# Chapter 14

# *Unicast Routing Protocols: RIP, OSPF, and BGP*

## Objectives

*Upon completion you will be able to:*

- *Distinguish between intra and interdomain routing*
- *Understand distance vector routing and RIP*
- *Understand link state routing and OSPF*
- *Understand path vector routing and BGP*

# Introduction

To forward a packet on outgoing link, router has to decide optimum pathway.

Router may assign a cost to packet called as *metric*.

In RIP, cost is same for each *hop count*.

In OSPF, route through network can have different costs based on *type of service* desired.

Routers use routing tables to decide best route.

In BGP, criteria is *policy* decided by Administrator.

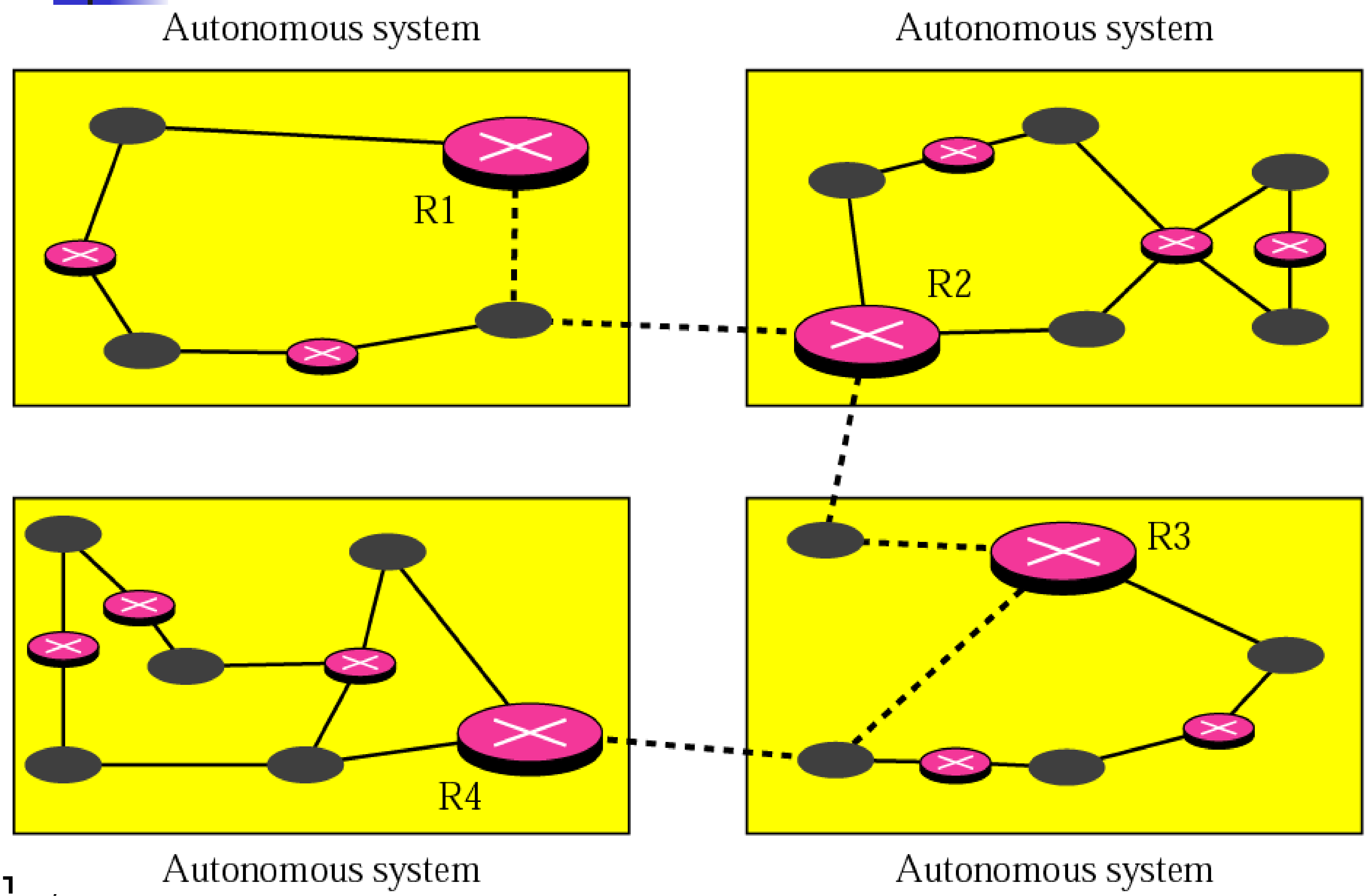
Routing table can be *static* or *dynamic*.
Internet needs dynamic routing tables.

# 14.1 INTRA- AND INTERDOMAIN ROUTING

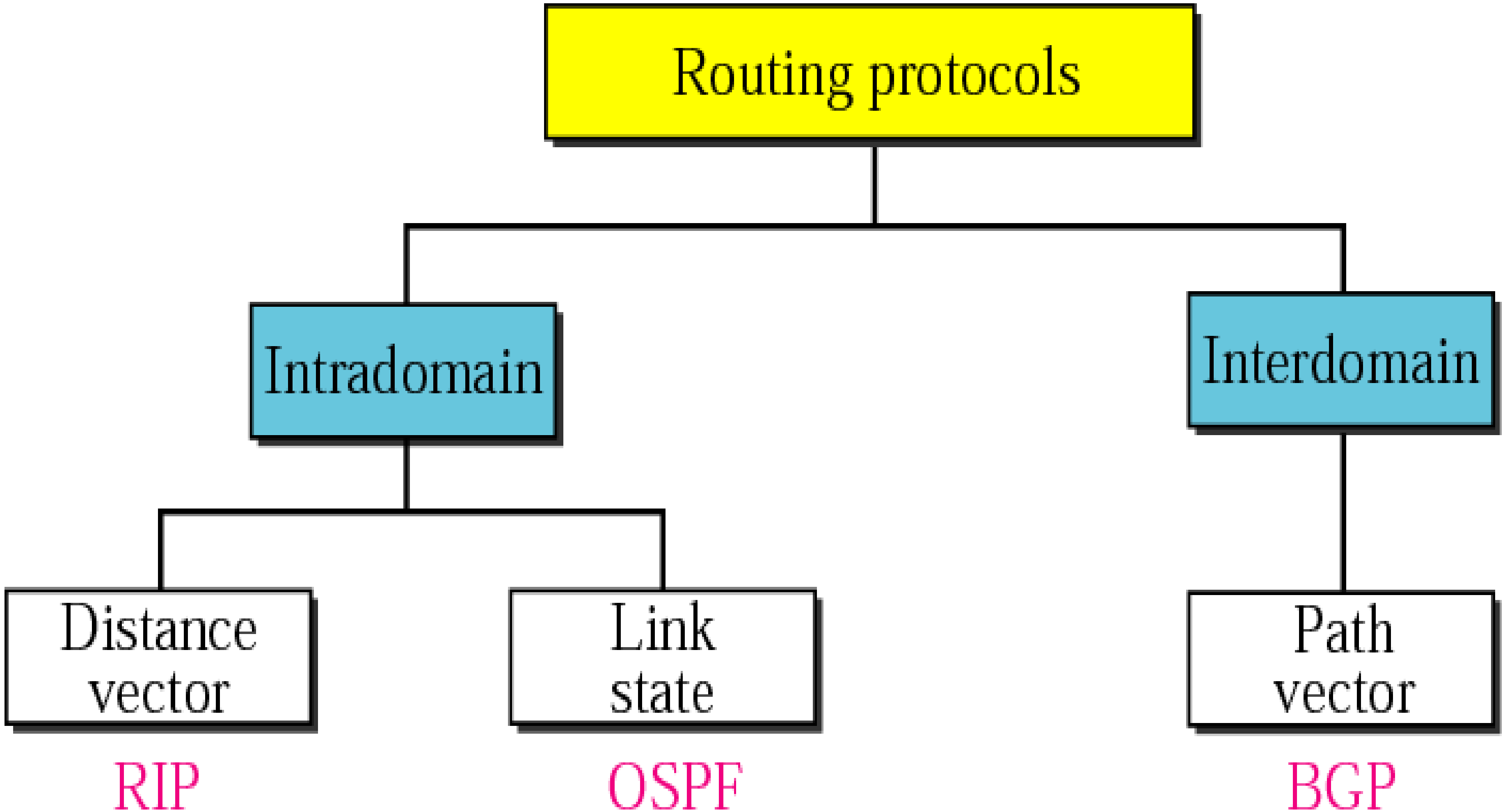
***An Autonomous System (AS) is a group of networks and routers under the authority of a single administration.***

***Routing inside an autonomous system is referred to as intradomain routing. Routing between autonomous systems is referred to as interdomain routing.***

# Autonomous systems



# *Popular routing protocols*



# 14.2 DISTANCE VECTOR ROUTING

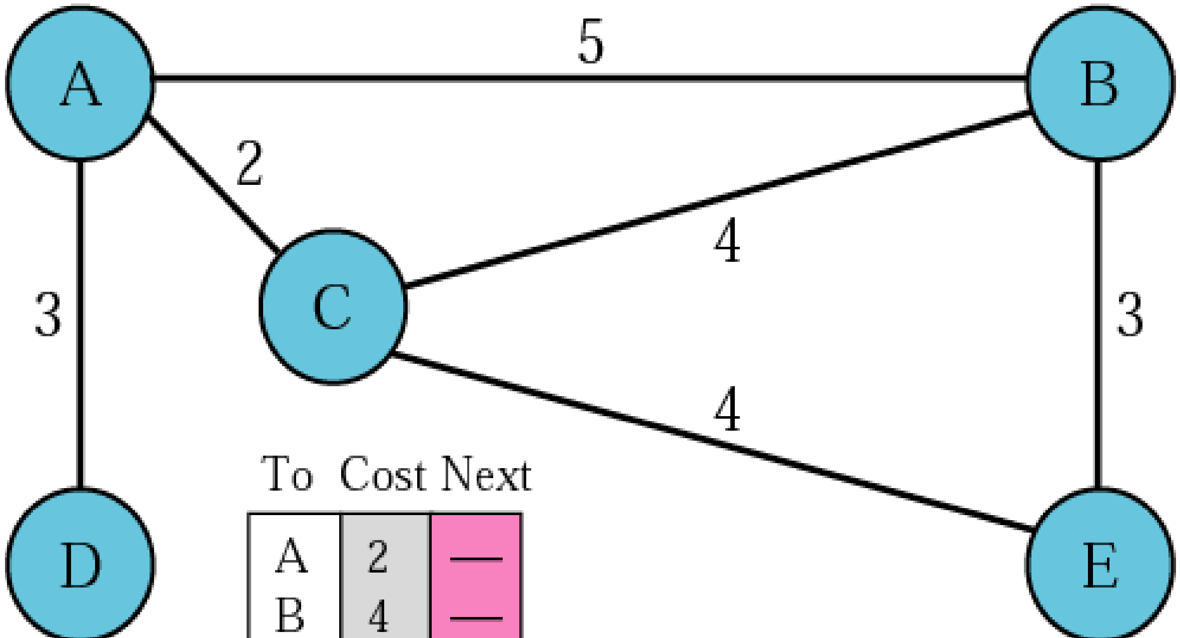
*In distance vector routing, the least cost route between any two nodes is the route with minimum distance. In this protocol each node maintains a vector (table) of minimum distances to every node*

*Ex. System of five nodes with their corresponding tables.*

***The topics discussed in this section include:***

*Initialization*
*Sharing*
*Updating*
*When to Share*
*Two-Node Loop Instability*
*Three-Node Instability*

# *Distance vector routing tables*




| To | Cost | Next |
|---|---|---|
| A | 0 | — |
| B | 5 | — |
| C | 2 | — |
| D | 3 | — |
| E | 6 | C |

**A's table**

| To | Cost | Next |
|---|---|---|
| A | 5 | — |
| B | 0 | — |
| C | 4 | — |
| D | 8 | A |
| E | 3 | — |

**B's table**

| To | Cost | Next |
|---|---|---|
| A | 2 | — |
| B | 4 | — |
| C | 0 | — |
| D | 5 | A |
| E | 4 | — |

**C's table**

| To | Cost | Next |
|---|---|---|
| A | 3 | — |
| B | 8 | A |
| C | 5 | A |
| D | 0 | — |
| E | 9 | A |

**D's table**

| To | Cost | Next |
|---|---|---|
| A | 6 | C |
| B | 3 | — |
| C | 4 | — |
| D | 9 | C |
| E | 0 | — |

**E's table**

# *Initialization of tables in distance vector routing*

At first, each node only knows the distance between itself and its immediate neighbors.

Each node sends a message to its neighbors and finds the distance.

Fig shows the initial tables.

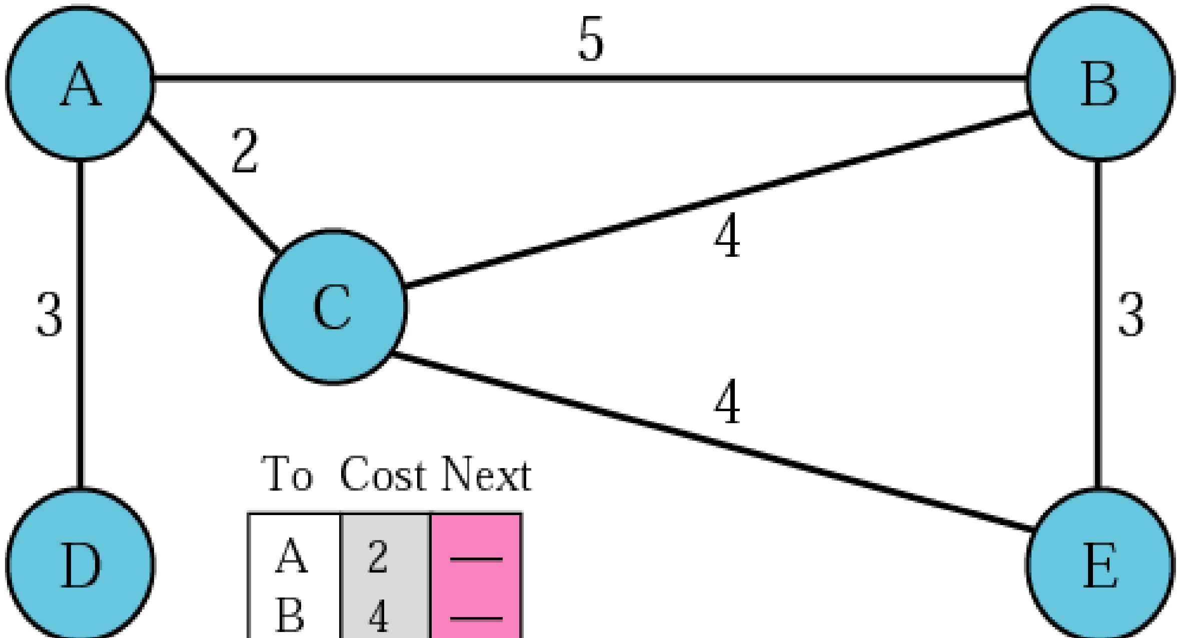
# *Initialization of tables in distance vector routing*

| To | Cost | Next |
|---|---|---|
| A | 0 | — |
| B | 5 | — |
| C | 2 | — |
| D | 3 | — |
| E | ∞ | |

**A's Table**

| To | Cost | Next |
|---|---|---|
| A | 5 | — |
| B | 0 | — |
| C | 4 | — |
| D | ∞ | |
| E | 3 | — |

**B's Table**




| To | Cost | Next |
|---|---|---|
| A | 3 | — |
| B | ∞ | |
| C | ∞ | |
| D | 0 | — |
| E | ∞ | |

**D's Table**

| To | Cost | Next |
|---|---|---|
| A | 2 | — |
| B | 4 | — |
| C | 0 | — |
| D | ∞ | |
| E | 4 | — |

**C's Table**

| To | Cost | Next |
|---|---|---|
| A | ∞ | |
| B | 3 | B |
| C | 4 | C |
| D | ∞ | |
| E | 0 | D |

**E's Table**

# *Initialization of tables in distance vector routing*

*Sharing:*

DV Routing is built on idea of sharing information.

How much of the table should be shared?

Soln. Send the complete table as message to neighbor. Neighbor only needs first two columns.

*Updating:*

1. Receiving node adds the cost of itself and sending node in second column.
2. Add the name of sending node to each row in third column.
3. Receiving node compares old table with new table
    a. If next-node entry is different, receiver chooses the low cost.
    b. If next-node entry is same, receiver chooses the new row.

## *Updating in distance vector routing*

| To | Cost |
|---|---|
| A | 2 |
| B | 4 |
| C | 0 |
| D | ∞ |
| E | 4 |

**Received from C**

| To | Cost | Next |
|---|---|---|
| A | 4 | C |
| B | 6 | C |
| C | 2 | C |
| D | ∞ | C |
| E | 6 | C |

**A's Modified Table**

| To | Cost | Next |
|---|---|---|
| A | 0 | — |
| B | 5 | — |
| C | 2 | — |
| D | 3 | — |
| E | ∞ | |

**A's Old Table**

**Compare**

| To | Cost | Next |
|---|---|---|
| A | 0 | — |
| B | 5 | — |
| C | 2 | — |
| D | 3 | — |
| E | 6 | C |

