4.2 Routing Algorithms

4.2 Routing algorithms :

- **4.2 Routing algorithms :** Shortest Path (Dijkstra's), Link state routing, Distance Vector Routing

Router

- A router is a physical or virtual appliance that passes information between two or more packet-switched computer networks.
- A router inspects a given data packet's destination Internet Protocol address (IP address), calculates the best way for it to reach its destination and then forwards it accordingly.
- A router is a common type of gateway. It is positioned where two or more networks meet at each point of presence on the internet.

Router

- Physical(Traditional) routers are stand-alone devices that use proprietary software.
- Virtual router is a software instance that performs the same functions as a physical router.
- Virtual routers typically run on commodity servers, either alone or packaged with other virtual network functions,
  - like firewall packet filtering,
  - load balancing and
  - wide area network (WAN) optimization capabilities.

Router

- A router examines a packet header's destination IP address and compares it against a routing table to determine the packet's best next hop.
- Routing tables list directions for forwarding data to particular network destinations, sometimes in the context of other variables, like cost.
- They amount to an **algorithmic set of rules that calculate the best way to transmit traffic toward** any given IP address.

Routing Table

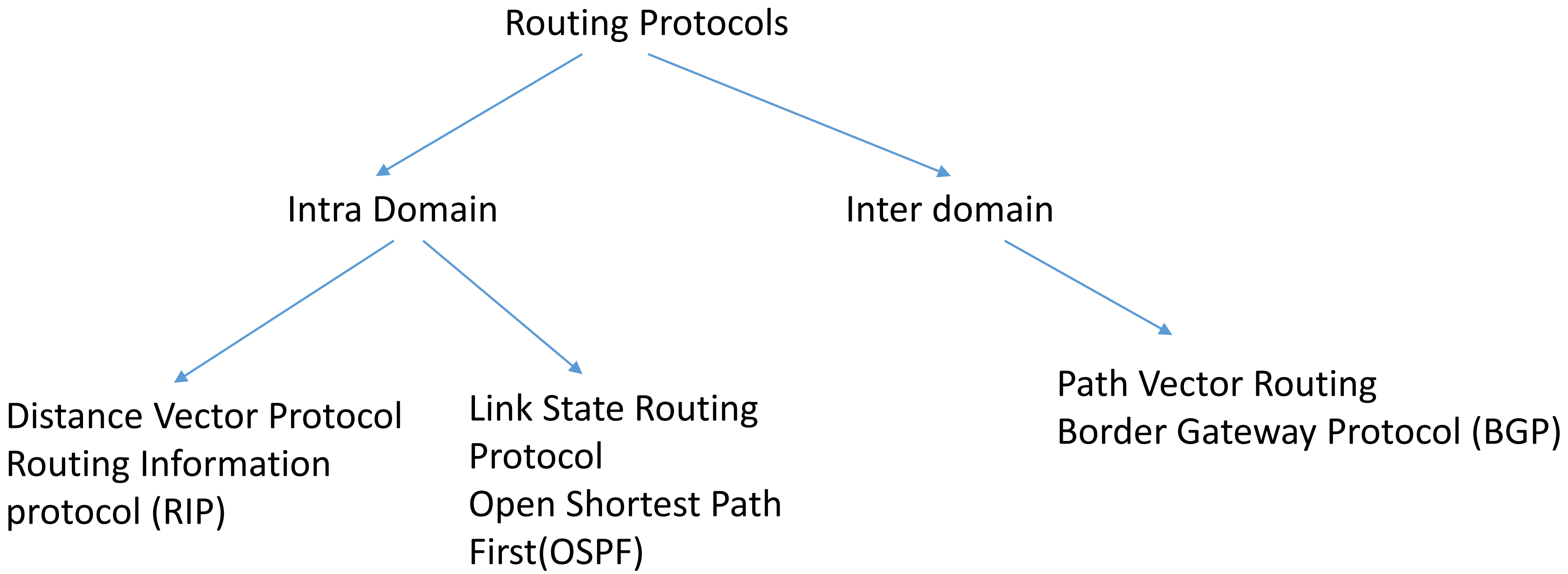
- A routing table often specifies a default route, which the router uses whenever it fails to find a better forwarding option for a given packet. For example, the typical home office router directs all outbound traffic along a single default route to its internet service provider (ISP).
- Routing tables can be static -- i.e., manually configured -- or dynamic.
- Dynamic routers automatically update their routing tables based on network activity, exchanging information with other devices via routing protocols.

Routing Table

Table 11.4 *Routing Table for Node A*

Destination	*Cost*	*Next Router*
A	0	—
B	2	—
C	7	B
D	3	—
E	6	B
F	8	B
G	9	B

Routing protocols



Routing Protocols

- **Routing Information Protocol** (RIP) - the original protocol for **defining how routers should share information when moving traffic among an interconnected group of local area networks**. The largest number of hops allowed for RIP is 15, which limits the size of networks that RIP can support.
- **Open Shortest Path First** (OSPF) - **used to find the best path for packets as they pass through a set of connected networks**. OSPF is designated by the Internet Engineering Task Force (IETF) as one of several Interior Gateway Protocols (IGPs)
- **Border Gateway Protocol** (BGP) - manages how packets are routed across the internet through the exchange of information between edge routers.

Routing Protocols

- **Interior Gateway Routing Protocol** (IGRP)- determines how routing information between gateways will be exchanged within an autonomous network. The routing information can then be used by other network protocols to specify how transmissions should be routed.
- **Enhanced Interior Gateway Routing Protocol** (EIGRP) - evolved from IGRP. If a router can't find a route to a destination in one of these tables, it queries its neighbours for a route and they in turn query their neighbours until a route is found. When a routing table entry changes in one of the routers, it notifies its neighbours of the change instead of sending the entire table.
- **Exterior Gateway Protocol** (EGP) - determines how routing information between two neighbour gateway hosts, each with its own router, is exchanged. EGP is commonly used between hosts on the Internet to exchange routing table information.

Distance Vector Routing

- Distance Vector Routing (DVR) Protocol
- A **distance-vector routing (DVR)** protocol requires that a router inform its neighbours of topology changes periodically. Historically known as the old ARPANET routing algorithm (or known as Bellman-Ford algorithm).
- **Bellman Ford Basics –** Each router maintains a Distance Vector table containing the distance between itself and ALL possible destination nodes. Distances based on a chosen metric, are computed using information from the neighbour's distance vectors.

Distance Vector Routing

<u>**Step-01:**</u> Each router prepares its routing table. By their local knowledge. each router knows about-

- All the routers present in the network
- Distance to its neighboring routers

<u>**Step-02:**</u>

- Each router exchanges its distance vector with its neighboring routers.
- Each router prepares a new routing table using the distance vectors it has obtained from its neighbors.
- This step is repeated for (n-2) times if there are n routers in the network.
- After this, routing tables converge / become stable.

