# 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.*

# Figure *Autonomous systems*



Autonomous system

Autonomous system

R1

R2

R3

R4

Autonomous system

Autonomous system
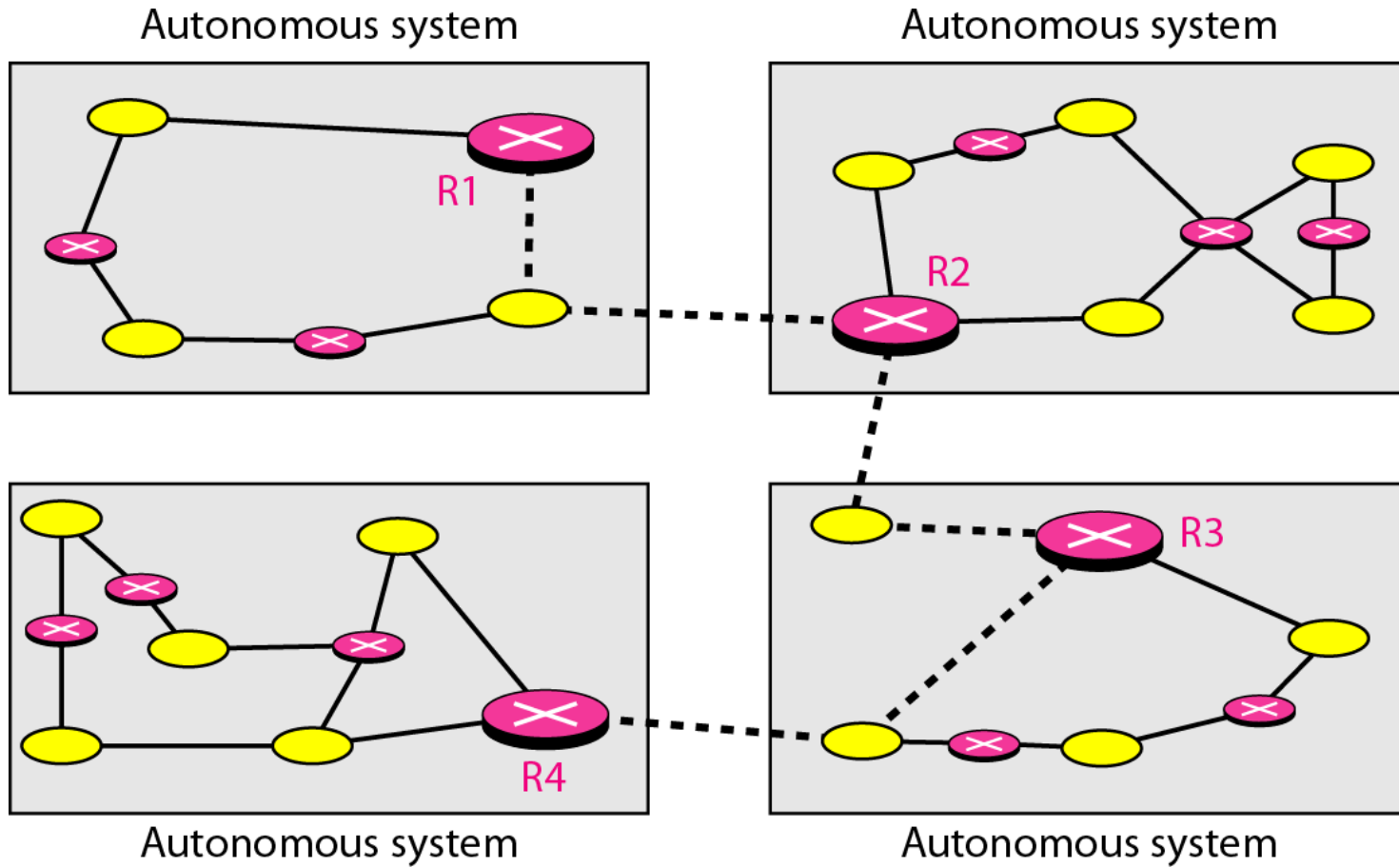
- When it receives a packet, to which network should it pass the packet?

- The decision is based on optimization: Which of the available pathways is the optimum pathway?

- One approach is to assign a cost for passing through a network. We call this cost a metric.

- However, the metric assigned to each network depends on the type of protocol.

- Some simple protocols, such as the Routing Information Protocol (RIP), treat all networks as equals. The cost of passing through a network is the same; it is one hop count.
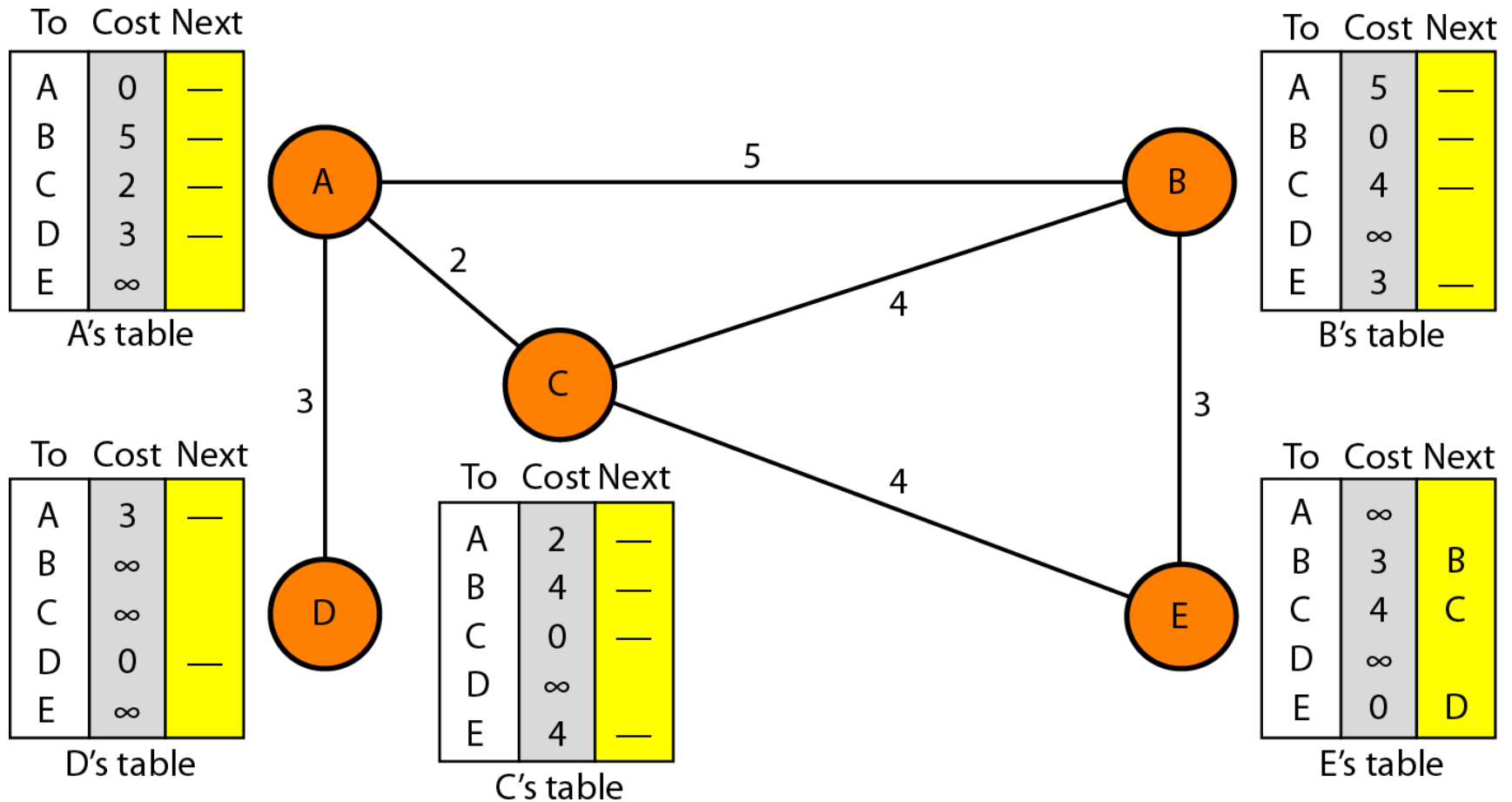
- So if a packet passes through 10 networks to reach the destination, the total cost is 10 hop counts

