SNS COLLEGE OF ENGINEERING

Kurumbapalayam(Po), Coimbatore – 641 107
Accredited by NAAC-UGC with 'A' Grade
Approved by AICTE, Recognized by UGC & Affiliated to Anna University, Chennai

Department of Information Technology

Course Name – 19IT401 Computer Networks

II Year / IV Semester

Unit 3 – Network Layer

Topic 5 – Routing Algorithms

Routing Algorithms

In unicast routing, a packet is routed, hop by hop, from its source to its destination by the help of forwarding tables.

There are three types of unicast routing algorithms, They are
1. Distance-Vector Routing
2. Link State Routing
3. Path vector Routing

Distance Vector Routing

- Each node creates its own least-cost tree with the information it has about its immediate neighbors
- The incomplete trees are exchanged between immediate neighbors to make the trees more and more complete and to represent the whole internet
- The heart of distance-vector routing is the famous **Bellman-Ford equation.**
- This equation is used to find the least cost (shortest distance) between a source node, x, and a destination node, y, through some intermediary nodes

Distance Vector Routing

Working Principle

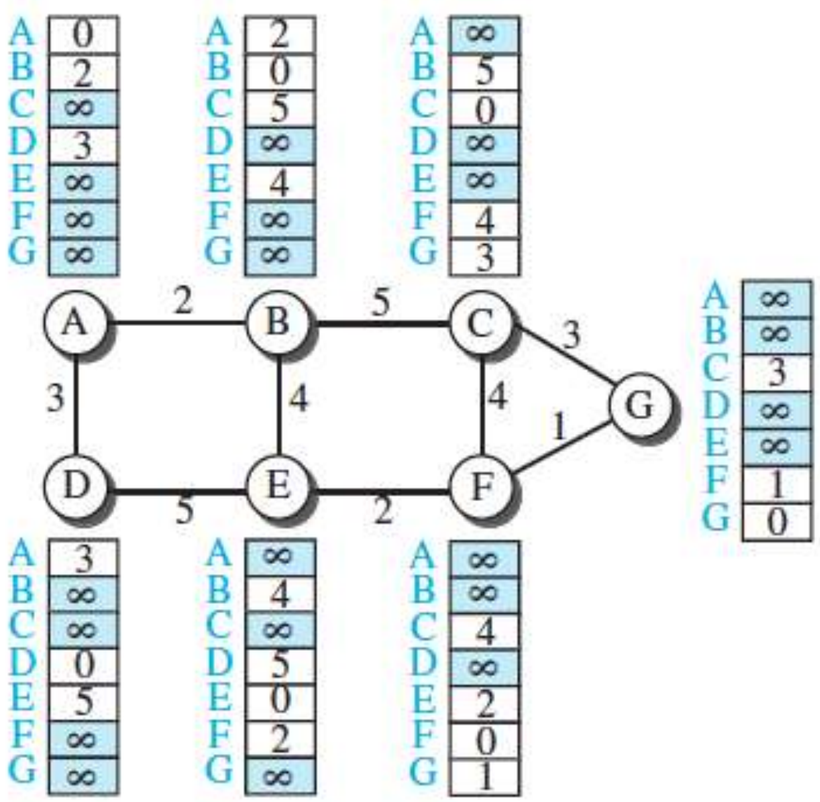
- Each node in an internet, when it is booted, creates a very rudimentary distance vector with the minimum information the node can obtain from its neighborhood.

- The node sends some greeting messages out of its interfaces and discovers the identity of the immediate neighbors and the distance between itself and each neighbor.

- It then makes a simple distance vector by inserting the discovered distances in the corresponding cells and leaves the value of other cells as infinity.

- After each node has created its vector, it sends a copy of the vector to all its immediate neighbors. After a node receives a distance vector from a neighbor, it updates its distance vector using the Bellman-Ford equation.

Distance Vector Routing



New B		Old B		A
A	2	A	2	A 0
B	0	B	0	B 2
C	5	C	5	C ∞
D	5	D	∞	D 3
E	4	E	4	E ∞
F	∞	F	∞	F ∞
G	∞	G	∞	G ∞

B[ ] = min (B[ ] , 2 + A[ ])

a. First event: B receives a copy of A's vector.

