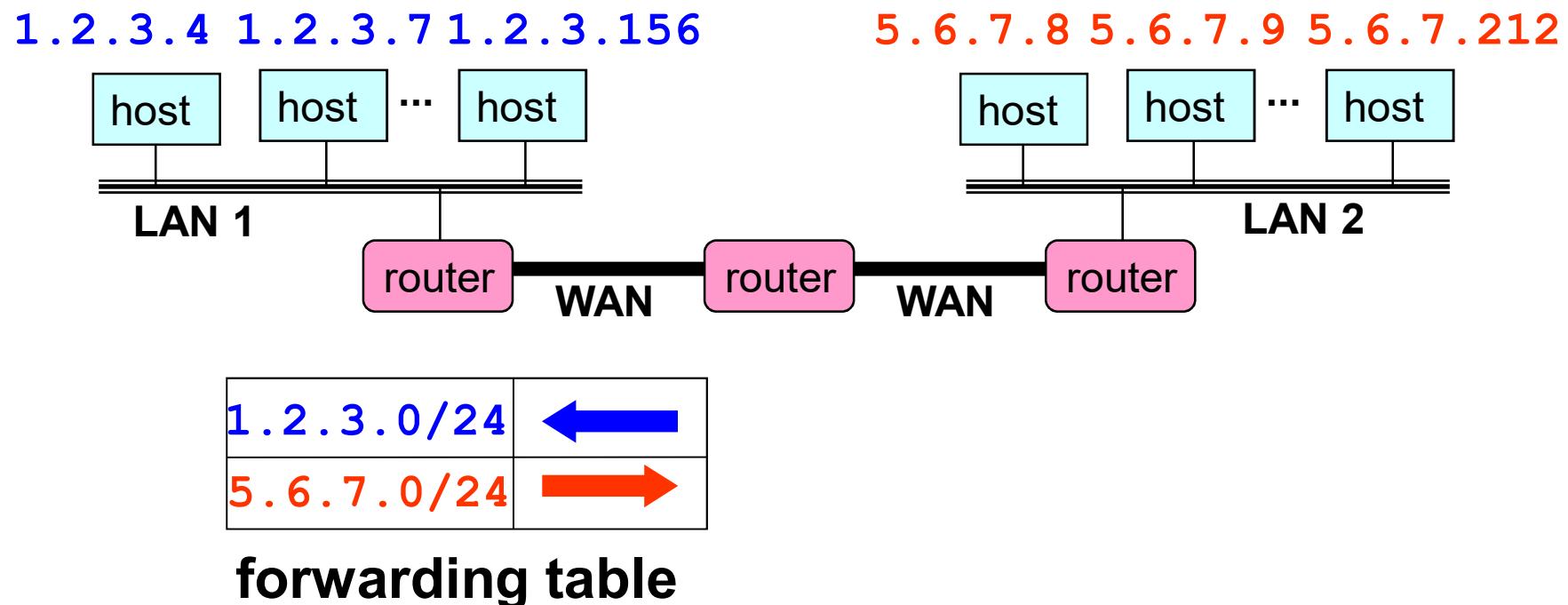
# Why classifying IP addresses?

- Can be assigned based on size and need.
- Improve Scalability.



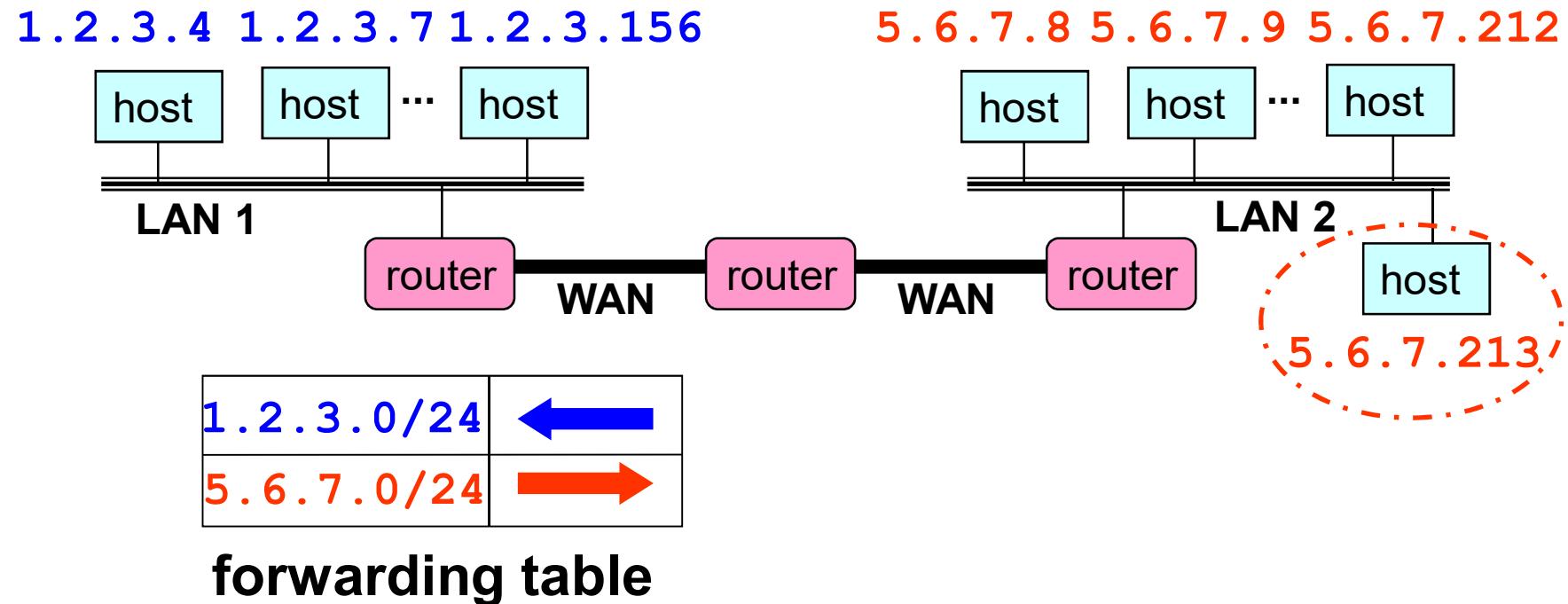
# Scalability Improved

- Number related hosts from a common subnet
  - 1.2.3.0/24 on the left LAN
  - 5.6.7.0/24 on the right LAN



# Scalability Improved

- Easy to Add New Hosts
  - No need to update the routers



# Special IP Addresses

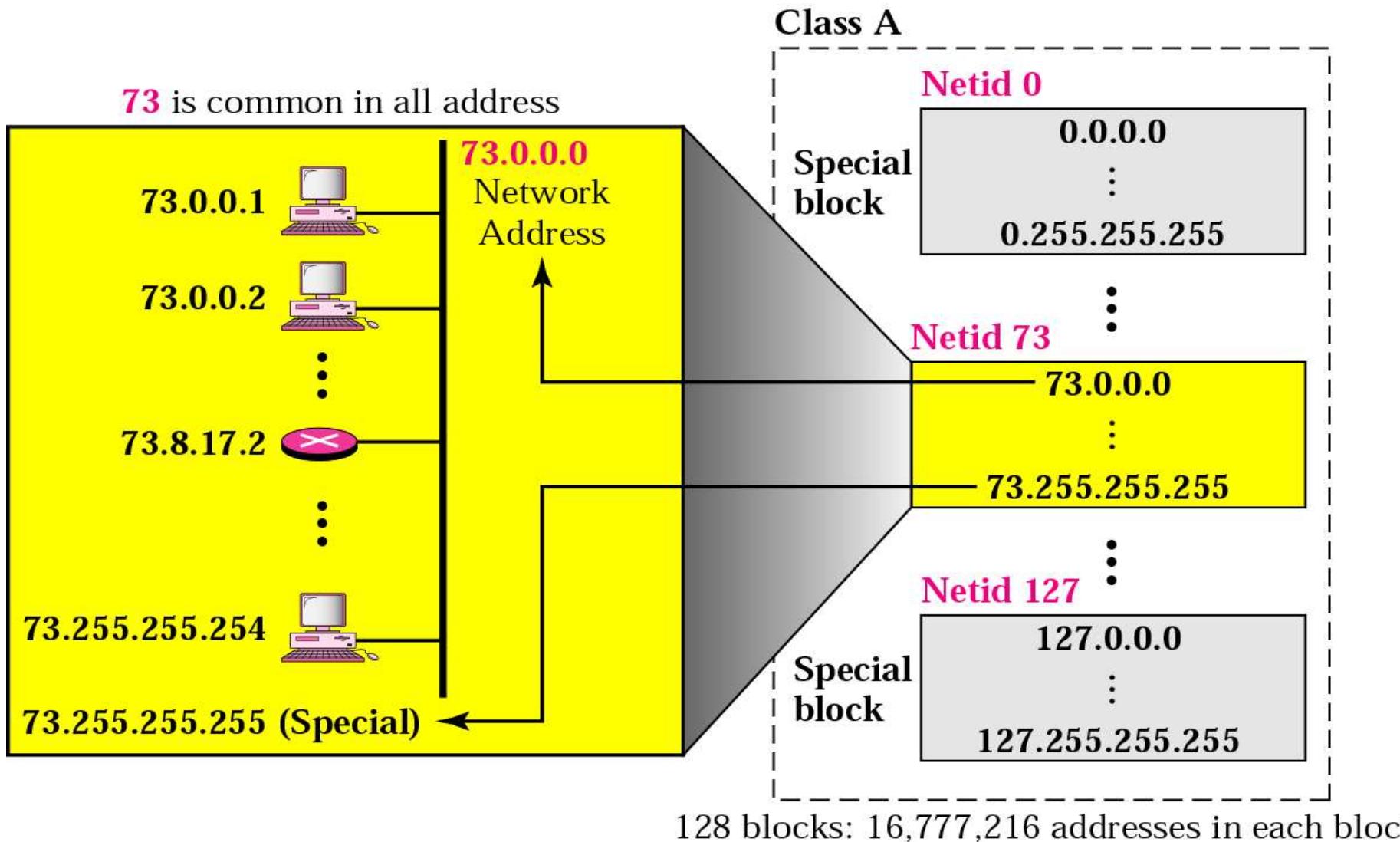
NetID	HostID	Source address	Dest. Address	meaning
0	0	Yes	No	This host
0	any	Yes	No	A host on this network
All 1	All 1	No	Yes	All host on this network
any	All 1	No	Yes	All host on network NetID
127	any	Yes	Yes	Loop test address

# Classful Addressing

## The range of Classful IP Address

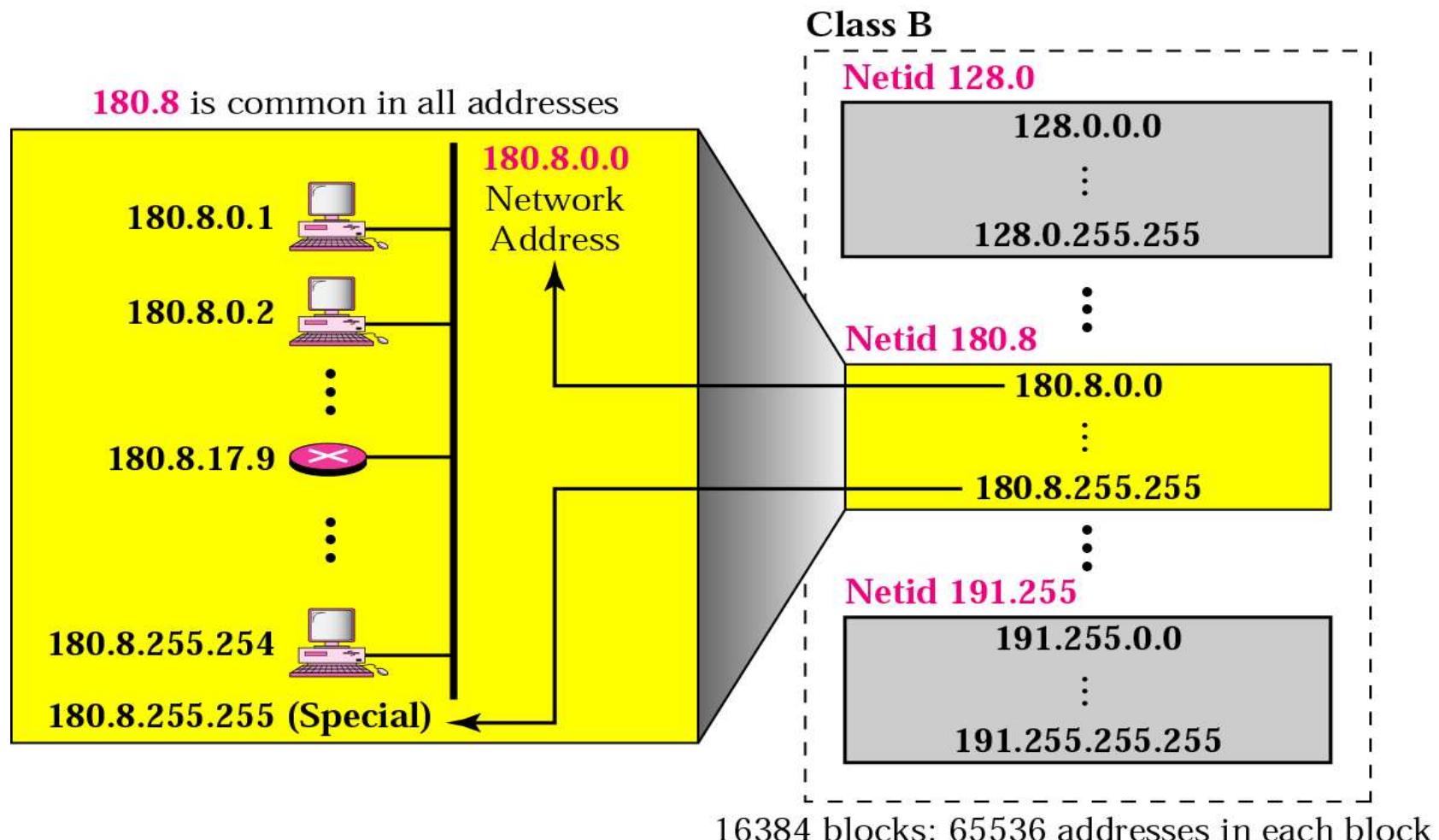
Class	Max. # of networks	First NetID	Last NetID	Max.# of hosts
A	126 $(2^7 - 2)$	1	126	16,777,214
B	16,384 $(2^{14})$	128.0	191.255	65,534
C	2,097,152 $(2^{21})$	192.0.0	223.255.255	254 $(2^8 - 2)$

# Blocks in class A



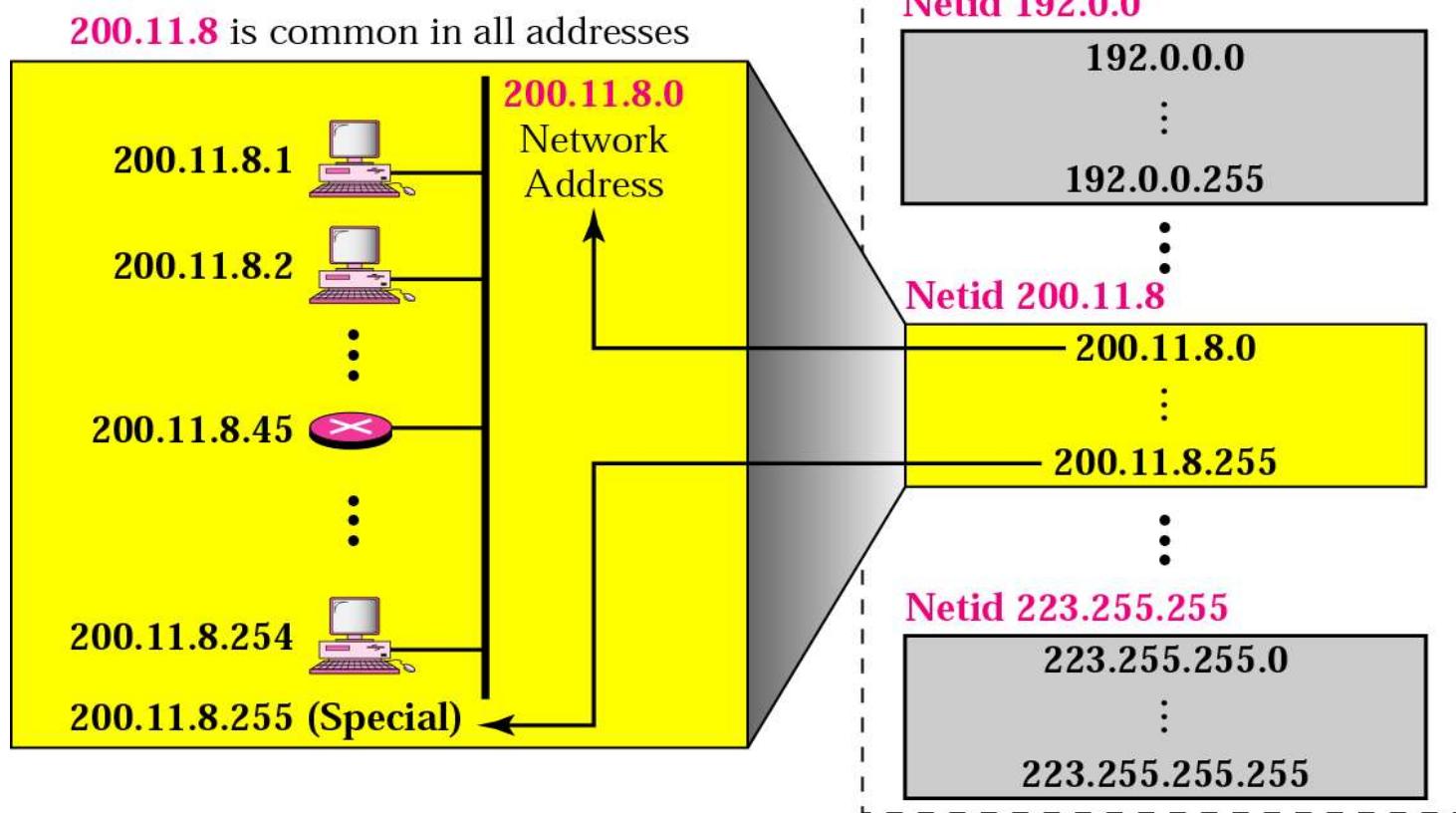
# Blocks in class B

- 16 blocks are reserved for private address.

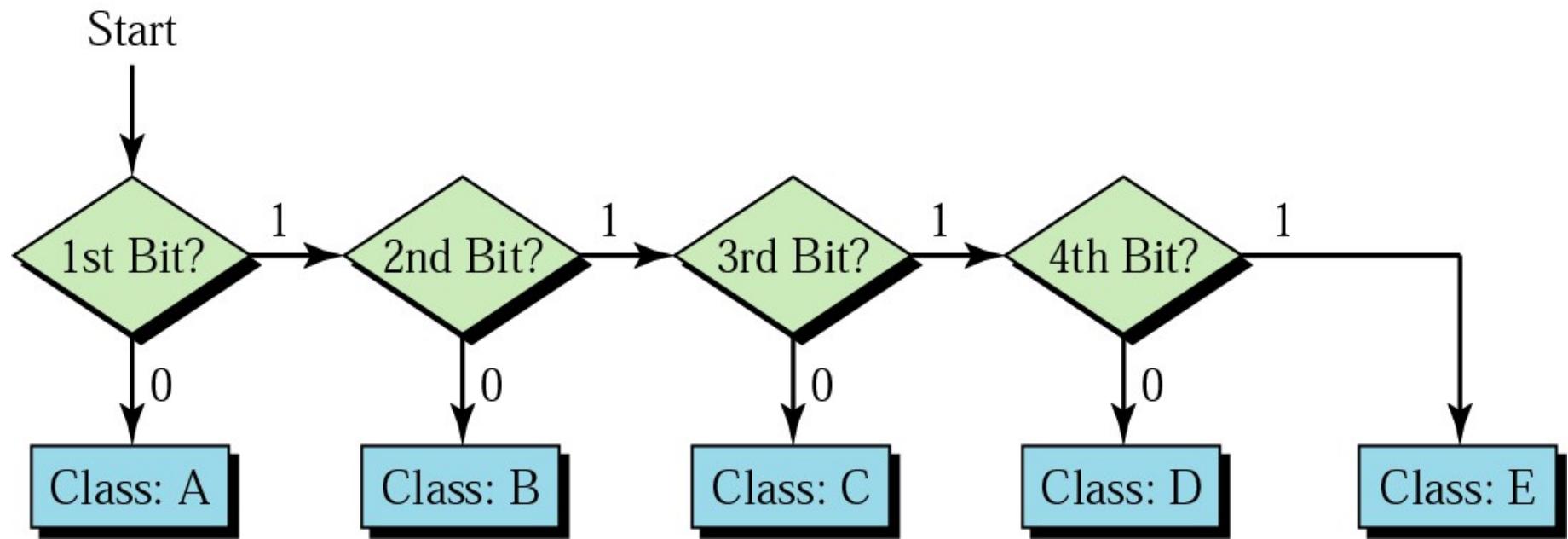


# Blocks in class C

- 256 blocks are used for private address.
- Designed for small organizations with a small number of computers.



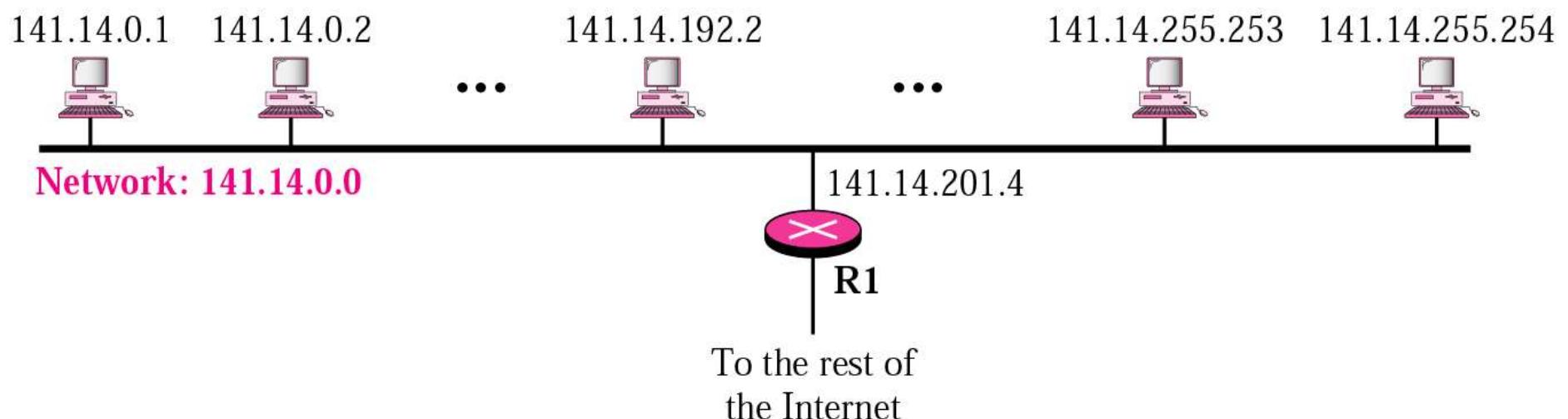
# Finding the Address Class



# Levels of hierarchy

## ■ Levels of Hierarchy

- To reach a host on the Internet, we must first reach the network by using the first portion of the address (netid)
- Then we must reach the host itself by using the second portion (hostid)
- IP addresses are designed with two levels of hierarchy.



# Levels of hierarchy

- Chief advantage of IP addresses:  
**Routers could keep one entry per network instead of one per destination host**

# Classful Addressing

## Example 3

Find the class of each address:

- a. 00000001 00001011 00001011 11101111
- b. 11110011 10011011 11111011 00001111

## Solution

- 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.

# Classful Addressing

## Example 4

Find the class of each address:

- a. 227.12.14.87
- b. 252.5.15.111
- c. 134.11.78.56

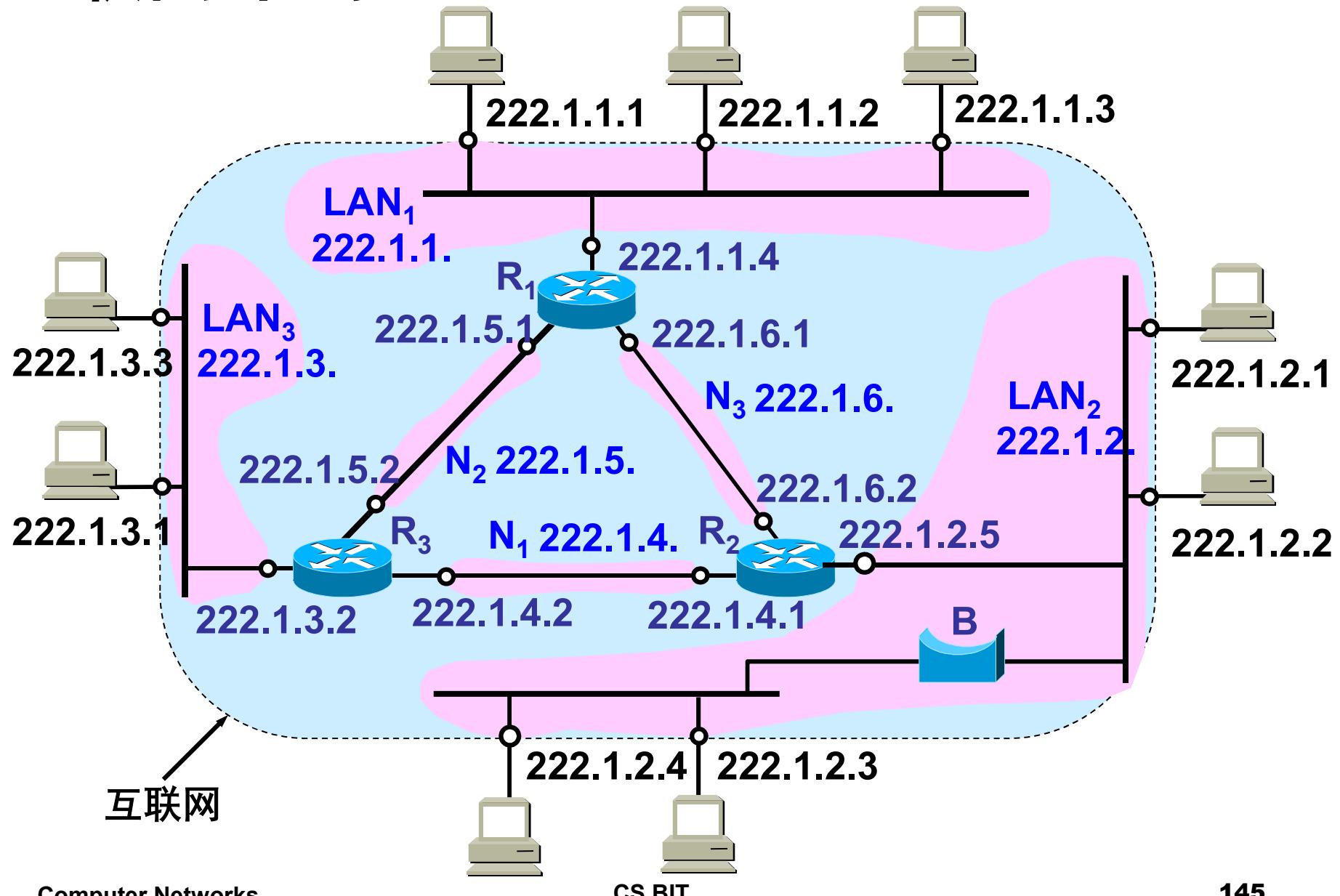
## Solution

- 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.

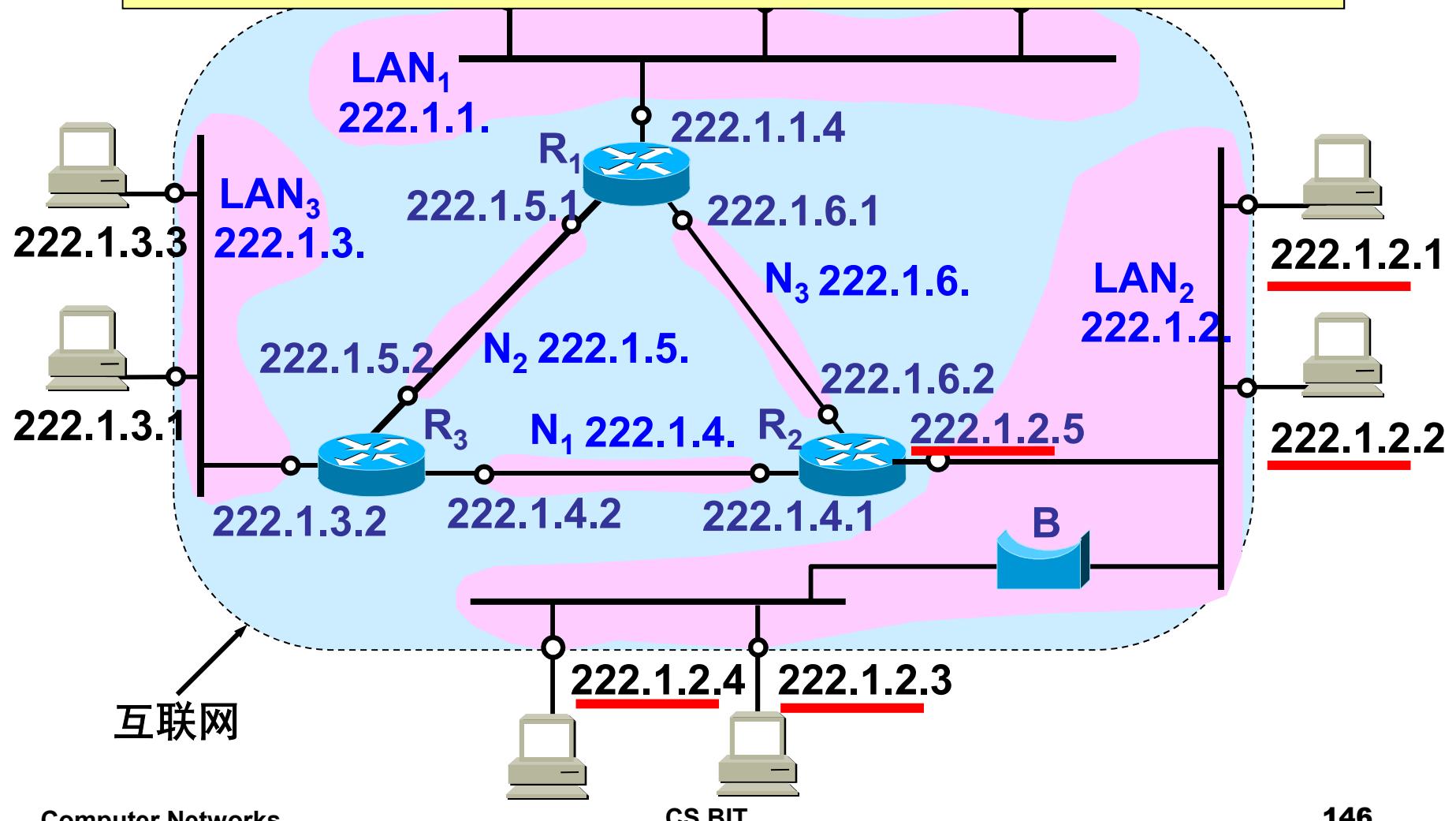
# IP Address Classes Exercise

Address	Class	Network	Host
10.2.1.1	A	10.0.0.0	0.2.1.1
128.63.2.100	B	128.63.0.0	0.0.2.100
201.222.5.64	C	201.222.5.0	0.0.0.64
192.6.141.2	C	192.6.141.0	0.0.0.2
130.113.64.16	B	130.113.0.0	0.0.64.16
256.241.201.10	Nonexistent		

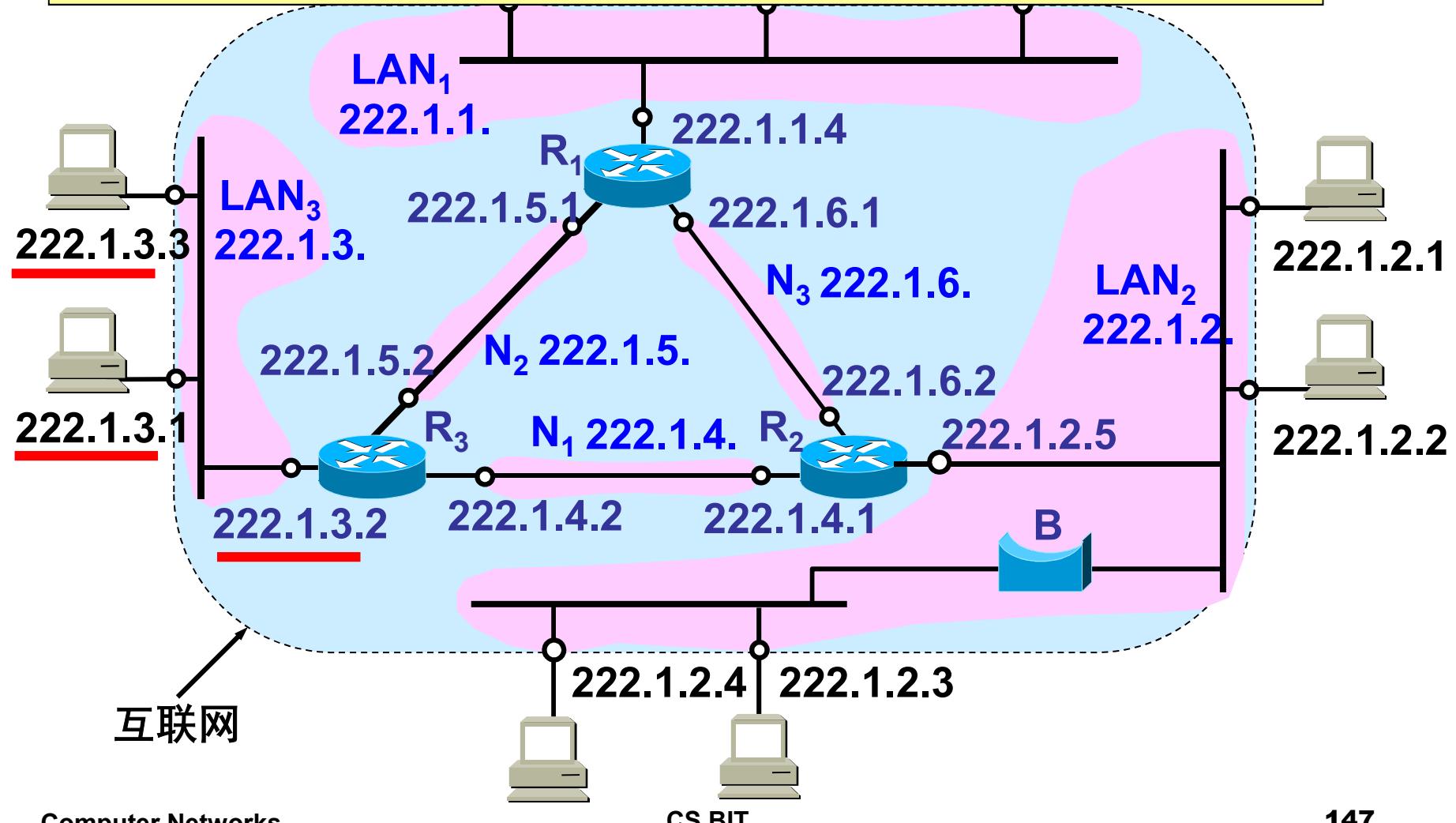
# 互联网中的 IP 地址



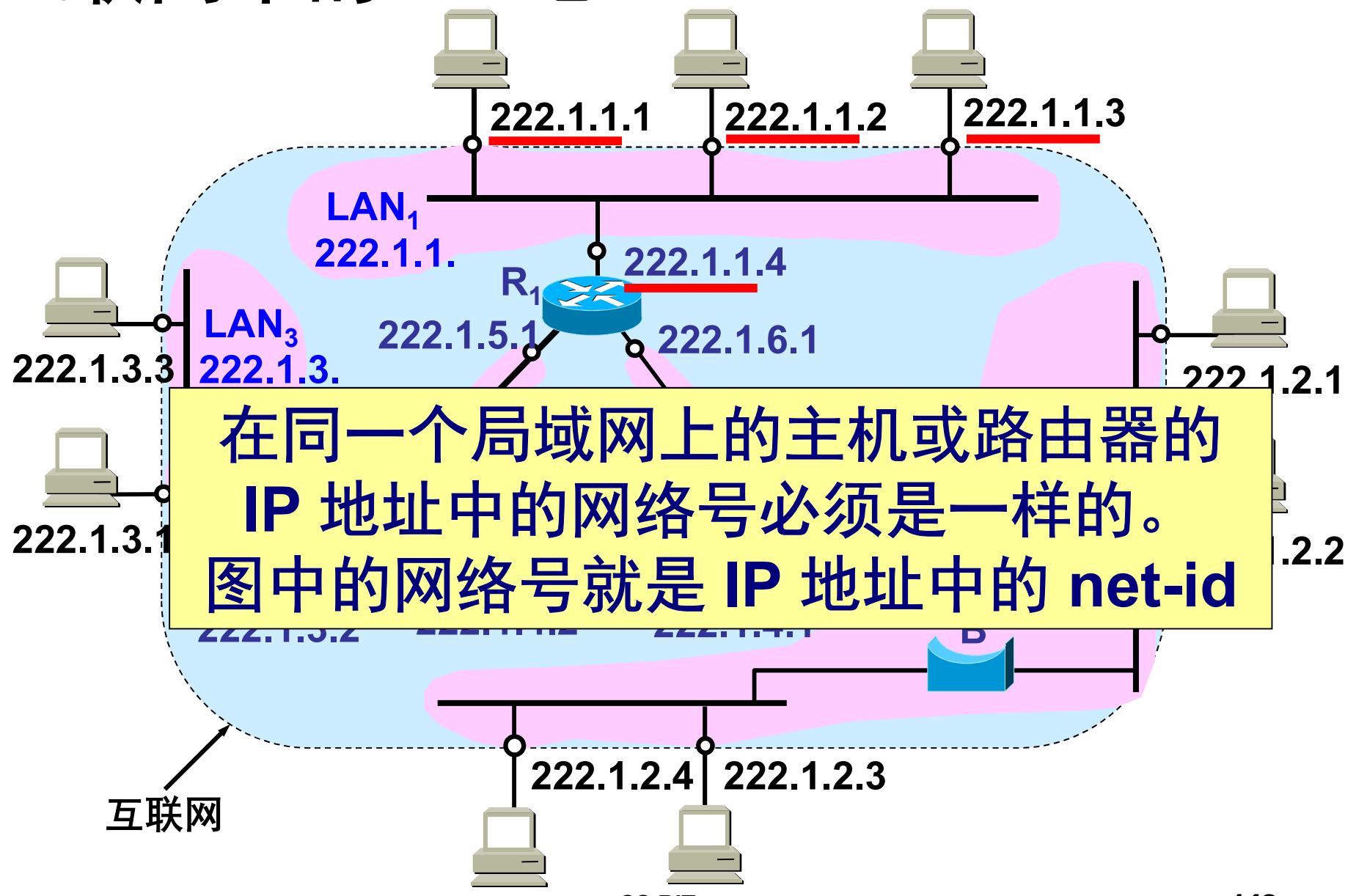
在同一个局域网上的主机或路由器的  
IP 地址中的网络号必须是一样的。  
图中的网络号就是 IP 地址中的 net-id



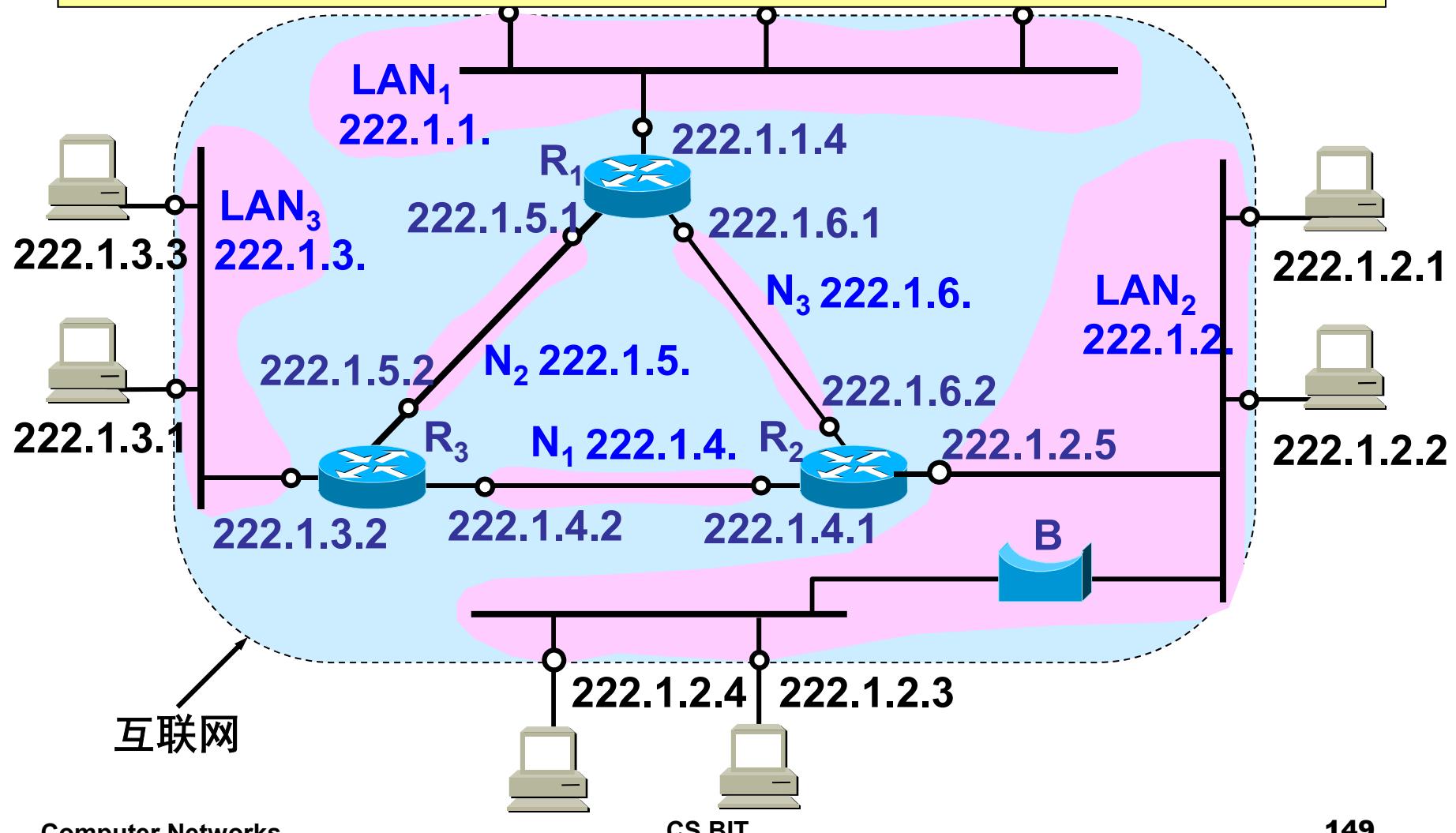
在同一个局域网上的主机或路由器的  
IP 地址中的网络号必须是一样的。  
图中的网络号就是 IP 地址中的 net-id