**A's NewTable**

# *Two-node instability*

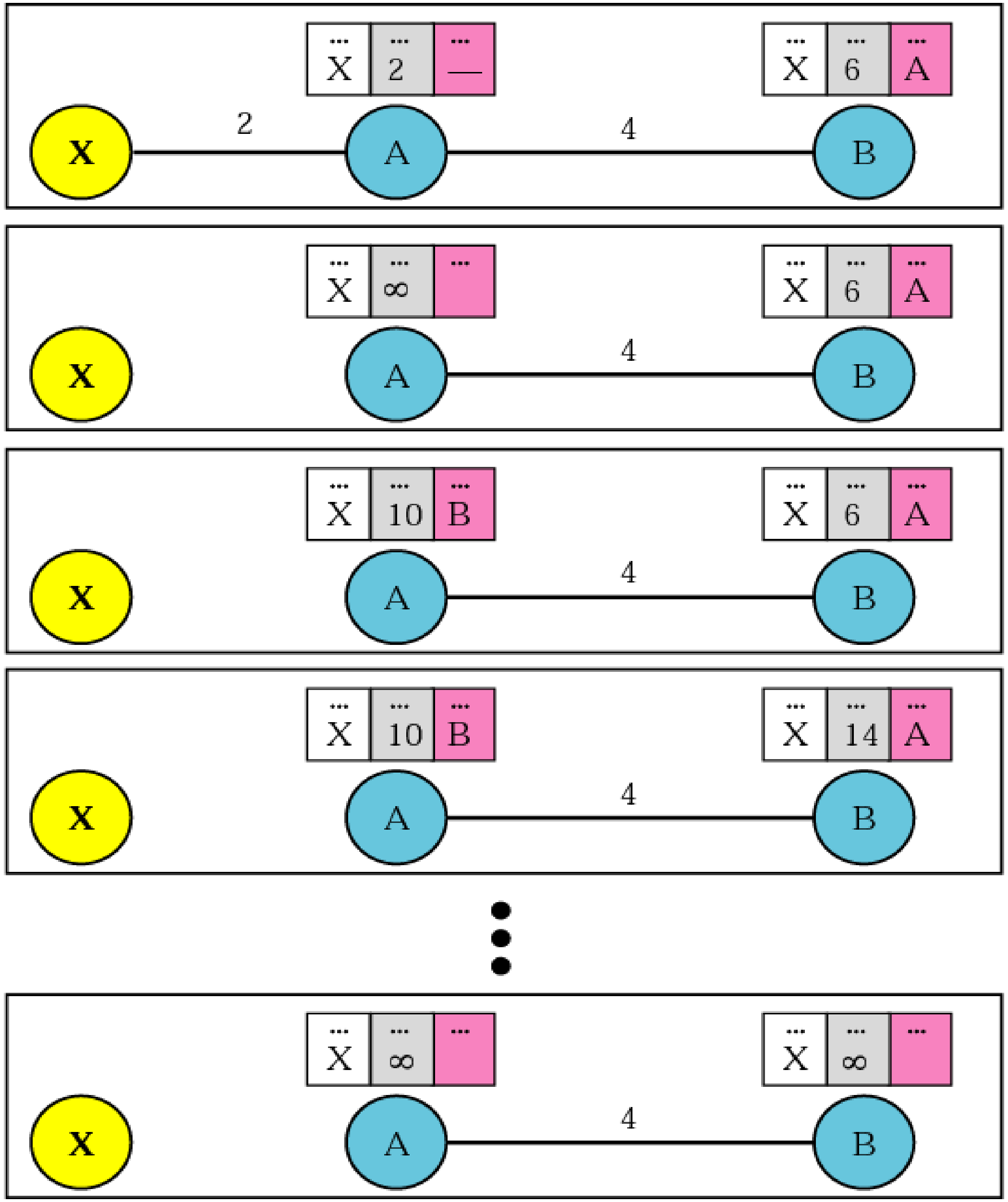
Before failure

After failure

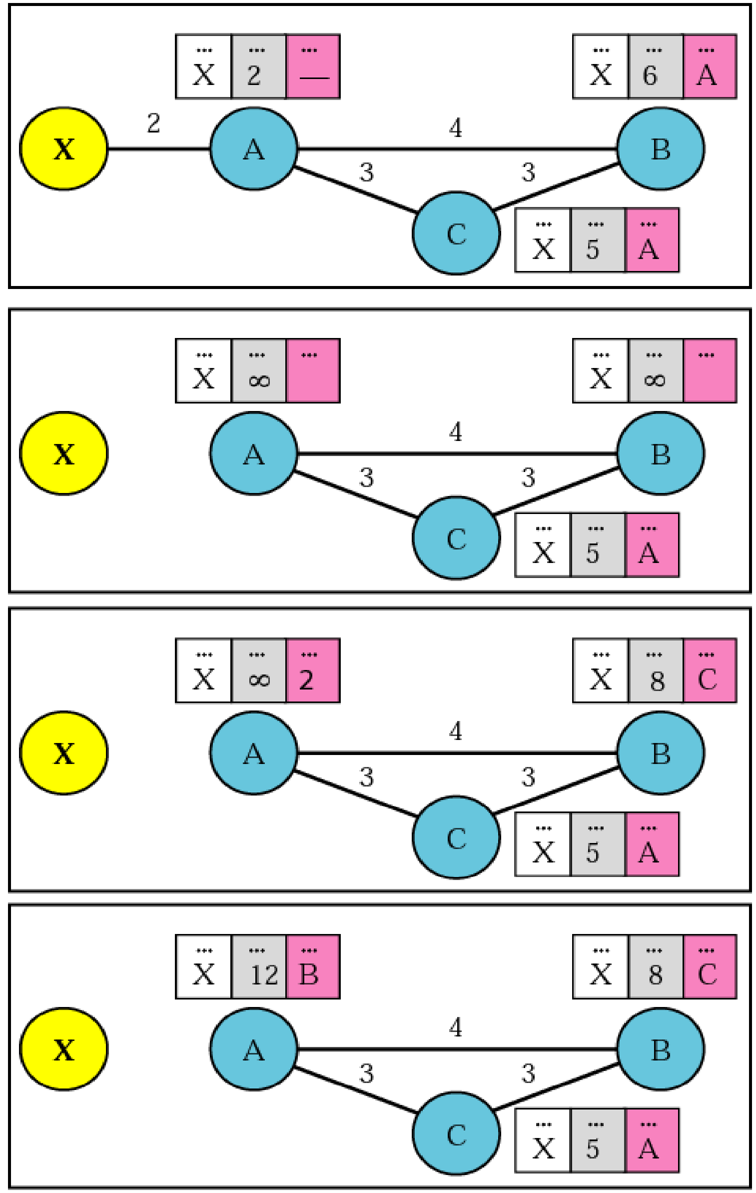
After A receives update from B

After B receives update from A

Finally



# **Figure 14.7** *Three-node instability*




Before failure

After A sends the route to B and C, but the packet to C is lost

After C sends the route to B

After B sends the route to A

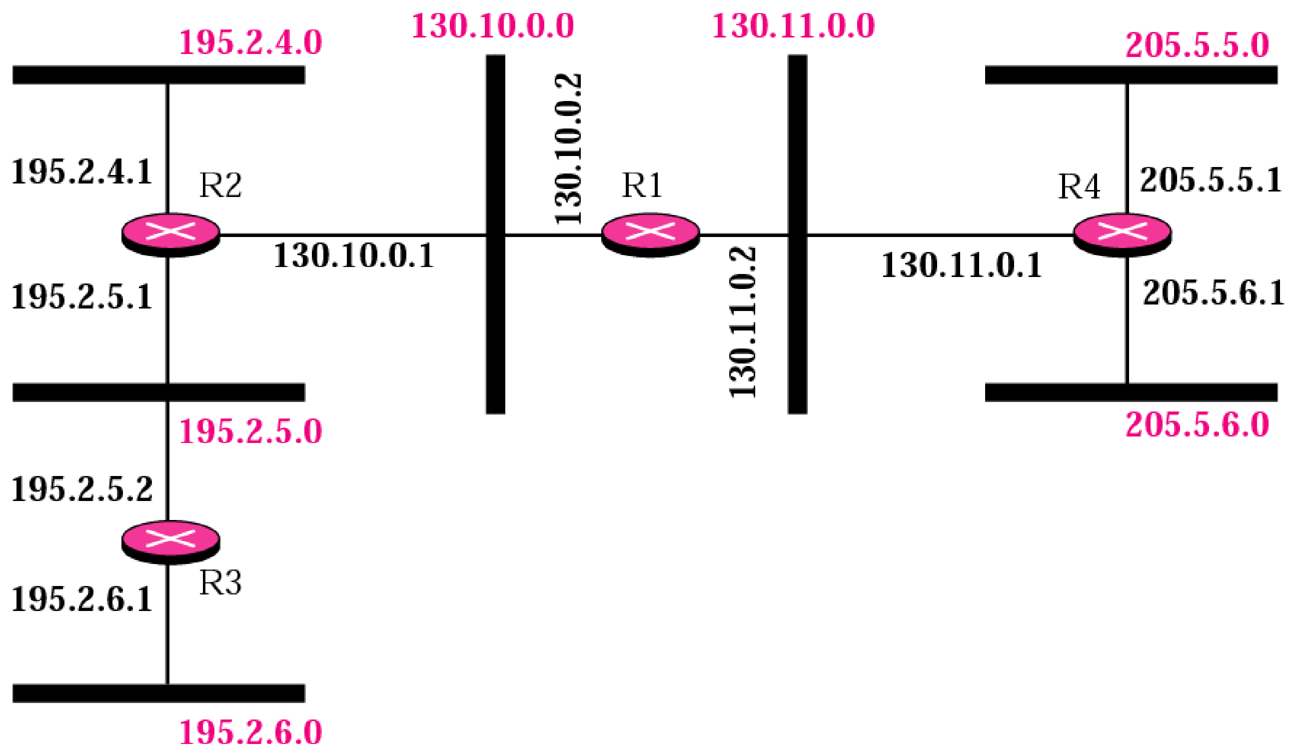
# 14.3 RIP

***The Routing Information Protocol (RIP) is an intradomain routing protocol used inside an autonomous system. It is a very simple protocol based on distance vector routing.***

***The topics discussed in this section include:***

***RIP Message Format***
***Requests and Responses***
***Timers in RIP***
***RIP Version 2***
***Encapsulation***

# **Figure 14.8** *Example of a domain using RIP*




| Dest. | Hop | Next |
|---|---|---|
| 130.10.0.0 | 1 | ——— |
| 130.11.0.0 | 1 | ——— |
| 195.2.4.0 | 2 | 130.10.0.1 |
| 195.2.5.0 | 2 | 130.10.0.1 |
| 195.2.6.0 | 3 | 130.10.0.1 |
| 205.5.5.0 | 2 | 130.11.0.1 |
| 205.5.6.0 | 2 | 130.11.0.1 |

R1 Table

| Dest. | Hop | Next |
|---|---|---|
| 130.10.0.0 | 1 | ——— |
| 130.11.0.0 | 2 | 130.10.0.2 |
| 195.2.4.0 | 1 | ——— |
| 195.2.5.0 | 1 | ——— |
| 195.2.6.0 | 2 | 195.2.5.2 |
| 205.5.5.0 | 3 | 130.10.0.2 |
| 205.5.6.0 | 3 | 130.10.0.2 |

R2 Table

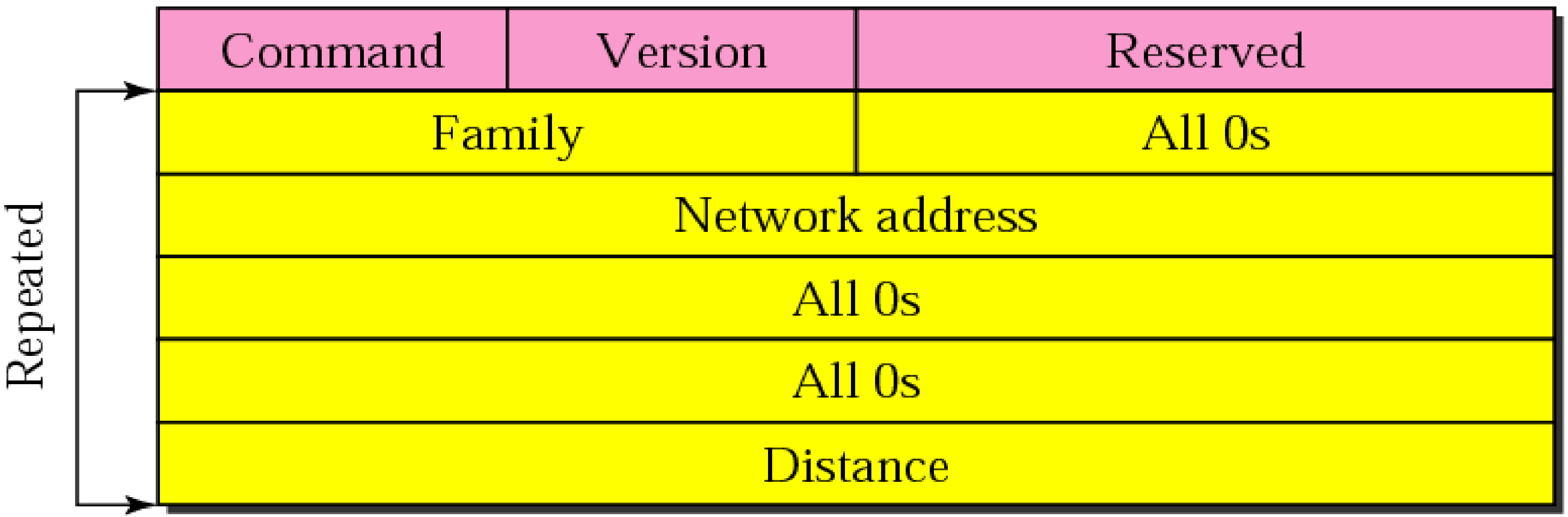
| Dest. | Hop | Next |
|---|---|---|
| 130.10.0.0 | 2 | 195.2.5.1 |
| 130.11.0.0 | 3 | 195.2.5.1 |
| 195.2.4.0 | 2 | 195.2.5.1 |
| 195.2.5.0 | 1 | ——— |
| 195.2.6.0 | 1 | ——— |
| 205.5.5.0 | 4 | 195.2.5.1 |
| 205.5.6.0 | 4 | 195.2.5.1 |

R3 Table

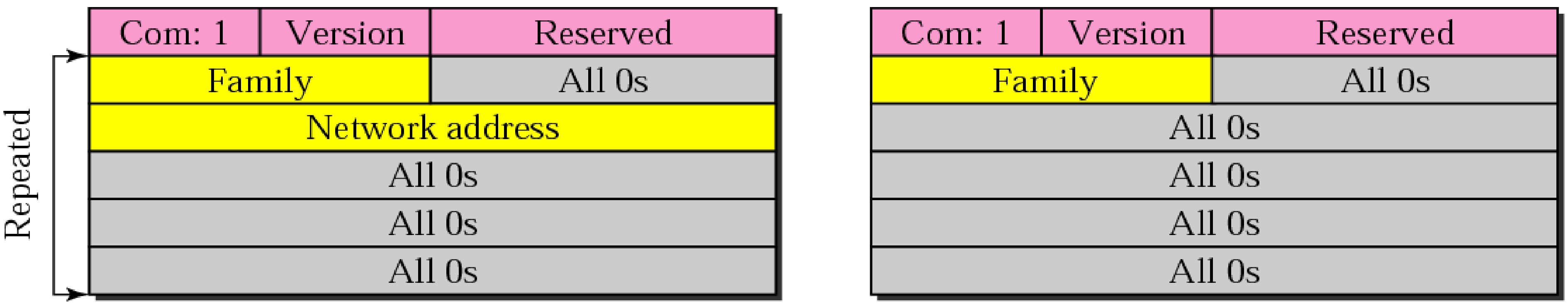
| Dest. | Hop | Next |
|---|---|---|
| 130.10.0.0 | 2 | 130.11.0.2 |
| 130.11.0.0 | 1 | ——— |
| 195.2.4.0 | 3 | 130.11.0.2 |
| 195.2.5.0 | 3 | 130.11.0.2 |
| 195.2.6.0 | 4 | 130.11.0.2 |
| 205.5.5.0 | 1 | ——— |
| 205.5.6.0 | 1 | ——— |

R4 Table

## **Figure 14.9** *RIP message format*



# **Figure 14.10** ***Request messages***




a. Request for some

b. Request for all

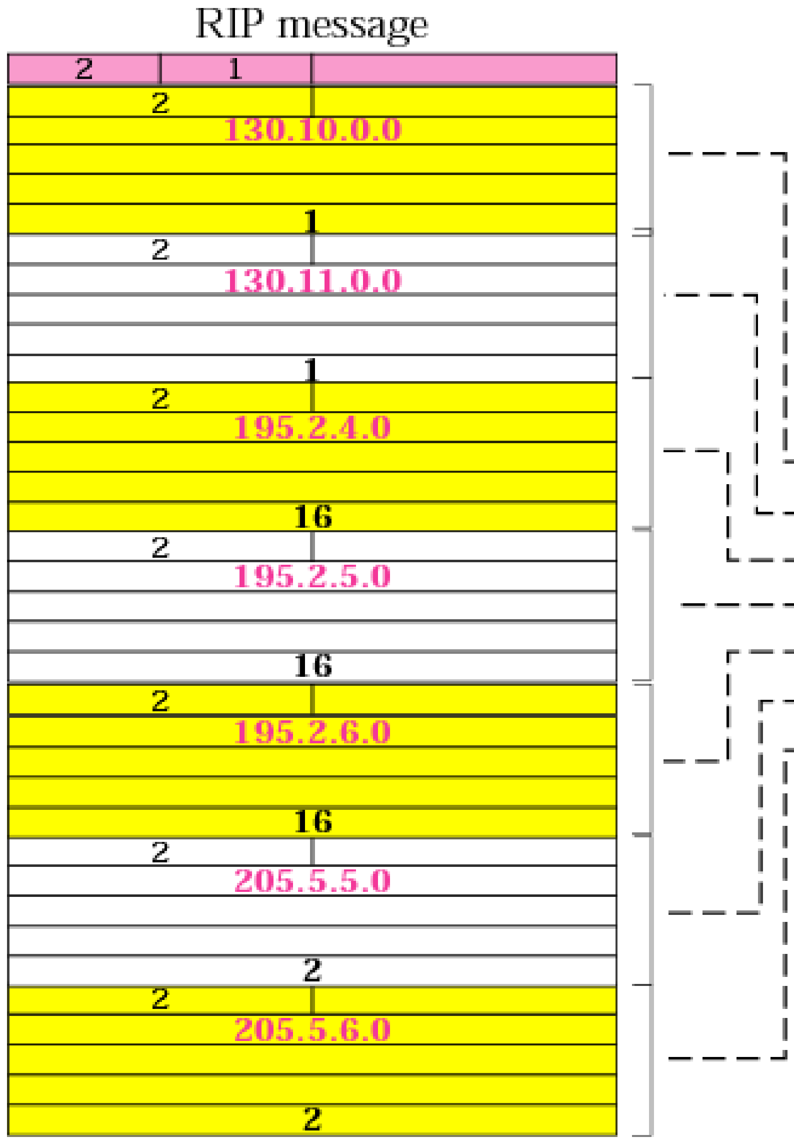
## Example 1

***Figure 14.11 shows the update message sent from router R1 to router R2 in Figure 14.8. 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.***

**See Next Slide**

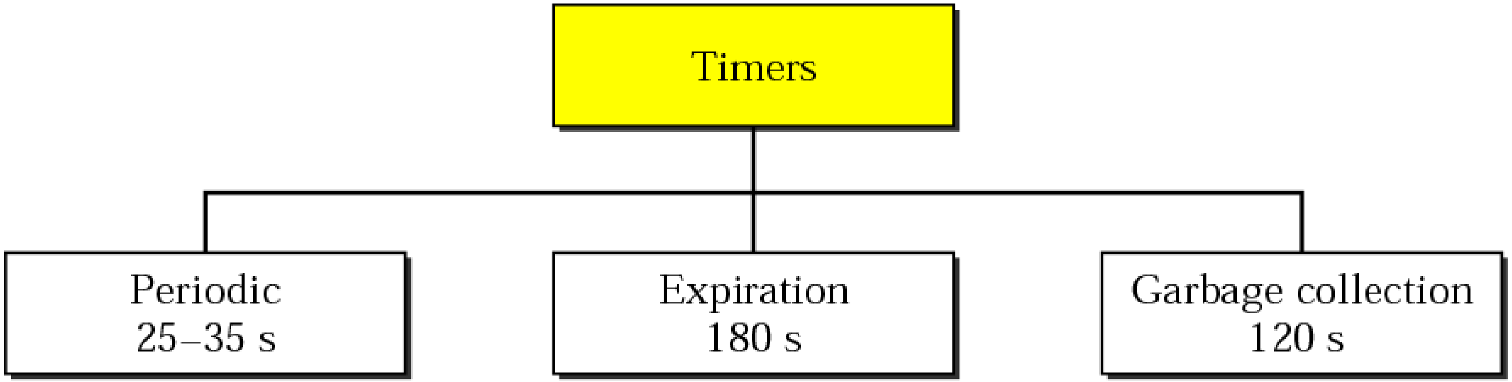
## Figure 14.11 *Solution to Example 1*




| Dest. | Hop |
|---|---|
| 130.10.0.0 | 1 |
| 130.11.0.0 | 1 |
| 195.2.4.0 | 16 |
| 195.2.5.0 | 16 |
| 195.2.6.0 | 16 |
| 205.5.5.0 | 2 |
| 205.5.6.0 | 2 |

Table extracted from message before incrementing

# Figure 14.12 *RIP timers*



## *Example 2*

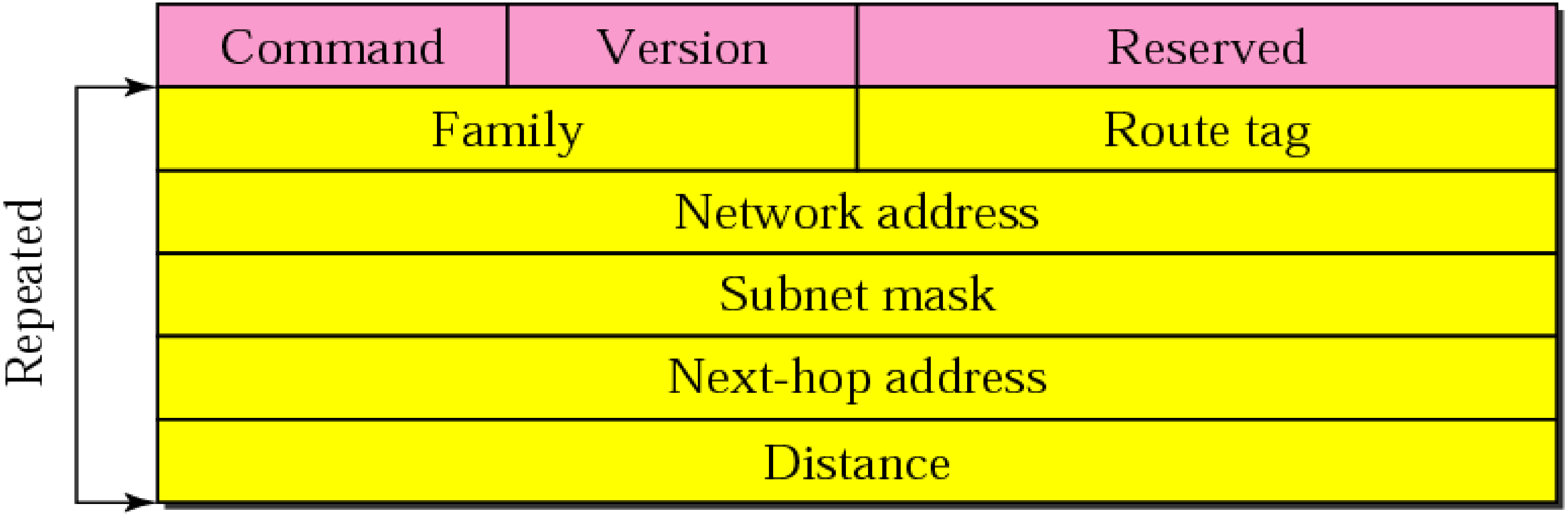
***A routing table has 20 entries. It does not receive information about five routes for 200 s. How many timers are running at this time?***

