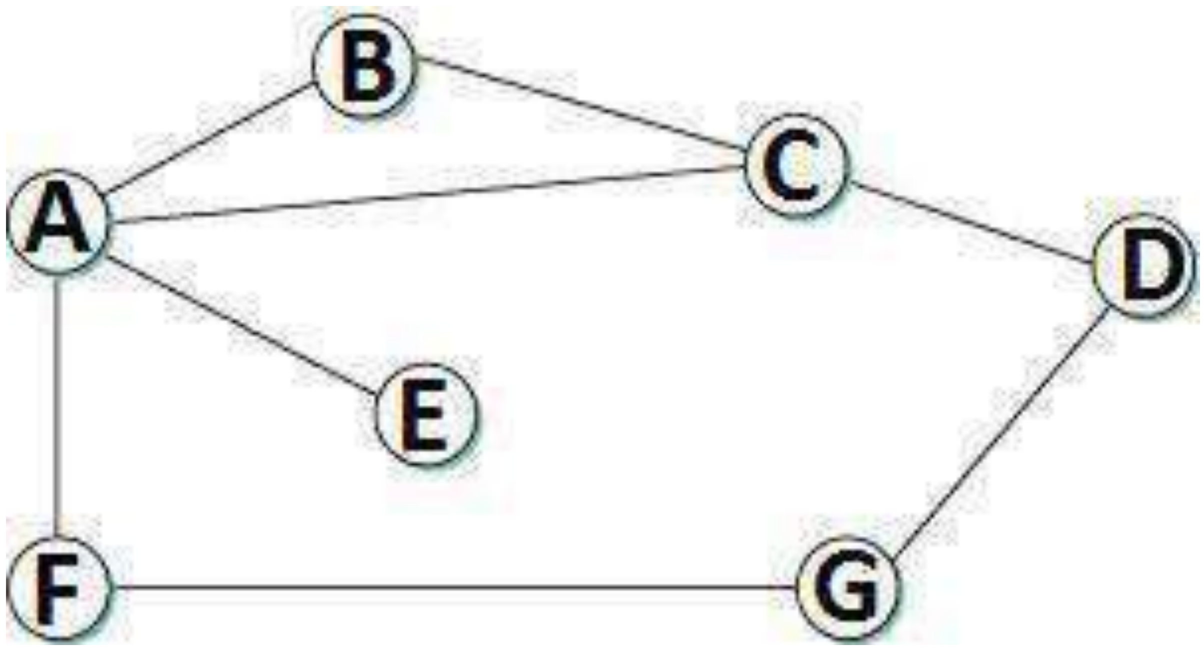
DISTANCE VECTOR ROUTING (DSR)

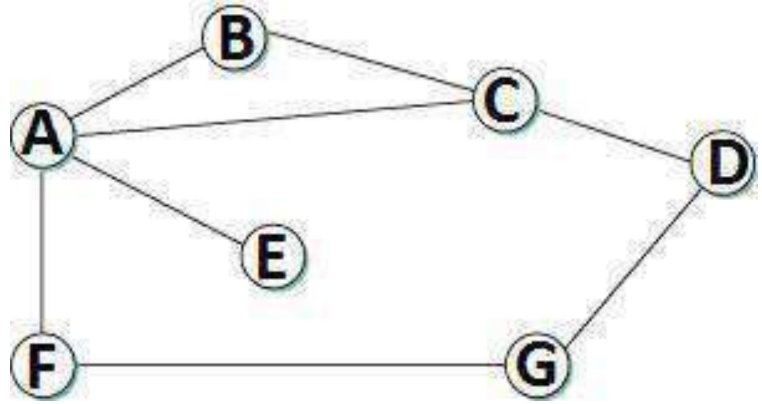
- Distance vector routing is *distributed*, i.e., algorithm is run on all nodes.
- Each node *knows* the distance (cost) to each of its directly connected neighbors.
- Nodes construct a *vector* (Destination, Cost, NextHop) and distributes to its neighbors.
- Nodes compute routing table of *minimum* distance to every other node via
- NextHop using information obtained from its neighbors.

Initial State



Initial State

- In given network, *cost* of each link is 1 hop.
- Each node sets a distance of 1 (hop) to its *immediate* neighbor and cost to itself as 0.
- Distance for non-neighbors is marked as *unreachable* with value ∞ (infinity).
- For node *A*, nodes *B*, *C*, *E* and *F* are *reachable*, whereas nodes *D* and *G* are *unreachable*.