# 互联网中的 IP 地址

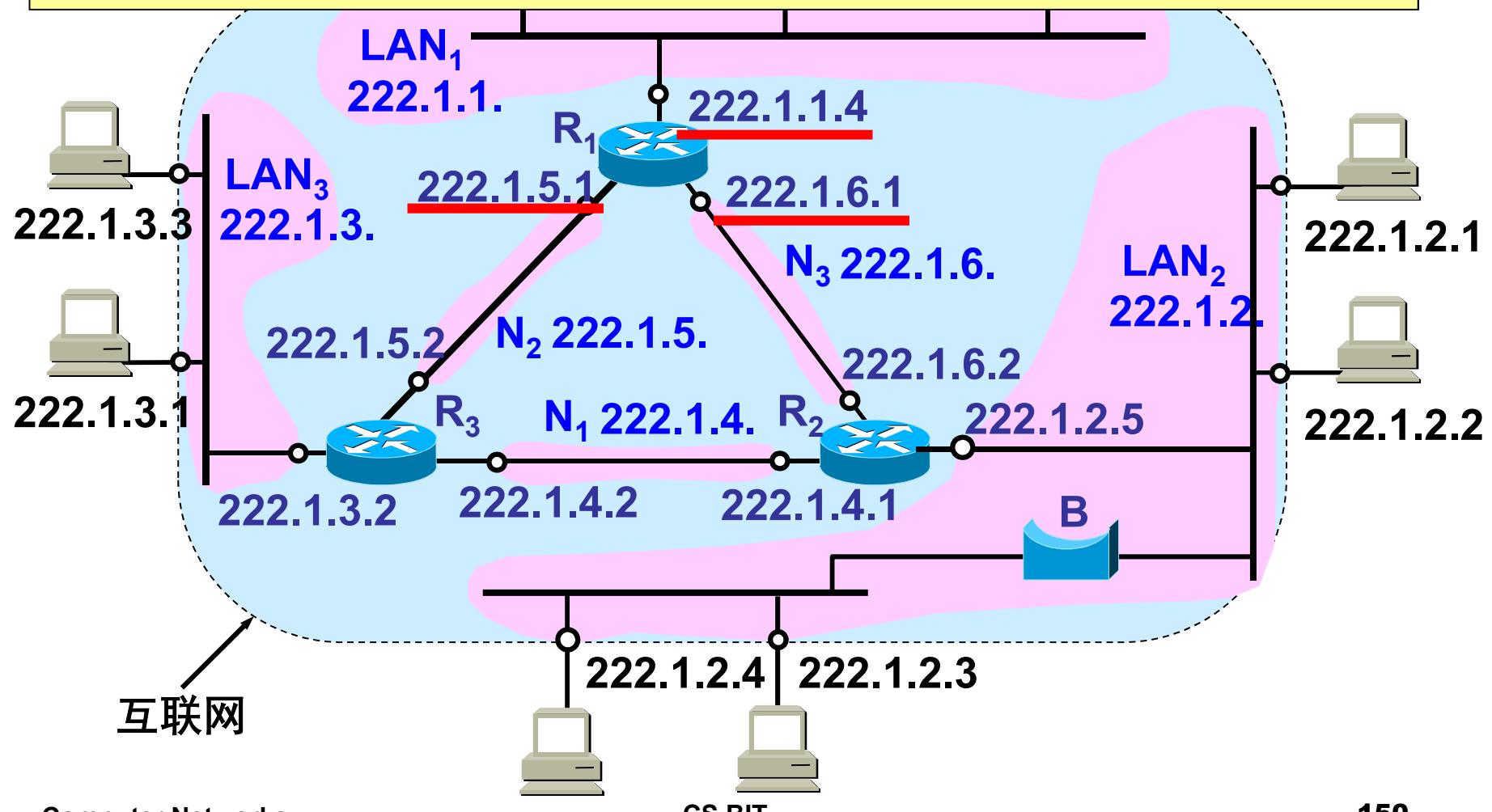


在同一个局域网上的主机或路由器的  
IP 地址中的网络号必须是一样的。  
图中的网络号就是 IP 地址中的 net-id



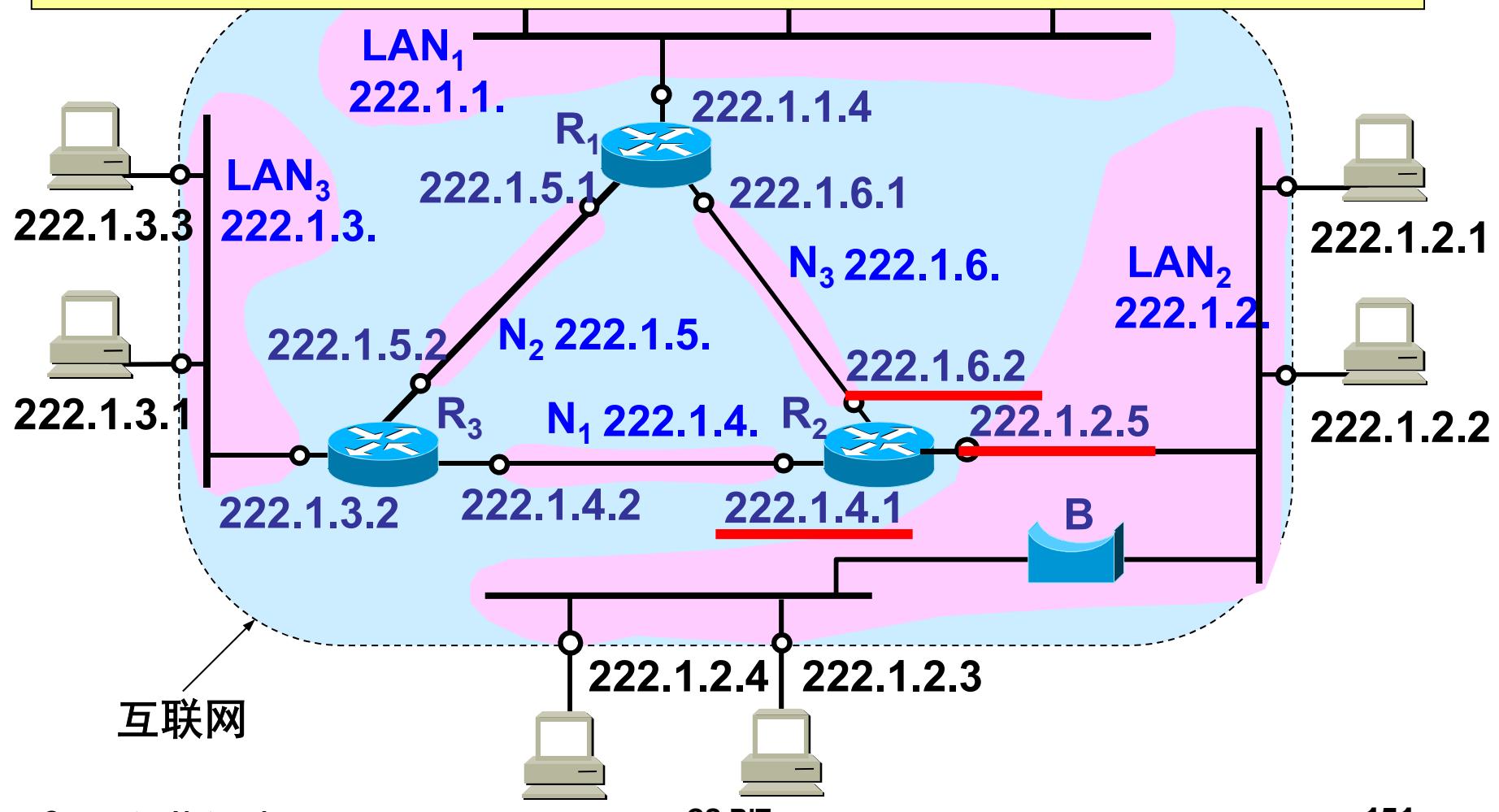
路由器总是具有两个或两个以上的 IP 地址。

路由器的每一个接口都有一个  
不同网络号的 IP 地址。



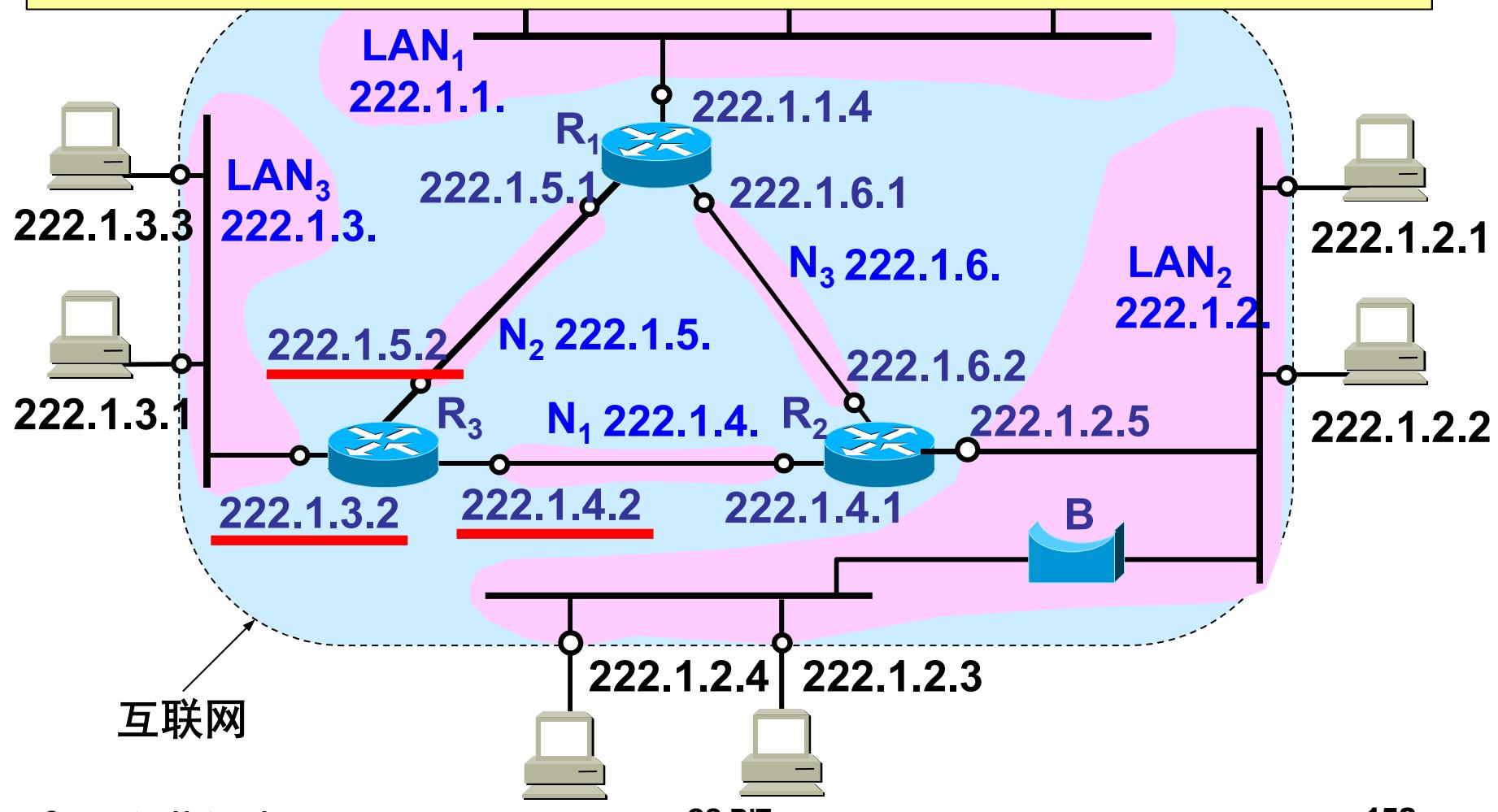
路由器总是具有两个或两个以上的 IP 地址。

路由器的每一个接口都有一个  
不同网络号的 IP 地址。



路由器总是具有两个或两个以上的 IP 地址。

路由器的每一个接口都有一个  
不同网络号的 IP 地址。



# Subnetting

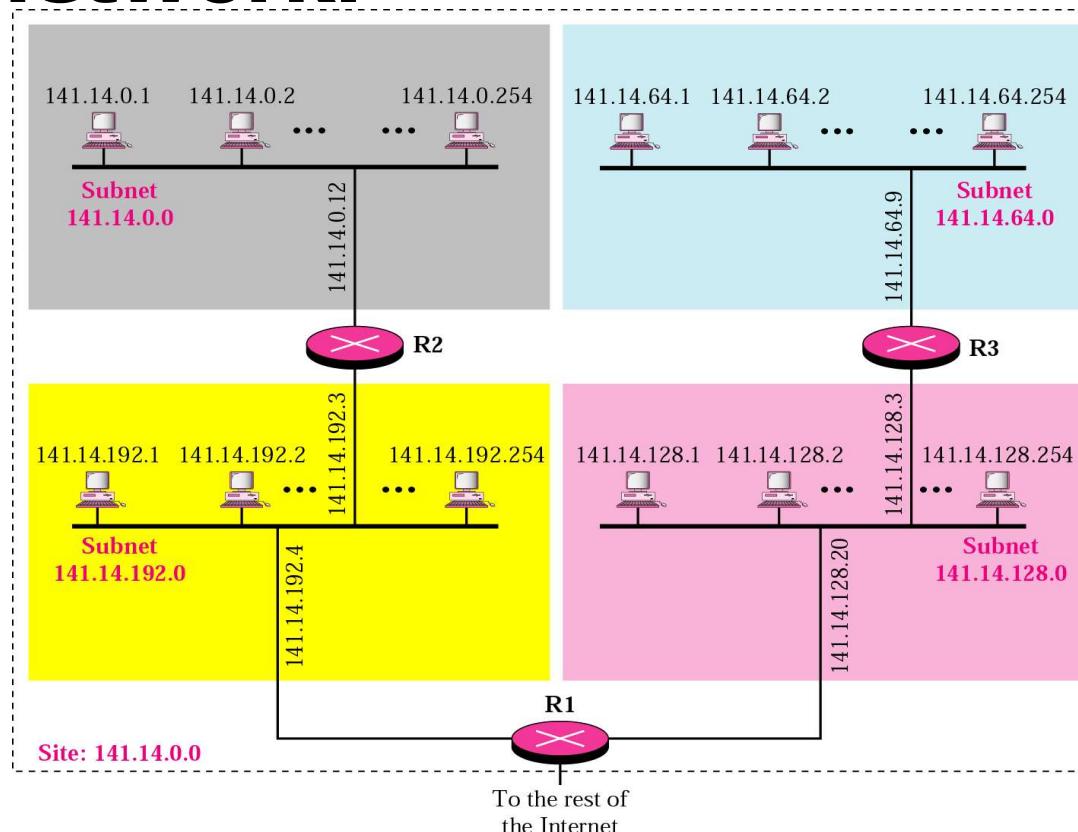
## ■ **Classful addressing Problems:**

**All hosts on the same network must have the same network number.**

□ **This may cause a single organization to acquire several classes of addresses (e.g. each time it adds a LAN), that would subsequently need to be announced worldwide (WHY?).**

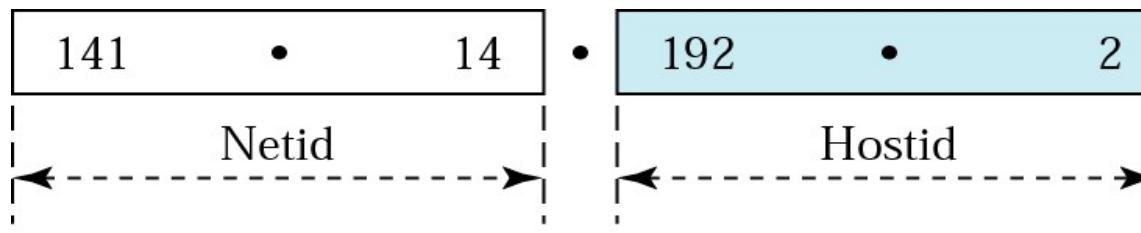
# Subnetting

- Divide a network into sub-networks while making the world knows only the main network.

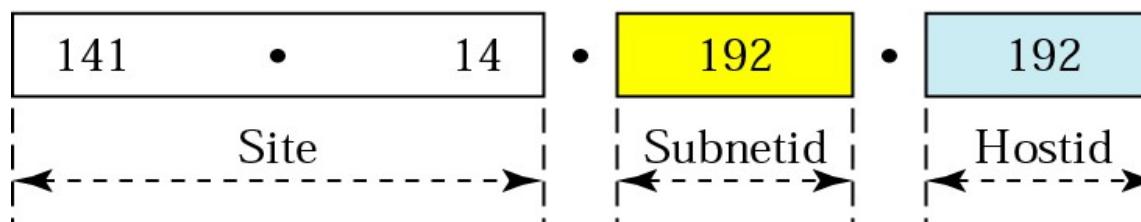


# Three Level Hierarchy

- Adding subnetworks creates an intermediate level of hierarchy in the IP addressing system.
  - The **site** is the first level.
  - The second level is the **subnet**.
  - The **host** is the third level.

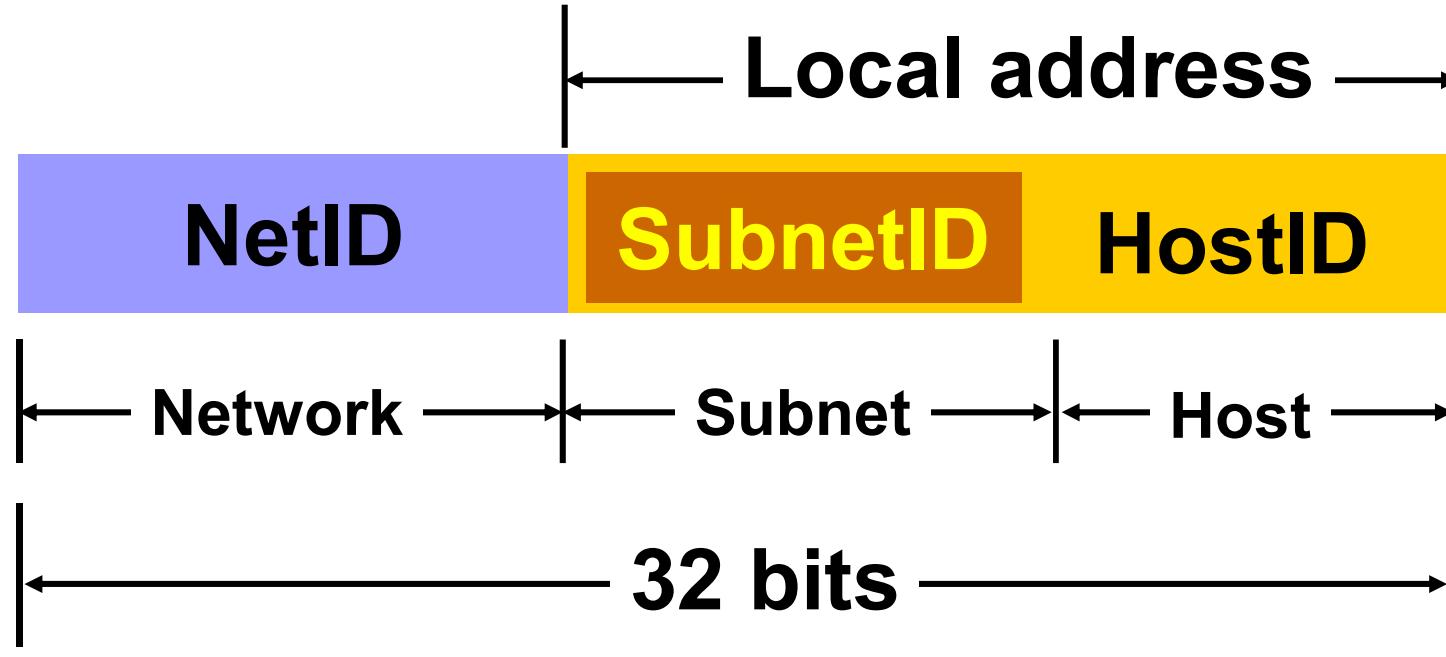


a. Without subnetting



b. With subnetting

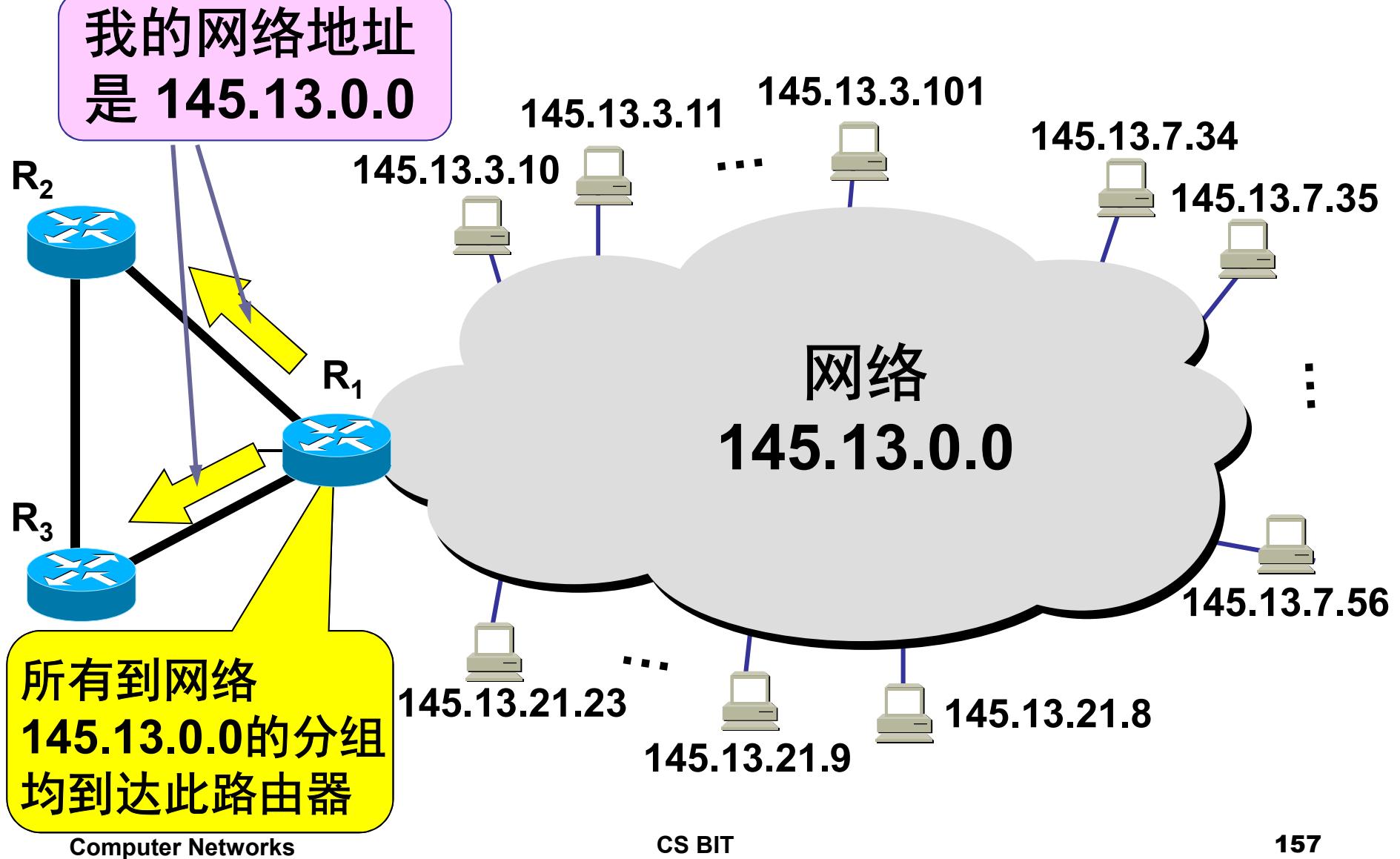
# Subnetting



Three Level Hierarchy

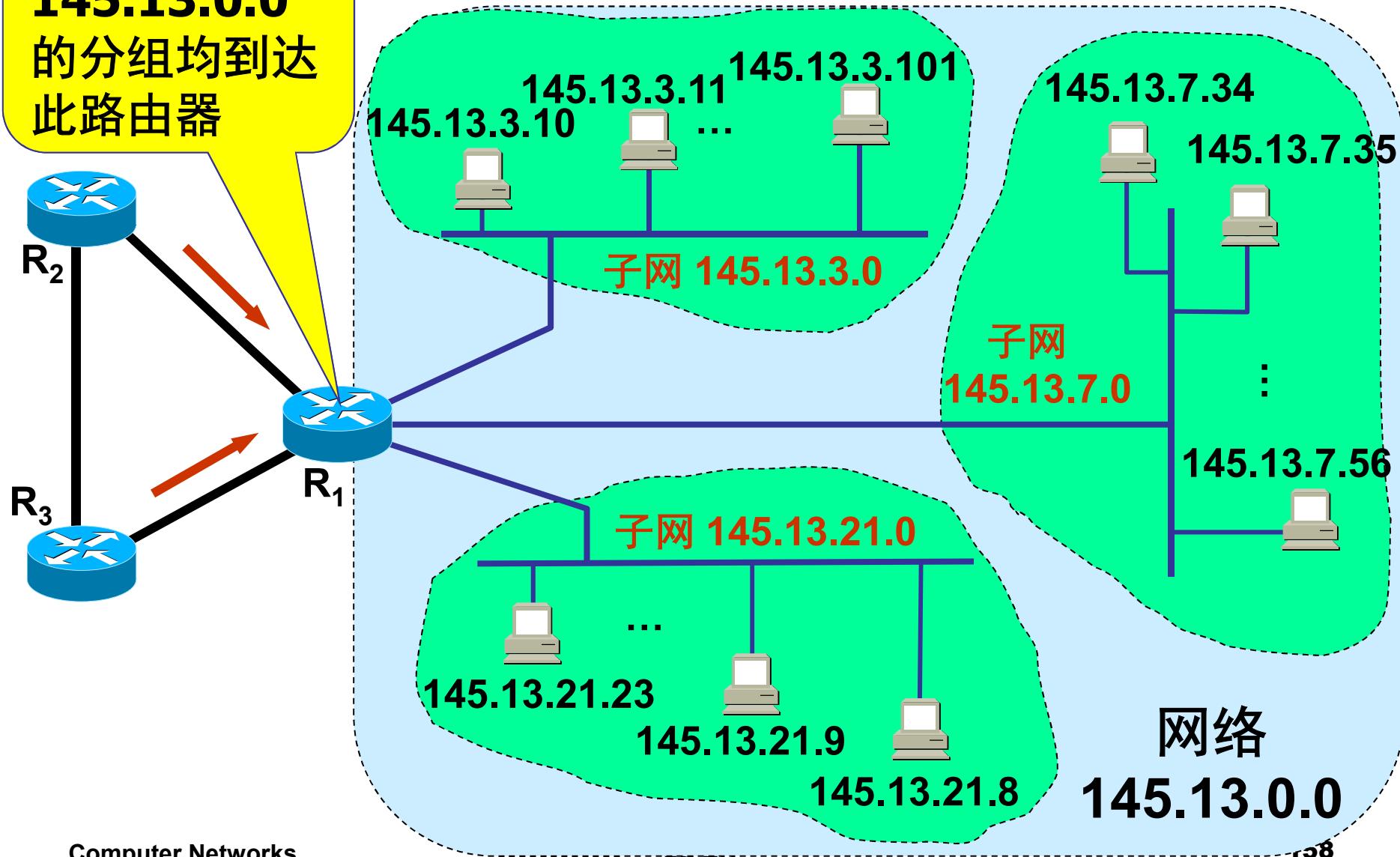
**Subnets are transparent to the Internet**

# Subnetting



# Subnetting

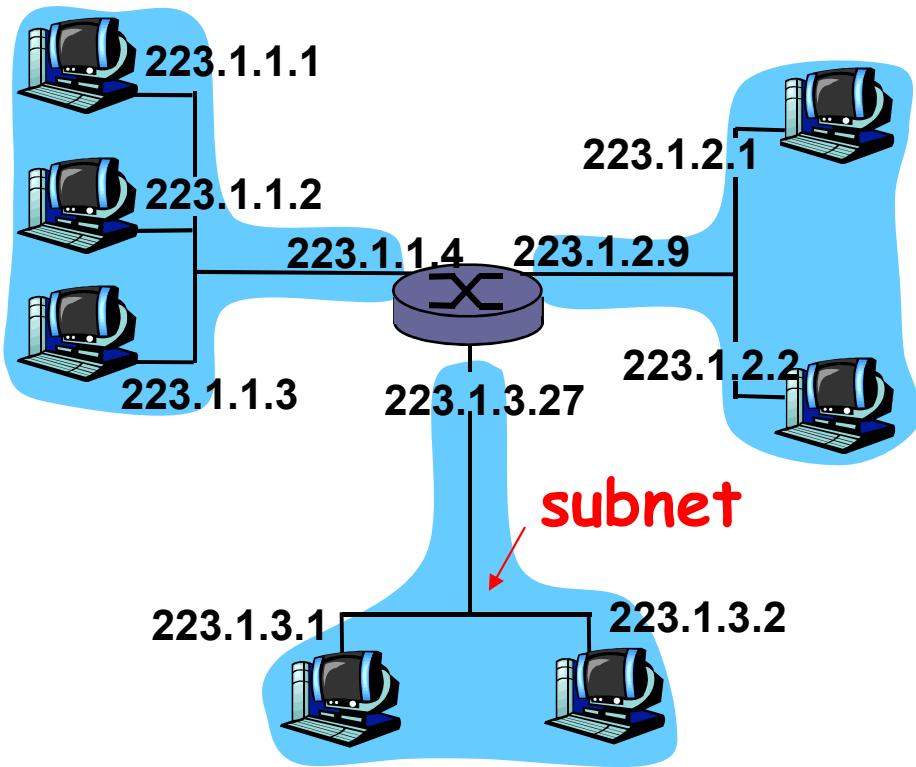
所有到达网络  
**145.13.0.0**  
的分组均到达  
此路由器



# Subnetting

- ***What's a subnet ?***

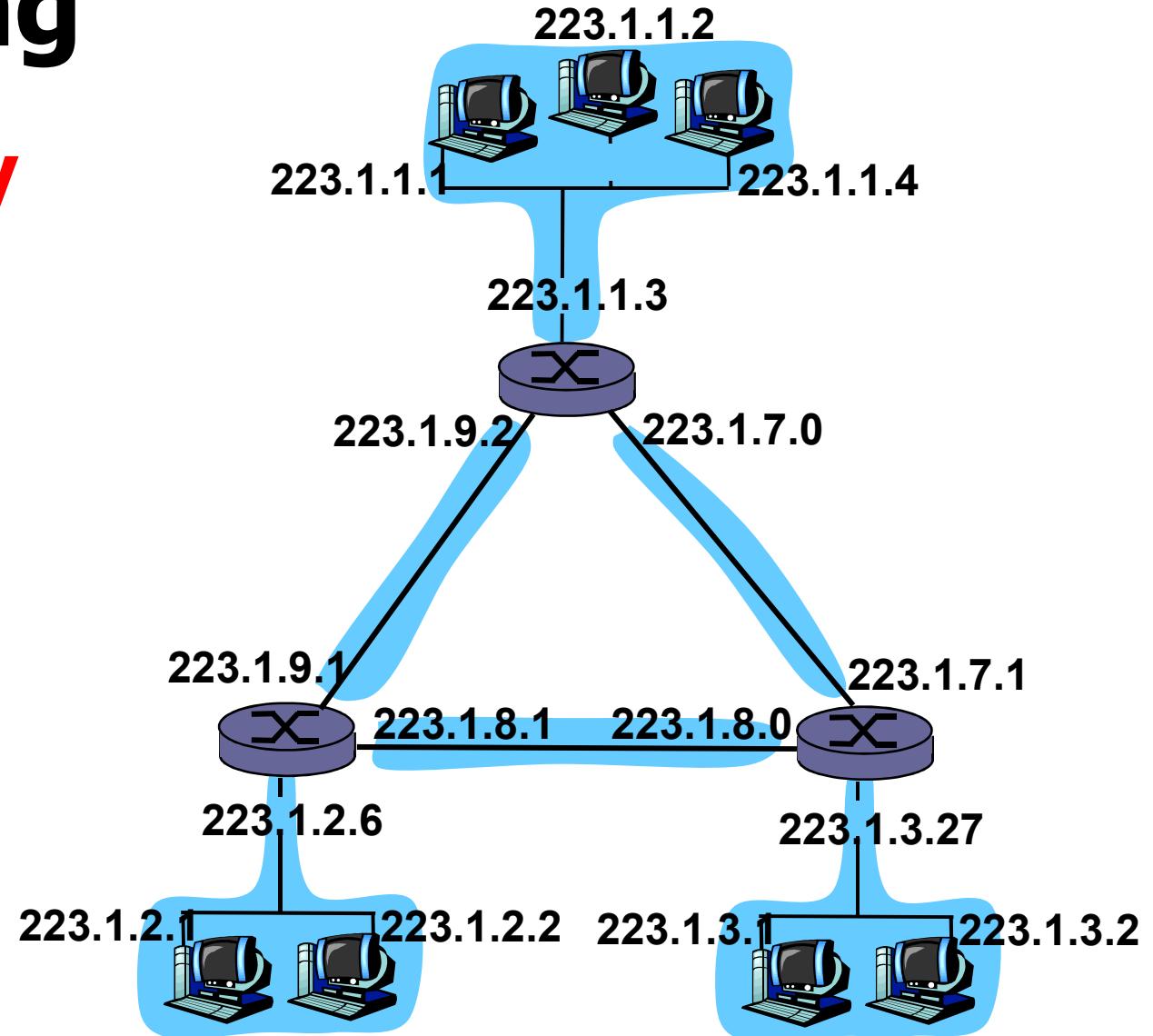
- device interfaces with same subnet part of IP address
  - can physically reach each other without intervening router

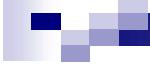


network consisting of 3 subnets

# Subnetting

**How many subnets ?**





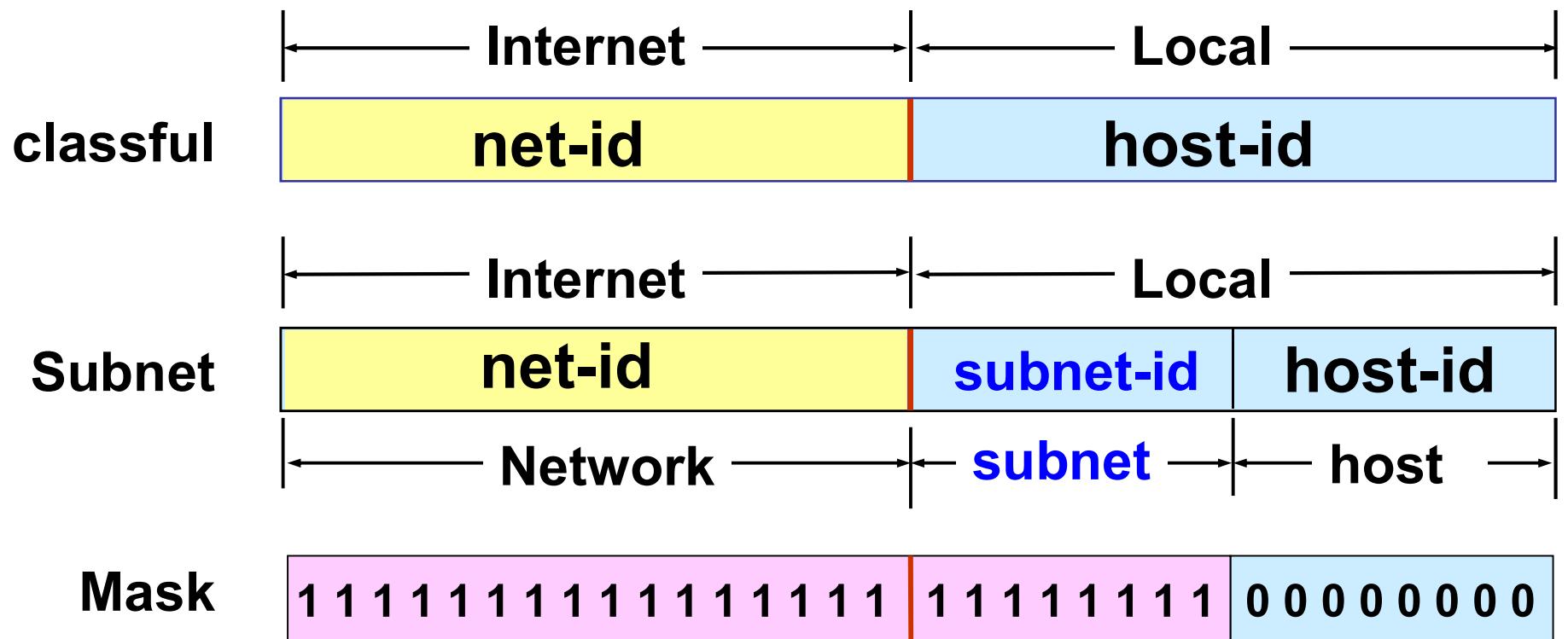
# Mask

- A router routes the packet based on network address and subnetwork address.
- A router **outside** a network routes based on network address.
- A router **inside** a network routes based on subnetwork address

# Mask

- Router uses the **32-bit mask** to identify the network address.
  - **One bit = 1:** corresponding bit in IP address is **NetID**.
  - **One bit = 0:** corresponding bit in IP address is **HostID**.

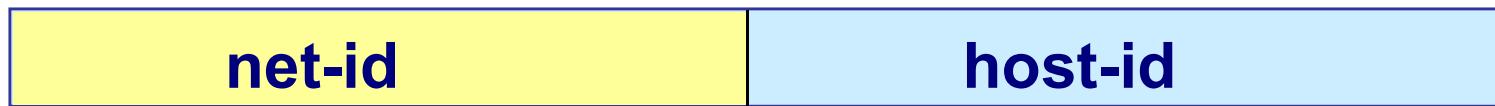
# Mask



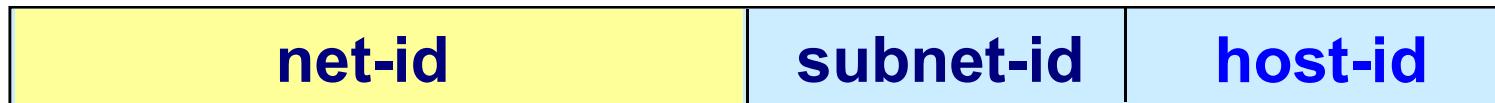
# Default Mask

# Network Address = (IP Address) AND (Mask)

classful



Subnet

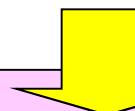
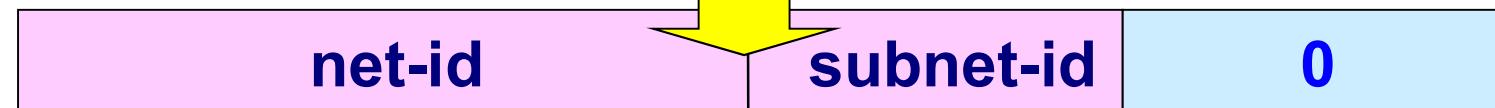


逐位进行 AND 运算

Mask



Network  
Address



# Mask

**Example:** IP address = 141.14.72.24, mask = 255.255.192.0. What's the network address?

(a) IP address

141	.	14	.	72	.	24
-----	---	----	---	----	---	----

(b) The 3rd octet in binary

141	.	14	.01001000.	24
-----	---	----	------------	----

(c) Mask= 255.255.192.0

11111111	11111111	11000000	00000000
----------	----------	----------	----------

(d) AND

141	.	14	.01000000.	0
-----	---	----	------------	---

(e) Network address

141	.	14	.	64	.	0
-----	---	----	---	----	---	---

# Mask

**Example: IP address = 141.14.72.24, mask = 255.255.224.0. What's the network address?**

(a) IP address

141	.	14	.	72	.	24
-----	---	----	---	----	---	----

(b) The 3rd octet in binary

141	.	14	.01001000.	24
-----	---	----	------------	----

(c) Mask= 255.255.224.0

11111111	11111111	11100000	00000000
----------	----------	----------	----------

(d) AND

141	.	14	.01000000.	0
-----	---	----	------------	---

(e) Network address

141	.	14	.	64	.	0
-----	---	----	---	----	---	---

# Mask

## Example

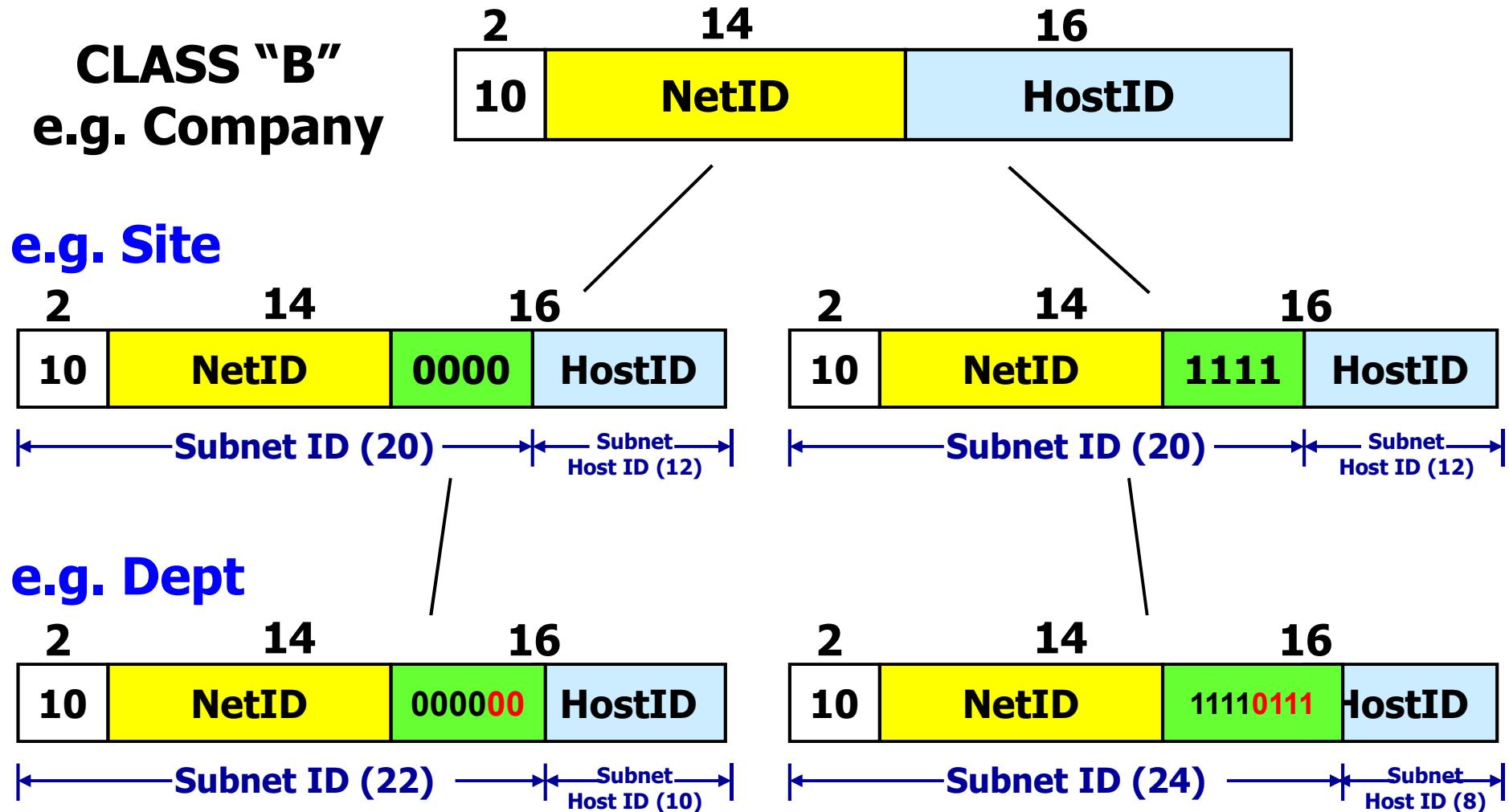
A router inside the organization receives the same packet with destination address 190.240.33.91. Assume the mask=255.255.224.0.

Show how it finds the subnetwork address to route the packet.

## Solution

- 1.The router applies the mask to the address, 190.240.33.91. The subnet address is **190.240.32.0**.
- 2.The router looks in its routing table to find how to route the packet to this destination.

# Subnetting



# Subnet Example

Network address **190.52.0.0** with /16 network mask

Using Subnets: subnet mask **255.255.255.0** or /24

Network	Network	Subnet	Host
190	52	0	Host
190	52	1	Host
190	52	2	Host
190	52	3	Host
190	52	Etc.	Host
190	52	254	Host
190	52	255	Host

**Subnets**

**255 Subnets**

**$2^8 - 1$**

**Cannot use last subnet as it contains broadcast address**

# Subnet Example

**Subnet 0 (all 0's subnet) issue:** The address of the subnet, 190.52.0.0/24 is the same address as the major network, 190.52.0.0/16.

Network	Network	Subnet	Host
---------	---------	--------	------

190	52	0	Host
190	52	1	Host
190	52	Etc.	Host
190	52	254	Host

Subnets

255  
Subnets

$2^8 - 1$

190	52	255	Host
-----	----	-----	------

**Last subnet (all 1's subnet) issue:** The broadcast address for the subnet, 190.52.255.255 is the same as the broadcast address as the major network, 190.52.255.255.

# Forwarding on Subnet

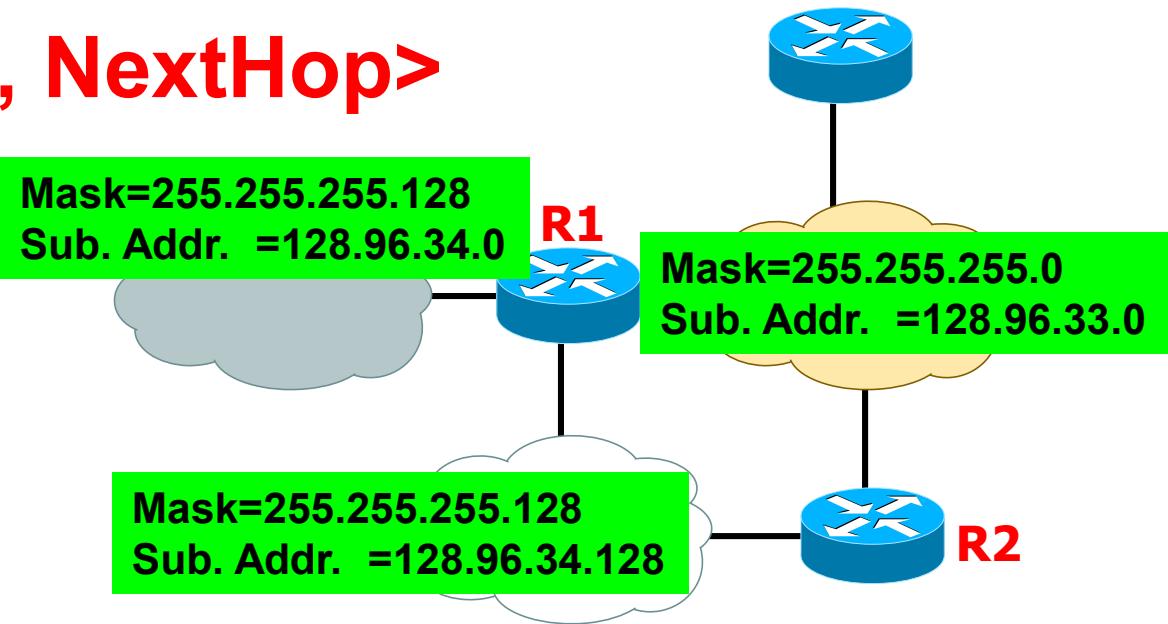
- **Resolution of subnet address:**
  - Bitwise ANDing Host IP address with Subnet Mask gives subnet number.
- **When send an IP packet:**
  - Perform BITwise AND between subnet mask and destination IP address
  - **If result == its subnet addr. , destination is on same subnet, send packet to the dest.**
  - **If not, send packet to default router R.**

# Router Forwarding Tables

- Table holds entries like  
**<Subnet, Mask, NextHop>**

- Router **ANDs** dest addr with subnet mask of each entry.

- Find the right entry (Match with subnet no.) and forward to Next hop.



Router R1's Routing table

Subnet	Mask	Next Hop
128.96.34.0	255.255.255.128	Int 0
128.96.34.128	255.255.255.128	Int 1
128.96.33.0	255.255.255.0	R2

# Forwarding Algorithm

```
D = destination IP address
for each entry (Subnet, Mask, NextHop)
    D1 = Mask & D
    if D1 = Subnet
        if NextHop is an interface
            deliver datagram directly to D
        else
            deliver datagram to NextHop
        end if
    end if
end for
```

Use a *default router* if nothing matches

# More about Subnetting

- Subnet Mask **need not align with byte boundaries** (e.g. **255.255.255.128**) -- 7 zeroes.
- **Non-contiguous** mask can be used
  - They make administration more difficult
  - e.g. **255.255.1.0**
- One could have multiple subnets on the same physical network ! **However**, now, hosts on the same network would need to go **through a router** in order to talk to each other.

# More about Subnetting

- Subnetting help solve scalability problems
  - Do not require us to use class B or C address for each physical network
- Subnetting is **not** the only way to solve scalability problems
- Additional router support is necessary to include mask and forwarding functionality

# More about Subnetting

- Routers outside a group of subnets see the group as a single network -- e.g. **178.96.0.0**
- **However**, once packet arrives to the group, routers within the group need to forward the packets to the proper subnet.

# Problems with Classful Addresses

- The different classes were different sizes:
  - Less than 17,000 class B network addresses
  - More than 2,000,000 class C network addresses
- The classes differed in popularity:
  - Class B addresses were very popular and almost exhausted
  - Class C addresses were hardly used at all

# **Problems with Classful Addresses**

- Potential exhaustion of IPv4 address space (due to inefficiency)
  - Class B network numbers are highly prized
    - Not everyone needs one
  - Lots of class C addresses but no one wants them
- Growth of back bone routing tables
  - We don't want lots of small networks since this causes large routing tables
  - Route calculation and management requires high computational overhead

# Problems with Classful Addresses

## ■ Example:

- If a network grows to more than 255 hosts, it may want a Class B address.
- One class B address: waste
- many Class C addresses: for this one network, each router has to maintain multiple routing entries.

