DRONACHARYA College of Engineering

Computer Science & Engineering

Data Communication and Computer Networks

(MTCSE-101-A)

22-3 UNICAST ROUTING PROTOCOLS

A routing table can be either static or dynamic. A static table is one with manual entries. A dynamic table is one that is updated automatically when there is a change somewhere in the Internet. A routing protocol is a combination of rules and procedures that lets routers in the Internet inform each other of changes.

Topics discussed in this section:

Optimization
Intra- and Interdomain Routing
Distance Vector Routing and RIP
Link State Routing and OSPF
Path Vector Routing and BGP

Figure 22.12 *Autonomous systems*

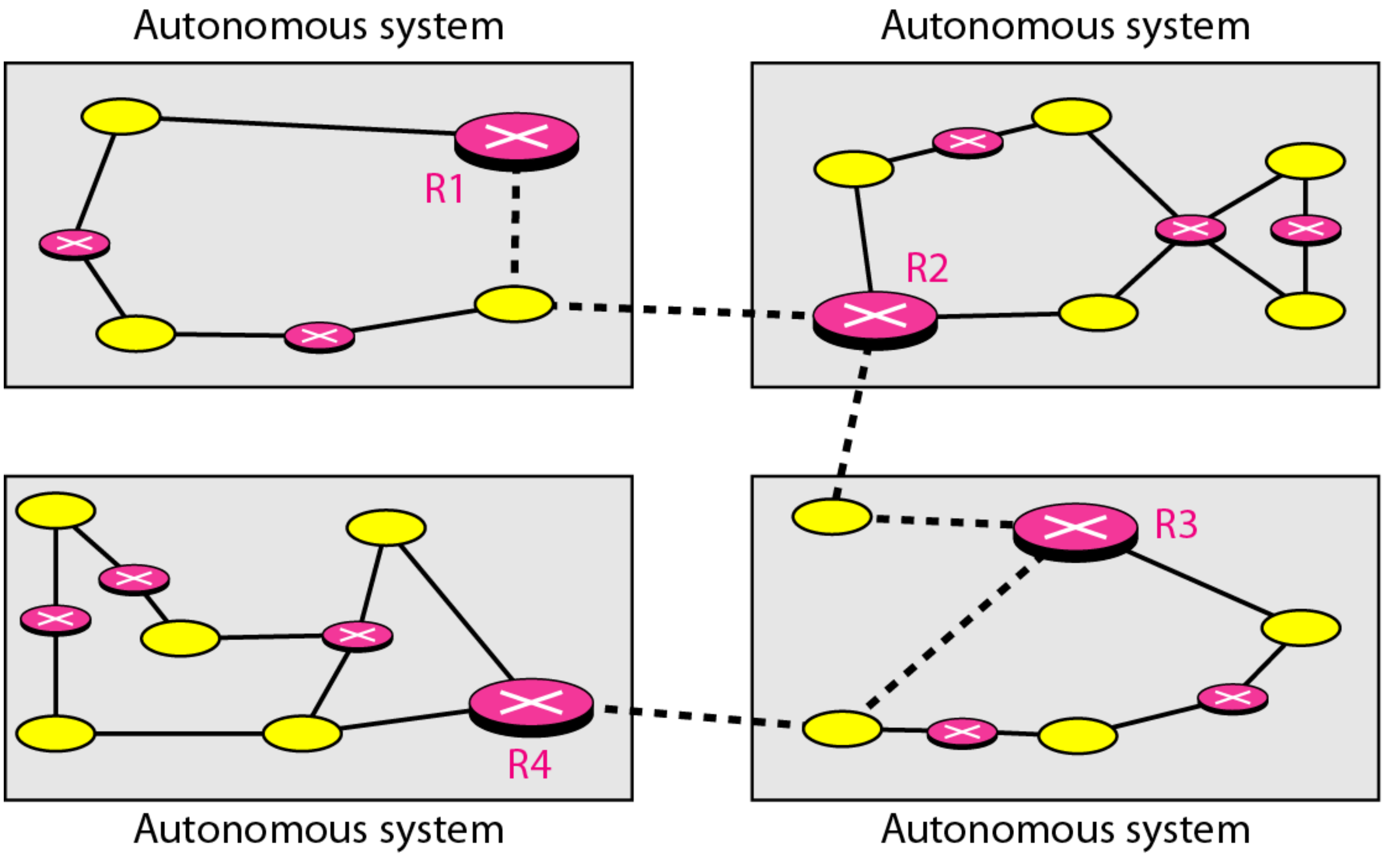
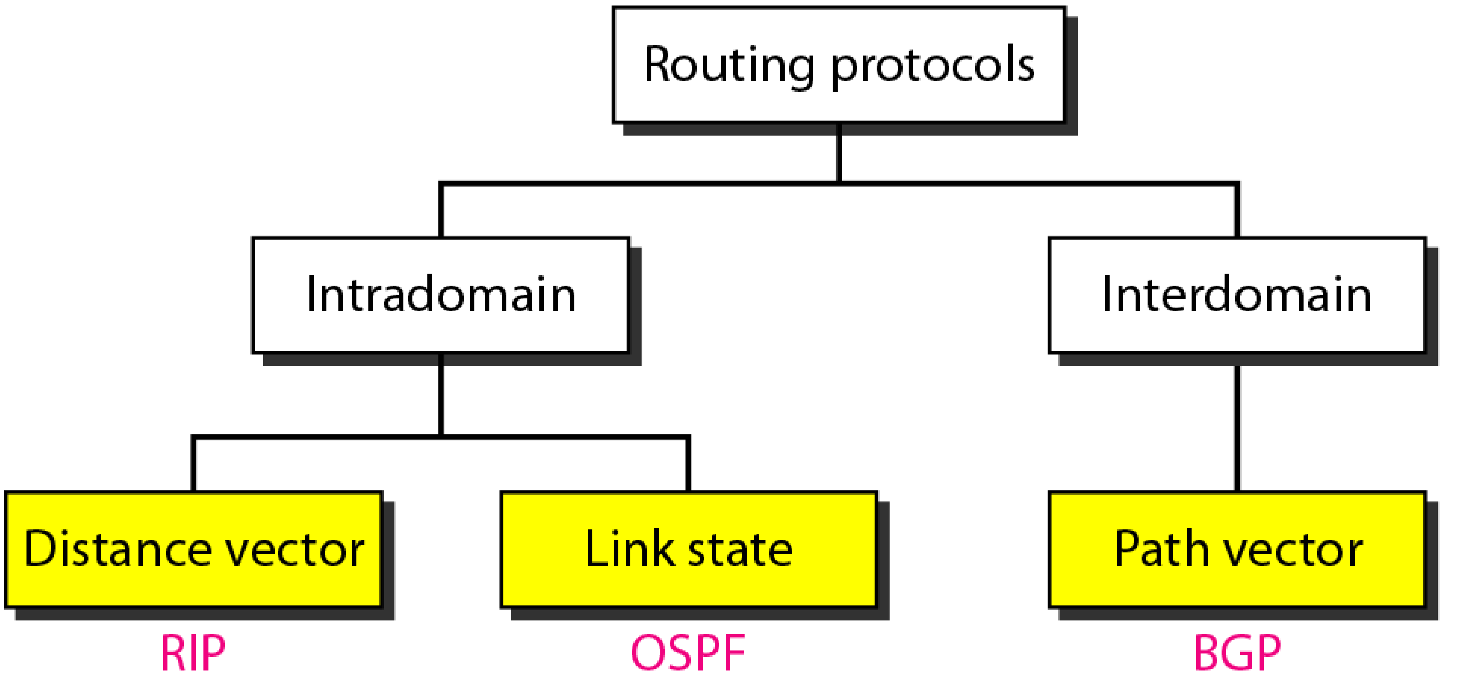
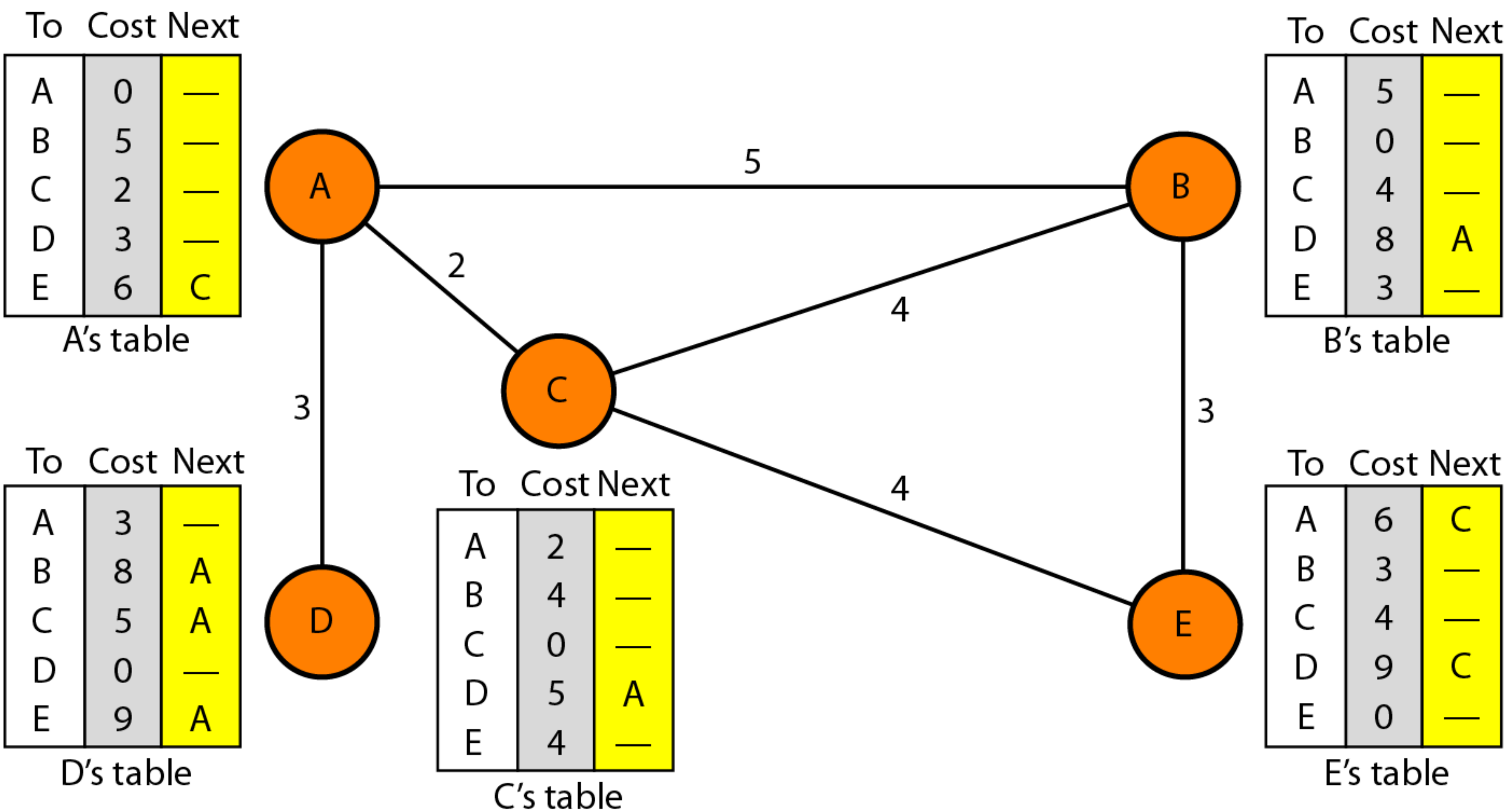