- Another method can be type of service

# Figure *Popular routing protocols*
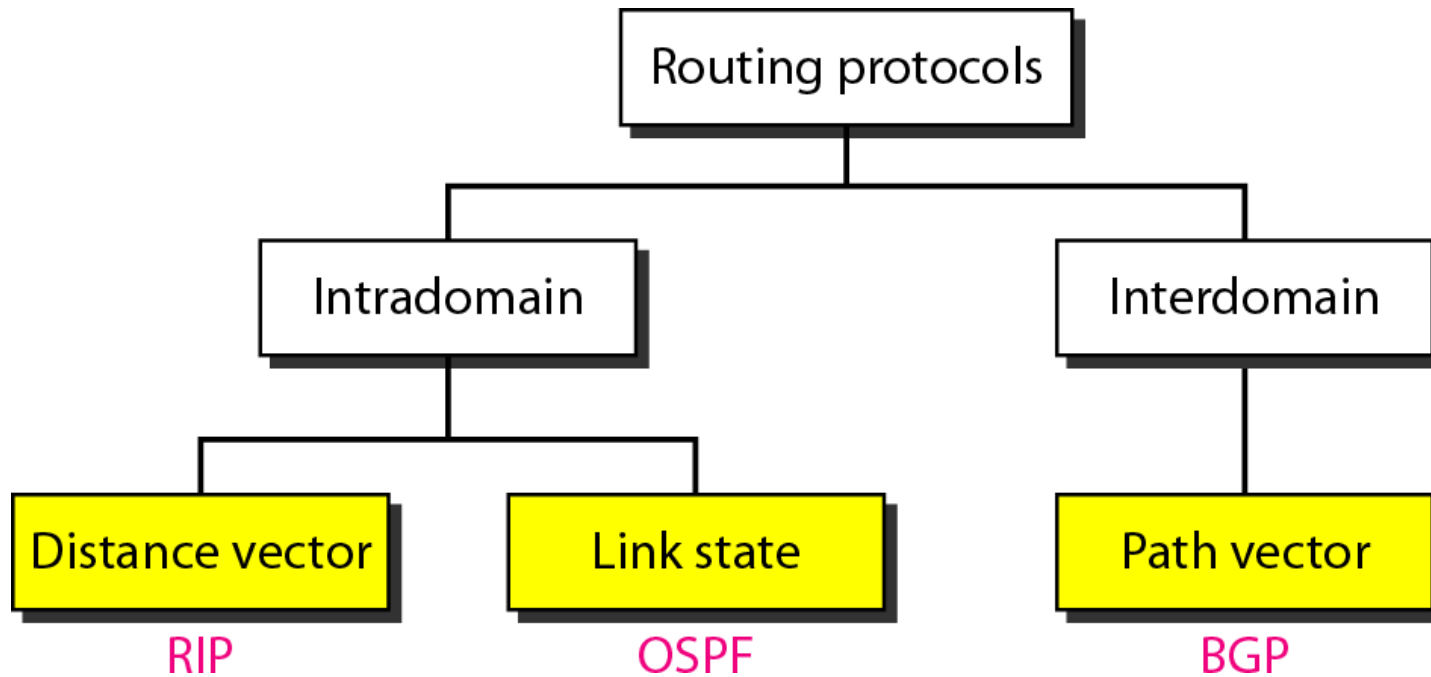
- In distance vector routing, the least-cost route between any two nodes is the route with minimum distance.

- 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 : *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

