# Unicast Routing Protocols

**Presented By**
**Dr. Md. Abir Hossain**

# Chapter Outline

# INTRODUCTION

- An internet is a combination of networks connected by routers.
- When a datagram goes from a source to a destination, it will probably pass through many routers until it reaches the router attached to the destination network.

# INTER- AND INTRA-DOMAIN ROUTING

- Today, an internet can be so large that one routing protocol cannot handle the task of updating the routing tables of all routers.
- For this reason, an internet is divided into autonomous systems.
- An autonomous system (AS) is a group of networks and routers under the authority of a single administration.
- Routing inside an autonomous system is called intra-domain routing.
- Routing between autonomous systems is called inter-domain routing

# Autonomous System

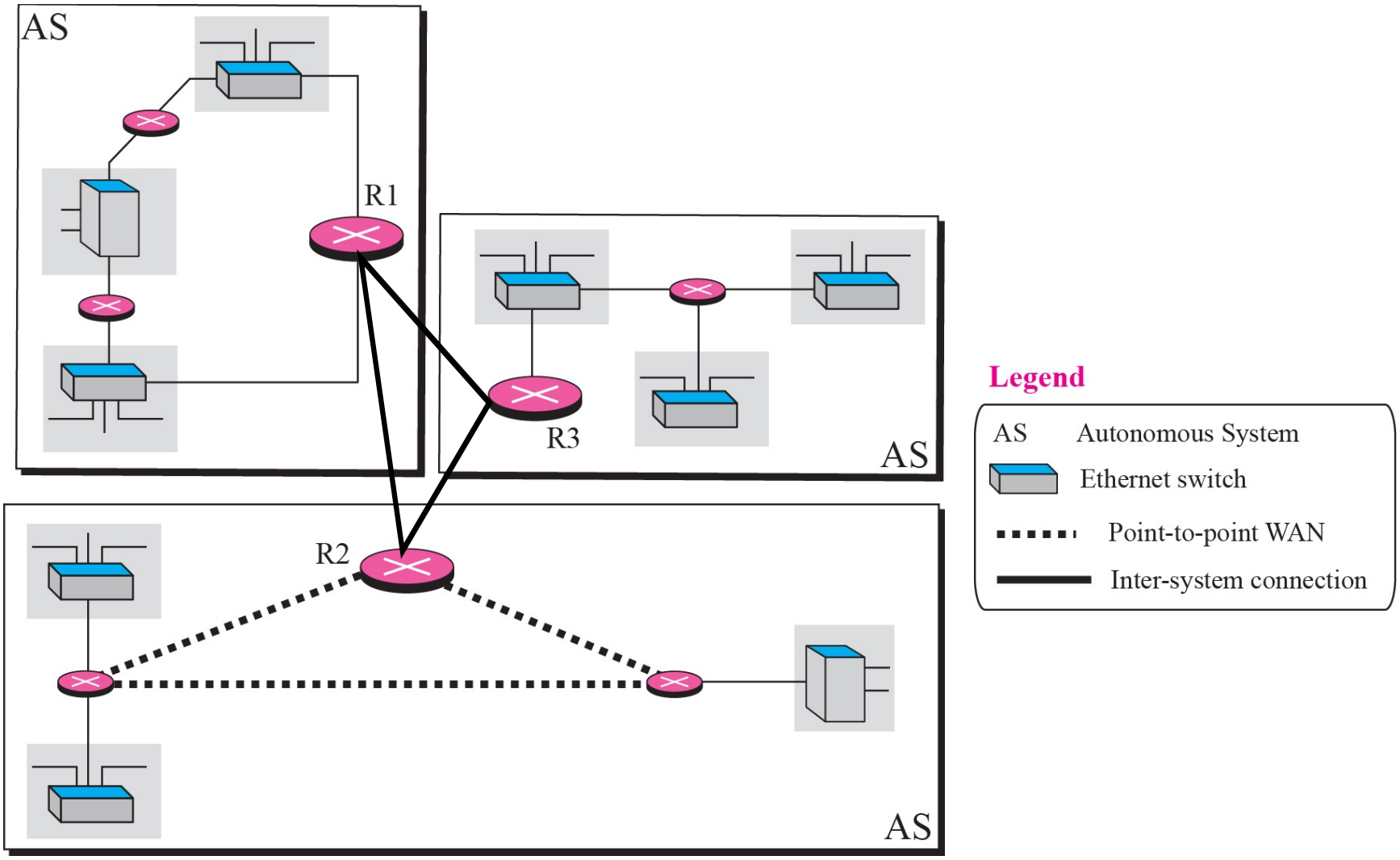



**Figure 11.1** *Autonomous systems*

# Routing Protocols

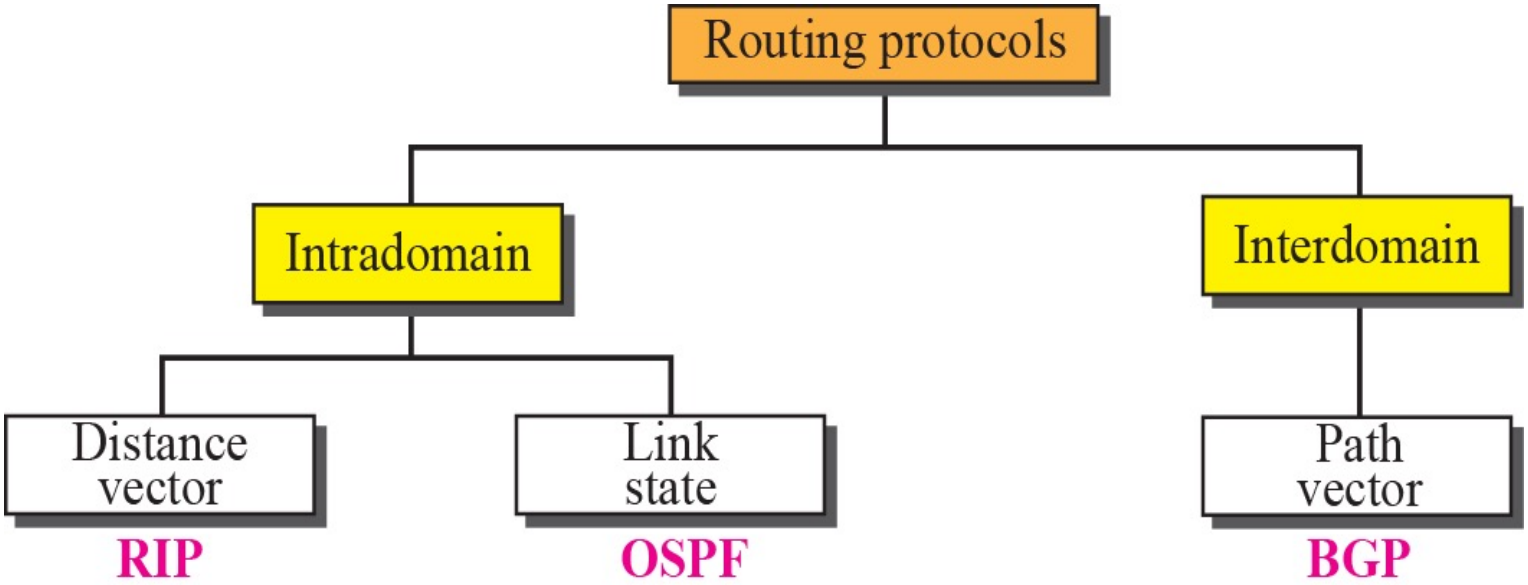



**Figure 11.2** *Popular routing protocols*

# *Topics Discussed in the Section*

- ✓ **Bellman-Ford Algorithm**
- ✓ **Distance Vector Routing Algorithm**
- ✓ **Count to Infinity**

# Graph for Bellman-Ford Algorithm

- Figure presents a map with nodes and lines.
- The cost of each line is given over the line.
- The algorithm can find the least cost between any two nodes.
- For example, if the nodes represent cities and the lines represent roads connecting them, the graph can find the shortest distance between any two cities.

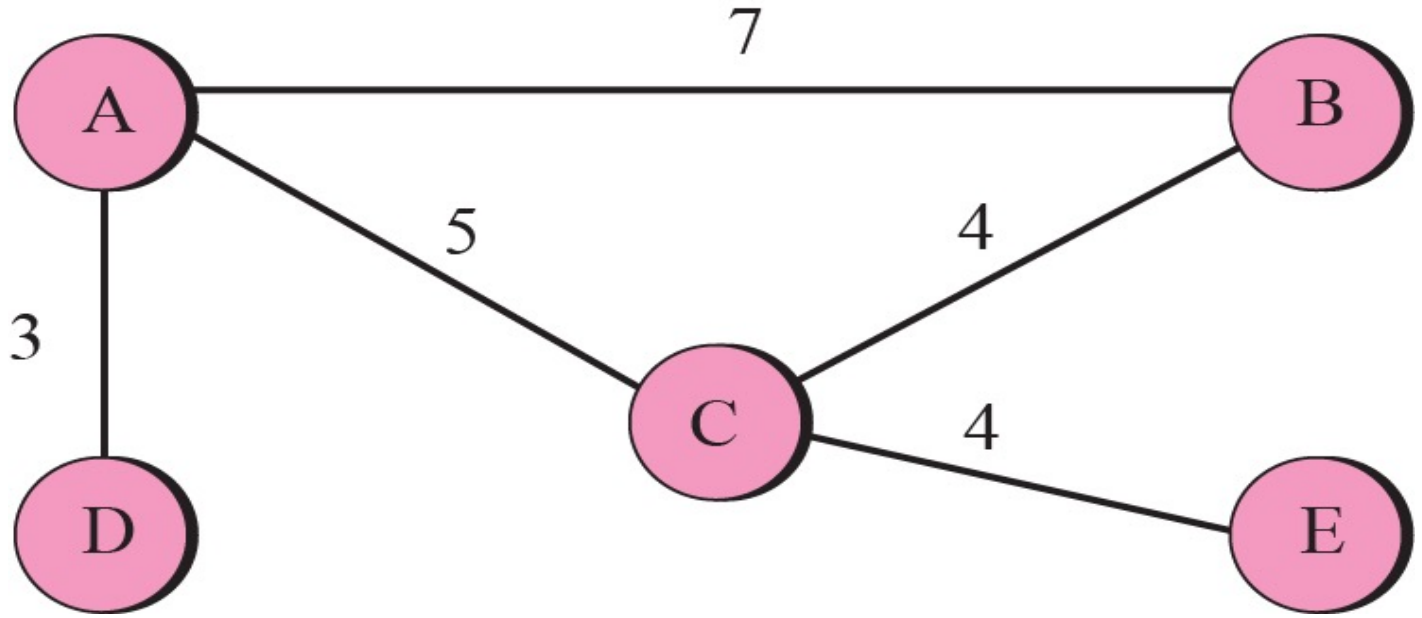



**Figure 11.3** *A graph for Bellman-Ford algorithm*

# Main Fact of Bellman-Ford Algorithm

$$D_{ij} = \text{minimum } \{(c_{i1} + D_{1j}), (c_{i2} + D_{2j}), \ldots (c_{iN} + D_{Nj})$$

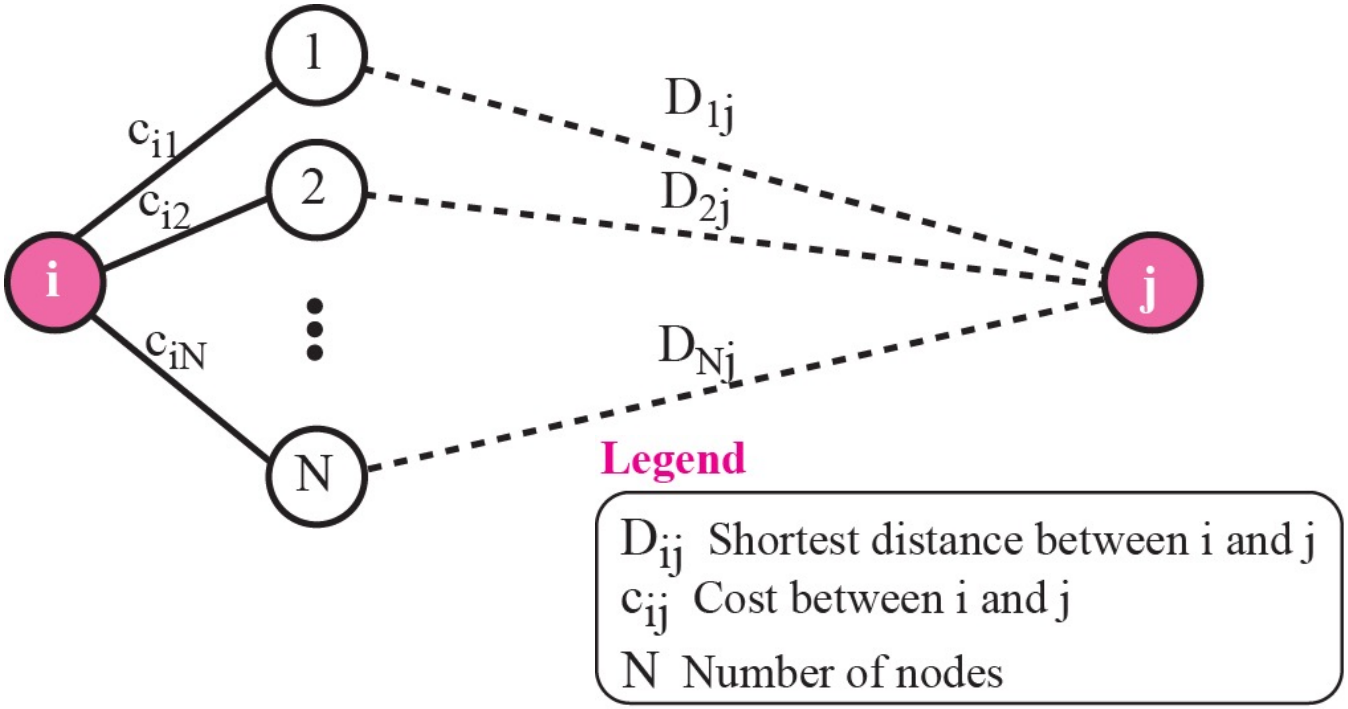



**Figure 11.4** *The fact behind Bellman-Ford algorithm*

# Bellman-Ford Algorithm

A shortest distance table (vector) for each node is created using the following steps:

- The shortest distance and the cost between a node and itself is initialized to 0.
- The shortest distance between a node and any other node is set to infinity. The cost between a node and any other node should be given (can be infinity if the nodes are not connected).
- The algorithm repeat as shown in Figure 11.4 until there is no more change in the shortest distance vector.

**Table 11.1** *Bellman-Ford Algorithm*

```
1  Bellman_Ford ( )
2  {
3     // Initialization
4     for (i = 1 to N; for j = 1 to N)
5     {
6        if(i == j)   Dij = 0   cij = 0
7        else         Dij = ∞   cij = cost between i and j
8     }
9     // Updating
10    repeat
11    {
12       for (i = 1 to N; for j = 1 to N)
13       {
14          Dij ← minimum [(ci1 + D1j)  ...  (ciN + DNj)]
15       } // end for
16    } until (there was no change in previous iteration)
17 } // end Bellman-Ford
```

# Distance-Vector Algorithm

- The Bellman-Ford algorithm can be very well applied to a map of roads between cities because we can have all of the initial information about each node at the same place.
- But If we want to use the algorithm for creating the routing table for **routers** in an AS, we need to change the algorithm:
- In distance vector routing, the cost is normally hop counts. So the cost between any two neighbour's is set to 1.
- Each router needs to update its routing table *asynchronously,* whenever it has received some information from its neighbour's.
- After a router has updated its routing table, it should send the result to its neighbour's so that they can also update their routing table.

**Table 11.2** *Distance Vector Algorithm Used by Each Router*

```
1  Distance_Vector_Algorithm ( )
2  {
3      // At startup
4      for (i = 1 to N)                  // N is number of ports
5      {
6          Table_i.dest = address of the attached network
7          Table_i.cost = 1
8          Table_i.next = —              // Means at home
9          Send a record R about each row to each neighbor
10     } // end for loop
11
12       // Updating
13     repeat (forever)
14     {
15         Wait for arrival of a record R from a neighbor
16         Update (R, T)                 // Call update module
17         for (i = 1 to N)              // N is the current table size
```

# Distance-Vector Algorithm

- Each router should keep at least three pieces of information for each route: destination network, the cost, and the next hop. We refer to the whole routing table as Table, to the row *i* in the table as Table*i*, to the three columns in row *i* as Table*i*.dest, Table*i*.cost, and Table*i*.next.
- We refer to information about each route received from a neighbor as R (record), which has only two pieces of information: R.dest and R.cost. The next hop is not included in the received record because it is the source address of the sender.

**Table 11.2** *Distance Vector Algorithm Used by Each Router (continued)*

```
18          {
19                Send a record R about each row to each neighbor
20          }
21        }  // end repeat
22
23  }  // end Distance_Vector
24  Update (R, T)              // Update module
25  {
26      Search T for a destination matching the one in R
27      if (destination is found in row i)
28      {
29          if (R.cost + 1 < Ti.cost     or     R.next == Ti.next)
30          {
31                Ti.cost = R.cost + 1
32                Ti.next = Address of sending router
33          }
```

**Address of the sender of R**

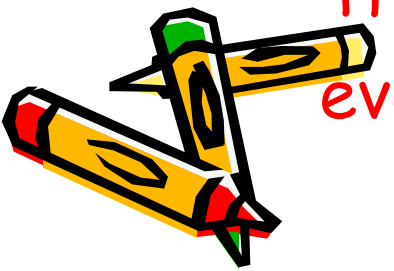
# Distance-Vector Algorithm

**Table 11.2** *Distance Vector Algorithm Used by Each Router (continued)*

```
34        else  discard  the  record      // No change is needed
35        }
36        else
37         // Insert the new router
38        {
39            T_{N+1}.dest = R.dest
40            T_{N+1}.cost = R.cost + 1
41            T_{N+1}.next = Address of sending router
42            Sort the table according to destination address
43        }
44  }  // end of Update module
```

# Updating Routing Table

- If the next-node entry is different
  - The receiving node chooses the row with the smaller cost
  - If there is a tie, the old one is kept
- If the next-node entry is the same
  - i.e. the sender of the new row is the provider of the old entry
  - The receiving node chooses the new row, even though the new value is infinity.



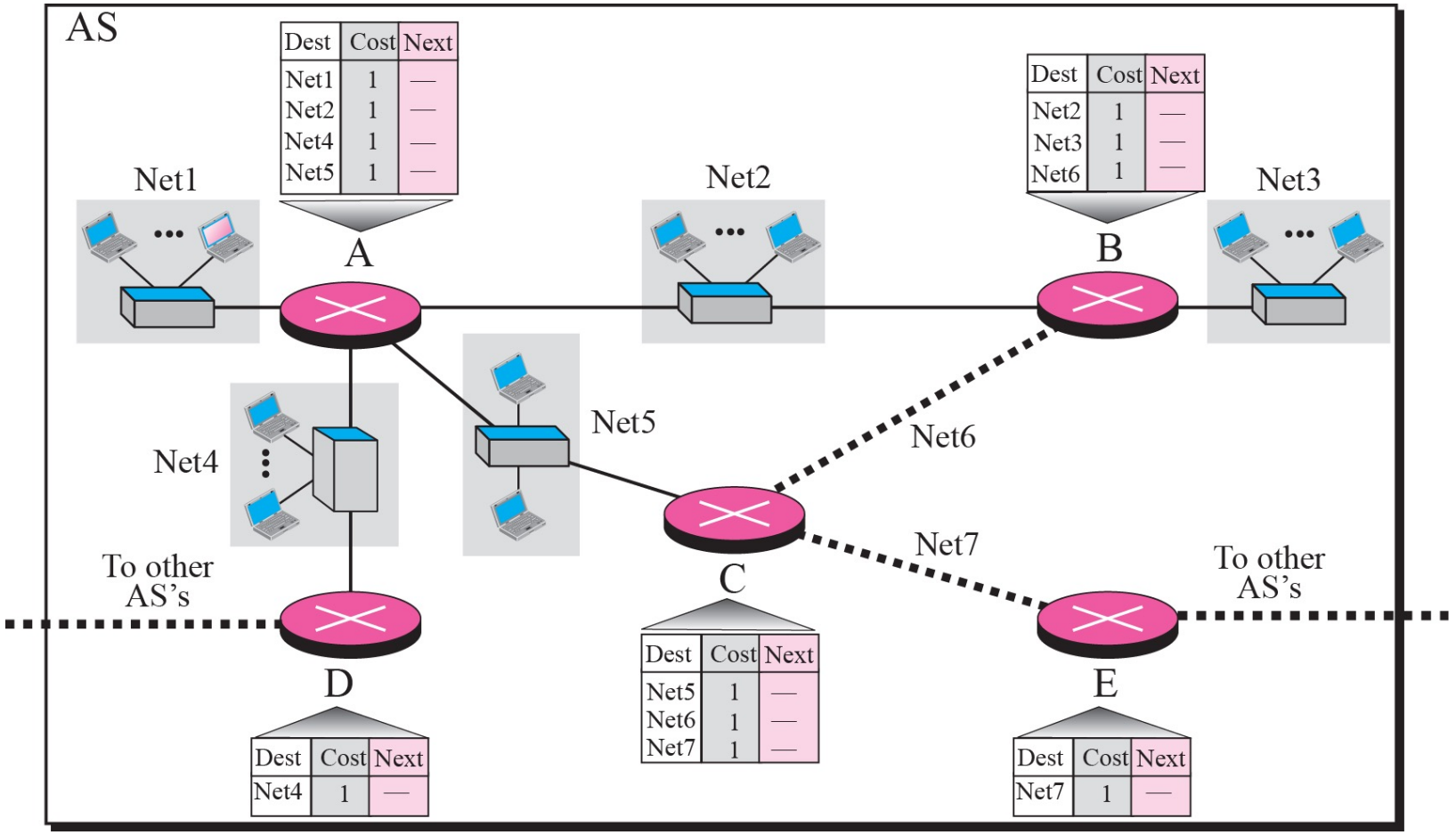
# When to Share

- Periodic Update
  - A node sends its routing table, normally 30 seconds, in a periodic update
- Triggered Update
  - A node sends its routing table to its neighbors any time when there is a change in its routing table
    - 1. After updating its routing table, or
    - 2. Detects some failure in the neighboring links

# Example 11.1

Figure 11.5 shows the initial routing table for an AS. Note that the figure does not mean that all routing tables have been created at the same time; each router creates its own routing table when it is booted.

## Figure 11.5 *Example 11.1*



# Example 11.2

Now assume router A sends four records to its neighbors, routers B, D, and C. Figure 11.6 shows the changes in B's routing table when it receives these records. We leave the changes in the routing tables of other neighbors as exercise.