# Problems with IPv4 Addresses

## ■ Solution:

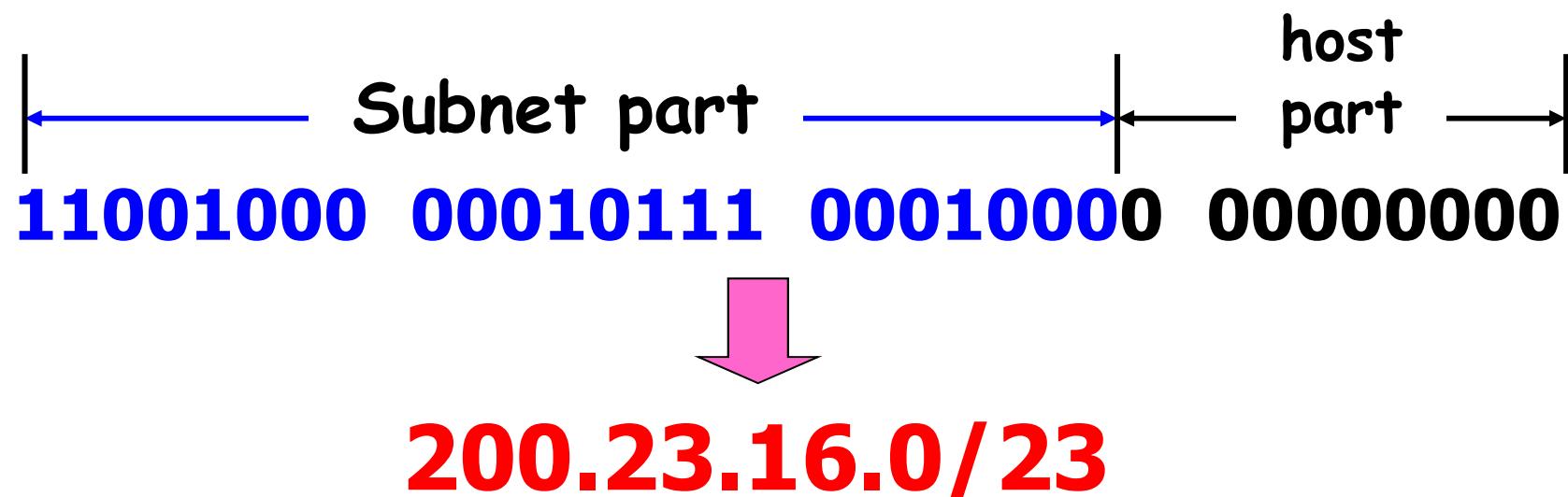
- Allow addresses assigned to a single entity to span multiple classed prefixes
- Enhance route aggregation

## ■ CIDR

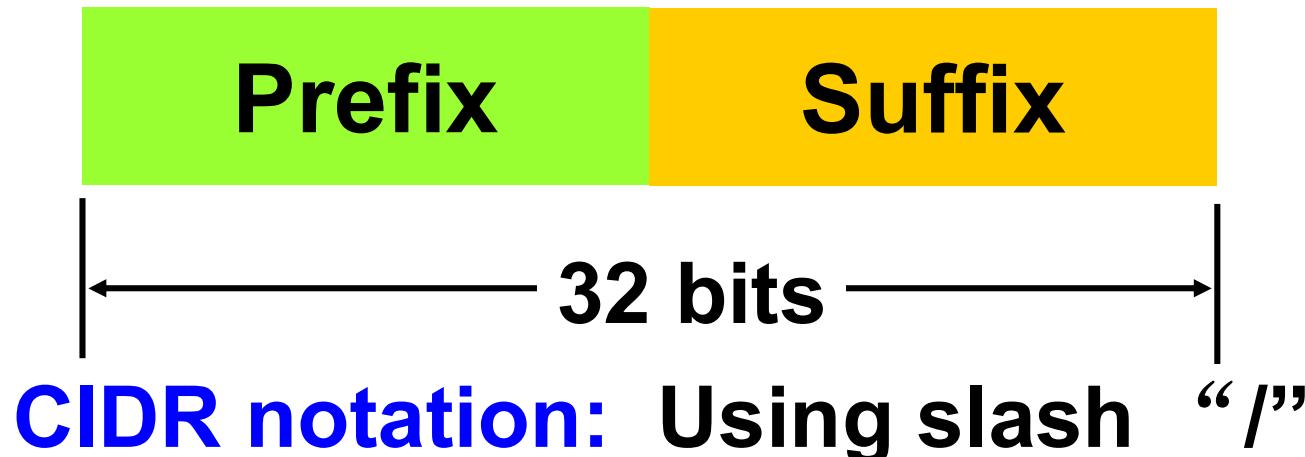
- an attempt to balance the desire to minimize the number of routes that a router needs to know versus the need to hand out addresses efficiently

# Classless Addressing: CIDR

- **CIDR: Classless InterDomain Routing**
  - subnet portion of address of **arbitrary length**
  - address format: **a.b.c.d/x**, where x is # of bits in subnet portion of address



# Classless Addressing: CIDR



**a.b.c.d / x**

(x is the length of prefix)

**Prefix** – common to all addresses in the block

**Suffix** – differentiates unique address in the block

**By default:**

Class A: a.b.c.d / 8

Class B: a.b.c.d / 16

Class C: a.b.c.d / 24

# Classless Addressing: CIDR

- Using CIDR, an organization can combine several class C blocks to create a larger range of addresses.
- Several networks are combined to create a **supernetwork**.
- Called **Supernetting**.

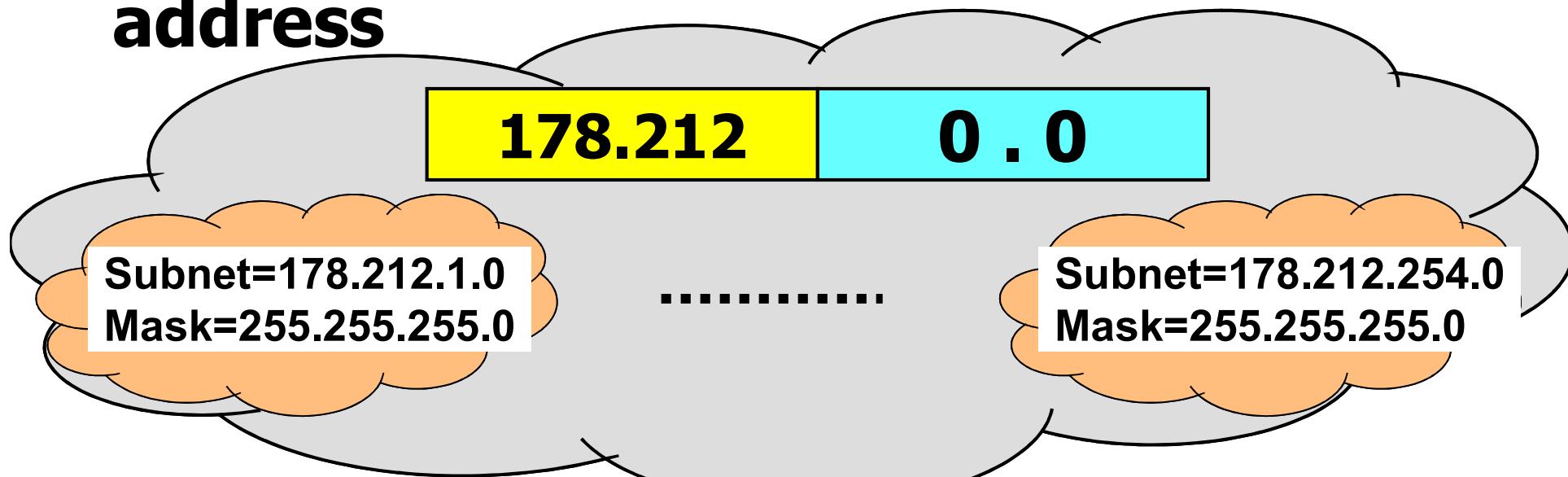
# Supernetting

- **Subnetting** allows an organization to share a single IP network address among multiple physical networks
- **Supernetting** allows the addresses assigned to an organization to span multiple IP network addresses
  - assign an organization a block of plentiful addresses (class C) rather than a single scarce (class B) address

# Supernetting

## Example:

- The organization would prefer a class B address

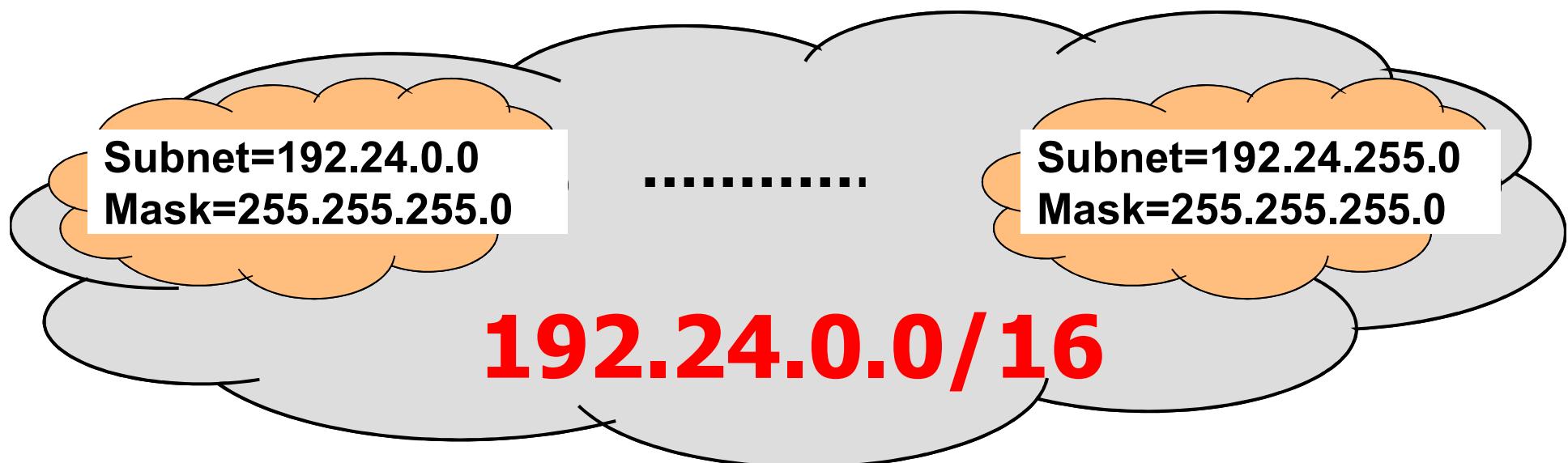


This would allow the organization to have 254 physical networks with up to 254 hosts per network.

# Supernetting

## Example (cont):

- Instead of getting a class B address, the organization can be given 256 contiguous class C addresses
  - E.g. **192.24.0.0 – 192.24.255.0**



所有地址的前16 bit前缀都是一样的

最小地址 →	11000000 00011000 ... 11000000 00011000 11000000 00001100 ... 11000000 00001100 ... 11000000 00001100 11000000 00001100 11000000 00001100 ... 11000000 00001100	00000000 00000000 ... 00000000 11111111 00000001 00000000 ... 00000001 11111111 ... 11111110 00000000 11111110 11111111 11111111 00000000 ... 11111111 11111111
--------	--	--



# CIDR

- Assign block of contiguous addresses to nearby networks
- Prefix represent a set of addresses that are allocated
- Restrict block sizes to powers of 2,  
**a.b.c.d/x** represent blocks with a single pair

**(first\_network\_address, count)**

## Examples: Finding the first address

- What is the first address in the block if one of the addresses is 167.199.170.82 / 27?
- Solution:

Address in binary:

10100111 11000111 10101010 01010010

Keep the left 27 bits:

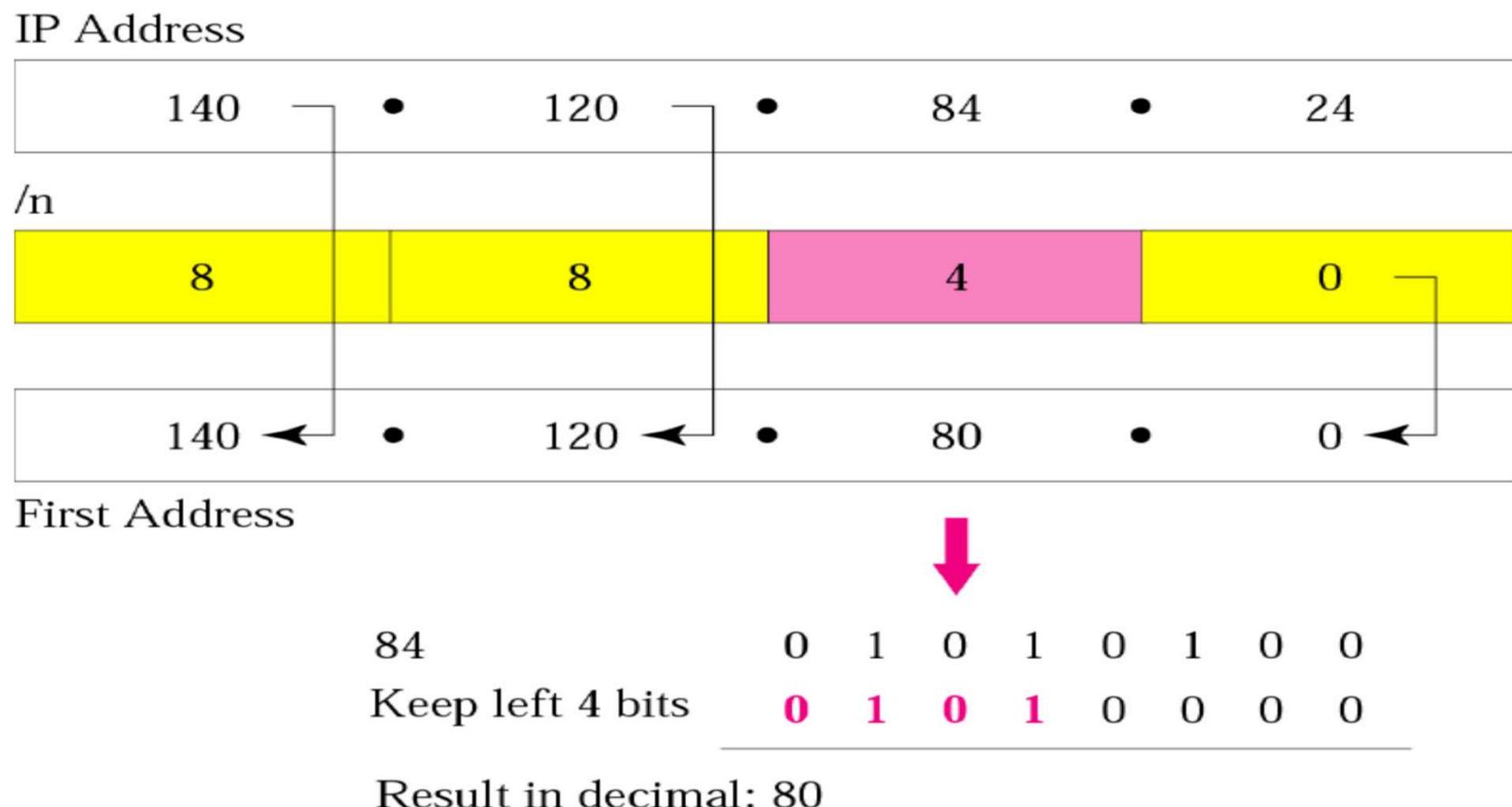
10100111 11000111 10101010 01000000

Result in CIDR notation:

167.199.170.64/27

# Examples: Finding the first address

- What is the first address in the block if one of the addresses is 140.120.84.24 / 20?



## **Examples: Finding the last address**

- To the first address, add the number of addresses, minus one

**OR**

- Set all bits that are not part of the CIDR prefix to 1

## Examples: Finding the last address

- **Find the number of addresses in the block if one of the addresses is 140.120.84.24 / 20.**
- **Solution:** The prefix length is 20. The number of addresses in the block is  $2^{32-20}$  or  $2^{12}$  or 4096.  
Note that this is a large block with 4096 addresses.

## Examples: Finding the last address

- Find the last address in the block if one of the addresses is 140.120.84.24 / 20.
- Solution
  - The first address = 140.120.80.0.
  - The number of addresses is 4096.
  - To find the last address, we need to add 4095 (4096 - 1) to the first address. OR,
  - Set all bits that are not part of the CIDR prefix to 1  
 $140.120.(0101\ 1111)_2.(1111\ 1111)_2 =$   
 $140.120.95.255$

# Effect of CIDR on Routing

192.24.0  
192.24.1  
192.24.2  
192.24.3  
192.24.4  
192.24.5  
192.24.6  
192.24.7  
192.24.8  
192.24.9  
192.24.10  
192.24.11  
192.24.12  
192.24.13  
192.24.14  
192.24.15

**192.24.0.0 / 20**

**One entry in routing table.  
Mask=255.255.240.0**

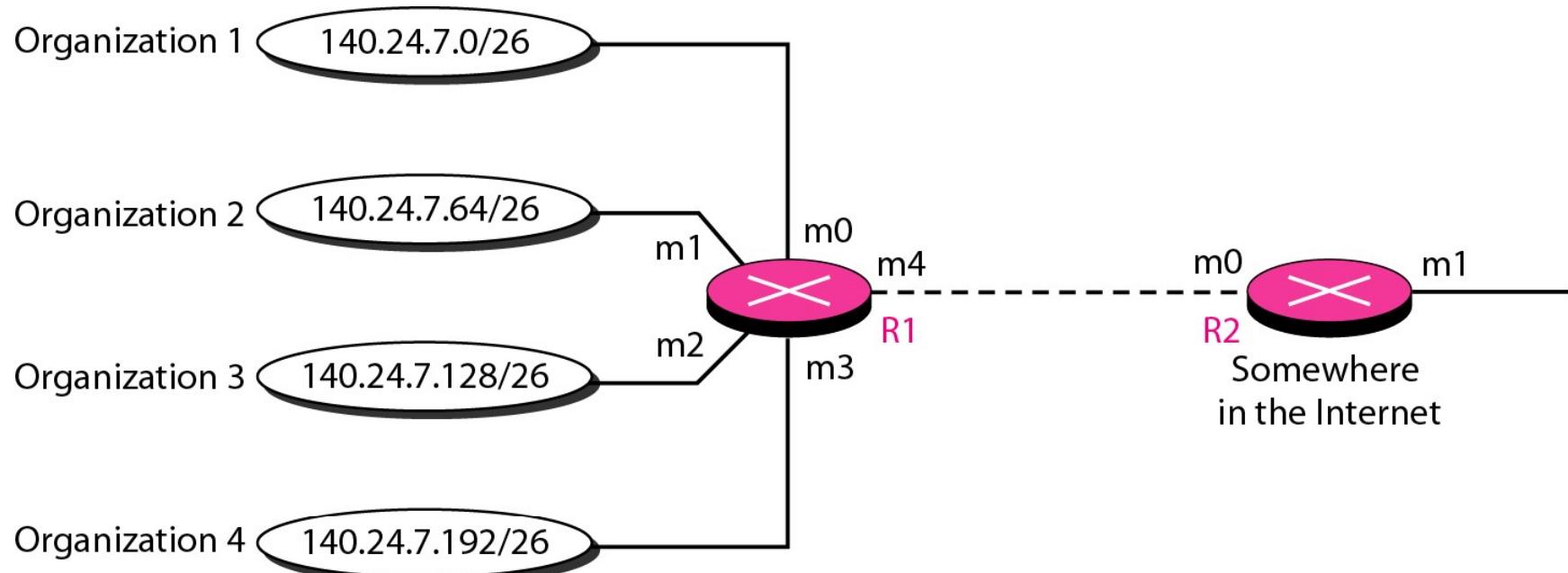
**Route aggregation**

**16 Class C addresses, 16 entries in routing table**

# Route aggregation

- Route aggregation is an alternate term for **route summarization**, which is a method used to **minimize the number of routing tables** required in an IP network from a set of child routes falling under a common parent prefix.

# Route aggregation



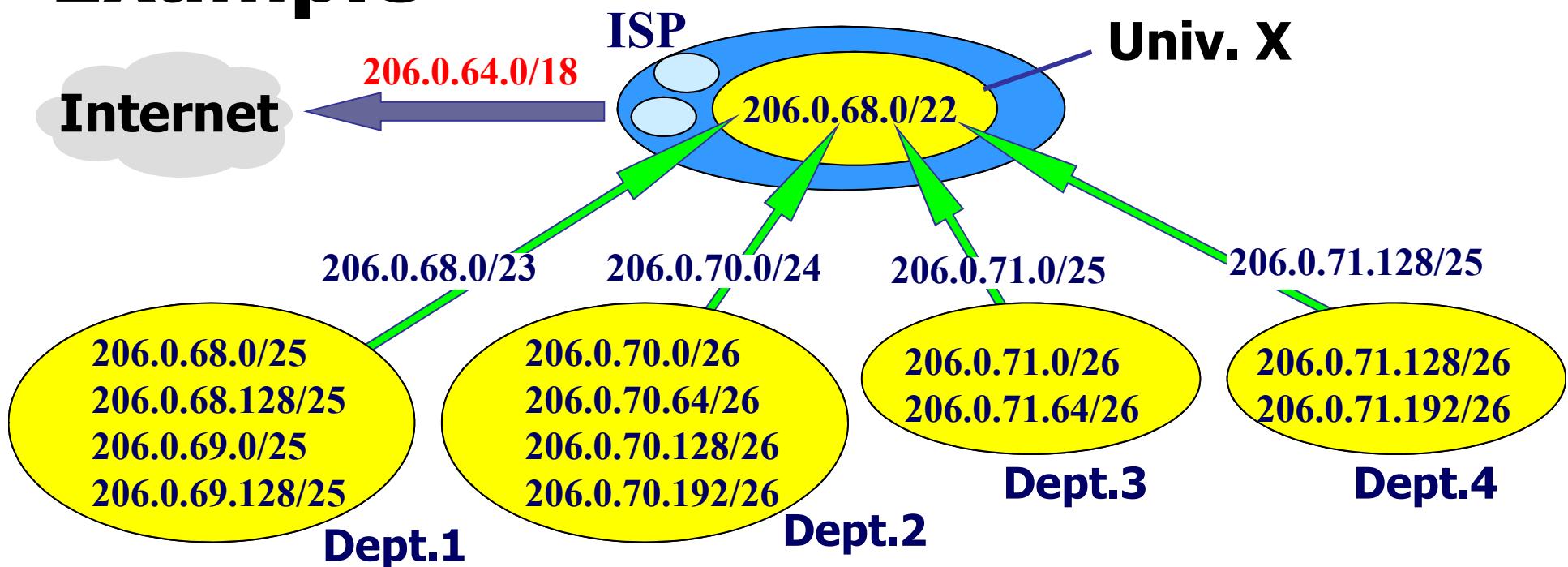
Mask	Network address	Next-hop address	Interface
/26	140.24.7.0	-----	m0
/26	140.24.7.64	-----	m1
/26	140.24.7.128	-----	m2
/26	140.24.7.192	-----	m3
/0	0.0.0.0	Default	m4

Routing table for R1

Mask	Network address	Next-hop address	Interface
/24	140.24.7.0	-----	m0
/0	0.0.0.0	Default	m1

Routing table for R2

# Example



单位	地址块	二进制表示	地址数
ISP	<b>206.0.64.0/18</b>	<b>11001110.00000000.01*</b>	<b>16384</b>
Univ	<b>206.0.68.0/22</b>	<b>11001110.00000000.010001*</b>	<b>1024</b>
Dept1	<b>206.0.68.0/23</b>	<b>11001110.00000000.0100010*</b>	<b>512</b>
Dept2	<b>206.0.70.0/24</b>	<b>11001110.00000000.01000110.*</b>	<b>256</b>
Dept3	<b>206.0.71.0/25</b>	<b>11001110.00000000.01000111.0*</b>	<b>128</b>
Dept4	<b>206.0.71.128/25</b>	<b>11001110.00000000.01000111.1*</b>	<b>128</b>

# Example

Internet

206.0.64.0/18



Univ. X

互联网 ISP 路由器转发表

不采用  
CIDR 时  
的路由表

目的	下一跳
206.0.64.0/24	ISP
206.0.65.0/24	ISP
.....	
206.0.127.0/24	ISP
a.b.c.d/??	其他 ISP

ISP 内路由器转发表

目的	下一跳
206.0.68.0/25	大学 X
206.0.68.128/25	大学 X
206.0.69.0/25	大学 X
206.0.69.128/25	大学 X
其他系 ...	大学 X
w.x.y.z/??	其他

采用  
CIDR 后  
的路由表

目的	下一跳
206.0.64.0/18	ISP
a.b.c.d/??	其他 ISP

ISP 共有 64 个 C 类

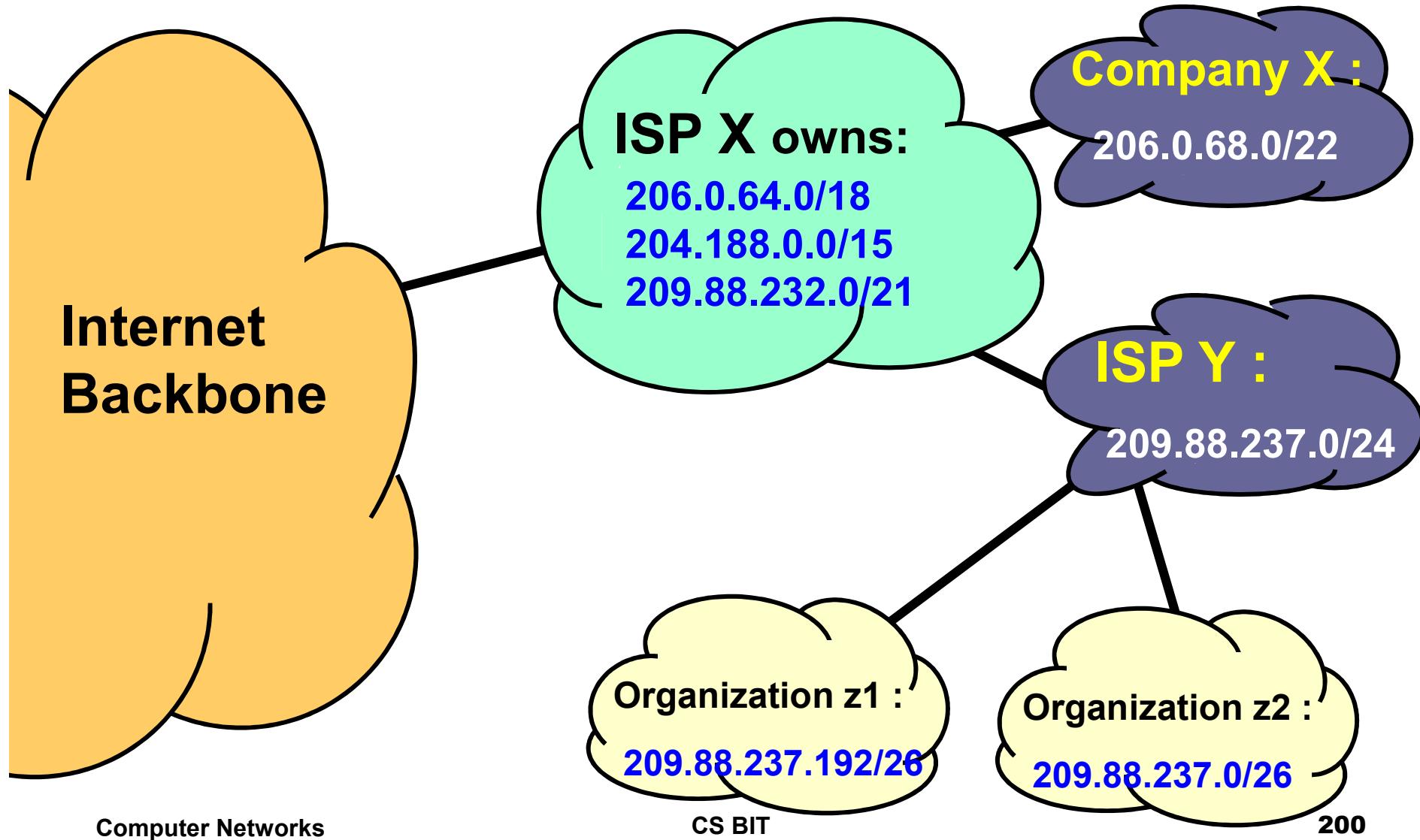
网络。

如果不采用 CIDR 技术，则在与该 ISP 的路由器交换路由信息的每一个路由器的路由表中，就需要有 64 个项目。

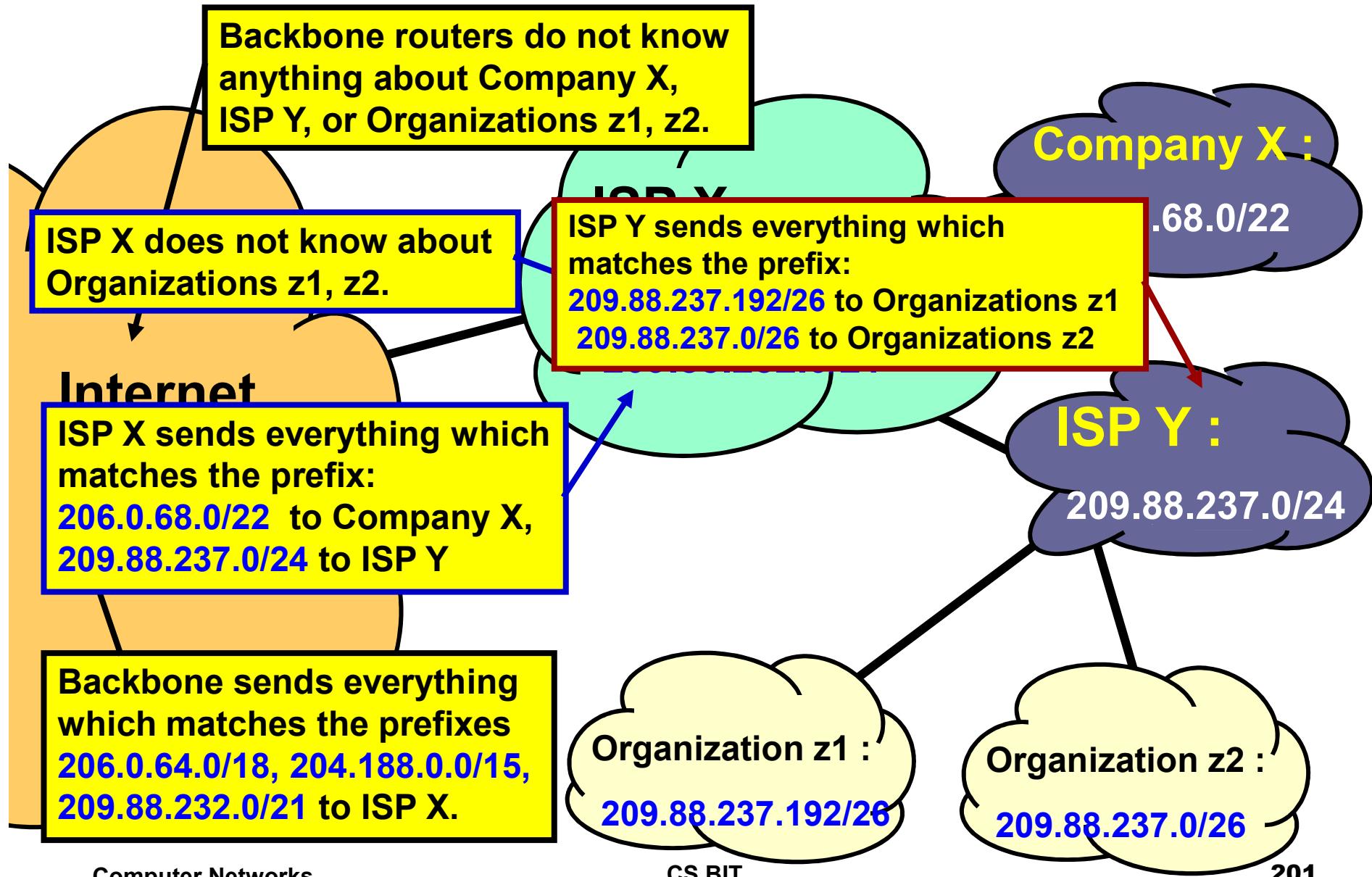
但采用地址聚合后，只需用路由聚合后的 1 个项目

**206.0.64.0/18** 就能找到该 ISP。

# CIDR and Routing Information

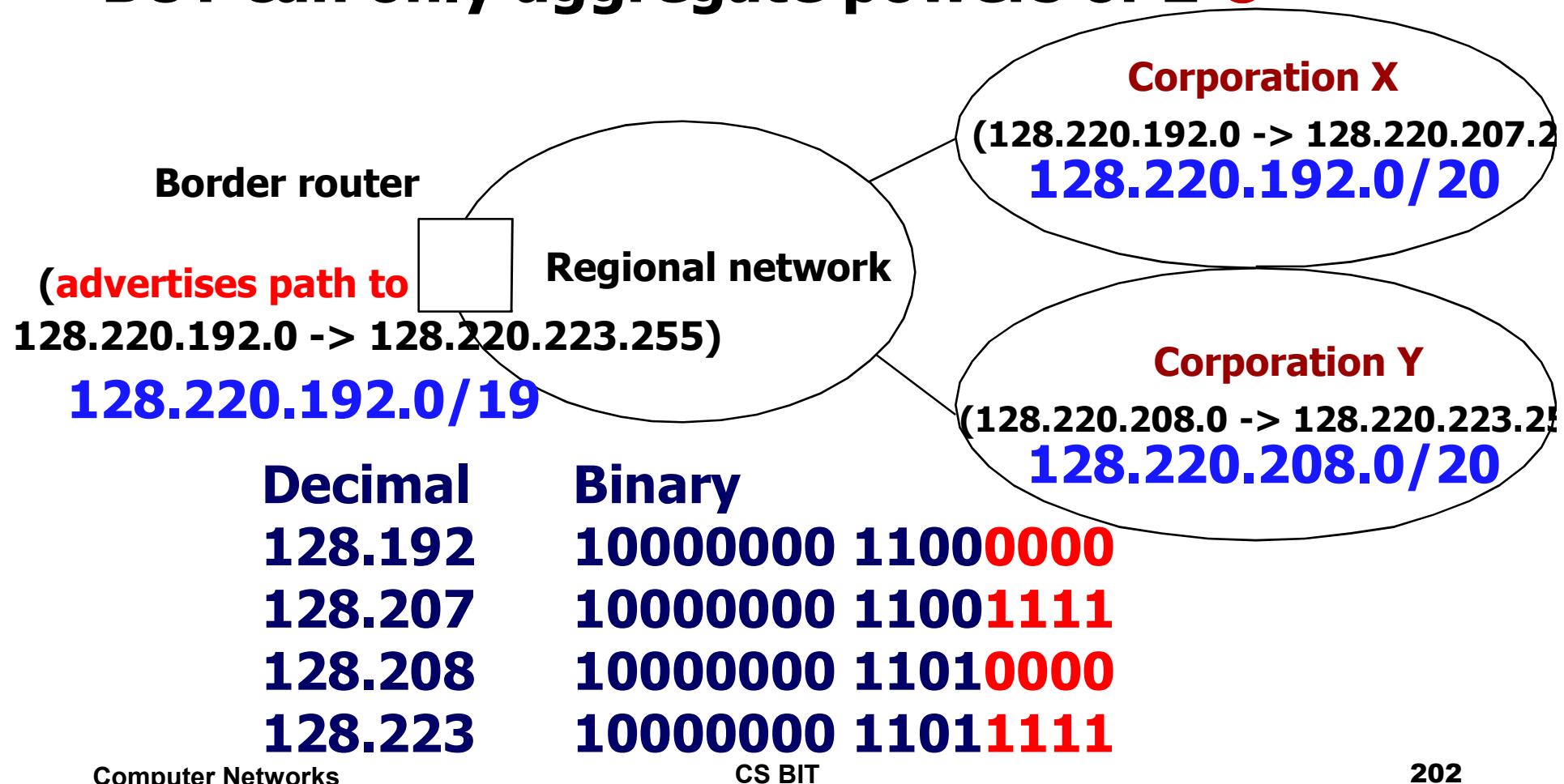


# CIDR and Routing Information



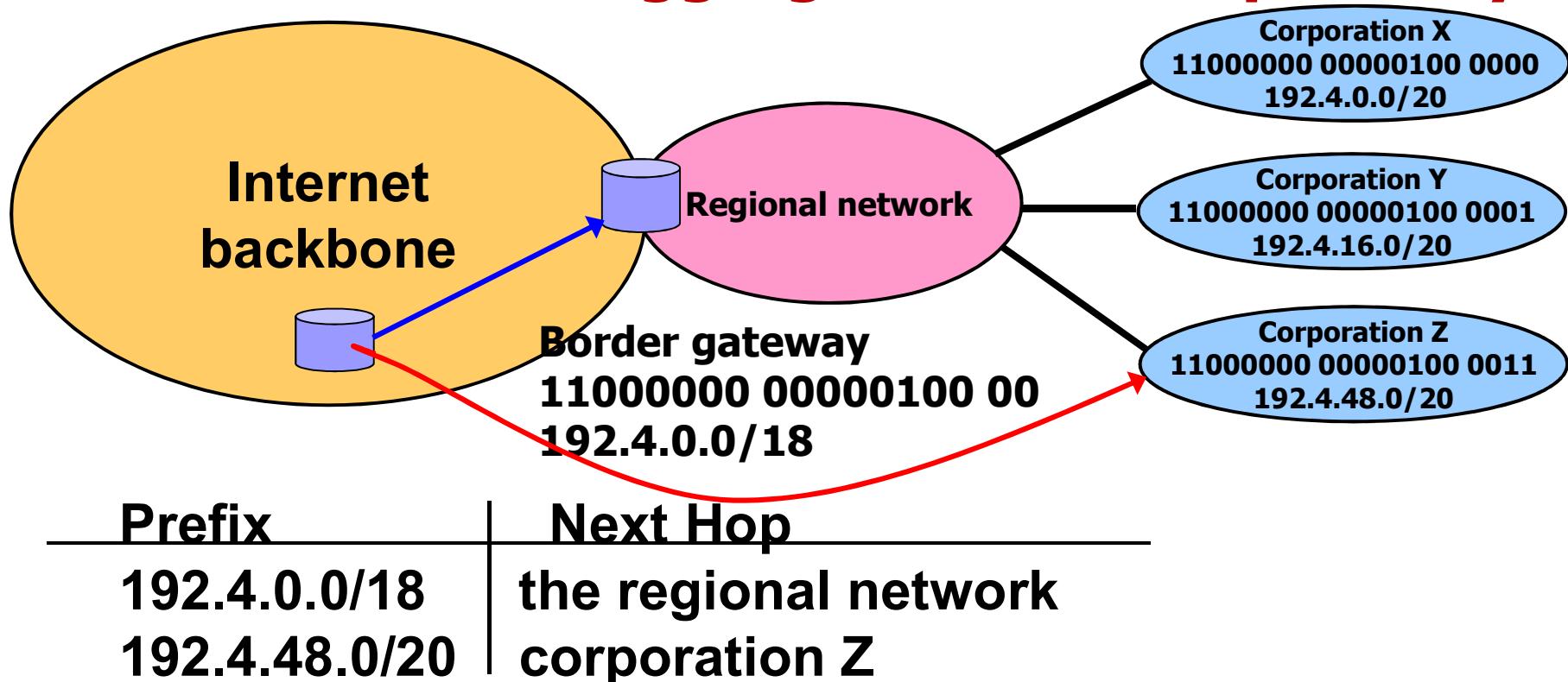
# Classless Addressing Examples

- X and Y routes can be aggregated because they form a bigger contiguous range.
- BUT can only aggregate powers of 2 😞



# Classless Addressing Examples

- CIDR allows to aggregate routes repeatedly



**Q:** what if there is a router capable of forwarding packets both to the regional network and to the corporation Z?

**S:** Use Principle of Longest Match

# Example

收到的分组的目的地址  $D = 206.0.71.130$

路由表中的项目：  
206.0.68.0/22      1  
206.0.71.128/25    2

查找路由表中的第 1 个项目

第 1 个项目 206.0.68.0/22 的掩码  $M$  有 22 个连续的 1。

$$M = 11111111 11111111 11111100 00000000$$

因此只需把  $D$  的第 3 个字节转换成二进制。

$$\begin{array}{rcl} M = & \boxed{11111111} & 11111111 11111100 00000000 \\ \text{AND } D = & \boxed{206.} & 0. \quad 01000111. \quad 130 \\ \hline & \boxed{206.} & 0. \quad 01000100. \quad 0 \end{array}$$

与 206.0.68.0/22 匹配

# Example

收到的分组的目的地址  $D = 206.0.71.130$

路由表中的项目：  
206.0.68.0/22 1  
206.0.71.128/25 2

再查找路由表中的第 2 个项目

第 2 个项目 206.0.71.128/25 的掩码  $M$  有 25 个连续的 1。

$$M = 11111111 11111111 11111111 10000000$$

因此只需把  $D$  的第 4 个字节转换成二进制。

$M =$	11111111	11111111	11111111	10000000
$AND$	206.	0.	71.	10000010
	206.	0.	71.	10000000

与 206.0.71.128/25 匹配

# Example

$D \text{ AND } (11111111\ 11111111\ 11111100\ 00000000)$

$= 206\ .\ 0\ .\ 68\ .\ 0 / 22$  匹配

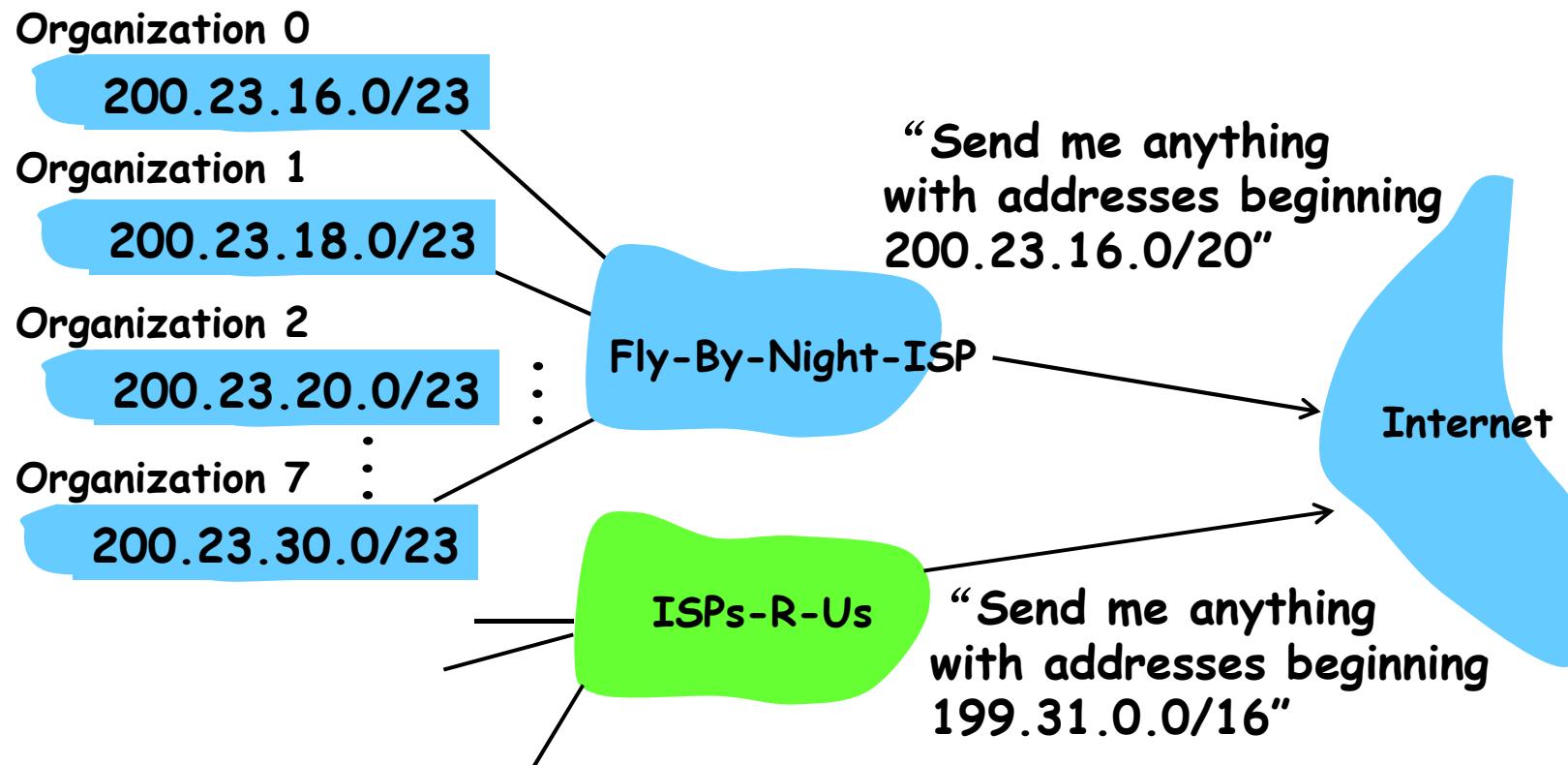
$D \text{ AND } (11111111\ 11111111\ 11111111\ 10000000)$

$= \underline{\underline{206\ .\ 0\ .\ 71\ .\ 128 / 25}}$  匹配

- 选择两个匹配的地址中更具体的一个，即选择最长前缀的地址

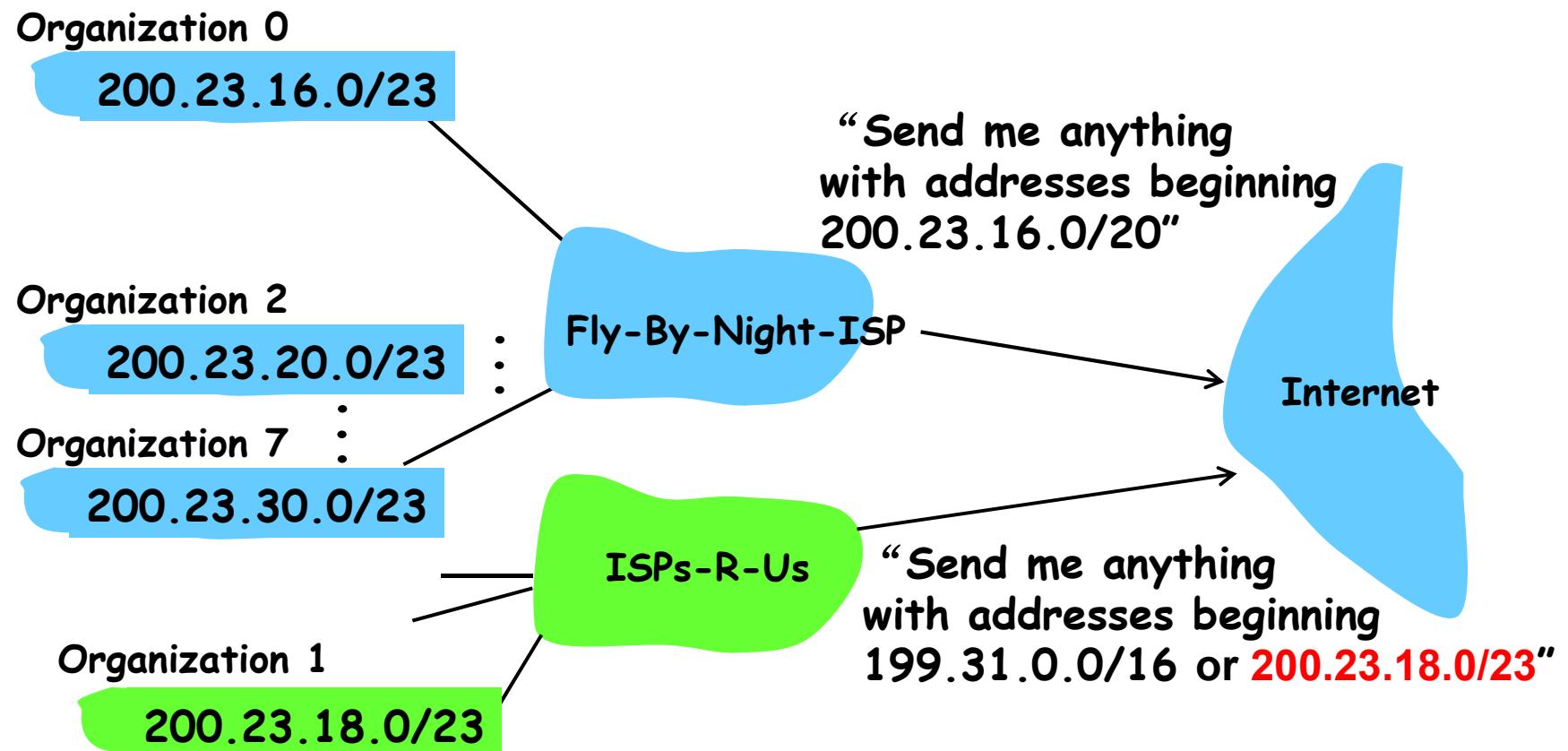
# Route aggregation: Hierarchical addressing

**Hierarchical addressing** allows efficient advertisement of routing information:



# Route aggregation: Hierarchical addressing

**ISPs-R-Us has a more specific route to Organization 1.**



# Advantage of Classless Addressing

- **Flexibility in allocating blocks of various sizes**
- **Assume:** an ISP has the following block of addresses: **128.211.0.0/16**
  - Can assign one customer 2048 addresses in the /21 range:

	Dotted Decimal	32-bit Binary Equivalent
Lowest	128.211.168.0	10000000 11010011 10101000 00000000
Highest	128.211.175.255	10000000 11010011 10101111 11111111

- Can assign another customer 4 addresses in the /29 range:

	Dotted Decimal	32-bit Binary Equivalent
Lowest	128.211.176.212	10000000 11010011 10110000 11010100
Highest	128.211.176.215	10000000 11010011 10110000 11010111

# Advantage of Classless Addressing

- **Allows a network administrator to assign addresses in contiguous blocks**
  - Number of addresses in a block must be a power 2
- **Allows for:**
  - Flexibility in assigning blocks of addresses
  - Ease of management of addresses
  - Minimizing the routing table

# NAT - Network Address Translation

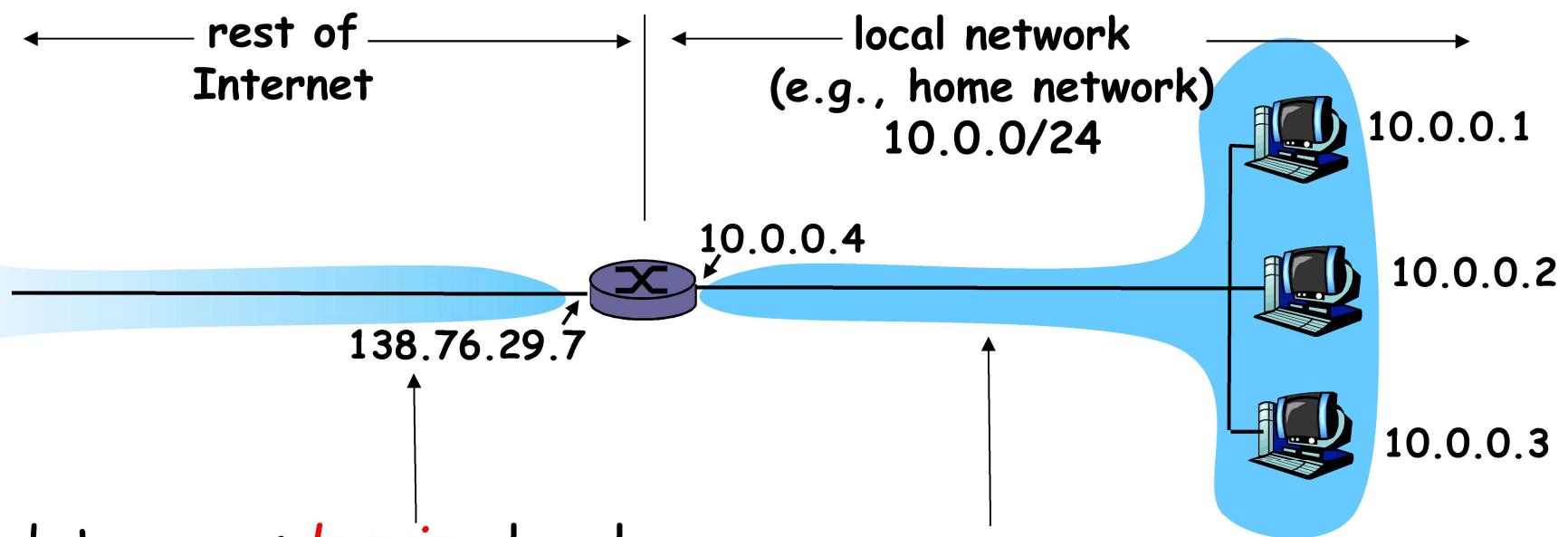
- NAT is a quick fix for the problem of running out of IP address
- NAT is to assign each company a single IP address for Internet traffic.
  - Within the company, every host gets a unique IP address, which is used for routing for intramural traffic.
  - When a packet exits the company and goes to the ISP, an address translation takes place.