Figure 22.13 *Popular routing protocols*



In distance vector routing, the least-cost route between any two nodes is the route with minimum distance.
In this protocol, as the name implies, each node maintains a vector (table) of minimum distances to every node. The table at each node also guides the packets to the desired node by showing the next stop in the route (next-hop routing).

Figure 22.14 *Distance vector routing tables*



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

A's table

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

B's table

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

C's table

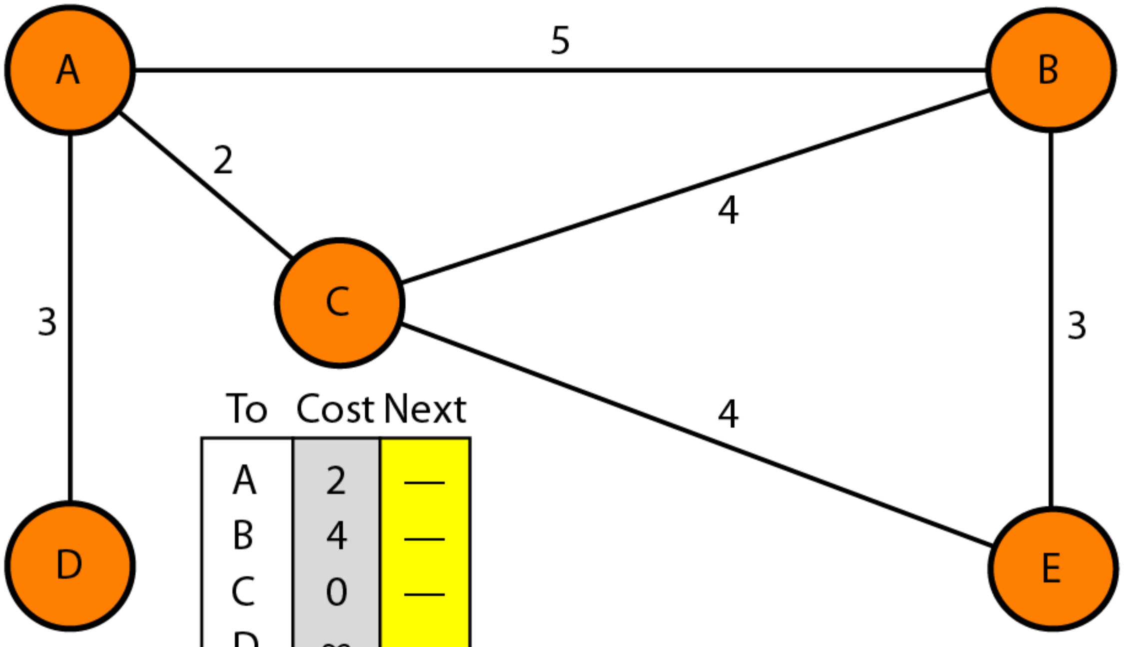
To	Cost	Next
A	3	—
B	8	A
C	5	A
D	0	—
E	9	A

