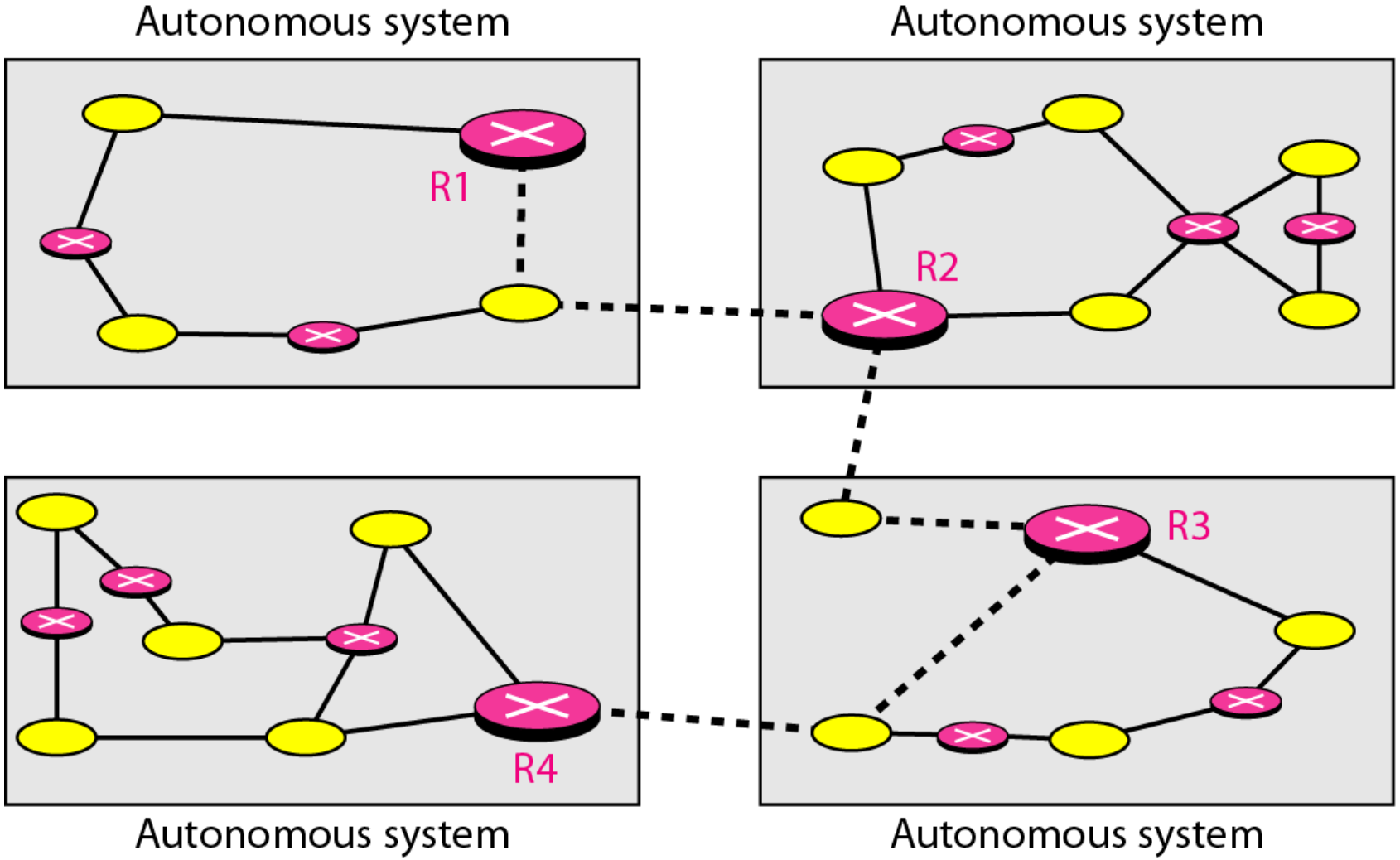
Traditional Routing Protocols

Beulah A. AP/CSE

Interdomain vs Intradomain Routing

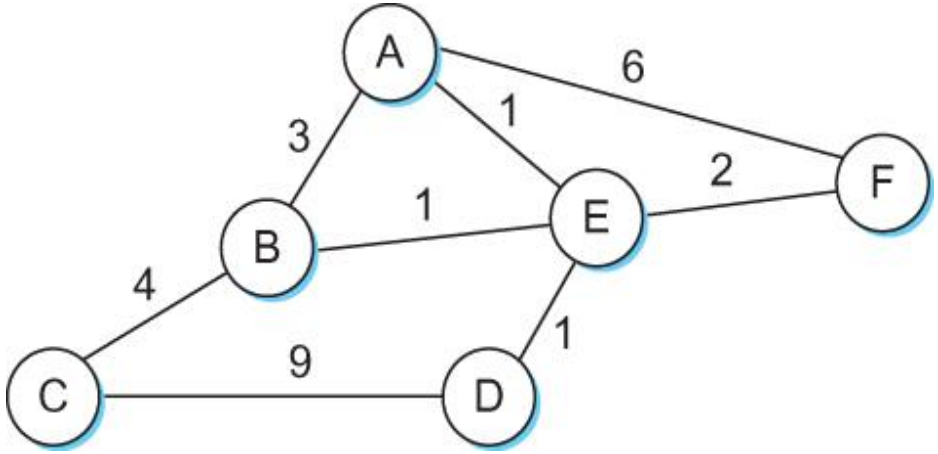
- Autonomous System
  - A group of networks and routers under the authority of a single administration.
- Intradomain Routing
  - Routing Inside autonomous systems
- Interdomain Routing
  - Routing between autonomous systems.

Interdomain vs Intradomain Routing



Network as a Graph

- The network is considered as a graph
- The basic problem of routing is to find the lowest-cost path between any two nodes
  - Where the cost of a path equals the sum of the costs of all the edges that make up the path