Distance Vector Algorithm –

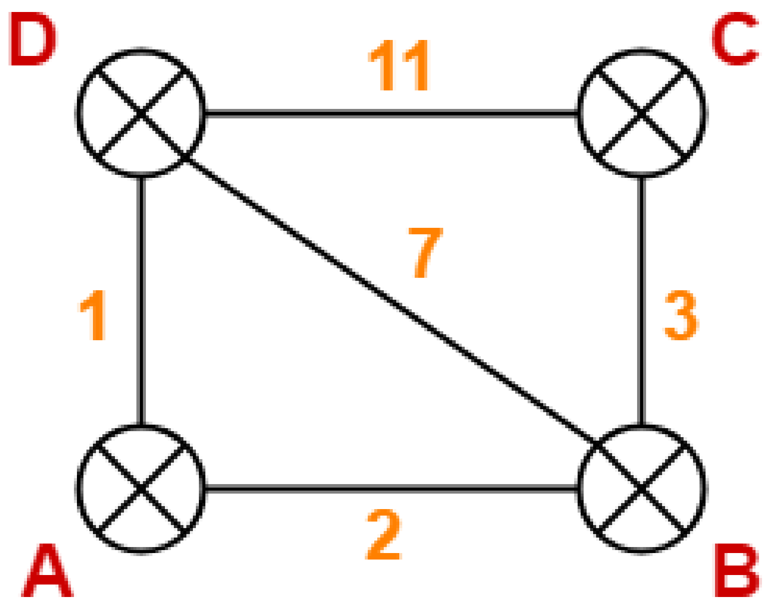
Note: The DV calculation is based on minimizing the cost to each destination

- From time-to-time, each node sends its own distance vector estimate to neighbours.
- When a node x receives new DV estimate from any neighbour v, it saves v’s distance vector and it updates its own DV using B-F equation:

Distance Vector Routing Example-

Consider-

- There is a network consisting of 4 routers.
- The weights are mentioned on the edges.
- Weights could be distances or costs or delays.



Step-01:

Each router prepares its routing table using its local knowledge. Routing table prepared by each router is shown below-

At Router A-

Destination	Distance	Next Hop
A	0	A
B	2	B
C	∞	–
D	1	D

At Router B-

Destination	Distance	Next Hop
A	2	A
B	0	B
C	3	C
D	7	D

At Router C-

Destination	Distance	Next Hop
A	∞	–
B	3	B
C	0	C
D	11	D

At Router D-

Destination	Distance	Next Hop
A	1	A
B	7	B
C	11	C
D	0	D

Step-02: Each router exchanges its distance vector obtained in Step-01 with its neighbours. After exchanging the distance vectors, each router prepares a new routing table.

At Router A- Router A receives distance vectors from its neighbours B and D. Router A prepares a new routing table as-

- Cost of reaching destination B from router A = min { 2+0 , 1+7 } = 2 via B.
- Cost of reaching destination C from router A = min { 2+3 , 1+11 } = 5 via B.
- Cost of reaching destination D from router A = min { 2+7 , 1+0 } = 1 via D.

From B

2
0
3
7

Cost(A→B) = 2

From D

1
7
11
0

Cost(A→D) = 1

Destination	Distance	Next hop
A	0	A
B		
C		
D		

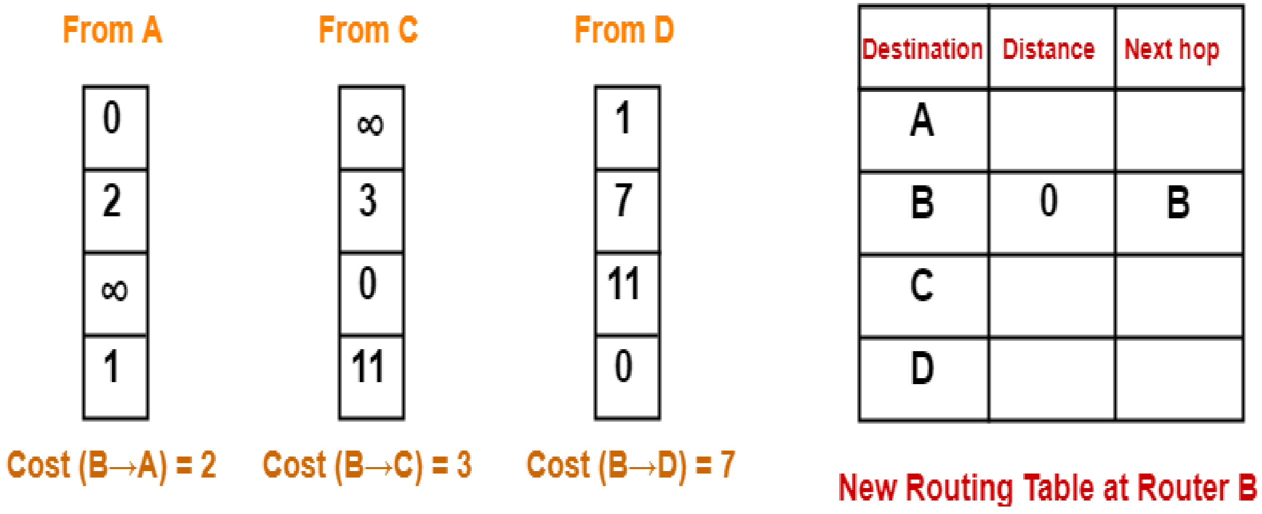
New Routing Table at Router A

Destination	Distance	Next Hop
A	0	A
B	2	B
C	5	B
D	1	D

At Router B-

Router B receives distance vectors from its neighbors A, C and D.
Router B prepares a new routing table as-

- Cost of reaching destination A from router B = min { 2+0 , 3+∞ , 7+1 } = 2 via A.
- Cost of reaching destination C from router B = min { 2+∞ , 3+0 , 7+11 } = 3 via C.
- Cost of reaching destination D from router B = min { 2+1 , 3+11 , 7+0 } = 3 via A.



Destination	Distance	Next Hop
A	2	A
B	0	B
C	3	C
D	3	A

At Router C-

Router C receives distance vectors from its neighbors B and D.
Router C prepares a new routing table as-

Cost of reaching destination A from router C = min { 3+2 , 11+1 } = 5 via B.
Cost of reaching destination B from router C = min { 3+0 , 11+7 } = 3 via B.
Cost of reaching destination D from router C = min { 3+7 , 11+0 } = 10 via B.

From B

2
0
3
7

Cost (C→B) = 3

From D

1
7
11
0

Cost (C→D) = 11

Destination	Distance	Next hop
A		
B		
C	0	C
D		

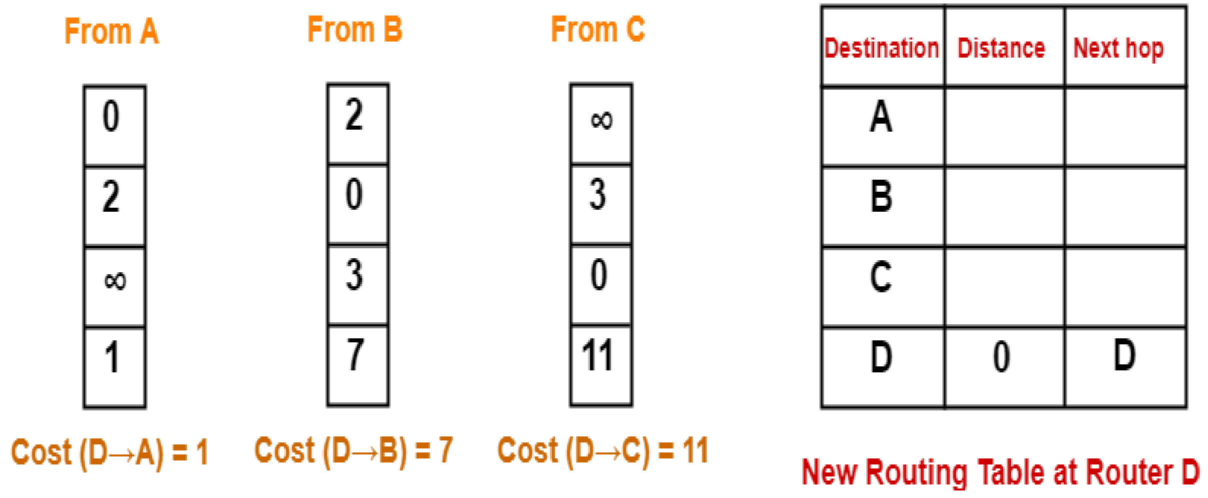
New Routing Table at Router C

Destination	Distance	Next Hop
A	5	B
B	3	B
C	0	C
D	10	B

At Router D-

Router D receives distance vectors from its neighbors A, B and C.
Router D prepares a new routing table as-

