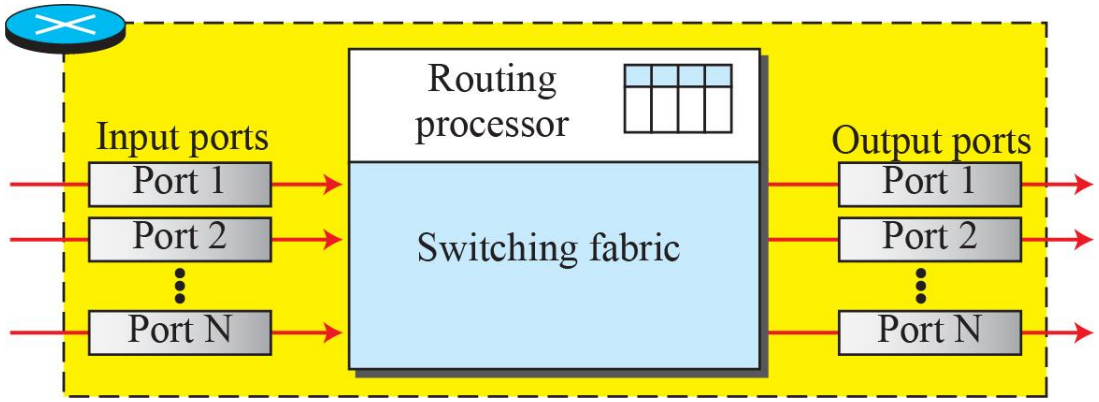
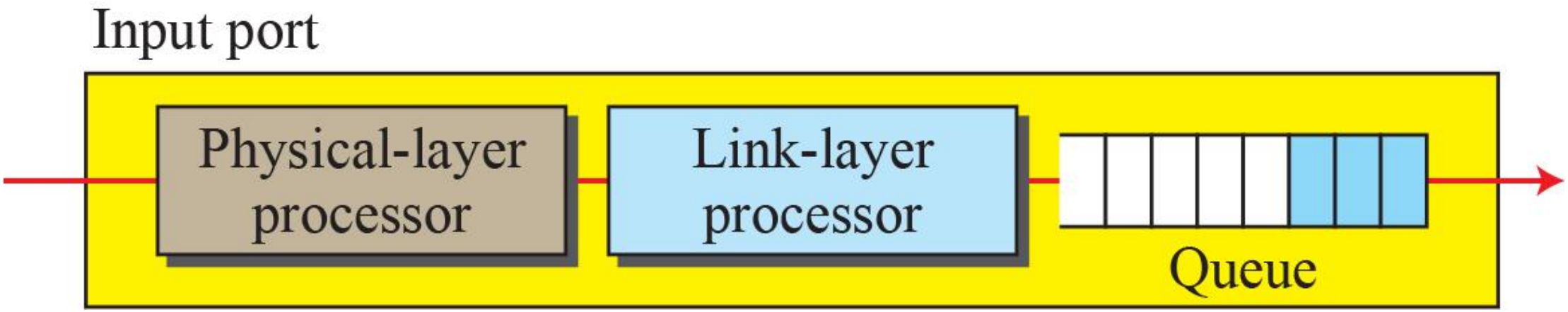
Router:

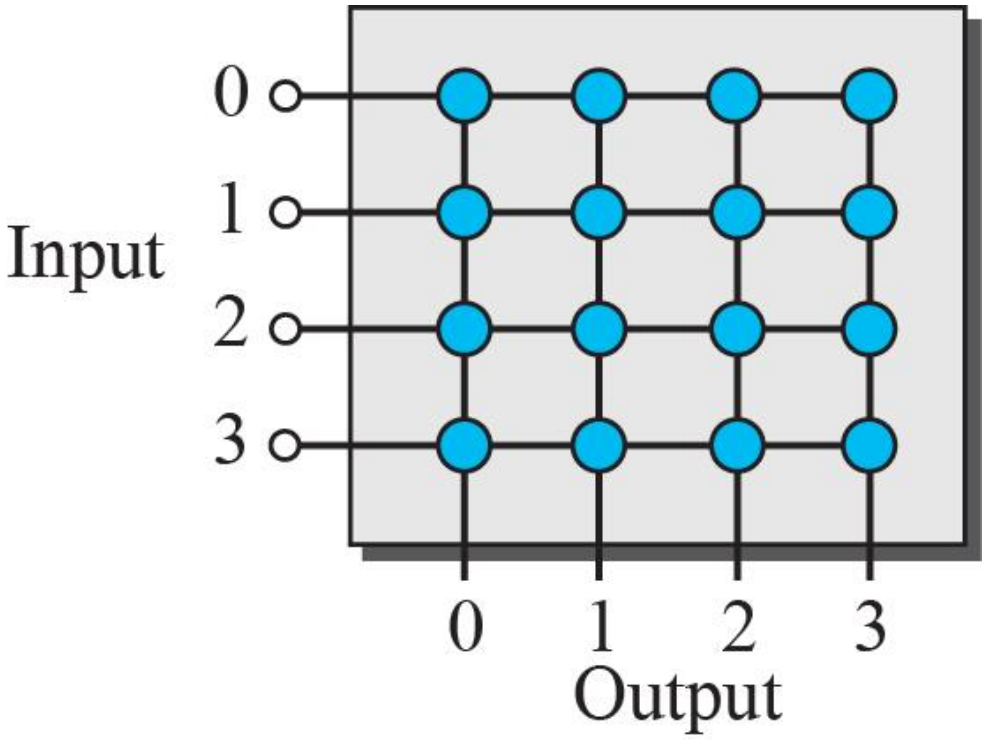
- A router as a device that accepts incoming packets from one of the input ports (interfaces), uses a forwarding table to find the output port from which the packet departs, and sends the packet from this output port.
- Componemts of Router:



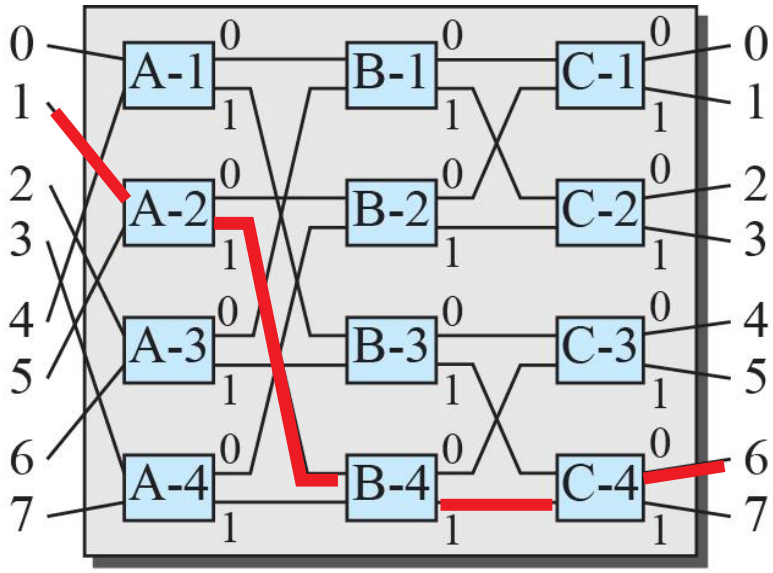
Input ports. The input port performs several functions. It performs the physical layer functionality of terminating an incoming physical link to a router. It performs the data link layer functionality needed to interoperate with the data link layer functionality on the other side of the incoming link. It also performs a lookup and forwarding function so that a datagram forwarded into the switching fabric of the router emerges at the appropriate output port. Control packets (e.g., packets carrying routing protocol information such as RIP, OSPF or IGMP) are forwarded from the input port to the routing processor. In practice, multiple ports are often gathered together on a single line card within a router.



Switching fabric. The switching fabric connects the router's input ports to its output ports. The most difficult task in a router is to move the packet from the input queue to the output queue. The speed with which this is done affects the size of the input/output queue and the overall delay in packet delivery.



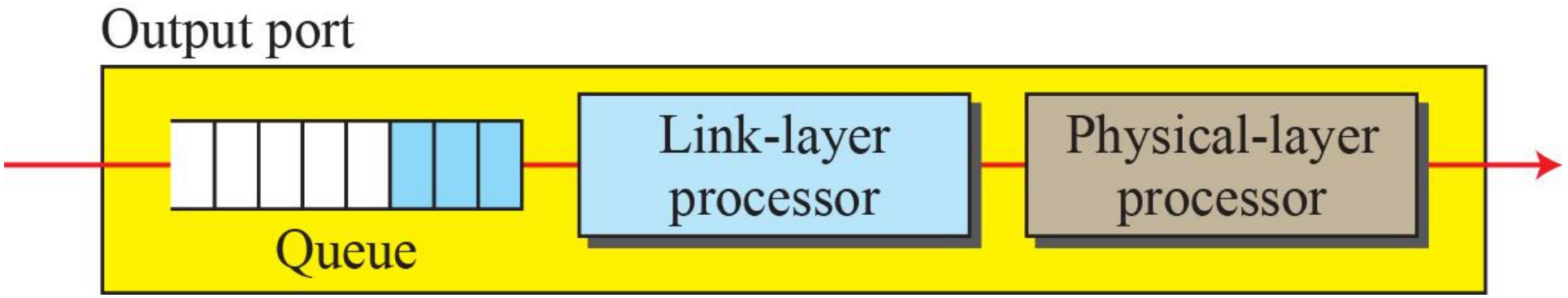
Crossbar switch



a. Input 1 sending to output 6 (110)

Banyan switch

Output ports. An output port stores the datagrams that have been forwarded to it through the switching fabric, and then transmits the datagrams on the outgoing link. The output port thus performs the reverse data link and physical layer functionality as the input port.

