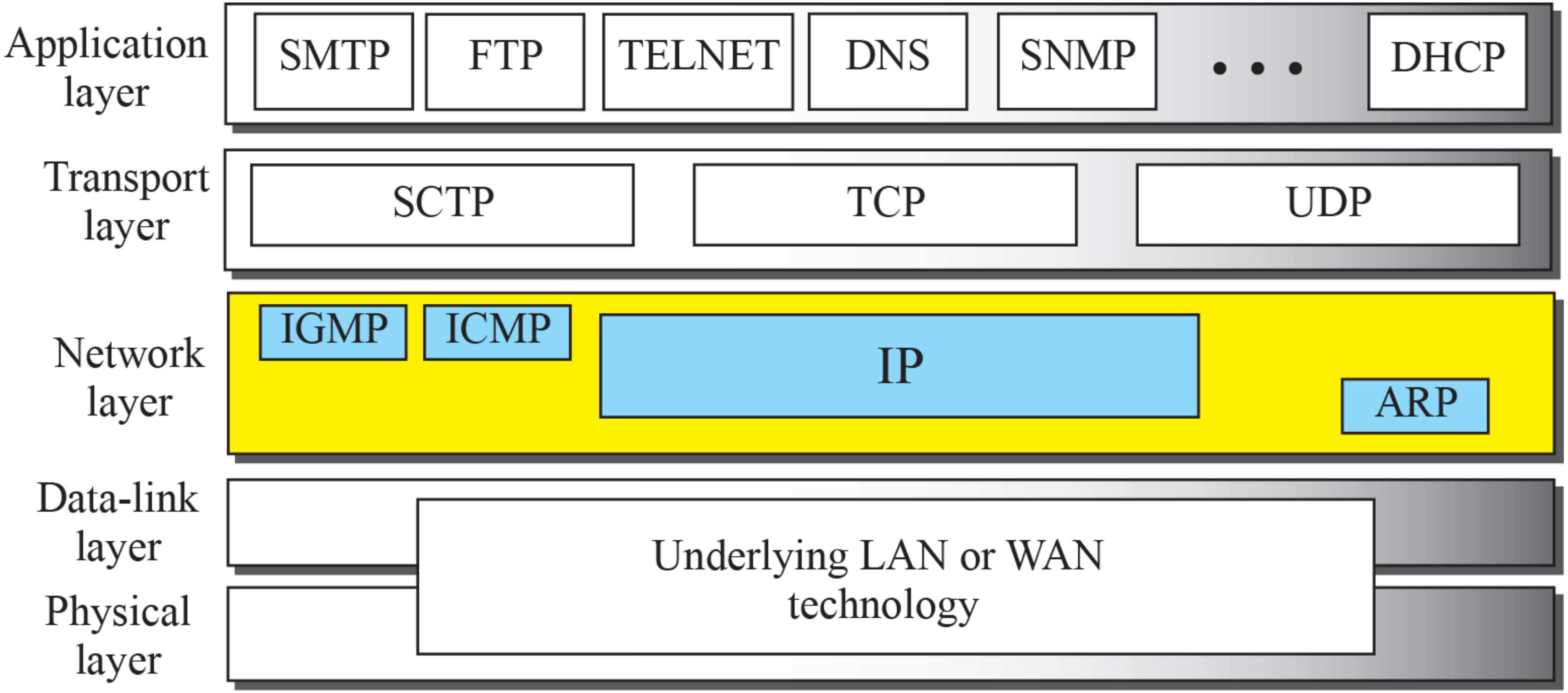
Network Layer

NETWORK-LAYER PROTOCOLS

In this section, we show how the network layer is implemented in the TCP/IP protocol suite. The protocols in the network layer have gone through several versions; in this section, we concentrate on the current version (4), in the last section of this chapter, we briefly discuss version 6, which is on the horizon.

Position of IP and other network-layer protocols in TCP/IP protocol suite



IPv4 Datagram Format

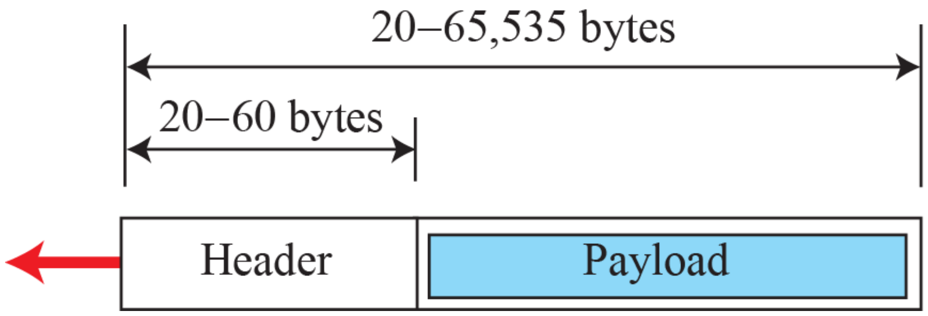
Packets used by the IP are called datagrams. A datagram is a variable-length packet consisting of two parts: header and payload (data). The header is 20 to 60 bytes in length and contains information essential to routing and delivery. It is customary in TCP/IP to show the header in 4-byte sections.

(continued)

- Fragmentation
  - Maximum Transfer Unit (MTU)
  - Fields Related to Fragmentation

- Security of IPv4 Datagrams
  - Packet Sniffing
  - Packet Modification
  - IP Spoofing
  - IPSec

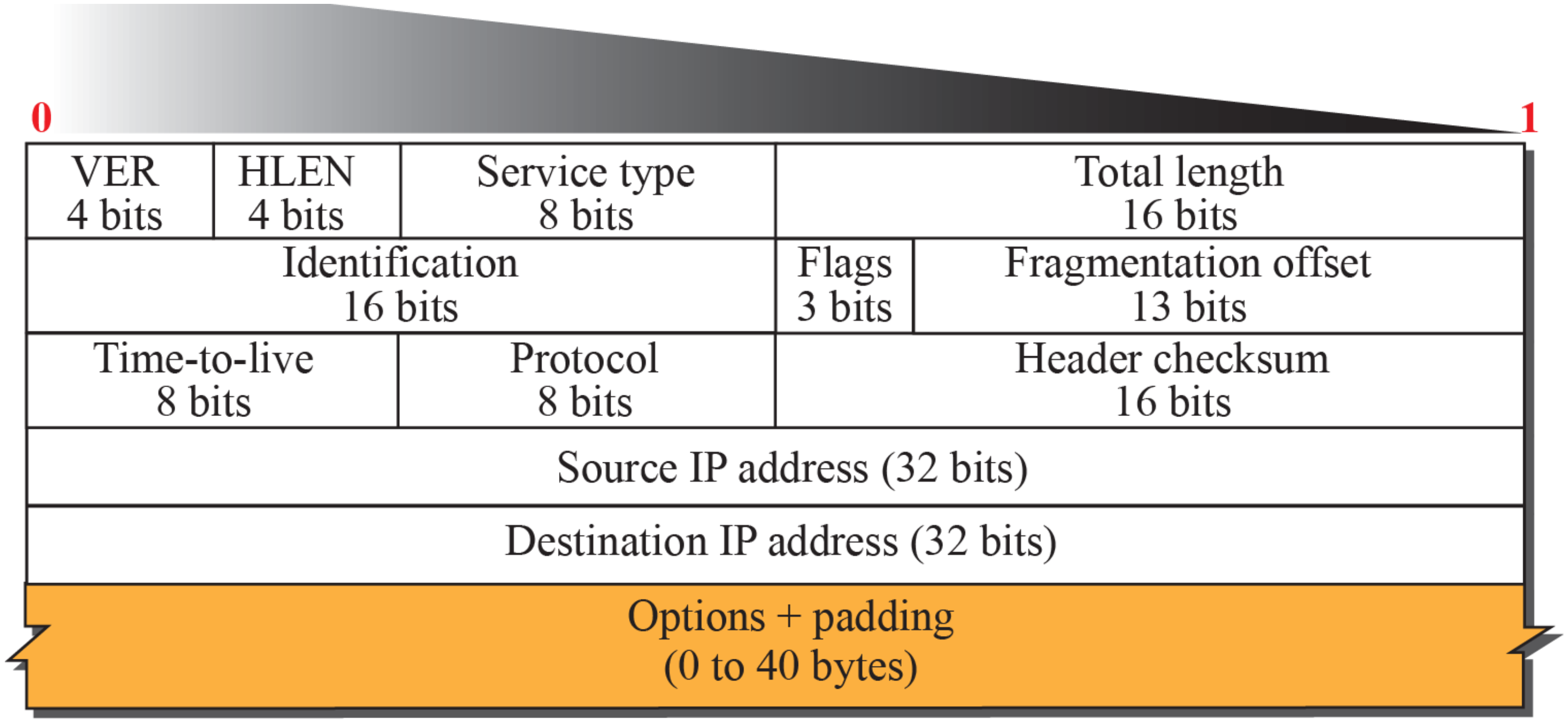
IP datagram



a. IP datagram

Legend

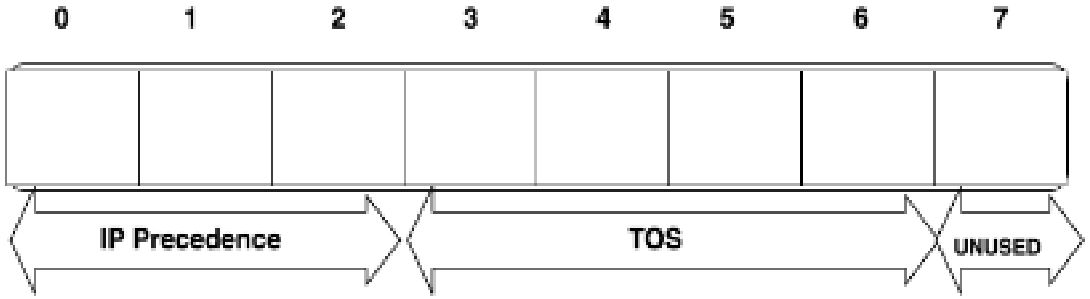
VER: version number
HLEN: header length
byte: 8 bits

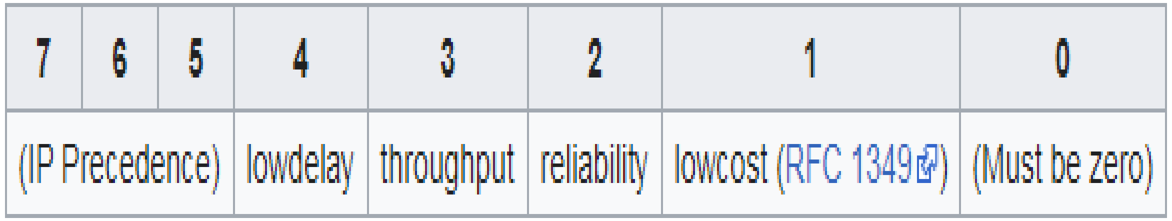


VER 4 bits	HLEN 4 bits	Service type 8 bits	Total length 16 bits
Identification 16 bits		Flags 3 bits	Fragmentation offset 13 bits
Time-to-live 8 bits		Protocol 8 bits	Header checksum 16 bits
Source IP address (32 bits)			
Destination IP address (32 bits)			
Options + padding (0 to 40 bytes)			

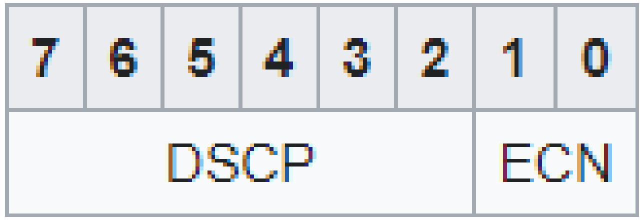
b. Header format

Type of Service - is how the datagram should be used, e.g. *reliability, delay, jitter,* and *bandwidth* etc. This TOS field is now used by **Differentiated Services** and is called the **Diff Serv Code Point (DSCP)**.





7	6	5	4	3	2	1	0
(IP Precedence)			lowdelay	throughput	reliability	lowcost (RFC 1349)	(Must be zero)

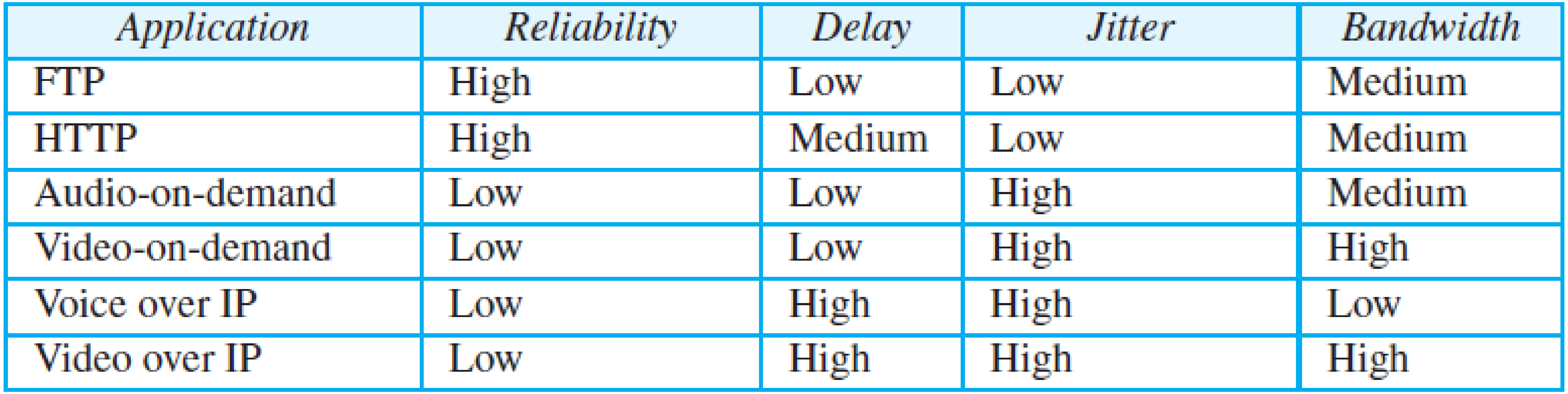


7	6	5	4	3	2	1	0
DSCP						ECN	

DSCP bit value, higer has more priority
000 (0) - Routine
001 (1) - Priority
010 (2) - Immediate
011 (3) - Flash
100 (4) - Flash Override
101 (5) - Critical
110 (6) - Internetwork Control
111 (7) - Network Control

ECN – Explicit Congestion Notification

Type of Service



Application	*Reliability*	*Delay*	*Jitter*	*Bandwidth*
FTP	High	Low	Low	Medium
HTTP	High	Medium	Low	Medium
Audio-on-demand	Low	Low	High	Medium
Video-on-demand	Low	Low	High	High
Voice over IP	Low	High	High	Low
Video over IP	Low	High	High	High

a. *Constant Bit Rate (CBR)*. This class is used for emulating circuit switching. CBR applications are quite sensitive to cell-delay variation. Examples of CBR are telephone traffic, video conferencing, and television.

b. *Variable Bit Rate-Non Real Time (VBR-NRT)*. Users in this class can send traffic at a rate that varies with time depending on the availability of user information. An example is multimedia e-mail.

c. *Variable Bit Rate-Real Time (VBR-RT)*. This class is similar to VBR–NRT but is designed for applications such as interactive compressed video that are sensitive to cell delay variation.

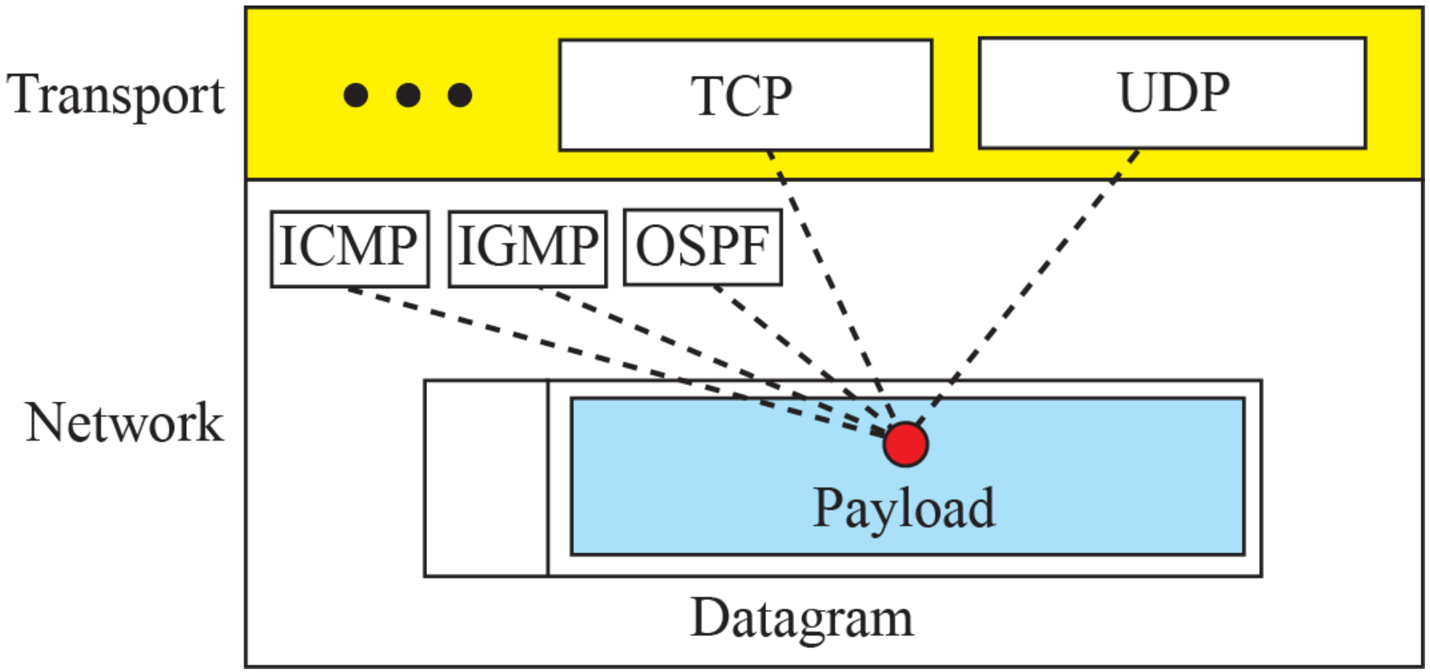
d. *Available Bit Rate (ABR)*. This class of ATM services provides rate-based flow control and is aimed at data traffic such as file transfer and e-mail.

e. *Unspecified Bit Rate (UBR)*. This class includes all other classes and is widely used today for TCP/IP.

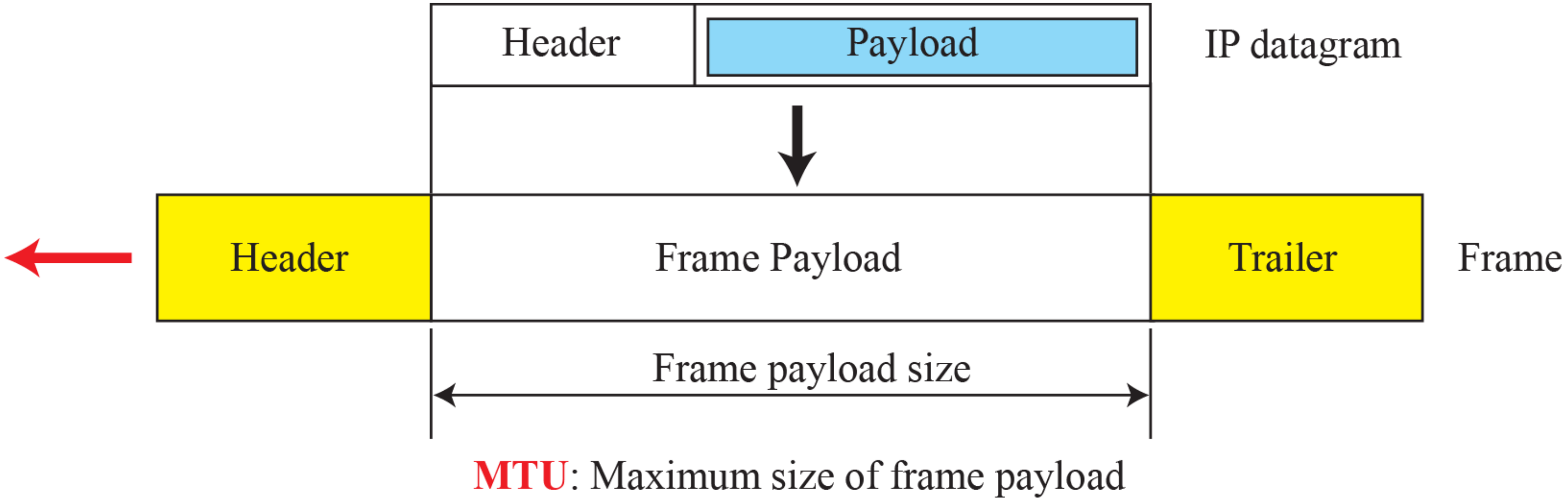
Multiplexing and demultiplexing using the value of the protocol field

ICMP: 01
IGMP: 02
TCP: 06
UDP: 17
OSPF: 89

Some protocol values

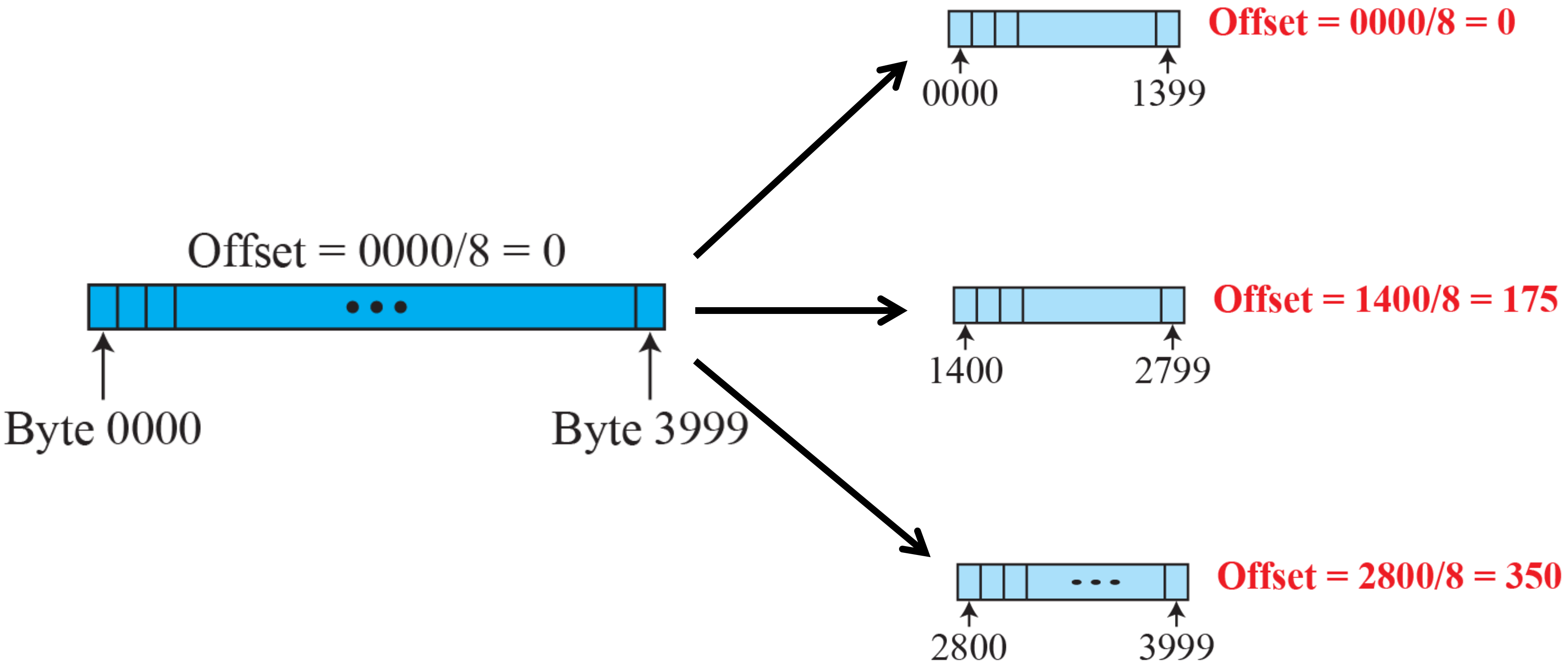


Fragmentation - Maximum transfer unit (MTU)