# NAT - Network Address Translation

- To make this scheme possible, three ranges of IP addresses have been declared as private.
- The reserved ranges for local networks are:
  - 10.0.0.0 - 10.255.255.255/8 (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)

# NAT - Network Address Translation

- A local network uses just one public IP address as far as outside world is concerned
- Each device on the local network is assigned a private IP address



All datagrams *leaving* local network have *same* single source NAT IP address:  
138.76.29.7,  
different source port numbers

Datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual)

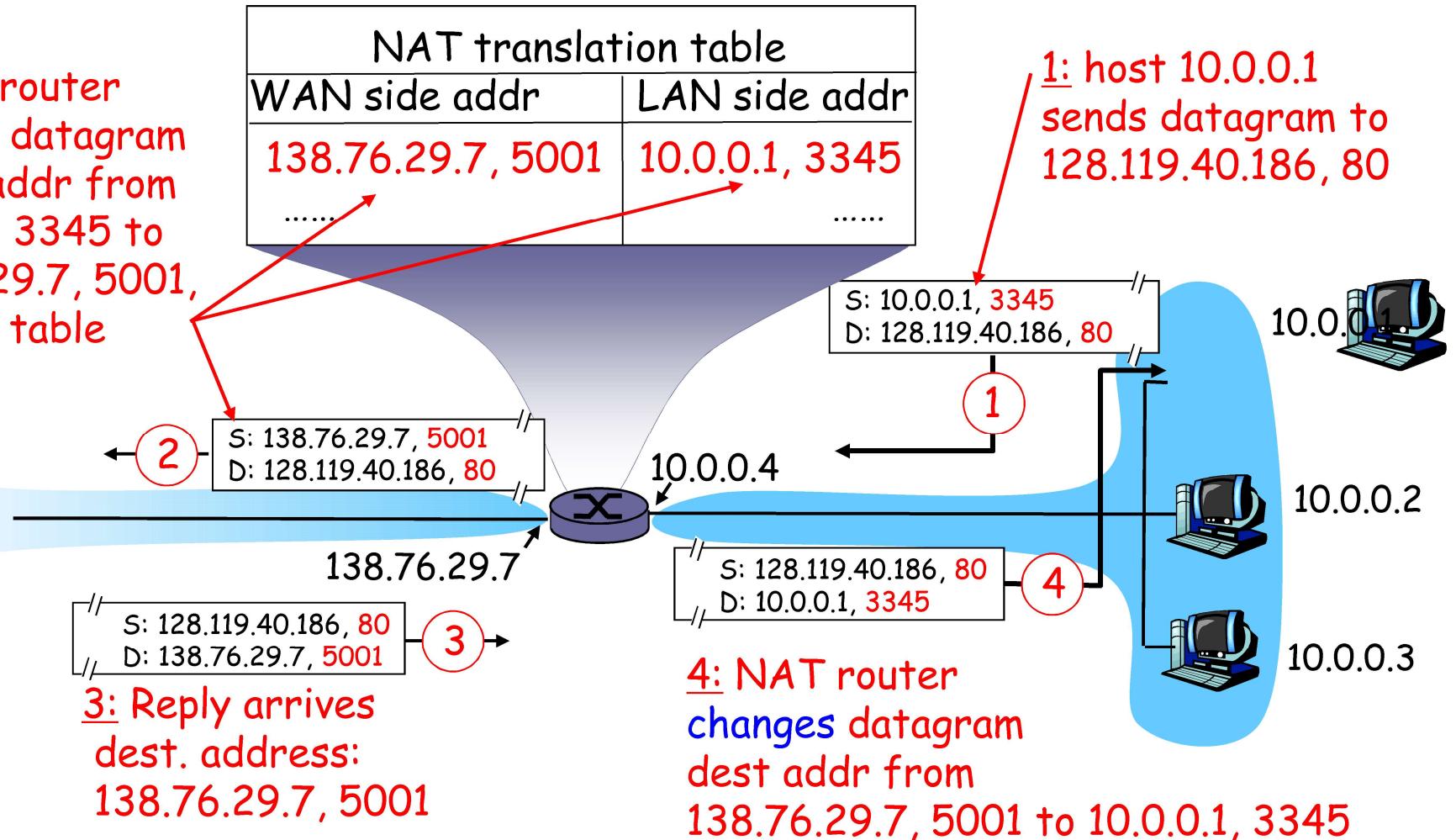
# NAT - Network Address Translation

**Implementation:** NAT router must:

- ***outgoing datagrams: replace*** (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)
  - remote clients/servers will respond using (NAT IP address, new port #) as destination addr.
- ***remember (in NAT translation table)*** every (source IP address, port #) to (NAT IP address, new port #) translation pair
- ***incoming datagrams: replace*** (NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

# NAT - Network Address Translation

2: NAT router changes datagram source addr from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table



# NAT - Advantage

- **No need** to be allocated range of addresses from ISP: - just one public IP address is used for all devices
  - 16-bit port-number field allows 60,000 simultaneous connections with a single LAN-side address !
  - can change ISP without changing addresses of devices in local network
  - can change addresses of devices in local network without notifying outside world
- Devices inside local net **not explicitly** addressable, visible by outside world (a security plus)

# NAT - Disadvantage

- If both hosts are behind NAT, they will have difficulty establishing connection
- NAT is controversial:
  - routers should process up to only layer 3
  - violates end-to-end argument
    - NAT possibility must be taken into account by app designers, e.g., P2P applications
  - address shortage should instead be solved by having more addresses --- IPv6 !

# IP Packet Routing and Forwarding

## ■ IP forwarding also known as Internet routing (or IP routing)

- A process used to determine which path a IP packet or datagram can be sent.
- The process uses **routing information** to make decisions and is designed to send a packet over multiple networks.

The **IP forwarding algorithm**, commonly known as **IP routing**, is a specific implementation of routing for IP networks.

# IP Packet Routing and Forwarding

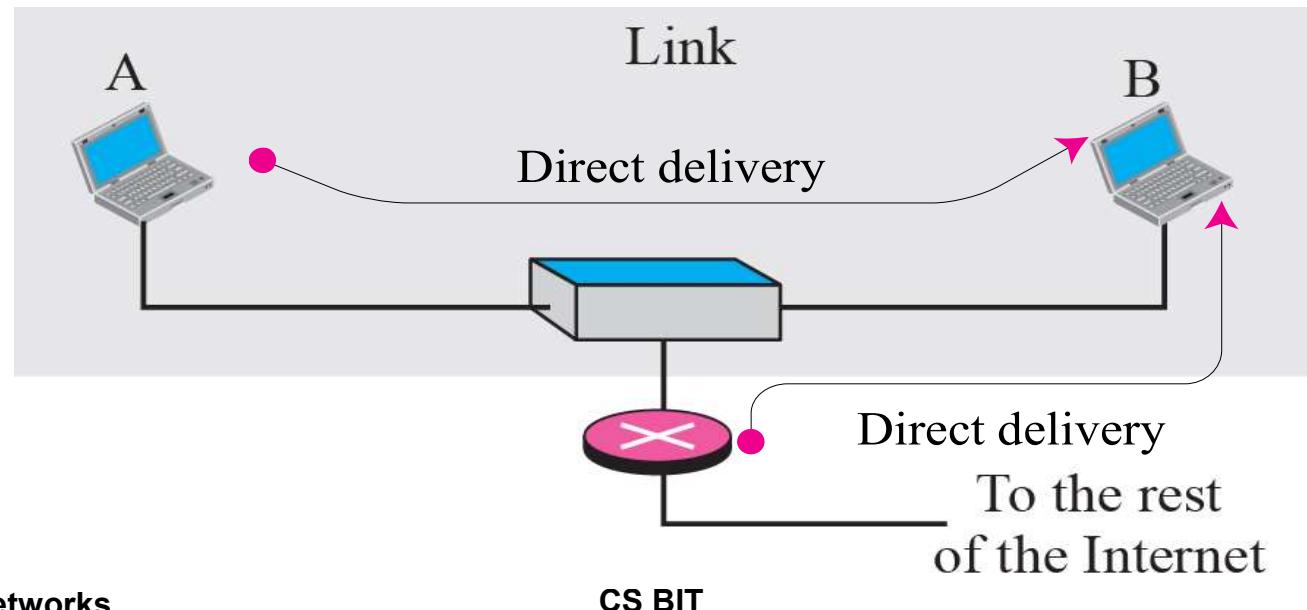
- There are two distinct processes to delivering IP datagrams:
  - 1. Forwarding: How to pass a packet from an input interface to the output interface?**
  - 2. Routing: How to find and setup the routing tables?**

# IP Packet Routing and Forwarding

- The delivery of a packet to its final destination is accomplished using two different methods of delivery:
  - Direct
  - Indirect.

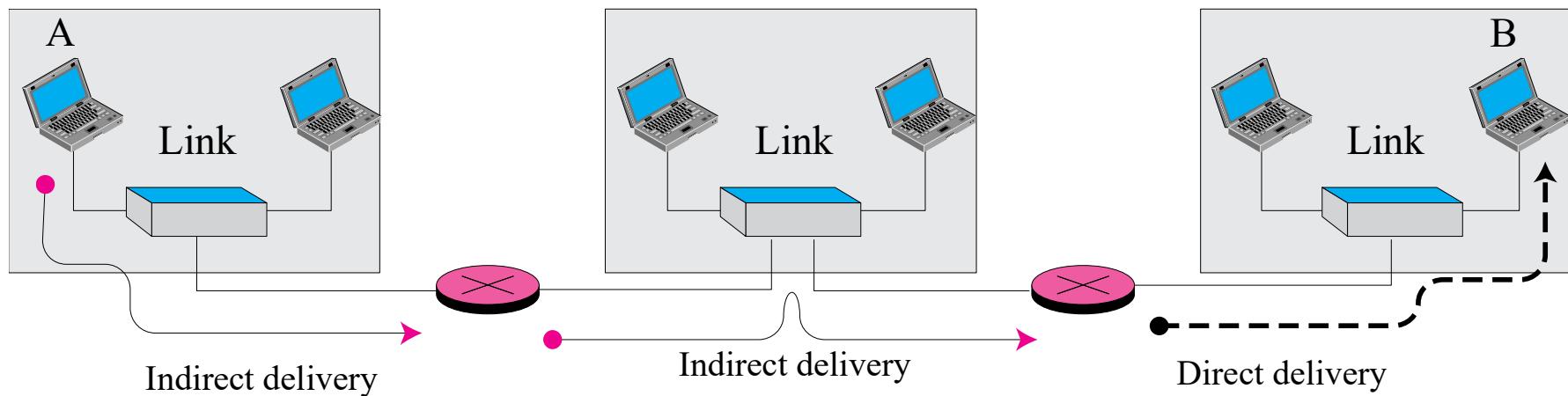
# Direct Delivery

- Occurs when the IP node (either the sending node or an IP router) forwards a packet to the final destination **on a directly attached network**.
- The IP node encapsulates the IP datagram in a frame format for the Network Interface addressed to the destination's physical address.

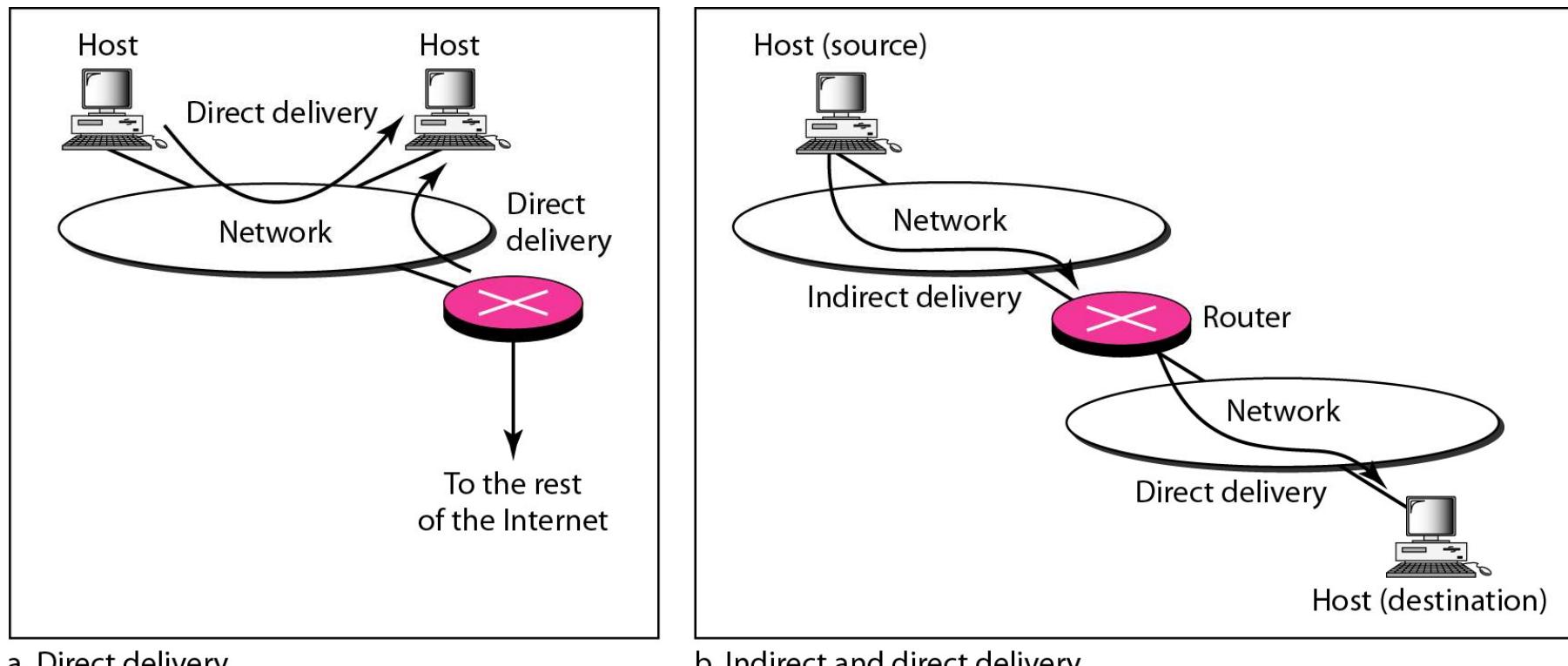


# Indirect Delivery

- Occurs when the IP node (either the sending node or an IP router) forwards a packet to an intermediate node (an IP router) because the final destination is **not on a directly attached network**.
- The IP node encapsulates the IP datagram in a frame format, addressed to the IP router's physical address, for the Network Interface.



# IP Packet Routing and Forwarding



a. Direct delivery

b. Indirect and direct delivery

- **直接交付:** 源和目的主机连接在同一个子网中，则将IP分组封装到数据链路层的帧中，利用数据链路层功能直接发送给目的主机。
- **间接交付:** 源和目的主机不在同一个子网中，则将IP分组发送下一跳路由器，有路由器转发给目的主机。

# IP Packet Forwarding

- **Forwarding means to place the packet in its route to its destination.**
  - **Forwarding Based on Destination Address.**
  - **Forwarding Based on Label.**

**Do you remember?**

The IP forwarding algorithm, commonly known as IP routing,

# Routing(1/2)

- a) Routing requires a host or a router to have a **routing table**.**
- b) Usually when a host has a packet to send or when a router has received a packet to be forwarded, it looks at this table to find the route to the final destination.**
- c) However, this simple solution is impossible in today's Internet world because the number of entries in the routing table makes the table lookups inefficient.**

# Routing(2/2)

- a) Need to make the size of table manageable and handles issues such security at the same time.**  
**The key question is how to design the routing table.**
- b) Routing:**
  - ① Next-hop routing**
  - ② Network-specific routing**
  - ③ Host specific routing**

# Next-hop routing

Routing table for host A

Destination	Route
Host B	R1, R2, Host B

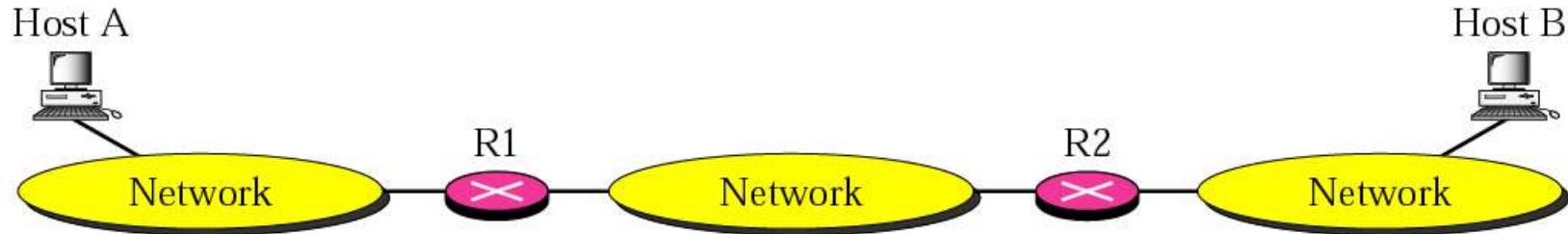
Routing table for R1

Destination	Route
Host B	R2, Host B

Routing table for R2

Destination	Route
Host B	Host B

a. Routing tables based on route



Routing table for host A

Destination	Next Hop
Host B	R1

Routing table for R1

Destination	Next Hop
Host B	R2

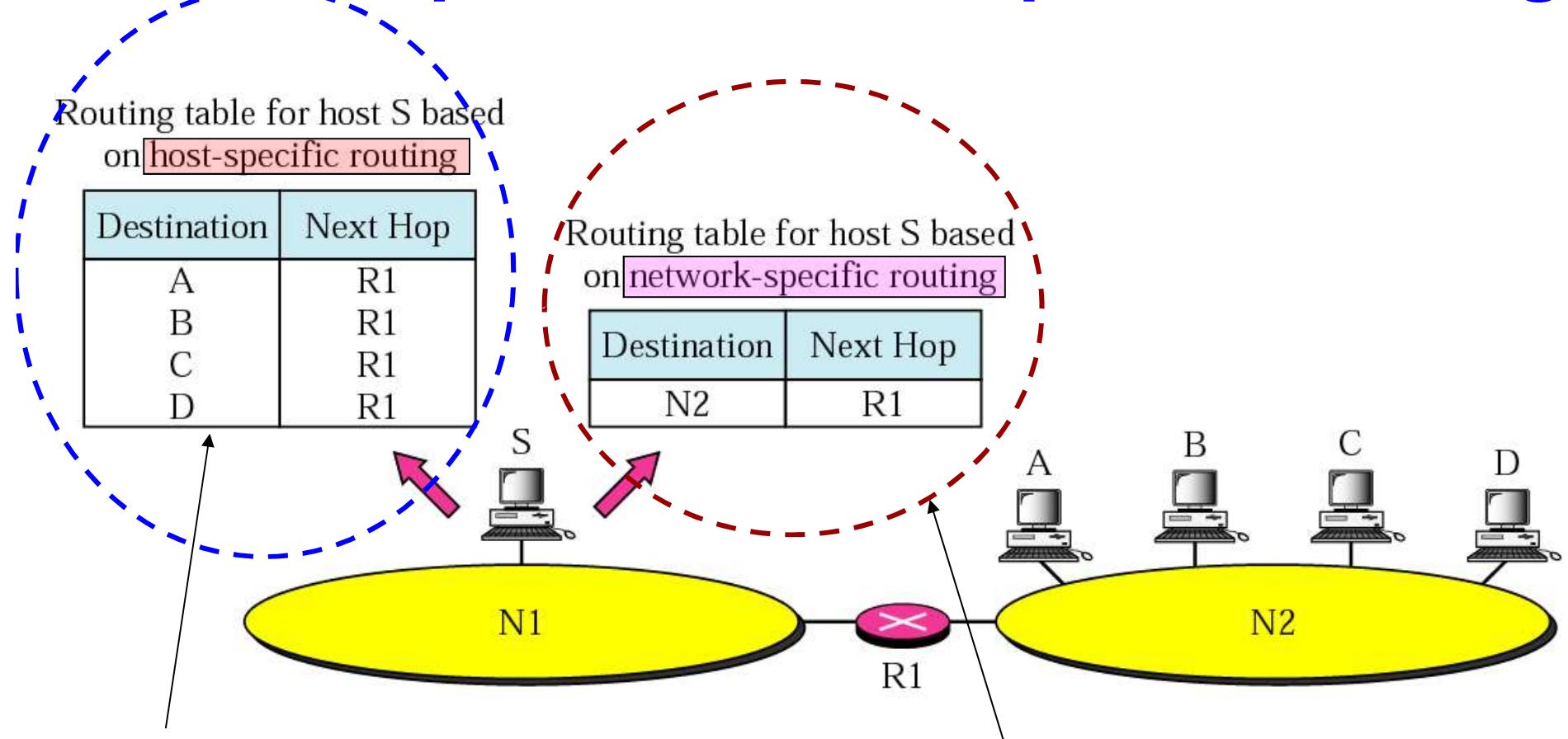
Routing table for R2

Destination	Next Hop
Host B	—

b. Routing tables based on next hop

**Next-hop routing holds only the information that leads to the next hop instead of complete route.**

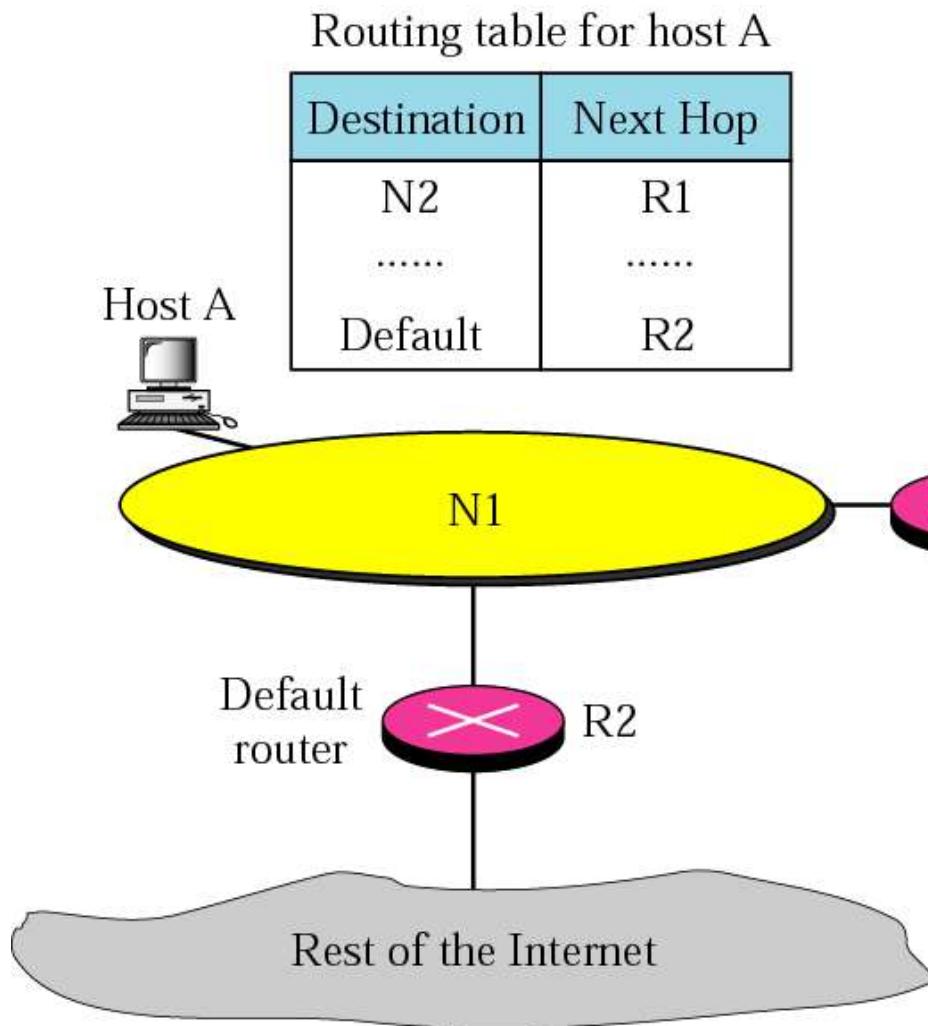
# Network-specific & host-specific routing



The destination host address is given in the routing table; to have greater control over routing.

Instead of having an entry for every host connected to the same network, only one entry is needed to define the address of the network itself. All hosts connected to the same network as one single entity.

# Default routing



**R1 is used to route packets to hosts connected to N2.**



**However, R2 is used to as default to route other packets to the rest of Internet without listing all the networks involved**

**Only one default routing is allowed with network address 0.0.0.0/0**

# General Routing Table

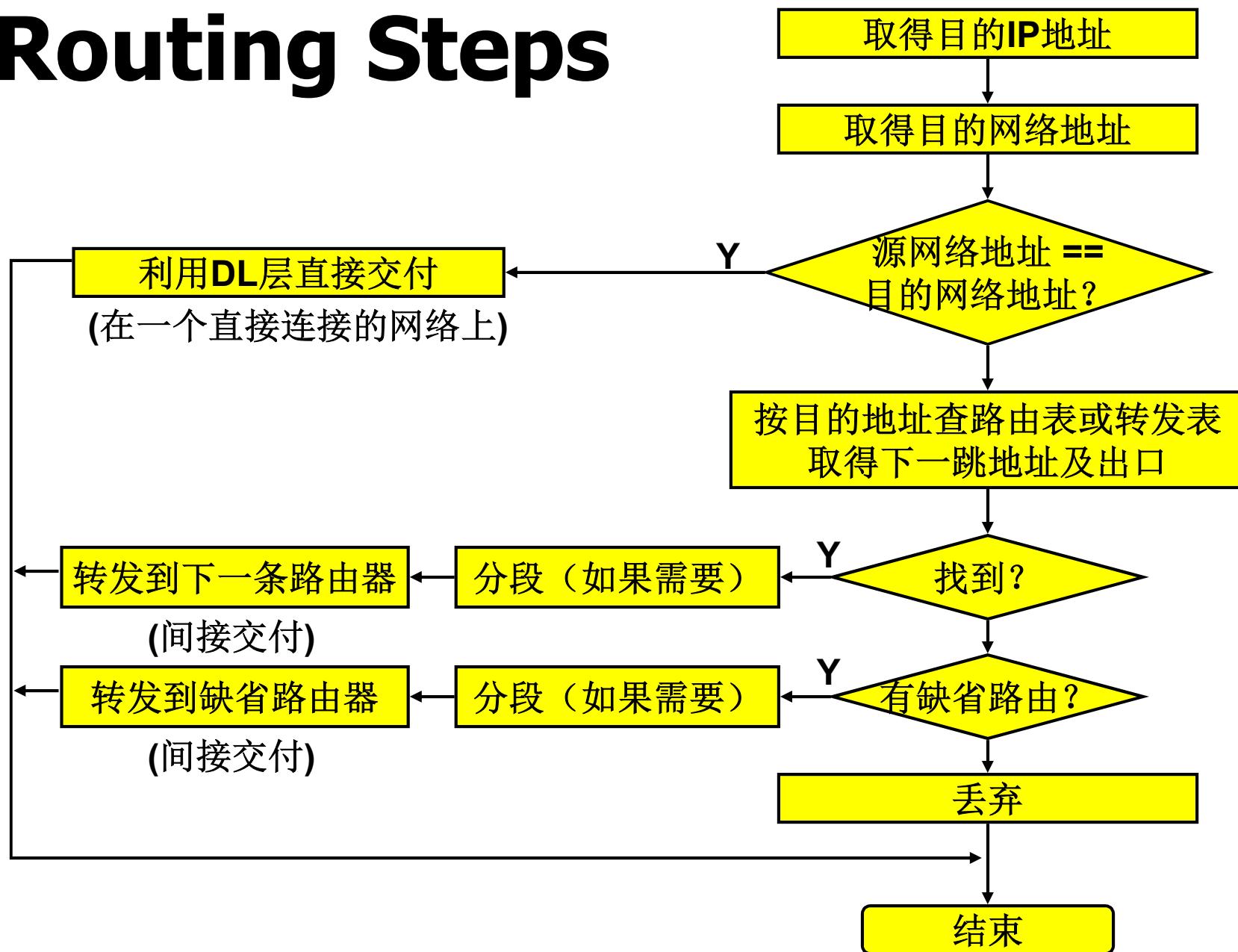
Mask	Destination address	Next-hop address	Flags	Reference count	Use	Interface
255.0.0.0	124.0.0.0	145.6.7.23	UG	4	20	m2
.....	.....	.....	...	...	...	....
.....	.....	.....	...	...	...	....

## Flags

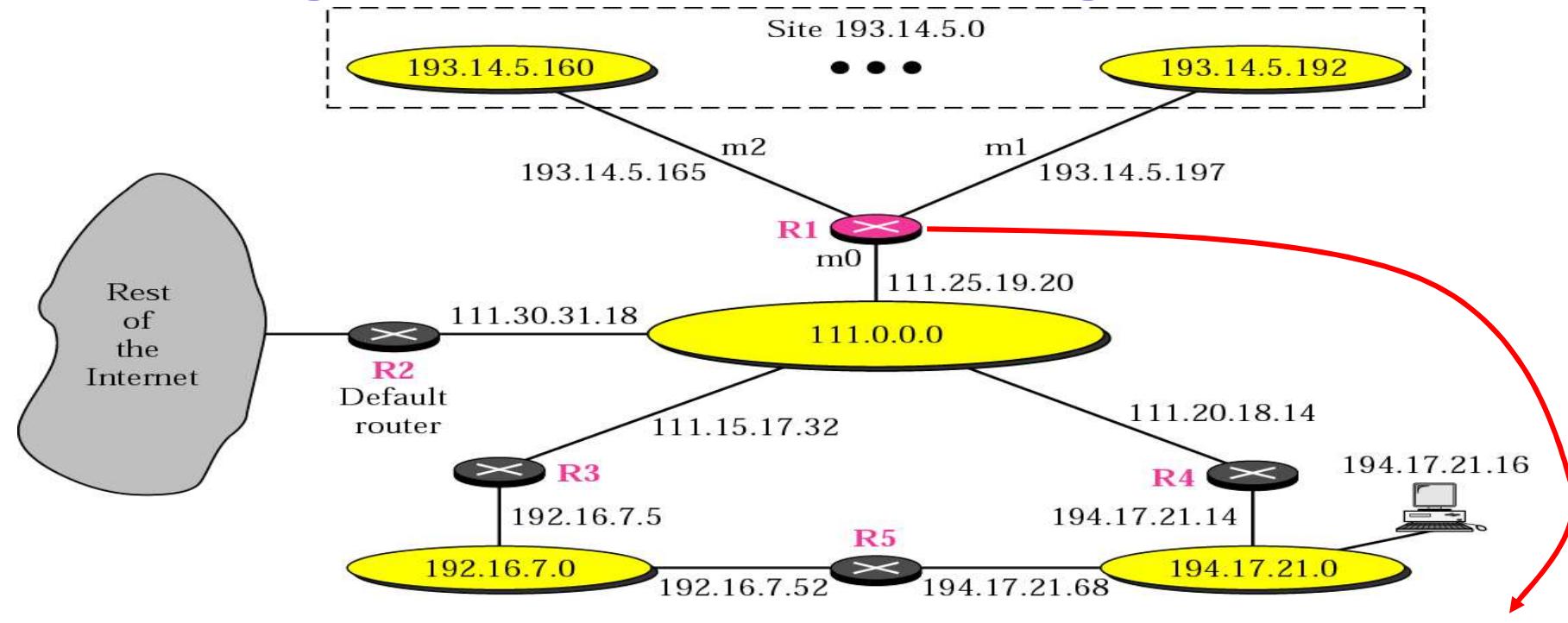
- U** The router is up and running.
- G** The destination is in another network.
- H** Host-specific address.
- D** Added by redirection.
- M** Modified by redirection.

Generally, a routing table needs **a minimum of 4 columns:** mask, destination network address, next hop address and interface.

# Routing Steps



# Configuration for routing example



**Standard delivery**

**Host-specific**

**Network-specific**

**Default**

	<u>Mask</u>	<u>Dest</u>	<u>Next Hop</u>	<u>Intf.</u>
	255.0.0.0	111.0.0.0	--	m0
<b>Standard delivery</b>	255.255.255.224	193.14.5.160	--	m2
	255.255.255.224	193.14.5.192	--	m1
<b>Host-specific</b>	255.255.255.255	194.17.21.16	111.20.18.14	m0
	255.255.255.0	192.16.7.0	111.15.17.32	m0
<b>Network-specific</b>	255.255.255.0	194.17.21.0	111.20.18.14	m0
<b>Default</b>	0.0.0.0	0.0.0.0	111.30.31.18	m0

## Example 1

Router R1 receives 500 packets for destination **192.16.7.14**; the algorithm applies the masks row by row to the destination address until a match is found.

### Solution

#### Direct delivery

→ 192.16.7.14 & **255.0.0.0** → 192.0.0.0 no match to 111.0.0.0  
192.16.7.14 & **255.255.255.224** → 192.16.7.0 no match to 193.14.5.160  
192.16.7.14 & **255.255.255.224** → 192.16.7.0 no match to 193.14.5.192

#### Host-specific

192.16.7.14 & **255.255.255.255** → 192.16.7.14 no match to 194.17.21.16

#### Network-specific

192.16.7.14 & **255.255.255.0** → 192.16.7.0 **match to 192.16.7.0**

## Example 2

Router R1 receives 100 packets for destination **193.14.5.176**; the algorithm applies the masks row by row to the destination address until a match is found.

## Solution

### Direct delivery

**193.14.5.176 & 255.0.0.0** → 193.0.0.0 no match

**193.14.5.176 & 255.255.255.224** → 193.14.5.160 match

## Example 3

Router R1 receives 20 packets for destination **200.34.12.34**; the algorithm applies the masks row by row to the destination address until a match is found.

### Solution

200.34.12.34 & **255.0.0.0** → 200.0.0.0 no match

200.34.12.34 & **255.255.255.224** → 200.34.12.32 no match

200.34.12.34 & **255.255.255.224** → 200.34.12.32 no match

---

200.34.12.34 & **255.255.255.255** → 200.34.12.34 no match

---

200.34.12.34 & **255.255.255.0** → 200.34.12.0 no match

---

200.34.12.34 & **255.255.255.0** → 200.34.12.0 no match

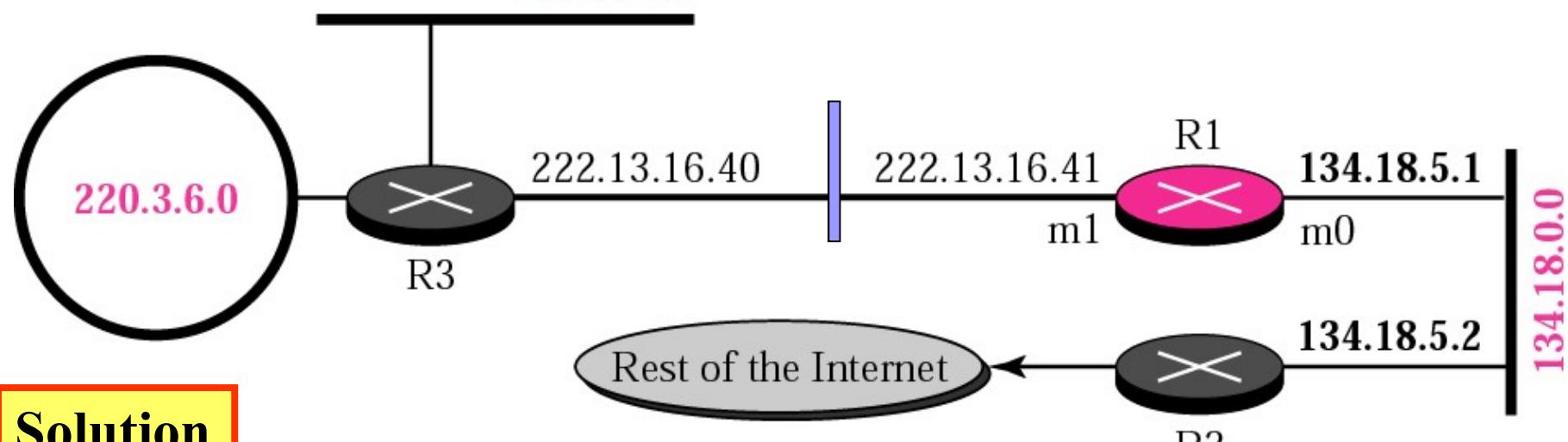
---

**Default**

200.34.12.34 & **0.0.0.0** → 0.0.0.0 **match**

## Example 4

Make the routing table for router R1 in figure below 129.8.0.0

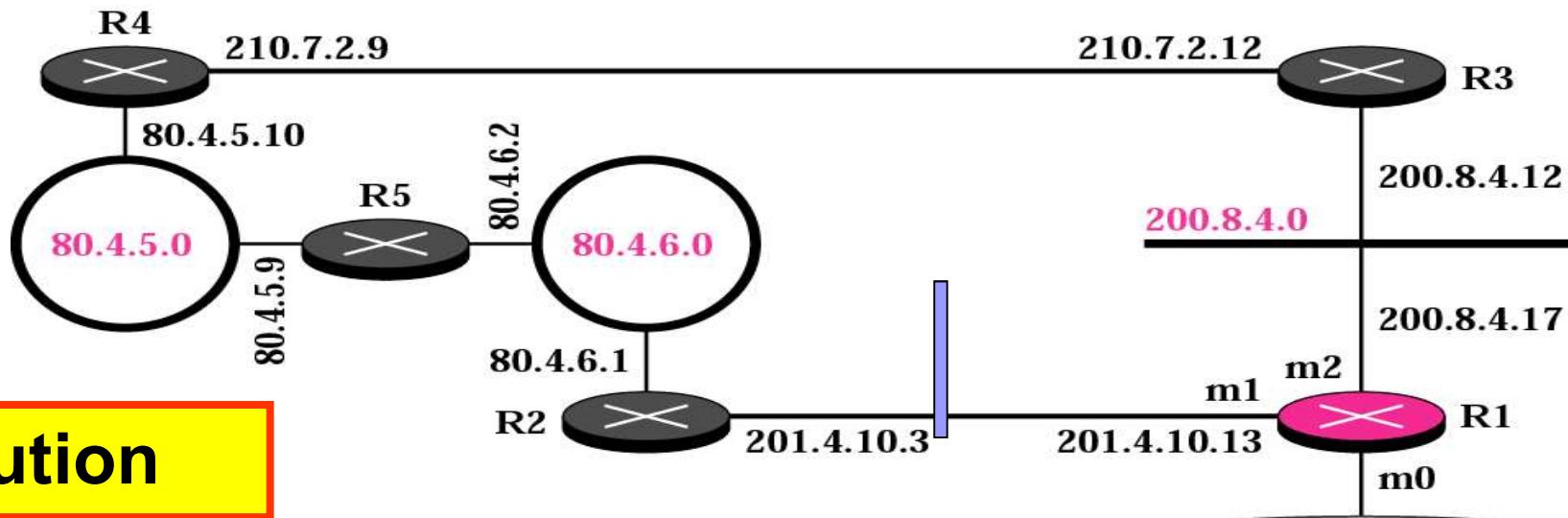


### Solution

Mask	Destination	Next Hop	Intf.
255.255.0.0	134.18.0.0	--	m0
255.255.0.0	129.8.0.0	222.13.16.40	m1
255.255.255.0	220.3.6.0	222.13.16.40	m1
0.0.0.0	0.0.0.0	134.18.5.2	m0

## Example 5

Make the routing table for router R1 in figure below



## Solution

Subnet mask	Destination	Next Hop	Intf.
255.255.255.0	200.8.4.0	--	m2
255.255.255.0	80.4.5.0	201.4.10.3	m1
		or 200.8.4.12	or m2
255.255.255.0	80.4.6.0	201.4.10.3	m1
		or 200.8.4.17	or m2
0.0.0.0	0.0.0.0		m0

# Routing Tables in IP with CIDR

Mask	Destination	Next Hop
/12	128.96.0.0	145.12.56.29
/17	128.125.0.0	153.202.12.128
/12	128.112.0.0	153.202.14.1
/26	128.105.14.64	153.2.45.101
/32	128.105.14.66	153.2.45.101

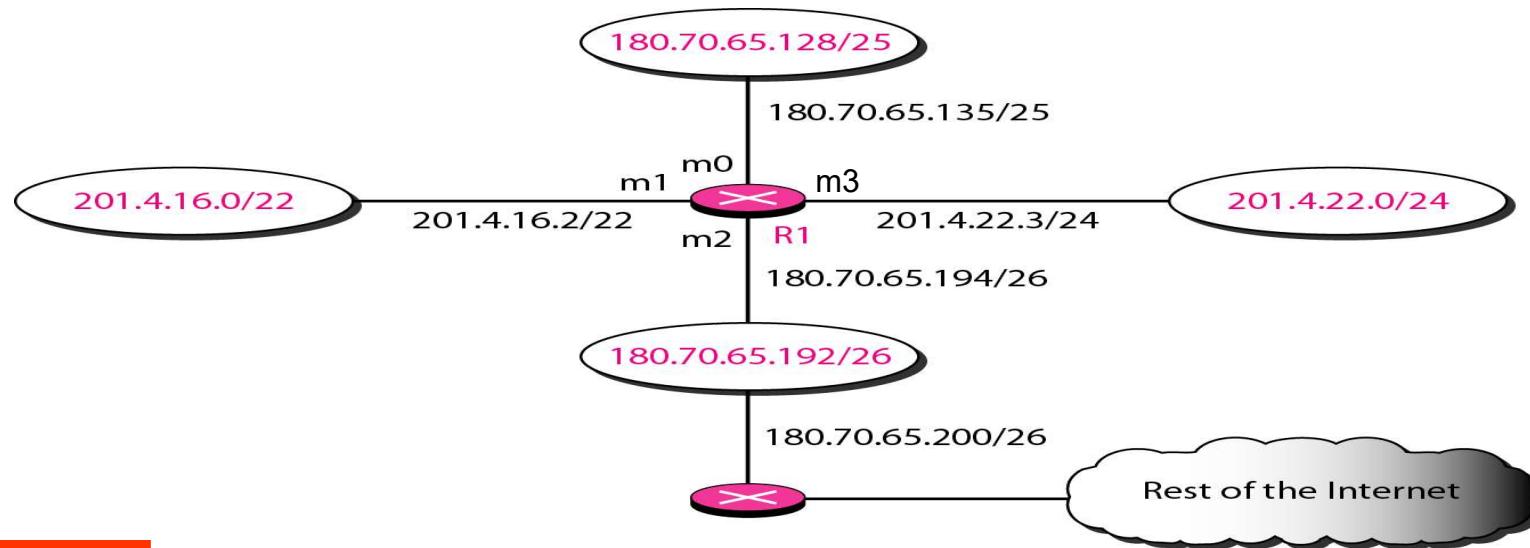
**For each entry in the routing table:**

```
MaskedAddress := EntryMask (bitAND) IPDatagramDestinationAddress;  
if (MaskedAddress == EntryDestination)  
    Mark the entry;
```

**Choose the marked entry with the longest Mask prefix.**

## Example 6a

Make a routing table for router R1, using the configuration in Figure below



### Solution

*Routing table for router R1 in Figure above*

Mask	Network Address	Next Hop	Interface
/26	180.70.65.192	—	m2
/25	180.70.65.128	—	m0
/24	201.4.22.0	—	m3
/22	201.4.16.0	....	m1
Any	Any	180.70.65.200	m2

The table is sorted from the longest mask to the shortest mask.