***Solution***

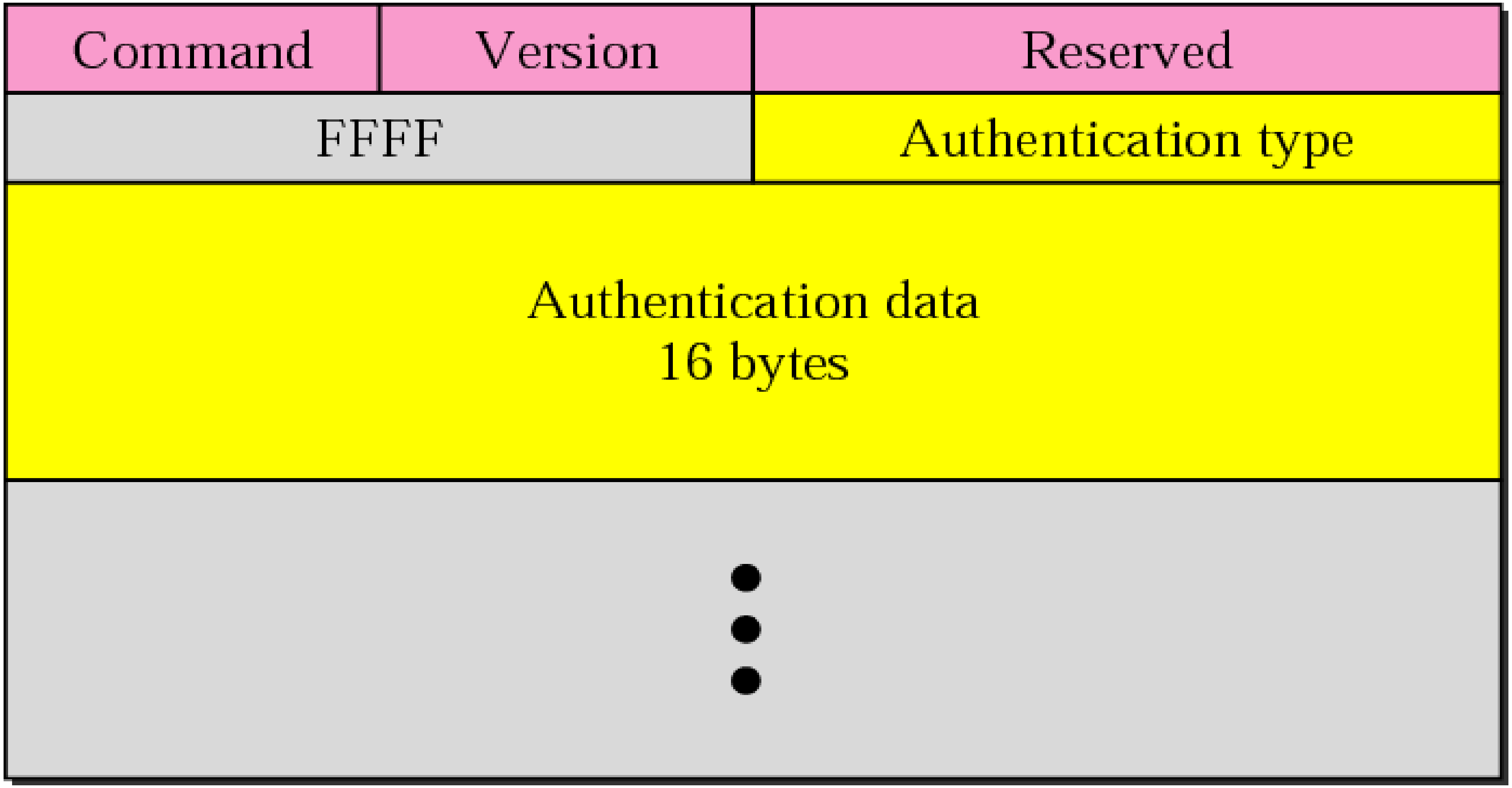
***The 21 timers are listed below:***

- ***Periodic timer: 1***
- ***Expiration timer: 20 − 5 = 15***
- ***Garbage collection timer: 5***

## **Figure 14.13** *RIP version 2 format*



## **Figure 14.14** *Authentication*



Note:

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

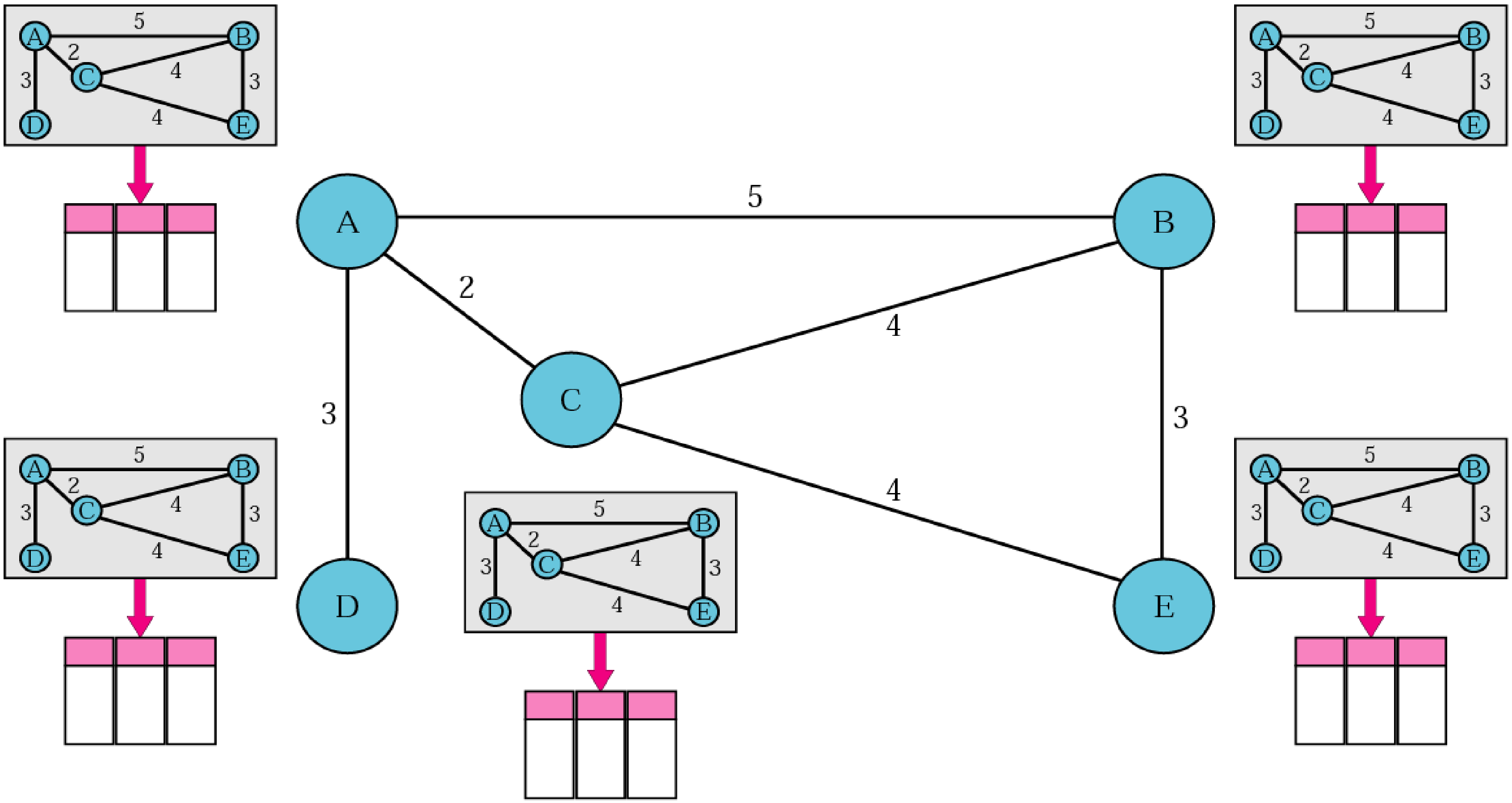
# 14.4 LINK STATE ROUTING

***In link state routing, if each node in the domain has the entire topology of the domain, the node can use Dijkstra's algorithm to build a routing table.***

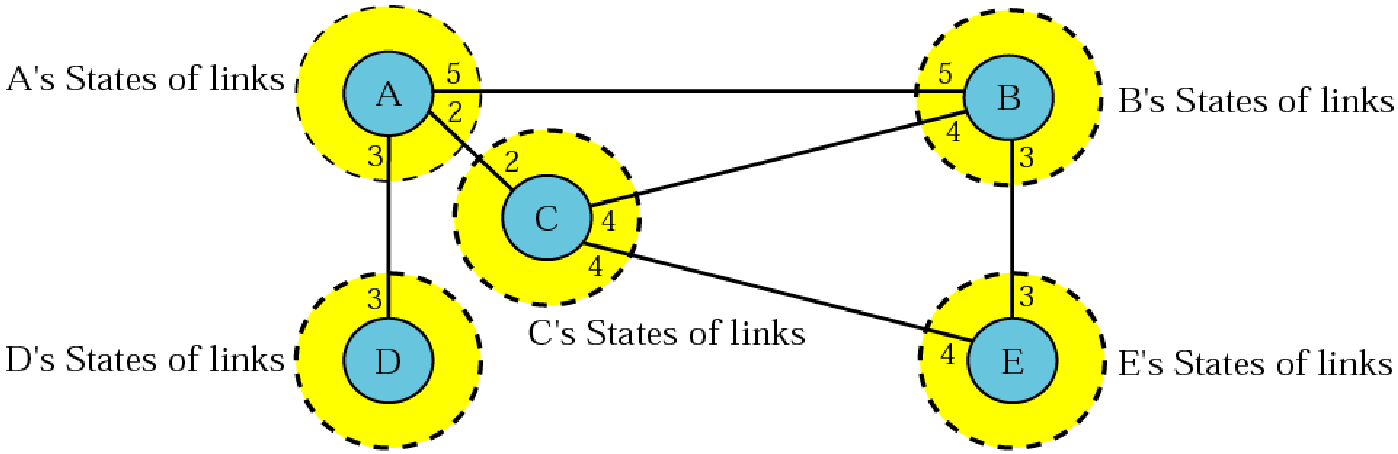
***The topics discussed in this section include:***

***Building Routing Tables***

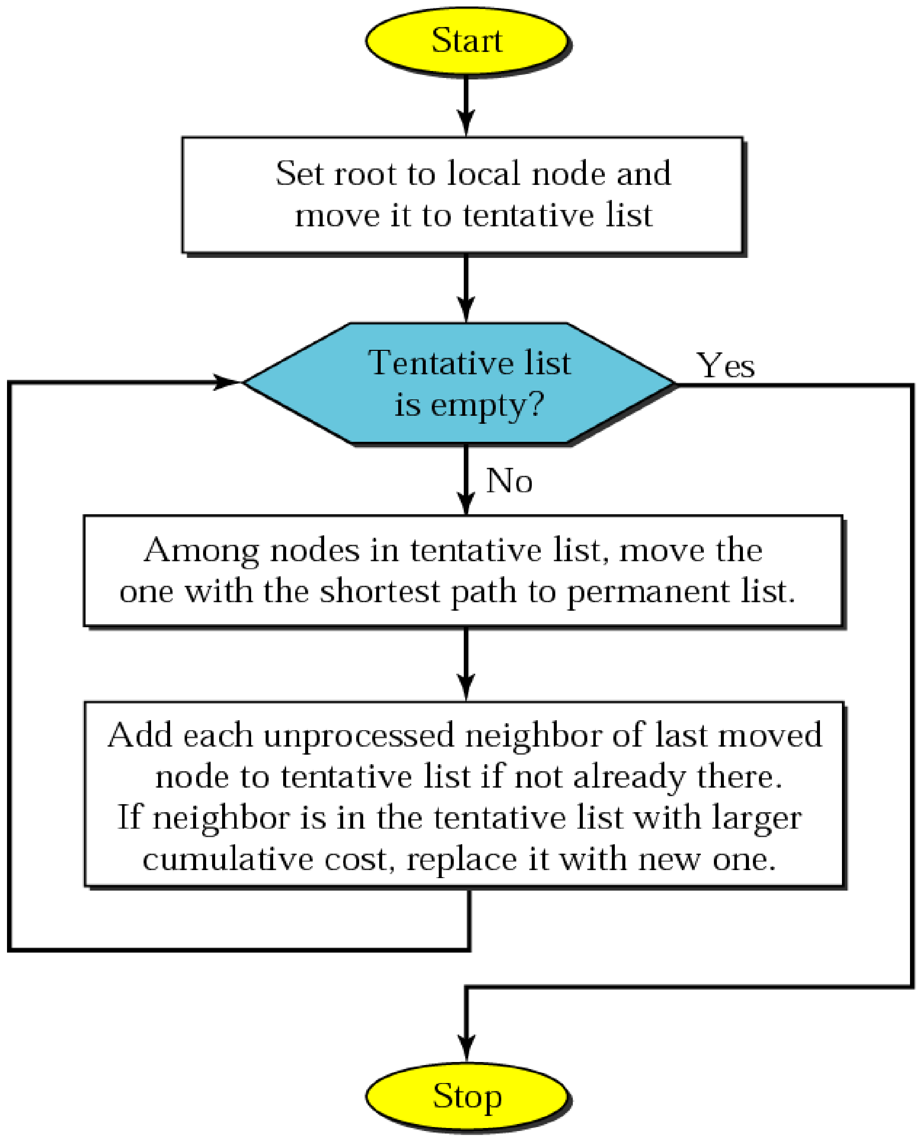
# Figure 14.15 *Concept of link state routing*



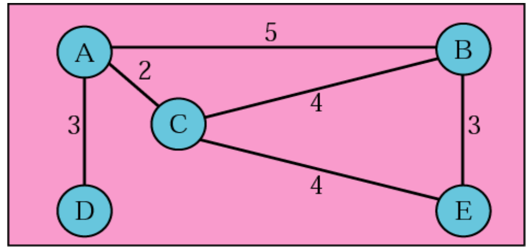
# Figure 14.16 *Link state knowledge*



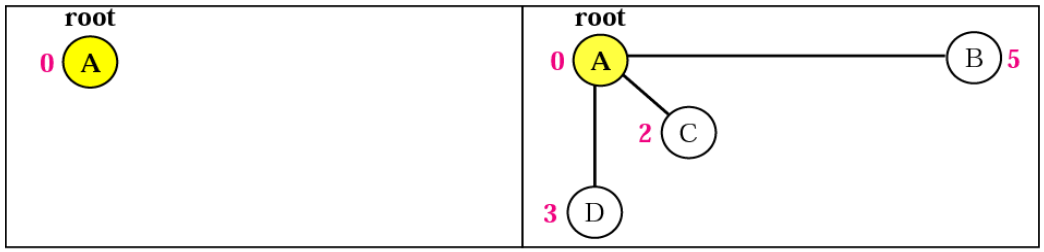
## Figure 14.17 *Dijkstra algorithm*



# Figure 14.18 *Example of formation of shortest path tree*

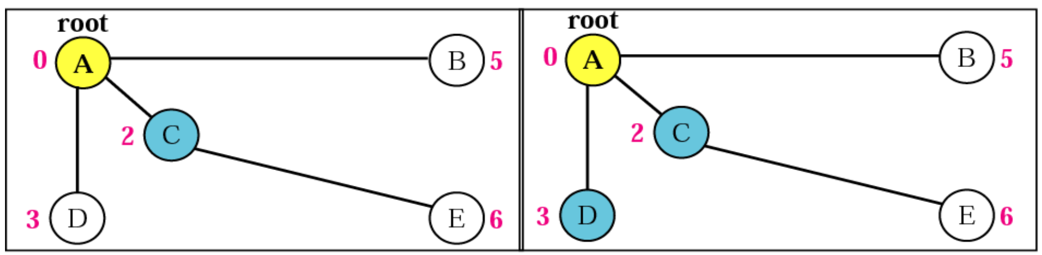



Topology



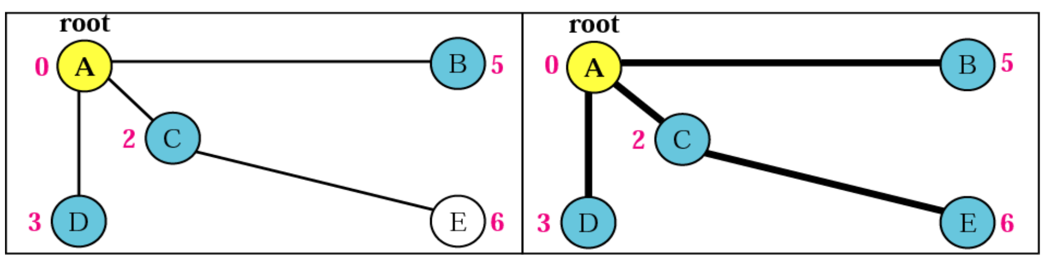

1. Set root to A and move A to tentative list

2. Move A to permanent list and add B, C, and D to tentative list




3. Move C to permanent and add E to tentative list

4. Move D to permanent list.




5. Move B to permanent list

6. Move E to permanent list (tentative list is empty)

**Table 14.1** ***Routing table for node A***

| *Node* | *Cost* | *Next Router* |
|---|---|---|
| A | 0 | — |
| B | 5 | — |
| C | 2 | — |
| D | 3 | — |
| E | 6 | C |