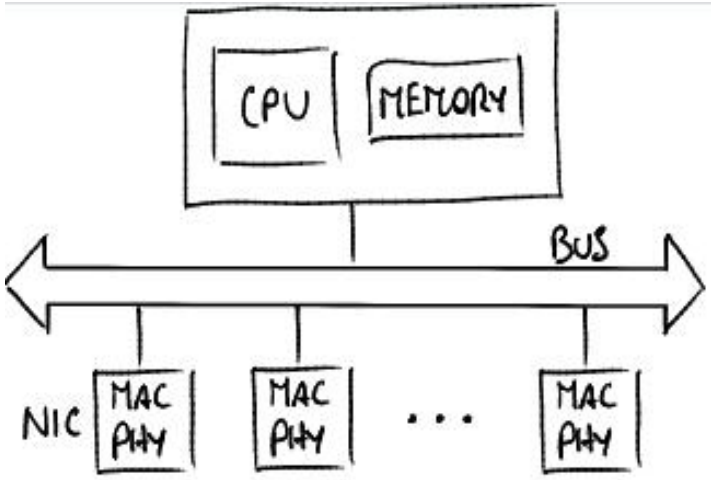


Routing processor. The routing processor executes the routing protocols, maintains the routing tables, and performs network management functions, within the router. The routing processor performs the functions of the network layer. The destination address is used to find the address of the next hop and, at the same time, the output port number from which the packet is sent out. This activity is sometimes referred to as table lookup because the routing processor searches the forwarding table. In the newer routers, this function of the routing processor is being moved to the input ports to facilitate and expedite the process.

1st Gen, 2nd Gen, and 3rd Gen router:

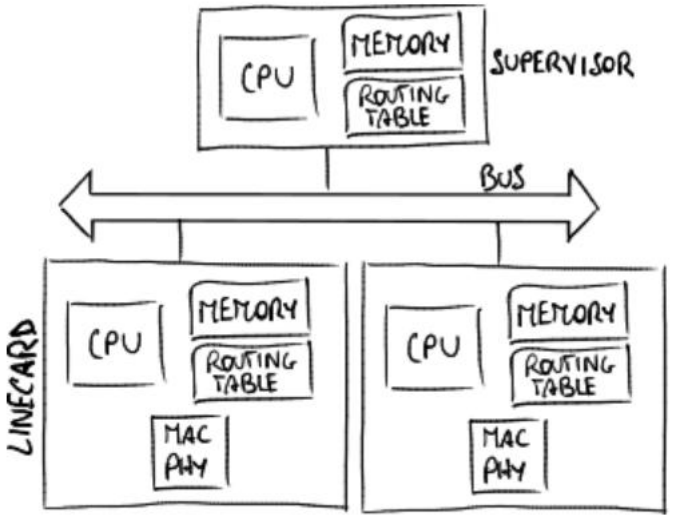
First-generation routers were basically modified PCs.

- network interfaces: they are normal NICs, in a greater number and in manifold kinds with respect to the ones on a normal PC.
- memory: solid-state memory (e.g. SD card) is preferred because less subject to failures with respect to a mechanical hard disk.
- operating system: it is optimized specifically for network traffic processing.



Second-generation routers try to solve the performance problem by moving the processing load from the core CPU to edge units: NICs are replaced by line card, 'smart' network cards which add forwarding modules to physical/MAC components:

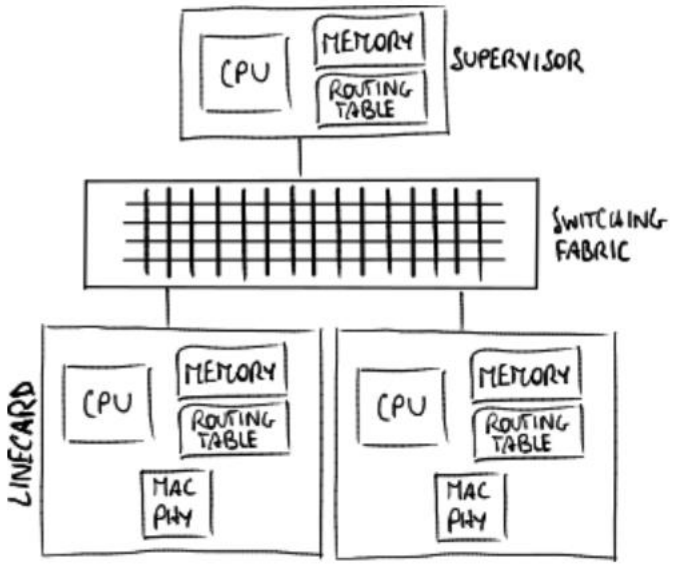
- CPU: the packet processor is a dedicated processor for 'simple' packet processing;
- memory: each incoming packet is stored into a local memory split from the one (generally TCAM) containing the routing table;
- routing table: sometimes the updated routing table is copied from the central unit to line cards.



Third-generation routers focus on the word parallelization problem by replacing the shared bus with a switching fabric (or crossbar) able to handle multiple transfers: a line card is connected to another line card by shorting the switch at the intersection of their 'wires' in the switching matrix.

Two line cards can not be connected to another same line card at the same time → an arbiter is needed which drives switching points and solves contention situations without collisions.

When output interfaces are slow in sending packets with respect to the switching speed of the switching fabric, packets can be queued in various ways

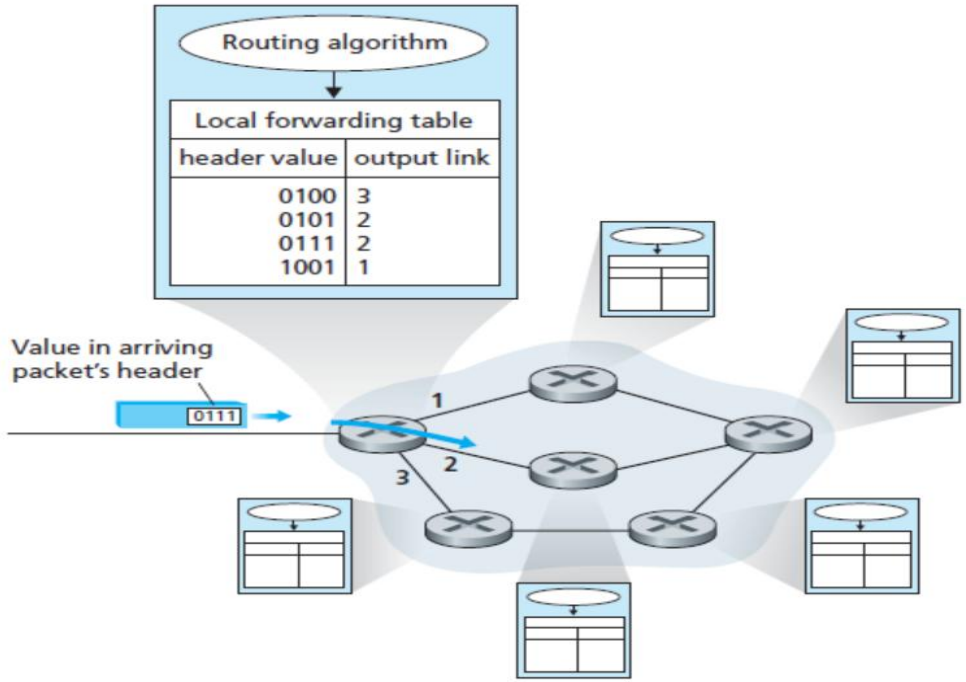


- **The distance between your wireless adapter and router can affect your internet speed. A general rule is that if you double the distance between the router and client (or device), throughput decreases by one-third of its original strength.**

Forwarding refers to the router-local action of transferring packet from an input link interface to the appropriate output link interface.

Routing refers to the network-wide process that determines the end-to-end paths that packets take from source to destination.

- Every router has a forwarding table. A router forward a packet by examining the value of a field in the arriving packet's header, and then using this header value to index into the router's forwarding table. The value stored in the forwarding table entry for that header indicates the router's outgoing link interface to which the packet is to be forwarded.
- Depending on the network layer protocol, the header value could be the destination address of the packet or an indication of the connection to which the packet belongs.



- 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. If the datagram is destined for several destinations (one-to-many delivery), we have multicast routing. Finally, if the datagram is supposed to be delivered to all hosts in the internet (one-to-all), we have broadcast routing.
- **Unicast routing** in the Internet, with a large number of routers and a huge number of hosts, can be done only by using hierarchical routing: routing in several steps using different routing algorithms.
- **Hierarchical Routing** is the method of routing in networks that is based on hierarchical addressing. Hierarchical routing is the procedure of arranging routers in a hierarchical manner.

Static Routing:

- Static Routing or Non-Adaptive Routing, follows user defined routing and routing table is not changed until network administrator changes it. Static Routing uses simple routing algorithms and provides more security than dynamic routing.

Dynamic Routing:

- Dynamic Routing or Adaptive Routing, as name suggests changes the routing table once any changes to network occurs or network topology changes. During network change, dynamic routing sends a signal to router, recalculates the routes and send the updated routing information.
- The **centralized routing** model performs routing using a centralized database, while the **distributed routing** model deals with performing routing using a distributed database. Simply put, a central node contains the routing table in the centralized model, while each node maintains a routing table in the distributed model.

