Data Communications and Networking Fourth Edition

Forouzan

Chapter 22

Network Layer: Delivery, Forwarding, and Routing

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

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

- Desirable properties of Routing Algorithms:
  - Correctness (applicable to all)
  - Simplicity (applicable to all)
  - Robustness: able to cope up with
    - changes in topology, load.
    - hardware and software failures
  - Stability (hard to achieve)
    - Converge to equilibrium
  - Fairness (conflicting)
  - Optimality (conflicting) see next **fig.**
- Types of Routing Algorithms:
  - **Non-Adaptive: Static. Routing decisions computed in advance, off-line and downloaded.**
  - Adaptive: Dynamic. Adaptive to the changes in topology **and load. Issue here is how to get the information? Locally, From adjacent routers, from all routers?**

Figure 22.12 *Autonomous systems*

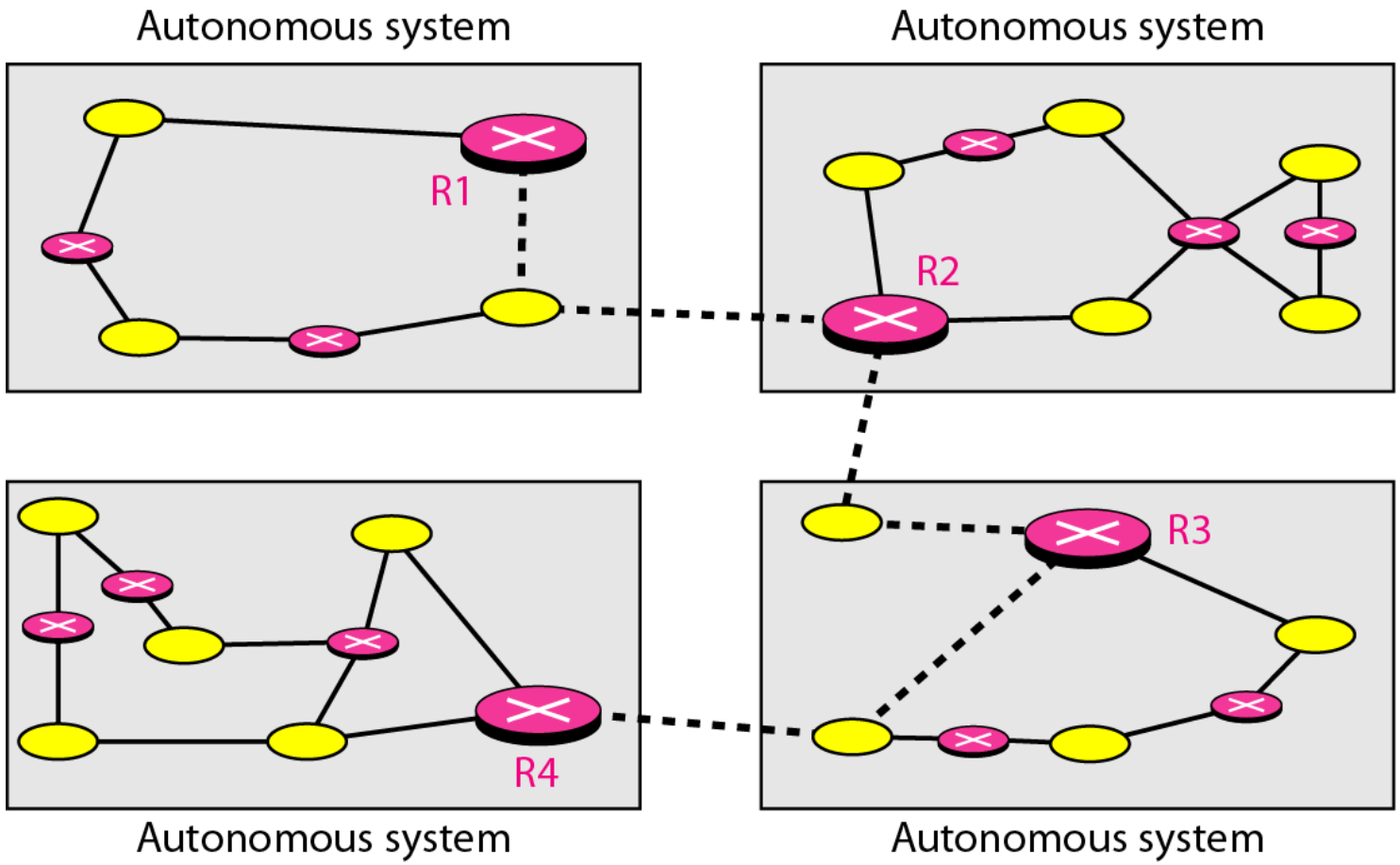
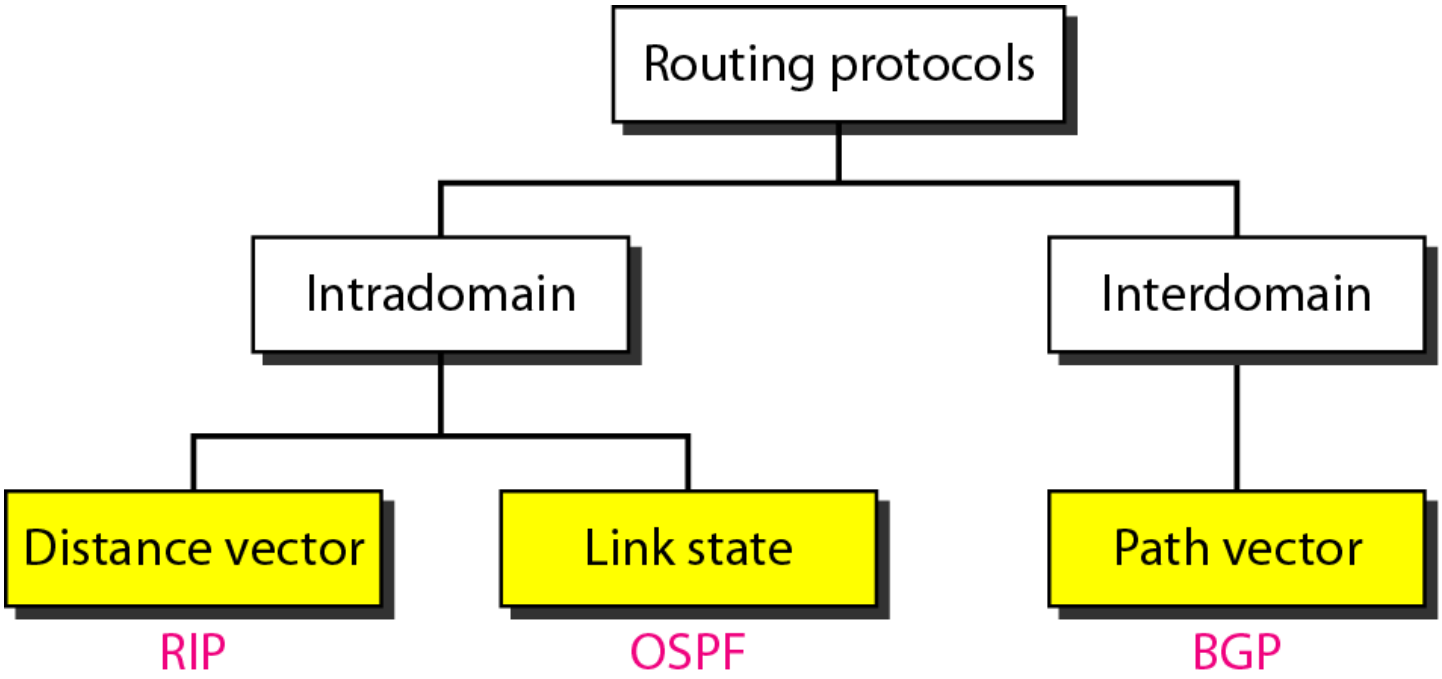


Figure 22.13 *Popular routing protocols*



Distance Vector Routing:

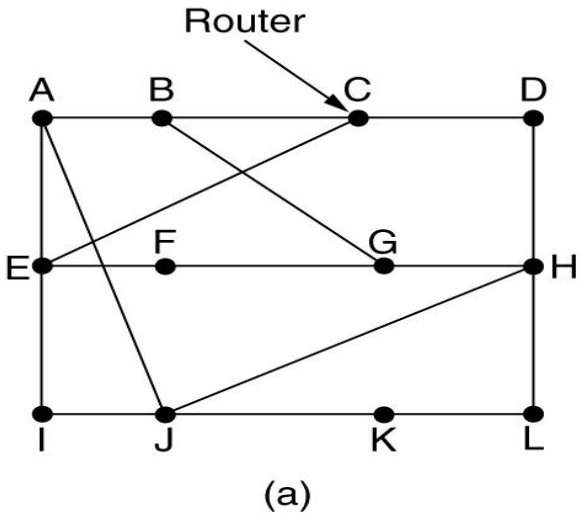
- So far we have studied Static Routing Algorithms.
- But practically dynamic Routing Algorithms are used.
- Following two are Dynamic Routing Algorithms:
  - 1. Distance Vector Routing Algorithm.
  - 2. Link State Routing Algorithm.
- **Distance Vector Routing Algorithm:**
- At each step within a router:
  - Get routing tables from neighbours
  - Compute distance to neighbours
  - Compute new routing table

1. Router transmits its *distance vector* to each of its neighbors.
2. Each router receives and saves the most recently received *distance vector* from each of its neighbors.
3. A router **recalculates** its distance vector when:
   a. It receives a *distance vector* from a neighbor containing different information than before.
   b. It discovers that a link to a neighbor has gone down (i.e., a topology change).

The DV calculation is based on minimizing the cost to each destination.