D's table

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

E's table

Figure 22.15 *Initialization of tables in distance vector routing*



A's table

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

B's table

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

D's table

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

C's table

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

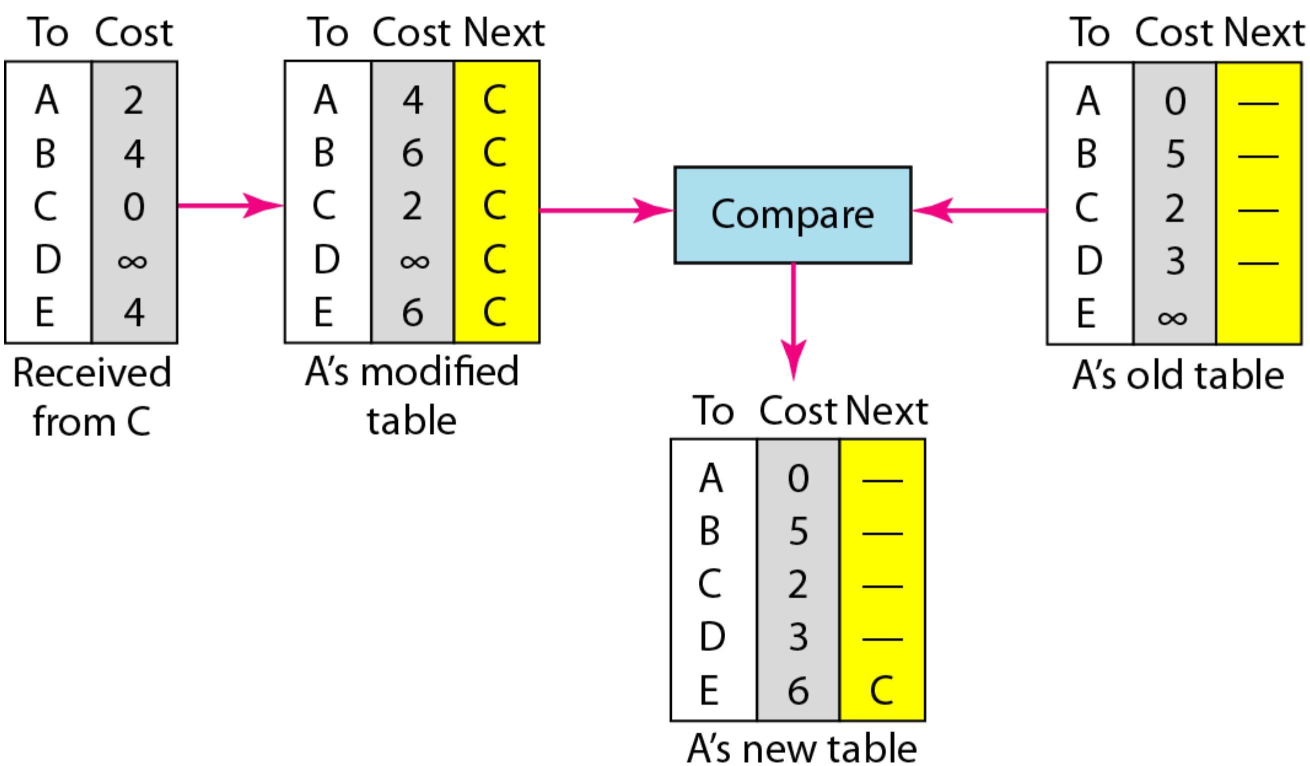
E's table

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

Note

In distance vector routing, each node shares its routing table with its immediate neighbors periodically and when there is a change.

Figure 22.16 *Updating in distance vector routing*



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

RIP implements distance vector routing directly with some considerations:

1. In an autonomous system, we are dealing with routers and networks (links). The routers have routing tables; networks do not.

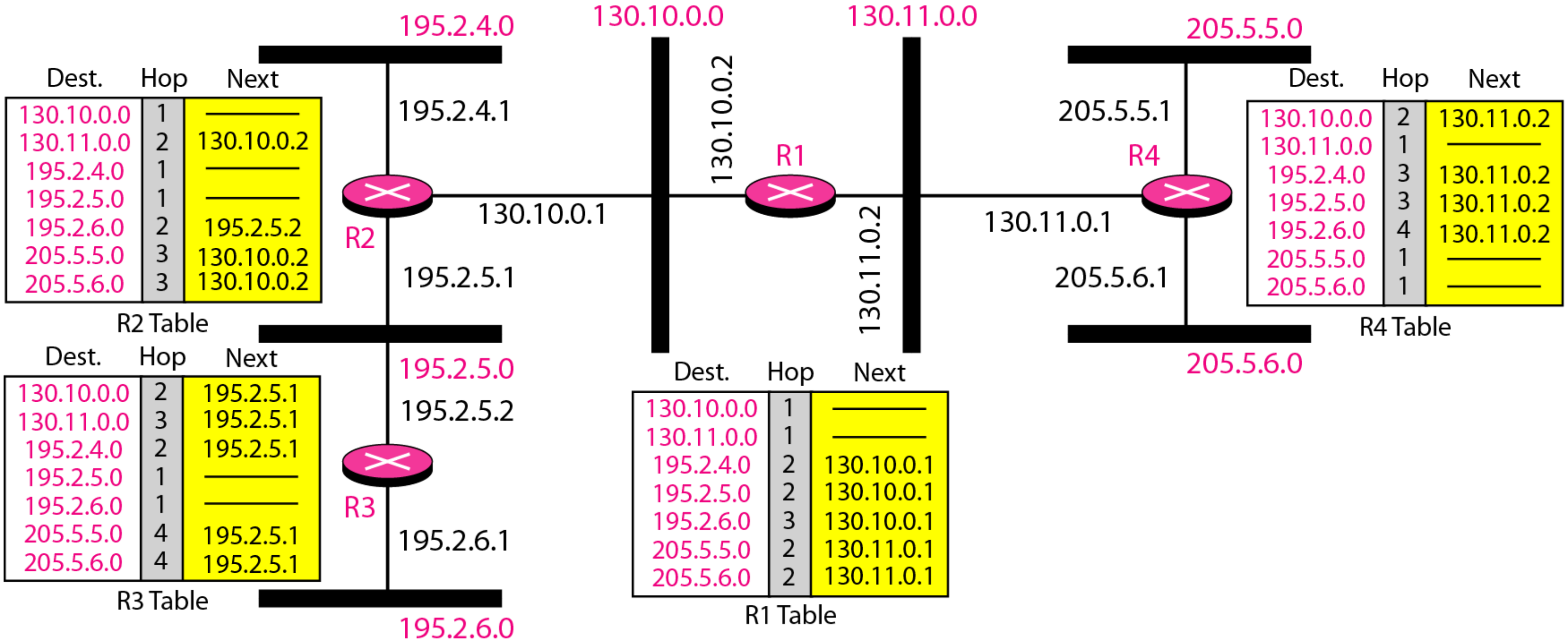
2. The destination in a routing table is a network, which means the first column defines a network address.

