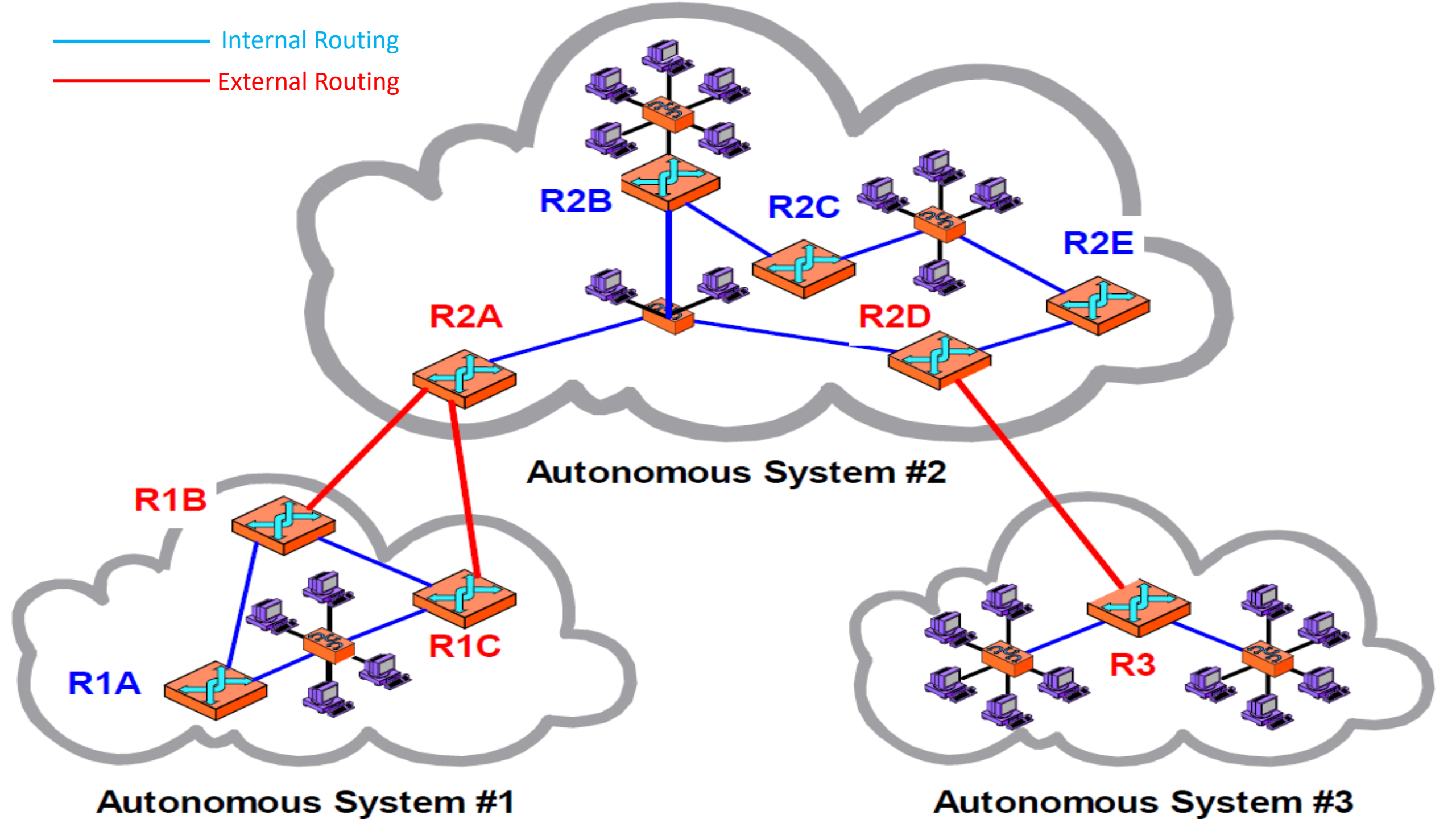
Interior and Exterior Gateway Routing Protocols

- One routing protocol cannot handle the task of updating routing tables of all routers on the internet. For this reason, an internet is divided into autonomous systems.
- An autonomous system (AS) exhibits the following characteristics:
  - AS is a set of routers and networks managed by a single organization.
  - An AS consists of a group of routers exchanging information via a common routing protocol.
  - AS is connected that is, there is a path between any pair of nodes , except in the times of failure.
- Interior gateway routing protocol is used for passing routing information between routers within an AS. The protocol used to pass routing information between routers in different ASs are refered as an exterior gateway routing protocol.
- AS are connected by linking a router in one AS to a router in another AS. An AS consists of a set of routers with two different types of connectivity:
- Internal Routers - These routers in an AS connect only to other routers in the same AS. These run interior gateway routing protocols.
- Border Routers – These routers in an AS connect both to routers within an AS and to routers in one or more AS. These devices are responsible for passing traffic between the AS and the rest of the internetwork. They run both interior and exterior routing protocols.


