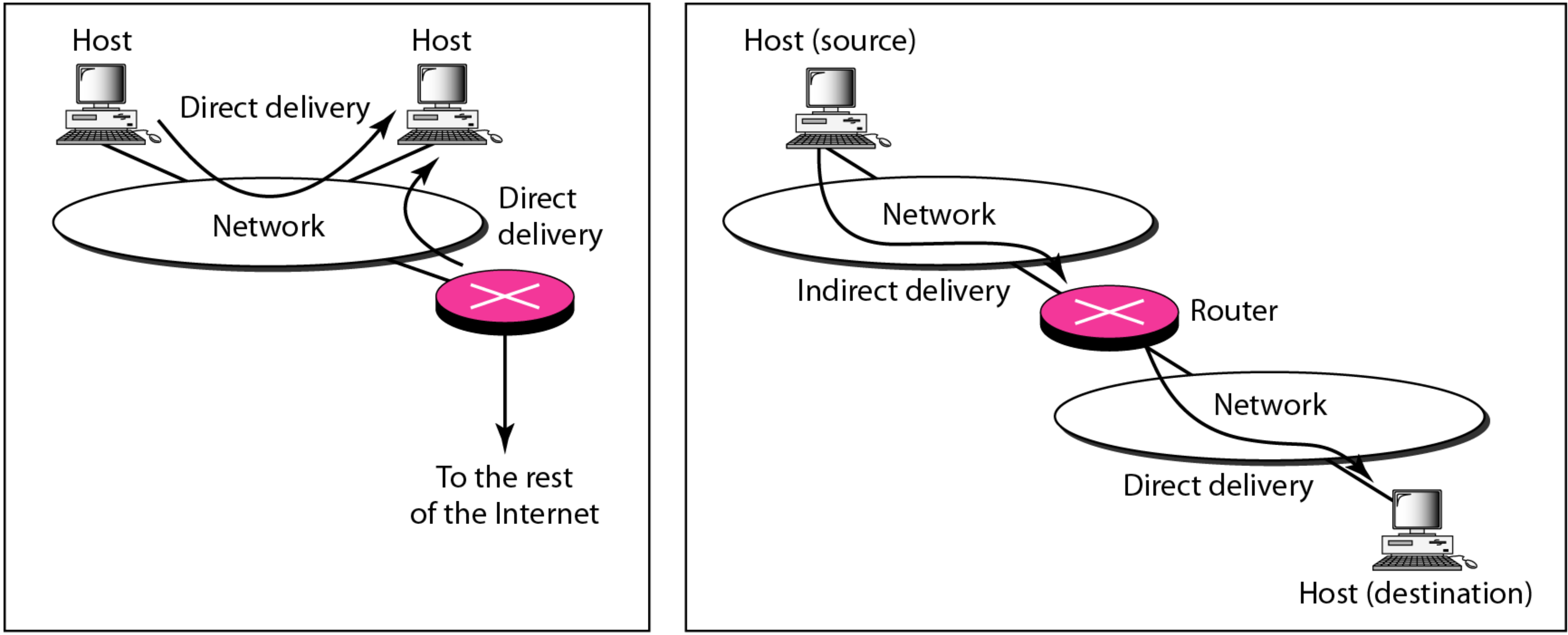
# Delivery, Forwarding and Routing

# Delivery

- Refers to the way the packets are handled by the internetworking devices.




a. Direct delivery

b. Indirect and direct delivery

# Forwarding

- Refers to the way the packets are delivered to the next node with the help of routing table.
- Forwarding Techniques
  - Route method
  - Next-hop method
  - Host-specific method
  - Network-specific method