## Figure 11.6 *Example 11.2*


Routing Table B

| Dest | Cost | Next |
|---|---|---|
| Net1 | 2 | A |
| Net2 | 1 | — |
| Net3 | 1 | — |
| Net6 | 1 | — |

After receiving record 1

Routing Table B

| Dest | Cost | Next |
|---|---|---|
| Net1 | 2 | A |
| Net2 | 1 | — |
| Net3 | 1 | — |
| Net6 | 1 | — |

After receiving record 2

Routing Table B

| Dest | Cost | Next |
|---|---|---|
| Net1 | 2 | A |
| Net2 | 1 | — |
| Net3 | 1 | — |
| Net4 | 2 | A |
| Net6 | 1 | — |

After receiving record 3

Routing Table B

| Dest | Cost | Next |
|---|---|---|
| Net1 | 2 | A |
| Net2 | 1 | — |
| Net3 | 1 | — |
| Net4 | 2 | A |
| Net5 | 2 | A |
| Net6 | 1 | — |

After receiving record 4

# Example 11.3

Figure 11.7 shows the final routing tables for routers in Figure 11.5.

## Figure 11.7 *Example 11.3*

A

| Dest | Cost | Next |
|---|---|---|
| Net1 | 1 | — |
| Net2 | 1 | — |
| Net3 | 2 | B |
| Net4 | 1 | — |
| Net5 | 1 | — |
| Net6 | 2 | C |
| Net7 | 2 | C |

B

| Dest | Cost | Next |
|---|---|---|
| Net1 | 2 | A |
| Net2 | 1 | — |
| Net3 | 1 | — |
| Net4 | 2 | A |
| Net5 | 2 | A |
| Net6 | 1 | — |
| Net7 | 2 | C |

C

| Dest | Cost | Next |
|---|---|---|
| Net1 | 2 | A |
| Net2 | 2 | A |
| Net3 | 2 | B |
| Net4 | 2 | A |
| Net5 | 1 | — |
| Net6 | 1 | — |
| Net7 | 1 | — |

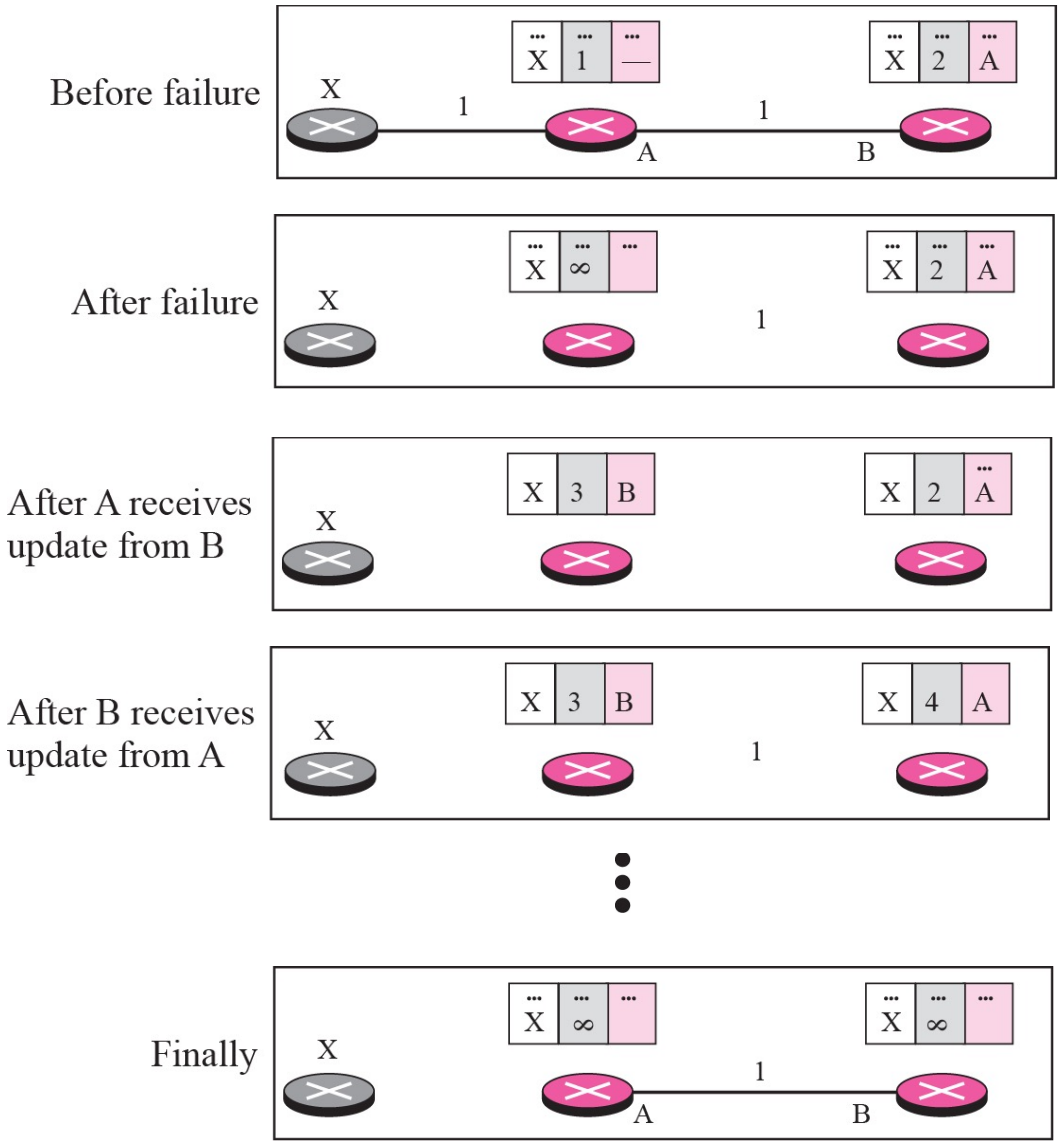
D

| Dest | Cost | Next |
|---|---|---|
| Net1 | 2 | A |
| Net2 | 2 | A |
| Net3 | 3 | A |
| Net4 | 1 | — |
| Net5 | 1 | A |
| Net6 | 3 | A |
| Net7 | 3 | A |

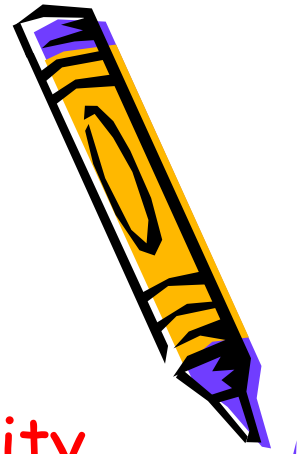
E

| Dest | Cost | Next |
|---|---|---|
| Net1 | 3 | C |
| Net2 | 3 | C |
| Net3 | 3 | C |
| Net4 | 3 | C |
| Net5 | 2 | C |
| Net6 | 2 | C |
| Net7 | 1 | — |

## Figure 11.8 *Two-node instability*



# Two-Node Instability (1)

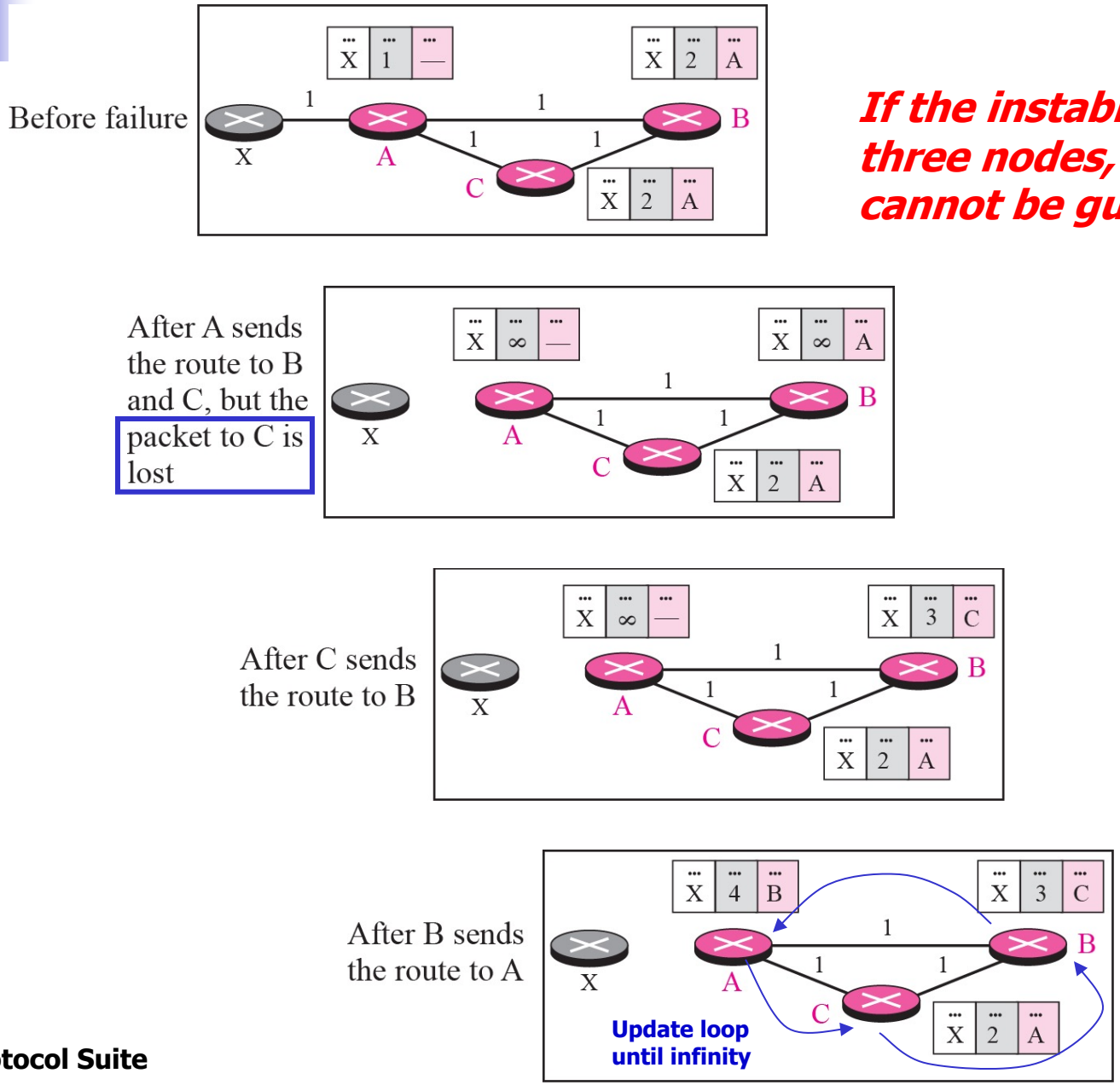


- Defining Infinity
  - Most implementations define 16 as infinity
- Split Horizon
  - Instead of flooding the table through each interface, each node sends only part of its table through each interface
  - E.g. node B thinks that the optimum route to reach X is via A, it does not need to advertise this piece of information to A

# Two-Node Instability (2)

- Split Horizon and Poison Reverse
  - One drawback of Split Horizon
    - Normally, the DV protocol uses a timer and if there is no news about a route, the node deletes the route from its table
    - In the previous e.g., node A cannot guess that this is due to split horizon or because B has not received any news about X recently
  - Poison Reverse
    - Node B can still advertise the value for X, but is the source of information is A, it can replace the distance with infinity as a warning

## Figure 11.9 *Three-node instability*




***If the instability is btw three nodes, stability cannot be guaranteed***

# 11-4 RIP

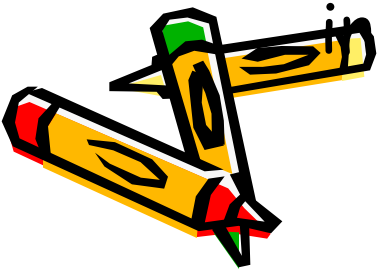
The Routing Information Protocol (RIP) is an intra-domain (interior) routing protocol used inside an autonomous system. It is a very simple protocol based on distance vector routing. RIP implements distance vector routing directly with some considerations.

# *Topics Discussed in the Section*

- ✓ RIP Message Format
- ✓ Request and Response
- ✓ Timers in RIP
- ✓ RIP Version 2
- ✓ Encapsulation

# RIP messages

- Request
  - A request message is sent by a router that has just come up or by a router that has some time-out entries
  - A request can ask about specific entries or all entries
- Response
  - A response can be either solicited or unsolicited (30s or when there is a change in the routing table)



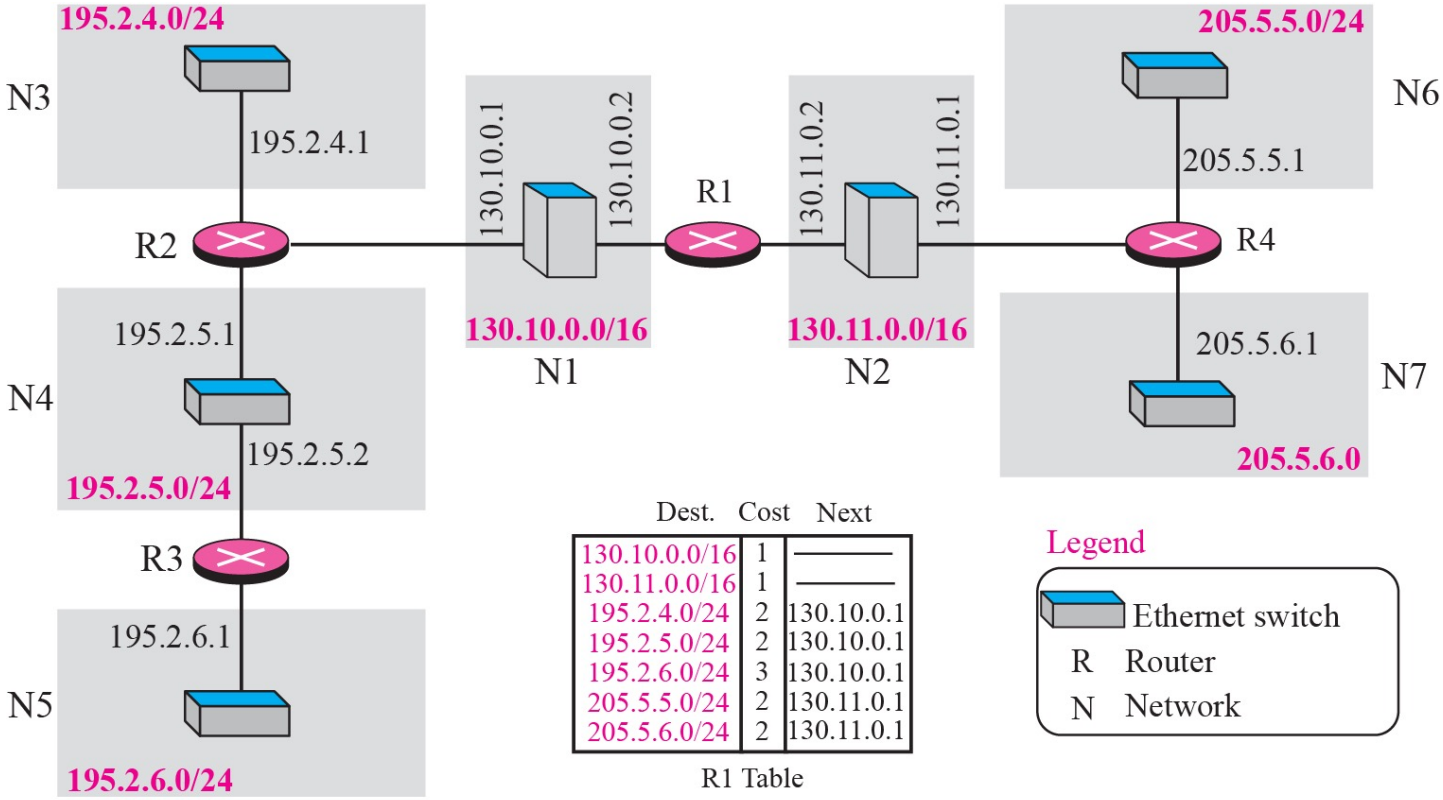
# RIP Timers

- Periodic timer
  - It controls the advertising of regular update message (25 ~ 30 sec)
- Expiration timer
  - It governs the validity of a route (180 sec)
  - The route is considered expired and the hop count of the route is set to 16
- Garbage collection timer
  - A invalid route is not purged from the routing table until this timer expires (120 sec)

# RIPv2 vs. RIPv1

- Classless Addressing
- Authentication
- Multicasting
  - RIPv1 uses broadcasting to send RIP messages to every neighbors. Routers as well as hosts receive the packets
  - RIPv2 uses the all-router multicast address to send the RIP messages only to RIP routers in the network

## Figure 11.10 *Example of a domain using RIP*




R1 Table

| Dest. | Cost | Next |
|---|---|---|
| 130.10.0.0/16 | 1 | ——— |
| 130.11.0.0/16 | 1 | ——— |
| 195.2.4.0/24 | 2 | 130.10.0.1 |
| 195.2.5.0/24 | 2 | 130.10.0.1 |
| 195.2.6.0/24 | 3 | 130.10.0.1 |
| 205.5.5.0/24 | 2 | 130.11.0.1 |
| 205.5.6.0/24 | 2 | 130.11.0.1 |

R2 Table

| Dest. | Cost | Next |
|---|---|---|
| 130.10.0.0/16 | 1 | ——— |
| 130.11.0.0/16 | 2 | 130.10.0.2 |
| 195.2.4.0/24 | 1 | ——— |
| 195.2.5.0/24 | 1 | ——— |
| 195.2.6.0/24 | 2 | 195.2.5.2 |
| 205.5.5.0/24 | 3 | 130.10.0.2 |
| 205.5.6.0/24 | 3 | 130.10.0.2 |

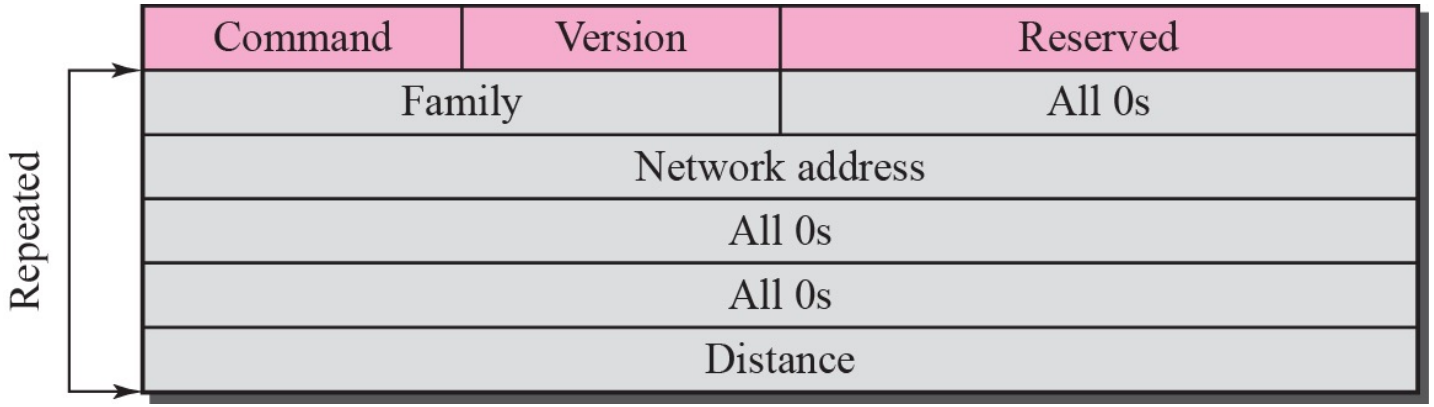
R3 Table

| Dest. | Cost | Next |
|---|---|---|
| 130.10.0.0/16 | 2 | 195.2.5.1 |
| 130.11.0.0/16 | 3 | 195.2.5.1 |
| 195.2.4.0/24 | 2 | 195.2.5.1 |
| 195.2.5.0/24 | 1 | ——— |
| 195.2.6.0/24 | 1 | ——— |
| 205.5.5.0/24 | 4 | 195.2.5.1 |
| 205.5.6.0/24 | 4 | 195.2.5.1 |

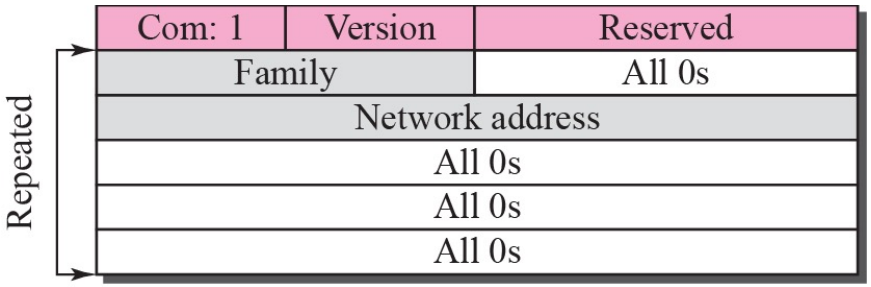
R4 Table

| Dest. | Cost | Next |
|---|---|---|
| 130.10.0.0/16 | 2 | 130.11.0.2 |
| 130.11.0.0/16 | 1 | ——— |
| 195.2.4.0/24 | 3 | 130.11.0.2 |
| 195.2.5.0/24 | 3 | 130.11.0.2 |
| 195.2.6.0/24 | 4 | 130.11.0.2 |
| 205.5.5.0/24 | 1 | ——— |
| 205.5.6.0/24 | 1 | ——— |

