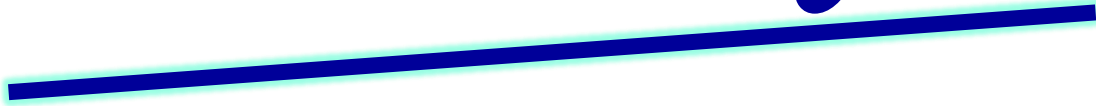


Network Layer

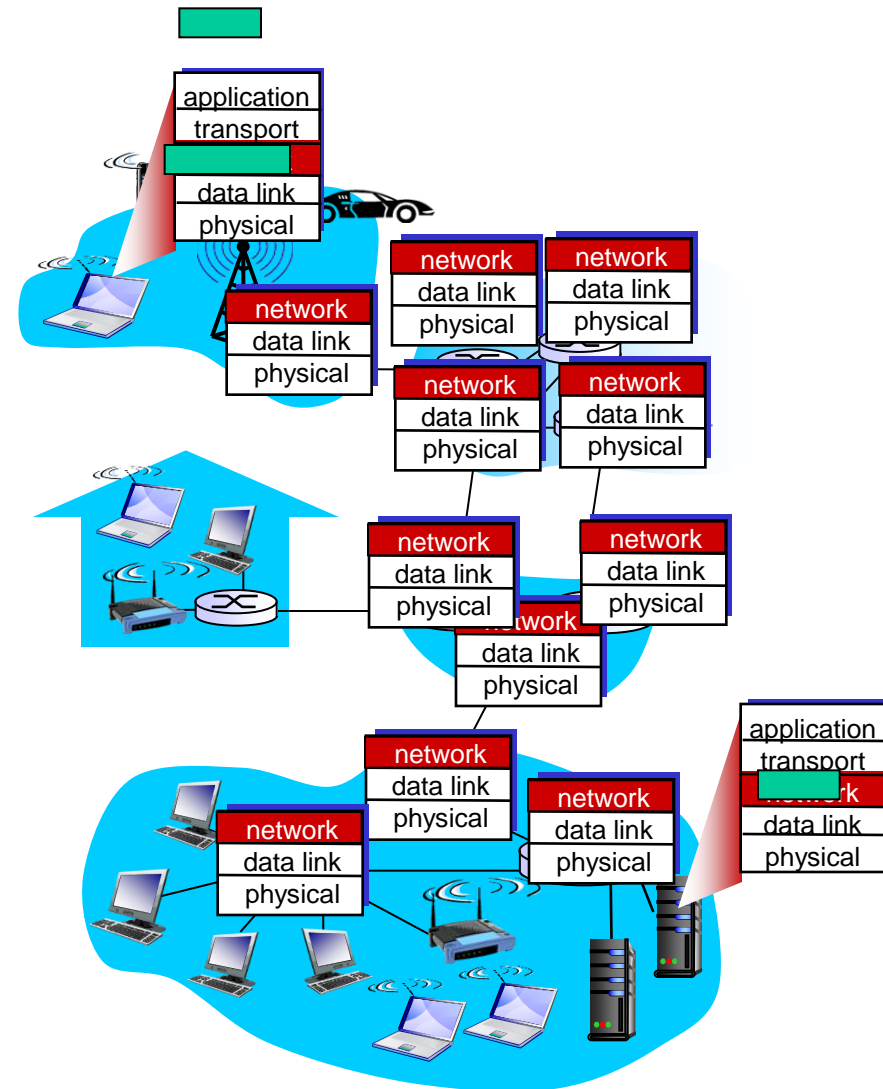


Outline

1. Virtual circuit and Datagram networks
2. IP: Internet Protocol
 - datagram format
 - IPv4 addressing
 - ICMP
 - IPv6
3. routing algorithms
 - link state
 - distance vector
4. routing in the Internet
 - RIP
 - OSPF
 - BGP
5. broadcast and multicast routing

Network layer

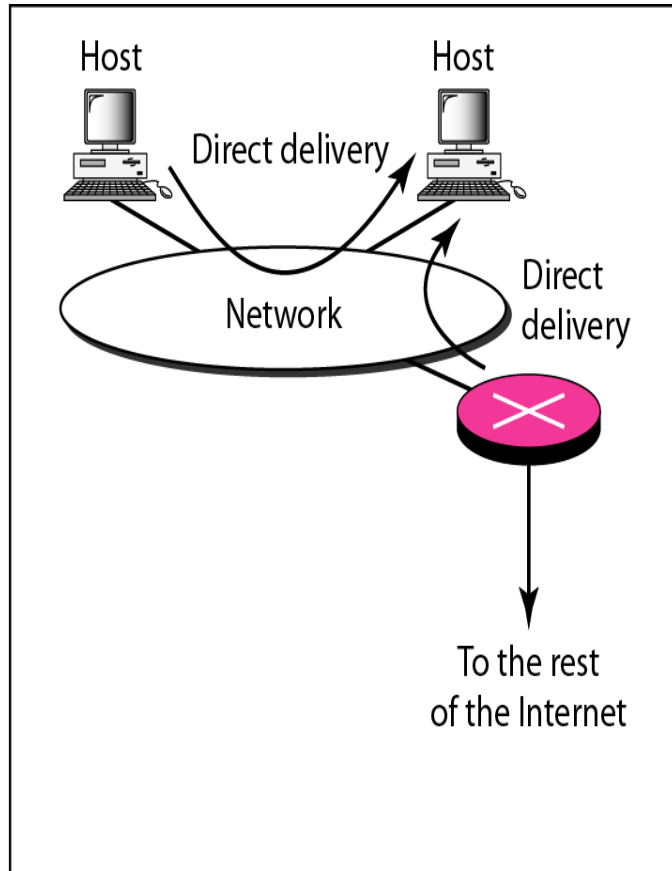
- ❖ transport segment from sending to receiving host
- ❖ on sending side encapsulates segments into datagrams
- ❖ on receiving side, delivers segments to transport layer
- ❖ network layer protocols in *every* host, router
- ❖ router examines header fields in all IP datagrams passing through it



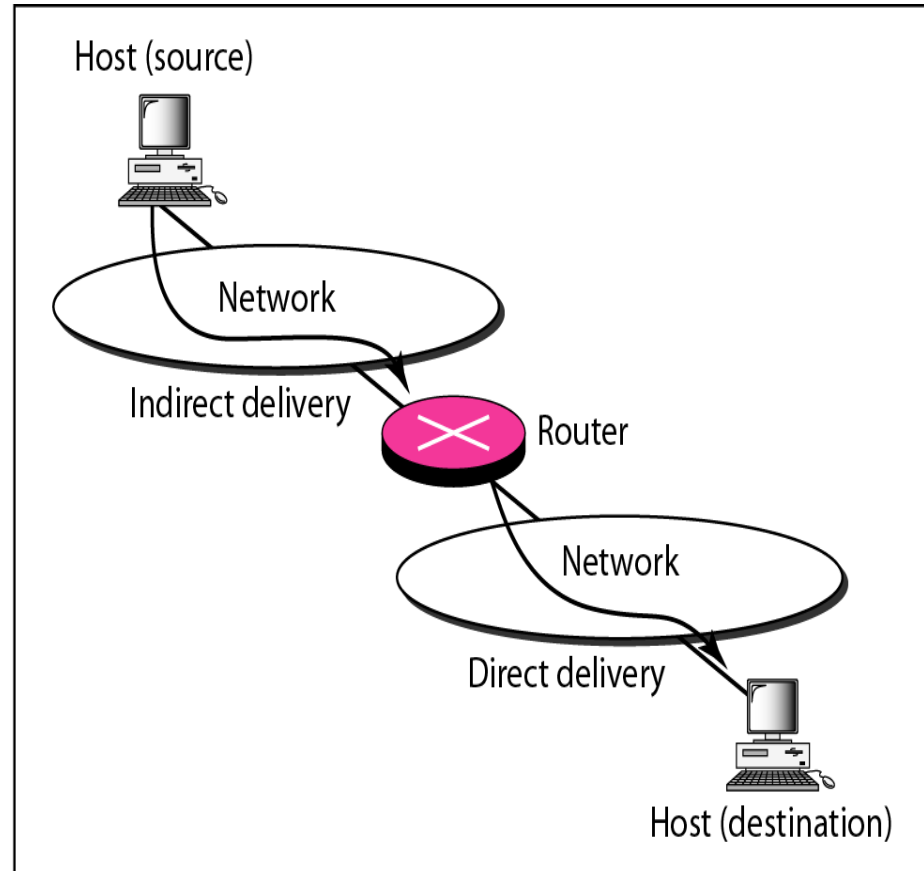
DELIVERY

The network layer supervises the handling of the packets by the underlying physical networks. We define this handling as the delivery of a packet.

Direct and indirect delivery



a. Direct delivery



b. Indirect and direct delivery

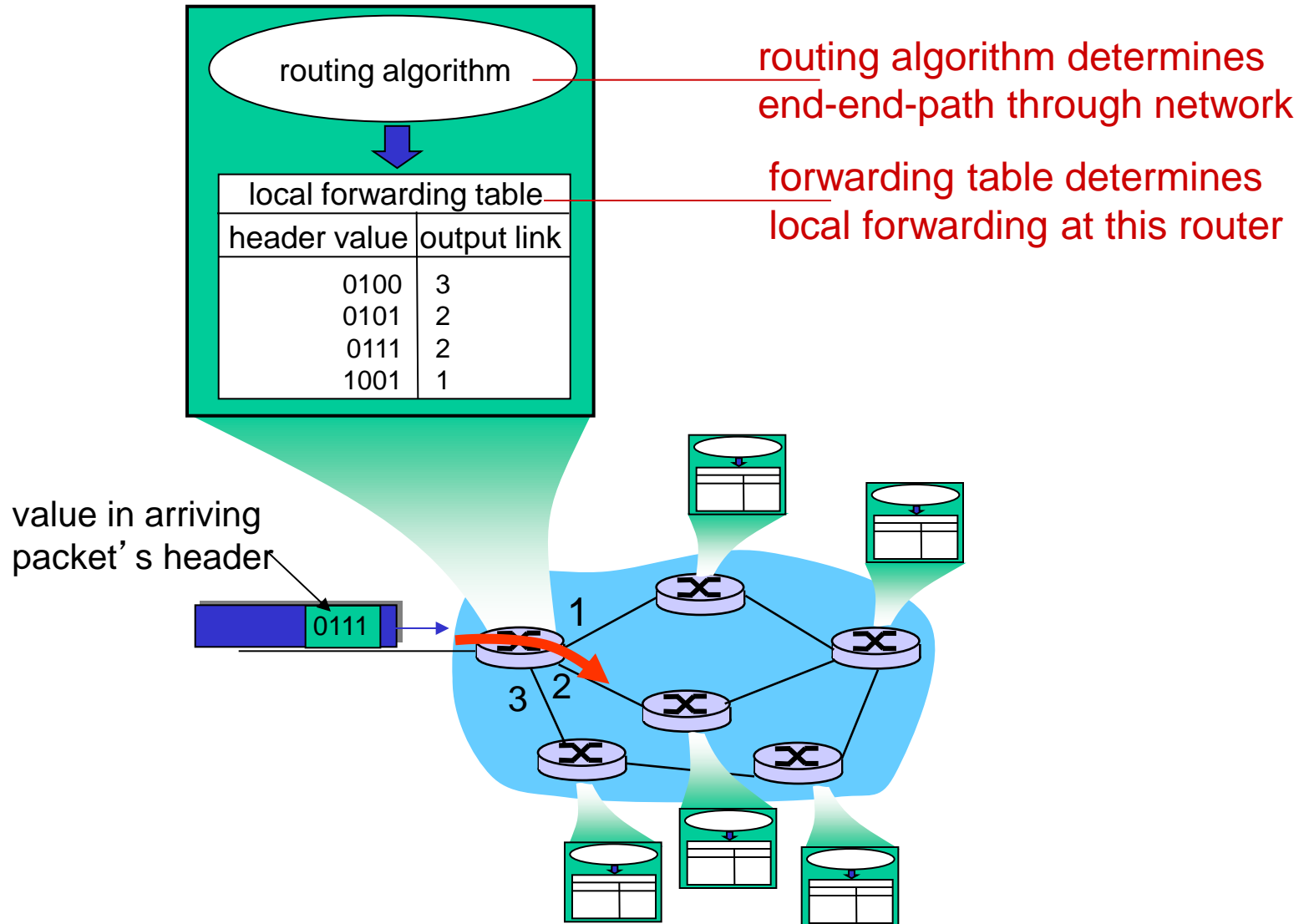
Two key network-layer functions

- ❖ *forwarding*: move packets from router's input to appropriate router output
- ❖ *routing*: determine route taken by packets from source to dest.
 - *routing algorithms*

analogy:

- ❖ *routing*: process of planning trip from source to dest
- ❖ *forwarding*: process of getting through single interchange

Interplay between routing and forwarding



Connection setup

- ❖ 3rd important function in *some* network architectures:
 - ATM, frame relay, X.25
- ❖ before datagrams flow, two end hosts *and* intervening routers establish virtual connection
 - routers get involved
- ❖ network vs transport layer connection service:
 - *network*: between two hosts (may also involve intervening routers in case of VCs)
 - *transport*: between two processes

Connection, connection-less service

- ❖ *datagram* network provides network-layer *connectionless* service
- ❖ *virtual-circuit* network provides network-layer *connection* service
- ❖ analogous to TCP/UDP connection-oriented / connectionless transport-layer services, but:
 - *service*: host-to-host
 - *no choice*: network provides one or the other
 - *implementation*: in network core