Cost of reaching destination A from router D = min { 1+0 , 7+2 , 11+∞ } = 1 via A.
Cost of reaching destination B from router D = min { 1+2 , 7+0 , 11+3 } = 3 via A.
Cost of reaching destination C from router D = min { 1+∞ , 7+3 , 11+0 } = 10 via B.



Destination	Distance	Next Hop
A	1	A
B	3	A
C	10	B
D	0	D

Step-03:

- Each router exchanges its distance vector obtained in Step-02 with its neighboring routers.

- After exchanging the distance vectors, each router prepares a new routing table.

At Router A-

Destination	Distance	Next Hop
A	0	A
B	2	B
C	5	B
D	1	D

At Router B-

Destination	Distance	Next Hop
A	2	A
B	0	B
C	3	C
D	3	A

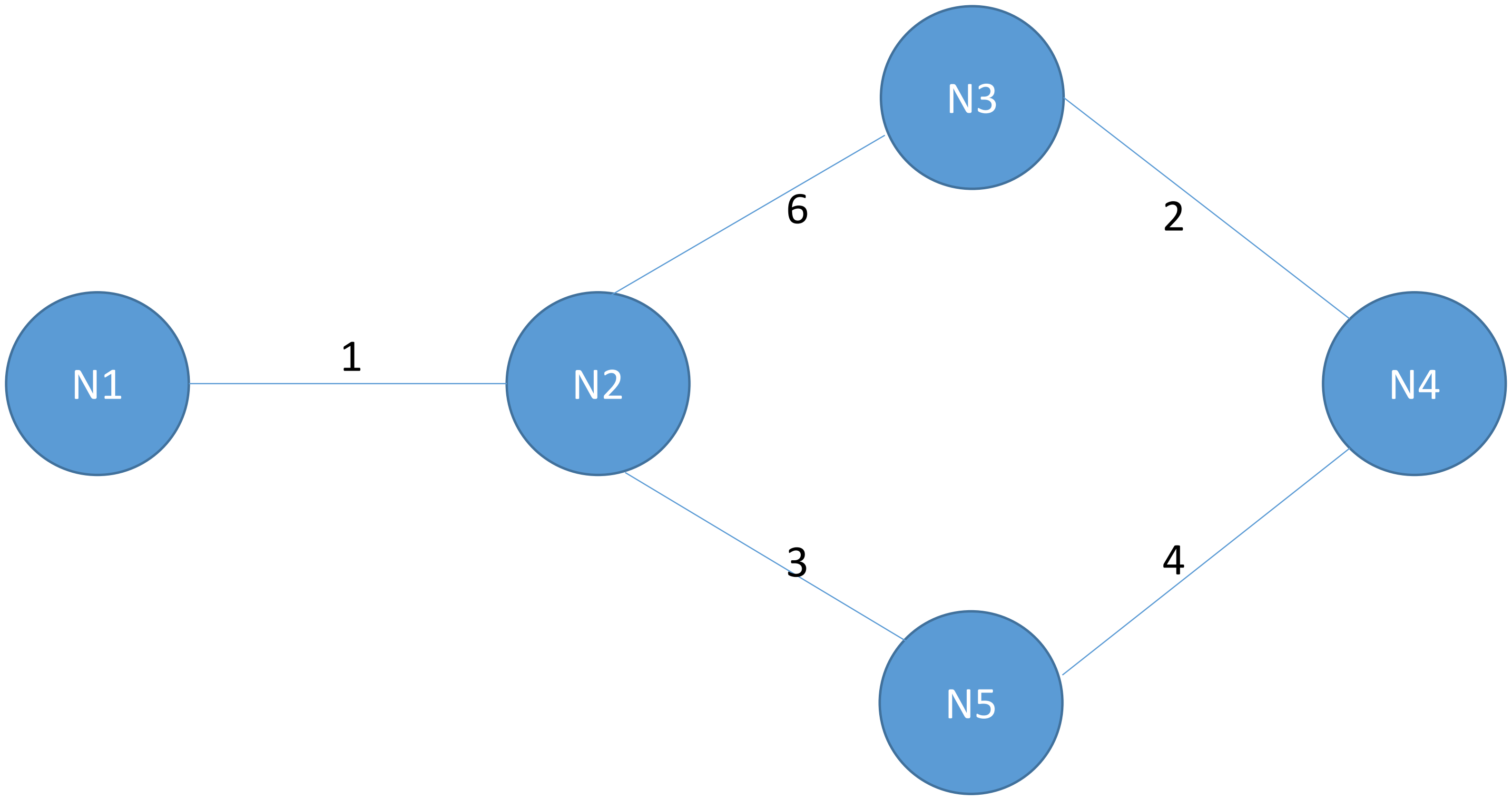
At Router C-

Destination	Distance	Next Hop
A	5	B
B	3	B
C	0	C
D	6	B

At Router D-

Destination	Distance	Next Hop
A	1	A
B	3	A
C	6	A
D	0	D

Distance Vector Protocol



N1

Dest.	Dist	Next
N1	0	N1
N2	1	N2
N3	∞	--
N4	∞	--
N5	∞	--

N2

Dest.	Dist	Next
N1	1	N1
N2	0	N2
N3	6	N3
N4	∞	--
N5	3	N5

N3

Dest.	Dist	Next
N1	∞	--
N2	6	N2
N3	0	N3
N4	4	N4
N5	∞	--

N4

Dest.	Dist	Next
N1	∞	--
N2	∞	--
N3	2	N3
N4	0	N4
N5	6	N5

N5

Dest.	Dist	Next
N1	∞	--
N2	3	N2
N3	∞	--
N4	4	N4
N5	0	N5

First Pass

- Routing table is shared with only Neighbours
- Only Distance Vector is saved.
- N1 shares Distance vector with N2
- N2 shares Distance vector with N1, N3 and N5
- N3 shares Distance vector with N2, N4
- N4 shares Distance vector with N3 and N5
- N5 shares Distance vector with N2, N4

N1

Dest.	Dist	Next
N1	0	N1
N2	1	N2
N3	7	N2,N3
N4	∞	--
N5	4	-2,N5

N2

Dest.	Dist	Next
N1	1	N1
N2	0	N2
N3	6	N3
N4	7	N5,N4
N5	3	N5

N3

Dest.	Dist	Next
N1	7	N2, N1
N2	6	N2
N3	0	N3
N4	4	N4
N5	7	N4, N5

N4

Dest.	Dist	Next
N1	∞	--
N2	7	N5, N2
N3	2	N3
N4	0	N4
N5	6	N5

N5

Dest.	Dist	Next
N1	4	N2,N1
N2	3	N2
N3	6	N4,N3
N4	4	N4
N5	0	N5

Second Pass

- Routing table is shared with only Neighbours
- Only Distance Vector is saved.
- N1 shares Distance vector with N2
- N2 shares Distance vector with N1, N3 and N5
- N3 shares Distance vector with N2, N4
- N4 shares Distance vector with N3 and N5
- N5 shares Distance vector with N2, N4