The distance vector routing algorithm is sometimes called by other names, the **distributed Bellman-Ford** routing algorithm and the **Ford-Fulkerson algorithm.**

Distance Vector Routing



(a)

New estimated delay from J

To	A	I	H	K	Line
A	0	24	20	21	8 A
B	12	36	31	28	20 A
C	25	18	19	36	28 I
D	40	27	8	24	20 H
E	14	7	30	22	17 I
F	23	20	19	40	30 I
G	18	31	6	31	18 H
H	17	20	0	19	12 H
I	21	0	14	22	10 I
J	9	11	7	10	0 –
K	24	22	22	0	6 K
L	29	33	9	9	15 K
	JA delay is 8	JI delay is 10	JH delay is 12	JK delay is 6	New routing table for J

Vectors received from J's four neighbors

(b)

(a) A subnet. (b) Input from A, I, H, K, and the new routing table for J.

- Drawback of Distance Vector Routing:
  - Count to Infinity Problem:
    - It reacts rapidly to good news,
    - But, leisurely to bad news.
    - Updates value fast when neighbor is down, but not when neighbor is again up. How?
    - Lie to neighbour about distance if routing via neighbour
    - The core of the problem is that when X tells Y that it has a path somewhere, Y has no way of knowing whether it itself is on the path? This is how problem is created.
  - It does not take bandwidth into account.
  - Take too long to converge changes in one node to all other nodes.
- Solution?
- Link State Routing.

Figure 22.14 *Distance vector routing tables*

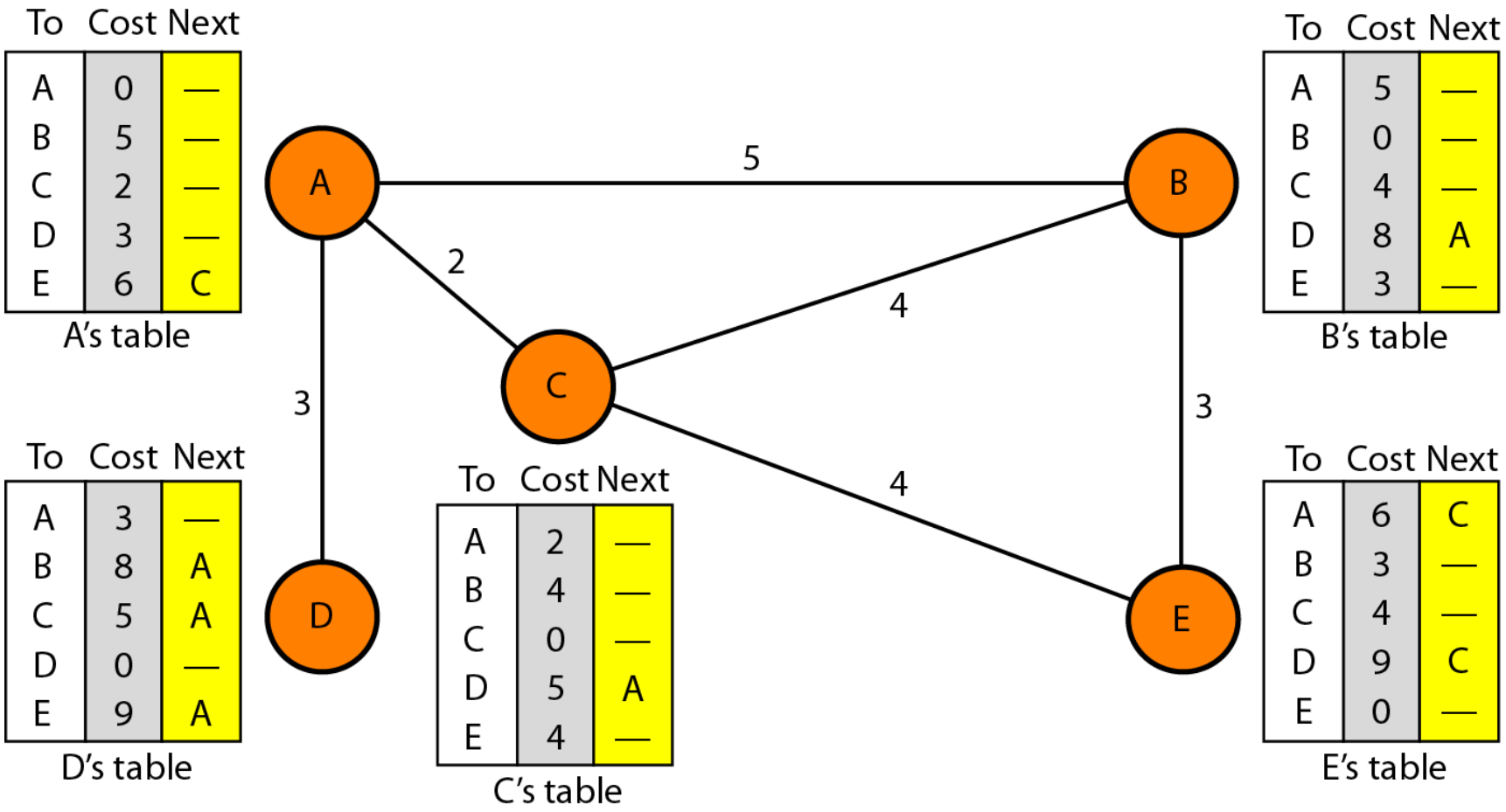
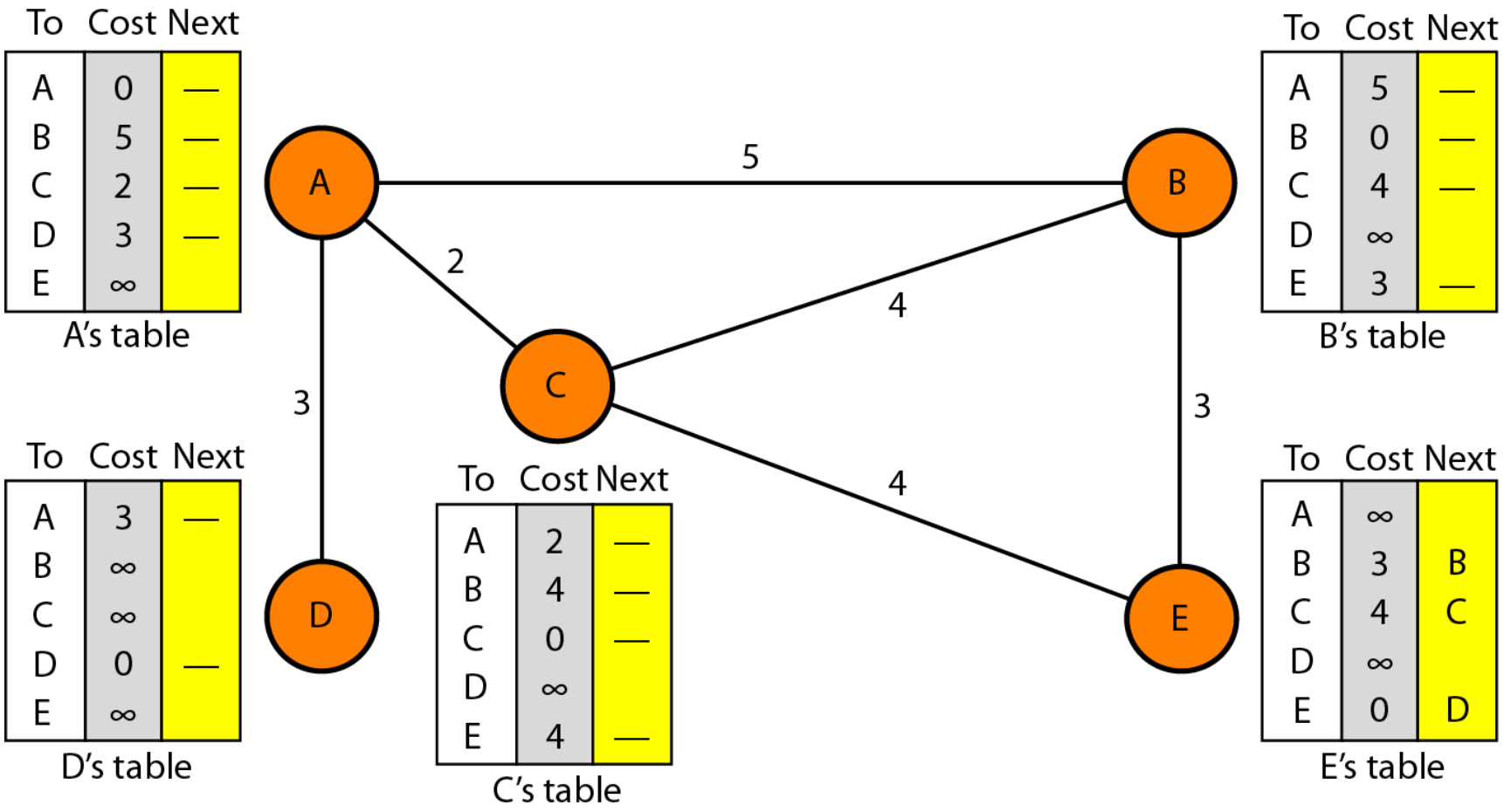