3. The metric used by RIP is very simple; the distance is defined as the number of links (networks) to reach the destination. For this reason, the metric in RIP is called a hop count.

4. Infinity is defined as 16, which means that any route in an autonomous system using RIP cannot have more than 15 hops.

5. The next-node column defines the address of the router to which the packet is to besent to reach its destination.

Figure 22.19 *Example of a domain using RIP*



R2 Table

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

R3 Table

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

R1 Table

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

R4 Table

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

Link State Routing

Link state routing **has a different philosophy from that of distance vector routing.**

In link state routing, if each node in the domain has the entire topology of the domain –the list of nodes and links, how they are connected including the type, cost (metric), and condition of the links (up or down)-the node can use Dijkstra's algorithm to build a routing table.

Figure 22.20 *Concept of link state routing*

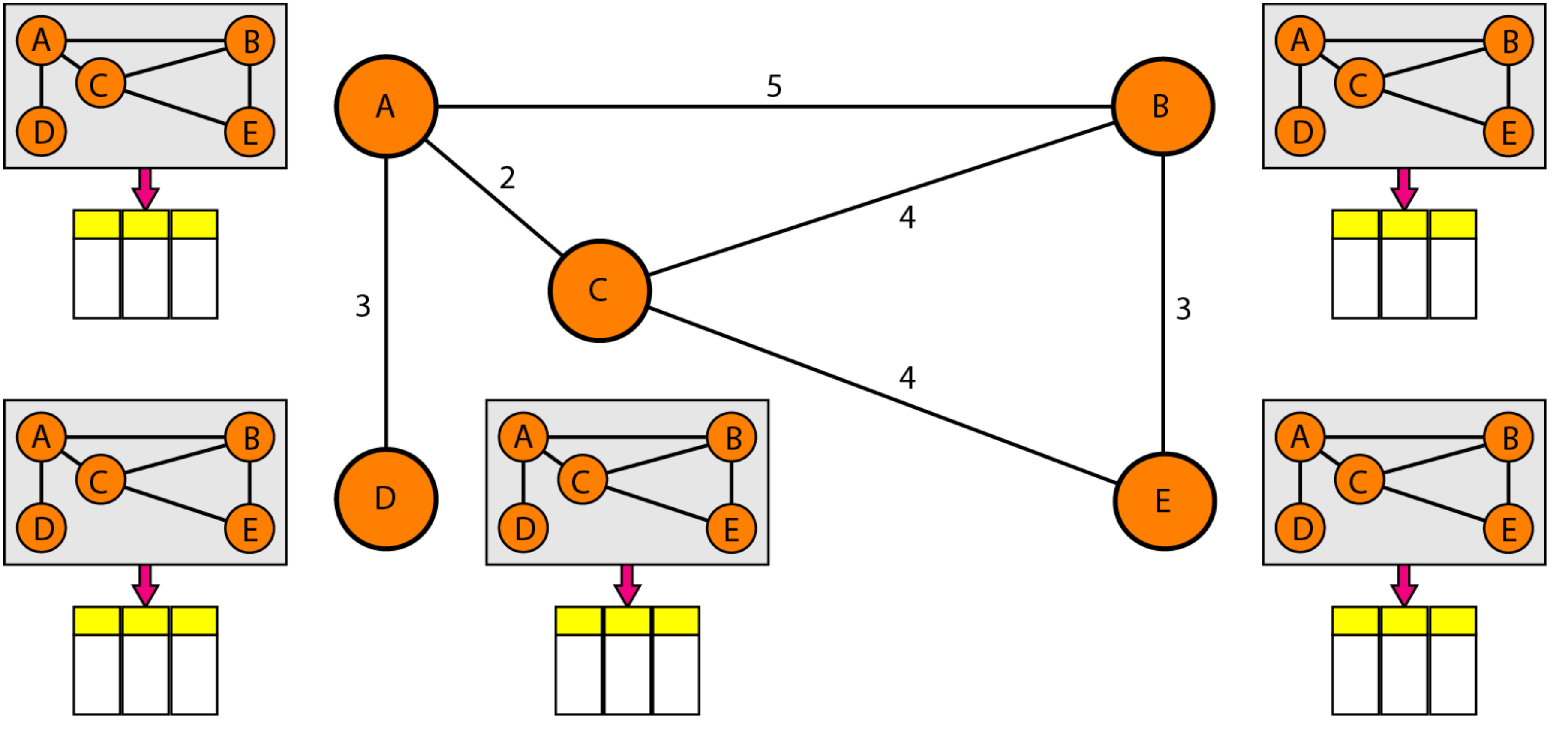
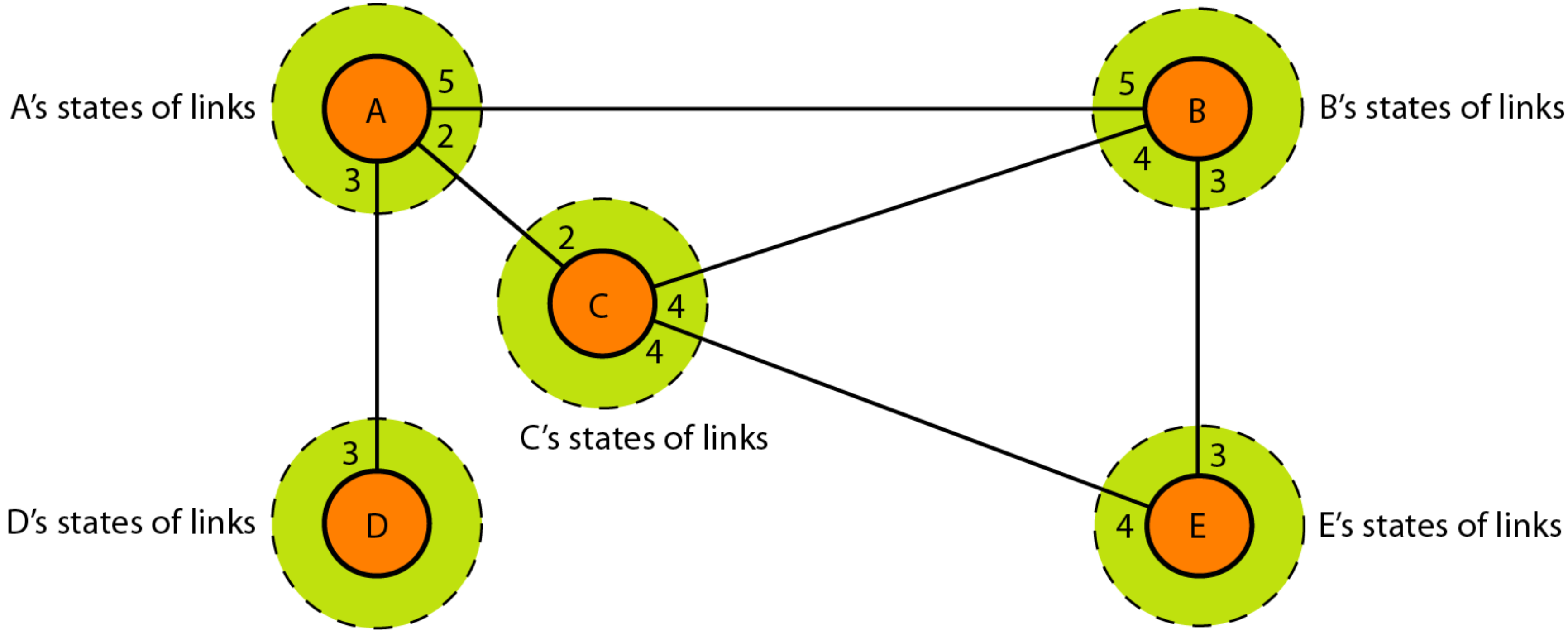