New B		Old B		E
A	2	A	2	A ∞
B	0	B	0	B 4
C	5	C	5	C ∞
D	5	D	5	D 5
E	4	E	4	E 0
F	6	F	∞	F 2
G	∞	G	∞	G ∞

B[ ] = min (B[ ] , 4 + E[ ])

b. Second event: B receives a copy of E's vector.

Distance Vector Routing

When to send routing update?
There are two different circumstances under which a given node decides to send a routing update to its neighbors.
1. Periodic Update
2. Triggered Update

Periodic Update - In this case, each node automatically sends an update message to its neighbors every so often, even if nothing has changed.

Triggered Update – Triggered update happens whenever a node's routing table is affected due to changes in the network (such as link failure). This forces the node to update its neighbors, neighbors update their neighbors and so on.

Distance Vector Routing

Disadvantages of Distance Vector

Count to Infinity

- For a routing protocol to work properly, if a link is broken (cost becomes infinity), every other router should be aware of it immediately, but in distance-vector routing, this takes some time.

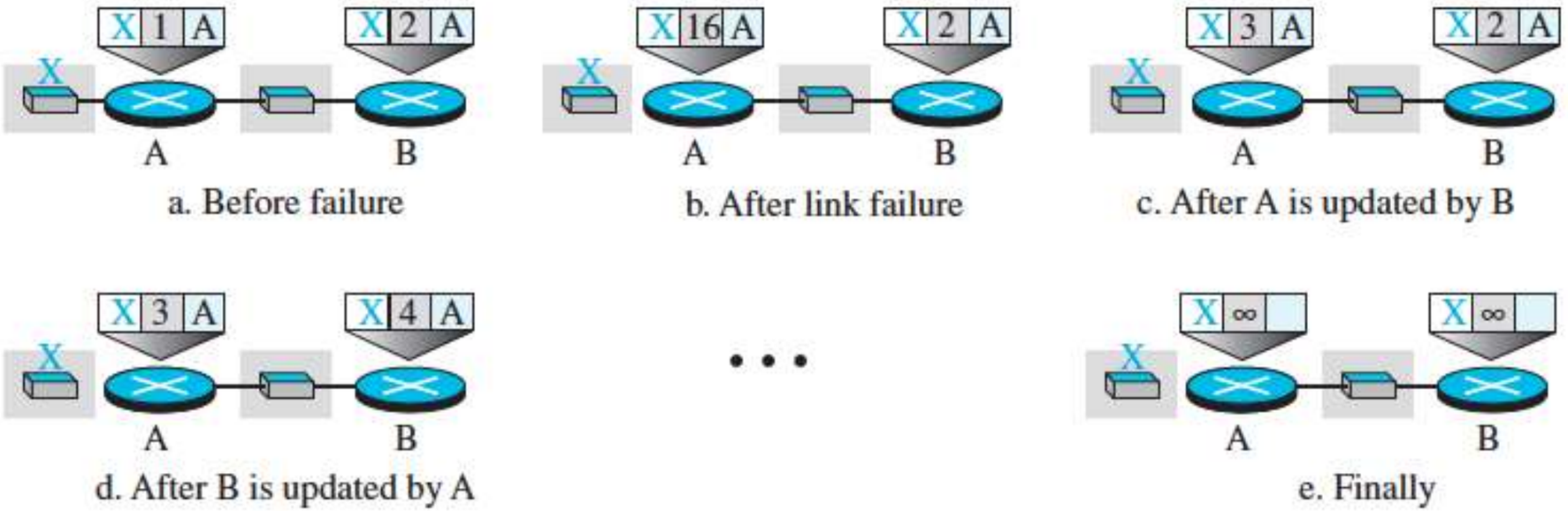
- The problem is referred to as count to infinity. It sometimes takes several updates before the cost for a broken link is recorded as infinity by all routers.

- In this case, the routing tables for the network do not stabilize.

- Infinity is redefined to a small number. Most implementations define 16 as infinity.