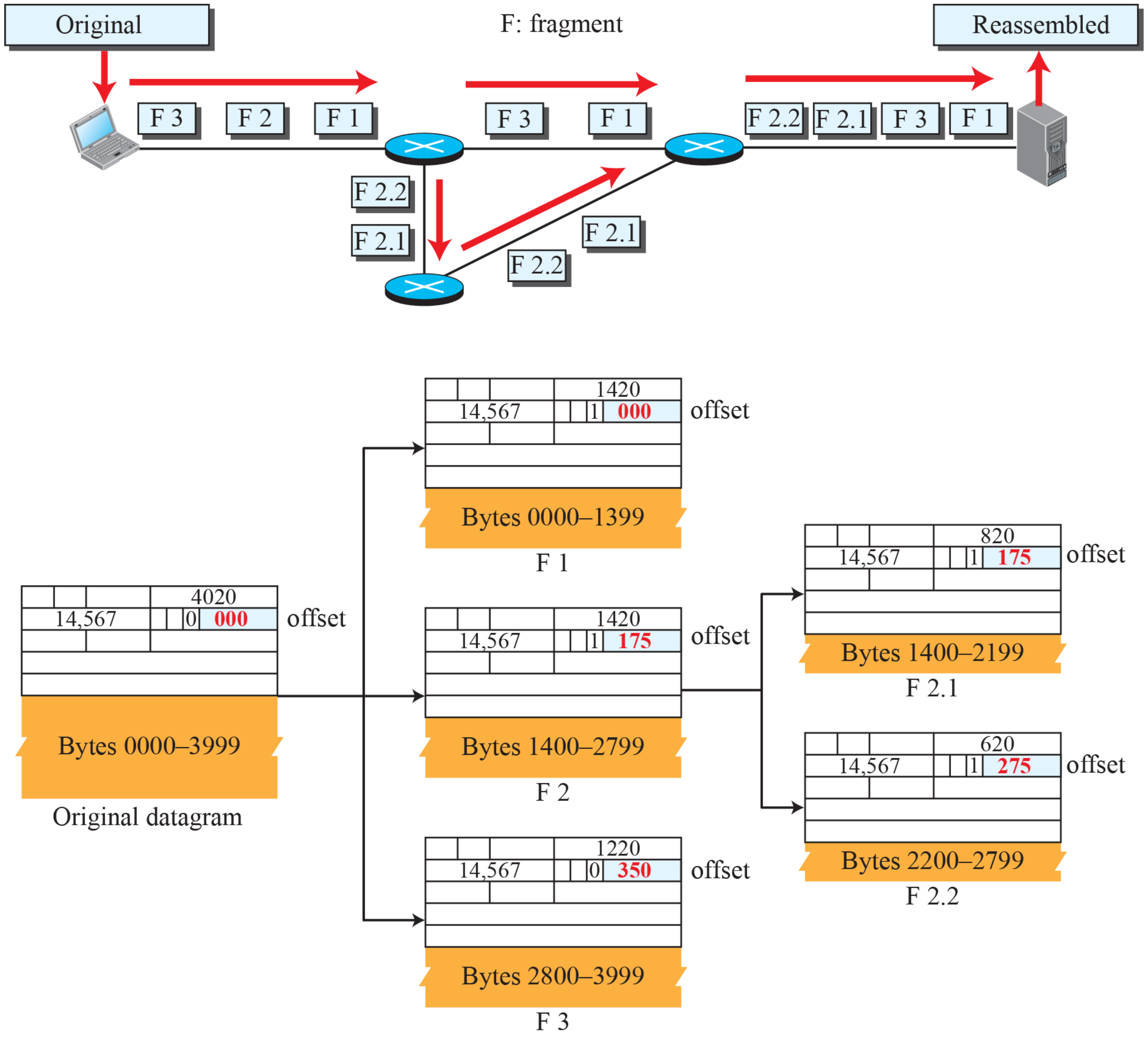
Fragmentation example

Flags - Bit 0 is always 0 and is reserved. Bit 1(DF) indicates whether a datagram can be fragmented (0) or not (1). Bit 2(MF) indicates to the receiving unit whether the fragment is the last one in the datagram (1) or if there are still more fragments to come (0).



The fragment offset field is measured in units of eight-byte blocks. It is 13 bits long and specifies the offset of a particular fragment relative to the beginning of the original unfragmented IP datagram.

Detailed fragmentation example



Options

The variable part comprises the options that can be a maximum of 40 bytes (in multiples of 4-bytes) to preserve the boundary of the header.

Single-Byte Options

- ***No Operation-*** A *no-operation option* is a 1-byte option used as a filler between options.
- ***End of Option -*** An *end-of-option option* is a 1-byte option used for padding at the end of the option field. It, however, can only be used as the last option.

Multliple-Byte Options

- ***Record Route -*** A *record route option* is used to record the Internet routers that handle the datagram. It can list up to nine router addresses.
- ***Strict Source Route -*** A *strict source route option* is used by the source to predetermine a route for the datagram as it travels through the Internet.
- ***Loose Source Route -*** A *loose source route option* is similar to the strict source route, but it is less rigid. Each router in the list must be visited, but the datagram can visit other routers as well.
- ***Timestamp -*** A *timestamp option* is used to record the time of datagram processing by a router.

Security of IPv4 Datagrams

Packet Sniffing
Packet Modification
IP Spoofing

IP Security(IPSec)
- Defining algorithms
- Packet Encryption & Keys
- Data Integrity
- Origin Authentication

Routing

Routing is the act of moving information across an inter-network from a source to a destination. Along the way, at least one intermediate node typically is encountered.

It’s also referred to as the process of choosing a path over which to send the packets.

The routing algorithm is the part of the network layer software responsible for deciding which output line an incoming packet should be transmitted on, i.e. what should be the next intermediate node for the packet.(Forwarding)

Routing protocols use metrics to evaluate what path will be the best for a packet to travel. A ***metric*** is a standard of measurement; such as path bandwidth, reliability, delay, current load on that path etc; that is used by routing algorithms to determine the optimal path to a destination.

Routing algorithms fill routing tables with a variety of information

Router is a network layer device which does the forwarding

Routing Algorithm Metrics

- Path length /Hop count
- Delay
- Bandwidth
- Load
- Communication cost
- Reliability

Desirable properties of a router

- **Correctness and simplicity:** The packets are to be correctly delivered. Simpler the routing algorithm, it is better.
- **Robustness:** Ability of the network to deliver packets via some route even in the face of failures.
- **Stability:** The algorithm should converge to equilibrium fast in the face of changing conditions in the network.
- **Fairness and optimality:** obvious requirements, but conflicting.
- **Efficiency:** Minimum overhead

Design parameters of Routing Protocol:

- **Performance Criteria:** Number of hops, Cost, Delay, Throughput, etc
- **Decision Time:** Per packet basis (Datagram) or per session (Virtual-circuit) basis
- **Decision Place:** Each node (distributed), Central node (centralized), Originated node (source)
- **Network Information Source:** None, Local, Adjacent node, Nodes along route, All nodes
- **Network Information Update Timing:** Continuous, Periodic, Major load change, Topology change

Classification of Routing algorithms

- Static versus Adaptive
- Single-path versus multi-path
- Intra-domain versus inter-domain
- Flat versus hierarchical
- Link-state versus distance vector
- Host-intelligent versus router-intelligent
- Unicast Routing versus Multicast routing

UNICAST ROUTING

In an internet, the goal of the network layer is to deliver a datagram from its source to its destination or destinations. If a datagram is destined for only one destination (one-to-one delivery), we have unicast routing. In this section and the next, we discuss only unicast routing; multicast and broadcast routing will be discussed later in the chapter.

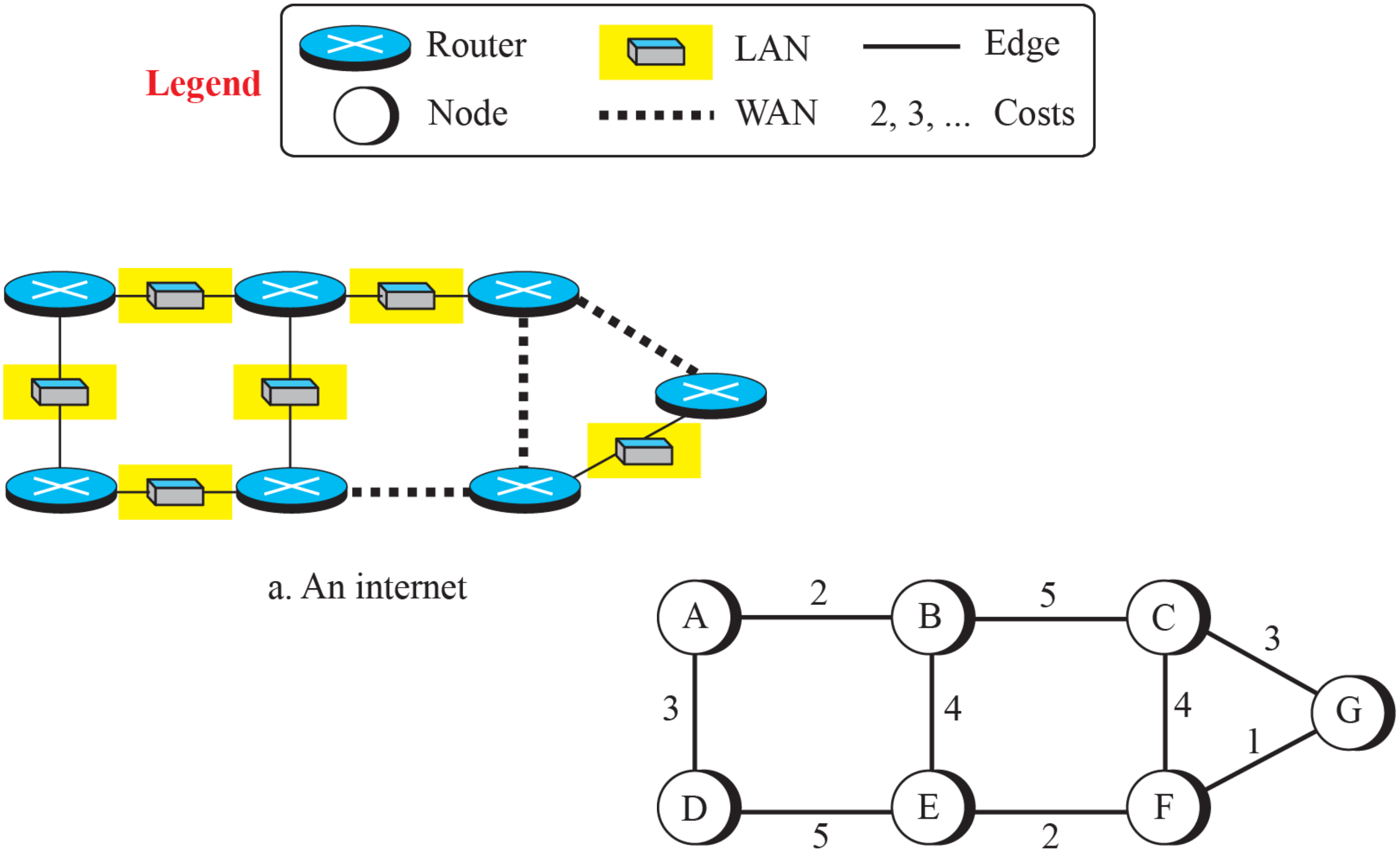
General Idea

In unicast routing, a packet is routed, hop by hop, from its source to its destination by the help of forwarding tables. The source host needs no forwarding table because it delivers its packet to the default router in its local network. The destination host needs no forwarding table either because it receives the packet from its default router in its local network. This means that only the routers that glue together the networks in the internet need forwarding tables.

(continued)

- ❑ An Internet as a Graph
- ❑ Least-Cost Routing
- ❑ Least-Cost Trees

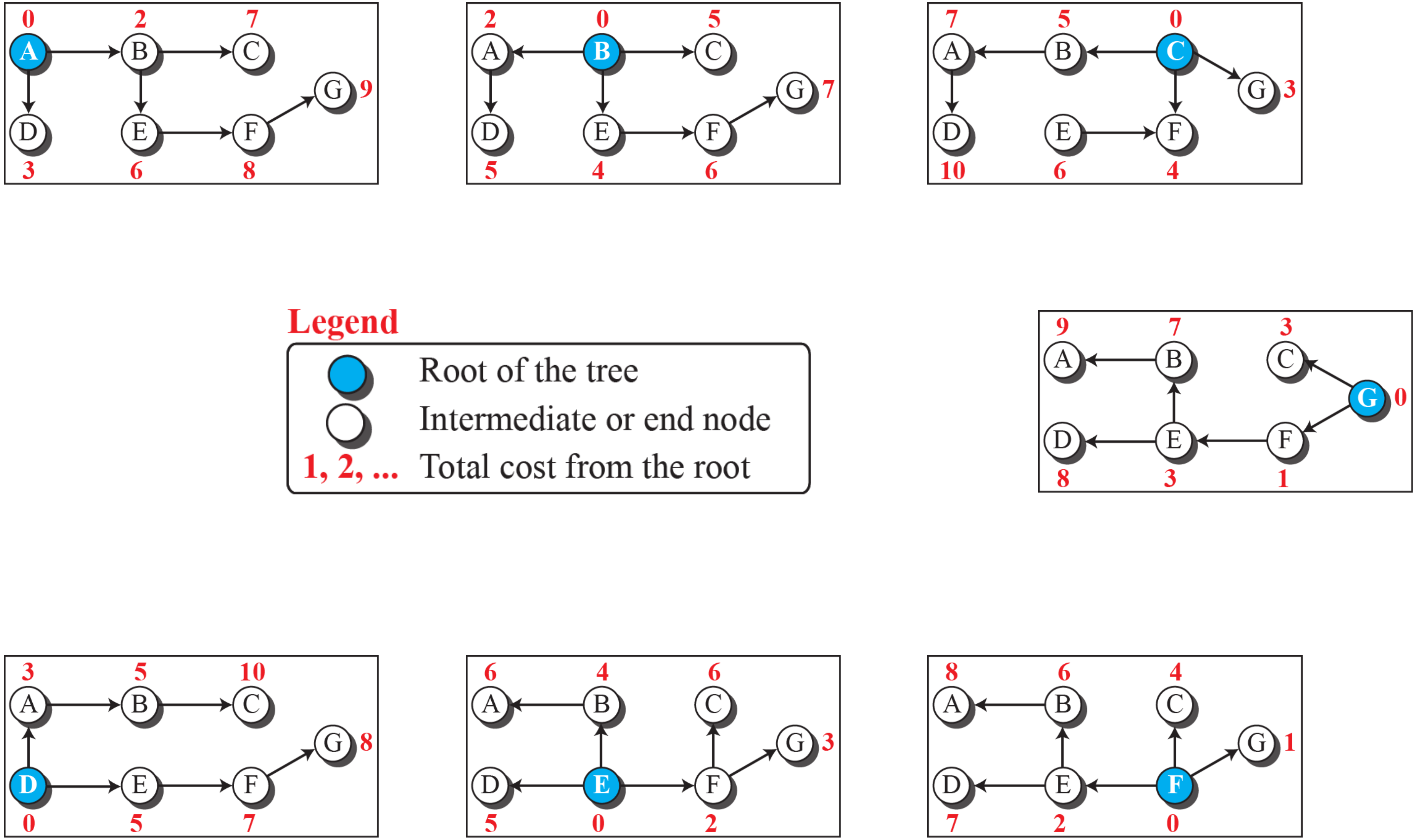
An internet and its graphical representation



a. An internet

b. The weighted graph

Least-cost trees for nodes in the internet of Figure



Legend

● Root of the tree

○ Intermediate or end node

1, 2, ... Total cost from the root

The least-cost trees for a weighted graph can have several properties if they are created using consistent criteria.

- The least-cost route from X to Y in X’s tree is the inverse of the least-cost route from Y to X in Y’s tree; the cost in both directions is the same.

- Instead of travelling from X to Z using X’s tree, we can travel from X to Y using X’s tree and continue from Y to Z using Y’s tree.

❑ Distance-Vector Routing

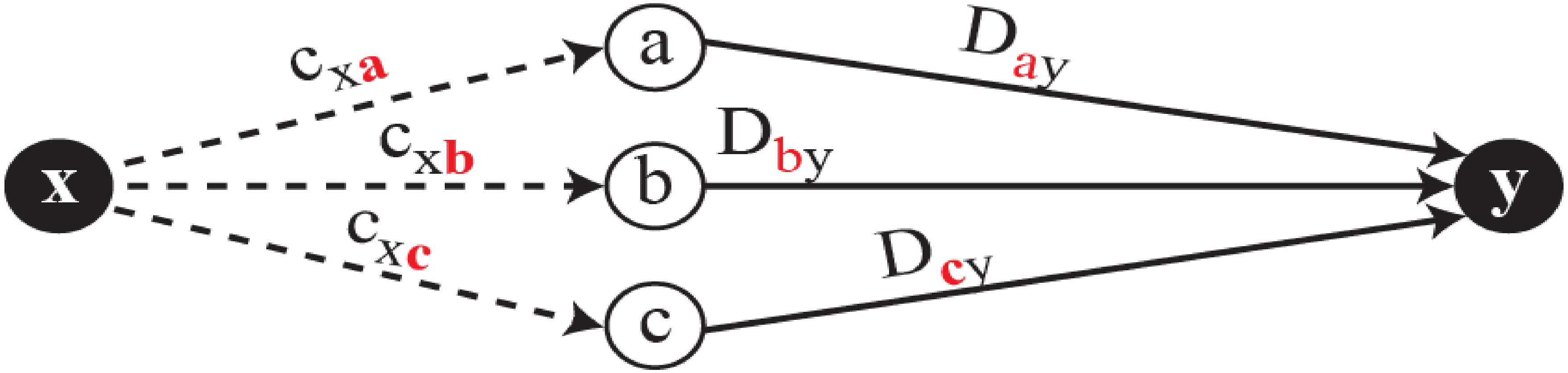
- ❖ Bellman-Ford Equation
- ❖ Distance Vectors
- ❖ Distance-Vector Routing Algorithm
- ❖ Count to Infinity
- ❖ Two-Node Loop
    - ▪ Split Horizon
    - ▪ Poisoned Reverse

- Link-State Routing
  - Link-State Database (LSDB)
  - Least-Cost Trees (Dijkstra’s Algorithm)
- Path-Vector Routing
  - Spanning Trees
  - Creation of Spanning Trees
  - Path-Vector Algorithm

Distance-Vector Routing

- The heart of distance-vector routing is the famous **Bellman-Ford** equation.
- A **distance vector** is the rationale for the name *distance-vector routing*, a one-dimensional array to represent the tree.
- A distance vector does not give the path to the destinations as the least-cost tree does; it gives only the least costs to the destinations.
- Each node in an internet originally creates the corresponding vector

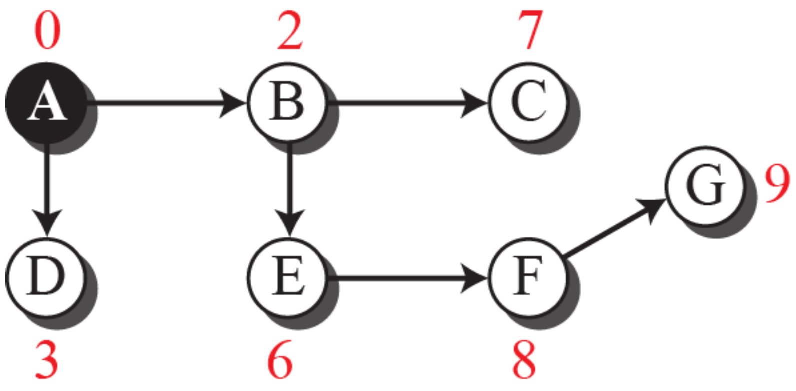
Graphical idea behind Bellman-Ford equation



a. General case with three intermediate nodes

$$Dxy = \min \{(c_{xa} + D_{ay}), (c_{xb} + D_{by}), (c_{xc} + D_{cy}), \ldots\}$$

The distance vector corresponding to a tree

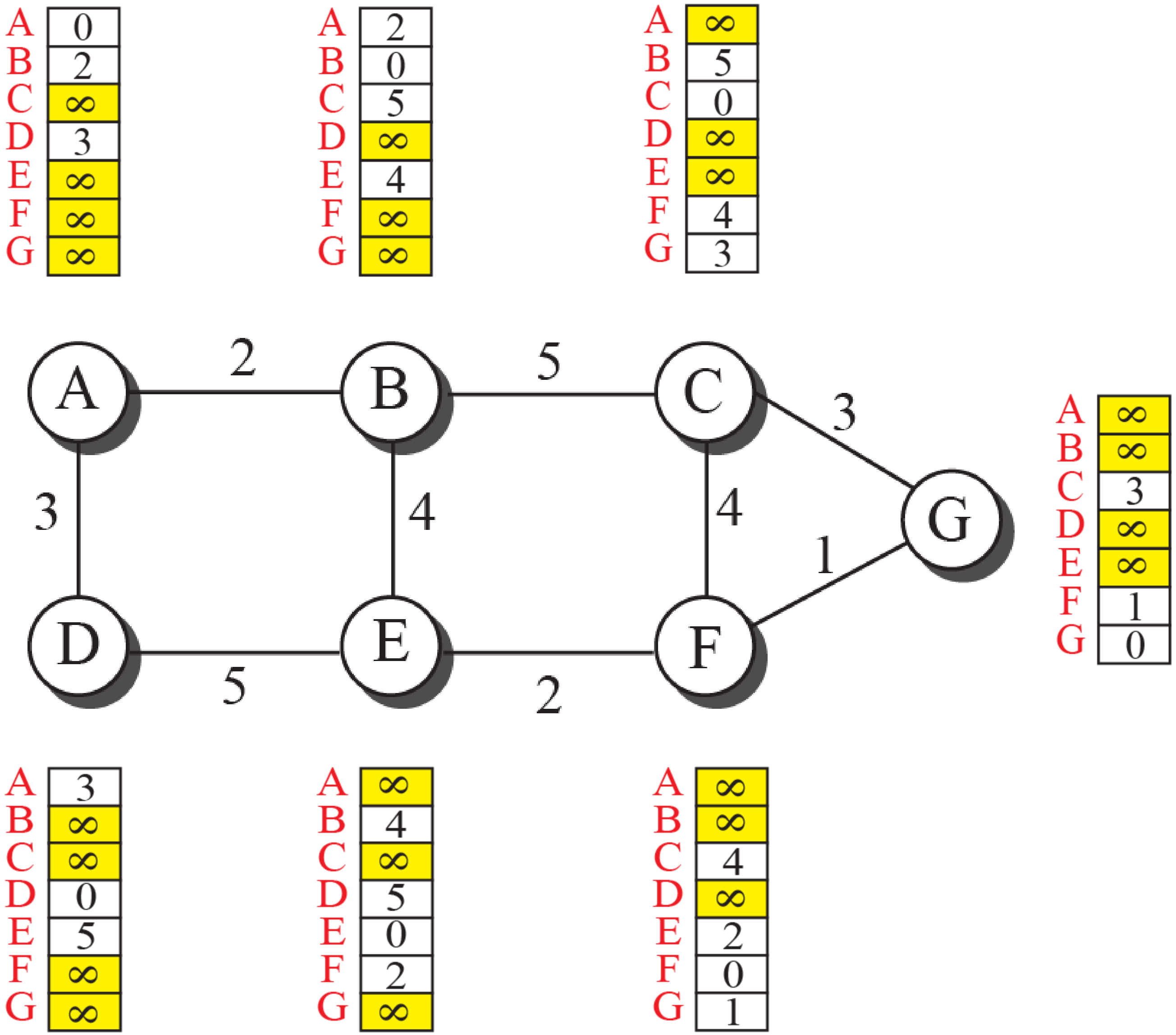


a. Tree for node A

	A
A	0
B	2
C	7
D	3
E	6
F	8
G	9

b. Distance vector for node A

Figure 4.60: *The first distance vector for an internet*



A	0
B	2
C	∞
D	3
E	∞
F	∞
G	∞

A	2
B	0
C	5
D	∞
E	4
F	∞
G	∞

A	∞
B	5
C	0
D	∞
E	∞
F	4
G	3

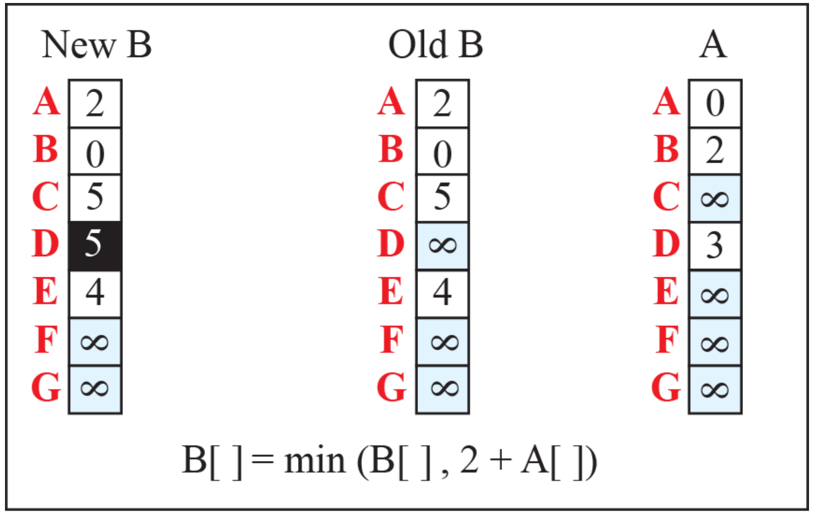
A	∞
B	∞
C	3
D	∞
E	∞
F	1
G	0

A	3
B	∞
C	∞
D	0
E	5
F	∞
G	∞

A	∞
B	4
C	∞
D	5
E	0
F	2
G	∞

A	∞
B	∞
C	4
D	∞
E	2
F	0
G	1

Figure 4.61: Updating distance vectors

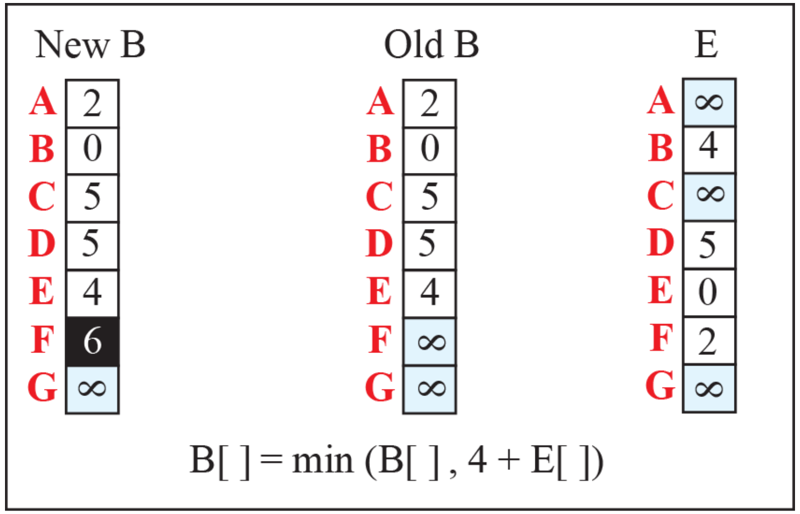


	New B	Old B	A
A	2	2	0
B	0	0	2
C	5	5	∞
D	5	∞	3
E	4	4	∞
F	∞	∞	∞
G	∞	∞	∞

B[ ] = min (B[ ] , 2 + A[ ])

a. First event: B receives a copy of A's vector.