Figure 22.21 *Link state knowledge*



Building Routing Tables

In link state routing, four sets of actions are required to ensure that each node has the routing table showing the least-cost node to every other node.

1. Creation of the states of the links by each node, called the link state packet (LSP).
2. Dissemination of LSPs to every other router, called flooding, in an efficient and reliable way.
3. Formation of a shortest path tree for each node.
4. Calculation of a routing table based on the shortest path tree.

Creation of Link State Packet (LSP):

A link state packet can carry a large amount of information. For the moment, however, we assume that it carries a minimum amount of data: the node identity, the list of links, a sequence number, and age.
The first two, node identity and the list of links, are needed to make the topology. The third, sequence number, facilitates flooding and distinguishes new LSPs from old ones. The fourth, age, prevents old LSPs from remaining in the domain for a long time.

Flooding of LSPs:

After a node has prepared an LSP, it must be disseminated to all other nodes, not only to its neighbors. The process is called flooding and based on the Following:

1.The creating node sends a copy of the LSP out of each interface.

2. A node that receives an LSP compares it 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, the node does the following:

a. It discards the old LSP and keeps the new one.

b. It 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 domain (where a node has only one interface).

Formation of Shortest Path Tree:

Dijkstra Algorithm: After receiving all LSPs, each node will have a copy of the whole topology. However, the topology is not sufficient to find the shortest path to every other node; a shortest path tree is needed.

The Dijkstra algorithm creates a shortest path tree from a graph. The algorithm divides the nodes into two sets: tentative and permanent. It finds the neighbors of a current node, makes them tentative, examines them, and if they pass the criteria, makes them permanent.