Internal Routing
External Routing
R2B
R2C
R2E
R2A
R2D
Autonomous System #2
R1B
R1C
R1A
R3
Autonomous System #1
Autonomous System #3

Shortest Path Routing Algorithms

- In shortest path routing, a path between the source and destination node is chosen that has the least cost.
- These algorithms are also called least cost path routing algorithms. In such algorithms, a cost is associated with each link.
- This link cost is a usually non-negative and proportional to link's current traffic load.
- The link cost is defined on both directions between each pair of nodes.
- Several least cost path routing algorithms have been developed for packet switched networks. In particular, following two algorithms have been most effective and widely used. They are:
  - Dijkstra's Algorithm
  - Bellman Ford Algorithm

Shortest Path Routing Algorithms

- A network is modelled as a graph $G = (N, E)$ which is a set N of nodes and a collection E of edges.
- $i \in N$ and $j \in N$ refer to nodes / stations in the network.
- $d_{i,j}$ is the direct link cost / metric between i and j , with:
  - $0 \leq d_{i,j} < \infty$ when i and j are adjacent nodes.
  - $d_{i,j} = \infty$ when i and j are non adjacent nodes.
- $D_{i,j}$ represents the total cost of the least cost path from i to j.
- N_i represents the set of nodes adjacent to node i.

Dijkstra Algorithm

- Dijkstra's algorithm can be stated as: Find the shortest paths from a given source node to all other nodes by developing the paths in order of increasing path length.
- Dijkstra Algorithm is the centralized routing algorithm that is it takes the connectivity between all nodes and all link costs as inputs and stores them in central location.
- Dijkstra's algorithm is iterative and has the property that after the kth iteration of the algorithm, the least cost paths are known to k destination nodes.
- This algorithm consists of an initialization step followed by a loop. The number of times the loop is executed is equal to the number of nodes in the network.
- Upon termination, the algorithm will have calculated the shortest paths from the source node to every other node in the network.
- Dijkstra cannot handle negative metrics.
- It is greedy: in every situation it makes the choice that is currently the best, without regard to future situations.

Dijkstra Algorithm

1. Define

s = source node

$P(v)$ = predecessor node(neighbor of v)

$\acute{N}$ = set of visited nodes by the algorithm

$d_{i,j}$ = direct link cost between node i to j

$D_{i,j}$ = total cost of the least cost path from i to j.