N1

Dest.	Dist	Next
N1	0	N1
N2	1	N2
N3	7	N2,N3
N4	8	N2,N5,N4
N5	4	-2,N5

N2

Dest.	Dist	Next
N1	1	N1
N2	0	N2
N3	6	N3
N4	7	N5,N4
N5	3	N5

N3

Dest.	Dist	Next
N1	7	N2, N1
N2	6	N2
N3	0	N3
N4	4	N4
N5	7	N4, N5

N4

Dest.	Dist	Next
N1	8	N5,N2, N1
N2	7	N5, N2
N3	2	N3
N4	0	N4
N5	6	N5

N5

Dest.	Dist	Next
N1	4	N2,N1
N2	3	N2
N3	6	N4,N3
N4	4	N4
N5	0	N5

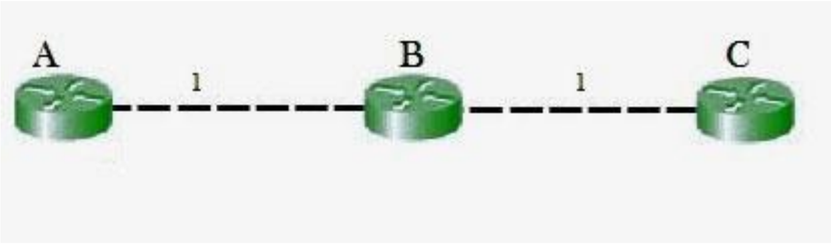
Third Pass

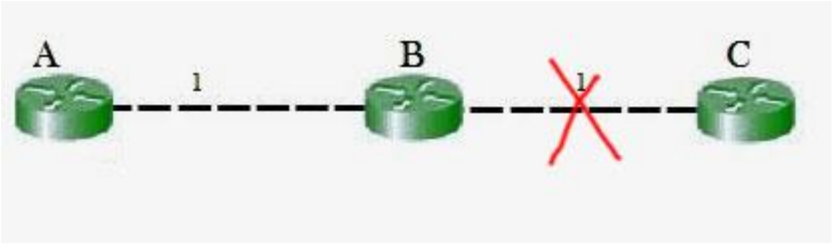
Distance Vector routing

- **Advantages of Distance Vector routing –**
  - It is **simpler to configure and maintain** than link state routing.
- **Disadvantages of Distance Vector routing –**
  - It is **slower to converge** than link state.
  - It is at risk from the **count-to-infinity problem**.
  - **It creates more traffic than link state** since a hop count change must be propagated to all routers and processed on each router. **Hop count updates take place on a periodic basis, even if there are no changes in the network topology**, so bandwidth-wasting broadcasts still occur.
  - For **larger networks, distance vector routing results in larger routing tables than link state** since each router must know about all other routers. This can also lead to congestion on WAN links.
- **Note – Distance Vector routing uses UDP(User datagram protocol) for transportation.**

Count to infinity problem in Routing

- Its The main issue with **D**istance Vector **R**outing (DVR) protocols is Routing Loops, since Bellman-Ford Algorithm cannot prevent loops. This routing loop in DVR network causes Count to Infinity Problem. **Routing loops usually occur when any interface goes down or two-routers send updates at the same time**.
- **Counting to infinity problem:**





Link State Routing

Link State Routing

- Also called shortest path first (SPF) forwarding
  - Named after Dijkstra's algorithm (1959) which it uses to compute routes
- **All routers have tables which contain a representation of the entire network topology.**
- **Each router creates a *link state packet* (LSP)** which contains names (e.g. network addresses) and cost to each of its neighbours
  - The **LSP is transmitted to *all* other routers (Flooding)**, who each update their own records
  - When a routers receives LSPs from all routers, it can use (collectively) that information to make topology-level decisions

Link State Packets

- LSP are essentially a list of tuples, containing:
  - The name of a neighbour to a router
    - Which may be a router or a network
  - The cost of the link to that neighbour
- LSPs are generated and distributed when:
  - A time period passes
  - New neighbours connect to the router
  - The link cost of a neighbour has changed
  - A link to a neighbour has failed (link failure)
  - A neighbour has failed (node failure)

Link State Packets

- Distribution of LSPs can be difficult
  - Routers themselves are the means for delivering messages
  - How do routers deliver their own messages, particularly when routers are in an inconsistent state
    - e.g. During link failure, before each router has been notified of the problem

Link State Packets

- One method for LSP distribution: Flooding
  - Each LSP received is transmitted to every direct neighbour (except the neighbour where the LSP came from)
  - This creates an exponential number of packets on the network (similar to $O(2^R)$, where R is the number of routers)
  - It does, however, guarantee that the LSP will be received by every router
    - Assuming that node or link failure does not occur, and LSPs are not somehow lost

Link State Packets

- An improvement on this scheme is as follows:
- When an LSP is received, it is compared with the stored copy
  - If it is identical to the stored copy, it is dropped
  - If it is different, the stored LSP is overwritten with the new LSP and the LSP is transmitted to every direct neighbour (except the source of the LSP)
- This scheme works because if a given router has already received a LSP from another neighbour, it will have also already distributed the LSP to all of its neighbours
- This scheme has a network complexity similar to $O(R^2)$

Link State Routing Algorithm

- Ok, now that we know how to distribute LSPs, how are they used to determine routes?
  - The algorithm used (mostly) was developed by Dijkstra
  - Essentially, the algorithm runs at each router, computing each possible path to the destination, adding up each cost
    - The path with the lowest cost is used

Link State Routing Algorithm

- The algorithm requires the following information:
  - **Link state database**: List of all the latest LSPs from each router on the network
  - **Path**: Tree structure storing previously computed best paths
    - Consider this a sort of cache
    - Data type for nodes: (ID, path cost, port)
  - **Tent**: Tree structure storing paths currently being tested and compared (tentative)
    - Consider this a sort of rough workspace
    - Data type for nodes: (ID, path cost, port)

Link State Routing Algorithm

- **Forwarding database**: Table storing all IDs that can be reached, and the port to which messages should be sent
  - This is simply a reduced version of the ‘Path’, which contains (destination , port) pairs
  - This can be used by the router to quickly forward packets for which the best path has already been determined
  - Data type for table rows: (ID, port)

Dijkstra Algorithm

- Remove the Loops
- Remove the parallel paths higher weights.
- Create Matrix with all vertices. And Set 0 to source and Infinity to other vertices.
- Mark the smallest valued vertex.
- Find all the vertices connected to lowest valued vertex and update the values of connected vertex.
- New Destination=min(old value, marked value + edge weight)