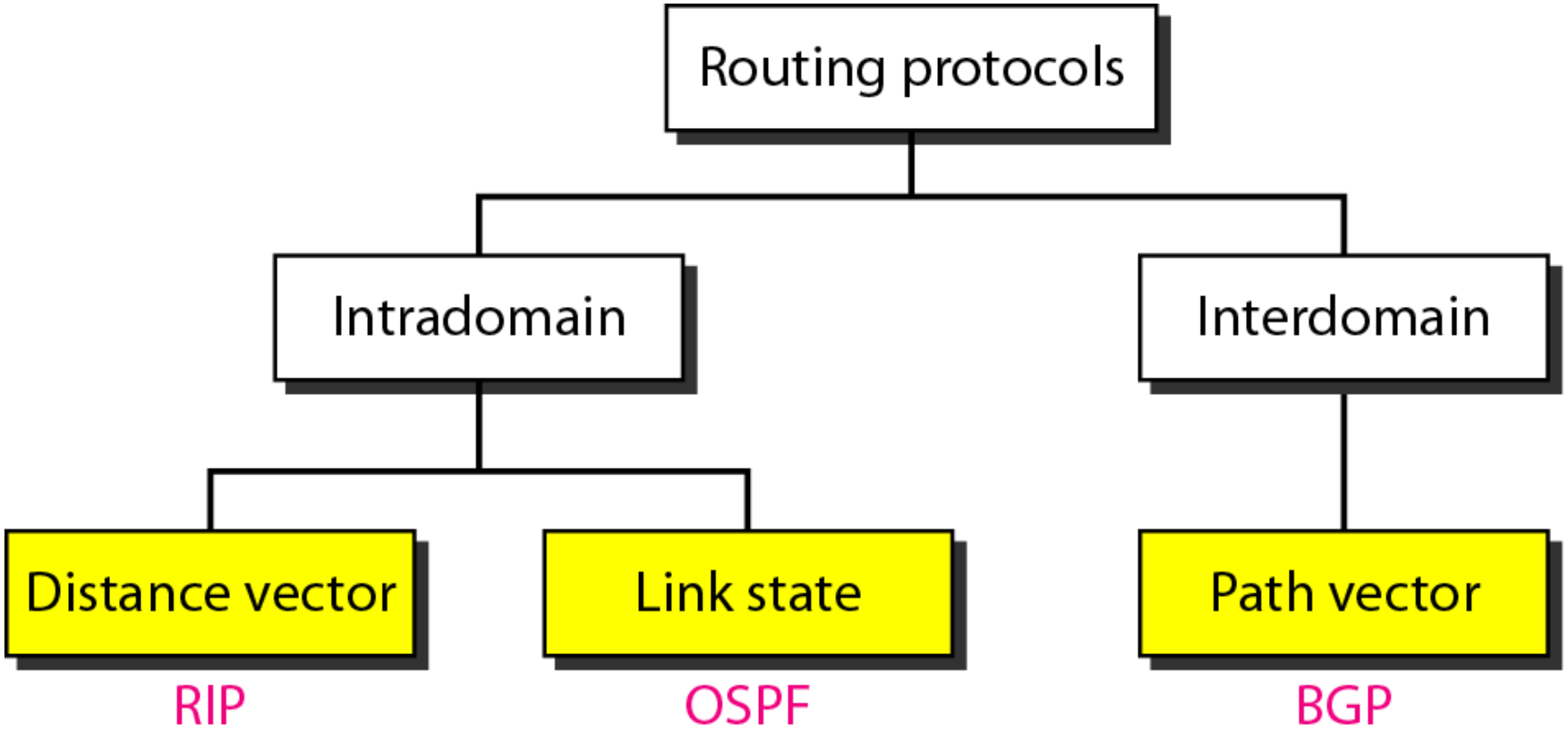


Network as a Graph

- For a simple network, calculate all shortest paths and load them into some nonvolatile storage on each node.
- Such a static approach has several shortcomings
  - It does not deal with node or link failures
  - It does not consider the addition of new nodes or links
  - It implies that edge costs cannot change

Network as a Graph

- What is the solution?
  - Need a distributed and dynamic protocol



Distance Vector Routing

- Commonly called as Distributed Bellman Ford Routing Algorithm and the Ford-Fulkerson Algorithm
- Completely decentralized algorithm
  - No node has complete information about the costs of all network links
- Gradual calculation of path by exchanging information with neighbors

Distance Vector Routing

- Each node constructs a one-dimensional array (vector) containing the distances (costs) to all other nodes (as it relates to its knowledge) and distributes it to its immediate neighbors.
- Each node knows the cost of links to its immediate neighbor.
- If no link exists between two nodes, the cost between the nodes is marked as infinity.

Representation

- The cost from X to Y via Z is the cost from X to Z plus the "minimum" cost from Z to Y.

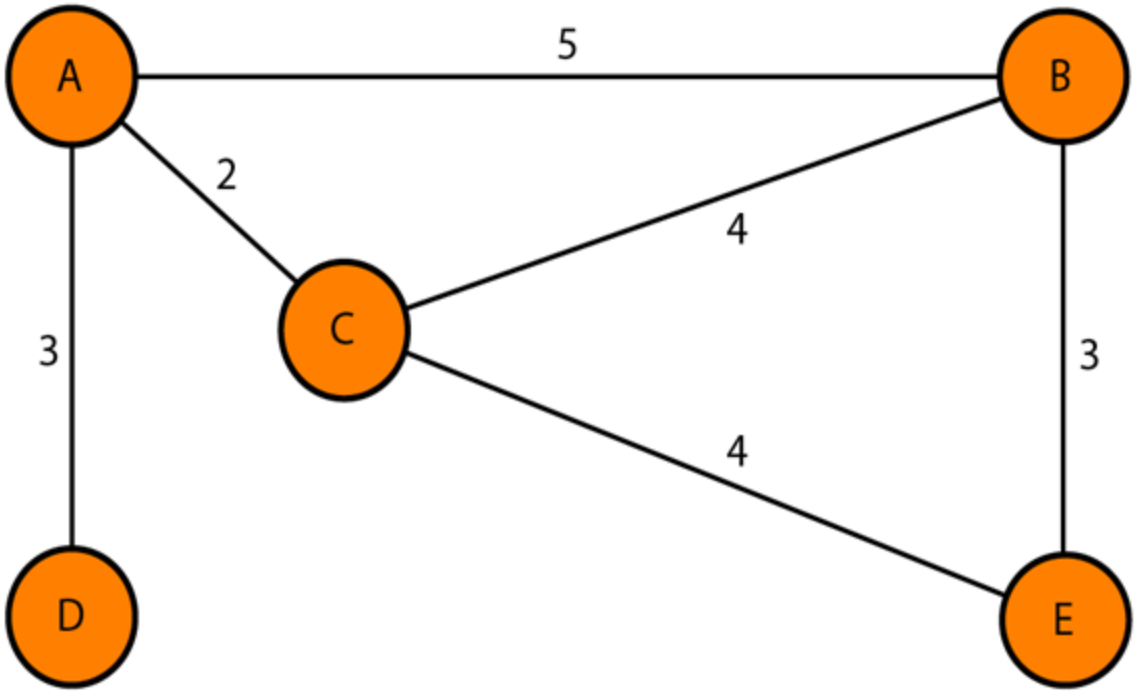
- $D^x(Y,Z) = C(X, Z) + \min_W \{ D^z(Y,W)\}$
  - $D^x(Y,Z)$ → Cost from X to Y via Z
  - $C(X, Z)$ → Cost from X to Z
  - $\min_W \{ D^z(Y,W)\}$ → Minimum cost from Z to Y via W.

- Minimum cost from Z to Y computed by taking all possible paths into consideration.