## Example 6b

Show the forwarding process if a packet arrives at R1 with the destination address **180.70.65.140**.



Mask	Network Address	Next Hop	Interface
/26	180.70.65.192	—	m2
/25	180.70.65.128	—	m0
/24	201.4.22.0	—	m3
/22	201.4.16.0	....	m1
Any	Any	180.70.65.200	m2

### Solution

The router performs the following steps:

1. The first mask (/26) is applied to the destination address. The result is 180.70.65.128, which does **not match** the corresponding network address.
2. The second mask (/25) is applied to the destination address. The result is 180.70.65.128, which **matches** the corresponding network address. The next-hop address and the interface number m0 are passed on for further processing.

## Example 6c

Show the forwarding process if a packet arrives at R1 with the destination address **201.4.22.35**.



### Solution

<i>Mask</i>	<i>Network Address</i>	<i>Next Hop</i>	<i>Interface</i>
/26	180.70.65.192	—	m2
/25	180.70.65.128	—	m0
/24	201.4.22.0	—	m3
/22	201.4.16.0	....	m1
Any	Any	180.70.65.200	m2

The router performs the following steps:

1. The first mask (/26) is applied to the destination address. The result is 201.4.22.0, which does **not match** the corresponding network address.
2. The second mask (/25) is applied to the destination address. The result is 201.4.22.0, which does **not match** the corresponding network address (row 2).
3. The third mask (/24) is applied to the destination address. The result is 201.4.22.0, which **matches** the corresponding network address..

## Example 6d

Show the forwarding process if a packet arrives at R1 with the destination address **18.24.32.78**.

<i>Mask</i>	<i>Network Address</i>	<i>Next Hop</i>	<i>Interface</i>
/26	180.70.65.192	—	m2
/25	180.70.65.128	—	m0
/24	201.4.22.0	—	m3
/22	201.4.16.0	....	m1
Any	Any	180.70.65.200	m2

### Solution

This time all masks are applied, one by one, to the destination address, but **no matching** network address is found.

When it reaches the end of the table, the module gives the **default next-hop address 180.70.65.200** (because it could not find the match) . This is probably an outgoing package that needs to be sent, via the default router, to someplace else in the Internet.

## Example 7

Assume that an ISP owns the address block **206.0.64.0/18**, which represents 16,384 ( $2^{14}$ ) IP addresses. Suppose a client requires **800** host addresses. Give 2 possible IP address arrangements.

**With classful addresses:** need to assign 1 class B address (and waste ~64,700 addresses) or 4 individual Class Cs (and introducing 4 new routes into the global Internet routing tables)

**With CIDR:** Assign a /22 block, e.g., **206.0.68.0/22**, and allocated a block of 1,024 ( $2^{10}$ ) IP addresses.

# Internet Message Control Protocols

- ICMP (RFC 792)
- Used to communicate IP **status** and **error** messages between host and routers

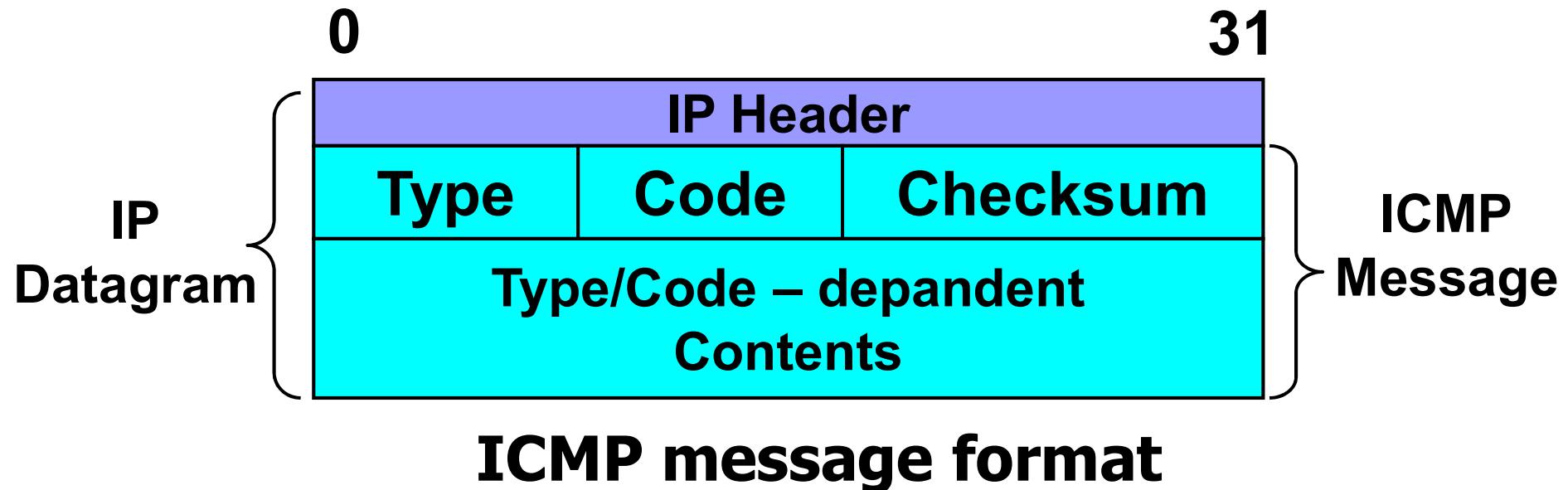
# Internet Message Control Protocols

- ICMP error messages **never** generates due to:
  - ICMP error message selves
  - Broadcast, multicast
  - Others fragments, except first fragment

# Internet Message Control Protocols

- **Uses IP to route its messages between hosts**
- **Must be implemented with IP**
- **ICMP data is carried as the payload of an IP datagram**
  - **specifies additional message formats within this area**

# Internet Message Control Protocols



- **Type:** 8 bit message type code
- **Code:** 8 bit; indicating why message is being sent
- **Checksum:** standard internet checksum
  - 16 bit 1's complement sum of the payload and header
- **Contents:** ICMP data, payload

# Internet Message Control Protocols

## The principal ICMP message types.

- 0 - Echo Reply
- 3 - Destination Unreachable
- 4 - Source Quench
- 5 - Redirect
- 8 - Echo
- 11 - Time Exceeded
- 12 - Parameter Problem
- 13 - Timestamp
- 14 - Timestamp Reply
- 15 - Information Request
- 16 - Information Reply
- 17 - Address Mask Request
- 18 - Address Mask Reply

# Destination Unreachable (3)

- Codes:

- **0** - net unreachable ; **1** - host unreachable
- **2** - protocol unreachable ; **3** - port unreachable
  - sent by destination host IP module
- **4** - fragmentation needed DF set ; **5** source route failed
- **6** - destination network unknown ; **7** destination host unknown
- **8** - source host isolated ; **9** - comm. with destn network prohibited
- **10** - comm. With dest host prohibited ; **11** - network unreachable for service
- **12** - host unreachable for service

# Destination Unreachable (3)

- Sent to **originating host** because destination is unreachable
  - may be determined by a router
  - destination IP may find the indicated protocol unavailable
  - Don't Fragment (DF) bit in the IP header is set but fragmentation is required to continue forwarding

# Source Quench (4)

- **Code : 0**
- **Sent to a host when an intermediate router or the destination host with the source host's transmission rate**
  - **may be sent to a source when a router is saturated**
  - **may be sent by a receiving host if its receive buffers are filling up**
- **Upon receipt the source host should throttle back on its transmission rate until the Source Quench goes away.**
  - **Can then increase its transmission rate**

# Redirect (5)

- **Code:**
  - 0 - redirect datagrams for the network
  - 1 - redirect datagrams for the host
  - 2 - redirect datagrams for the type of service and the network
  - 3 - redirect datagrams for the type of service and host
- **a router sends a message to a host when it determines a datagram that originated from the host must be forwarded to router that can be directly reached**
  - allows the host to sent future datagrams to the optimal first-hop router increasing network efficiency
- **not used for datagrams that have source routing options**

# Echo (8)/Echo Reply (0)

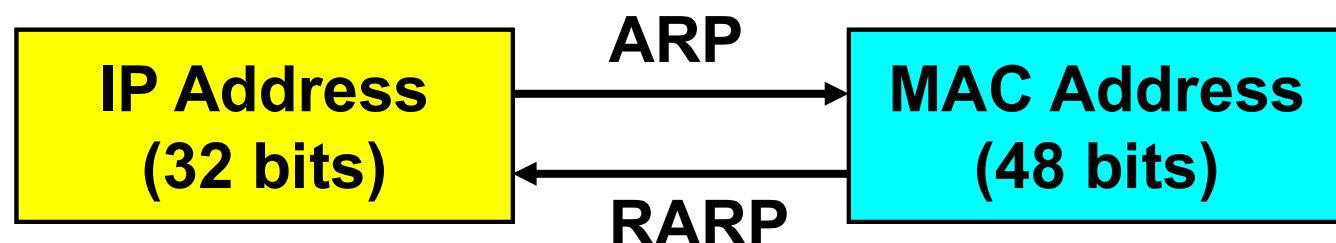
- **ICMP header (4 bytes) + identifier (2 bytes) + sequence number (2 bytes) + data (4 bytes)**
  - identifier - used to match Echoes and Echo Replies
  - sequence - used to match Echoes with Echo Replies
- **Used to determine if a host is reachable**
  - **a host receiving an echo message**
    - reverses the IP source and destination addresses
    - sets the ICMP type field to zero (echo reply)
    - recomputes the ICMP checksum
    - identifier, sequence and data are sent back unchanged

# Time Exceeded (11)

- Same format as type 3
- Code:
  - 0 - time to live exceeded in transit
  - 1 - fragment reassembly time exceeded
- Time exceeded message is sent if:
  - a router finds a datagram with TTL set to zero
    - router discards the datagram and sends message with code field set to 0
  - a host does not receive all of the fragments of a datagram before its local reassemble timer expires
    - host discards all fragments and return message with code field set to 1

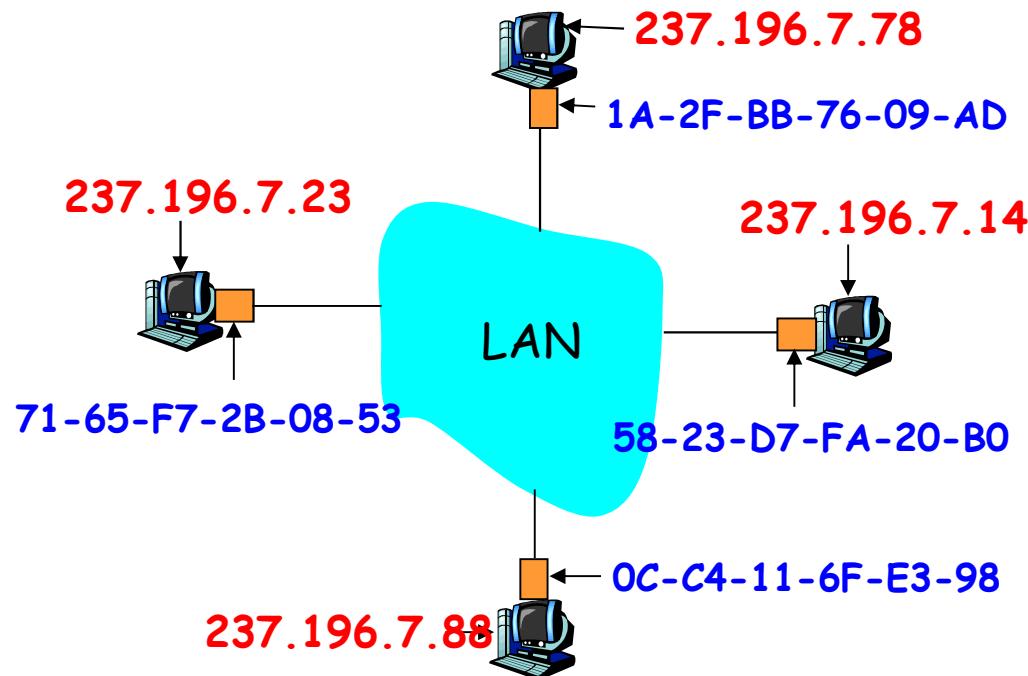
# ARP: Address Resolution Protocol

- The Internet is based on IP addresses
  - not recognized by the hardware of hosts
- Data link protocols (Ethernet, FDDI, ATM) may have different (MAC) addresses
- The ARP (and RARP) perform the translation between IP addresses and MAC layer addresses



# ARP: Address Resolution Protocol

**Question:** how to determine MAC address of B knowing B's IP address?



- Each IP node (Host, Router) on LAN has ARP table
- **ARP Table:** IP/MAC address mappings for some LAN nodes  
    < IP address; MAC address; TTL >
- **TTL (Time To Live):** time after which address mapping will be forgotten (typically 20 min)

# ARP: Address Resolution Protocol

- ARP Table (ARP Cache)
- Command: arp -a

IP Address	Hardware Address
197.15.3.2	0A:07:4B:12:82:36
197.15.3.3	0A:9C:28:71:32:8D
197.15.3.4	0A:11:C3:68:01:99
197.15.3.5	0A:74:59:32:CC:1F
197.15.3.6	0A:04:BC:00:03:28
197.15.3.7	0A:77:81:0E:52:FA

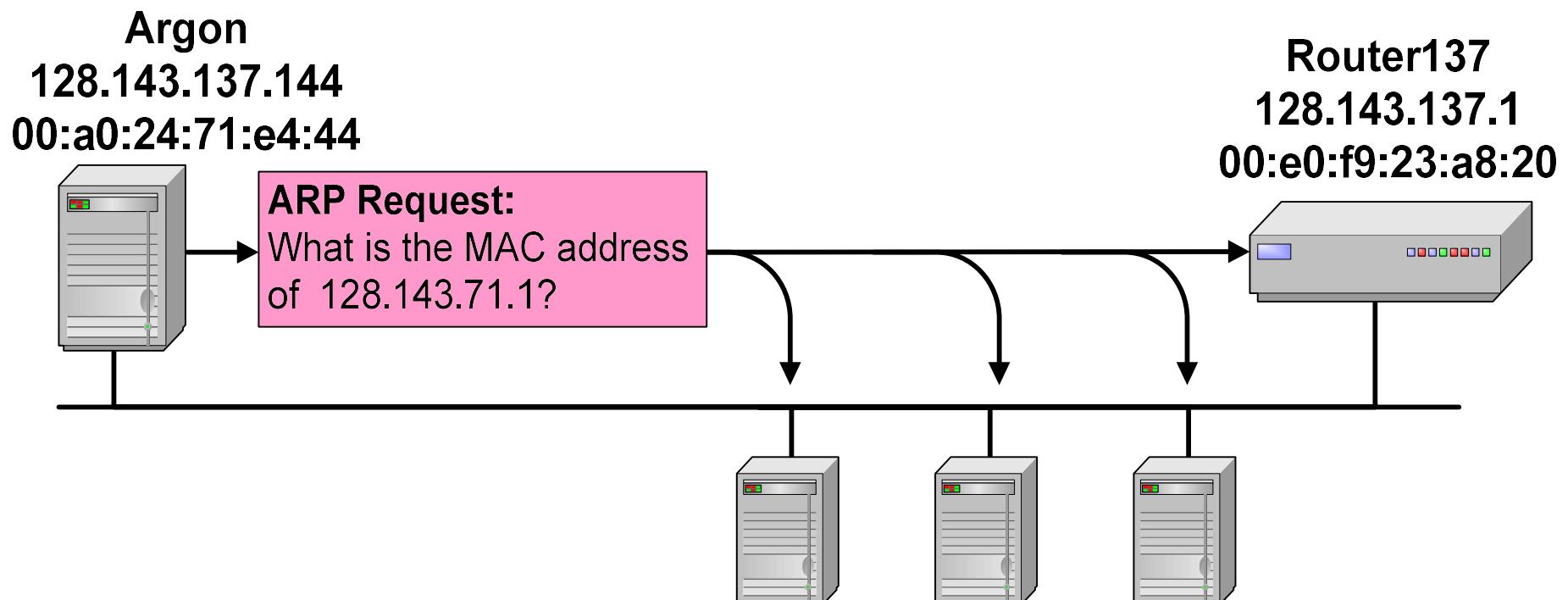
# ARP: Address Resolution Protocol

- ARP provides a **dynamic mapping** from an IP address to the corresponding hardware address.
- **Dynamic mapping:**
  - since it happens automatically and
  - is normally not a concern of either the application user or the system administrator.

# Address Translation with ARP

## ARP Request:

Argon **broadcasts** an ARP request to all stations on the network: “**What is the hardware address of 128.143.137.1 ?**”



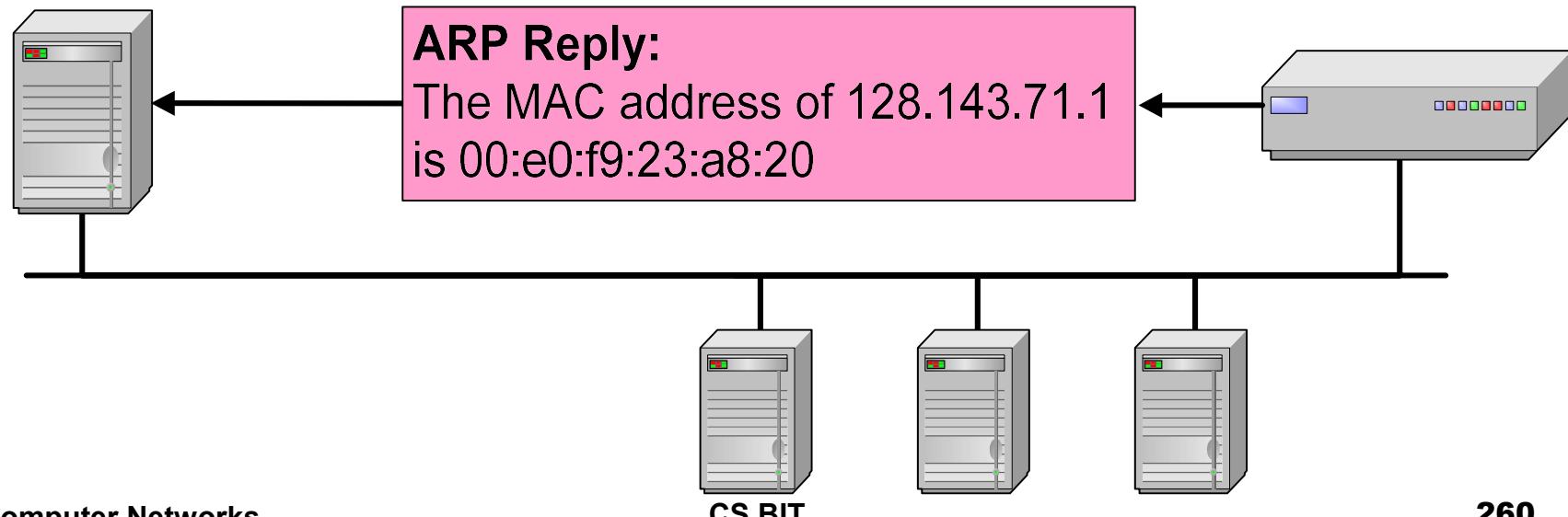
# Address Translation with ARP

## ARP Reply:

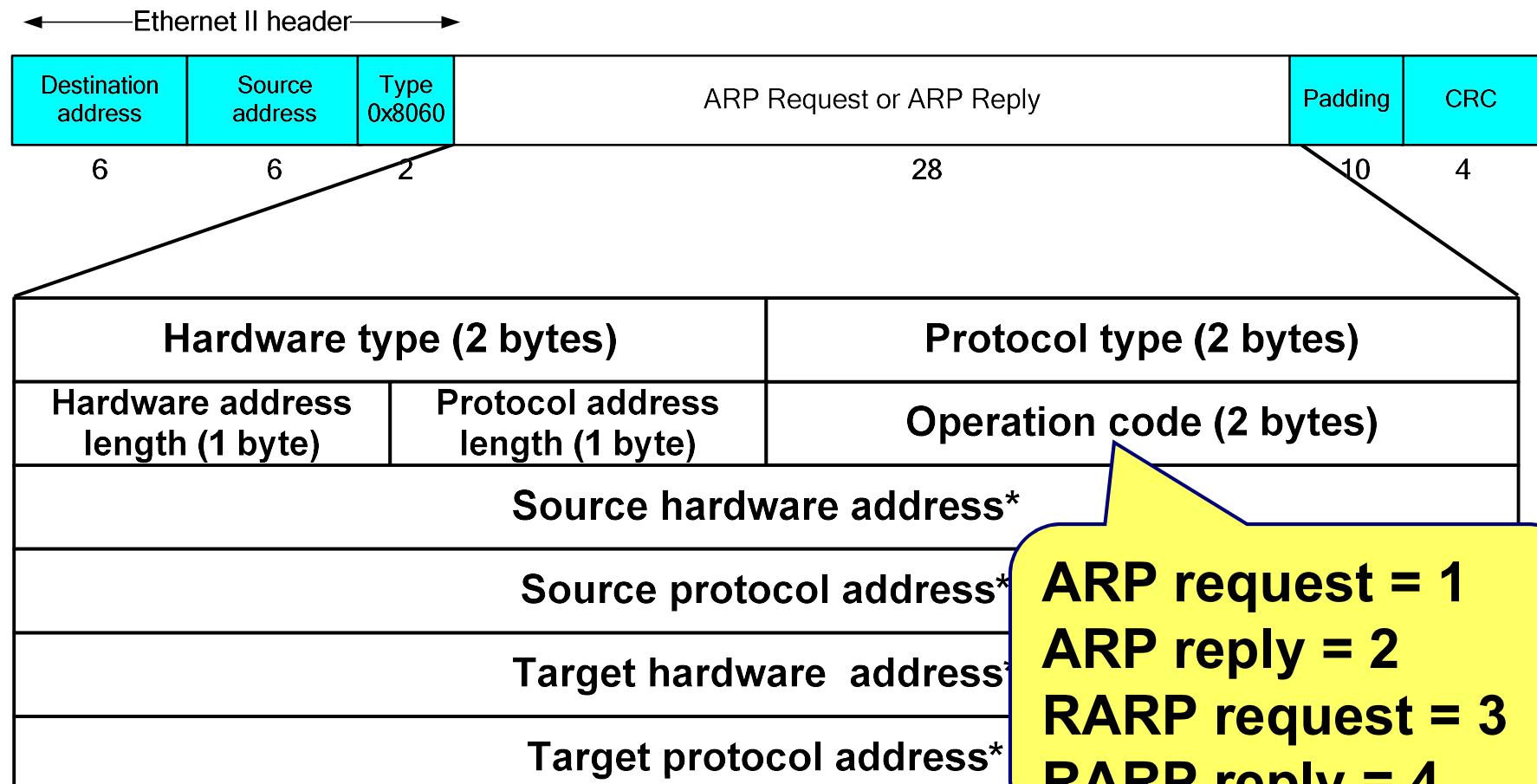
**Router 137 (128.143.137.1) responds with an ARP Reply (**unicast**) which contains the hardware address**

Argon  
128.143.137.144  
00:a0:24:71:e4:44

Router137  
128.143.137.1  
00:e0:f9:23:a8:20



# ARP Packet Format



\* Note: The length of the address fields is determined by the corresponding address length fields

# Example 1

## ■ ARP Request from Argon:

**Source hardware address:** 00:a0:24:71:e4:44

**Source protocol address:** 128.143.137.144

**Target hardware address:** 00:00:00:00:00:00

**Target protocol address:** 128.143.137.1

## ■ ARP Reply from Router137:

**Source hardware address:** 00:e0:f9:23:a8:20

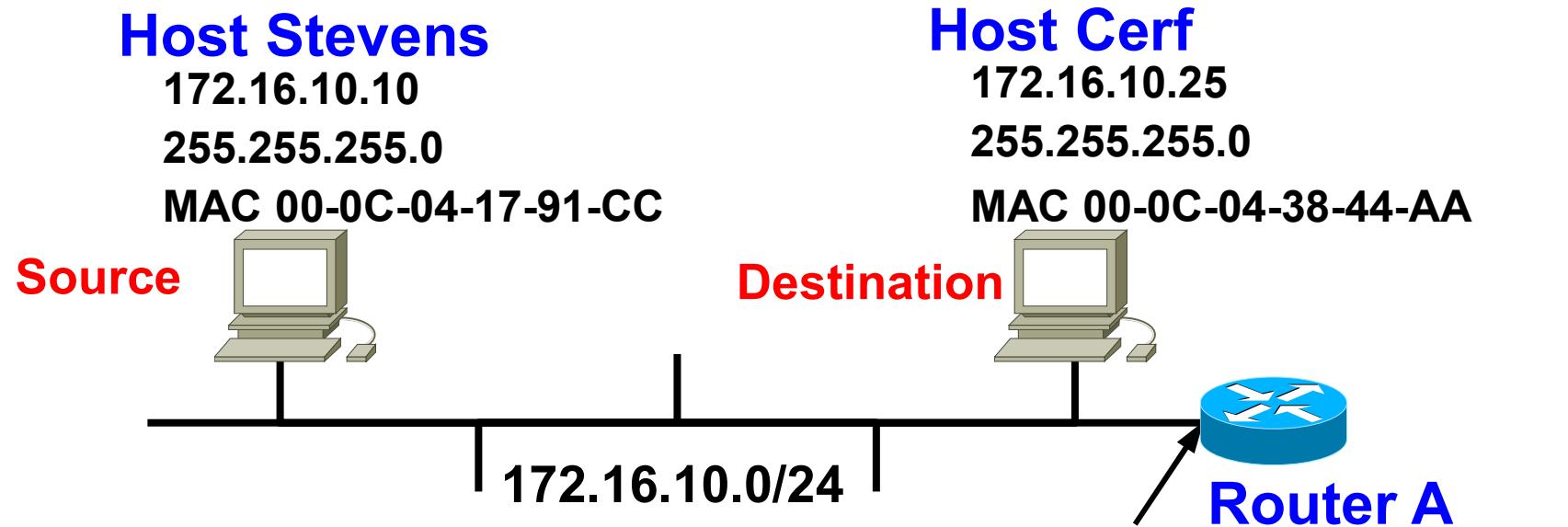
**Source protocol address:** 128.143.137.1

**Target hardware address:** 00:a0:24:71:e4:44

**Target protocol address:** 128.143.137.144

# Example 2

Two devices (hosts) are on the same subnet



Host Stevens at IP address  
172.16.10.10 want to send an IP  
packet to Host Cerf at IP address  
172.16.10.25

Ethernet 0  
172.16.10.1  
255.255.255.0  
MAC 03-0D-17-8A-F1-32

# Example 2

**Host Stevens IP Address**

**172.16.10.10**

**Host Stevens Subnet Mask**

**255.255.255.0**

**Host Stevens Network**

**172.16.10.0**

**Host Cerf IP Address**

**172.16.10.25**

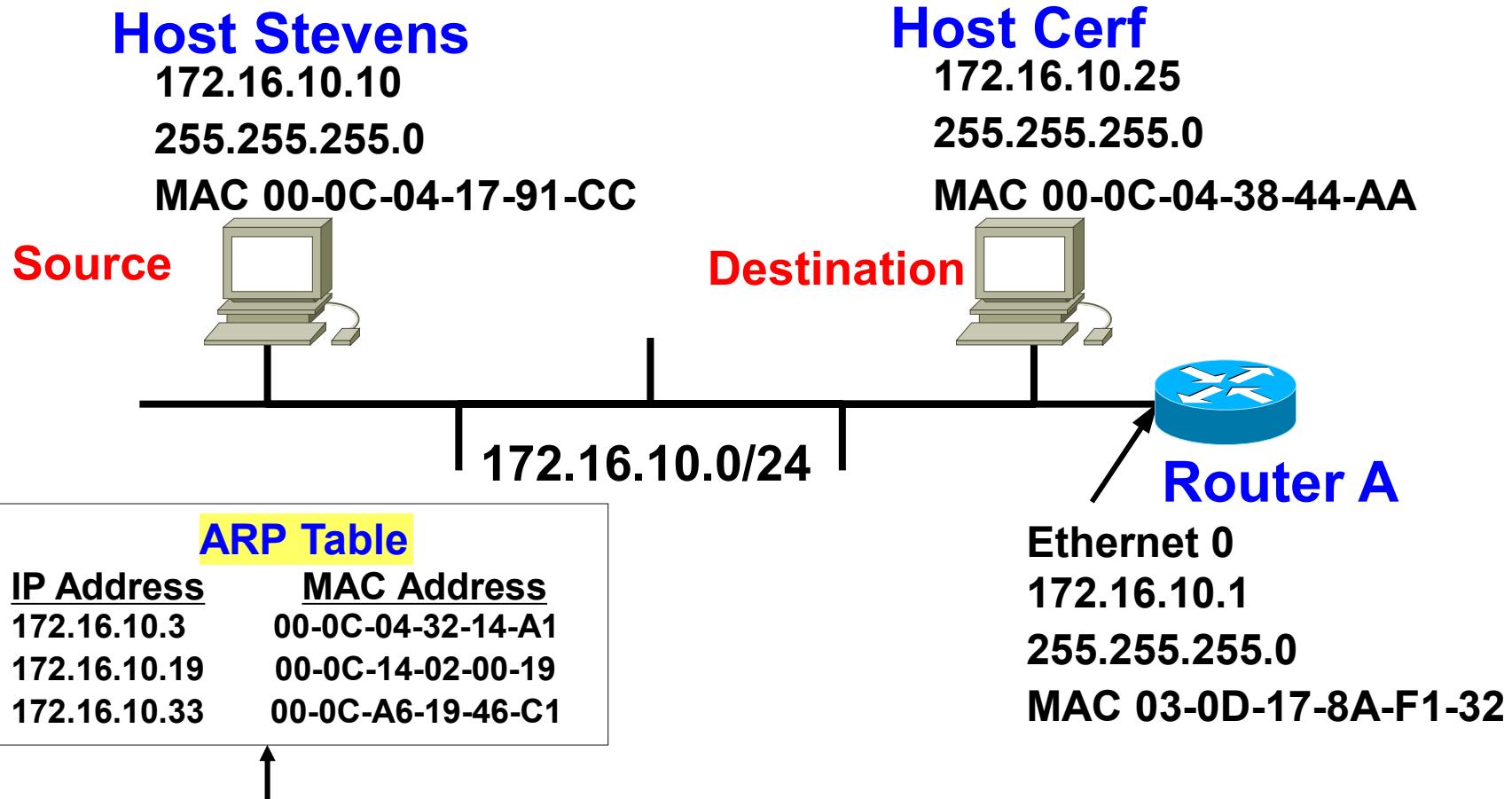
**Host Cerf Subnet Mask**

**255.255.255.0**

**Host Cerf Network**

**172.16.10.0**

# Example 2



**Destination MAC Address???**  
**Host Stevens checking its ARP table for Host Cerf's MAC address...**

# Example 2

**ARP Request from Host Stevens at 172.16.10.10**

**“Hey everyone! I have this IP Address and I need the host this belongs to, to send me their MAC address.”**

**ARP Request from 172.16.10.10**

Ethernet Header			Ethernet Data – 28 byte ARP request/reply				
Ethernet Destination Address (MAC)	Ethernet Source Address (MAC)	Frame Type	ARP header s, i.e. op field	Sender's Ethernet Address (MAC)	Sender's IP Address	Target's Ethernet Address (MAC)	Target's IP Address
FF-FF-FF-FF-FF-FF	00-0C-04-17-91-CC	0x806	op = 1	00-0C-04-17-91-CC	172.16.10.10		172.16.10.25

# Example 2

ARP Reply from Host Cerf at 172.16.10.25

“Hey sender of ARP Request! Here is my MAC address that you wanted for that IP address.”

ARP Reply from 172.16.10.25

Ethernet Header			Ethernet Data – 28 byte ARP request/reply				
Ethernet Destination Address (MAC)	Ethernet Source Address (MAC)	Frame Type	ARP header s, i.e. op field	Sender's Ethernet Address (MAC)	Sender's IP Address	Target's Ethernet Address (MAC)	Target's IP Address
00-0C-04-17-91-CC	00-0C-04-38-44-AA	0x806	op = 2	00-0C-04-38-44-AA	172.16.10.25	00-0C-04-17-91-CC	172.16.10.10

Here it is!

# Example 2

- Host Stevens receives the ARP Reply and enters Host Cerf's IP address and MAC address into its ARP Table.
- Host Stevens now has all it needs to encapsulate the IP packet into the Ethernet frame and send that packet directly to Host Cerf.

Ethernet Frame

Ethernet Header			IP Datagram from above				Ethernet Trailer
MAC Destination Address <b>00-0C-04-38-44-AA</b>	MAC Source Address <b>00-0C-04-17-91-CC</b>	Other Header Info	IP Header Info	IP Original Source Address <b>172.17.10.10</b>	IP Final Destination Address <b>172.16.10.25</b>	Data	FCS

# Example 3

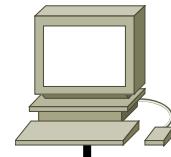
**Two hosts are on different subnets**

**Host Stevens**

172.16.10.10

255.255.255.0

MAC 00-0C-04-17-91-CC



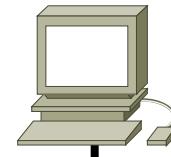
**Source**

**Host Perlman**

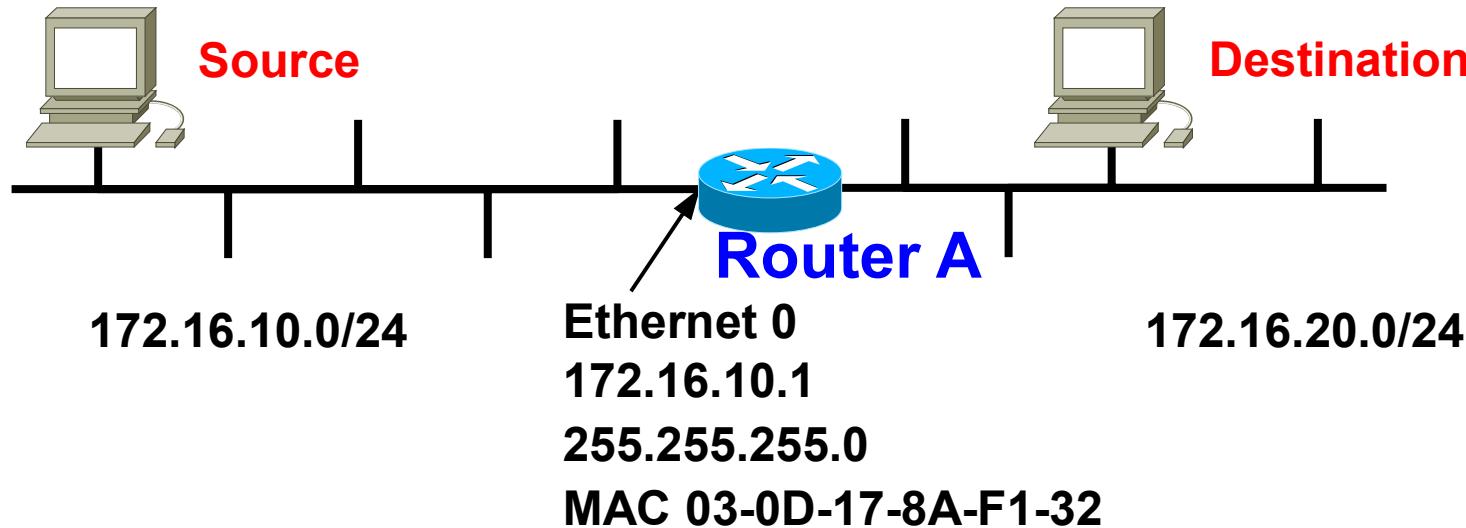
172.16.20.12

255.255.255.0

MAC 00-0C-22-A3-14-01



**Destination**



**Host Stevens at IP address 172.16.10.10 want to send an IP packet to Host Perlman at IP address 172.16.20.12**

# Example 3

- Host Stevens needs to send the packet either:
  - directly to the final destination, Host Perlman
  - OR
  - to the default gateway, the router, so it can forward it onward
- How does Host Stevens know where it needs to send this packet?

# Example 3

- Depending upon the answer, Host Stevens will either look for Host Perlman's IP address of 172.16.20.12 in its ARP table **OR** that of the default gateway, Router A's IP address of 172.16.10.1
- This is the “**big question!**”

# The BIG Question

- Which IP address does the sending host (**Stevens**) look for in its ARP table? And if that IP address is not there, what IP Address does it send an ARP Request for?
  
- Is it:
  - The IP Address of the destination host?
  - The IP Address of the default gateway (the router)?

# The Answer

- It depends on whether the final destination address is on its same subnet or that of a different subnet or network.
- The sending host **must** determine whether the final destination IP address is **on the same subnet** as itself.

# Example 3

**Host Stevens IP Address**

**172.16.10.10**

**Host Stevens Subnet Mask**

**255.255.255.0**

-----  
**Host Stevens Network**

**172.16.10.0**

**Host Perlman's IP Address**

**172.16.20.12**

**Host Perlman Subnet Mask**

**255.255.255.0**

-----  
**Host Perlman's Network**

**172.16.20.0**

# Example 3

- **Different subnets!**
- This means that Host Stevens **can not** send the packet directly to Host Perlman.
- Now, that Host Stevens knows that Host Cerf is on a different subnet, it knows that it must send the packet to the **default gateway**, the router.
- Host Stevens will look up the default gateway's IP address (which has been entered by the user or received by a DHCP server), in its ARP Table.

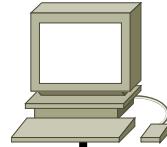
# Example 3

**Host Stevens**

172.16.10.10

255.255.255.0

MAC 00-0C-04-17-91-CC



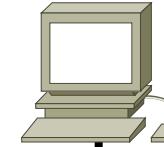
Source

**Host Perlman**

172.16.20.12

255.255.255.0

MAC 00-0C-22-A3-14-01



Destination

**Router A**

172.16.10.0/24

Ethernet 0

172.16.20.0/24

**ARP Table**

<u>IP Address</u>	<u>MAC Address</u>
172.16.10.3	00-0C-04-32-14-A1
172.16.10.19	00-0C-14-02-00-19
172.16.10.33	00-0C-A6-19-46-C1

172.16.10.1

255.255.255.0

MAC 03-0D-17-8A-F1-32



Destination MAC Address???

**Host Stevens checking its ARP table for Host Perlman's MAC address...**

# Example 3

ARP Request from Host Stevens at 172.16.10.10

“Hey everyone! I have this IP Address, 172.16.10.1, and I need the device this belongs to, to send me their MAC address.”

ARP Request from 172.16.10.10

Ethernet Header			Ethernet Data – 28 byte ARP request/reply				
Ethernet Destination Address (MAC)	Ethernet Source Address (MAC)	Frame Type	ARP header s, i.e. op field	Sender's Ethernet Address (MAC)	Sender's IP Address	Target's Ethernet Address (MAC)	Target's IP Address
FF-FF-FF-FF-FF-FF	00-0C-04-17-91-CC	0x806	op = 1	00-0C-04-17-91-CC	172.16.10.10		172.16.10.1

# Example 3

ARP Reply from Router A at 172.16.10.1

“Hey sender of ARP Request! Here is my MAC address that you wanted for that IP address.”

ARP Reply from 172.16.10.1

Ethernet Header			Ethernet Data – 28 byte ARP request/reply				
Ethernet Destination Address (MAC)	Ethernet Source Address (MAC)	Frame Type	ARP header s, i.e. op field	Sender's Ethernet Address (MAC)	Sender's IP Address	Target's Ethernet Address (MAC)	Target's IP Address
00-0C-04-17-91-CC	03-0D-17-8A-F1-32	0x806	op = 2	03-0D-17-8A-F1-32	172.16.10.1	00-0C-04-17-91-CC	172.16.10.10

Here it is!

# Example 3

- Host Stevens receives the ARP Reply and enters Router A's IP address and MAC address into its ARP Table.
- Host Stevens now has all it needs to encapsulate the IP packet into the Ethernet frame and send that packet to Router A.

Ethernet Frame

Ethernet Header			IP Datagram from above				Ethernet Trailer
MAC Destination Address <b>03-0D-17-8A-F1-32</b>	MAC Source Address <b>00-0C-04-17-91-CC</b>	Other Header Info	IP Header Info	IP Original Source Address <b>172.17.10.10</b>	IP Final Destination Address <b>172.16.10.1</b>	Data	FCS

# Example 3

- It is now up to Router A to forward the packet onward.