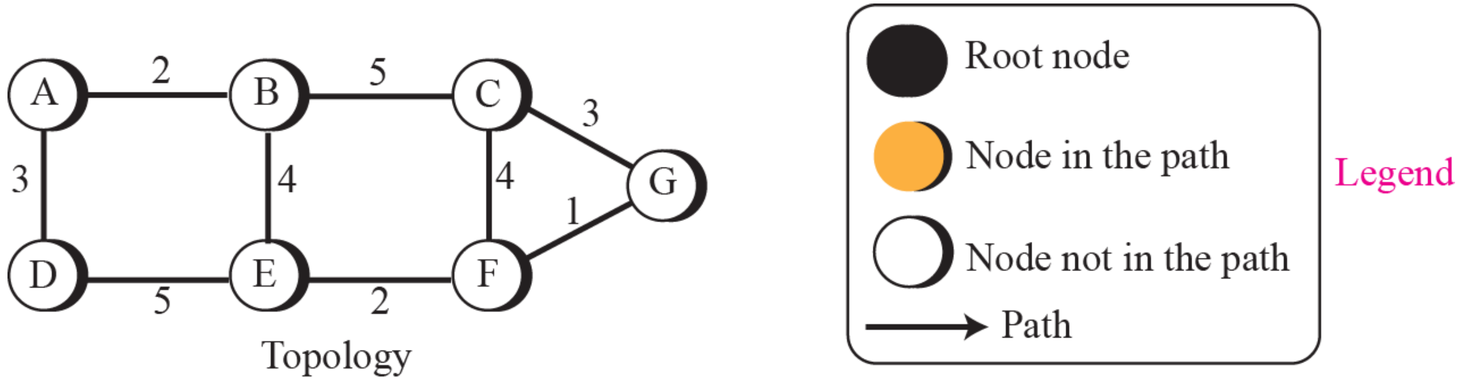
Table 11.3 *Dijkstra's Algorithm*

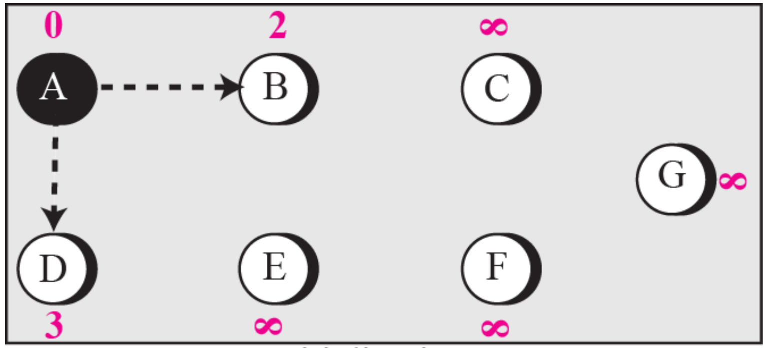
```
Dijkstra ( )
{
    // Initialization
    Path = {s}              // s means self
    for (i = 1 to N)
    {
        if(i is a neighbor of s and i ≠ s)    D_i = c_si
        if (i is not a neighbor of s)         D_i = ∞
    }
    D_s = 0

} // Dijkstra
```

Continued

```
// Iteration
Repeat
{
    // Finding the next node to be added
 Path = Path ∪ i   if D_i is minimum among all remaining nodes

    // Update the shortest distance for the rest
 for (j = 1 to M)     // M number of remaining nodes
 {
     D_j = minimum (D_j ,  D_j  + c_ij)
 }
} until (all nodes included in the path, M = 0)
```

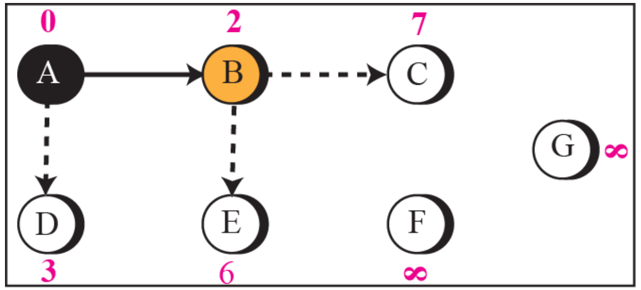
Figure 11.19 *Forming shortest path three for router A in a graph*



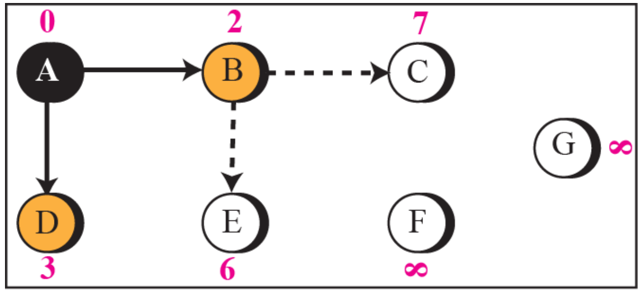


Initialization

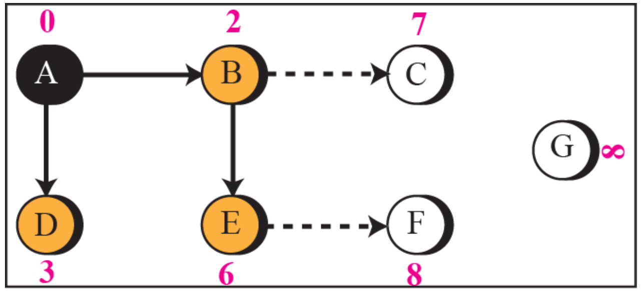
Figure 11.19 *Continued*



Iteration 1

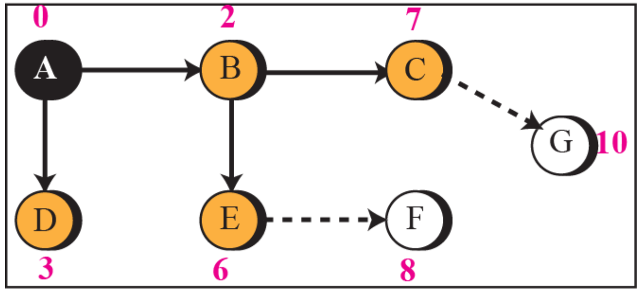


Iteration 2

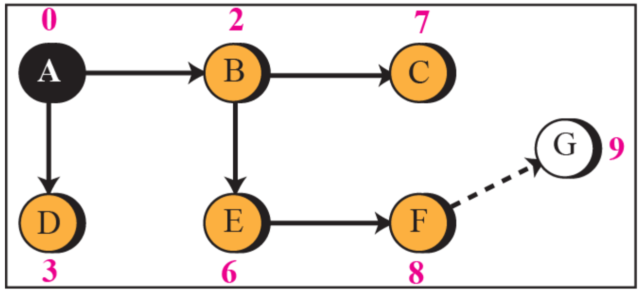


Iteration 3

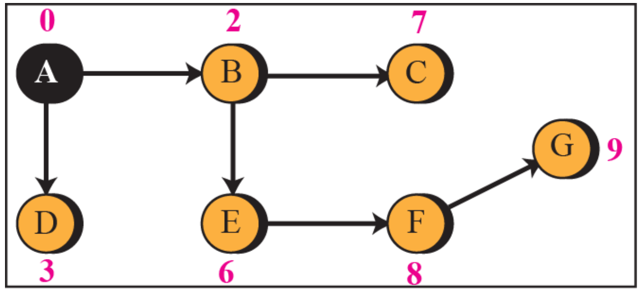
Figure 11.19 *Continued*



Iteration 4



Iteration 5



Iteration 6

Example 11.6

To show that the shortest path tree for each node is different, we found the shortest path tree as seen by node C (Figure 11.20). We leave the detail as an exercise.