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 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. A good example of a stub AS is the costumer 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. A good example of such an AS is some of the costumer 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 ASs and also allows the traffic to pass through. The provider networks and the backbone are good examples of transient ASs.*

Figure 4.76: Areas in an autonomous system

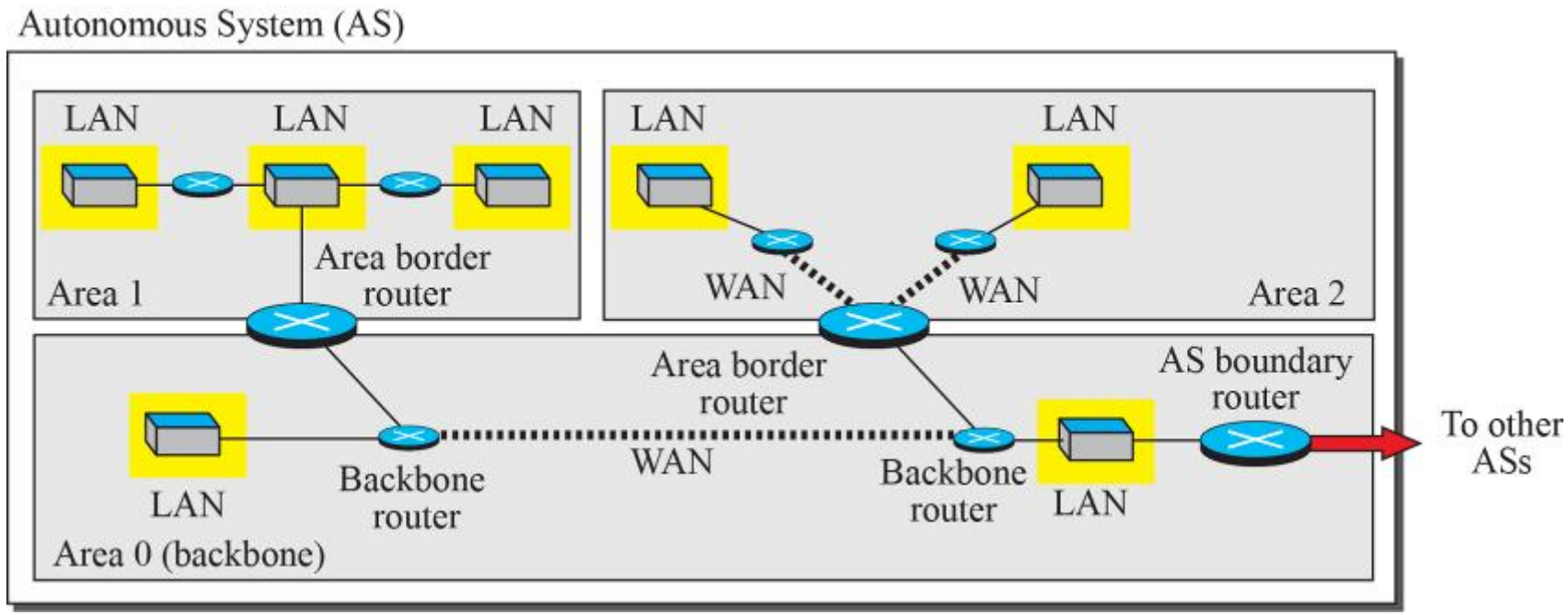
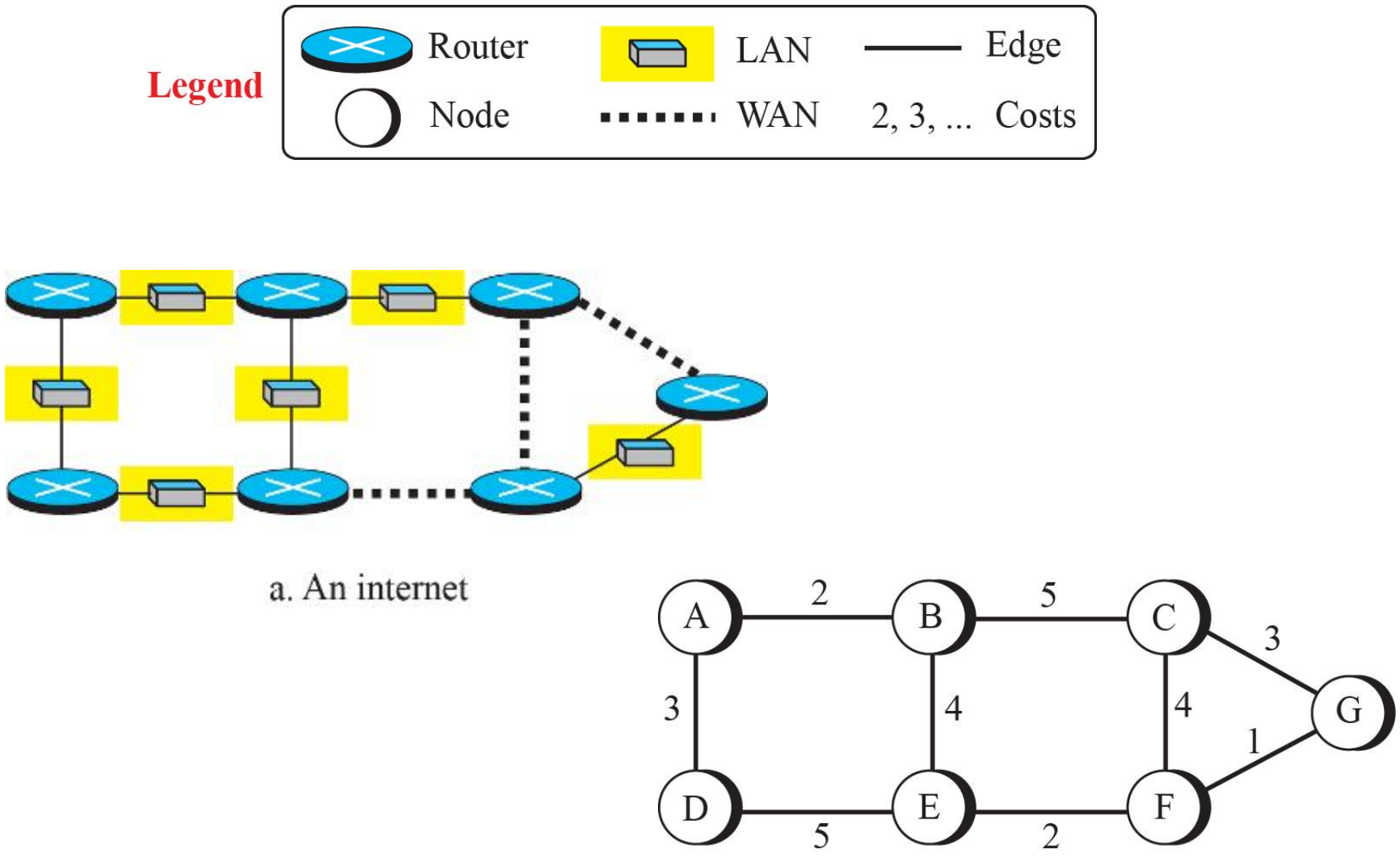


Figure 4.56: An internet and its graphical representation



a. An internet

b. The weighted graph

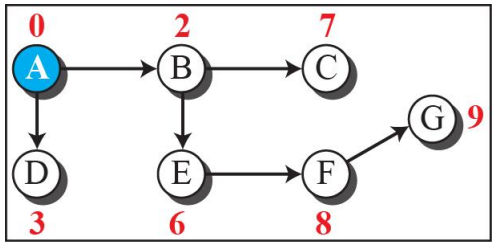
Least-Cost Routing

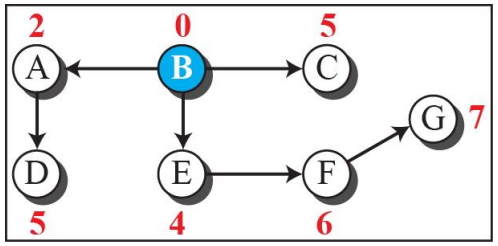
When an internet is modeled as a weighted graph, one of the ways to interpret the best route from the source router to the destination router is to find the least cost between the two.

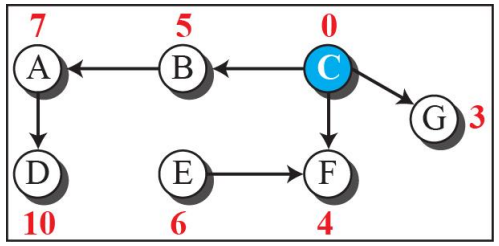
Least-Cost Trees

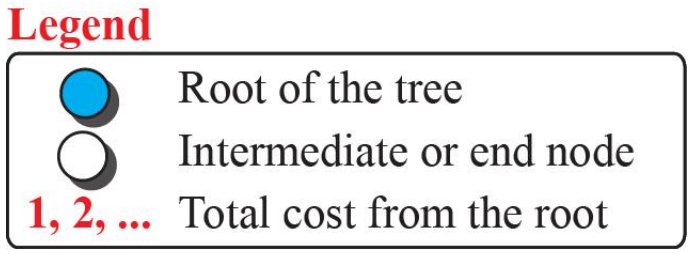
If there are N routers in an internet, there are (N − 1) least-cost paths from each router to any other router. This means we need N × (N − 1) least-cost paths for the whole internet. If we have only 10 routers in an internet, we need 90 least-cost paths. A better way to see all of these paths is to combine them in a least-cost tree.

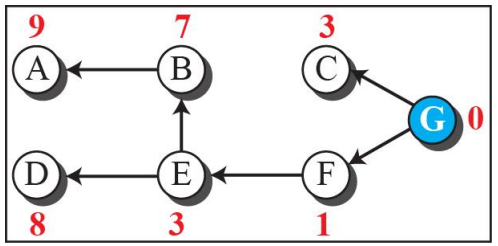
Figure 4.57: Least-cost trees for nodes in the internet of Figure 4.56