# IP Forwarding Algorithm

w=source-IP-address W=source-MAC-address

y=source-subnet-mask

x=dest-IP-address X=dest-MAC-address

z=source-default-router-IP-address

Z= source-default-router-MAC-address

if (w and y)=(x and y) then //源主机和目的主机在一个子网

ARP.lookup\_MAC(x) //在ARP缓存中查找目的主机的MAC地址

if found then

IP.send\_packet(X,x)

else

ARP.send\_ARP(x) //广播ARP请求，目的主机响应ARP，

//源主机更新ARP缓存

IP.send\_packet(X,x) //发送分组给目的主机

end if

//next slide

# IP Forwarding Algorithm

w=source-IP-address W=source-MAC-address

y=source-subnet-mask

x=dest-IP-address X=dest-MAC-address

z=source-default-router-IP-address

Z= source-default-router-MAC-address

//previous slide

else //源主机和目的主机在不同的子网

ARP.lookup\_MAC(z) //在ARP缓存中查找缺省路由器的MAC地址

if found then

IP.send\_packet(Z,x) //发送分组给缺省路由器

else

ARP.send\_ARP(z) //广播ARP请求，缺省路由器用代理ARP响应

//ARP，源主机更新ARP缓存

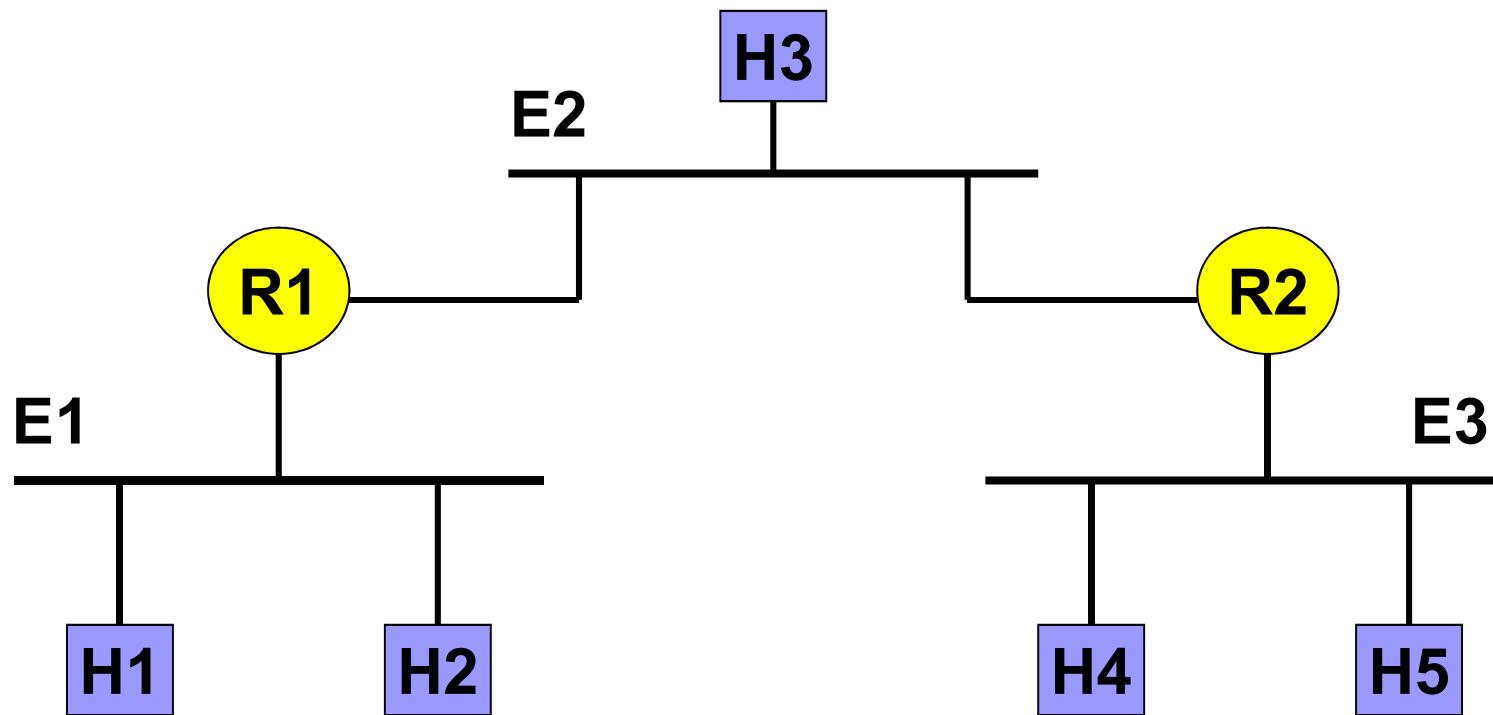
IP.send\_packet(Z,x) //发送分组给缺省路由器

end if

end if

# IP Forwarding Algorithm

■ Example: 某网络如下:



# IP Forwarding Algorithm

- 例：每个设备的IP地址和MAC地址如下表：

设备	IP地址	MAC地址
H1	IP_H1	MAC_H1
H2	IP_H2	MAC_H2
R1	IP_R1_E1 IP_R1_E2	MAC_R1_E1 MAC_R1_E2
R2	IP_R2_E2 IP_R2_E3	MAC_R2_E2 MAC_R2_E3

# IP Forwarding Algorithm

- 例： H1将R1作为缺省路由器， 向H5发送一个IP分组。

**问题：**按发送顺序写出每个帧中所包括的**MAC地址**和**IP地址**。

- **解：**帧中可能包含**IP分组**、**ARP请求分组**或**ARP响应分组**

# IP Forwarding Algorithm

- 解：用BCAST表示广播地址，用?????表示未知地址，用空表示未使用的字段。对于ARP，用第二行表示ARP分组中包含的MAC地址和IP地址，并用括号()括起来。

# IP Forwarding Algorithm

帧序号	协议类型	发送方	接收方	源MAC	目的MAC	源IP	目的IP
1	ARP	H1	E1	MAC_H1 (MAC_H1)	BCAST ?????	IP_H1	IP_R1_E1)
2	ARP	R1	H1	MAC_R1_E1 (MAC_R1_E1)	MAC_H1 MAC_H1	IP_R1_E1	IP_H1)
3	IP	H1	R1	MAC_H1	MAC_R1_E1	IP_H1	IP_H5
4	ARP	R1	E2	MAC_R1_E2 (MAC_R1_E2)	BCAST ?????	IP_R1_E2	IP_R2_E2)
5	ARP	R2	R1	MAC_R2_E2 (MAC_R2_E2)	MAC_R1_E2 MAC_R1_E2	IP_R2_E2	IP_R1_E2)
6	IP	R1	R2	MAC_R1_E2	MAC_R2_E2	IP_H1	IP_H5

# IP Forwarding Algorithm

帧序号	协议类型	发送方	接收方	源MAC	目的MAC	源IP	目的IP
7	ARP	R2	E3	MAC_R2_E3 (MAC_R2_E3)	BCAST ?????	IP_R2_E3	IP_H5)
8	ARP	H5	R2	MAC_H5 (MAC_H5)	MAC_R2_E3 MAC_R2_E3	IP_H5	IP_R2_E3)
9	IP	R2	H5	MAC_R2_E3	MAC_H5	IP_H1	IP_H5

# RARP

- **Reverse Address Resolution Protocol**
- **Problem:** How does a host know its own IP address?
- **Solution:** The host uses a limited broadcast (i.e. restricted to its own network) to query a RARP server for its IP address. The RARP server maintains a table of (data link address, IP address) mappings.
- **Note:** we need a RARP server for every broadcast segment, that comes BOOTP(using UDP).

# Routing in the Internet

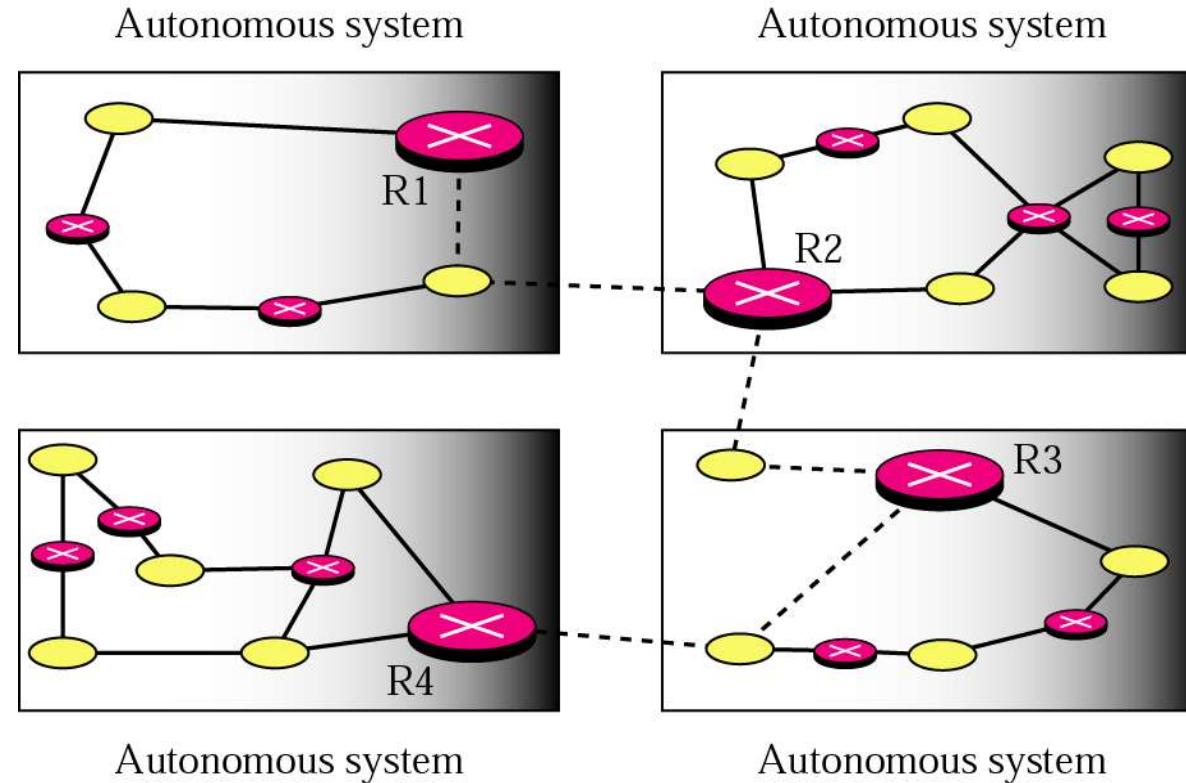
- The Global Internet consists of **Autonomous Systems (AS)** interconnected with each other:
- **Two-level routing:**
  - **Intra-AS:** administrator is responsible for choice
  - **Inter-AS:** unique standard

**Hierarchical Routing**

# Routing in the Internet

## ■ Autonomous Systems (AS)

**Group of networks and routers under the authority of a single administration.**

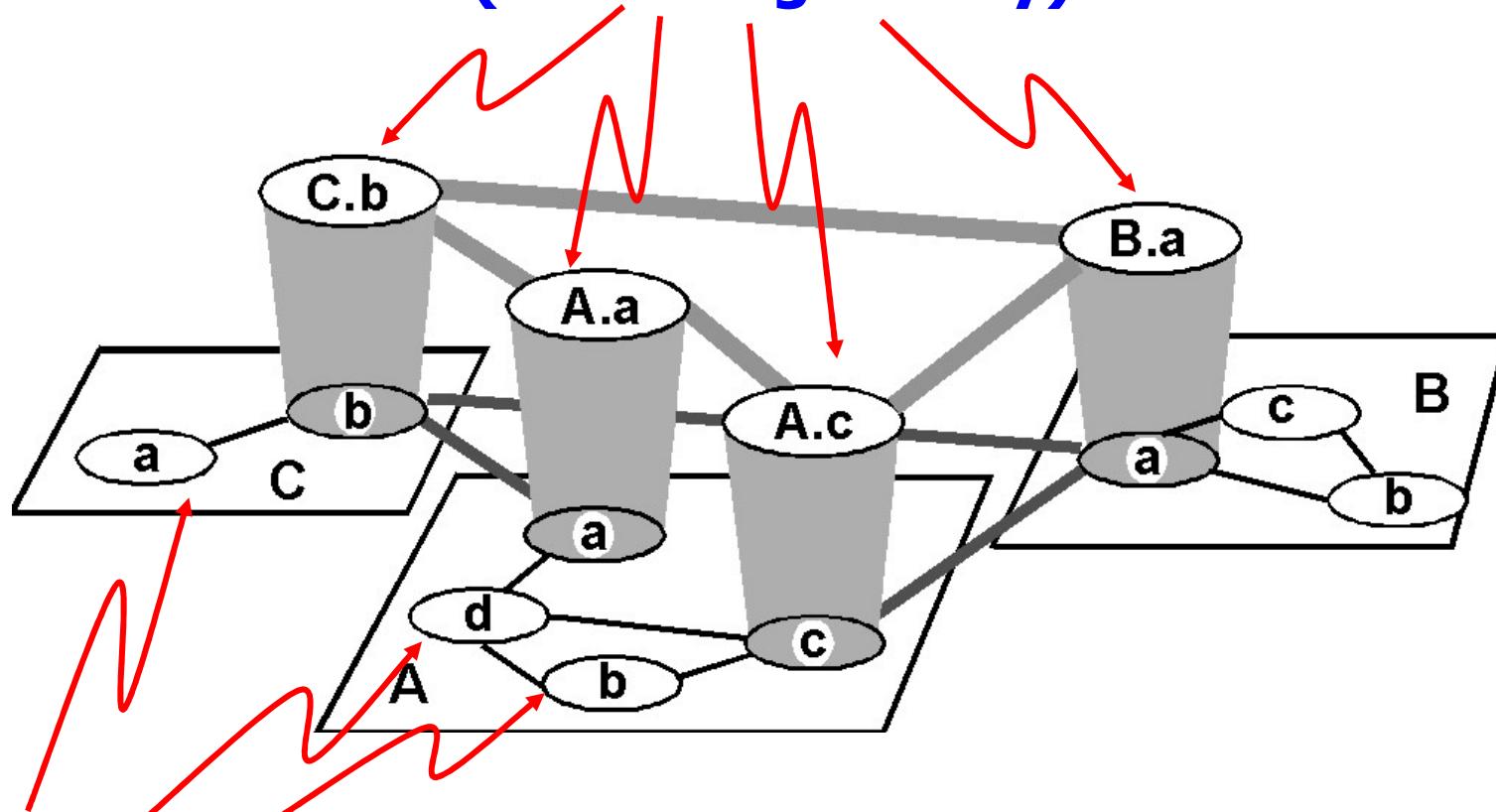


# Routing in the Internet

- Routing inside an autonomous system is referred to as **interior routing**.
- Routing between autonomous systems is referred to as **exterior routing**.

# Internet AS Hierarchy

**Inter-AS border (exterior gateway) routers**



**Intra-AS interior (gateway) routers**

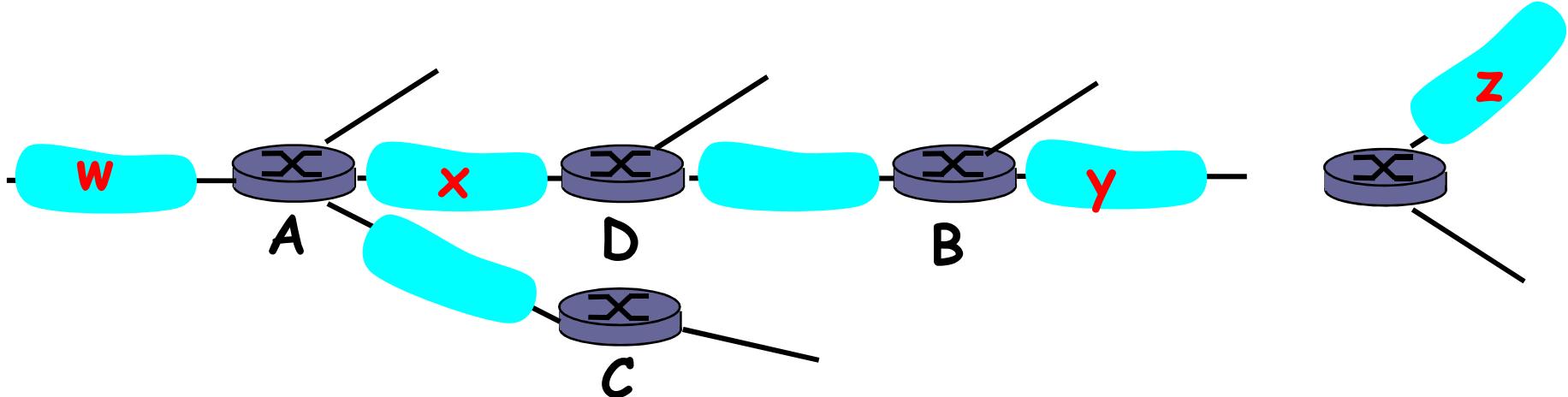
# Intra-AS Routing Protocols

- Also known as **Interior Gateway Protocols (IGP)**
- Most common IGPs:
  - **RIP: Routing Information Protocol**
  - **OSPF: Open Shortest Path First**
  - **IGRP: Interior Gateway Routing Protocol (Cisco propri.)**

# RIP ( Routing Information Protocol)

- Based on **distance vector** algorithm
- Included in **BSD-UNIX Distribution in 1982**
- Distance metric: # of hops (**max = 15 hops**)
- **Distance vectors:** exchanged every 30 sec via Response Message (also called advertisement) **using UDP**
- Each advertisement: route to up to **25 destination nets**

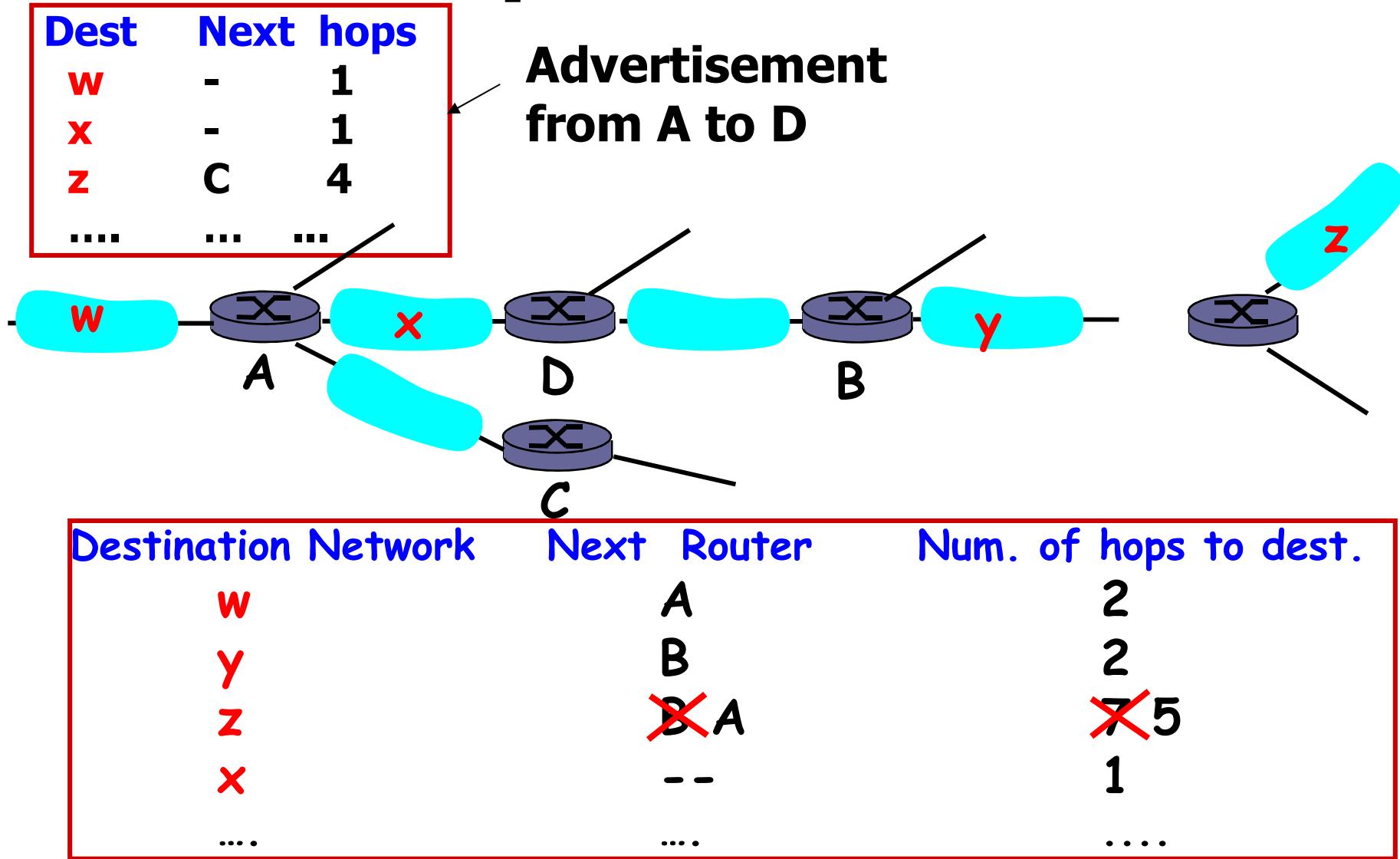
# RIP: Example



Destination Network	Next Router	Num. of hops to dest.
W	A	2
Y	B	2
Z	B	7
X	--	1
....	....	....

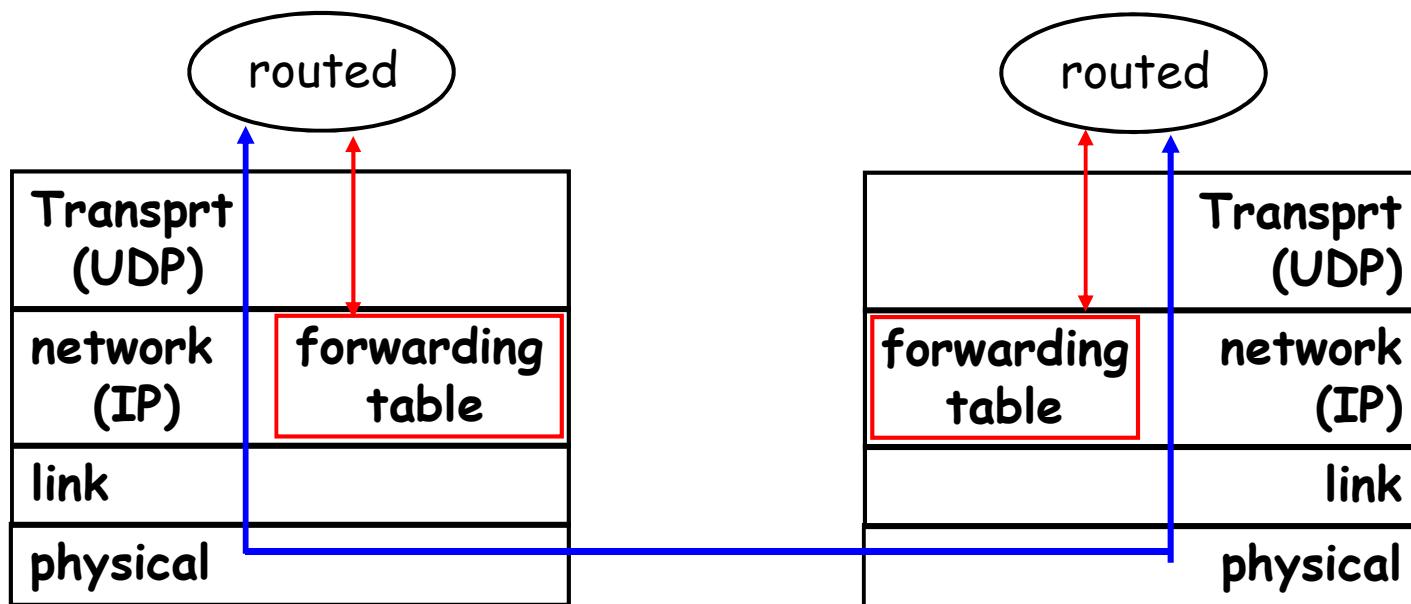
Routing table in D

# RIP: Example



# RIP Table processing

- RIP routing tables managed by **application-level** process called **route-d (daemon)**
- advertisements sent in **UDP** packets, periodically repeated



# RIP: Link Failure and Recovery

- If no advertisement heard after **180 sec**  
--> neighbor/link declared dead
  - routes via neighbor invalidated
  - new advertisements sent to neighbors
  - neighbors in turn send out new advertisements (if tables changed)
  - link failure info quickly propagates to entire net
  - poison reverse used to prevent ping-pong loops (**infinite distance = 16 hops**)

# OSPF (Open Shortest Path First)

- “open”: publicly available
- **Uses Link State algorithm**
  - LS packet dissemination
  - Topology map at each node
  - Route computation using Dijkstra’s algorithm
- OSPF advertisement carries one entry per neighbor router
- Advertisements disseminated to **entire AS (via flooding)**
  - **Carried in OSPF messages directly over IP (rather than TCP or UDP)**

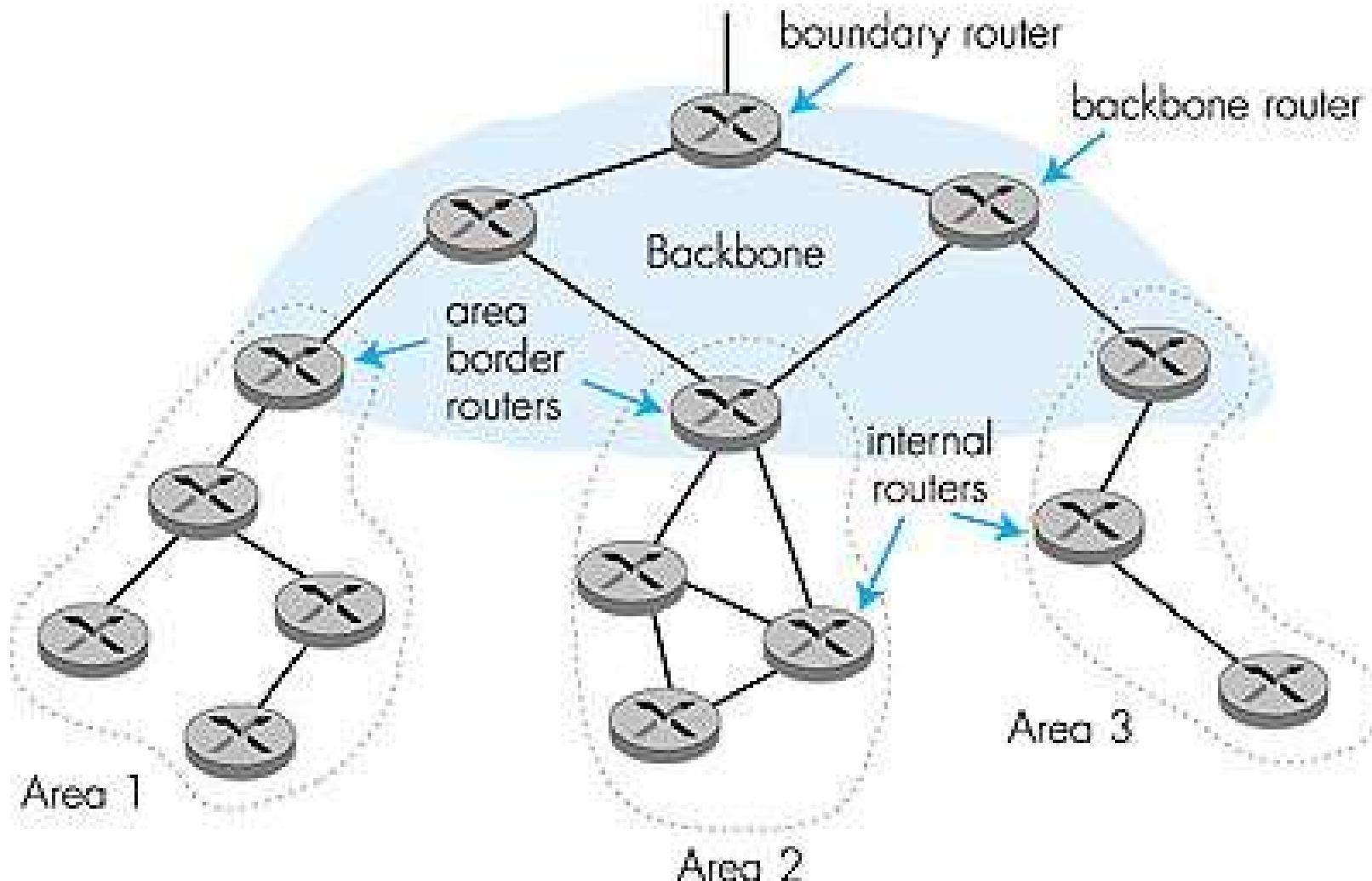
# OSPF “advanced” features (not in RIP)

- **Security:** all OSPF messages authenticated (to prevent malicious intrusion); TCP connections used
- **Multiple same-cost paths allowed (load balancing)**
  - only one path in RIP
- For each link, **multiple cost metrics** for different **ToS** (eg, satellite link cost set “low” for best effort; high for real time)

# OSPF “advanced” features (not in RIP)

- Integrated uni- and **multicast** support:
  - Multicast OSPF (MOSPF) uses same topology data base as OSPF
- **Hierarchical** OSPF in large domains.
- Support **IP tunneling**.

# Hierarchical OSPF



# Hierarchical OSPF

- **Two-level hierarchy:** local area, backbone.
  - Link-state advertisements only in area
  - each node has detailed area topology; only know direction (shortest path) to nets in other areas.
- **Area border routers:** “summarize” distances to nets in own area, advertise to other Area Border routers.
- **Backbone routers:** run OSPF routing limited to backbone.
- **Boundary routers:** connect to other ASs.

# OSPF

## The five types of OSPF messages.

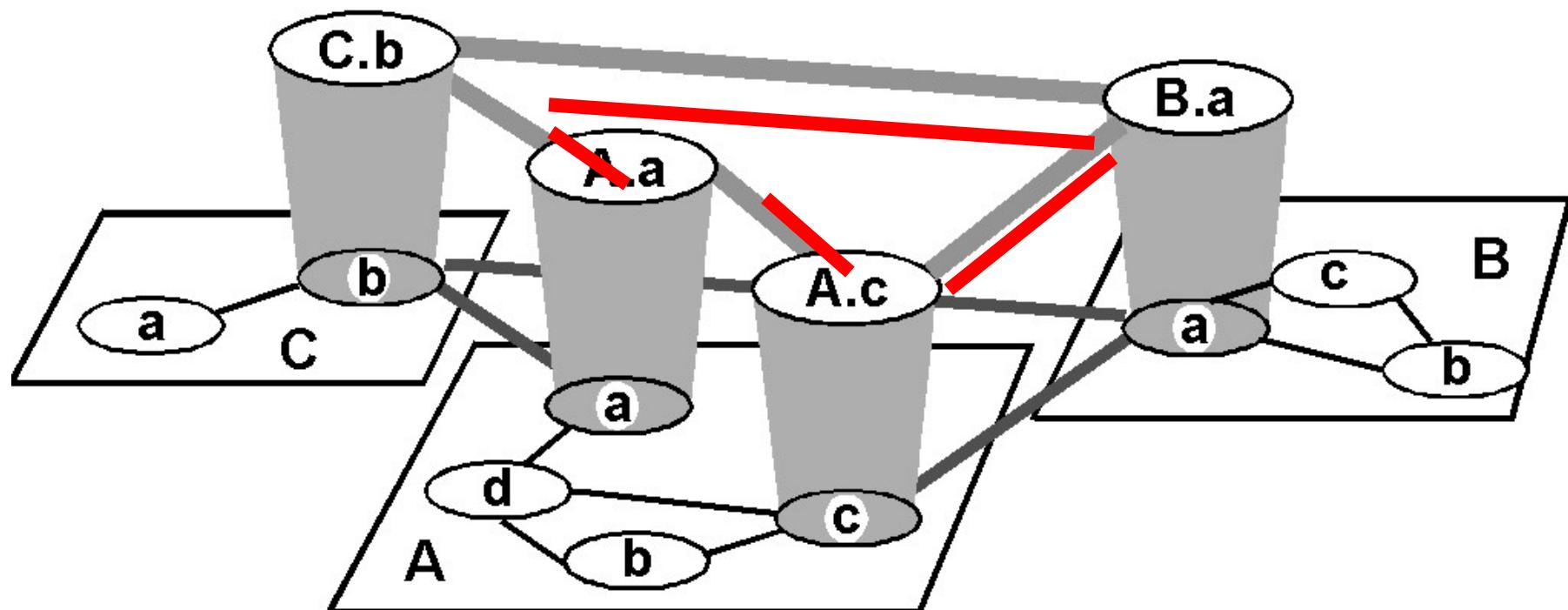
Message type	Description
Hello	Used to discover who the neighbors are
Link state update	Provides the sender's costs to its neighbors
Link state ack	Acknowledges link state update
Database description	Announces which updates the sender has
Link state request	Requests information from the partner

- **Note:** We can use the same algorithm for the areas and the backbone. In fact, the backbone behaves as just another area. This means it should be contiguous.

# IGRP (Interior Gateway Routing Protocol)

- CISCO proprietary; successor of RIP (mid 80s)
- Distance Vector, like RIP
- several cost metrics (delay, bandwidth, reliability, load etc)
- uses TCP to exchange routing updates
- Loop-free routing via Distributed Updating Alg. (DUAL) based on *diffused computation*

# Inter-AS routing



# Internet inter-AS routing: BGP

- **BGP (Border Gateway Protocol)**
  - Current version is BGP-4
- **BGP provides each AS a means to:**
  1. Obtain subnet **reachability information** from neighboring ASs.
  2. Propagate the **reachability information** to all routers internal to the AS.
  3. Determine “**good**” routes to subnets based on **reachability information and policy**.
- **Allows a subnet to advertise its existence to rest of the Internet: “*I am here*”**

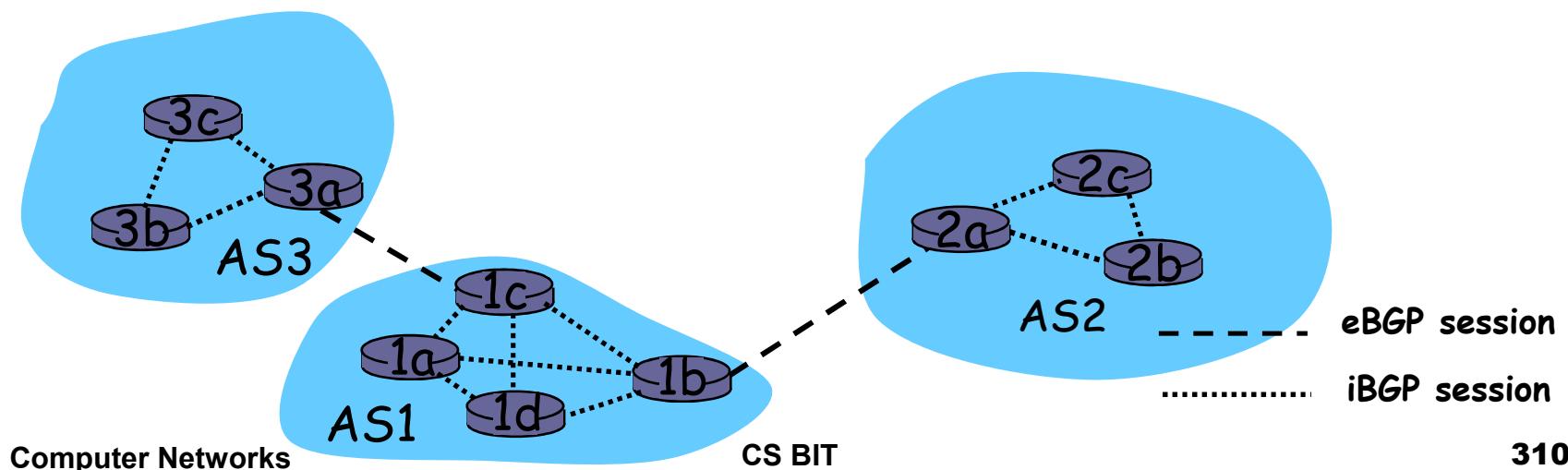
# Internet inter-AS routing: BGP

- **Path Vector protocol:**
  - similar to Distance Vector protocol
  - each Border Gateway broadcast to neighbors (peers) *entire path* (I.e, sequence of ASs) to destination
  - E.g., Gateway X may send its path to dest. Z:

**Path (X,Z) = X,Y<sub>1</sub>,Y<sub>2</sub>,Y<sub>3</sub>,...,Z**

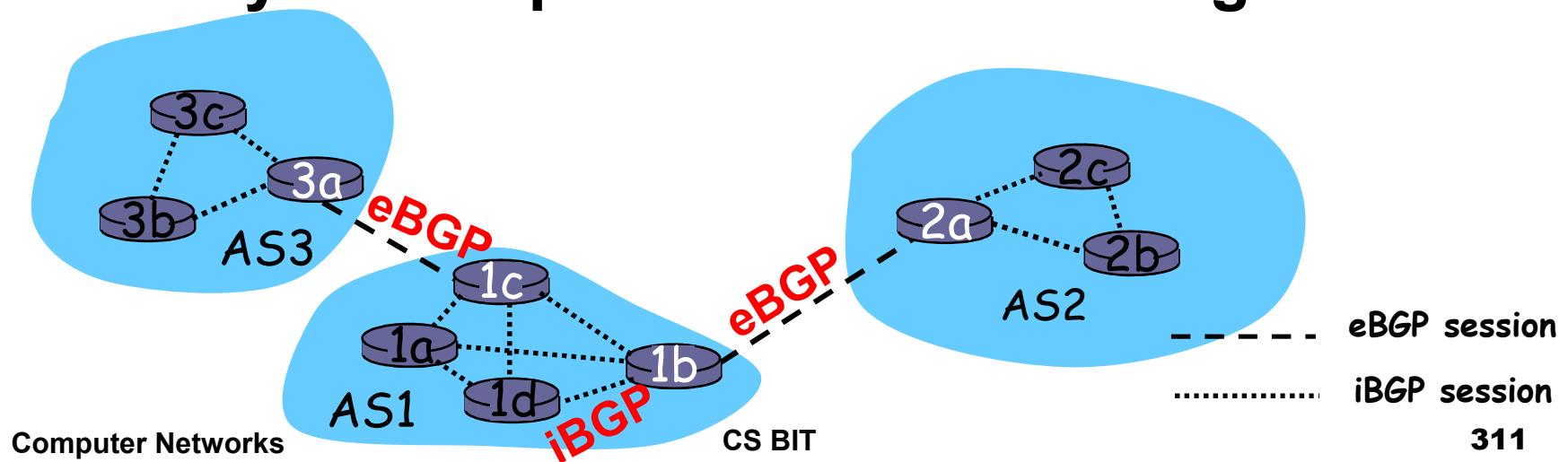
# BGP basics

- Pairs of routers (BGP peers) exchange routing info over semi-permanent **TCP connections: BGP sessions**. Note that BGP sessions do not correspond to physical links.
- When AS2 advertises a prefix to AS1, AS2 is *promising* it will forward any datagrams destined to that prefix towards the prefix.
  - AS2 can aggregate prefixes in its advertisement



# Distributing reachability info

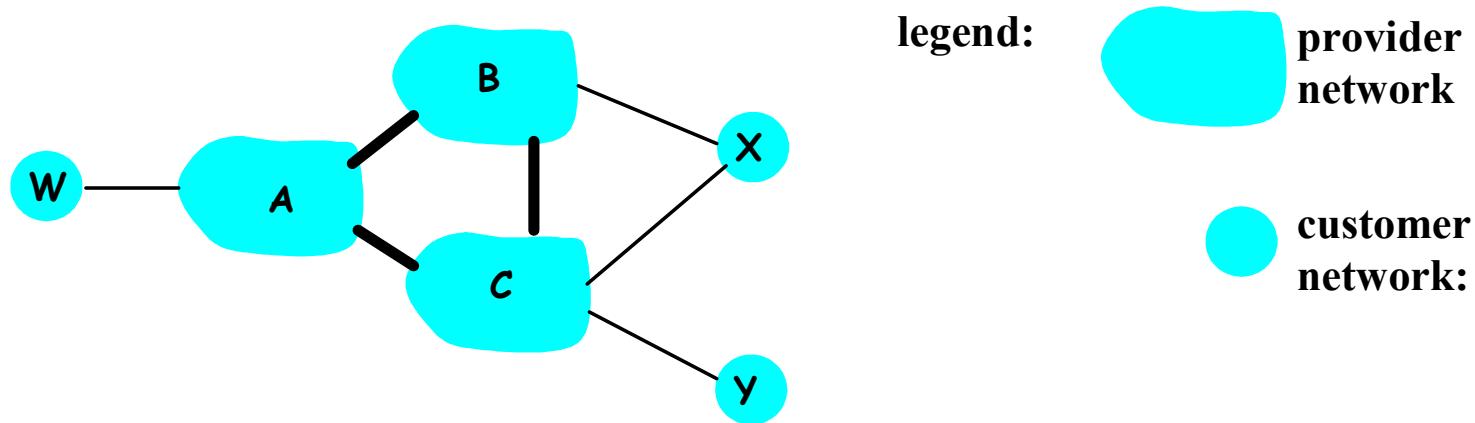
- With **eBGP** session between 3a and 1c, AS3 sends prefix reachability info to AS1.
- 1c can then use **iBGP** to distribute this new prefix reach info to all routers in AS1
- 1b can then re-advertise the new reach info to AS2 over the 1b-to-2a **eBGP** session
- When router learns about a new prefix, it creates an entry for the prefix in its forwarding table.



# Path attributes & BGP routes

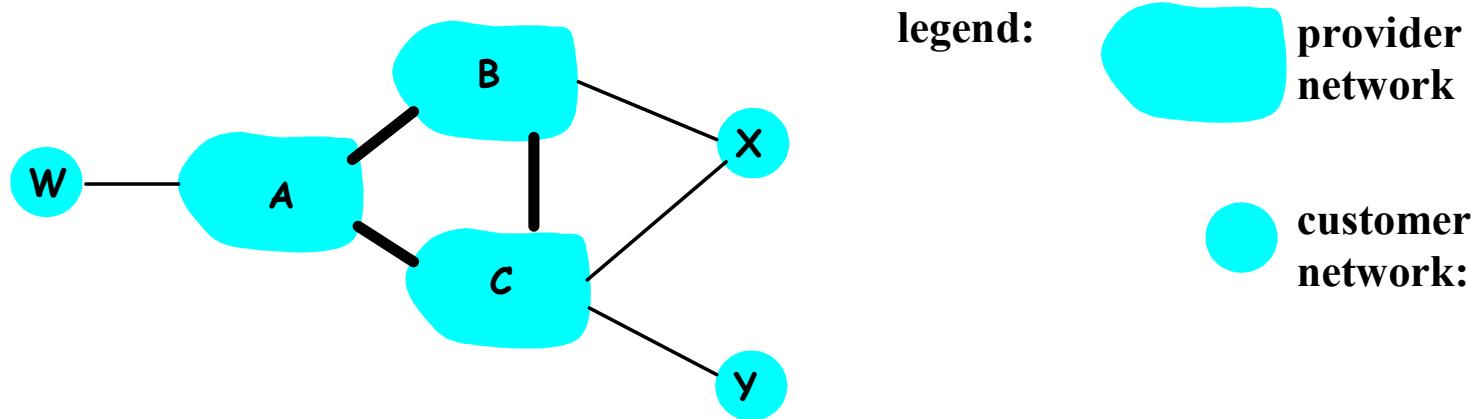
- When advertising a prefix, advert includes BGP attributes.  
**prefix + attributes = “route”**
- Two important attributes:
  - **AS-PATH:** contains the ASs through which the advert for the prefix passed: AS 67 AS 17
  - **NEXT-HOP:** Indicates the specific internal-AS router to next-hop AS. (There may be multiple links from current AS to next-hop-AS.)
- When gateway router receives route advert, uses **import policy** to accept/decline.

# BGP routing policy (1)



- A,B,C are **provider networks**
- X,W,Y are customer (of provider networks)
- X is **dual-homed**: attached to two networks
  - X does not want to route from B via X to C
  - .. so X will not advertise to B a route to C

# BGP routing policy (2)



- A advertises to B the path AW
- B advertises to X the path BAW
- Should B advertise to C the path BAW ?
  - No way! B gets no “revenue” for routing CBAW since neither W nor C are B's customers
  - B wants to force C to route to w via A
  - B wants to route **only** to/from its customers!

# Internet inter-AS routing: BGP

- **BGP messages exchanged using TCP.**
- **BGP messages:**
  - **OPEN:** opens TCP connection to peer and authenticates sender
  - **UPDATE:** advertises new path (or withdraws old)
  - **KEEPALIVE:** keeps connection alive in absence of UPDATES; also ACKs OPEN request
  - **NOTIFICATION:** reports errors in previous msg; also used to close connection

# Why different Intra- and Inter-AS routing ?

## Policy:

- Inter-AS: admin wants control over how its traffic routed, who routes through its net.
- Intra-AS: single admin, so no policy decisions needed

## Scale:

- hierarchical routing saves table size, reduced update traffic

## Performance:

- Intra-AS: can focus on performance
- Inter-AS: policy may dominate over performance

# IPv6 - IP Version 6

## ■ **IP Version 6**

- the successor to the currently used IPv4
- Specification completed in 1994
- Makes improvements to IPv4 (no revolutionary changes)

# Problems in IPv4

## ■ Scalability

- Address run out

- Explosive routing (bottle neck of Internet speed)



The most urgent  
thing!!

## ■ QoS

- Best effort is not enough

- Commercialized Internet

## ■ Security

# Effort On Saving IPv4

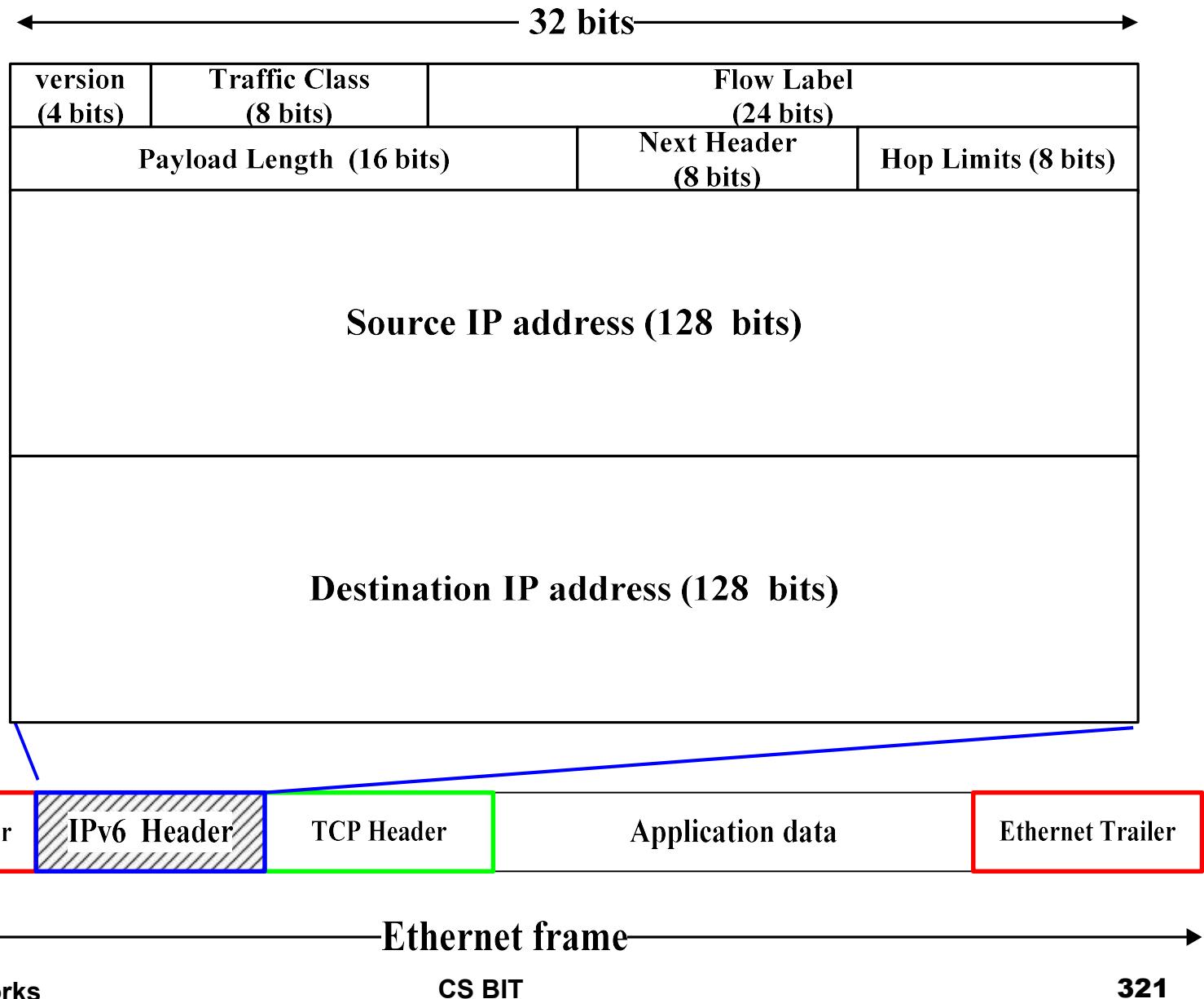
- VLSM(Variable Length Subnet Mask)
- CIDR (Classless Inter-Domain Routing)
- NAT (Net Address Translation)
- L3 Switching, MPLS
- RSVP, RTP/RTCP, DirectRoute, SSL
  
- **However**, due to scalability reason, a new IP protocol has to be developed

# IPv6 - IP Version 6

- One (not the only !) feature of IPv6 is a significant increase in of the IP address to **128 bits (16 bytes)**
  - IPv6 will solve – for the foreseeable future – the problems with IP addressing
  - $10^{24}$  addresses per square inch on the surface of the Earth.