Note:
X[ ]: the whole vector

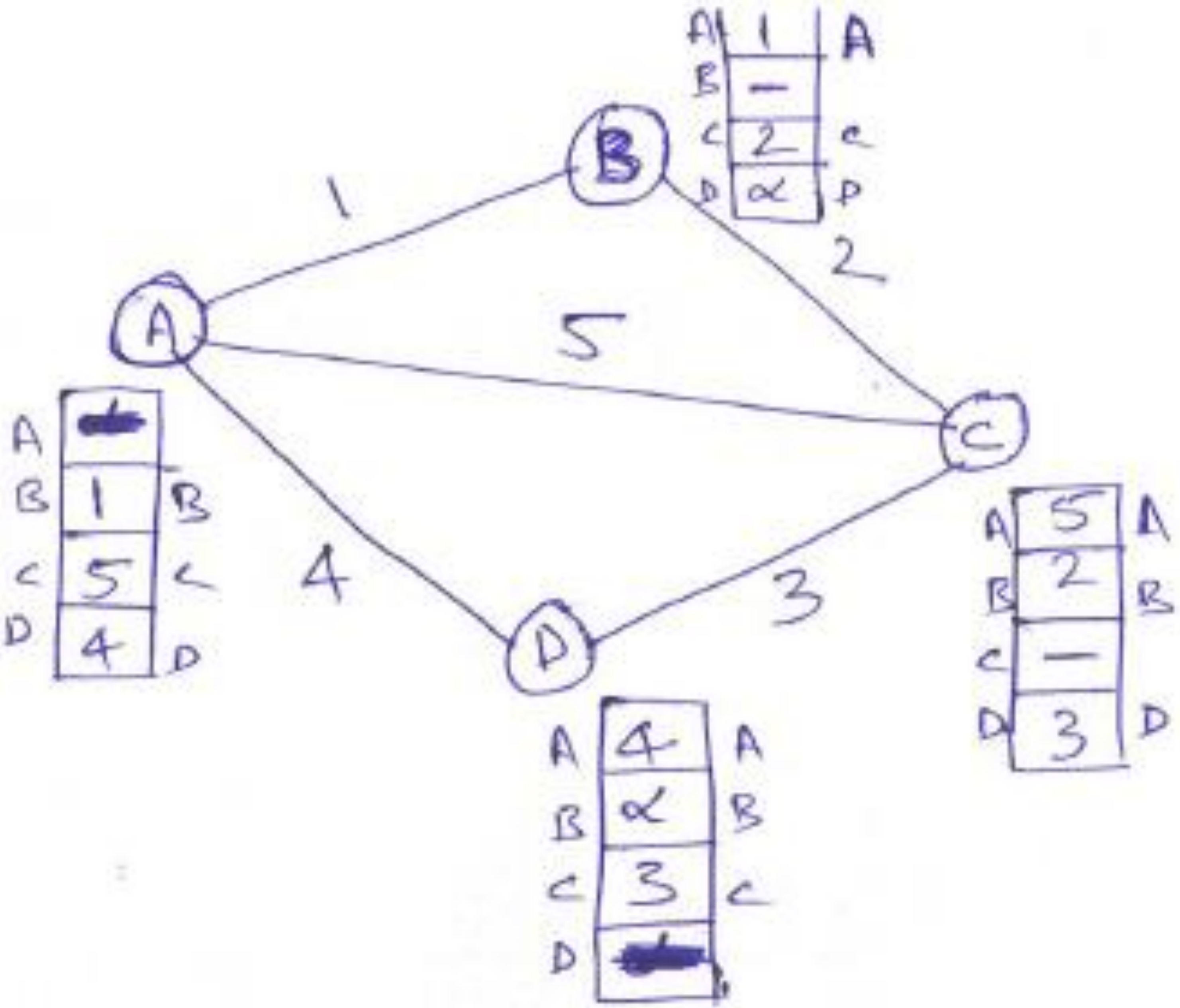


	New B	Old B	E
A	2	2	∞
B	0	0	4
C	5	5	∞
D	5	5	5
E	4	4	0
F	6	∞	2
G	∞	∞	∞

B[ ] = min (B[ ] , 4 + E[ ])

b. Second event: B receives a copy of E's vector.

Initial



Aftr Iteratn-I

At A

	New A	Next Hop	Old A	B's	C's	D's	
A	—		—	1	5	4	~~[illegible]~~ —
B	1	B	1	—	2	α	min(0+1, 2+5, α+4) = 1, B
C	3	B	5	2	—	3	min(2+1, —+5, 3+4) = 3, B
D	4	D	4	α	3	—	min(α+1, 3+5, 4+0) = 4, D
				1	5	4	

At B

	New B	Next Hop	Old B	A	C	
A	1	A	1	—	5	min(0+1, 5+2) = 1, A
B	—		—	1	2	
C	2	C	2	5	—	min(5+1, 0+2) = 2, C
D	5	A	α	4	3	min(4+1, 3+2) = 5, A
				1	2	

At C

	New C	Next hop	Old C	A	B	D
A	3	B	5	—	1	4
B	2	B	2	1	–	α
C	—		–	5	2	3
D	3	D	3	4	α	—
				5	2	3

min(5+0, 1+2, 4+3) = 3, B

min(5+1, 2+0, 3+α) = 2, B

min(4+5, α+2, —+3) = 3, D

At D

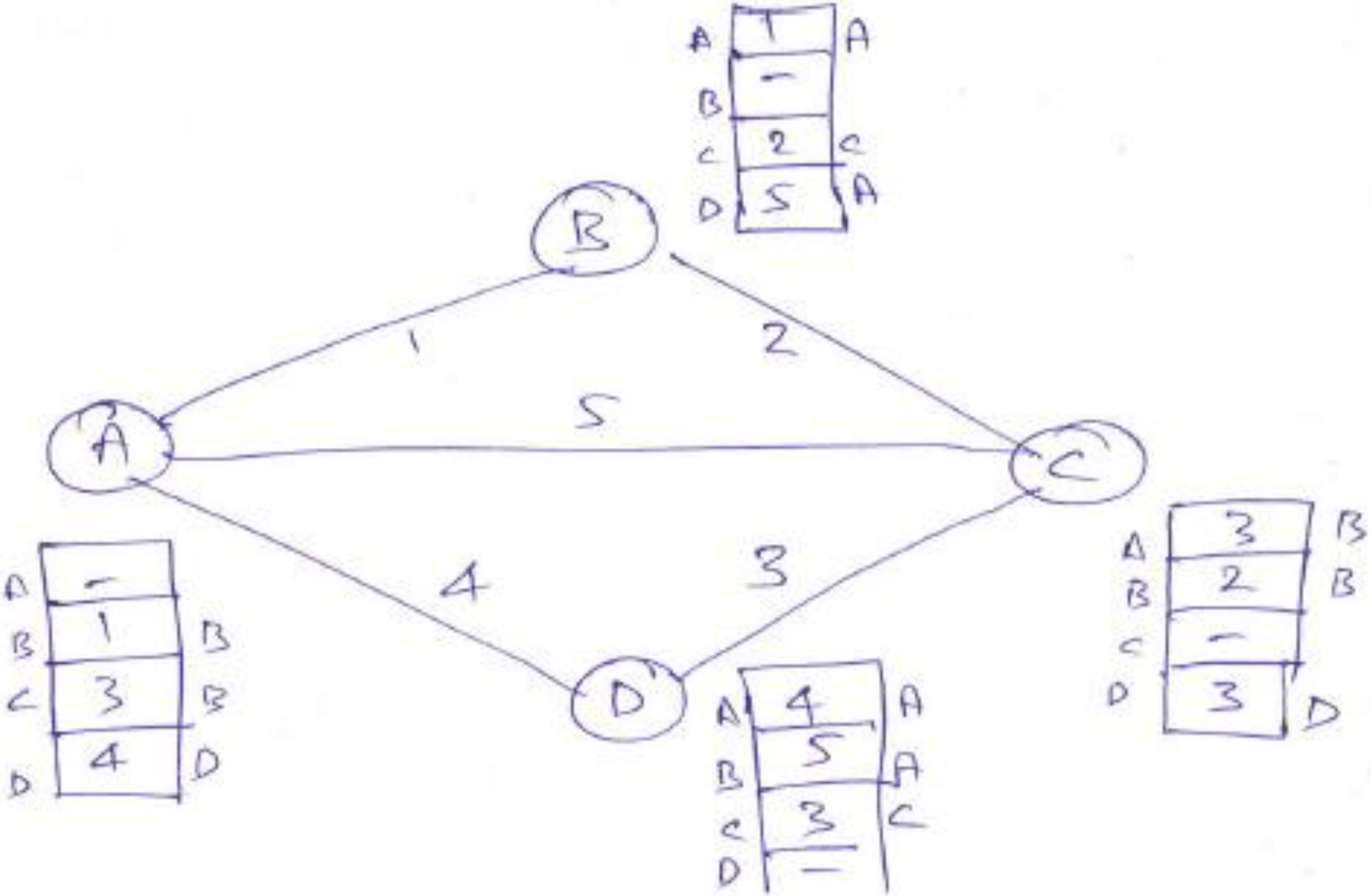
	New D	Next Hop	Old D.	A	C
A	4	A	4	—	5
B	5	A	α	1	2
C	3	C	3	5	—
D	—		—	4	3
				4	3

min(0+4, 5+3) = 4, A

min(1+4, 2+3) = 5, A

min(5+4, —+3) = 3, C

After Iteration-I



A	–	
B	1	B
C	3	B
D	4	D

A	1	A
B	–	
C	2	C
D	5	A

A	3	B
B	2	B
C	–	
D	3	D

A	4	A
B	5	A
C	3	C
D	–	

Aftr Iteration - 2

At - A

	New A	Next hop
A	—	
B	1	B
C	3	B
D	4	D

	old A	B	C	D	
A	—	1	3	4	——
B	1	—	2	5	min(—+1, 2+5, 5+4) = 1, B
C	3	2	[illegible]	3	min(2+1, —+5, 3+4) = 3, B
D	4	5	3	—	min(5+1, 3+5, —+4) = 4, D
		1	5	4	

At - B

	New B	Next hop
A	1	A
B	—	
C	2	C
D	5	A

	old B	A	C	
A	1	—	3	min(—+1, 3+2) = 1, A
B	—	1	2	——
C	2	3	—	min(3+1, —+2) = 2, C
D	5	4	3	min(4+1, 3+2) = 5, A
		1	2	

At C

	New C	Next Hop
A	3	B
B	2	B
C	—	
D	3	D

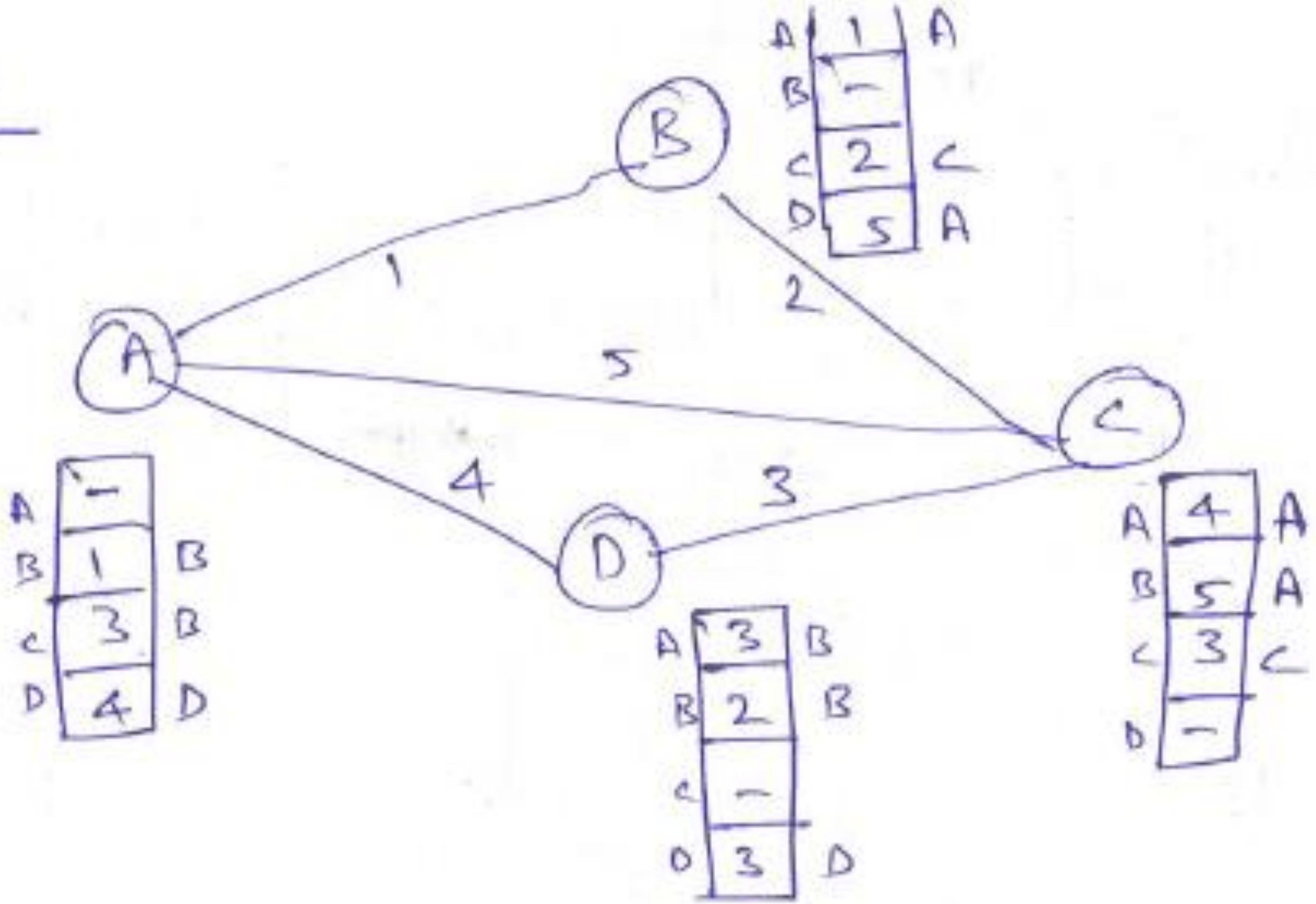
old C	A	B	D	
3	—	1	4	min(-+5, 1+2, 4+3) = 3, B
2	1	—	5	min(1+5, 0+2, 5+3) = 2, B
—	3	2	3	———
3	4	5	—	min(-+5, 5+1, -+3) = 3, D
	5	2	3	

At D

	New D	Next Hop.
A	4	A
B	5	A
C	3	C
D	—	

	old D	A	C	
A	4	—	3	min(-+4, 3+3) = 4, A
B	5	1	2	min(1+4, 2+3) = 5, A
C	3	3	—	min(3+4, -+3) = 3, C
D	—	4	3	———
		4	3	

Aftr Iteratu-2



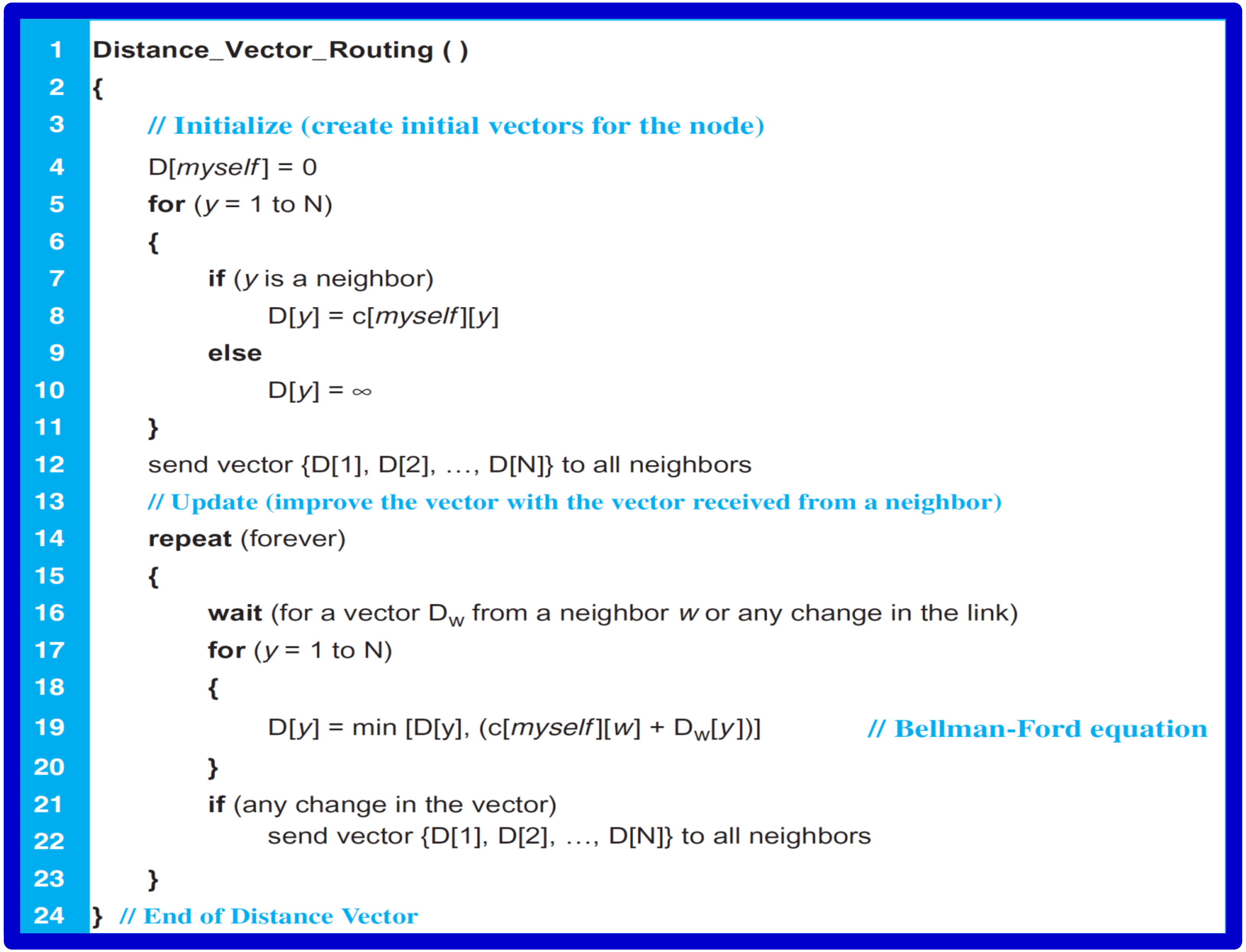
A	–	
B	1	B
c	3	B
D	4	D

A	1	A
B	–	
c	2	c
D	5	A

A	4	A
B	5	A
c	3	c
D	–	

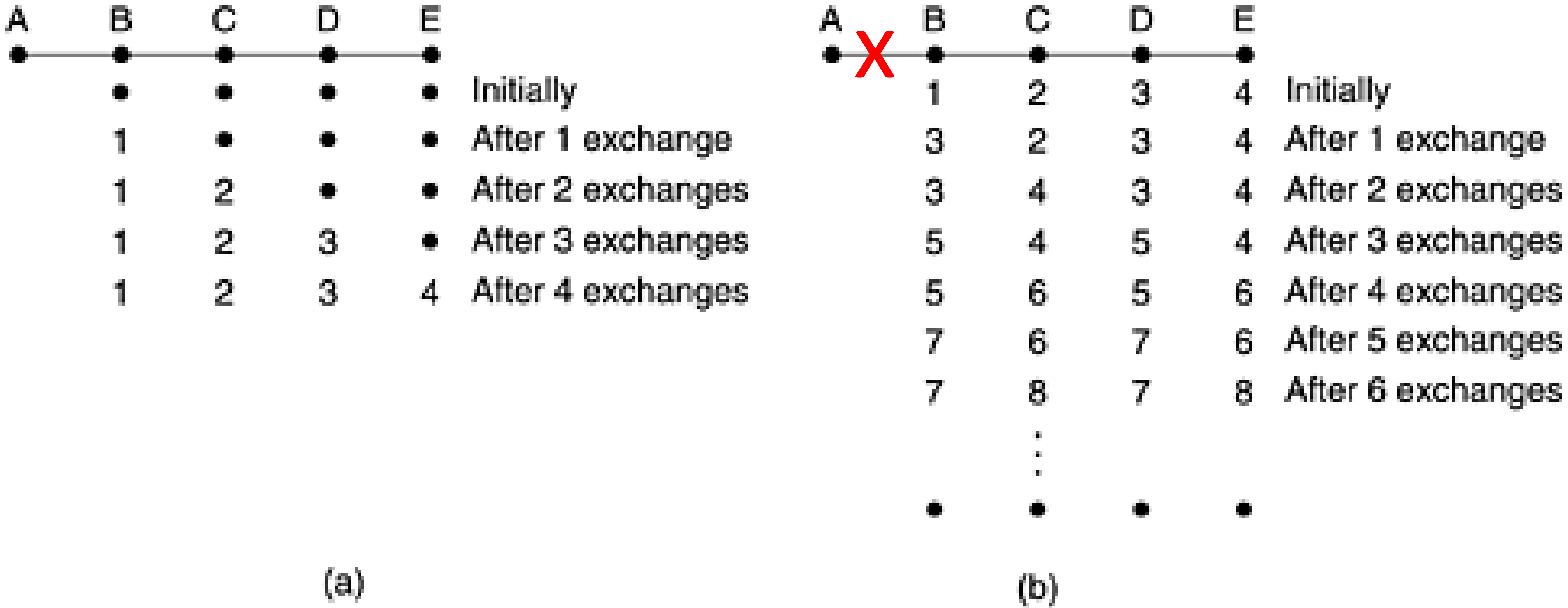
A	3	B
B	2	B
c	–	
D	3	D

Table 4.4: Distance-Vector Routing Algorithm for A Node



```
Distance_Vector_Routing ( )
{
    // Initialize (create initial vectors for the node)
    D[myself] = 0
    for (y = 1 to N)
    {
        if (y is a neighbor)
            D[y] = c[myself][y]
        else
            D[y] = ∞
    }
    send vector {D[1], D[2], …, D[N]} to all neighbors
    // Update (improve the vector with the vector received from a neighbor)
    repeat (forever)
    {
        wait (for a vector $D_w$ from a neighbor w or any change in the link)
        for (y = 1 to N)
        {
            D[y] = min [D[y], (c[myself][w] + $D_w[y]$)]        // Bellman-Ford equation
        }
        if (any change in the vector)
            send vector {D[1], D[2], …, D[N]} to all neighbors
    }
} // End of Distance Vector
```

The count-to-infinity problem.