Figure 11.20 *Example 11.6*

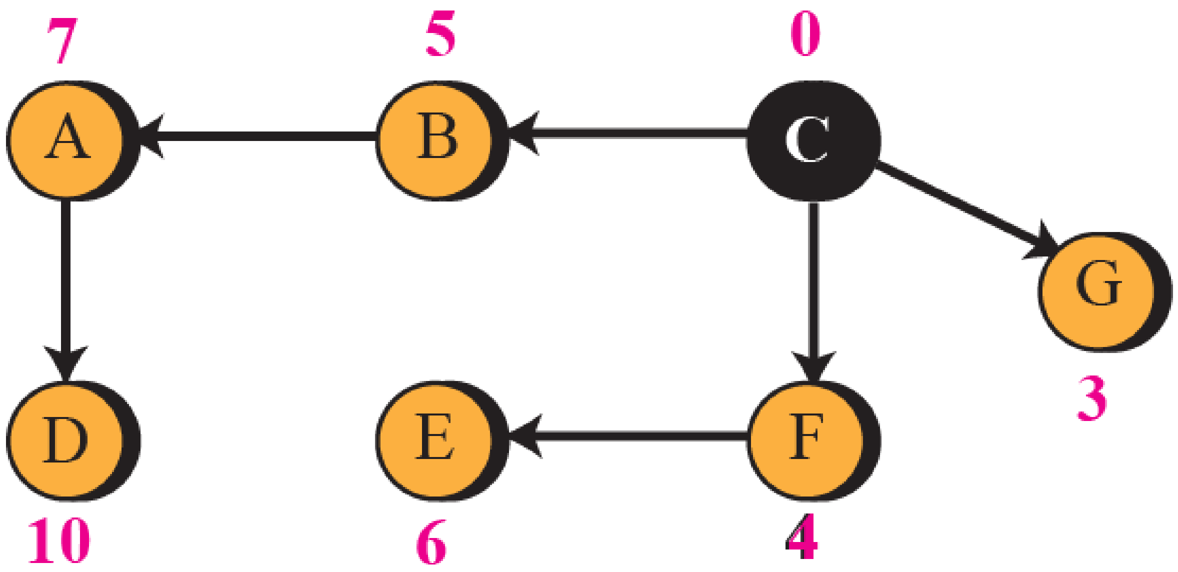


Table 11.4 *Routing Table for Node A*

Destination	*Cost*	*Next Router*
A	0	—
B	2	—
C	7	B
D	3	—
E	6	B
F	8	B
G	9	B

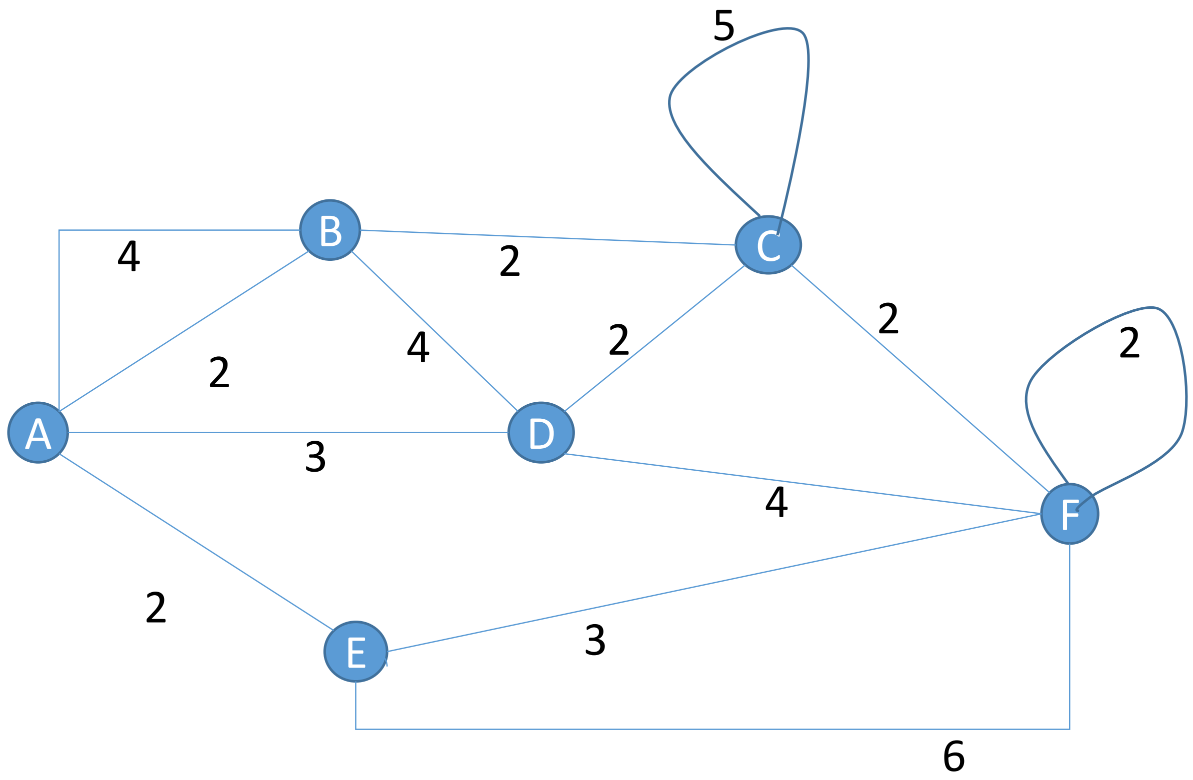
Table 11.4 *Routing Table for Node A*

Destination	*Cost*	*Next Router*
A	0	—
B	2	—
C	7	B
D	3	—
E	6	B
F	8	B
G	9	B

Dijkstra Algorithm

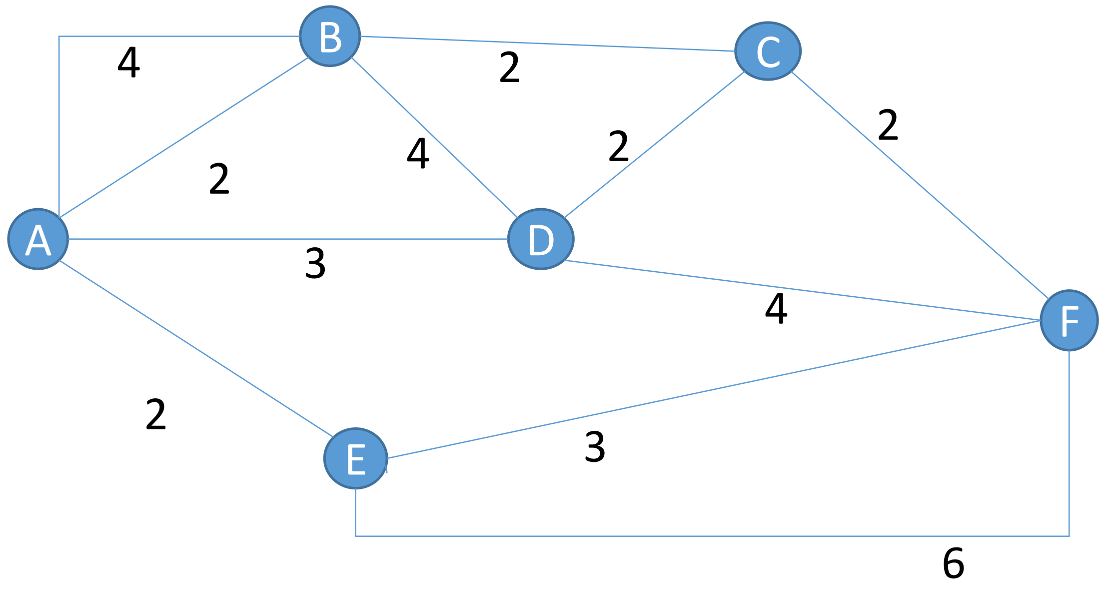
- Remove the Loops
- Remove the parallel paths higher weights.
- Create Matrix with all vertices. And Set 0 to source and Infinity to other vertices.
- Mark the smallest valued vertex.
- Find all the vertices connected to lowest valued vertex and update the values of connected vertex.
- New Destination=min(old value, marked value + edge weight)