# Route vs. next-hop method

**Routing table for host A**

a. Routing tables based on route

| Destination | Route |
|---|---|
| Host B | R1, R2, host B |

b. Routing tables based on next hop

| Destination | Next hop |
|---|---|
| Host B | R1 |

**Routing table for R1**

a. Routing tables based on route

| Destination | Route |
|---|---|
| Host B | R2, host B |

b. Routing tables based on next hop

| Destination | Next hop |
|---|---|
| Host B | R2 |

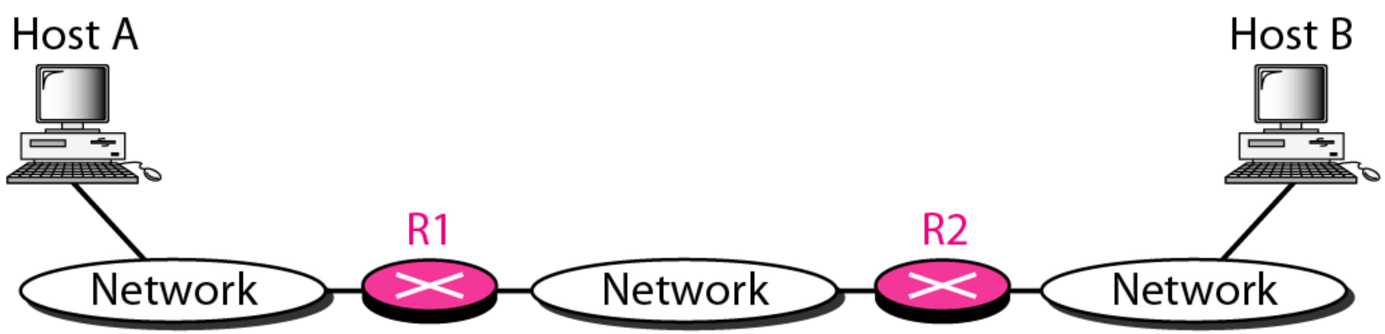
**Routing table for R2**

a. Routing tables based on route

| Destination | Route |
|---|---|
| Host B | Host B |

b. Routing tables based on next hop

| Destination | Next hop |
|---|---|
| Host B | --- |



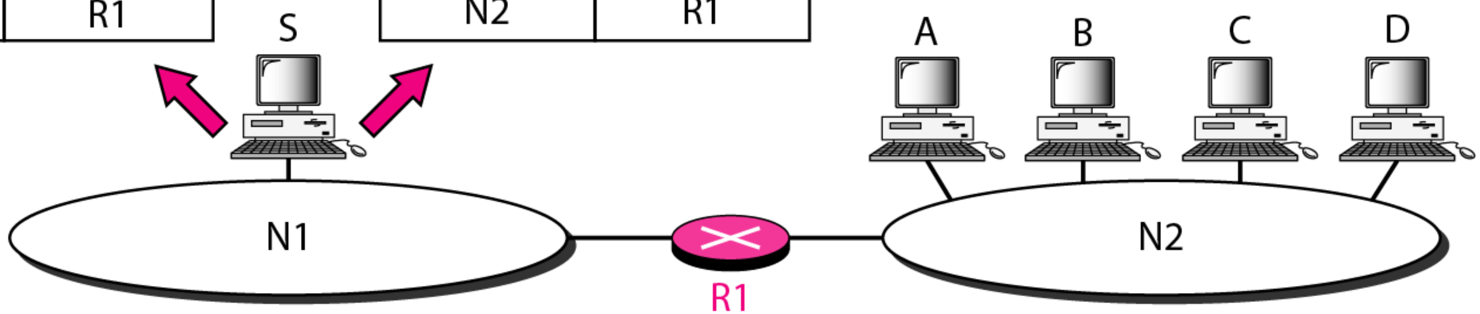
# Host-specific vs. Network-specific method