A	B	C	D	E	
	•	•	•	•	Initially
	1	•	•	•	After 1 exchange
	1	2	•	•	After 2 exchanges
	1	2	3	•	After 3 exchanges
	1	2	3	4	After 4 exchanges

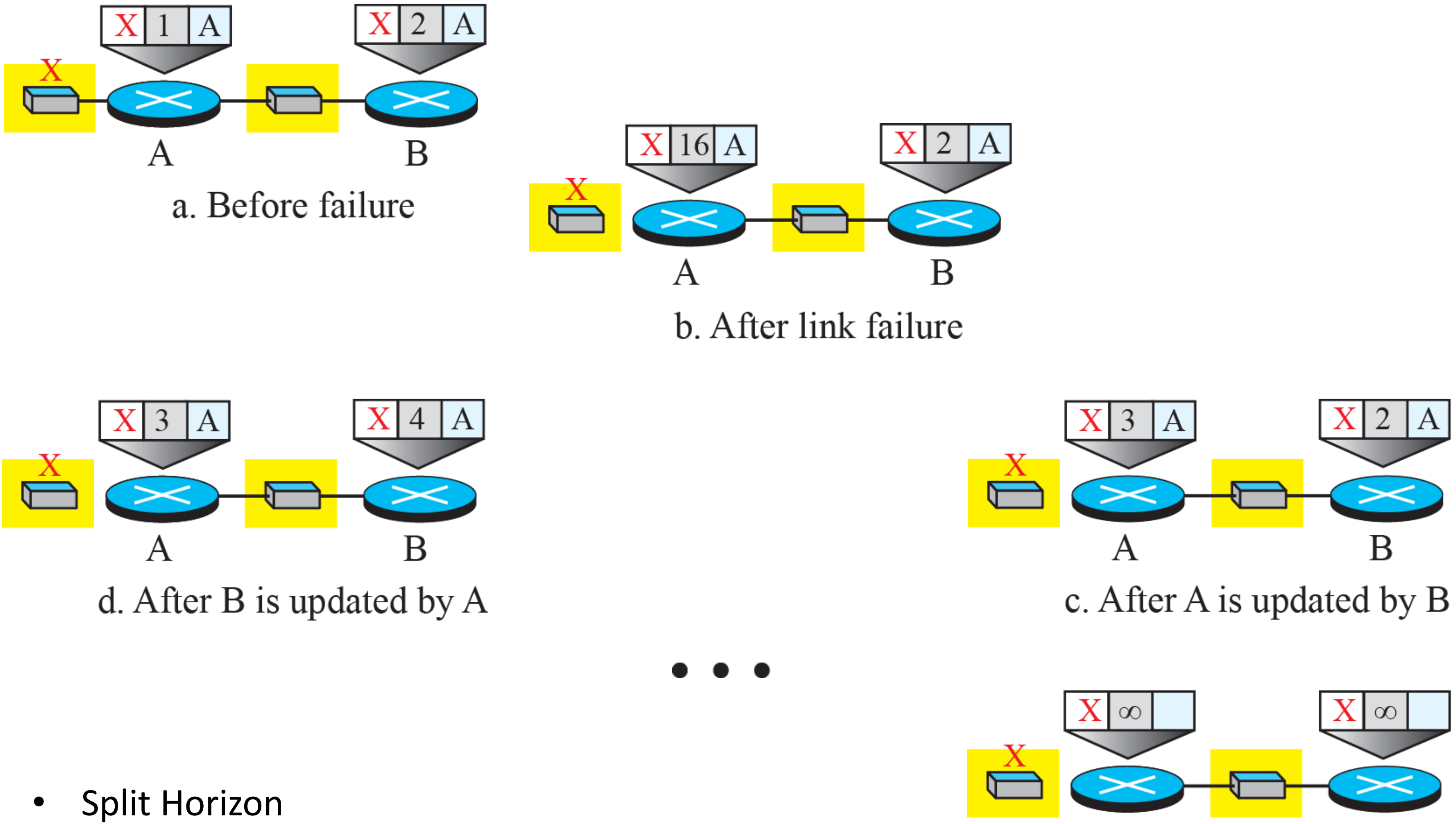
(a)

A	B	C	D	E	
	1	2	3	4	Initially
	3	2	3	4	After 1 exchange
	3	4	3	4	After 2 exchanges
	5	4	5	4	After 3 exchanges
	5	6	5	6	After 4 exchanges
	7	6	7	6	After 5 exchanges
	7	8	7	8	After 6 exchanges
		⋮			
	•	•	•	•	

(b)

Count to Infinity

***Ex:* Two-node instability**



a. Before failure

b. After link failure

d. After B is updated by A

c. After A is updated by B

e. Finally

- Split Horizon
- Poisoned Reverse

Link-State Routing

- A routing algorithm that directly follows our discussion for creating least-cost trees and forwarding tables is link-state (LS) routing.
- This method uses the term link-state to define the characteristic of a link (an edge) that represents a network in the internet.
- In this algorithm the cost associated with an edge defines the state of the link.
- Links with lower costs are preferred to links with higher costs; if the cost of a link is infinity, it means that the link does not exist or has been broken.
- To create a least-cost tree with this method, each node needs to have a complete map of the network, which means it needs to know the state of each link. The collection of states for all links is called the **link-state database (LSDB).**

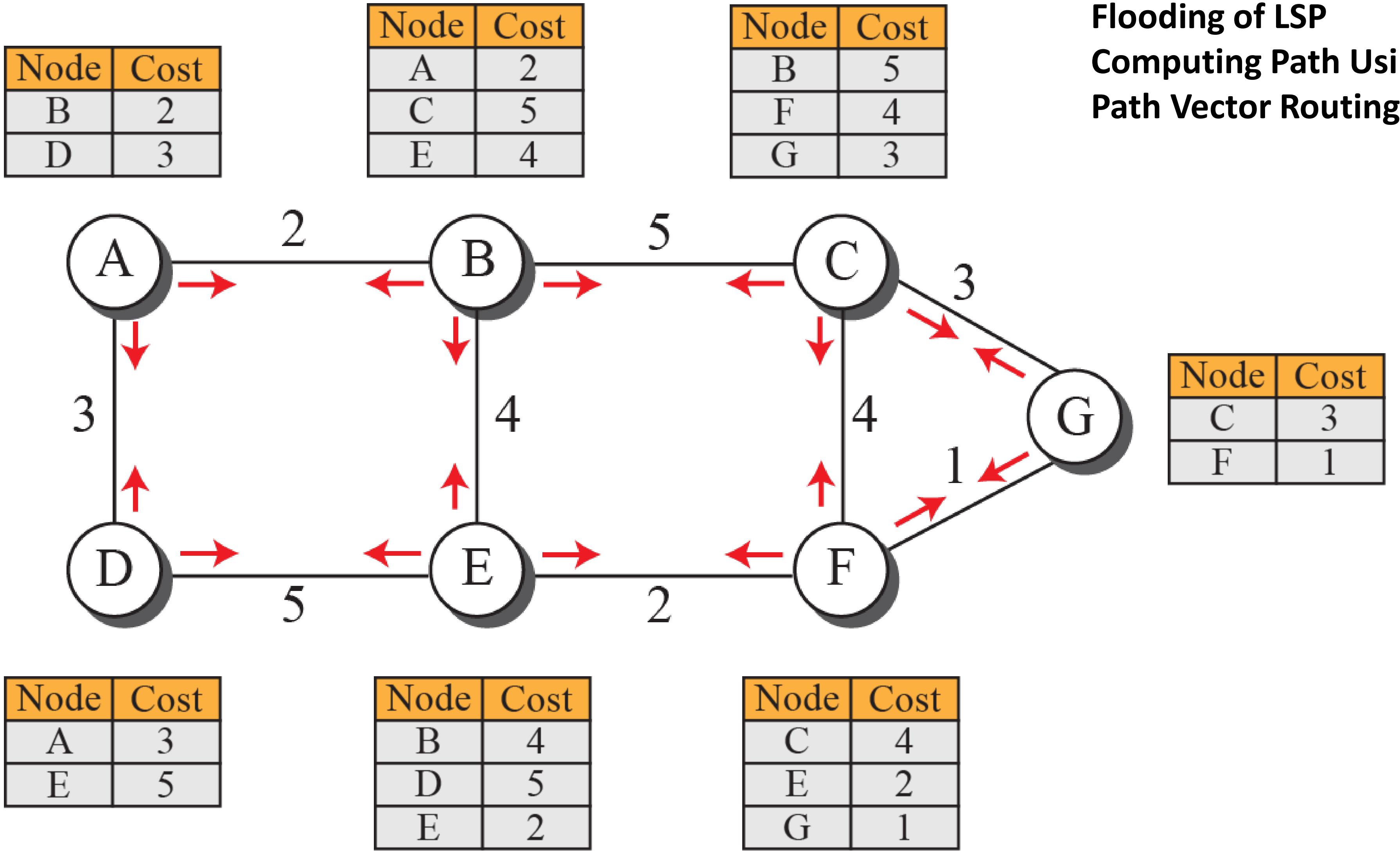
Each node can send some greeting messages to all its immediate neighbors collect two pieces of information for each neighboring node: the identity of the node and the cost of the link. The combination of these two pieces of information is called the *LS packet* (LSP); the LSP is sent out of each interface. This can be done by a process called **flooding.**

When a node receives an LSP from one of its interfaces, it compares the LSP with the copy it may already have. If the newly arrived LSP is older than the one it has (found by checking the sequence number), it discards the LSP. If it is newer or the first one received, the node discards the old LSP (if there is one) and keeps the received one. It then sends a copy of it out of each interface except the one from which the packet arrived. This guarantees that flooding stops somewhere in the network (where a node has only one interface). Once all LSP of all nodes are received at each node, each node creates LSDB.

It then calculates the spanning tree/ **least cost tree using Dijstraw's Algorithm** and LDSB.

Figure 4.64: LSPs created and sent out by each node to build LSDB

Flooding of LSP
Computing Path Using Dijstraws Alg.
Path Vector Routing alg



Node	Cost
B	2
D	3

Node	Cost
A	2
C	5
E	4

Node	Cost
B	5
F	4
G	3

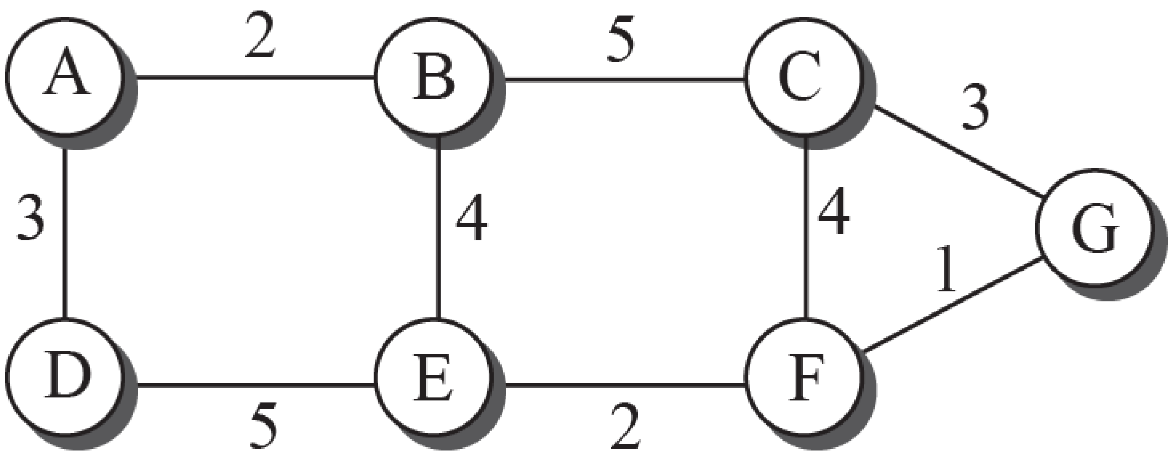
Node	Cost
C	3
F	1

Node	Cost
A	3
E	5

Node	Cost
B	4
D	5
E	2

Node	Cost
C	4
E	2
G	1

Figure 4.63: Example of a link-state database

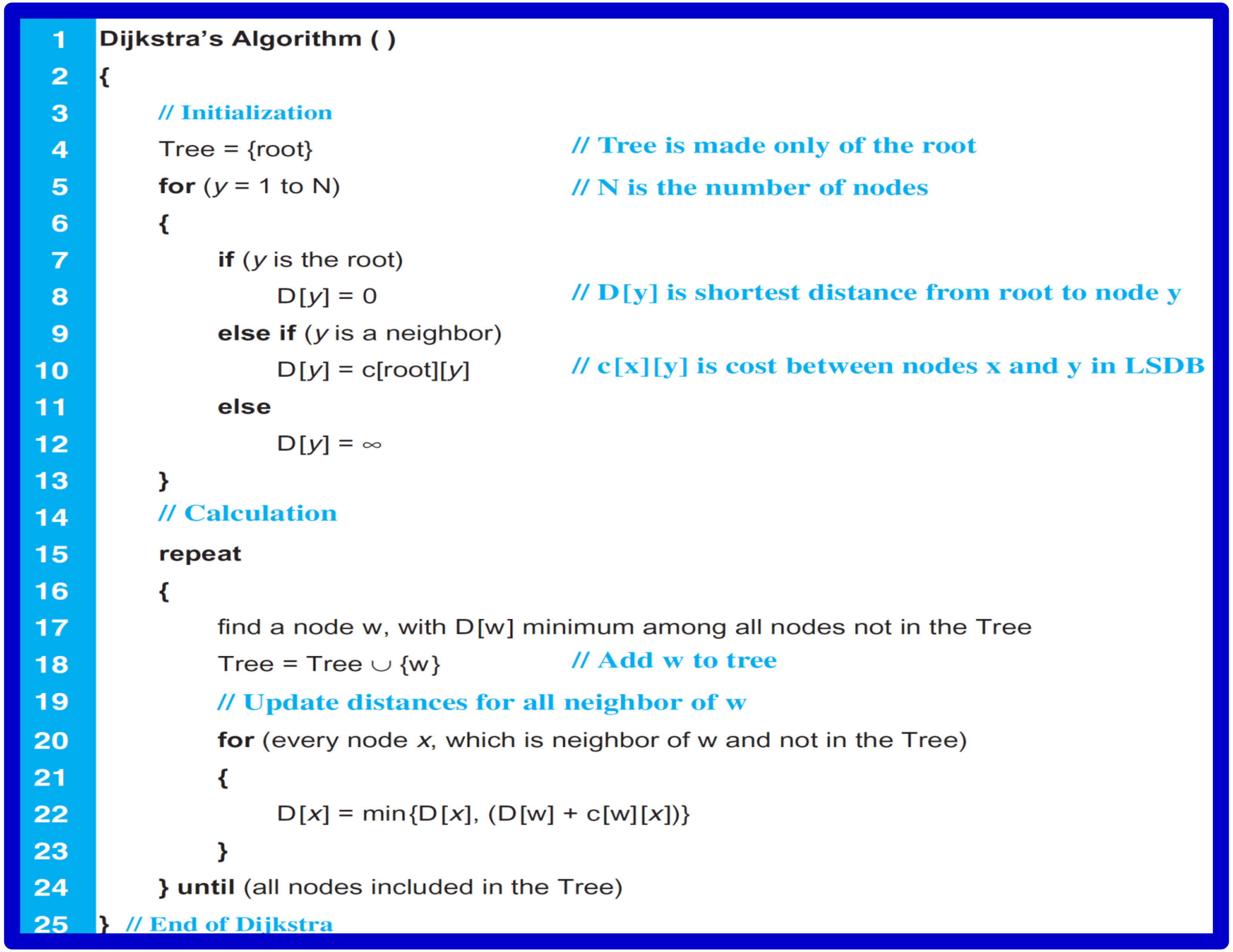


a. The weighted graph

	A	B	C	D	E	F	G
A	0	2	∞	3	∞	∞	∞
B	2	0	5	∞	4	∞	∞
C	∞	5	0	∞	∞	4	3
D	3	∞	∞	0	5	∞	∞
E	∞	4	∞	5	0	2	∞
F	∞	∞	4	∞	2	0	1
G	∞	∞	3	∞	∞	1	0

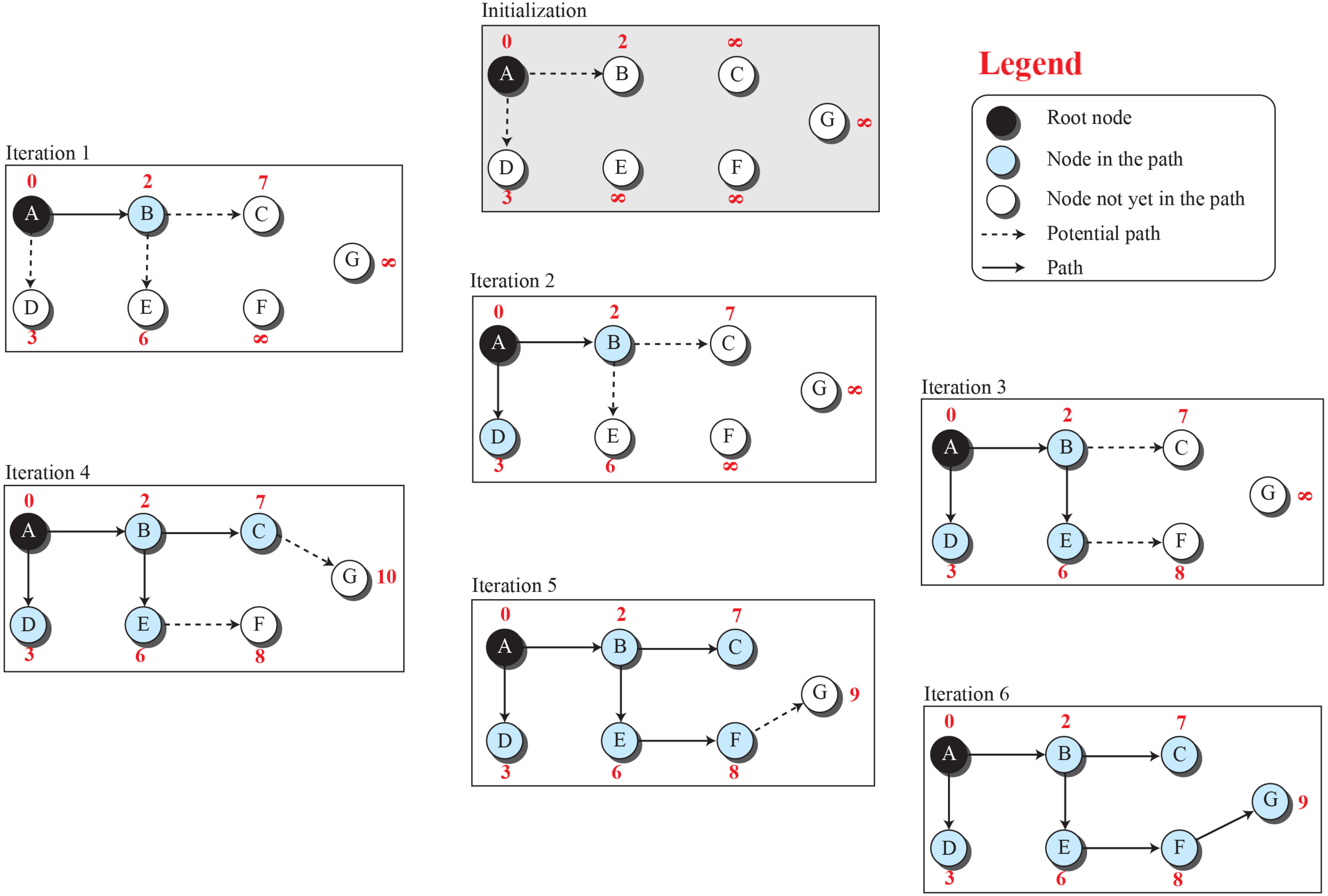
b. Link state database

Table 4.5: Dijkstra's Algorithm



```
Dijkstra's Algorithm ( )
{
    // Initialization
    Tree = {root}                      // Tree is made only of the root
    for (y = 1 to N)                   // N is the number of nodes
    {
        if (y is the root)
            D[y] = 0                   // D[y] is shortest distance from root to node y
        else if (y is a neighbor)
            D[y] = c[root][y]          // c[x][y] is cost between nodes x and y in LSDB
        else
            D[y] = ∞
    }
    // Calculation
    repeat
    {
        find a node w, with D[w] minimum among all nodes not in the Tree
        Tree = Tree ∪ {w}              // Add w to tree
        // Update distances for all neighbor of w
        for (every node x, which is neighbor of w and not in the Tree)
        {
            D[x] = min{D[x], (D[w] + c[w][x])}
        }
    } until (all nodes included in the Tree)
} // End of Dijkstra
```

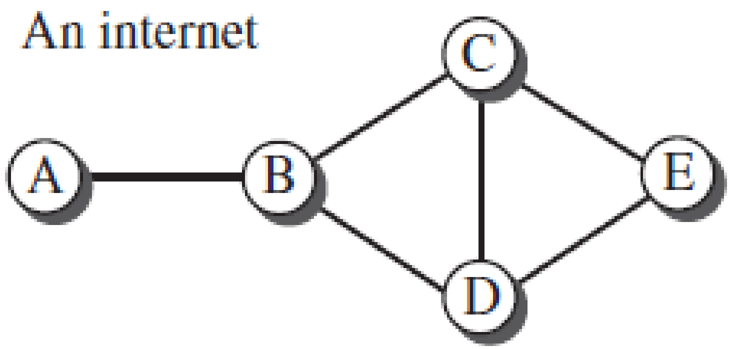
Figure 4.65: ***Least-cost tree***

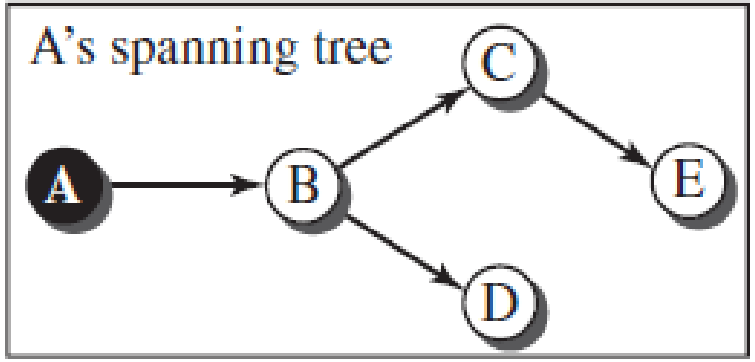


Path-Vector Routing

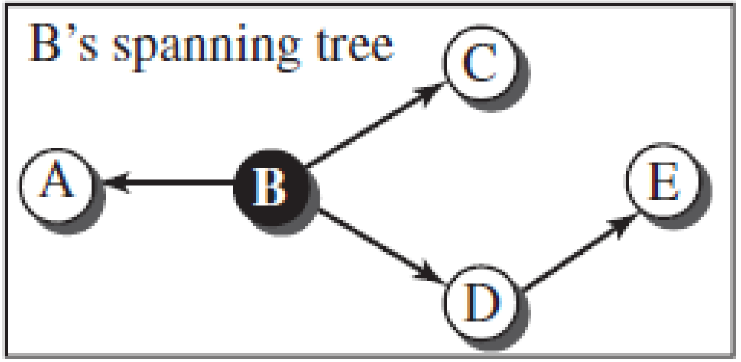
- Both link-state and distance-vector routing are based on the least-cost goal. However, there are instances where this goal is not the priority.
- The least-cost goal, applied by LS or DV routing, does not allow a sender to apply specific policies to the route a packet may take
- **Path-Vector (PV) routing** has been devised to respond to these demands. Ex: Security reasons, commercial viability etc..
- Path-vector routing does not have the drawbacks of LS or DV. The best route is determined by the source using the policy it imposes on the route.