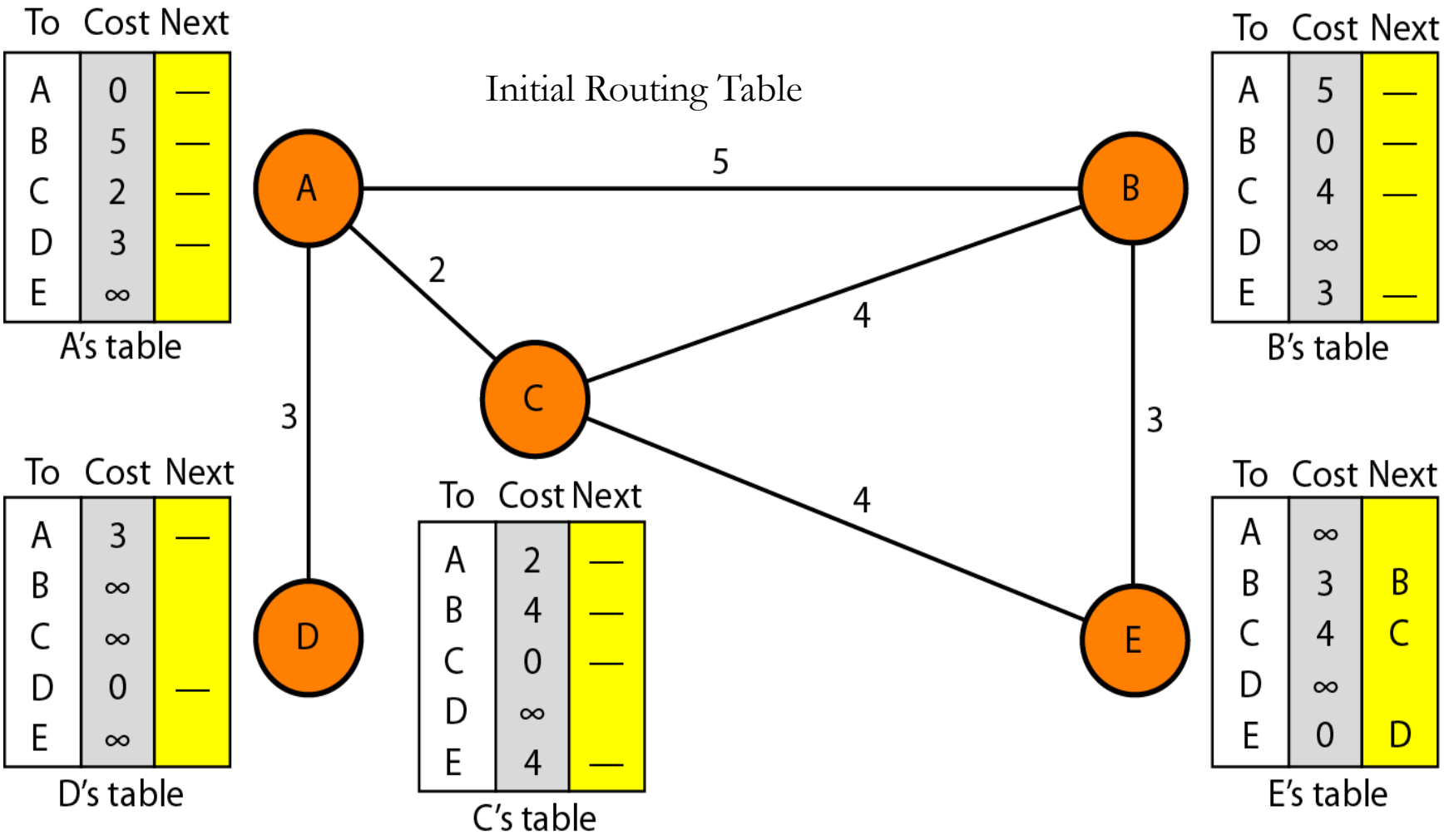
Routing Table

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

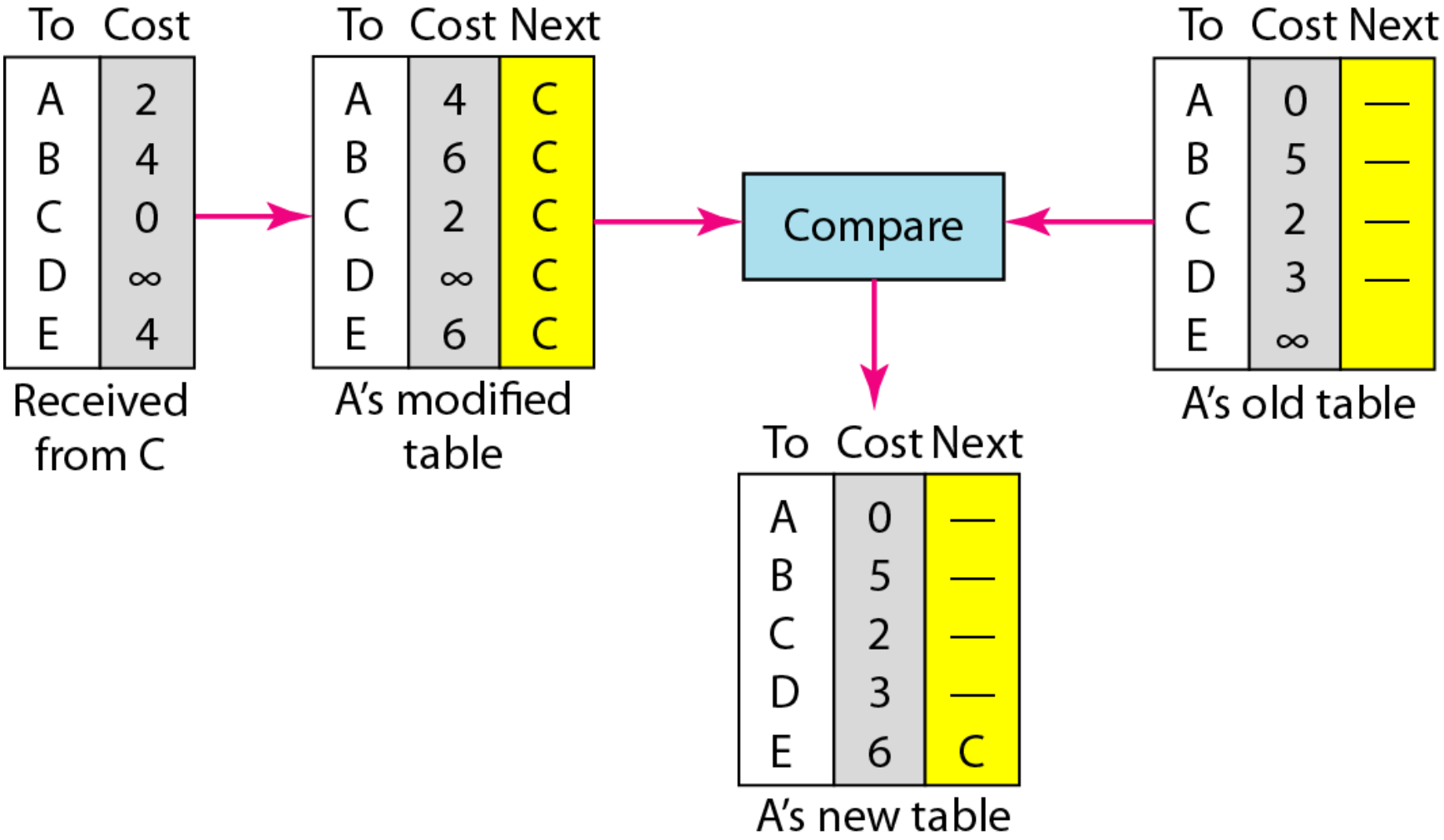
A's table



Routing Table



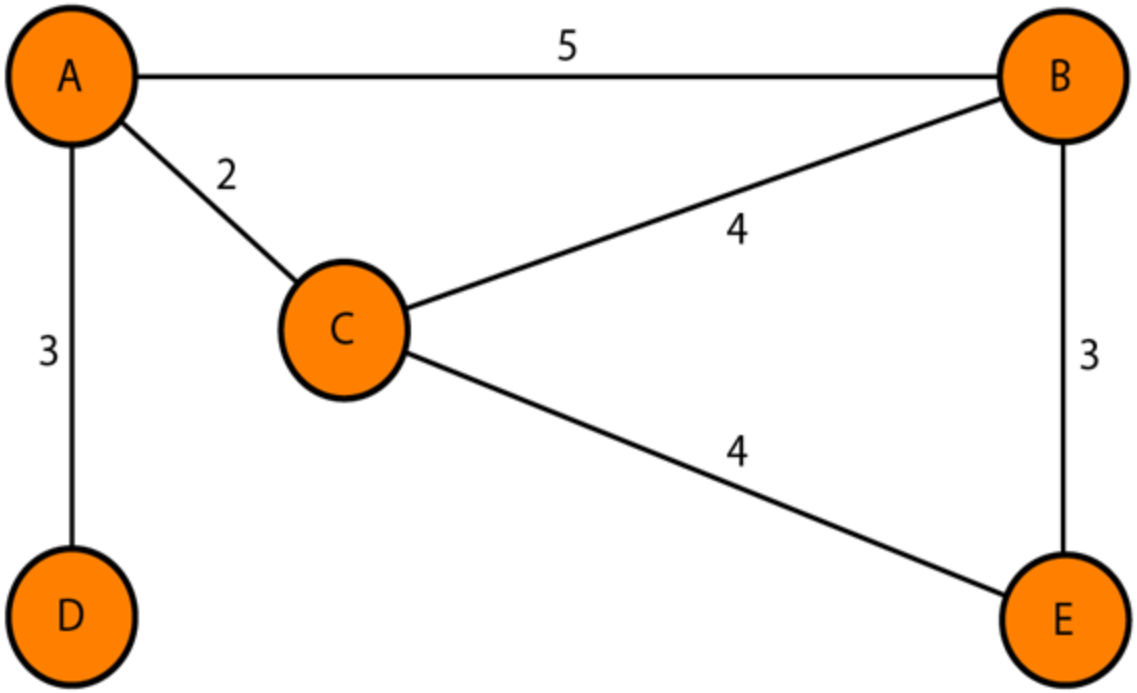
Routing Table



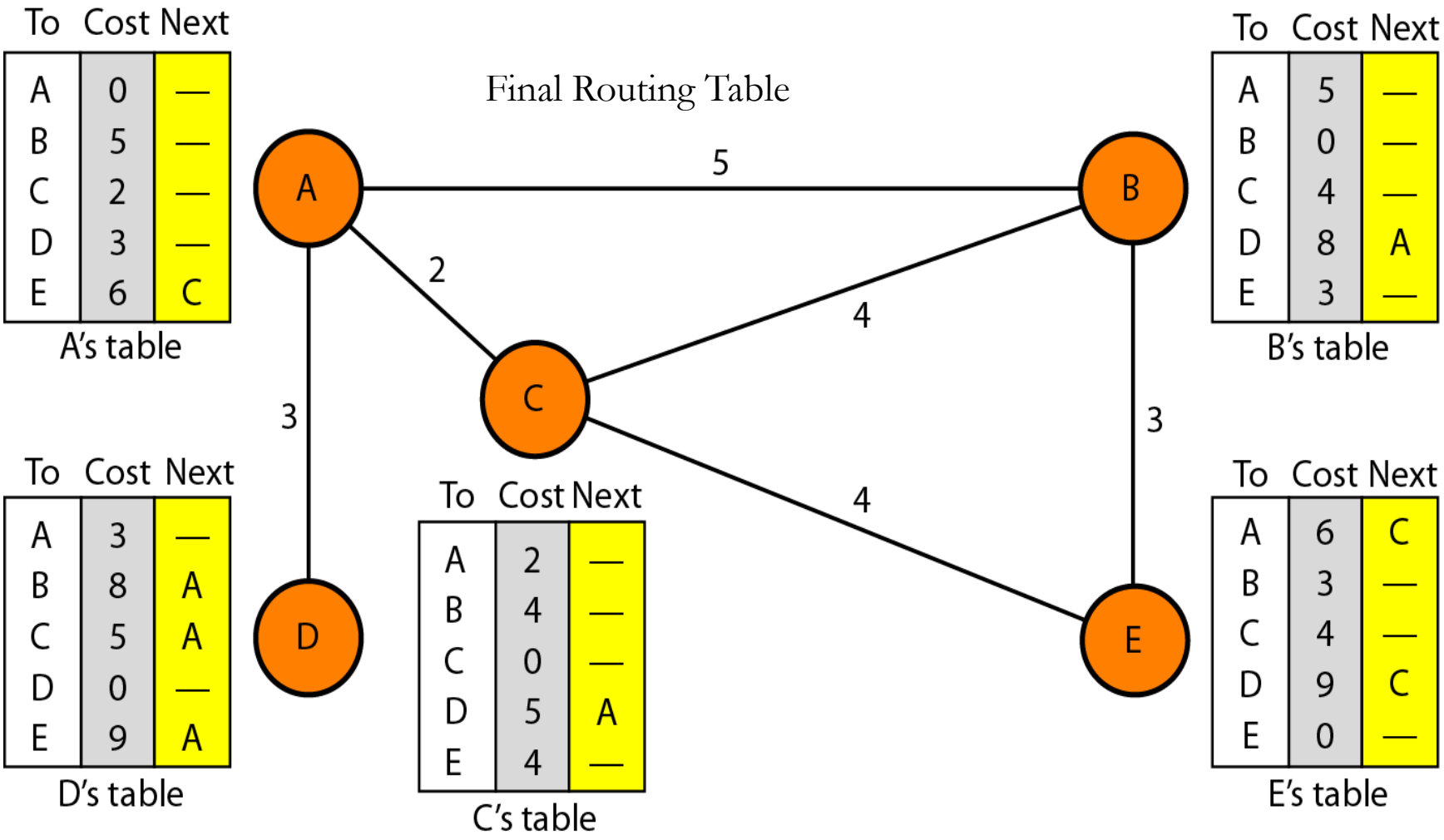
Routing Table

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

A's table



Routing Table



Convergence

- In the absence of topological changes -- few exchanges between neighbors before complete routing table is formed.
- As this table is consistent, Convergence is achieved.
- Notice -- no centralized authority
- There is no one node in the network that has all the information in the table—each node only knows about the contents of its own routing table

Routing Updates

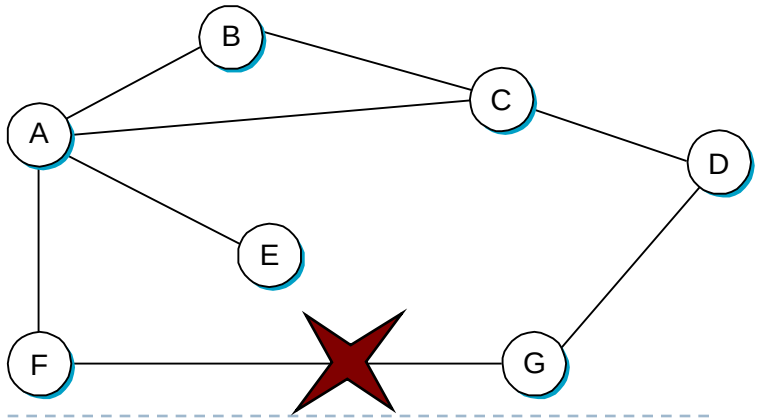
- When are routing updates sent ?
- Periodic Updates
  - Even if nothing has changed, send periodically. Every T seconds.