2. Initialization

$N' = \{s\}$

for all nodes v, if v is a neighbor of s

then $D_{s,v}=d_{s,v}$

else $D_{s,v} = \infty$

3. Loop

find w not in N' such that $D_{s,w}$ is a minimum

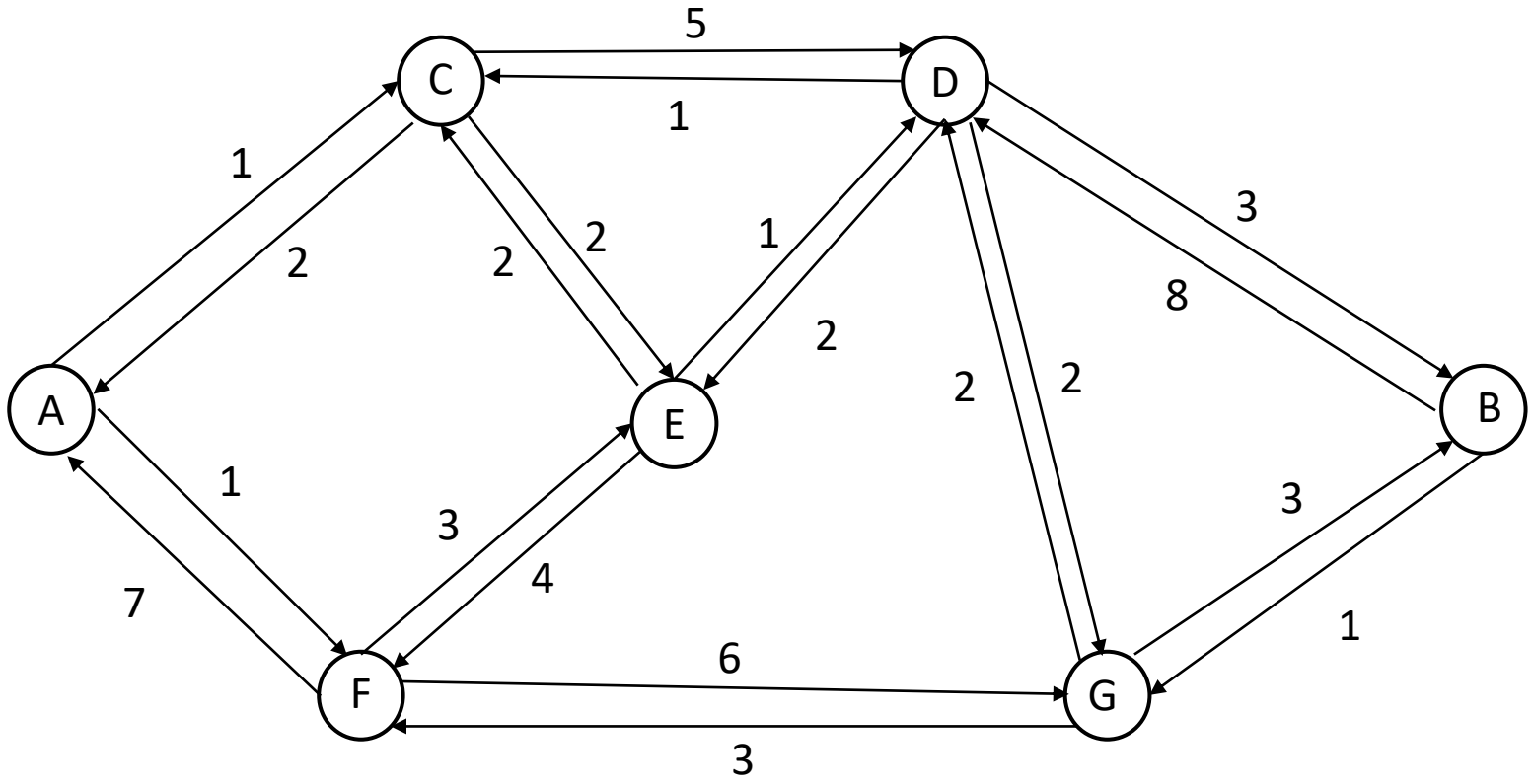
add w to N', update $D_{s,v}$ for each neighbor v of w and not in N'

$D_{s,v} = \min(D_{s,v}, D_{s,w} + d_{w,v})$ /* new cost to v is either old cost to v or known least path cost to w plus cost from w to v */

until $N' = N$

Execution of Dijkstra Algorithm

Using Dijkstra's algorithm, find the least cost path from node A to node B in the given network.



Execution of Dijkstra Algorithm

Step 1 & 2 - Define & Initialization

s = A

P(A) = A

$\acute{N}$ = {A}

for all nodes in the network,

$D_{A,C}=d_{A,C} = 1$ (Direct cost since it is a neighbor of A)

$D_{A,F}=d_{A,F} = 1$ (Direct cost since it is a neighbor of A)

$D_{A,E} = \infty$ (Not directly connected)

$D_{A,D} = \infty$ (Not directly connected)