- In path-vector routing, the path from a source to all destinations is also determined by the *best* spanning tree. The best spanning tree, however, is not the least-cost tree; it is the tree determined by the source when it imposes its own policy.



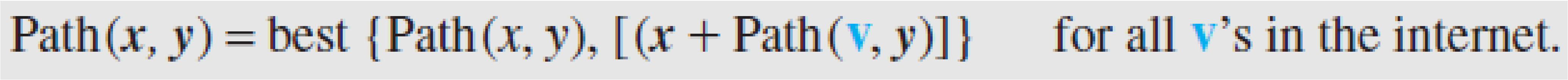


The spanning tree selected by A is such that the communication uses the minimum number of nodes to be visited but not delay.



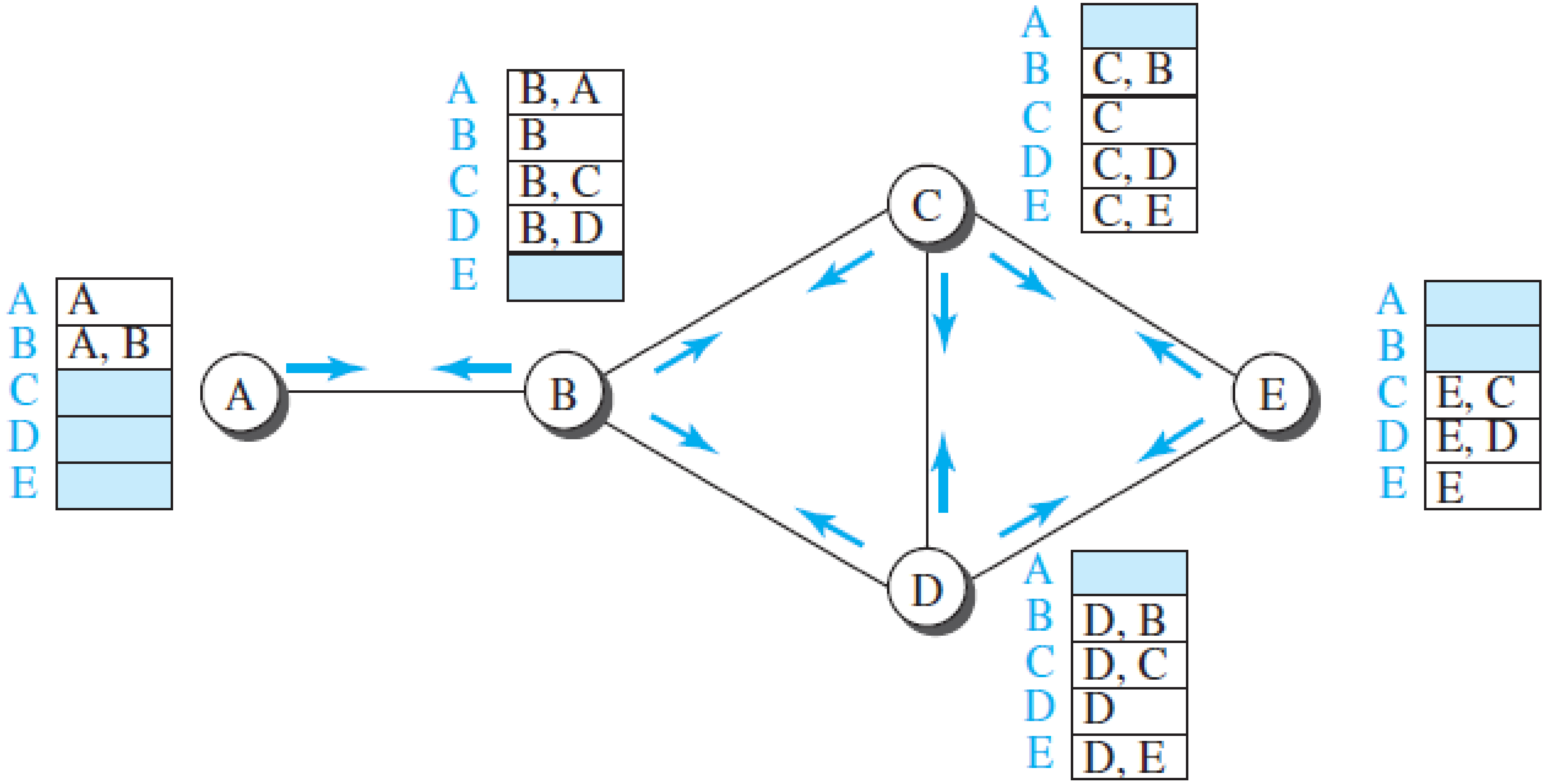
The spanning tree selected by B is such that the communication does not pass through C as a middle node.

- Path-vector routing, like distance-vector routing, is an asynchronous and distributed routing algorithm. The spanning trees are made, gradually and asynchronously, by each node.
- When a node is booted, it creates a *path vector* based on the information it can obtain about its immediate neighbor.
- Each node, after the creation of the initial path vector, sends it to all its immediate neighbors. Each node, when it receives a path vector from a neighbor, updates its path vector using an equation similar to the Bellman-Ford, but applying its own policy instead of looking for the least cost.



$$\text{Path}(x, y) = \text{best}\ \{\text{Path}(x, y), [(x + \text{Path}(\text{v}, y)]\} \qquad \text{for all v's in the internet.}$$

Path vectors made at booting time



A	A
B	A, B
C	
D	
E	

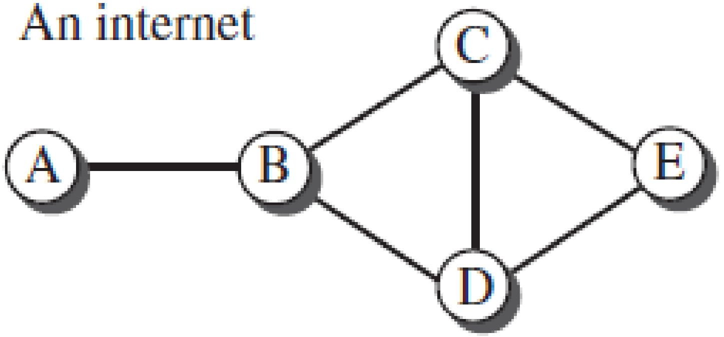
A	B, A
B	B
C	B, C
D	B, D
E	

A	
B	C, B
C	C
D	C, D
E	C, E

A	
B	D, B
C	D, C
D	D
E	D, E

A	
B	
C	E, C
D	E, D
E	E

Updating path vectors


An internet

Note:
X [ ]: vector X
Y: node Y

	New C	Old C	B
A	C, B, A		B, A
B	C, B	C, B	B
C	C	C	B, C
D	C, D	C, D	B, D
E	C, E	C, E	

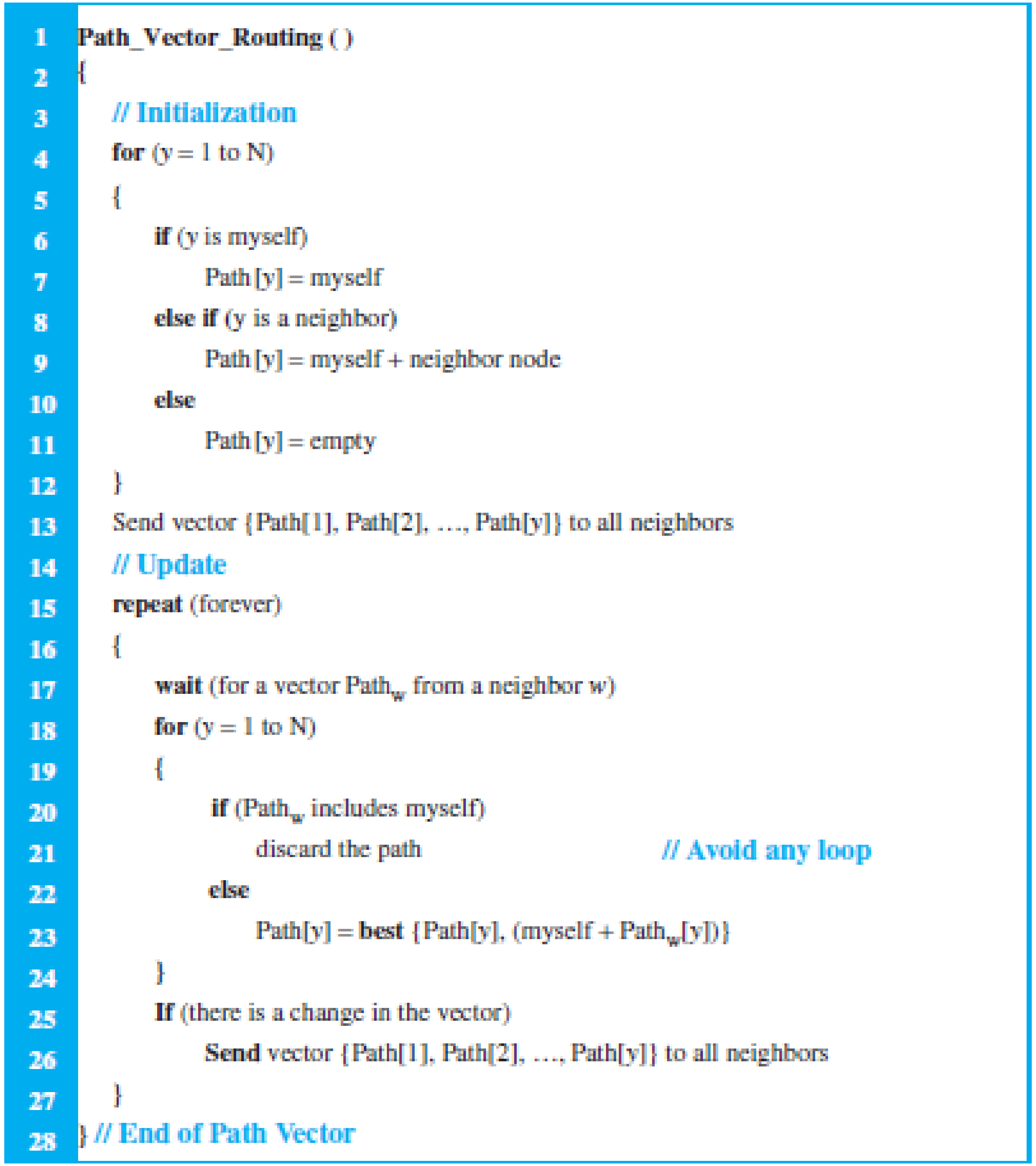
C[ ] = best (C[ ], C + B[ ])

Event 1: C receives a copy of B's vector

	New C	Old C	D
A	C, B, A	C, B, A	
B	C, B	C, B	D, B
C	C	C	D, C
D	C, D	C, D	D
E	C, E	C, E	D, E

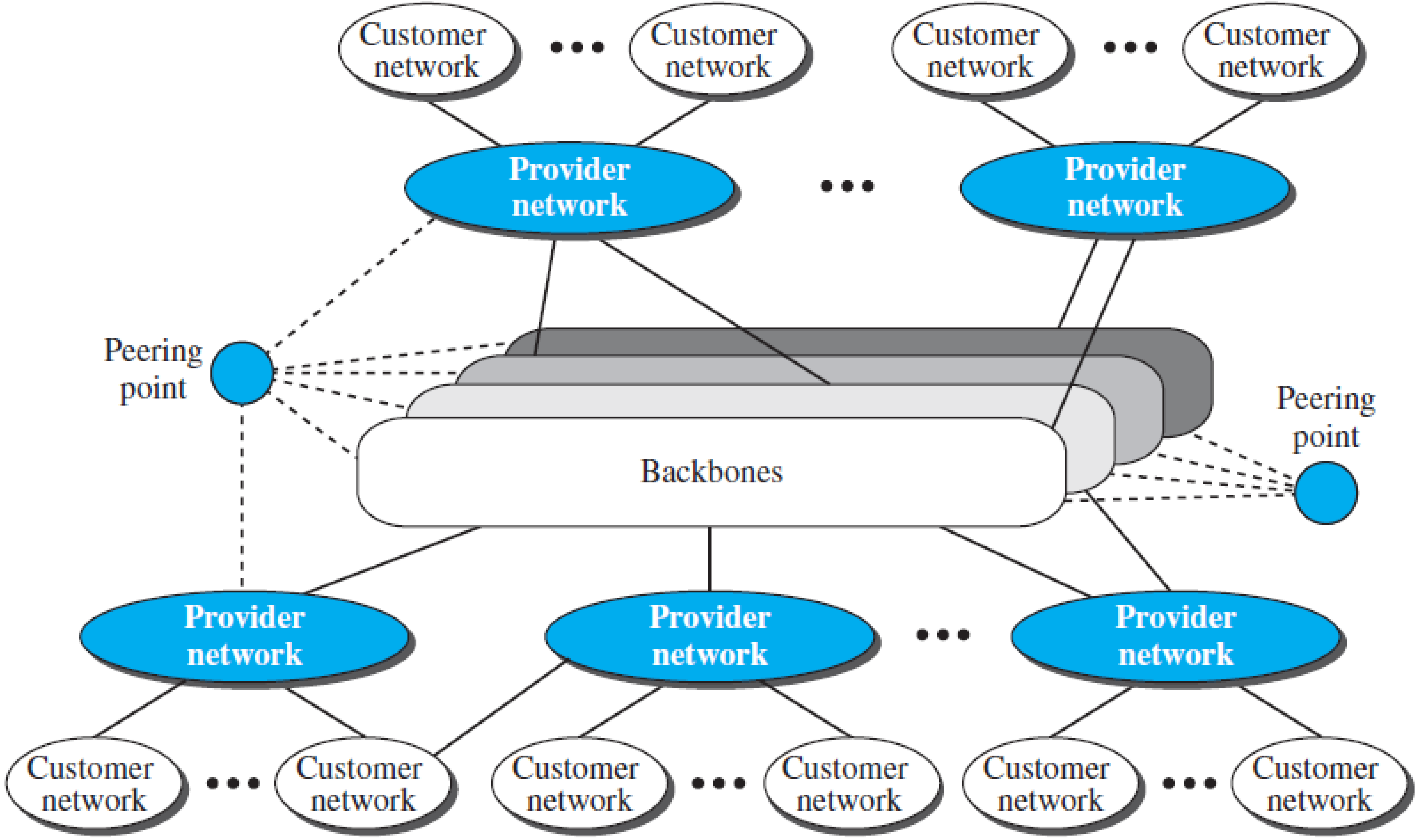
C[ ] = best (C[ ], C + D[ ])

Event 2: C receives a copy of D's vector



```
Path_Vector_Routing ( )
{
    // Initialization
    for (y = 1 to N)
    {
        if (y is myself)
            Path [y] = myself
        else if (y is a neighbor)
            Path [y] = myself + neighbor node
        else
            Path [y] = empty
    }
    Send vector {Path[1], Path[2], …, Path[y]} to all neighbors
    // Update
    repeat (forever)
    {
        wait (for a vector $\text{Path}_w$ from a neighbor $w$)
        for (y = 1 to N)
        {
            if ($\text{Path}_w$ includes myself)
                discard the path                        // Avoid any loop
            else
                Path[y] = best {Path[y], (myself + $\text{Path}_w$[y])}
        }
        If (there is a change in the vector)
            Send vector {Path[1], Path[2], …, Path[y]} to all neighbors
    }
} // End of Path Vector
```

Internet structure



- The Internet has changed from a tree-like structure, with a single backbone, to a multi-backbone structure run by different private corporations today.
- There are several *backbones* run by private communication companies that provide
- global connectivity. These backbones are connected by some *peering points* that allow connectivity between backbones.
- At a lower level, there are some *provider networks* that use the backbones for global connectivity but provide services to Internet customers.
- Finally, there are some customer *networks* that use the services provided by the provider networks.
- Any of these three entities (backbone, provider network, or customer network) can be called an Internet Service Provider or ISP. They provide services, but at different levels.

Hierarchical Routing

- It is obvious that routing in the Internet cannot be done using a single protocol for two reasons:
  - Scalability problem and an administrative issue. *Scalability problem* means that the size of the forwarding tables becomes huge, searching for a destination in a forwarding table becomes time-consuming, and updating creates a huge amount of traffic.
  - The *administrative issue* is related to the Internet structure described, each ISP is run by an administrative authority. The administrator needs to have control in its system. The organization must be able to use as many subnets and routers as it needs, may desire that the routers be from a particular manufacturer, may wish to run a specific routing algorithm to meet the needs of the organization, and may want to impose some policy on the traffic passing through its ISP.

- Hierarchical routing means considering each ISP as an **autonomous system (AS).**
- Each AS can run a routing protocol that meets its needs, but the global Internet runs a global protocol to glue all ASs together.
- The routing protocol run in each AS is referred to as *intra-AS routing protocol*, ***intradomain routing protocol***, or *interior gateway protocol (IGP)*. The two common intradomain routing protocols are RIP and OSPF
- The global routing protocol is referred to as *inter-AS routing protocol,* ***interdomain routing protocol,*** or *exterior gateway protocol (EGP)*. The only interdomain routing protocol is BGP.

Autonomous Systems

- Each ISP is an autonomous system when it comes to managing networks and routers under its control. Although we may have small, medium-size, and large ASs.
- Each AS is given an autonomous number (ASN) by the ICANN. Each ASN is a 16-bit unsigned integer that uniquely defines an AS.
- The autonomous systems, however, are not categorized according to their size; they are categorized according to the way they are connected to other ASs.
    - Stub ASs,
    - Multihomed ASs, and
    - Transient ASs

***Stub AS*:**

A stub AS has only one connection to another AS. The data traffic can be either initiated or terminated in a stub AS; the data cannot pass through it. Ex. the customer network, which is either the source or the sink of data.

***Multihomed AS*:**

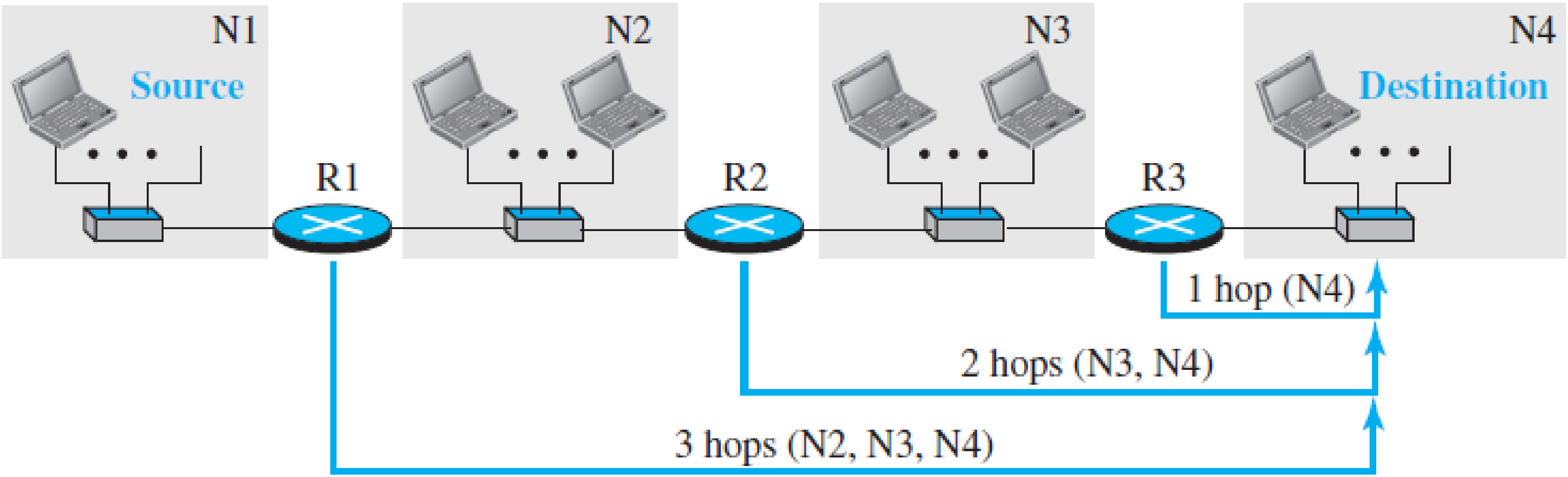
A multihomed AS can have more than one connection to other ASs, but it does not allow data traffic to pass through it. Ex. some of the customer ASs that may use the services of more than one provider network, but their policy does not allow data to be passed through them.

***Transient AS*:**

A transient AS is connected to more than one other AS and also allows the traffic to pass through. Ex. the provider networks and the backbone are good examples of transient ASs.

Routing Information Protocol (RIP)

- **Routing Information Protocol** (**RIP**) is one of the most widely used routing protocols based on the distance-vector routing algorithm.
- The DV algorithm used by RIP has been modified as described below
  - the cost is defined as the number of hops, which means the number of networks (subnets) a packet needs to travel through from the source router to the final destination host.
  - In RIP, the maximum cost of a path can be 15, which means 16 is considered as infinity.
  - As DV uses exchanging distance vectors between neighboring nodes, the routers in RIP keep forwarding tables to forward packets to their destination networks.
- A forwarding table in RIP is a three-column table in which
  - the first column is the address of the destination network,
  - the second column is the address of the next router to which the packet should be forwarded, and
  - the third column is the cost (the number of hops) to reach the destination network



Forwarding table for R1

Destination network	Next router	Cost in hops
N1	——	1
N2	——	1
N3	R2	2
N4	R2	3

Forwarding table for R2

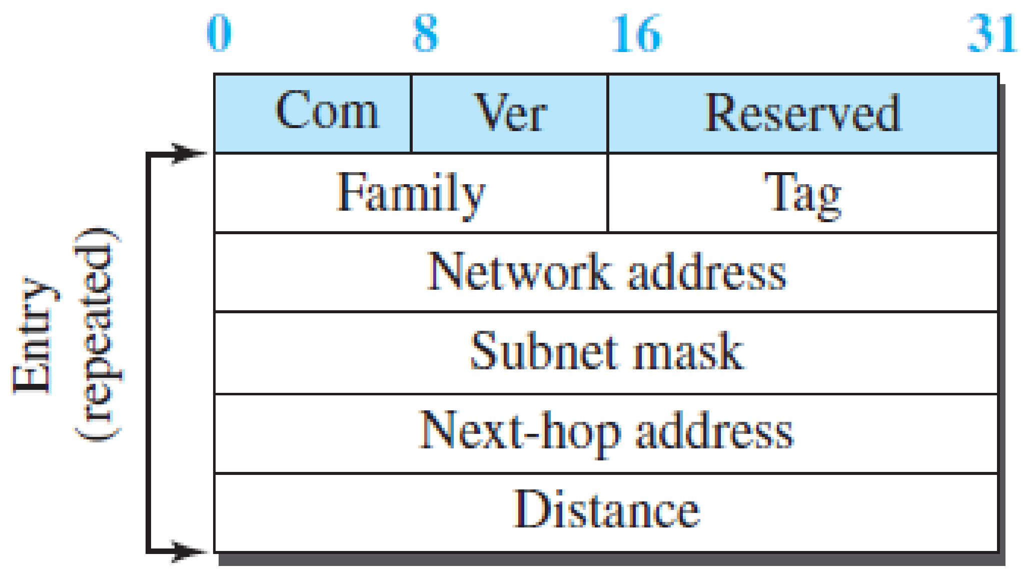
Destination network	Next router	Cost in hops
N1	R1	2
N2	——	1
N3	——	1
N4	R3	2

Forwarding table for R3

Destination network	Next router	Cost in hops
N1	R2	3
N2	R2	2
N3	——	1
N4	——	1

RIP Messages

- RIP is implemented as a process that uses the service of UDP on the well-known port number 520.
- RIP has gone through two versions: RIP-1 and RIP-2.
- Two RIP processes, a client and a server, like any other processes, need to exchange messages.
- RIP has two types of messages: request and response.
- 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 message can ask about specific entries or all entries.
- A response (or update) message can be either solicited or unsolicited.
    - A solicited response message is sent only in answer to a request message. It contains information about the destination specified in the corresponding request message.
    - An unsolicited response message, on the other hand, is sent periodically, every 30 seconds or when there is a change in the forwarding table.