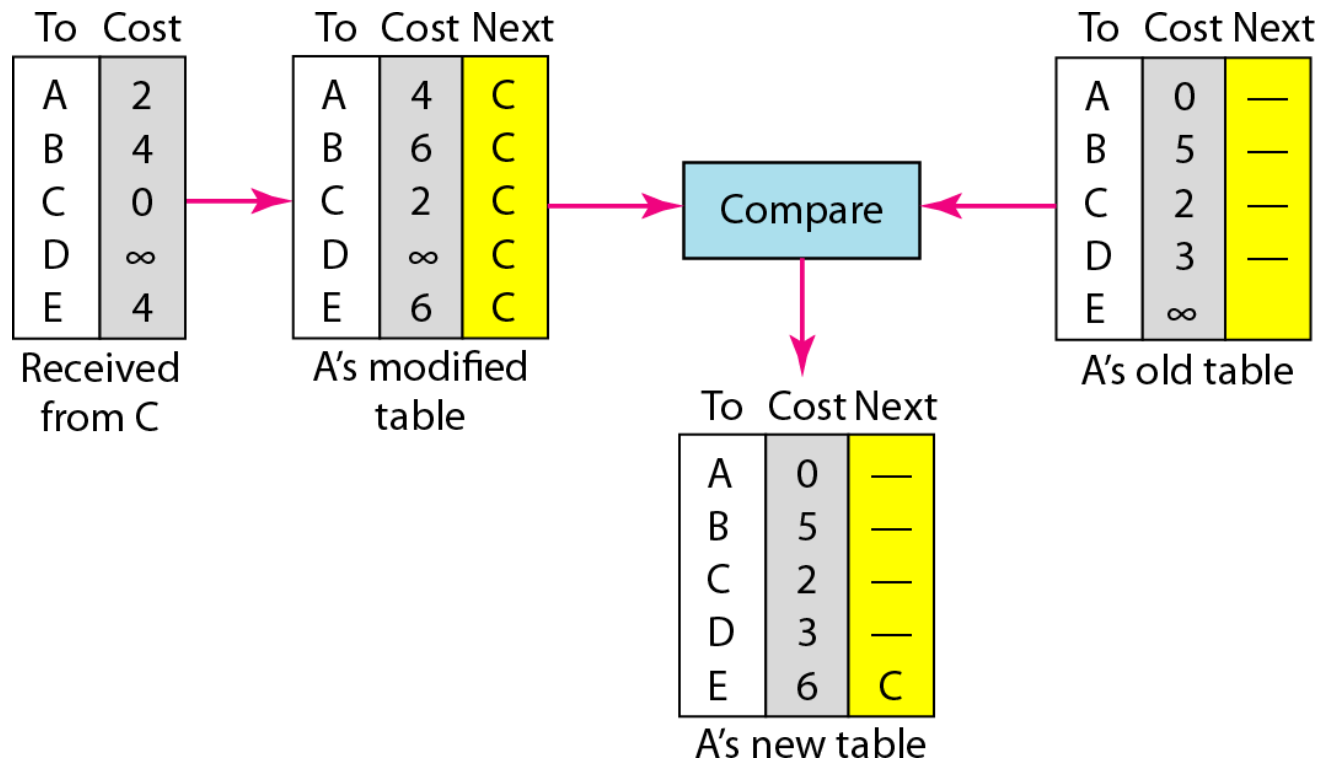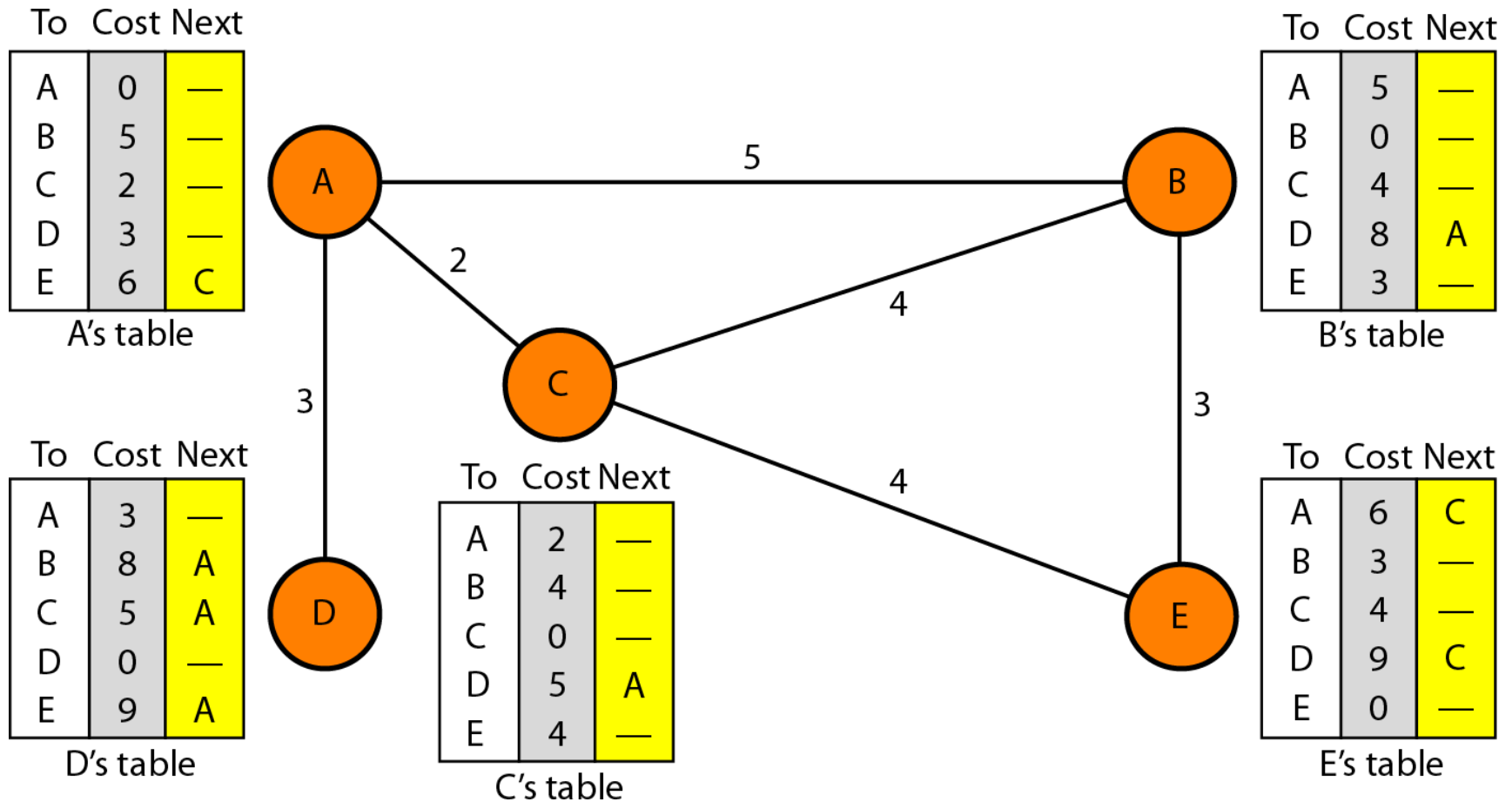| To | Cost | Next |
|----|------|------|
| A | ∞ | |
| B | 3 | B |
| C | 4 | C |
| D | ∞ | |
| E | 0 | D |

*Note*

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

**Figure** *Updating in distance vector routing*

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

Received from C

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

A's modified table

Compare

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

A's old table

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

A's new table

# **Figure** *Distance vector routing tables*



**A's table**

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

**B's table**

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

**D's table**

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

**C's table**

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

**E's table**

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

# When does a node send its partial routing table (only two columns) to all its immediate neighbors?

**Periodic Update:** A node sends its routing table, normally every 30 s, in a periodic update. The period depends on the protocol that is using distance vector routing.

**Triggered Update:** A node sends its two-column routing table to its neighbors anytime there is a change in its routing table. This is called a triggered update.

The change can result from the following:

1.A node receives a table from a neighbor, resulting in changes in its own table after updating.

2.A node detects some failure in the neighboring links which results in a distance change to infinity.