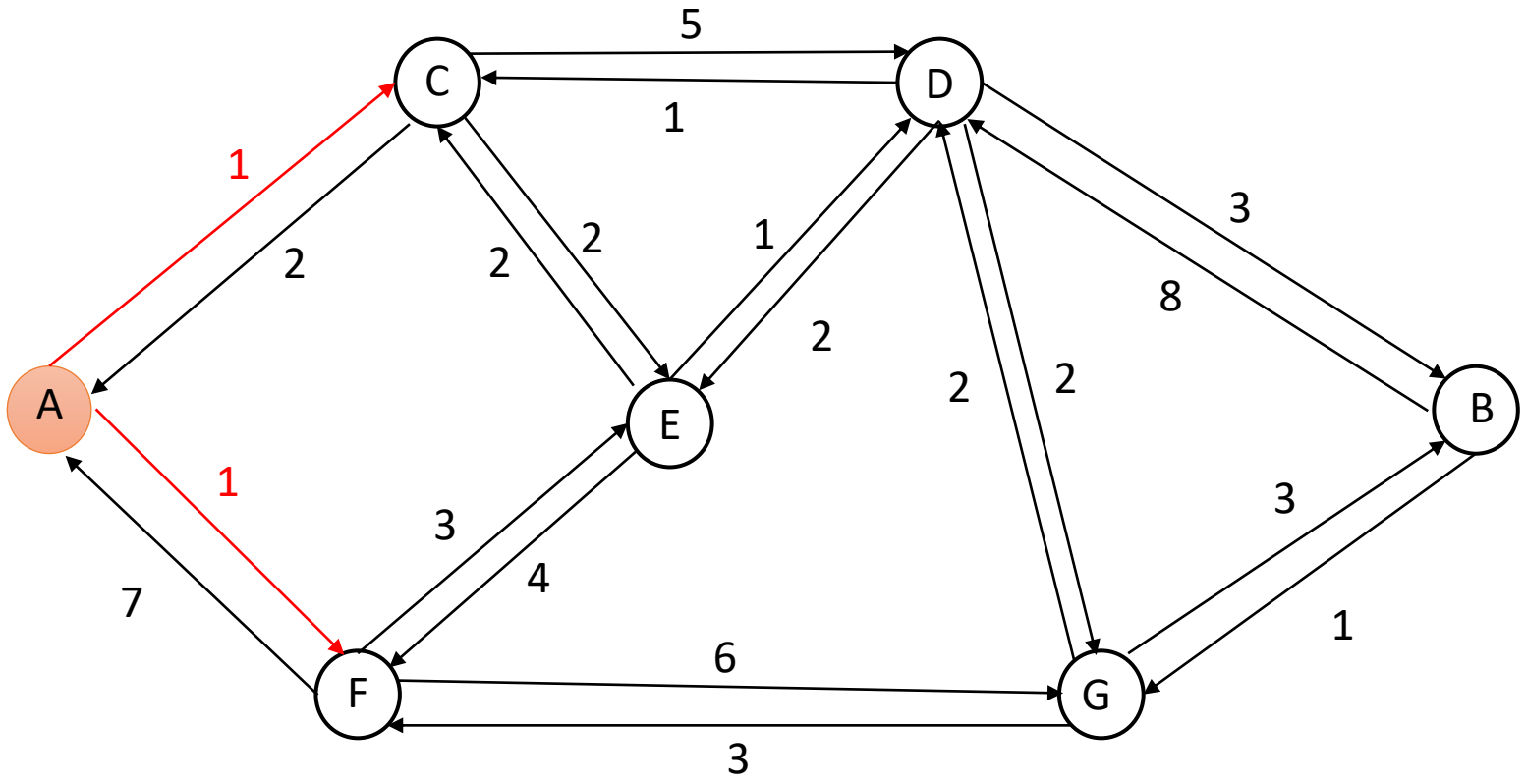
$D_{A,G} = \infty$ (Not directly connected)

$D_{A,B} = \infty$ (Not directly connected)

Both nodes 'C' and 'F' have cost of 1 from source node. Anyone of these nodes can be chosen but we choose Node C for next iteration.

Execution of Dijkstra Algorithm

Step	$\bar{N}$	$D_{A,C}$, P(C)	$D_{A,F}$,$P(F)$	$D_{A,E}$, P(E)	$D_{A,D}$, P(D)	$D_{A,G}$,P(G)	$D_{A,B}$, P(B)
1	{A}	1 , A (AC)	1 , A (AF)	∞	∞	∞	∞



Execution of Dijkstra Algorithm

Step 3 - Loop

$$\acute{N} = \{A, C\}$$

$$P(C) = A \text{ (predecessor node of C) path : } A \rightarrow C$$

for each neighbor of C and not in N' we update the cost

$D_{A,E} = \min(D_{A,E}, D_{A,C} + d_{C,E})$ /* new cost to E is either old cost to E or known least path cost to C plus cost from C to E */