Virtual circuits

“source-to-dest path behaves much like telephone circuit”

- performance-wise
- network actions along source-to-dest path

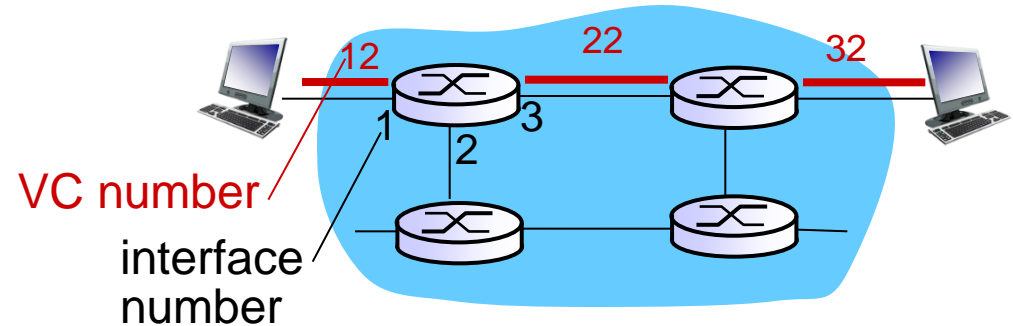
- ❖ call setup, teardown for each call *before* data can flow
- ❖ each packet carries VC identifier (not destination host address)
- ❖ every router on source-dest path maintains “state” for each passing connection
- ❖ link, router resources (bandwidth, buffers) may be *allocated* to VC (dedicated resources = predictable service)

VC implementation

a VC consists of:

1. *path* from source to destination
 2. *VC numbers*, one number for each link along path
 3. *entries in forwarding tables* in routers along path
- ❖ packet belonging to VC carries VC number (rather than dest address)
 - ❖ VC number can be changed on each link.
 - new VC number comes from forwarding table

VC forwarding table



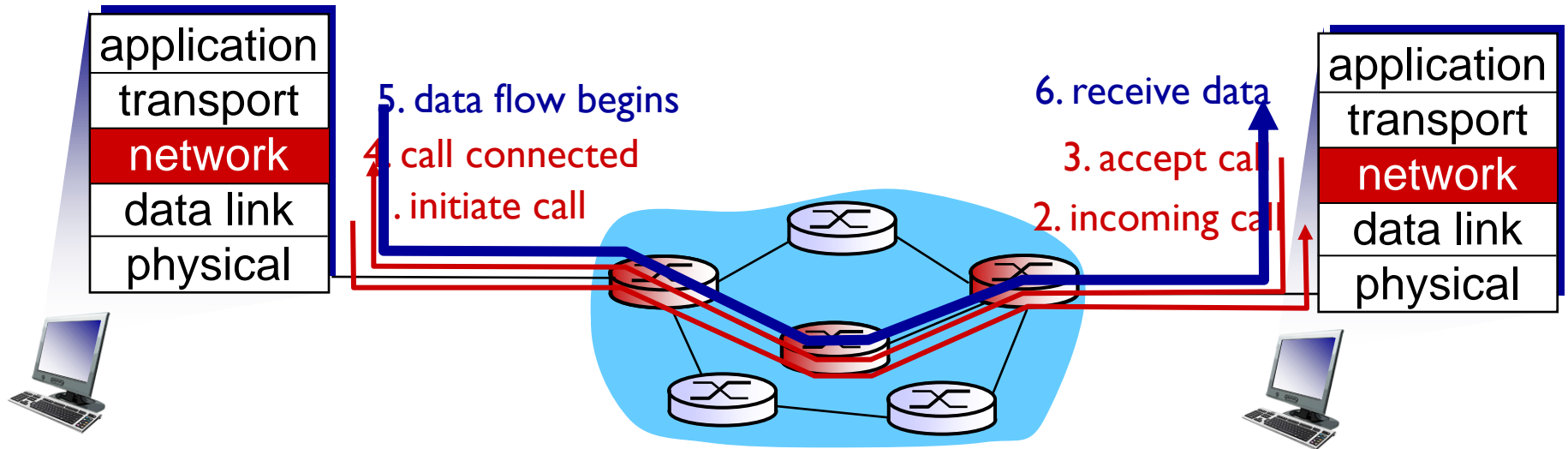
forwarding table in northwest router:

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

VC routers maintain connection state information!

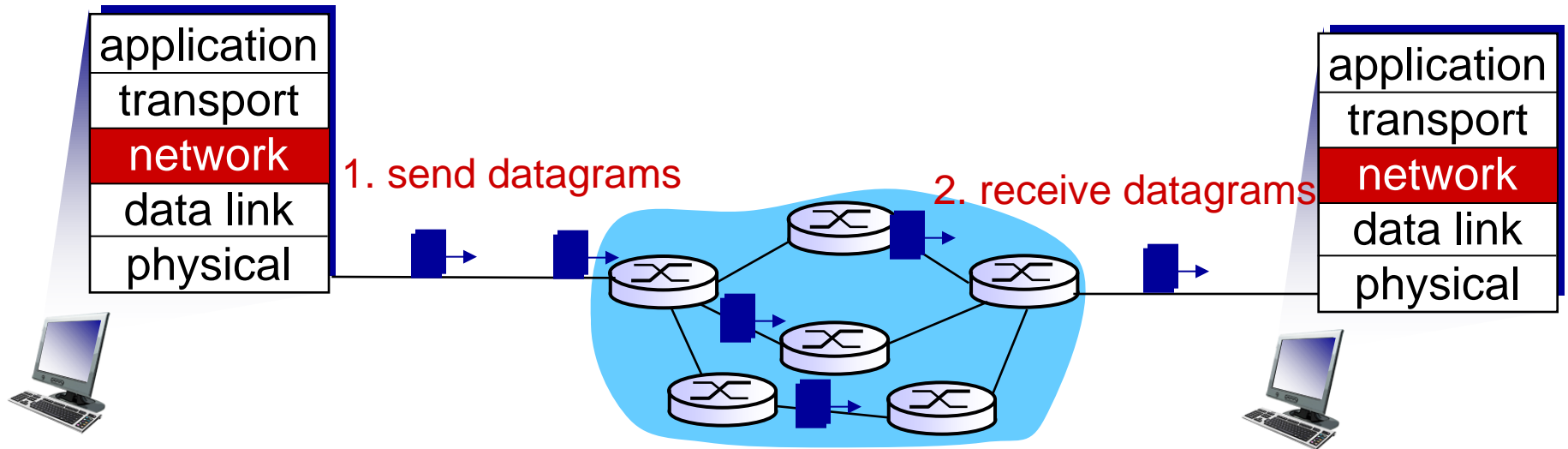
Virtual circuits: signaling protocols

- ❖ used to setup, maintain teardown VC
- ❖ used in ATM, frame-relay, X.25
- ❖ not used in today's Internet

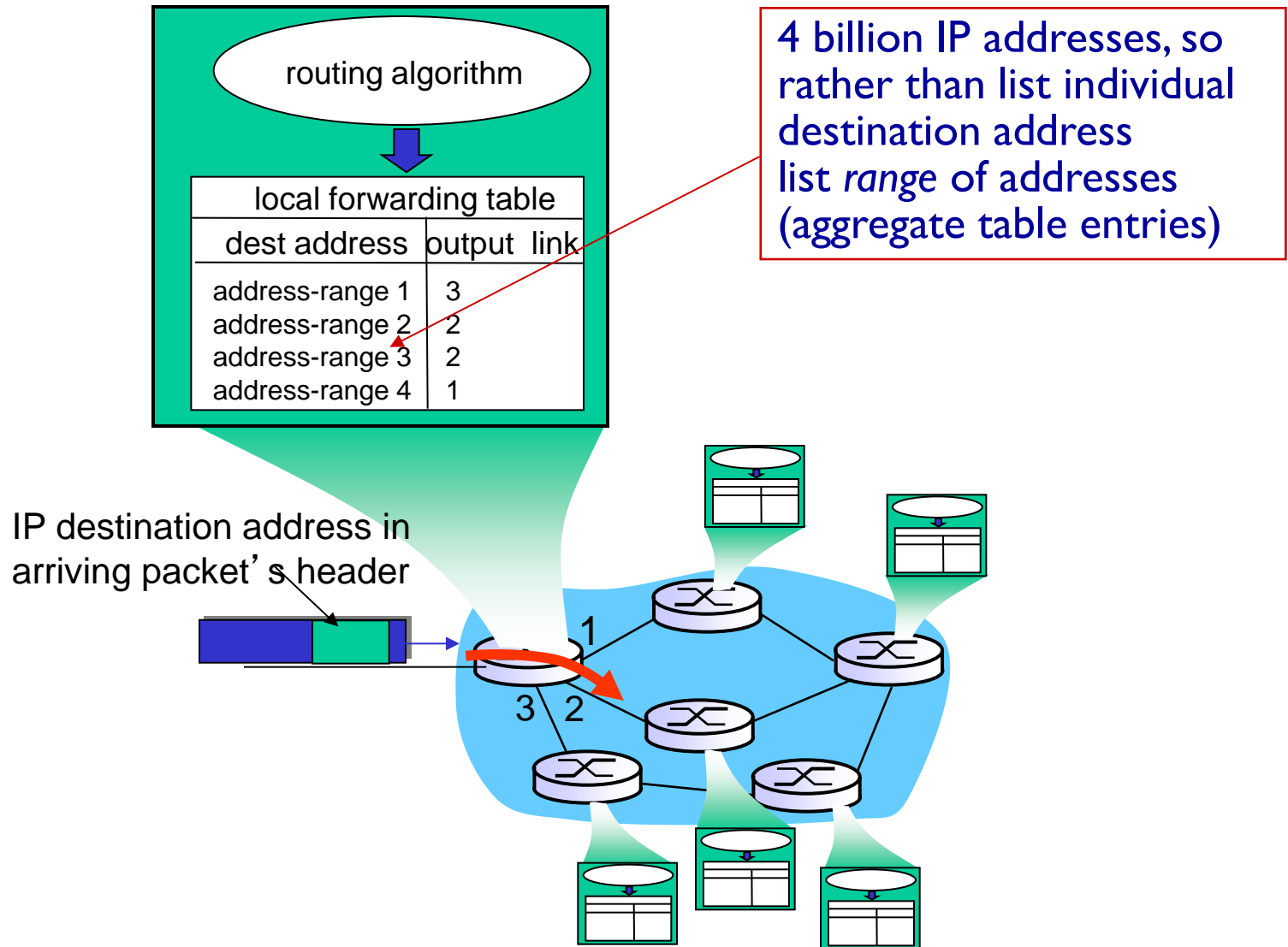


Datagram networks

- ❖ no call setup at network layer
- ❖ routers: no state about end-to-end connections
 - no network-level concept of “connection”
- ❖ packets forwarded using destination host address



Datagram forwarding table



Datagram forwarding table

Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
otherwise	3

Q: but what happens if ranges don't divide up so nicely?

Longest prefix matching

longest prefix matching

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010*** *****	0
11001000 00010111 00011000 *****	1
11001000 00010111 00011*** *****	2
otherwise	3

examples:

DA: 11001000 00010111 00010110 10100001

which interface?

DA: 11001000 00010111 00011000 10101010

which interface?

Datagram or VC network: why?

Internet (datagram)

- ❖ data exchange among computers
 - “elastic” service, no strict timing req.
- ❖ many link types
 - different characteristics
 - uniform service difficult
- ❖ “smart” end systems (computers)
 - can adapt, perform control, error recovery
 - **simple inside network, complexity at “edge”**

ATM (VC)

- ❖ evolved from telephony
- ❖ human conversation:
 - strict timing, reliability requirements
 - need for guaranteed service
- ❖ “dumb” end systems
 - telephones
 - **complexity inside network**

