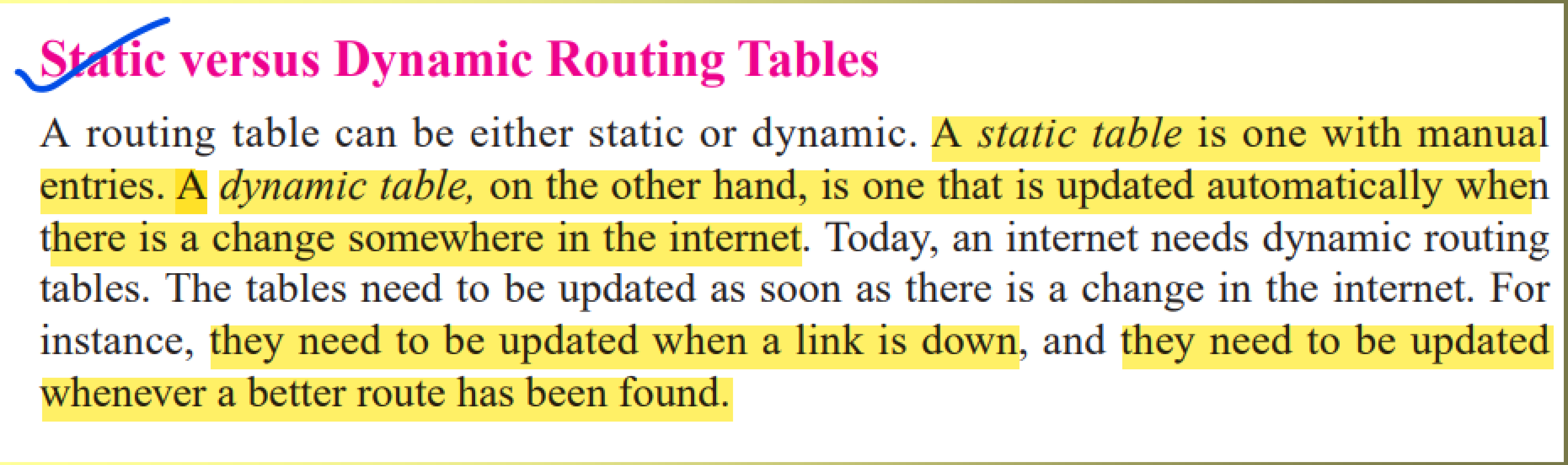
Topics Discussed in the Section

- ✓ **Cost or Metric**
- ✓ **Static versus Dynamic Routing Table**
- ✓ **Routing Protocol**



Static versus Dynamic Routing Tables

A routing table can be either static or dynamic. A *static table* is one with manual entries. A *dynamic table,* on the other hand, is one that is updated automatically when there is a change somewhere in the internet. Today, an internet needs dynamic routing tables. The tables need to be updated as soon as there is a change in the internet. For instance, they need to be updated when a link is down, and they need to be updated whenever a better route has been found.

11-2 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

Figure 11.1 *Autonomous systems*

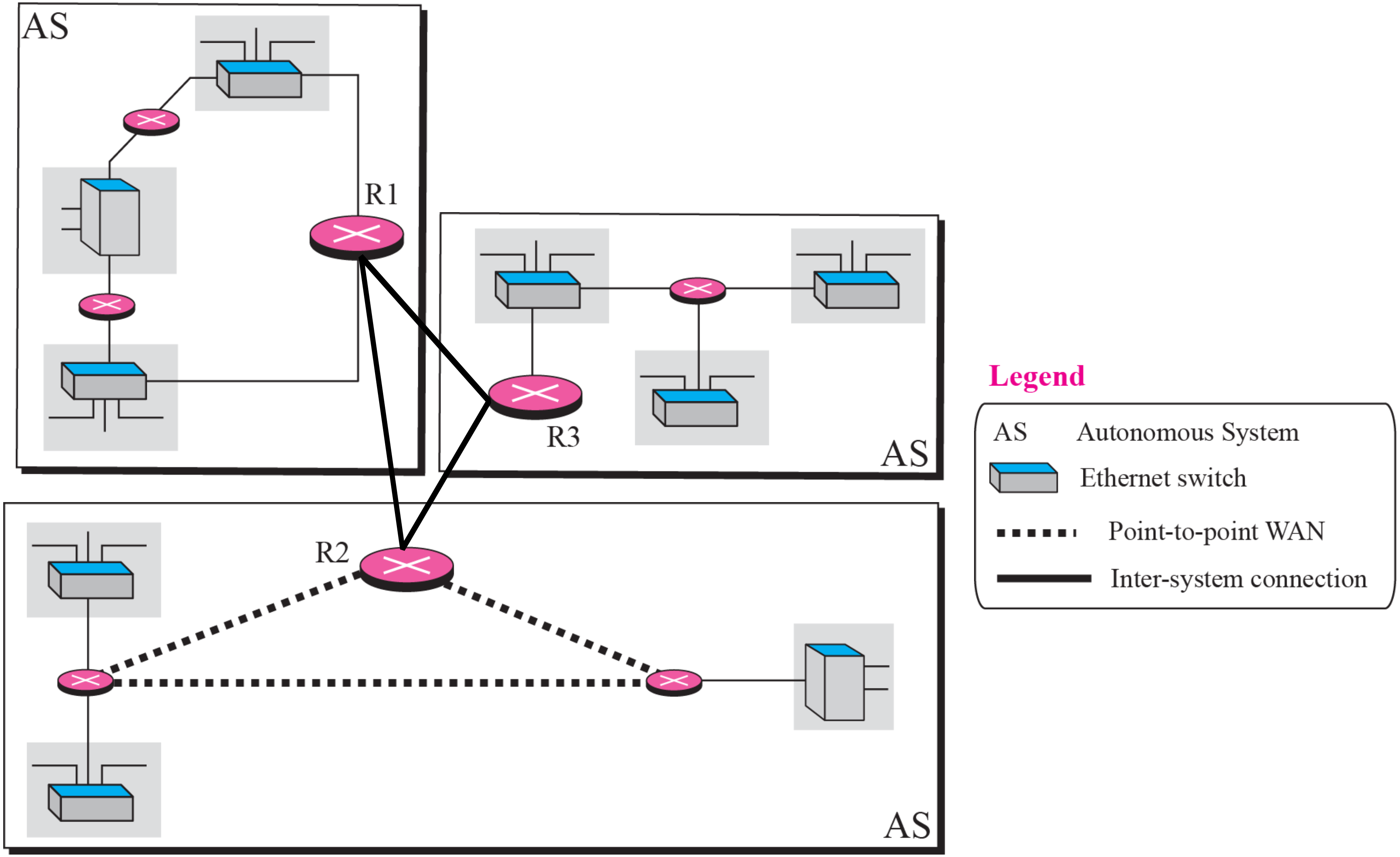
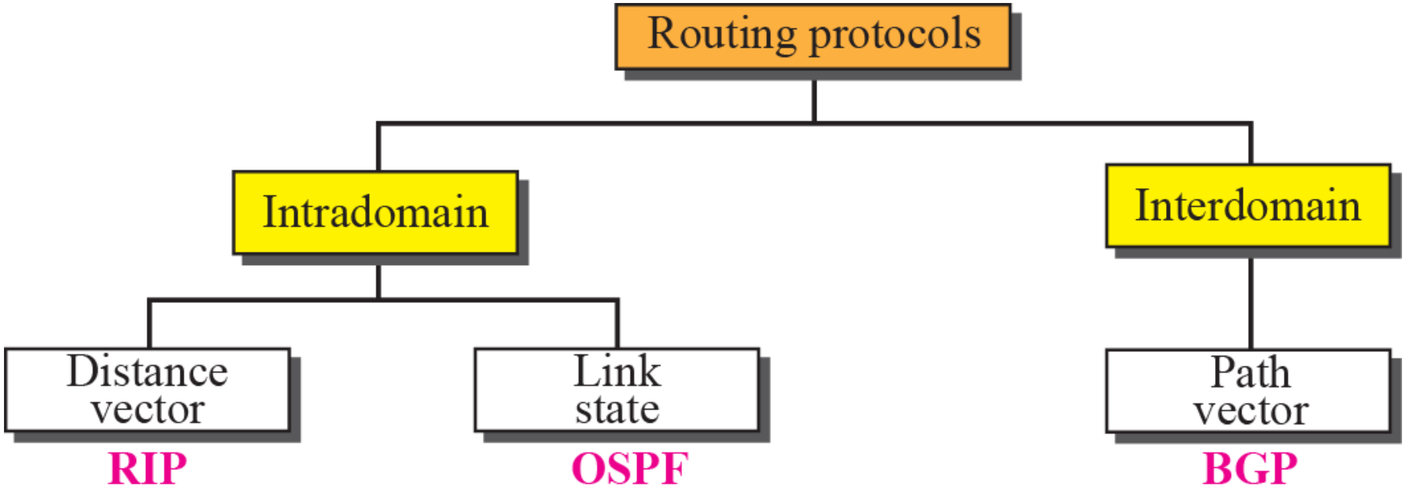


Figure 11.2 *Popular routing protocols*



11-3 DISTANCE VECTOR 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

此段說明重複**11-2**的標題投影片內容

Topics Discussed in the Section

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

Figure 11.3 *A graph for Bellman-Ford algorithm*

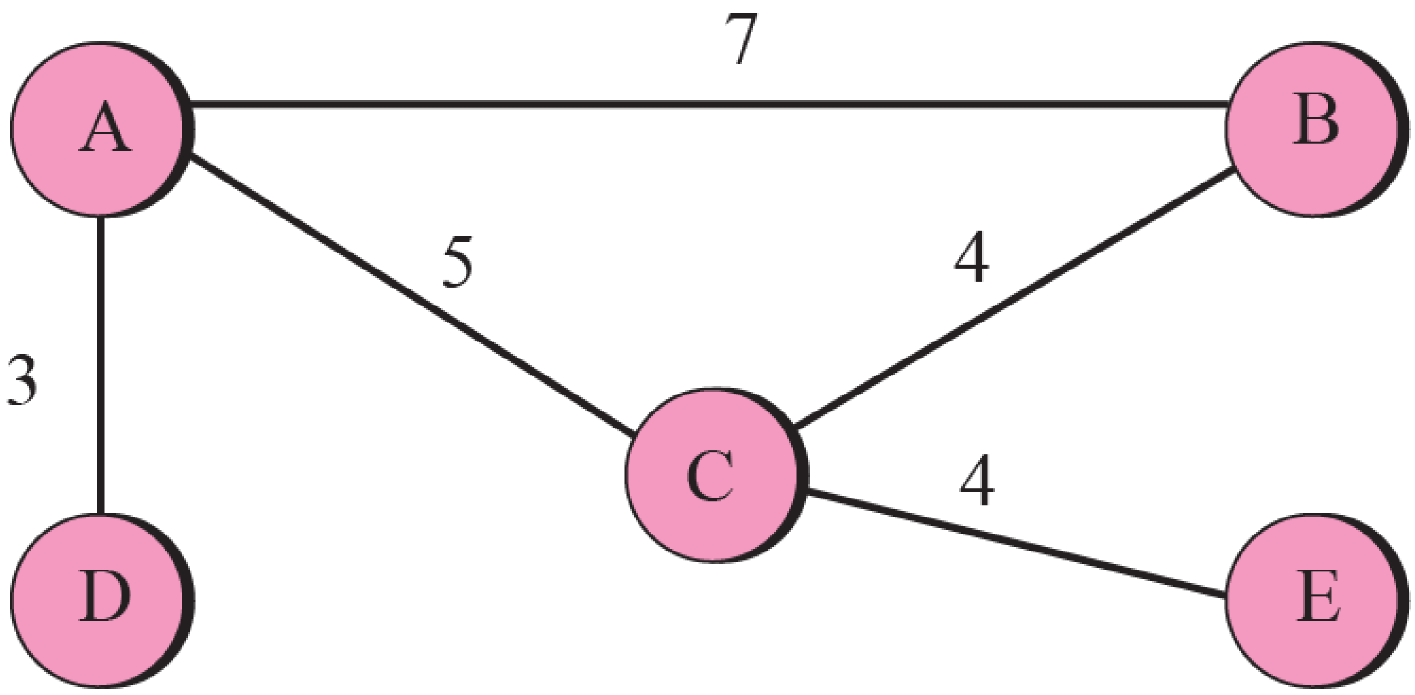
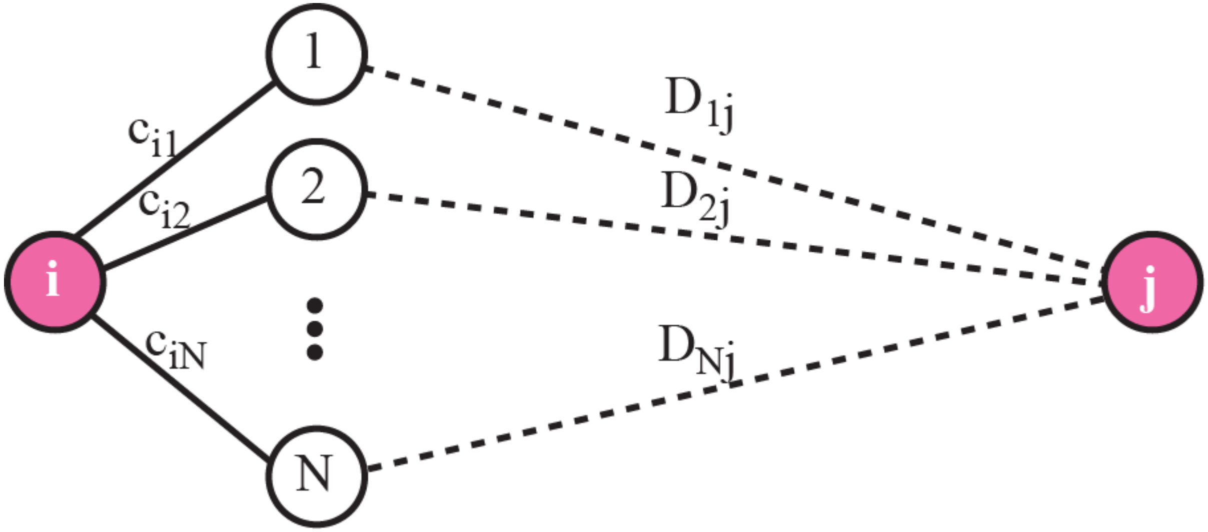