Example



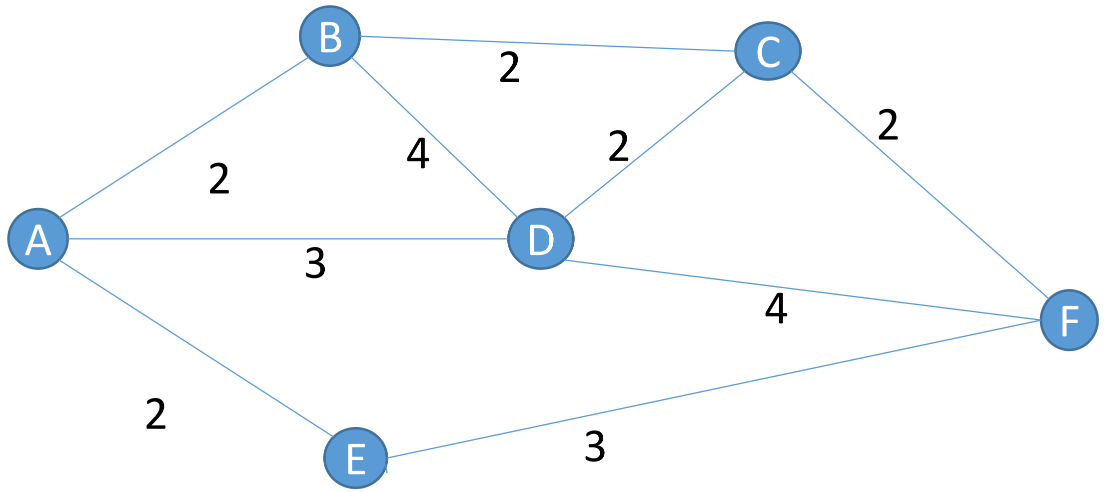
Example

- Removed the Loops



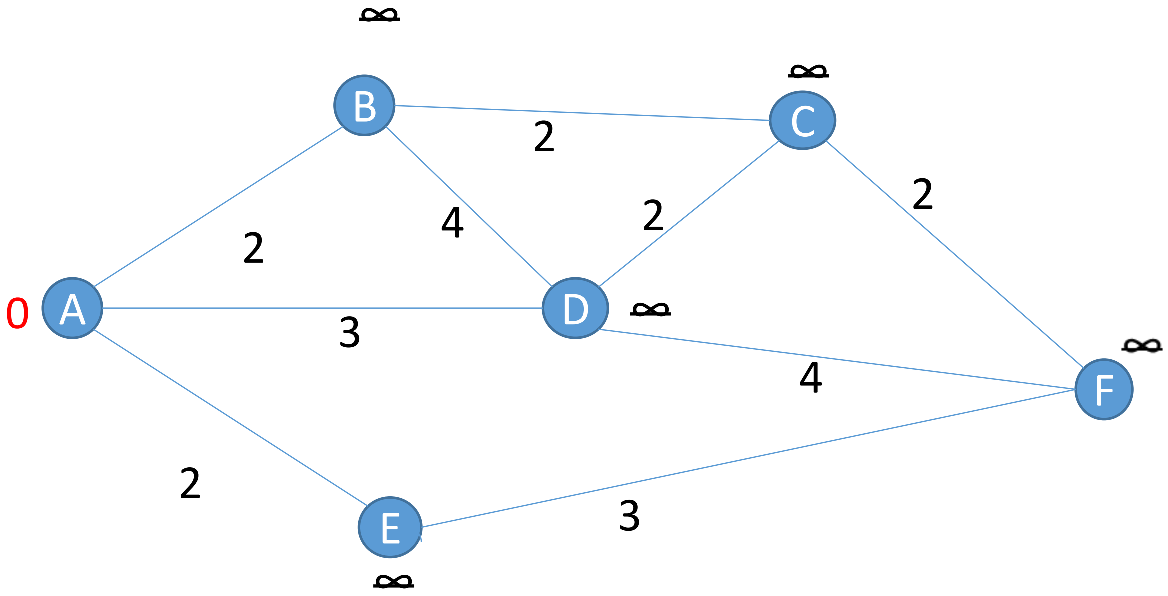
Example

- Remove the parallel paths higher weights.