Forwarding Methods

Route method versus next-hop method

a. Routing tables based on route

Destination	Route
Host B	R1, R2, host B

Routing table for host A

Destination	Route
Host B	R2, host B

Routing table for R1

Destination	Route
Host B	Host B

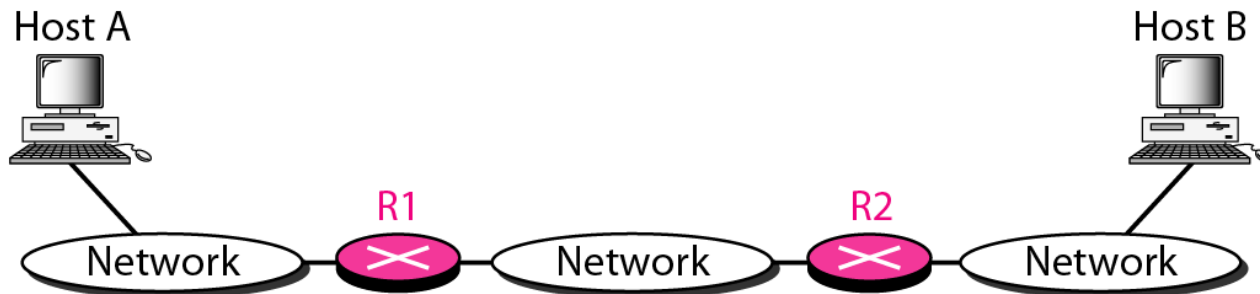
Routing table for R2

b. Routing tables based on next hop

Destination	Next hop
Host B	R1

Destination	Next hop
Host B	R2

Destination	Next hop
Host B	---



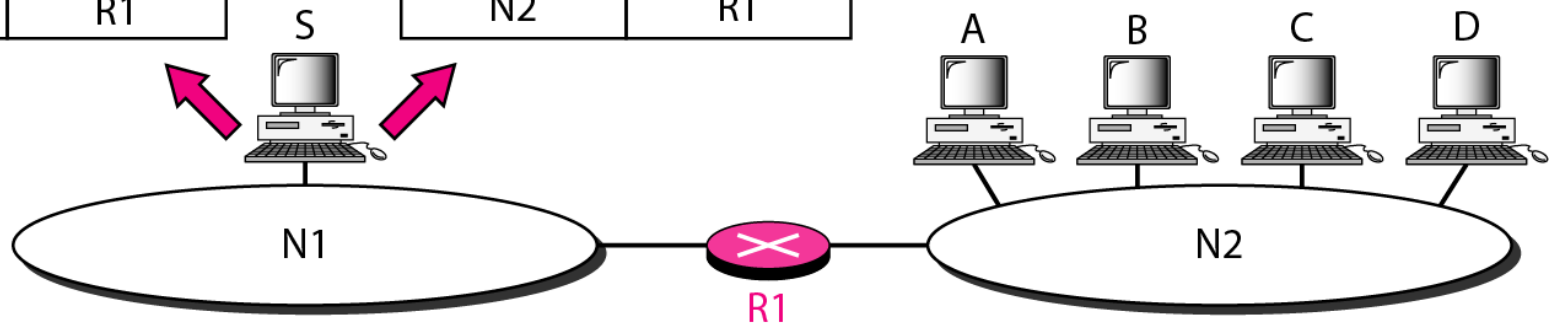
Host-specific versus network-specific method

Routing table for host S based on host-specific method

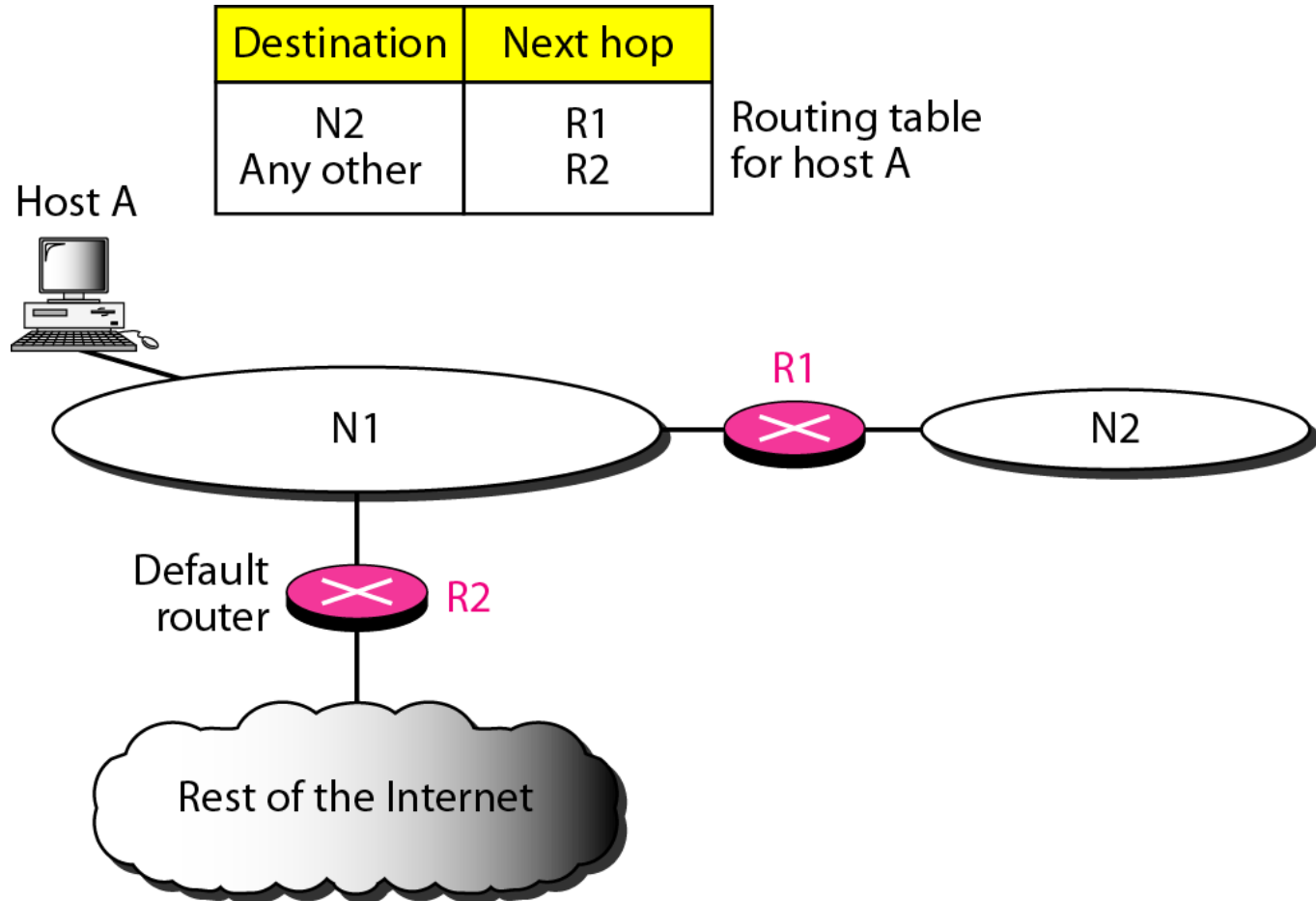
Destination	Next hop
A	R1
B	R1
C	R1
D	R1

Routing table for host S based on network-specific method

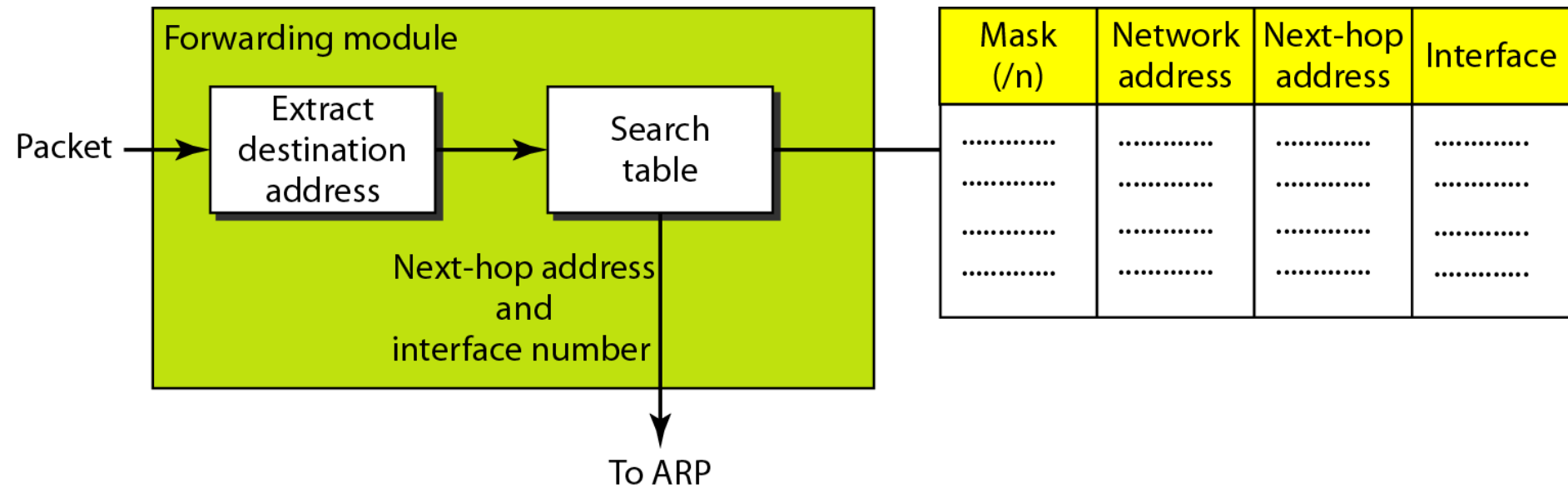
Destination	Next hop
N2	R1



Default method



Simplified forwarding module in classless address





In classless addressing, we need at least four columns in a routing table.

Example 1

Make a routing table for router R1, using the configuration in Figure 1.

Solution

Table 22.1 shows the corresponding table.

Figure 1 Configuration

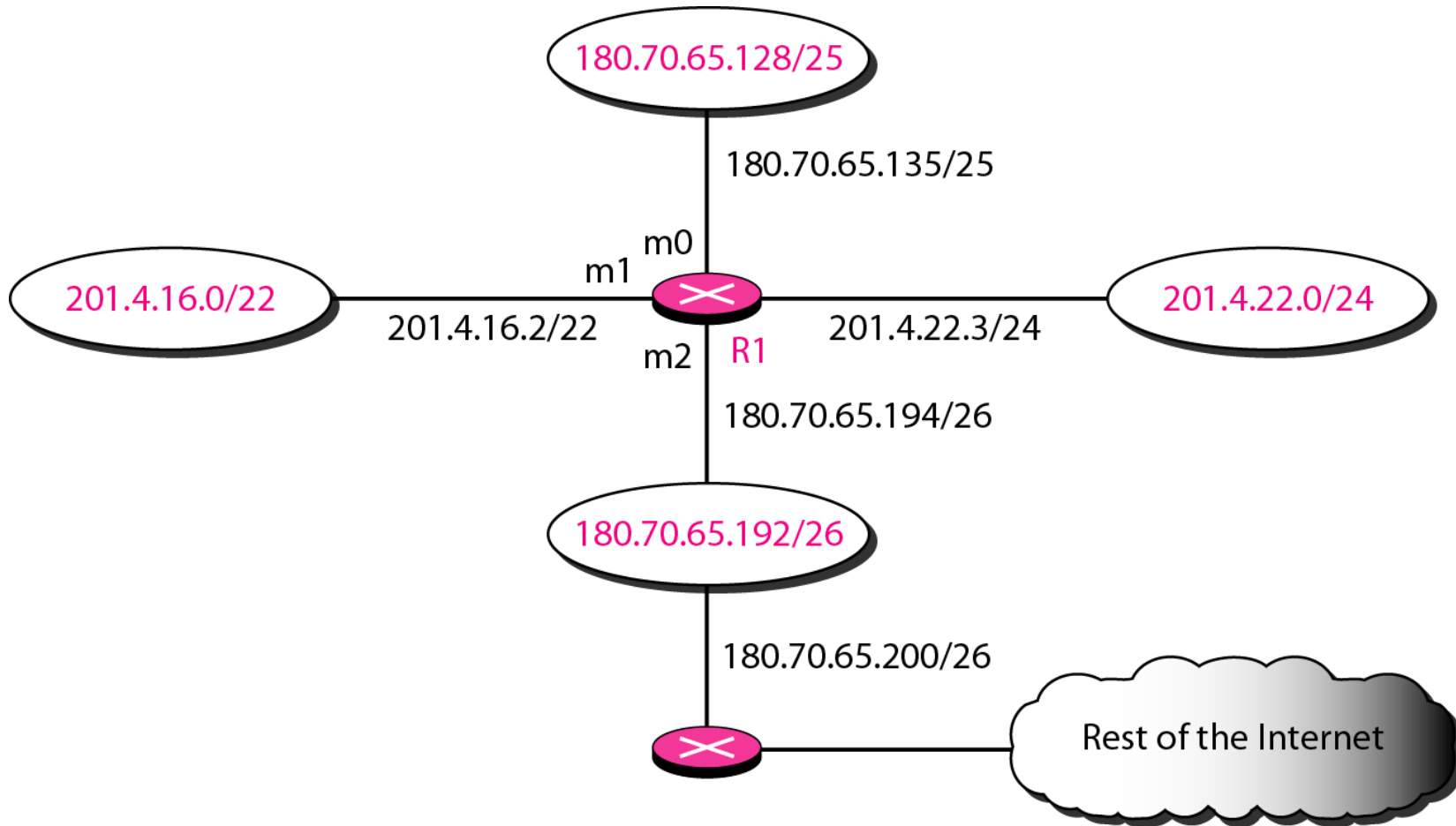


Table .1 *Routing table for router R1 in Figure 1*

<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

Example. 2

Show the forwarding process if a packet arrives at R1 in Figure 1 with the destination address 180.70.65.140.

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 to ARP for further processing.*

Example. 3

Show the forwarding process if a packet arrives at R1 in Figure 1 with the destination address 201.4.22.35.

Solution

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).*

Example. 3 (continued)

- 3. The third mask (/24) is applied to the destination address. The result is 201.4.22.0, which matches the corresponding network address. The destination address of the packet and the interface number m3 are passed to ARP.*

Example .4

Show the forwarding process if a packet arrives at R1 in Figure 1 with the destination address 18.24.32.78.

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 next-hop address 180.70.65.200 and interface number m2 to ARP. This is probably an outgoing package that needs to be sent, via the default router, to someplace else in the Internet.