Figure 11.4 *The fact behind Bellman-Ford algorithm*

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



Legend

D_{ij} Shortest distance between i and j

c_{ij} Cost between i and j

N Number of nodes

Table 11.1 *Bellman-Ford Algorithm*

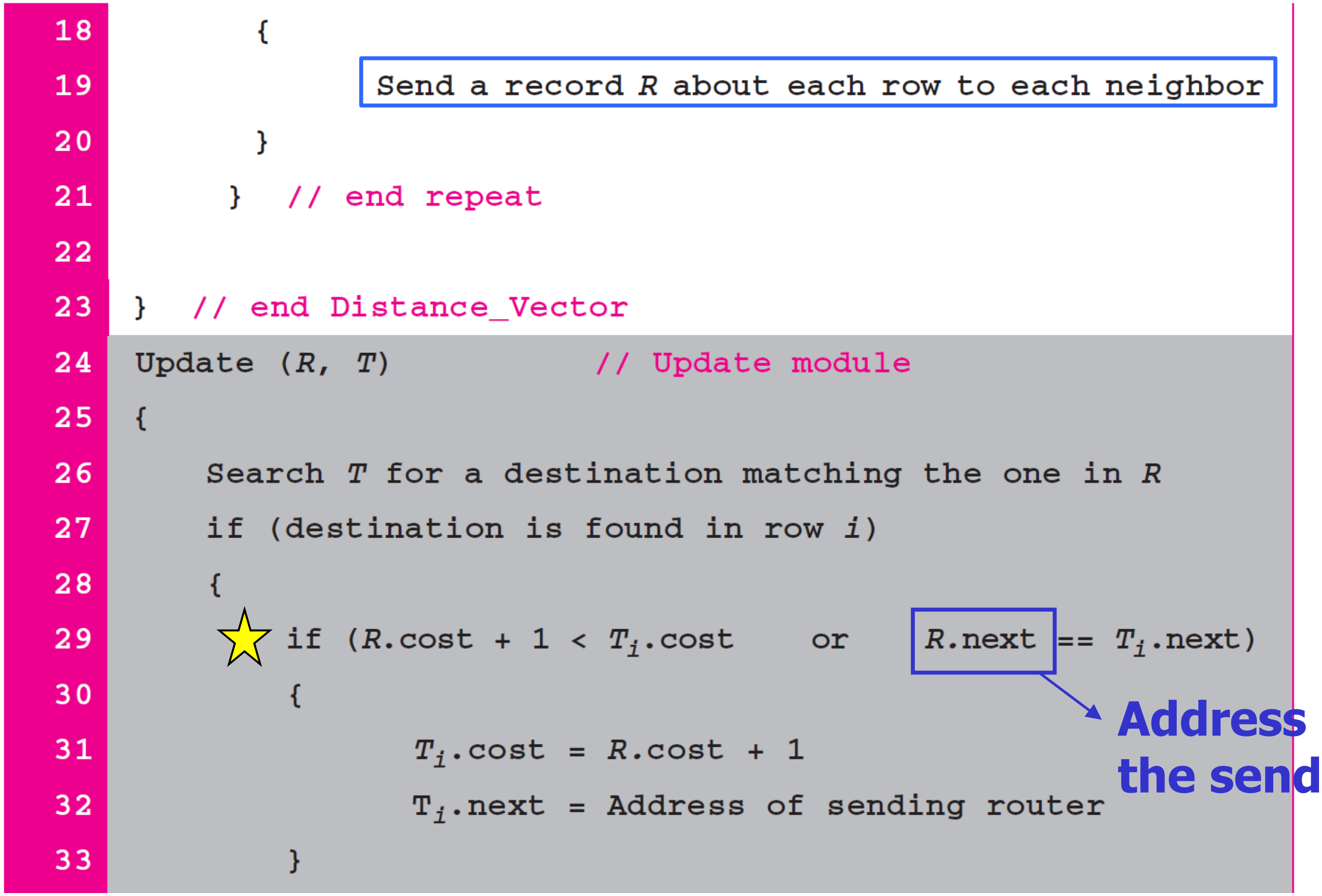
```
Bellman_Ford ( )
{
   // Initialization
   for ($i$ = 1 to $N$; for $j$ = 1 to $N$)
   {
       if($i$ == $j$)   $D_{ij}$ = 0    $c_{ij}$ = 0
       else           $D_{ij}$ = $\infty$    $c_{ij}$ = cost between $i$ and $j$
   }
   // Updating
   repeat
   {
      for ($i$ = 1 to $N$; for $j$ = 1 to $N$)
      {
          $D_{ij}$ $\leftarrow$ minimum [($c_{i1}$ + $D_{1j}$)  ...  ($c_{iN}$ + $D_{Nj}$)]
      } // end for
   } until (there was no change in previous iteration)
} // end Bellman-Ford
```

Table 11.2 *Distance Vector Algorithm Used by Each Router*

```
Distance_Vector_Algorithm ( )
{
    // At startup
    for ($i$ = 1 to $N$)                  // $N$ is number of ports
    {
        Table$_i$.dest = address of the attached network
        Table$_i$.cost = 1
        Table$_i$.next = —                // Means at home
        Send a record $R$ about each row to each neighbor
    } // end for loop

    // Updating
    repeat (forever)
    {
        Wait for arrival of a record $R$ from a neighbor
        Update ($R$, $T$)                 // Call update module
        for ($i$ = 1 to $N$)              // $N$ is the current table size
```

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

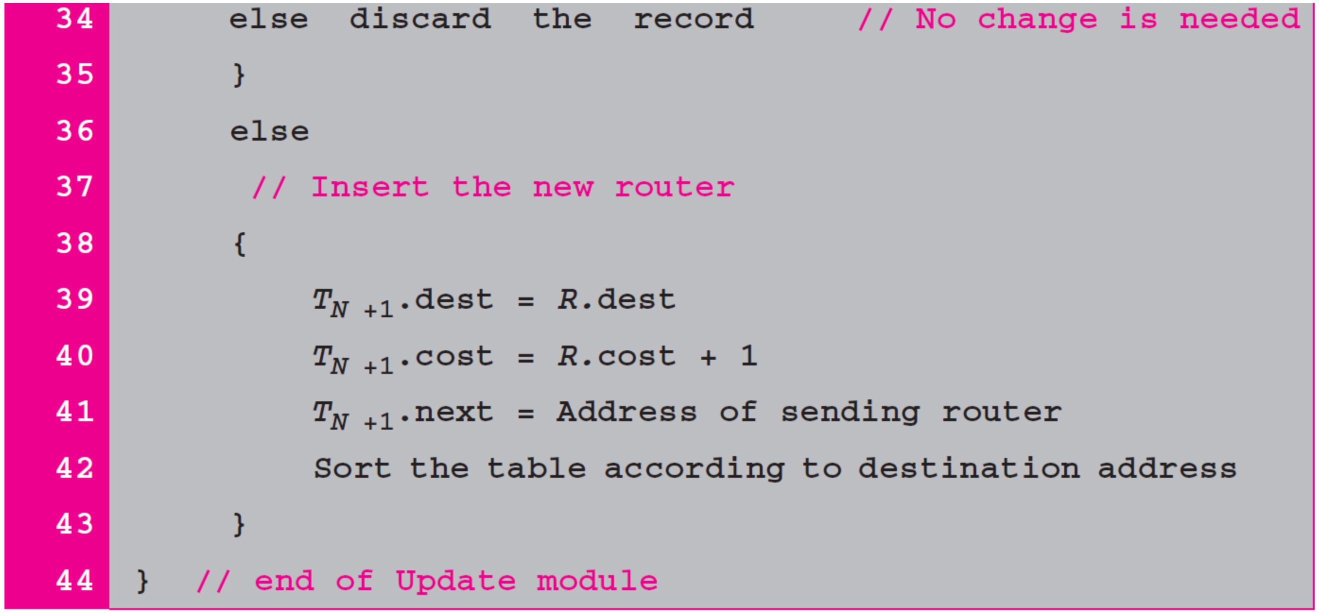


```
        {
            Send a record R about each row to each neighbor
        }
      }  // end repeat

}  // end Distance_Vector
Update (R, T)               // Update module
{
    Search T for a destination matching the one in R
    if (destination is found in row i)
    {
        if (R.cost + 1 < T_i.cost    or    R.next == T_i.next)
        {
              T_i.cost = R.cost + 1
              T_i.next = Address of sending router
        }
```

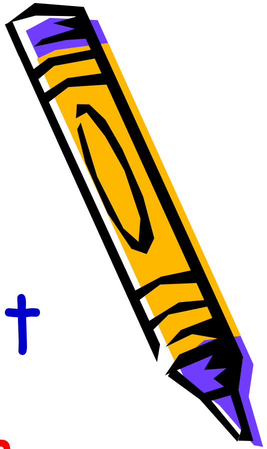
Address of the sender of R

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

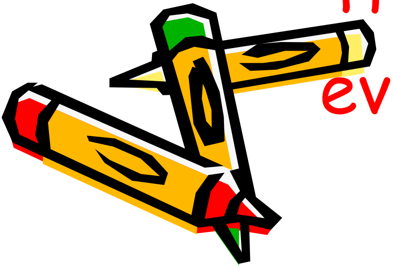


```
            else  discard  the  record     // No change is needed
        }
        else
         // Insert the new router
        {
            $T_{N+1}$.dest = $R$.dest
            $T_{N+1}$.cost = $R$.cost + 1
            $T_{N+1}$.next = Address of sending router
            Sort the table according to destination address
        }
    }  // end of Update module
```

Updating Routing Table

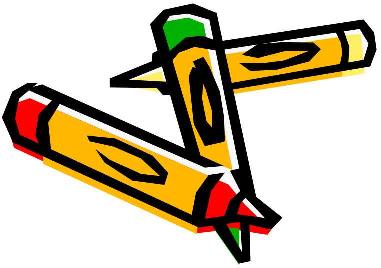


- If the next-node entry is different
  - The receiving node chooses the row with the smaller cost
  - If there is a tie, the old one is kept
- If the next-node entry is the same
  - i.e. the sender of the new row is the provider of the old entry
  - The receiving node chooses the new row, even though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



When to Share

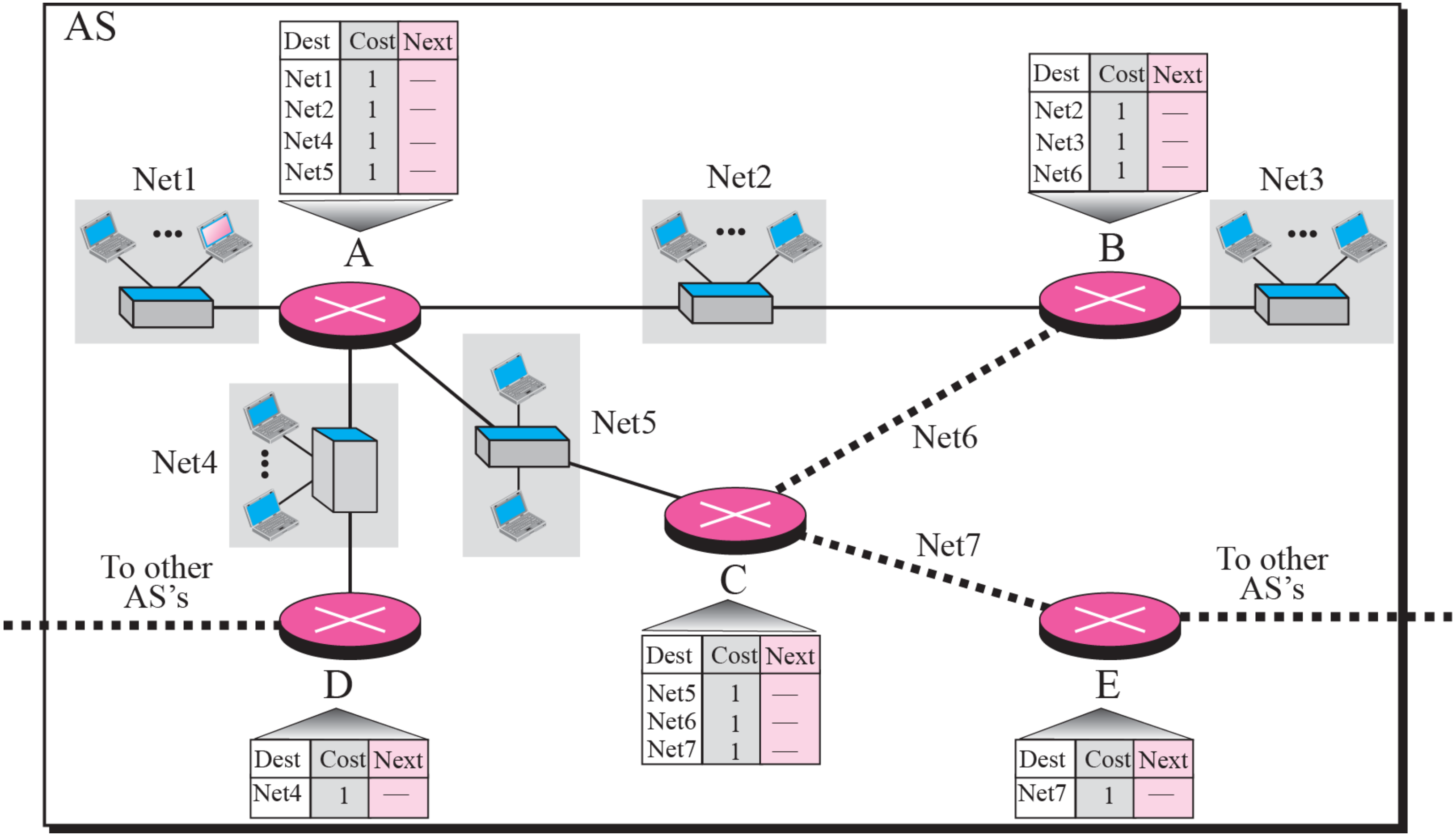
- Periodic Update
  - A node sends its routing table, normally 30 seconds, in a periodic update
- Triggered Update
  - A node sends its routing table to its neighbors any time when there is a change in its routing table
    - 1. After updating its routing table, or
    - 2. Detects some failure in the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



