CN (IT-3001)

Network Layer: Routing Algorithm

Prof. Amit Jha

School of Electronics Engineering (SOEE)

KIIT Deemed to be University

Disclaimer: *The contents in this slide have been referred from many sources which I do not claim as my own. Some of the content has been modified for easier understanding of the students without any malafide intention. This slide is only for educational purpose strictly, and not for the commercial purpose. Images portrayed (if any) are not to hurt the sentiments of any person.*

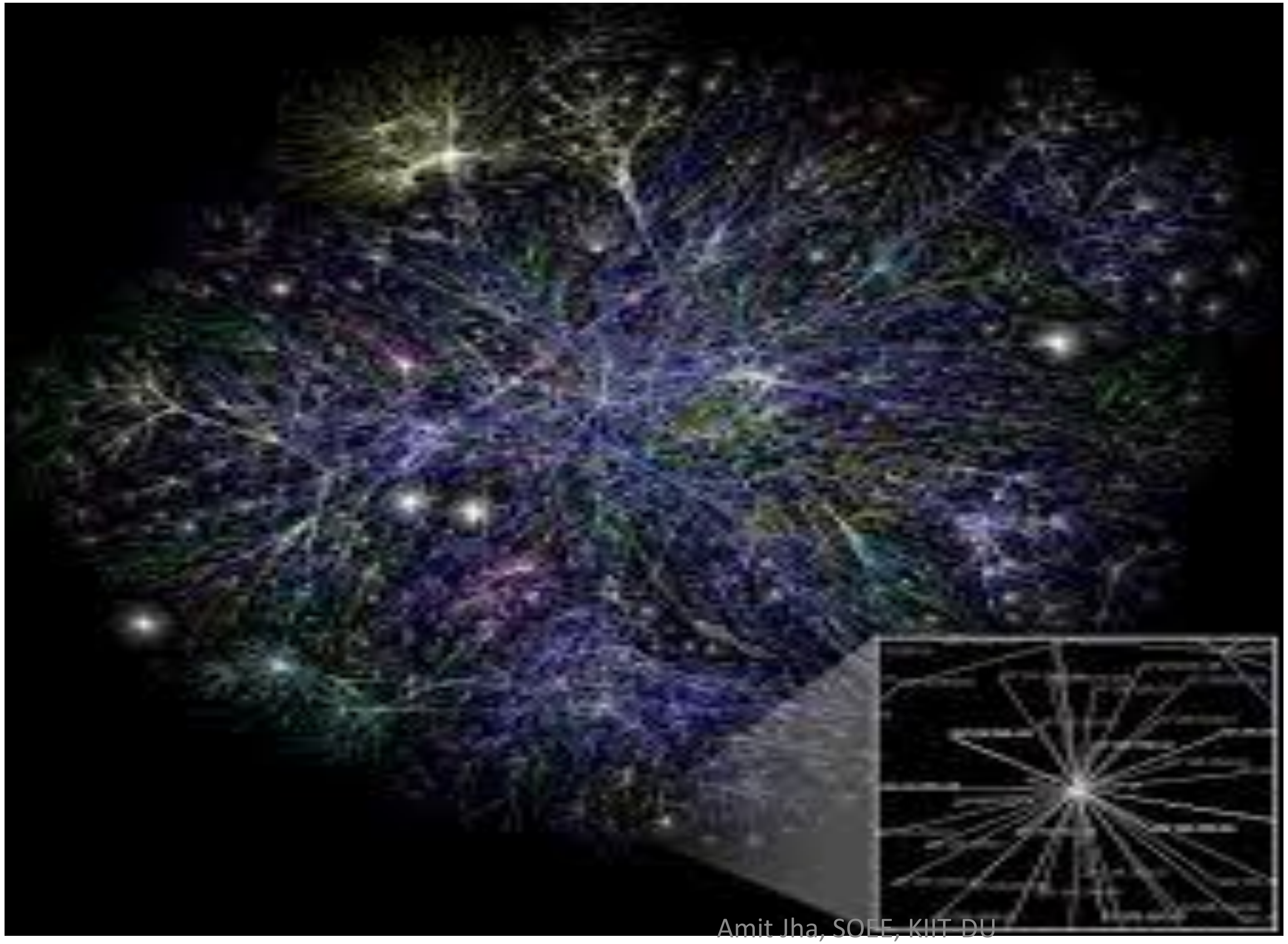
Objectives

- The objective of this module is to discus following concepts…
    1. Routing Algorithm

Delivery, Forwarding, and Routing: A view

- **Delivery:** The handling of the packets by the network layer. If the final destination of the packet is a host connected to the same physical network as the deliverer, it is known as ***direct delivery***; otherwise ***indirect delivery***.
- **Forwarding:** It means to place the packet in its route to its destination, i.e., to show a path to packets.
- **Routing:** A mechanism to show an **optimum** path for forwarding.

Why do we need Routing?



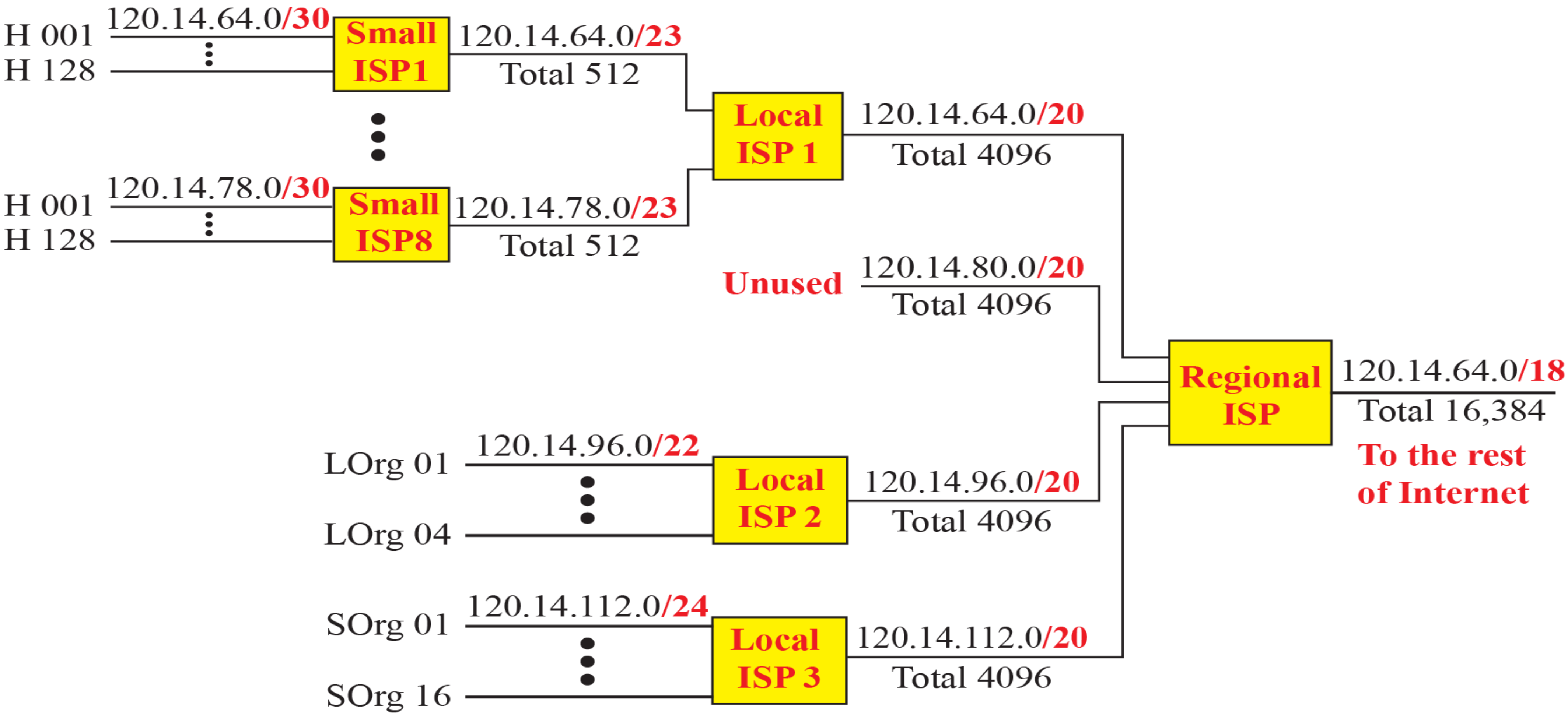
Forwarding

Let's retake example 4.16 of this module to understand the significance of forwarding.

Example 4.16:- Consider a regional ISP is granted 16,384 addresses, starting from 120.14.64.0/18. The regional ISP has decided to divide these addresses into 4 subblocks, each with 4096 addresses. All these subblocks are assigned to Local ISP namely, *Local ISP1*, *Local ISP2*, *Local ISP3* and *Local ISP4*. The *Local ISP1* has divided its assigned subblock into 8 smaller sub blocks and assigned each to 8 smaller ISP, namely *Small ISP1 to Small ISP8*. Each Small ISP provides services to 128 households, namely *H001 to H128*, each using 4 addresses. The *Local ISP3* has divided its block into 4 blocks and has assigned addresses to 4 large organizations, namely LOrg1 to LOrg4, with 1024 addresses in each. The *Local ISP4* has divided its block into 16 blocks and has assigned each block to 16 small organizations, namely SOrg1 to SOrg16, with 256 addresses in each. Design the hierarchy of addressing using the concept of VLSM. Clearly state the assumptions, if any.

Solution:- refer next slide.

- To be noted,
  - This example of IP address distribution shows the perfect example, where the hierarchical routing has been used.
  - All routers in the Internet send a packet with destination address 120.14.64.0 to 120.14.127.255 to the regional ISP.
  - The regional ISP sends every packet with destination address 120.14.64.0 to 120.14.79.255 to the local ISP1.
  - The Local ISP1 sends every packet with destination address 120.14.64.0 to 120.14.64.3 to the household H001.

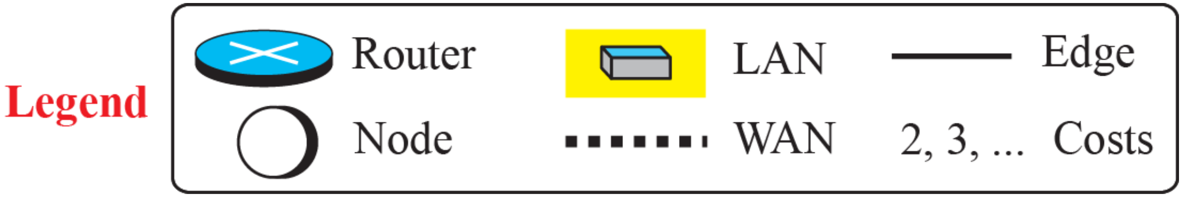

H 001
H 128
120.14.64.0/30
Small ISP1
120.14.64.0/23
Total 512
H 001
H 128
120.14.78.0/30
Small ISP8
120.14.78.0/23
Total 512
Local ISP 1
120.14.64.0/20
Total 4096
Unused
120.14.80.0/20
Total 4096
Regional ISP
120.14.64.0/18
Total 16,384
To the rest of Internet
LOrg 01
LOrg 04
120.14.96.0/22
Local ISP 2
120.14.96.0/20
Total 4096
SOrg 01
SOrg 16
120.14.112.0/24
Local ISP 3
120.14.112.0/20
Total 4096

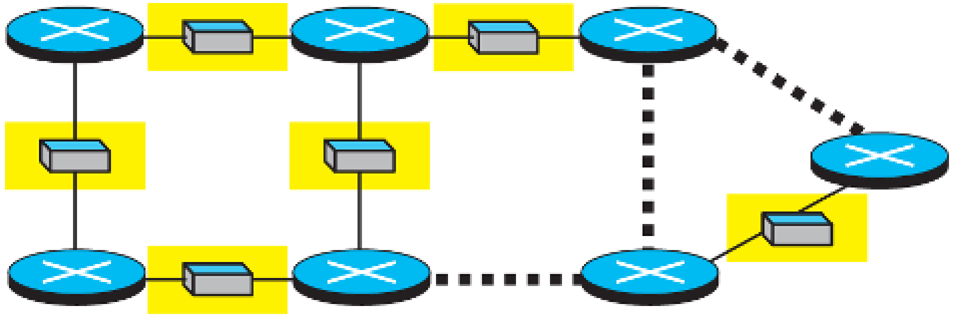
Basics of Routing:

- The following are the broad classification of routing.

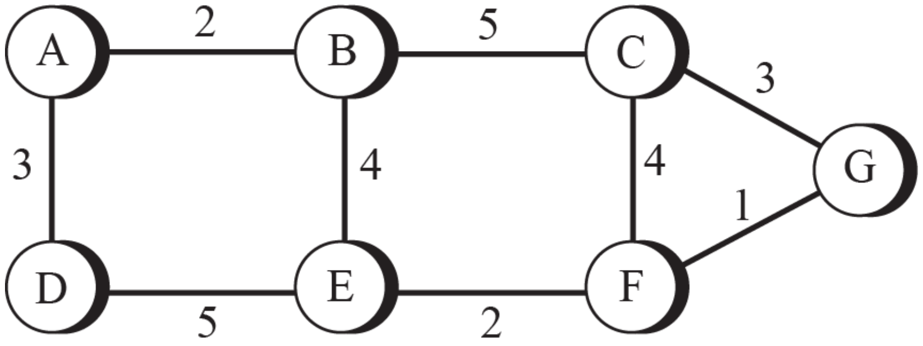
1. *Unicast Routing*: In this, a datagram is destined for only one destination.
2. *Multicast Routing:* In this, a datagram is destined for several destination.
3. *Broadcast Routing:* In this, a datagram is supposed to be delivered to all hosts on the Internet.

Basics of Routing: An internet and its graphical representation





a. An internet



b. The weighted graph

Unicast Routing:

- The unicast routing in the Internet, with a large number of routers and a huge number of hosts, can be done only by using hierarchical routing: routing in several steps using different routing algorithm.
- First, we discuss the different routing algorithms and then we will see their implementation in the today’s Interenet.

Routing Table

- It consists of routing information.
- It is of two types:
  1. **Static Routing Table** and 2. **Dynamic Routing Table**
- ***Static Routing Table:*** Routing information is entered and maintain manually by administrator.
- ***Dynamic Routing Table:*** Routing table is updated periodically by using one of the dynamic routing protocols such as RIP, OSPF, or BGP.

Note: Despite of having drawback of static routing table, it can be used in small internet that does not change very often or in an experimental internet for troubleshooting.