Routing Table

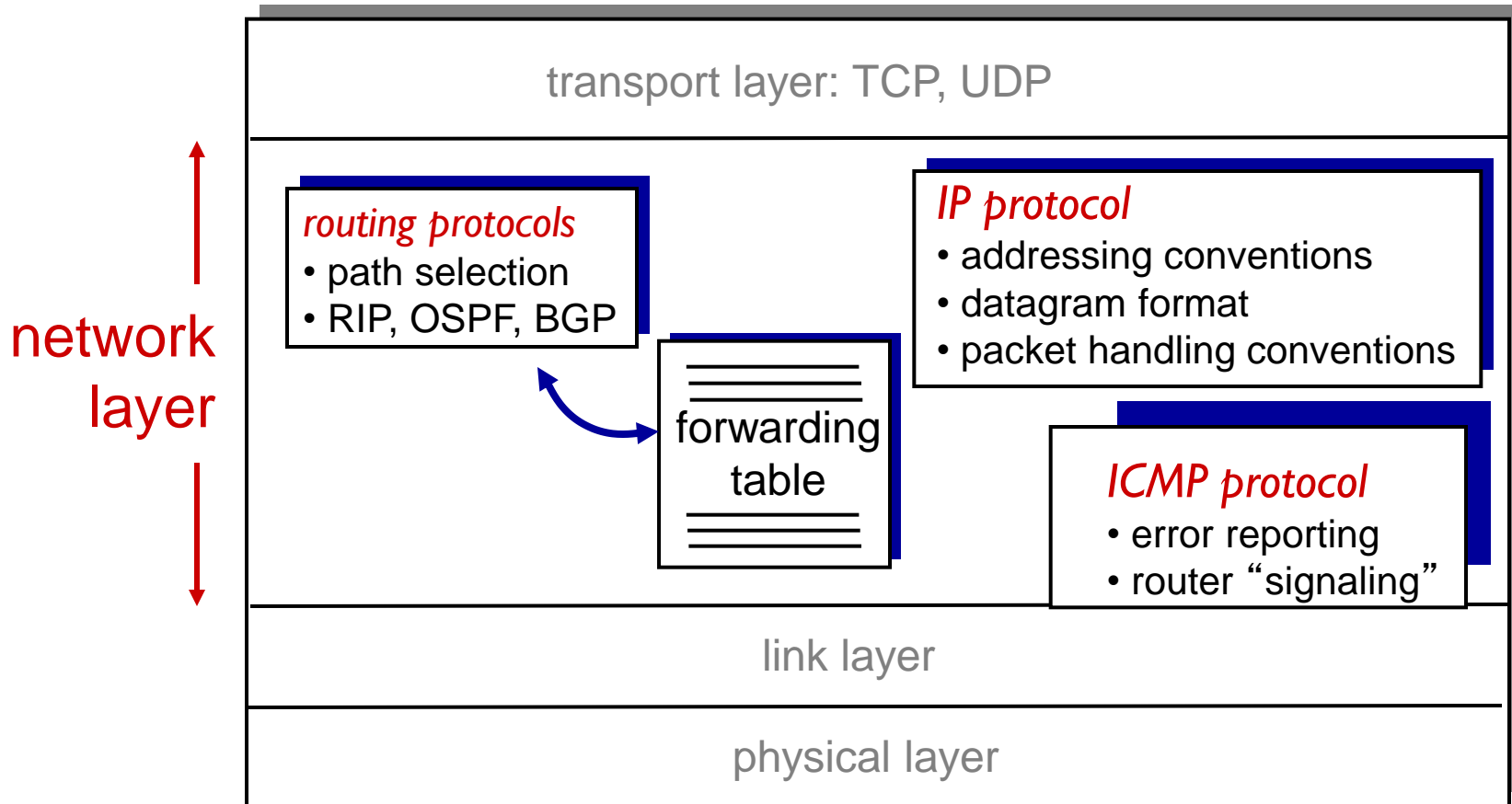
- Static Routing Table.
- Dynamic Routing Table

Figure 22.10 *Common fields in a routing table*

Mask	Network address	Next-hop address	Interface	Flags	Reference count	Use
.....

The Internet network layer

host, router network layer functions:



DHCP: Dynamic Host Configuration Protocol

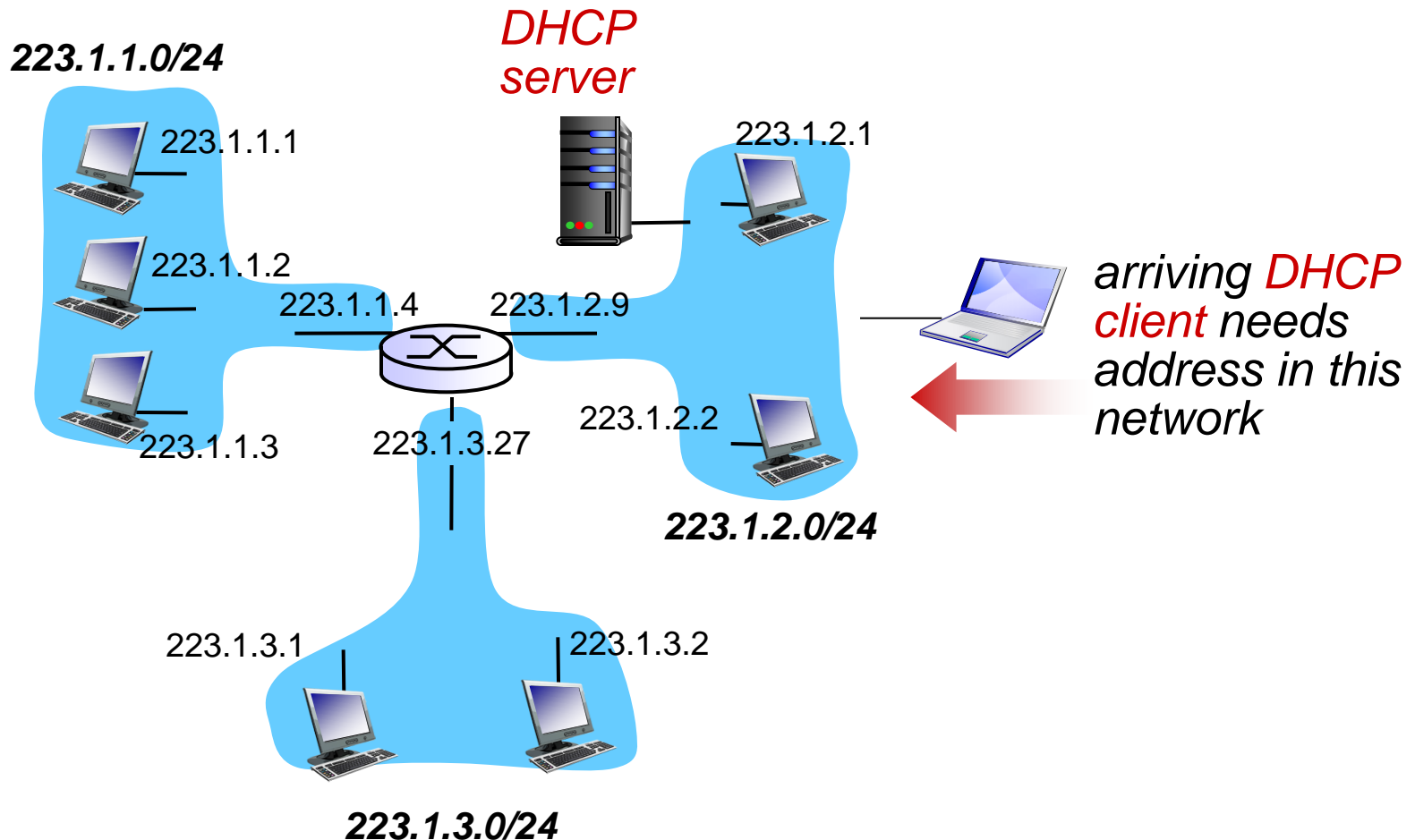
goal: allow host to *dynamically* obtain its IP address from network server when it joins network

- can renew its lease on address in use
- allows reuse of addresses (only hold address while connected/“on”)
- support for mobile users who want to join network (more shortly)

DHCP overview:

- host broadcasts “DHCP discover” msg [optional]
- DHCP server responds with “DHCP offer” msg [optional]
- host requests IP address: “DHCP request” msg
- DHCP server sends address: “DHCP ack” msg

DHCP client-server scenario



DHCP client-server scenario

DHCP server: 223.1.2.5



DHCP discover

Broadcast: is there a DHCP server out there?

arriving client



DHCP offer

Broadcast: I'm a DHCP server! Here's an IP address you can use

DHCP request

Broadcast: OK. I'll take that IP address!

DHCP ACK

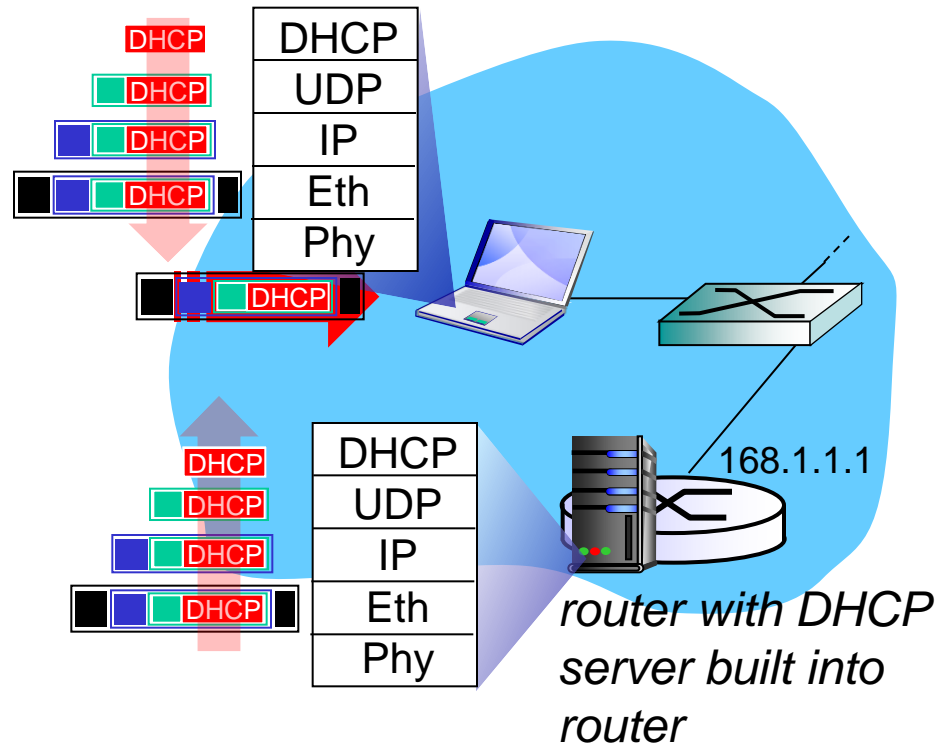
Broadcast: OK. You've got that IP address!

DHCP: more than IP addresses

DHCP can return more than just allocated IP address on subnet:

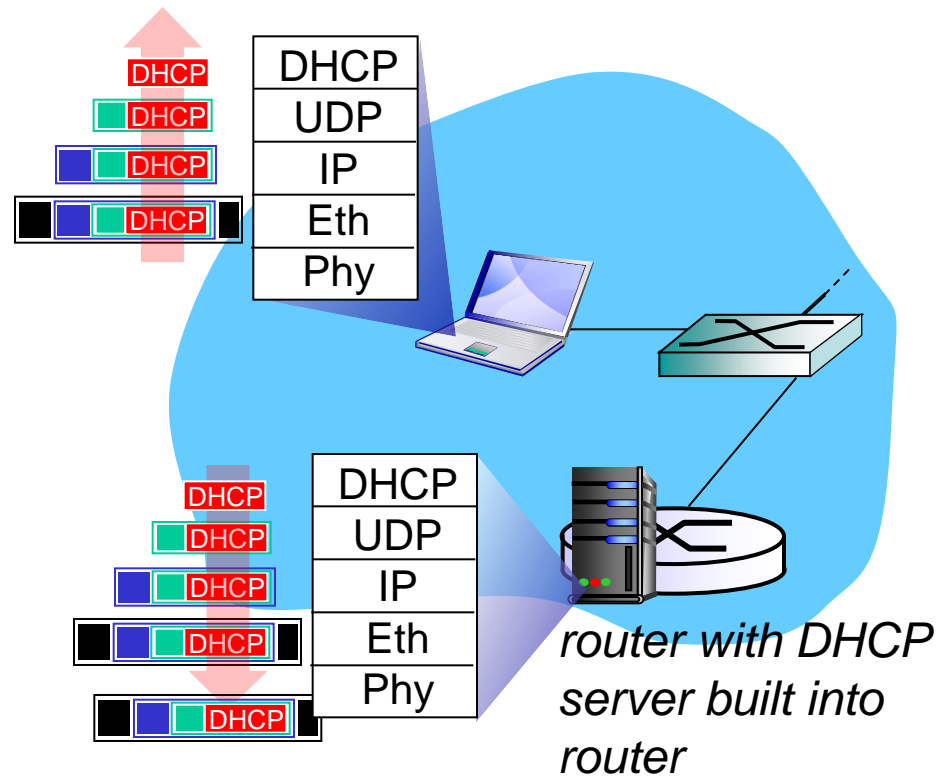
- address of first-hop router for client
- name and IP address of DNS sever
- network mask (indicating network versus host portion of address)