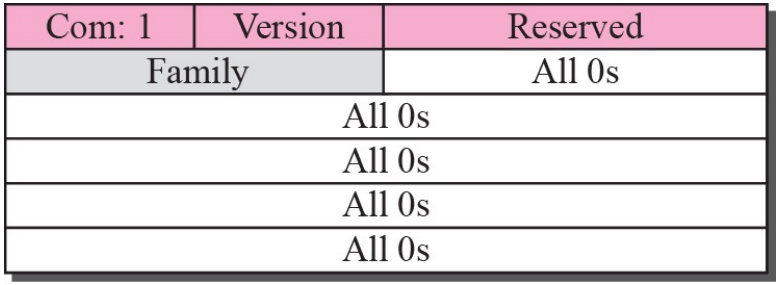
## Figure 11.11 *RIP message format*



## Figure 11.12 *Request messages*




a. Request for some



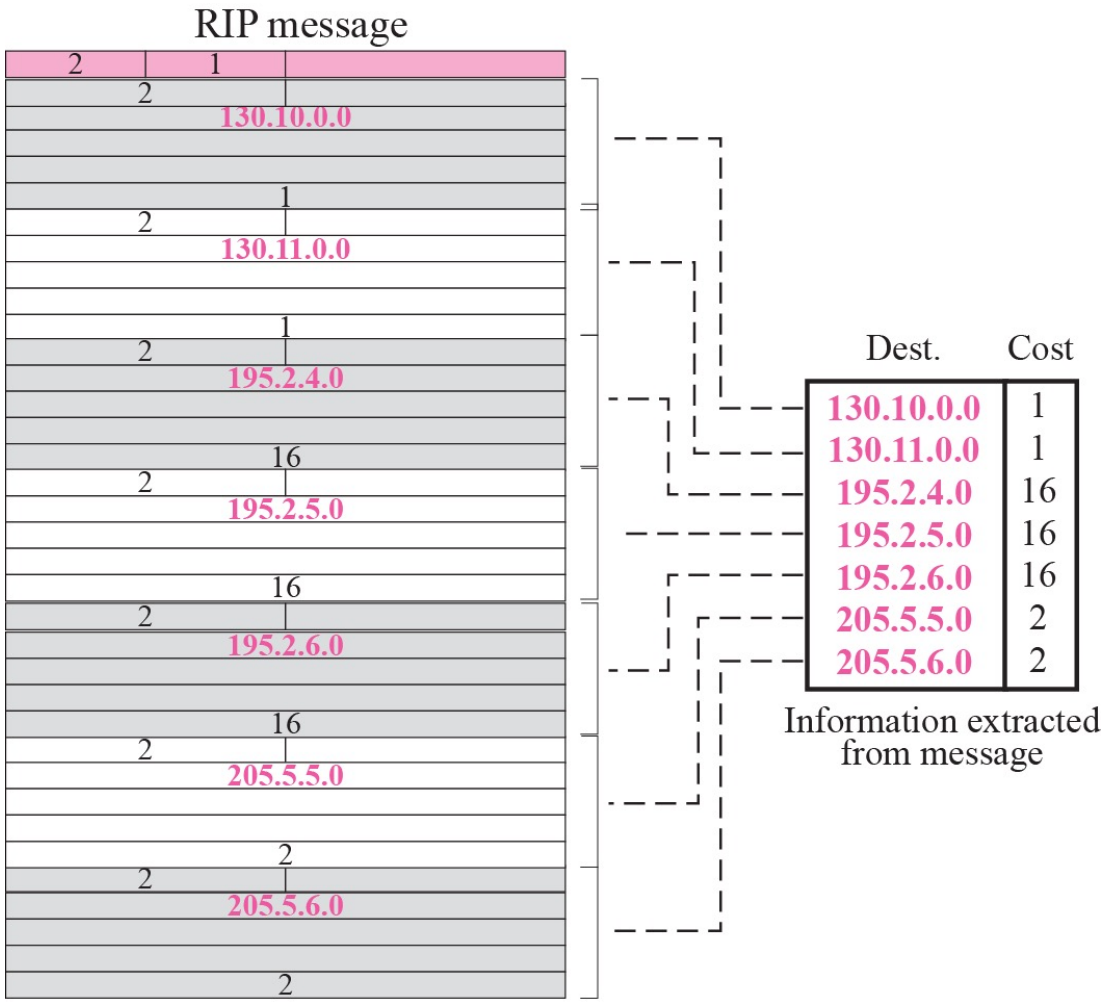

b. Request for all

# Example 11.4

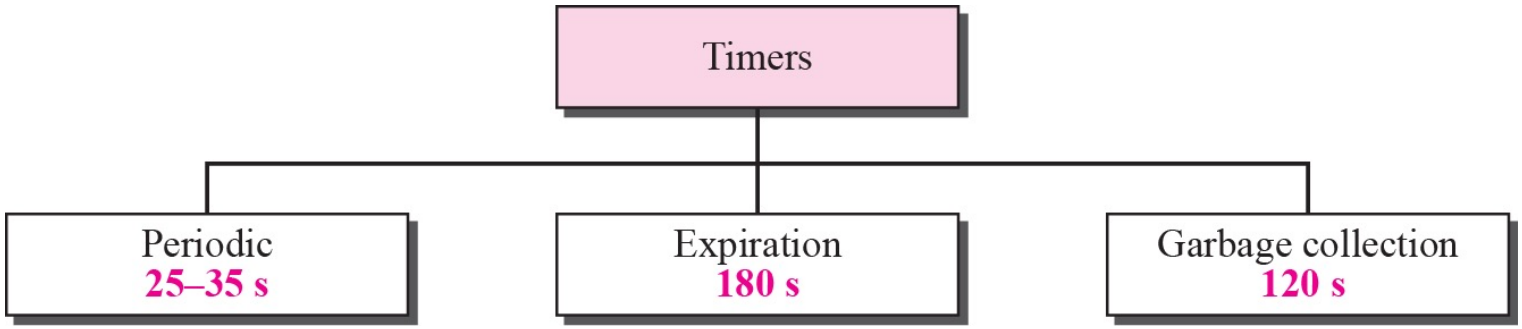
Figure 11.13 shows the update message sent from router R1 to router R2 in Figure 11.10. The message is sent out of interface 130.10.0.2.

The message is prepared with the combination of split horizon and poison reverse strategy in mind. Router R1 has obtained information about networks 195.2.4.0, 195.2.5.0, and 195.2.6.0 from router R2. When R1 sends an update message to R2, it replaces the actual value of the hop counts for these three networks with 16 (infinity) to prevent any confusion for R2. The figure also shows the table extracted from the message. Router R2 uses the source address of the IP datagram carrying the RIP message from R1 (130.10.02) as the next hop address. Router R2 also increments each hop count by 1 because the values in the message are relative to R1, not R2.

## Figure 11.13 *Solution to Example 11.4*



## Figure 11.14 *RIP timers*



# Example 11.5

A routing table has 20 entries. It does not receive information about five routes for 200 s. How many timers are running at this time?

*Solution*

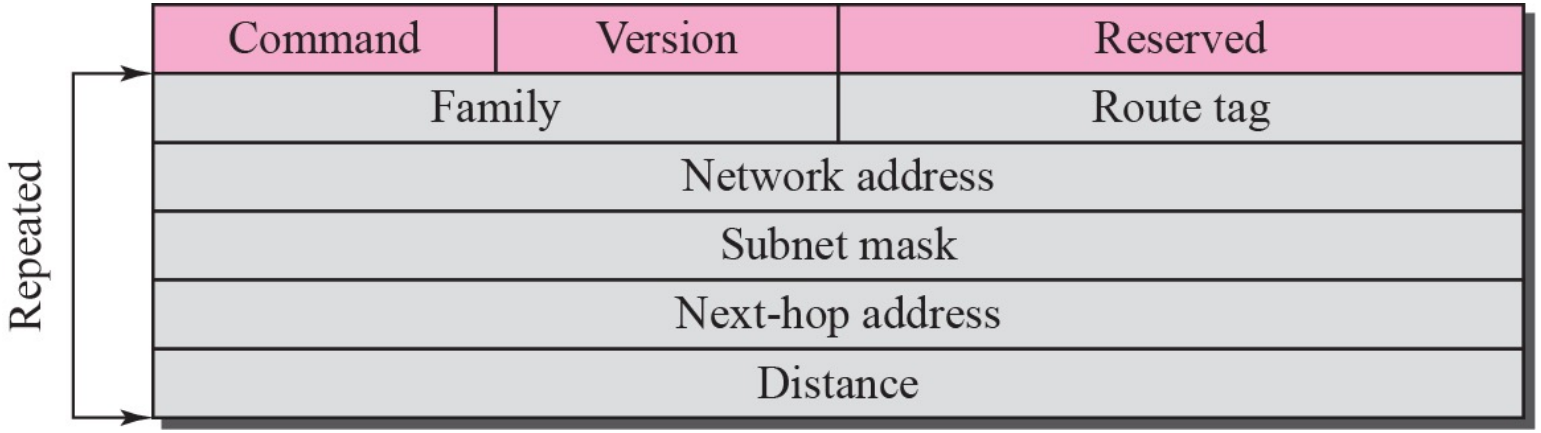
The 21 timers are listed below:

Periodic timer: 1

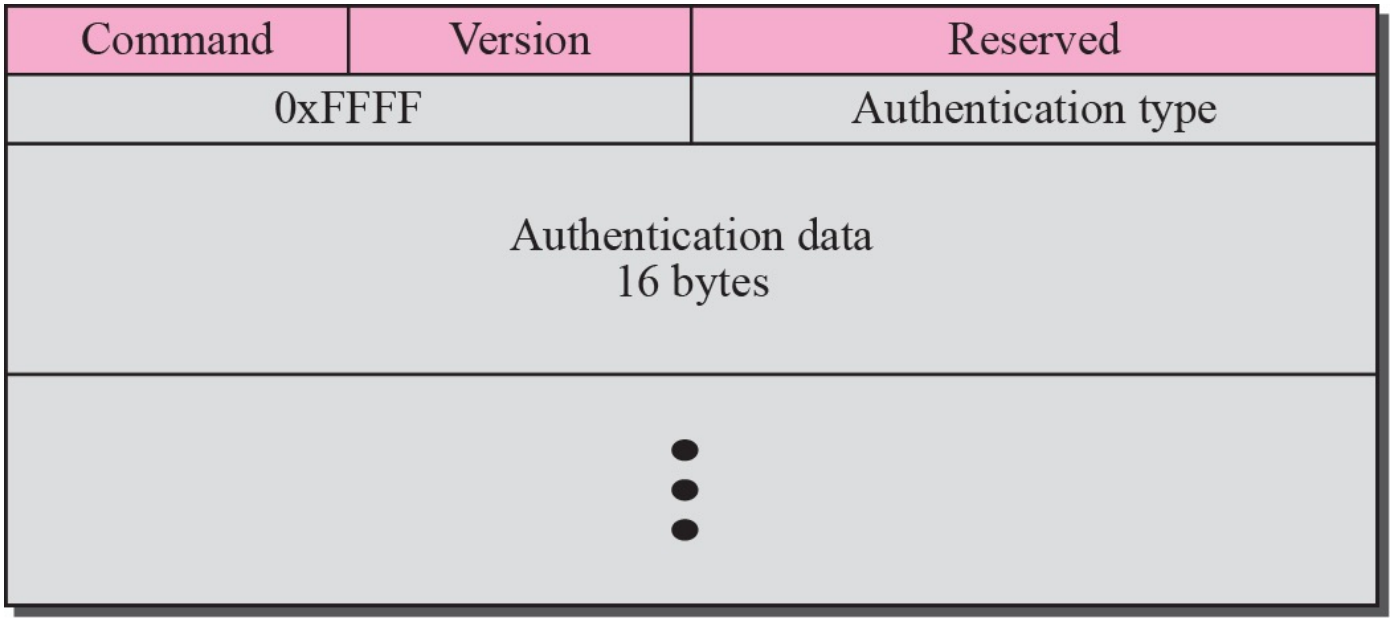
Expiration timer: 20 − 5 = 15

Garbage collection timer: 5

## Figure 11.15 *RIP version 2 format*



## Figure 11.16 *Authentication*



Note

***RIP uses the services of UDP on well-known port 520.***

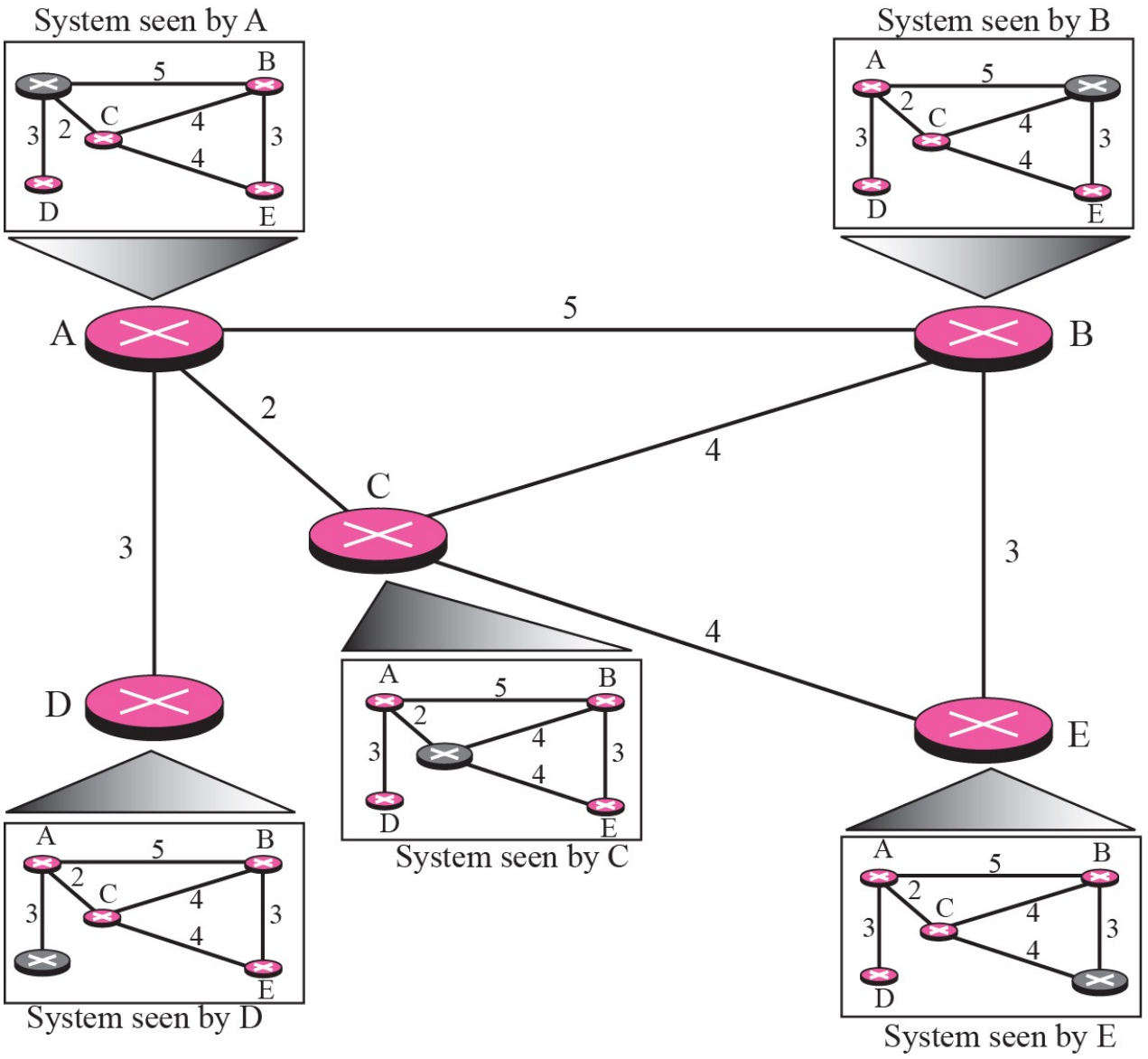
# 11-5 LINK STATE ROUTING

Link state routing has a different philosophy from that of distance vector routing. In link state routing, if each node in the domain has the entire topology of the domain—the list of nodes and links, how they are connected including the type, cost (metric), and the condition of the links (up or down)—the node can use the Dijkstra algorithm to build a routing table.

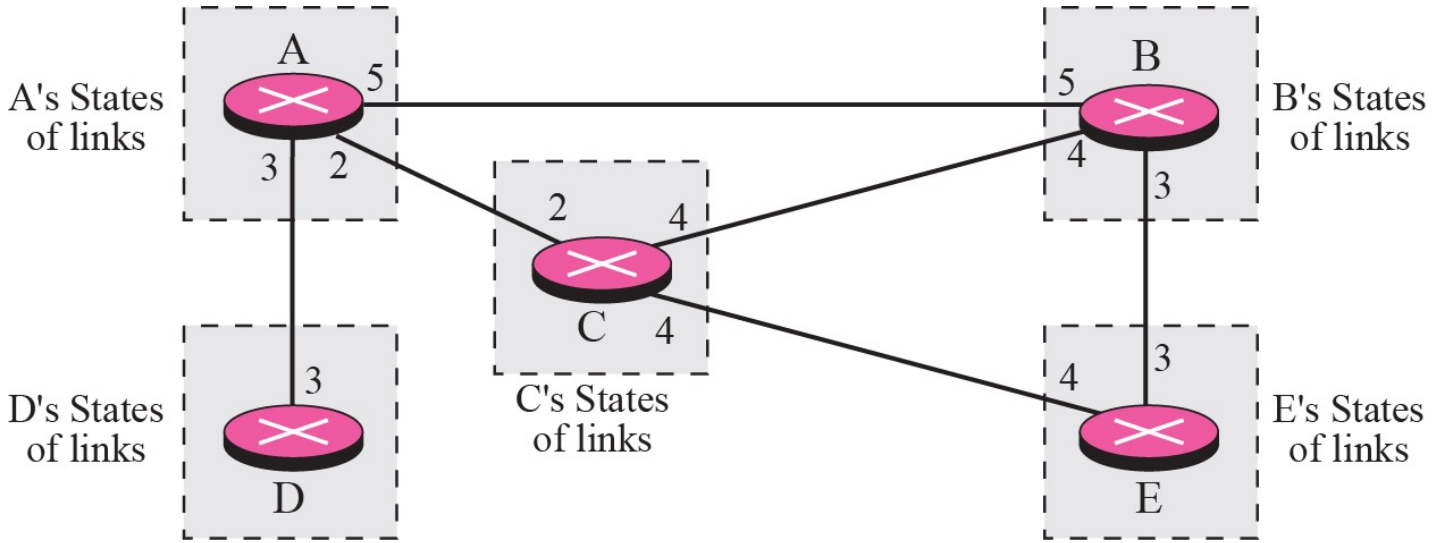
# *Topics Discussed in the Section*

- ✓ **Building Routing tables**

## Figure 11.17 *Concept of Link state routing*



## Figure 11.18 *Link state knowledge*



# Building Routing Tables

- Creation of the states of the links by each node, called the link state packets (LSP)
- Dissemination of LSPs to every other routers, called flooding (efficiently)
- Formation of a shortest path tree for each node
- Calculation of a routing table based on the shortest path tree

# Creation of LSP

- LSP data: E.g. the node ID, the list of links, a sequence number, and age.
- LSP Generation
  - When there is a change in the topology of the domain
  - On a periodic basis
    - There is no actual need for this type of LSP, normally 60 minutes or 2 hours

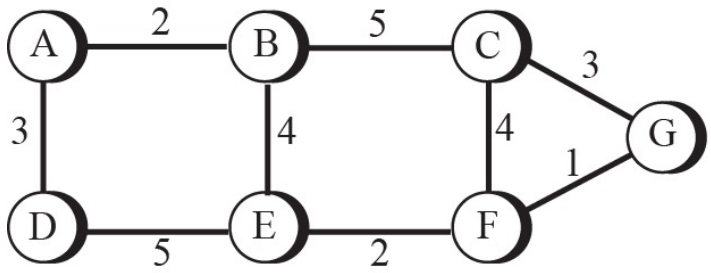
**Table 11.3** *Dijkstra's Algorithm*

```
1  Dijkstra ( )
2  {
3      // Initialization
4      Path = {s}              // s means self
5      for (i = 1 to N)
6      {
7          if(i is a neighbor of s and i ≠ s)    D_i = c_si
8          if (i is not a neighbor of s)         D_i = ∞
9      }
10     D_s = 0
11
12 } // Dijkstra
```

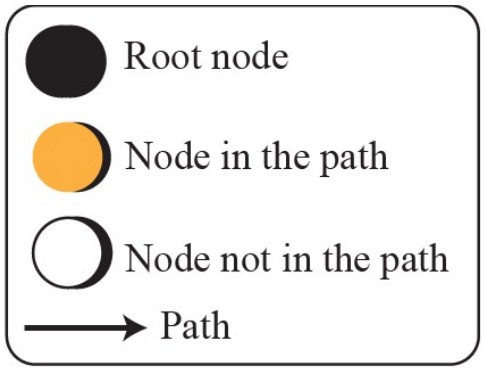
## Continued

```
13    // Iteration
14    Repeat
15    {
16        // Finding the next node to be added
17     Path = Path ∪ i   if D_i is minimum among all remaining nodes
18
19       // Update the shortest distance for the rest
20     for (j = 1 to M)     // M number of remaining nodes
21     {
22          D_j = minimum (D_j ,  D_j  + c_ij)
23      }
24    } until (all nodes included in the path, M = 0)
25
```

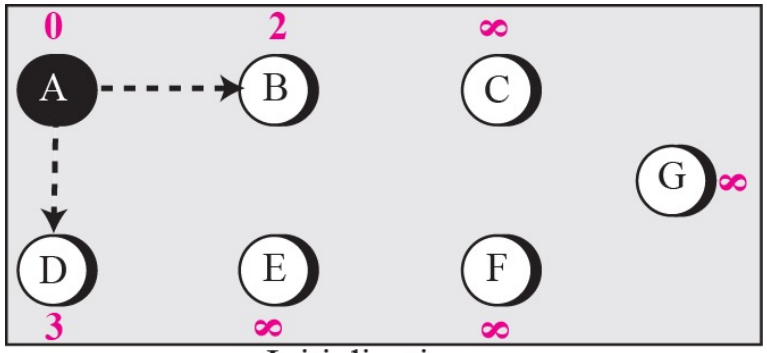
## Figure 11.19 *Forming shortest path three for router A in a graph*




Topology

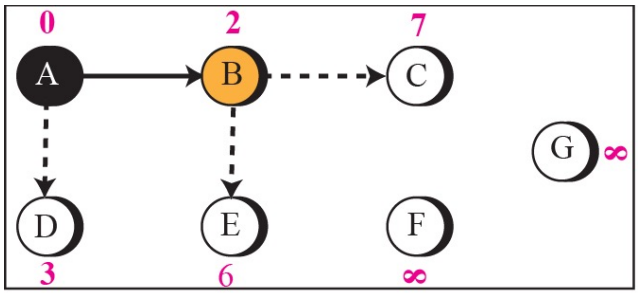



Legend

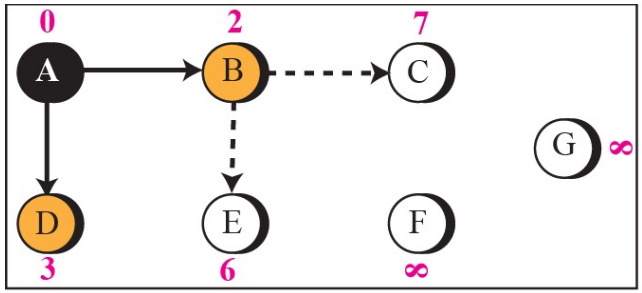



Initialization

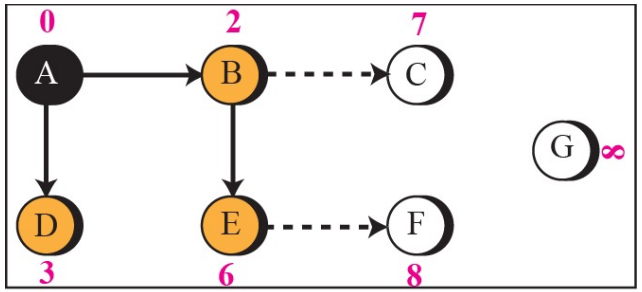
# Figure 11.19 *Continued*




Iteration 1

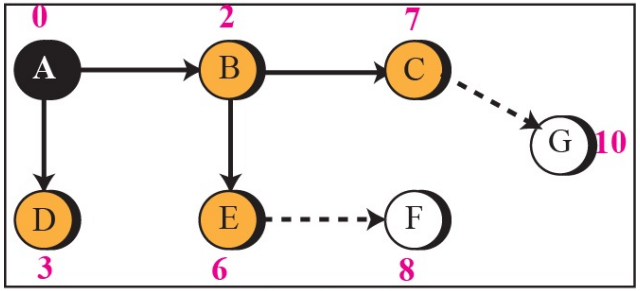



Iteration 2

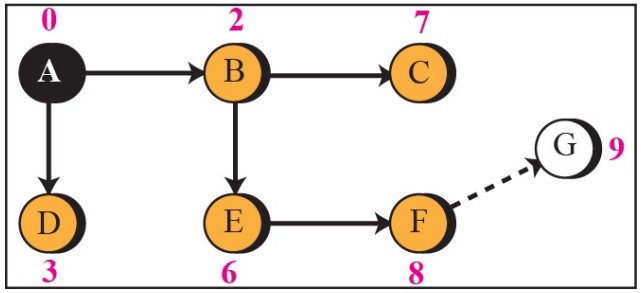



Iteration 3

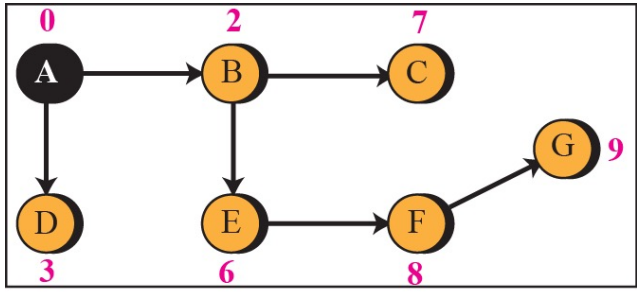
# Figure 11.19 *Continued*




Iteration 4




Iteration 5




Iteration 6

# Example 11.6

To show that the shortest path tree for each node is different, we found the shortest path tree as seen by node C (Figure 11.20). We leave the detail as an exercise.