Routing table for host S based on host-specific method

| Destination | Next hop |
|---|---|
| A | R1 |
| B | R1 |
| C | R1 |
| D | R1 |

Routing table for host S based on network-specific method

| Destination | Next hop |
|---|---|
| N2 | R1 |

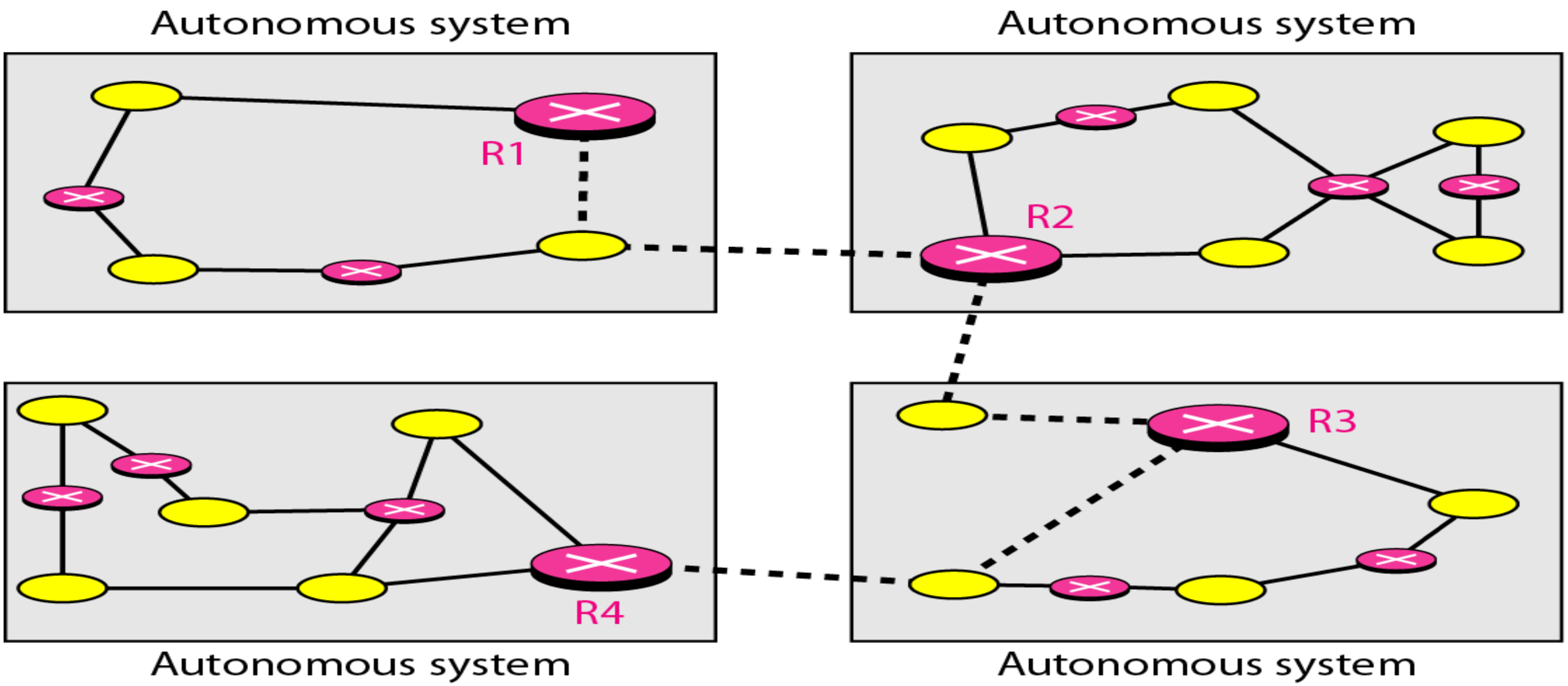


# Unicast Routing Protocols

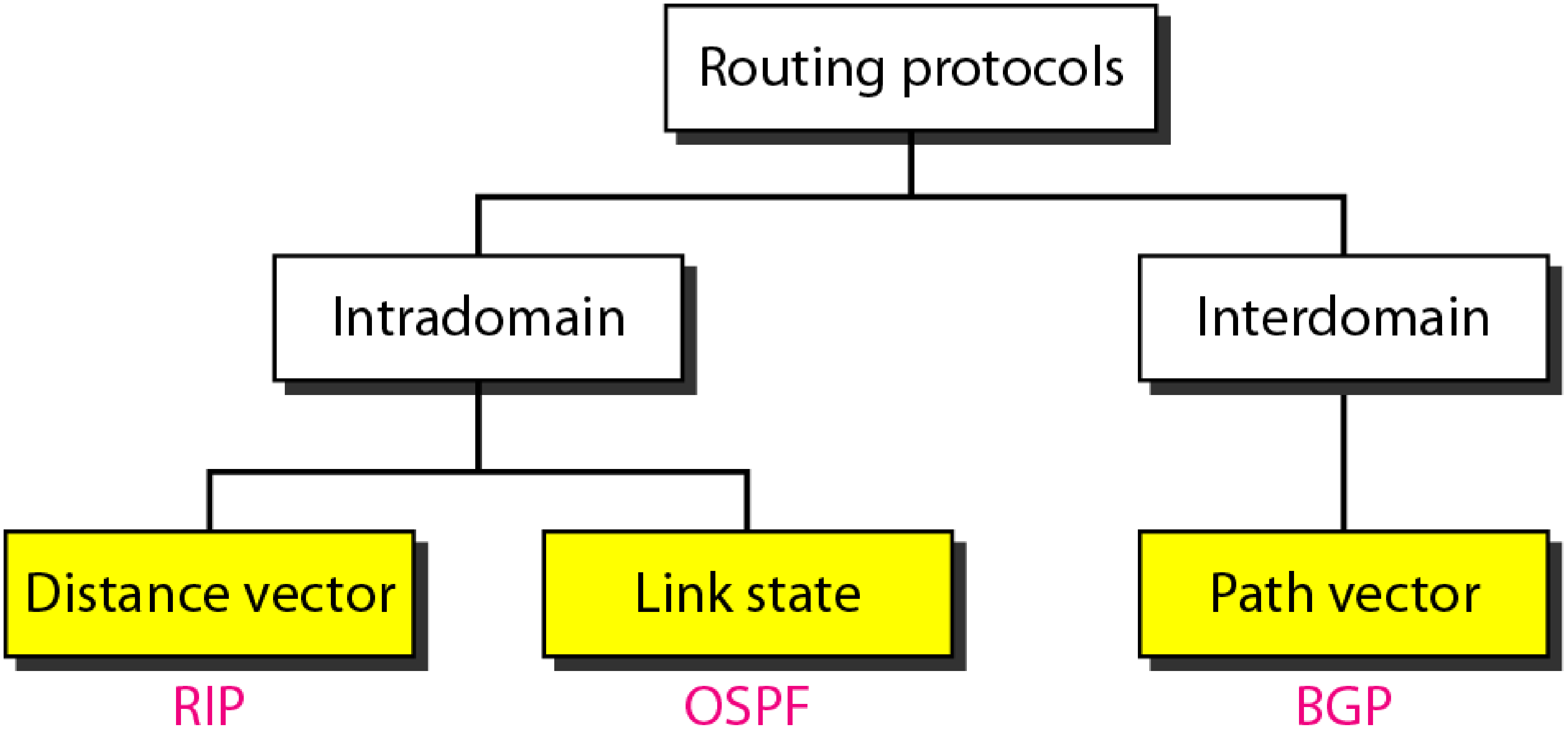
- A routing table can be either static or dynamic.
- A static table is one with manual entries.
- A dynamic table is one that is <u>updated automatically</u> when there is a change somewhere in the network.
  - Whenever the link/node is down, the routing table needs to be updated.
- Today, internet needs dynamic routing tables.

# Autonomous System

- Updating the routing table with one routing algorithm is very tedious task in an Internet.
- AS - Is a group of networks and routers under the authority of single administration.



# Popular Routing Protocols



# Distance Vector Routing

- Simple and distributed protocol.
- Used in ARPAnet.
- Goes by the name Bellman-Ford algorithm.
- Later in Internet it is used under the routing protocol standard RIP (Routing Information Protocol).
- Now, it is not much used.
- As name implies, each node maintains a vector(table) of minimum distances to every node.
  - Destination
  - Cost
  - Next Hop to reach destination
- Initial State : Each node can know only its neighbour cost.
- Final State : Each node knows cost of all nodes.
- In DV, the least-cost route between any two nodes is the route with minimum distance.
- Builds routing table using three steps.
  - Initialization
  - Sharing
  - Updating