$$D_{A,E} = \min(\infty, 1 + 2) = 3$$

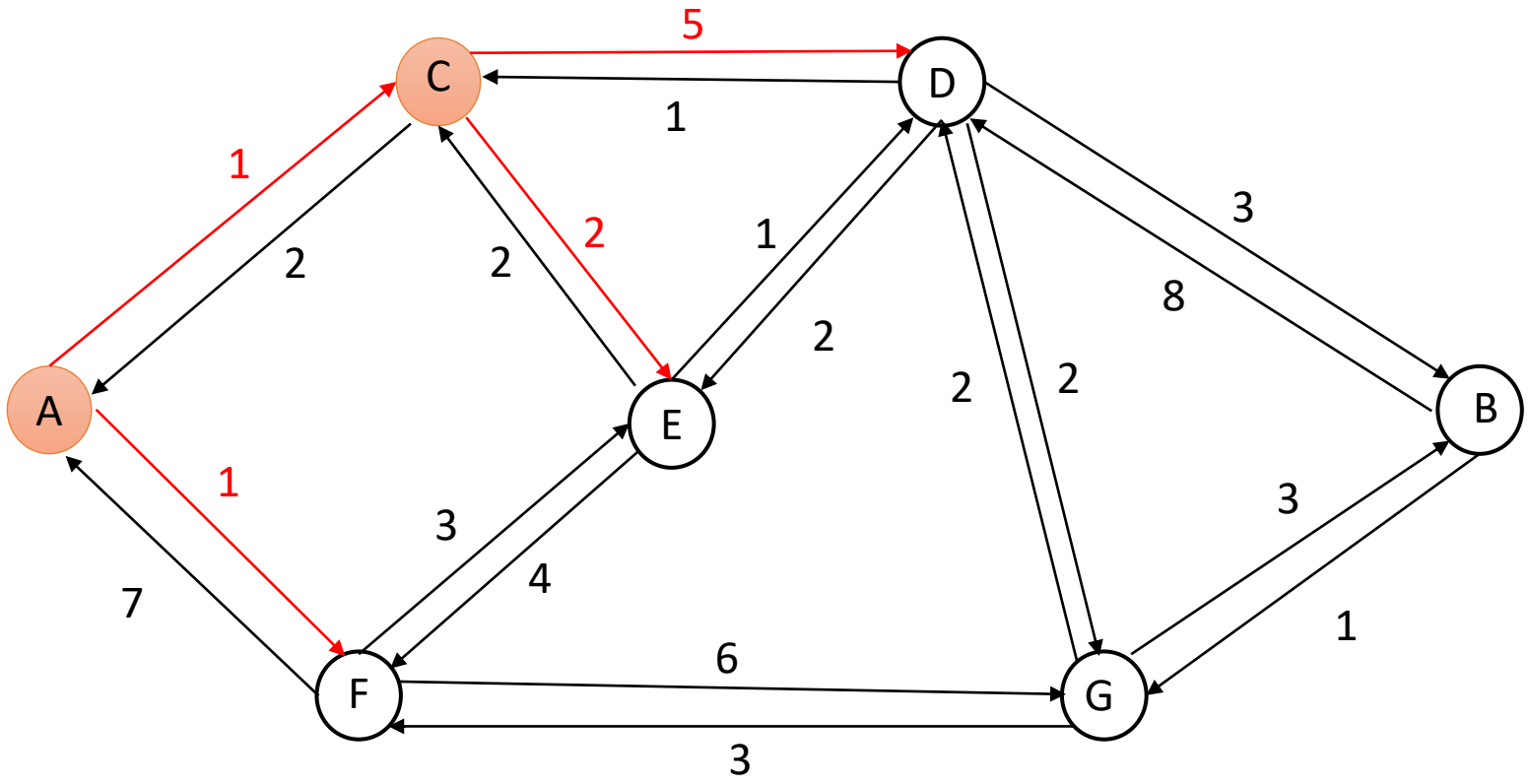
$D_{A,D} = \min(D_{A,D}, D_{A,C} + d_{C,D})$ /* new cost to D is either old cost to D or known least path cost to C plus cost from C to D */

$$D_{A,D} = \min(\infty, 1 + 5) = 6$$

Both nodes 'C' and 'F' have cost of 1 from source node. We chose Node C for second iteration. Now we choose Node 'F' for third iteration and continue step no. 3 of algorithm until all the nodes have been visited and included in $\acute{N}$

Execution of Dijkstra Algorithm

Step	$\bar{N}$	$D_{A,C}$, P(C)	$D_{A,F}$,P(F)	$D_{A,E}$, P(E)	$D_{A,D}$, P(D)	$D_{A,G}$,P(G)	$D_{A,B}$, P(B)
1	{A}	1 , A (AC)	1 , A (AF)	∞	∞	∞	∞
2	{A,C}	1 , A (AC)	1 , A (AF)	3 , C (ACE)	6 , C (ACD)	∞	∞