## Figure 11.20 *Example 11.6*

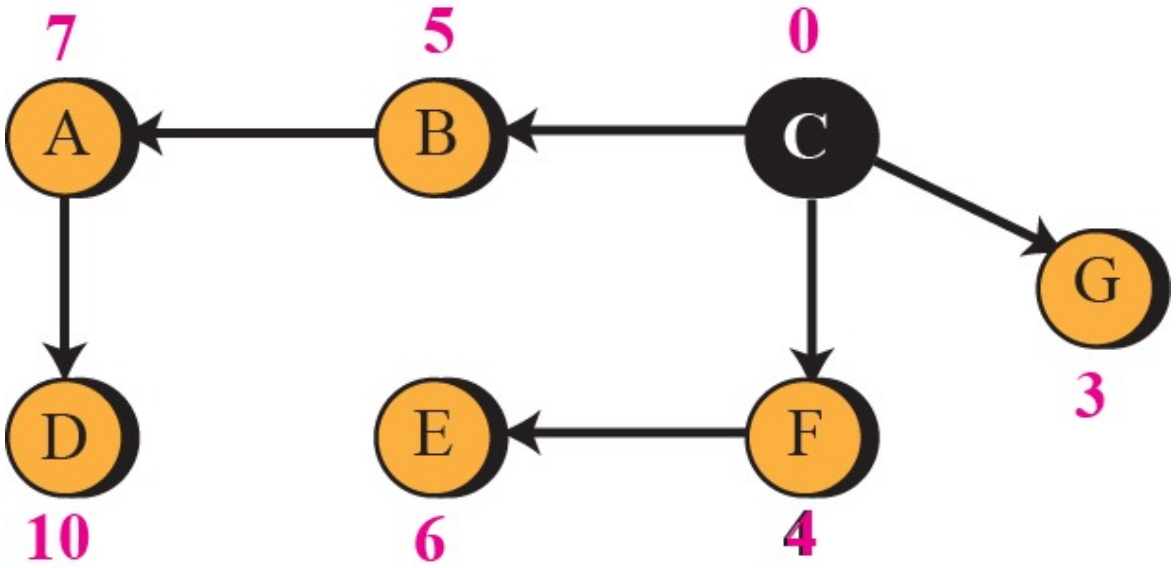


**Table 11.4** *Routing Table for Node A*

| *Destination* | *Cost* | *Next Router* |
|---|---|---|
| A | 0 | — |
| B | 2 | — |
| C | 7 | B |
| D | 3 | — |
| E | 6 | B |
| F | 8 | B |
| G | 9 | B |

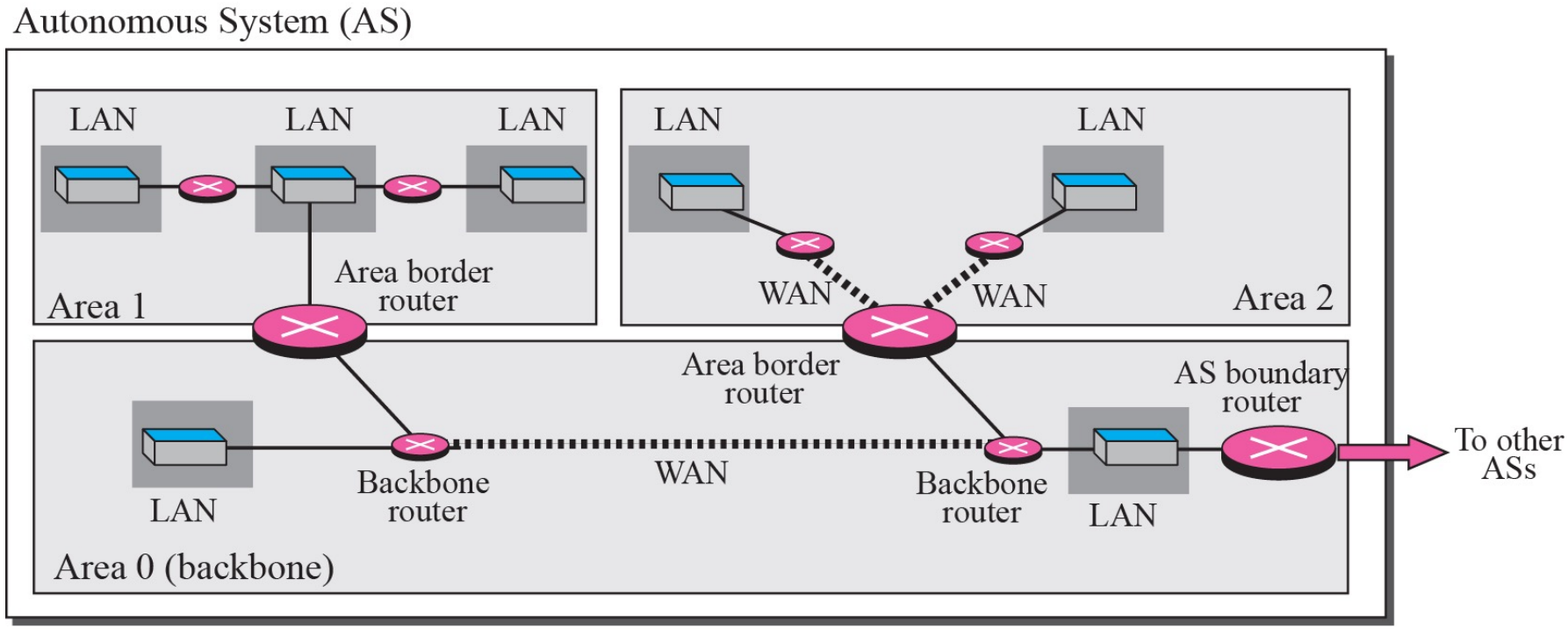
# 11-6 OSPF

The Open Shortest Path First (OSPF) protocol is an intra-domain routing protocol based on link state routing. Its domain is also an autonomous system.

# *Topics Discussed in the Section*

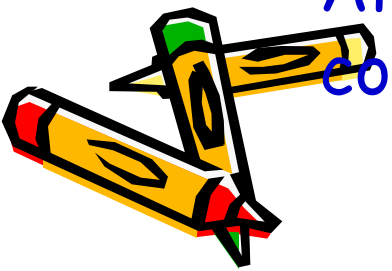
- ✓ **Area**
- ✓ **Metric**
- ✓ **Types of Links**
- ✓ **Graphical Representation**
- ✓ **OSPF Packets**
- ✓ **Link State Update Packet**
- ✓ **Other Packets**
- ✓ **Encapsulation**

## Figure 11.21 *Areas in an autonomous system*



# Area in OSPF (1)

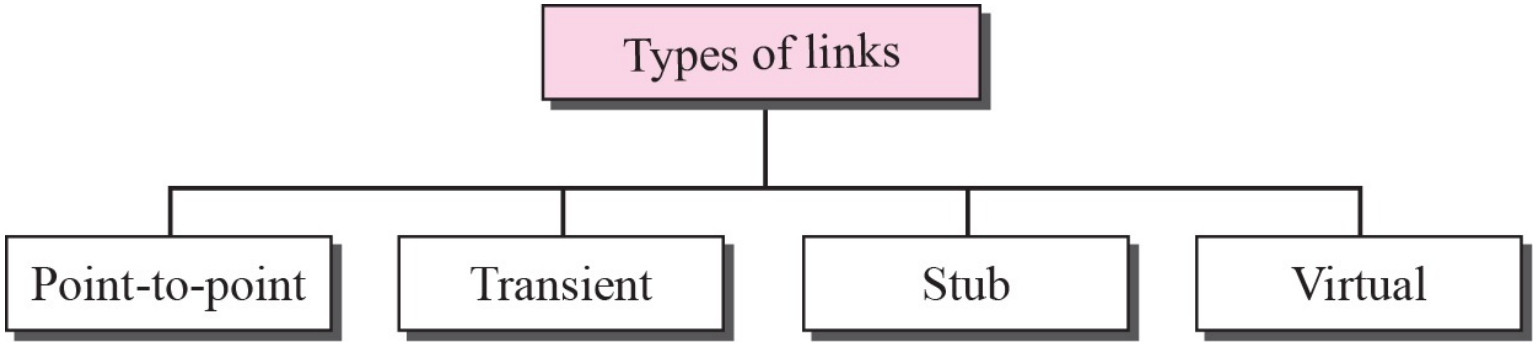
- A collection of networks with area ID
- Routers inside an area flood the area with routing information
- Area border routers summarize the information about the area and send it to other areas
- Backbone area and backbone routers
  - All of the area inside an AS must be connected to the backbone



# Area in OSPF (2)

- Virtual link
  - If, because of some problem, the connectivity between a backbone and an area is broken, a virtual link between routers must be created by the administration to allow continuity of the functions of the backbone as the primary area

## Figure 11.22 *Types of links*

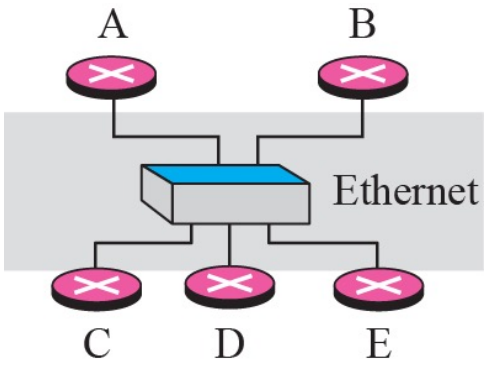


## Figure 11.23 *Point-to-point link*

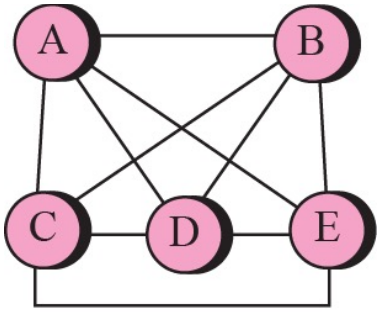

a. Point-to-point network


b. Representation

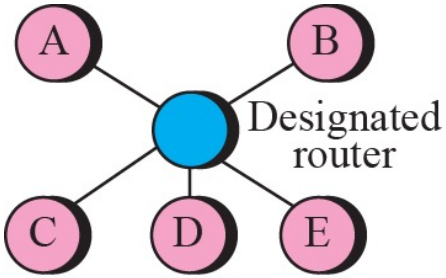
## Figure 11.24 *Transient link*




a. Transient network

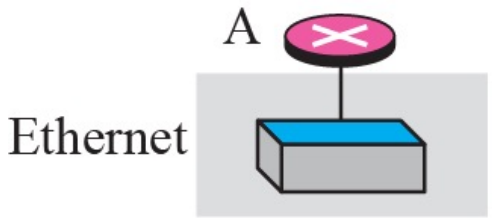



b. Unrealistic

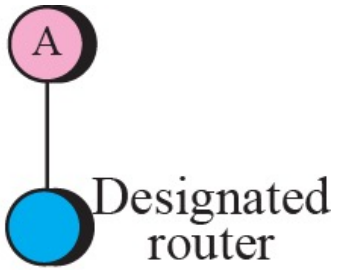



c. Realistic

## Figure 11.25 *Stub link*

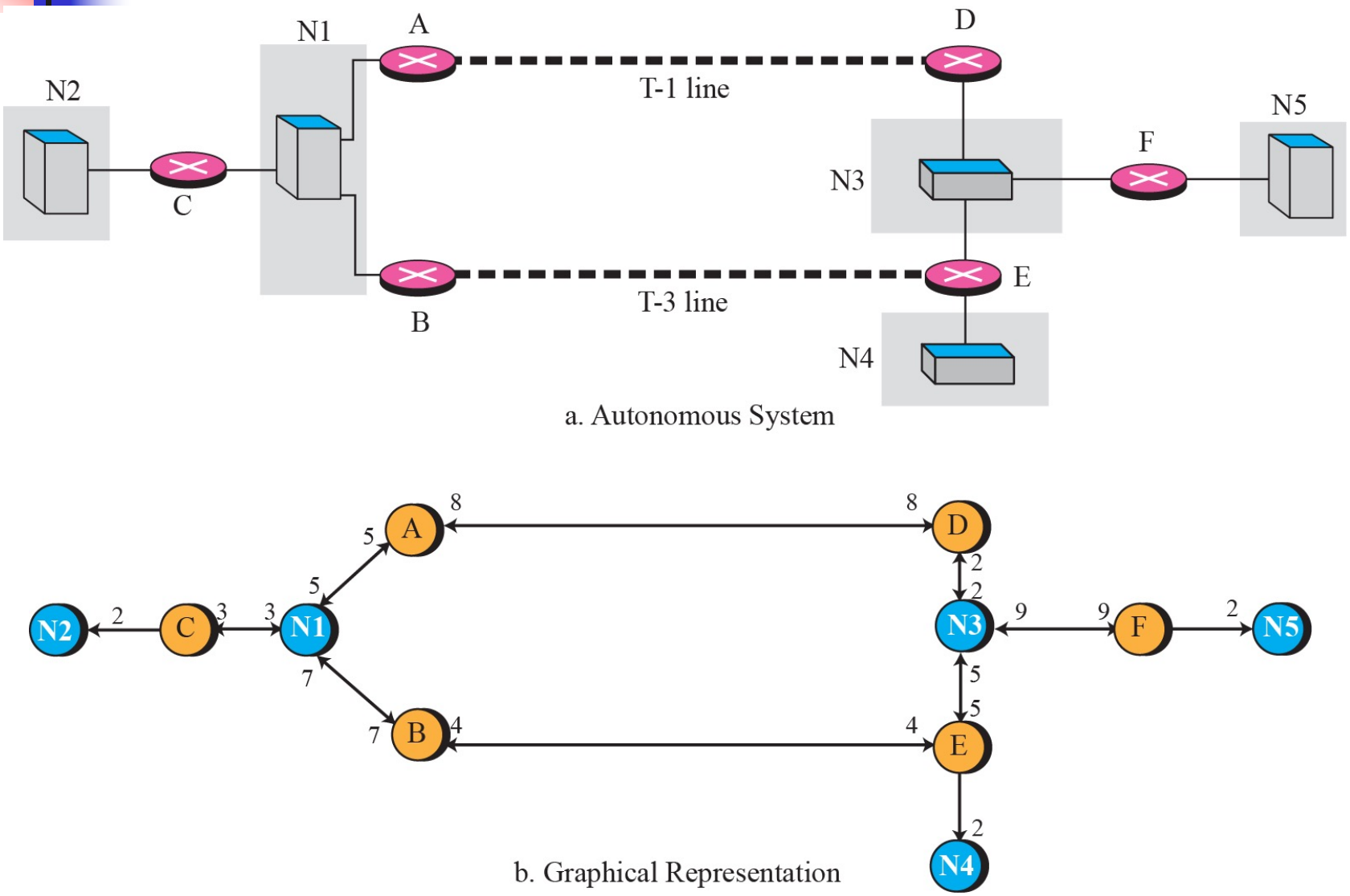



a. Stub network

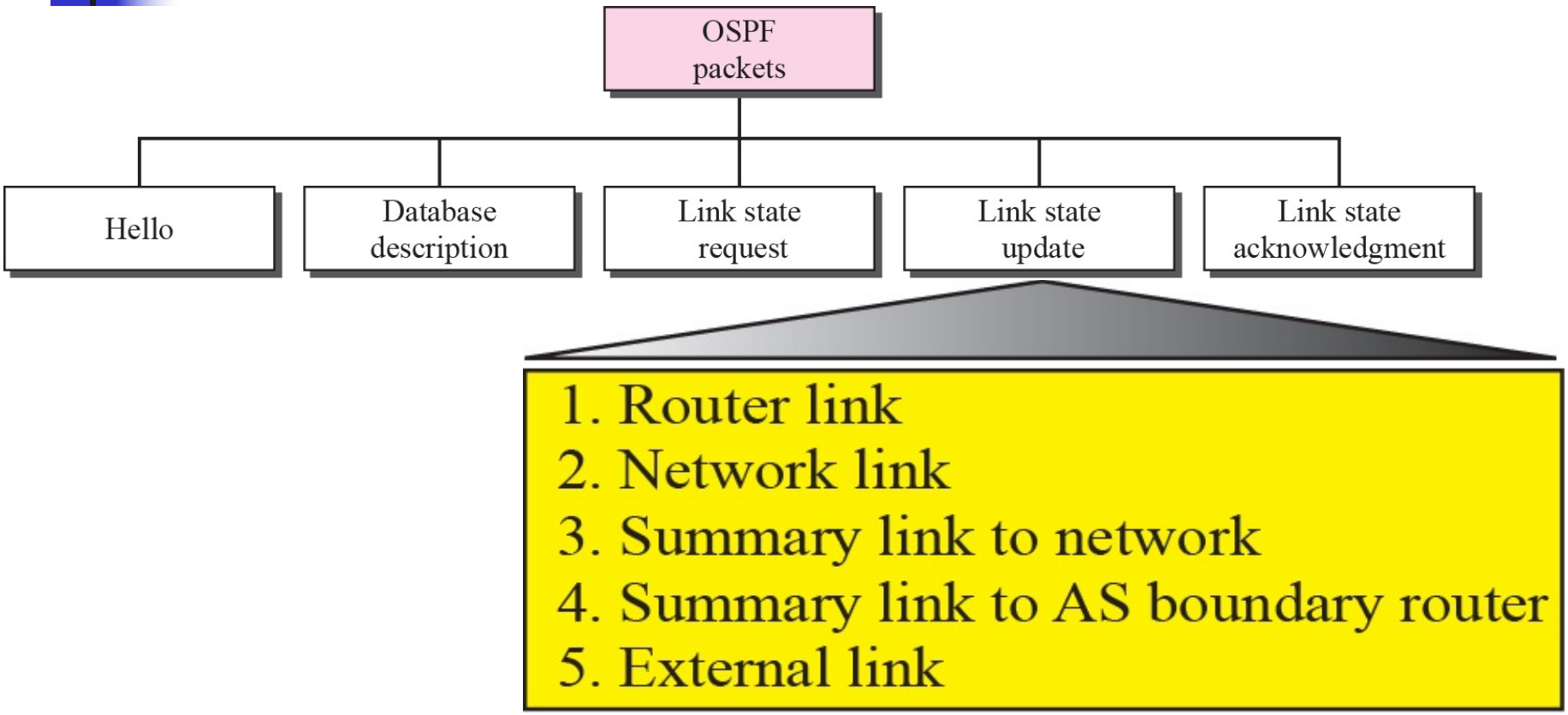



b. Representation

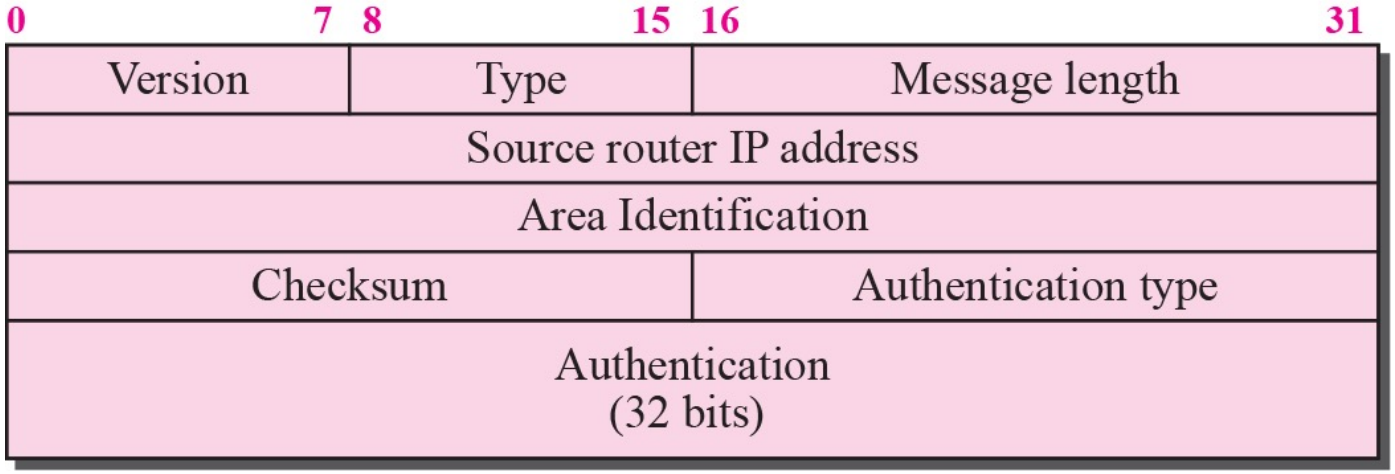
## Figure 11.26 *Example of an AS and its graphical representation in OSPF*