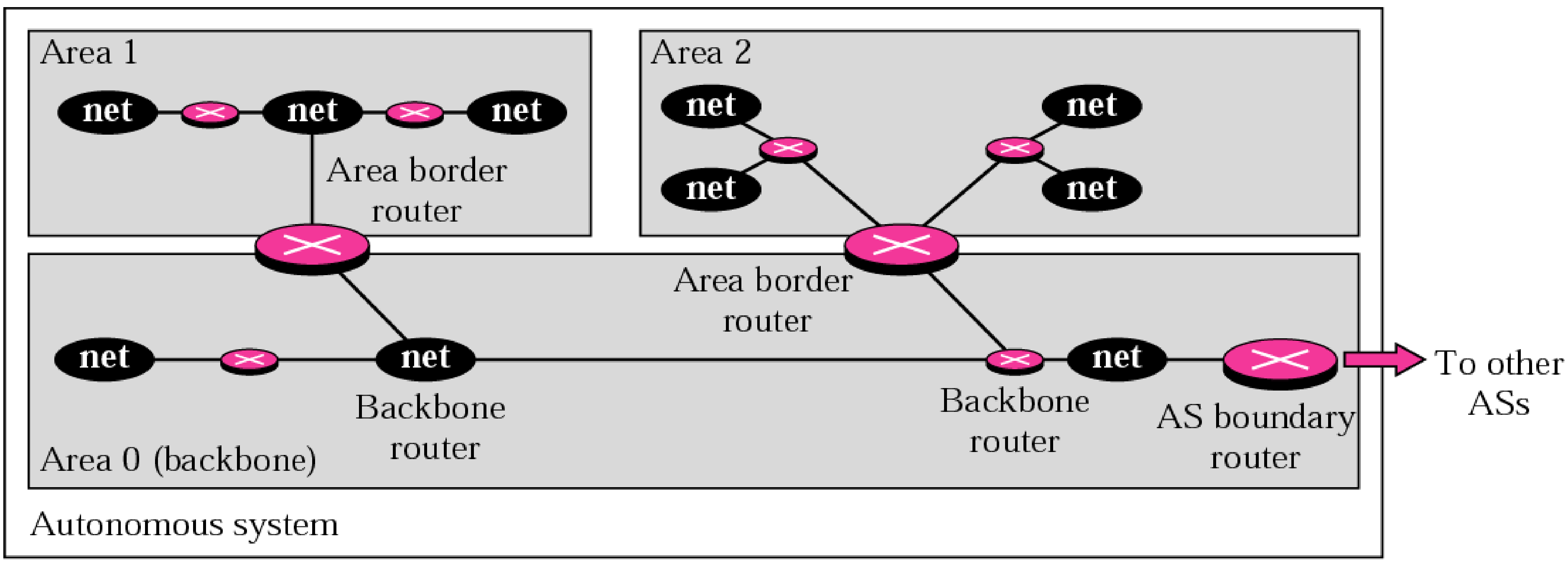
# 14.5 OSPF

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

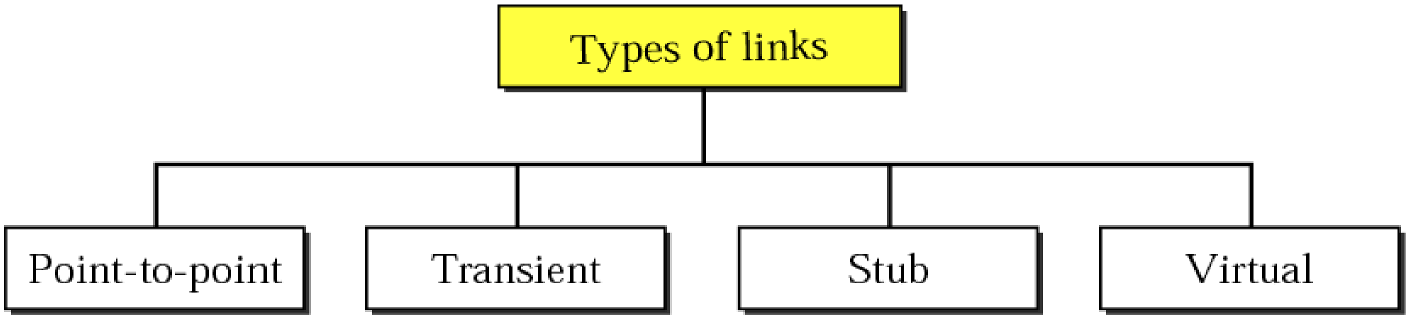
***The topics discussed in this section include:***

***Areas***
***Metric***
***Types of Links***
***Graphical Representation***
***OSPF Packets***
***Link State Update Packet***
***Other Packets***
***Encapsulation***

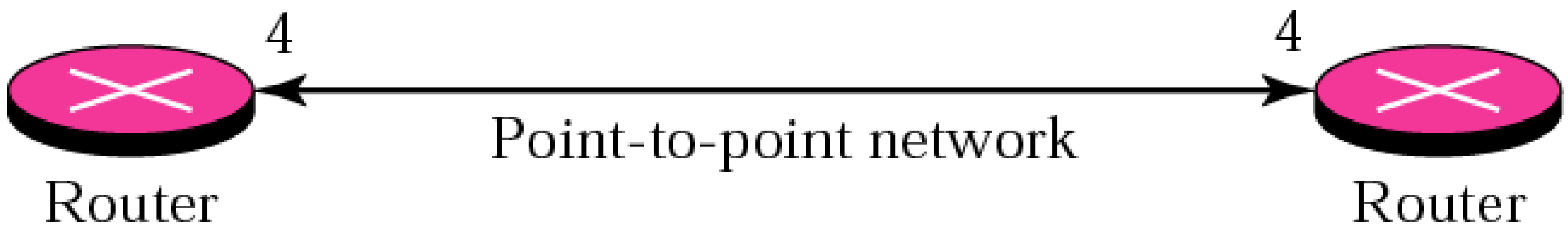
## Figure 14.19 *Areas in an autonomous system*



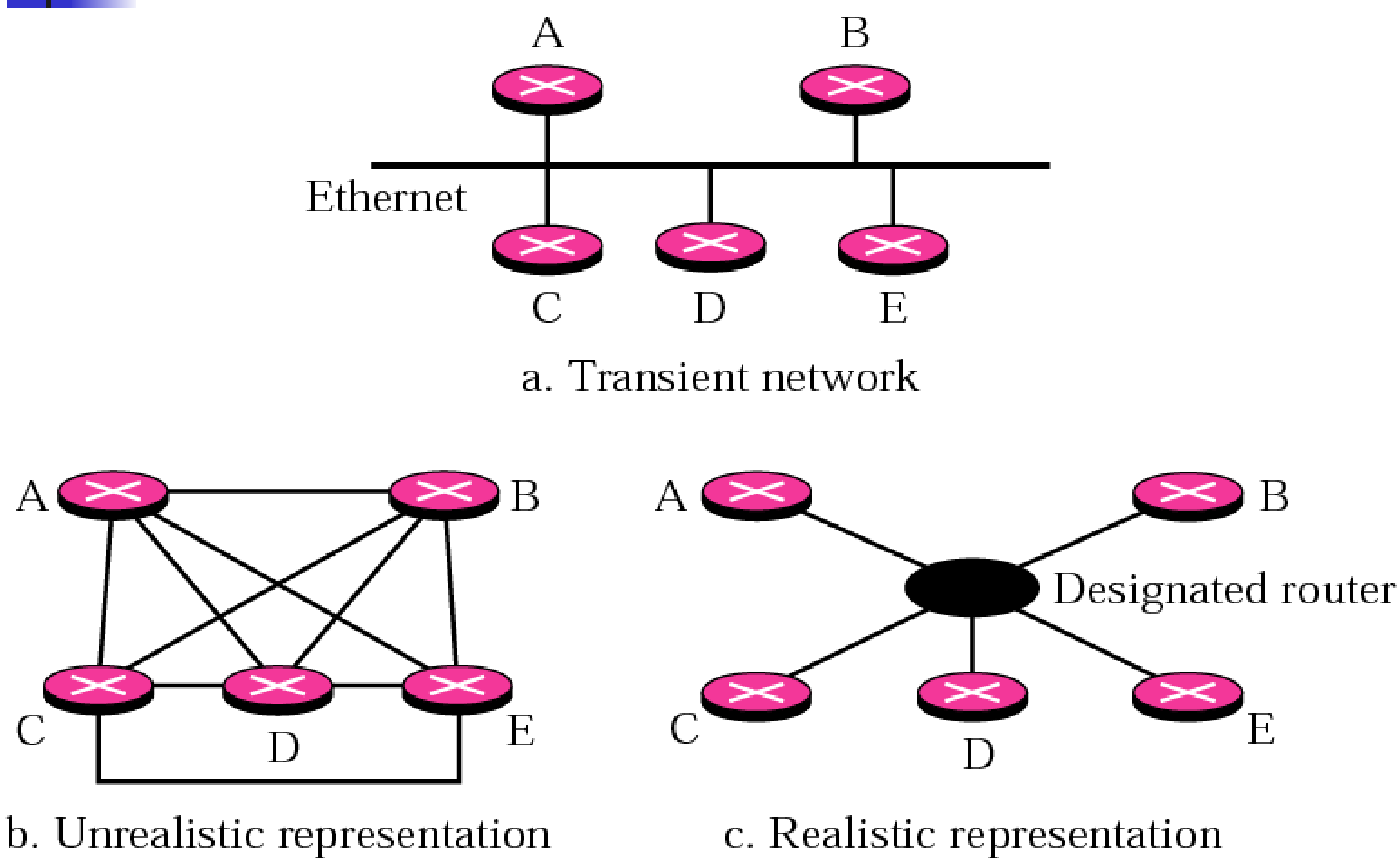
## Figure 14.20 *Types of links*



# Figure 14.21 *Point-to-point link*



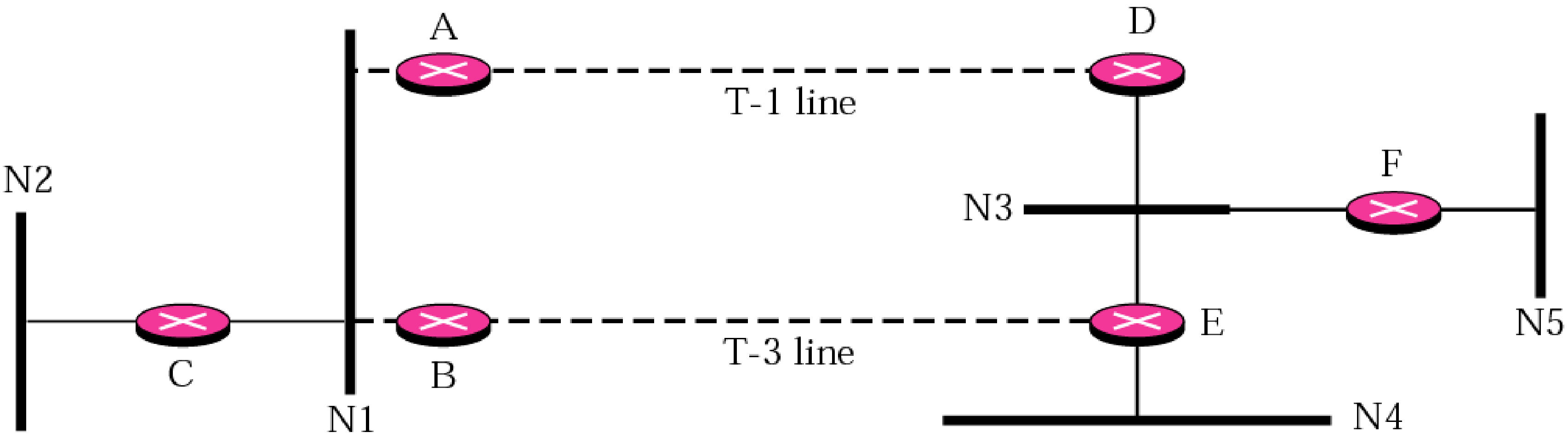
## Figure 14.22 *Transient link*



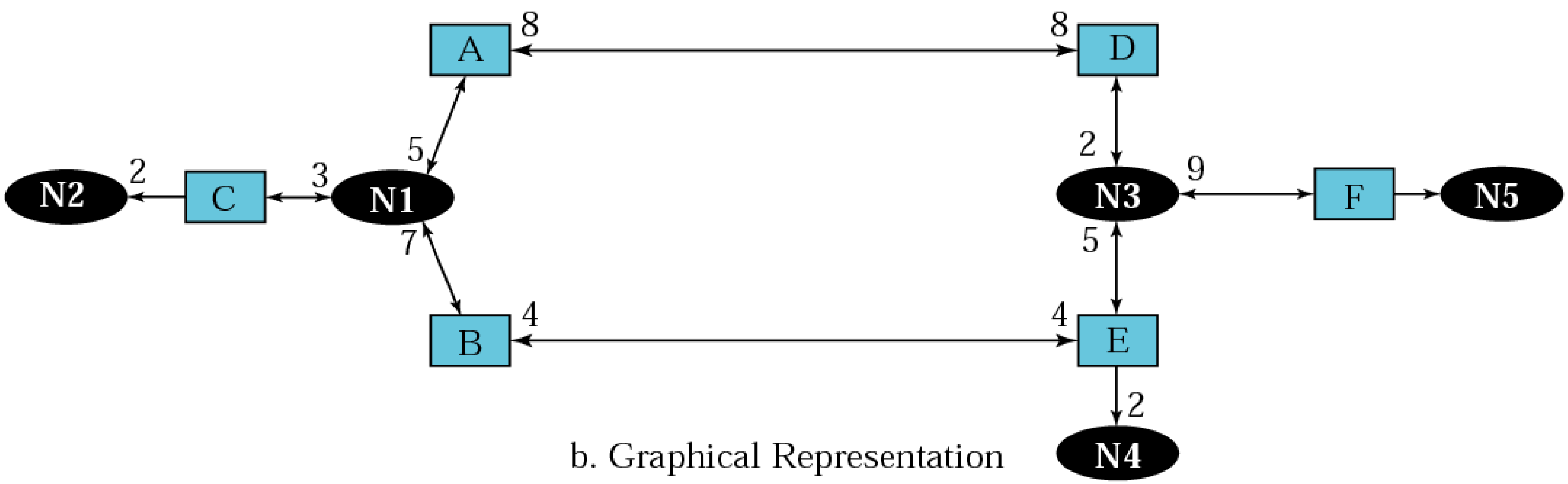
## Figure 14.23 *Stub link*


a. Stub network　　b. Representation

# Figure 14.24 *Example of an AS and its graphical representation in OSPF*

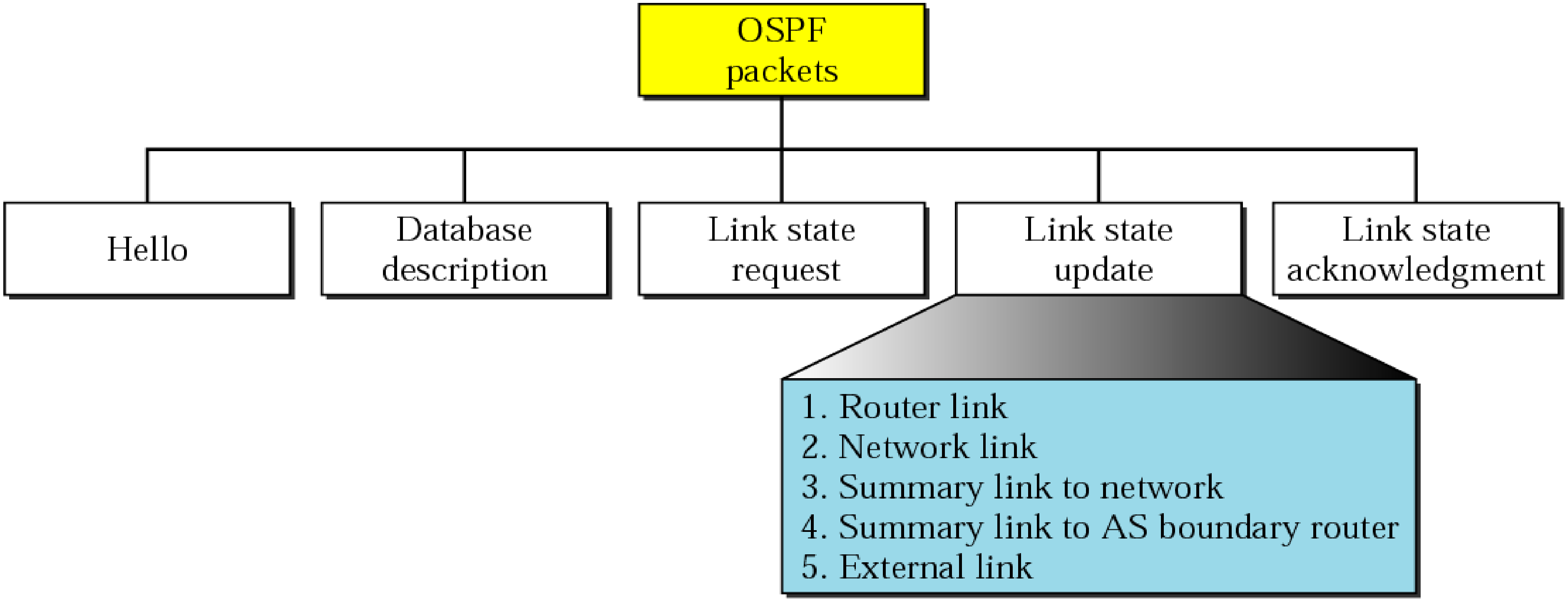



a. Autonomous System




b. Graphical Representation

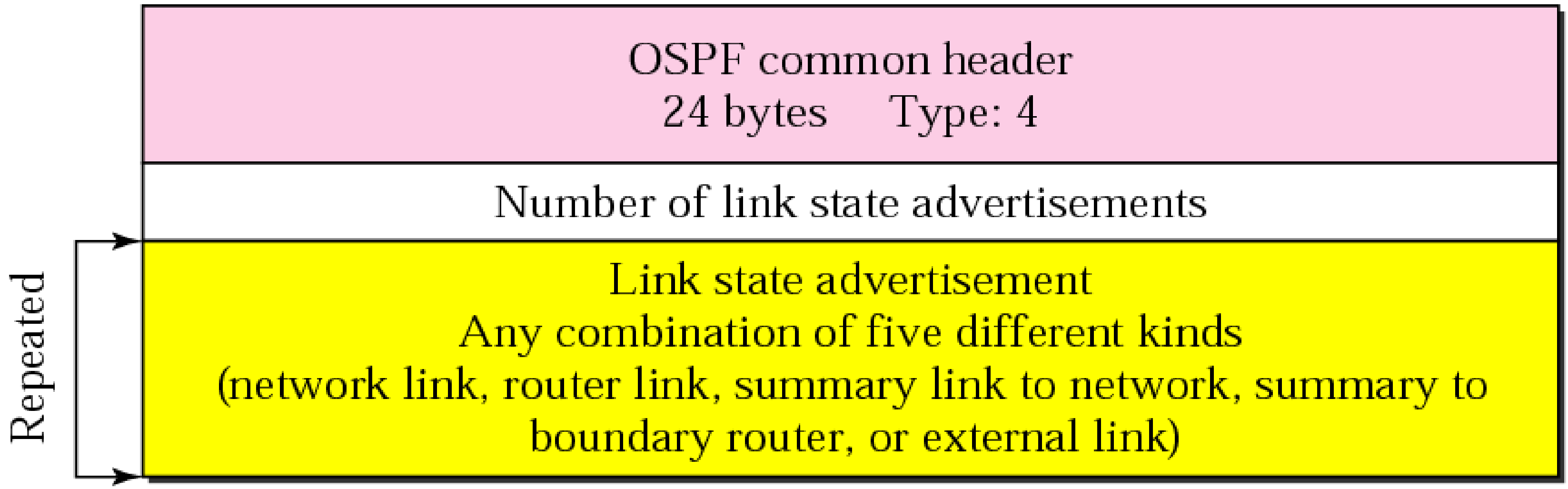
## Figure 14.25 *Types of OSPF packets*



## Figure 14.26 *OSPF common header*

| 0–7 | 8–15 | 16–31 |
|---|---|---|
| Version | Type | Message length |
| Source router IP address | | |
| Area Identification | | |
| Checksum | | Authentication type |
| Authentication (32 bits) | | |

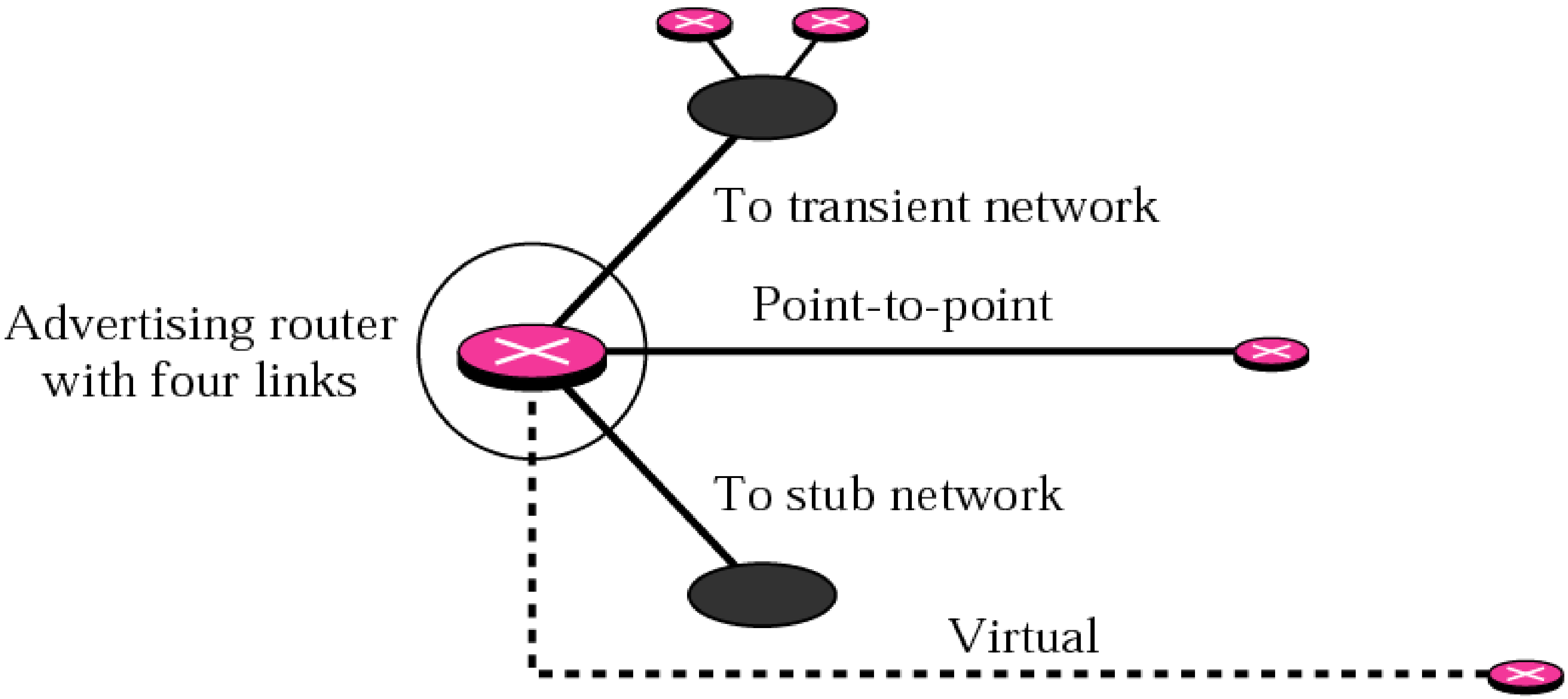
## Figure 14.27 *Link state update packet*