# Figure *Two-node instability*
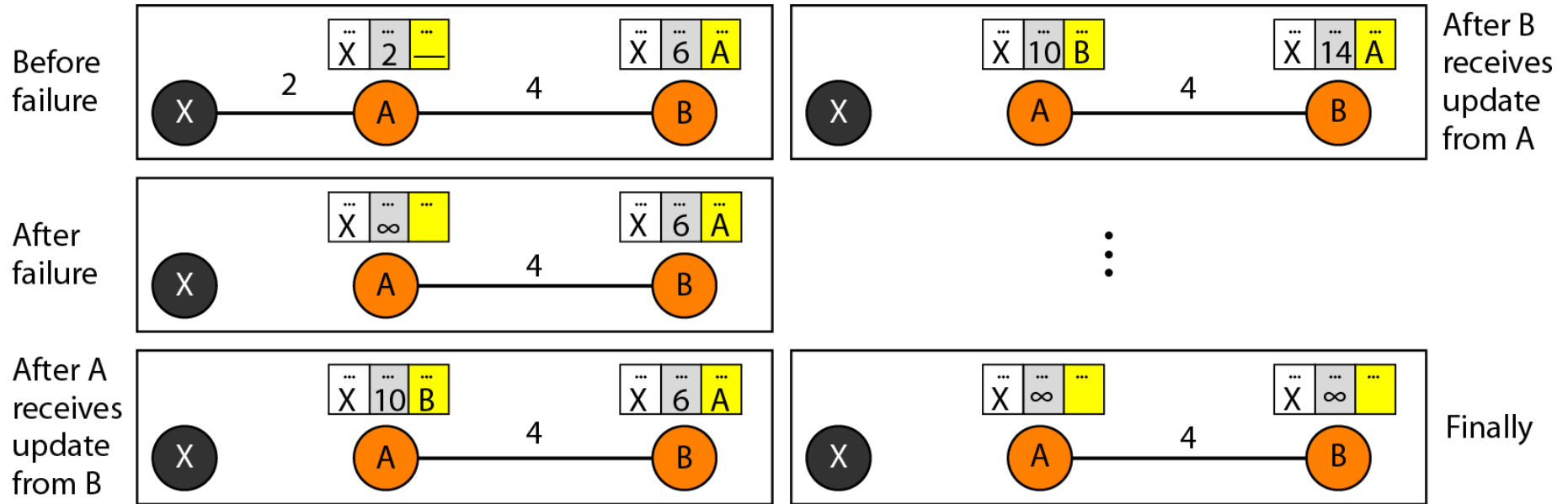
# Defining Infinity

- The first obvious solution is to redefine infinity to a smaller number, such as 100.

- For our previous scenario, the system will be stable in less than 20 updates. As a matter of fact, most implementations of the distance vector protocol define the distance between each node to be I and define 16 as infinity.

- However, this means that the distance vector routing cannot be used in large systems.

- The size of the network, in each direction, can not exceed 15 hops.

# Split Horizon

- If node B thinks that the optimum route to reach X is via A, it does not need to advertise this piece of information to A; the information has come from A (A already knows).

- Taking information from node A, modifying it, and sending it back to node A creates the confusion.

# Split Horizon and Poison Reverse

- Normally, the distance vector protocol uses a timer, and if there is no news about a route, the node deletes the route from its table.

- When node B in the previous scenario eliminates the route to X from its advertisement to A, node A cannot guess that this is due to the split horizon strategy (the source of information was A) or because B has not received any news about X recently.

- The split horizon strategy can be combined with the poison reverse strategy.

- Node B can still advertise the value for X, but if the source of information is A, it can replace the distance with infinity as a warning: "Do not use this value; what I know about this route comes from you."
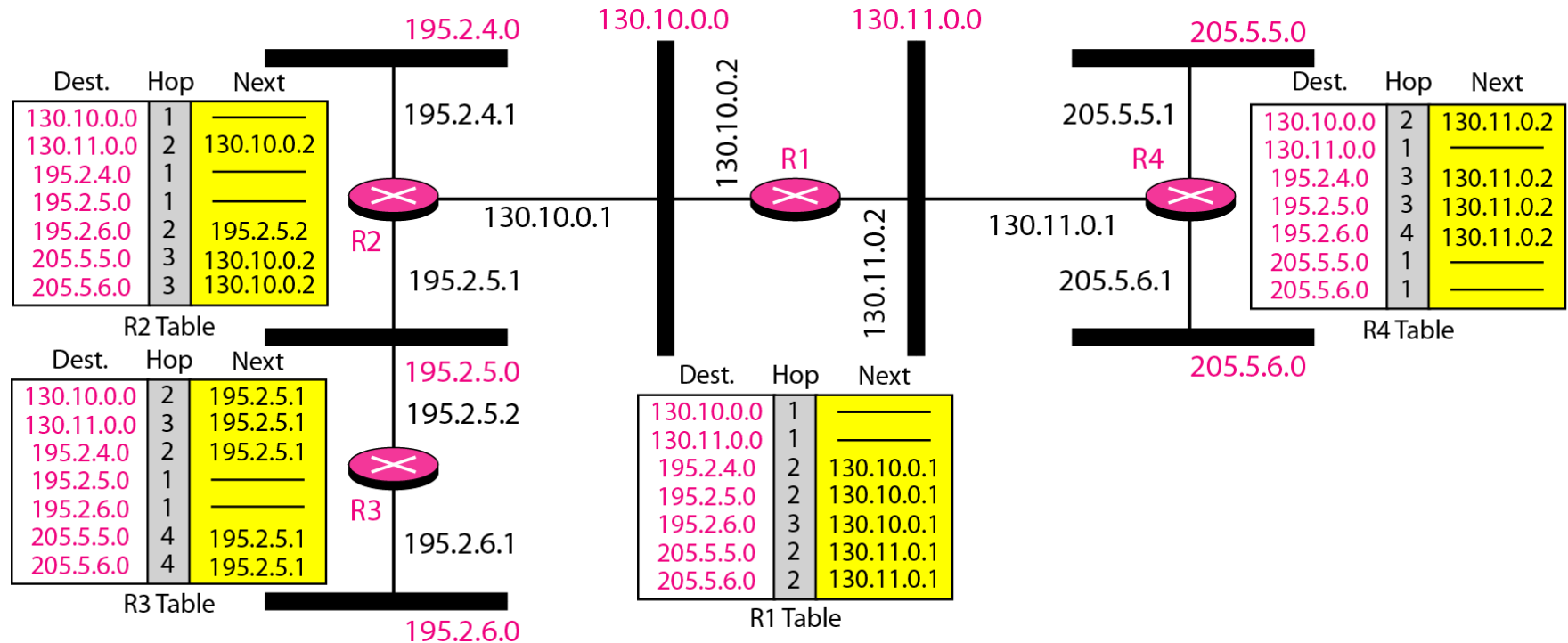
# Figure *Three-node instability*
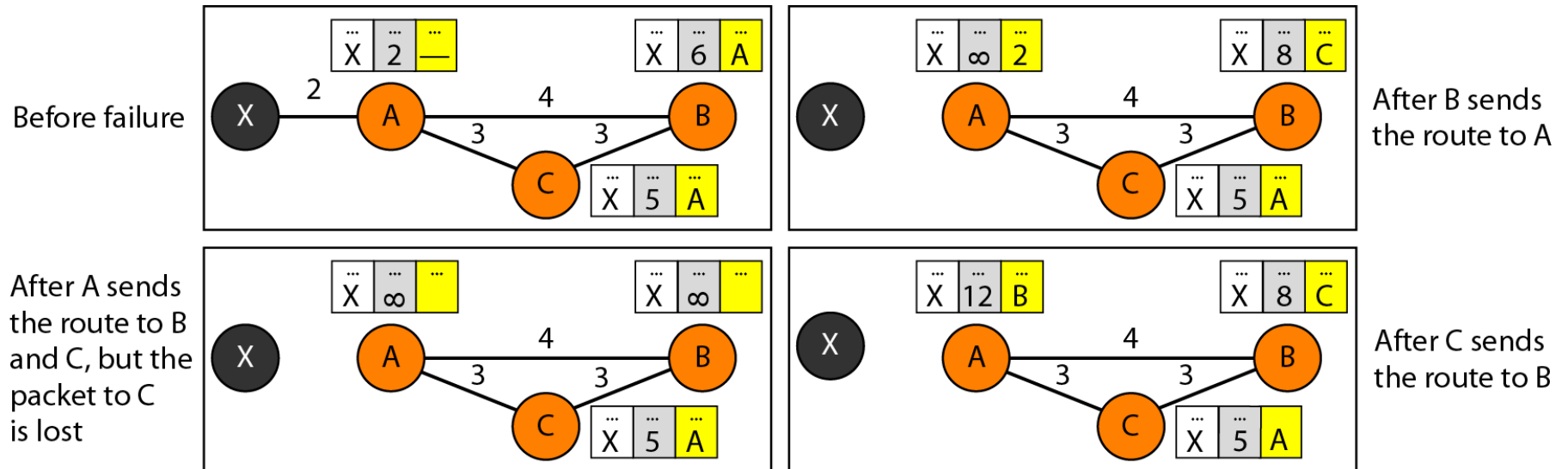
# Routing Information Protocol (RIP)

It 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 be sent to reach its destination.