  - Main reason is to let other nodes know that the sender is alive.
  - Refresh information that might be needed if some of the current routes become unavailable.
- Triggered Updates
  - When a node receives an update from one of its neighbors which may lead to a change in its routing tables (could be due to change in link cost).
- Note: typically order of periodicity is seconds to several minutes.

Link/Node Failures

- Nodes that notice a failure, first send new lists of distances to neighbors. And the system settles down quickly.
- How do nodes detect failure ?
  - A node continually tests the link to another node by sending a control packet and seeing if it receives an acknowledgment
  - A node determines that the link (or the node at the other end of the link) is down if it does not receive the expected periodic routing update for the last few update cycles

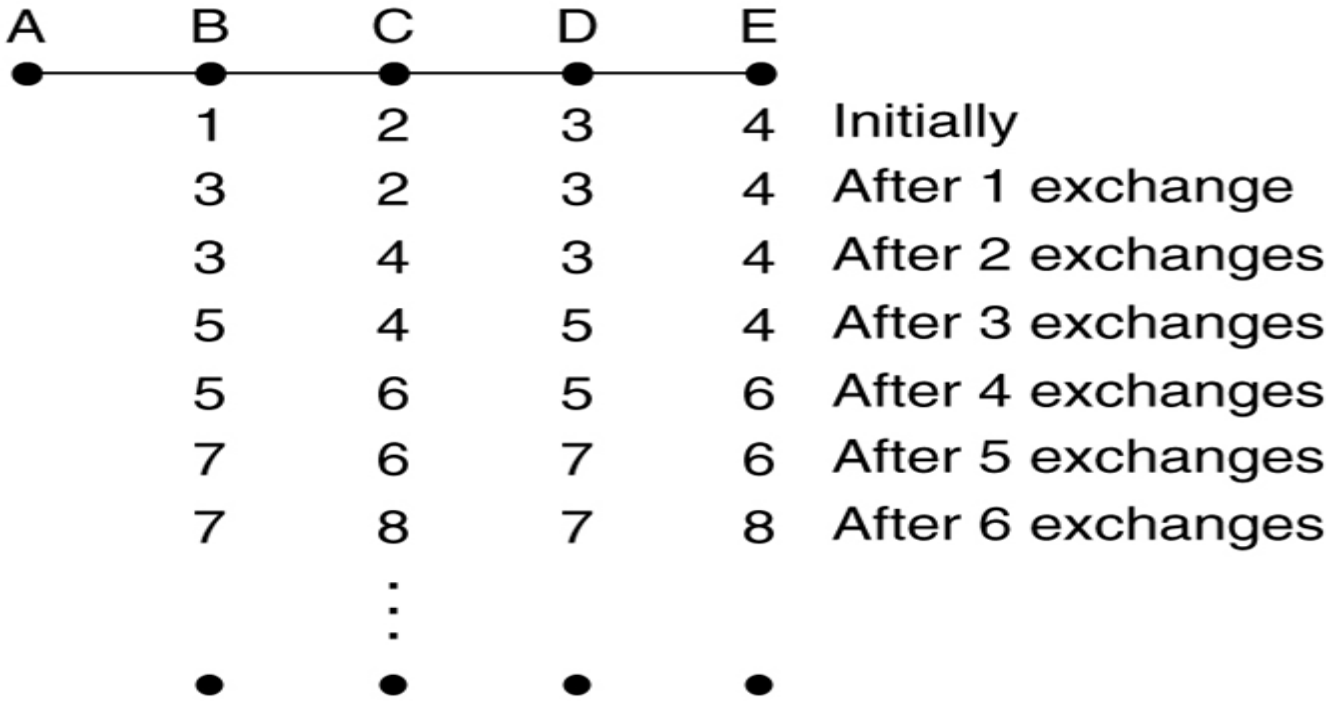
What happens when link fails?