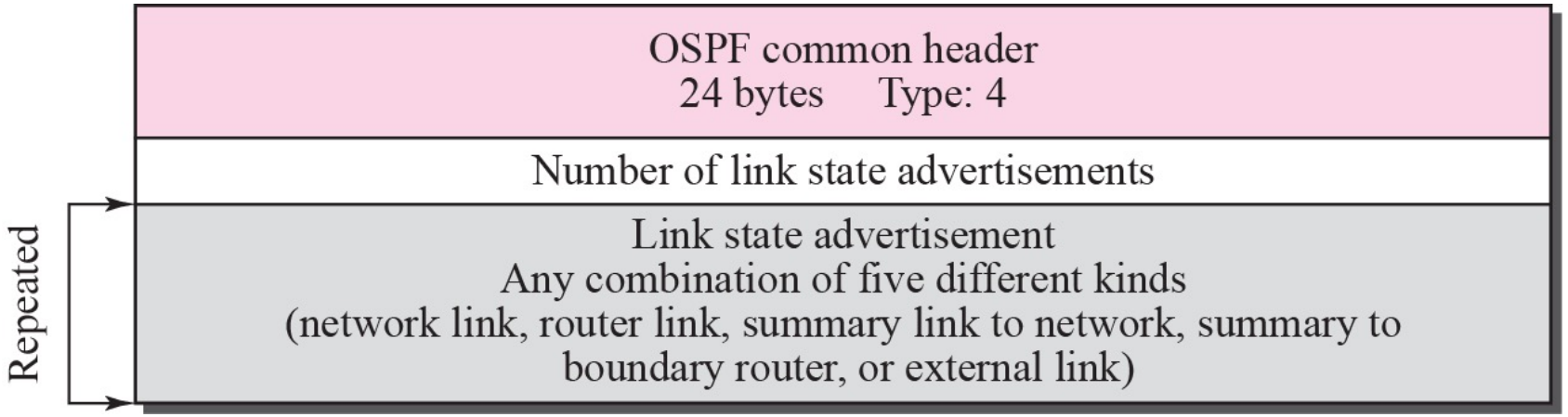
## Figure 11.27 *Types of OSPF packet*



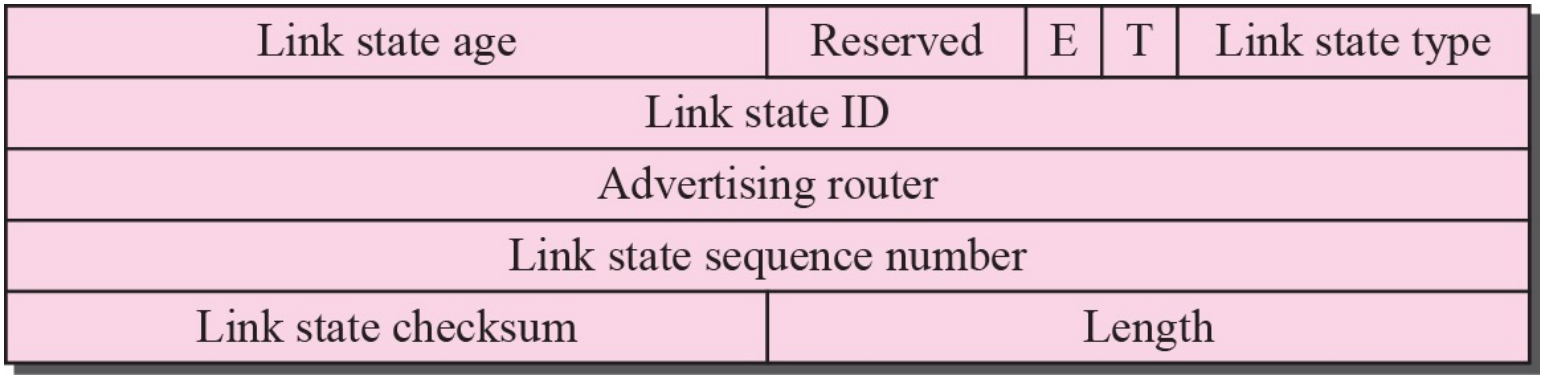
## Figure 11.28 *OSPF common header*



## Figure 11.29 *Link state update packet*



## Figure 11.30 *LSA general header*



# LSA General Header (1)

- Link state age
  - When a router creates the message, the value of this field is 0
  - When each successive router forwards this message, it estimates the transit time and adds it to the cumulative value of this field

# LSA General Header (2)

- E flag
  - If this flag is set to 1, it means the area is a stub area (an area that is connected to the backbone area by only one path
- T flag
  - If this flag is set to 1, it means the router can handle multiple types of services

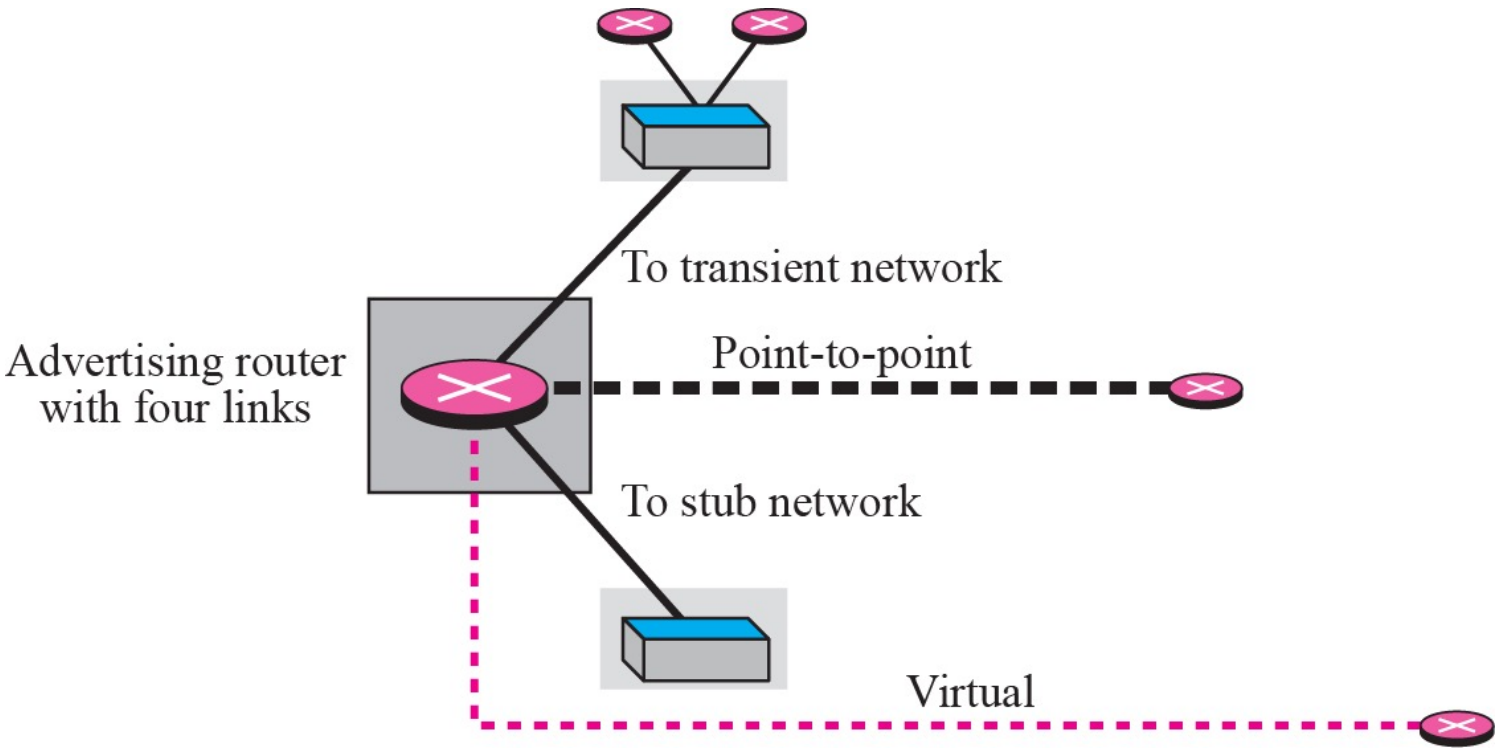
# LSA General Header (3)

- Advertising router
  - The IP address of the router advertising this message
- Link state sequence number
  - A sequence number assigned to each link state update message

# LS Type and LS ID

| Link state type | Link state ID |
|---|---|
| Router link | IP address of the router |
| Network link | IP address of the designated router |
| Summary link to network | Address of the network |
| Summary link to AS boundary | IP address of the boundary router |
| External link | Address of the network |

## Figure 11.31 *Router link*



## Figure 11.32 *Router link LSA*

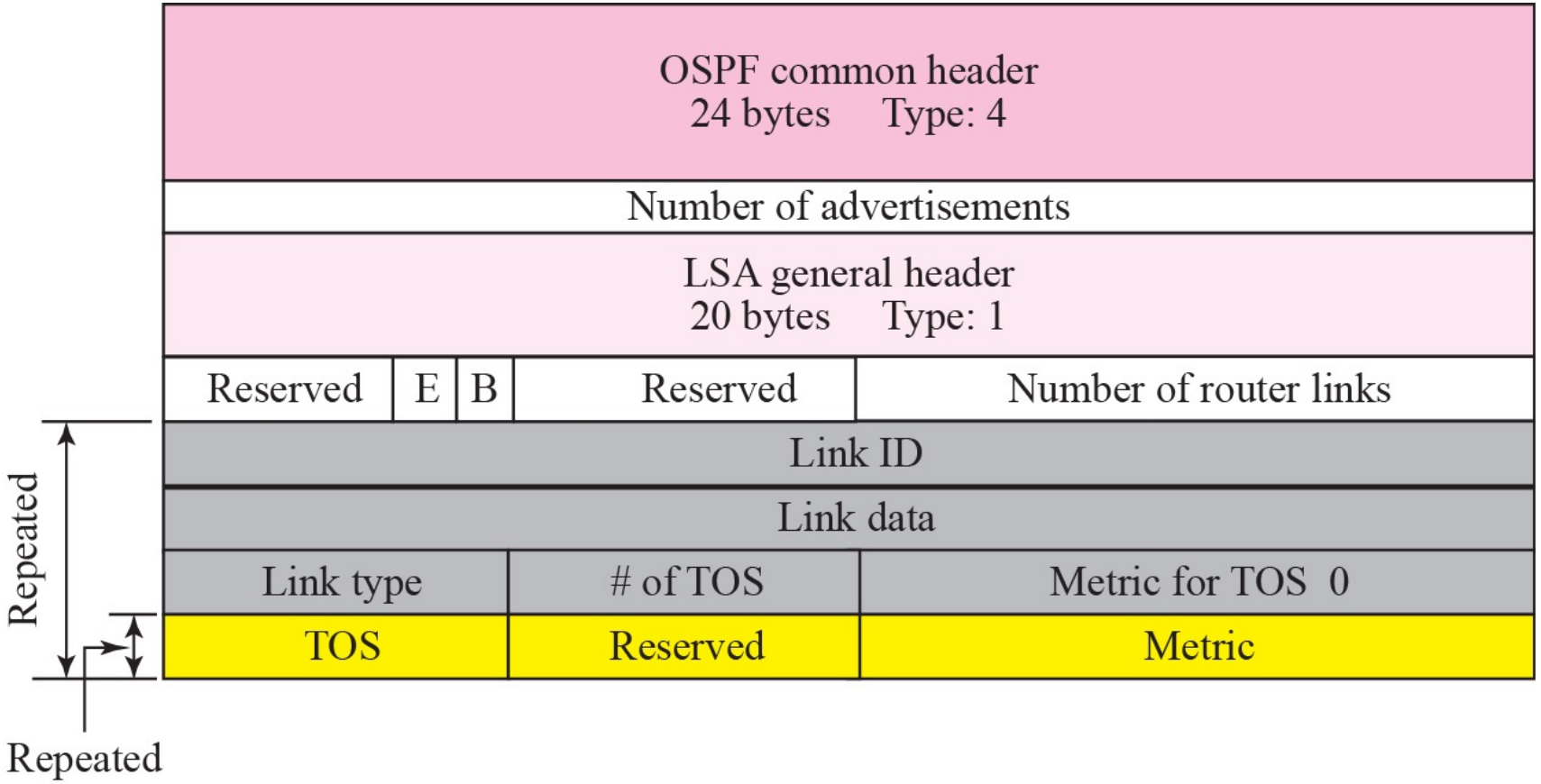


**Table 11.5** *Link Types, Link Identification, and Link Data*

| *Link Type* | *Link Identification* | *Link Data* |
|---|---|---|
| Type 1: Point-to-point | Address of neighbor router | Interface number |
| Type 2: Transient | Address of designated router | Router address |
| Type 3: Stub | Network address | Network mask |
| Type 4: Virtual | Address of neighbor router | Router address |

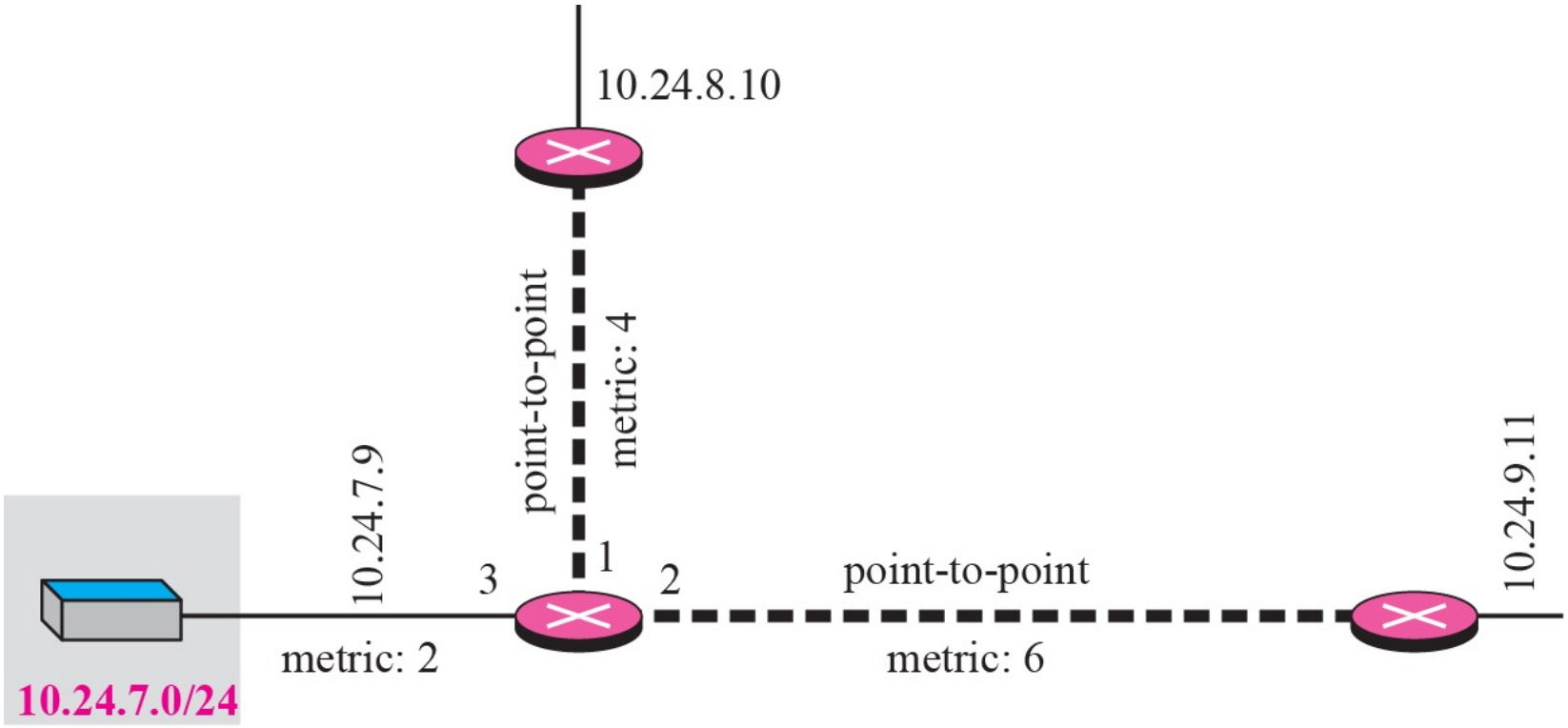
# Example 11.7

Figure 11.7 shows the final routing tables for routers in Figure 11.5.

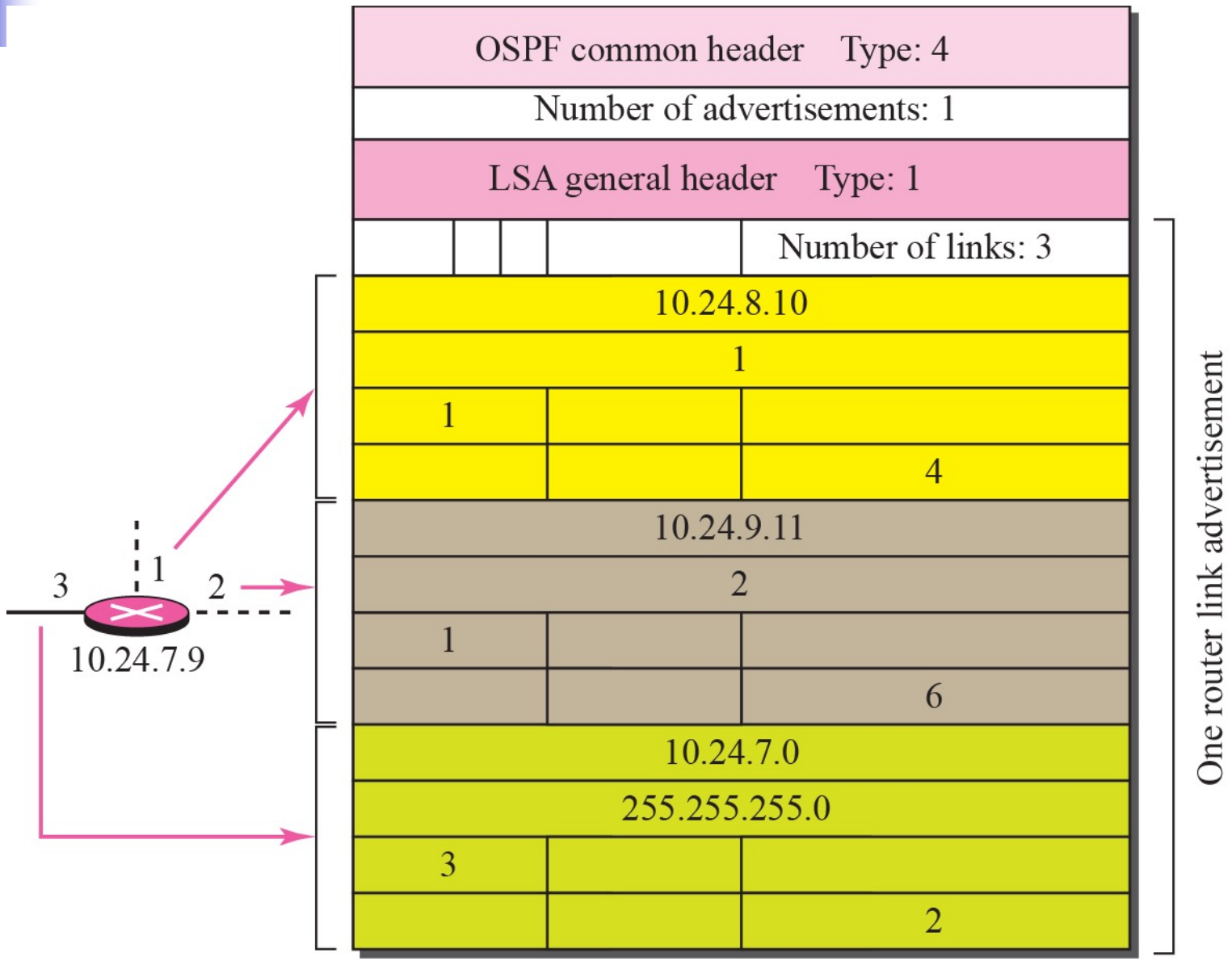
*Solution*

This router has three links: two of type 1 (point-to-point) and one of type 3 (stub network). Figure 11.34 shows the router link LSA.