## Figure 14.28 *LSA general header*

<table>
<tr><td>Link state age</td><td>Reserved</td><td>E</td><td>T</td><td>Link state type</td></tr>
<tr><td colspan="5">Link state ID</td></tr>
<tr><td colspan="5">Advertising router</td></tr>
<tr><td colspan="5">Link state sequence number</td></tr>
<tr><td>Link state checksum</td><td colspan="4">Length</td></tr>
</table>

## Figure 14.29 *Router link*



## **Figure 14.30** *Router link LSA*

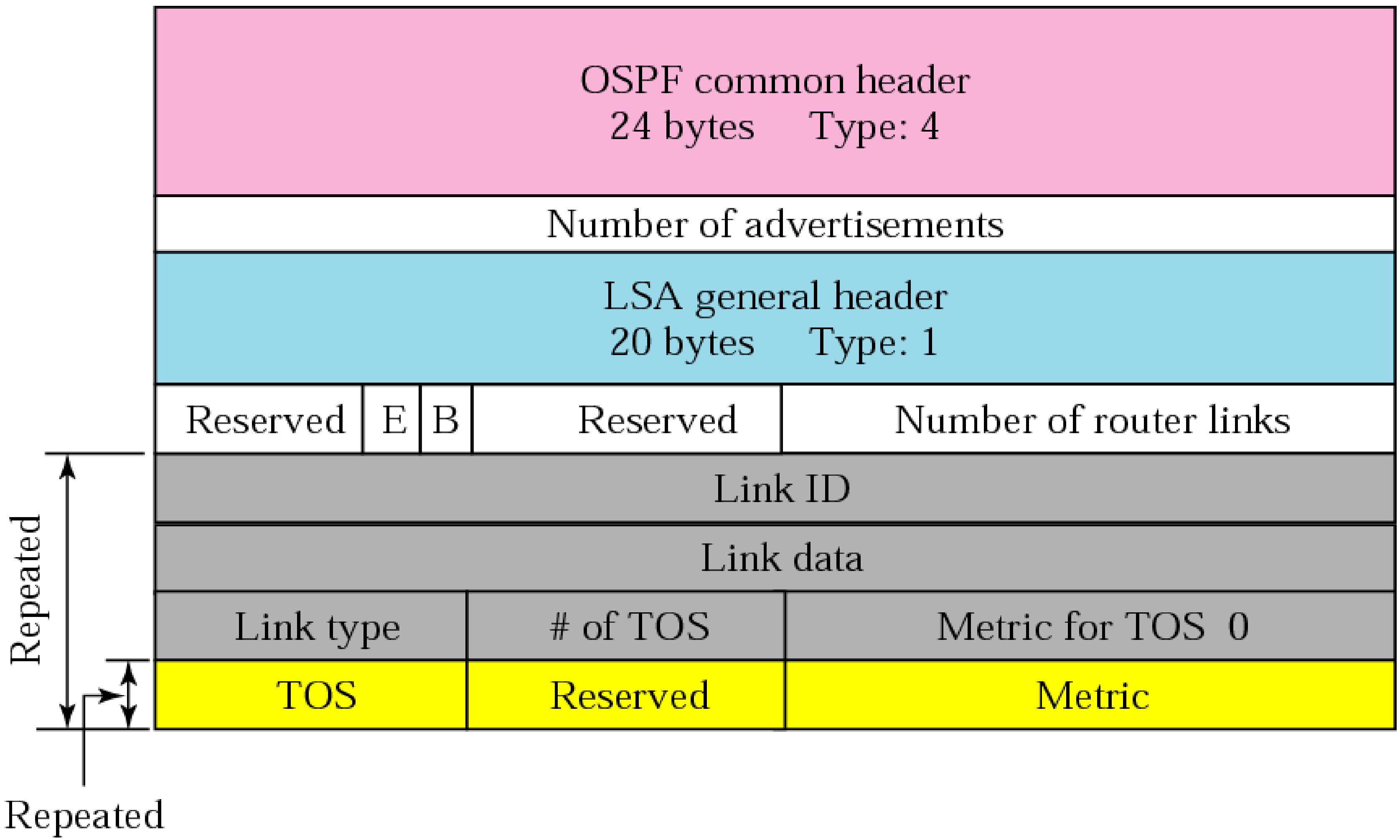


**Table 14.2** ***Link types, link identification, and link data***

| *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 3*

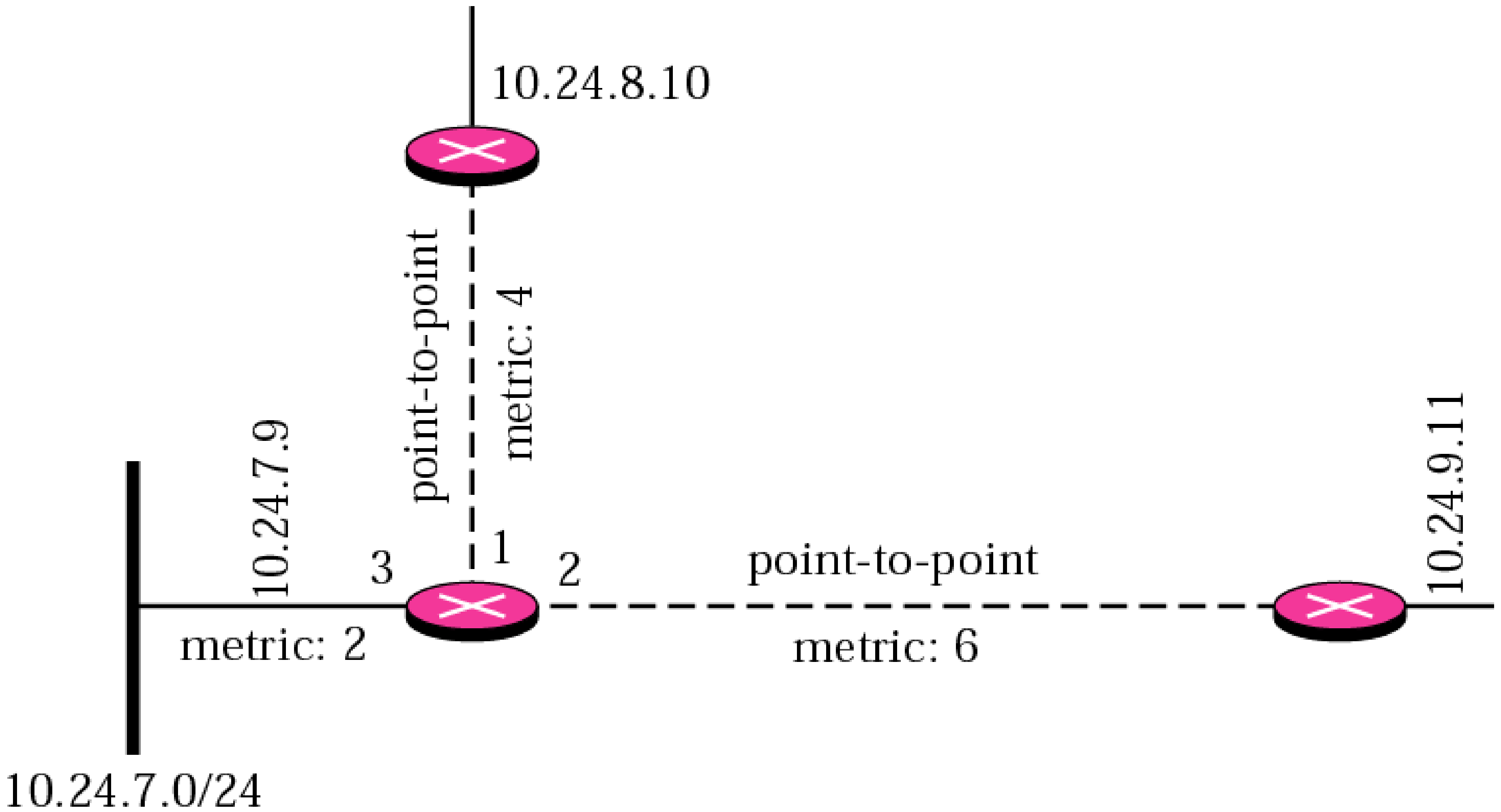
***Give the router link LSA sent by router 10.24.7.9 in Figure 14.31.***

**See Next Slide**

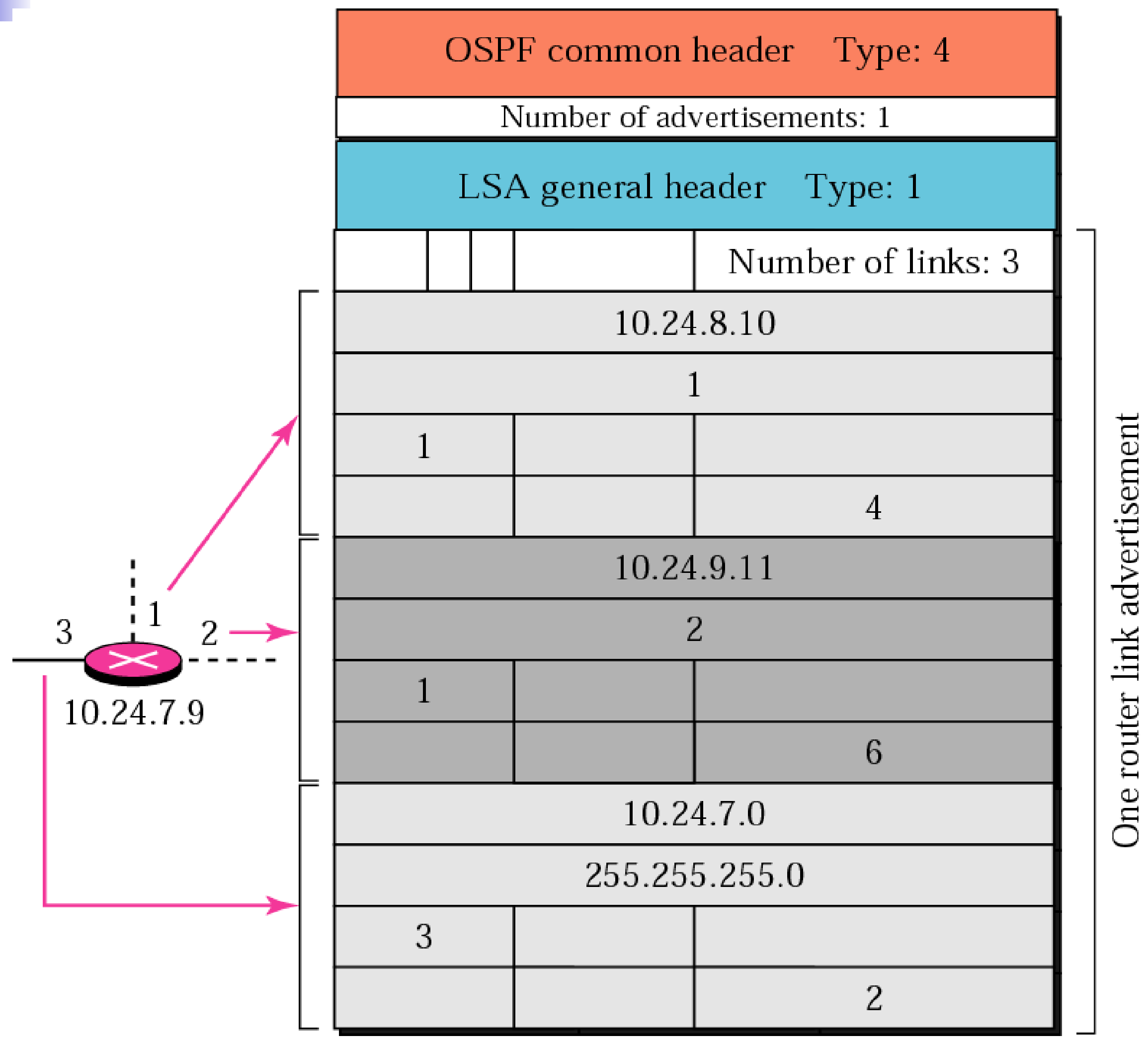
### *Solution*

***This router has three links: two of type 1 (point-to-point) and one of type 3 (stub network). Figure 14.32 shows the router link LSA.***

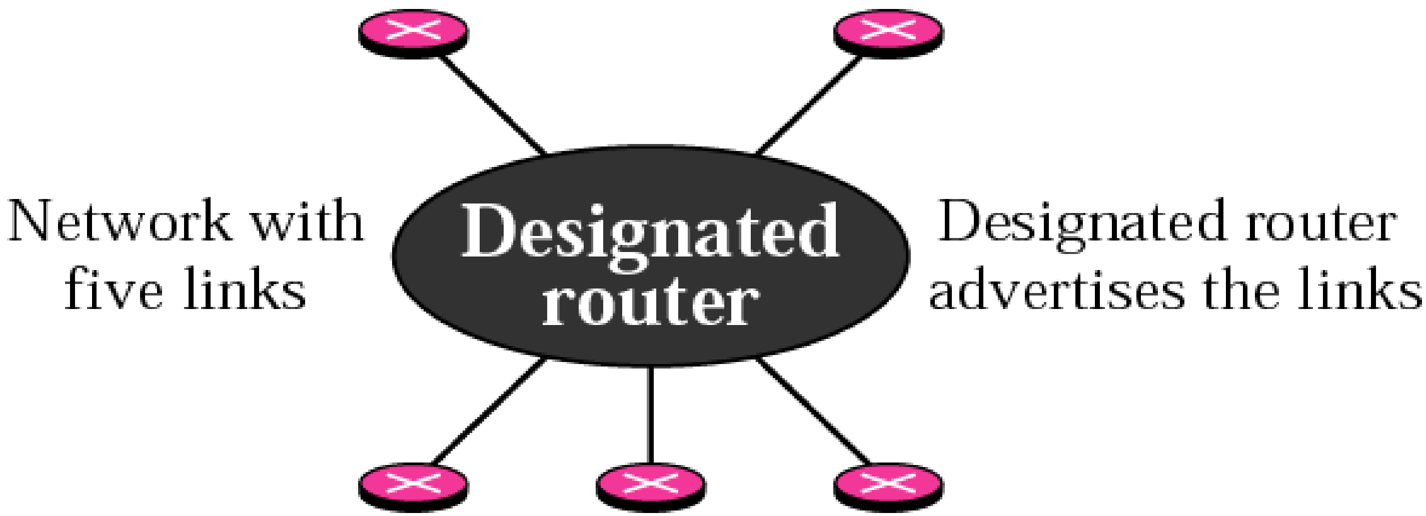
**See Figure 14.32**


10.24.8.10
point-to-point
metric: 4
10.24.7.9
3
1
2
point-to-point
10.24.9.11
metric: 2
metric: 6
10.24.7.0/24

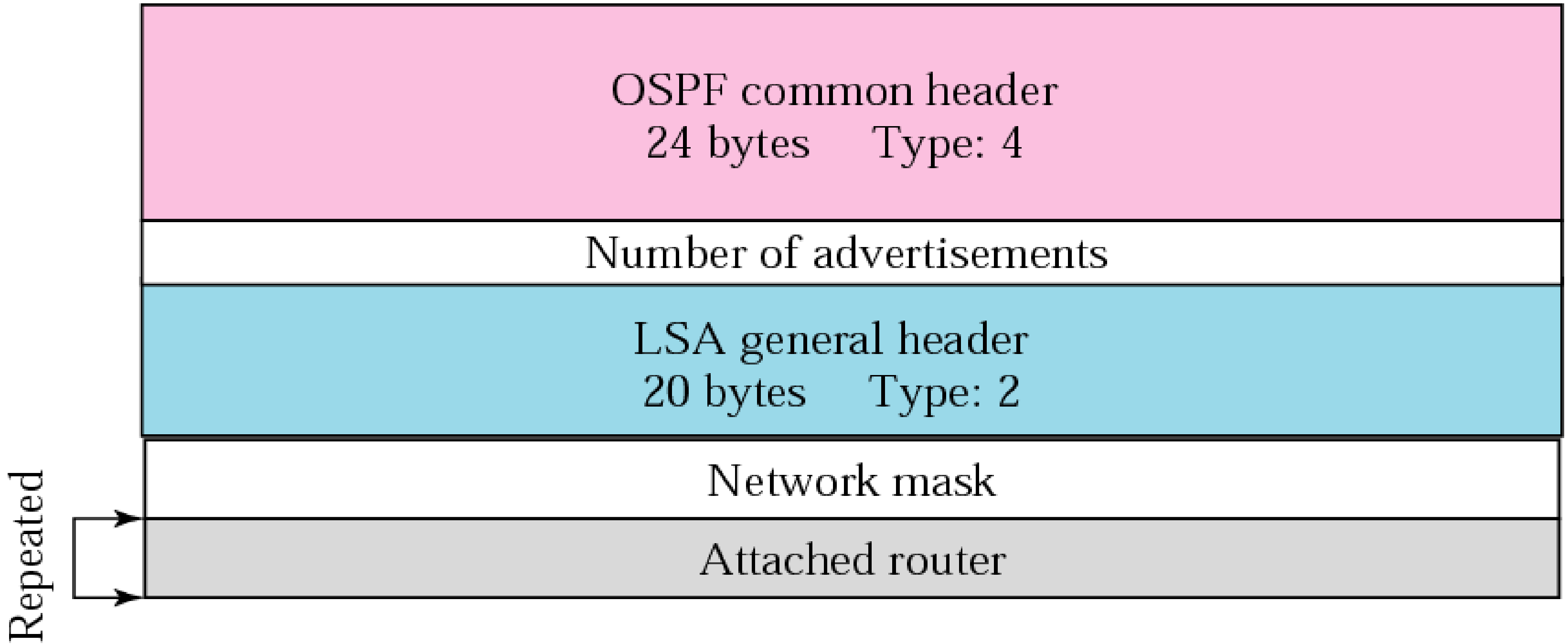
## Figure 14.32 *Solution to Example 3*



# Figure 14.33 *Network link*



## **Figure 14.34** *Network link advertisement format*



## *Example 4*

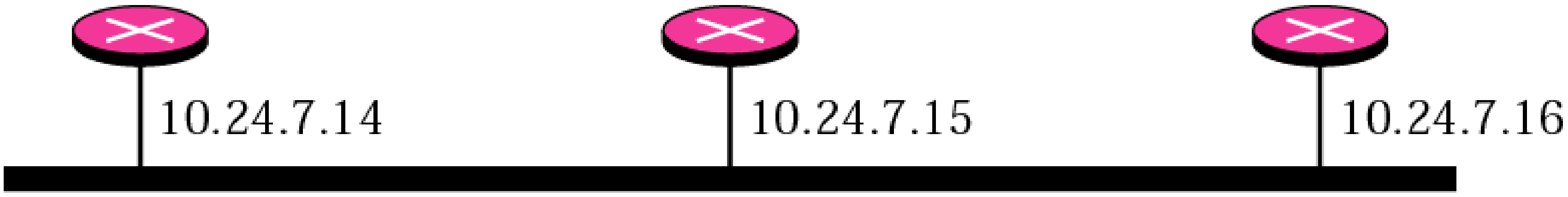
***Give the network link LSA in Figure 14.35.***