- Let link from F to G fail.
- F sets new distance to G to ∞ ; sends update to A.
- A was initially routing to G via F. So it now sets link cost to G to ∞ .
- Next update from C; A learns that C has 2 hop path to G.
- A now can reach G in 3 hops via C.
- A sends an update to F.
- Thus, F now, can reach G via A in 4 hops.



Count to Infinity

- A goes down.
- None of the nodes actually knows that A is unreachable
- This situation is known as the count-to-infinity problem.



Split Horizon

- One solution would be to approximate ∞ to say 16 hops.
- When a node sends a routing table update to its neighbors, it "does not" send those routes it learned from "a particular" neighbor, back to that neighbor.
  - For example, C had A 2 B. When it sends a route update to B, it does not include this.
- In split horizon with poison reverse, this update is reported but the link weight is set to ∞
  - For example C sends (A, ∞) to B.

Disadvantage of DVR

- Slow convergence
- Count to infinity problem
- Lack of variety of metrics
- No possibility of hierarchical routing
- Bad performance in large networks

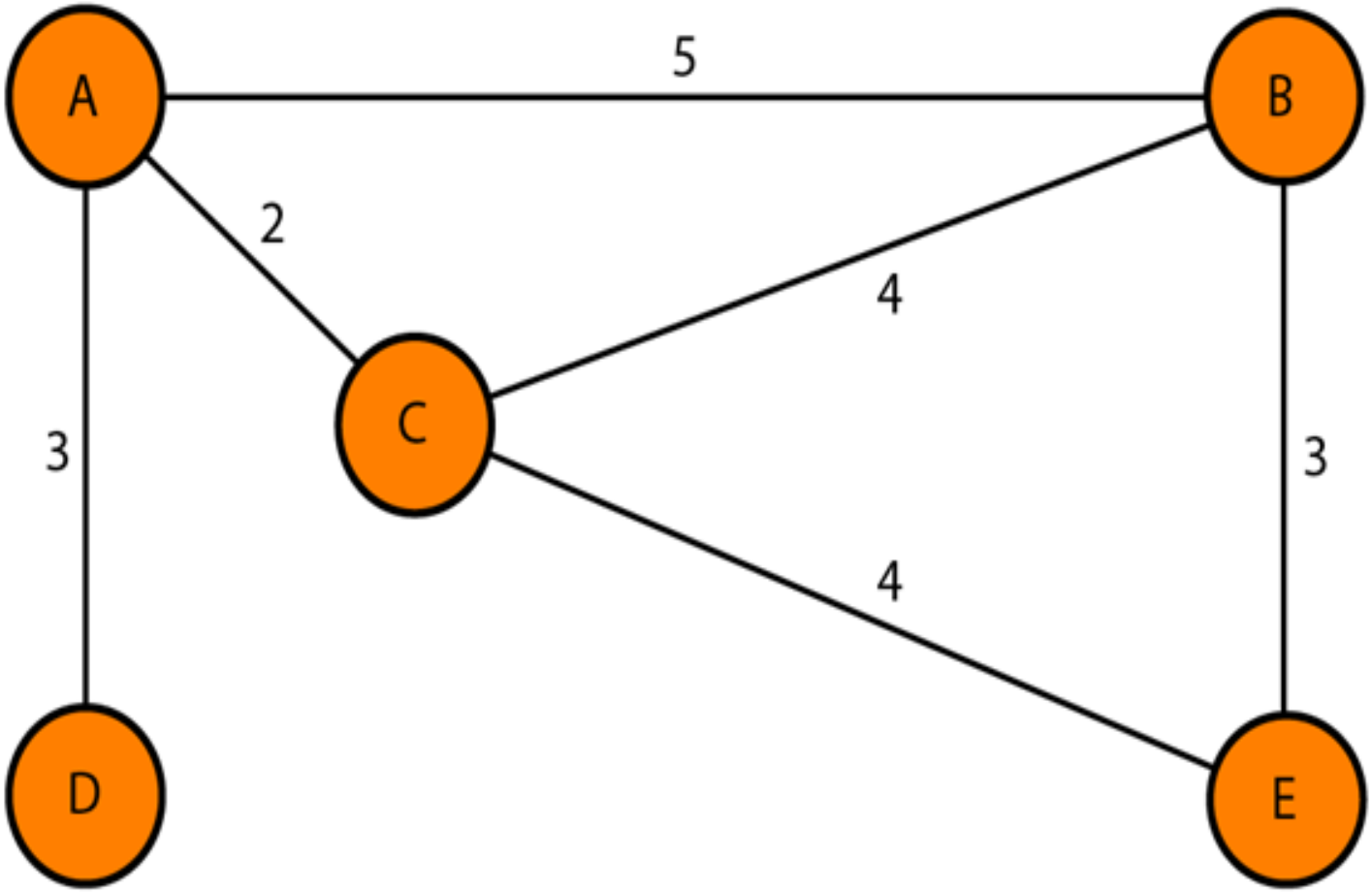
Link State Routing

- Developed to overcome the disadvantages of the distance vector protocols
- Centralized database describes the topology of the whole network
- Calculation and routing are still distributed