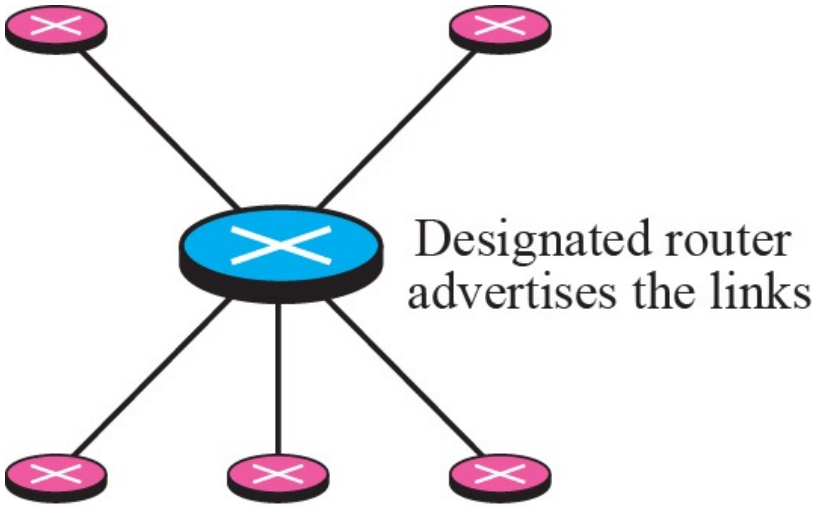
## Figure 11.33 *Example 11.7*



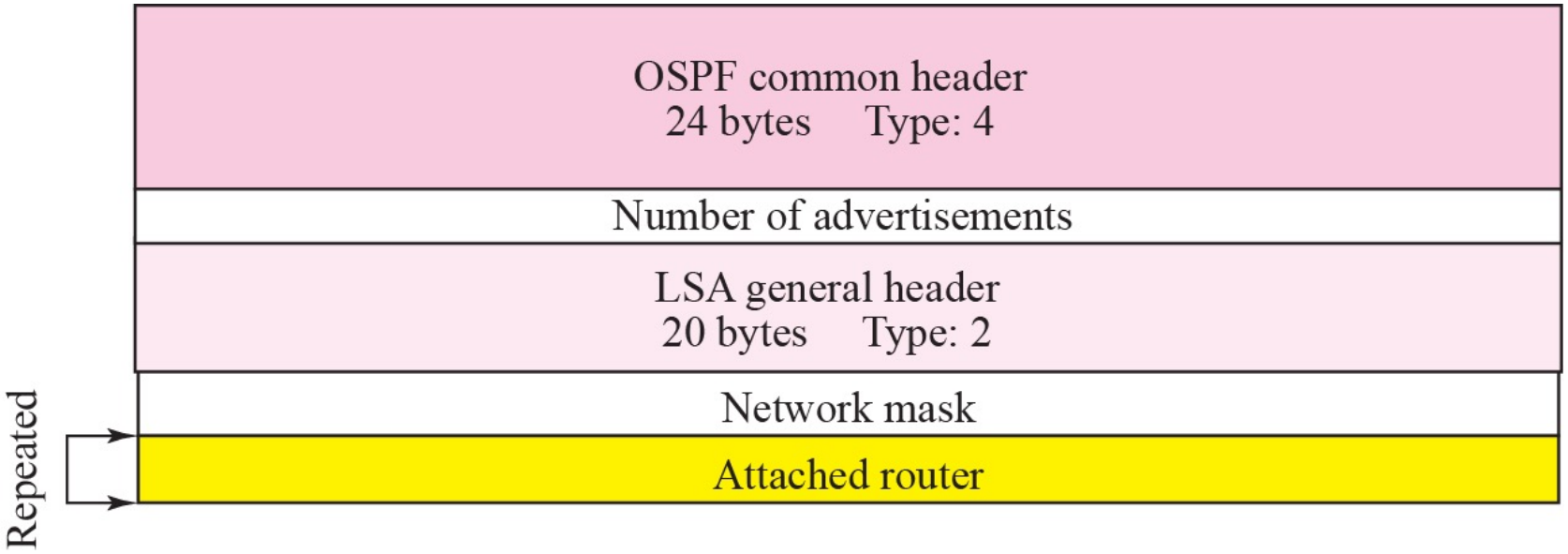
## Figure 11.34 *Solution to Example 11.7*



## Figure 11.35 *Network link*



## Figure 11.36 *Network link advertisement format*



# Example 11.8

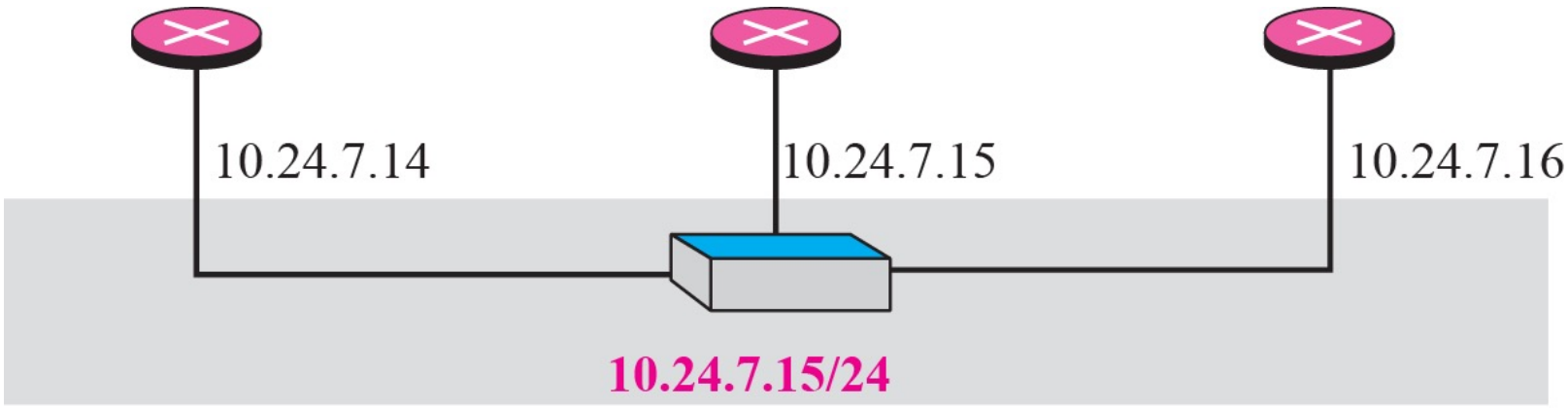
Give the network link LSA in Figure 11.37.

*Solution*

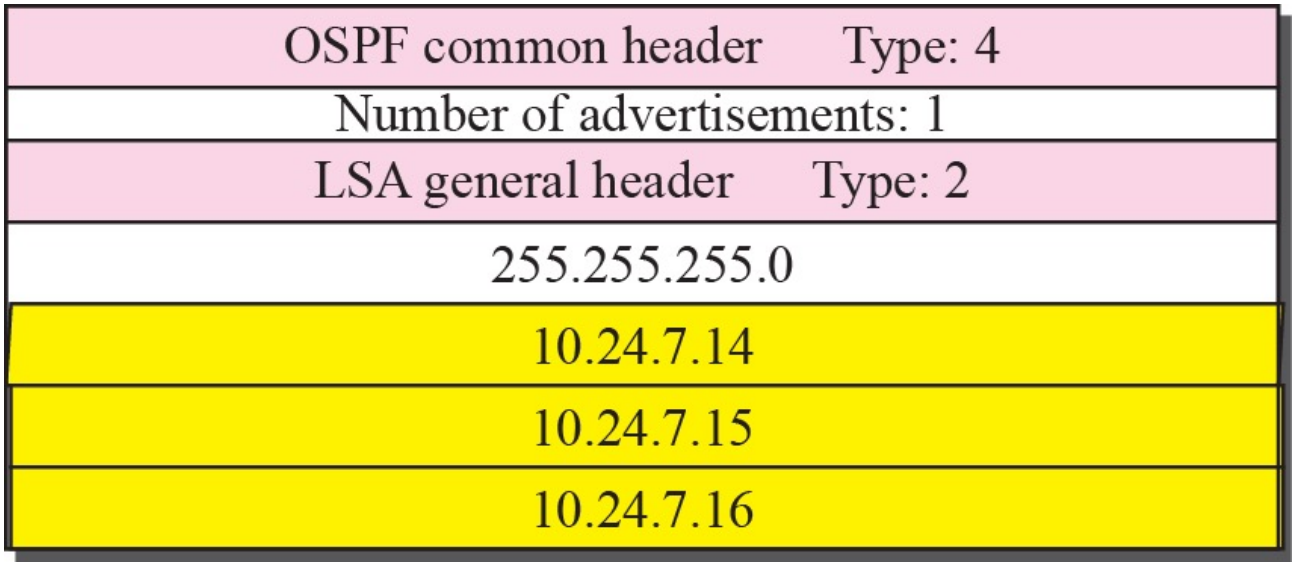
Solution

The network for which the network link advertises has three routers attached. The LSA shows the mask and the router addresses. Figure 11.38 shows the network link LSA.

## Figure 11.37 *Example 11.8*



## Figure 11.38 *Solution to Example 11.8*



# Example 11.9

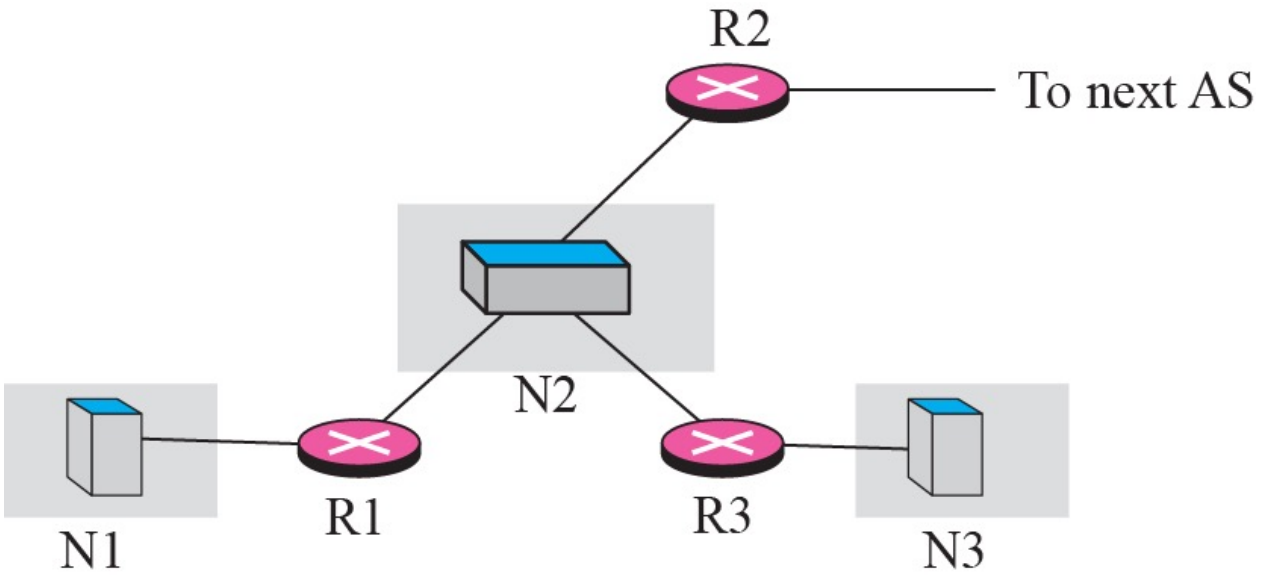
In Figure 11.39, which router(s) sends out router link LSAs?

*Solution*

All routers advertise router link LSAs.

a. R1 has two links, N1 and N2.
b. R2 has one link, N1.
c. R3 has two links, N2 and N3.

## Figure 11.39 *Examples 11.9 and 11.10*



# Example 11.10

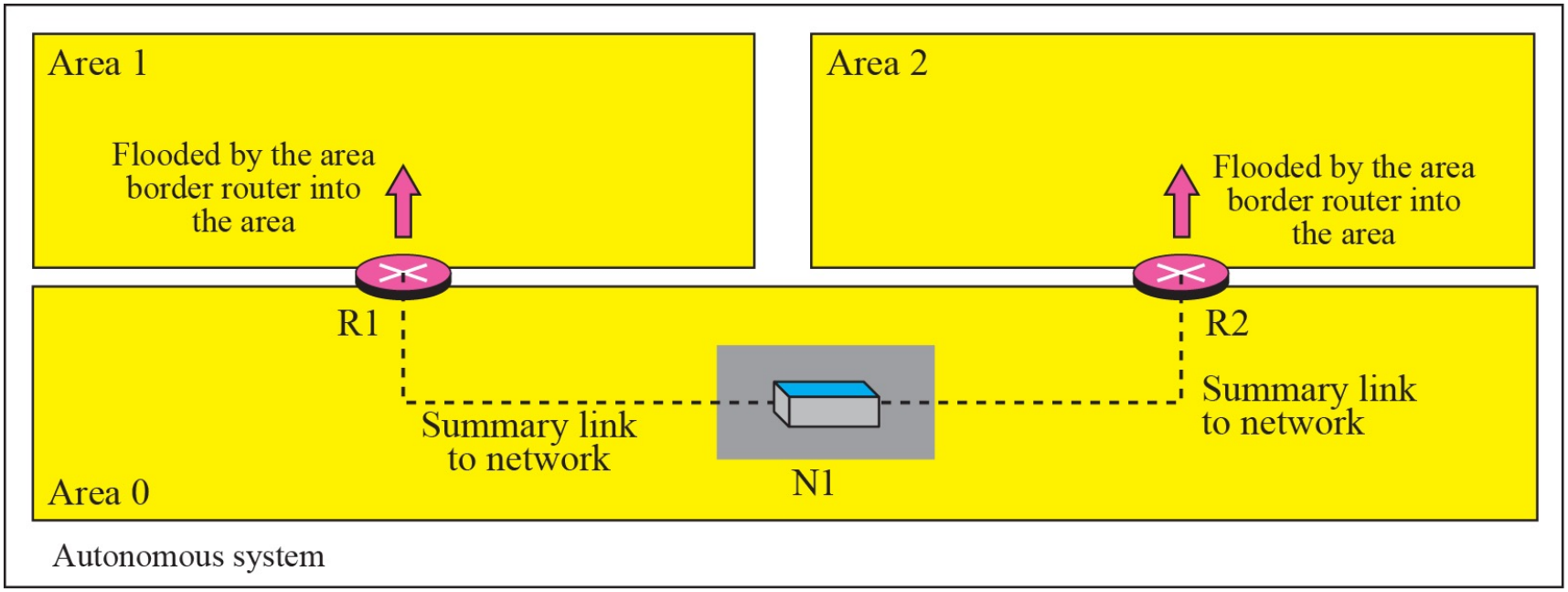
In Figure 11.39, which router(s) sends out the network link LSAs?

Solution

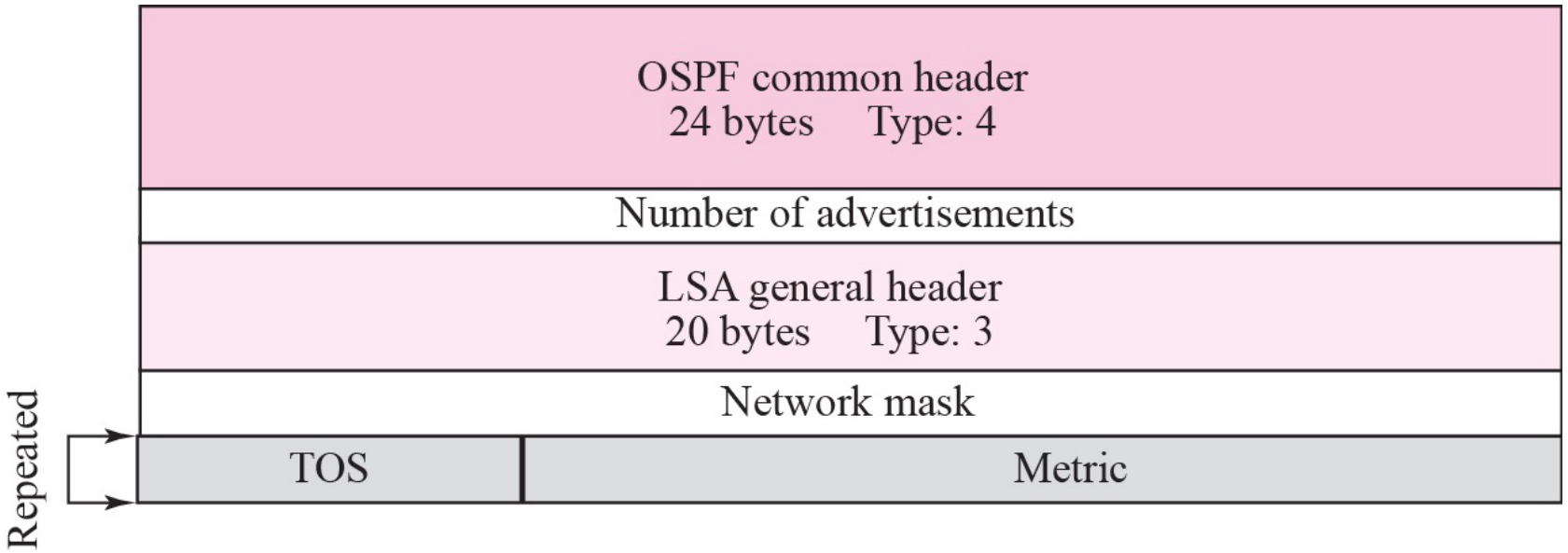
All three networks must advertise network links:

a. Advertisement for N1 is done by R1 because it is the only attached router and therefore the designated router.
b. Advertisement for N2 can be done by either R1, R2, or R3, depending on which one is chosen as the designated router.
c. Advertisement for N3 is done by R3 because it is the only attached router and therefore the designated router

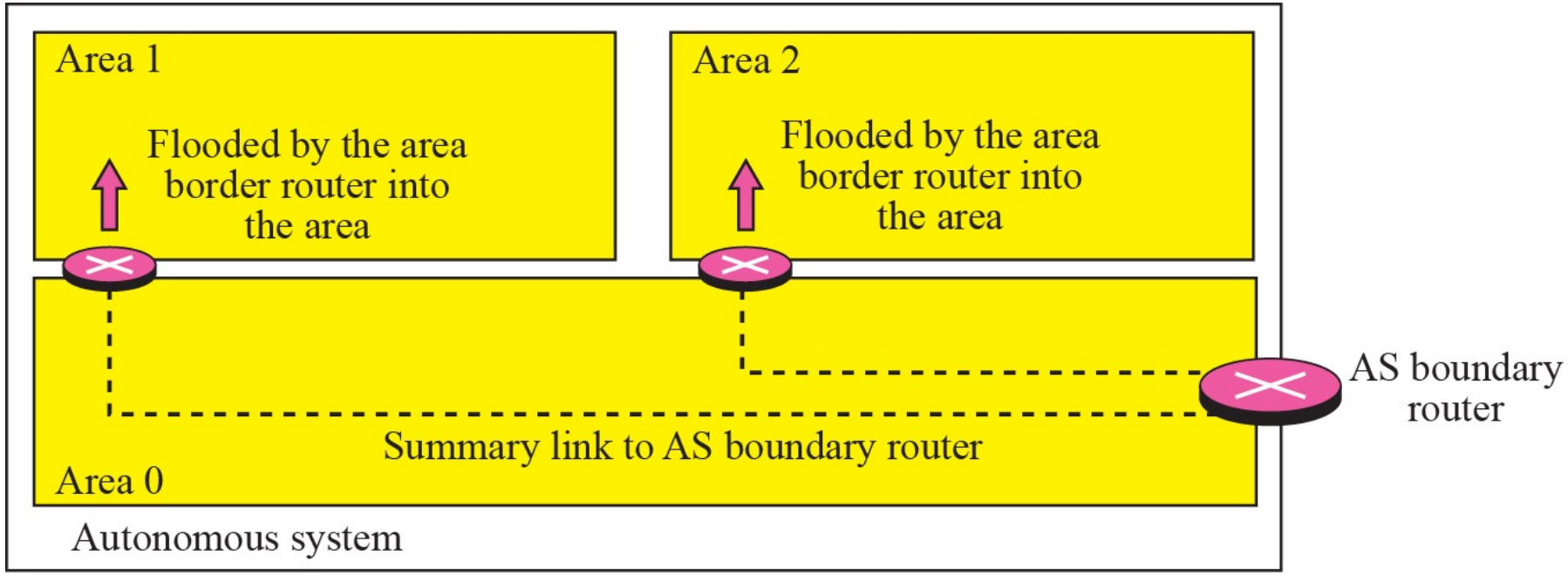
## Figure 11.40 *Summary link to network*



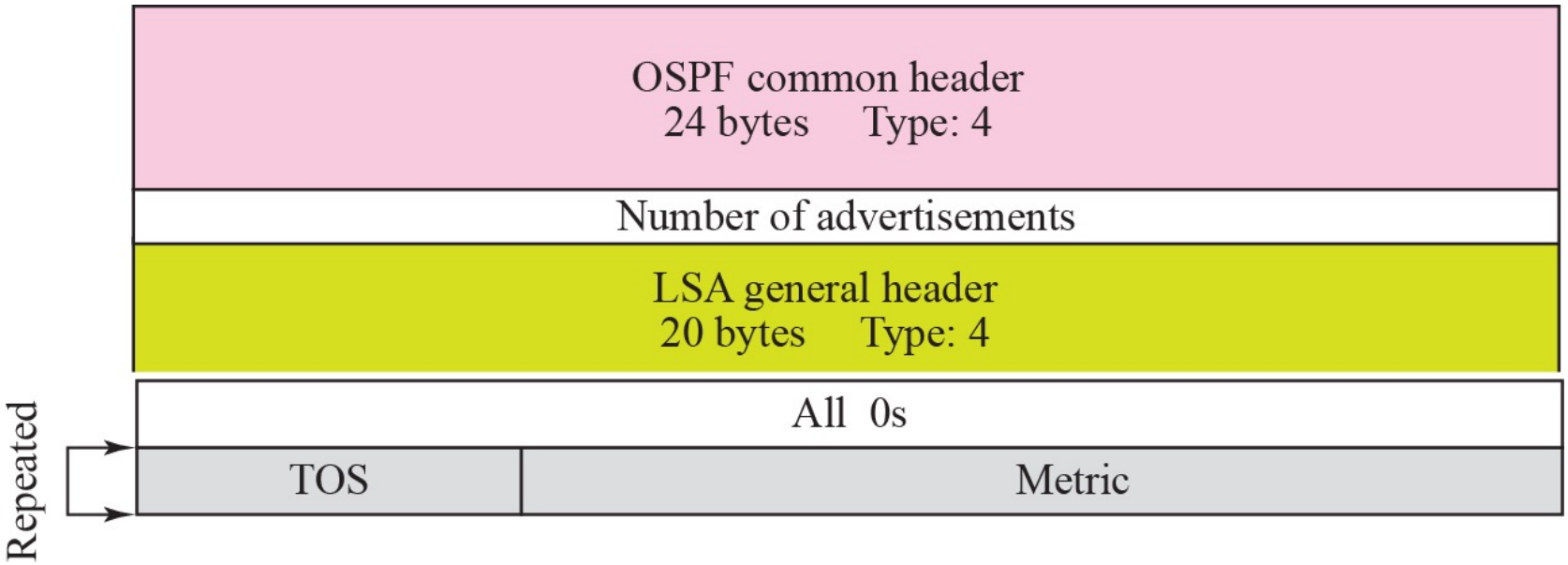
## Figure 11.41 *Summary link to network LSA*