# Initialization

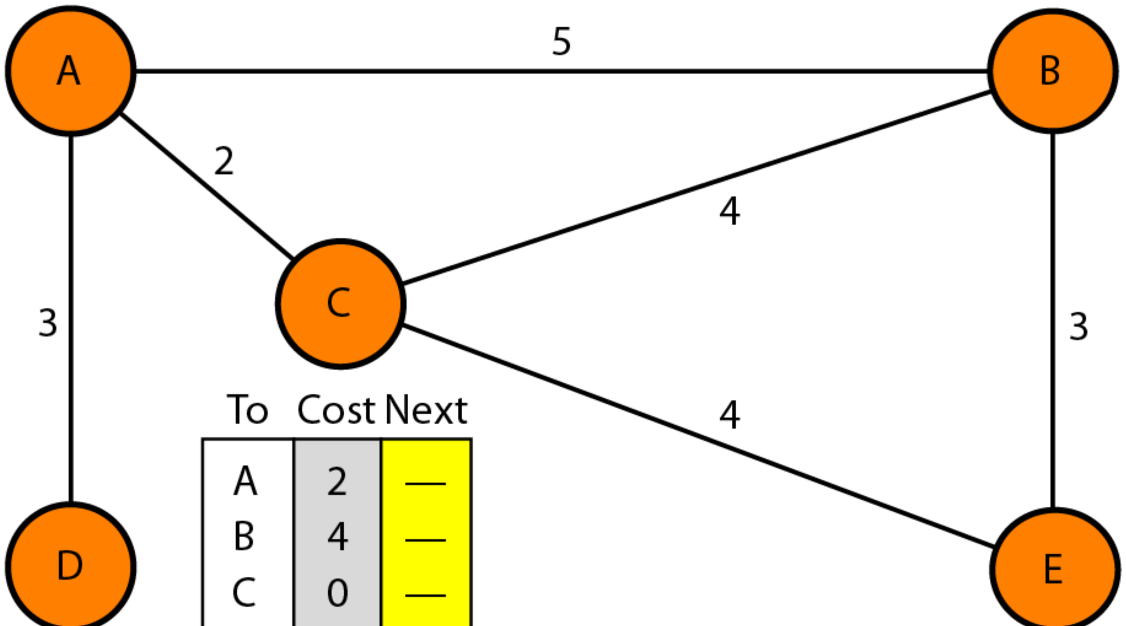
- At the beginning, each node knows only the distance between itself and its immediate(directly) connected neighbours.
- The distance for the node which is not connected directly i.e not a direct neighbour is set to infinity (∞) .

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

A’s table

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

B’s table




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

D’s table

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

C’s table

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

E’s table

# Sharing

- The whole idea behind DV routing is the sharing of information between neighbours.
- Although node A does not know about node E, but node C knows about it.
- So, if node C shares its routing table with node A, then node A can also know how to reach node E.
- Similarly, node C does not know about how to reach node D but node A knows. So if node A shares its routing table with node C then node C also knows how to reach node D.
- Exchanging routing table with neighbours can improve the routing table information.

What kind of information are exchanged?

A node can send/share only first two columns not the complete routing table.

# When to share?

When does a node sends a partial routing table (only two columns) to all its immediate neighbours?

- **Periodic Update** - A node sends its routing table to its neighbour for every 30 s. The period depends on the routing protocol.
- **Triggered Update** - A node sends its routing table to its neighbour any time when there is a change in its routing table. The triggered update makes the following changes:
  - A node receives a table from a neighbour, resulting in its own table after updating.
  - A node detects some failure in the neighbouring links which results in a distance change to infinity.

# Updating

- When node receives a two-column from a neighbor, it needs to update its routing table. Updating done in three steps:

1) The receiving node needs to add the cost between itself and the sending node to each value in the second column.
2) The receiving node needs to add the name of the sending nodes to each row as the third column if the receiving node uses the information from any row. The sending node is next node in the route.
3) The receiving node needs to compare each row of its table with the corresponding row of the modified version of received table.
   i. If the next node entry is different, the receiving node chooses with the smaller cost. If there is a tie, then the old one is kept.
   ii. If the next node entry is same, the receiving node chooses the next row .