Distance Vector Routing

Two-Node Loop (Example for count to infinity problem)



- Node A thinks that the route to X is via B; node B thinks that the route to X is via A.
- If A receives a packet destined for X, the packet goes to B and then comes back to A.
- Similarly, if B receives a packet destined for X, it goes to A and comes back to B.
- Packets bounce between A and B, creating a two-node loop problem.

Distance Vector Routing

Split Horizon

- One solution to instability is called split horizon. In this strategy, instead of flooding the table through each interface, each node sends only part of its table through each interface.

- When a node sends a routing update to its neighbors, it does not send those routes it learned from each neighbor back to that neighbor.

Link-State Routing

- In this algorithm the cost associated with an edge defines the state of the link.

- Links with lower costs are preferred to links with higher costs; if the cost of a link is infinity, it means that the link does not exist or has been broken.

Link-State Database (LSDB)

- To create a least-cost tree with this method, each node needs to have a complete map of the network, which means it needs to know the state of each link.
- The collection of states for all links is called the link-state database (LSDB).

Flooding

Flooding is the process of ensuring all nodes having a copy of the link-state information from all the other nodes. Each node creates an update packet called a **link-state packet (LSP)** that contains

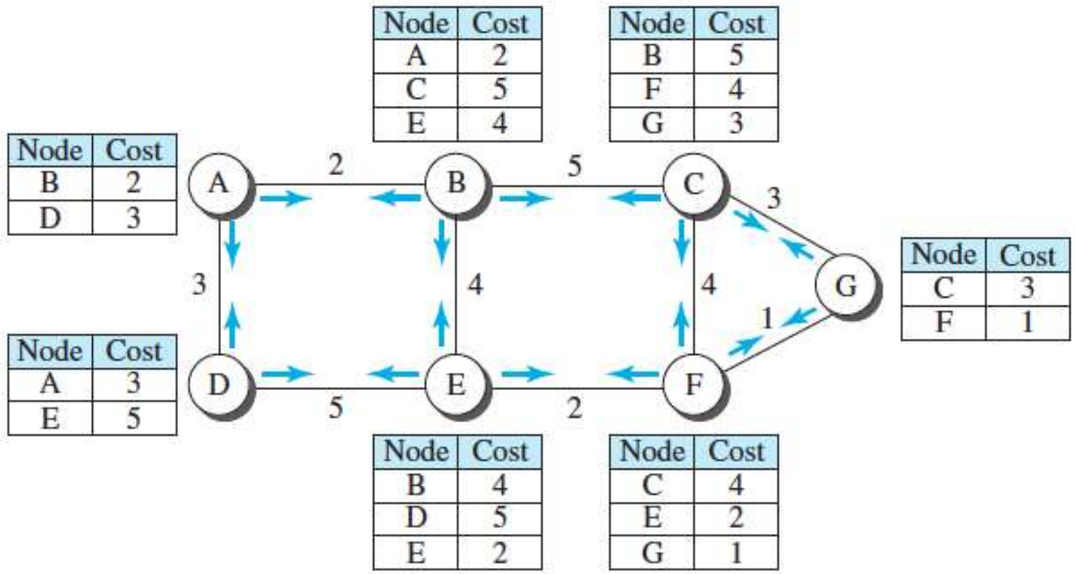
- ID of the node
- List of directly connected neighbors of that node and the cost to each one
- Sequence number
- Time to live for this packet

Link-State Routing

Each node that receives a LSP checks to see whether it has a copy.

- If not, it stores and forwards the LSP on all of its links
- If it has a copy, it compares the sequence numbers.
  - If the received LSP has a bigger sequence number, then it is stored and forwarded, since it is a recent LSP. The old one discarded.

LSP is generated either periodically or when there is a change in the topology detected using hello packets.