Dest	Cost	Next
Net1	1	—
Net2	1	—
Net4	1	—
Net5	1	—

Dest	Cost	Next
Net2	1	—
Net3	1	—
Net6	1	—

Dest	Cost	Next
Net5	1	—
Net6	1	—
Net7	1	—

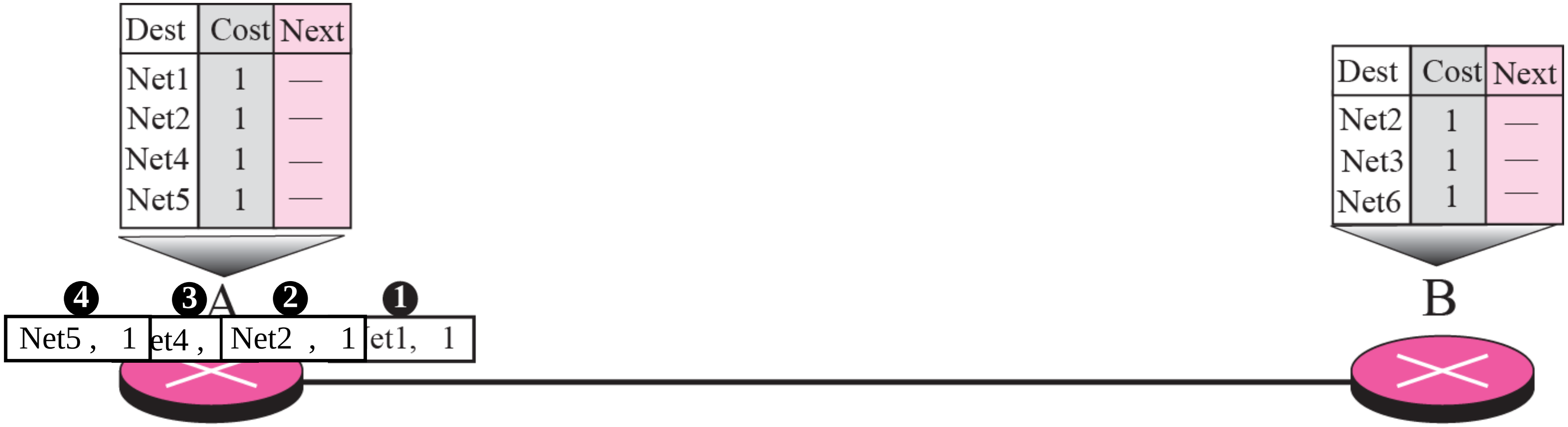
Dest	Cost	Next
Net4	1	—

Dest	Cost	Next
Net7	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*



Dest	Cost	Next
Net1	1	—
Net2	1	—
Net4	1	—
Net5	1	—

Dest	Cost	Next
Net2	1	—
Net3	1	—
Net6	1	—

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

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	—

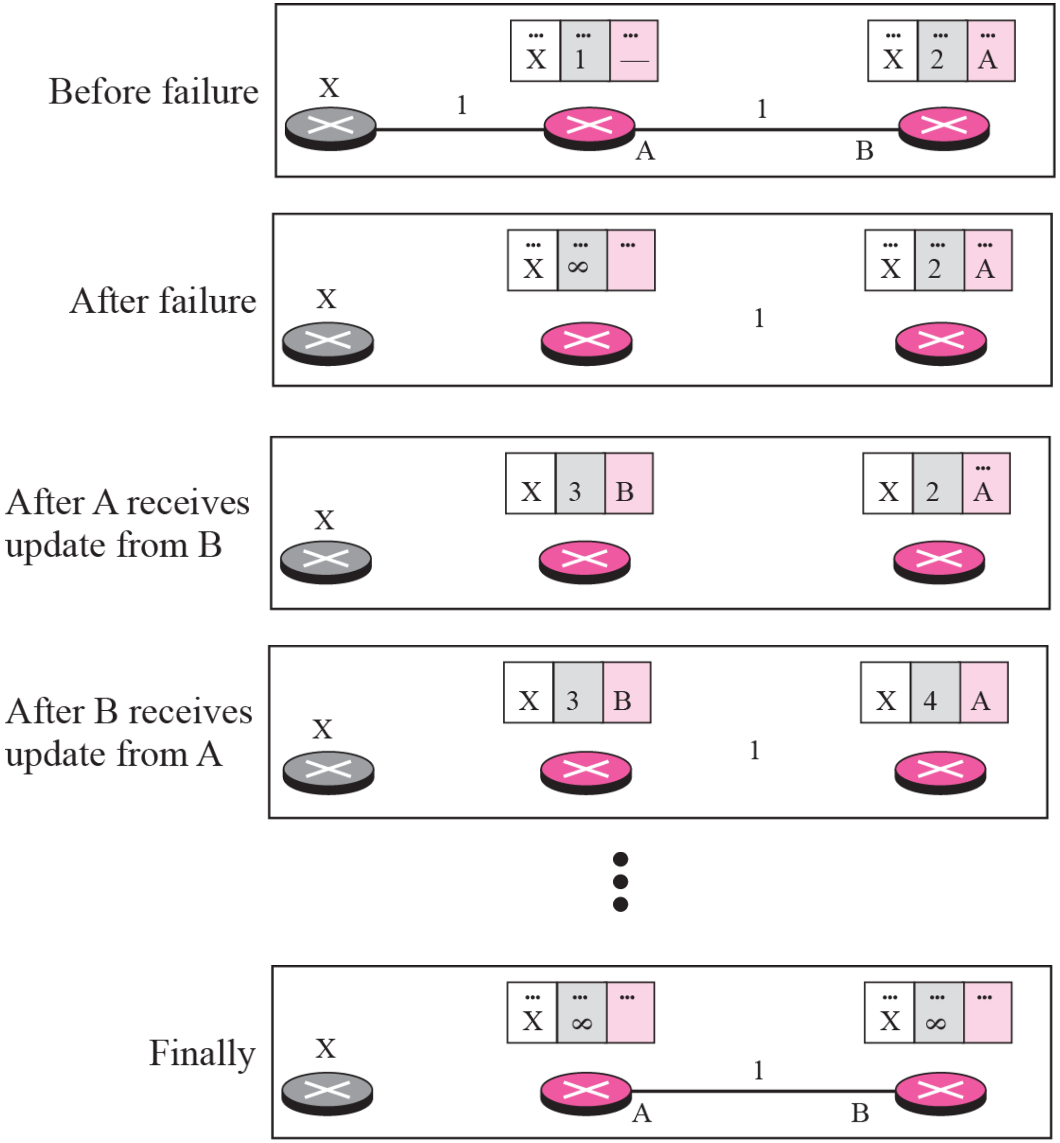
D

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

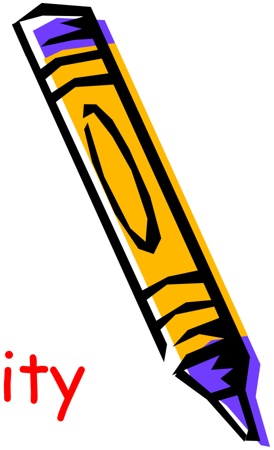
E

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

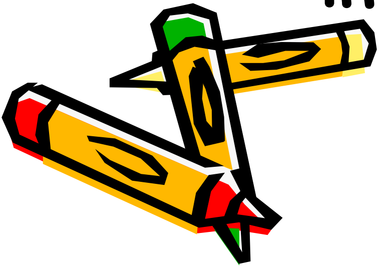
Figure 11.8 *Two-node instability*



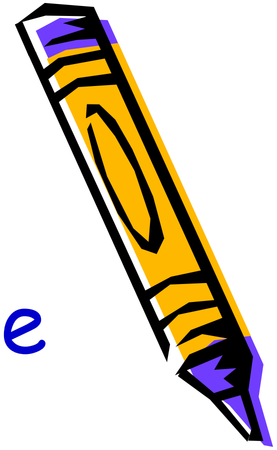
Two-Node Instability (1)



- 1) Defining Infinity, 2) Hold Down
  - Most implementations define 16 as infinity
- 3) 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)



- 4) 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

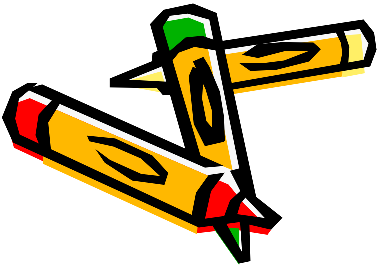
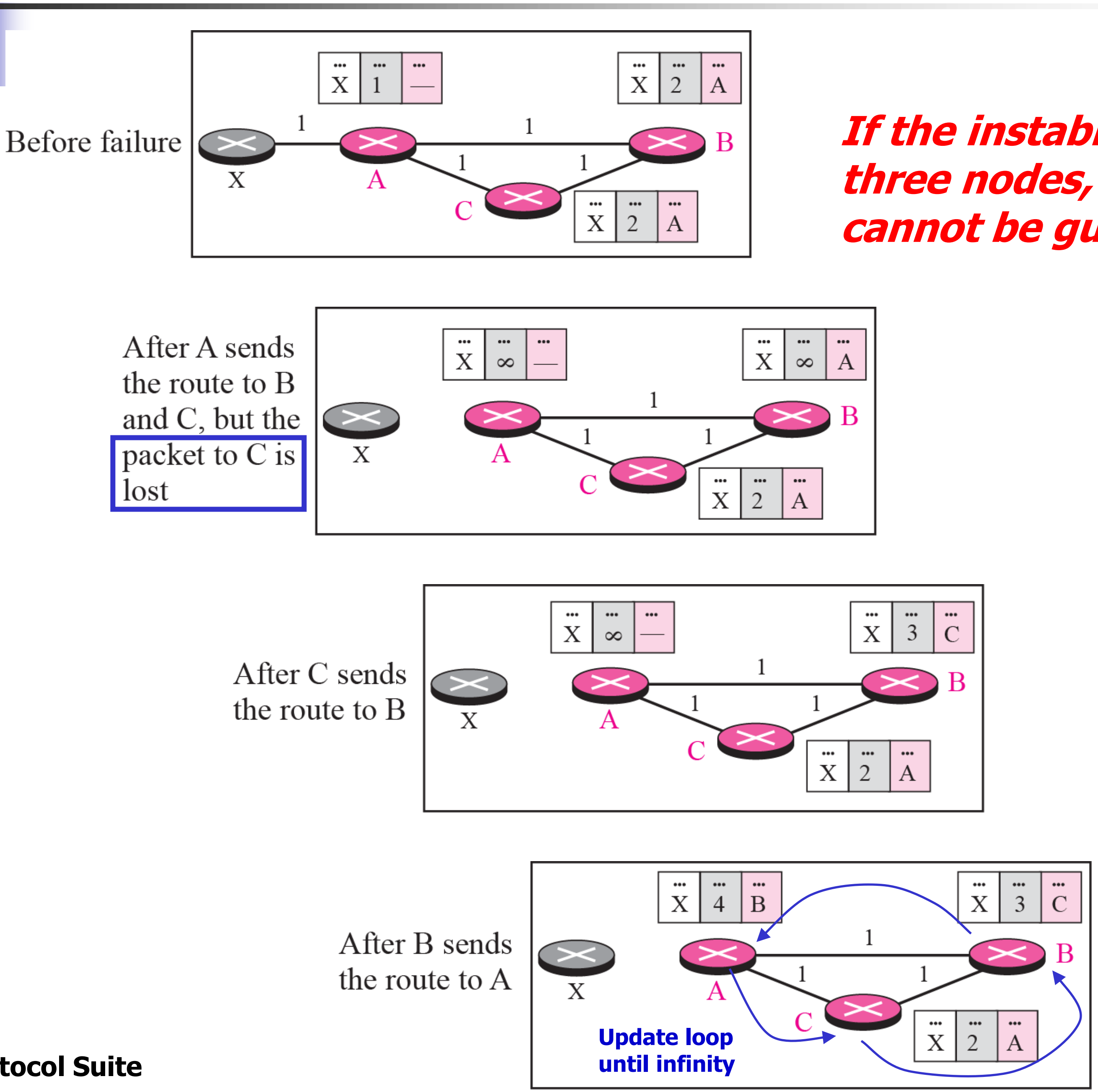


Figure 11.9 *Three-node instability*

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



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

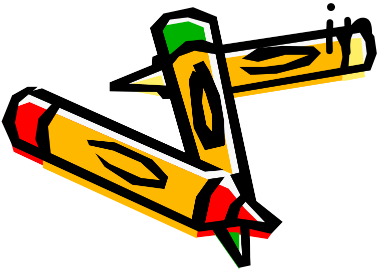
Update loop until infinity

11-4 RIP

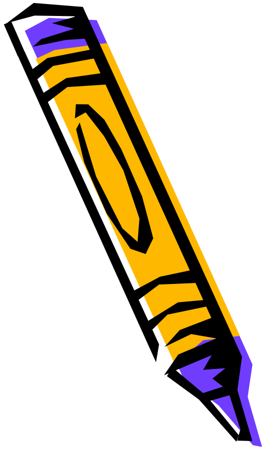
The Routing Information Protocol (RIP) is an intra-domain (interior) routing protocol used inside an autonomous system. It is a very simple protocol based on distance vector routing. RIP implements distance vector routing directly with some considerations.

RIP messages

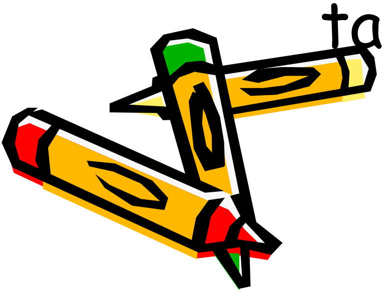
- Request
  - A request message is sent by a router that has just come up or by a router that has some time-out entries
  - A request can ask about specific entries or all entries
- Response
  - A response can be either solicited or unsolicited (30s or when there is a change in the routing table)



RIP Timers



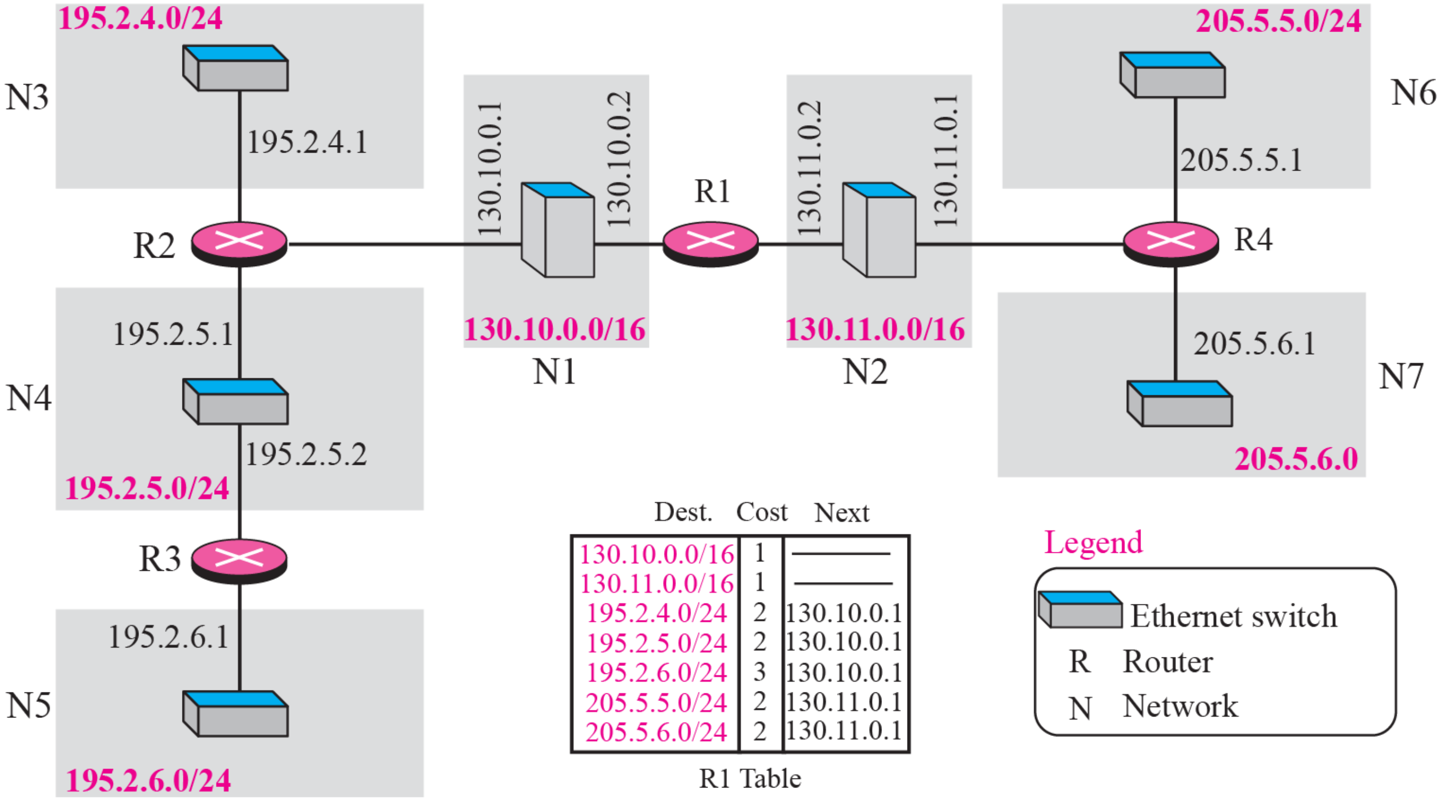
- Periodic timer
  - It controls the advertising of regular update message (25 ~ 30 sec)
- Expiration timer
  - It governs the validity of a route (180 sec)
  - The route is considered expired and the hop count of the route is set to 16
- Garbage collection timer
  - A invalid route is not purged from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