DHCP: example



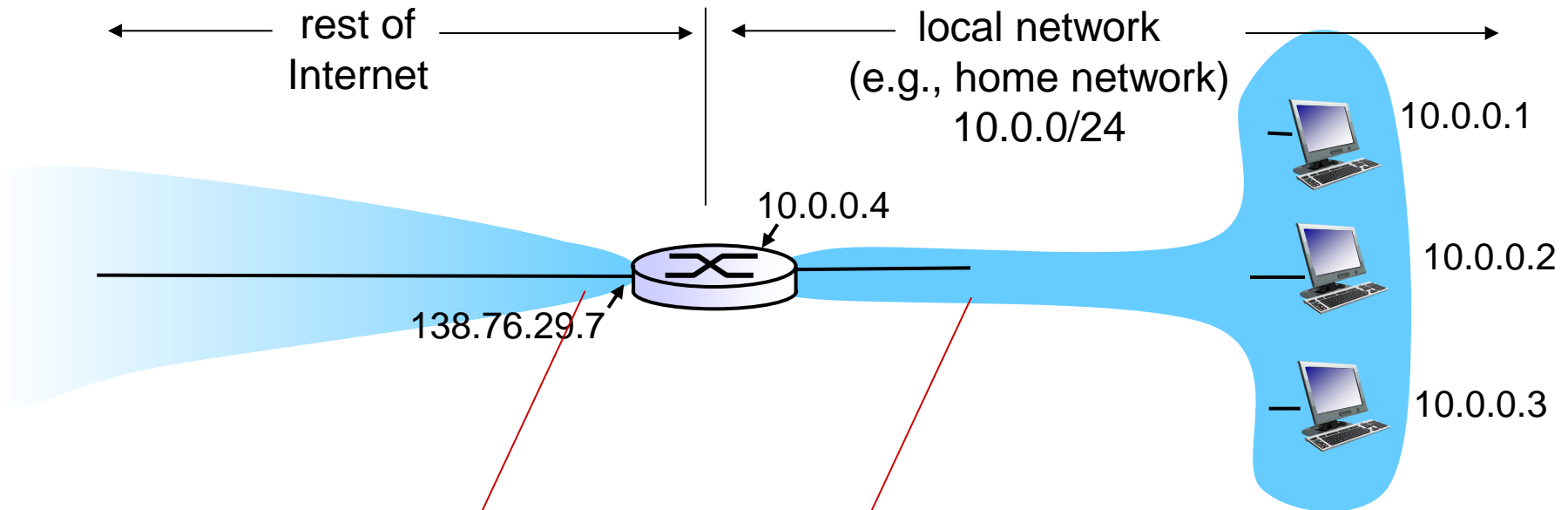
- ❖ connecting laptop needs its IP address, addr of first-hop router, addr of DNS server: use DHCP
- ❖ DHCP request encapsulated in UDP, encapsulated in IP, encapsulated in 802.1 Ethernet
- ❖ Ethernet frame broadcast (dest: FFFFFFFF) on LAN, received at router running DHCP server
- ❖ Ethernet demuxed to IP demuxed, UDP demuxed to DHCP

DHCP: example



- ❖ DHCP server formulates DHCP ACK containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- ❖ encapsulation of DHCP server, frame forwarded to client, demuxing up to DHCP at client
- ❖ client now knows its IP address, name and IP address of DNS server, IP address of its first-hop router

NAT: network address translation



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

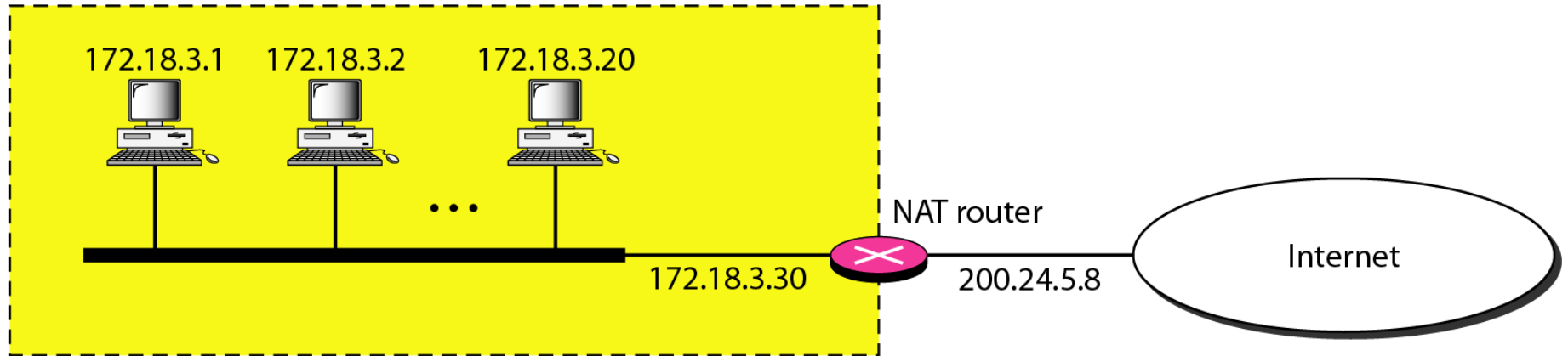
datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual)

Table Addresses for private networks

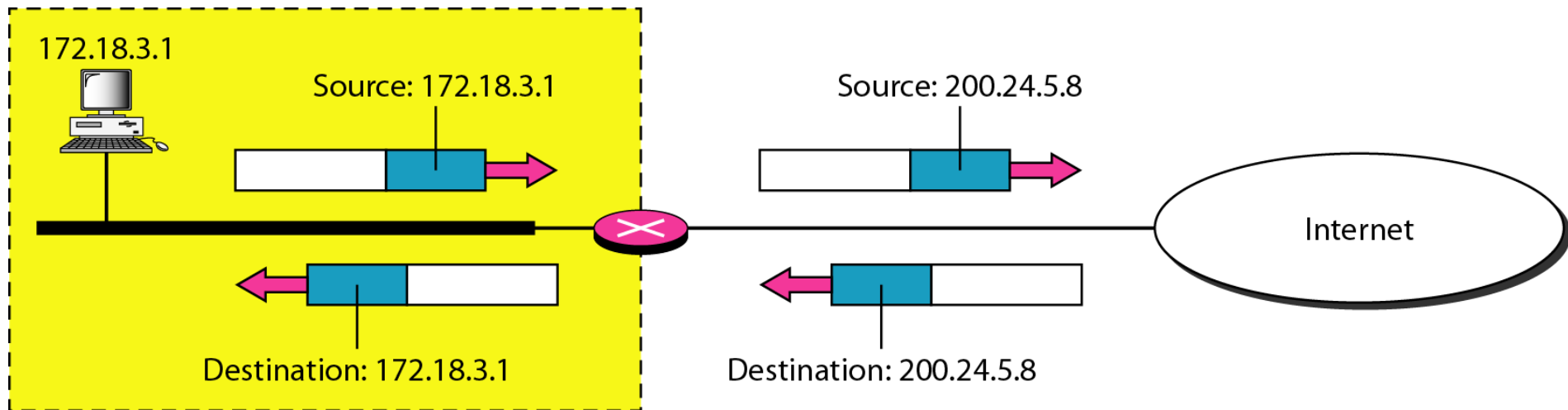
<i>Range</i>			<i>Total</i>
10.0.0.0	to	10.255.255.255	2^{24}
172.16.0.0	to	172.31.255.255	2^{20}
192.168.0.0	to	192.168.255.255	2^{16}

A NAT implementation

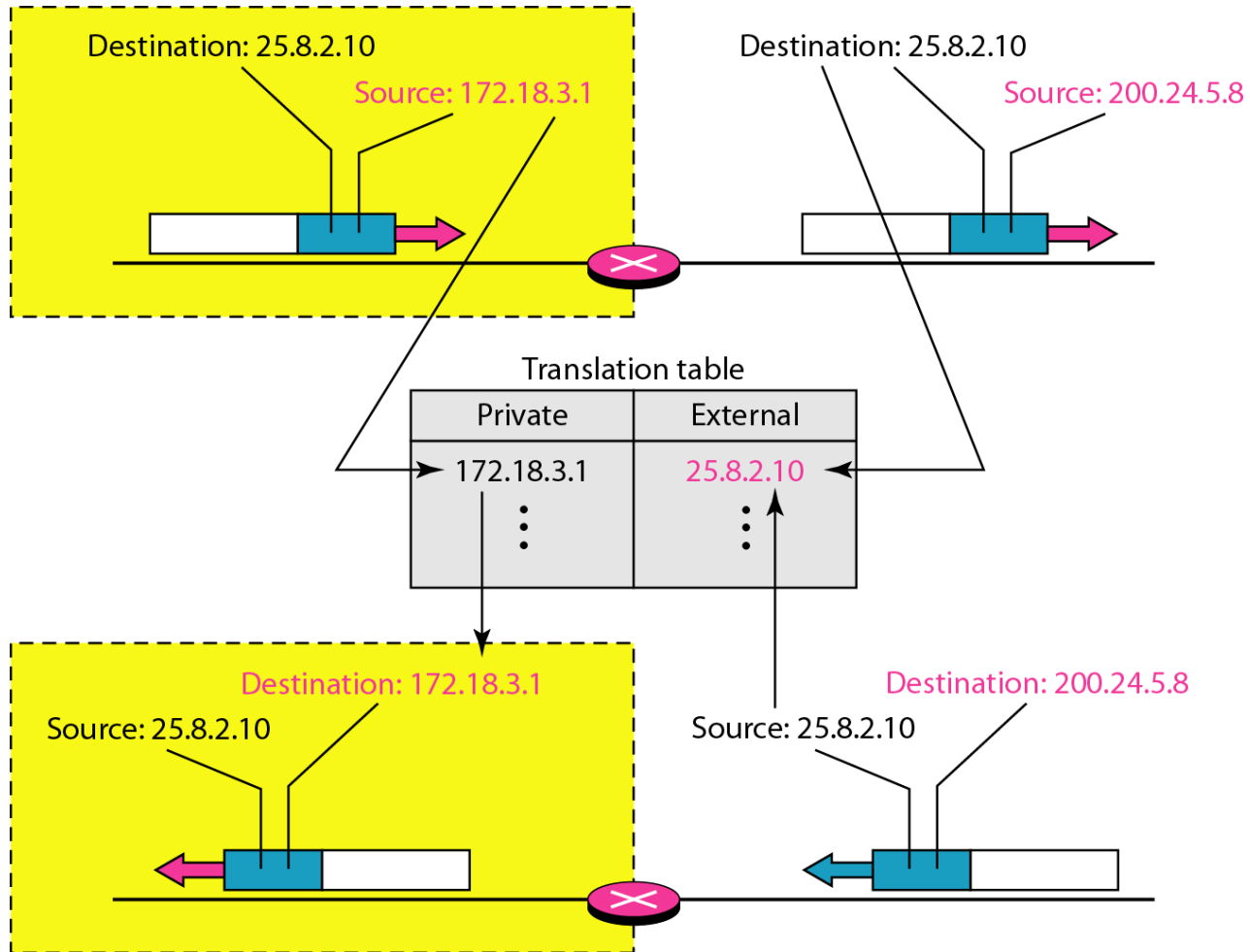
Site using private addresses



Addresses in a NAT



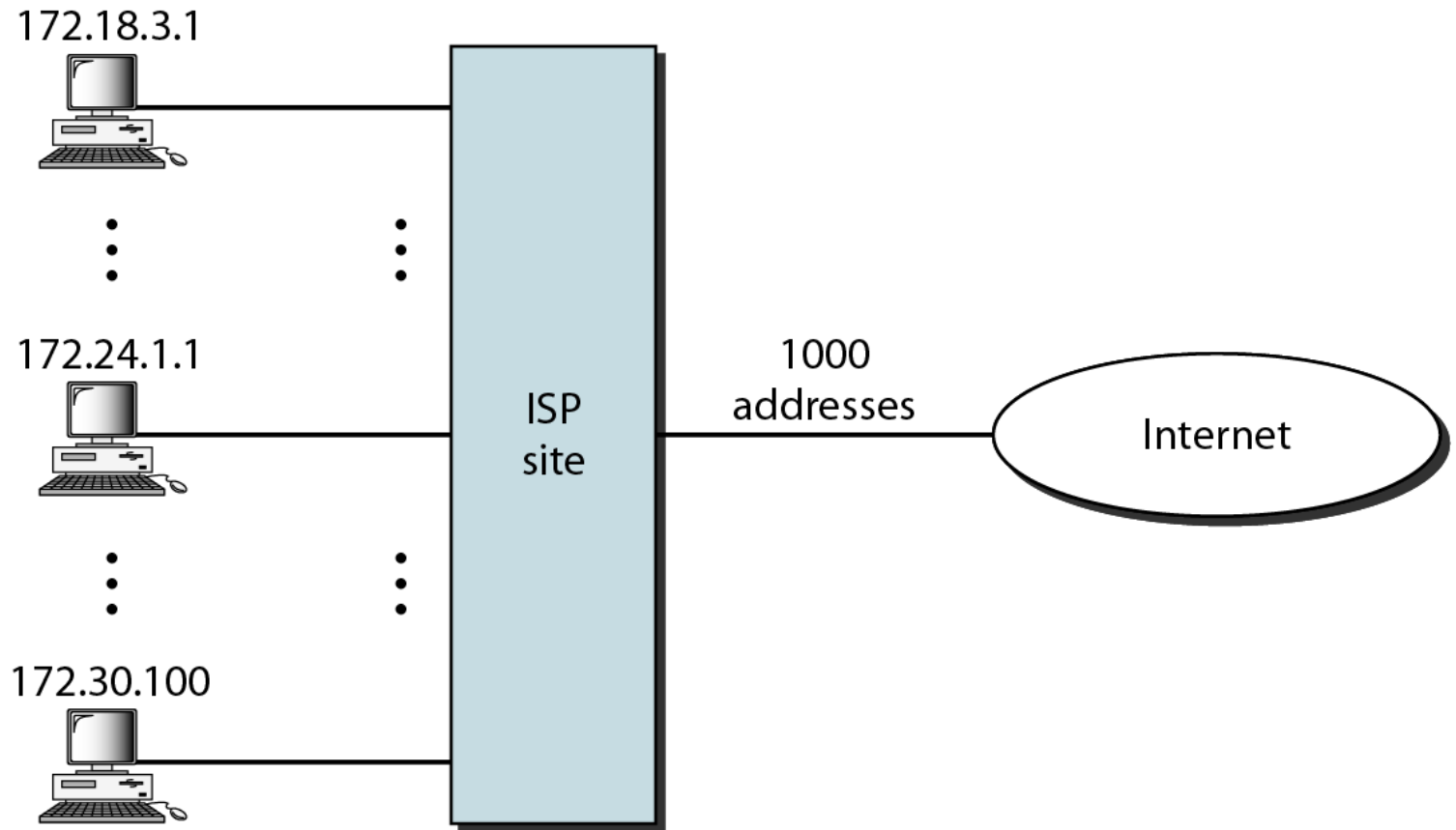
NAT address translation



Five-column translation table

<i>Private Address</i>	<i>Private Port</i>	<i>External Address</i>	<i>External Port</i>	<i>Transport Protocol</i>
172.18.3.1	1400	25.8.3.2	80	TCP
172.18.3.2	1401	25.8.3.2	80	TCP
...