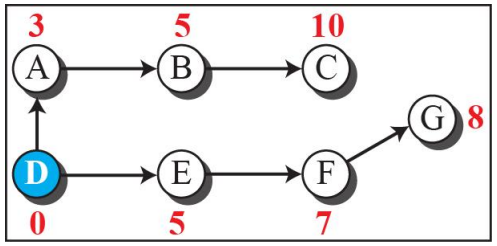


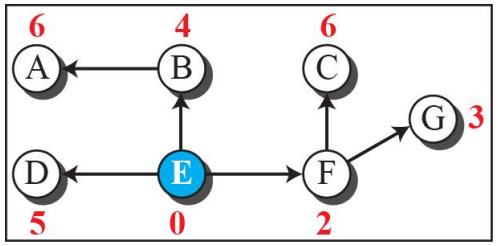


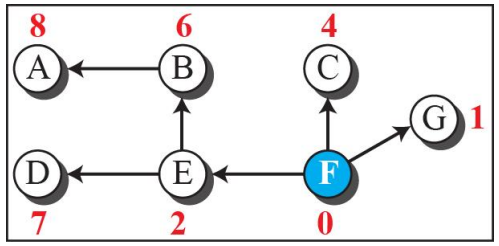




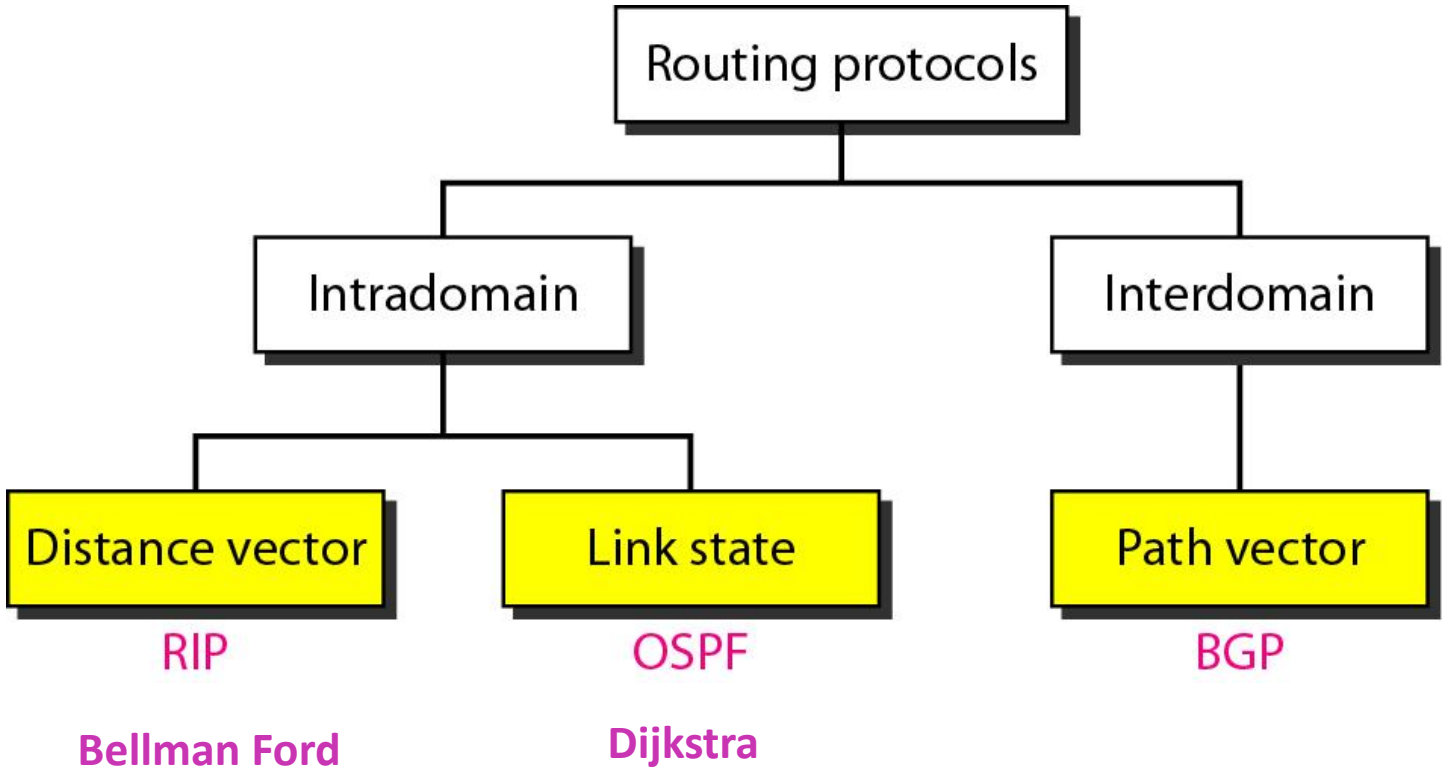








Popular routing protocols



Distance vector routing

A distance-vector routing protocol in data networks determines the best route for data packets based on distance.

- **Initialization:** Each node can know only the distance between itself and its immediate neighbors, those directly connected to it. So, in the initial table only nearby node distance will be present and other node distance is taken as ∞.

- **Sharing:** Each node shares its routing table with its nearby nodes or immediate nodes periodically and when it changes.

- **Updating:** According to the new shared routing table, each node updates their routing table.

Distance-Vector Routing:

In distance-vector routing, a router continuously tells all of its neighbors about what it knows about the whole internet (although the knowledge can be incomplete).

Bellman-Ford Equation:

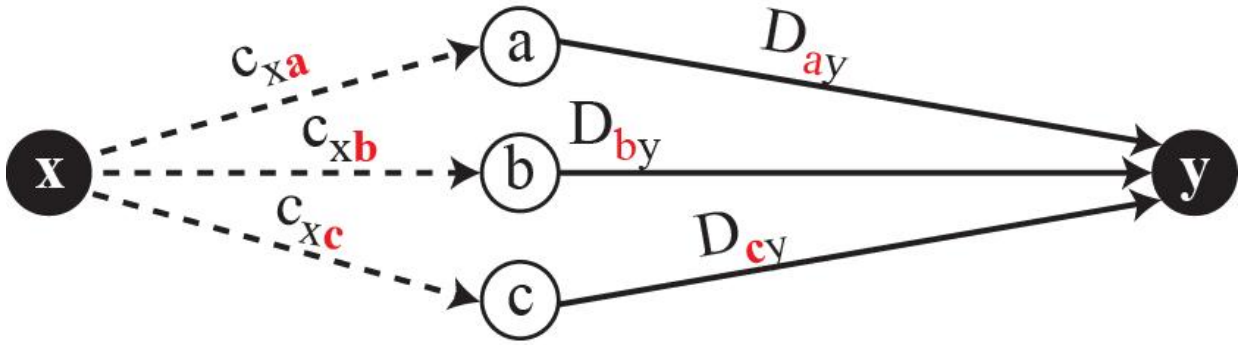
The heart of distance-vector routing is the famous Bellman-Ford equation. This equation is used to find the least cost (shortest distance) between a source node, x, and a destina_x0002_tion node, y, through some intermediary nodes (a, b, c, . . .) when the costs between the source and the intermediary nodes and the least costs between the intermediary nodes and the destination are given. The following shows the general case in which Dij is the short_x0002_est distance and cij is the cost between node i and j.

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

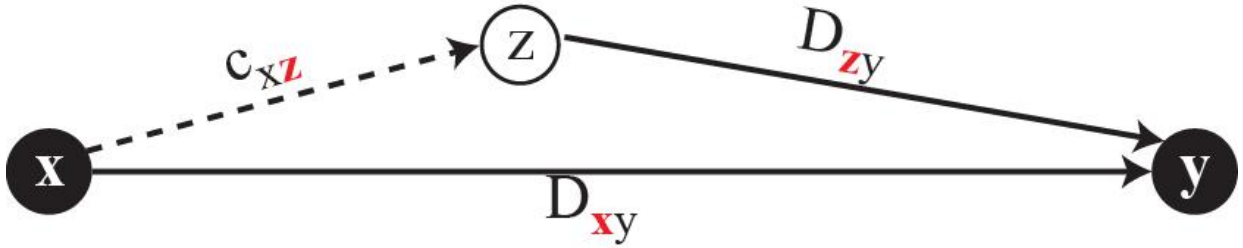
In distance-vector routing, normally we want to update an existing least cost with a least cost through an intermediary node, such as z, if the latter is shorter. In this case, the equation becomes simpler, as shown below:

$$D_{xy} = \min\{D_{xy}, (c_{xz} + D_{zy})\}$$

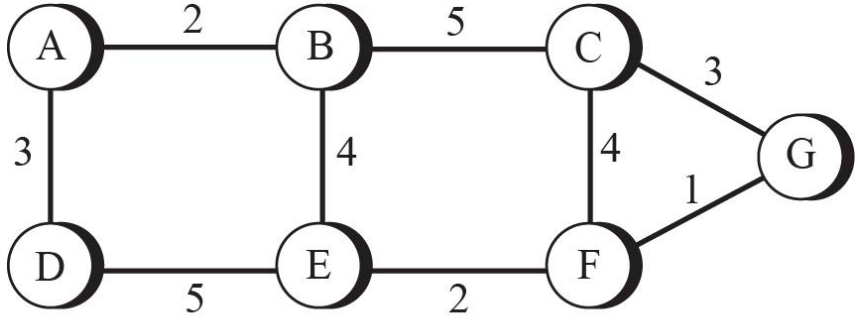
Figure 4.58: Graphical idea behind Bellman-Ford equation

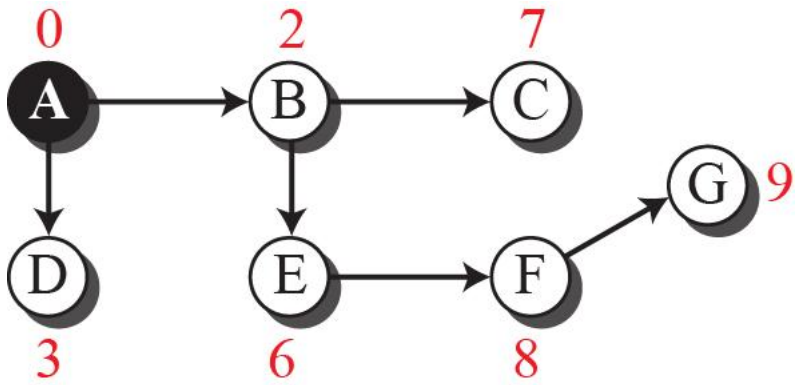


a. General case with three intermediate nodes



b. Updating a path with a new route





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 4.59: The distance vector corresponding to a tree for node A