Link-State Routing

Formation of Least-Cost Trees

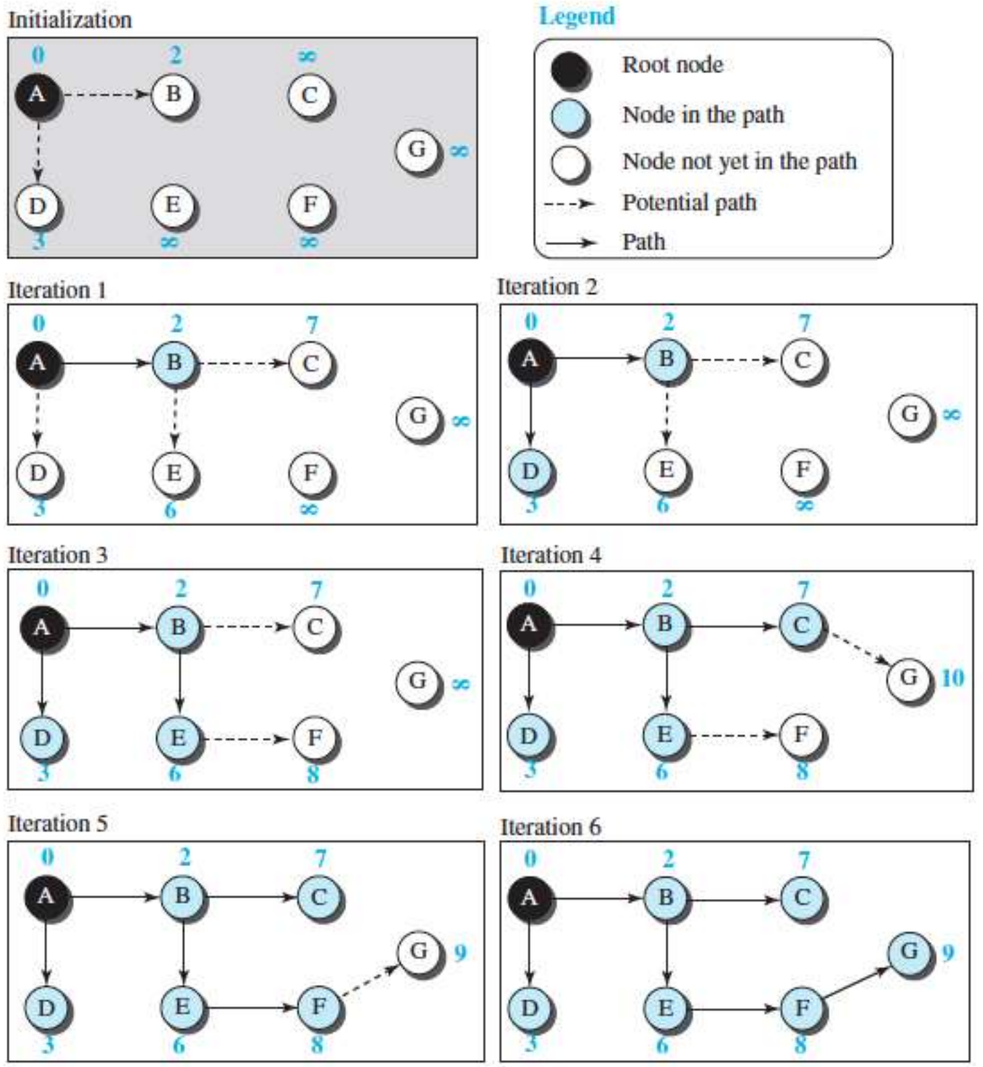
To create a least-cost tree for itself, using the shared LSDB, each node needs to run the famous **Dijkstra Algorithm**. This iterative algorithm uses the following steps:

1. The node chooses itself as the root of the tree, creating a tree with a single node, and sets the total cost of each node based on the information in the LSDB.

2. The node selects one node, among all nodes not in the tree, which is closest to the root, and adds this to the tree. After this node is added to the tree, the cost of all other nodes not in the tree needs to be updated because the paths may have been changed.

3. The node repeats step 2 until all nodes are added to the tree.

Link-State Routing



Path-Vector Routing

Path-Vector Routing

- Both link-state and distance-vector routing are based on the least-cost goal.
- Least-cost routing does not prevent a packet from passing through an area when that area is in the least-cost path.
- In other words, the least-cost goal, applied by LS or DV routing, does not allow a sender to apply specific policies to the route a packet may take.
- Path-vector routing does not have the drawbacks of LS or DV routing as described above because it is not based on least-cost routing.
- The best route is determined by the source using the policy it imposes on the route.
- In other words, the source can control the path

Path-Vector Routing

Spanning Trees

In path-vector routing, the path from a source to all destinations is also determined by the best spanning tree.