An ISP and NAT



Routing algorithm classification

Q: global or decentralized information?

global:

- ❖ all routers have complete topology, link cost info
- ❖ “link state” algorithms

decentralized:

- ❖ router knows physically-connected neighbors, link costs to neighbors
- ❖ iterative process of computation, exchange of info with neighbors
- ❖ “distance vector” algorithms

Q: static or dynamic?

static:

- ❖ routes change slowly over time

dynamic:

- ❖ routes change more quickly
 - periodic update
 - in response to link cost changes

Figure 2 *Autonomous systems*

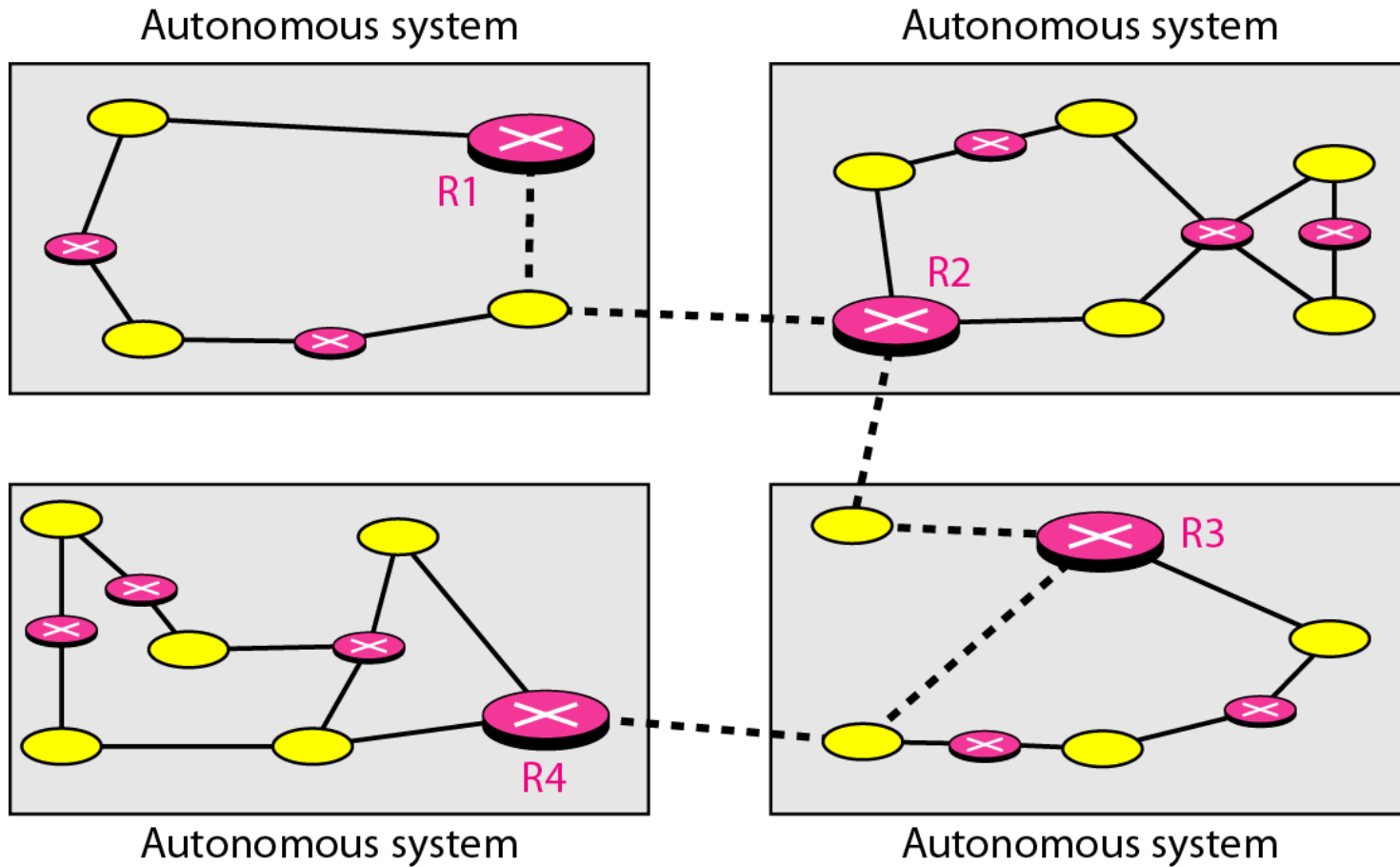


Figure 3 *Popular routing protocols*

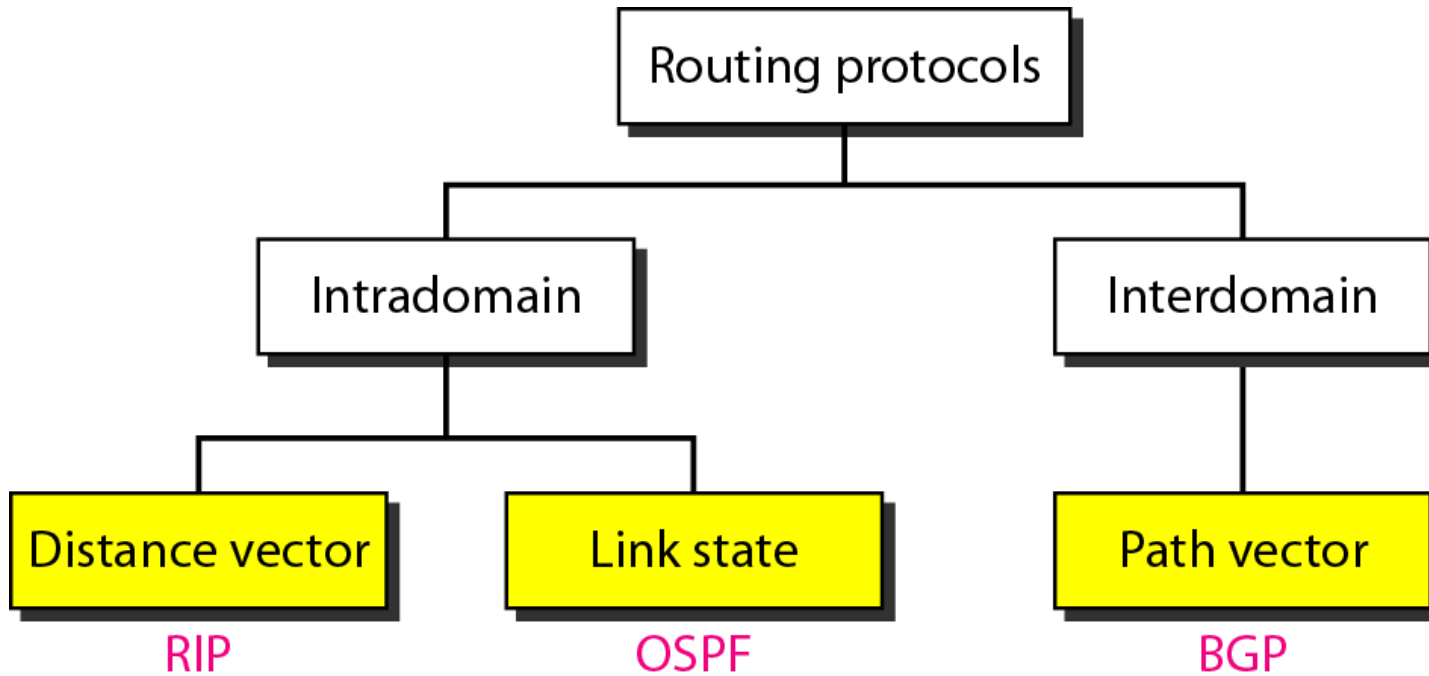


Figure 4 *Distance vector routing tables*

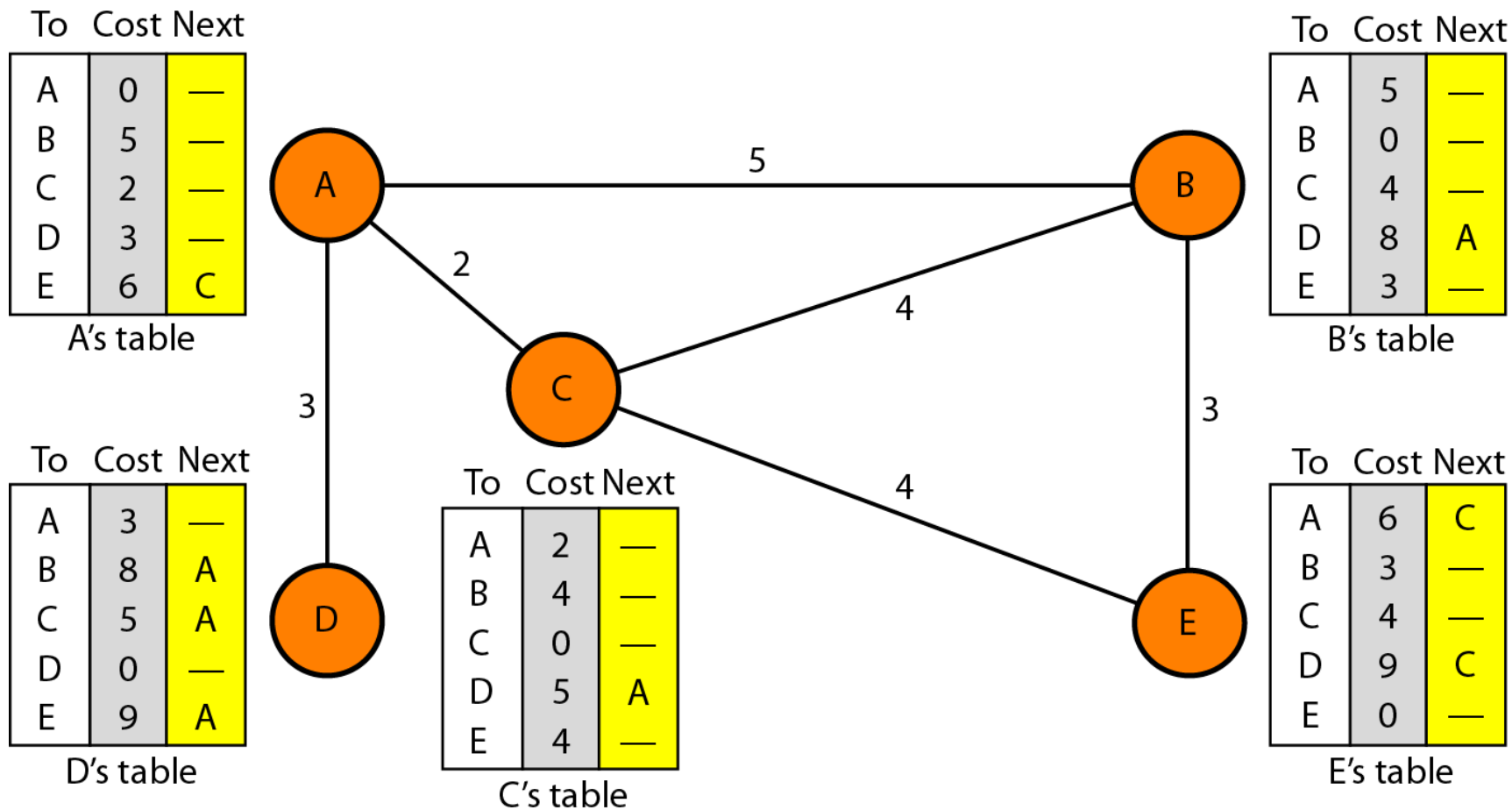
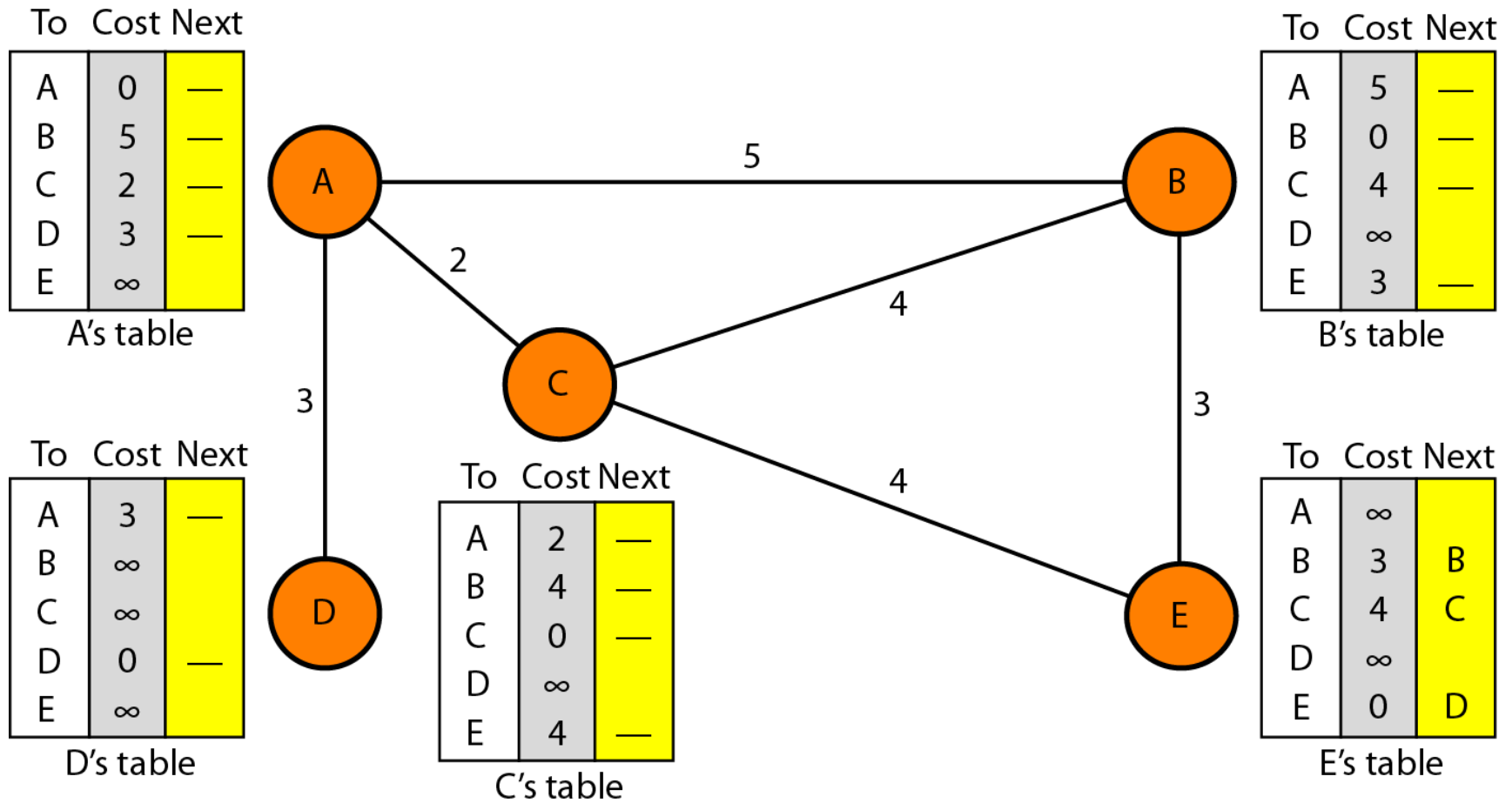