Figure 4.60: The first distance vector for an internet

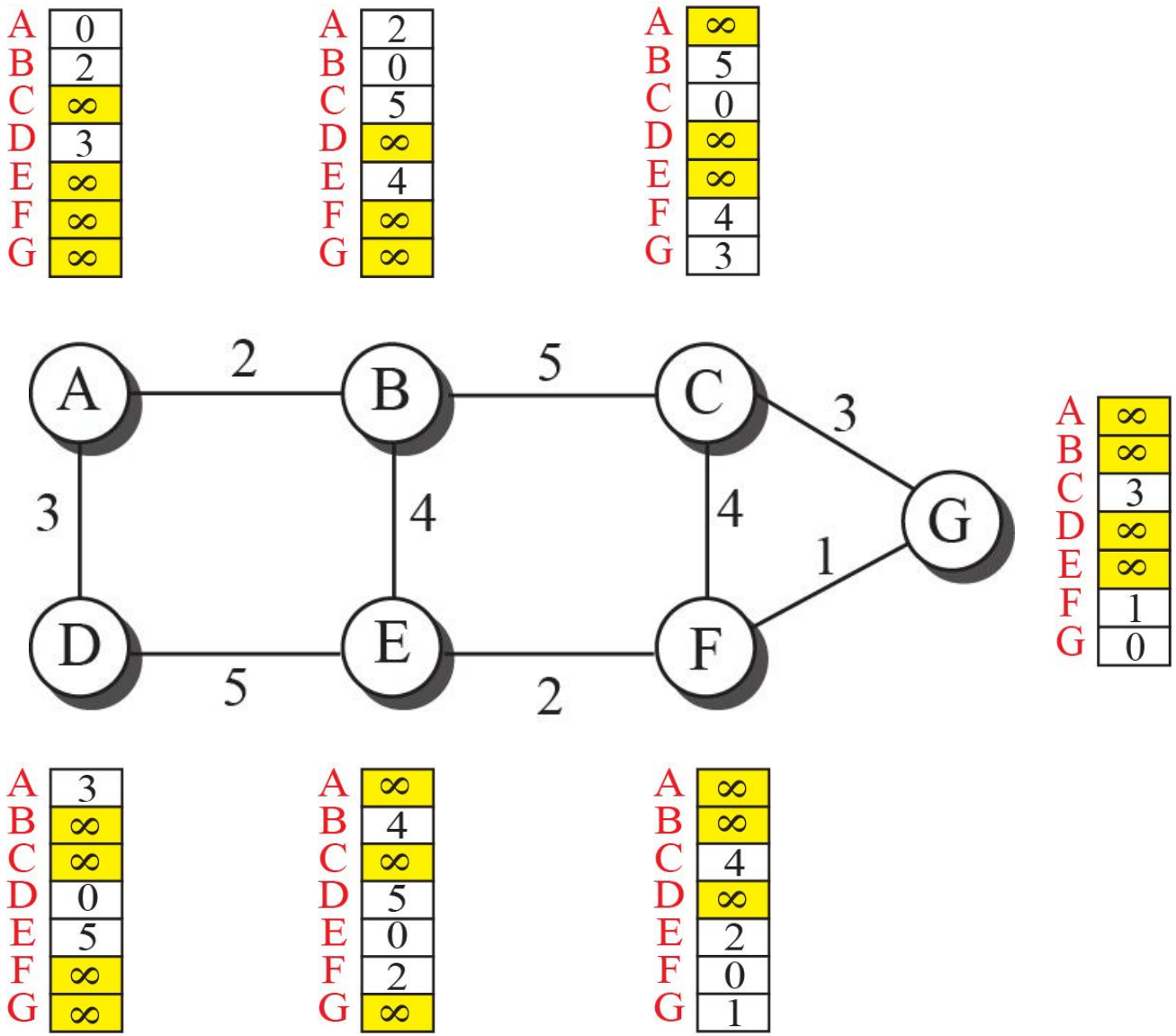
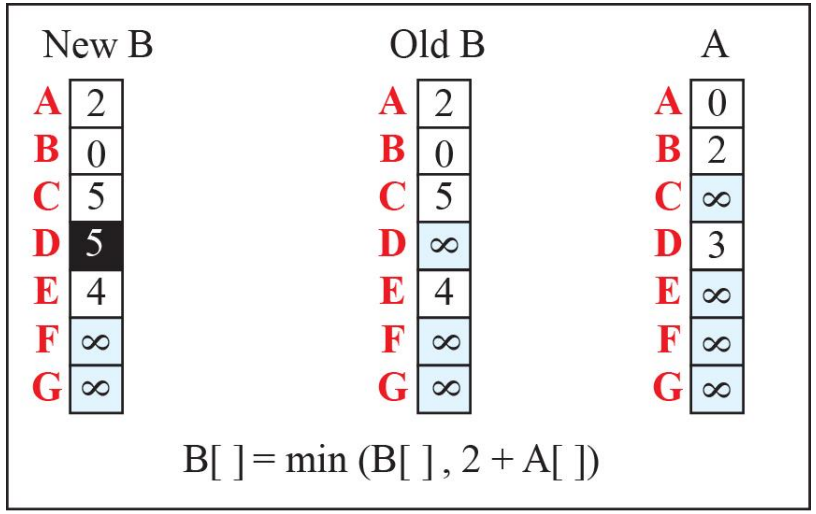
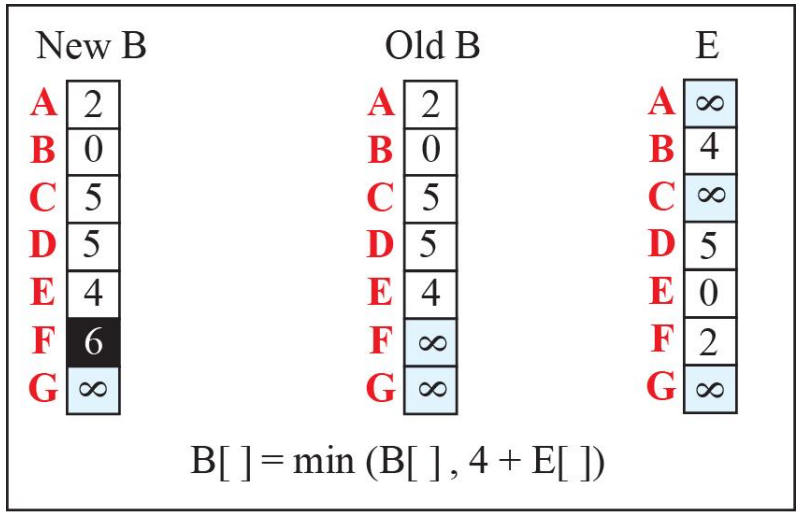


Figure 4.61: Updating distance vectors



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

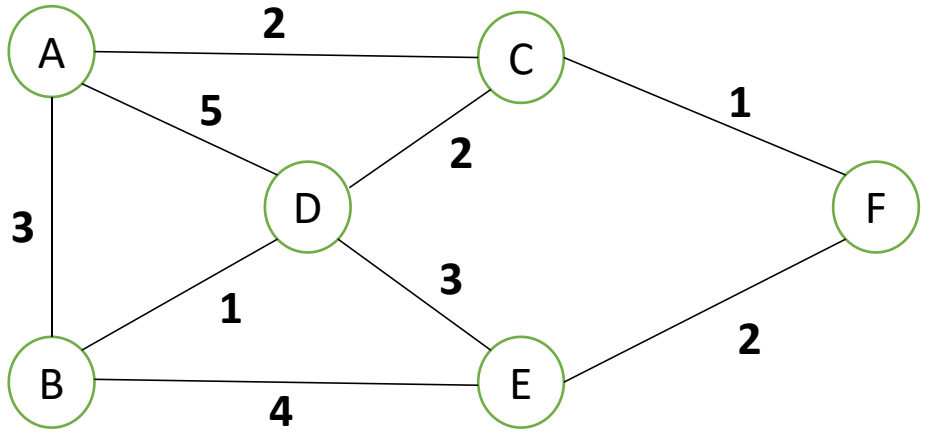
Note:
X[ ]: the whole vector



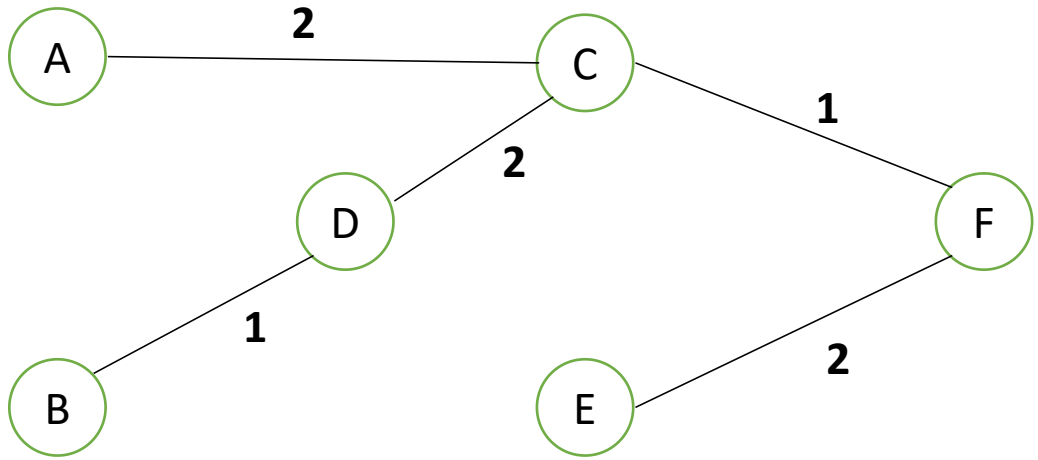
b. Second event: B receives a copy of E's vector.

Bellman-Ford routing algotithm:

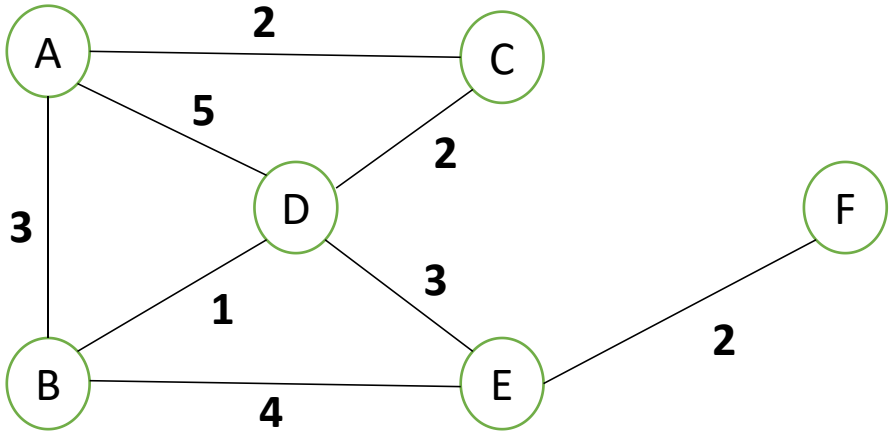
Determine the shortest path to node F using Bellman-Ford distance vector algorithm