Figure 22.22 *Dijkstra algorithm*

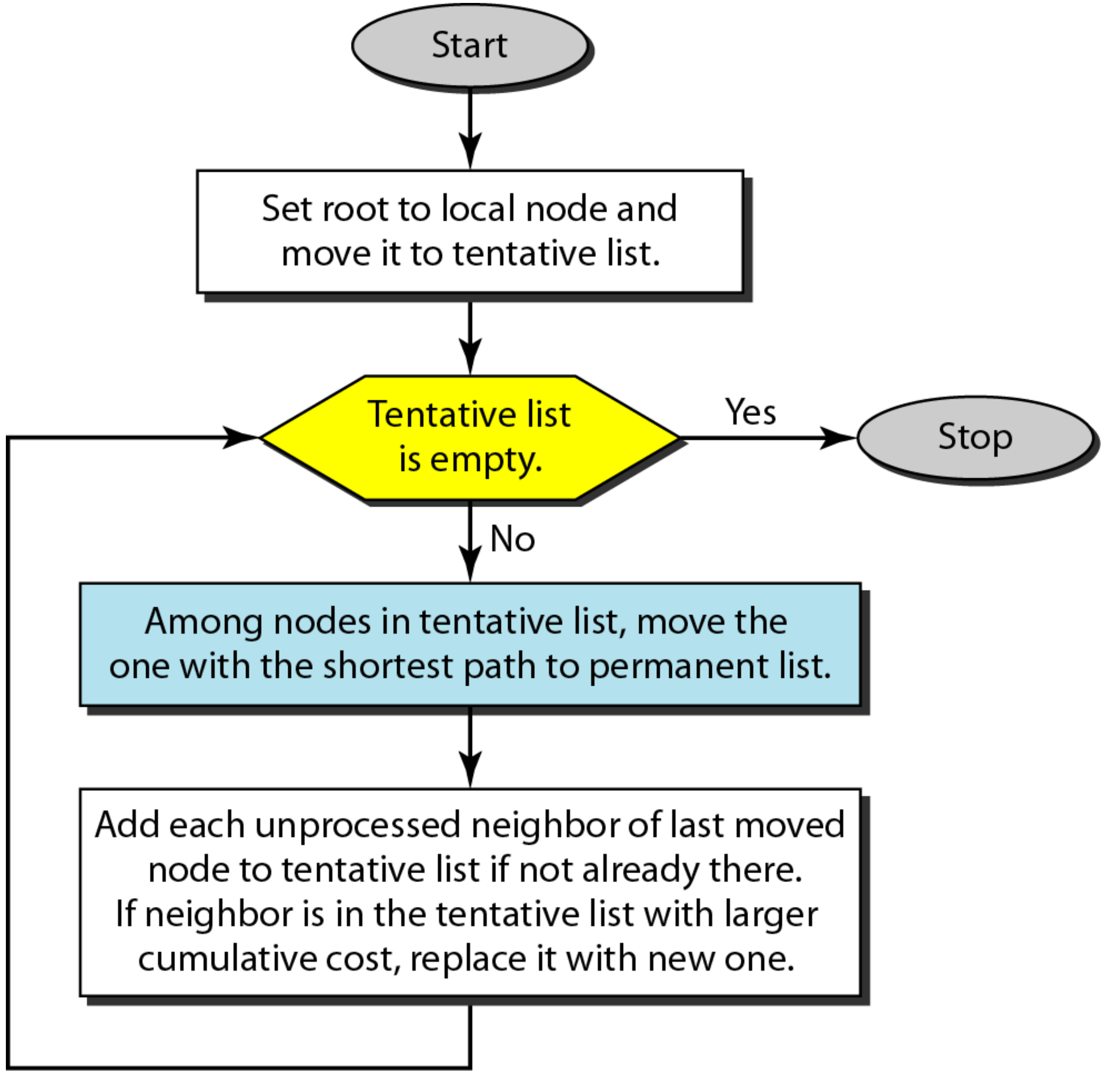
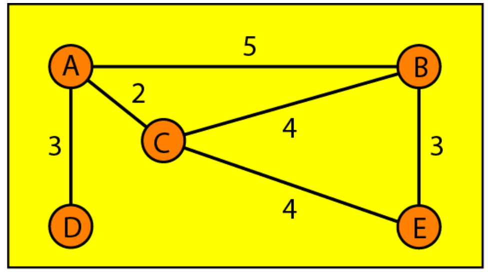
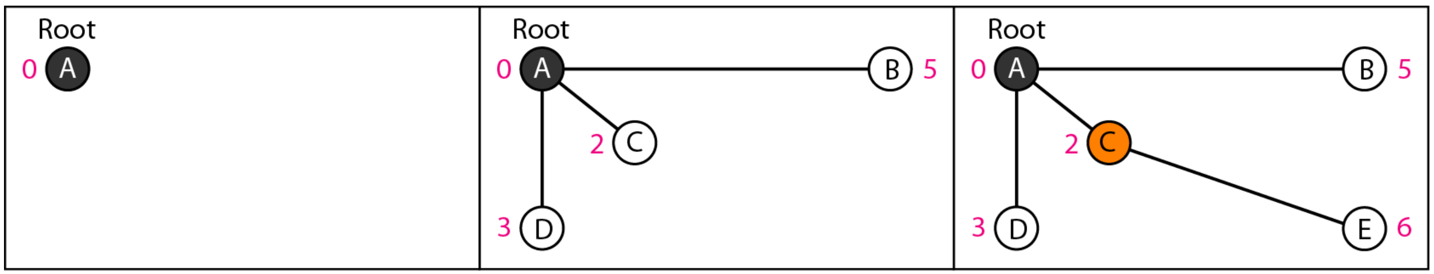