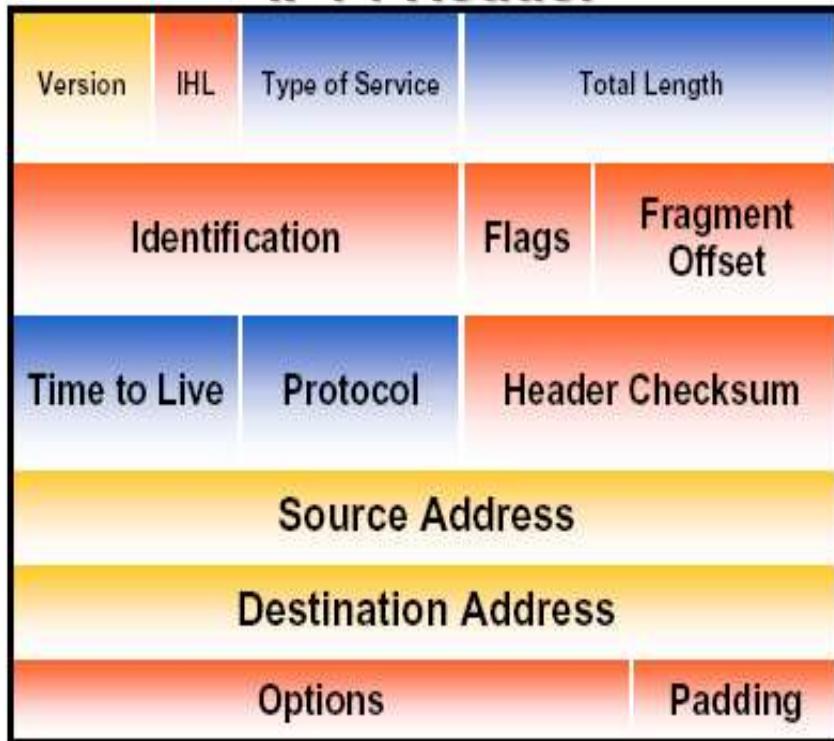
# IPv6 Header

The IPv6 fixed header (required).

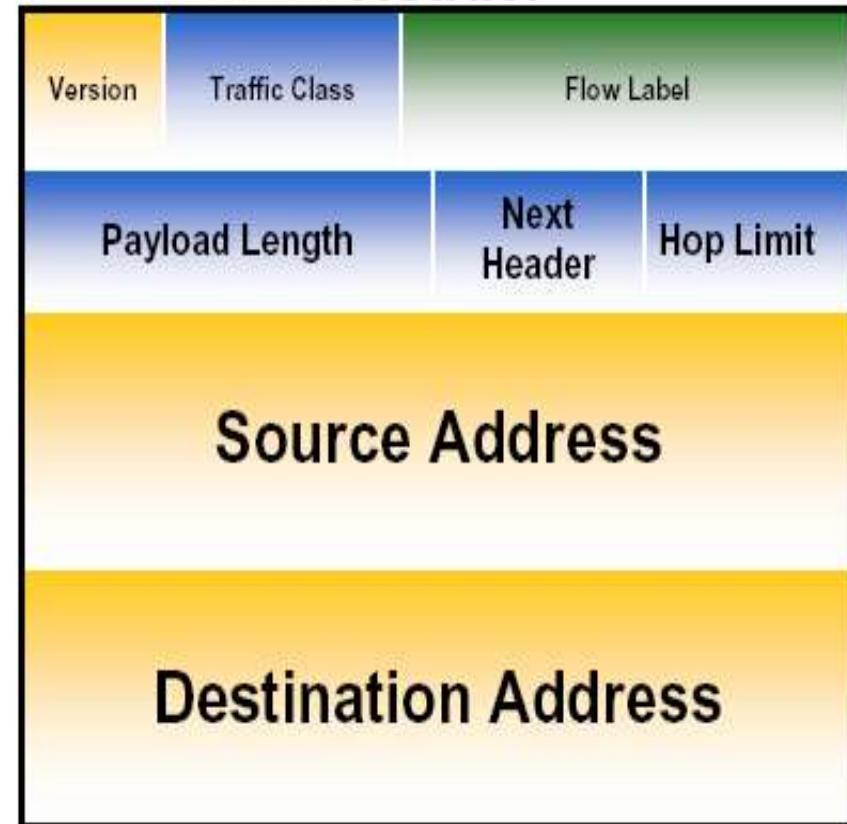


# Header: IPv4 vs. IPv6

IPv4 Header



IPv6 Header



Legend

- Field's name kept from IPv4 to IPv6
- Fields not kept in IPv6
- Name & position changed in IPv6
- New field in IPv6

# Extension Headers

## IPv6 extension headers.

Extension header	Description
Hop-by-hop options	Miscellaneous information for routers
Destination options	Additional information for the destination
Routing	Loose list of routers to visit
Fragmentation	Management of datagram fragments
Authentication	Verification of the sender's identity
Encrypted security payload	Information about the encrypted contents

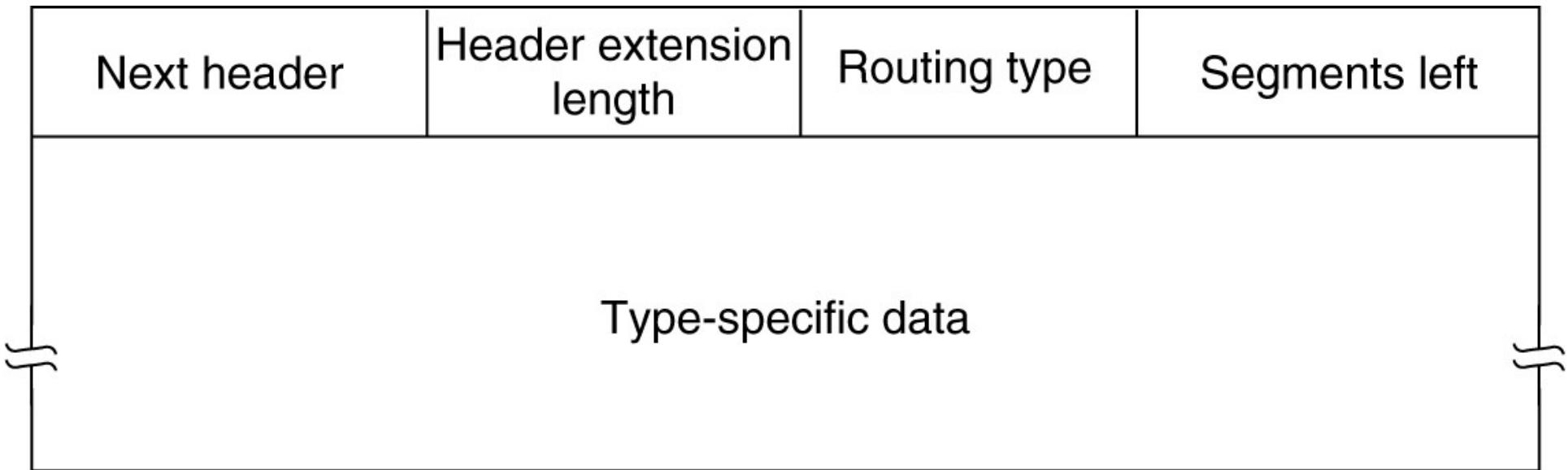
# Extension Headers (2)

**The hop-by-hop extension header for large datagrams (jumbograms).**

Next header	0	194	4
Jumbo payload length			

# Extension Headers (3)

## The extension header for routing.



# IPv6 vs. IPv4: Address Comparison

- **IPv4** has a maximum of  
 $2^{32} \approx 4 \text{ billion addresses}$
- **IPv6** has a maximum of  
 $2^{128} = (2^{32})^4 \approx 4 \text{ billion} \times 4 \text{ billion} \times 4 \text{ billion} \times 4 \text{ billion}$   
addresses

# Notation of IPv6 addresses

- **Convention:** The 128-bit IPv6 address is written as eight 16-bit integers (using hexadecimal digits for each integer)

**CEDF:BP76:3245:4464:FACE:2E50:3025:DF12**

- **Short notation:**

- Abbreviations of leading zeroes:

**CEDF:BP76:0000:0000:009E:0000:3025:DF12 →**

**CEDF:BP76:0:0:9E :0:3025:DF12**

- “:0000:0000:0000” can be written as “::”

**CEDF:BP76:0:0:FACE:0:3025:DF12 →**

**CEDF:BP76::FACE:0:3025:DF12**

# Notation of IPv6 addresses

## ■ Short notation:

- CIDR slash notation

- 60 bit prefix **12AB00000000CD3:**

**12AB:0000:0000:CD30:0000:0000:0000:00  
00/60**

OR **12AB:CD30:0:0:0:0/60**

OR **12AB:0:0:CD30::/60**

# IPv6 Provider-Based Addresses

- The first IPv6 addresses will be allocated to a provider-based plan

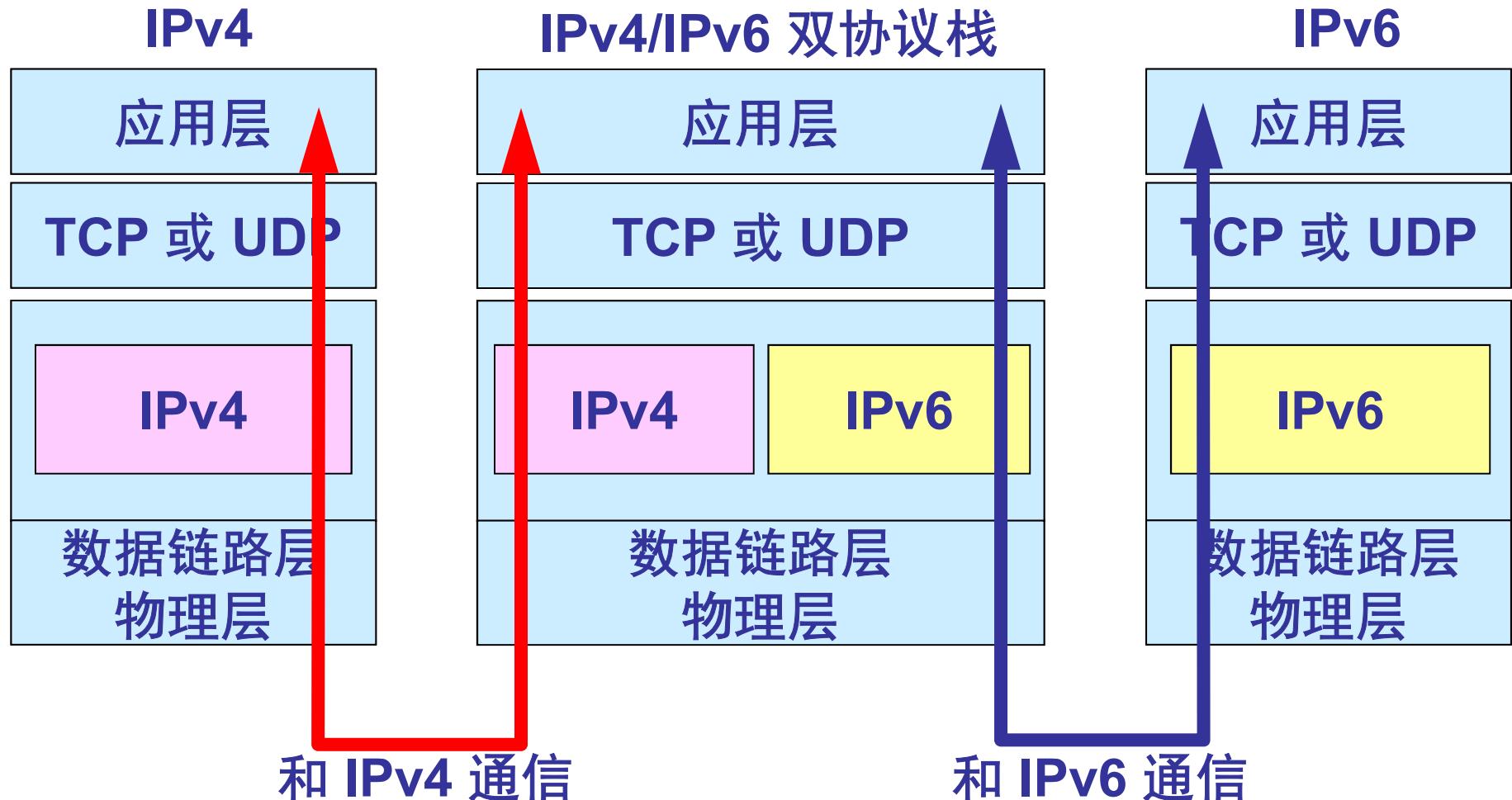


- **Type:** Set to “010” for provider-based addresses
- **Registry:** identifies the agency that registered the address
- **Provider:** Id of Internet access provider (*16 bits*)
- **Subscriber:** Id of the organization at provider (*24 bits*)
- **Subnetwork:** Id of subnet within organization (*32 bits*)
- **Interface:** identifies an interface at a node (*48 bits*)

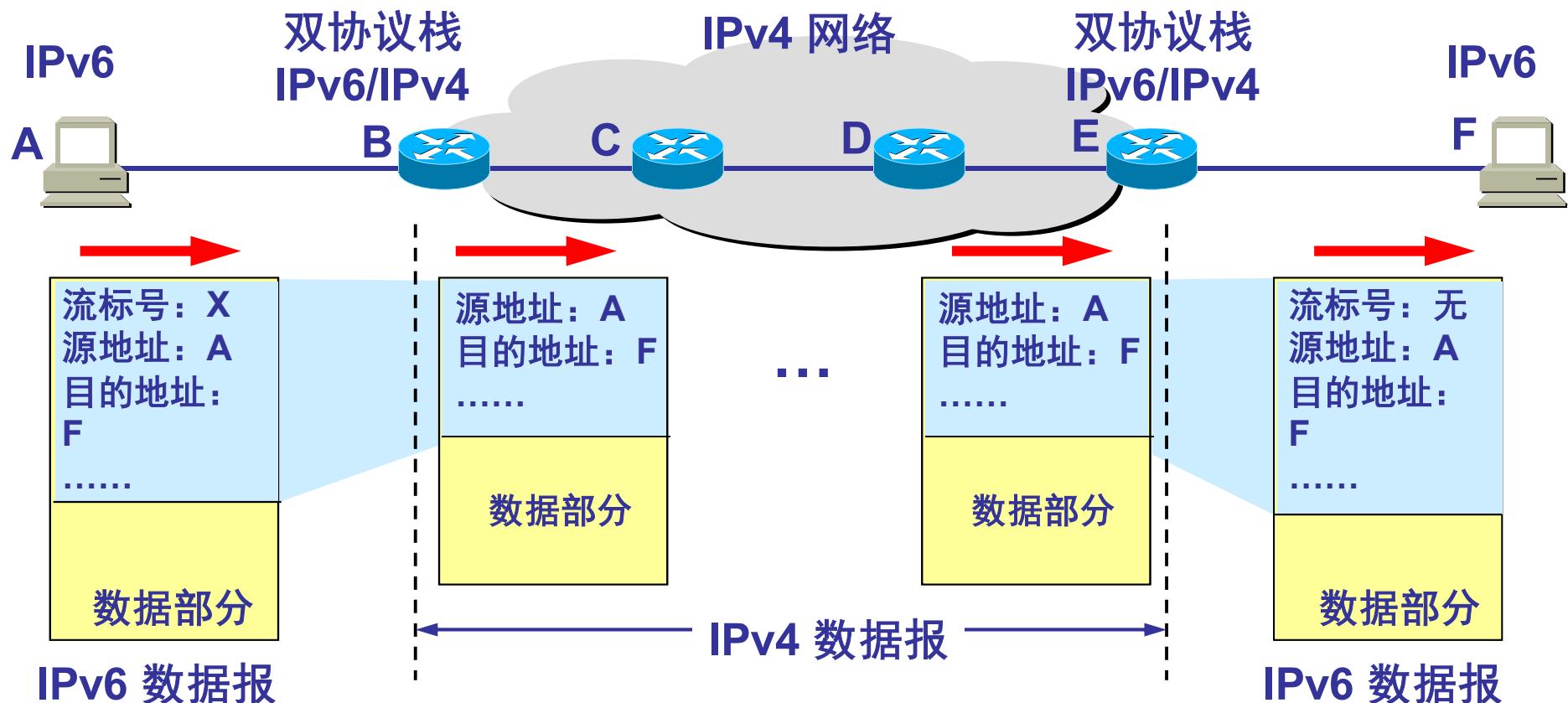
# Transition From IPv4 To IPv6

- Not all routers can be upgraded simultaneous
  - no “flag days”
  - How will the network operate with mixed IPv4 and IPv6 routers?
- *Dual Stack*
- *Tunneling:* IPv6 carried as payload in IPv4 datagram among IPv4 routers

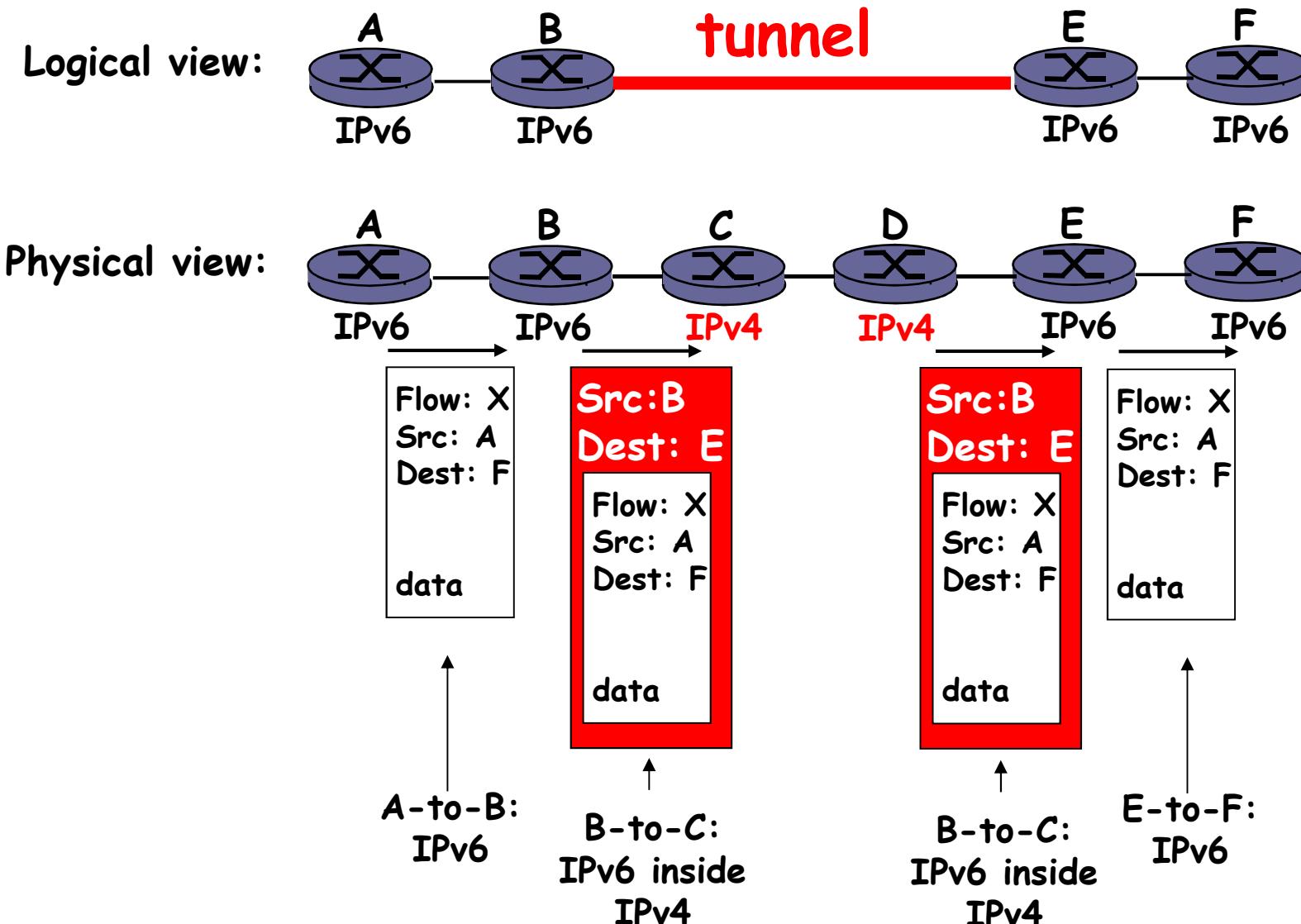
# Transition From IPv4 To IPv6



# Transition From IPv4 To IPv6

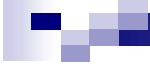


# Tunneling



# Chapter 5: Roadmap

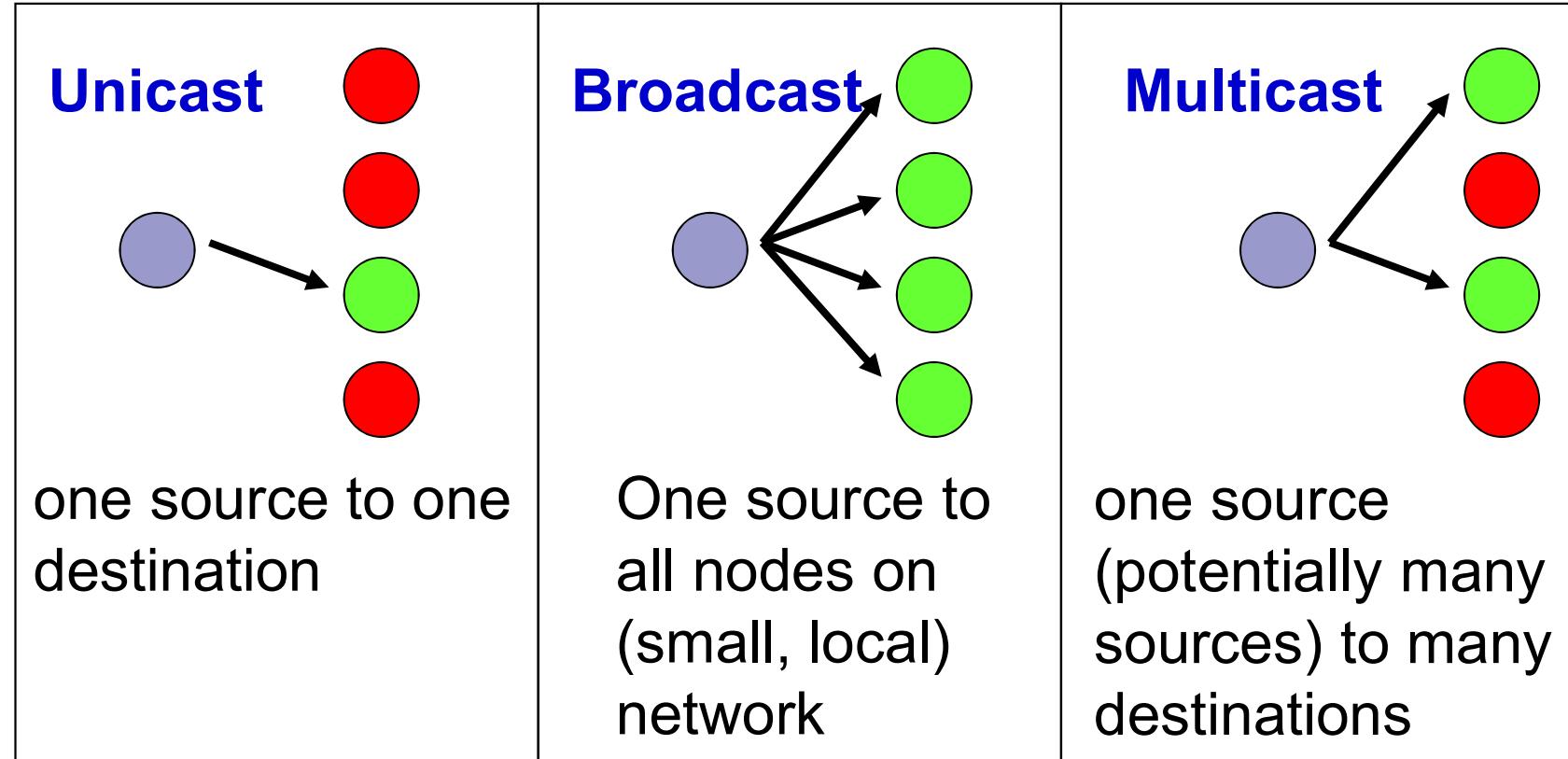
- Network Layer and Design Issues
- Routing Algorithms
- Congestion Control Algorithm
- IP
- **IP Multicasting**
- Mobile IP



# **IP Multicast Outline**

- **What is IP Multicast?**
- **IP Multicast Address**
- **Layer-2 Multicast**
- **IGMP**
- **Multicast Routing**

# Three Transmission Mode



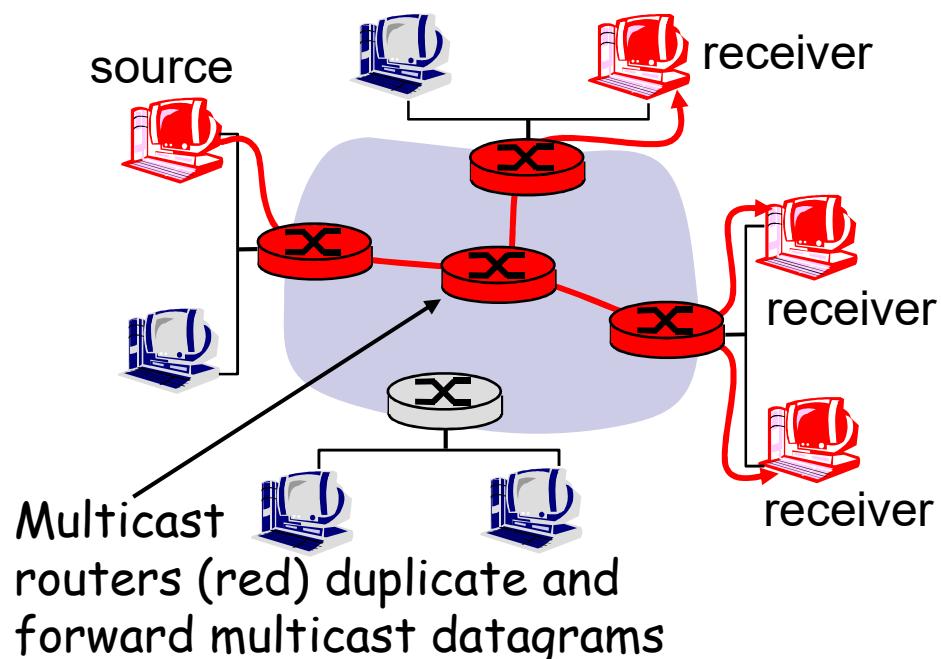
1-1

1-ALL

1-Subset  
of ALL

# Multicast: one sender to many receivers

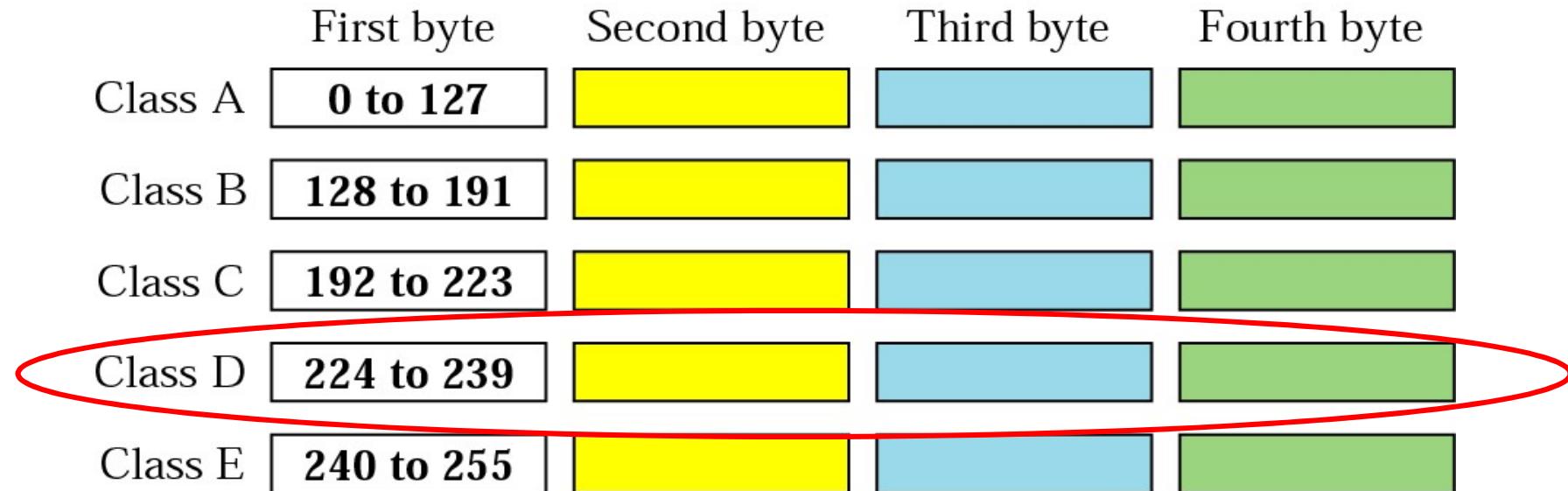
- **Multicast:** act of sending datagram to multiple receivers with **single** “transmit” operation
  - analogy: one teacher to many students
- **Question:** how to achieve multicast?



## Network multicast

- Router actively participate in multicast, making copies of packets **as needed** and forwarding towards multicast receivers

# IP Multicast Address: identify host group



- **Permanent group:** identified by permanent group addresses

- 224.0.0.1** all systems on a LAN
- 224.0.0.2** all routers on a LAN
- 224.0.0.5** all OSPF routers on a LAN
- 224.0.0.6** all designated OSPF routers on a LAN

- **Temporary group:** must be created before use

## IP Multicast Address: identify **host group**

- The Class D address range is used **only** for the **group address or destination address** of IP multicast traffic.
- The source address for multicast datagrams is **always** the unicast source address

# Layer-2 Multicast Address

- Source unicast multicast packets, routers make copies of packets as needed and forward towards multicast receivers.
- **Multicast MAC:** when packets arrive to the **last hop router**, it needs to send the packets to a group of hosts in the LAN by a common destination MAC address.

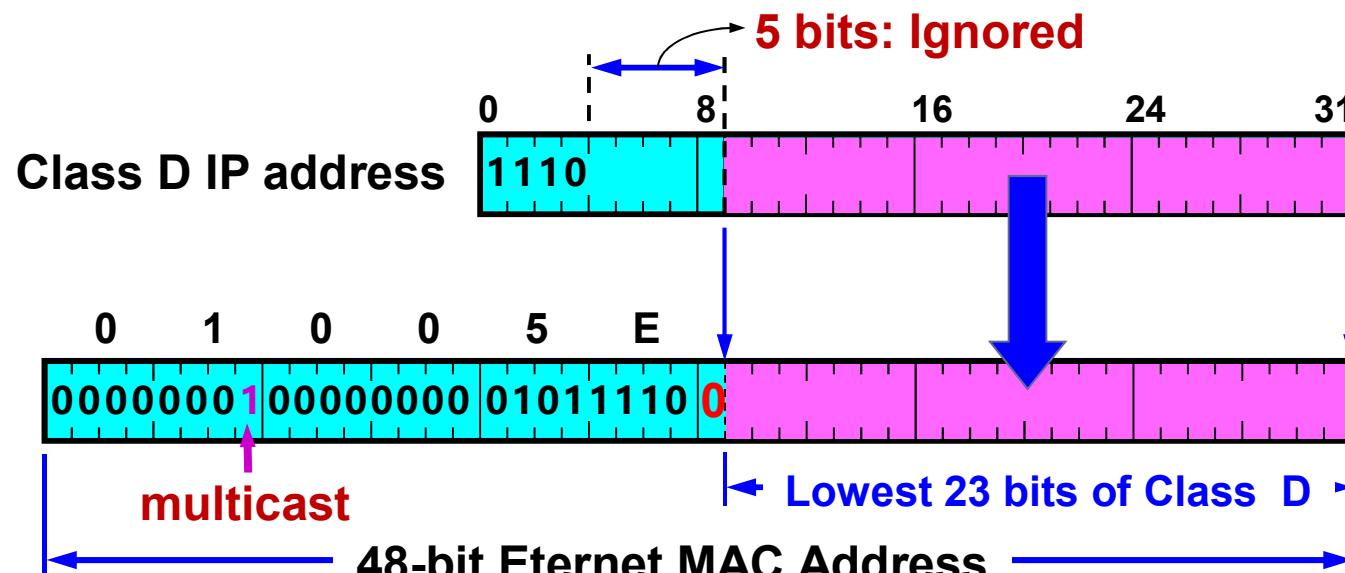
# Layer-2 Multicast Address

- Source unicast multicast packets, routers make copies of packets as needed and forward towards multicast receivers.
- **Multicast MAC:** when packets arrive to the **last hop router**, it needs to send the packets to a group of hosts in the LAN by a common destination MAC address.

# Layer-2 Multicast Address

01-00-5E-00-00-00 to 01-00-5E-7F-FF-FF

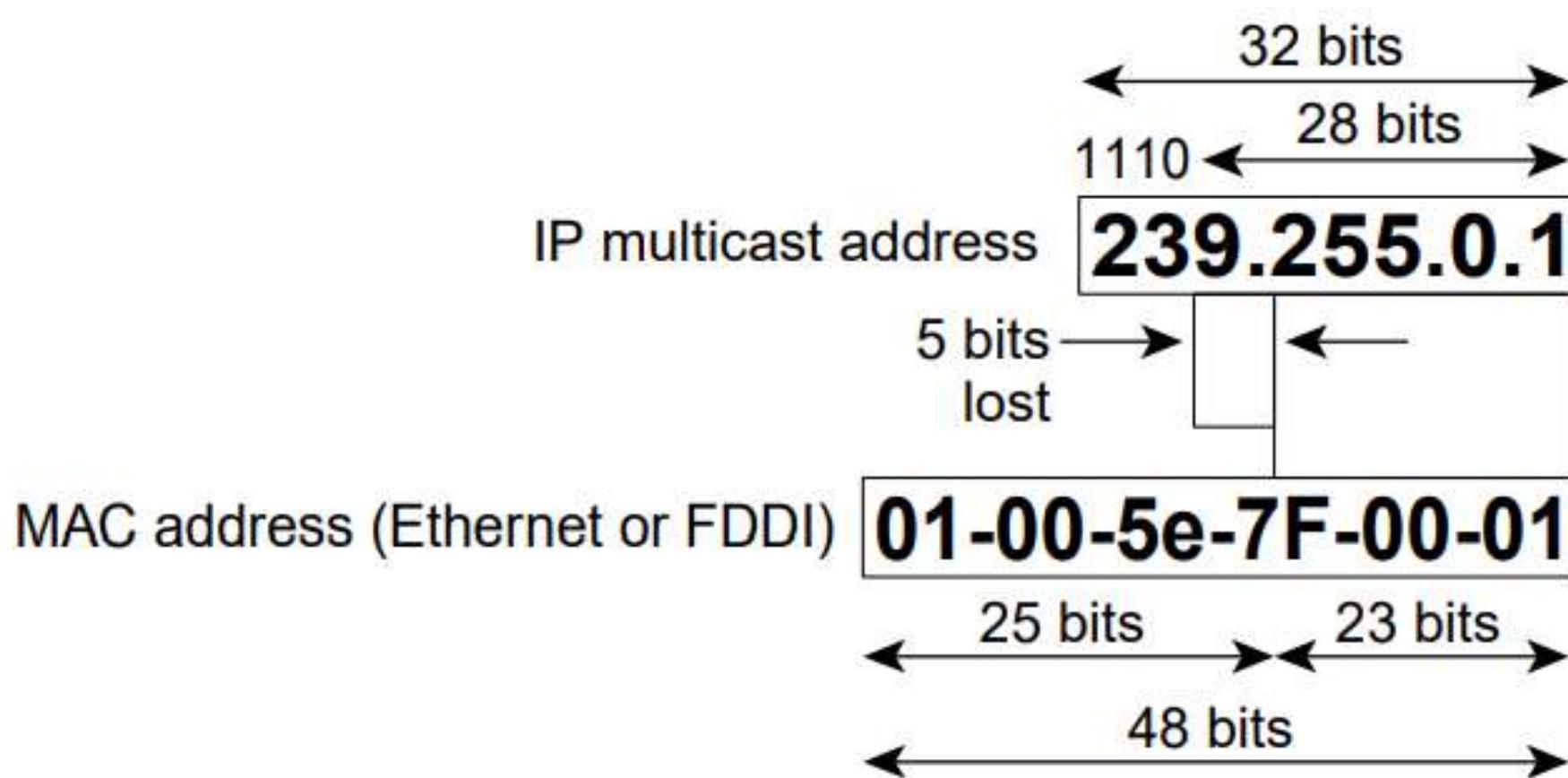
Mapping: Class D to Ethernet Multicast



32:1

最终收到组播数据报的主机在 IP 层依据 IP 组播地址进行过滤，把不是本主机要接收的组播数据丢弃。

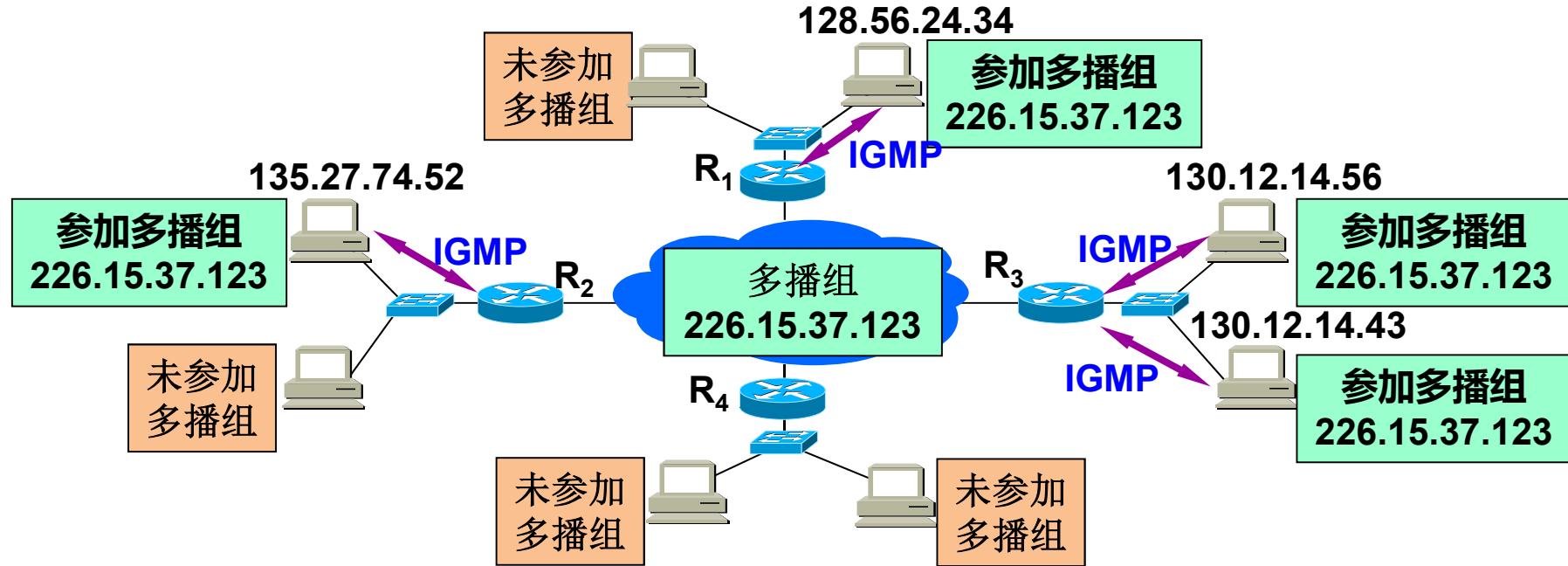
# Layer-2 Multicast Address



# Internet Group Management Protocol (IGMP)

- **Dynamically register individual hosts in a multicast group **on a particular LAN**.**
- **Hosts identify group memberships by sending IGMP messages to their **local** multicast router.**
- **Routers **listen** to IGMP messages and periodically **send out queries** to discover which groups are active or inactive on a particular subnet.**

# Internet Group Management Protocol (IGMP)



IGMP 协议是让连接在本地局域网上的多播路由器知道本局域网上是否有主机参加了或退出了某个组。IGMP 不知道 IP 多播组包含的成员数，也不知道这些成员都分布在哪些网络上。

# The two basic type messages

## ■ Membership Report

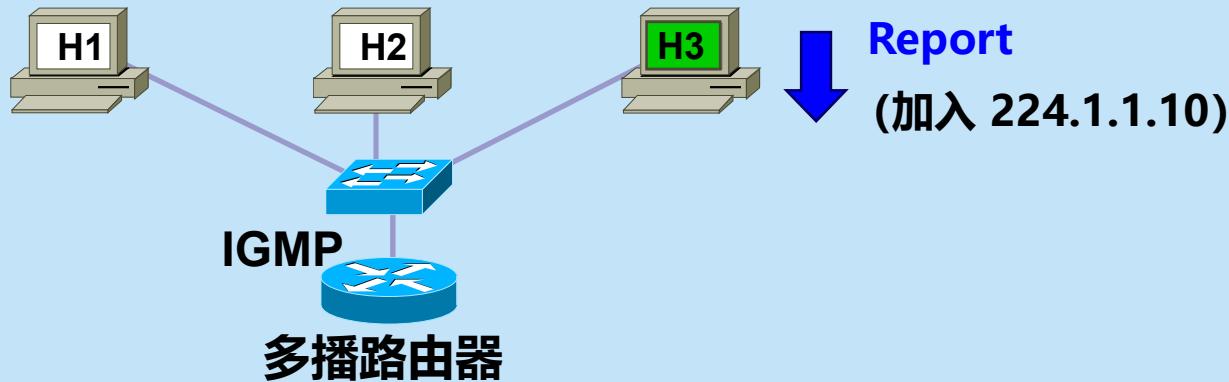
□ Hosts send out IGMP membership reports corresponding to a particular multicast group to indicate that they are interested in joining that group.

## ■ Membership Query

□ The router periodically sends out an IGMP membership query to verify that at least one host on the subnet is still interested in receiving traffic directed to that group.