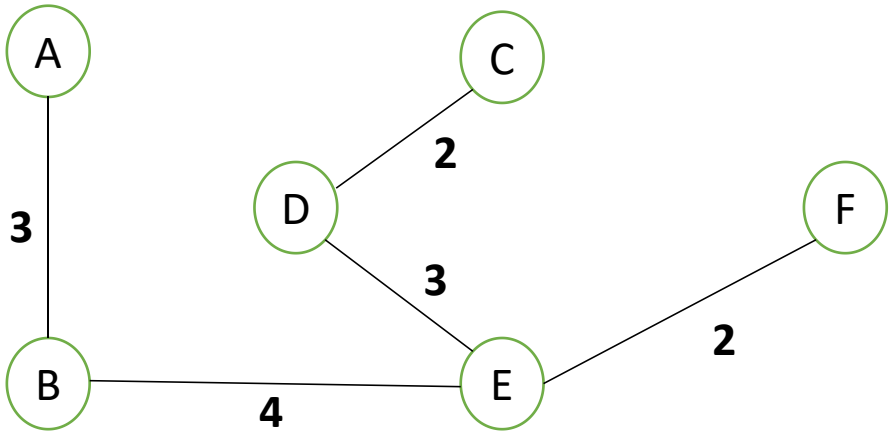


Iteration	NodeA	NodeB	NodeC	NodeD	NodeE
Initial	(-1,∞)	(-1,∞)	(-1,∞)	(-1,∞)	(-1,∞)
1st	(-1,∞)	(-1,∞)	(F,1)	(-1,∞)	(F,2)
2nd	(C,3)	(E,6)	(F,1)	(C,3)	(F,2)
3rd	(C,3)	(D,4)	(F,1)	(C,3)	(F,2)



Consider the link between node C and F is failed. Then re-route for node F.



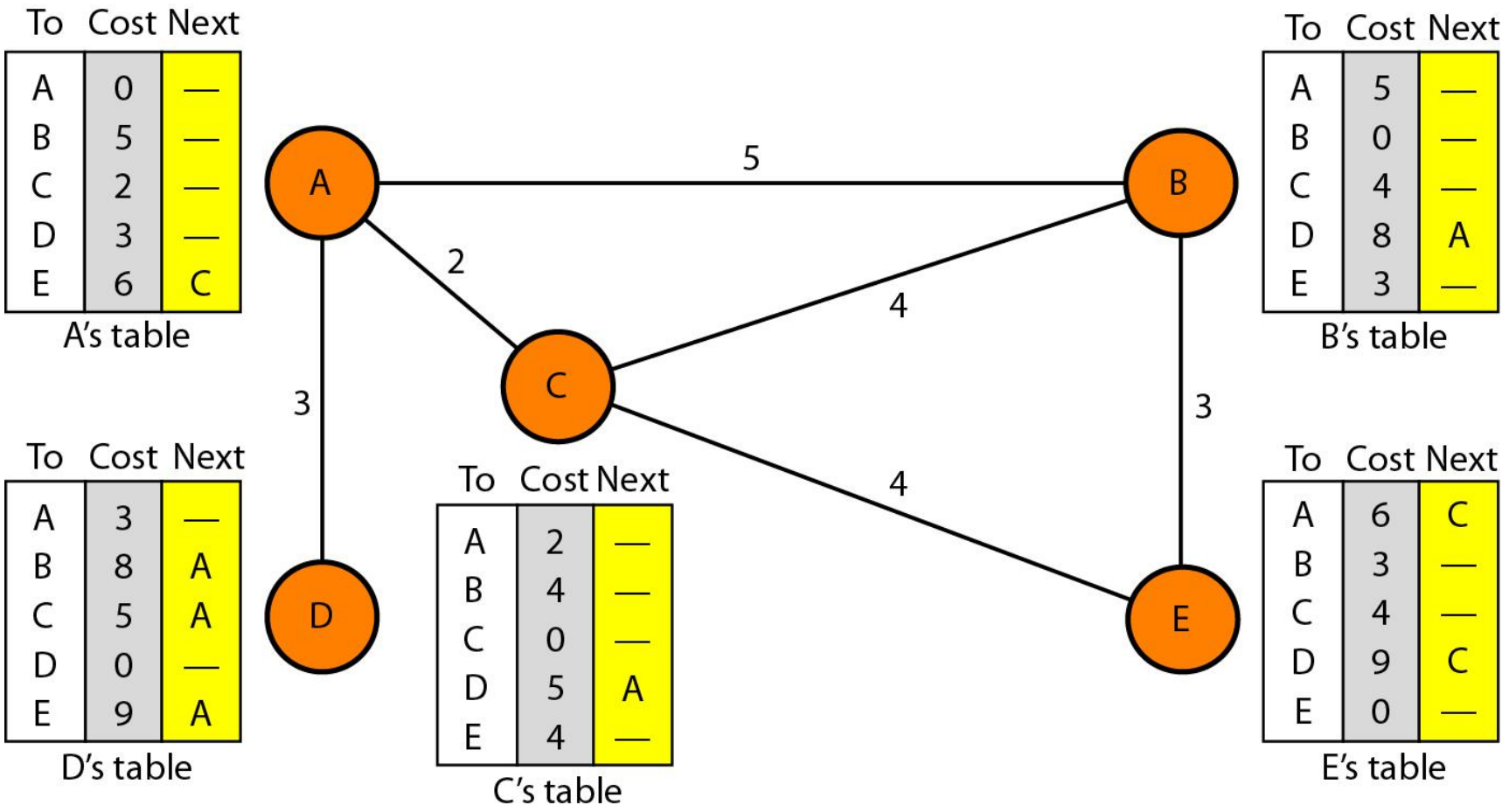


Iteration	NodeA	NodeB	NodeC	NodeD	NodeE
Initial	(C,3)	(D,4)	(F,1)	(C,3)	(F,2)
1st	(C,3)	(D,4)	**(D,5)**	(C,3)	(F,2)
2nd	**(C,7)**	(D,4)	(D,5)	**(E,5)**	(F,2)
3rd	(C,7)	**(E,6)**	**(D,7)**	(E,5)	(F,2)
4th	**(B,9)**	(E,6)	(D,7)	(E,5)	(F,2)

Table 4.4: Distance-Vector Routing Algorithm for A Node

```
1   Distance_Vector_Routing ( )
2   {
3       // Initialize (create initial vectors for the node)
4       D[myself] = 0
5       for (y = 1 to N)
6       {
7           if (y is a neighbor)
8               D[y] = c[myself][y]
9           else
10              D[y] = ∞
11      }
12      send vector {D[1], D[2], ..., D[N]} to all neighbors
13      // Update (improve the vector with the vector received from a neighbor)
14      repeat (forever)
15      {
16          wait (for a vector Dw from a neighbor w or any change in the link)
17          for (y = 1 to N)
18          {
19              D[y] = min [D[y], (c[myself][w] + Dw[y])]          // Bellman-Ford equation
20          }
21          if (any change in the vector)
22              send vector {D[1], D[2], ..., D[N]} to all neighbors
23      }
24  } // End of Distance Vector
```

Distance vector routing tables



Count to Infinity:

For a routing protocol to work properly, if a link is broken (cost becomes infinity), every other router should be aware of it immediately, but in distance-vector routing, this takes some time. The problem is referred to as count to infinity. It sometimes takes several updates before the cost for a broken link is recorded as infinity by all routers.

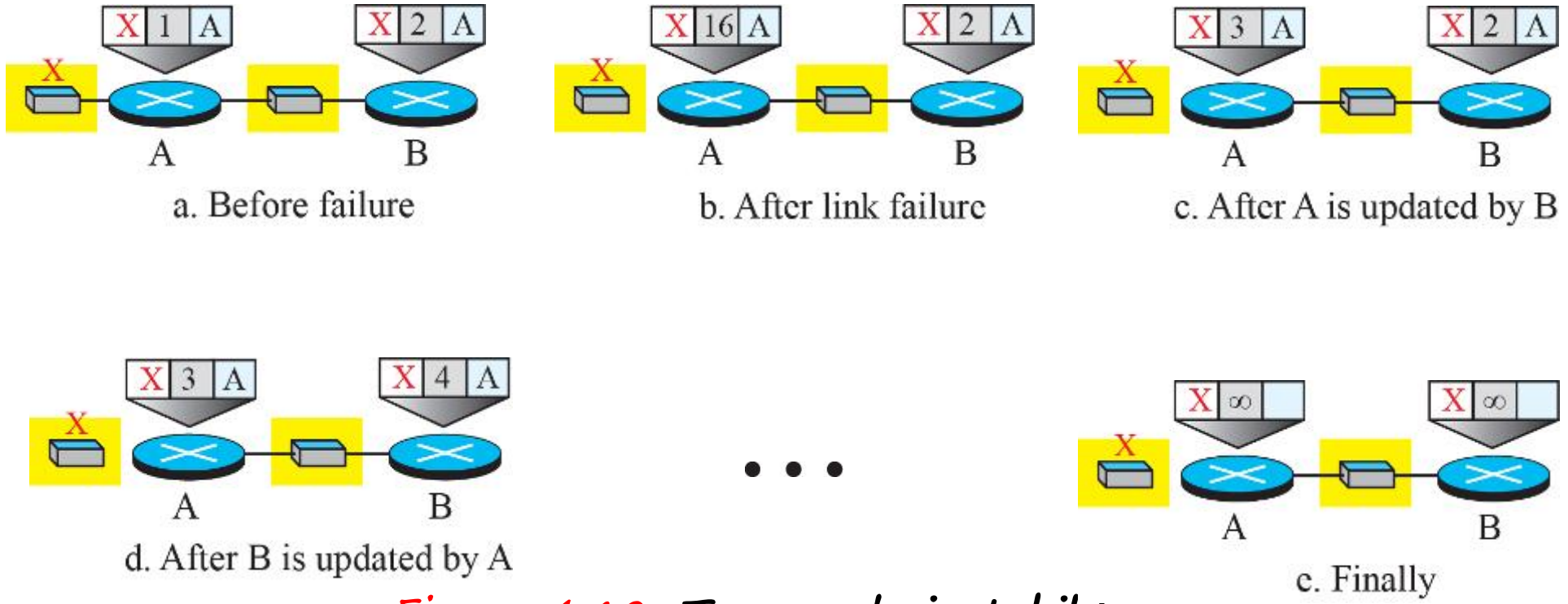


Figure 4.62: Two-node instability

Solutions:

- **Defining Infinity:** Redefine infinity to a smaller number. In maximum systems the size of network in each direction can not exceed 15 hops.
- **Split Horizon:** split-horizon route advertisement is a method of preventing routing loops in distance-vector routing protocols by prohibiting a router from advertising a route back onto the interface from which it was learned.
- **Poison Reverse:** a poison reverse is a way in which a gateway node tells its neighbor gateways that one of the gateways is no longer connected.