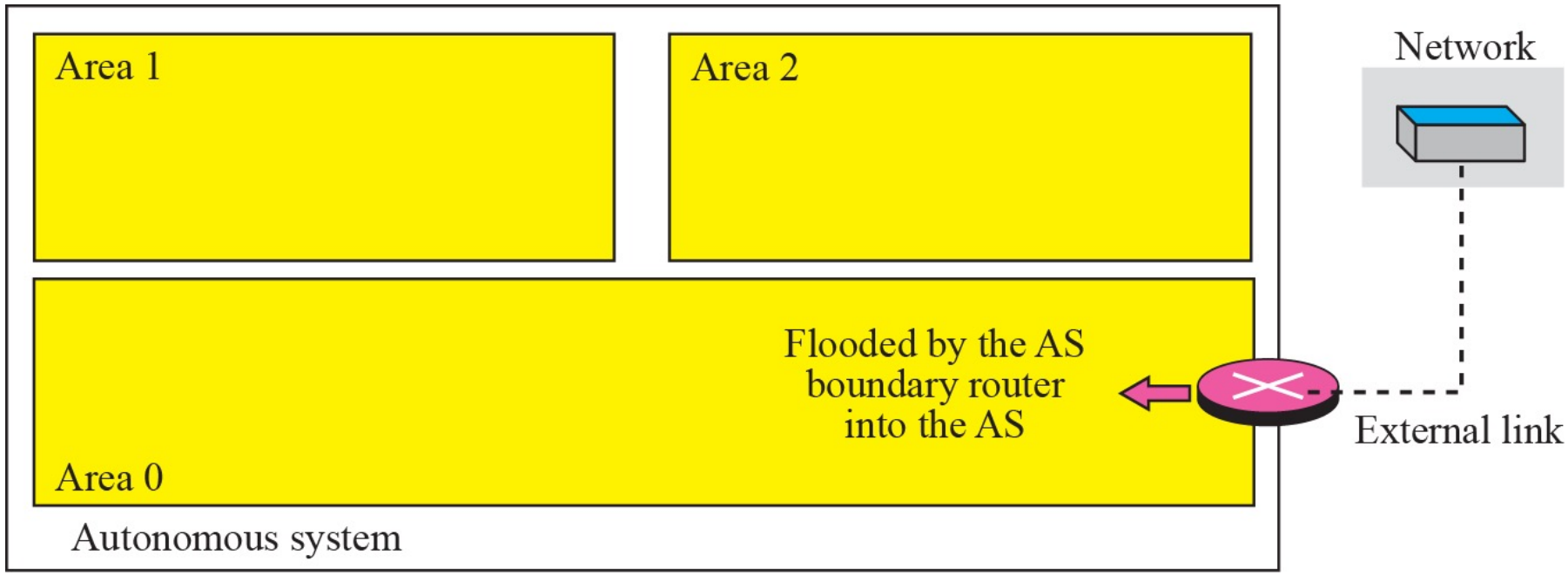
## Figure 11.42 *Summary link to AS boundary router*



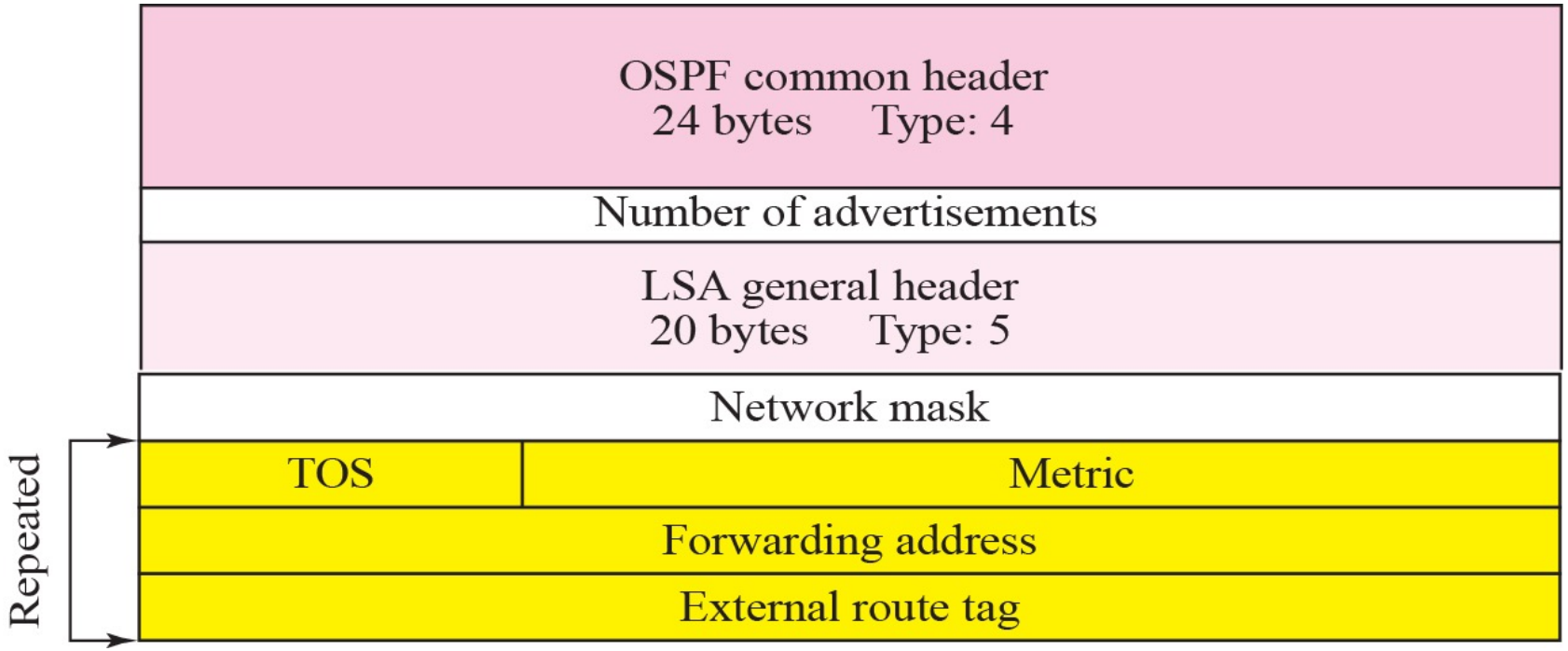
## Figure 11.43 *Summary link to AS boundary router LSA*



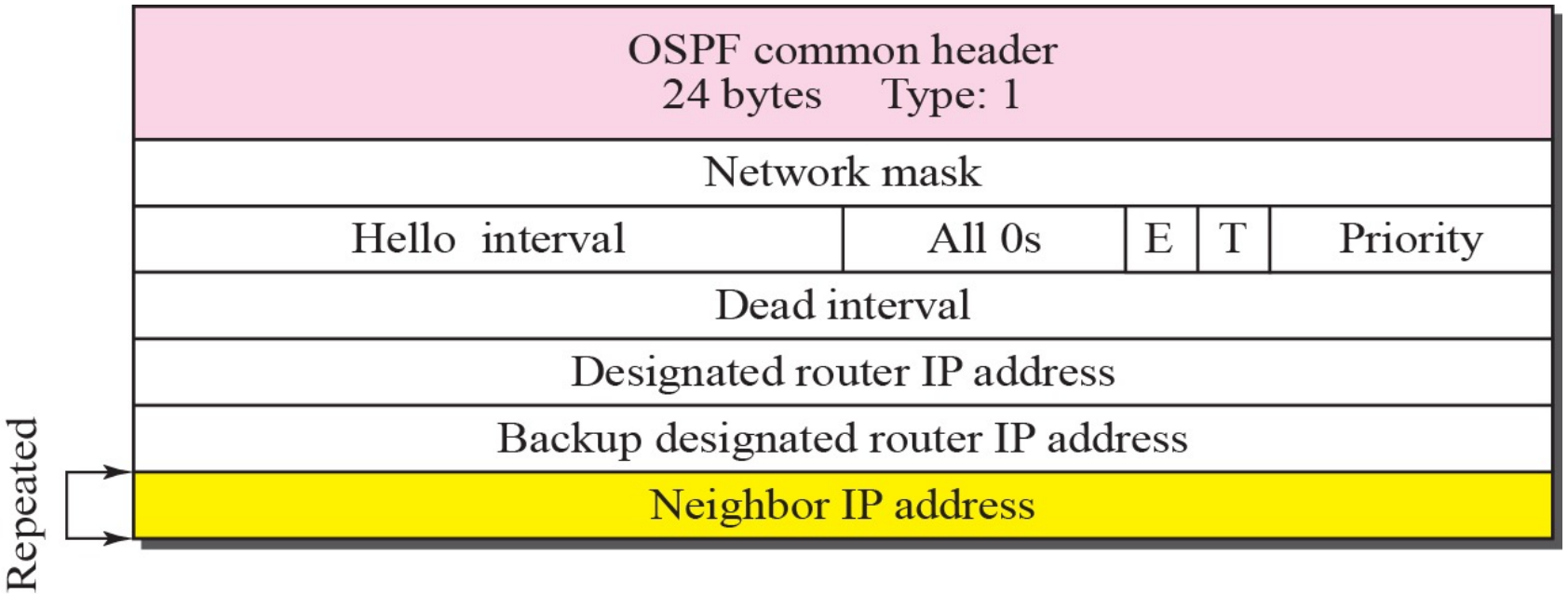
## Figure 11.44 *External link*



## Figure 11.45 *External link LSA*



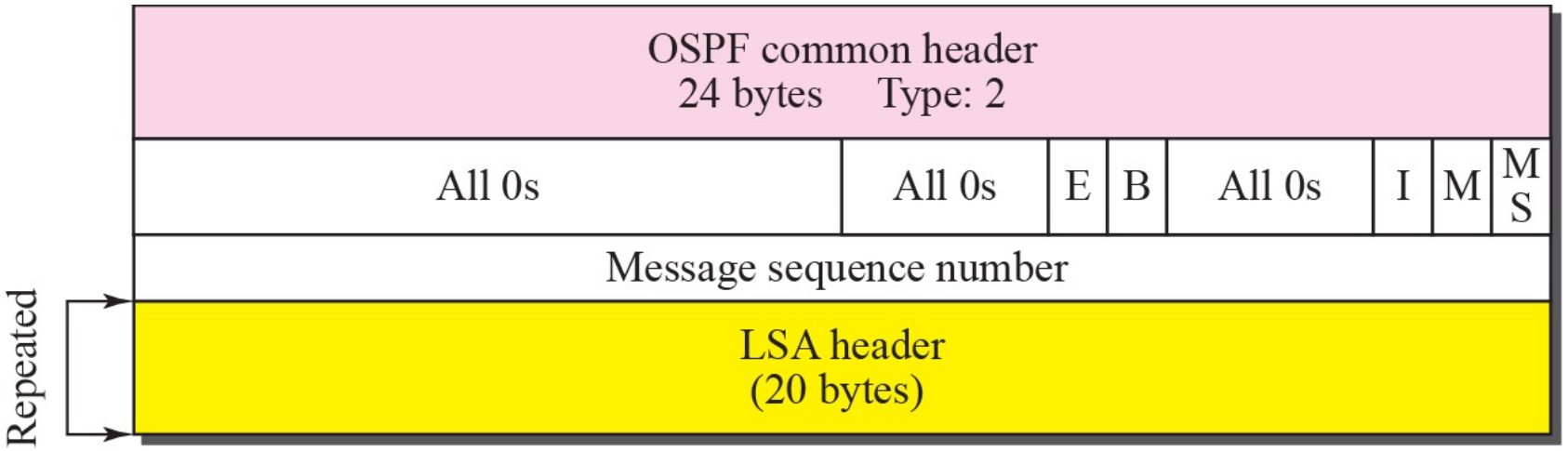
## Figure 11.46 *Hello packet*




***OSPF uses the hello message to create neighborhood relationship and to test the reachability of neighbors.***

***This is the first step in link state routing. Before a router can flood all of the other routers with information about its neighbors, it must first greet it neighbors.***

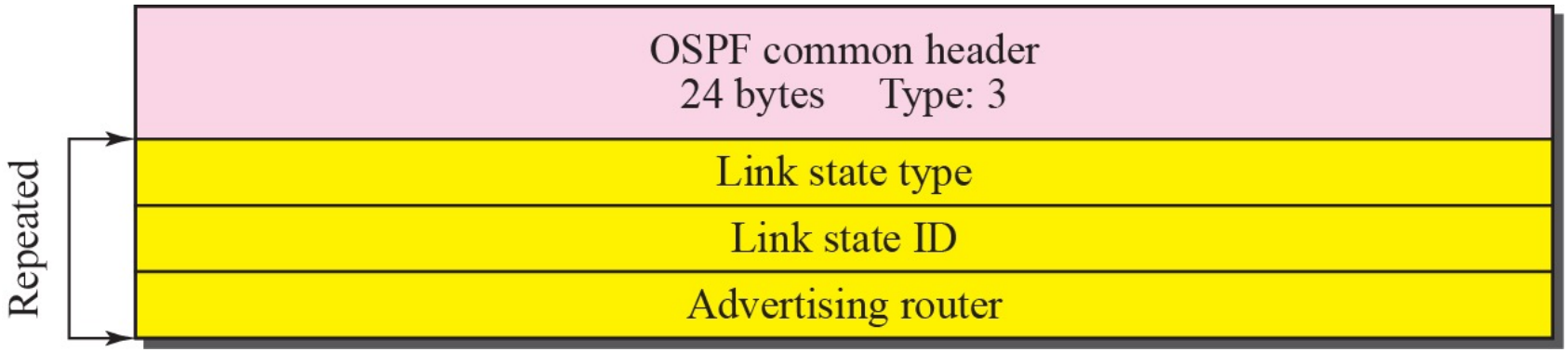
## Figure 11.47 *Database description packet*



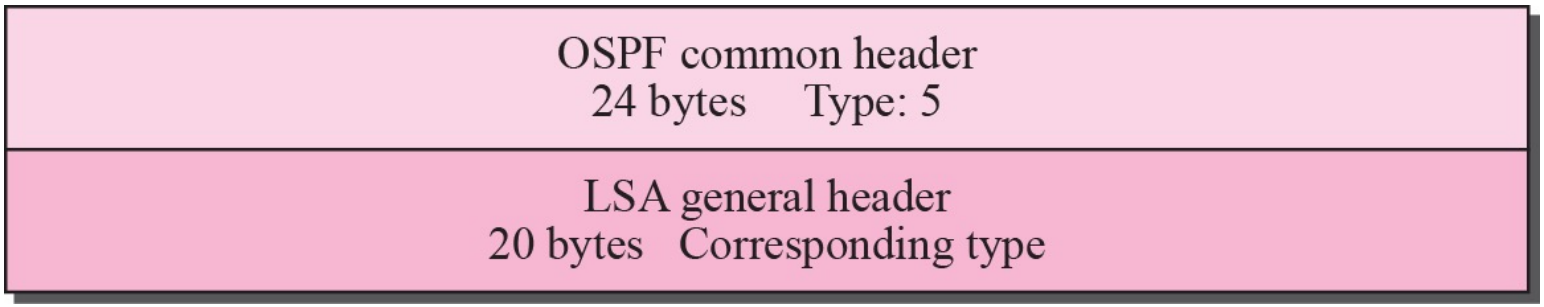

***When a router is connected to the system for the first time or after a failure, it needs the complete link state database immediately. Therefore, it sends hello packets to greet its neighbors. If this is the first time that the neighbors hear from the router, they send a database description message.***

***The database description packet does not contain complete database information; it only gives an outline, the title of each lines in the database.***

## Figure 11.48 *Link state request packet*



## Figure 11.49 *Link state acknowledgment packet*



Note

***OSPF packets are encapsulated in IP datagrams.***

# 11-7 PATH VECTOR ROUTING

Distance vector and link state routing are both interior routing protocols. They can be used inside an autonomous system. Both of these routing protocols become intractable when the domain of operation becomes large. Distance vector routing is subject to instability if there is more than a few hops in the domain of operation. Link state routing needs a huge amount of resources to calculate routing tables. It also creates heavy traffic because of flooding. There is a need for a third routing protocol which we call path vector routing.

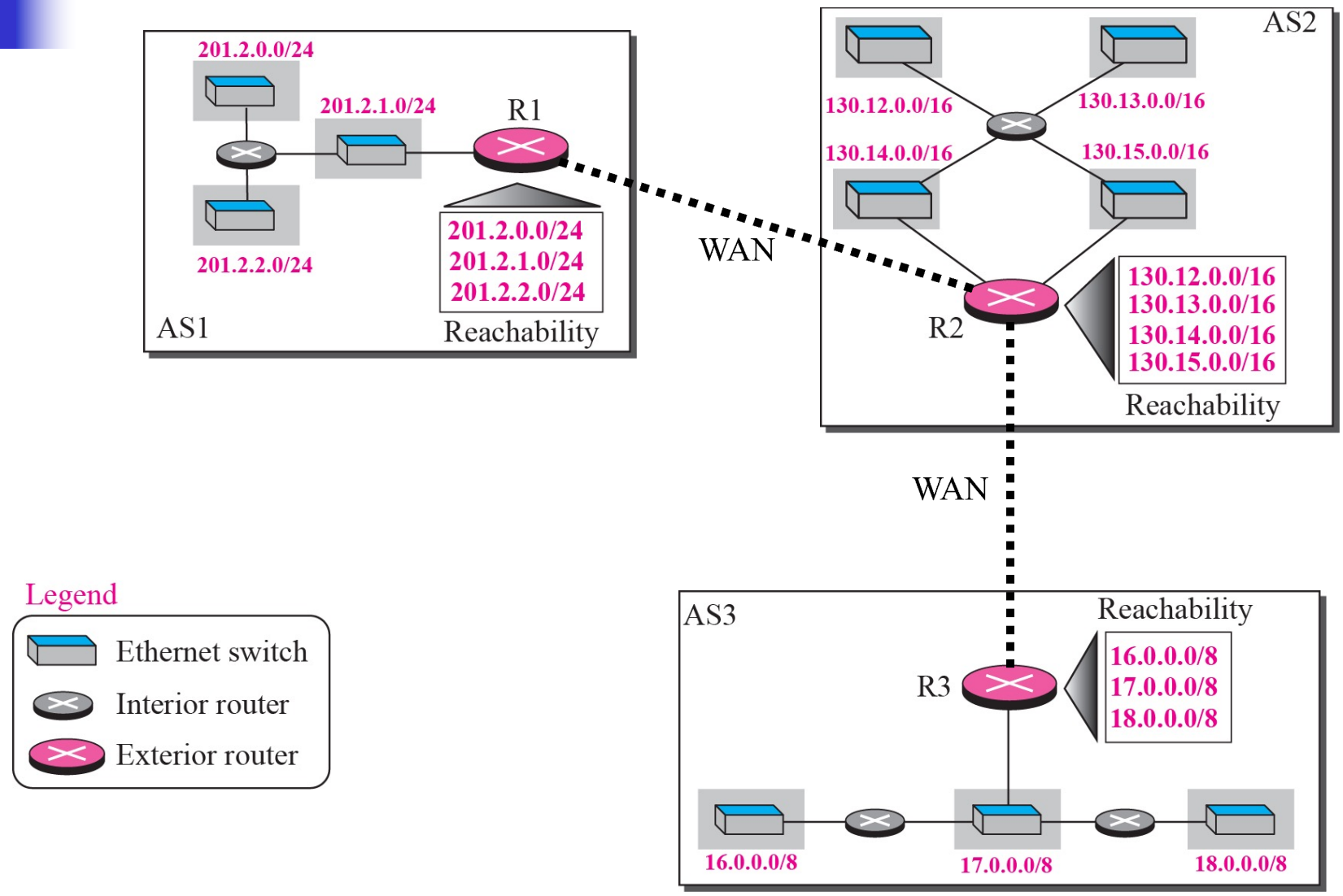
# *Topics Discussed in the Section*

- ✓ **Reachability**
- ✓ **Routing Table**

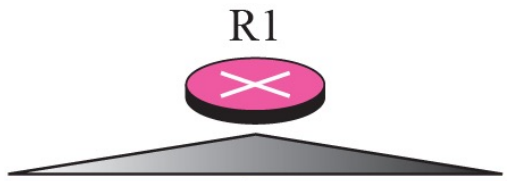
# Example 11.10

The difference between the distance vector routing and path vector routing can be compared to the difference between a national map and an international map. A national map can tell us the road to each city and the distance to be traveled if we choose a particular route; an international map can tell us which cities exist in each country and which countries should be passed before reaching that city.

## Figure 11.50 *Reachability*

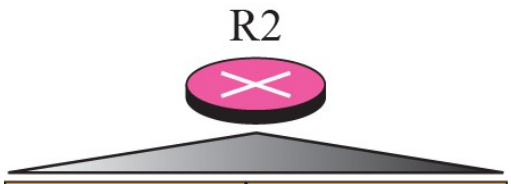


## Figure 11.51 *Stabilized table for three autonomous system*



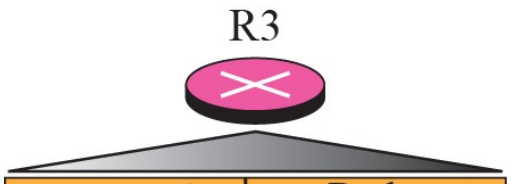

| Network | Path |
|---|---|
| 201.2.0.0/24 | **AS1** (This AS) |
| 201.2.1.0/24 | **AS1** (This AS) |
| 201.2.2.0/24 | **AS1** (This AS) |
| 130.12.0.0/16 | AS1, AS2 |
| 130.13.0.0/16 | AS1, AS2 |
| 130.14.0.0/16 | AS1, AS2 |
| 130.15.0.0/16 | AS1, AS2 |
| 16.0.0.0/8 | AS1, AS2, AS3 |
| 17.0.0.0/8 | AS1, AS2, AS3 |
| 18.0.0.0/8 | AS1, AS2, AS3 |

Path-Vector Routing Table




| Network | Path |
|---|---|
| 201.2.0.0/24 | AS2, AS1 |
| 201.2.1.0/24 | AS2, AS1 |
| 201.2.2.0/24 | AS2, AS1 |
| 130.12.0.0/16 | **AS2** (This AS) |
| 130.13.0.0/16 | **AS2** (This AS) |
| 130.14.0.0/16 | **AS2** (This AS) |
| 130.15.0.0/16 | **AS2** (This AS) |
| 16.0.0.0/8 | AS2, AS3 |
| 17.0.0.0/8 | AS2, AS3 |
| 18.0.0.0/8 | AS2, AS3 |

Path-Vector Routing Table




| Network | Path |
|---|---|
| 201.2.0.0/24 | AS3, AS2, AS1 |
| 201.2.1.0/24 | AS3, AS2, AS1 |
| 201.2.2.0/24 | AS3, AS2, AS1 |
| 130.12.0.0/16 | AS3, AS2 |
| 130.13.0.0/16 | AS3, AS2 |
| 130.14.0.0/16 | AS3, AS2 |
| 130.15.0.0/16 | AS3, AS2 |
| 16.0.0.0/8 | **AS3** (This AS) |
| 17.0.0.0/8 | **AS3** (This AS) |
| 18.0.0.0/8 | **AS3** (This AS) |

Path-Vector Routing Table

## Figure 11.52 *Routing tables after aggregation*

R1

| Network | Path |
|---|---|
| 201.2.0.0/22 | **AS1** (This AS) |
| 130.12.0.0/18 | AS1, AS2 |
| 16.0.0.0/6 | AS1, AS2, AS3 |

Path-Vector Routing Table

R2

| Network | Path |
|---|---|
| 201.2.0.0/22 | AS2, AS1 |
| 130.12.0.0/18 | **AS2** (This AS) |
| 16.0.0.0/6 | AS2, AS3 |

Path-Vector Routing Table

R3

| Network | Path |
|---|---|
| 201.2.0.0/22 | AS3, AS2, AS1 |
| 130.12.0.0/18 | AS3, AS2 |
| 16.0.0.0/6 | **AS3** (This AS) |

Path-Vector Routing Table

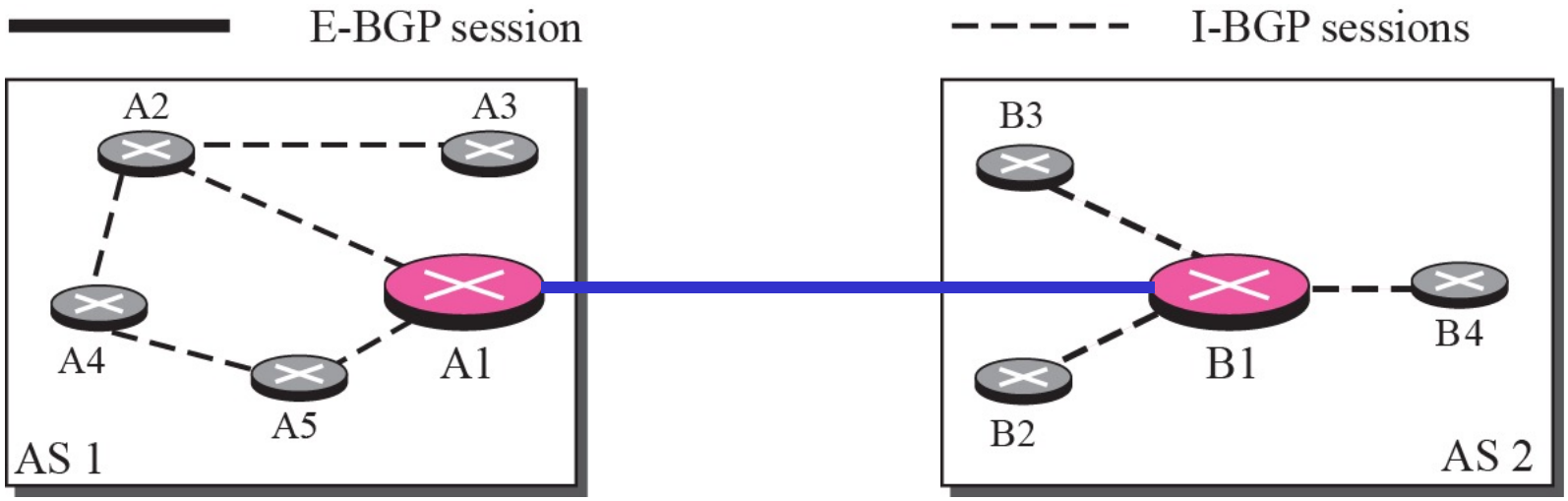
# 11-8 BGP

Border Gateway Protocol (BGP) is an interdomain routing protocol using path vector routing. It first appeared in 1989 and has gone through four versions.