Routing Algorithm

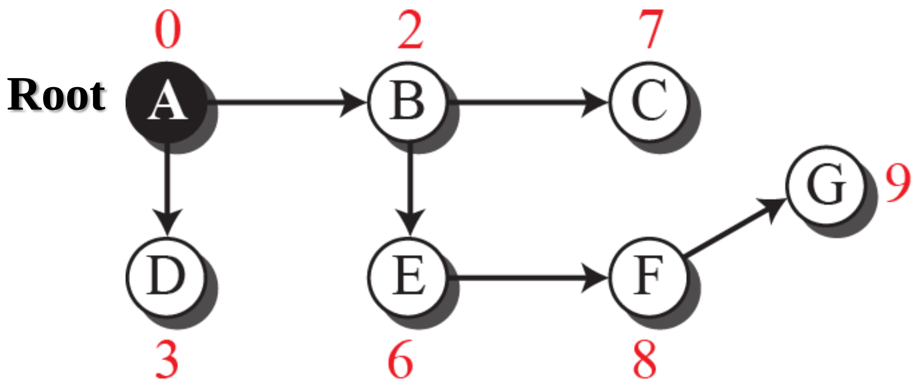
- Two most popular routing algorithm:

1) **Bellman-Ford Algorithm:** It is used in **Distance Vector Routing Protocols**.
    - Neighbours exchange list of distances to destinations.
    - Best next hop is determined for each destination.
    - Uses distributed approach

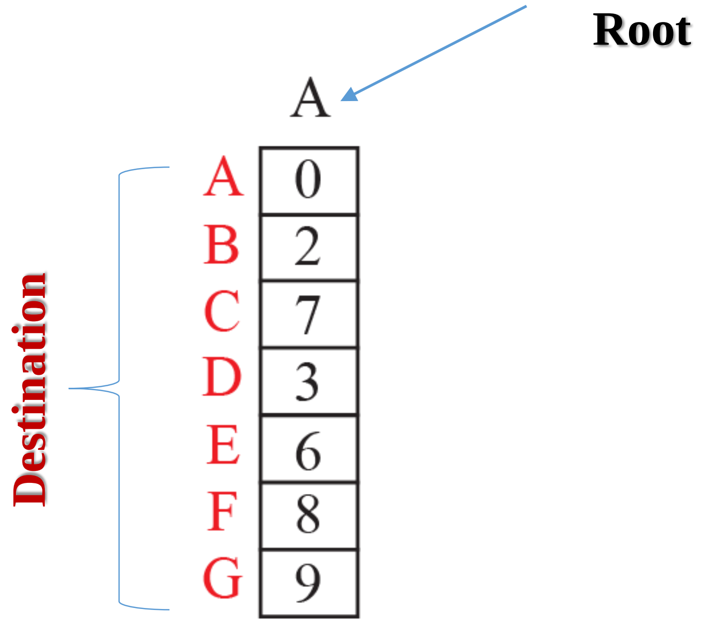
2) **Dijkstra's Algorithm:** It is used in **Link State Routing Protocols**.
    - Link state information is flooded to all routers.
    - Uses centralized approach

Note: Both computes the shortest path (i.e., least cost) from a single node vertex.

What is Distance Vector?



a. Tree for node A

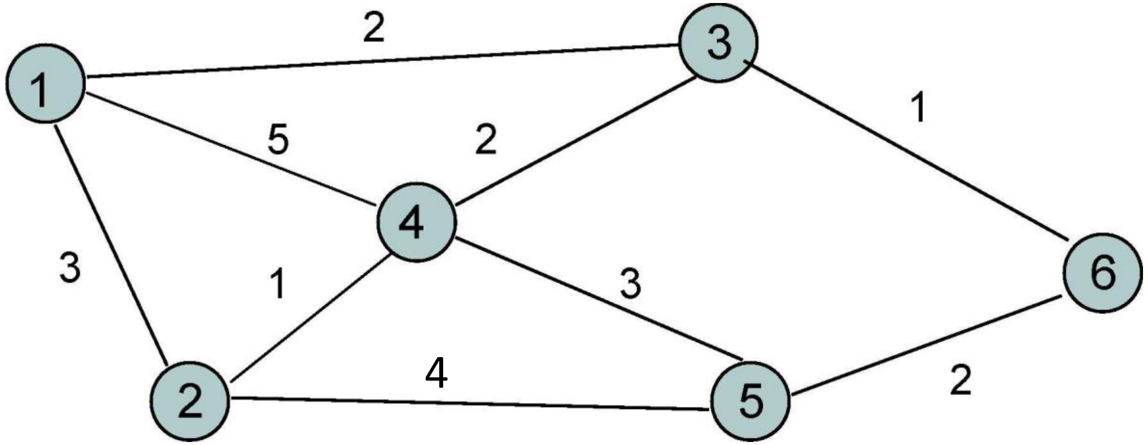


Root
Destination

	A
A	0
B	2
C	7
D	3
E	6
F	8
G	9

b. Distance vector for node A

Bellman-Ford (Distance Vector Protocols)
The simplest approach…….



Prob: *Find the shortest path to 6.*

Iteration	Node 1	Node 2	Node 3	Node 4	Node 5
Initial	(-1, ∞)	(-1, ∞)	(-1, ∞)	(-1, ∞)	(-1, ∞)

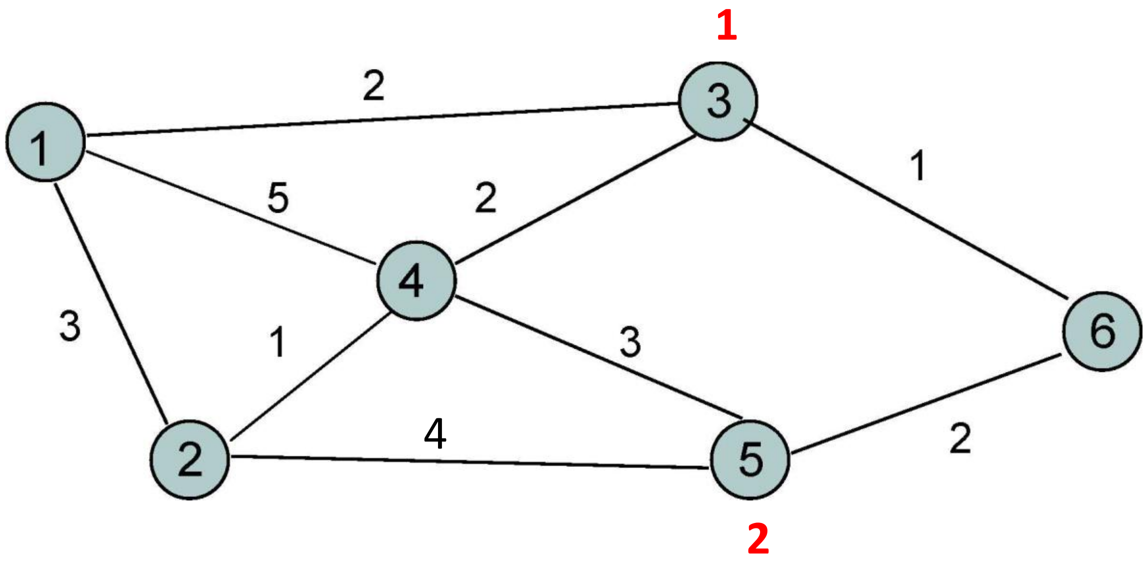
Next node from Node 6

Distance of node 6 from Node 5

Blind Rule: The only question to be asked to yourself at every node Nx for given destination N:

Can we reach to node N from node Nx *using the nodes which have been previously defined* ?????

Bellman-Ford (Distance Vector Protocols)
The simplest approach…….



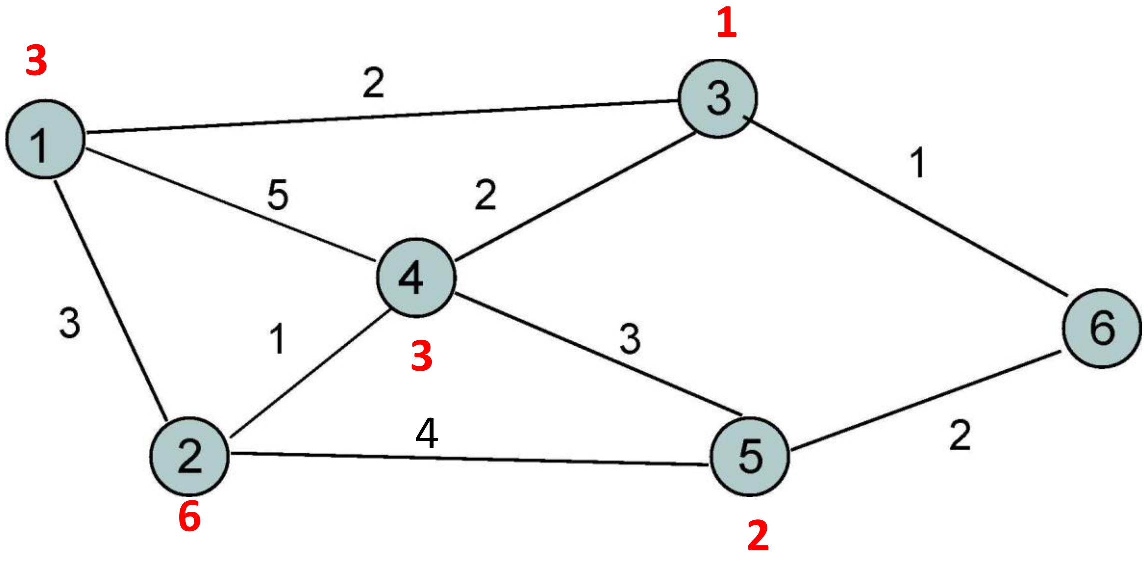
***Prob:** Find the shortest path to 6.*

Iteration	Node 1	Node 2	Node 3	Node 4	Node 5
Initial	(-1, ∞)	(-1, ∞)	(-1, ∞)	(-1, ∞)	(-1, ∞)
1^{st}	(-1, ∞)	(-1, ∞)	(6, 1)	(-1, ∞)	(6, 2)

Next node from Node 6

Distance of node 6 from Node 5

Bellman-Ford (Distance Vector Protocols)
The simplest approach…….



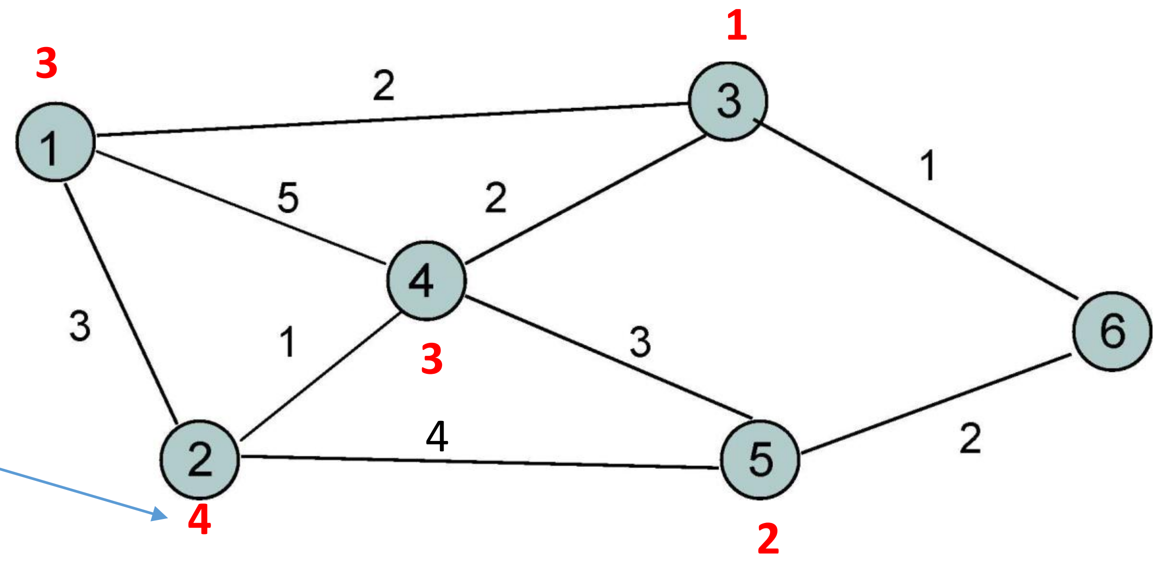
Prob: *Find the shortest path to 6.*

Iteration	Node 1	Node 2	Node 3	Node 4	Node 5
Initial	(-1, ∞)	(-1, ∞)	(-1, ∞)	(-1, ∞)	(-1, ∞)
1st	(-1, ∞)	(-1, ∞)	(6, 1)	(-1, ∞)	(6, 2)
2nd	(3,3)	(5,6)	(6, 1)	(3, 3)	(6, 2)

Next node from Node 6

Distance of node 6 from Node 5

Bellman-Ford (Distance Vector Protocols)
The simplest approach…….



Updated

Prob: *Find the shortest path to 6.*

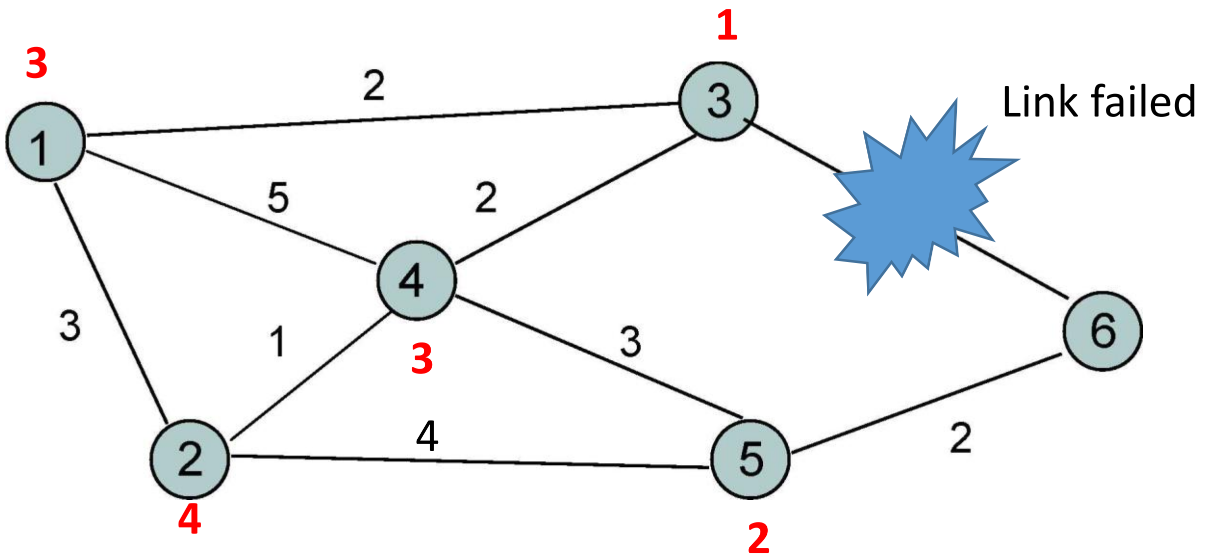
Next node from Node 6

Distance of node 6 from Node 5

Iteration	Node 1	Node 2	Node 3	Node 4	Node 5
Initial	(-1, ∞)	(-1, ∞)	(-1, ∞)	(-1, ∞)	(-1, ∞)
1st	(-1, ∞)	(-1, ∞)	(6, 1)	(-1, ∞)	(6, 2)
2nd	(3,3)	(5,6)	(6, 1)	(3, 3)	(6, 2)
3rd	(3,3)	(4, 4)	(6, 1)	(3, 3)	(6, 2)