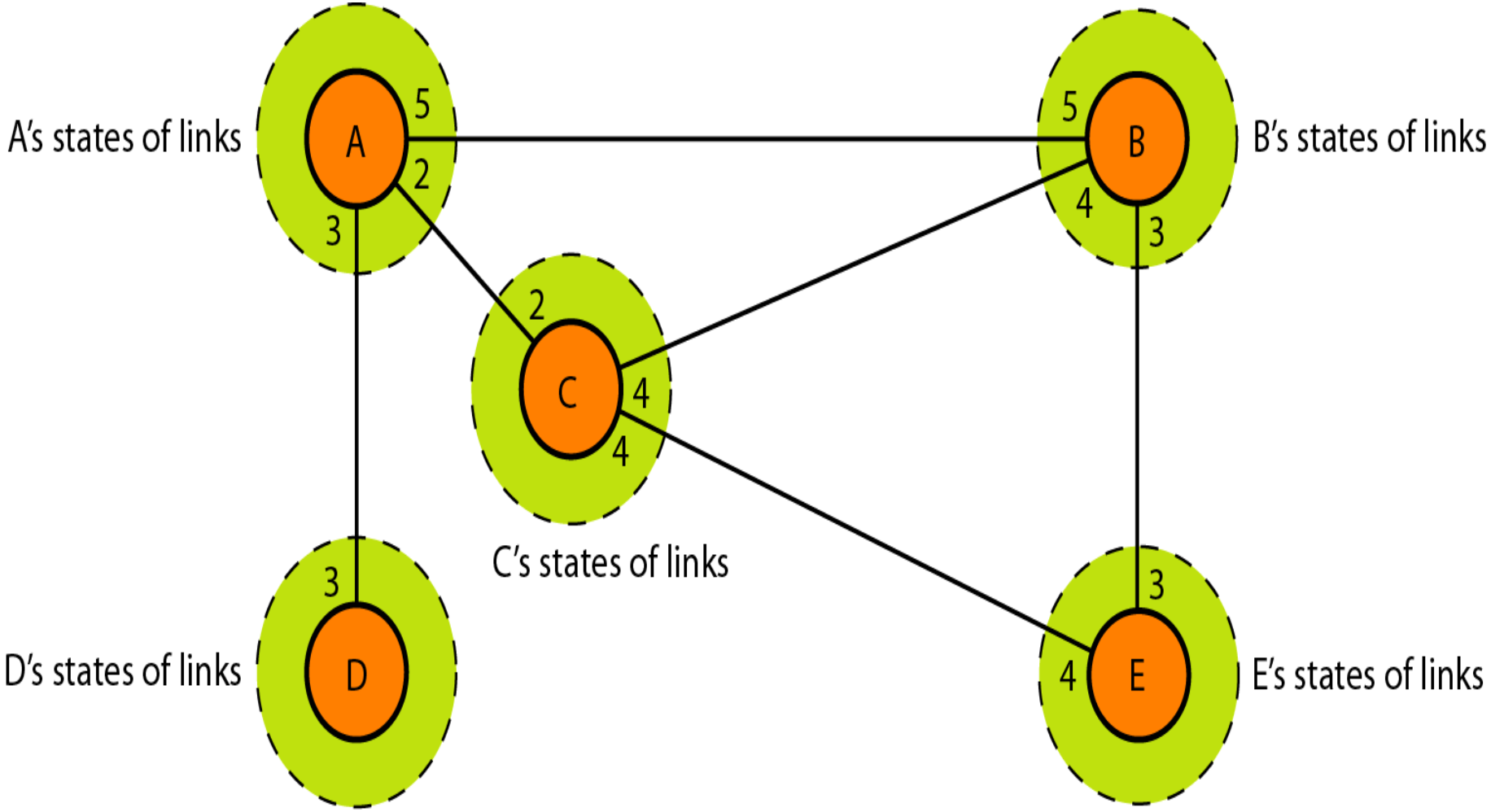
Introduction

- Initial state : similar to distance vector i.e., state of link to neighbors known.
- Goal: To find the path of least cost to destination.
- Basic Idea -- Every node knows how to reach its neighbors. If this info is dissemination (broadcast) to every node, every node ultimately has the information to build the complete map of the network.

Link state Knowledge



Link state Knowledge



Mechanisms

- Link State Packet (LSP) → Creation of states of links by each node
- Reliable Flooding →Reliable dissemination of link state information to every other router
- Formation of shortest path tree for each node
- Calculation of routing table based on the shortest path tree → Dijkstra's Algorithm

Link State Information

- Each node creates a Link State Packet (LSP) that contains:
  - ID of the node that created LSP
  - A list of directly connected nodes and the cost to reach each node.
  - Sequence Number
  - TTL

} → for reliability

Link State Information

- Sequence numbers help in identifying new information
- TTL helps in ensuring that packets don't stay in the network indefinitely.

Dissemination of LSPs

- When Dissemination of LSPs are done?
- Periodic Updates
  - LSPs are sent periodically (upon the expiry of a timer)
  - Periodicity is of the order of hours.
- Triggered Updates
  - Due to a change in topology

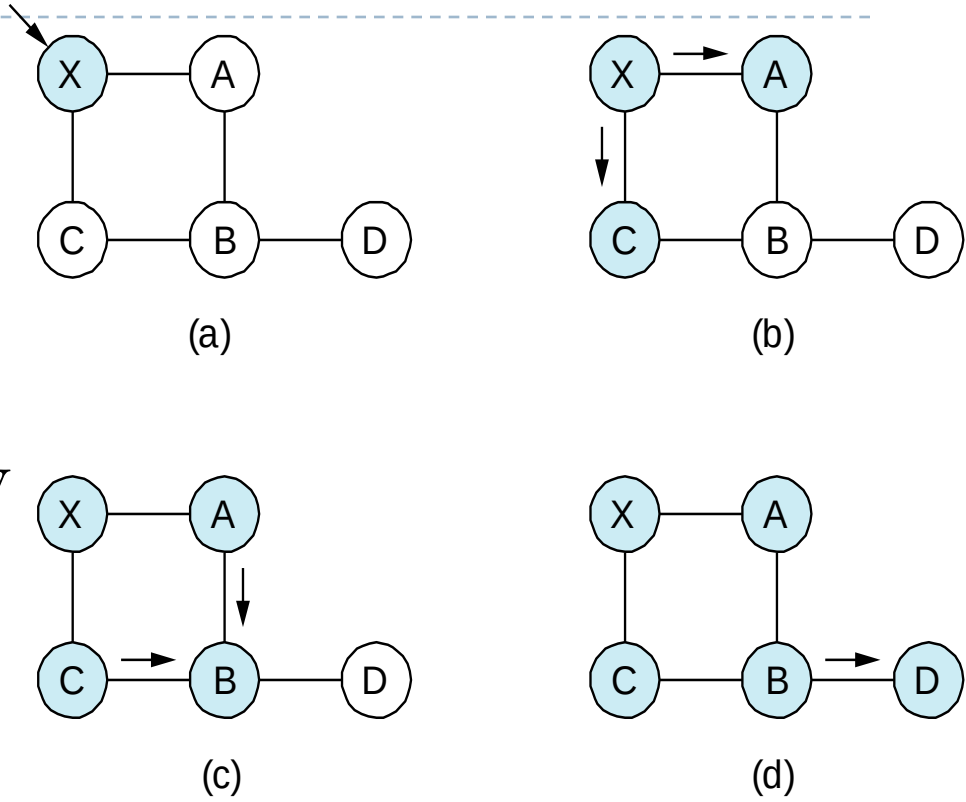
- To minimize overhead, LSPs are not created unless needed.

Reliable Flooding

- Store most recent LSP from each node
- Forward LSP to all nodes but not to the node that sent it
- Generate new LSP periodically; increment SEQNO
- Start SEQNO at 0 when reboot
- Decrement TTL of each stored LSP; discard when TTL=0

An Example

- X receives LSP from some node Y.
- X checks to see if it already has an update from Y. If it does, it compares the sequence number in the new LSP to the one stored.
- If New SEQNO < Old SEQNO, then, discard LSP.
- Else store LSP and send the LSP to all neighbors except the one that sent the LSP.
- If no update from Y, keep it.