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

R1 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

R2 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

R3 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	—

R4 Table

Figure 11.11 *RIP message format*

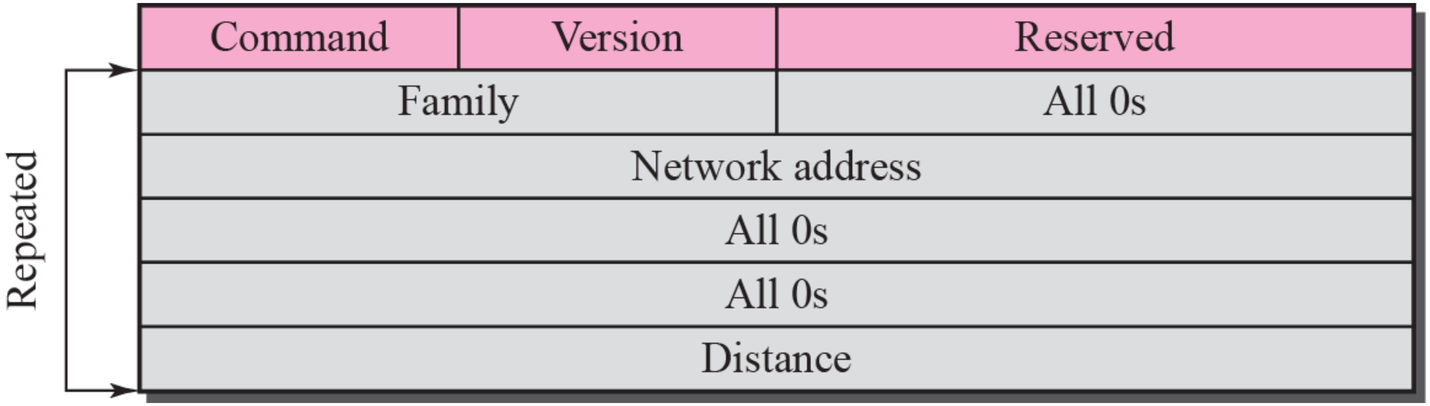
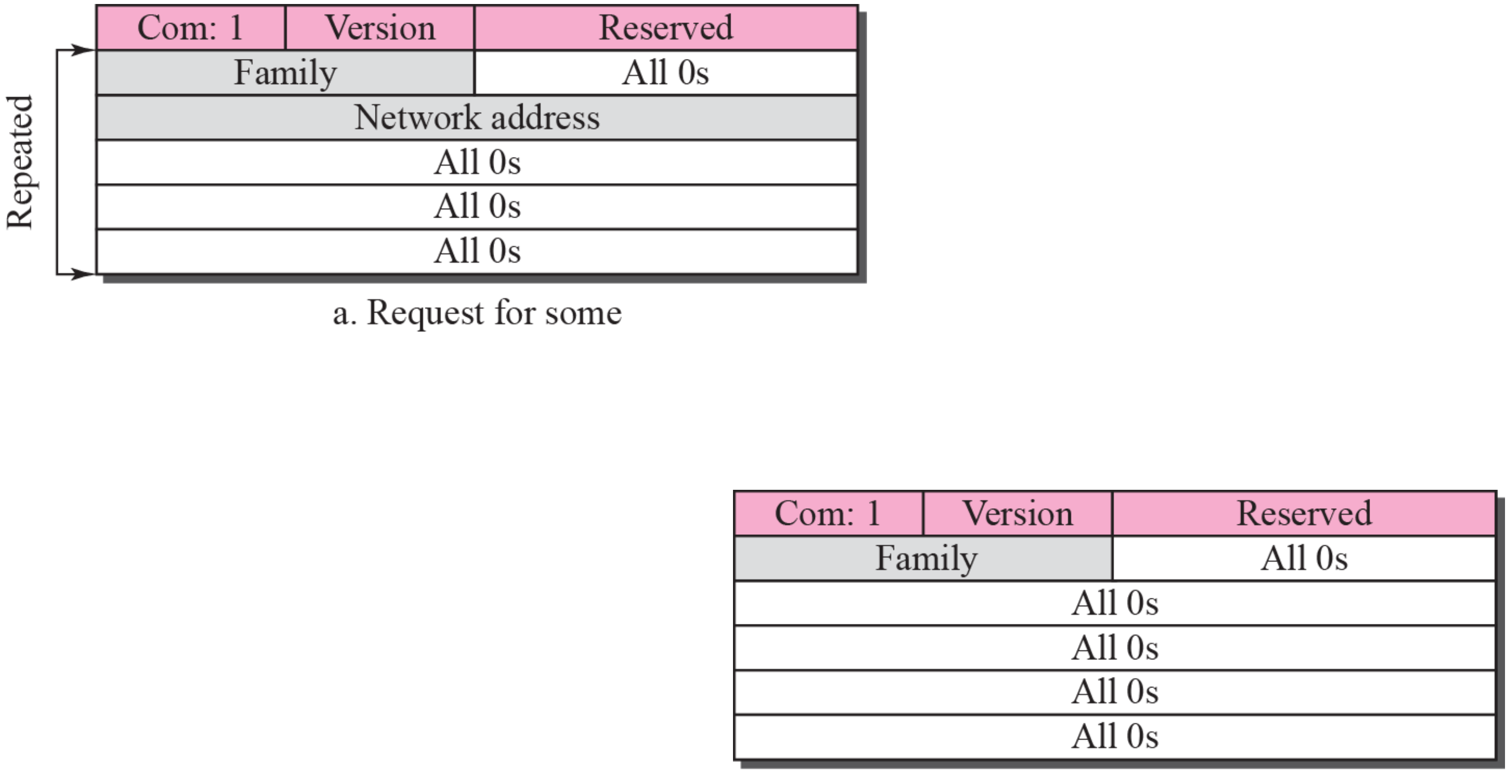


Figure 11.12 *Request messages*



Com: 1	Version	Reserved
Family		All 0s
Network address		
All 0s		
All 0s		
All 0s		

a. Request for some

Com: 1	Version	Reserved
Family		All 0s
All 0s		
All 0s		
All 0s		
All 0s		

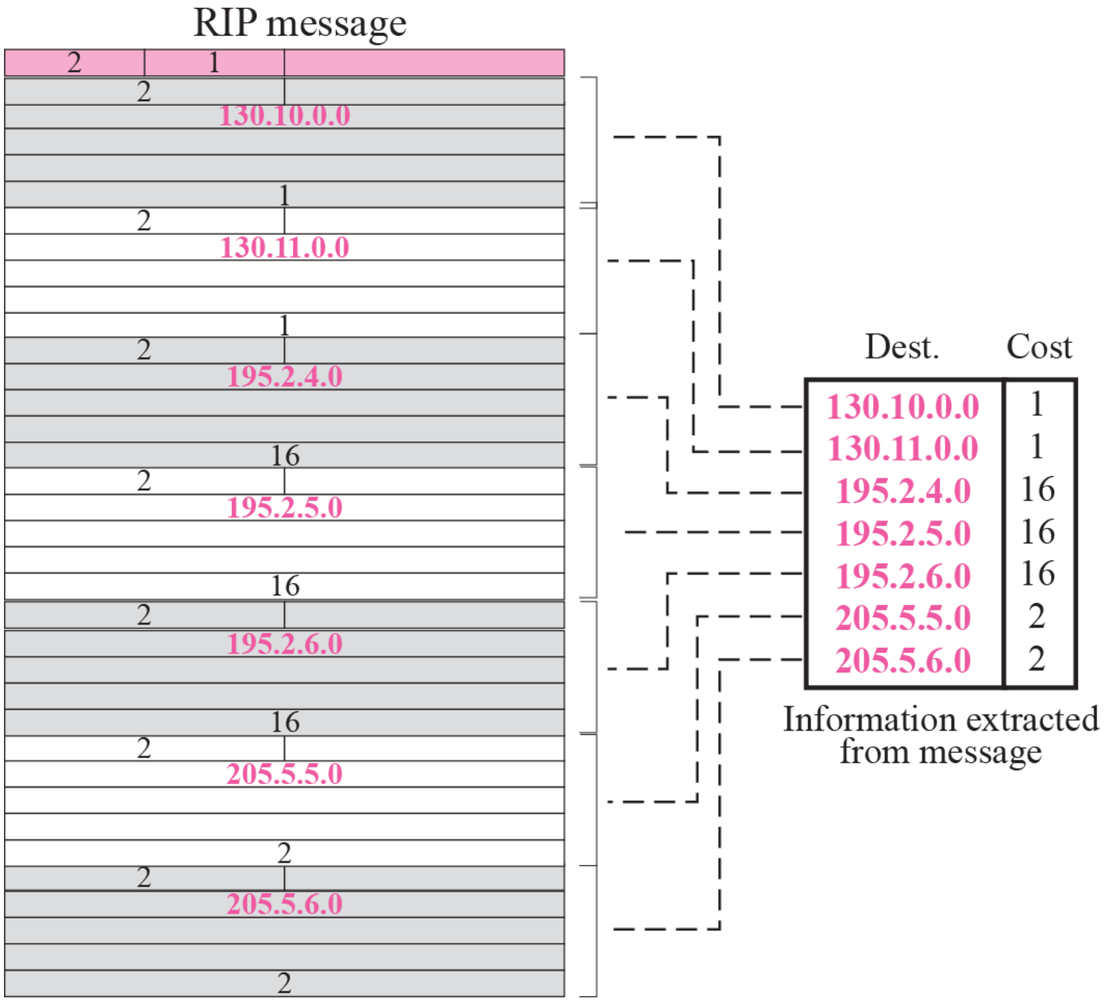
b. Request for all

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.

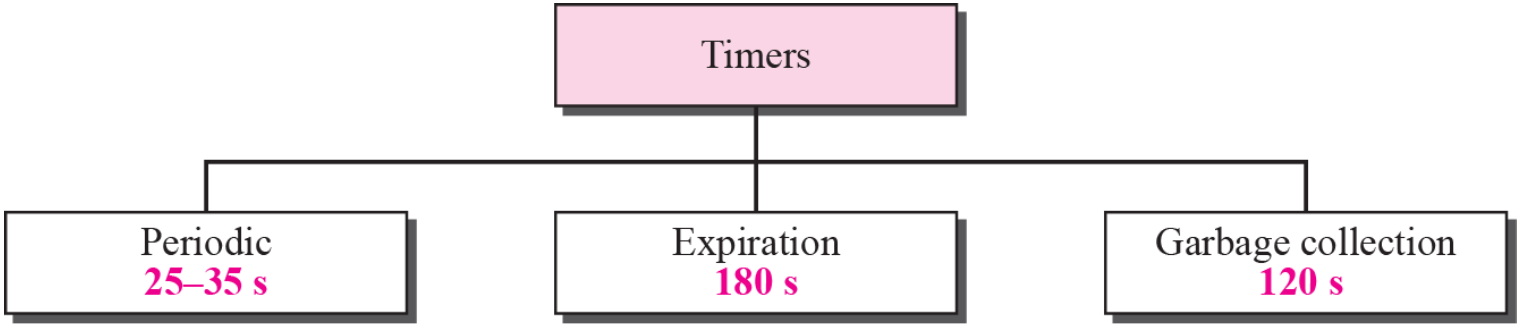
Figure 11.13 *Solution to Example 11.4*



Dest.	Cost
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

Information extracted from message

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

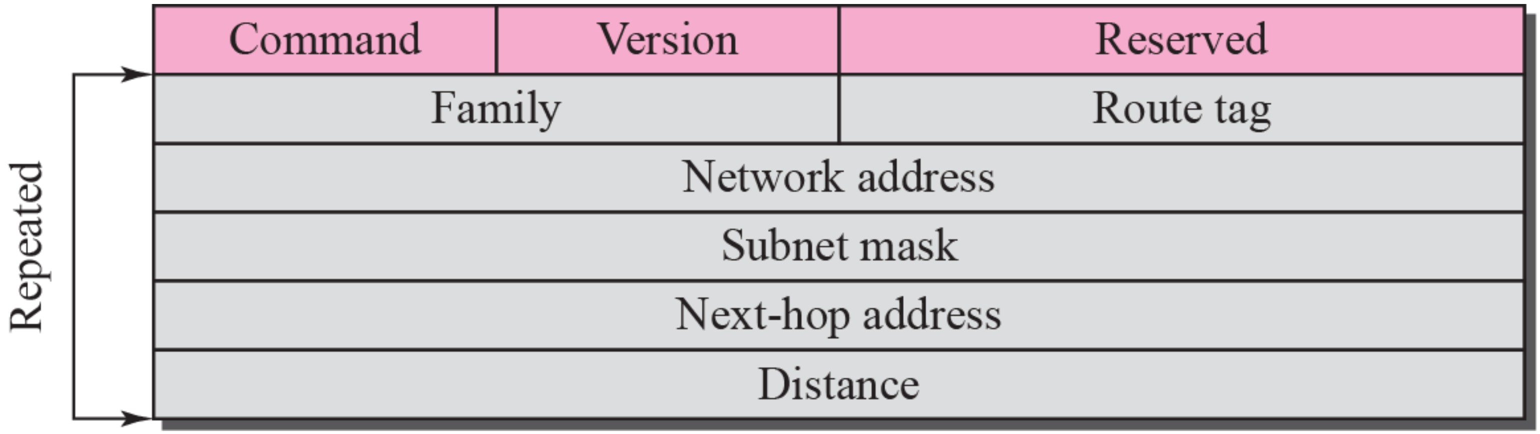
The 21 timers are listed below:

Periodic timer: 1

Expiration timer: 20 − 5 = 15

Garbage collection timer: 5

Figure 11.15 *RIP version 2 format*



Command | Version | Reserved

Family | Route tag

Network address

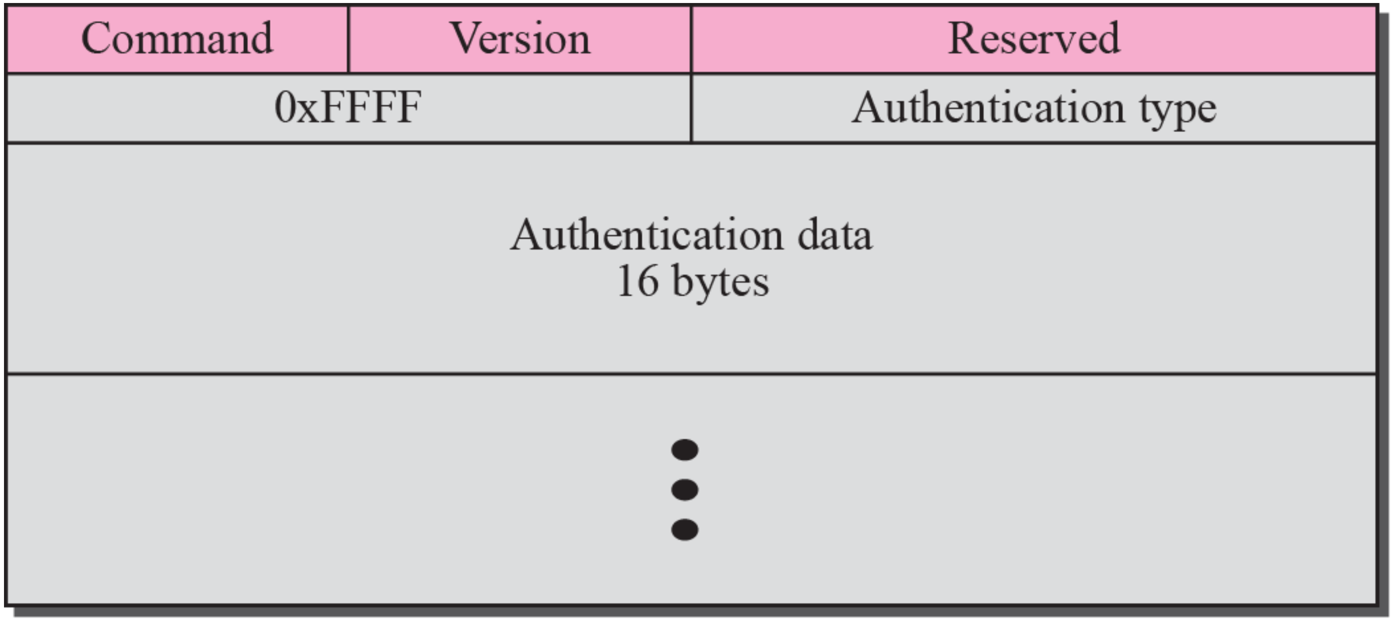
Subnet mask

Next-hop address

Distance

Repeated

Figure 11.16 *Authentication*



Command

Version

Reserved

0xFFFF

Authentication type

Authentication data
16 bytes

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.