Have you understood

Solve This……



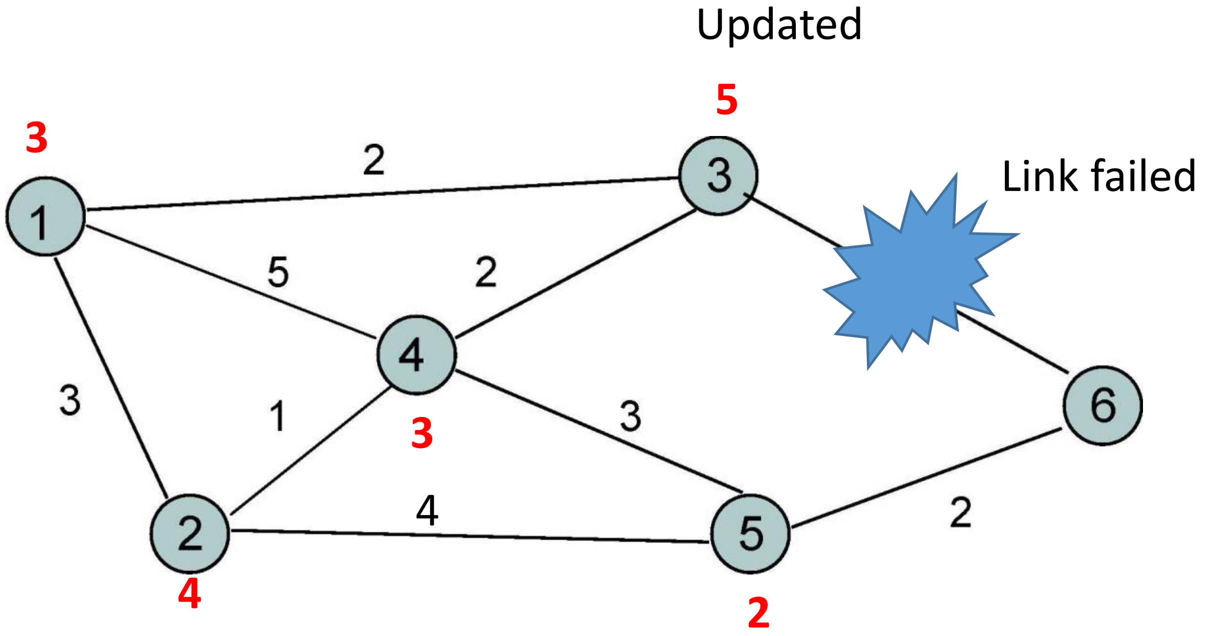
Prob: *Find the shortest path to 6.*

Iteration	Node 1	Node 2	Node 3	Node 4	Node 5
Initial	(3, 3)	(4, 4)	(6, 1)	(3, 3)	(6, 2)

Next node from Node 6

Distance of node 6 from Node 5

Solve This……



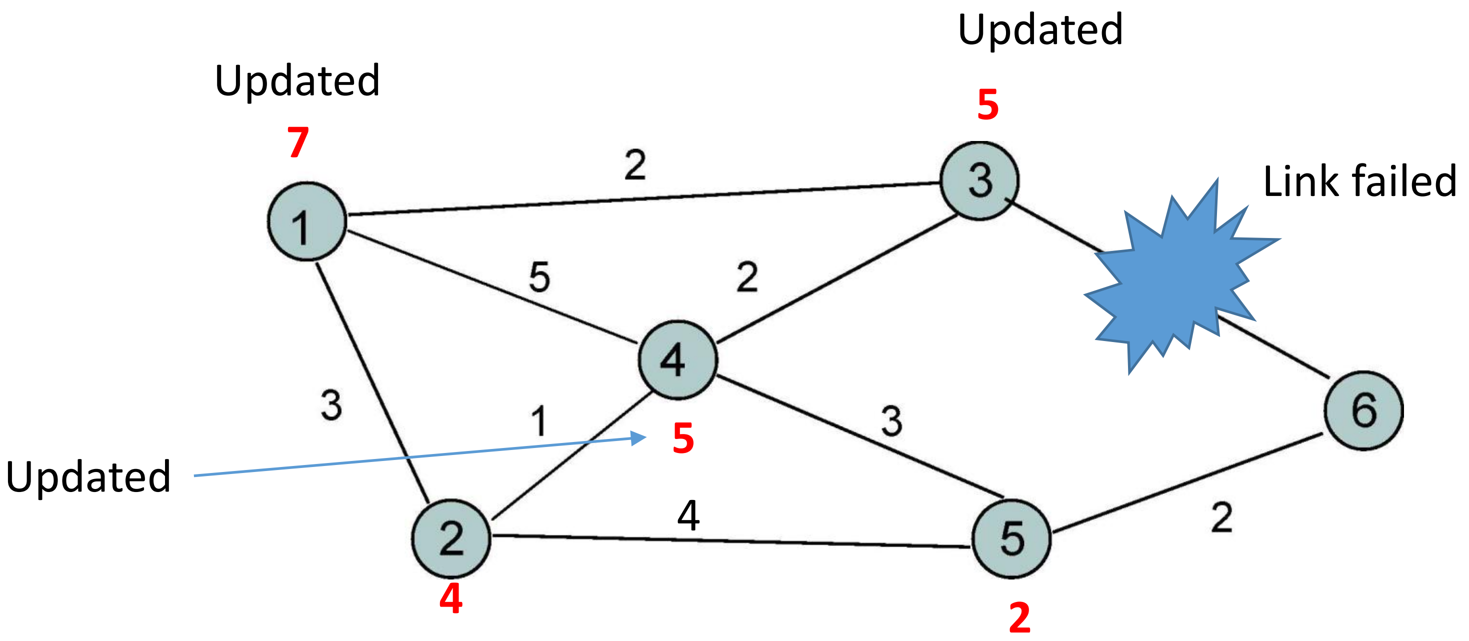
Prob: *Find the shortest path to 6.*

Iteration	Node 1	Node 2	Node 3	Node 4	Node 5
Initial	(3, 3)	(4, 4)	(6, 1)	(3, 3)	(6, 2)
1st	(3, 3)	(4, 4)	(4, 5)	(3, 3)	(6, 2)

Next node from Node 6

Distance of node 6 from Node 5

Solve This……



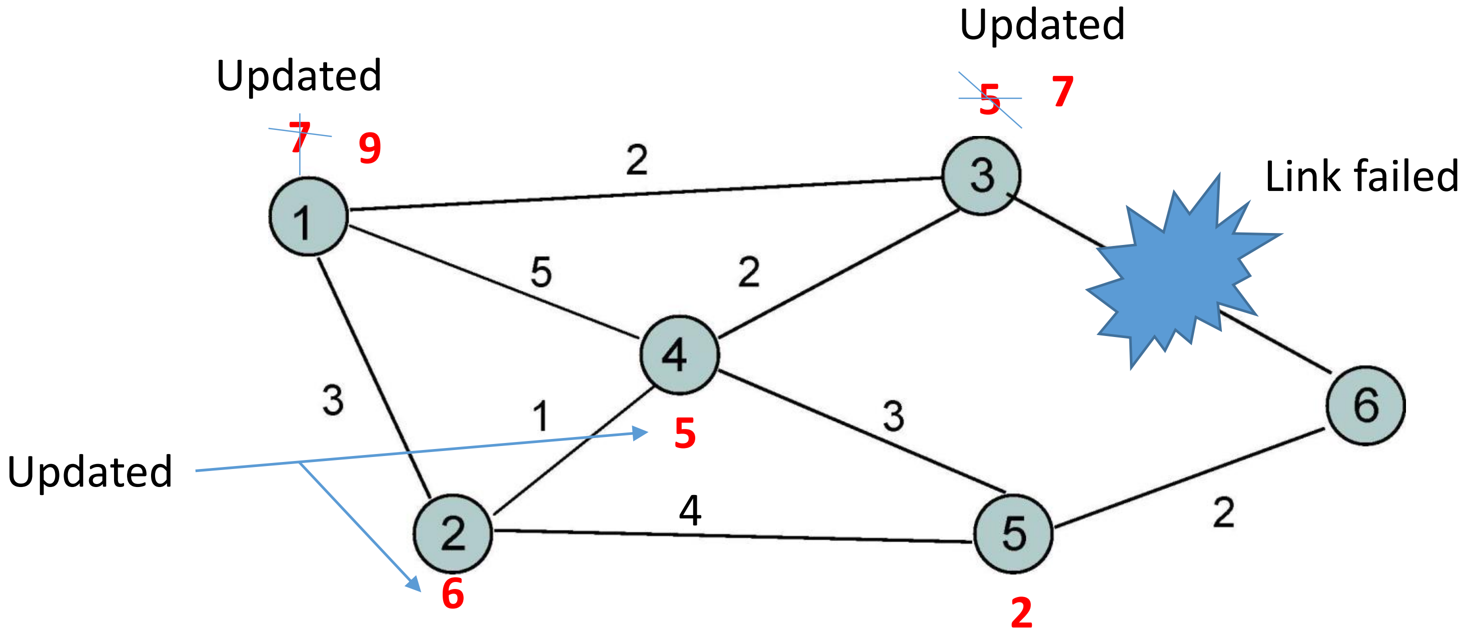
Prob: *Find the shortest path to 6.*

Next node from Node 6

Distance of node 6 from Node 5

Iteration	Node 1	Node 2	Node 3	Node 4	Node 5
Initial	(3, 3)	(4, 4)	(6, 1)	(3, 3)	(6, 2)
1st	(3, 3)	(4, 4)	(4, 5)	(3, 3)	(6, 2)
2nd	(3, 7)	(4, 4)	(4, 5)	(5, 5)	(6, 2)

Solve This……



Prob: *Find the shortest path to 6.*

Next node from Node 6

Distance of node 6 from Node 5

Iteration	Node 1	Node 2	Node 3	Node 4	Node 5
Initial	(3, 3)	(4, 4)	(6, 1)	(3, 3)	(6, 2)
1st	(3, 3)	(4, 4)	(4, 5)	(3, 3)	(6, 2)
2nd	(3, 7)	(4, 4)	(4, 5)	(5, 5)	(6, 2)
3rd	(3, 7)	(5, 6)	(4, 7)	(5, 5)	(6, 2)
4th	(2, 9)	(5, 6)	(4, 7)	(5, 5)	(6, 2)

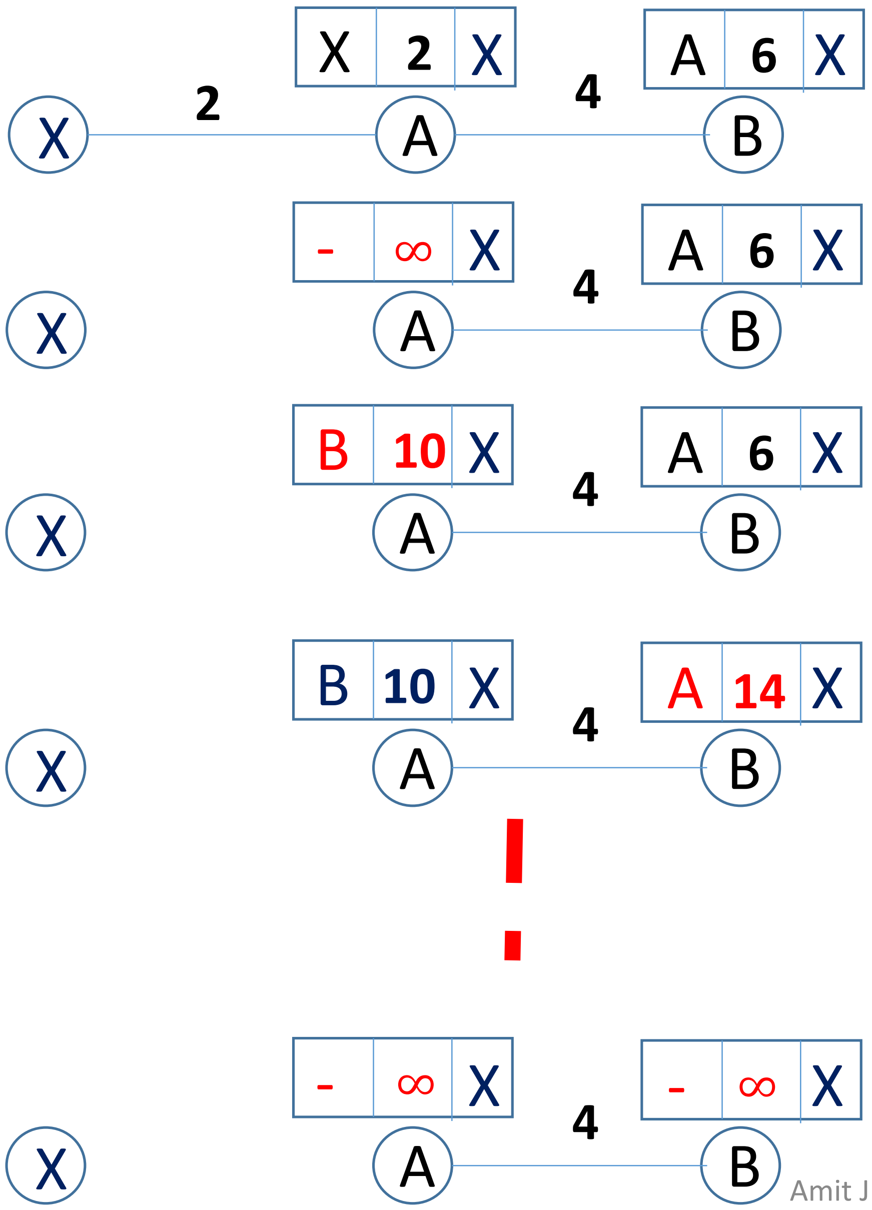
Some Key Points of Distance Vector(DV)

Note: DV uses hop counts to assign values to edges.

- **Advantage:** This algorithm does not require all of the nodes to operate in synchronous manner.
- **Disadvantage:** It does not scale well.

- Changes in network topology are not reflected quickly, since updates are spread node-by-node.
- May exists a loop and thus count to infinity, (if node failed is not traceable by other set of nodes).
- Thus, Can't be used in larger networks which require more robust and precise algorithm.

Counting to Infinity Problem in DV



Before failure

After failure

After A receives update from B

After B receives update from A

Finally……….

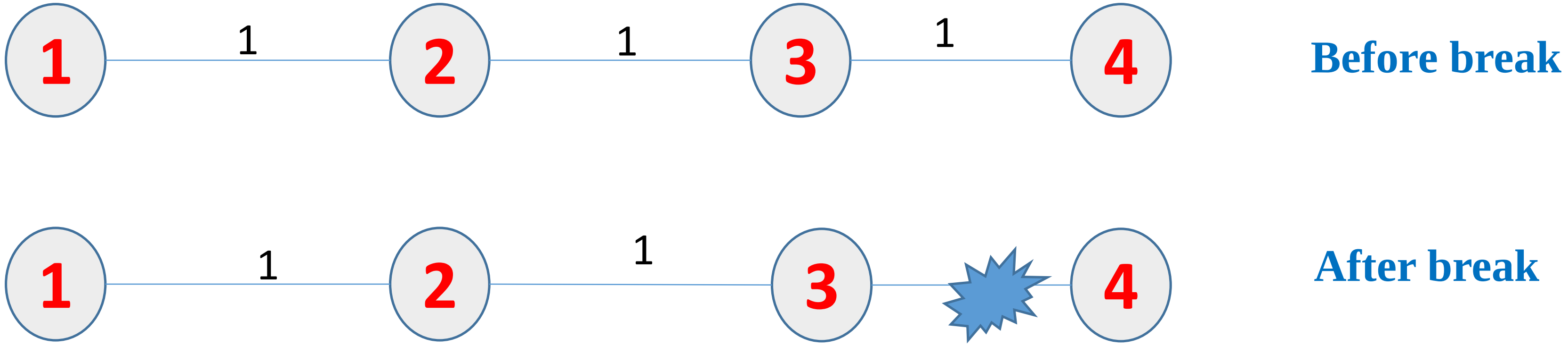
Iteration	Node A	Node B
Before Break	(X, 2)	(A, 6)
After Break		
1st	(B, 10)	(A, 6)
2nd	(B, 10)	(A, 14)
3rd	(B, 18)	(A, 14)
4th	(B, 18)	(A, 22)
. . .	. . .	. . .