Execution of Dijkstra Algorithm

Third Iteration (Loop)

$$\acute{N} = \{A, C, F\}$$

$$P(F) = A \text{ (predecessor node of F) path : } A \rightarrow F$$

for each neighbor of F and not in N' we update the cost

$D_{A,E} = \min(D_{A,E}, D_{A,F} + d_{F,E})$ /* new cost to E is either old cost to E or known least path cost to F plus cost from F to E */

$$D_{A,E} = \min(3, 1 + 3) = 3$$

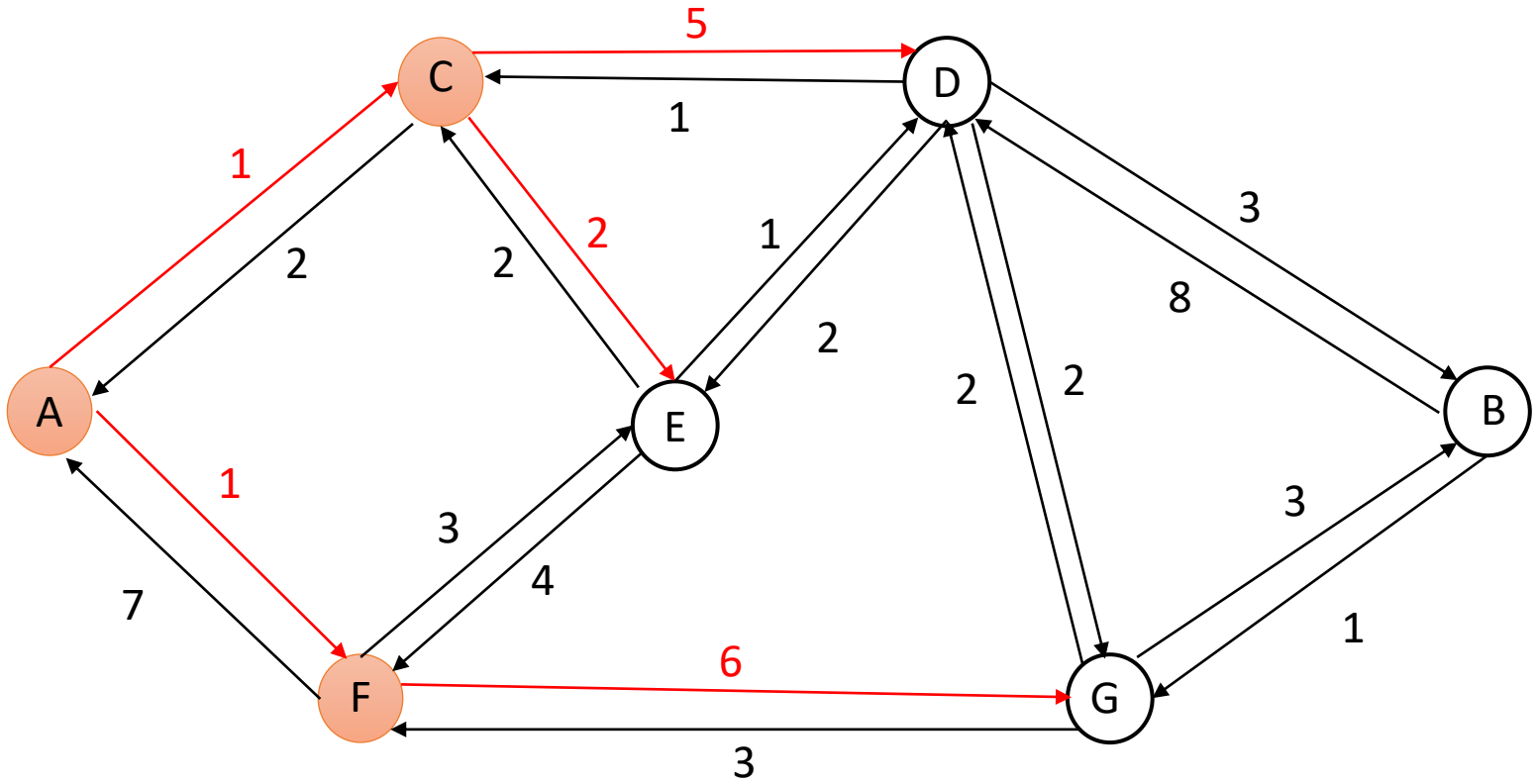
$D_{A,G} = \min(D_{A,G}, D_{A,F} + d_{F,G})$ /* new cost to G is either old cost to G or known least path cost to F plus cost from F to G */

$$D_{A,G} = \min(\infty, 1 + 6) = 7$$

Both nodes 'C' and 'F' have cost of 1 from source node and have been added to $\acute{N}$. Now we choose Node 'E ' , which has a cost of 3 from source node, for fourth iteration and continue step no. 3 of algorithm until all the nodes have been visited and included in $\acute{N}$.