# Membership Report

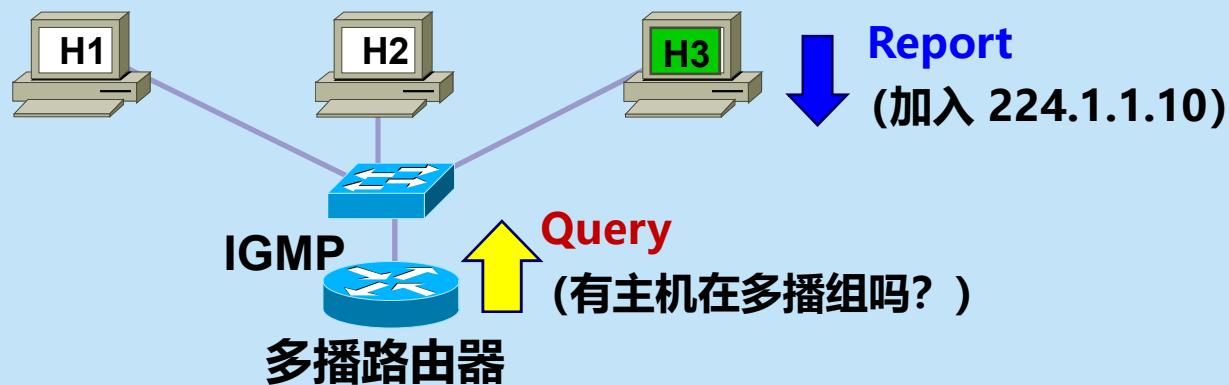
## Joining a group



1. 当某个主机加入多播组时，该主机向多播组的多播地址发送 IGMP Report 报文，声明自己要成为该组的成员。
2. 本地的多播路由器收到 IGMP 报文后，将组成员关系转发给互联网上的其他多播路由器。

# Membership Query

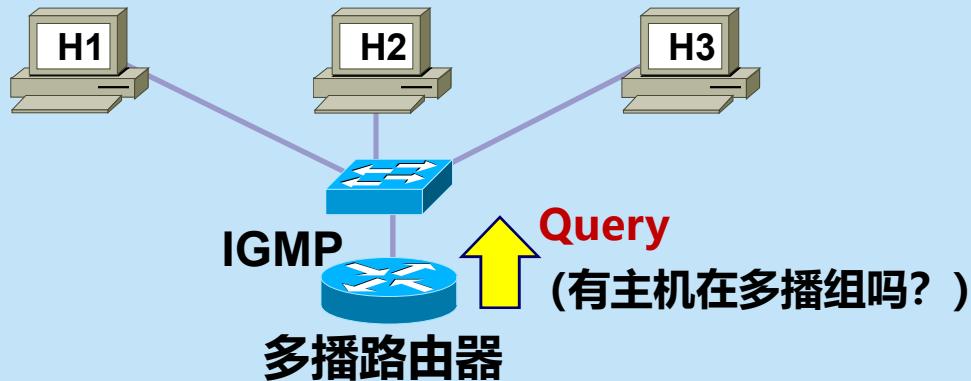
## Maintaining a Group



1. 本地多播路由器周期性地发送IGMP Query 探询本地局域网上的主机，以便知道这些主机是否还继续是组的成员。
2. 只要对某个组有一个主机响应，那么多播路由器就认为这个组有成员。

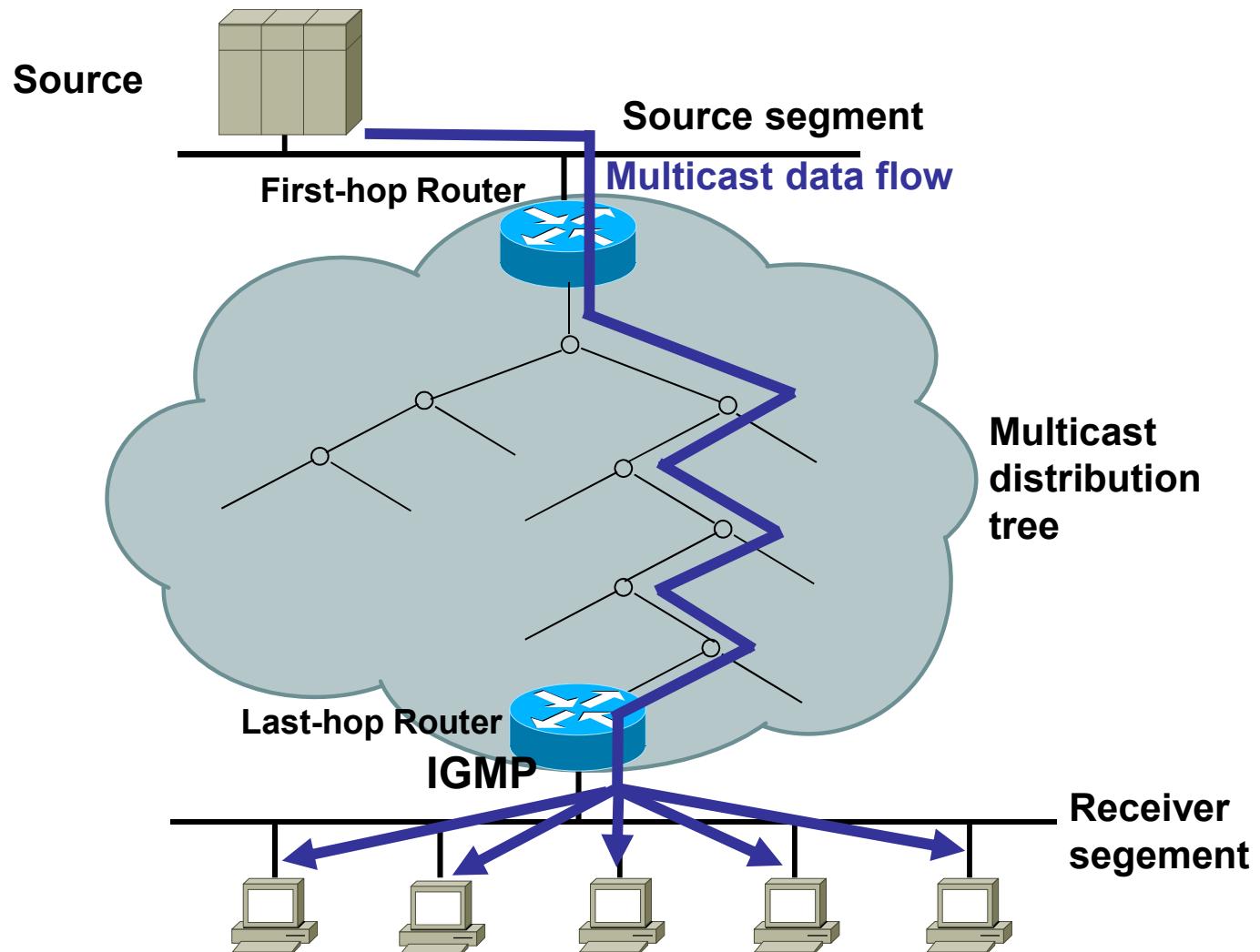
# Membership Query

## Maintaining a Group



1. 本地多播路由器周期性地发送IGMP Query 探询本地局域网上的主机，以便知道这些主机是否还继续是组的成员。
2. 只要对某个组有一个主机响应，那么多播路由器就认为这个组有成员。

# Multicast Routing (1)



# Multicast Routing (2)

## ■ Multicast distribution tree:

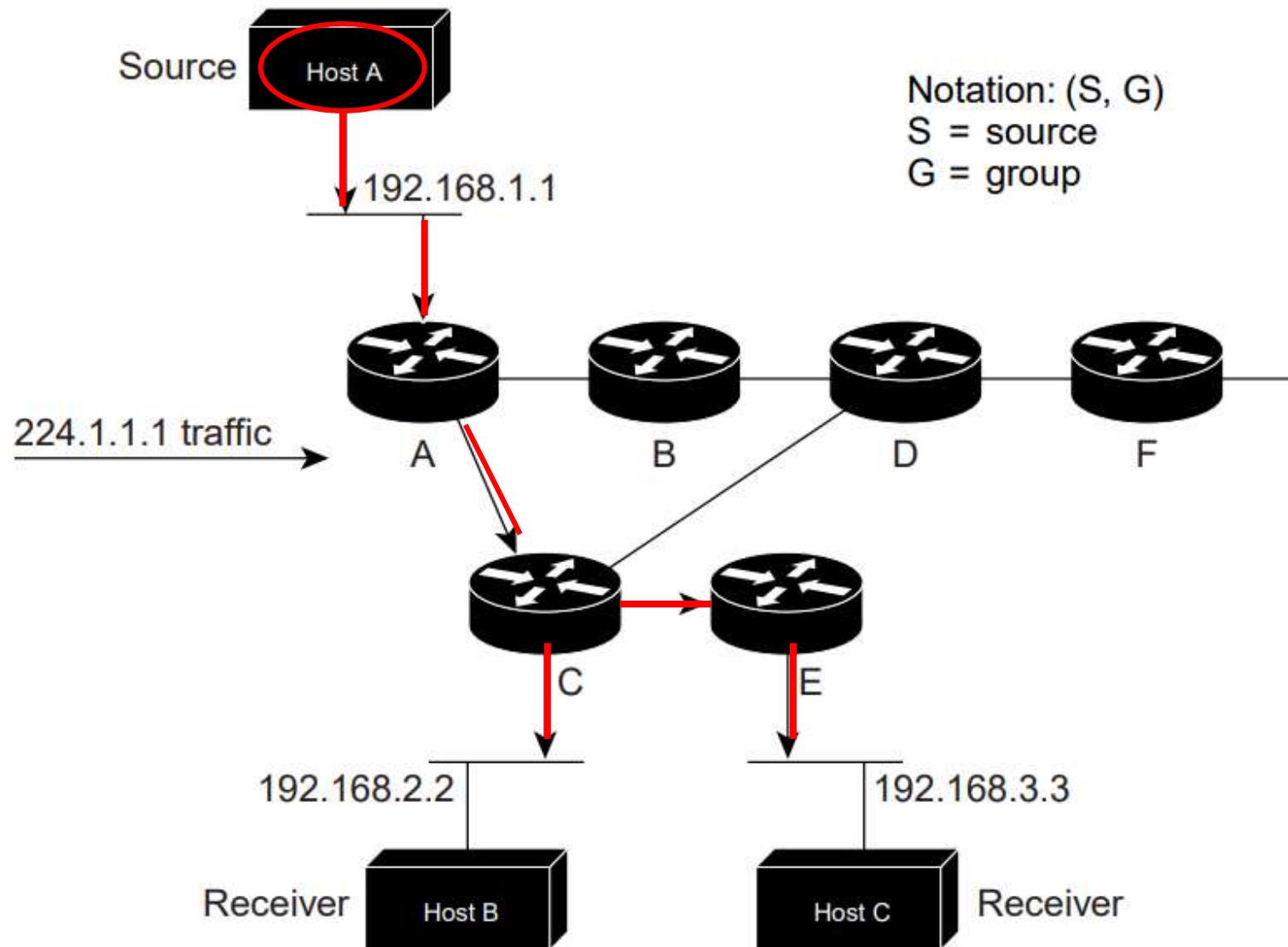
### □ Source tree

- Entry in multicast routing table: (S,G)

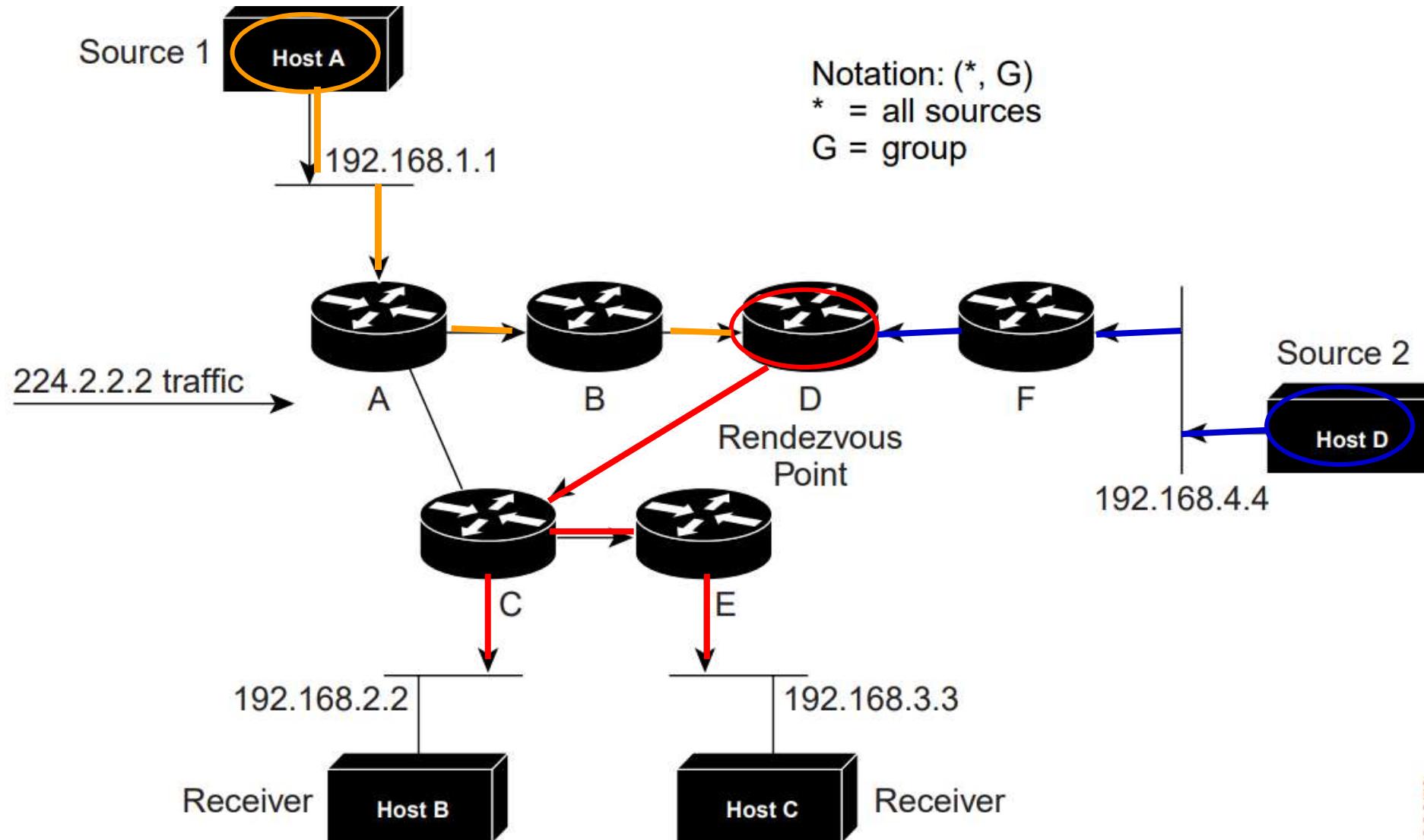
### □ Shared tree

- Entry in multicast routing table: (\*,G)

# Source Tree (SPT)



# Shared Tree (RPT)



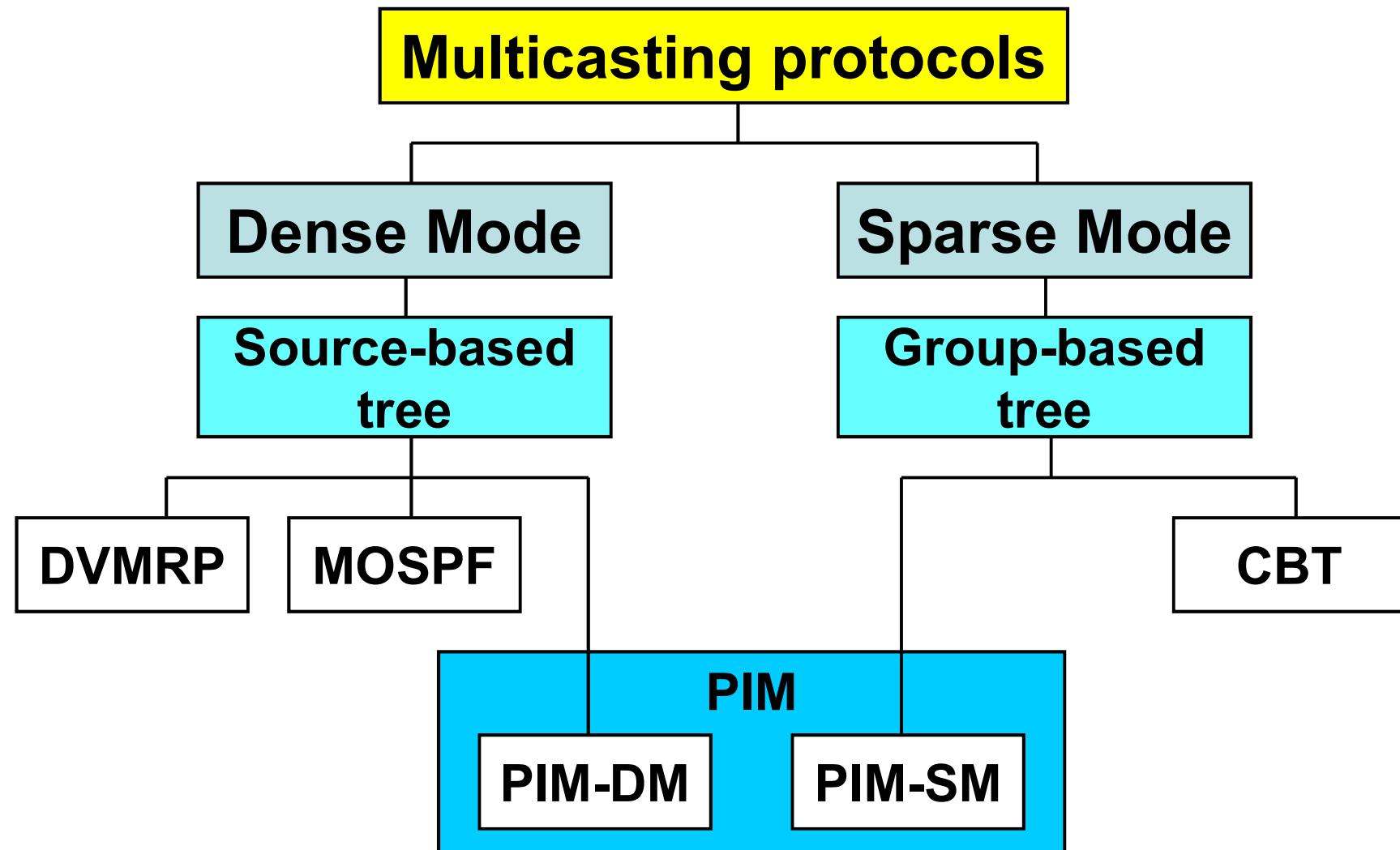
# Multicast Routing Table

Source IP address	Multicast group	Incoming interface (RPF interface)	Outgoing interface list
192.168.1.1	224.2.2.2	I1	I2, I3
*	224.4.4.4	I2	I1, I3

**Source tree:** (Source, Group) entry.

**Shared tree:** (\*, Group) entry.

# Multicast Routing Protocols



# Chapter 5: Roadmap

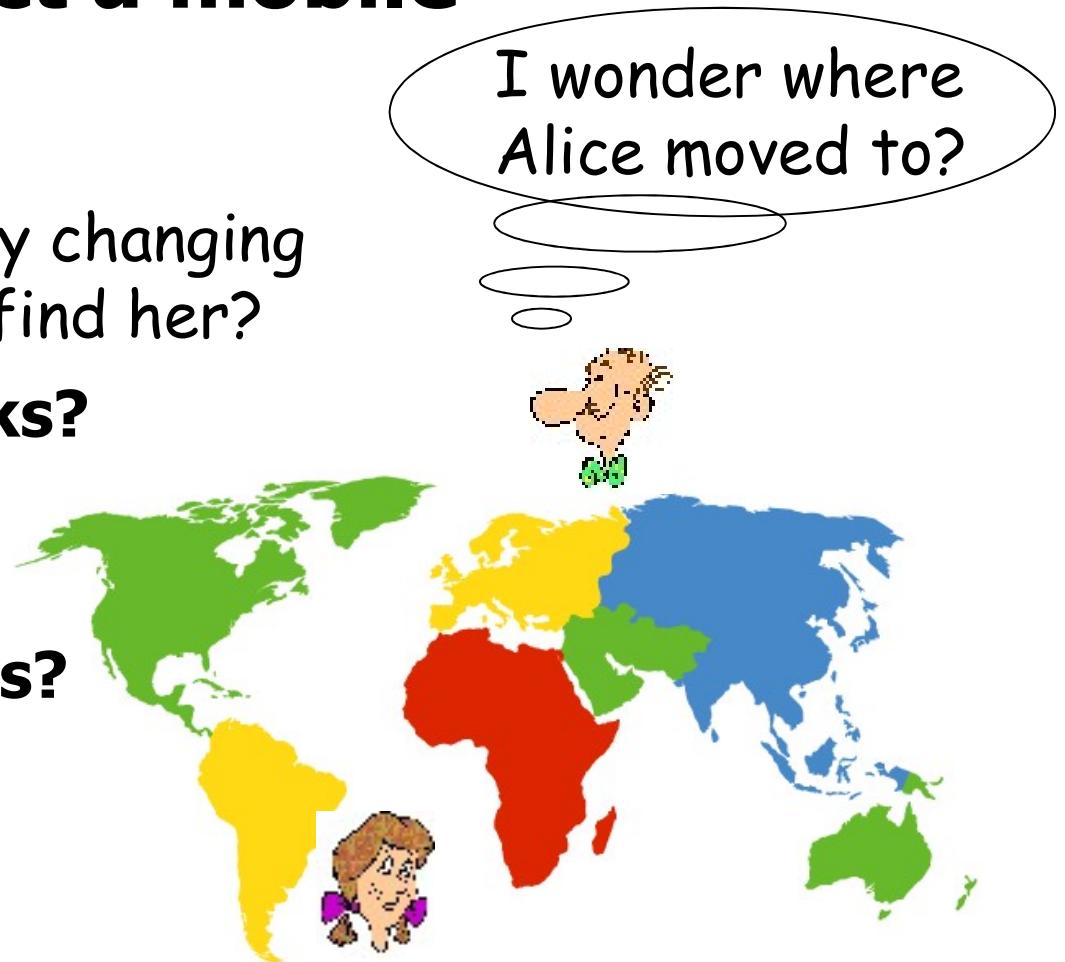
- Network Layer and Design Issues
- Routing Algorithms
- Congestion Control Algorithm
- IP
- IP Multicast
- Mobile IP

# Mobility

How do ***you*** contact a mobile friend?

Consider friend frequently changing addresses, how do you find her?

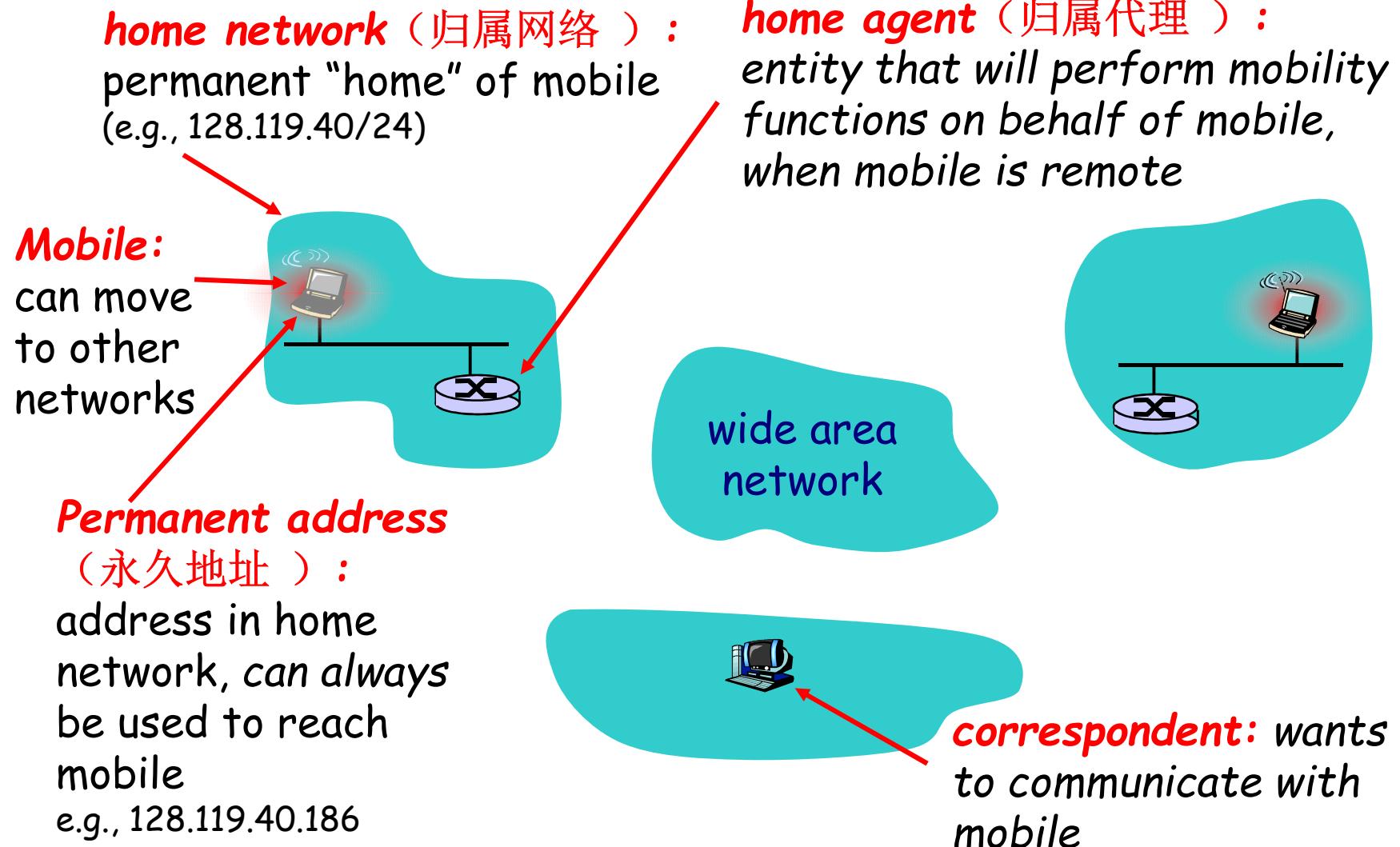
- **search all phone books?**
- **call her parents?**
- **expect her to let you know where he/she is?**



# Mobility

- mobile node moves from network to network
- correspondents want to send packets to mobile node
- Two approaches:
  - *indirect routing*: communication from correspondent to mobile goes **through home agent**, then forwarded to remote
  - *direct routing*: correspondent gets **foreign address** of mobile, sends **directly** to mobile

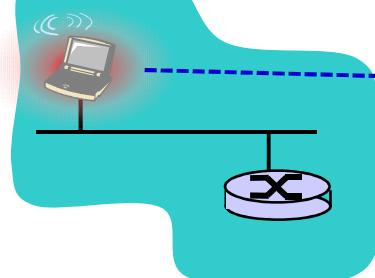
# Mobility: Vocabulary



# Mobility: more vocabulary

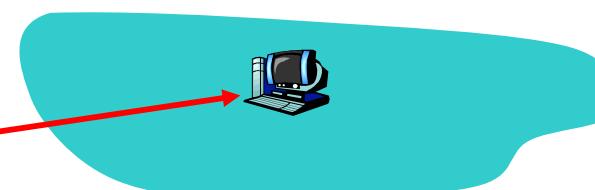
**Permanent address:** remains constant (e.g., 128.119.40.186)

**Care-of-address** (转交地址) : address in visited network. (e.g., 79.129.13.2)

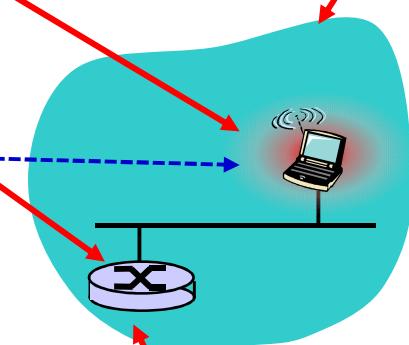


wide area network

**correspondent:** wants to communicate with mobile

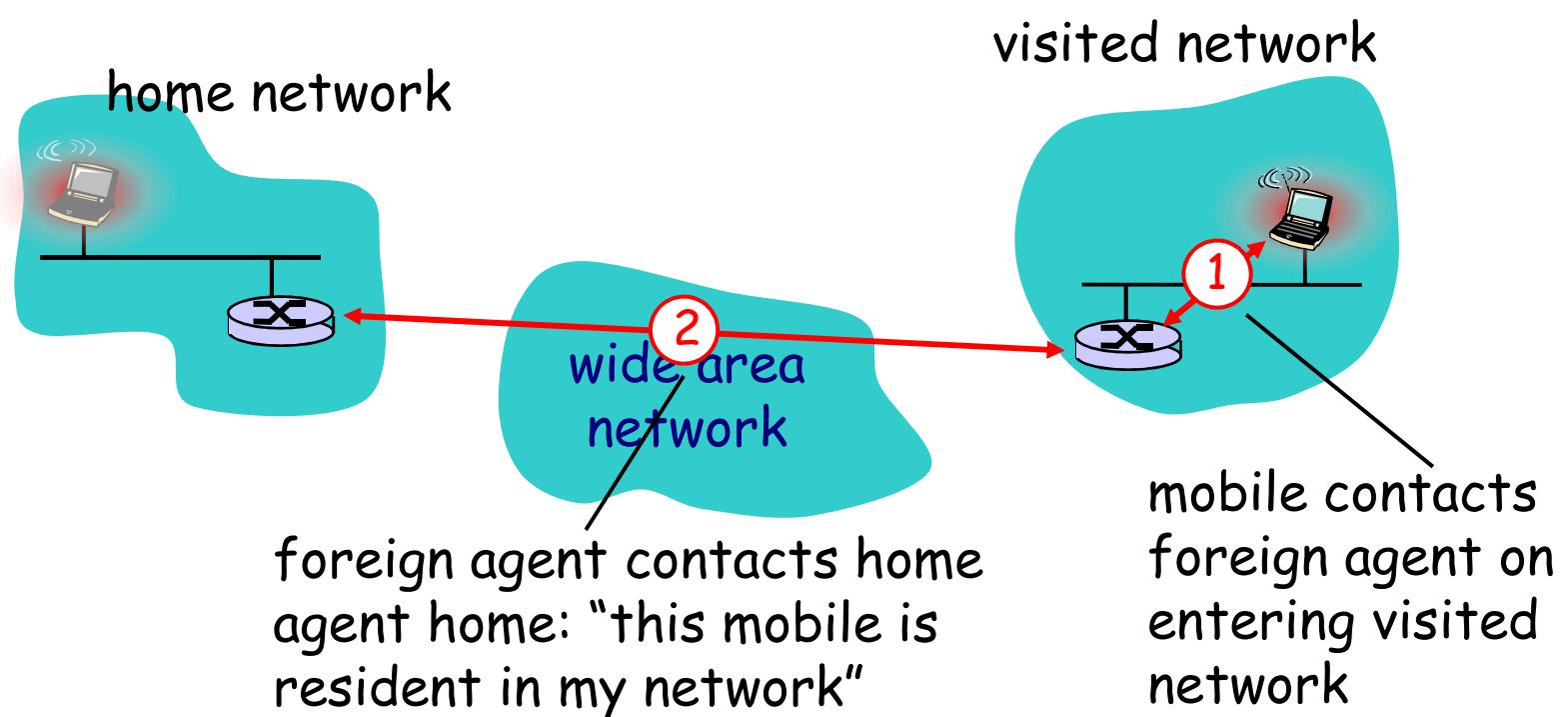


**visited network** (被访网络) : network in which mobile currently resides (e.g., 79.129.13/24)



**foreign agent** (外地代理) : entity in visited network that performs mobility functions on behalf of mobile.

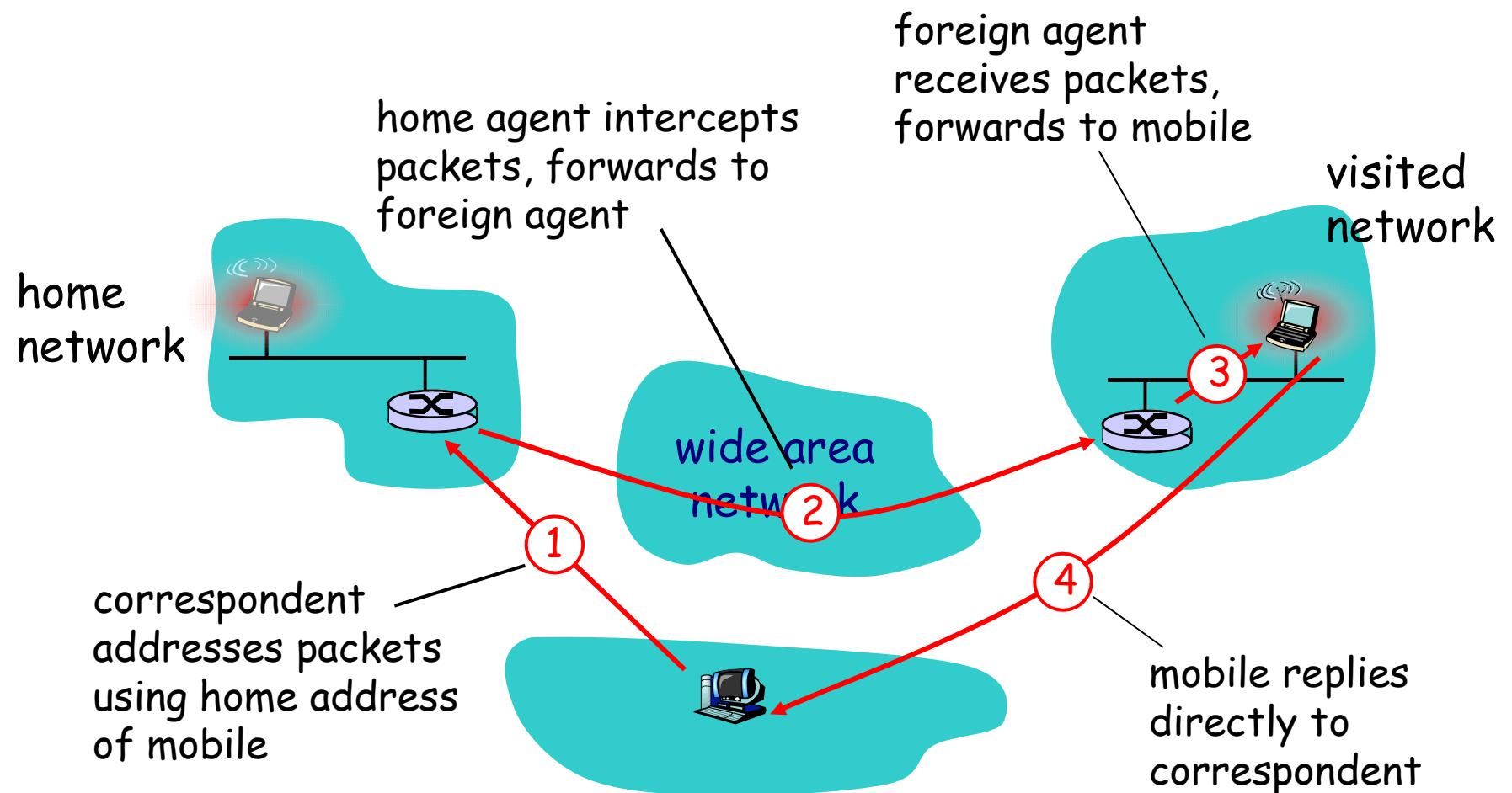
# Mobility: registration



## Result:

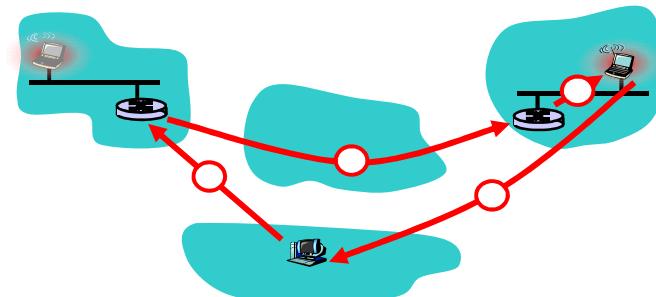
- **Foreign agent knows about mobile**
- **Home agent knows location of mobile**

# Mobility via Indirect Routing

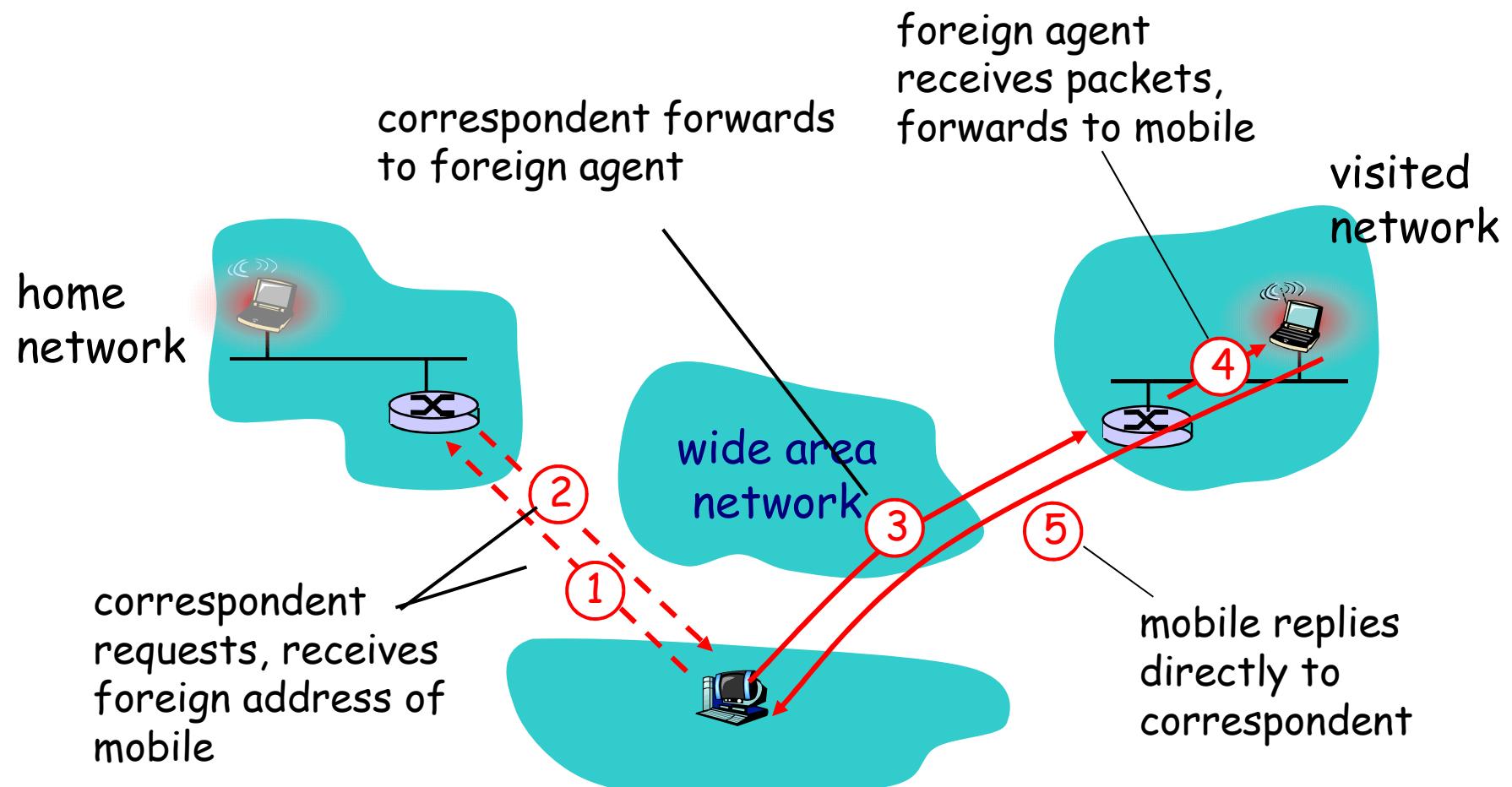


# Indirect Routing: comments

- Mobile uses **two addresses**:
  - **permanent address**: used by correspondent (hence mobile location is *transparent* to correspondent)
  - **care-of-address**: used by home agent to forward datagrams to mobile
- Foreign agent functions may be done by mobile itself
- **Triangle Routing**: correspondent-home-network-mobile
  - **Problems**: inefficient when correspondent, mobile are in same network



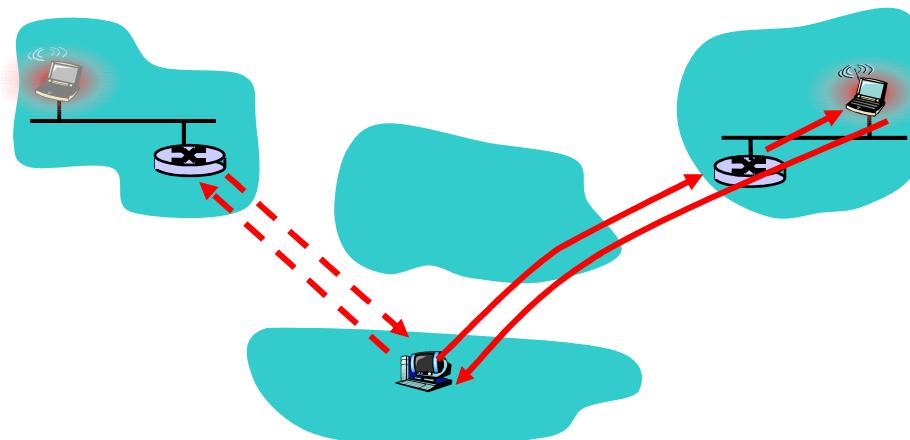
# Mobility via Direct Routing



让通信者创建一个**通信者代理** (correspondent agent), 让通信者代理向**归属代理**询问到移动站在被访网络的**转交地址**。然后由通信者代理把数据报用**隧道技术**发送到被访网络的**外地代理**, 最后再由这个**外地代理**拆封, 把数据报转发给移动站。

# Mobility via Direct Routing: comments

- Overcome triangle routing problem
- Non-transparent to correspondent:  
correspondent must get care-of-address  
from home agent
  - What happens if mobile changes networks?



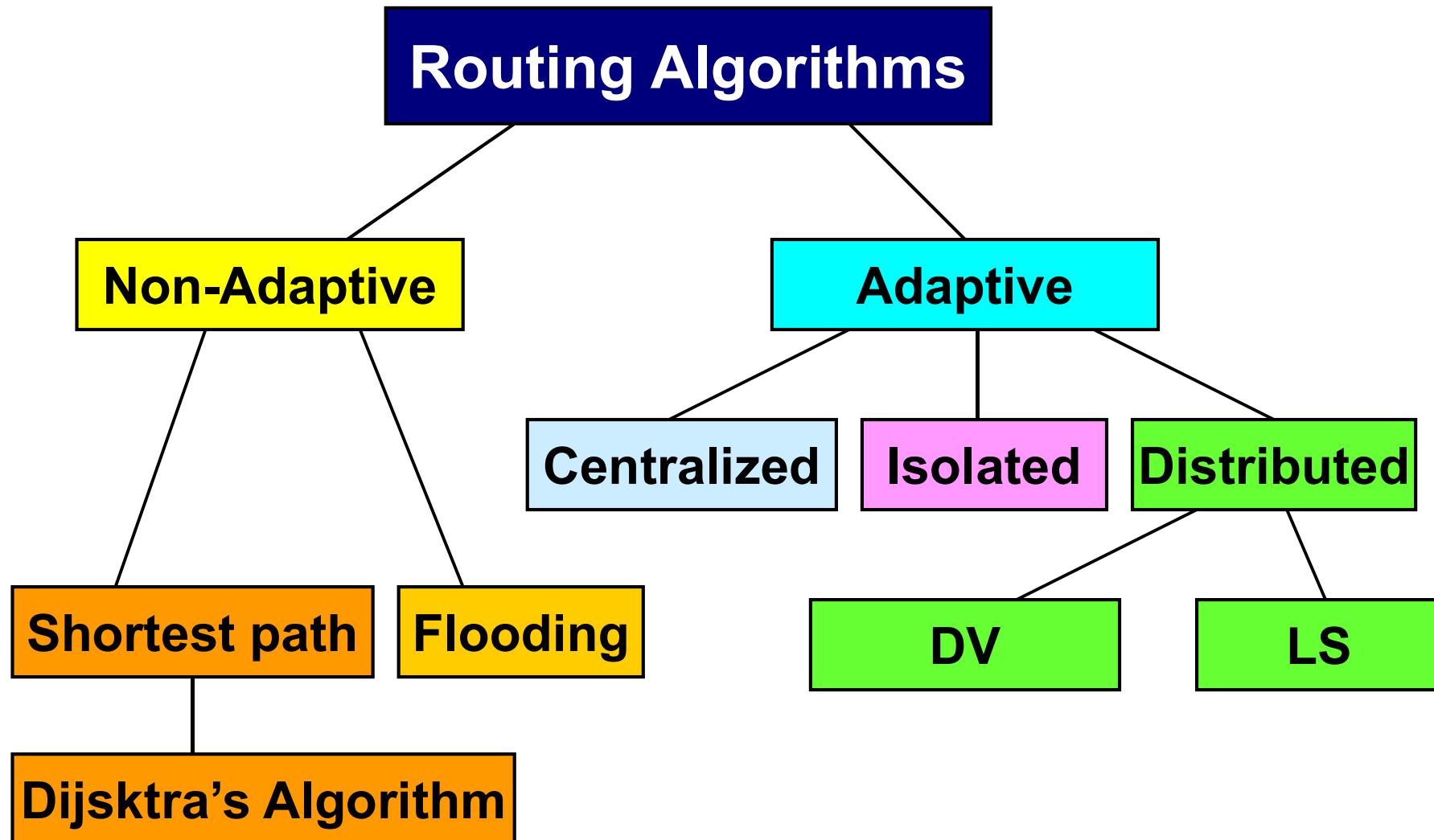
# Mobile IP

- RFC 3220
- Many features we've seen:
  - home agents, foreign agents, foreign-agent registration, care-of-addresses, encapsulation (packet-within-a-packet)
- Three components to standard:
  - agent discovery
  - registration with home agent
  - indirect routing of datagrams

# Summary

- Network layer: sending host (source) to receiving host (destination) communication support.
- Services:
  - Virtual circuit : reliable
  - Datagram: best-effort (unreliable)
- Key functions:
  - Forwarding
  - Routing
  - Congestion Control

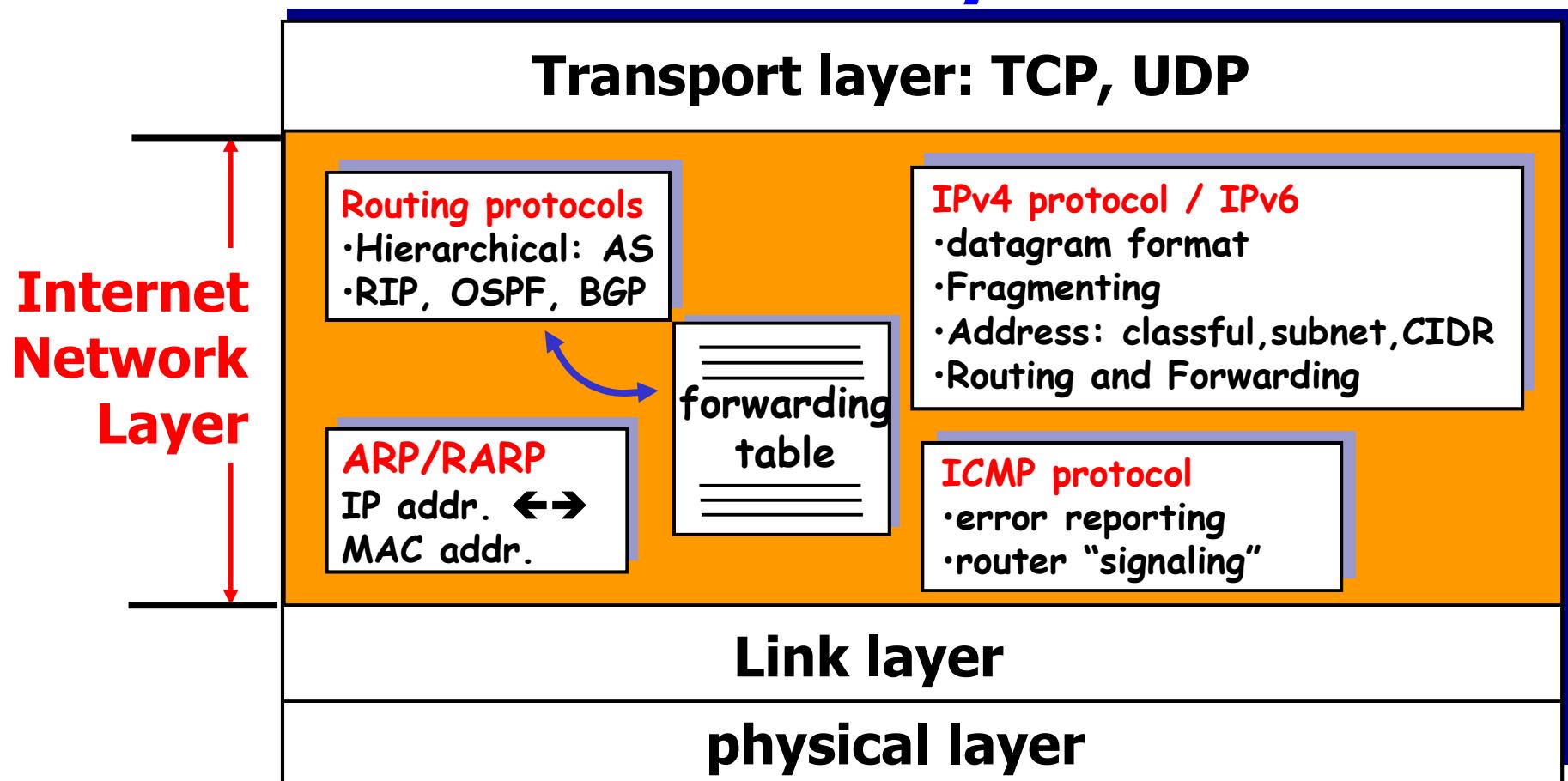
# Summary



# Summary

## The Internet Network Layer

### Host & router network layer functions:



# Summary

- **IP: IPv4, IPv6**

  - **IPv4 Address: 32 bits**

  - **Ipv6 Address: 128 bits**

- **Routing: hierarchical**

  - **Intra-AS : IGP**

    - **RIP: DV, UDP, 16-unreachable**

    - **OSPF: LS, IP**

  - **Inter-AS : EGP**

    - **BGP: Path Vector, TCP, eBGP, iBGP**

# Summary

## ■ IP Multicasting

- Host group.
- Map Class D IP addr. to 48-bit MAC addr.
- IGMP: maintaining group
- Source tree vs. Shared tree

## ■ Mobile IP

- Indirect Routing (triangle routing)
- Direct Routing