Remedies to count to Infinity Problem

- **Note:** This problem does not exist if link failed node shares its routing information before the other nodes. In our scenario, if A shares its information to node B before, node B shares its information to A, this problem will not exist.
- Remedies to the problem:

1. ***Defining infinity:*** redefine infinity to a smaller number such as 100. But in this case, it can not be useful for a larger network.
2. ***Split Horizon:*** Share only part of the routing table instead of all to each interface. If, according to its table, node B thinks that the optimum route to reach X is via A, it does not need to advertise this piece of information to A; the information has come from A (A already knows). Taking information from node A, modifying it, and sending it back to node A creates the confusion. In our scenario, node B eliminates the last line of its routing table before it sends it to A. In this case, node A keeps the value of infinity as the distance to X. Later when node A sends its routing table to B, node B also corrects its routing table. The system becomes stable after the first update: both node A and B know that X is not reachable.
3. ***Split Horizon and Poison Reverse:*** Using the split horizon strategy has one draw back. Normally, the distance vector protocol uses a timer, and if there is no news about a route, the node deletes the route from its table. When node B in the previous scenario eliminates the route to X from its advertisement to A, node A can not guess that this is due to the split horizon strategy (the source of information was A) or because B has not received any news about X recently. The split horizon strategy can be combined with the poison reverse strategy. Node B can still advertise the value for X, but if the source of information is A, it can replace the distance with infinity as a warning: " Do not use this value; what I know about this route comes from you."

Practice: Observe Counting to Infinity Problem for the given Network



Destination is node 4

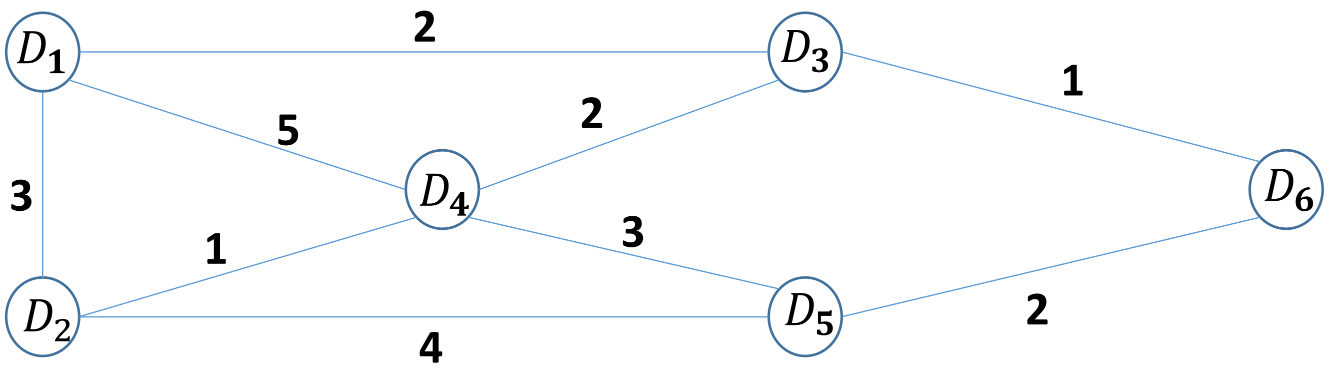
Update	Node 1	Node 2	Node 3
Before Break	(2, 3)	(3, 2)	(4, 1)
After Break	(2, 3)	(3, 2)	(2, 3)
1st	(2, 3)	(3, 4)	(2, 3)
2nd	(2, 5)	(3, 4)	(2, 5)
3rd	(2, 5)	(3, 6)	(2, 5)
4th	(2, 7)	(3, 6)	(2, 7)
…	…	…	…

Dijkstra Algorithm (Link State Protocol)

- Some key-points to remember:
  - Initially, choose the source node and make distance from source node to itself as 0, and from source to all other nodes infinity.
  - In the first iteration, select the source node in the visited queue and find the distance from source node to all other nodes which is directly connected to it.
  - In the second iteration, choose the node with **minimum** cost from the source node as visited node; and find the minimum distance from the source node to all other nodes **using the visited node**.
  - Repeat second step until all nodes become visited nodes.

Note: If a node can not be reached through from the source using all of the visited nodes then distance of the node from source will be infinite.

Execution of Dijkstra's Algorithm



Find the shortest path from D1

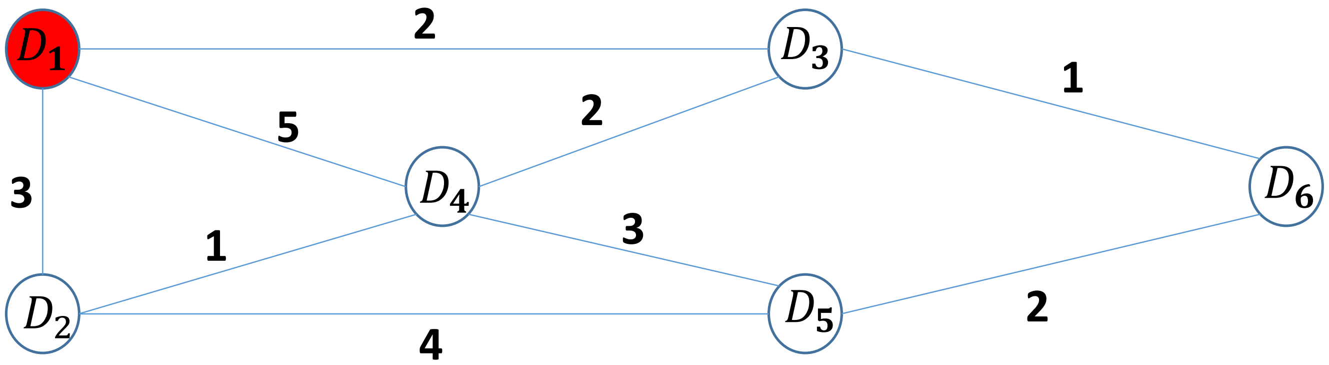
Iteration	Visited Nodes	D_2	D_3	D_4	D_5	D_6
Initial	{-}	∞	∞	∞	∞	∞

Blind Rule: The only question to be asked to yourself at every node Nx:

Can we reach to node Nx from the source node N *using the nodes in the visited queue* ?????

Note: The node having minimum cost will become a member of visited queue in every iteration.

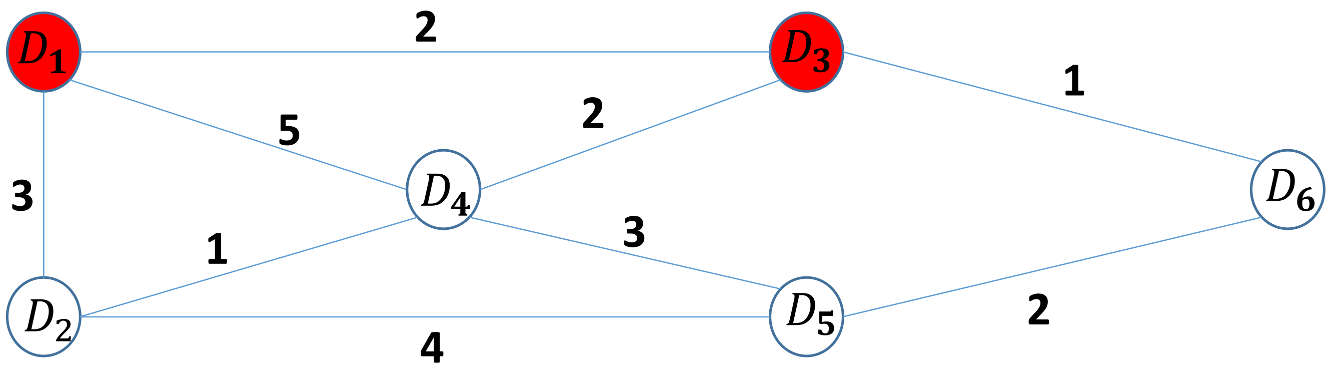
Execution of Dijkstra's Algorithm



Find the shortest path from D1

Iteration	Visited Nodes	D_2	D_3	D_4	D_5	D_6
Initial	{-}	∞	∞	∞	∞	∞
1st	$\{D_1\}$	**3** $(D_1\text{-}D_2)$	**2** $(D_1\text{-}D_3)$	**5** $(D_1\text{-}D_4)$	∞	∞

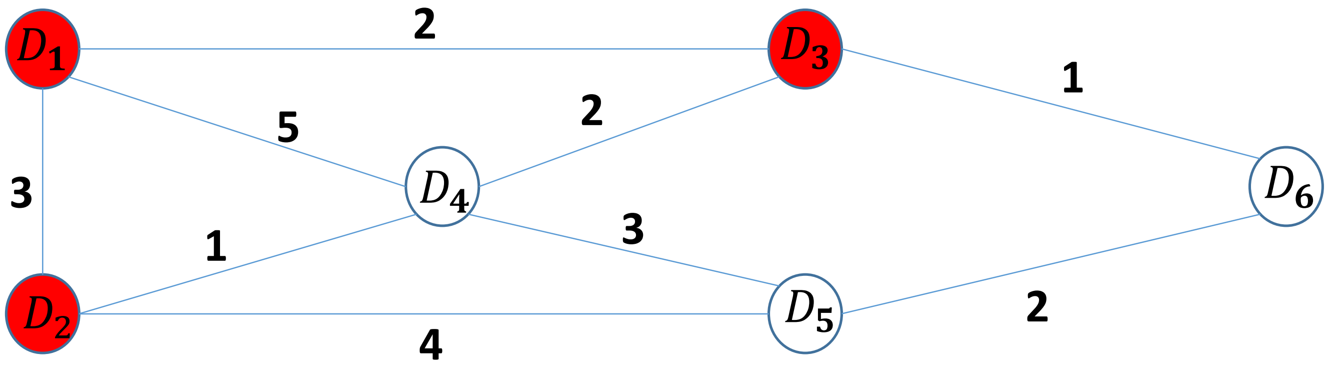
Execution of Dijkstra’s Algorithm



Find the shortest path from D1

Iteration	Visited Nodes	D_2	D_3	D_4	D_5	D_6
Initial	{-}	∞	∞	∞	∞	∞
1st	$\{D_1\}$	**3** (D_1-D_2)	**2** (D_1-D_3)	**5** (D_1-D_4)	∞	∞
2nd	$\{D_1, D_3\}$	**3** (D_1-D_2)	**2** (D_1-D_3)	**4** $(D_1-D_3-D_4)$	∞	**3** $(D_1-D_3-D_6)$

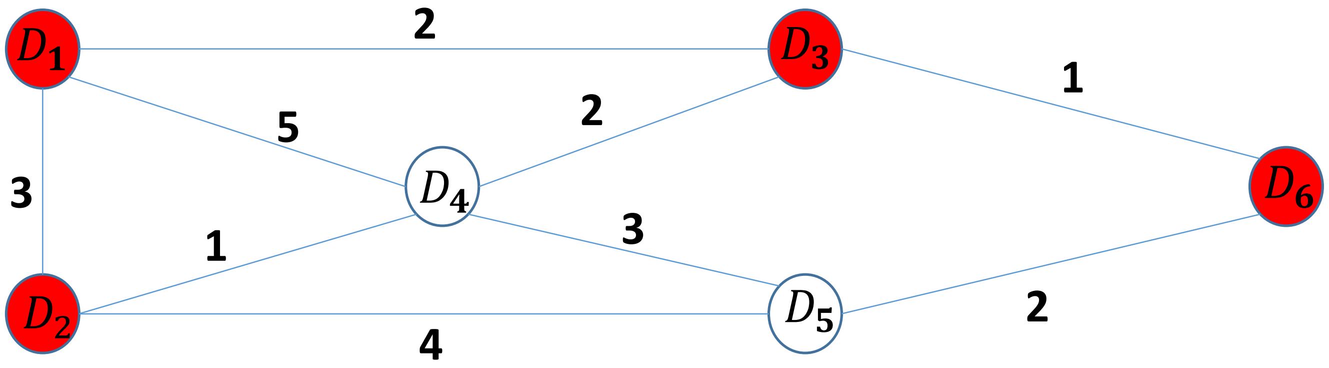
Execution of Dijkstra’s Algorithm



Find the shortest path from D1

Iteration	Visited Nodes	D_2	D_3	D_4	D_5	D_6
Initial	{-}	∞	∞	∞	∞	∞
1st	$\{D_1\}$	**3** $(D_1\text{-}D_2)$	**2** $(D_1\text{-}D_3)$	**5** $(D_1\text{-}D_4)$	∞	∞
2nd	$\{D_1,\ D_3\}$	**3** $(D_1\text{-}D_2)$	**2** $(D_1\text{-}D_3)$	**4** $(D_1\text{-}D_3-D_4)$	∞	**3** $(D_1\text{-}D_3-D_6)$
3rd	$\{D_1,D_2,D_3\}$	**3** $(D_1\text{-}D_2)$	**2** $(D_1\text{-}D_3)$	**4** $(D_1\text{-}D_3-D_4)$	**7** $(D_1\text{-}D_2-D_5)$	**3** $(D_1\text{-}D_3-D_6)$

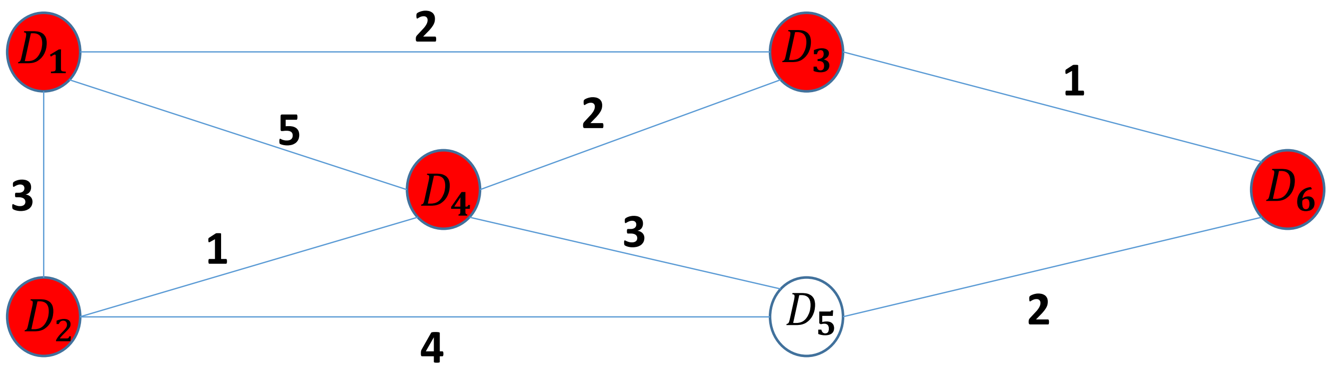
Execution of Dijkstra’s Algorithm



Find the shortest path from D1

Iteration	Visited Nodes	D_2	D_3	D_4	D_5	D_6
Initial	{-}	∞	∞	∞	∞	∞
1st	$\{D_1\}$	**3** $(D_1\text{-}D_2)$	**2** $(D_1\text{-}D_3)$	**5** $(D_1\text{-}D_4)$	∞	∞
2nd	$\{D_1,\ D_3\}$	**3** $(D_1\text{-}D_2)$	**2** $(D_1\text{-}D_3)$	**4** $(D_1\text{-}D_3-D_4)$	∞	**3** $(D_1\text{-}D_3-D_6)$
3rd	$\{D_1,D_2,D_3\}$	**3** $(D_1\text{-}D_2)$	**2** $(D_1\text{-}D_3)$	**4** $(D_1\text{-}D_3-D_4)$	**7** $(D_1\text{-}D_2-D_5)$	**3** $(D_1\text{-}D_3-D_6)$
4th	$\{D_1,D_2,D_3,\ D_6\}$	**3** $(D_1\text{-}D_2)$	**2** $(D_1\text{-}D_3)$	**4** $(D_1\text{-}D_3-D_4)$	**5** $(D_1-D_3-D_6-D_5)$	**3** $(D_1\text{-}D_3-D_6)$