Execution of Dijkstra Algorithm

Step	$\bar{N}$	$D_{A,C}$, P(C)	$D_{A,F}$,$P(F)$	$D_{A,E}$, P(E)	$D_{A,D}$, P(D)	$D_{A,G}$,P(G)	$D_{A,B}$, P(B)
1	{A}	1 , A (AC)	1 , A (AF)	∞	∞	∞	∞
2	{A,C}	1 , A (AC)	1 , A (AF)	3 , C (ACE)	6 , C (ACD)	∞	∞
3	{A,C,F)	1 , A (AC)	1 , A (AF)	3 , C (ACE)	6 , C (ACD)	7 , F (AFG)	∞



Execution of Dijkstra Algorithm

Fourth Iteration (Loop)

$$\acute{N} = \{A, C, F, E\}$$

$$P(E) = C \text{ (predecessor node of E) path : } A \rightarrow C \rightarrow E$$

for each neighbor of E and not in N' we update the cost

$$D_{A,D} = \min(D_{A,D}, D_{A,E} + d_{E,D}) \quad /* \text{ new cost to } D \text{ is either old cost to } D \text{ or known least path cost to } E \text{ plus cost from } E \text{ to } D \ */$$

$$D_{A,D} = \min(6, 3 + 1) = 4$$

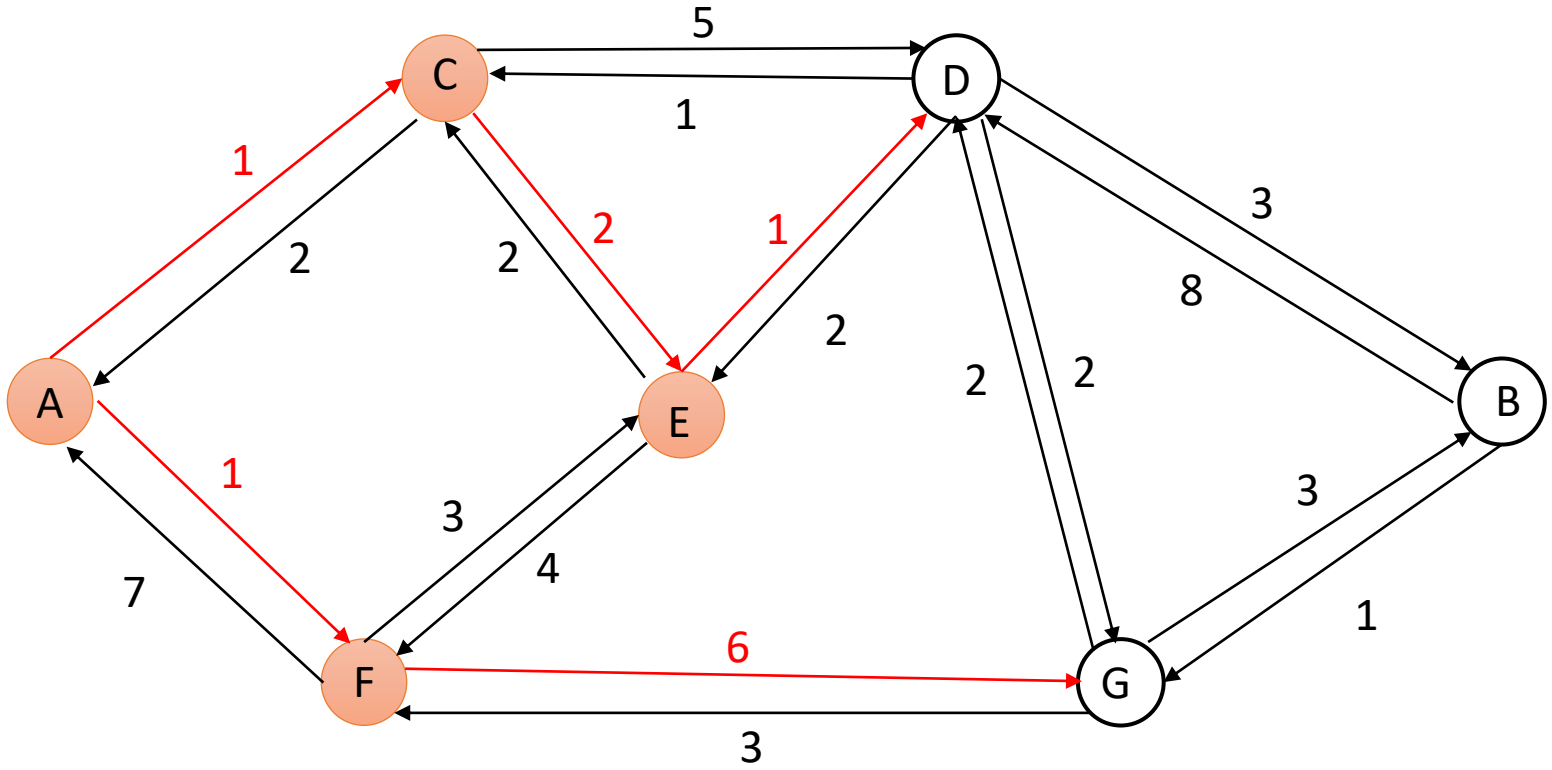
$$D_{A,F} = \min(D_{A,F}, D_{A,E} + d_{E,F}) \quad /* \text{ new cost to } F \text{ is either old cost to } F \text{ or known least path cost to } E \text{ plus cost from } E \text{ to } F \ */$$