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

195.2.4.0     130.10.0.0     130.11.0.0     205.5.5.0

195.2.4.1

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

R2 Table

R2    130.10.0.1

195.2.5.1

130.10.0.2

R1

130.11.0.2

205.5.5.1

R4    130.11.0.1

205.5.6.1

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

R4 Table

195.2.5.0

195.2.5.2

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

R3 Table

R3

195.2.6.1

205.5.6.0

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

R1 Table

195.2.6.0

# Link state routing

- In link state routing, each node in the domain has the entire topology of the domain

- List of nodes and links, how they are connected including cost (metric), and condition of the links (up or down)

# Link state knowledge
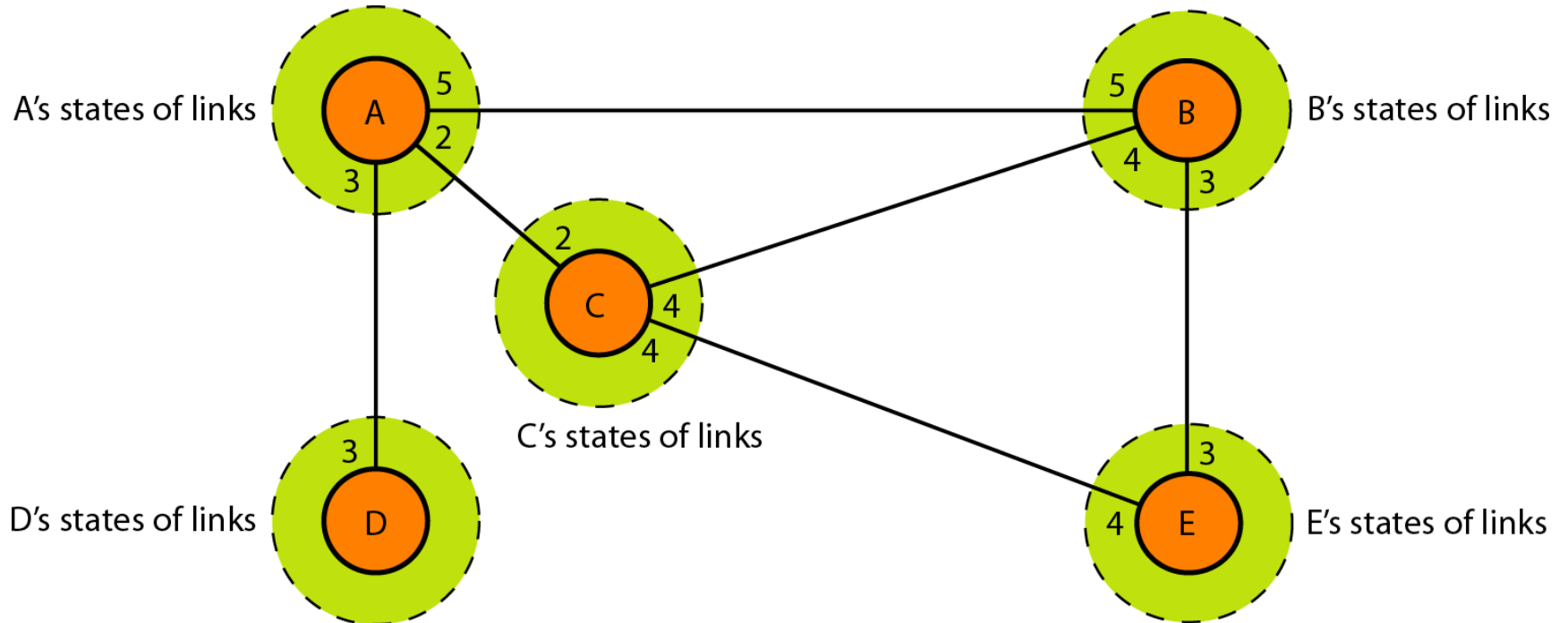
# Build routing table

**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:

  - node identity

  - list of links

  - sequence number,

  - 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

- LSPs are generated on two occasions:

  - *When there is a change in the topology of the domain*

  - On a periodic basis

**Flooding of LSPs-**

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

- 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, 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.