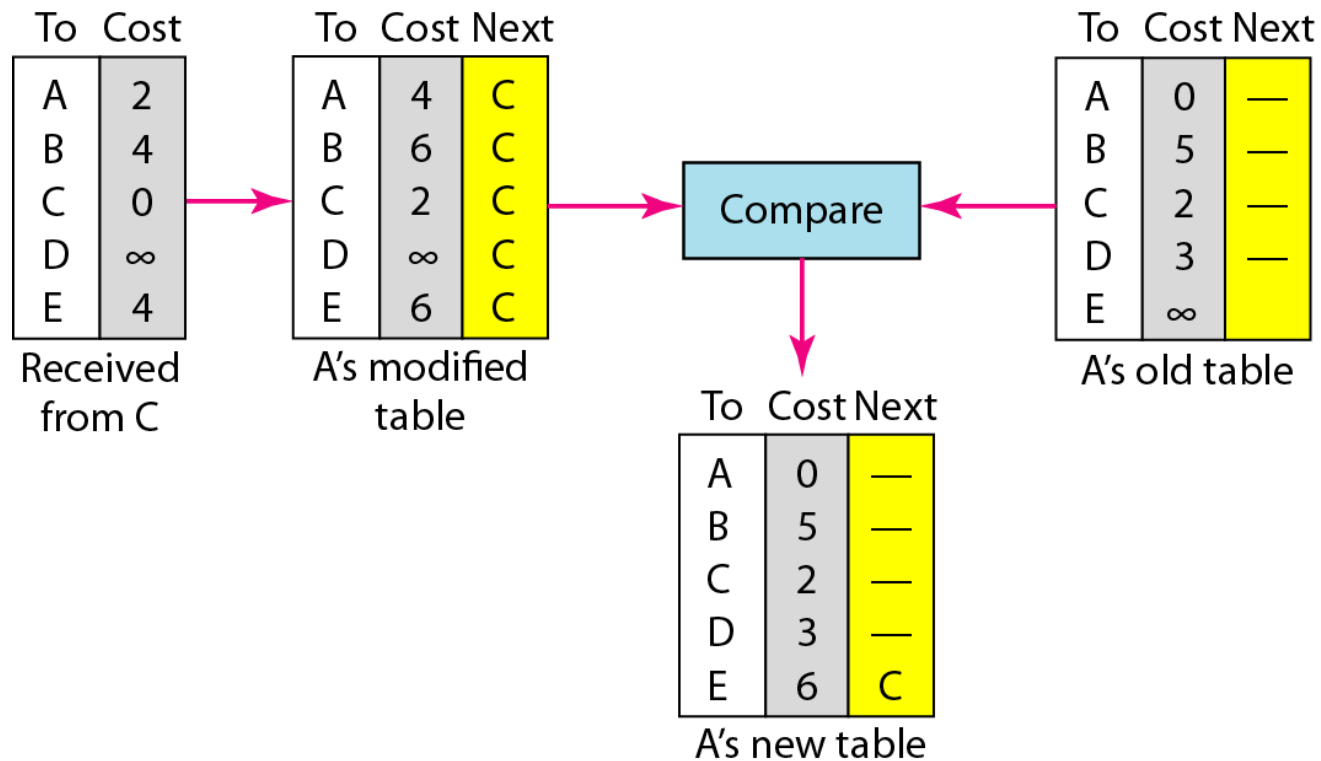
Figure 5 *Initialization of tables in distance vector routing*





In distance vector routing, each node shares its routing table with its immediate neighbors periodically and when there is a change.