$$D_{A,F} = \min(1, 3 + 4) = 1$$

Nodes 'C', 'F' and 'E' have minimum cost from source node and have been added to $\acute{N}$. Now we choose Node 'D', which has a cost of 4 from source node, for fifth iteration and continue step no. 3 of algorithm until all the nodes have been visited and included in $\acute{N}$.

Execution of Dijkstra Algorithm

Step	$\bar{N}$	$D_{A,C}$, P(C)	$D_{A,F}$,$P(F)$	$D_{A,E}$, P(E)	$D_{A,D}$, P(D)	$D_{A,G}$,P(G)	$D_{A,B}$, P(B)
1	{A}	1 , A (AC)	1 , A (AF)	∞	∞	∞	∞
2	{A,C}	1 , A (AC)	1 , A (AF)	3 , C (ACE)	6 , C (ACD)	∞	∞
3	{A,C,F)	1 , A (AC)	1 , A (AF)	3 , C (ACE)	6 , C (ACD)	7 , F (AFG)	∞
4	{A,C,F,E}	1 , A (AC)	1 , A (AF)	3 , C (ACE)	4 , E (ACED)	7 , F (AFG)	∞



Execution of Dijkstra Algorithm

Final Picture (All Nodes have been added to $\acute{N}$)

Step	$\bar{N}$	$D_{A,C}$, P(C)	$D_{A,F}$,$P(F)$	$D_{A,E}$, P(E)	$D_{A,D}$, P(D)	$D_{A,G}$,P(G)	$D_{A,B}$, P(B)
1	{A}	1 , A (AC)	1 , A (AF)	∞	∞	∞	∞
2	{A,C}	1 , A (AC)	1 , A (AF)	3 , C (ACE)	6 , C (ACD)	∞	∞
3	{A,C,F)	1 , A (AC)	1 , A (AF)	3 , C (ACE)	6 , C (ACD)	7 , F (AFG)	∞
4	{A,C,F,E}	1 , A (AC)	1 , A (AF)	3 , C (ACE)	4 , E (ACED)	7 , F (AFG)	∞
5	{A,C,F,E,D}	1 , A (AC)	1 , A (AF)	3 , C (ACE)	4 , E (ACED)	6 , D (ACEDG)	7 , D (ACEDB)
6	{A,C,F,E,D,G}	1 , A (AC)	1 , A (AF)	3 , C (ACE)	4 , E (ACED)	6 , D (ACEDG)	7 , D (ACEDB)
7	{A,C,F,E,D,G,B}	1 , A (AC)	1 , A (AF)	3 , C (ACE)	4 , E (ACED)	6 , D (ACEDG)	7 , D (ACEDB)

Dijkstra's Algorithm - Performance

- When this algorithm terminates, we have, for each node, its predecessor along the least cost path from the source node.
- For each predecessor, we also have its predecessor, and so in this manner we can construct the entire path from the source to all destinations.
- What is the computational complexity of this algorithm?
- If we have n nodes (excluding the source), then in the first iteration, we need to search through all n nodes(not in $\acute{N}$) to determine the nodes that has the minimum cost.
- In the second iteration, we need to check $n - 1$ nodes to determine the minimum cost; in the third iteration $n - 2$ nodes, and so on.
- Overall, the total number of nodes we need to search through over all the iterations is

$$n(n + 1)/2.$$

- So this algorithm has worst case complexity of $O(n)^2$.

Dijkstra Algorithm - Performance

- Dijkstra's algorithm requires that each node must have complete topological information about the network. That is, each node must know the link costs of all links in the network. Thus, for this algorithm, information must be exchanged with all other nodes.
- The algorithm converges under static conditions of topology, and link costs. If the link costs change over time, the algorithm will attempt to catch up with these changes.