Figure 22.15 *Initialization of tables in distance vector routing*



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*

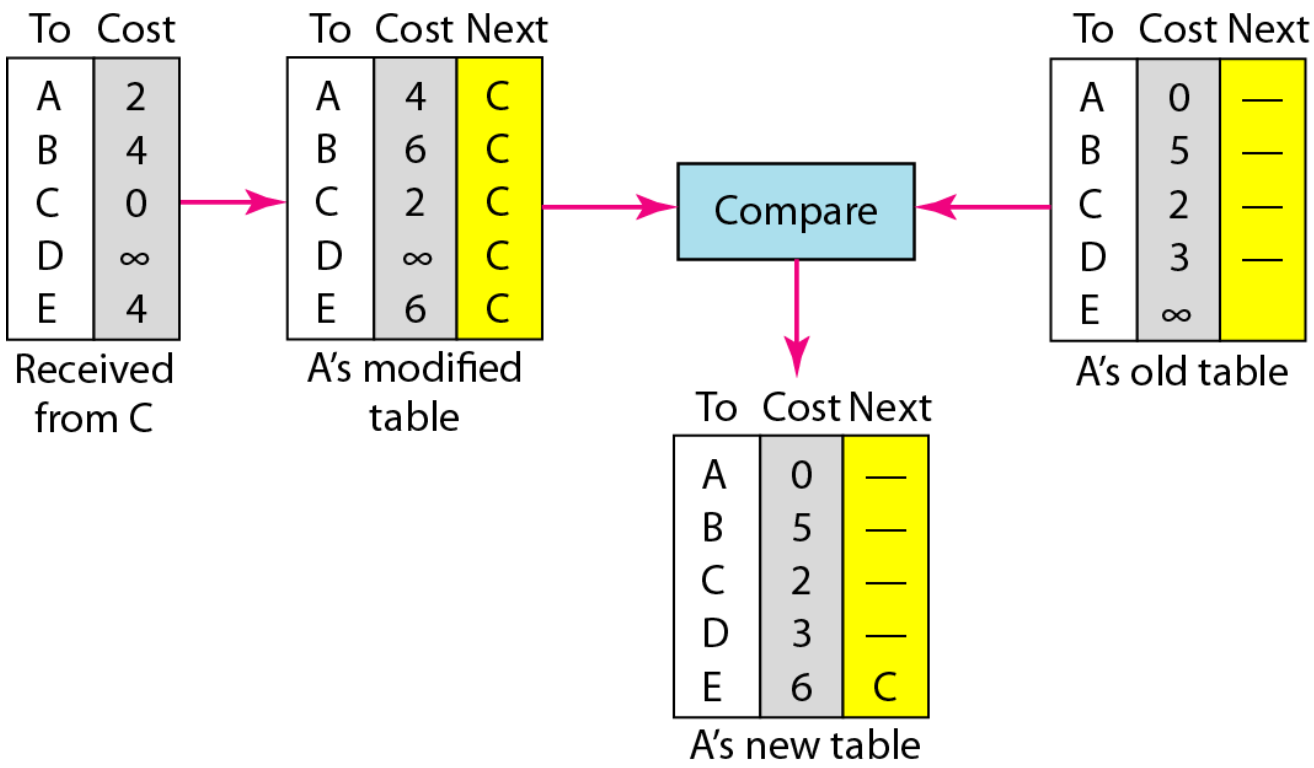


Figure 22.17 *Two-node instability*

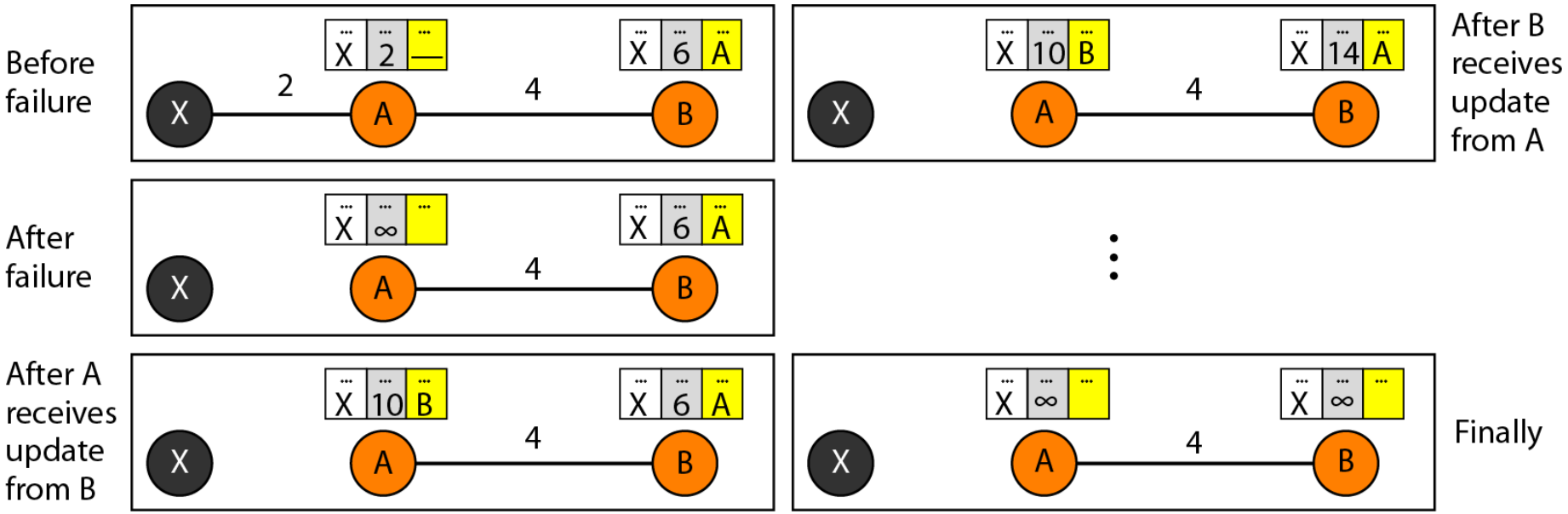


Figure 22.18 *Three-node instability*

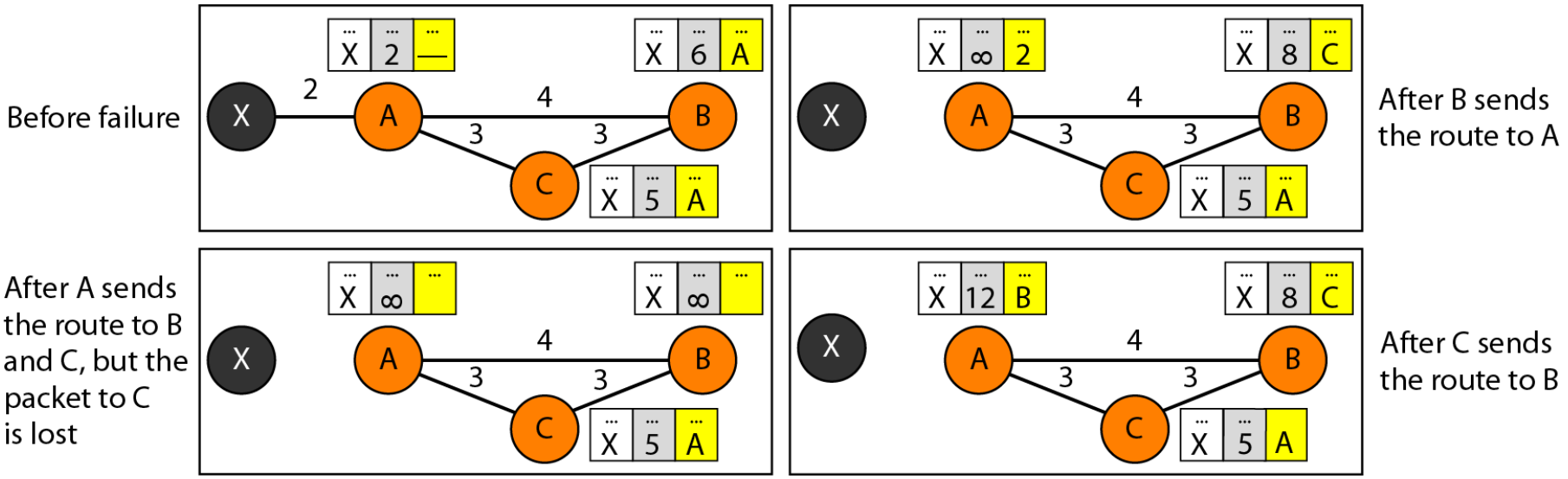
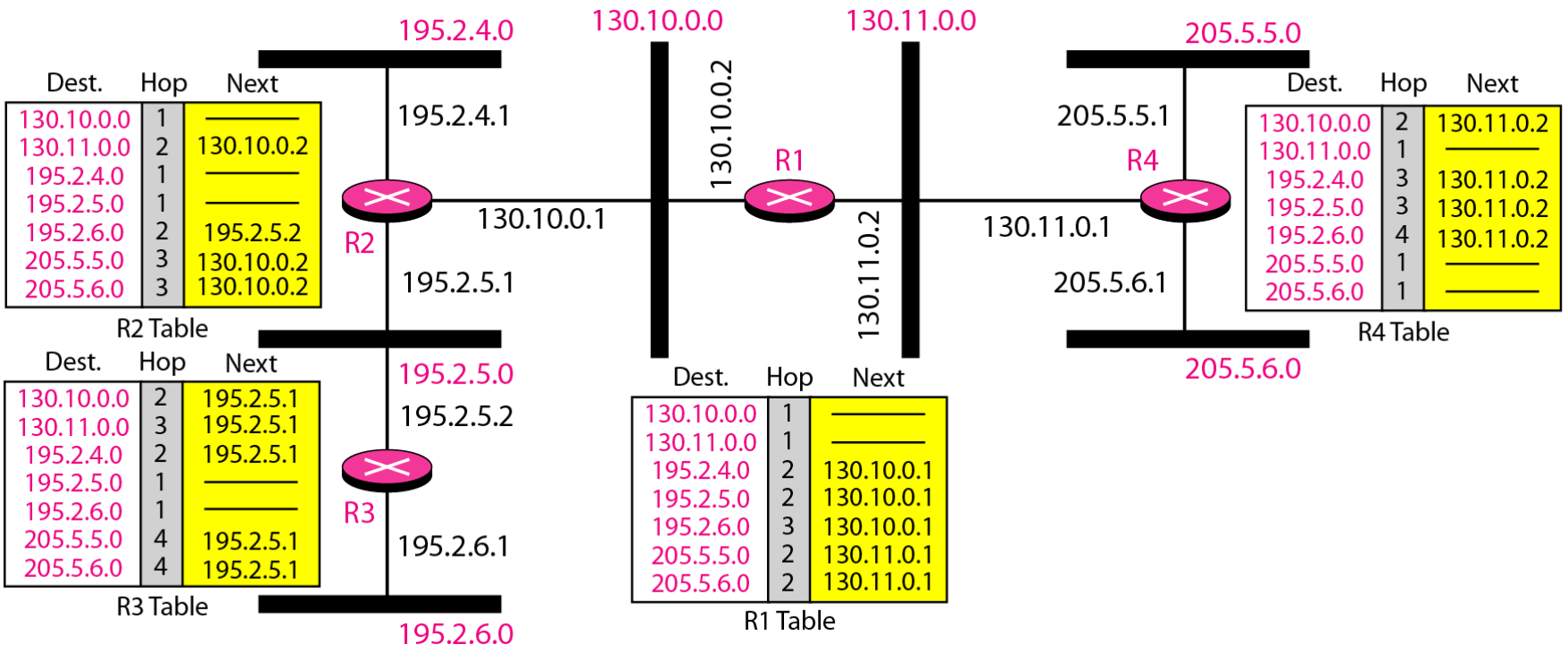