Figure 6 *Updating in distance vector routing*



link state routing

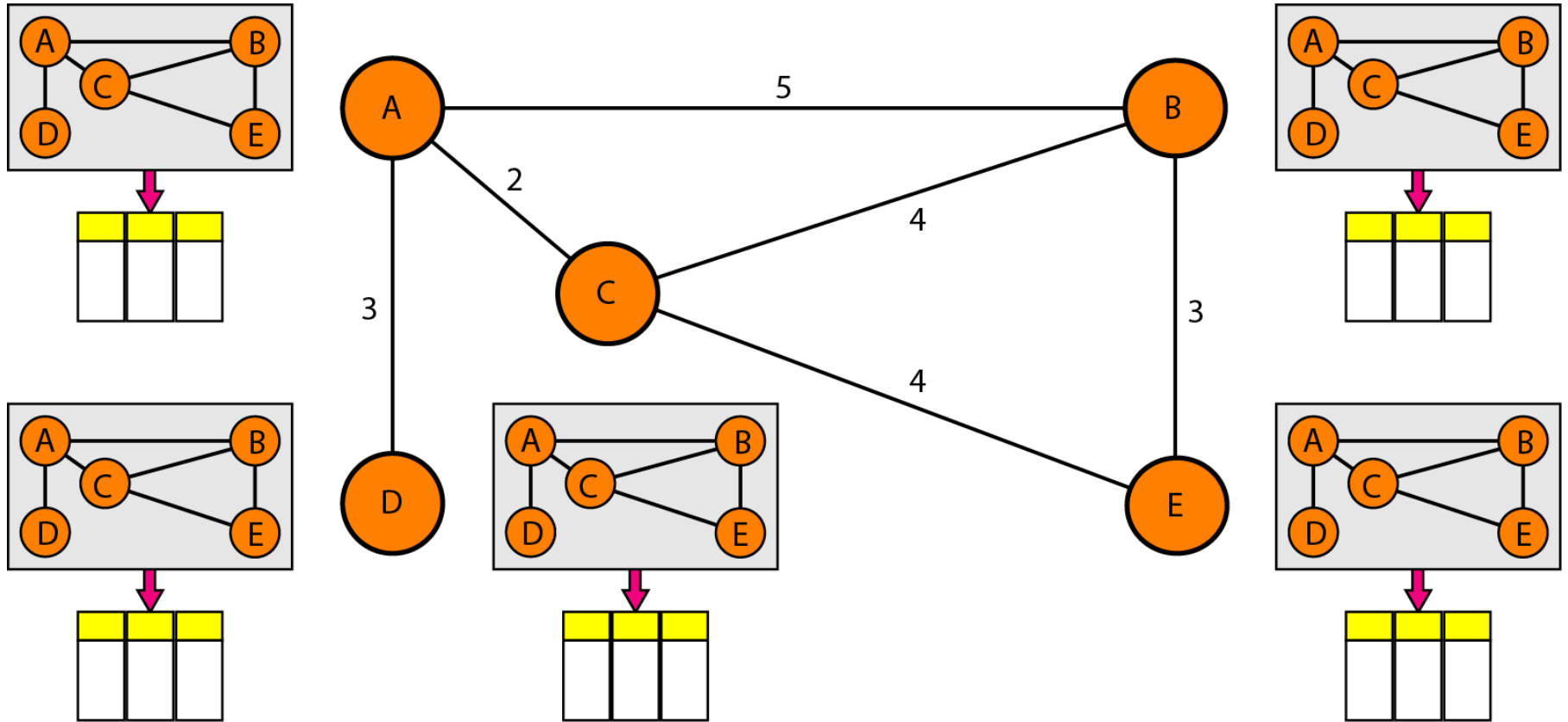


Figure 22.21 *Link state knowledge*

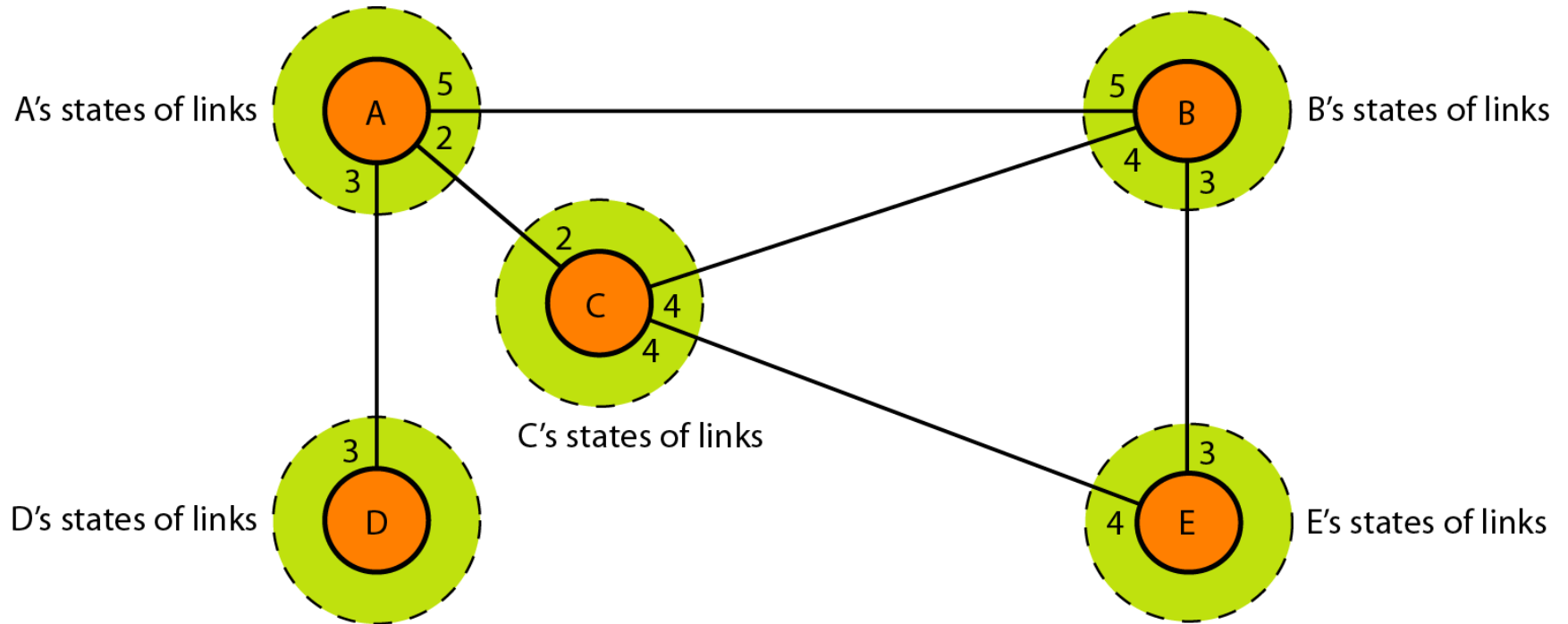


Figure 22.22 *Dijkstra algorithm*

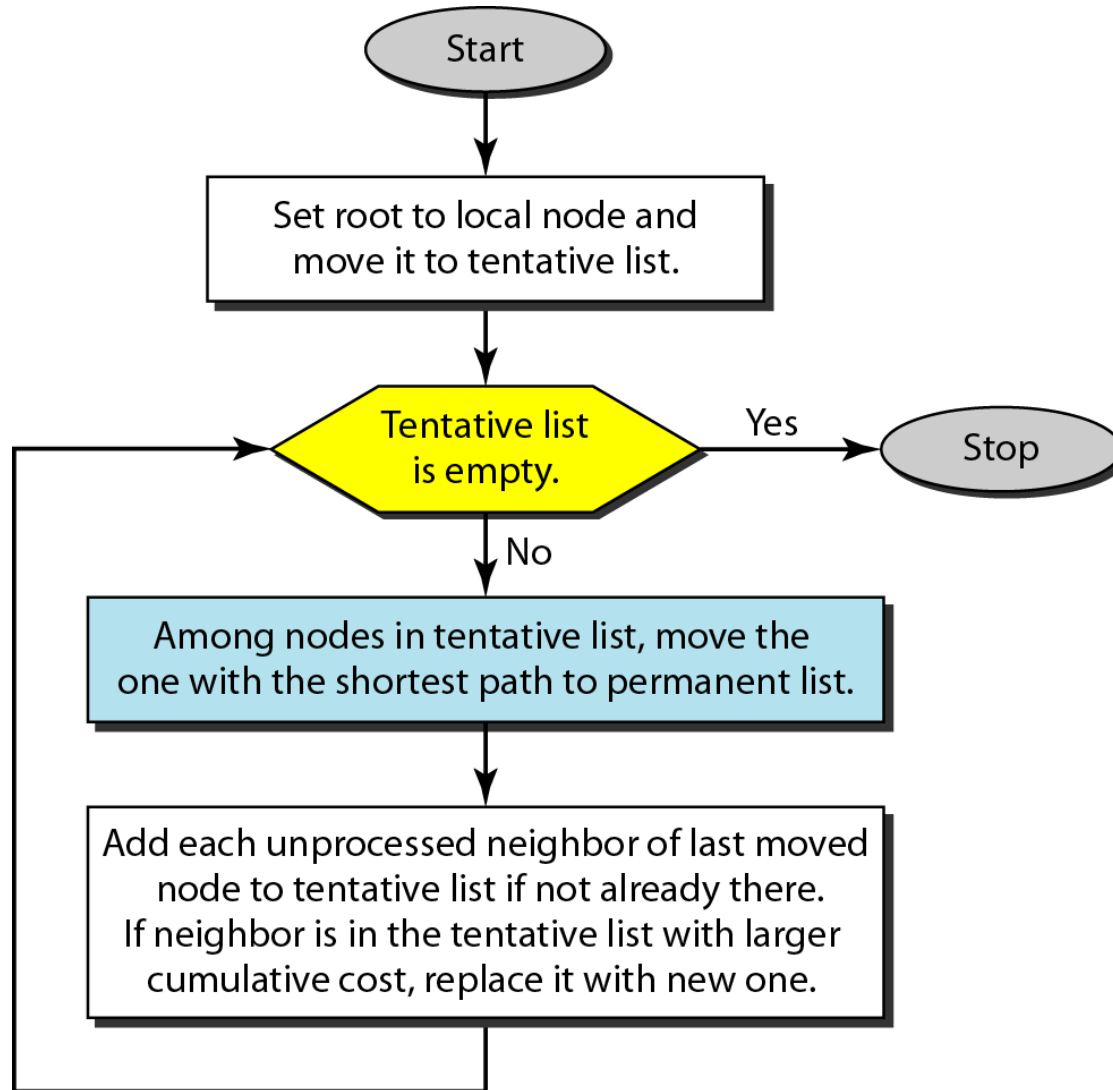
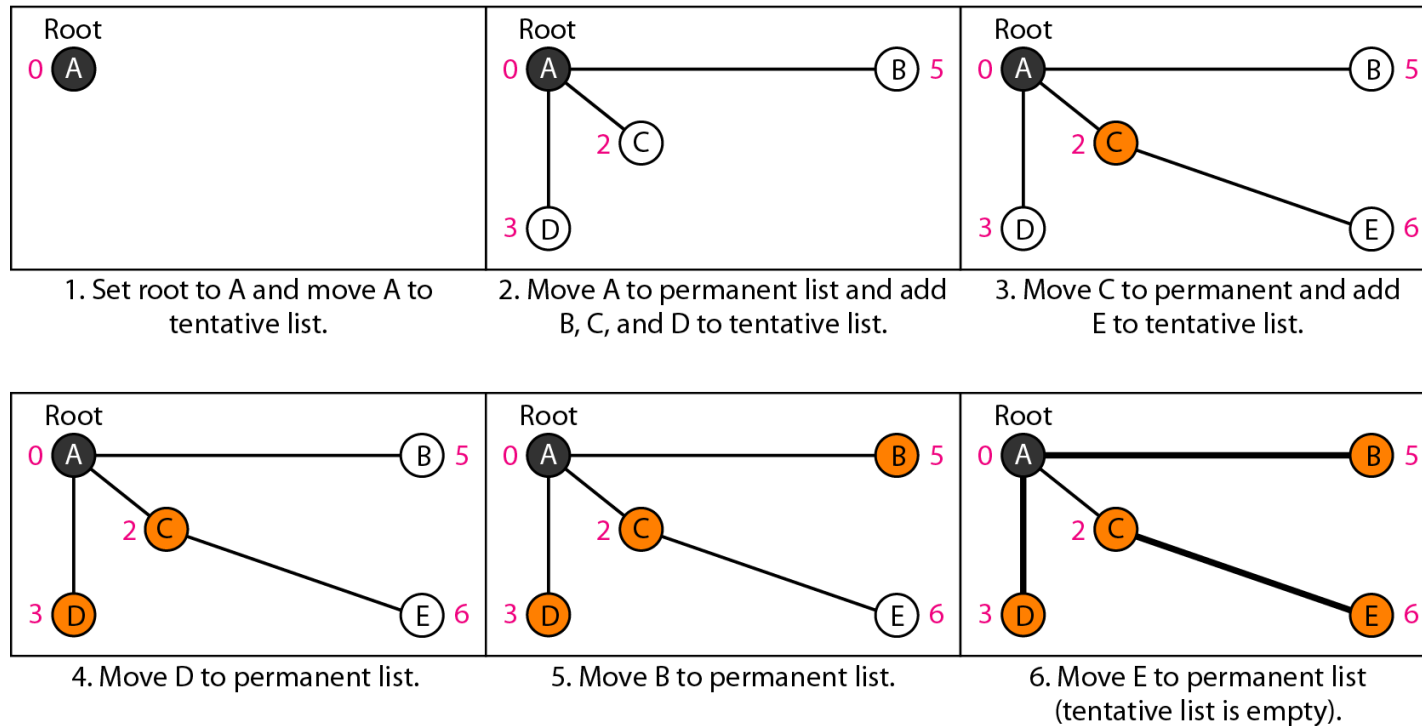
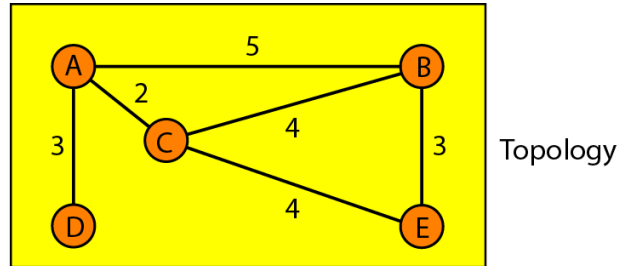


Figure 22.23 *Example of formation of shortest path tree*



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

ICMP: internet control message protocol

- ❖ used by hosts & routers to communicate network-level information

- error reporting: unreachable host, network, port, protocol
- echo request/reply (used by ping)

- ❖ network-layer “above” IP:

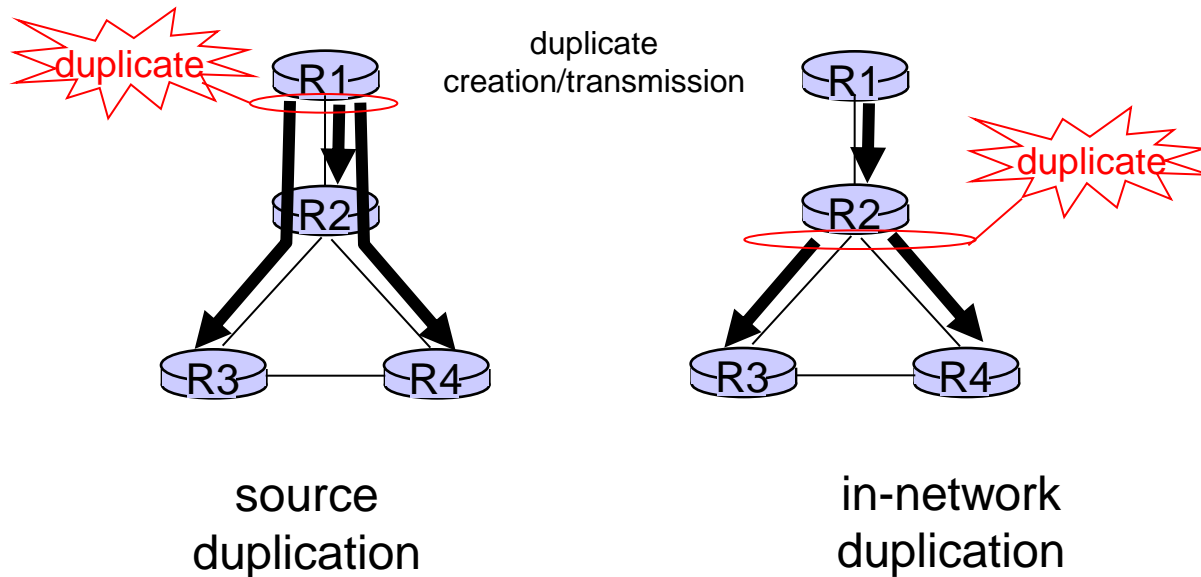
- ICMP msgs carried in IP datagrams

- ❖ **ICMP message:** type, code plus first 8 bytes of IP datagram causing error

<u>Type</u>	<u>Code</u>	<u>description</u>
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

Broadcast routing

- ❖ deliver packets from source to all other nodes
- ❖ source duplication is inefficient:



- ❖ source duplication: how does source determine recipient addresses?

Multicast routing: problem statement

goal: find a tree (or trees) connecting routers having local mcast group members

- ❖ *tree:* not all paths between routers used
- ❖ *shared-tree:* same tree used by all group members
- ❖ *source-based:* different tree from each sender to rcvrs

legend



group member



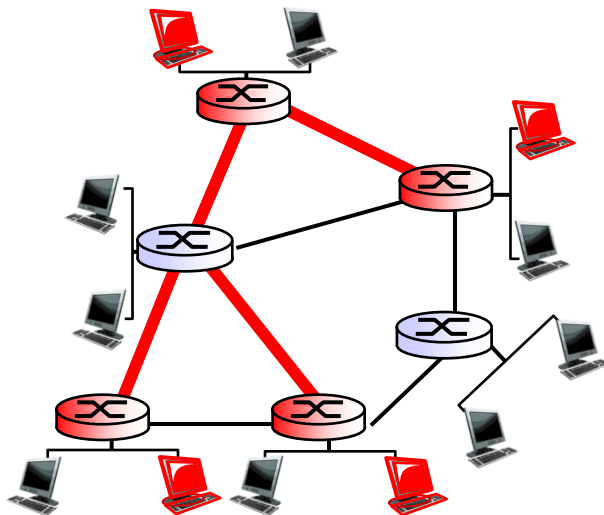
not group member



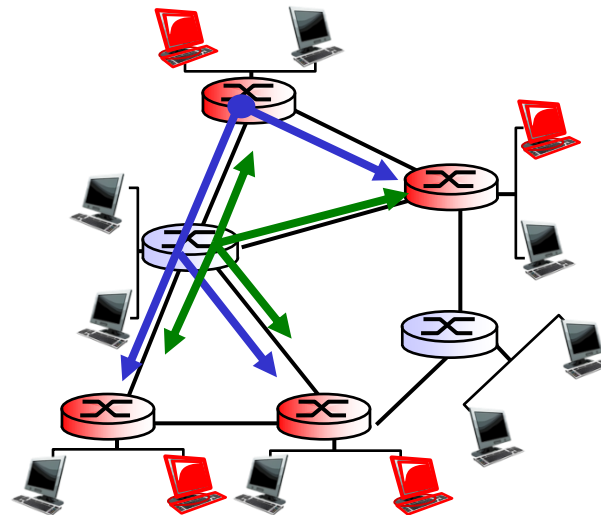
router with a group member



router without group member



shared tree



source-based trees