# **Figure** *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).
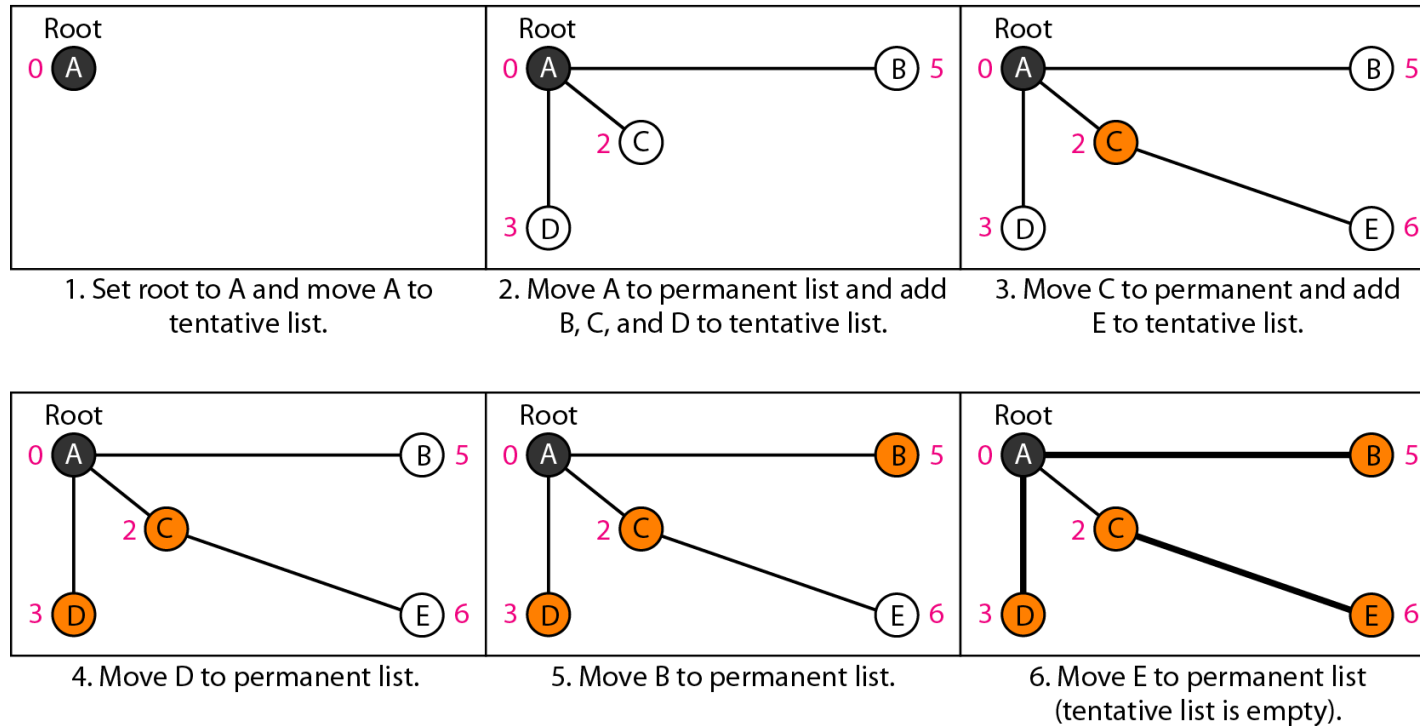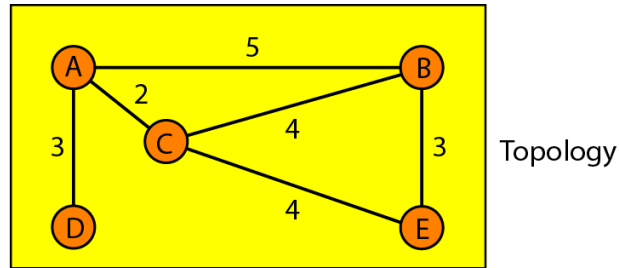
## Dijkstra algorithm
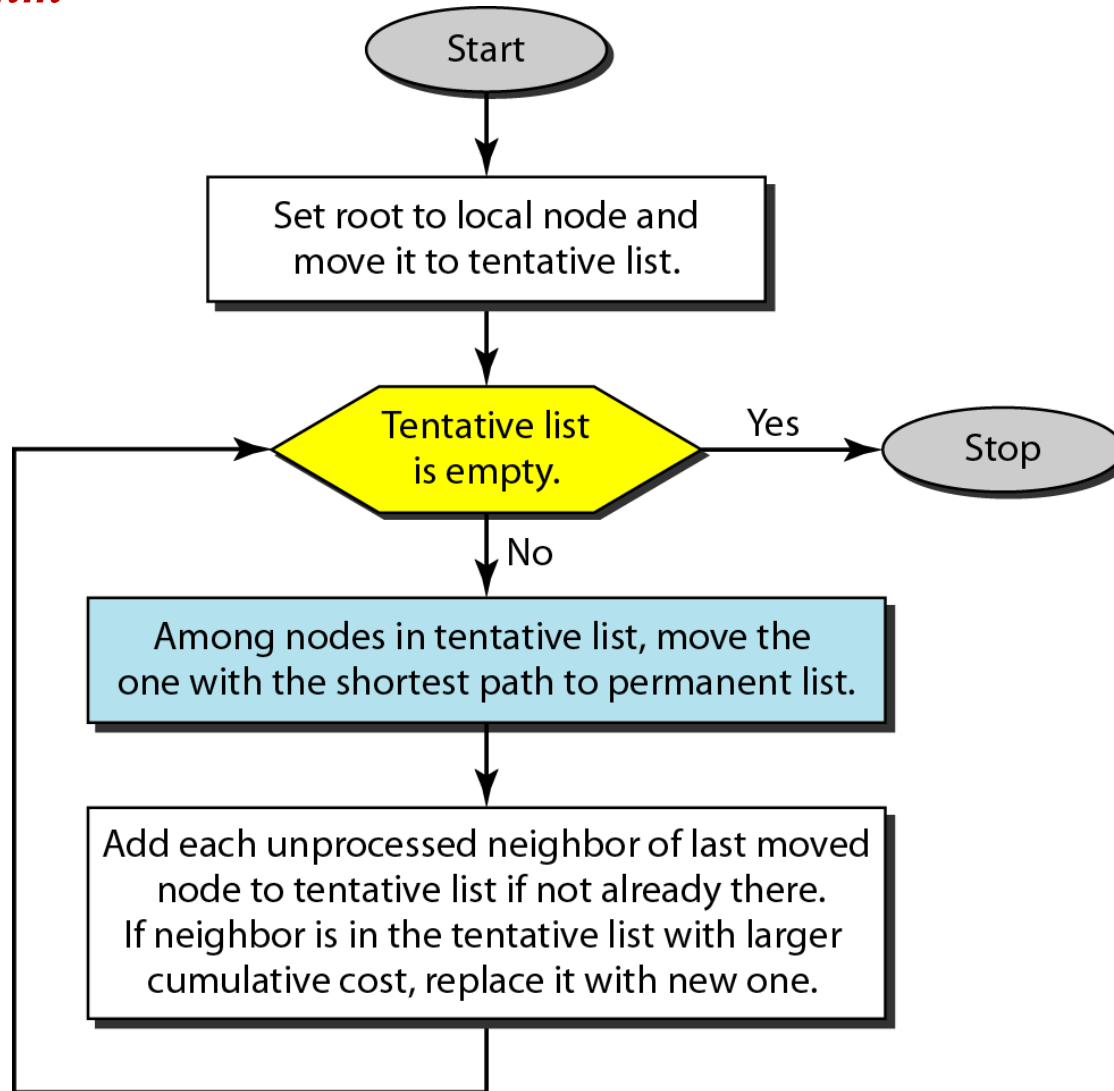
**Table** *Routing table for node A*