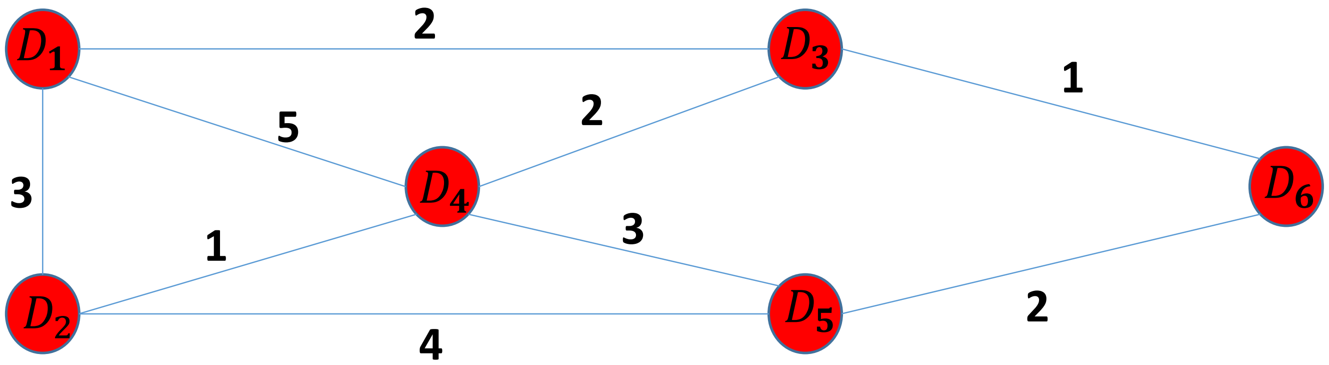
Execution of Dijkstra’s Algorithm



Find the shortest path from D1

Iteration	Visited Nodes	D_2	D_3	D_4	D_5	D_6
Initial	{-}	∞	∞	∞	∞	∞
1st	$\{D_1\}$	3 $(D_1\text{-}D_2)$	2 $(D_1\text{-}D_3)$	5 $(D_1\text{-}D_4)$	∞	∞
2nd	$\{D_1,\ D_3\}$	3 $(D_1\text{-}D_2)$	2 $(D_1\text{-}D_3)$	4 $(D_1\text{-}D_3\text{−}D_4)$	∞	3 $(D_1\text{-}D_3\text{−}D_6)$
3rd	$\{D_1,D_2,D_3\}$	3 $(D_1\text{-}D_2)$	2 $(D_1\text{-}D_3)$	4 $(D_1\text{-}D_3\text{−}D_4)$	7 $(D_1\text{-}D_2\text{−}D_5)$	3 $(D_1\text{-}D_3\text{−}D_6)$
4th	$\{D_1,D_2,D_3,\ D_6\}$	3 $(D_1\text{-}D_2)$	2 $(D_1\text{-}D_3)$	4 $(D_1\text{-}D_3\text{−}D_4)$	5 $(D_1-D_3\ -D_6\ -D_5)$	3 $(D_1\text{-}D_3\text{−}D_6)$
5th	$\{D_1,D_2,D_3,D_4,\ D_6\}$	3 $(D_1\text{-}D_2)$	2 $(D_1\text{-}D_3)$	4 $(D_1\text{-}D_3\text{−}D_4)$	5 $(D_1-D_3\ -D_6\ -D_5)$	3 $(D_1\text{-}D_3\text{−}D_6)$

Execution of Dijkstra’s Algorithm



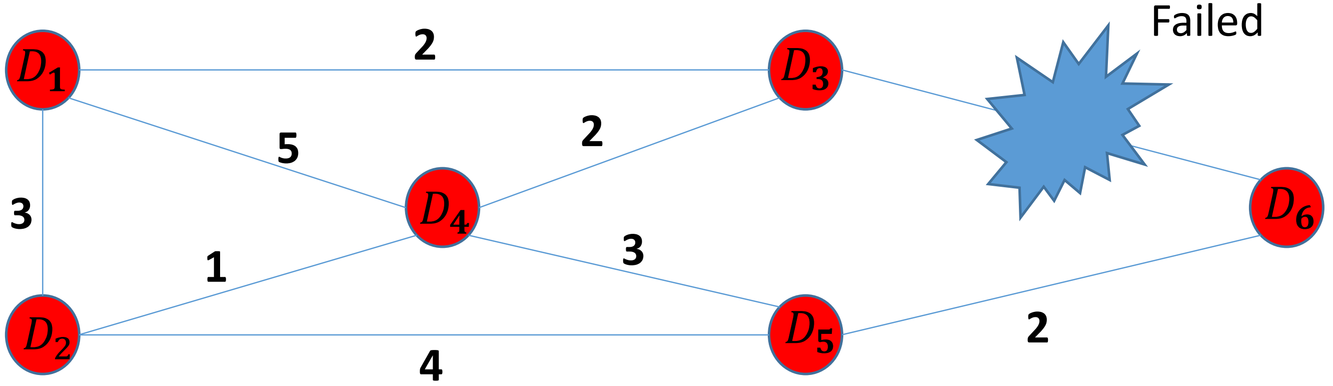
Find the shortest path from D1

Iteration	Visited Nodes	D_2	D_3	D_4	D_5	D_6
Initial	{-}	∞	∞	∞	∞	∞
1st	$\{D_1\}$	3 $(D_1\text{-}D_2)$	2 $(D_1\text{-}D_3)$	5 $(D_1\text{-}D_4)$	∞	∞
2nd	$\{D_1,\ D_3\}$	3 $(D_1\text{-}D_2)$	2 $(D_1\text{-}D_3)$	4 $(D_1\text{-}D_3 - D_4)$	∞	3 $(D_1\text{-}D_3 - D_6)$
3rd	$\{D_1, D_2, D_3\}$	3 $(D_1\text{-}D_2)$	2 $(D_1\text{-}D_3)$	4 $(D_1\text{-}D_3 - D_4)$	7 $(D_1\text{-}D_2 - D_5)$	3 $(D_1\text{-}D_3 - D_6)$
4th	$\{D_1, D_2, D_3,\ D_6\}$	3 $(D_1\text{-}D_2)$	2 $(D_1\text{-}D_3)$	4 $(D_1\text{-}D_3 - D_4)$	5 $(D_1\text{-}D_2 - D_5)$	3 $(D_1\text{-}D_3 - D_6)$
5th	$\{D_1, D_2, D_3, D_4,\ D_6\}$	3 $(D_1\text{-}D_2)$	2 $(D_1\text{-}D_3)$	4 $(D_1\text{-}D_3 - D_4)$	5 $(D_1 - D_3 - D_6 - D_5)$	3 $(D_1\text{-}D_3 - D_6)$
6th	$\{D_1, D_2, D_3, D_4, D_5, D_6\}$	3 $(D_1\text{-}D_2)$	2 $(D_1\text{-}D_3)$	4 $(D_1\text{-}D_3 - D_4)$	5 $(D_1 - D_3 - D_6 - D_5)$	3 $(D_1\text{-}D_3 - D_6)$

Routing Table at Node D_1

Destination	Next Node	Cost
D_2	D_2	3
D_3	D_3	2
D_4	D_3	4
D_5	D_3	5
D_6	D_3	3

Reaction to Failure



Iteration	Visited Nodes	D_2	D_3	D_4	D_5	D_6
Initial	$\{D_1, D_2, D_3, D_4, D_5, D_6\}$	**3** $(D_1 - D_2)$	**2** $(D_1 - D_3)$	**4** $(D_1 - D_3 - D_4)$	**5** $(D_1 - D_3 - D_6 - D_5)$	**3** $(D_1 - D_3 - D_6)$
1st	$\{D_1, D_2, D_3, D_4, D_5, D_6\}$	**3** $(D_1\text{-}D_2)$	**2** $(D_1\text{-}D_3)$	**4** $(D_1 - D_3 - D_4)$	**7** $(D_1 - D_3 - D_4 - D_5)$	**9** $(D_1 - D_2 - D_5 - D_6)$

Updated quickly

Advantages and Disadvantages of Link State

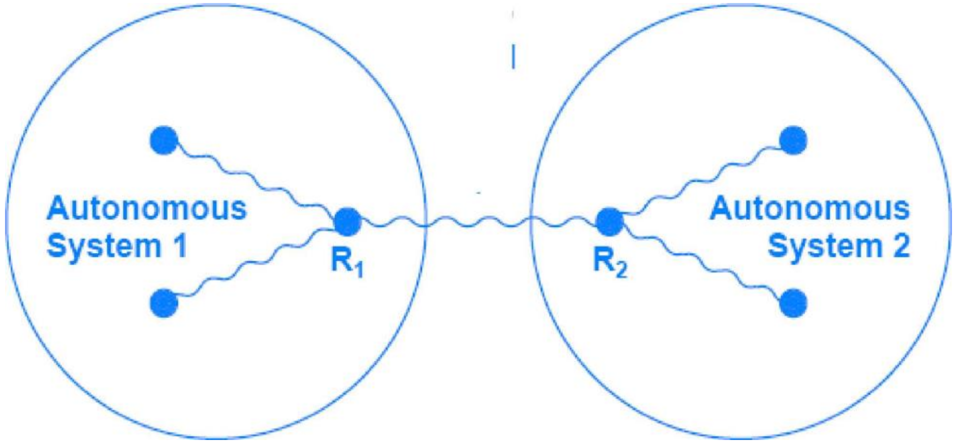
- **Advantages:**
  - Fast and loopless convergence (thus, it can be used in larger networks) as it uses centralized approach.
  - Support for precise metrics , and multiple metrics if necessary (throughput, delay, cost, reliability)
  - Supporting for multiple paths to a destination
    - Algorithm can be supported to find the two best paths to a destination
  - Scales better than distance vector
  - Each routers computes routes independently from original data (i.e. not relying on intermediate nodes)
- **Disadvantage:**
  - Computational load on routers and
  - Negative edge cost can not be handled

Key points to Remember

- **For Bellman Ford:**
  - **Blind Rule:** The only question to be asked to yourself at every node Nx for given destination N:
    - Can we reach to node N from node Nx *using the nodes which have been **previously defined*** ?????
- **For Dijkstra's Algorithm:**
  - **Blind Rule:** The only question to be asked to yourself at every node Nx for given Source N:
    - Can we reach to node Nx from the source node N *using the nodes in the **visited queue*** ?????
- **For Both:**
  - If you get two or more different paths of same cost then you can choose any one as per your wish.
  - Thus, if different students solve question **independently**, then path in their answer may vary but the cost of different nodes has to match for all students.

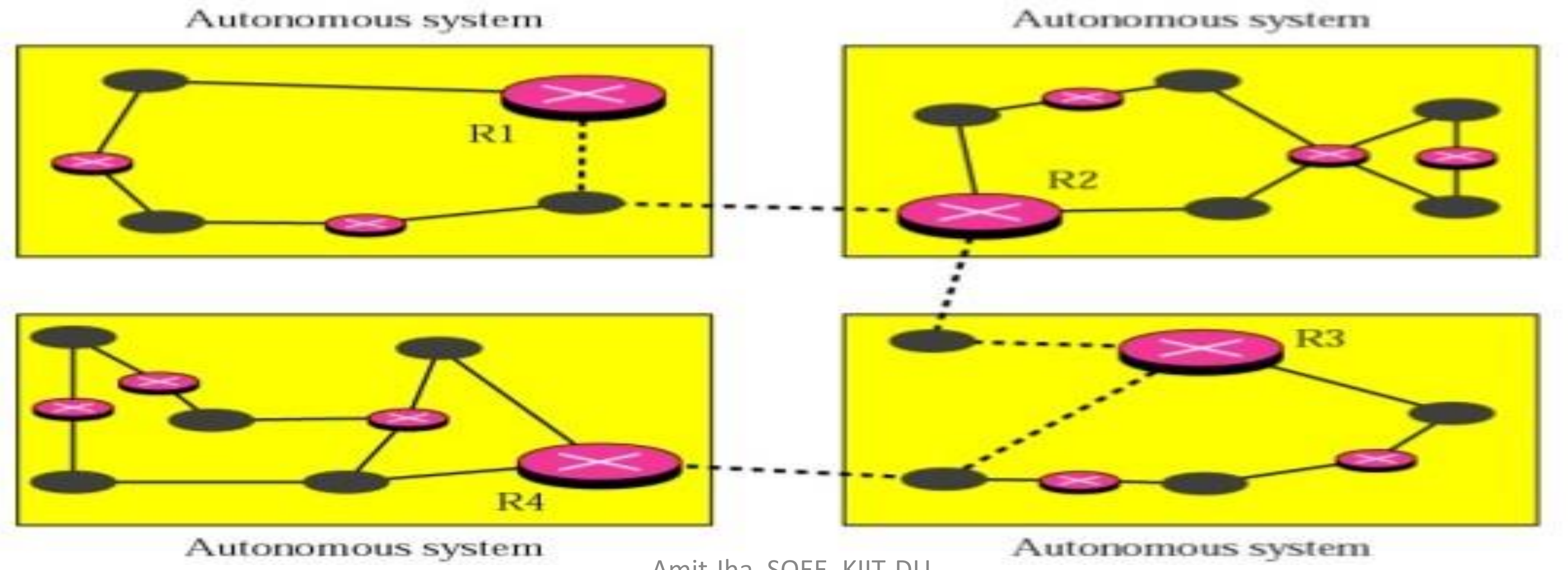
The concept of Autonomous System

- We cannot run an automatic routing protocol for the entire Global Internet, because of its complexity.
- So, networks and routers are owned by organizations and individuals.
- Within each, an administrative authority can guarantee that internal routes remain consistent and viable.
- For purposes of routing, a group of networks controlled by a single administrative authority is called ***an autonomous system (AS)*** and identified by an *autonomous system number.*



Implementation of Routing in Today’s Internet

- The Global Internet consists of AS interconnected with each other.
- *Intradomain Routing:* Routing within an *Autonomous System* (AS).
- *Interdomain Routing:* Routing between Autonomous Systems (ASs).



Routing Protocol

- A routing protocol is a combination of rules and procedures that lets routers in the internet to share whatever they know about the internet or their neighbourhood. The sharing of information allows a router in Bhubaneswar to know about the failure of a network in Greenwich.