Figure 22.23 *Example of formation of shortest path tree*



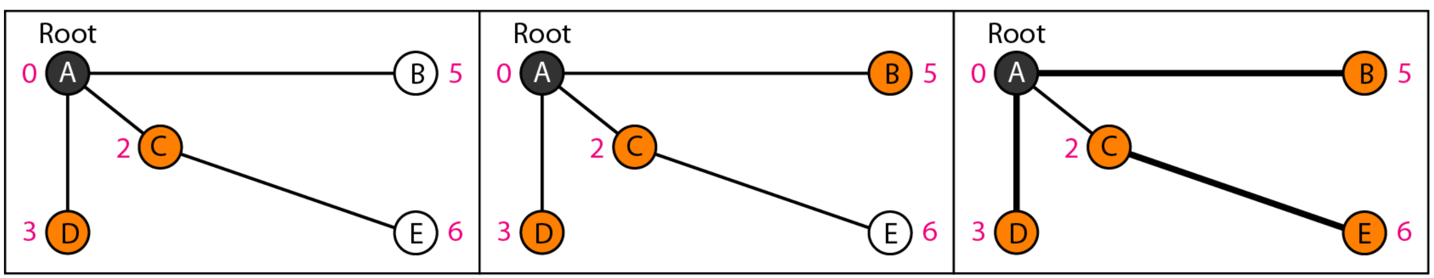
Topology



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

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

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



4. Move D to permanent list.

5. Move B to permanent list.

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

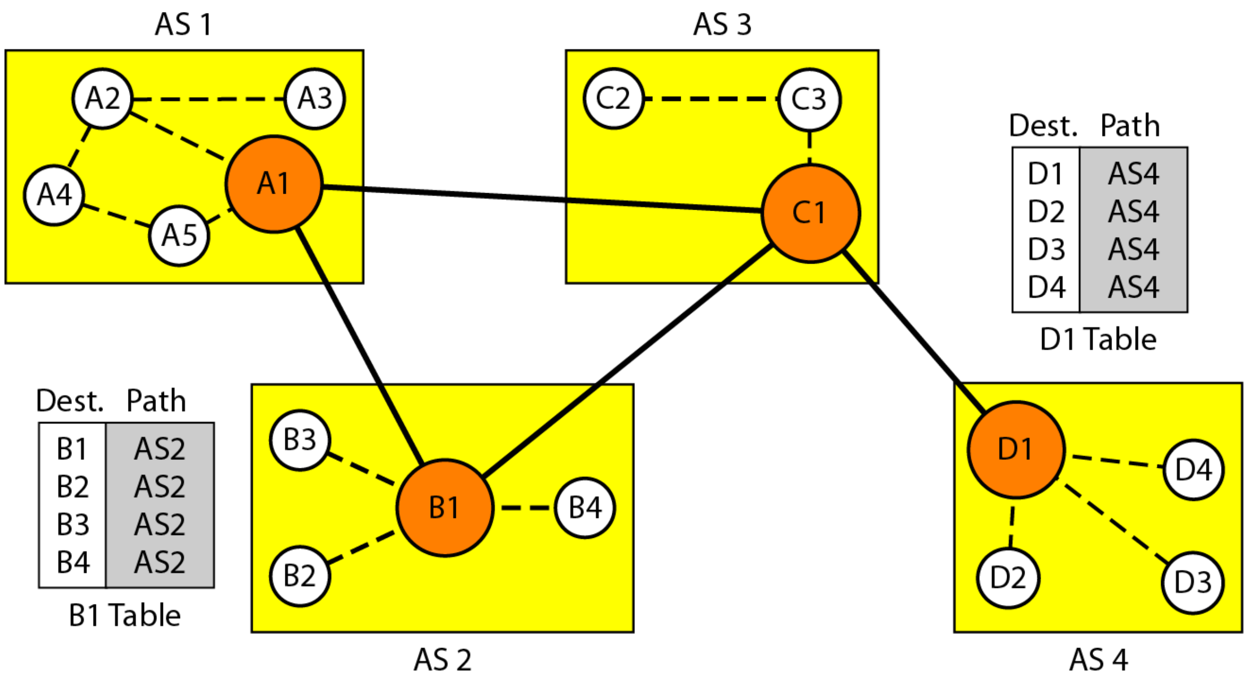
Path Vector Routing

Path vector routing proved to be useful for interdomain routing. The principle of path vector routing is similar to that of distance vector routing.

In path vector routing, we assume that there is one node (there can be more, but one is enough for our conceptual discussion) in each autonomous system that acts on behalf of the entire autonomous system. Let us call it the speaker node.
The speaker node in an AS creates a routing table and advertises it to speaker nodes in the neighboring ASs. The idea is the same as for distance vector routing except that only speaker nodes in each AS can communicate with each other. However, what is advertised is different. A speaker node advertises the path, not the metric of the nodes, in its autonomous system or other autonomous systems.

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

Figure 22.30 *Initial routing tables in path vector routing*



A1 Table

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

C1 Table

Dest.	Path
C1	AS3
C2	AS3
C3	AS3

D1 Table

Dest.	Path
D1	AS4
D2	AS4
D3	AS4
D4	AS4

B1 Table

Dest.	Path
B1	AS2
B2	AS2
B3	AS2
B4	AS2

Figure 22.31 *Stabilized tables for three autonomous systems*

A1 Table

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

B1 Table

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

C1 Table

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

D1 Table

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