The best spanning tree, however, is not the least-cost tree; it is the tree determined by the source when it imposes its own policy.

If there is more than one route to a destination, the source can choose the route that meets its policy best.

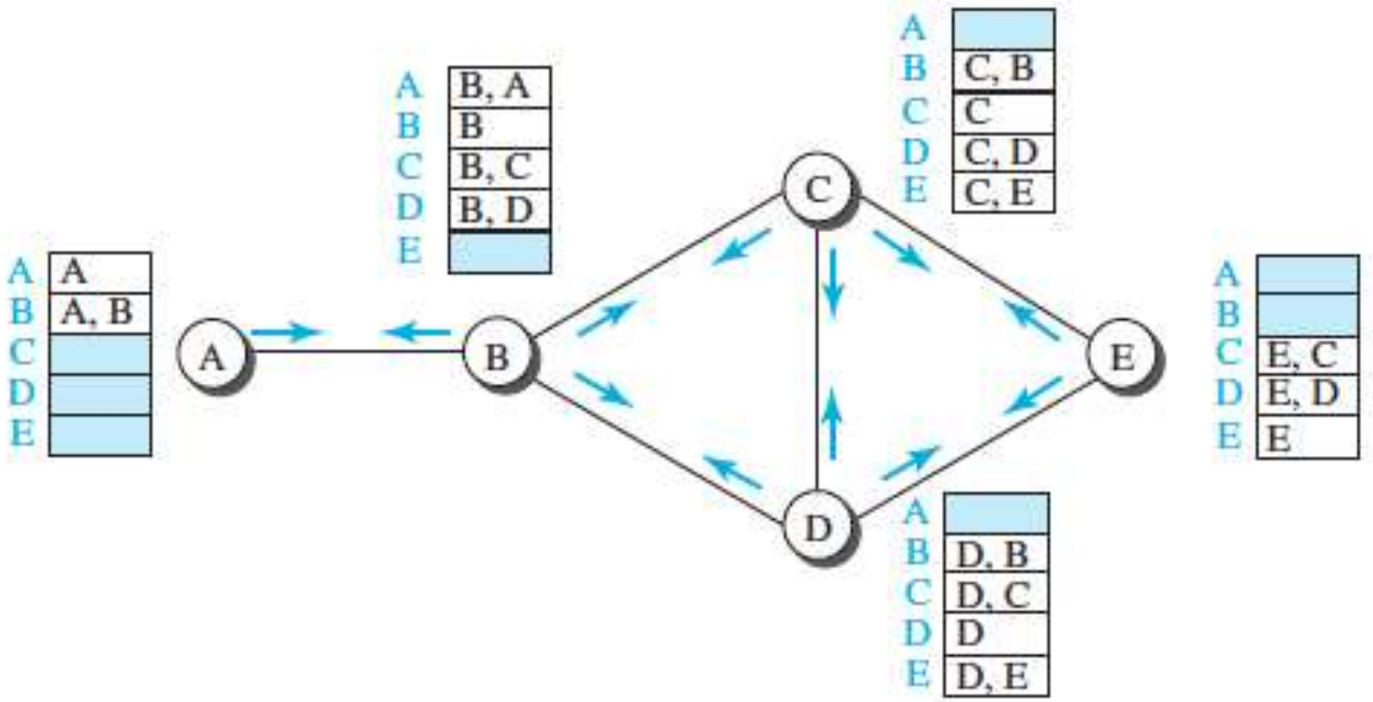
Creation of Spanning Trees

When a node is booted, it creates a path vector based on the information it can obtain about its immediate neighbor.

A node sends greeting messages to its immediate neighbors to collect these pieces of information.

Path-Vector Routing

- Each node, after the creation of the initial path vector, sends it to all its immediate neighbors.
- Each node, when it receives a path vector from a neighbor, updates its path vector using an equation similar to the Bellman-Ford, but applying its own policy instead of looking for the least cost.



THANK YOU