**See Next Slide**

***Solution*.**

**See Figure 14.36**

## Figure 14.35 *Example 4*



## **Figure 14.36** *Solution to Example 4*

| |
|---|
| OSPF common header       Type: 4 |
| Number of advertisements: 1 |
| LSA general header       Type: 2 |
| 255.255.255.0 |
| 10.24.7.14 |
| 10.24.7.15 |
| 10.24.7.16 |

## *Example 5*

***In Figure 14.37, which router(s) sends out router link LSAs?***

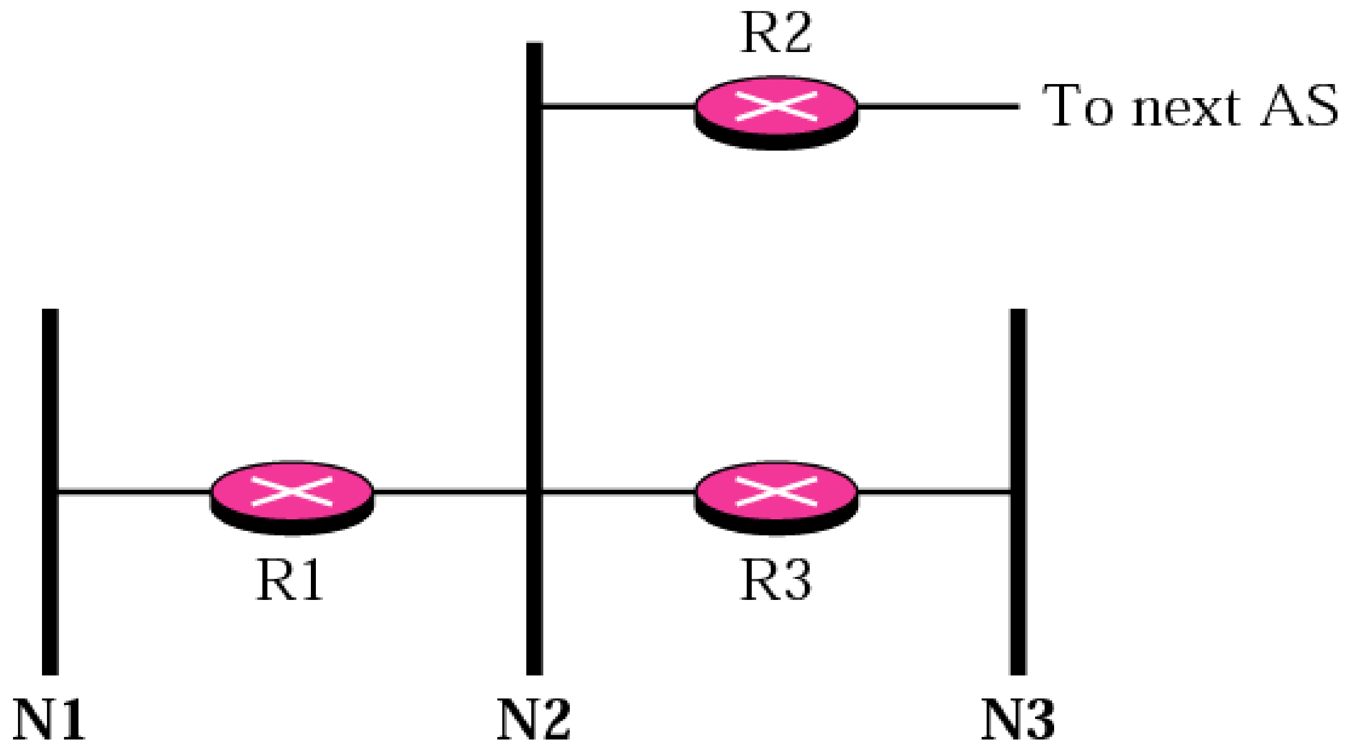
**See Next Slide**

***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 14.37** *Example 5 and Example 6*



# *Example 6*

***In Figure 14.37, which router(s) sends out the network link LSAs?***

## *Solution*

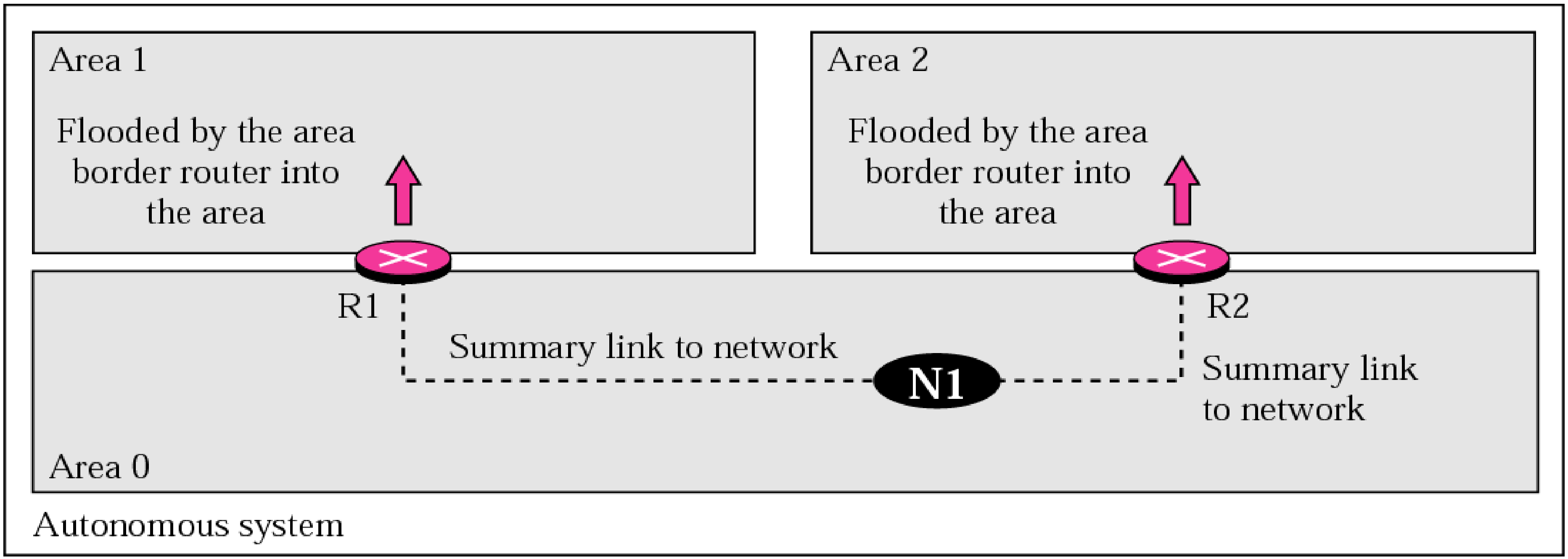
***All three network must advertise network links:***

*a.* ***Advertisement for N1 is done by R1 because it is the only attached router and therefore the designated router.***

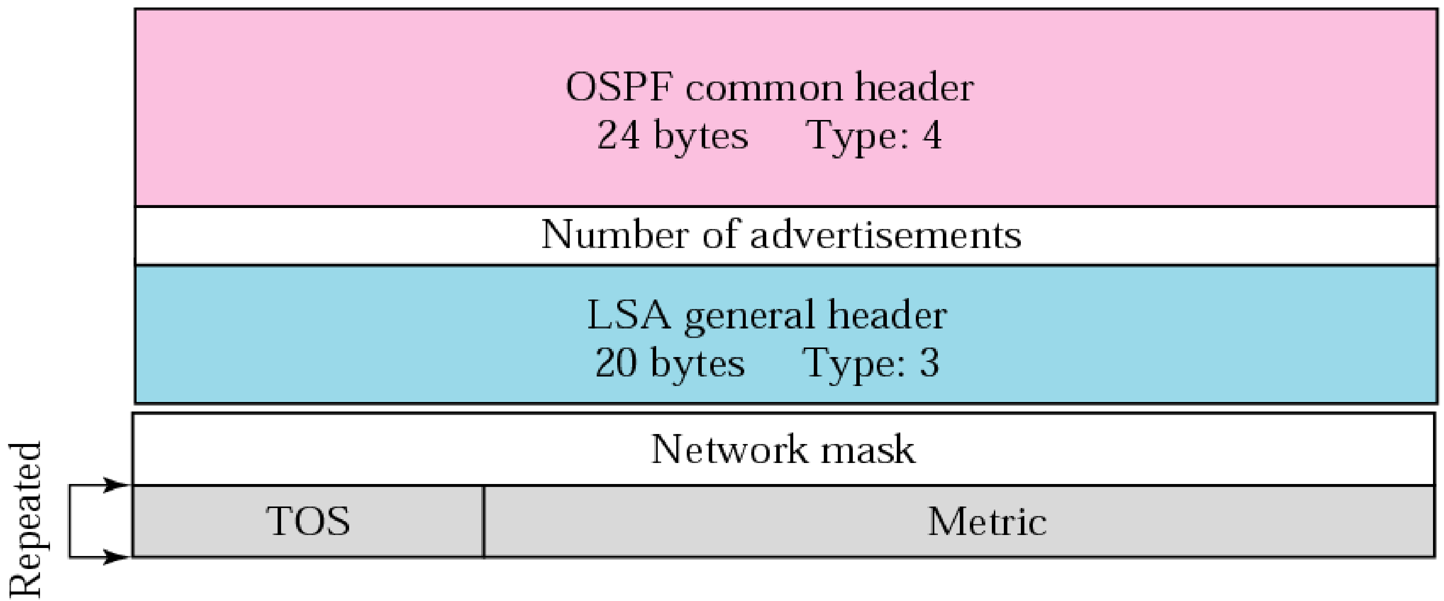
*b.* ***Advertisement for N2 can be done by either R1, R2, or R3, depending on which one is chosen as the designated router.***

*c.* ***Advertisement for N3 is done by R3 because it is the only attached router and therefore the designated router.***

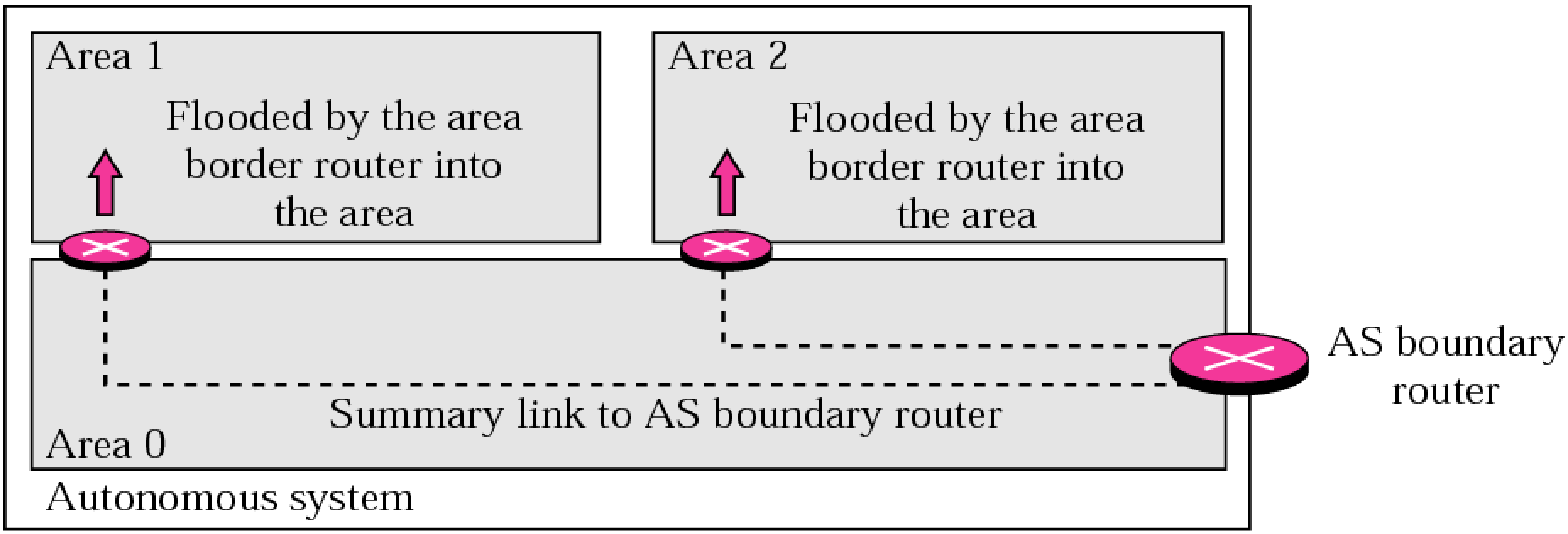
## Figure 14.38 *Summary link to network*



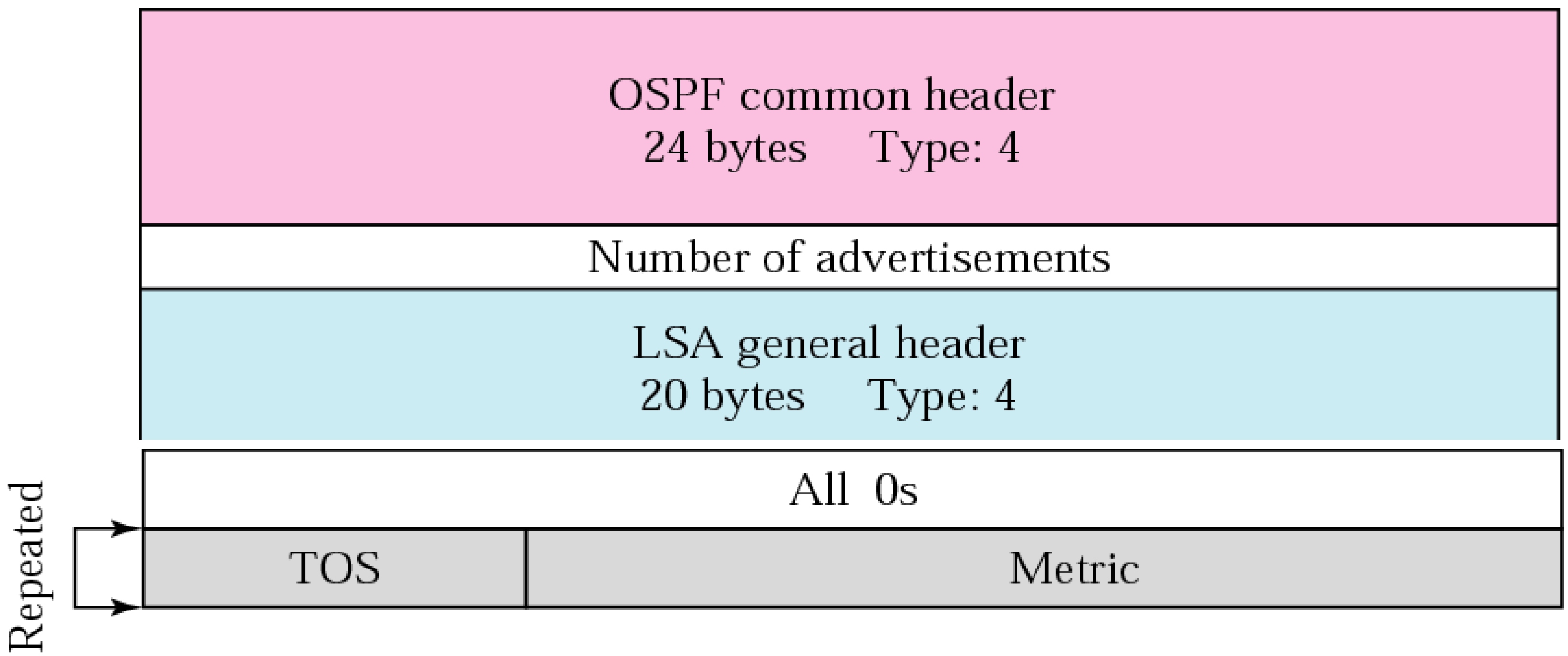
## Figure 14.39 *Summary link to network LSA*



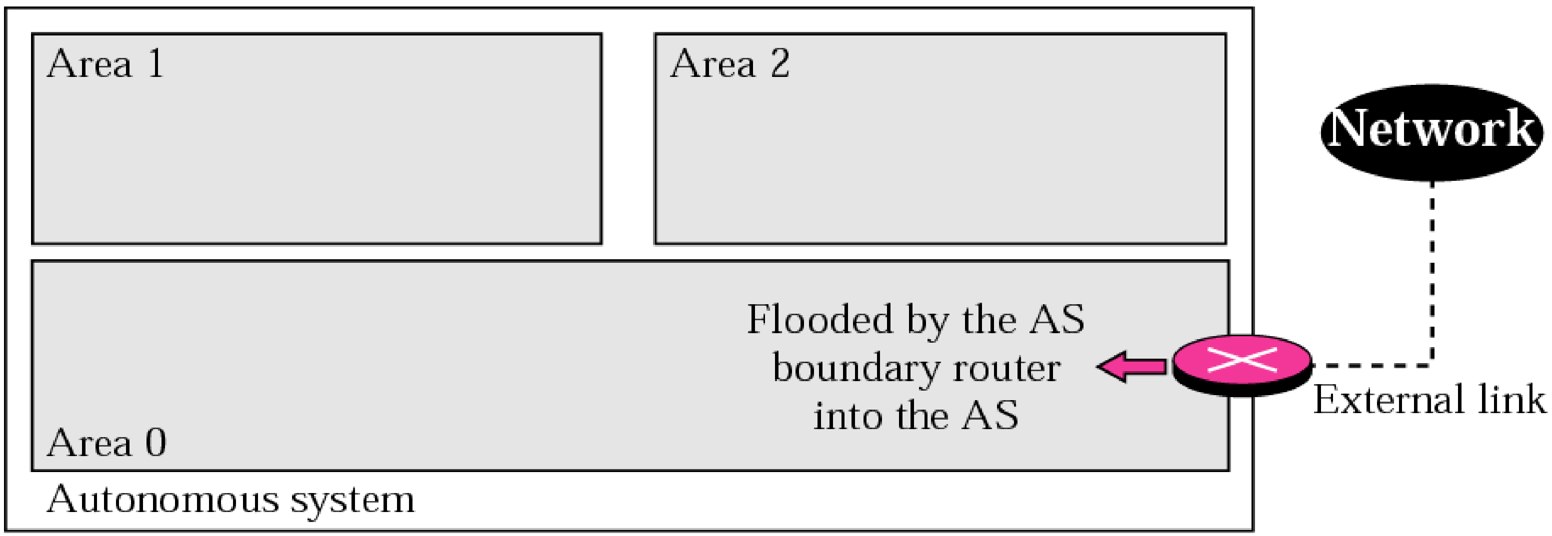
## Figure 14.40 *Summary link to AS boundary router*