Figure 22.19 *Example of a domain using RIP*



Link State	Distance Vector
link states algorithm is an algorithm **using global information**	the distance vector algorithm is **iterative, asynchronous, and distributed**
each node **talks with all other nodes**, but tell them only the cost of it's directly comparison of some of their attribute	each node **talks to only its directly connected neighbors**, but provides its neighbor with least cost estimates from itself to all the nodes.
Message complexity: With link state, every node has to keep the information about the cost of each link within the network.	**Message complexity**: with distance vector algorithm, message is exchanged between two hosts which are directly connected to each other.
very times, if any of the link cost is changed, all the nodes are **updated.**	change of cost in the link which is belong to the least cost path for one of the nodes, the DV algorithm will update the new value. But if the change doesn't belong to the least cost part between 2 hosts, there will **no updating**.
Speed of convergence: can converge faster in comparison of later.	**Speed of convergence**: can converge slowly and have routing loops while the algorithm is converging.
Such probability is less.	DV algorithm also suffers from the **count to infinity** problem.
Robustness: For LS, when a router is down, it can broadcast a wrong cost for the closest one. LS node is computing for its own forwarding table and other node do the calculation for themselves. **Better than DV.**	**Robustness:** DV, the wrong least cost path can be passed to more than one or all of the node so the wrong calculation will be process in the entire net work. This problem of DV is much **worse than LS algorithm**.

Link State Routing

Each router must do the following:

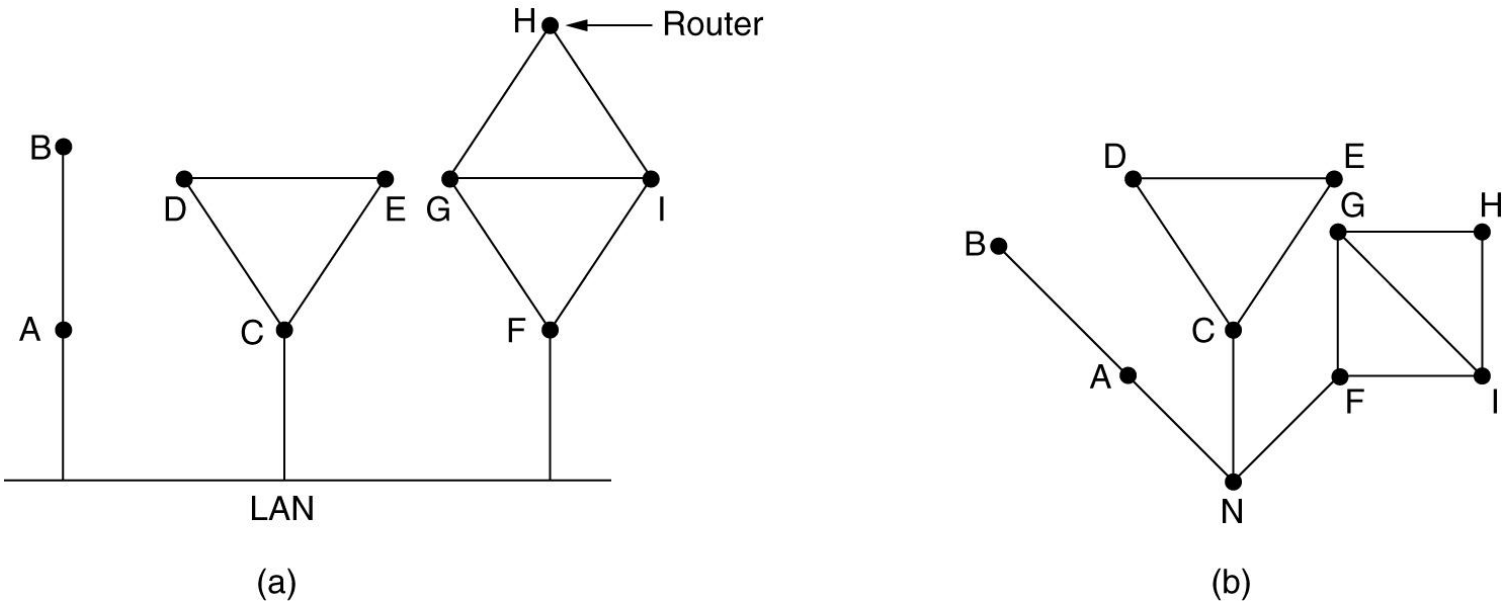
1. Discover its neighbors, learn their network address.
2. Measure the delay or cost to each of its neighbors.
3. Construct a packet telling all it has just learned.
4. Send this packet to all other routers.
5. Compute the shortest path to every other router.

A complete topology is developed. Then Dijkstra's Algorithm can be used to compute the shortest path.

Following 5 steps are followed to implement it.

1. Learning about the Neighbors
2. Measuring Line Cost.
3. Building Link State Packets.
4. Distributing the Link State Packets.
5. Computing the New Routes.

Learning about the Neighbors

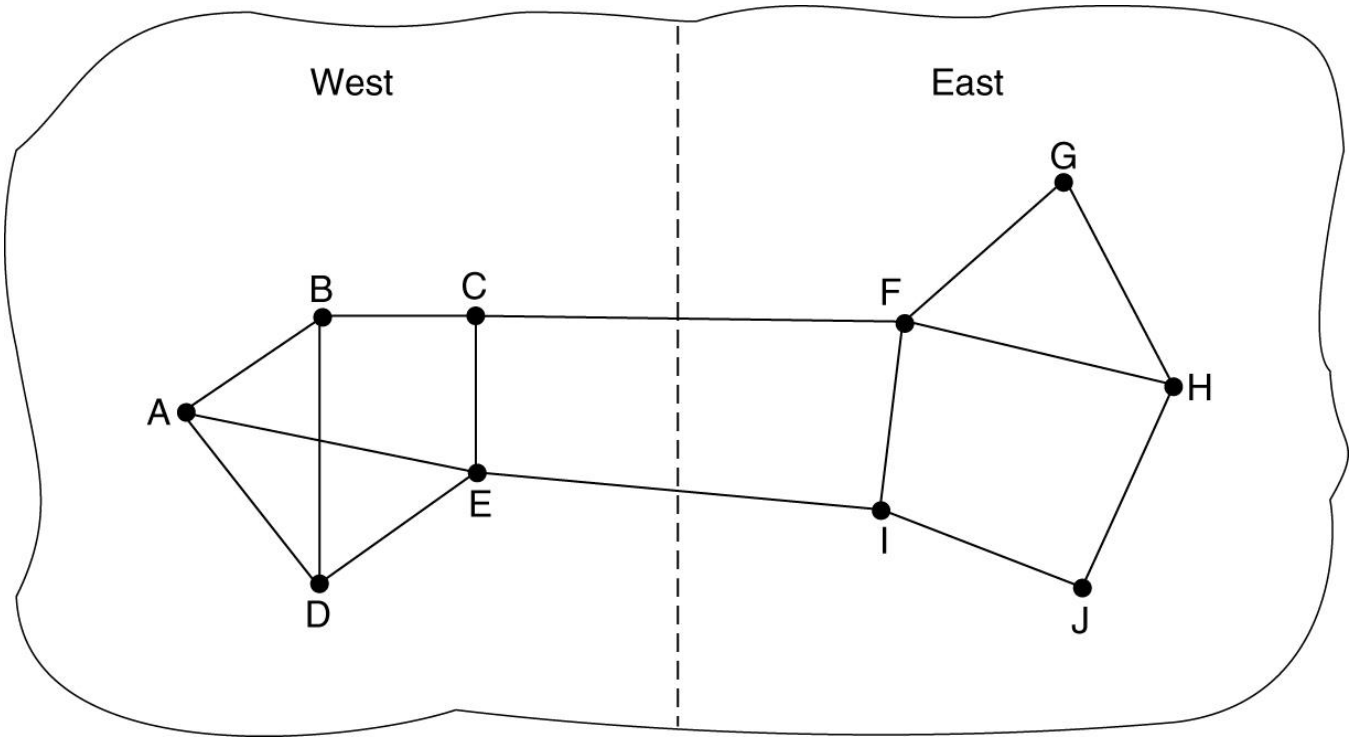


(a) Nine routers and a LAN. (b) A graph model of (a).

- Step 1: Learning about the Neighbours:
  - Upon boot of router,
    - Send HELLO packet on each point-to-point line
    - Routers are supposed to send reply with a globally unique name
- Step 2: Measuring the Line Cost:
  - Measure round-trip delay using ECHO Packet and wait for its reply
  - Take load into account? Yes. Arguments both ways: when choice is given to router having same number of hops from S to D.
    - Yes! preference for unloaded line as shortest path.

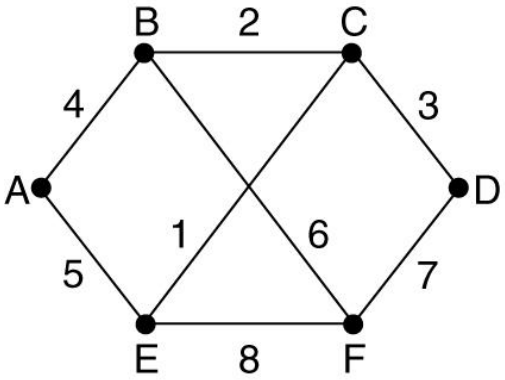
Measuring Line Cost

A subnet in which the East and West parts are connected by two lines.