Routing Information Protocol (RIP):

- The Routing Information Protocol (RIP) is one of the most widely used intradomain routing protocols based on the distance-vector routing algorithm.
- In RIP, the maximum cost of a path can be 15, which means 16 is considered as infinity (no connection). For this reason, RIP can be used only in autonomous systems in which the diameter of the AS is not more than 15 hops.

Figure 4.70: Hop counts in RIP

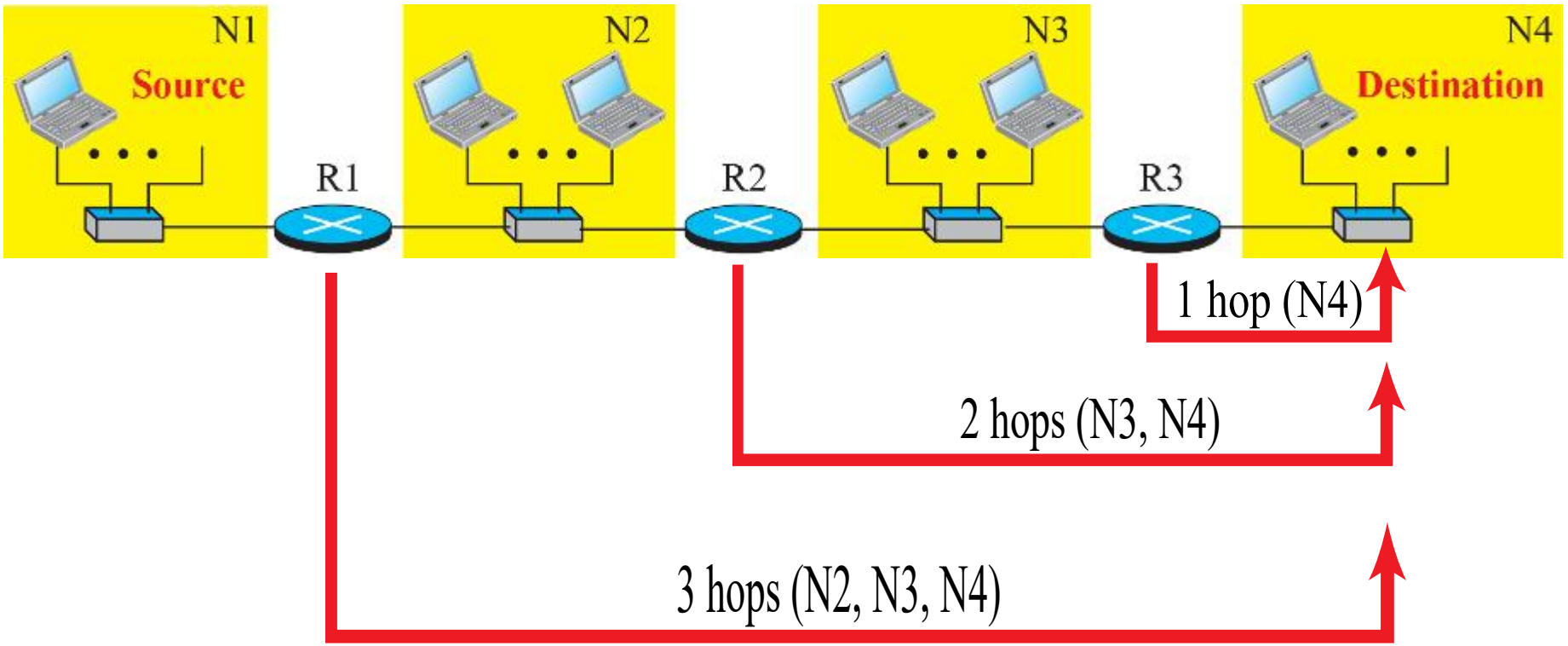
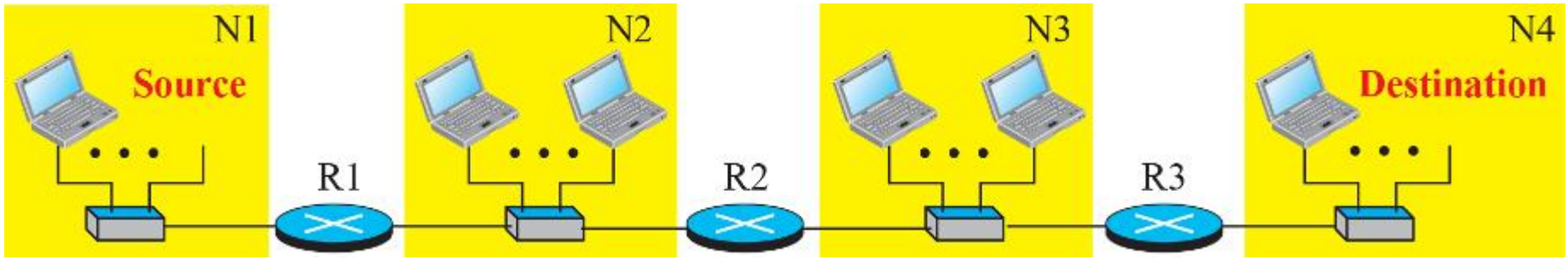


Figure 4.71: Forwarding tables



Forwarding table for R1

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

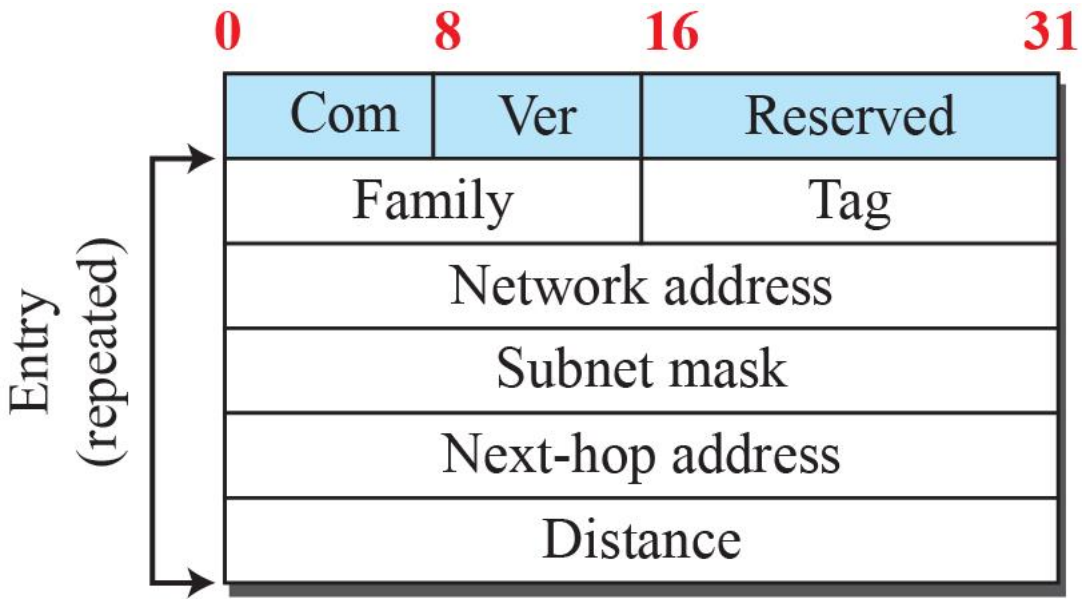
Forwarding table for R2

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

Figure 4.72: RIP message format



Fields

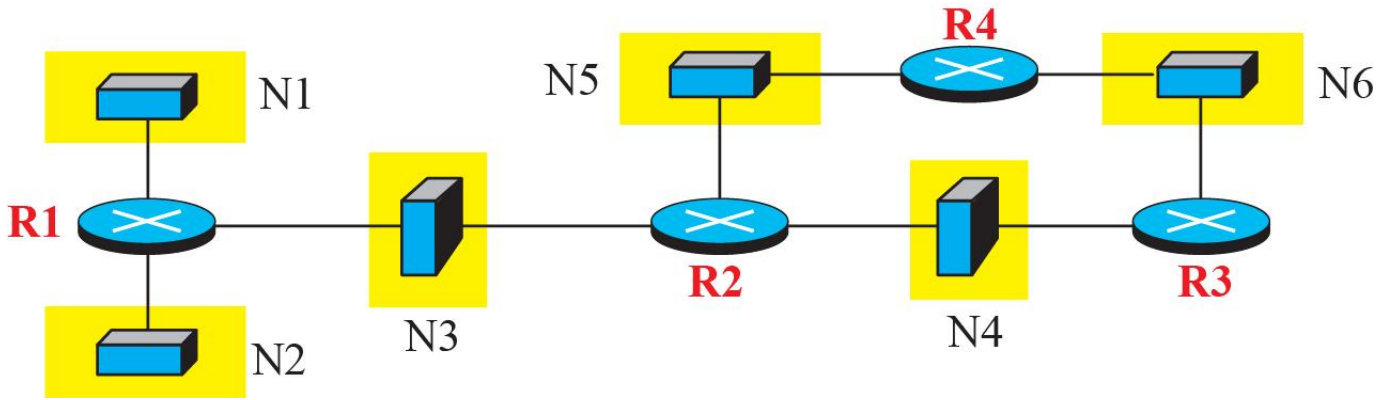
Com: Command, request (1), response (2)
Ver: Version, current version is 2
Family: Family of protocol, for TCP/IP value is 2
Tag: Information about autonomous system
Network address: Destination address
Subnet mask: Prefix length
Next-hop address: Address length
Distance: Number of hops to the destination

- 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.

Example 4.15

Figure 4.73 shows a more realistic example of the operation of RIP in an autonomous system. First, the figure shows all forwarding tables after all routers have been booted. Then we show changes in some tables when some update messages have been exchanged. Finally, we show the stabilized forwarding tables when there is no more change.