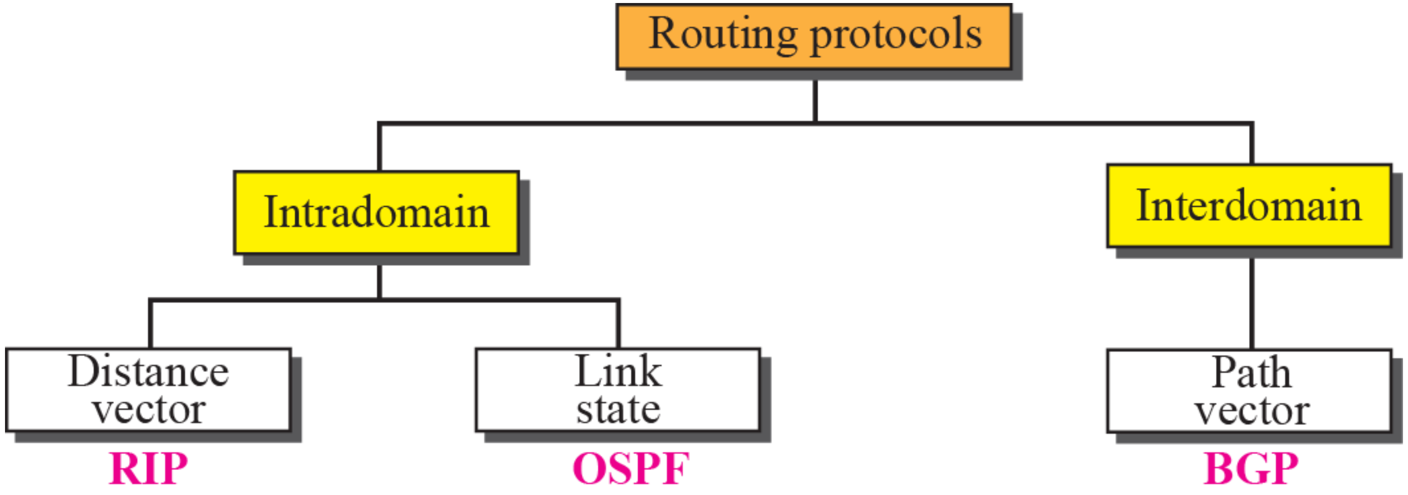


Fig: Popular Routing Protocols

Routing Information Protocol (RIP)

- RFC 1058
- It is an intradomain routing protocol used inside an autonomous system
- It implements the distance vector algorithm (like Bellman ford) in the Internet.
- Runs on UDP, port number 520
- Uses number of hops as metric
- Maximum limit is 15
  - Suitable for smaller networks (local area environment)
  - Value of 16 is reserved to represent infinity

RIP Operation: Implementation of DVP

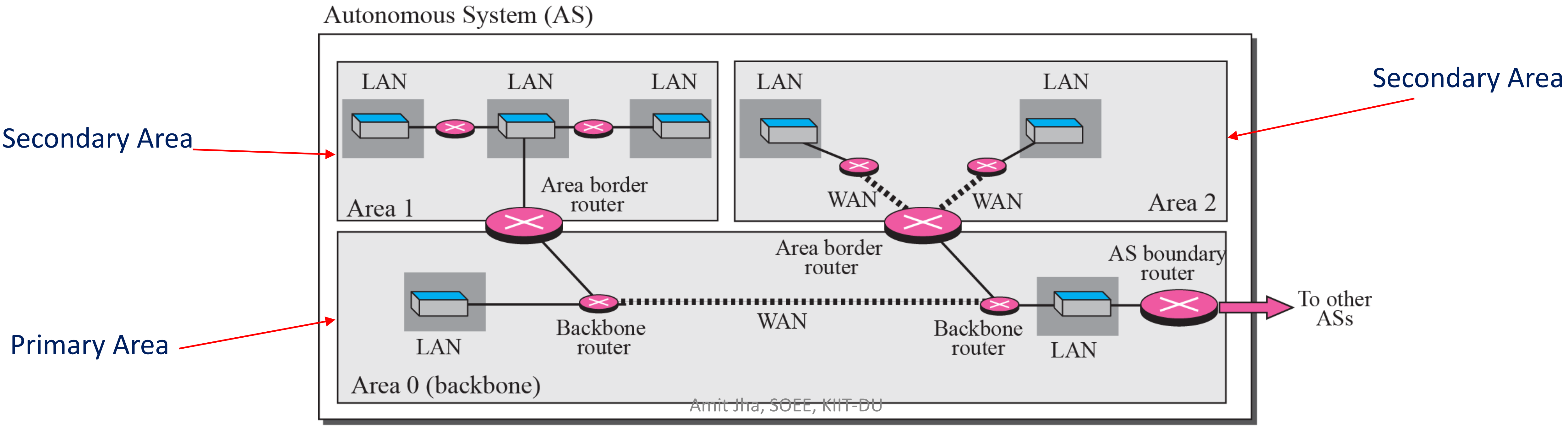
- Router sends update message to neighbours every 30 sec.
- A router expects to receive an update message from each of its neighbours within 180 sec in the worst case.
- If a router does not receive an update message from it neighbour X within 180 sec, it assumes that the link to X has failed and sets the minimum cost to 16 (i.e., infinity).
- Uses ***split horizon with poisoned reverse.***
- Two RIP packet types:
  1) Request to ask neighbours for distance vector table
  2) Response to advertise distance vector table

Open Shortest Path First (OSPF)

- RFC 2328 (v2)
- It is also an intradomain routing protocol i.e., used within an AS.
- Uses the link state algorithm (like Dijkstra’s algorithm)
- Fixes some of the deficiencies of RIP (Distance Vector algorithm)
- Its domain is also an autonomous system.
- **Area:** unlike RIP, it is used to handle **larger networks**, so its domain system is divided into smaller areas. An area is a collection of networks, hosts, and routers all contained within an autonomous system.

OSPF: Operation

- Routers inside an area flood the area with routing information.
- At the border of an area, special routers called area ***border routers*** summarize the information about the area and send it to the other areas.
- Among the areas inside an AS, there exists a ***backbone area*** (primary area) to which all other areas are connected (called secondary area).



Can we use Distance Vector (Bellman Ford) or Link State (Dijkstra) For Inter domain (between Autonomous Systems) routing ??

Can we use Distance Vector (Bellman Ford) or Link State (Dijkstra) For Inter domain (between Autonomous Systems) routing ??

No…. Because of the following issues !!!

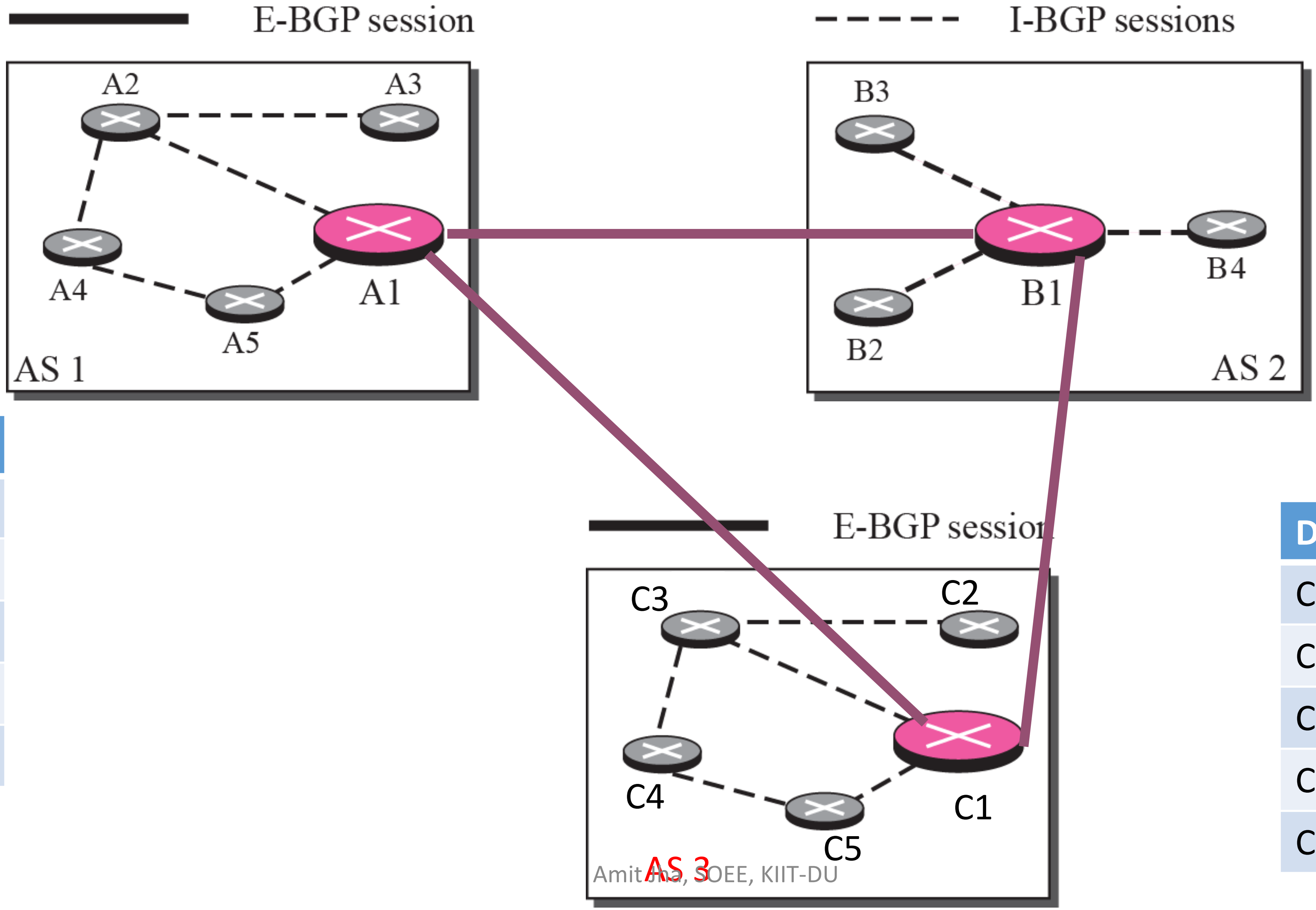
Both are not useful for a larger network

- **Bellman Ford** can not be used for a large network because; not well scalable, update is not quick, counting to infinity problem.
- **Dijkstra's Algorithm** is not useful because it floods complete routing table thus may create congestion in the network.
- Moreover, different AS may use different metric for their internal routing. For e.g., if routing is to be done between AS1 (using Link state) and AS2 (using Distance Vector), then AS1 can use delay as its metric whereas AS2 is **compelled** to use hope count as its metric.

Path Vector Routing: Introduction

- It works on the principle of Distance Vector routing.
- Each AS has one node which acts on the behalf of it called ***speaker node.***
- The speaker node in an AS creates a routing table and advertises it to speaker nodes in the neighbouring ASs.
- **Only speaker** node in each AS **can communicate** to each other.
- A speaker node **advertises the path**, **not the metric** of the nodes, in its autonomous system or other autonomous systems.

Path Vector Routing: Operation: Initially



Dest.	Path
B1	AS2
B2	AS2
B3	AS2
B4	AS2

B1 Table

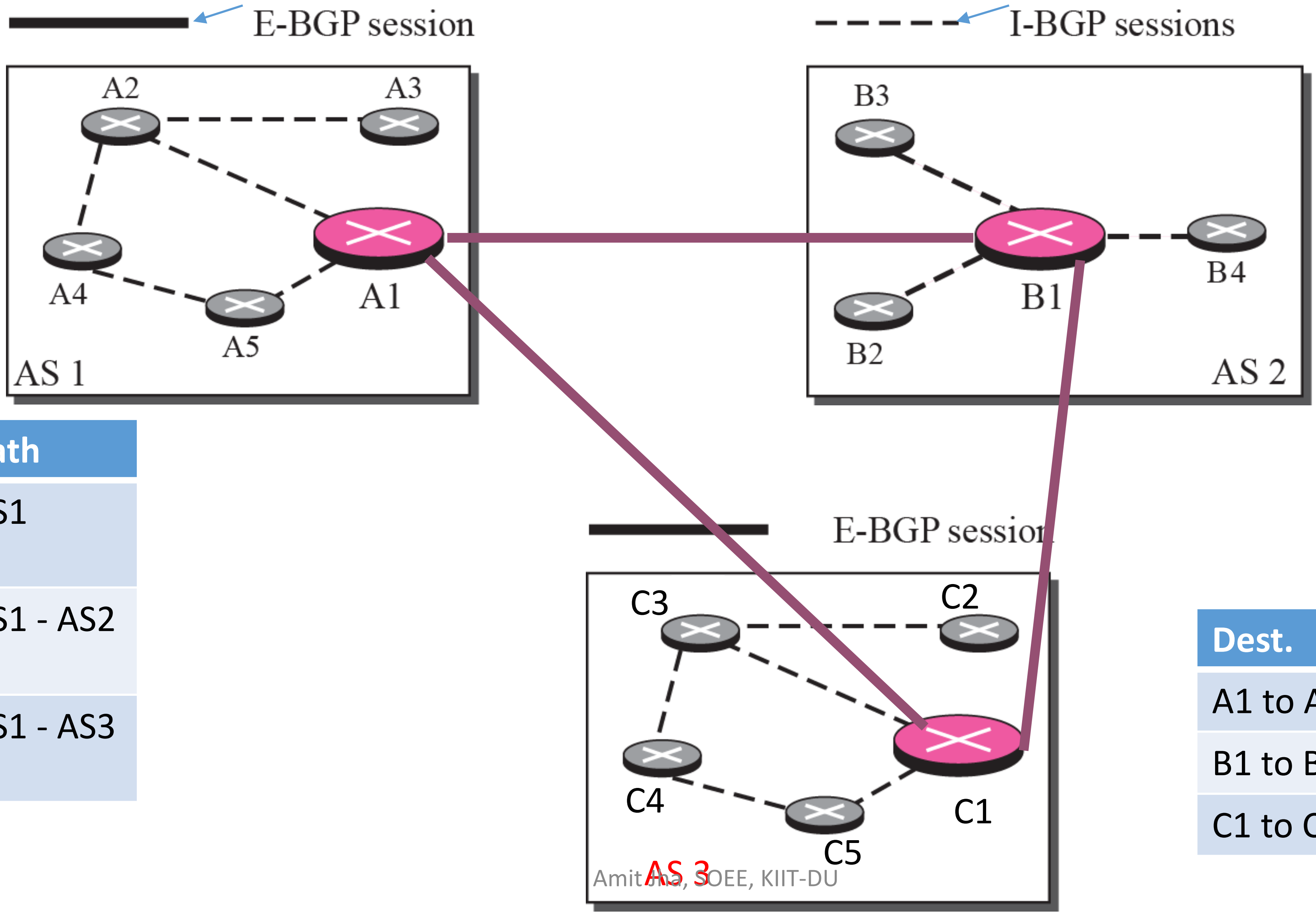
Dest.	Path
A1	AS1
A2	AS1
A3	AS1
A4	AS1
A5	AS1

A1 Table

Dest.	Path
C1	AS3
C2	AS3
C3	AS3
C4	AS3
C5	AS3

C1 Table

Path Vector Routing: Operation: After becoming Stable



Dest.	Path
A1 to A5	AS2 – AS1
B1 to B4	AS2
C1 to C5	AS2 – AS3

B1 Table

Dest.	Path
A1 to A5	AS1
B1 to B4	AS1 - AS2
C1 to C5	AS1 - AS3

A1 Table

Dest.	Path
A1 to A5	AS3 - AS1
B1 to B4	AS3 – AS2
C1 to C4	AS3

C1 Table

What is the optimum path in path vector routing ?

Ans: While deciding the optimum path we can not include the metric. This is because two different AS can use different metric as seen previously (DV using hop count whereas, link state using delay as metric). Thus, we choose optimum path as a path having less number of autonomous system.