- Step 3: Building Link State Packets:
  - Packet containing:
    - Identity of sender
    - Sequence number + age
    - For each neighbour:
    - name + distance
  - When to build the link state packets?
    - Periodically
    - when significant events occur
- See next figure.

Building Link State Packets



(a)

Link State Packets

A		B		C		D		E		F	
Seq.		Seq.		Seq.		Seq.		Seq.		Seq.	
Age		Age		Age		Age		Age		Age	
B	4	A	4	B	2	C	3	A	5	B	6
E	5	C	2	D	3	F	7	C	1	D	7
		F	6	E	1			F	8	E	8

(b)

(a) A subnet. (b) The link state packets for this subnet.

- Step 4: Distributing Link State Packets:
- Distributing link state packets
  - Trickiest part of algorithm
    - Arrival time for packets different
    - How to keep consistent routing tables?
  - Basic algorithm
    - Flooding +
    - Sequence number (in each packet) to limit duplicates.
  - Manageable problems
    - Wrap around of sequence numbers results to wrong data. Solution? Use 32 bit sequence number.
    - Wrong sequence number used in case of :
      - lost in case of crash
      - Corrupted data transmitted.
  - Solution? include the age of each packet after the sequence number and decrement it once per second. When the age hits zero, the information from that router is discarded.
      - duplicates are discarded
      - Old packets are thrown out

Distributing the Link State Packets

The packet buffer for router B in the previous slide.

Source	Seq.	Age	Send flags A	Send flags C	Send flags F	ACK flags A	ACK flags C	ACK flags F	Data
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

Step 5: Computing new routes:

- With a full set of link state packets, a router can:
  - Construct the entire subnet graph
  - Run Dijkstra's algorithm to compute the shortest path to each destination
- Problems for large subnets
  - Memory to store data
  - Compute time for developing these tables.

- Usage:
  - IS-IS protocol (Intermediate System, Intermediate System)
    - Designed for DECnet(digital equipment corporation network protocol suite), adopted by ISO(international standardization organization), used still in internet.
    - Supports multiple network layer protocols
  - OSPF(Open Shortest Path First) protocol used in Internet
  - Common features:
    - Self-stabilizing method of flooding link state updates
    - Concept of a designated router on a LAN
    - Method of computing and supporting path splitting and multiple metrics.
    - Useful in Multi Protocol Environment.

Figure 22.20 *Concept of link state routing*

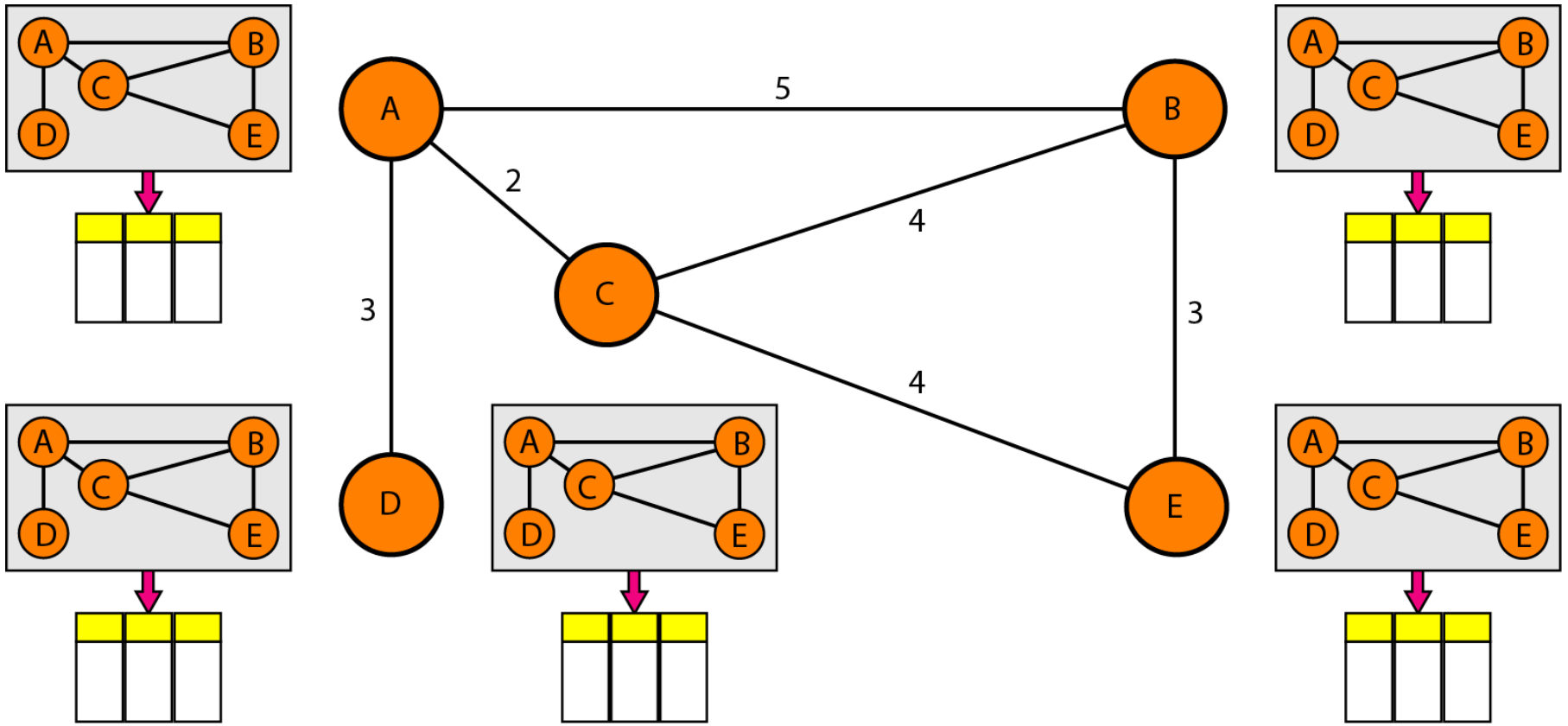
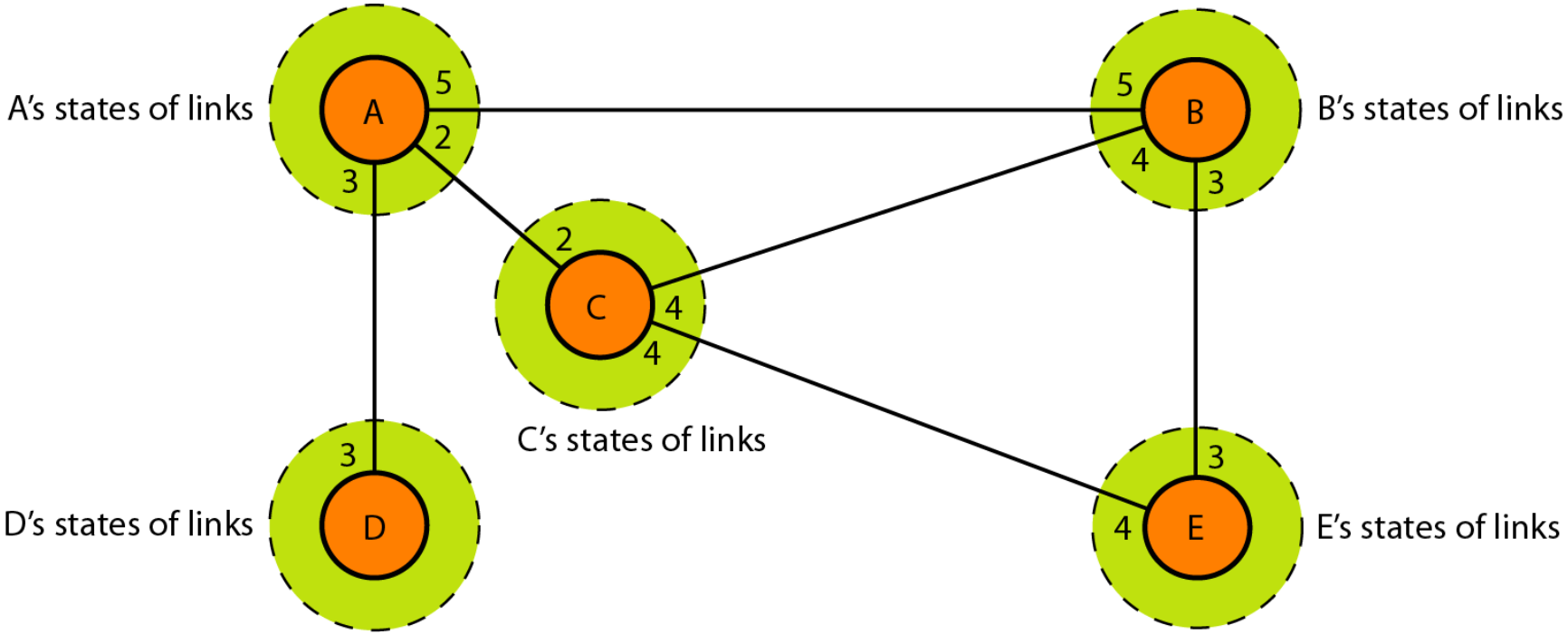


Figure 22.21 *Link state knowledge*



What does it mean to be the shortest (or optimal) route?

a. Minimize mean packet delay
b. Maximize the network throughput
c. Minimize the number of hops along the path

Dijkstra algorithm:

- Each node is labeled (in parentheses) with its distance from the source node along the best known path.
- Initially, no paths are known, so all nodes are labeled with infinity.
- As the algorithm proceeds and paths are found, the labels may change, reflecting better paths.
- A label may be either **tentative** or **permanent**.
- Initially, all labels are tentative.
- When it is discovered that a label represents the shortest possible path from the source to that node, it is made permanent and never changed thereafter.