# *Topics Discussed in the Section*

- ✓ Types of Autonomous Systems
- ✓ Path Attributes
- ✓ BGP Sessions
- ✓ External and Internal BGP
- ✓ Types of Packets
- ✓ Packet Format
- ✓ Encapsulation

## Figure 11.53 *Internal and external BGP sessions*



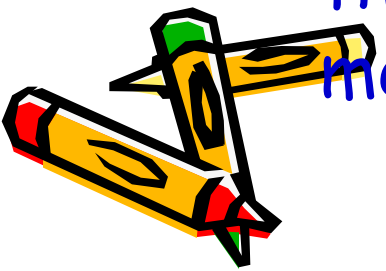

***A speaker node advertises the path, not the metric of the nodes, in its AS or other ASs.***

# Path Vector Routing (1)

- Sharing
  - A speaker in an AS shares its table with immediate neighbors
- Updating
  - Adding the nodes that are not in its routing table and adding its own AS and the AS that sent the table
  - The routing table shows the path completely

# Path Vector Routing (2)

- Loop prevention
  - A route checks to see if its AS is in the path list to the destination
- Policy routing
  - If one of the ASs listed in the path is against its policy, it can ignore that path and that destination
  - It does not update its routing table with the path, and it does not send this message to its neighbors



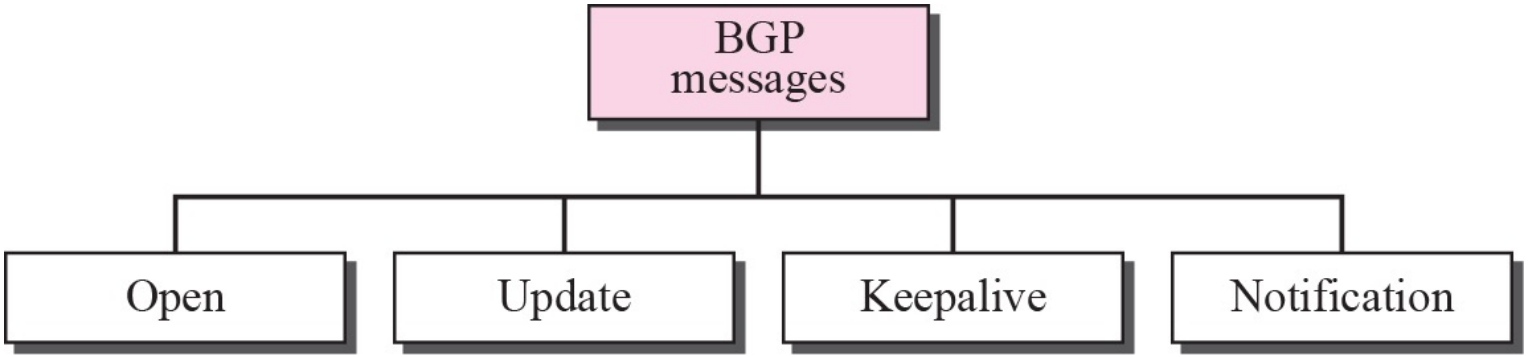
# Path Vector Routing (3)

- Optimum path
  - Problem: each AS that is included in the path may use a different criteria for the metric
  - The optimum path is the path that fits the organization
  - For Fig. 14-49, the author chose the one that had the smaller number of ASs
  - Other criteria: security, safety, reliability, etc.

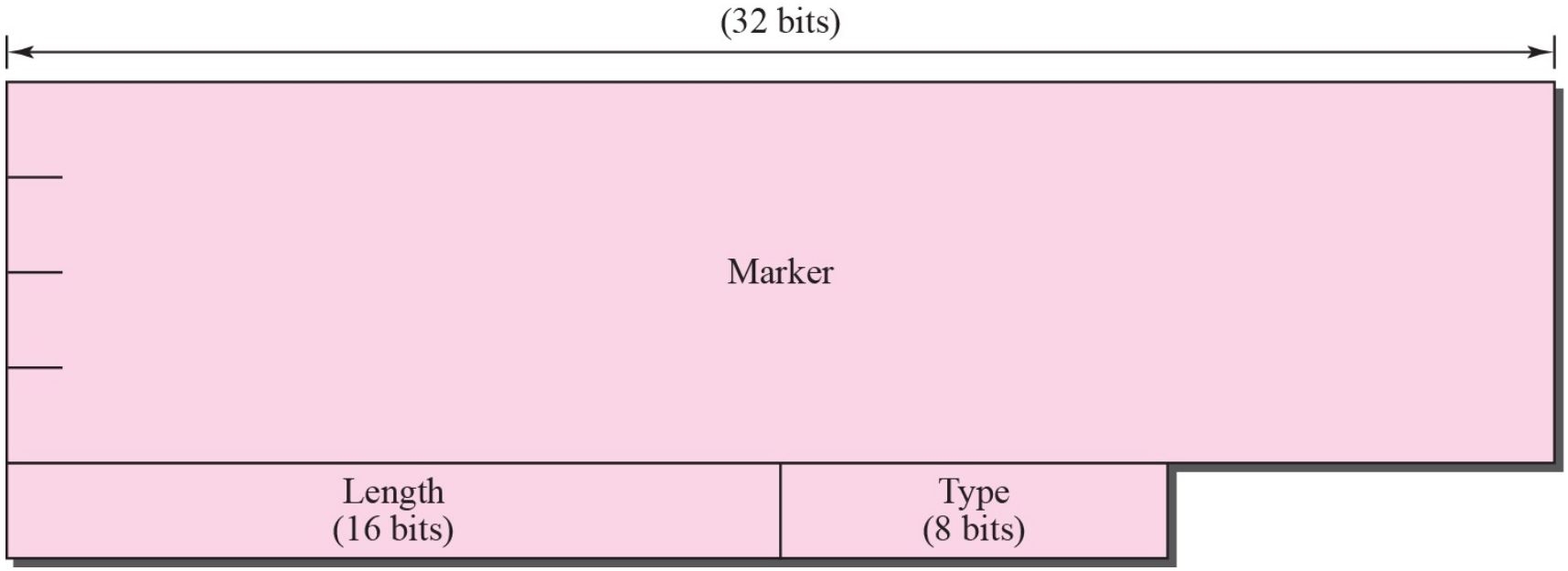
# Types of AS

- Stub AS
  - Only one connection to another AS (only a source or sink for data traffic)
- Multihomed AS
  - More than one connection to other AS, but it is still only a source or sink for data traffic
- Transit AS
  - Multihomed AS that also allows transient traffic

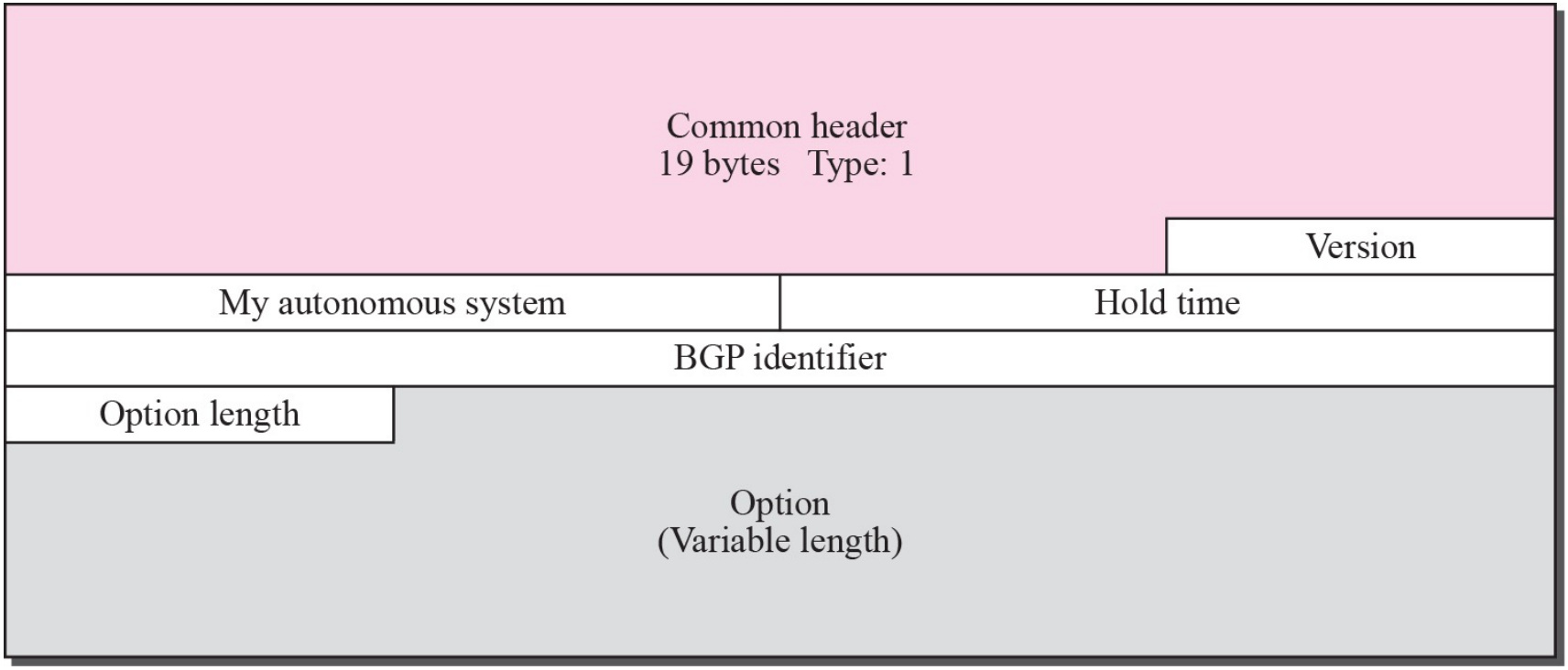
## Figure 11.54 *Types of BGP messages*



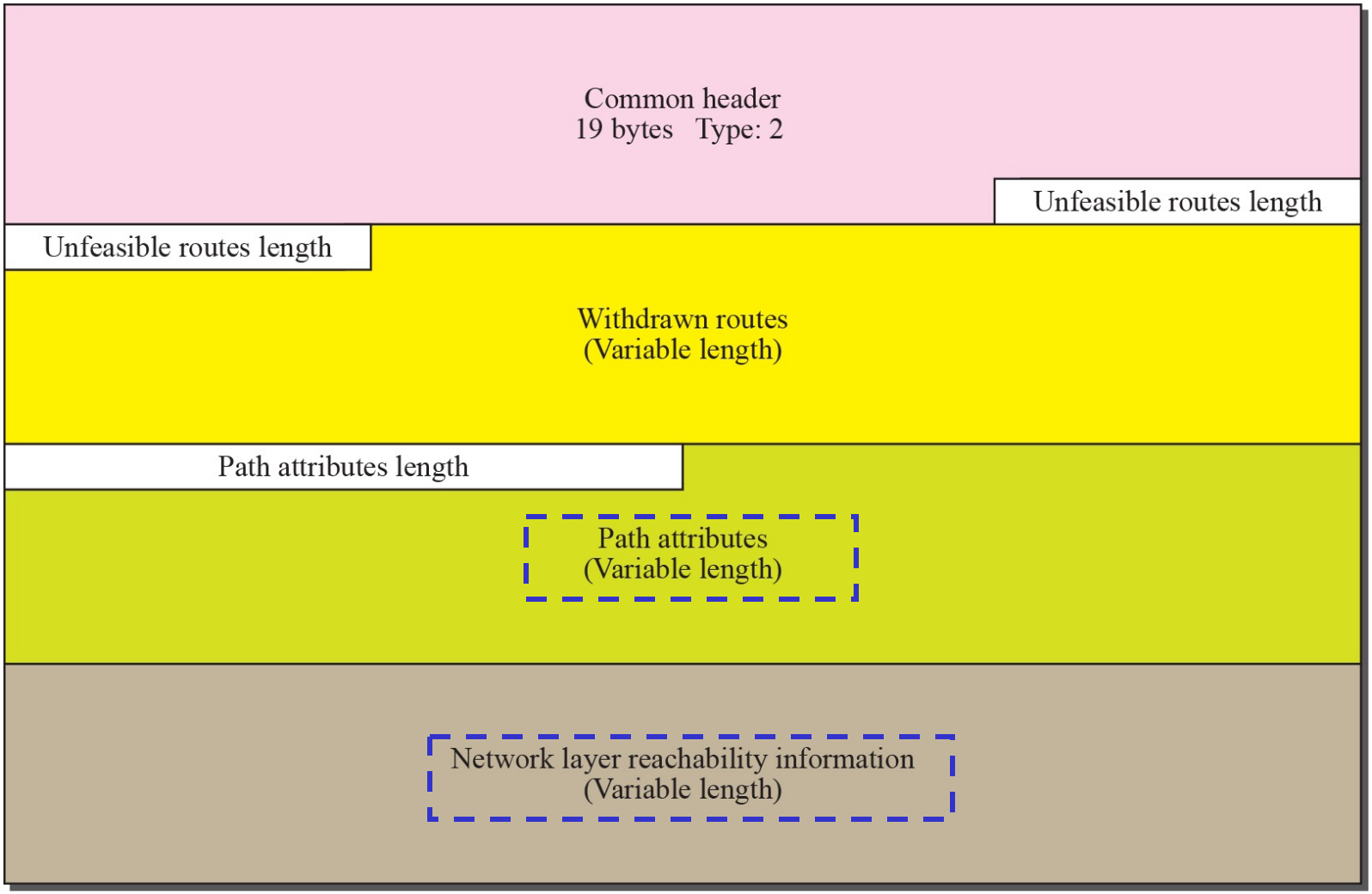
## Figure 11.55 *BGP packet header*



## Figure 11.56 *Open message*

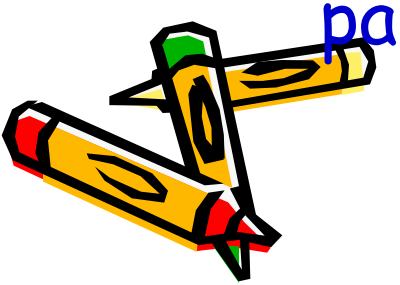


## Figure 11.57 *Update message*



# Path Attributes

- ORIGIN
  - The source of the routing information (RIP, OSPF, etc)
- AS_PATH
  - The list of ASs through which the destination can be reached
- NEXT-HOP
  - The next router to which the data packet should be sent



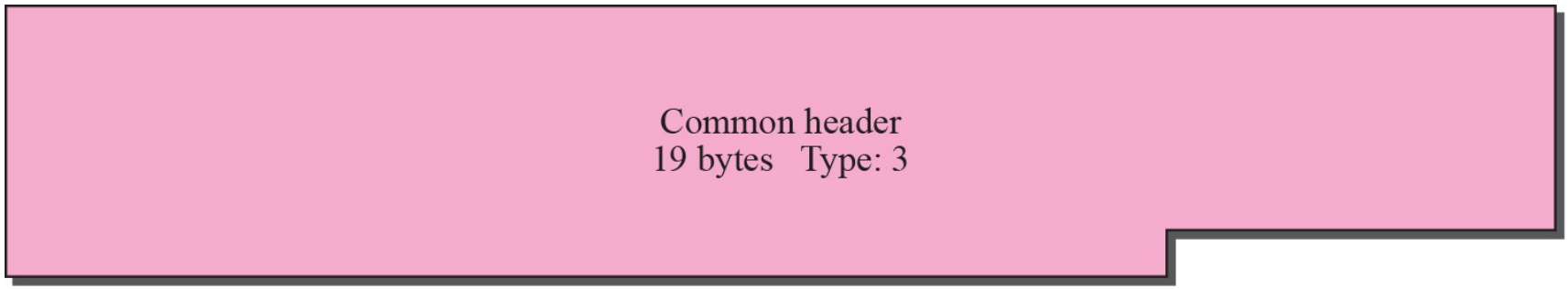
# NLRI

- Network layer reachability information
  - It defines the network that is actually advertised by this message
  - Length field and IP address prefix
  - BGP4 supports classless addressing and CIDR

Note

**BGP supports classless addressing and CIDR.**

## Figure 11.58 *Keepalive message*



## Figure 11.59 *Notification message*

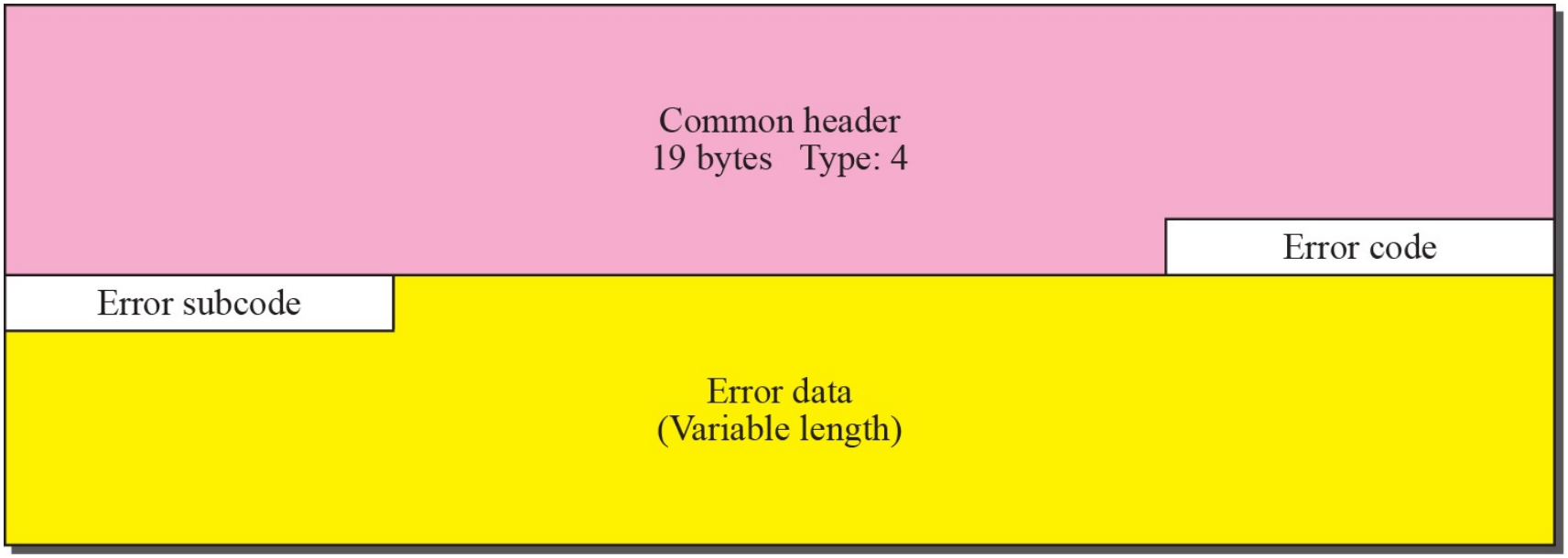


**Table 11.6** *Error Codes*

| *Error Code* | *Error Code Description* | *Error Subcode Description* |
|---|---|---|
| 1 | Message header error | Three different subcodes are defined for this type of error: synchronization problem (1), bad message length (2), and bad message type (3). |
| 2 | Open message error | Six different subcodes are defined for this type of error: unsupported version number (1), bad peer AS (2), bad BGP identifier (3), unsupported optional parameter (4), authentication failure (5), and unacceptable hold time (6). |
| 3 | Update message error | Eleven different subcodes are defined for this type of error: malformed attribute list (1), unrecognized well-known attribute (2), missing well-known attribute (3), attribute flag error (4), attribute length error (5), invalid origin attribute (6), AS routing loop (7), invalid next hop attribute (8), optional attribute error (9), invalid network field (10), malformed AS_PATH (11). |
| 4 | Hold timer expired | No subcode defined. |
| 5 | Finite state machine error | This defines the procedural error. No subcode defined. |
| 6 | Cease | No subcode defined. |

Note

***BGP uses the services of TCP on port 179.***