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

# OSPF

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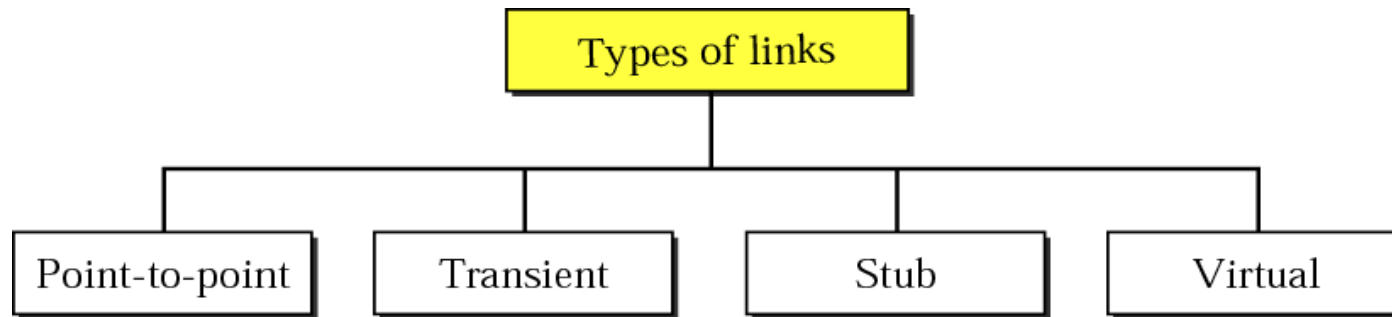
**Figure** *Areas in an autonomous system*
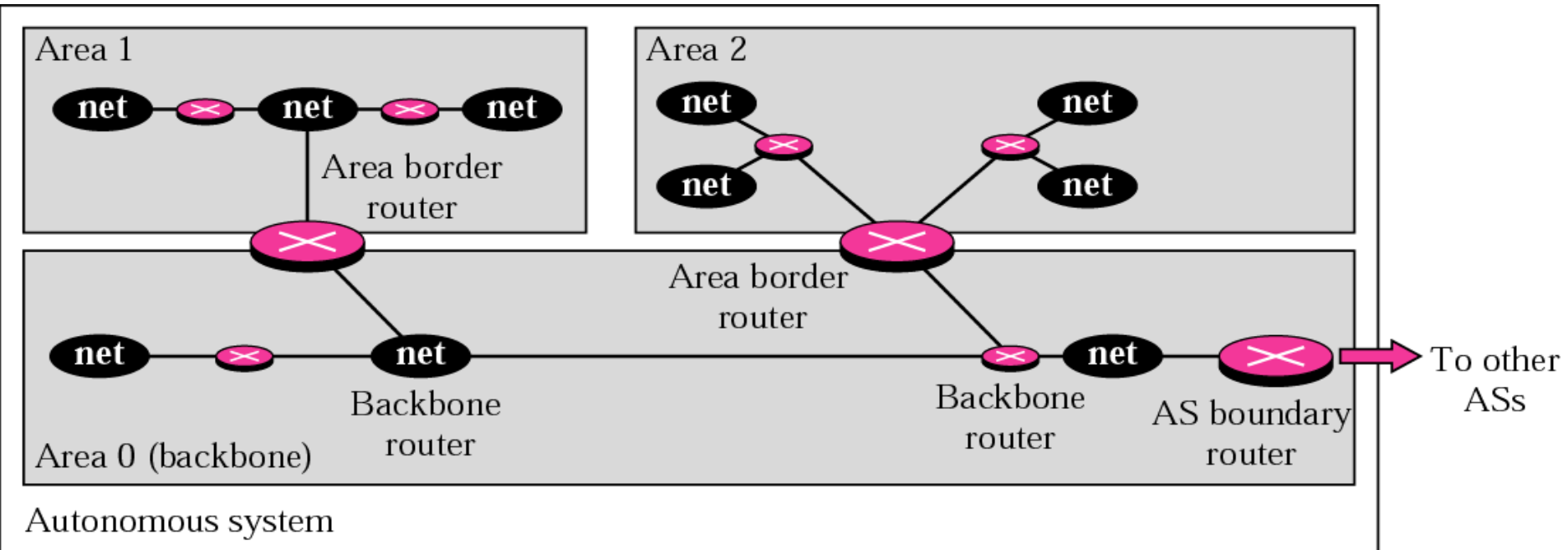
# Metric in OSPF

- OSPF allows administrator to assign a cost called metric.

- The metric can be based on type of service- minimum delay, maximum throughput and so on.

- Therefore, a router can have multiple routing tables based on different type of service.

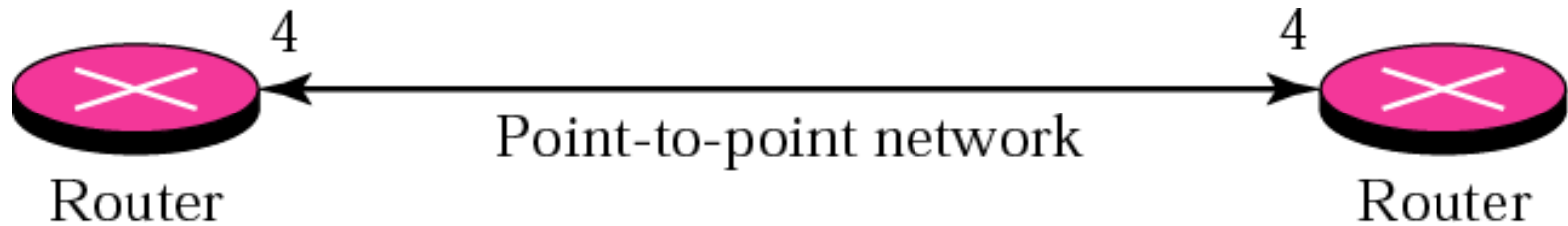**Figure** *Types of links*
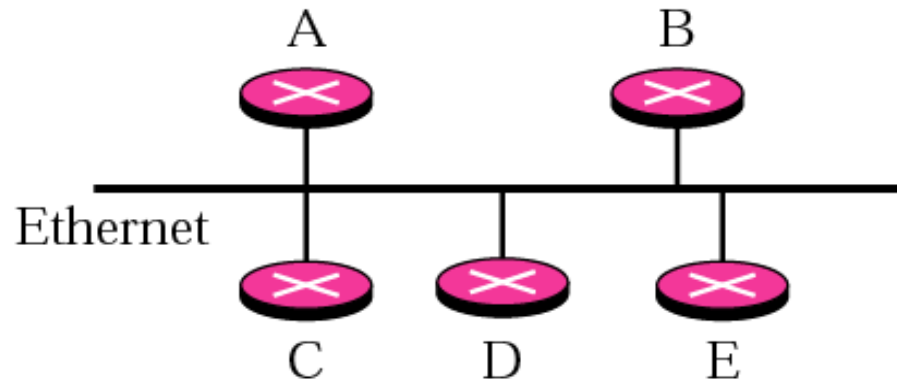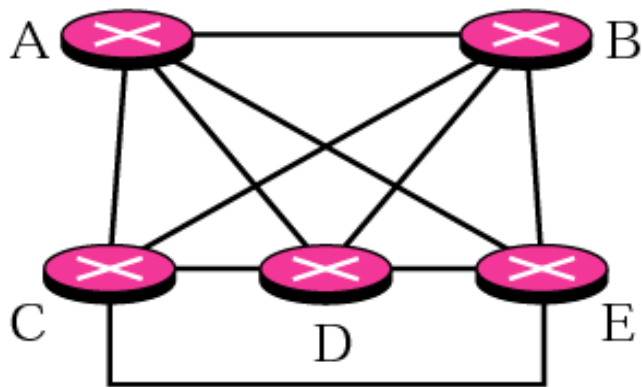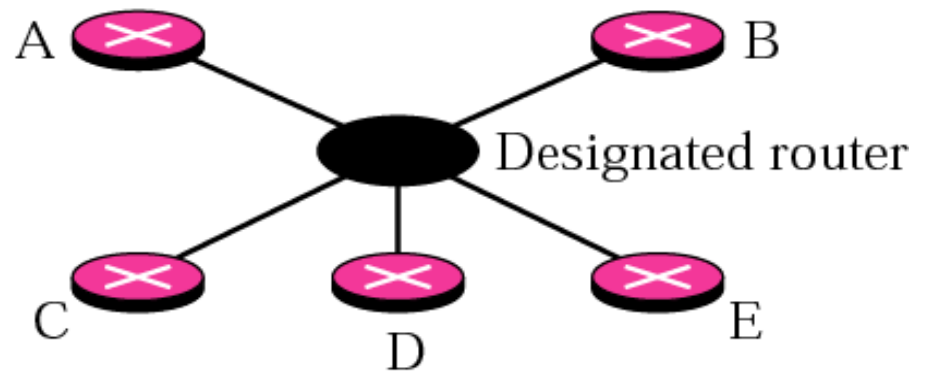
**Figure** *Point-to-point link*