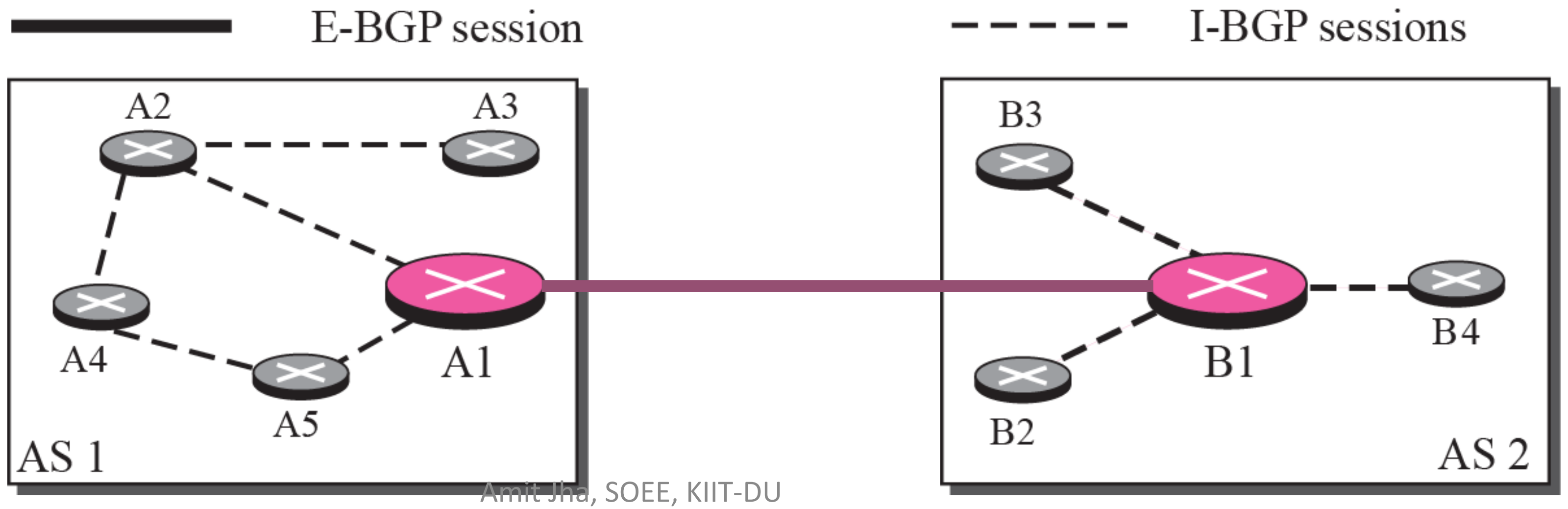
For e.g., if a sender from AS1 wants to send a packet to a receiver in AS3 then the optimum path will be AS1 - AS3, but not AS1 - AS2 - AS3.

Border Gateway Protocol (BGP)

- It is an inter domain routing protocol.
- It is based on path vector routing algorithm.
- BGP uses the services of TCP to create a reliable environment.
- BGP Session: A session is a connection that is established between two BGP routers only for the sake of exchanging routing information.
- There are two types of BGP Session:
    1. External BGP (E-BGP)
    2. Internal BGP (I-BGP)

Border Gateway Protocol (BGP): BGP Sessions

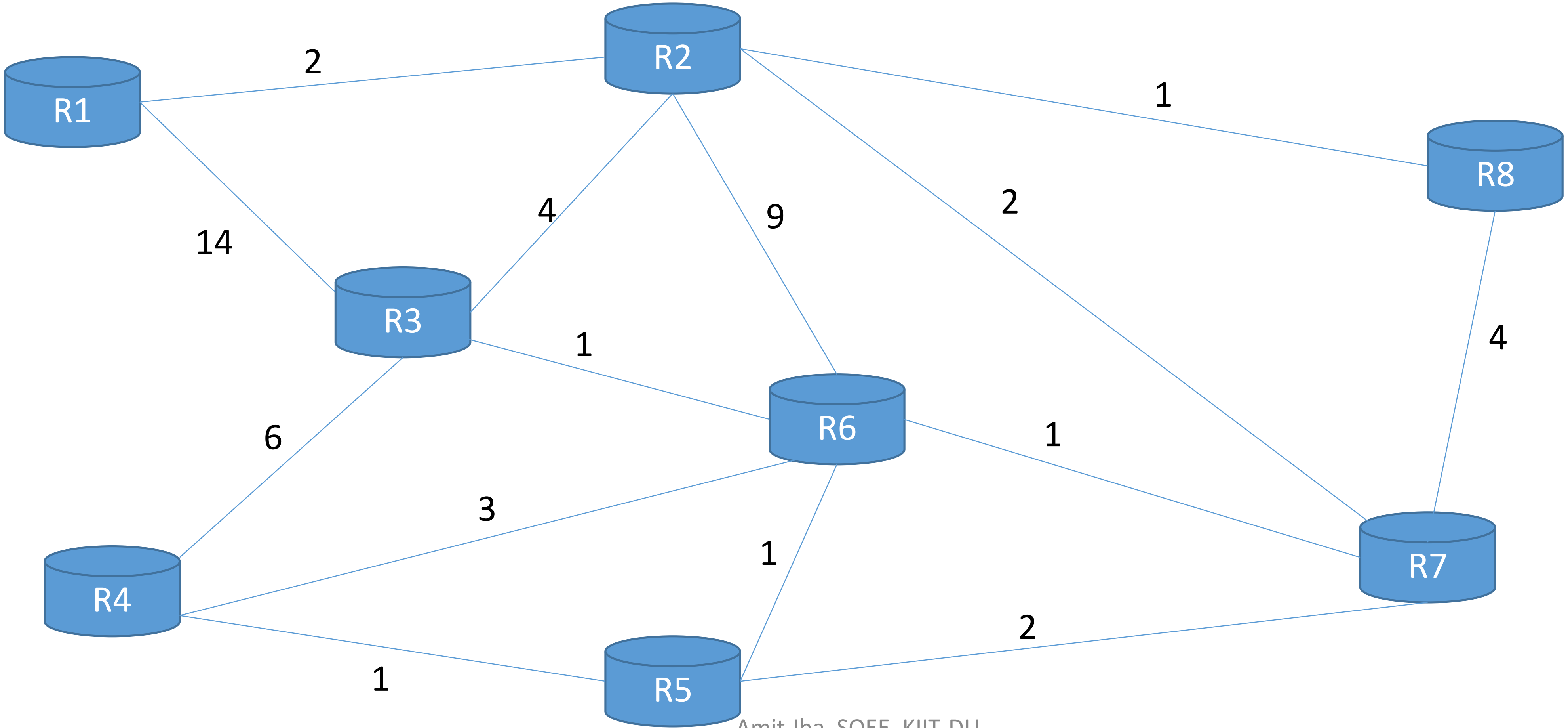
- External BGP (E-BGP): The E-BGP session is used to exchange information between two speaker nodes belonging to two different autonomous systems.
- Internal BGP (I-BGP): The I-BGP session, on the other hand, is used to exchange routing information between two routers inside an autonomous system.



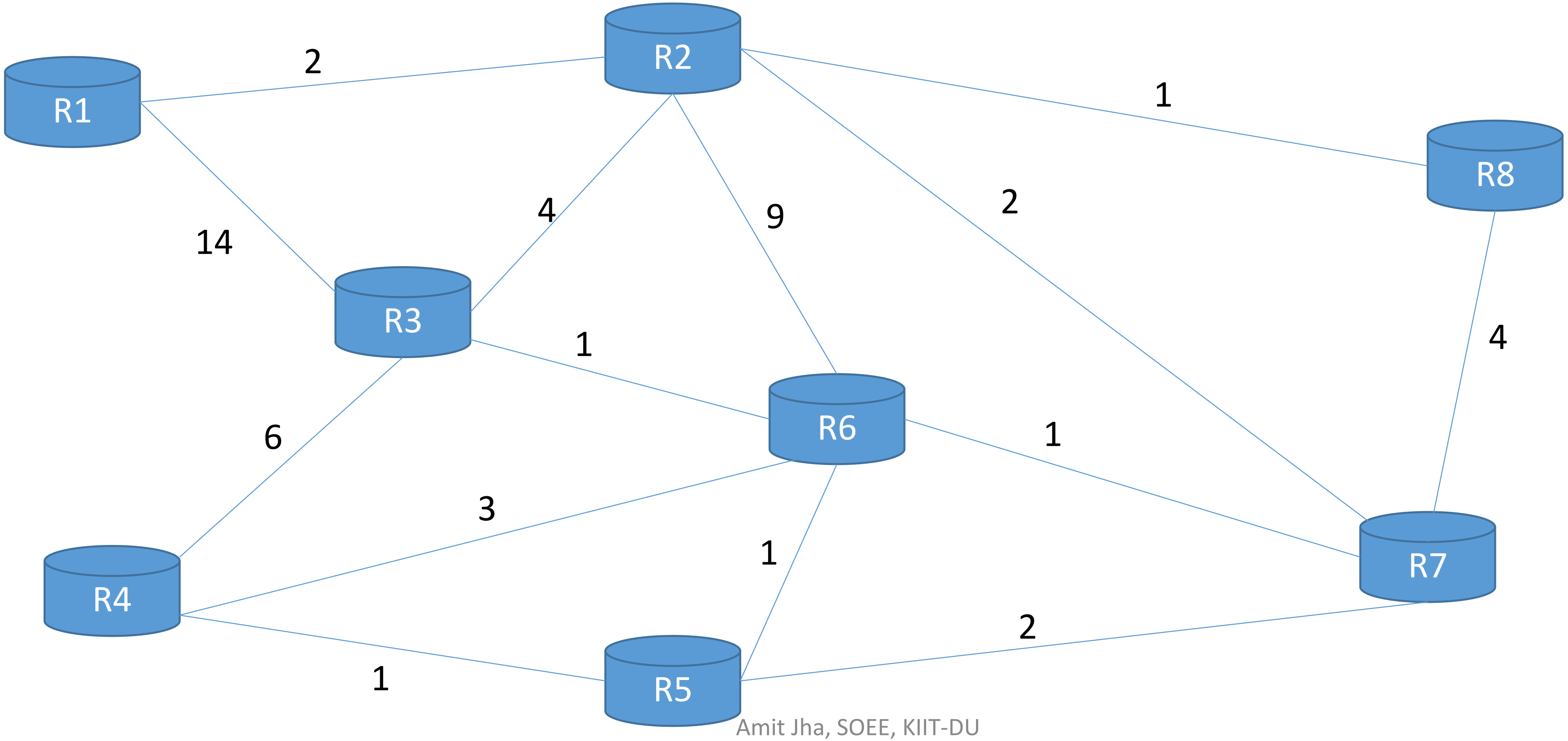
Practice Questions

- Compare the following:
  1. Bellman Ford Vs. Dijkstra’s Algorithm
  2. Distance Vector Vs. Link State
- Write short notes on the following:
  1. RIP
  2. OSPF
- Explain briefly what do you understand by BGP.

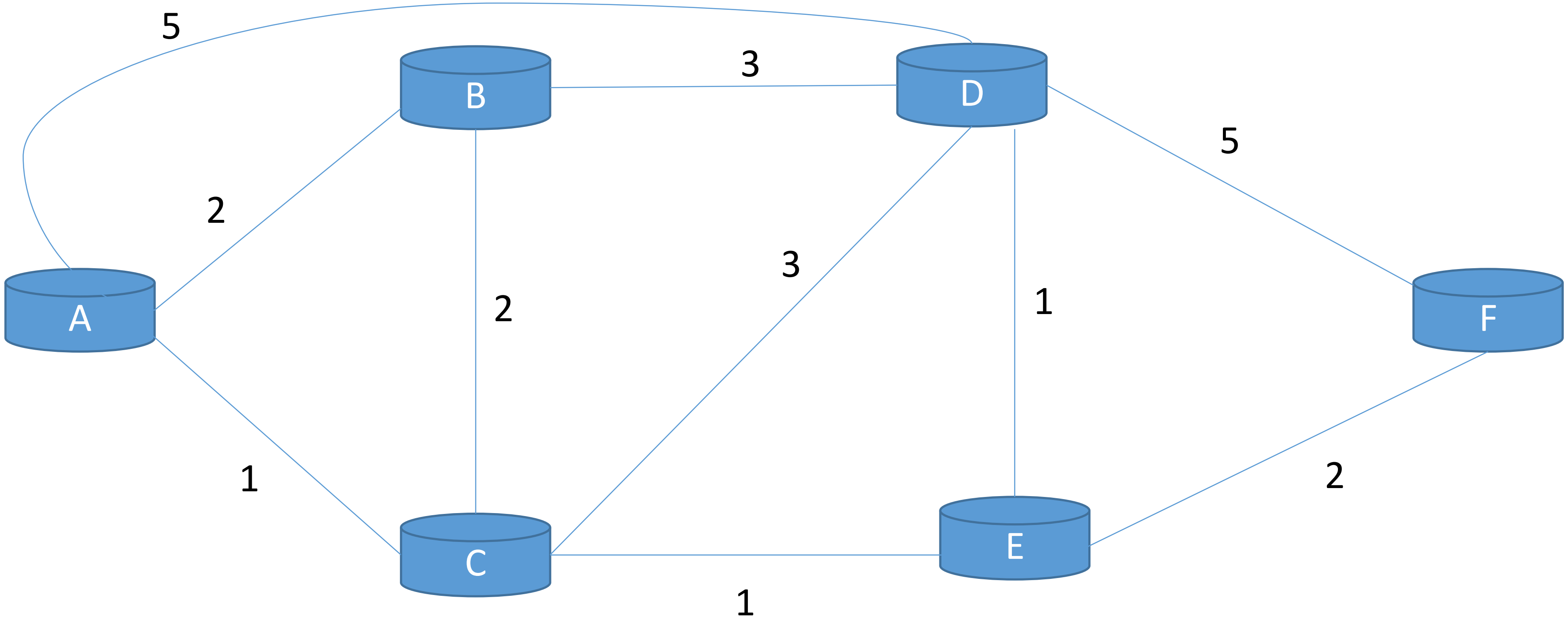
HYU-1: Find the shortest path from router R1 to all routers using Dijkstra’s Algorithm.



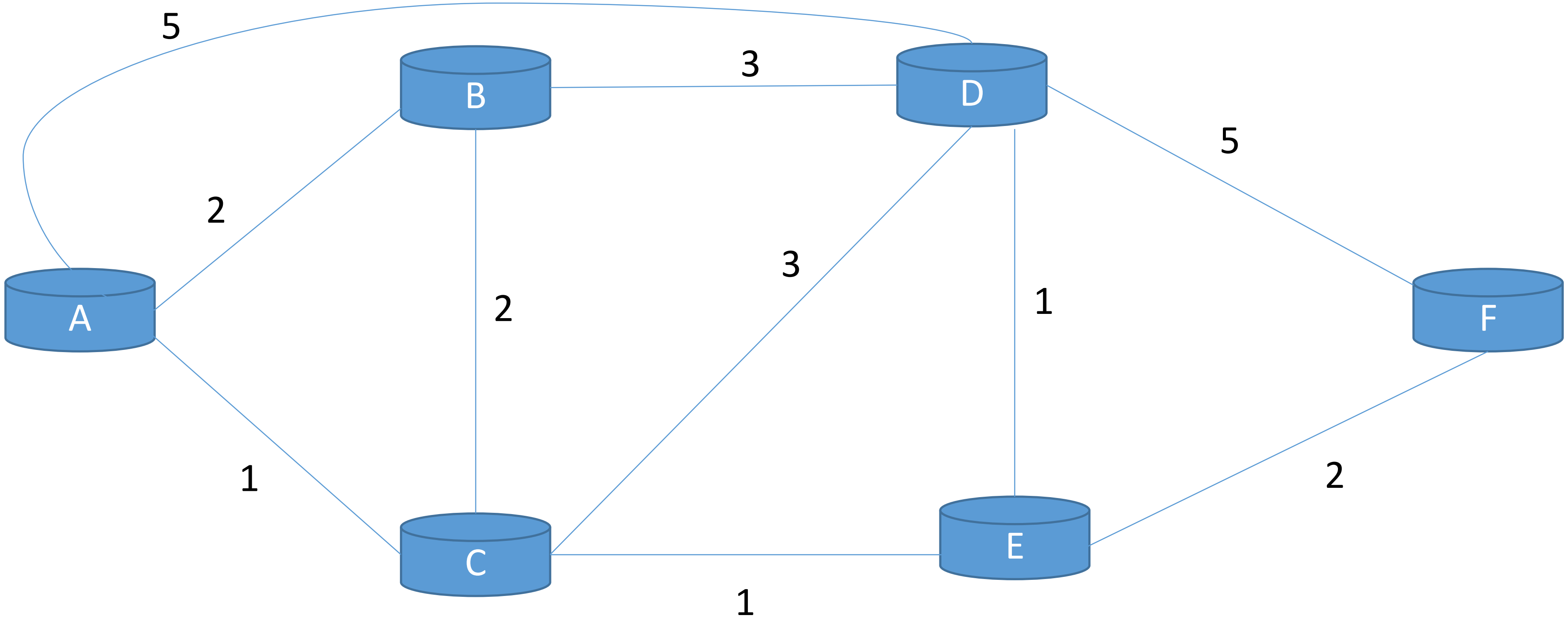
HYU-2: Find the shortest path to router R8 from all routers using Bellman Ford Algorithm.