# 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

# Points to Note

- No topology change, convergence in few seconds.
- After one message exchange, each node knows about two hops away.
- After two message exchange, each node knows about three hops away.
- And so on…
- No node has a global knowledge.

# Link State Routing

- Working principle of link state routing differs from DV routing.
- In link state routing, each node in the network knows entire network topology such as:
  - the list of nodes and links
  - how they are connected
  - cost, and
  - links up/down.
- Each node uses **Dijikstra's algorithm** to build a shortest path tree.

# Building Routing Table

- In link state routing, four sets of actions are required that each node has a least cost routing table 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 node(router) – Flooding.
  3. Formation of a shortest path tree for each node.
  4. Calculation of routing table based on a shortest path tree.

# 1. Creation of LSP

- Each LSP contains
  - Node identity
  - List of links
  - Sequence number
  - Age
- The first two values are needed to make a network topology.
- The third parameters distinguishes new LSPs from old ones.
- The fourth parameters define how long the advertised LSPs can present in the network.
- LSPs are generated on two occasions:

1. When there is a change in the topology
2. Periodic interval – 60 min – 2 hr > DV.

   NOTE: Longer periodic interval eliminates excessive traffic due to flooding

# 2. Dissemination/Flooding of LSPs

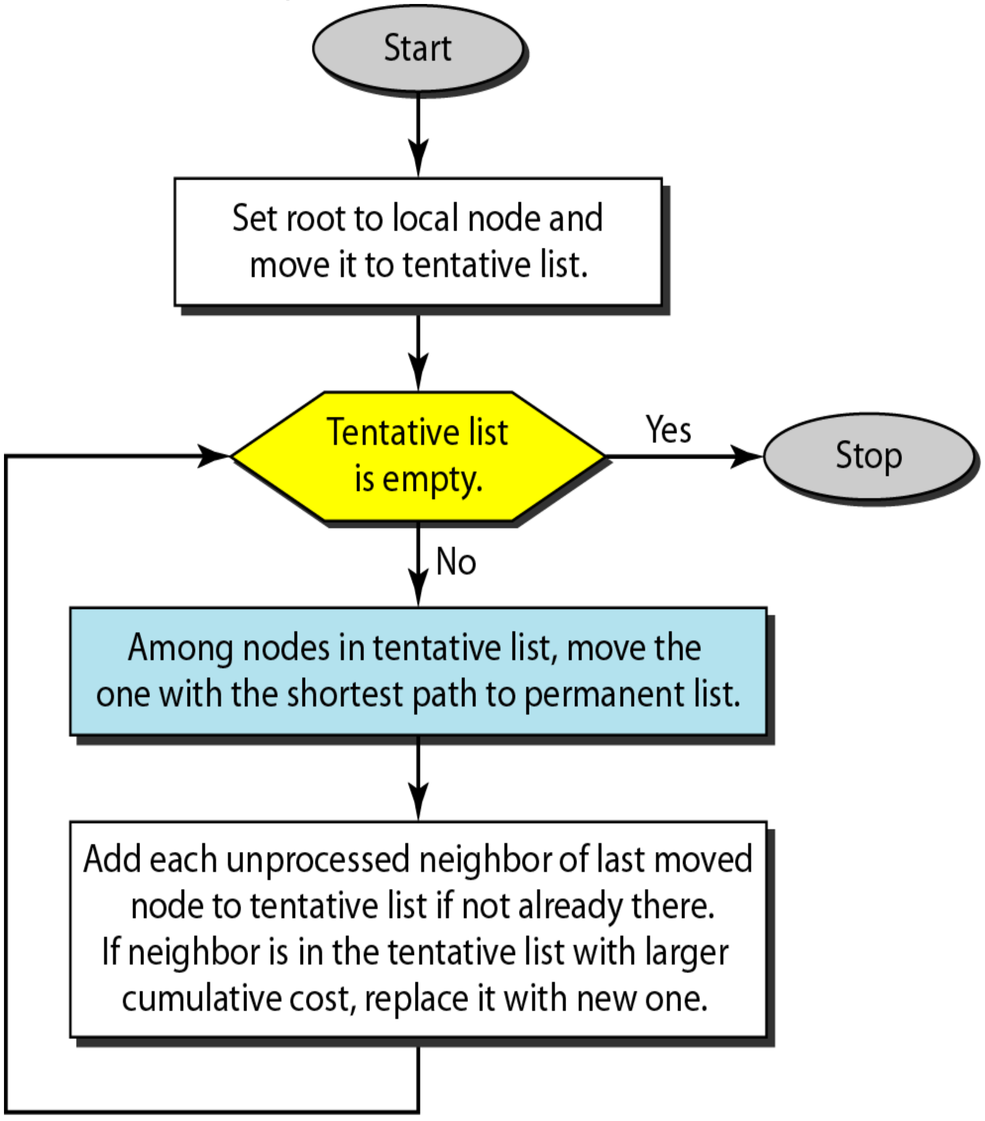
- After a node has prepared LSP, it must be disseminated to all the nodes, not only to its neighbours. This process is called flooding.