Figure   *Transient link*



A   B

Ethernet

C   D   E

a. Transient network

A   B

C   D   E

b. Unrealistic representation

A   B

Designated router
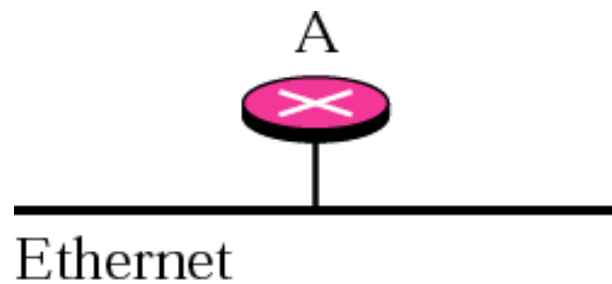
C   D   E

c. Realistic representation

**Figure** *Stub link*



a. Stub network   b. Representation

# PATH VECTOR ROUTING

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

**The topics discussed in this section include:**
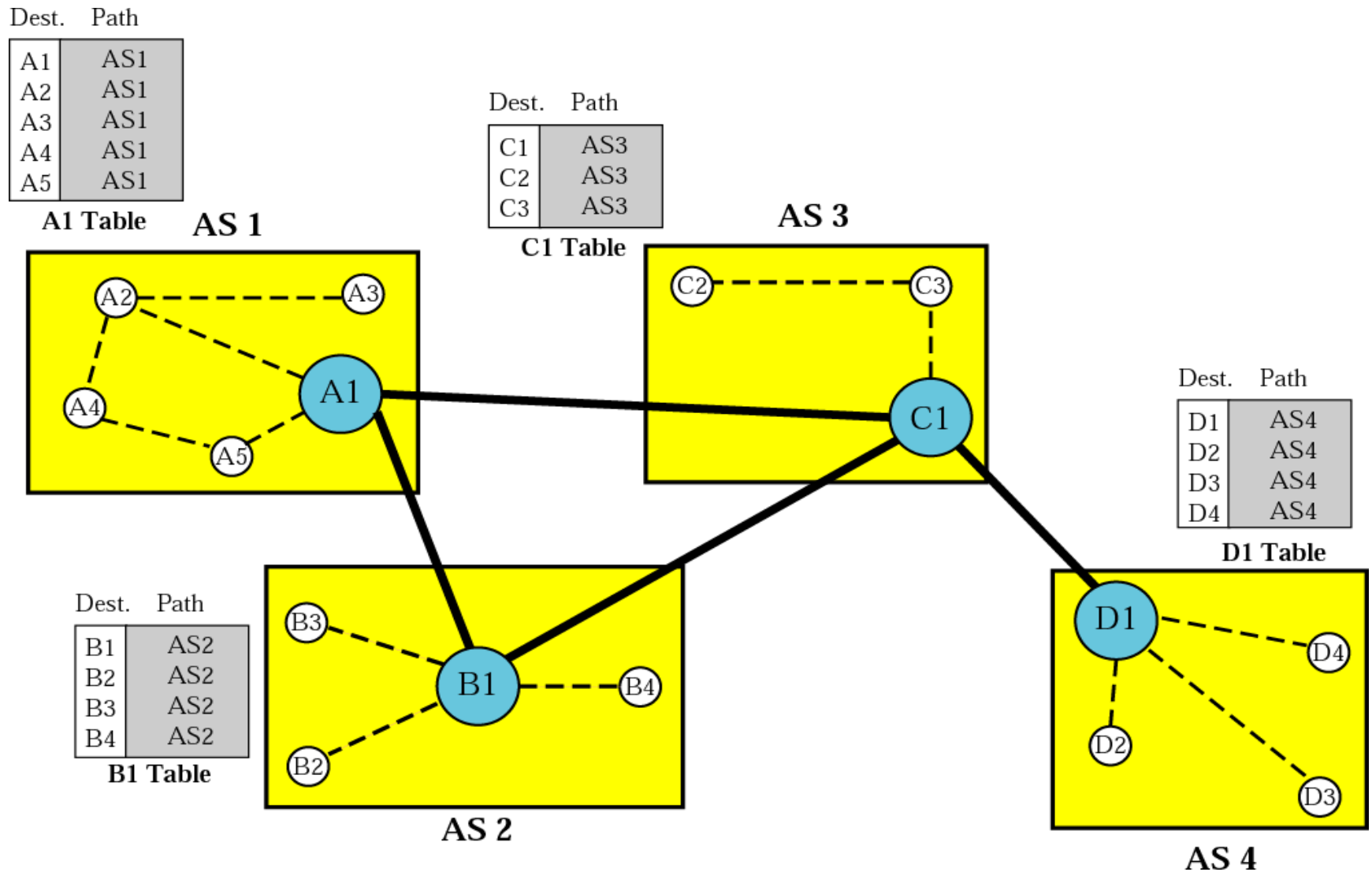
*Initialization*
*Sharing*
*Updating*

- Problem in distance vector is node instability.

- Problem in link state routing is heavy traffic due to flooding and extra resources required for generating routing tables.

# Steps in path vector Routing

- Initialization

- Sharing

- Updating– Loop prevention, policy routing, optimum path.

# Figure *Initial routing tables in path vector routing*



Dest.  Path

| A1 | AS1 |
|----|-----|
| A2 | AS1 |
| A3 | AS1 |
| A4 | AS1 |
| A5 | AS1 |

**A1 Table**   **AS 1**

Dest.  Path

| C1 | AS3 |
|----|-----|
| C2 | AS3 |
| C3 | AS3 |

**C1 Table**   **AS 3**

Dest.  Path

| D1 | AS4 |
|----|-----|
| D2 | AS4 |
| D3 | AS4 |
| D4 | AS4 |

**D1 Table**

Dest.  Path

| B1 | AS2 |
|----|-----|
| B2 | AS2 |
| B3 | AS2 |
| B4 | AS2 |

**B1 Table**

**AS 2**

**AS 4**

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



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

**A1 Table**

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

**B1 Table**

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

**C1 Table**

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

**D1 Table**

# BGP

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

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

*Types of Autonomous Systems*
*Path Attributes*
*BGP Sessions*
*External and Internal BGP*

# *Path Attributes*

- Well Known attribute
  - Mandatory
  - Discretionary
- Optional attribute
  - Transitive
  - Non-transitive

# BGP Sessions

- Using TCP connections—reliable

- Last for long time so it is called as semi permanent connections.

**Figure** *Internal and external BGP sessions*