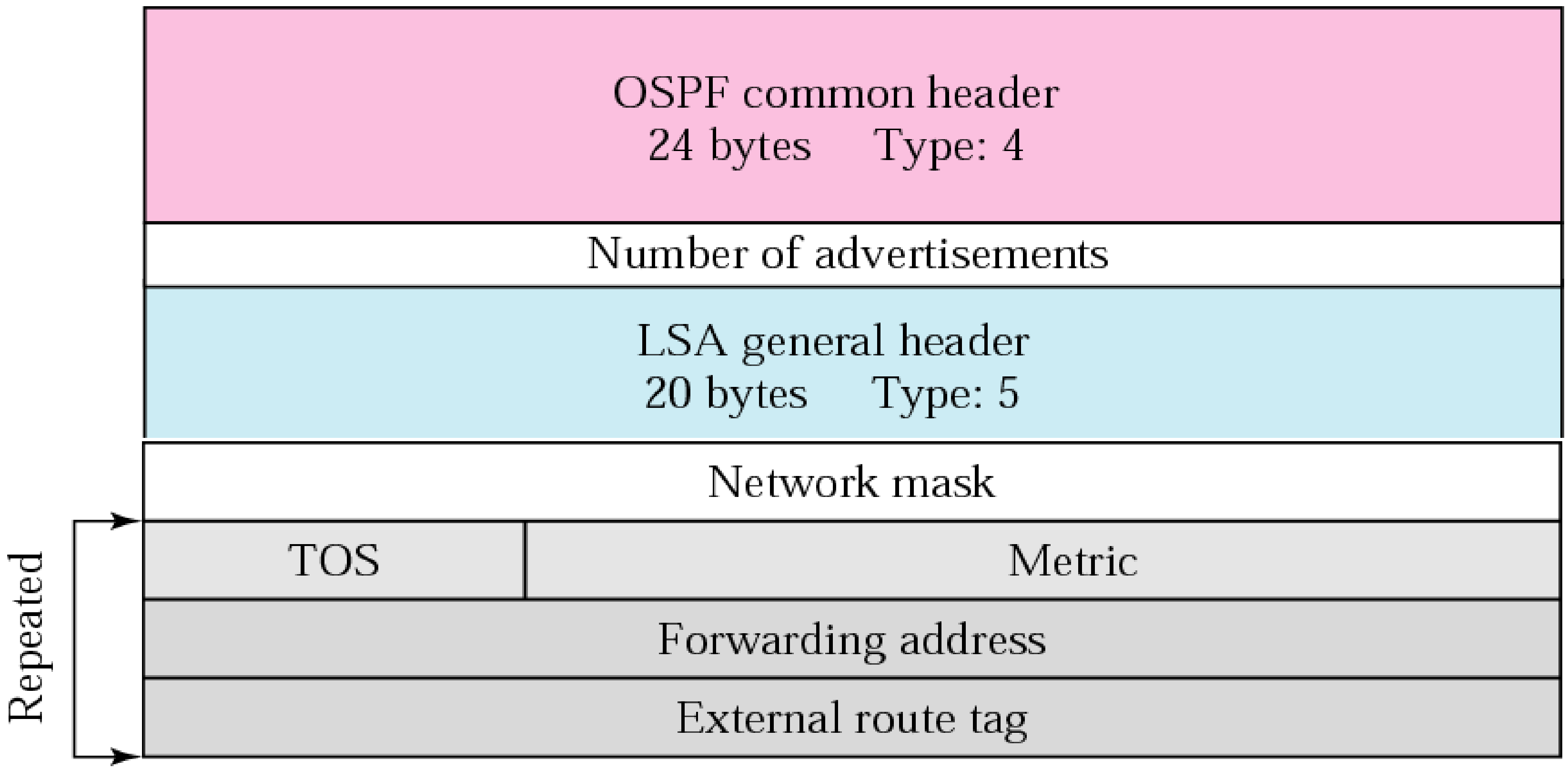
## Figure 14.41 *Summary link to AS boundary router LSA*

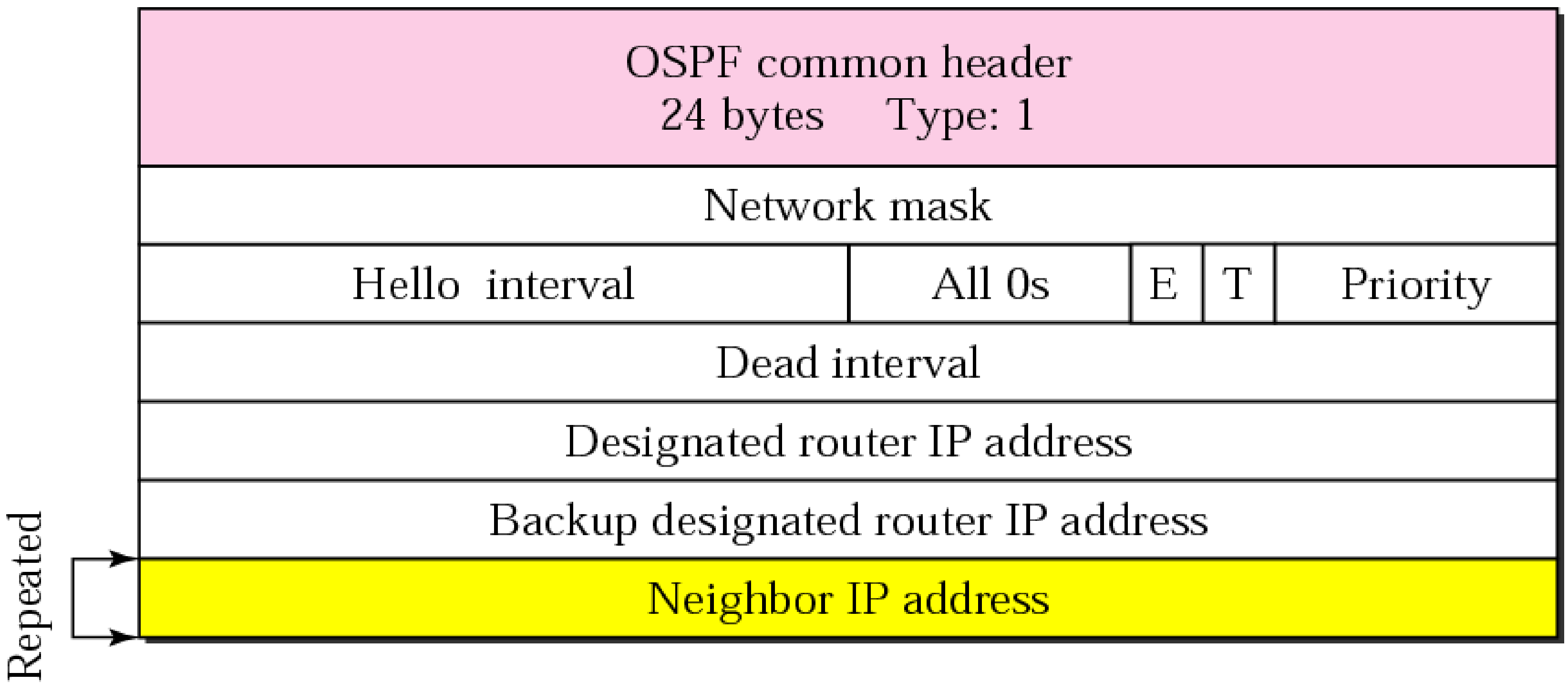


## Figure 14.42 *External link*

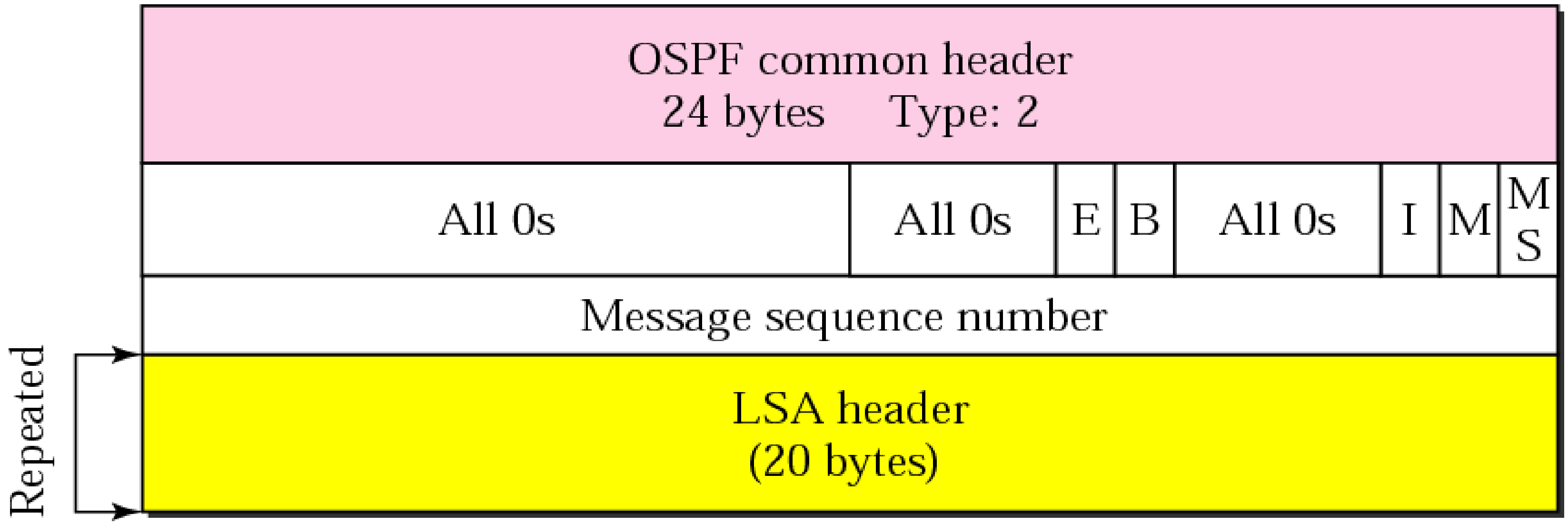


## Figure 14.43 *External link LSA*

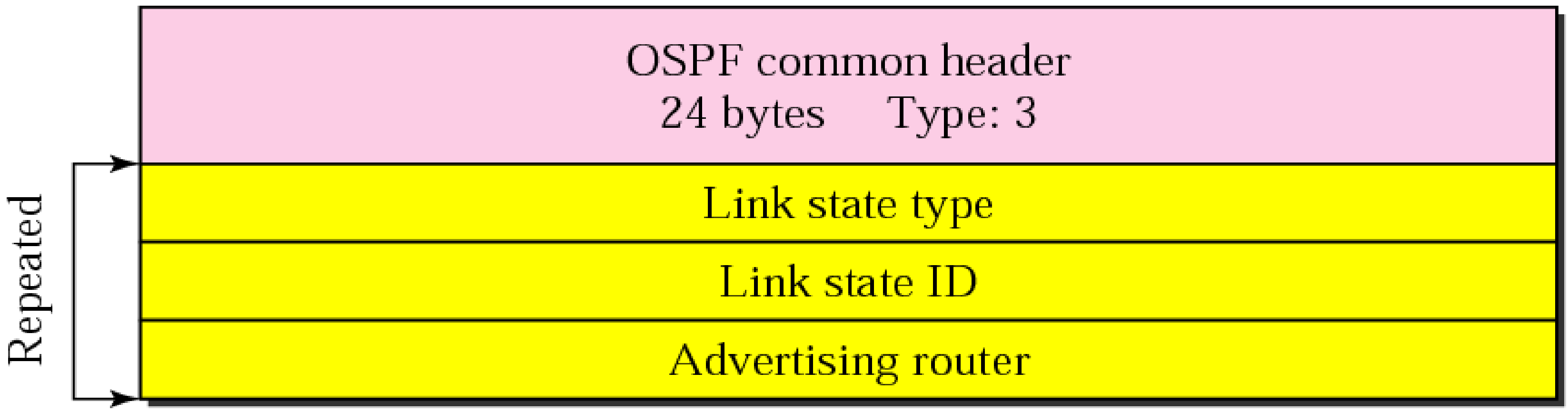



OSPF common header
24 bytes Type: 1
Network mask
Hello interval
All 0s
E
T
Priority
Dead interval
Designated router IP address
Backup designated router IP address
Neighbor IP address
Repeated

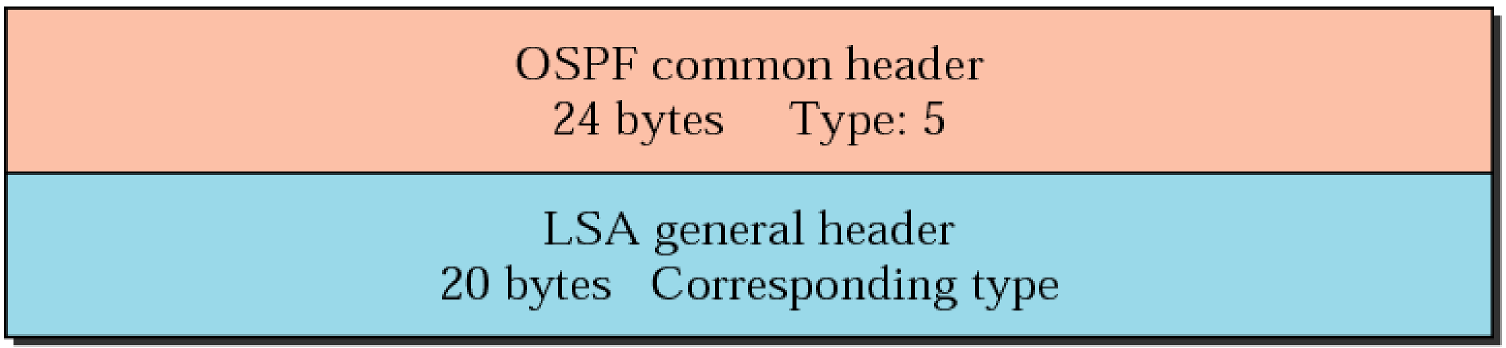
## Figure 14.45 *Database description packet*



# Figure 14.46 *Link state request packet*



## Figure 14.47 *Link state acknowledgment packet*



Note:

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

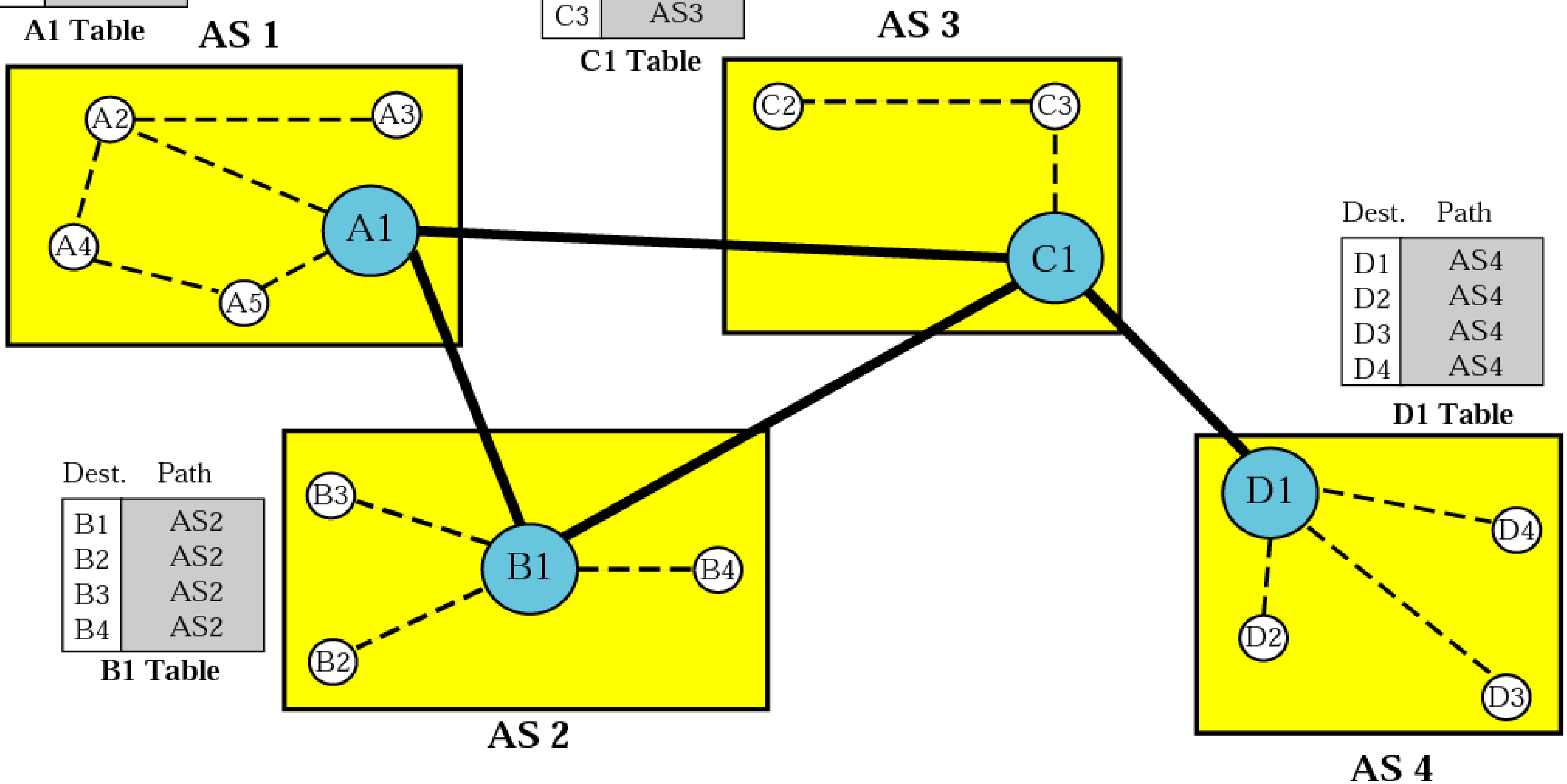
# 14.6 PATH VECTOR ROUTING

***Path vector routing is similar to distance vector routing. There is at least one node, called the speaker node, in each AS that creates a routing table and advertises it to speaker nodes in the neighboring ASs..***

***The topics discussed in this section include:***

***Initialization***
***Sharing***
***Updating***

## **Figure 14.48** *Initial routing tables in path vector routing*




| Dest. | Path |
|---|---|
| A1 | AS1 |
| A2 | AS1 |
| A3 | AS1 |
| A4 | AS1 |
| A5 | AS1 |

**A1 Table**

| Dest. | Path |
|---|---|
| C1 | AS3 |
| C2 | AS3 |
| C3 | AS3 |

**C1 Table**

| Dest. | Path |
|---|---|
| D1 | AS4 |
| D2 | AS4 |
| D3 | AS4 |
| D4 | AS4 |

**D1 Table**

| Dest. | Path |
|---|---|
| B1 | AS2 |
| B2 | AS2 |
| B3 | AS2 |
| B4 | AS2 |

**B1 Table**

## **Figure 14.49** *Stabilized tables for four autonomous systems*

| Dest. | Path |
|---|---|
| A1 | AS1 |
| ... | |
| A5 | AS1 |
| B1 | AS1-AS2 |
| ... | ... |
| B4 | AS1-AS2 |
| C1 | AS1-AS3 |
| ... | ... |
| C3 | AS1-AS3 |
| D1 | AS1-AS2-AS4 |
| ... | ... |
| D4 | AS1-AS2-AS4 |

**A1 Table**

| Dest. | Path |
|---|---|
| A1 | AS2-AS1 |
| ... | |
| A5 | AS2-AS1 |
| B1 | AS2 |
| ... | ... |
| B4 | AS2 |
| C1 | AS2-AS3 |
| ... | ... |
| C3 | AS2-AS3 |
| D1 | AS2-AS3-AS4 |
| ... | ... |
| D4 | AS2-AS3-AS4 |

**B1 Table**

| Dest. | Path |
|---|---|
| A1 | AS3-AS1 |
| ... | |
| A5 | AS3-AS1 |
| B1 | AS3-AS2 |
| ... | ... |
| B4 | AS3-AS2 |
| C1 | AS3 |
| ... | ... |
| C3 | AS3 |
| D1 | AS3-AS4 |
| ... | ... |
| D4 | AS3-AS4 |

**C1 Table**

| Dest. | Path |
|---|---|
| A1 | AS4-AS3-AS1 |
| ... | |
| A5 | AS4-AS3-AS1 |
| B1 | AS4-AS3-AS2 |
| ... | ... |
| B4 | AS4-AS3-AS2 |
| C1 | AS4-AS3 |
| ... | ... |
| C3 | AS4-AS3 |
| D1 | AS4 |
| ... | ... |
| D4 | AS4 |