Fields

Com: Command, request (1), response (2)
Ver: Version, current version is 2
Family: Family of protocol, for TCP/IP value is 2
Tag: Information about autonomous system
Network address: Destination address
Subnet mask: Prefix length
Next-hop address: Address length
Distance: Number of hops to the destination

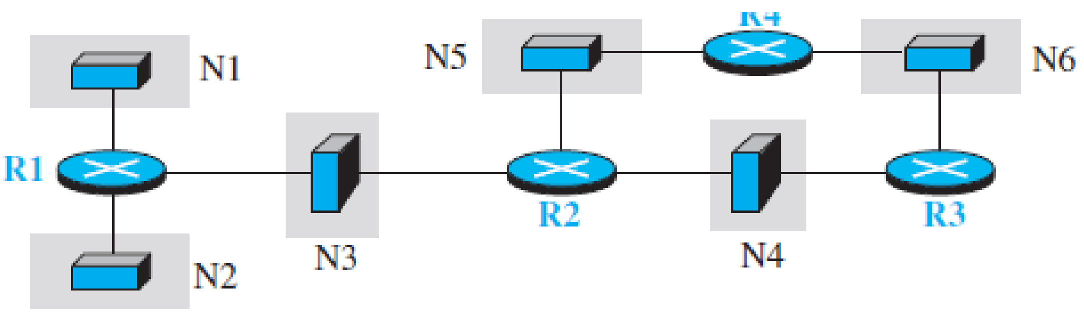
- Part of the message, which we call *entry*, can be repeated as needed in a message. Each entry carries the information related to one line in the forwarding table of the router that sends the message.

Timers in RIP

- RIP uses three timers to support its operation.
- The *periodic timer* controls the advertising of regular update messages. Each router has one periodic timer that is randomly set to a number between 25 and 35 seconds.
- The *expiration timer* governs the validity of a route. When a router receives update information for a route, the expiration timer is set to 180 seconds for that particular route, the route is considered expired and the hop count of the route is set to 16, which means the destination is unreachable.
- The *garbage collection timer* is used to purge a route from the forwarding table. When the information about a route becomes invalid, the router does not immediately purge that route from its table.

Performance of RIP:

- ***Update Messages.*** The update messages in RIP have a very simple format and are sent only to neighbors; they are local.
- ***Convergence of Forwarding Tables*****.** RIP uses the distance-vector algorithm, which can converge slowly if the domain is large, but, since RIP allows only 15 hops in a domain (16 is considered as infinity), there is normally no problem in convergence.
- ***Robustness*****.** As we said before, distance-vector routing is based on the concept that each router sends what it knows about the whole domain to its neighbors. If there is a failure or corruption in one router, the problem will be propagated to all routers and the forwarding in each router will be affected.



Legend

Des.: Destination network
N. R.: Next router
Cost: Cost in hops
← : New route
← : Old route

R1

Des.	N. R.	Cost
N1	—	1
N2	—	1
N3	—	1

R2

Des.	N. R.	Cost
N3	—	1
N4	—	1
N5	—	1

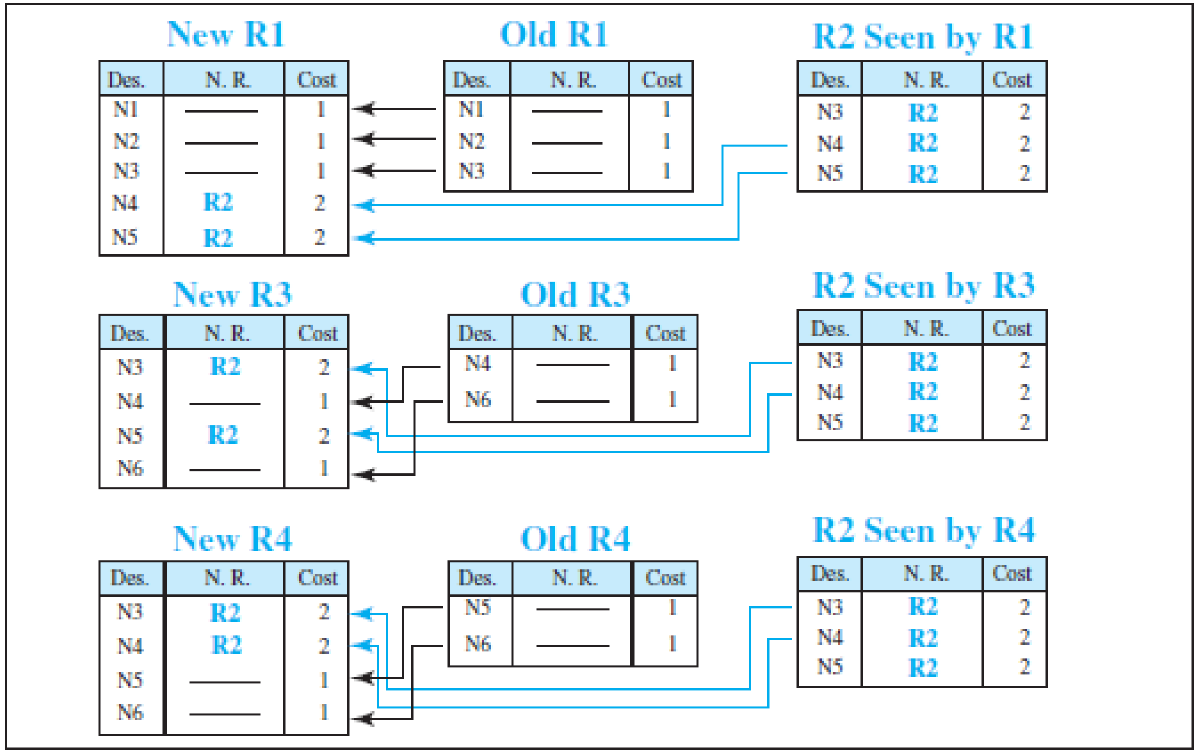
R3

Des.	N. R.	Cost
N4	—	1
N6	—	1

R4

Des.	N. R.	Cost
N5	—	1
N6	—	1

Forwarding tables after all routers booted



New R1

Des.	N. R.	Cost
N1	—	1
N2	—	1
N3	—	1
N4	R2	2
N5	R2	2

Old R1

Des.	N. R.	Cost
N1	—	1
N2	—	1
N3	—	1

R2 Seen by R1

Des.	N. R.	Cost
N3	R2	2
N4	R2	2
N5	R2	2

New R3

Des.	N. R.	Cost
N3	R2	2
N4	—	1
N5	R2	2
N6	—	1

Old R3

Des.	N. R.	Cost
N4	—	1
N6	—	1

R2 Seen by R3

Des.	N. R.	Cost
N3	R2	2
N4	R2	2
N5	R2	2

New R4

Des.	N. R.	Cost
N3	R2	2
N4	R2	2
N5	—	1
N6	—	1

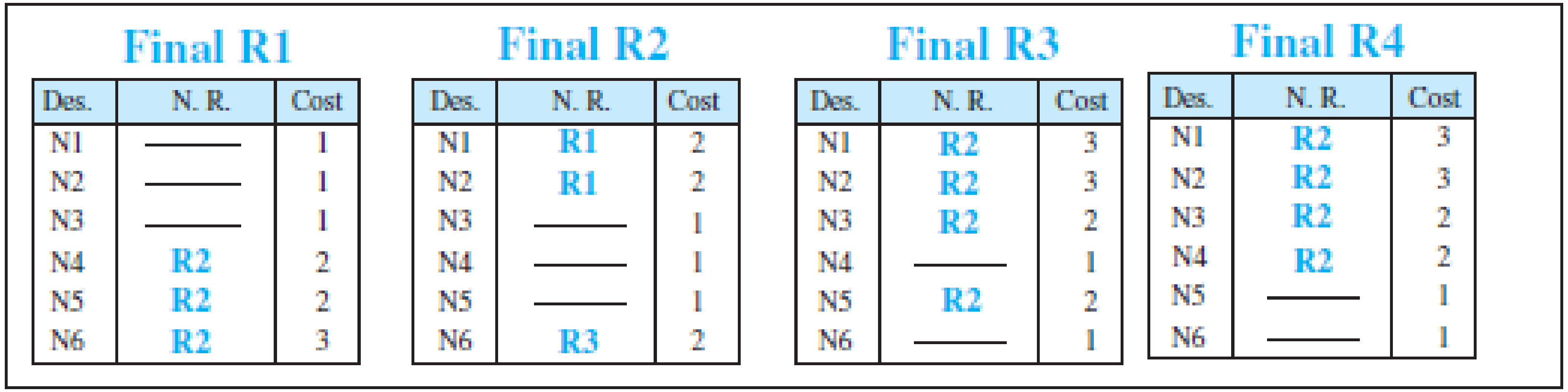
Old R4

Des.	N. R.	Cost
N5	—	1
N6	—	1

R2 Seen by R4

Des.	N. R.	Cost
N3	R2	2
N4	R2	2
N5	R2	2

Changes in the forwarding tables of R1, R3, and R4 after they receive a copy of R2's table



Final R1

Des.	N. R.	Cost
N1	———	1
N2	———	1
N3	———	1
N4	R2	2
N5	R2	2
N6	R2	3

Final R2

Des.	N. R.	Cost
N1	R1	2
N2	R1	2
N3	———	1
N4	———	1
N5	———	1
N6	R3	2

Final R3

Des.	N. R.	Cost
N1	R2	3
N2	R2	3
N3	R2	2
N4	———	1
N5	R2	2
N6	———	1

Final R4

Des.	N. R.	Cost
N1	R2	3
N2	R2	3
N3	R2	2
N4	R2	2
N5	———	1
N6	———	1

Forwarding tables
for all routers
after they have
been stablized

Open Shortest Path First (OSPF)

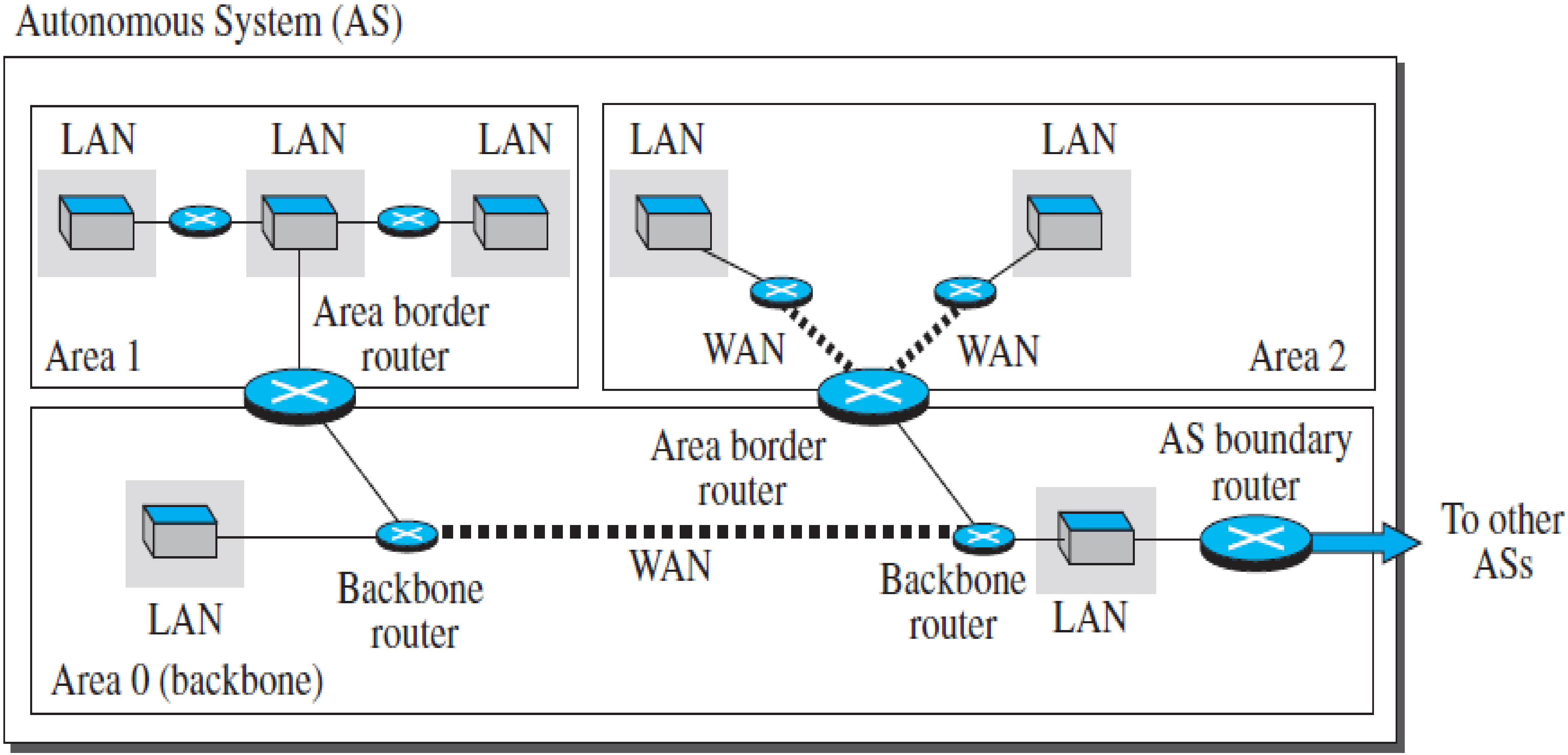
- It is also an intradomain routing protocol like RIP, but it is based on the link-state routing protocol
- Each link (network) can be assigned a weight based on the throughput, round-trip time, reliability, and so on. An administration can also decide to use the hop count as the cost. An interesting point about the cost in OSPF is that different service types (TOSs) can have different weights as the cost.

Forwarding Tables

- Each OSPF router can create a forwarding table after finding the shortest-path tree between itself and the destination using Dijkstra’s algorithm

Areas

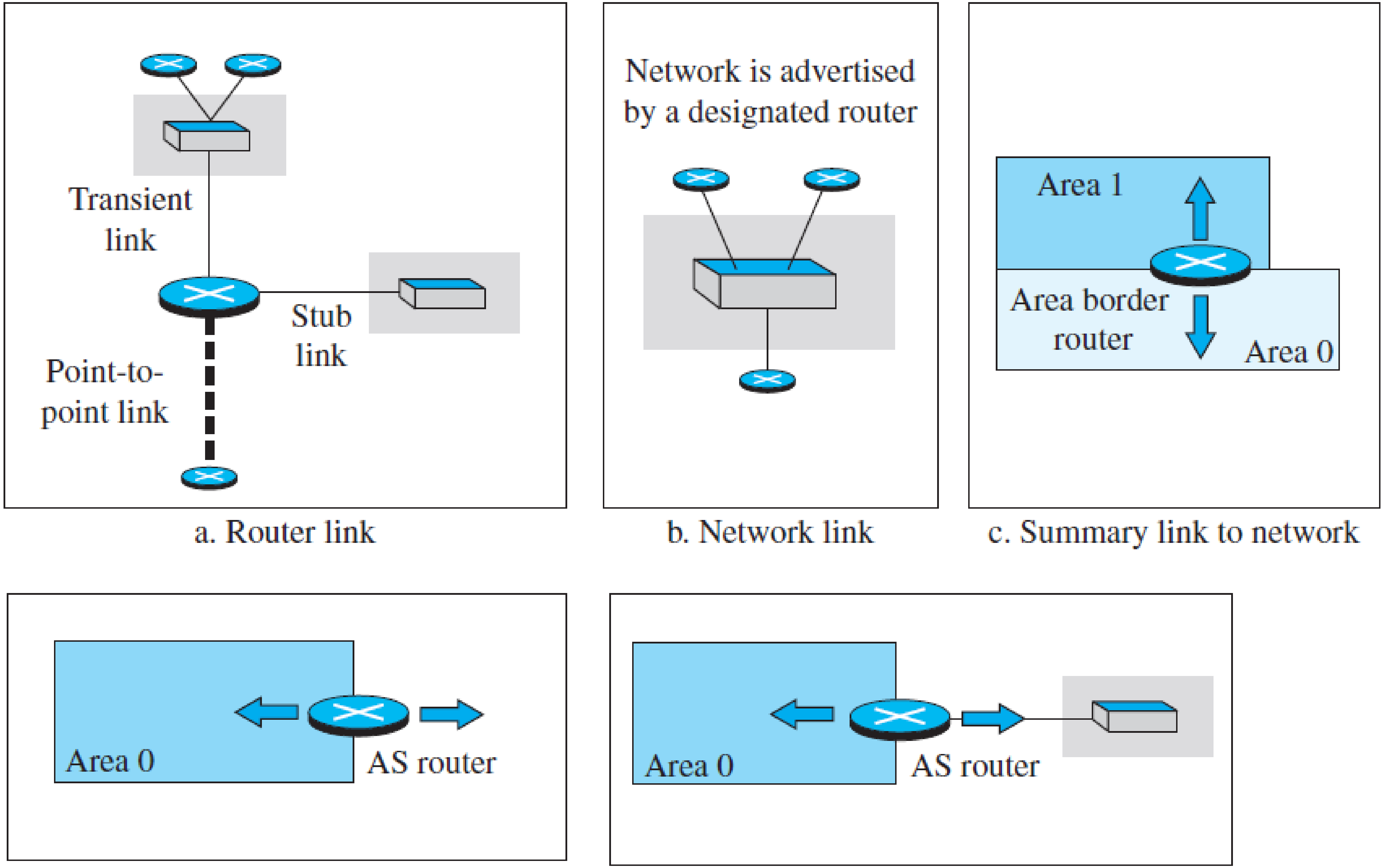
- OSPF was designed to be able to handle routing in a small or large autonomous system.
- The AS needs to be divided into small sections called *areas*. Each area acts as a small independent domain for flooding LSPs. In other words, OSPF uses another level of hierarchy in routing: the first level is the autonomous system, the second is the area.
- However, each router in an area needs to know the information about the link states not only in its area but also in other areas. For this reason, one of the areas in the AS is designated as the *backbone area,* responsible for gluing the areas together.
- The routers in the backbone area are responsible for passing the information collected by each area to all other areas. In this way, a router in an area can receive all LSPs generated in other areas. For the purpose of communication, each area has an area identification. The area identification of the backbone is zero.


Autonomous System (AS)
LAN
LAN
LAN
LAN
LAN
Area border router
Area 1
WAN
WAN
Area 2
Area border router
AS boundary router
To other ASs
LAN
Backbone router
WAN
Backbone router
LAN
Area 0 (backbone)

Link-State Advertisement

5 types of advertisements:

- ***router link****:* A router link advertises the existence of a router as a node. In addition to giving the address of the announcing router
  - *transient link:* define the address of the transient network and the cost of the link
  - *stub link* advertises a link to a stub network
  - *point-to-point link* should define the address of the router at the end of the point-to-point line and the cost to get the
- ***network link****:* A network link advertises the network as a node. However, since a network cannot do announcements itself (it is a passive entity), one of the routers is assigned as the designated router and does the advertising. In addition to the address of the designated router, this type of LSP announces the IP address of all routers, but not the cost.
- ***summary link to network****:* it advertises the summary of links collected by the backbone to an area or the summary of links collected by the area to the backbone
- ***Summary link to AS border router:*** This is done by an AS router that advertises the summary links from other ASs to the backbone area of the current AS
- ***external link****:* This is also done by an AS router to announce the existence of a single network outside the AS to the backbone area to be disseminated into the areas



a. Router link

b. Network link

c. Summary link to network

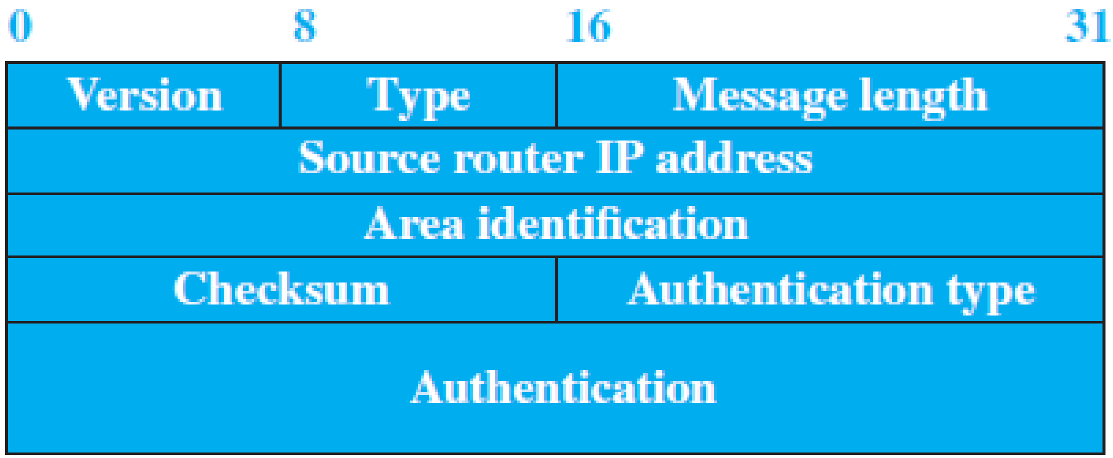
d. Summary link to AS

e. External link

OSPF Messages

It uses five different types of messages:

- The *hello* message (type 1) is used by a router to introduce itself to the neighbors and announce all neighbors that it already knows.
- The *database description* message (type 2) is normally sent in response to the hello message to allow a newly joined router to acquire the full LSDB.
- The *linkstate request* message (type 3) is sent by a router that needs information about a specific LS.
- The *link-state update* message (type 4) is the main OSPF message used for building the LSDB. This message, in fact, has five different versions (router link, network link, summary link to network, summary link to AS border router, and external link)
- The *link-state acknowledgment* message (type 5) is used to create reliability in OSPF; each router that receives a link-state update message needs to acknowledge it.