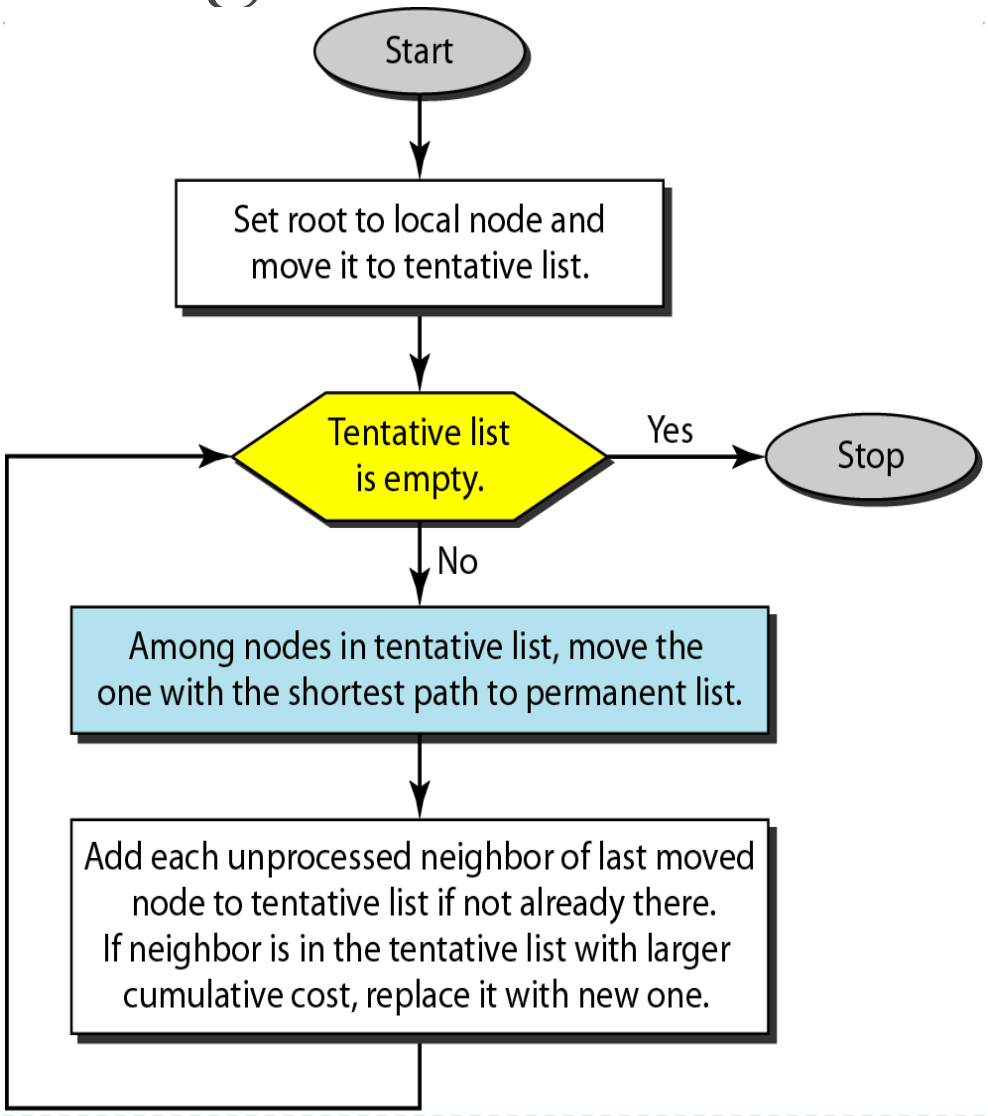


Flooding of link-state packets.
a) LSP arrives at node X;
b) X floods LSP to A and C;
c) A and C flood LSP to B (but not X);
d) flooding is complete

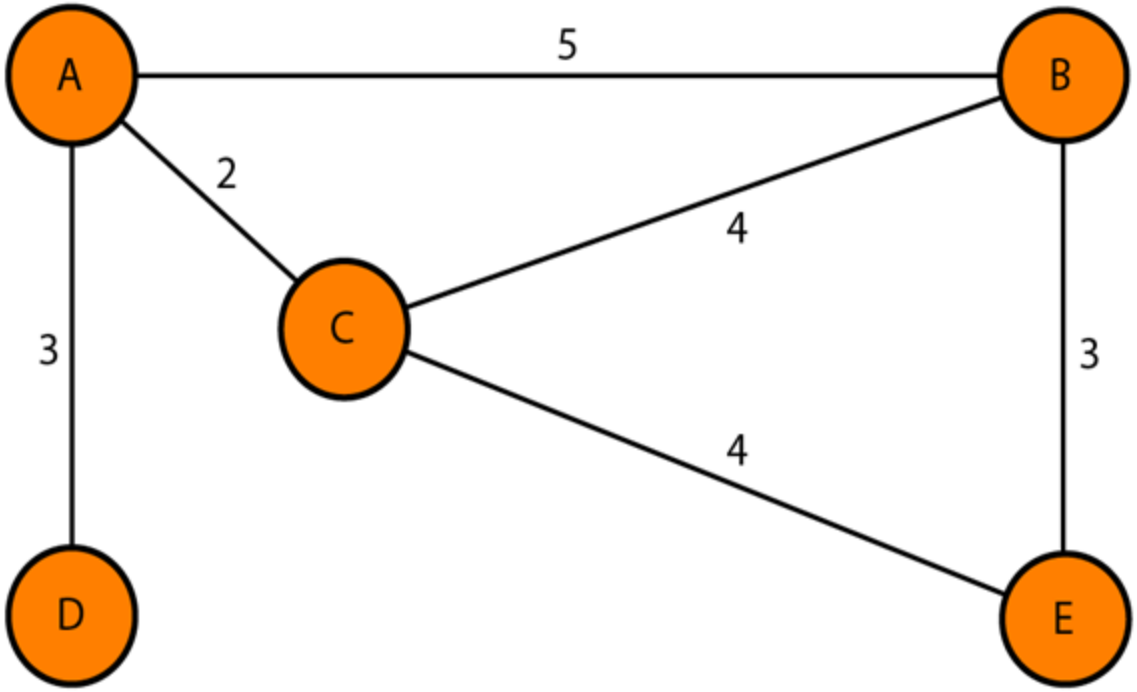
Dijkstra's Algorithm

- Each node will have the copy of whole topology, after receiving all LSP's
- Computation of shortest path using Dijkstra's.
- Nodes are divided into 2 sets:
  - Permanent list/Confirmed list
  - Tentative list
- Find the neighbors of a current node, make them tentative. If they pass the criteria, make them permanent.

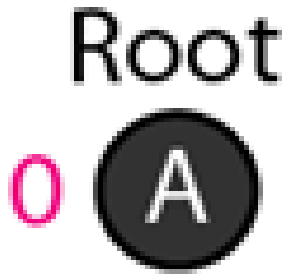
Dijkstra's Algorithm



Example

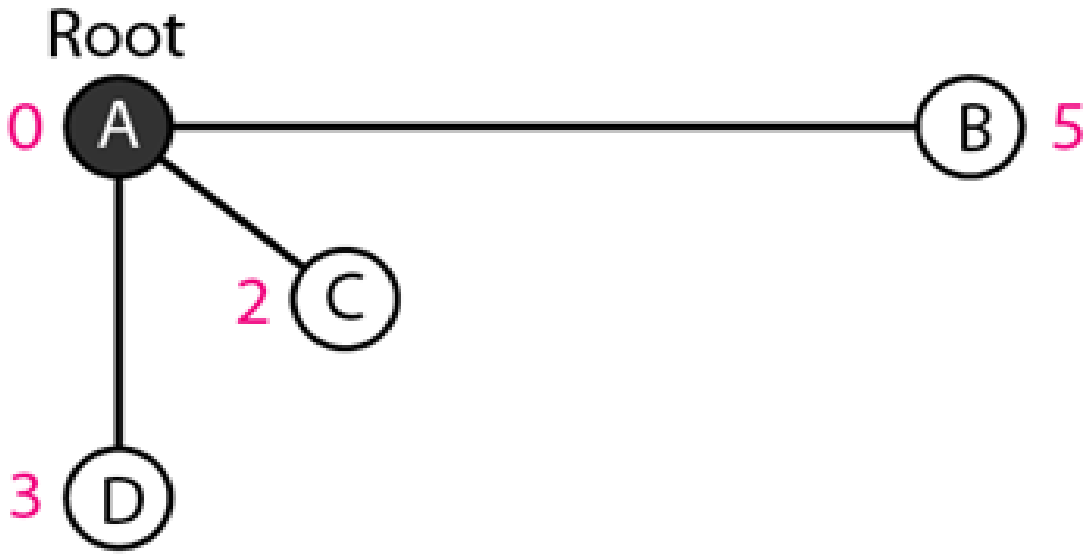


Dijkstra's Algorithm



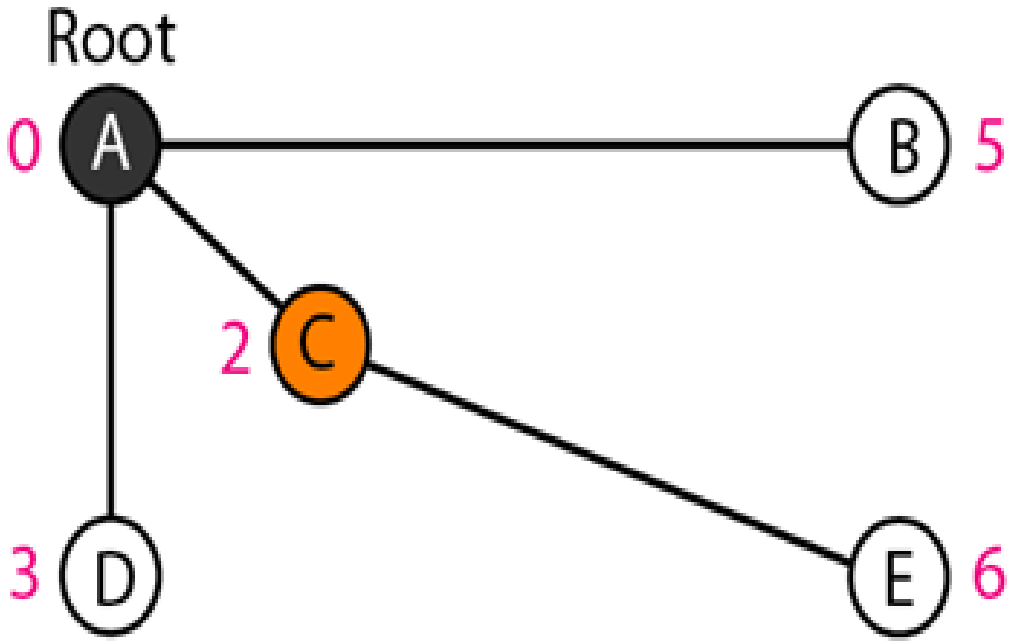
Set root to A and move A to Tentative List