Figure 4.73: Example of an autonomous system using RIP (Part I)



Legend

Des.: Destination network
N. R.: Next router
Cost: Cost in hops

R1

Des.	N. R.	Cost
N1	———	1
N2	———	1
N3	———	1

R2

Des.	N. R.	Cost
N3	———	1
N4	———	1
N5	———	1

R3

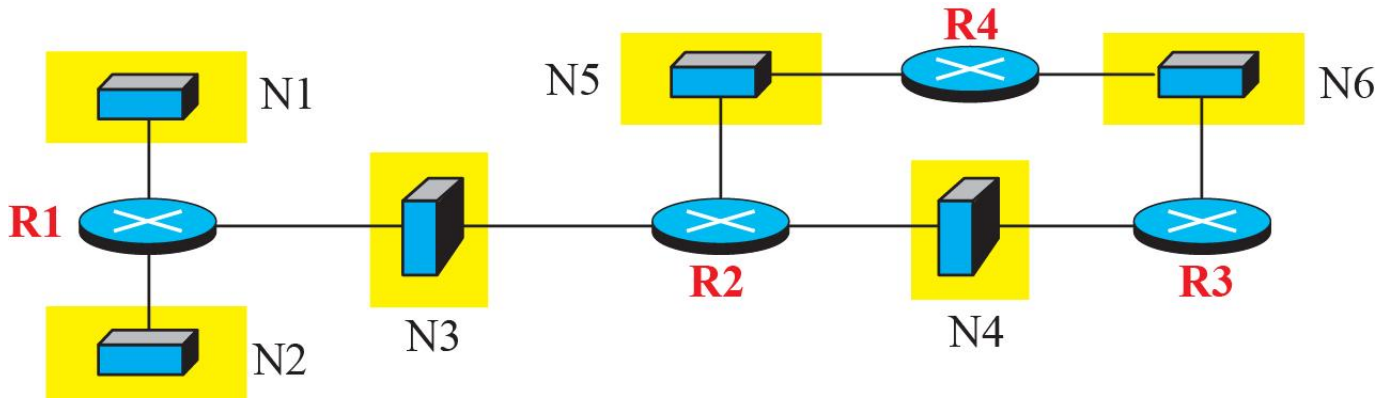
Des.	N. R.	Cost
N4	———	1
N6	———	1

R4

Des.	N. R.	Cost
N5	———	1
N6	———	1

Forwarding tables after all routers booted

Figure 4.73: Example of an autonomous system using RIP (Part II)



Legend

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

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

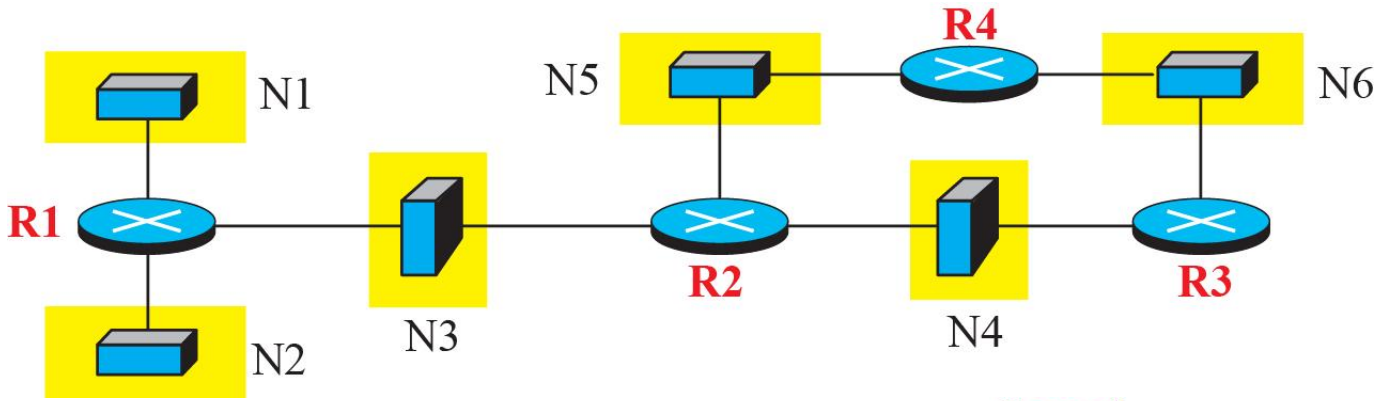
Des.	N. R.	Cost
N5	——	1
N6	——	1

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

Figure 4.73: Example of an autonomous system using RIP (Part III)



Legend

Des.: Destination network
N. R.: Next router
Cost: Cost in hops

Forwarding tables for all routers after they have been stablized

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

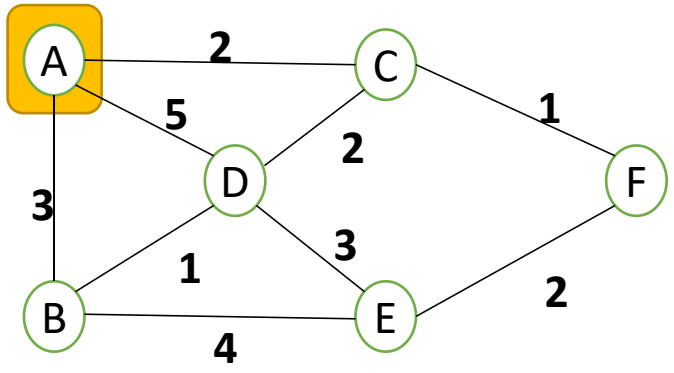
Des.	N. R.	Cost
N1	R2	3
N2	R2	3
N3	R2	2
N4	R2	2
N5	———	1
N6	———	1

Link State Routing:

The basic concept of link-state routing is that every node constructs a map of the connectivity to the network, in the form of a graph, showing which nodes are connected to which other nodes. Each node then independently calculates the next best logical path from it to every possible destination in the network.

Applying **Dijkstra Algorithm** determine the shortest path from node A to rest all nodes in the network.

Solution: The node having minimum cost will become a member of visited queue in every iteration.



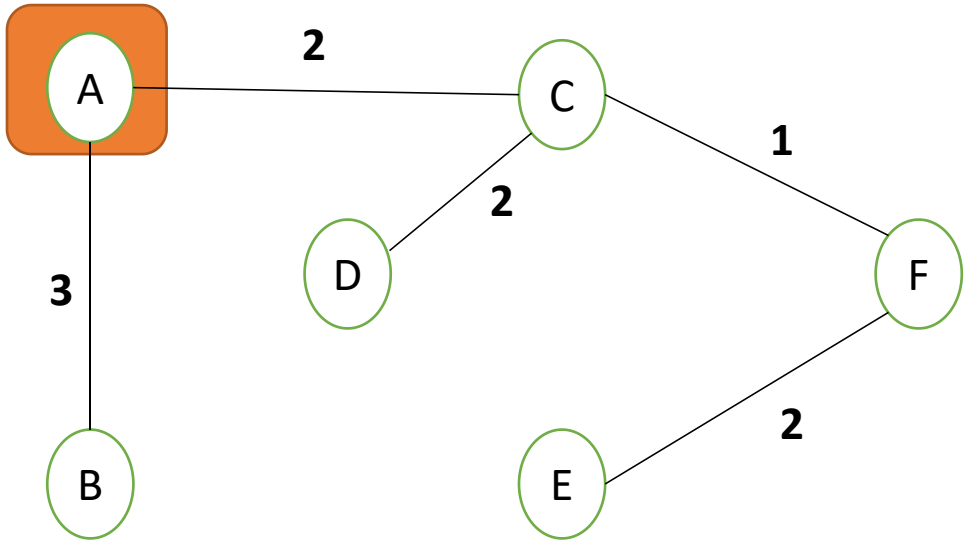
ROUTING TABLE AT NODE A:

Dest. Node	Next node	Cost
B	B	3
C	C	2
D	C	4
E	C	5
F	C	3

Iteration	Visited Nodes	NodeB	NodeC	NodeD	NodeE	NodeF
Initial	{-}	∞	∞	∞	∞	∞
1st	{A}	3 (A-B)	**2 (A-C)**	5 (A-D)	∞	∞
2nd	{A,C}	**3 (A-B)**	**2 (A-C)**	4 (A-C-D)	∞	3 (A-C-F)
3rd	{A,B,C}	**3 (A-B)**		4 (A-C-D)	7 (A-B-E)	**3 (A-C-F)**
4th	{A,B,C,F}			**4 (A-C-D)**	5 (A-C-F-E)	**3 (A-C-F)**
5th	{A,B,C,D,F}			**4 (A-C-D)**	**5 (A-C-F-E)**	
6th	{A,B,C,D,E,F}	**3 (A-B)**	**2 (A-C)**	**4 (A-C-D)**	**5 (A-C-F-E)**	**3 (A-C-F)**

ROUTING TABLE AT NODE A:

Dest. Node	Next node	Cost
A	-	0
B	B	3
C	C	2
D	C	4
E	C	5
F	C	3



Iteration	Visited Nodes	NodeB	NodeC	NodeD	NodeE	NodeF
6th	{A,B,C,D,E,F}	3 (A-B)	2 (A-C)	4 (A-C-D)	5 (A-C-F-E)	3 (A-C-F)

Table 4.5: Dijkstra's Algorithm

```
1  Dijkstra's Algorithm ( )
2  {
3      // Initialization
4      Tree = {root}                    // Tree is made only of the root
5      for (y = 1 to N)                 // N is the number of nodes
6      {
7          if (y is the root)
8              D[y] = 0                 // D[y] is shortest distance from root to node y
9          else if (y is a neighbor)
10             D[y] = c[root][y]        // c[x][y] is cost between nodes x and y in LSDB
11         else
12             D[y] = ∞
13     }
14     // Calculation
15     repeat
16     {
17         find a node w, with D[w] minimum among all nodes not in the Tree
18         Tree = Tree ∪ {w}            // Add w to tree
19         // Update distances for all neighbor of w
20         for (every node x, which is neighbor of w and not in the Tree)
21         {
22             D[x] = min{D[x], (D[w] + c[w][x])}
23         }
24     } until (all nodes included in the Tree)
25 } // End of Dijkstra
```