0 8 16 31

Version | Type | Message length

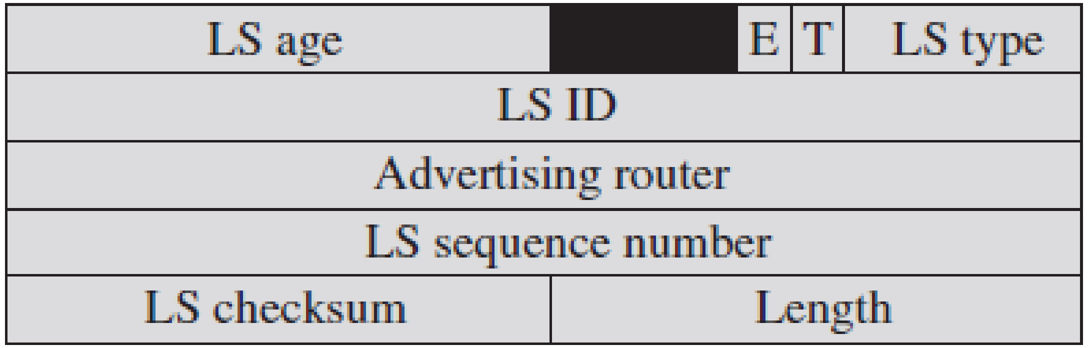
Source router IP address

Area identification

Checksum | Authentication type

Authentication

OSPF common header



LS age | E | T | LS type

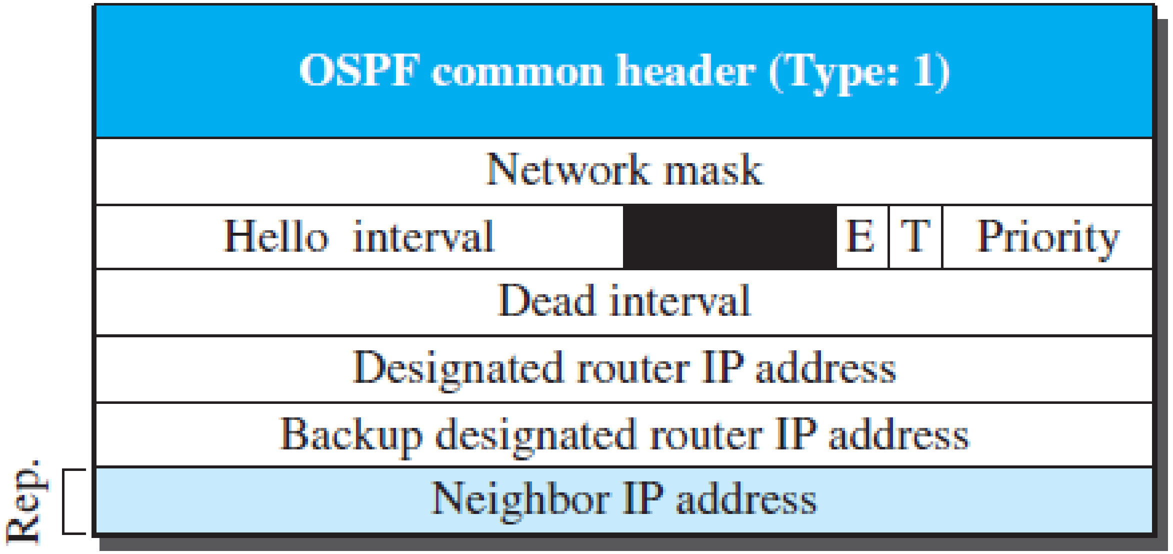
LS ID

Advertising router

LS sequence number

LS checksum | Length

Link-state general header



OSPF common header (Type: 1)

Network mask

Hello interval | E | T | Priority

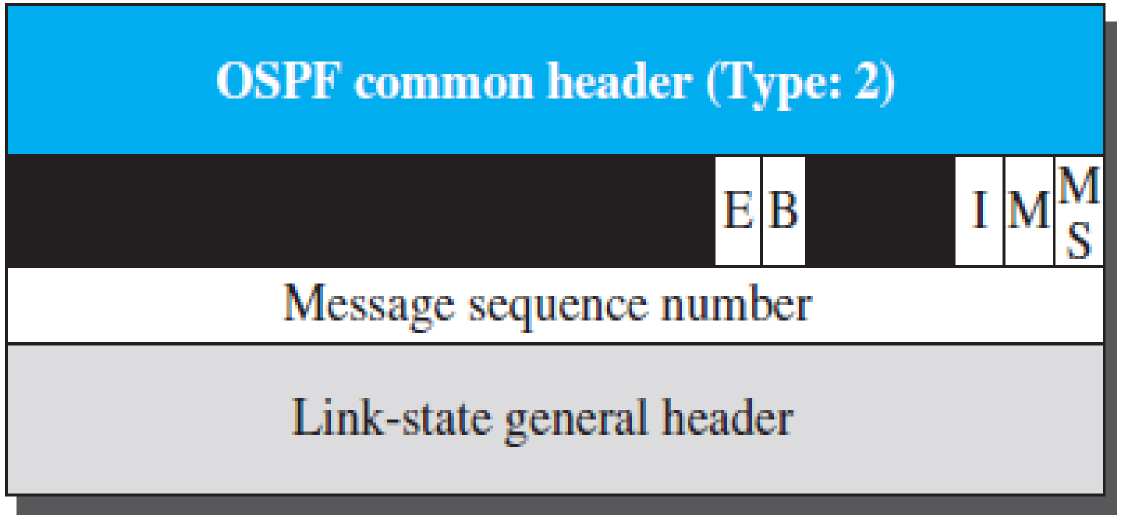
Dead interval

Designated router IP address

Backup designated router IP address

Rep. Neighbor IP address

Hello message



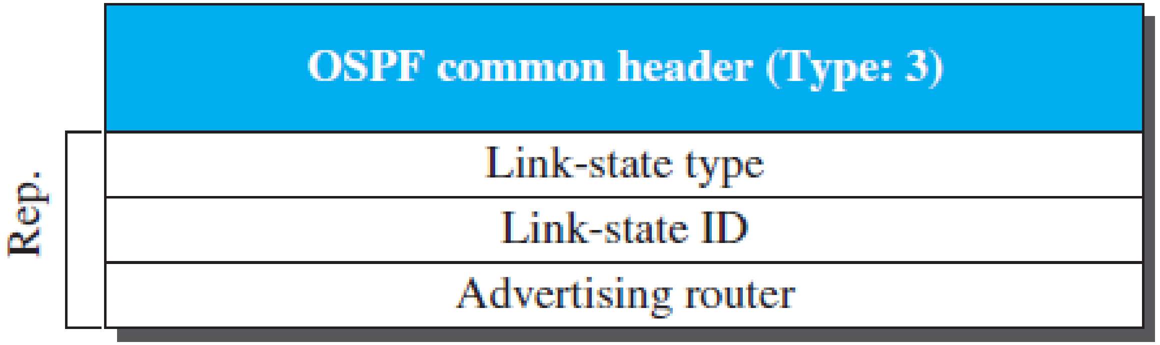
OSPF common header (Type: 2)

E | B | I | M | MS

Message sequence number

Link-state general header

Database description



OSPF common header (Type: 3)

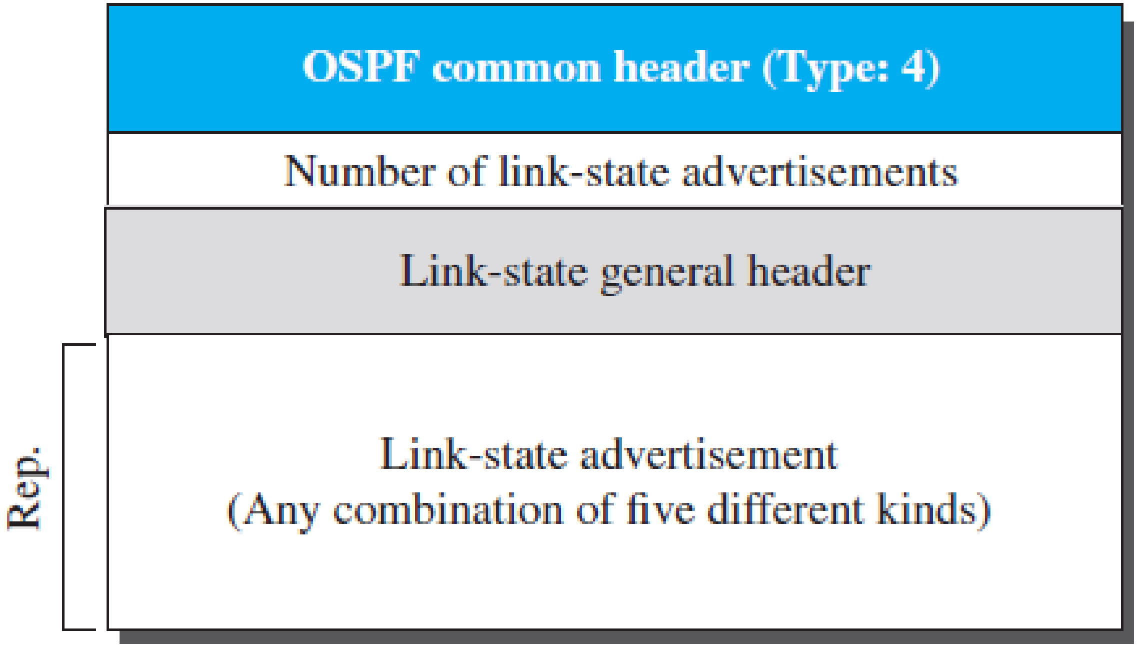
Rep.

Link-state type

Link-state ID

Advertising router

Link-state request



OSPF common header (Type: 4)

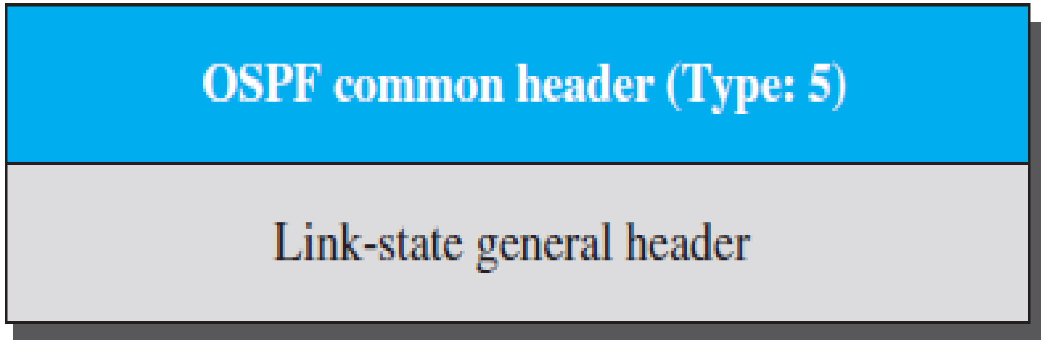
Number of link-state advertisements

Link-state general header

Rep.

Link-state advertisement
(Any combination of five different kinds)

Link-state update



OSPF common header (Type: 5)

Link-state general header

Link-state acknowledgment

Legend

E, T, B, I, M, MS: flags used by OSPF
Priority: used to define the designated router
Rep.: Repeated as required

OSPF Algorithm:

OSPF implements the link-state routing algorithm with a few changes:

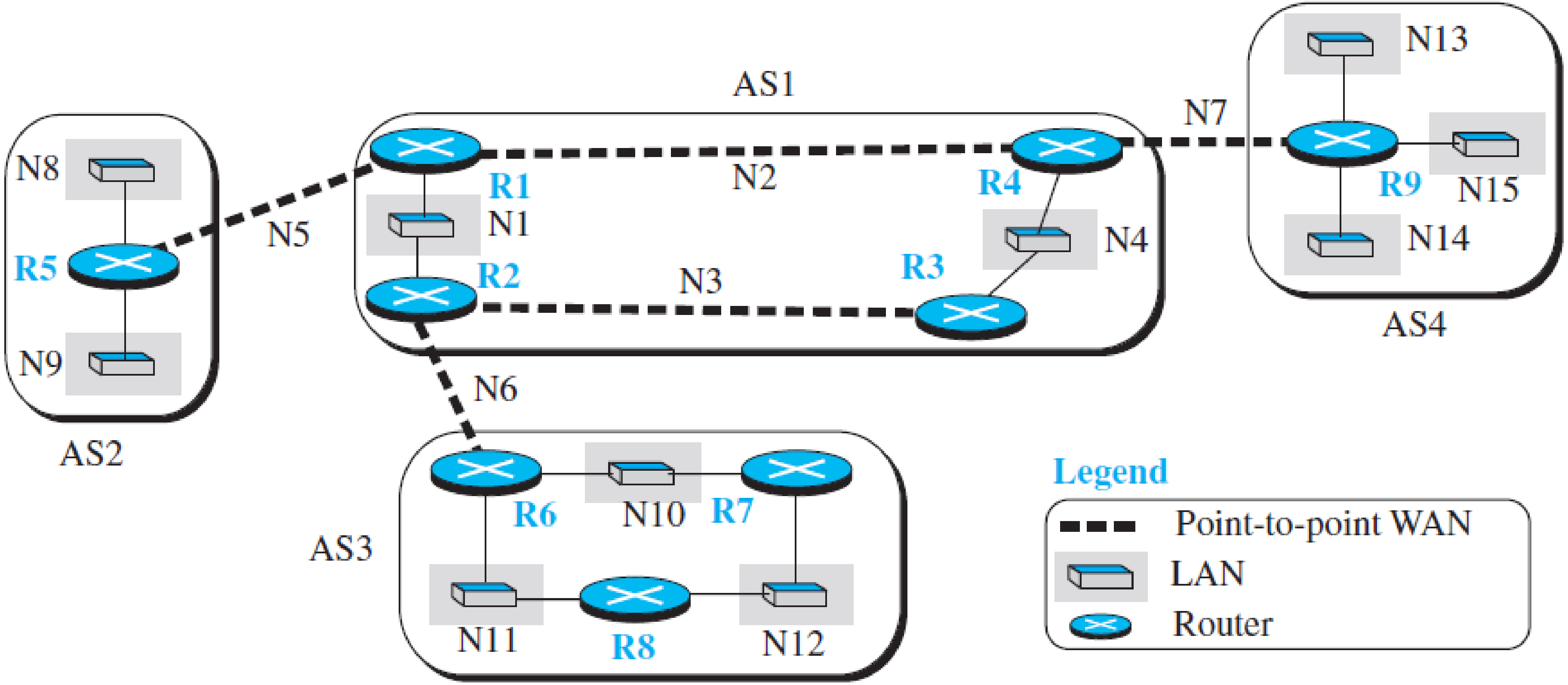
- After each router has created the shortest-path tree, the algorithm needs to use it to create the corresponding routing algorithm.
- The algorithm needs to be augmented to handle sending and receiving all five types of messages.

Performance:

- ***Update Messages.*** The link-state messages in OSPF have a somewhat complex format. They also are flooded to the whole area. If the area is large, these messages may create heavy traffic and use a lot of bandwidth.
- ***Convergence of Forwarding Tables.*** When the flooding of LSPs is completed, each router can create its own shortest-path tree and forwarding table; convergence is fairly quick. However, each router needs to run Dijkstra’s algorithm, which may take some time.
- ***Robustness*. ** The OSPF protocol is more robust than RIP because, after receiving the completed LSDB, each router is independent and does not depend on other routers in the area. Corruption or failure in one router does not affect other routers as seriously as in RIP.

Border Gateway Protocol (BGP)

- It is the only interdomain routing protocol used in the Internet today.
- BGP4 is based on the path-vector algorithm.
- Example of an internet with four autonomous systems. AS2, AS3, and AS4 are *stub* autonomous systems; AS1 is a *transient* one. In our example, data exchange between AS2, AS3, and AS4 should pass through AS1.



- Each autonomous system uses one of the two common intradomain protocols, RIP or OSPF.
- To enable each router to route a packet to any network in the internet:
  - A variation of BGP4, called *external BGP* (*eBGP*) on each *border router* (the one at the edge of each AS which is connected to a router at another AS).
  - We then install the second variation of BGP, called *internal BGP* (*iBGP*), on all routers.
- This means that the border routers will be running three routing protocols (intradomain, eBGP, and iBGP), but other routers are running two protocols (intradomain and iBGP).

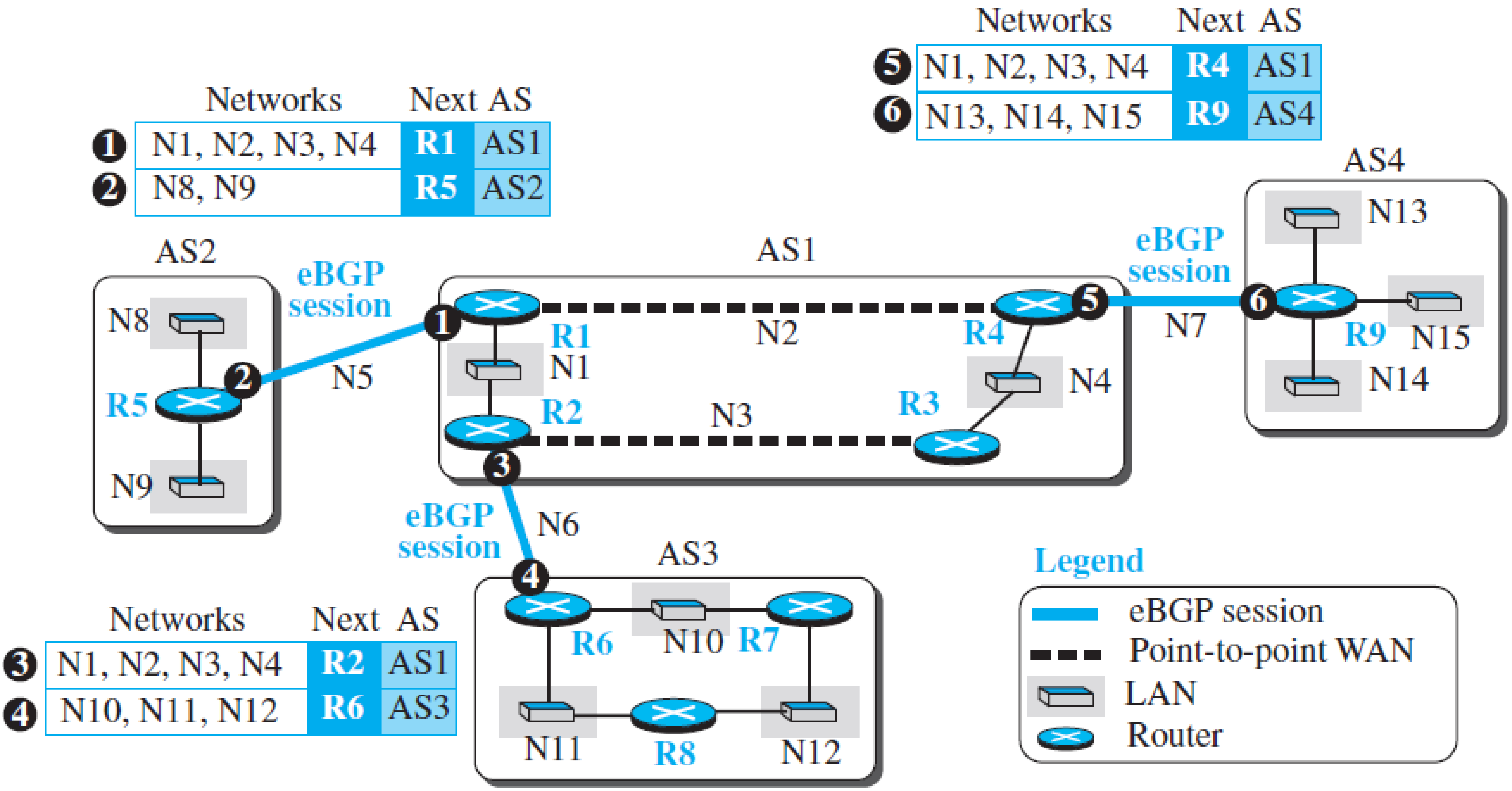
Operation of External BGP (eBGP)

BGP is a kind of point-to-point protocol.

BGP software installed on routers create a TCP connection using the well-known port 179.

- The two routers that run the BGP processes are called *BGP peers* or *BGP speakers*.
- The eBGP variation of BGP allows two physically connected border routers in two different ASs to form pairs of eBGP speakers and exchange messages.
- Each logical connection in BGP parlance is referred to as a *session*.
- the messages exchanged during three eBGP sessions help some routers know how to route packets to some networks in the internet, but the reachability information is not complete. There are two problems:
    - Some border routers do not know how to route a packet destined for nonneighbor ASs.
    - None of the nonborder routers know how to route a packet destined for any networks in other ASs.
- To address the above two problems, we need to allow all pairs of routers (border or nonborder) to run the second variation of the BGP protocol, iBGP.

eBGP operation



	Networks	Next	AS
❶	N1, N2, N3, N4	R1	AS1
❷	N8, N9	R5	AS2

	Networks	Next	AS
❸	N1, N2, N3, N4	R2	AS1
❹	N10, N11, N12	R6	AS3

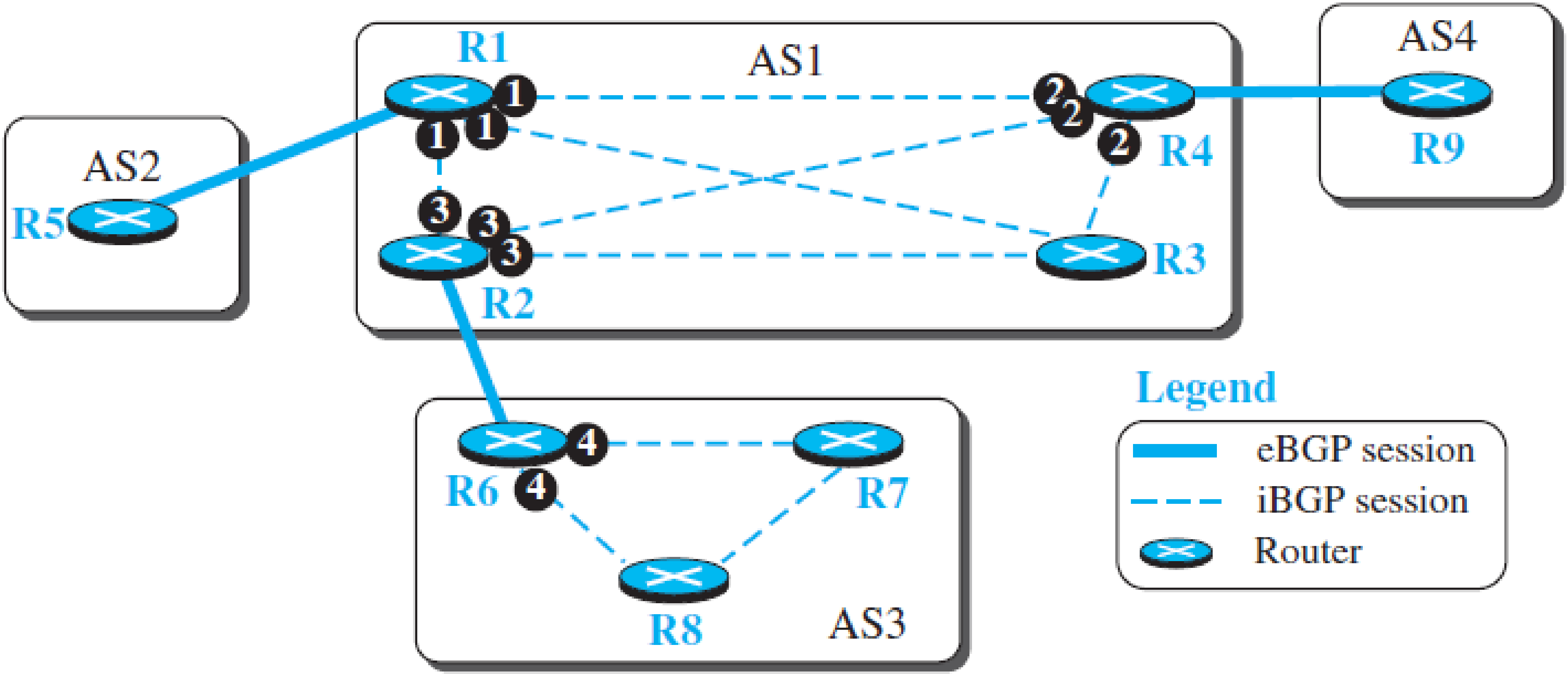
	Networks	Next	AS
❺	N1, N2, N3, N4	R4	AS1
❻	N13, N14, N15	R9	AS4

Operation of Internal BGP (iBGP)

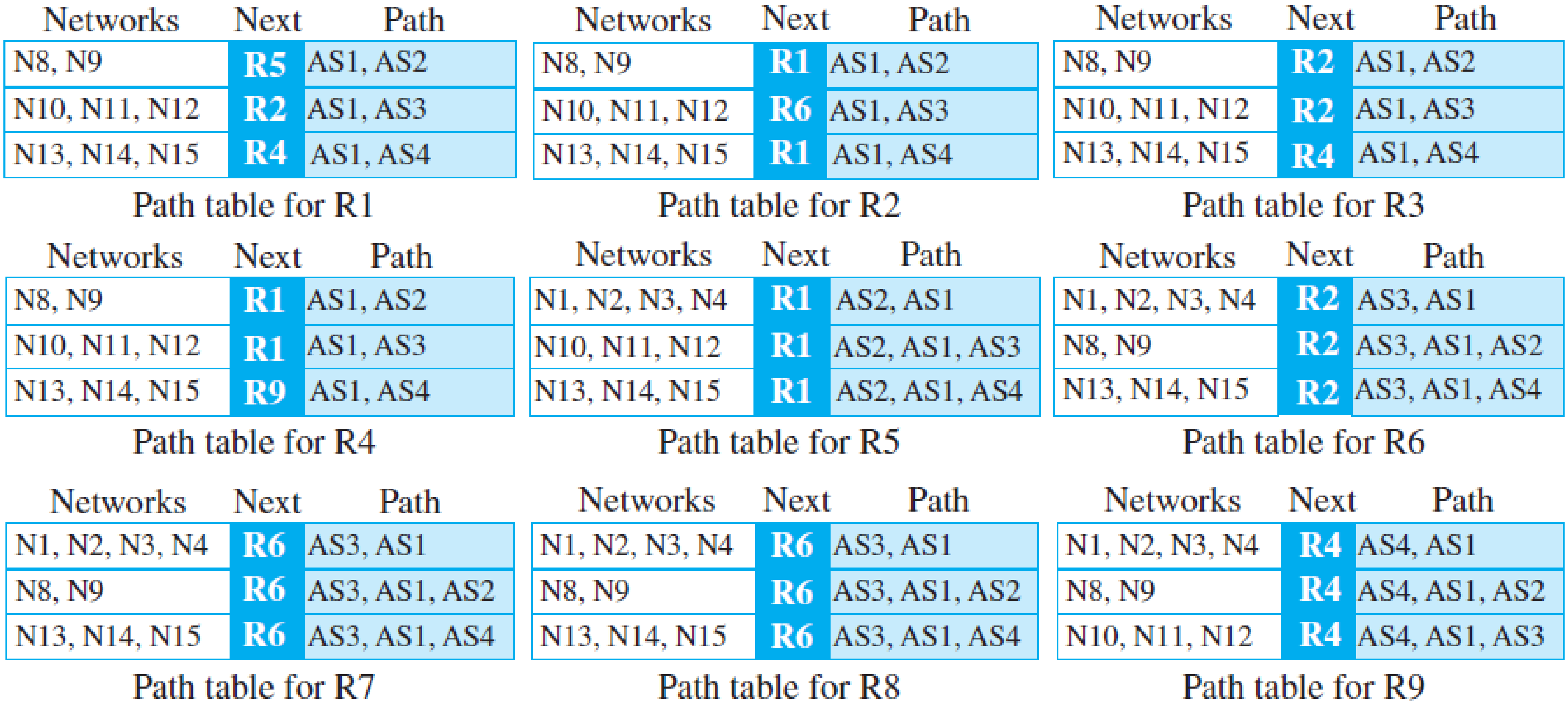
- The iBGP protocol is similar to the eBGP protocol in that it uses the service of TCP on the well-known port 179
- It creates a session between any possible pair of routers inside an autonomous system.
- If an AS has only one router, there cannot be an iBGP session.
- If there are *n* routers in an autonomous system, there should be [$n \times (n - 1) / 2$] iBGP sessions in that autonomous system (a fully connected mesh) to prevent loops in the system.
- Each router needs to advertise its own reachability to the peer in the session instead of flooding what it receives from another peer in another session.