Destination	Cost	NextHop
A	0	A
B	1	B
C	1	C
D	∞	—
E	1	E
F	1	F
G	∞	—

Node A's initial table

Destination	Cost	NextHop
A	1	A
B	1	B
C	0	C
D	1	D
E	∞	—
F	∞	—
G	∞	—

Node C's initial table

Destination	Cost	NextHop
A	1	A
B	∞	—
C	∞	—
D	∞	—
E	∞	—
F	0	F
G	1	G

Node F's initial table

- The initial table for all the nodes are given below

Initial Distances Stored at Each Node (Global View)							
Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	∞	1	1	∞
B	1	0	1	∞	∞	∞	∞
C	1	1	0	1	∞	∞	∞
D	∞	∞	1	0	∞	∞	1
E	1	∞	∞	∞	0	∞	∞
F	1	∞	∞	∞	∞	0	1
G	∞	∞	∞	1	∞	1	0

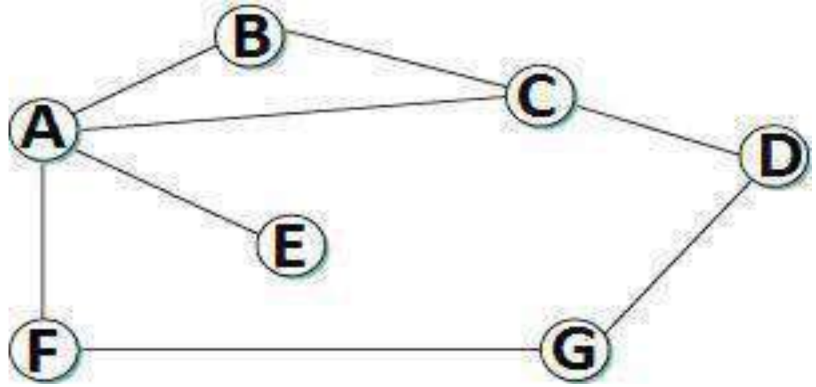
- Each node *sends* its initial table (distance vector) to neighbors and receives their estimate.
- Node *A* sends its table to nodes *B*, *C*, *E* & *F* and receives tables from nodes *B*, *C*, *E* & *F*.
- Each node *updates* its routing table by comparing with each of its neighbor's table
- For each destination, Total Cost is computed as:
  - ***Total Cost*** = Cost (*Node* to *Neighbor*) + Cost (*Neighbor* to *Destination*)
- If Total Cost < Cost then
  - ***Cost*** = Total Cost and NextHop = *Neighbor*
- Node *A* *learns* from *C*'s table to reach node *D* and from *F*'s table to reach node *G*.
- Total Cost to reach node *D* via *C* = Cost (*A* to *C*) + Cost(*C* to *D*)

  Cost = 1 + 1 = 2.
  - Since 2 < ∞, entry for destination *D* in *A*'s table is changed to (*D*, 2, *C*)
  - Total Cost to reach node *G* via *F* = Cost(*A* to *F*) + Cost(*F* to *G*) = 1 + 1 = 2
  - Since 2 < ∞, entry for destination *G* in *A*'s table is changed to (*G*, 2, *F*)
- Each node builds *complete* routing table after few exchanges amongst its neighbors.

Node A's final routing table

Destination	Cost	NextHop
A	0	A
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	2	F

- System stabilizes when all nodes have complete routing information, i.e., **convergence.**
- Routing tables are exchanged *periodically or* in case of *triggered update*.
- The final distances stored at each node is given below:



Final Distances Stored at Each Node (Global View)							
Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	1
G	2	3	2	1	3	1	0

Updation of Routing Tables

- 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 every so often, even if nothing has changed.
- The frequency of these periodic updates varies from protocol to protocol, but it is typically on the order of several seconds to several minutes.

Triggered Update

- In this case, whenever a node notices a link failure or receives an update from one of its neighbors that causes it to change one of the routes in its routing table.
- Whenever a node's routing table changes, it sends an update to its neighbors, which may lead to a change in their tables, causing them to send an update to their neighbors.

Count-To-Infinity (or) Loop Instability Problem

- Suppose link from node *A* to *E* goes *down*.
  - Node *A* advertises a distance of ∞ to *E* to its neighbors
  - Node B receives periodic update from C before A's update reaches B
  - Node *B* updated by *C*, concludes that *E* can be reached in 3 hops via *C*
  - Node *B* advertises to *A* as 3 hops to reach *E*
  - Node *A* in turn updates *C* with a distance

- Thus nodes update each other until cost to *E* reaches *infinity*, i.e., *no convergence*.
- Routing table does not stabilize.
- This problem is called *loop instability* or *count to infinity*

Solution to Count-To-Infinity (or) Loop Instability Problem :

- *Infinity* is redefined to a small number, say 16.
- Distance between any two nodes can be 15 hops maximum. Thus distance vector routing *cannot be used* in large networks.
- When a node updates its neighbors, it does not send those routes it learned from each neighbor back to that neighbor. This is known as **split horizon**.
- **Split horizon with poison reverse** allows nodes to advertise routes it learnt from a node back to that node, but with a warning.