Figure 11.17 *Concept of Link state routing*

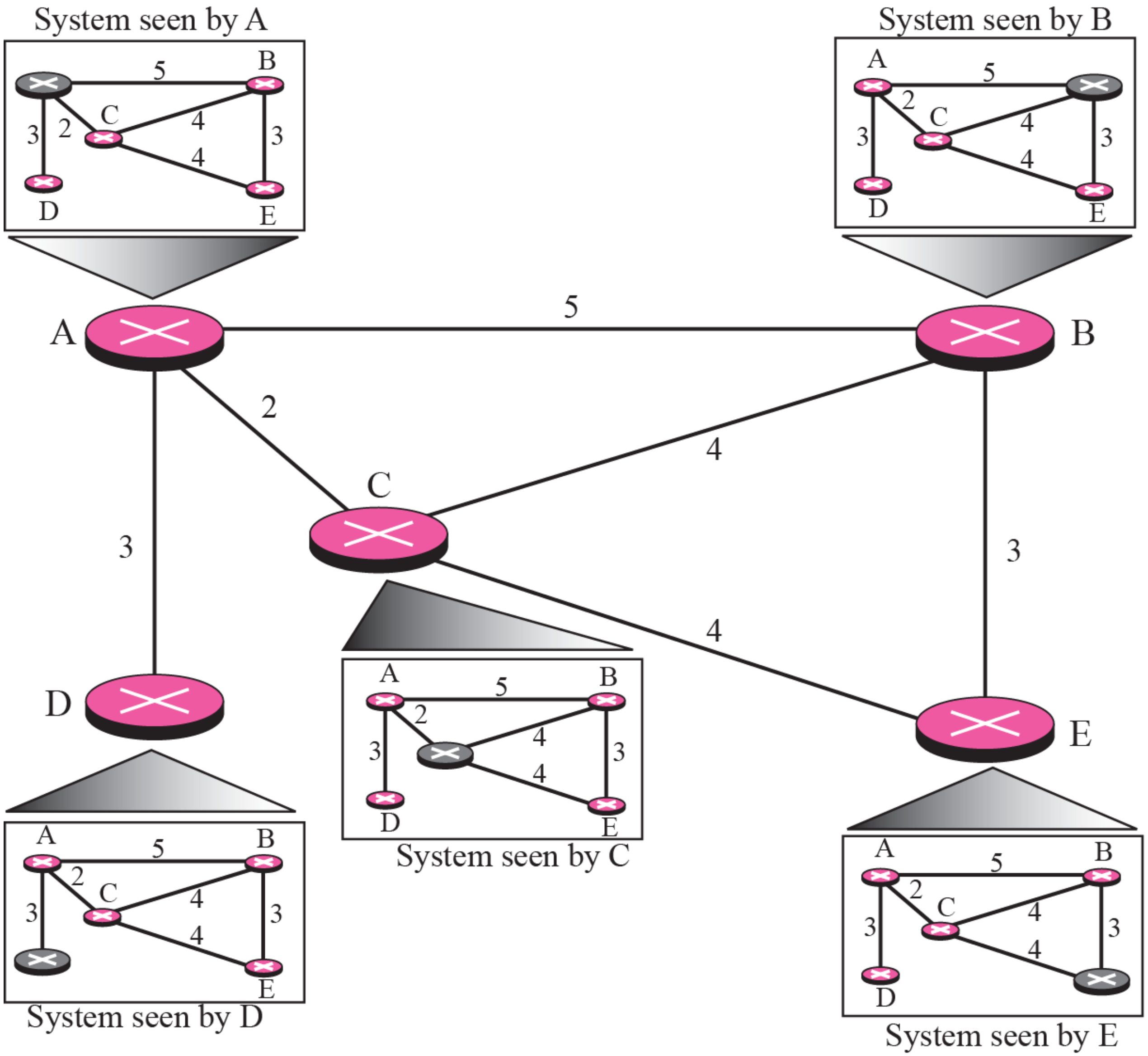
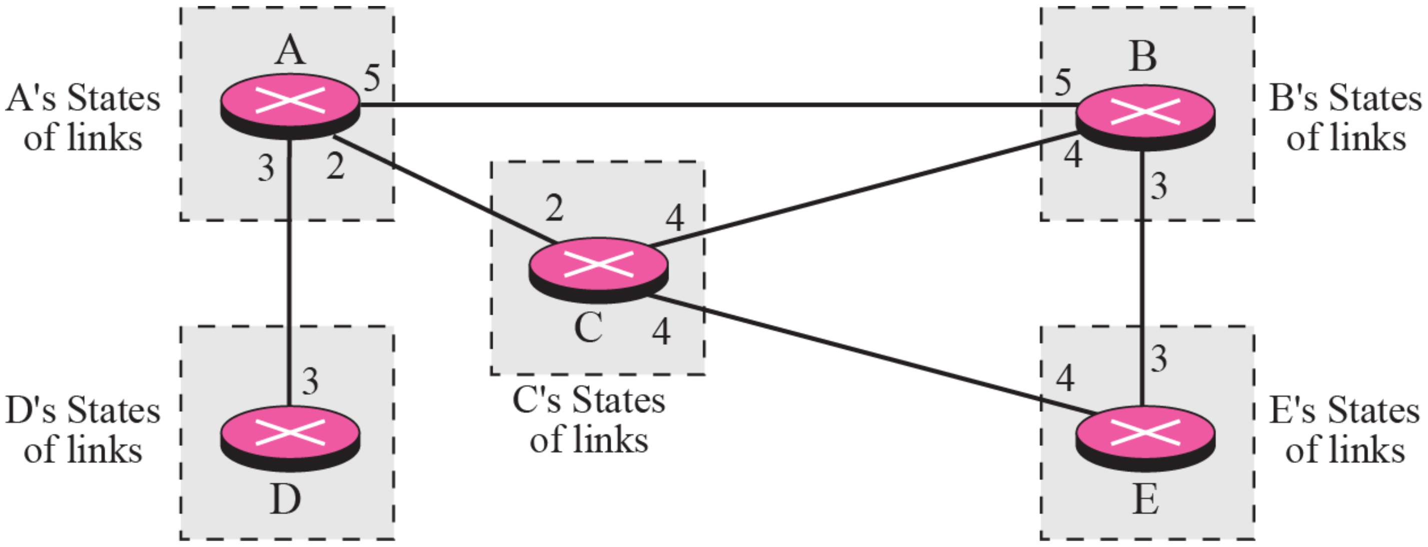
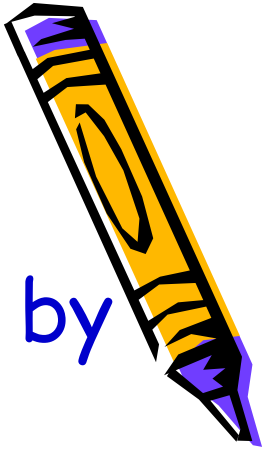


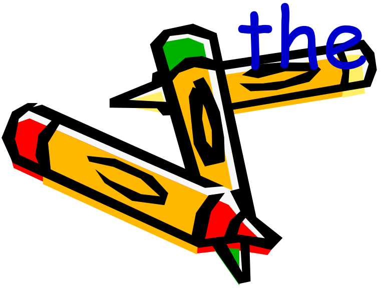
Figure 11.18 *Link state knowledge*



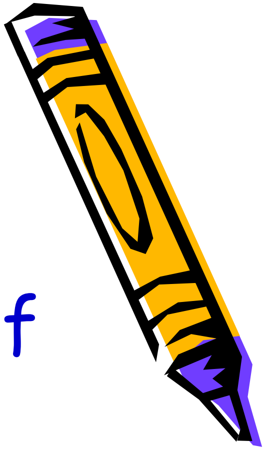
Building Routing Tables



- 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

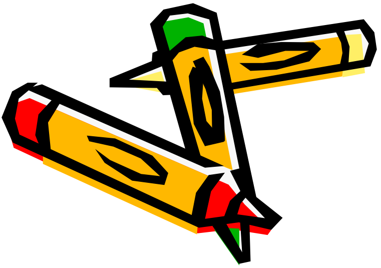
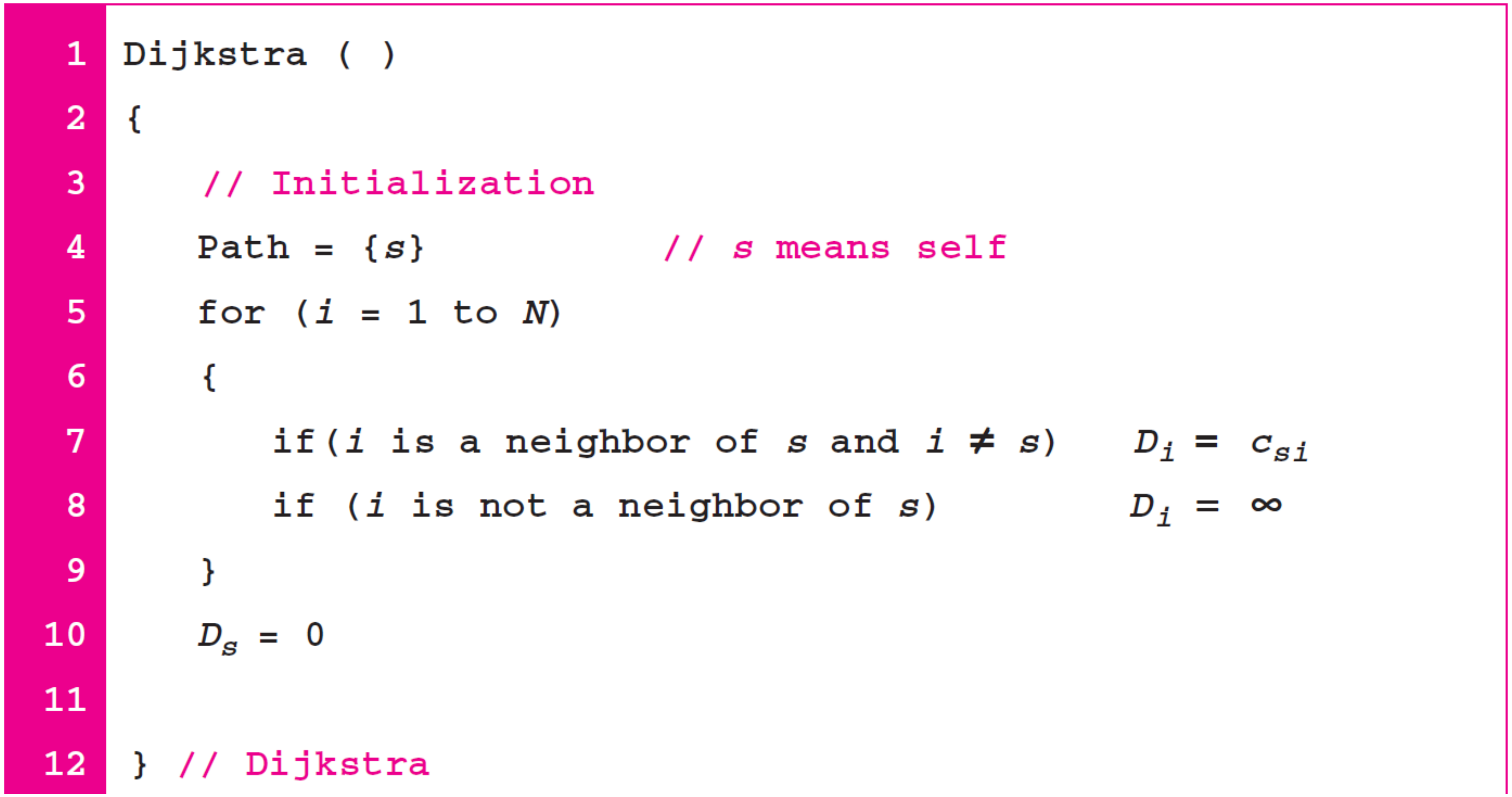


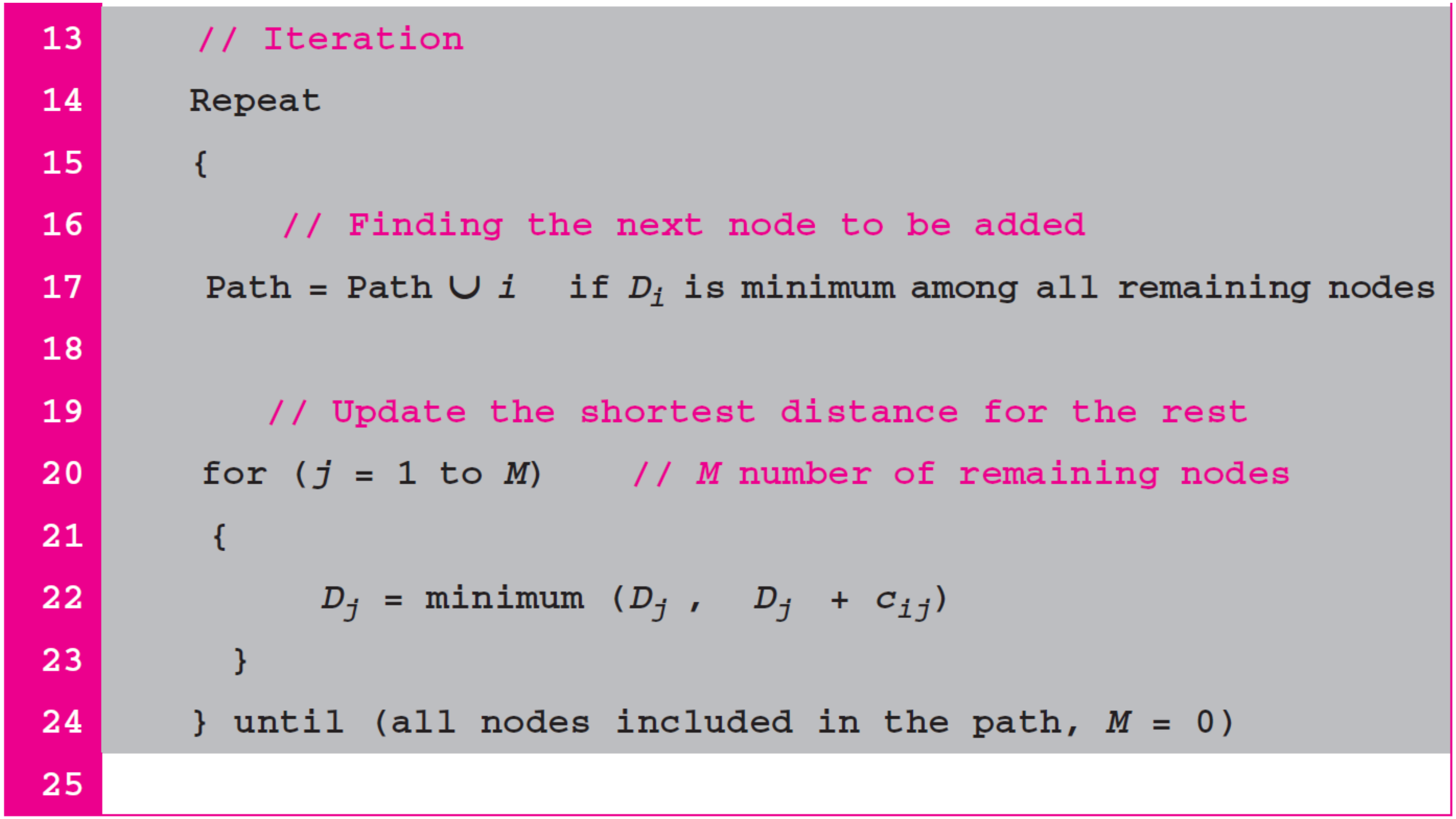
Table 11.3 *Dijkstra's Algorithm*



```
Dijkstra ( )
{
    // Initialization
    Path = {$s$}            // $s$ means self
    for ($i$ = 1 to $N$)
    {
        if($i$ is a neighbor of $s$ and $i \neq s$)    $D_i = c_{si}$
        if ($i$ is not a neighbor of $s$)          $D_i = \infty$
    }
    $D_s = 0$

} // Dijkstra
```

Continued



```
// Iteration
Repeat
{
    // Finding the next node to be added
 Path = Path $\cup$ $i$   if $D_i$ is minimum among all remaining nodes

   // Update the shortest distance for the rest
 for ($j$ = 1 to $M$)    // $M$ number of remaining nodes
  {
       $D_j$ = minimum ($D_j$ ,  $D_j$ + $c_{ij}$)
   }
} until (all nodes included in the path, $M$ = 0)
```