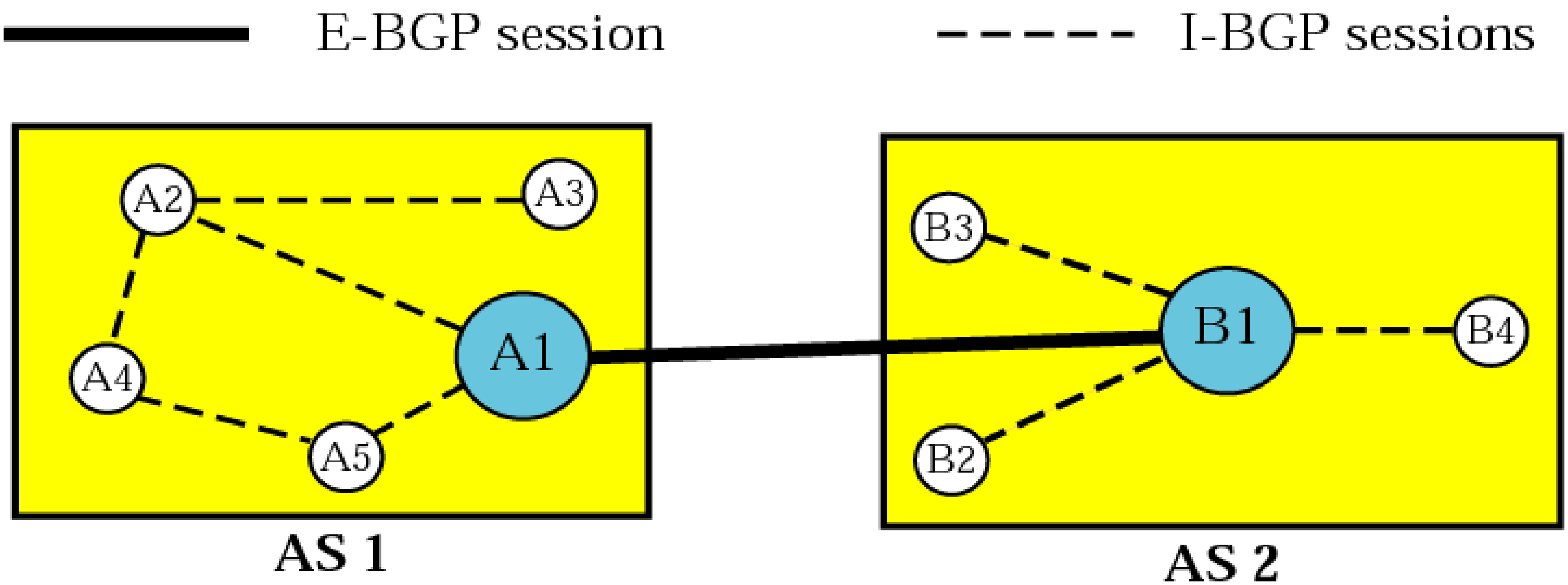
**D1 Table**

# 14.7 BGP

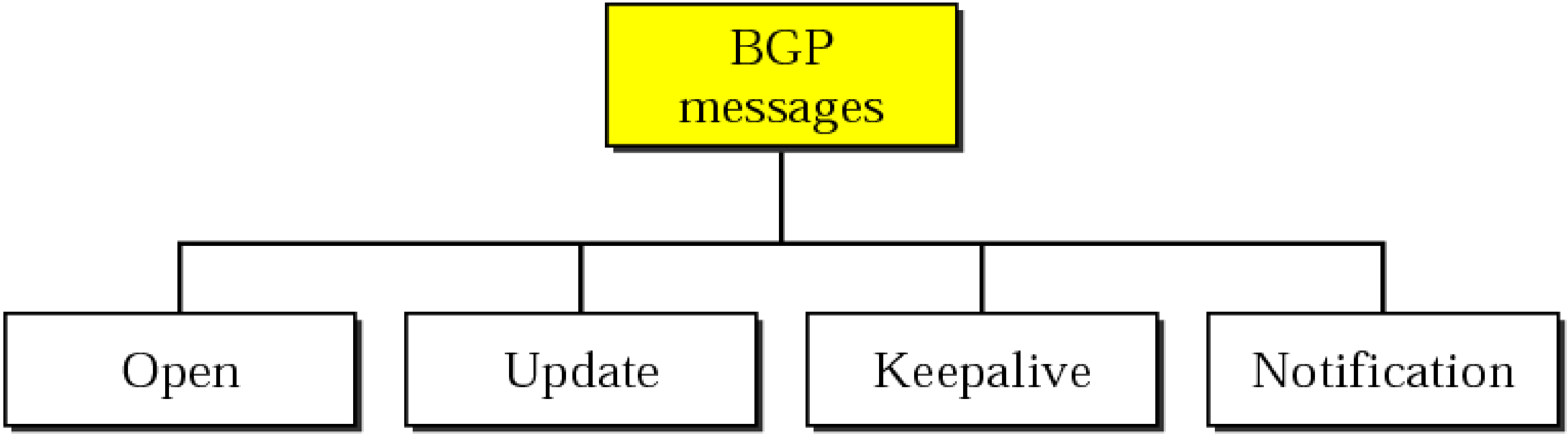
***Border Gateway Protocol (BGP) is an interdomain routing protocol using path vector routing. It first appeared in 1989 and has gone through four versions.***

***The topics discussed in this section include:***

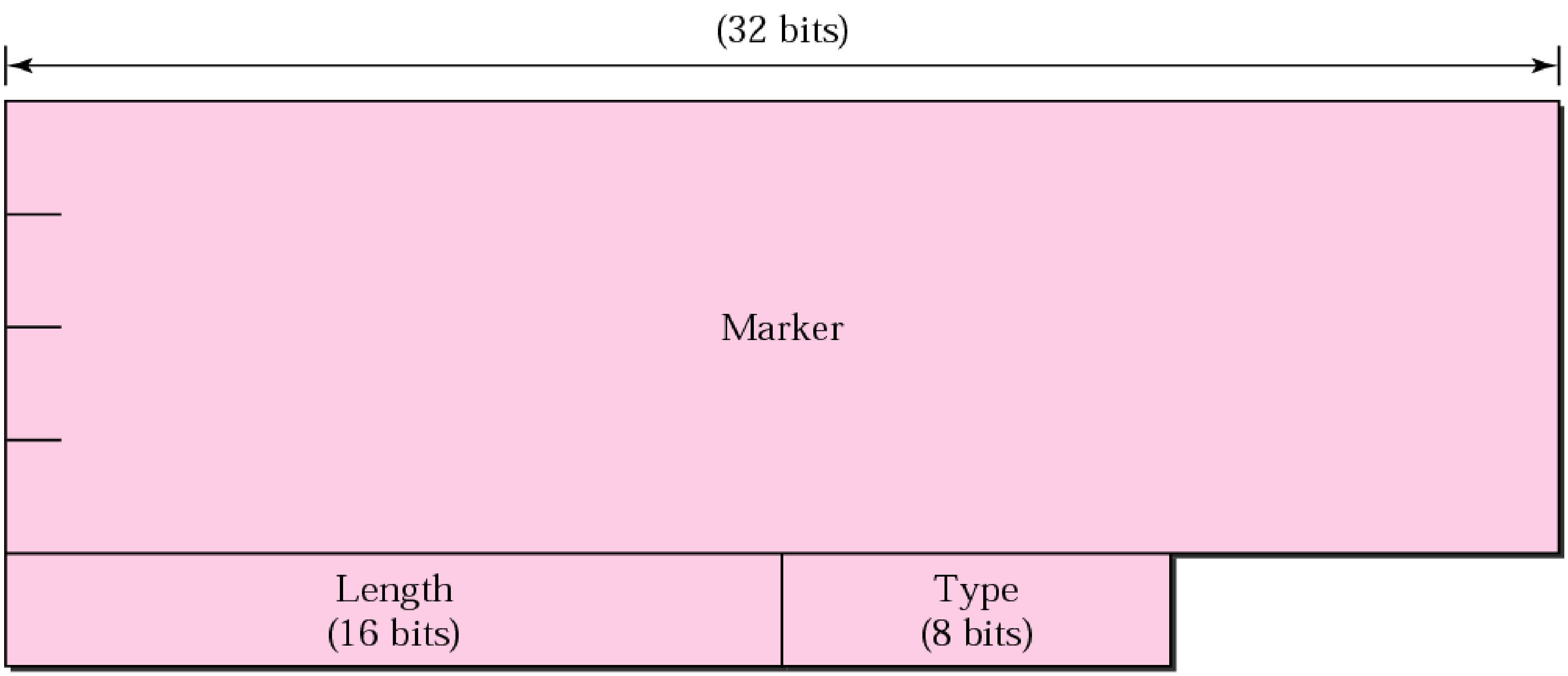
***Types of Autonomous Systems***
***Path Attributes***
***BGP Sessions***
***External and Internal BGP***
***Types of Packets***
***Packet Format***
***Encapsulation***


E-BGP session
I-BGP sessions
A1
A2
A3
A4
A5
B1
B2
B3
B4
AS 1
AS 2

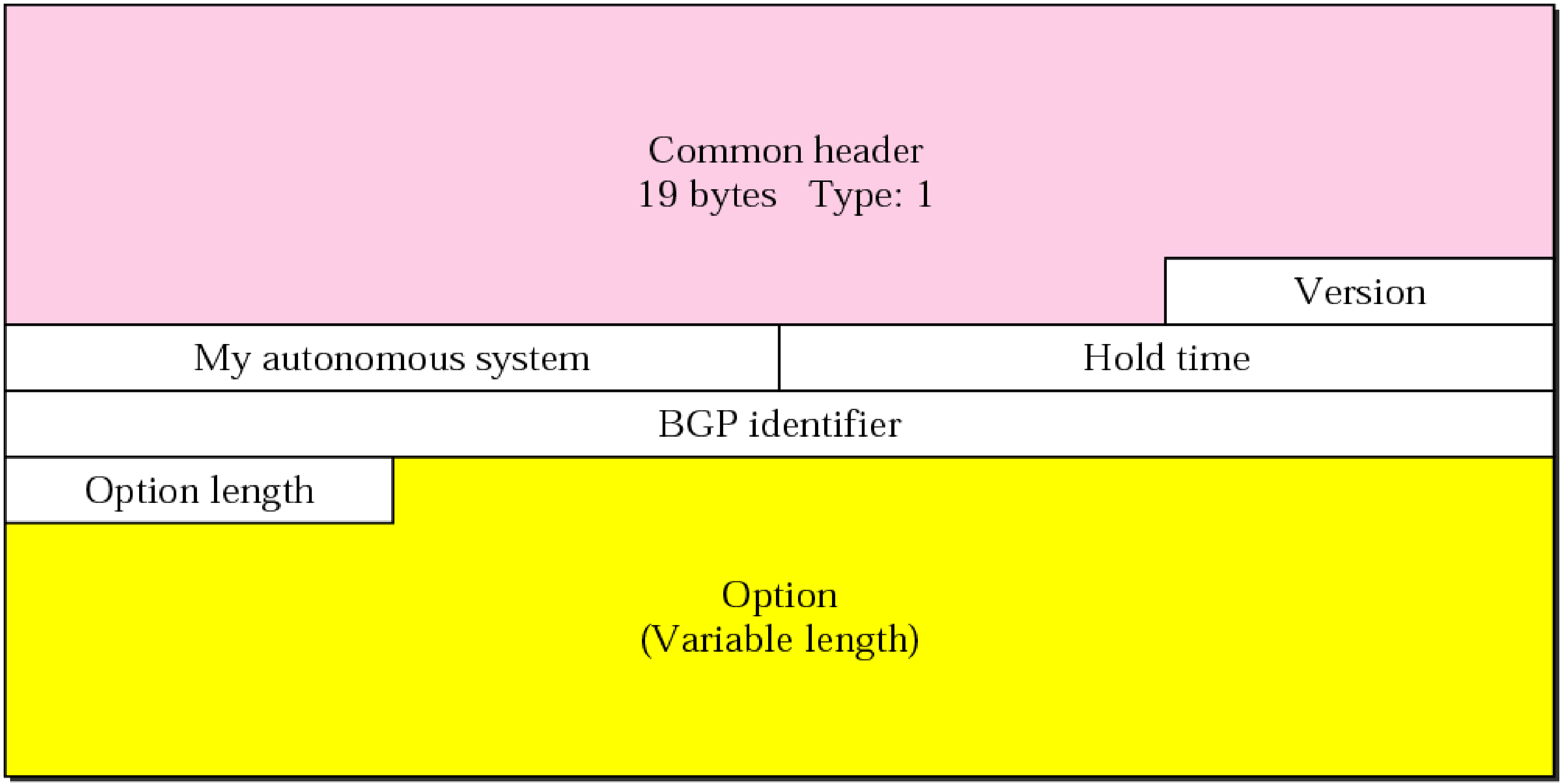
## Figure 14.51 *Types of BGP messages*



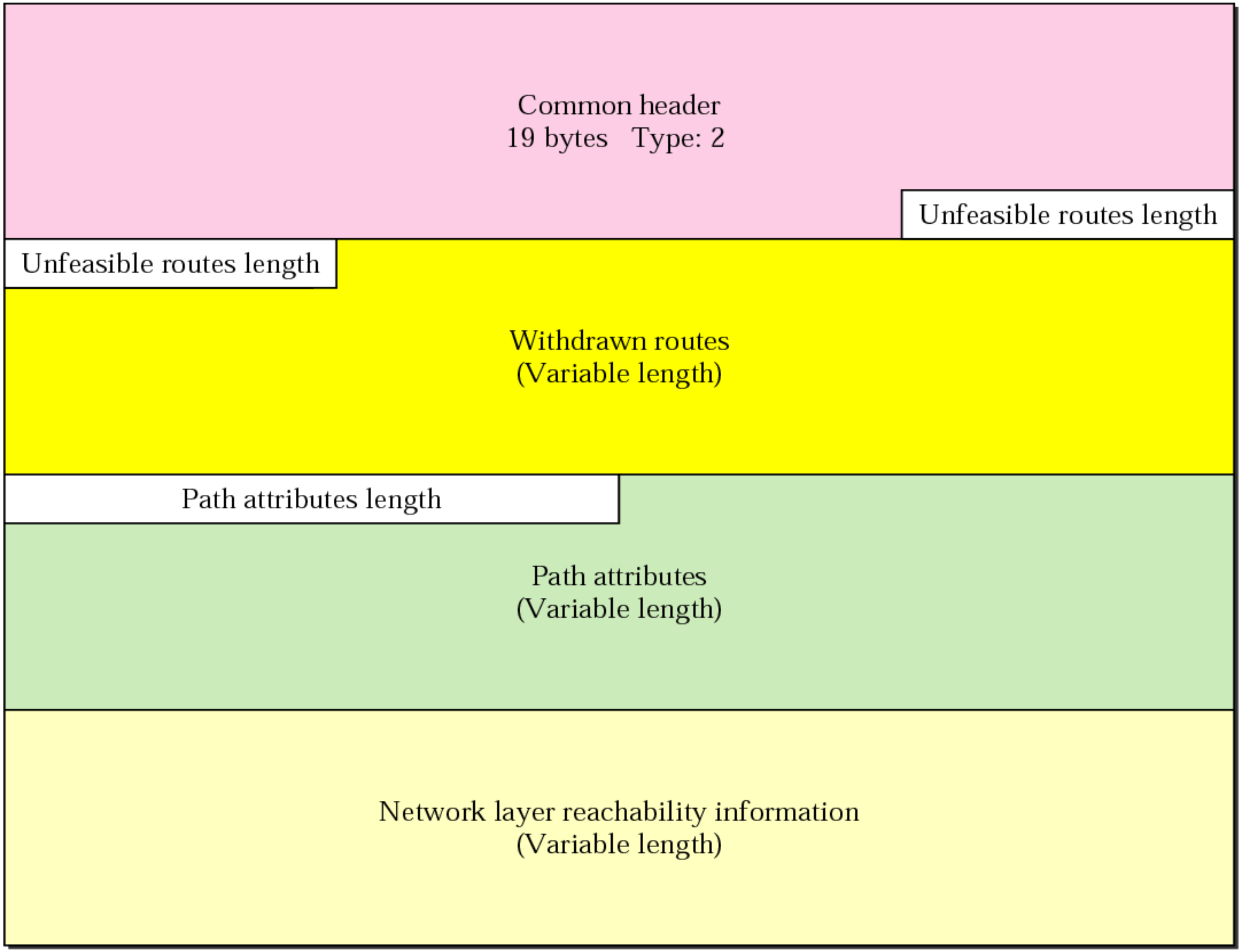
## Figure 14.52 *BGP packet header*



## Figure 14.53 *Open message*



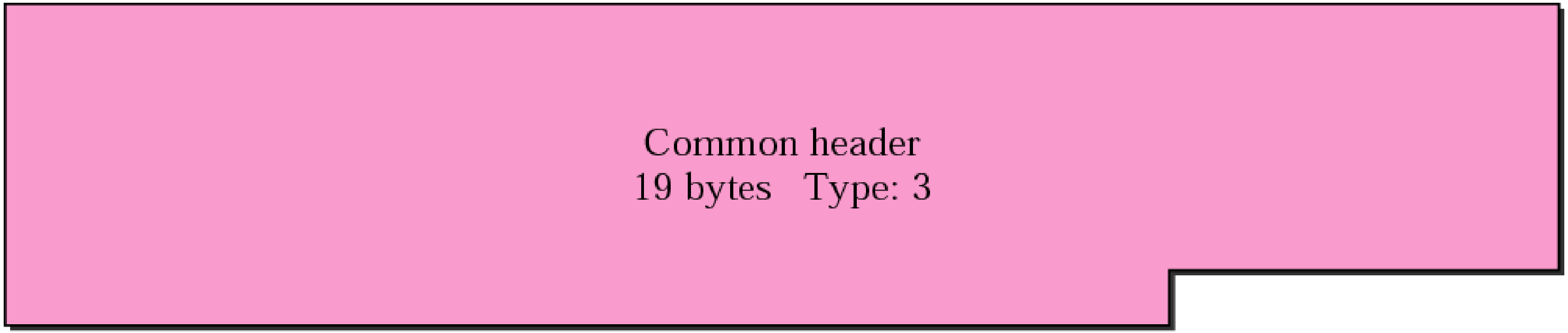
## Figure 14.54 *Update message*



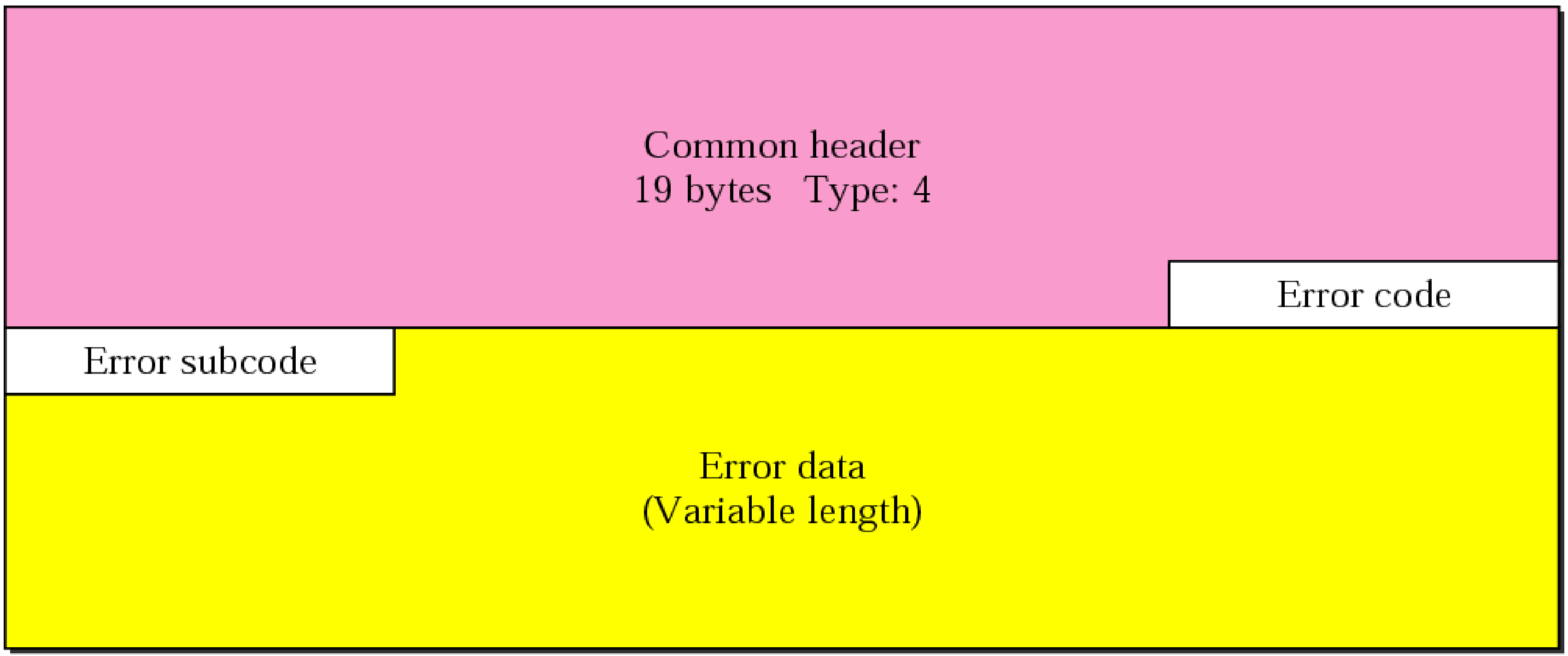
Note:

***BGP supports classless addressing and CIDR.***

## Figure 14.55 *Keepalive message*



## Figure 14.56 *Notification message*



# Table 14.3 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.***