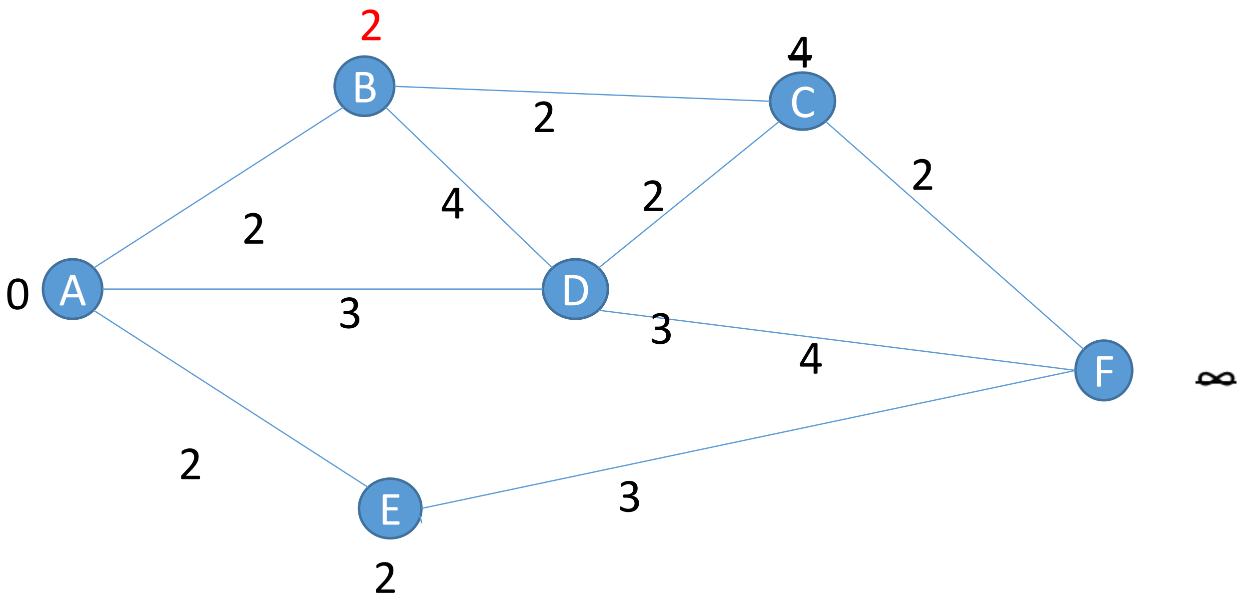


Example

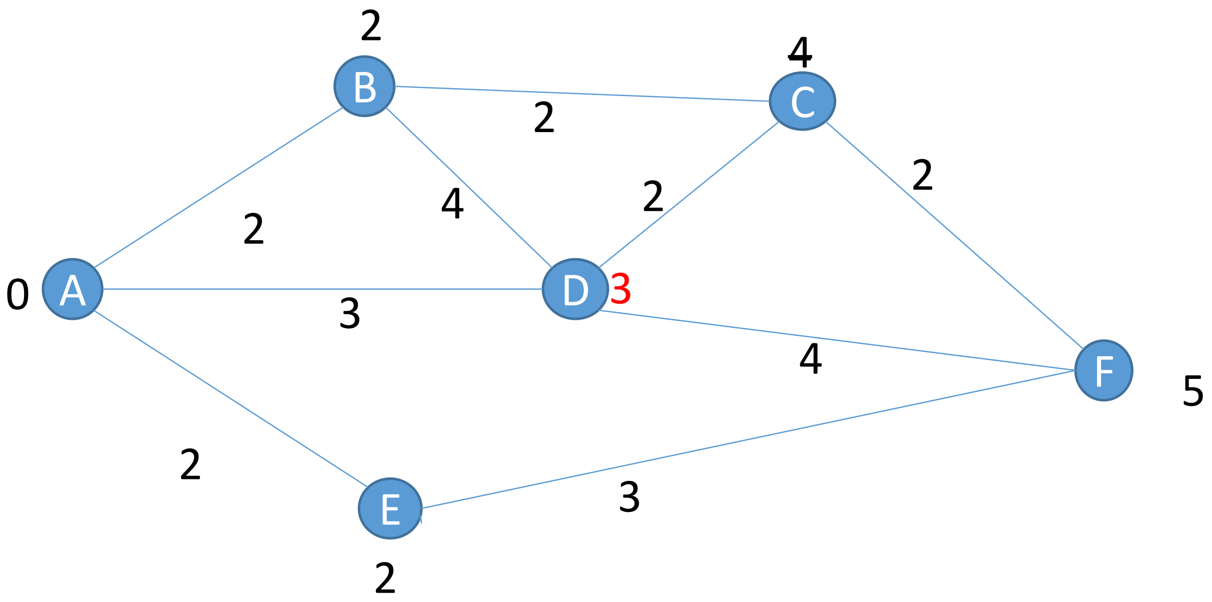
- Create Matrix with all vertices. And Set 0 to source and Infinity to other vertices.



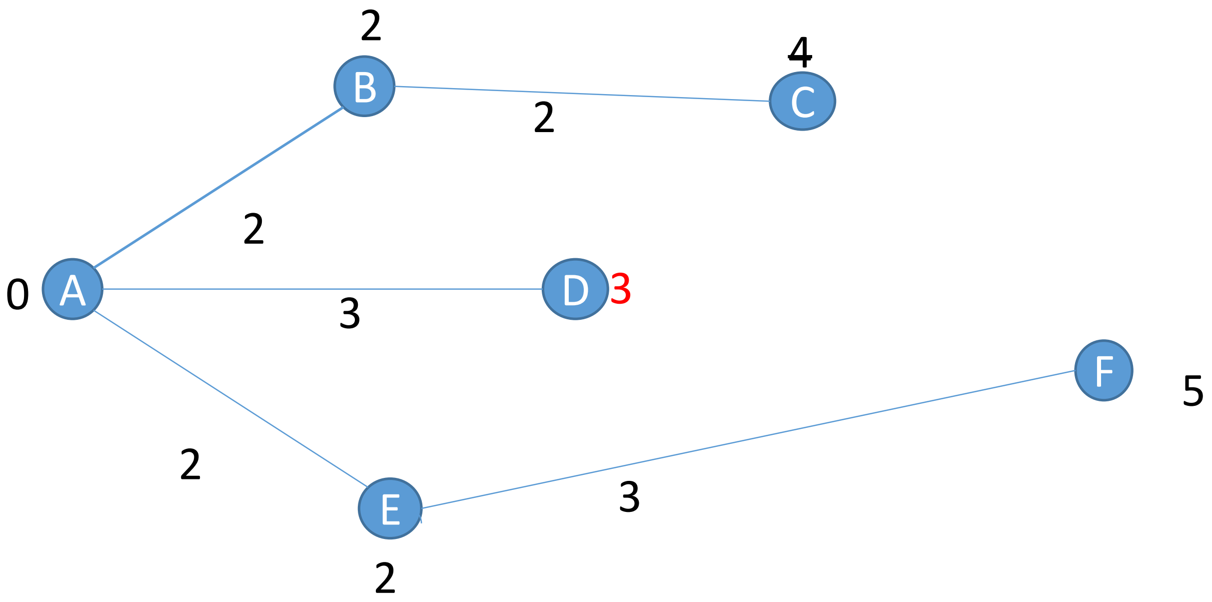
Example



Example



Example



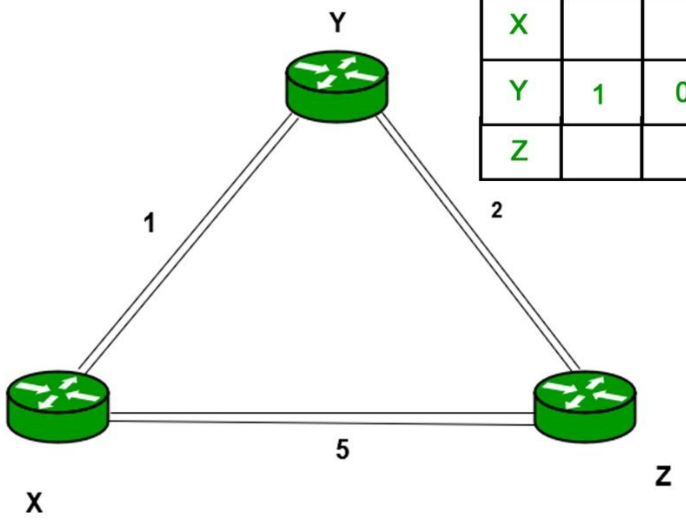
DESTINATION	COST	NEXT ROUTER
A	0	--
B	2	--
C	4	B
D	3	--
E	2	--
F	5	E

Distance Vector Routing	Link State Routing
--> Bandwidth required is less due to local sharing, small packets and no flooding.	--> Bandwidth required is more due to flooding and sending of large link state packets.
--> Based on local knowledge since it updates table based on information from neighbors.	--> Based on global knowledge i.e. it have knowledge about entire network.
--> Make use of Bellman Ford algo	--> Make use of Dijkastra's algo
--> Traffic is less	--> Traffic is more
--> Converges slowly i.e. good news spread fast and bad news spread slowly.	--> Converges faster.
--> Count to infinity problem.	--> No count to infinity problem.
--> Persistent looping problem i.e. loop will there forever.	--> No persistent loops, only transient loops.
--> Practical implementation is RIP and IGRP.	--> Practical implementation is OSPF and ISIS.

Distance Vector Algorithm –

- **Example –** Consider 3-routers X, Y and Z as shown in figure. Each router have their routing table. Every routing table will contain distance to the destination nodes

	X	Y	Z
X			
Y	1	0	2
Z			



	X	Y	Z
X	0	1	5
Y			
Z			

	X	Y	Z
X			
Y			
Z	5	2	0