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

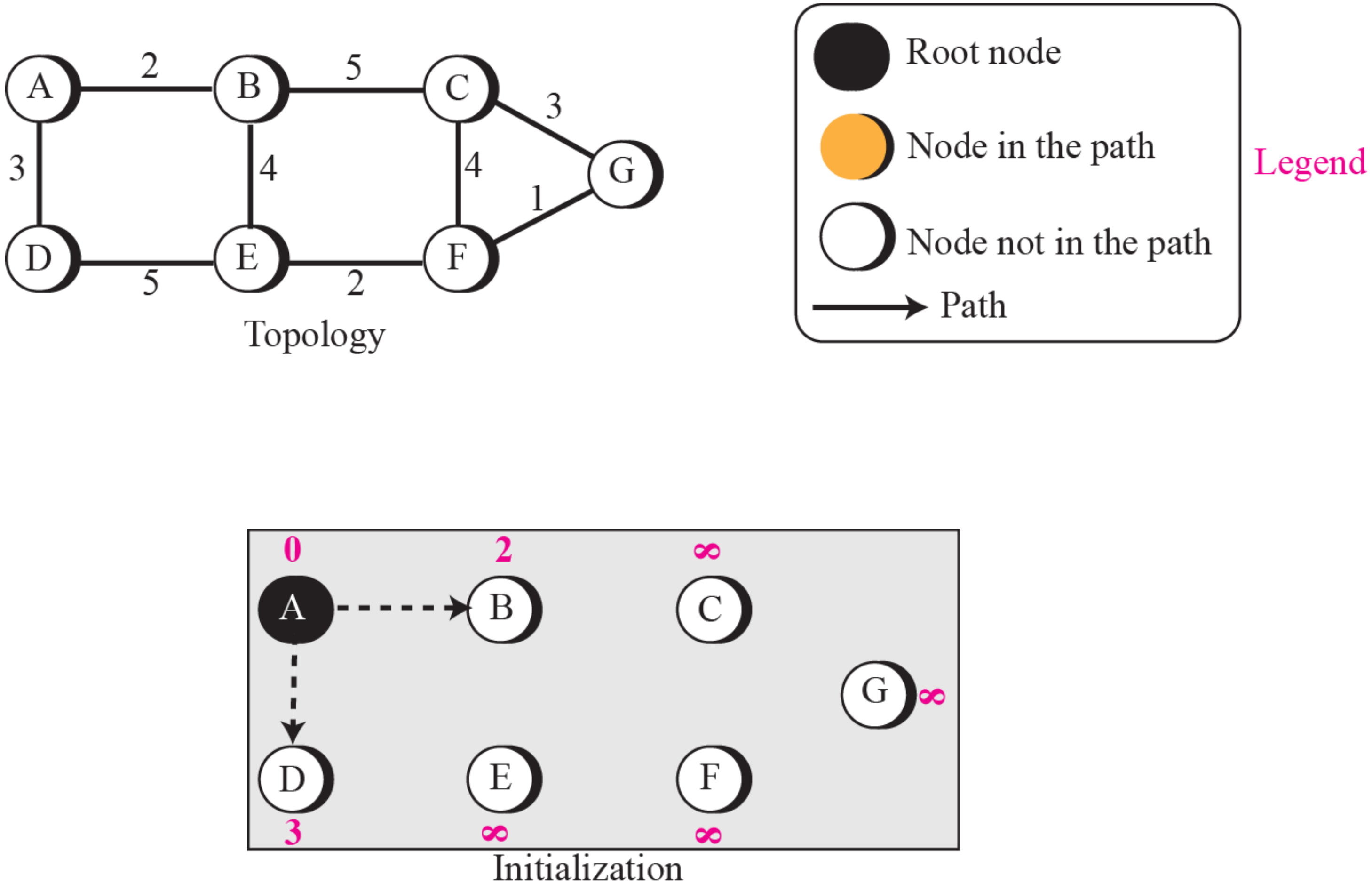
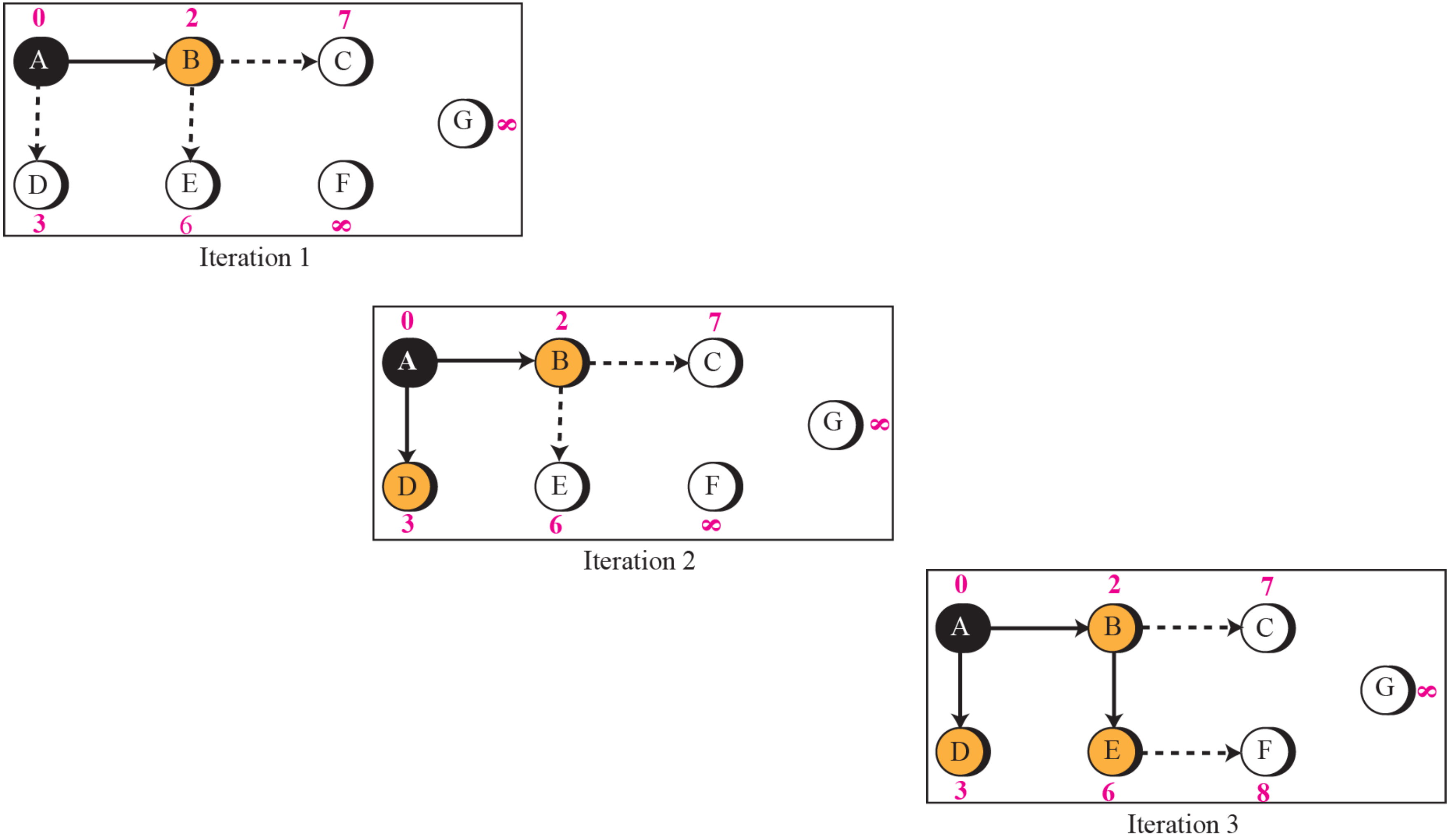


Figure 11.19 *Continued*

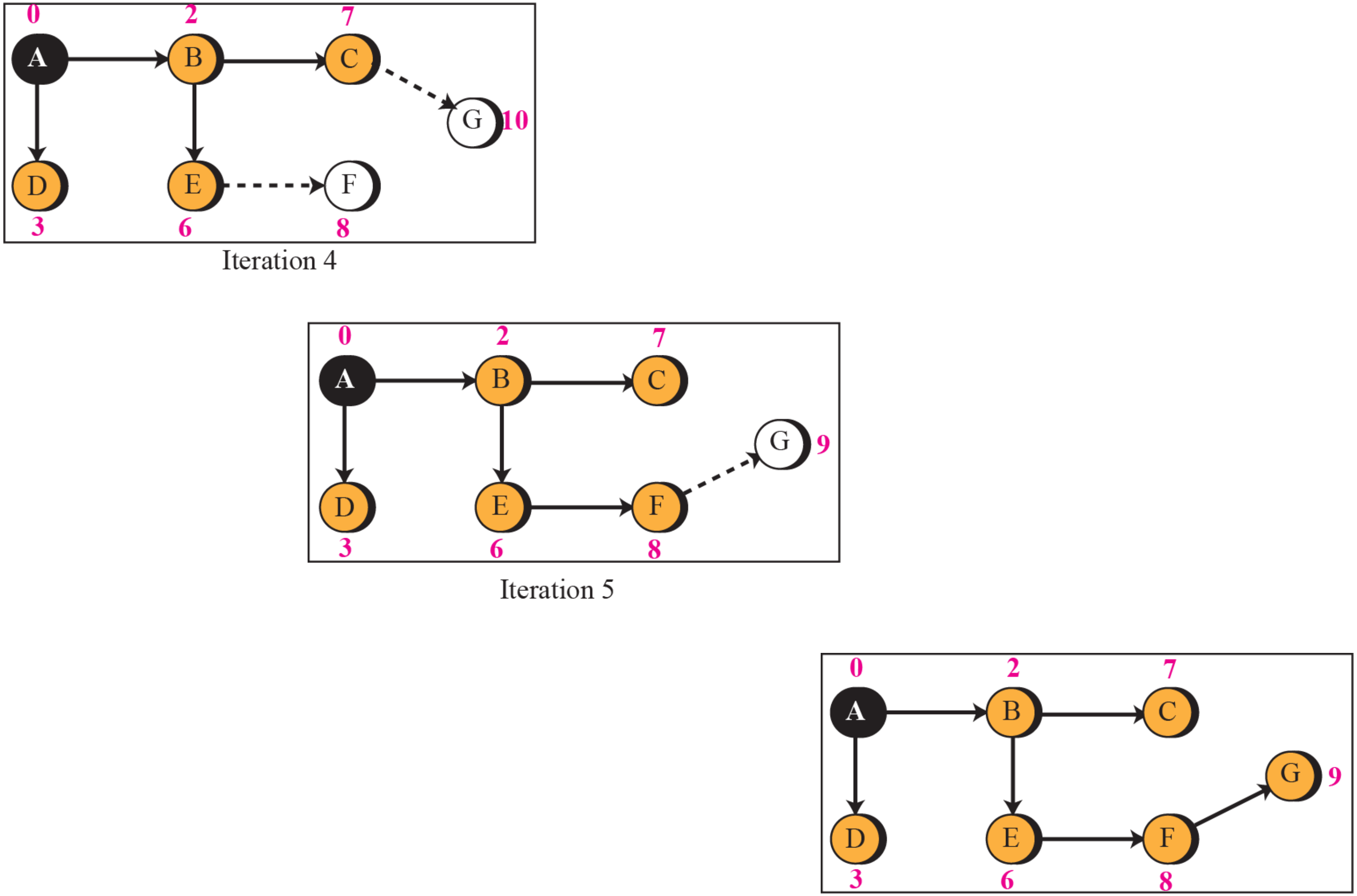


Iteration 1

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*

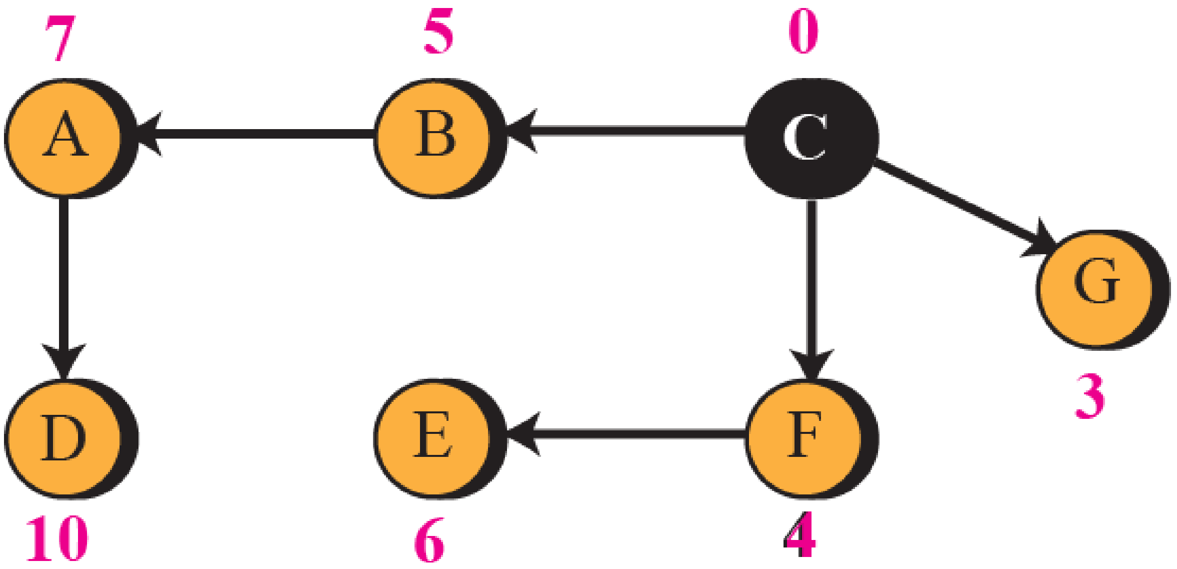


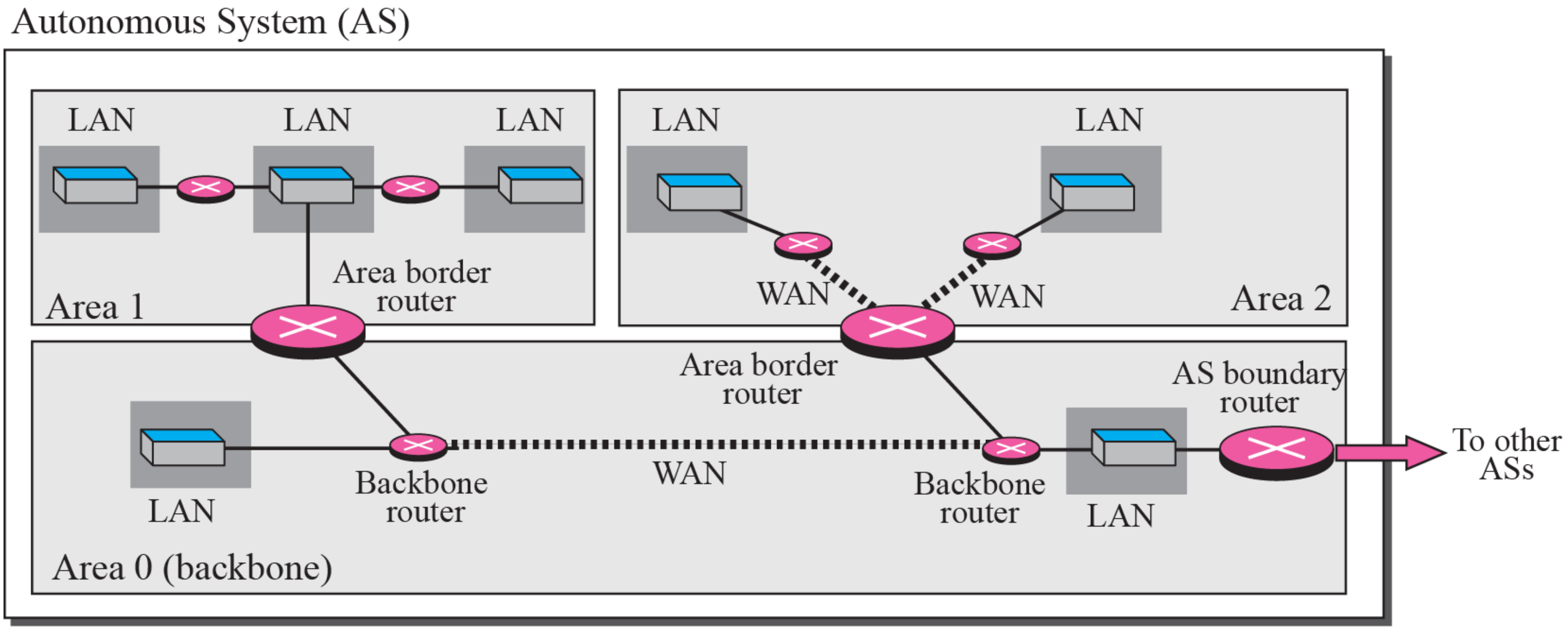
Table 11.4 *Routing Table for Node A*

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

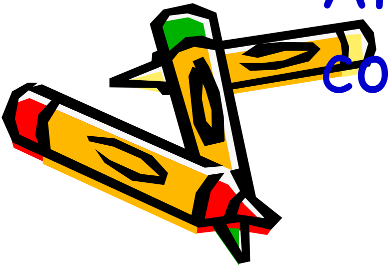
The Open Shortest Path First (OSPF) protocol is an intra-domain routing protocol based on link state routing. Its domain is also an autonomous system.