*Now lets see the **algorithm** which describes this procedure to develop shortest path from source to destination.*

Figure 22.22 *Dijkstra algorithm*

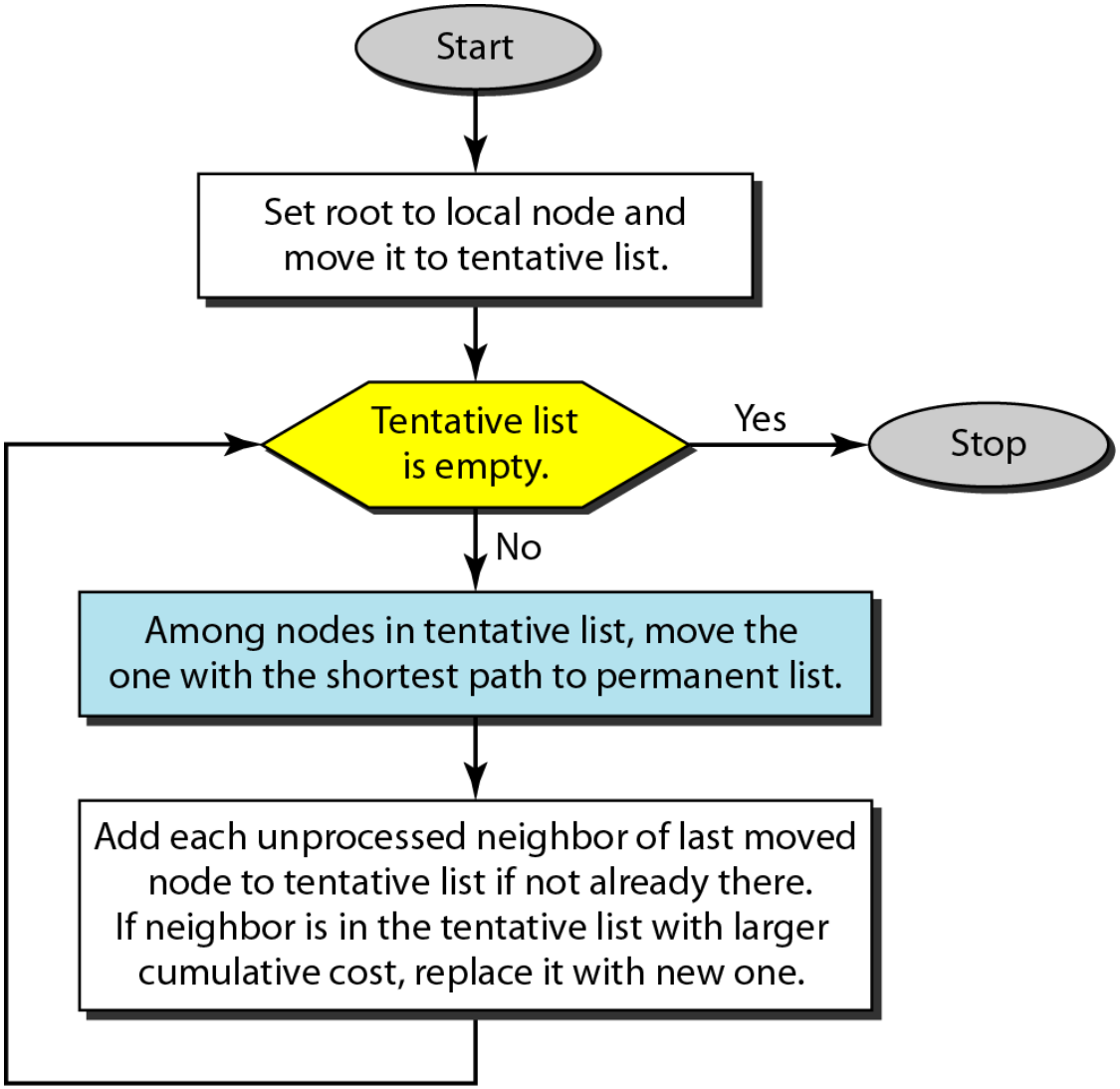


Figure 22.23 *Example of formation of shortest path tree*

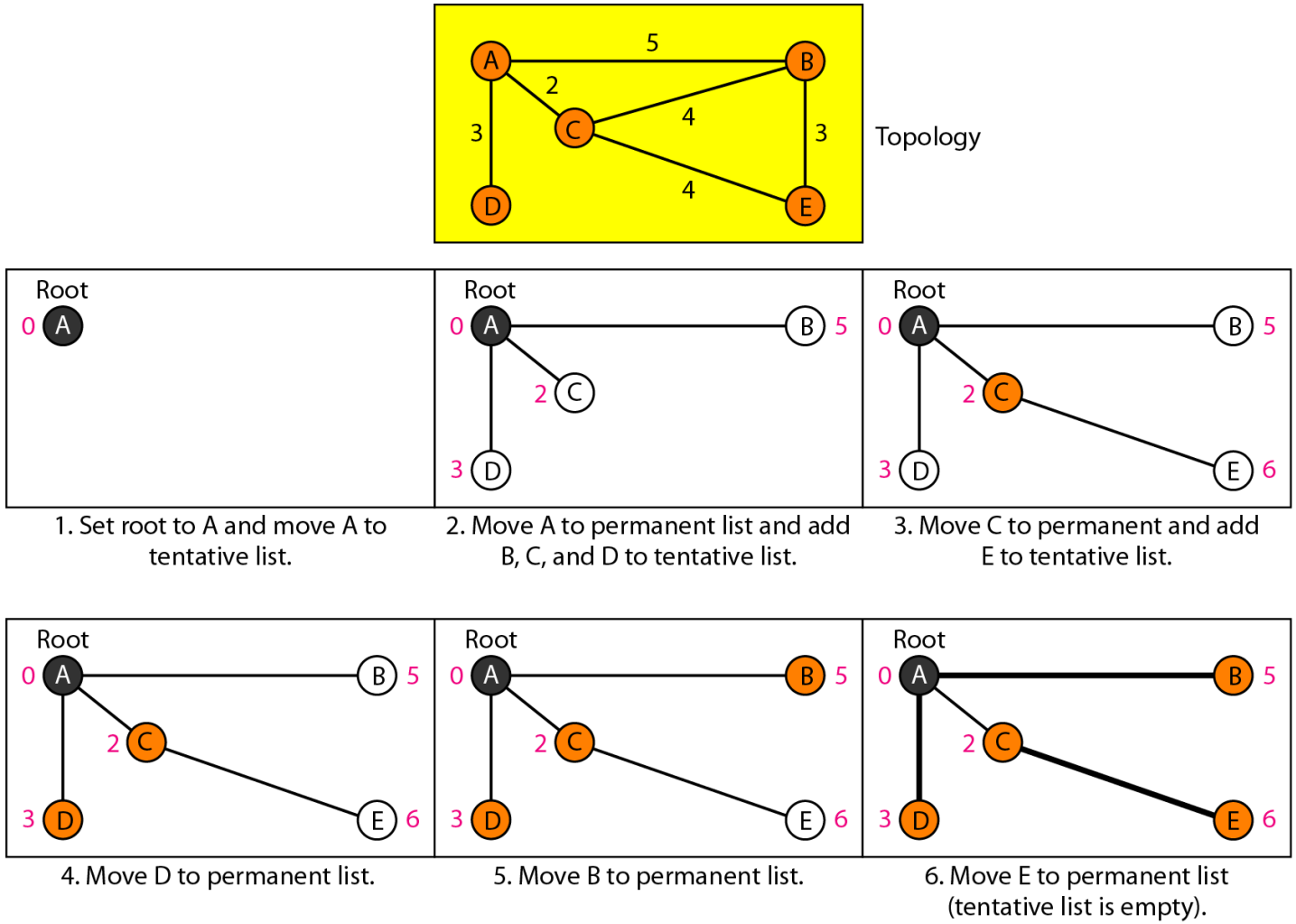


Table 22.2 *Routing table for node A*