Combination of eBGP and iBGP sessions

	Networks	Next	AS
1	N8, N9	R1	AS1, AS2
2	N13, N14, N15	R4	AS1, AS4
3	N10, N11, N12	R2	AS1, AS3
4	N1, N2, N3, N4	R6	AS3, AS1



Finalized BGP path tables



Networks	Next	Path
N8, N9	R5	AS1, AS2
N10, N11, N12	R2	AS1, AS3
N13, N14, N15	R4	AS1, AS4

Path table for R1

Networks	Next	Path
N8, N9	R1	AS1, AS2
N10, N11, N12	R6	AS1, AS3
N13, N14, N15	R1	AS1, AS4

Path table for R2

Networks	Next	Path
N8, N9	R2	AS1, AS2
N10, N11, N12	R2	AS1, AS3
N13, N14, N15	R4	AS1, AS4

Path table for R3

Networks	Next	Path
N8, N9	R1	AS1, AS2
N10, N11, N12	R1	AS1, AS3
N13, N14, N15	R9	AS1, AS4

Path table for R4

Networks	Next	Path
N1, N2, N3, N4	R1	AS2, AS1
N10, N11, N12	R1	AS2, AS1, AS3
N13, N14, N15	R1	AS2, AS1, AS4

Path table for R5

Networks	Next	Path
N1, N2, N3, N4	R2	AS3, AS1
N8, N9	R2	AS3, AS1, AS2
N13, N14, N15	R2	AS3, AS1, AS4

Path table for R6

Networks	Next	Path
N1, N2, N3, N4	R6	AS3, AS1
N8, N9	R6	AS3, AS1, AS2
N13, N14, N15	R6	AS3, AS1, AS4

Path table for R7

Networks	Next	Path
N1, N2, N3, N4	R6	AS3, AS1
N8, N9	R6	AS3, AS1, AS2
N13, N14, N15	R6	AS3, AS1, AS4

Path table for R8

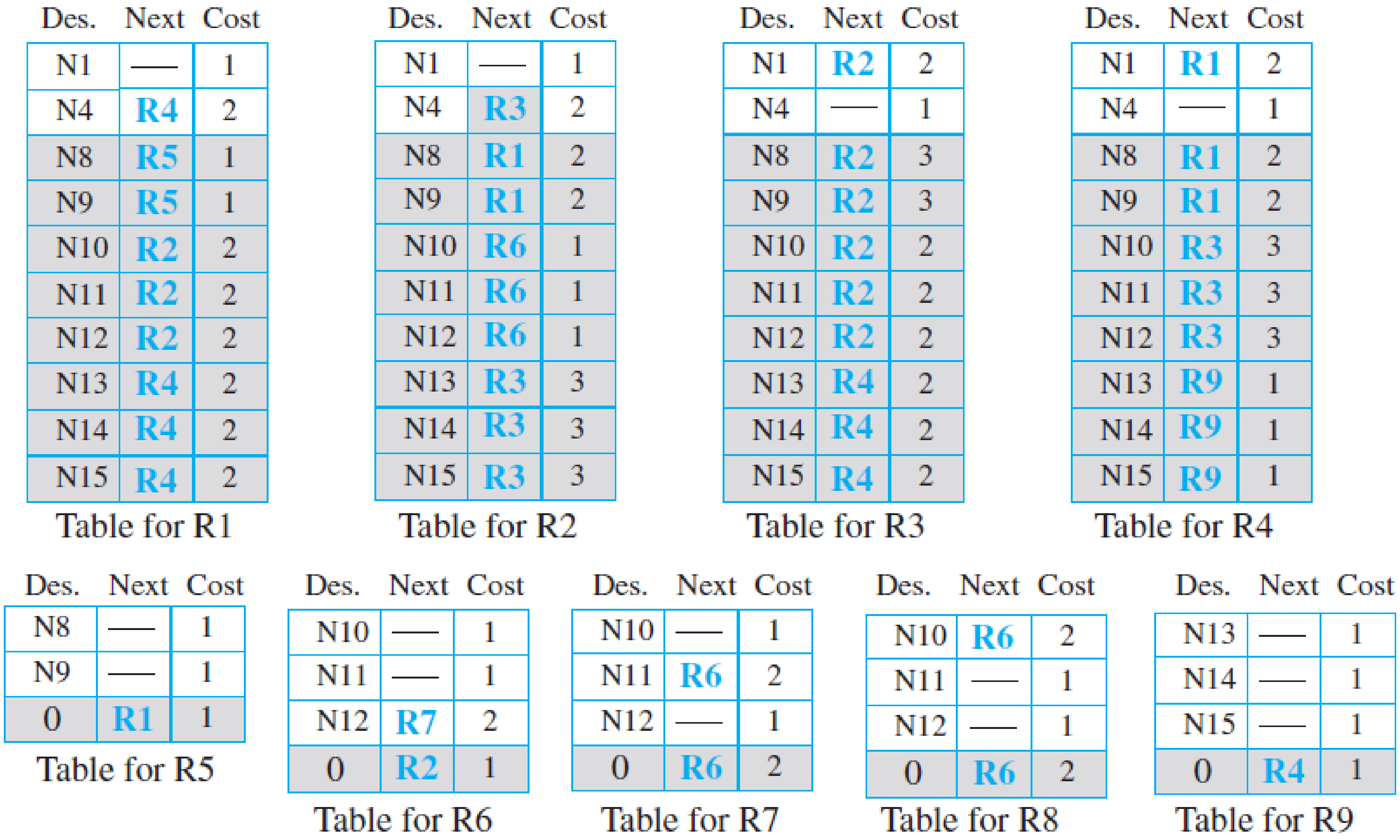
Networks	Next	Path
N1, N2, N3, N4	R4	AS4, AS1
N8, N9	R4	AS4, AS1, AS2
N10, N11, N12	R4	AS4, AS1, AS3

Path table for R9

Injection of Information into Intradomain Routing

- The role of an interdomain routing protocol such as BGP is to help the routers inside the AS to augment their routing information. In other words, the path tables collected and organized by BPG are not used, per se, for routing packets; they are injected into intradomain forwarding tables (RIP or OSPF) for routing packets. This can be done in several ways depending on the type of AS.
- In the case of a stub AS, the only area border router adds a default entry at the end of its forwarding table and defines the next router to be the speaker router at the end of the eBGP connection.
- In the case of a transient AS, the situation is more complicated. One issue to be resolved is the cost value. We know that RIP and OSPF use different metrics. One solution, which is very common, is to set the cost to the foreign networks at the same cost value as to reach the first AS in the path.

Forwarding tables after injection from BGP



Des.	Next	Cost
N1	—	1
N4	R4	2
N8	R5	1
N9	R5	1
N10	R2	2
N11	R2	2
N12	R2	2
N13	R4	2
N14	R4	2
N15	R4	2

Table for R1

Des.	Next	Cost
N1	—	1
N4	R3	2
N8	R1	2
N9	R1	2
N10	R6	1
N11	R6	1
N12	R6	1
N13	R3	3
N14	R3	3
N15	R3	3

Table for R2

Des.	Next	Cost
N1	R2	2
N4	—	1
N8	R2	3
N9	R2	3
N10	R2	2
N11	R2	2
N12	R2	2
N13	R4	2
N14	R4	2
N15	R4	2

Table for R3

Des.	Next	Cost
N1	R1	2
N4	—	1
N8	R1	2
N9	R1	2
N10	R3	3
N11	R3	3
N12	R3	3
N13	R9	1
N14	R9	1
N15	R9	1

Table for R4

Des.	Next	Cost
N8	—	1
N9	—	1
0	R1	1

Table for R5

Des.	Next	Cost
N10	—	1
N11	—	1
N12	R7	2
0	R2	1

Table for R6

Des.	Next	Cost
N10	—	1
N11	R6	2
N12	—	1
0	R6	2

Table for R7

Des.	Next	Cost
N10	R6	2
N11	—	1
N12	—	1
0	R6	2

Table for R8

Des.	Next	Cost
N13	—	1
N14	—	1
N15	—	1
0	R4	1

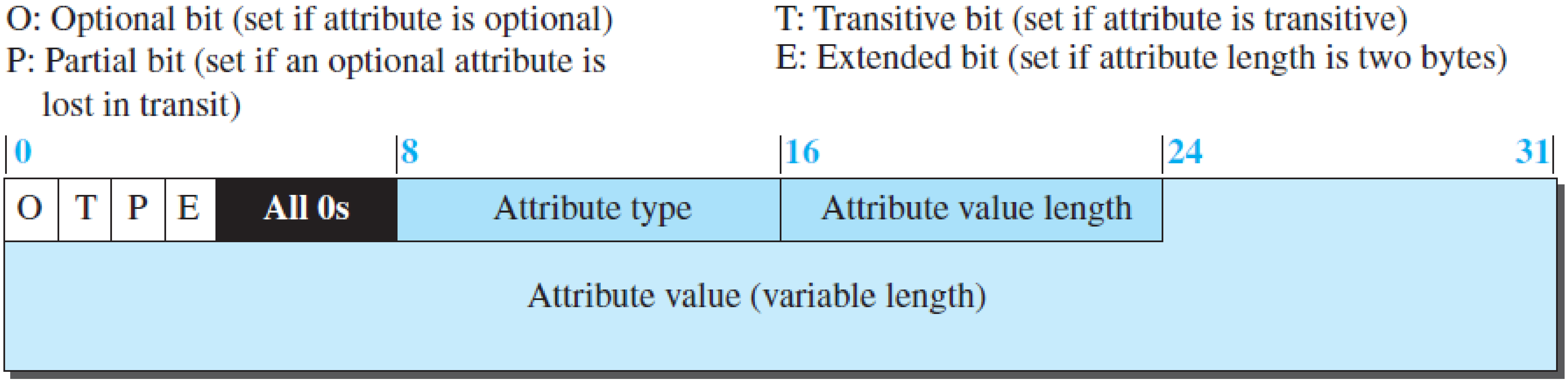
Table for R9

Address Aggregation

- Many destination networks may be included in a forwarding table. Fortunately, BGP4 uses the prefixes as destination identifiers and allows the aggregation of these prefixes.

Path Attributes

- Interdomain routing is more involved and naturally needs more information about how to reach the final destination. In BGP these pieces are called *path attributes.* BGP allows a destination to be associated with up to seven path attributes. Path attributes are divided into two broad categories:
  - **A well-known attribute** must be recognized by all routers; an optional attribute need not be. A well-known attribute can be mandatory, which means that it must be present in any BGP update message, or discretionary, which means it does not have to be.
  - **An optional attribute** can be either transitive, which means it can pass to the next AS, or intransitive, which means it cannot. All attributes are inserted after the corresponding destination prefix in an update message



O: Optional bit (set if attribute is optional)

P: Partial bit (set if an optional attribute is lost in transit)

T: Transitive bit (set if attribute is transitive)

E: Extended bit (set if attribute length is two bytes)

- The first byte in each attribute defines the four attribute flags.
- The next byte defines the type of attributes assigned by ICANN
- The attribute value length defines the length of the attribute value field.

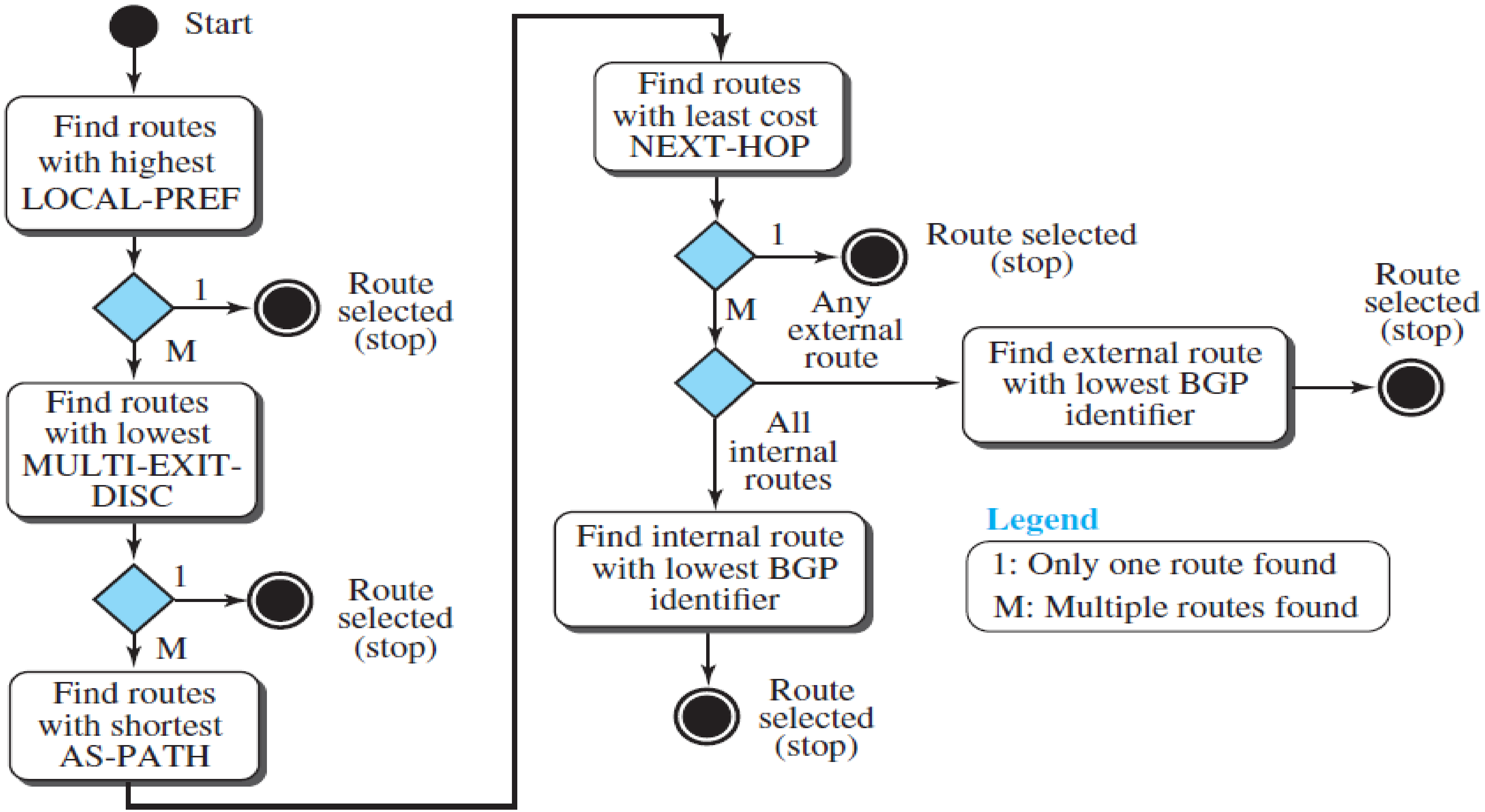
The following gives a brief description of each attribute.

- ***ORIGIN (type 1)*.** This is a well-known mandatory attribute, which defines the source of the routing information. This attribute can be defined by one of the three values: 1, 2, and 3. Value 1 means that the information about the path has been taken from an intradomain protocol (RIP or OSPF). Value 2 means that the information comes from BGP. Value 3 means that it comes from an unknown source.
- ***AS-PATH (type 2).*** This is a well-known mandatory attribute, which defines the list of autonomous systems through which the destination can be reached.
- ***NEXT-HOP (type 3)*.** This is a well-known mandatory attribute, which defines the next router to which the data packet should be forwarded.
- ***MULT-EXIT-DISC (type 4)*.** The multiple-exit discriminator is an optional intransitive attribute, which discriminates among multiple exit paths to a destination.

- ***LOCAL-PREF (type 5)*****.** The local preference attribute is a well-known discretionary attribute. It is normally set by the administrator, based on the organization policy. The routes the administrator prefers are given a higher local preference value.
- ***ATOMIC-AGGREGATE (type 6)*****.** This is a well-known discretionary attribute, which defines the destination prefix as not aggregate; it only defines a single destination network. This attribute has no value field, which means the value of the length field is zero.
- ***AGGREGATOR (type 7).*** This is an optional transitive attribute, which emphasizes that the destination prefix is an aggregate. The attribute value gives the number of the last AS that did the aggregation followed by the IP address of the router that did so.

Route Selection

- Where multiple routes are received to a destination, BGP needs to select one among them.



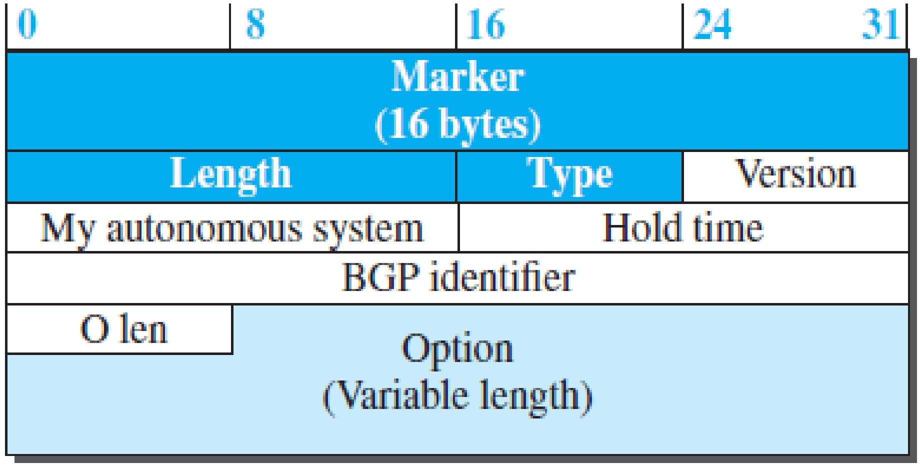
Start
Find routes with highest LOCAL-PREF
1
Route selected (stop)
M
Find routes with lowest MULTI-EXIT-DISC
1
Route selected (stop)
M
Find routes with shortest AS-PATH
Find routes with least cost NEXT-HOP
1
Route selected (stop)
M
Any external route
Find external route with lowest BGP identifier
Route selected (stop)
All internal routes
Find internal route with lowest BGP identifier
Route selected (stop)
Legend
1: Only one route found
M: Multiple routes found

Messages

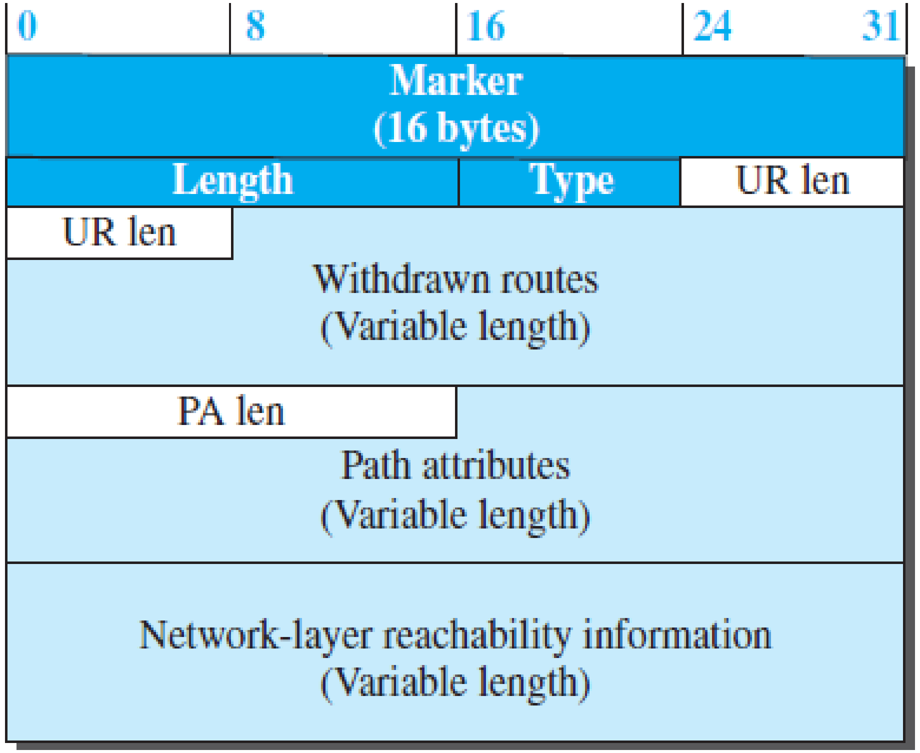
- BGP uses four types of messages for communication between the BGP speakers across the ASs and inside an AS. All BGP packets share the same common header:
- ***Open Message*.** To create a neighborhood relationship, a router running BGP opens a TCP connection with a neighbor and sends an *open message.*
- ***Update Message*.** The *update message* is the heart of the BGP protocol. It is used by a router to withdraw destinations that have been advertised previously, to announce a route to a new destination, or both.
- ***Keepalive Message*.** The BGP peers that are running exchange keepalive messages regularly (before their hold time expires) to tell each other that they are alive.
- ***Notification.*** A notification message is sent by a router whenever an error condition is detected or a router wants to close the session.

Performance

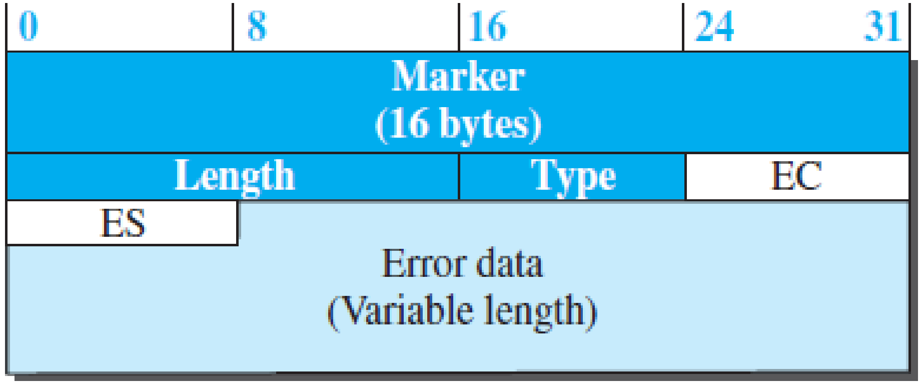
- BGP speakers exchange a lot of messages to create forwarding tables, but BGP is free from loops and count-to-infinity.



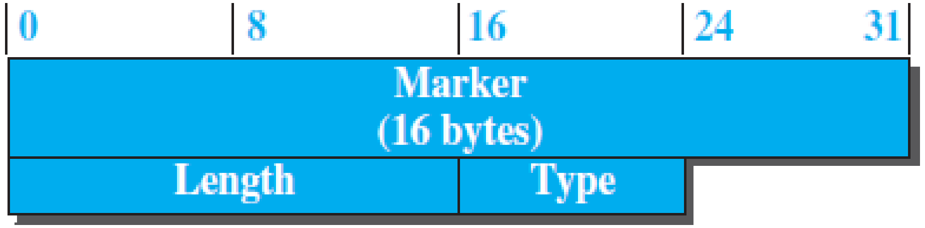
Open message (type 1)



Update message (type 2)



Notification message (type 3)



Keepalive message (type 4)

Fields in common header

Marker: Reserved for authentication
Length: Length of total message in bytes
Type: Type of message (1 to 4)

Abbreviations

O len: Option length
EC: Error code
ES: Error subcode
UR len: Unfeasible route length
PA len: Path attribute length