Figure 11.21 *Areas in an autonomous system*

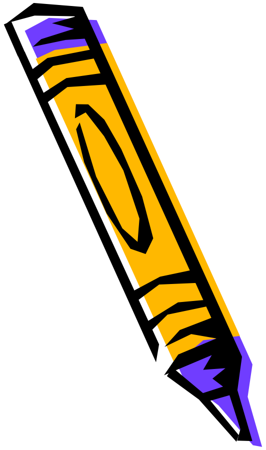


Area in OSPF (1)

- 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

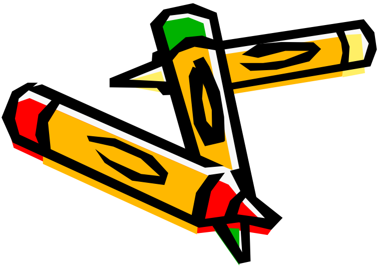


Figure 11.22 *Types of links*

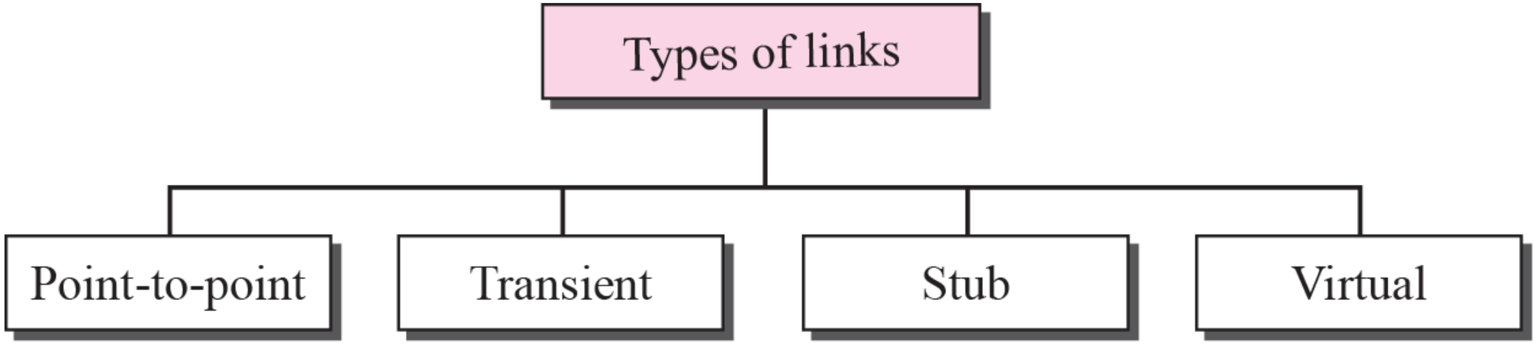
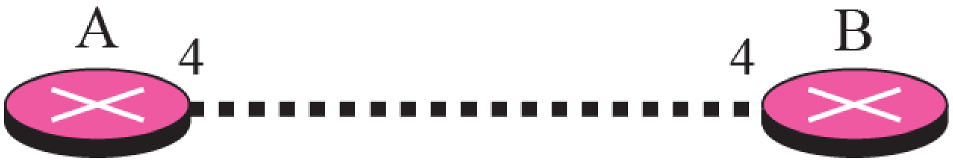
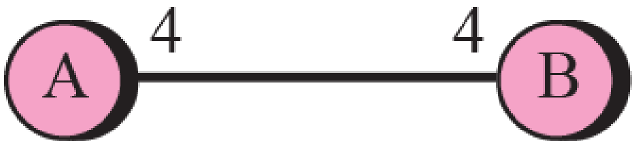


Figure 11.23 *Point-to-point link*

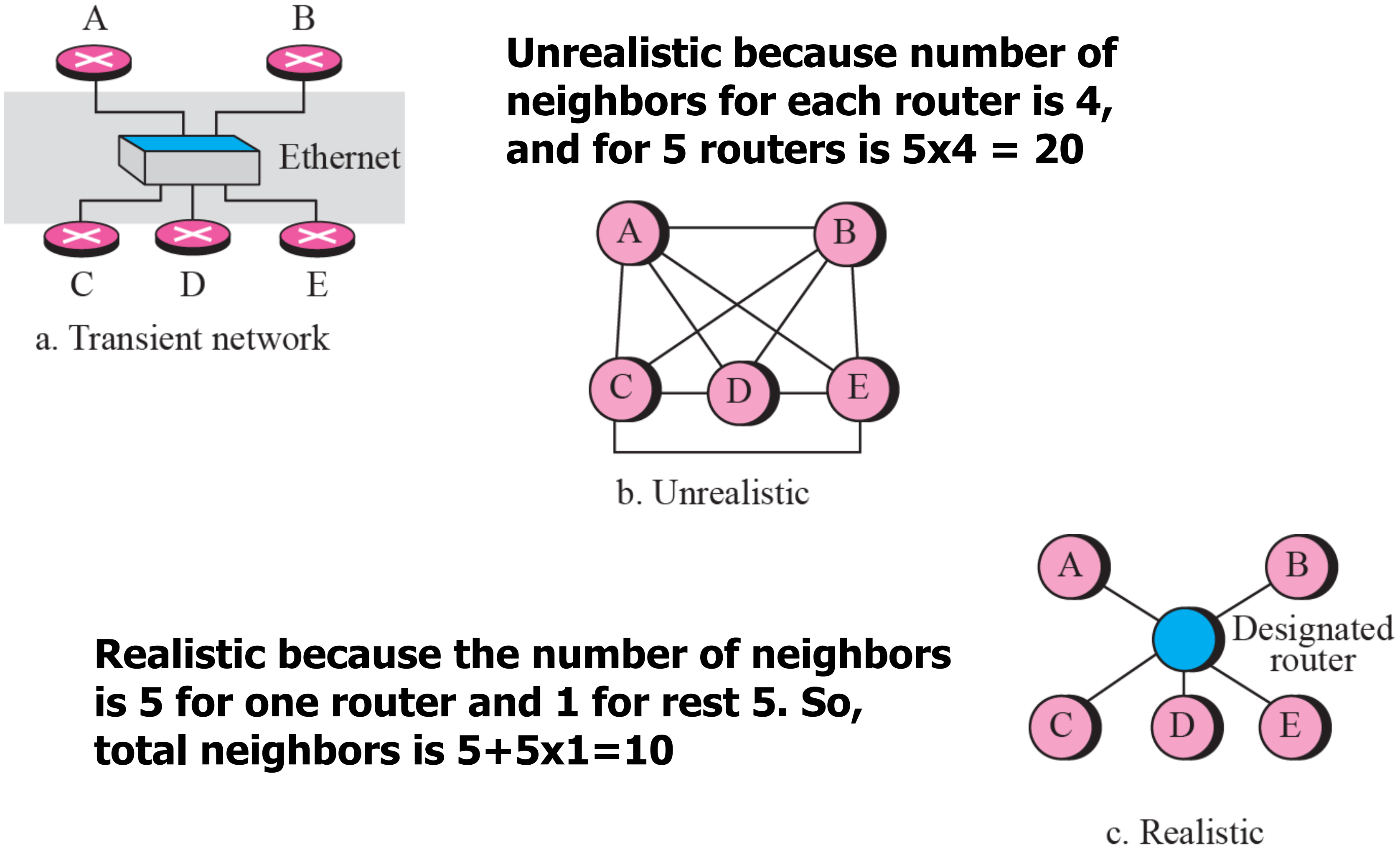


a. Point-to-point network



b. Representation

Figure 11.24 *Transient link*



a. Transient network

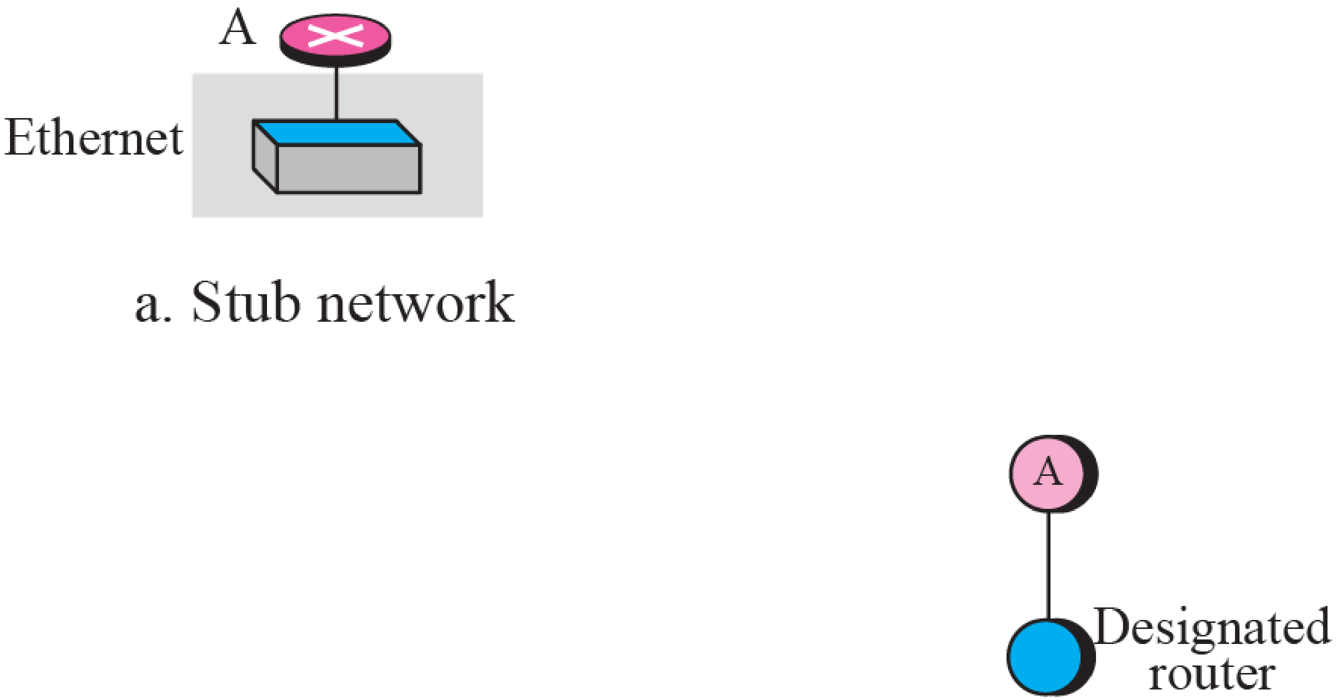
b. Unrealistic

c. Realistic

Unrealistic because number of neighbors for each router is 4, and for 5 routers is 5x4 = 20

Realistic because the number of neighbors is 5 for one router and 1 for rest 5. So, total neighbors is 5+5x1=10

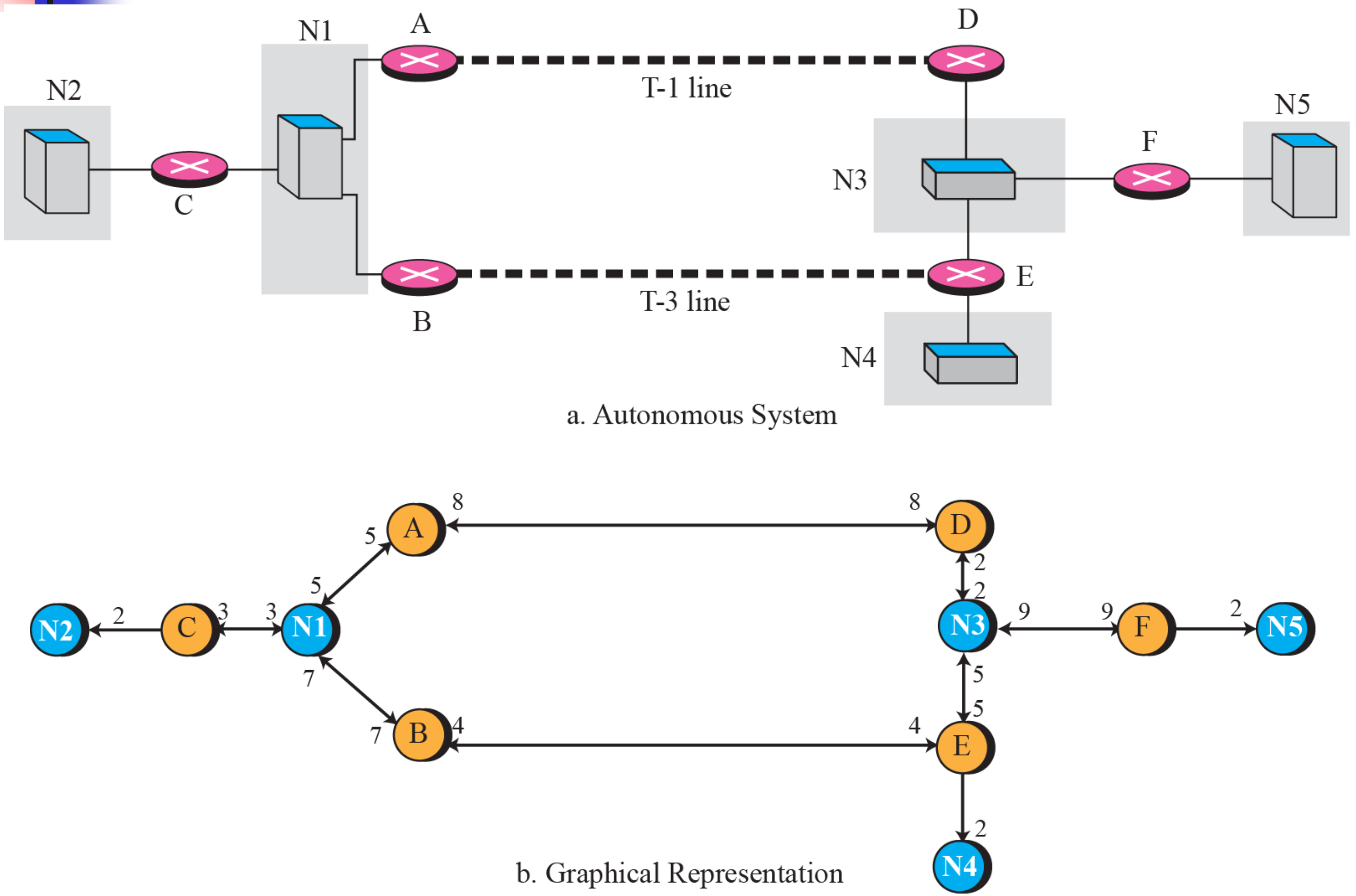
Figure 11.25 *Stub link*



a. Stub network

b. Representation

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



a. Autonomous System

b. Graphical Representation

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.