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

Figure 22.24 *Areas in an autonomous system*

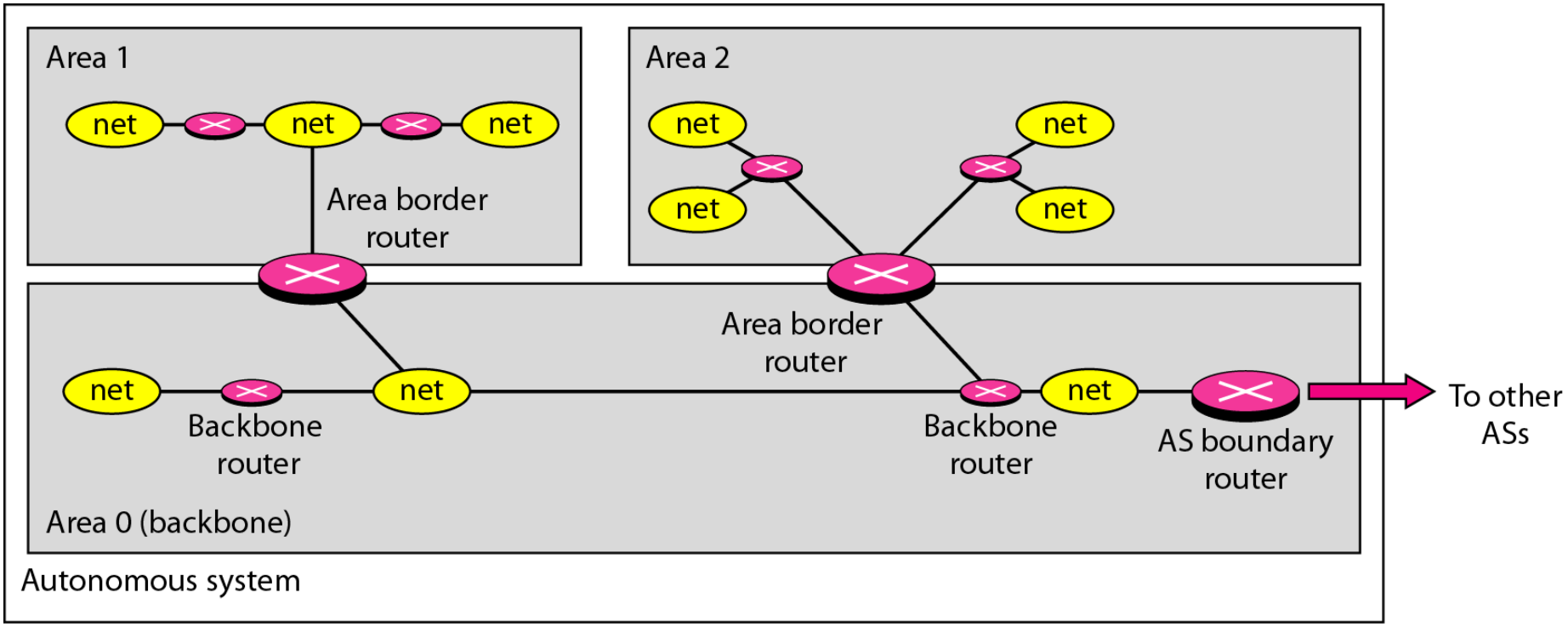


Figure 22.25 *Types of links*

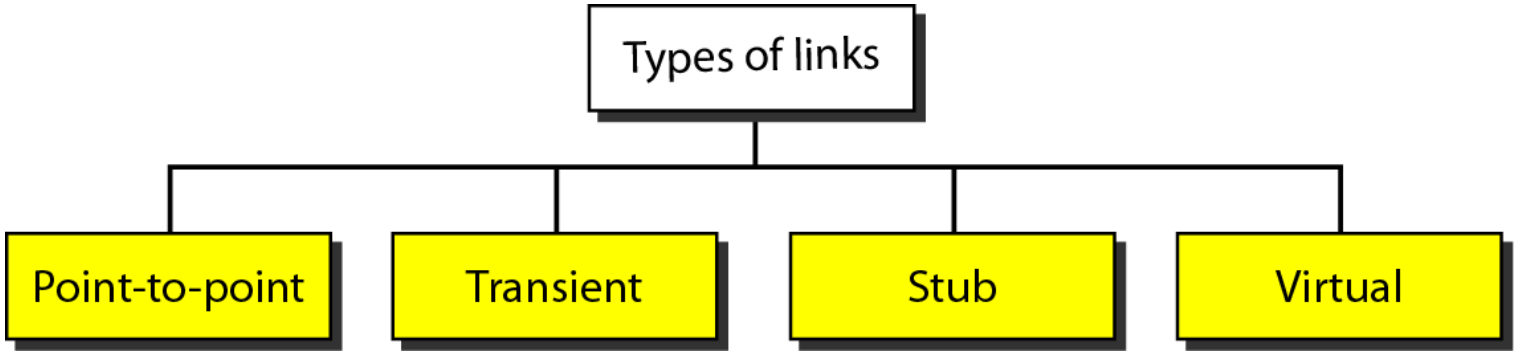


Figure 22.26 *Point-to-point link*

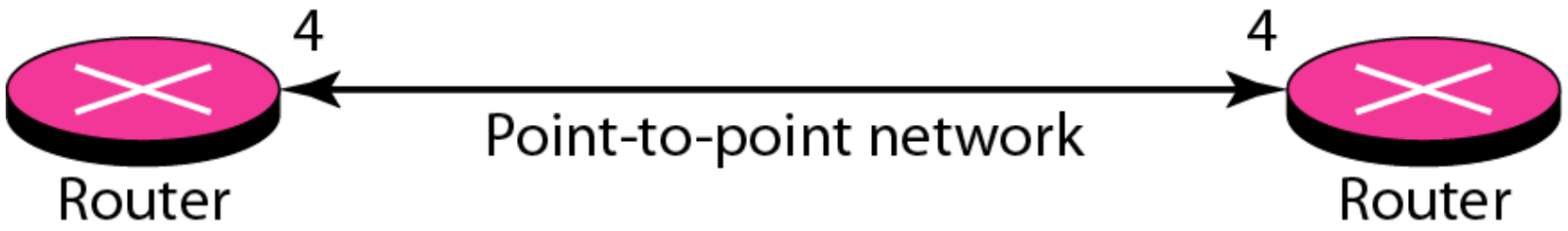
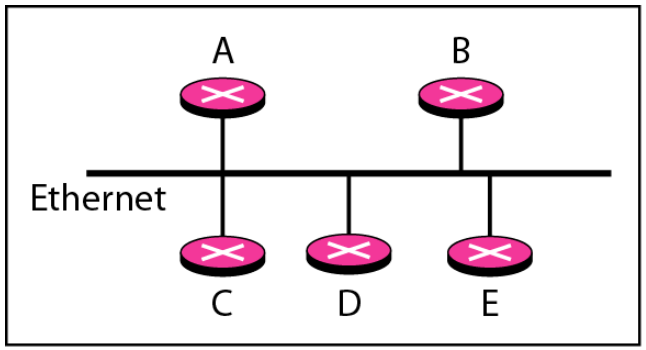
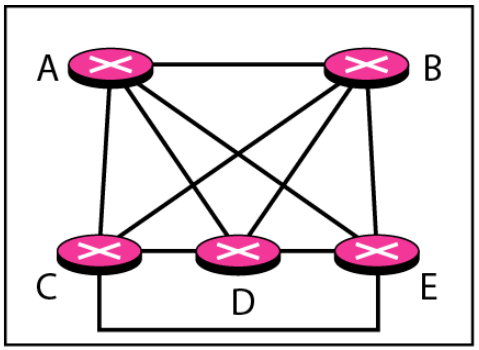