Dijkstra's Algorithm



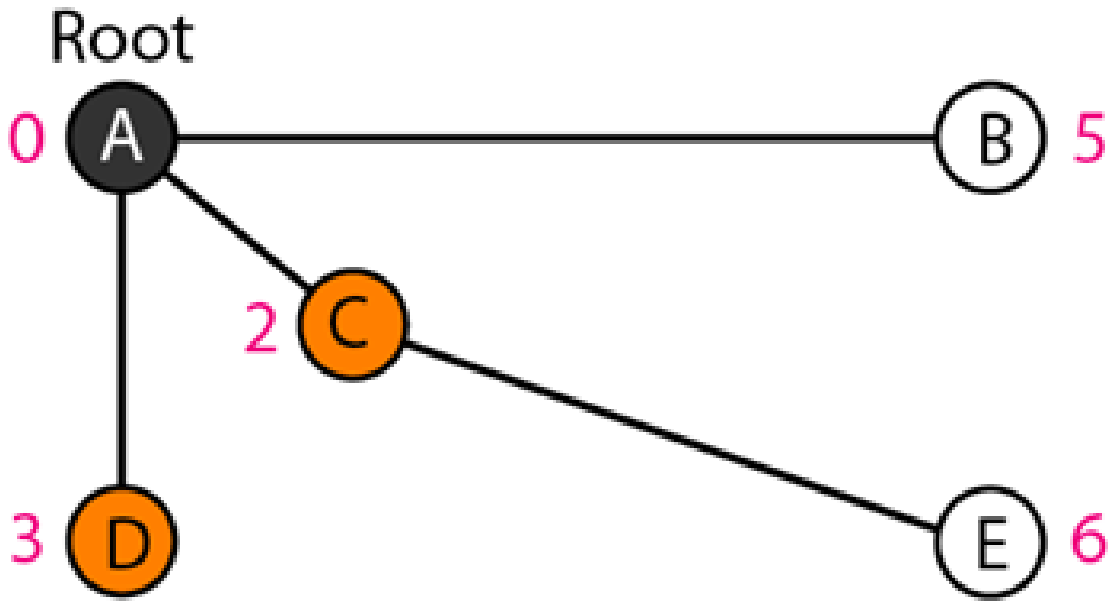
Move A to Permanent List and add B, C, and D to Tentative list

Dijkstra's Algorithm



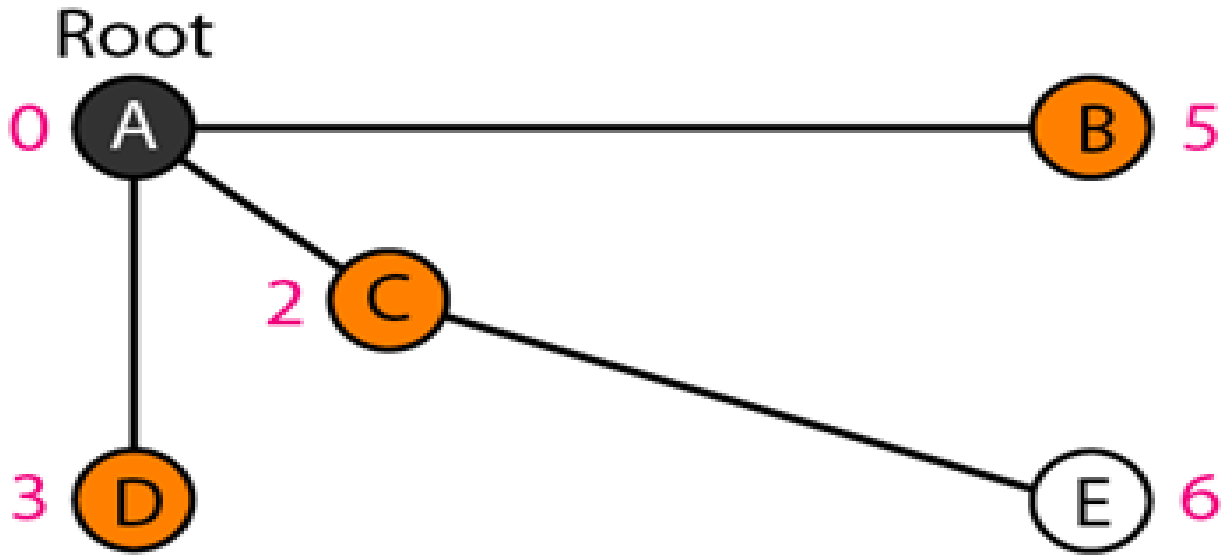
Move C to Permanent List and add E to Tentative List

Dijkstra's Algorithm



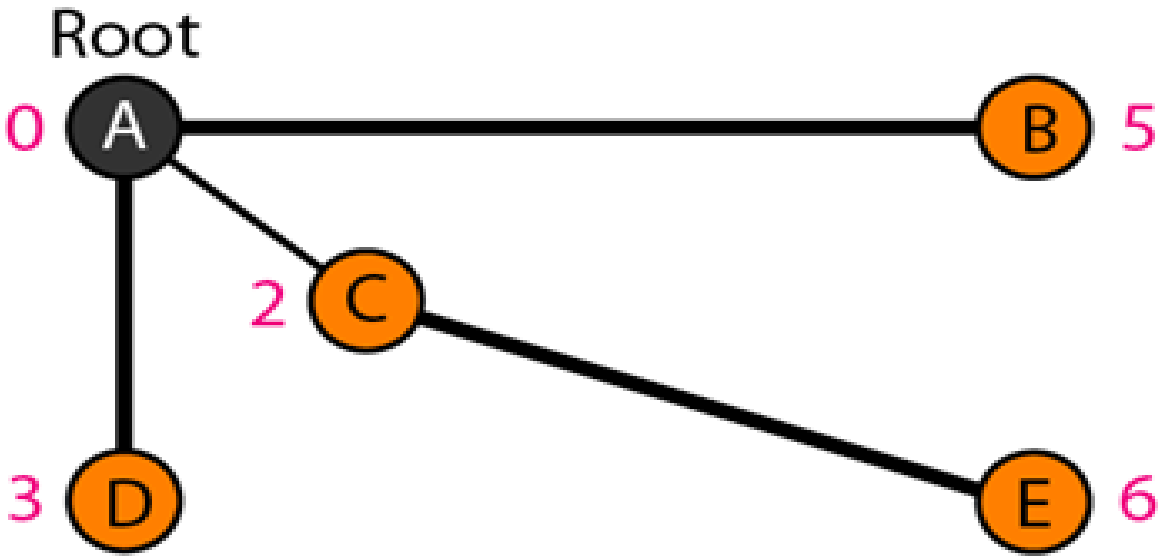
Move D to Permanent List

Dijkstra's Algorithm



Move B to Permanent List

Dijkstra's Algorithm



Move E to Permanent List. Tentative List is empty

Routing Table

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

A's table