Figure 11.50 *Reachability*

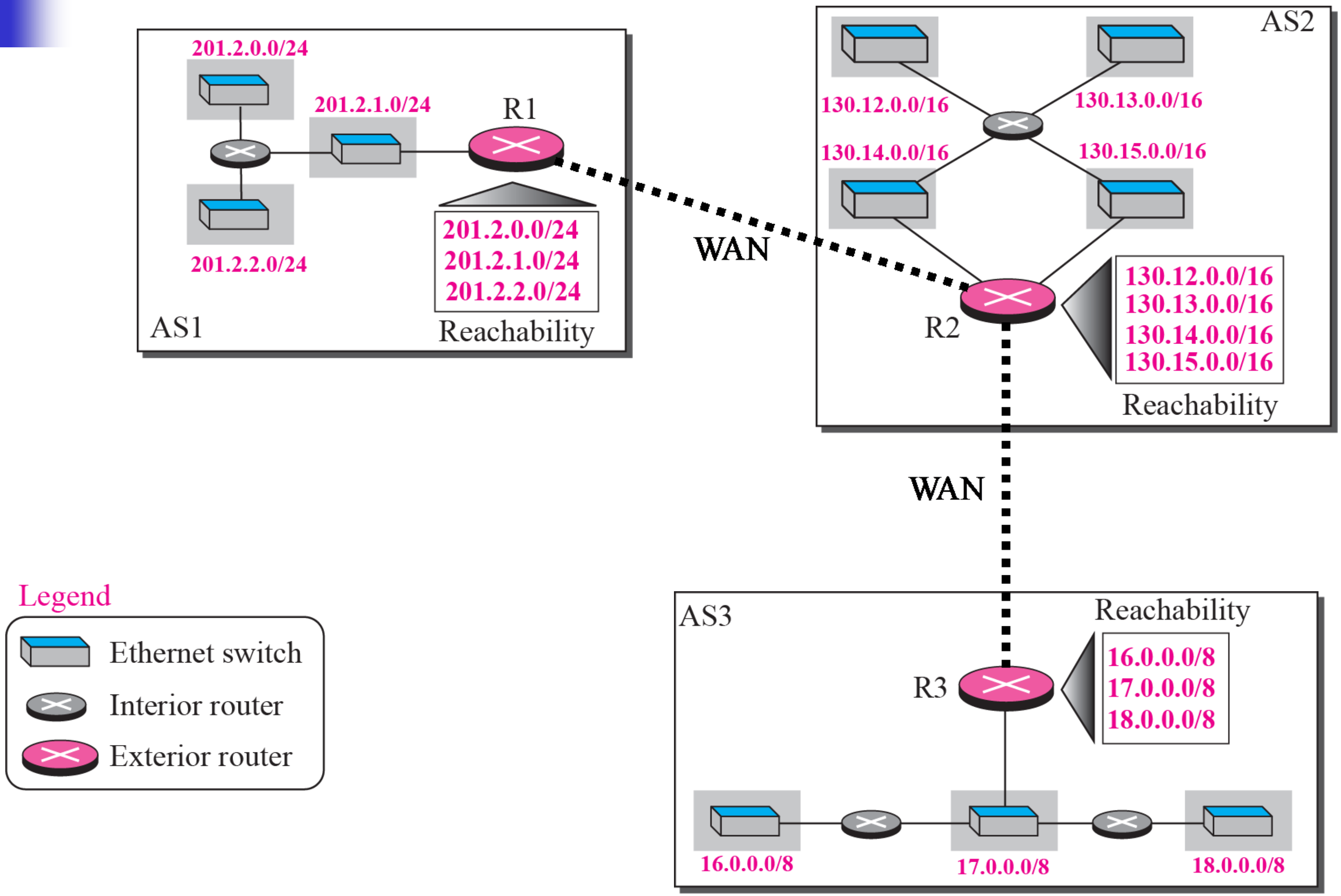
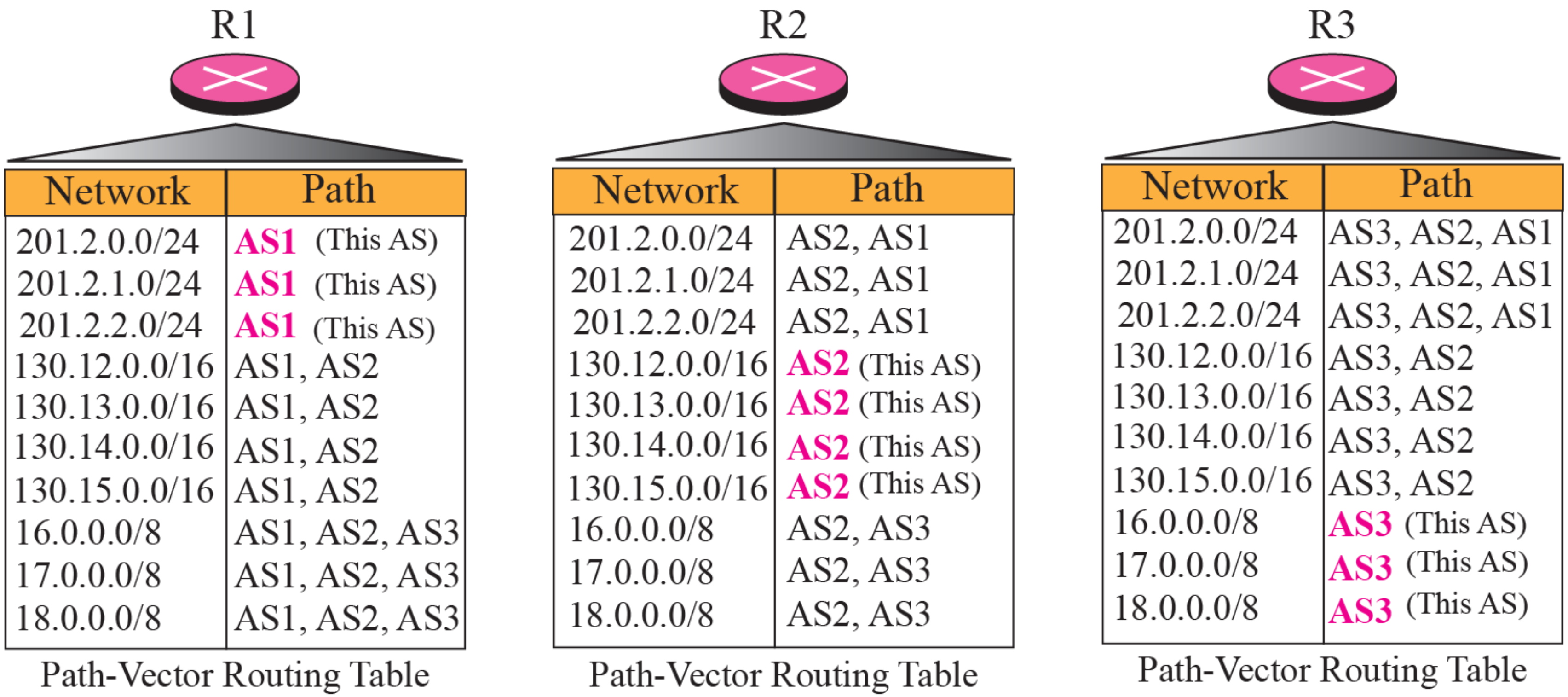


Figure 11.51 *Stabilized table for three autonomous system*



R1

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

R2

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

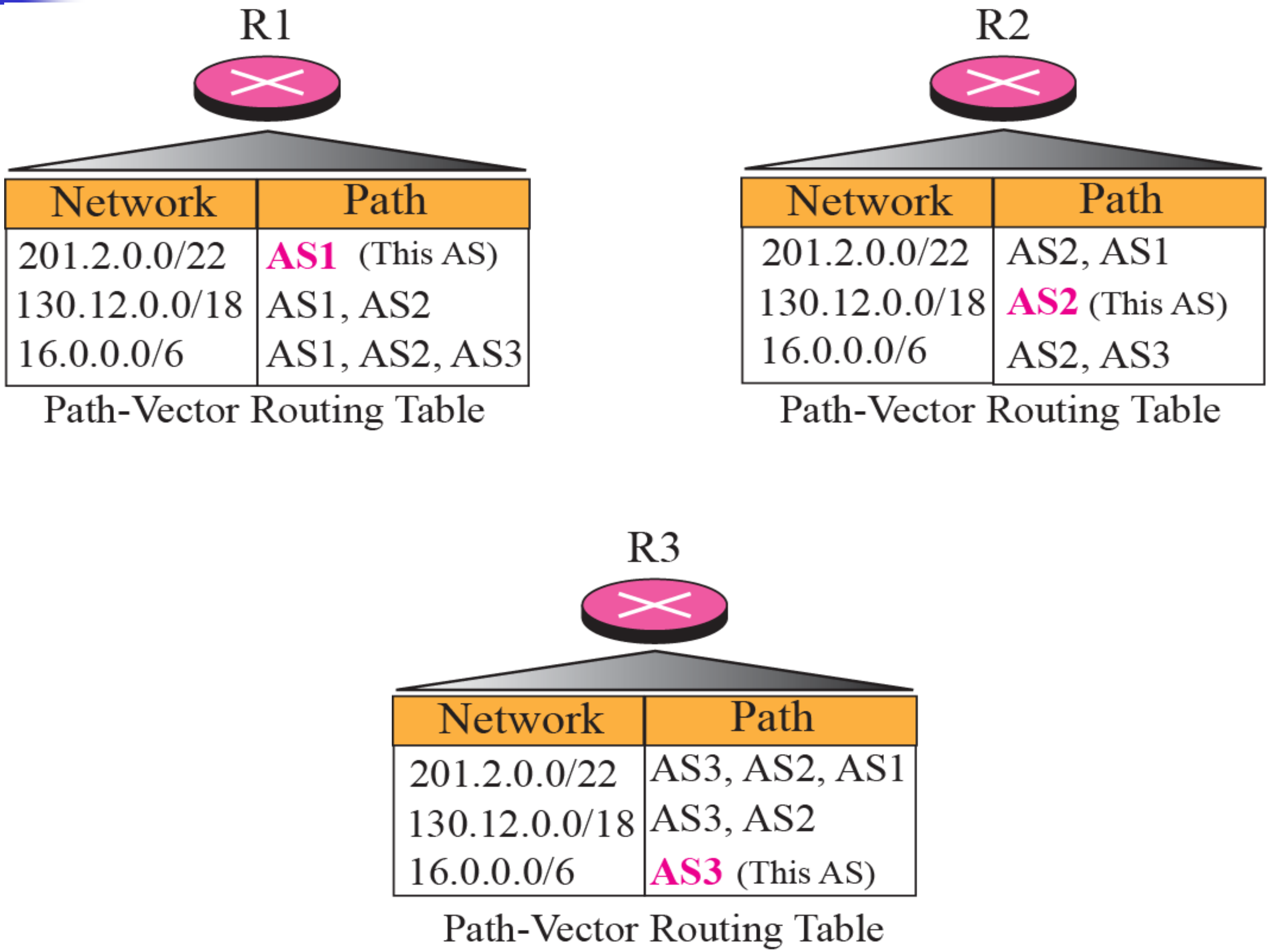
Path-Vector Routing Table

R3

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

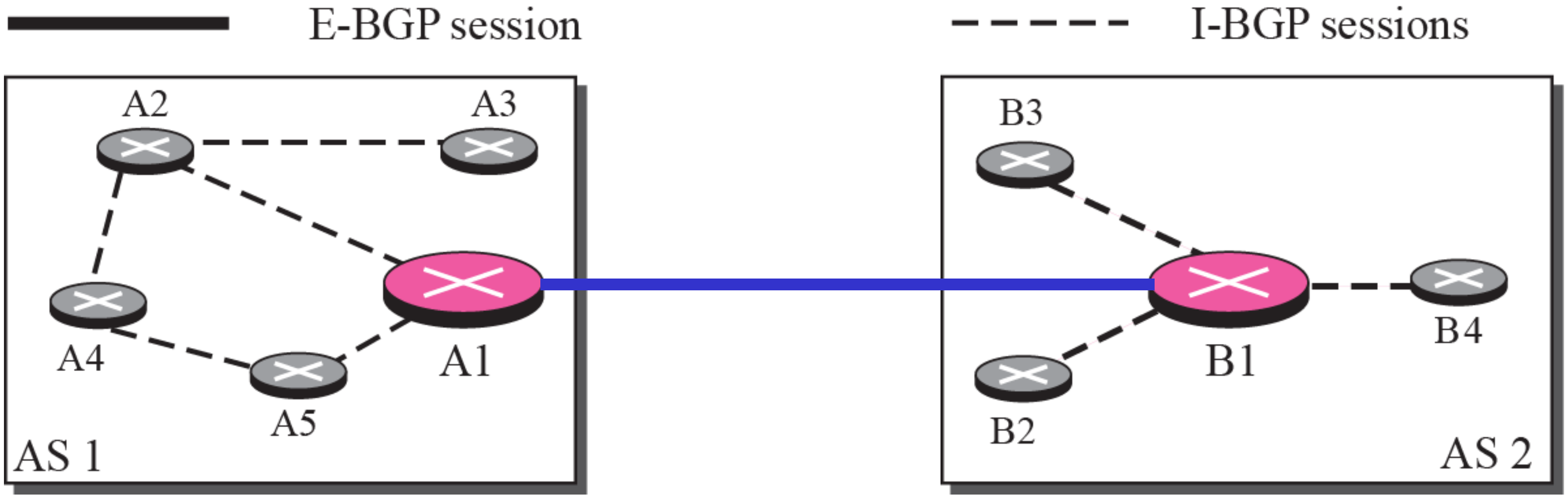
Network	Path
201.2.0.0/22	AS3, AS2, AS1
130.12.0.0/18	AS3, AS2
16.0.0.0/6	**AS3** (This AS)

Path-Vector Routing Table

11-8 BGP

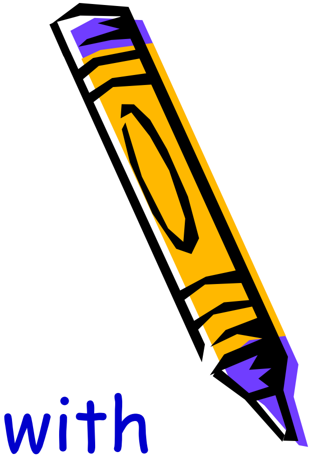
Border Gateway Protocol (BGP) is an interdomain routing protocol using path vector routing. It first appeared in 1989 and has gone through four versions.

Figure 11.53 *Internal and external BGP sessions*

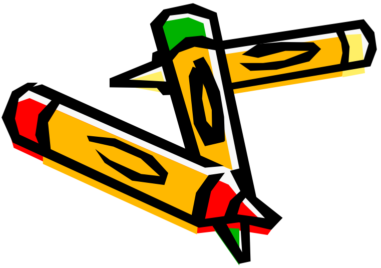


A speaker node advertises the path, not the metric of the nodes, in its AS or other ASs.

Path Vector Routing (1)

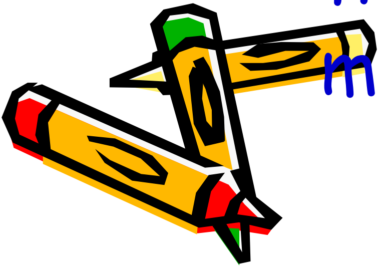


- Sharing
  - A speaker in an AS shares its table with immediate neighbors
- Updating
  - Adding the nodes that are not in its routing table and adding its own AS and the AS that sent the table
  - The routing table shows the path completely



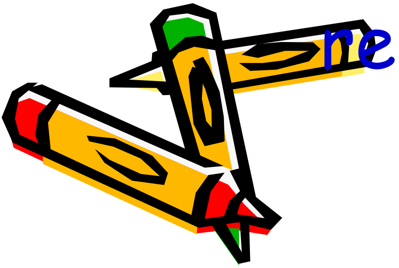
Path Vector Routing (2)

- 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