1. The creating node sends a copy of LSPs to its neighbour.
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, it does the following:
   a. It discards the old LSP and keeps the new one.
   b. It sends the copy of new LSP to its neighbour.

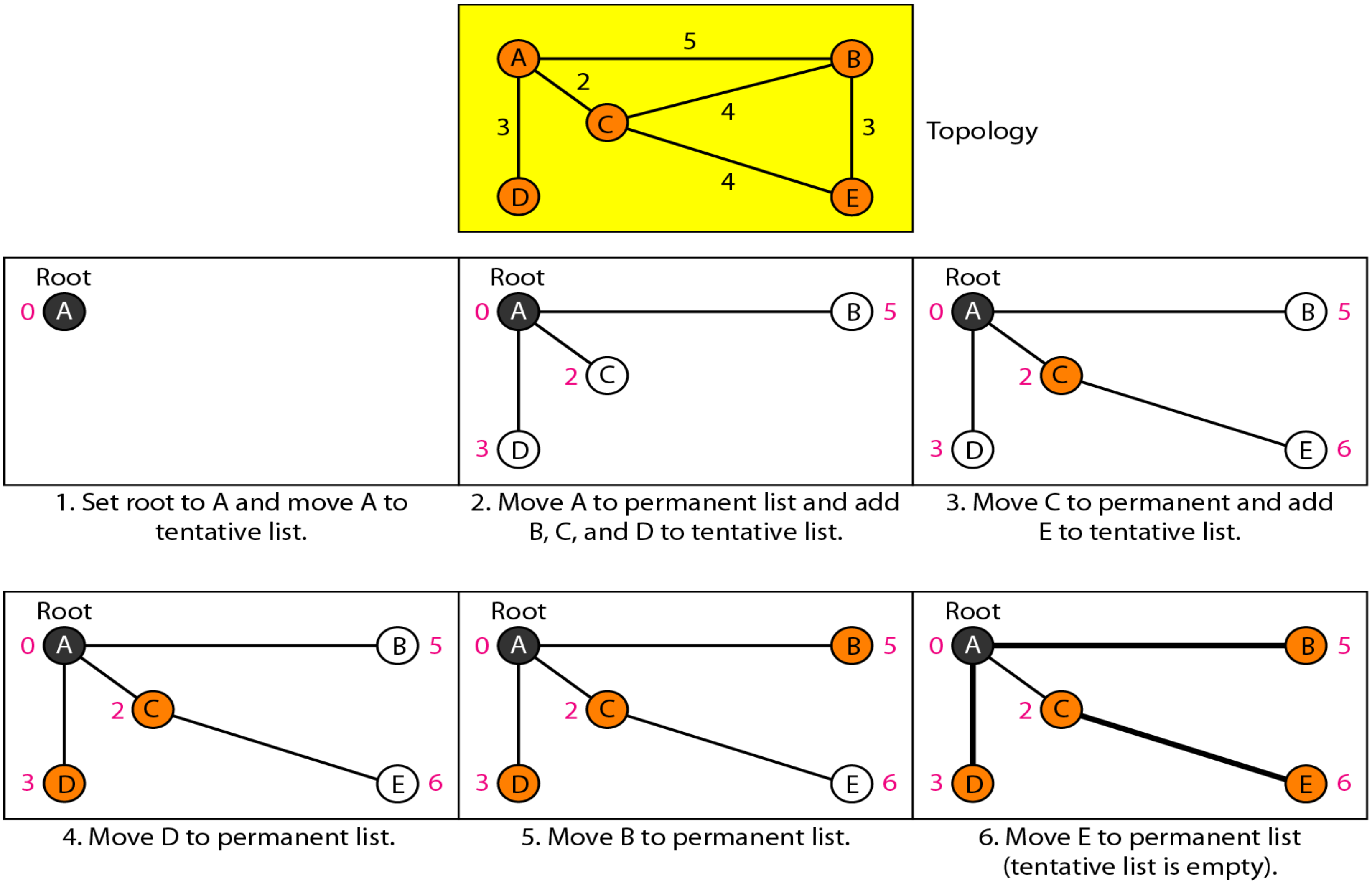
# 3. Formation of Shortest Path Tree – Dijikstra's Algorithm