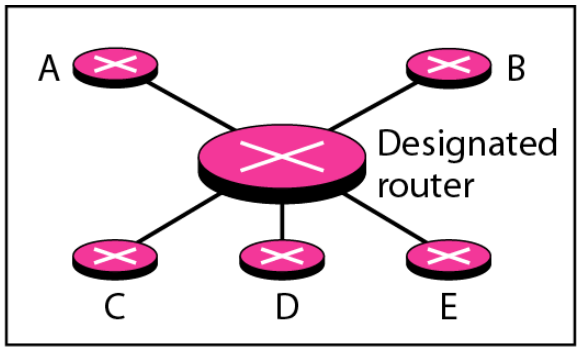
Figure 22.27 *Transient link*



a. Transient network

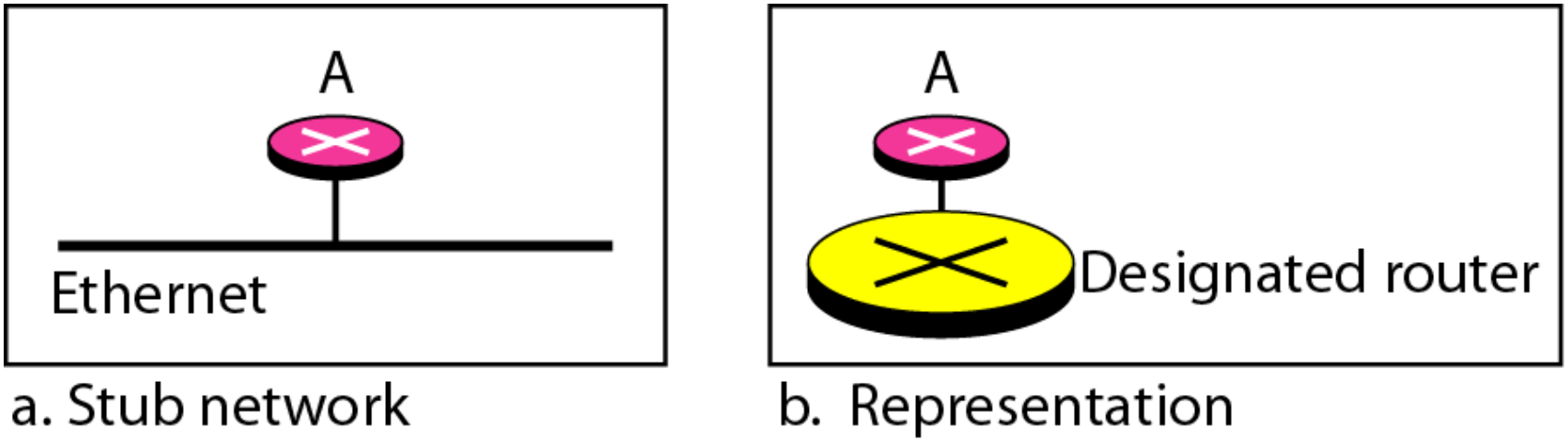


b. Unrealistic representation



c. Realistic representation

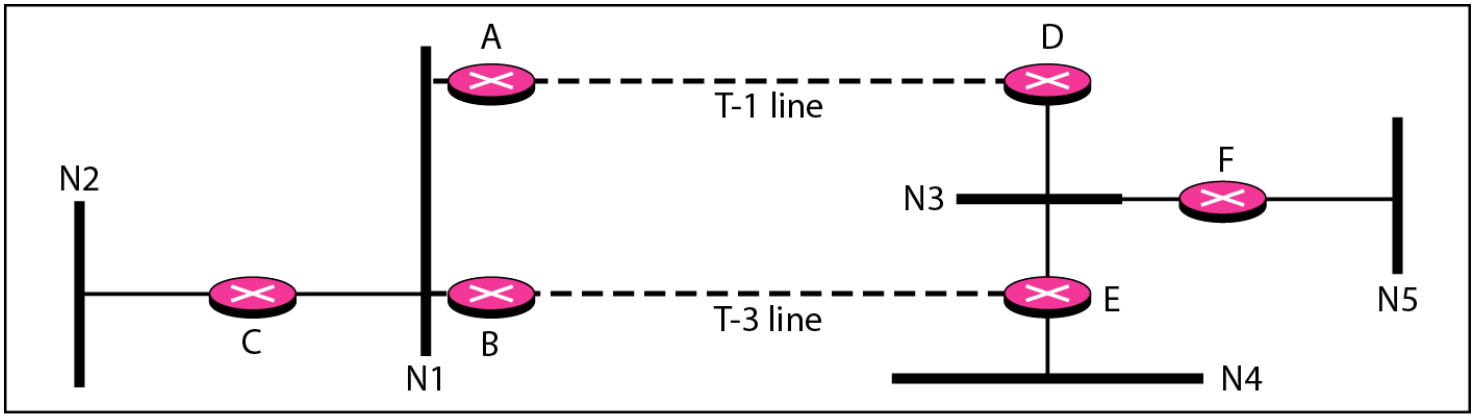
Figure 22.28 *Stub link*



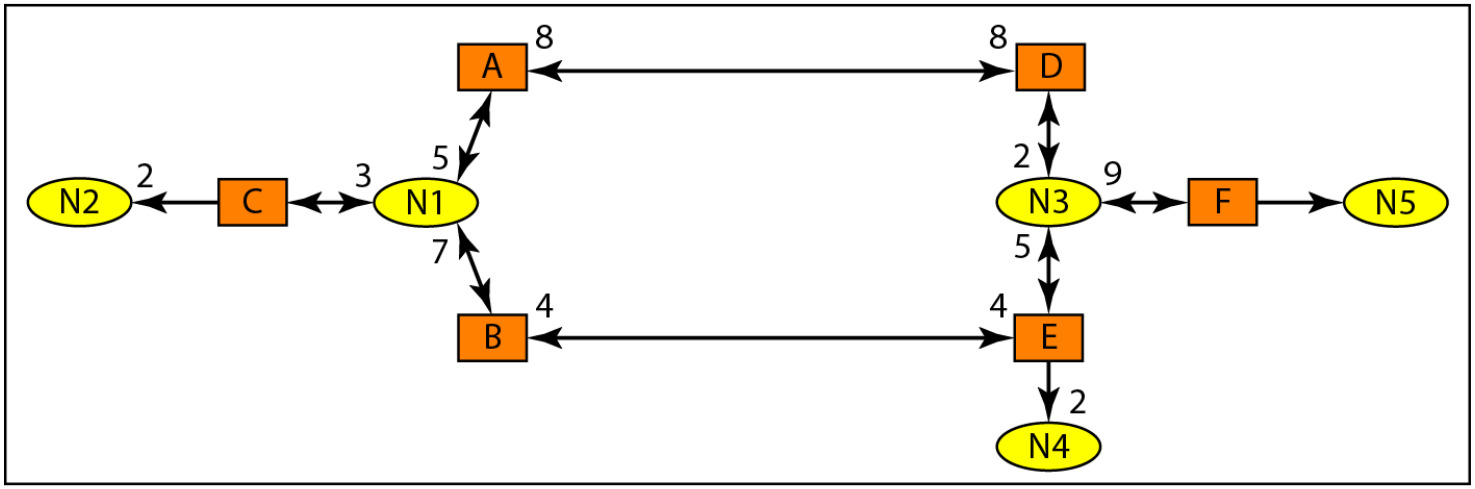
a. Stub network

b. Representation

Figure 22.29 *Example of an AS and its graphical representation in OSPF*



a. Autonomous system



b. Graphical representation

Figure 22.30 *Initial routing tables in path vector routing*

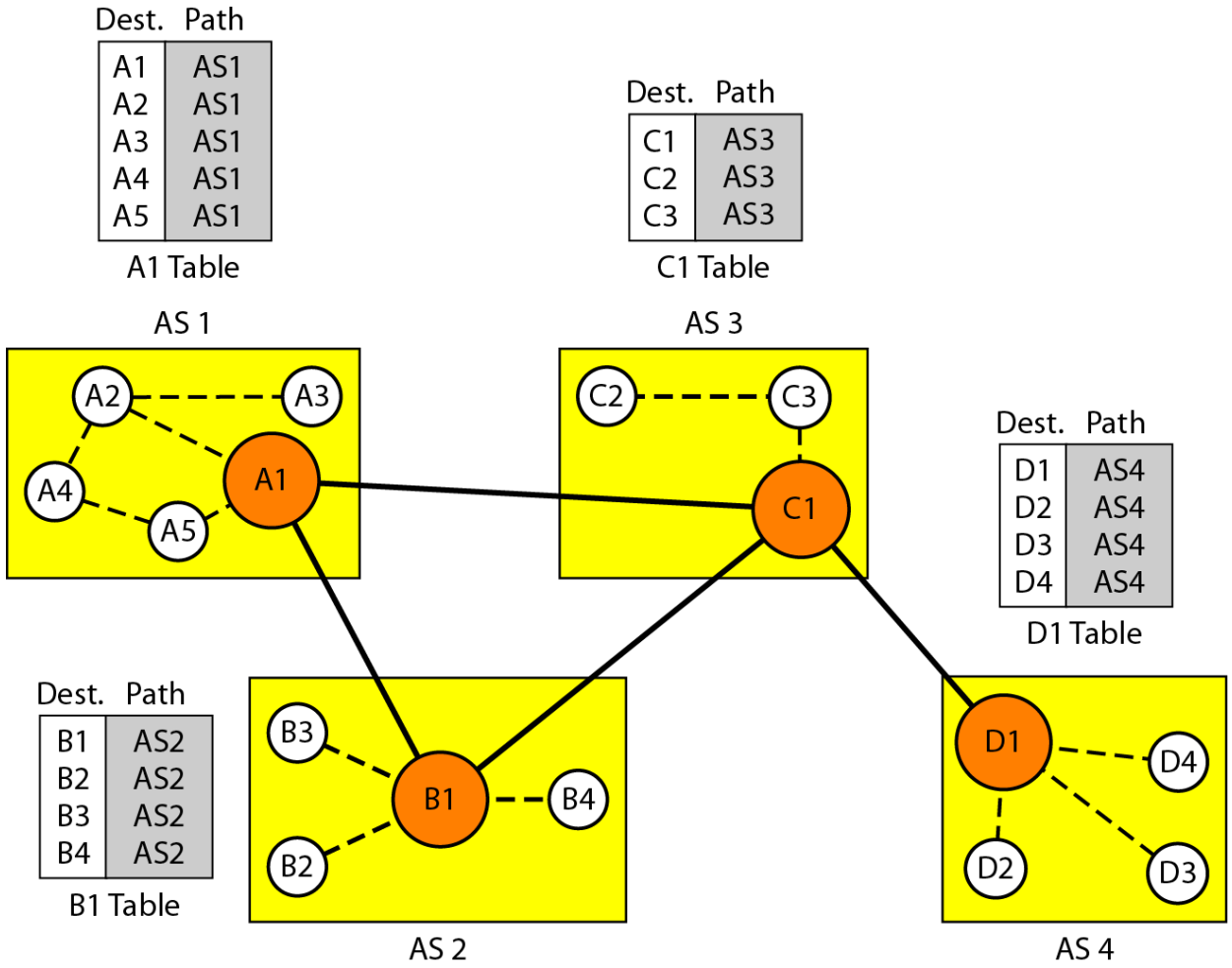


Figure 22.31 *Stabilized tables for three autonomous systems*

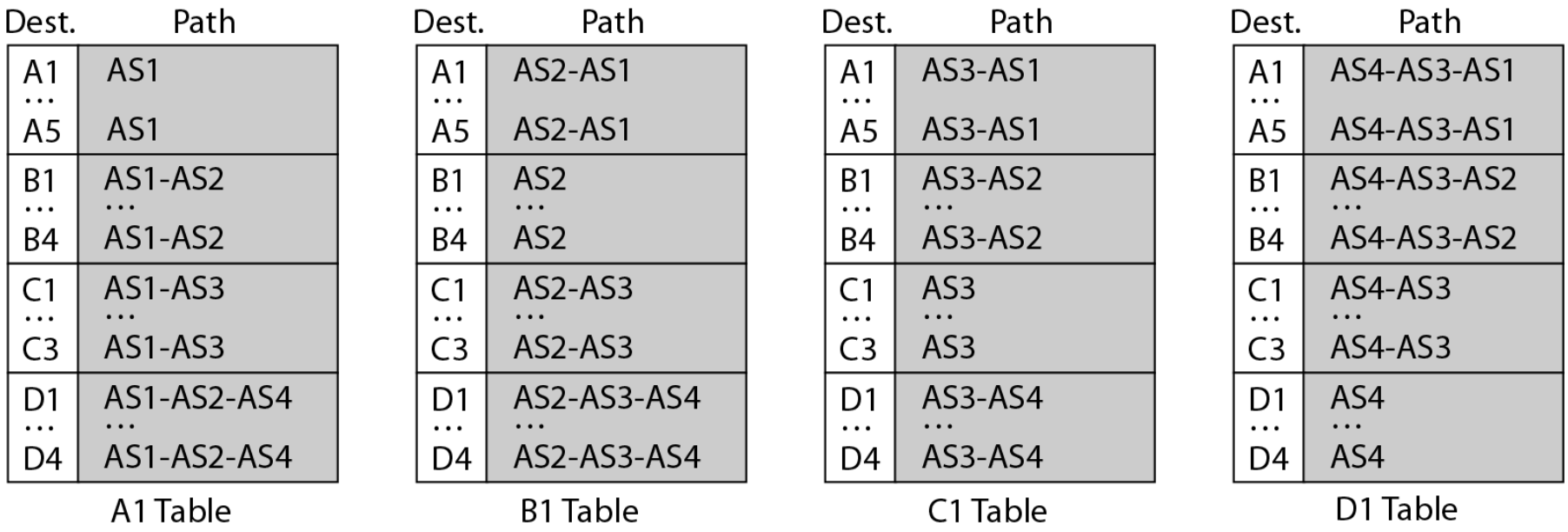


Figure 22.32 *Internal and external BGP sessions*