- After receiving all LSPs, each node will have a copy of whole topology. However, this is not sufficient to find a shortest path to every other node; a shortest path tree is needed.
- A tree is a graph of nodes and links. One node is called root. All other nodes can be reached from the root through only one single route.
- A shortest path tree is a tree in which the path between the root and every other node is the shortest.
- NEED: For each node is a shortest path tree with that node as a root.
- The Dijikstra's algorithm finds a shortest path tree from a graph.
  - The algorithm divides the nodes into two sets as tentative and permanent.
  - It finds a neighbour of a current node, makes them tentative, examines them, if they pass the criteria, makes them permanent .

# By Flowchart



# *Formation of shortest path tree*



1. We make node A the root of the tree and move it to tentative list. Our two lists are:

   Permanent list: empty    Tentative list: A(0)

2. Node A has the shortest cumulative cost from all nodes in the tentative list. We move A to permanent list and add all neighbours of A to tentative list. Our new lists are:

   Permanent list: A(0)    Tentative list: B(5), C(2), D(3)

3. Node C has the shortest cumulative cost from all nodes in the tentative list. We move C to the permanent list. Node C has three neighbours, but node A is already processed, which makes the unprocessed neighbours just B and E. However, B is already in the tentative list with the cumulative cost of 5. Node A could also reach node B through C with a cumulative cost of 6. Since 5 is less than 6, we keep node B with a cumulative cost of 5 in the tentative list and do not replace it. Our new lists are:

   Permanent list: A(0), C(2)    Tentative list: B(5), D(3), E(6)

4. Node D has the shortest cumulative cost of all the nodes in the tentative list. We move D to the permanent list. Node D has no unprocessed neighbour to be added to the tentative list. Our new list are:

   Permanent list: A(0),C(2), D(3)    Tentative list: B(5), E(6)

5. Node B has the shortest cumulative cost of all the nodes in the tentative list. We move B to the permanent list. We need to add all unprocessed neighbours of B to the tentative list( Just node E). However, E(6) is already in the list with a smaller cumulative cost. The cumulative cost to node Em as the neighbor of B is 8. We keep node E(6) in the tentative list. Our new lists are:

   Permanent list: A(0),B(5),C(2), D(3)    Tentative list: E(6)

6. Node E has the shortest cumulative cost from all nodes in the tentative list. We move E to the permanent list. Node E has no neighbour. Now the tentative list is empty. We Stop; Shortest path is ready. The final lists are:

   Permanent list: A(0),B(5),C(2), D(3),E(6)    Tentative list: empty

# 4. Calculation of Routing table from Shortest Path tree

- Each node uses the shortest path tree protocol to construct its routing table.

***Routing table for node A***

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