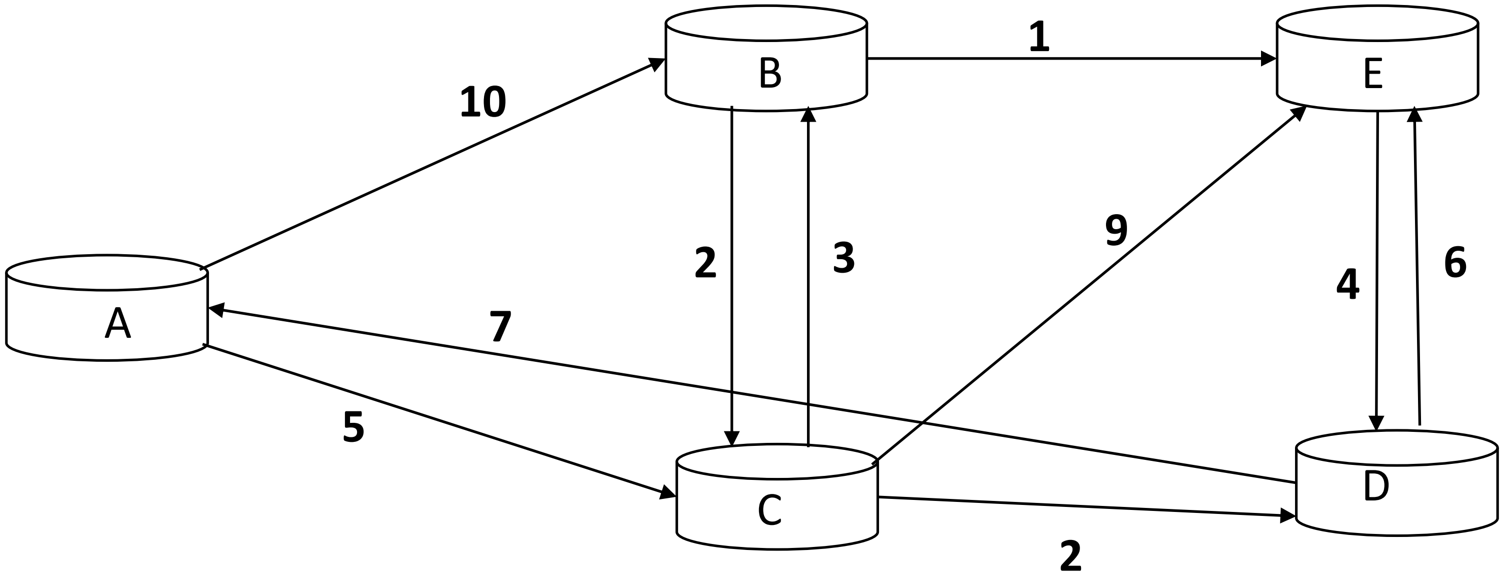
HYU-3: Find the shortest path from node A to node F using Dijkstra's algorithm. Also, mention the routing table at node E.



HYU-4: Find the shortest path to node F using Bellman Ford algorithm.

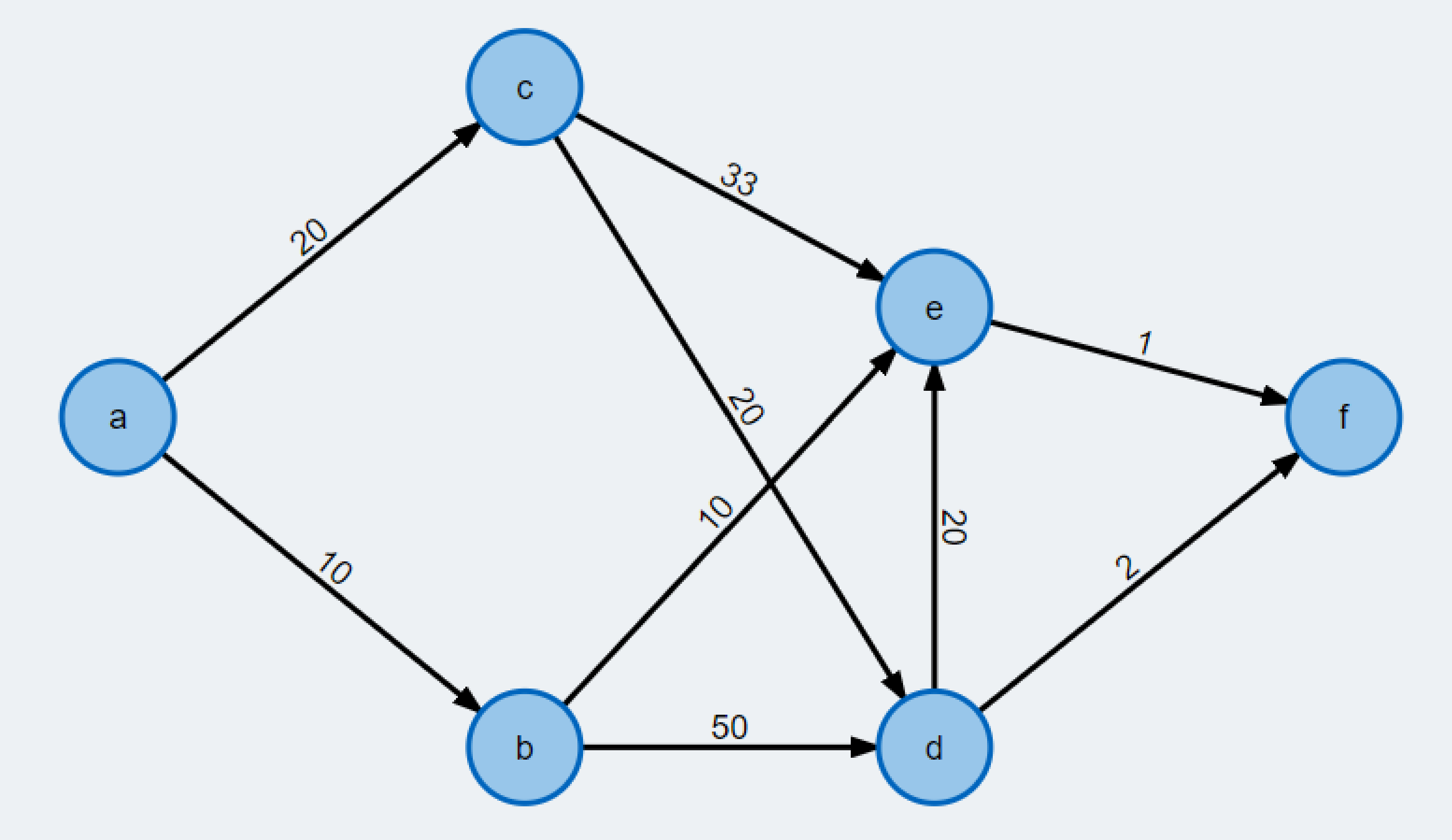


HYU-6: Find Shortest path using Dijkstra’s and Bellman Ford algorithm.

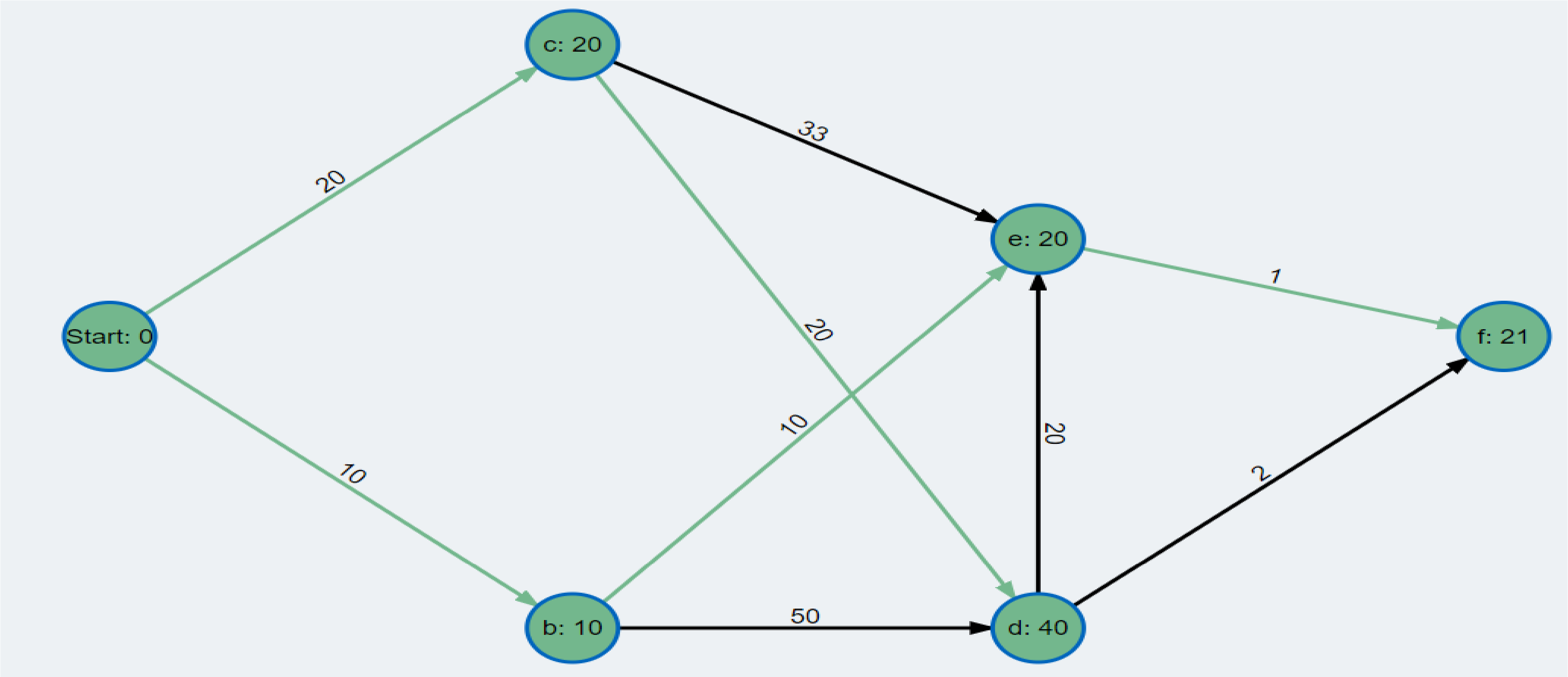


Note: When source or destination is not mentioned, you can assume any one as source and destination for Dijkstra and Bellman ford respectively.

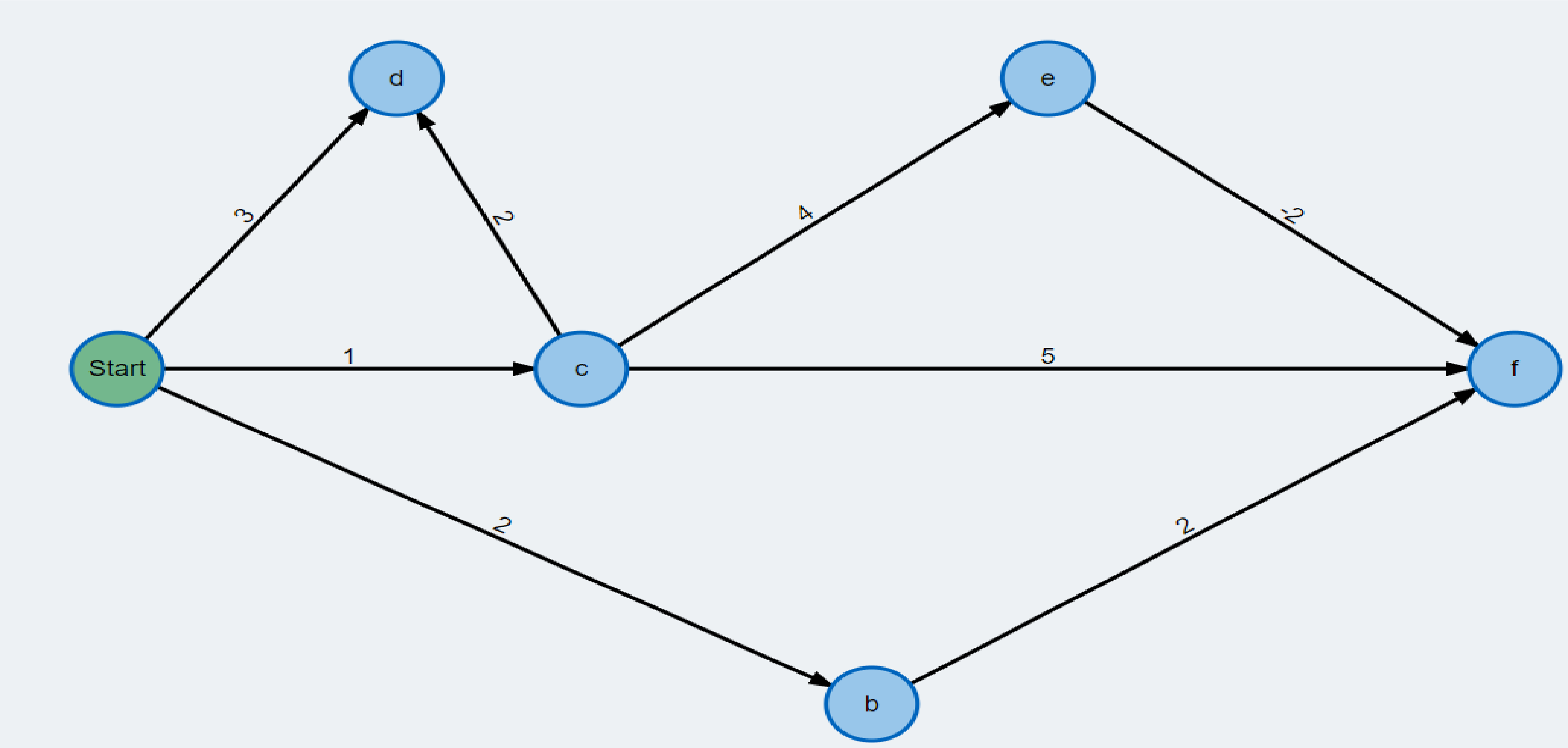
HYU-7: Use Dijkstra's Algorithm to find Shortest path from **a** to all other nodes.



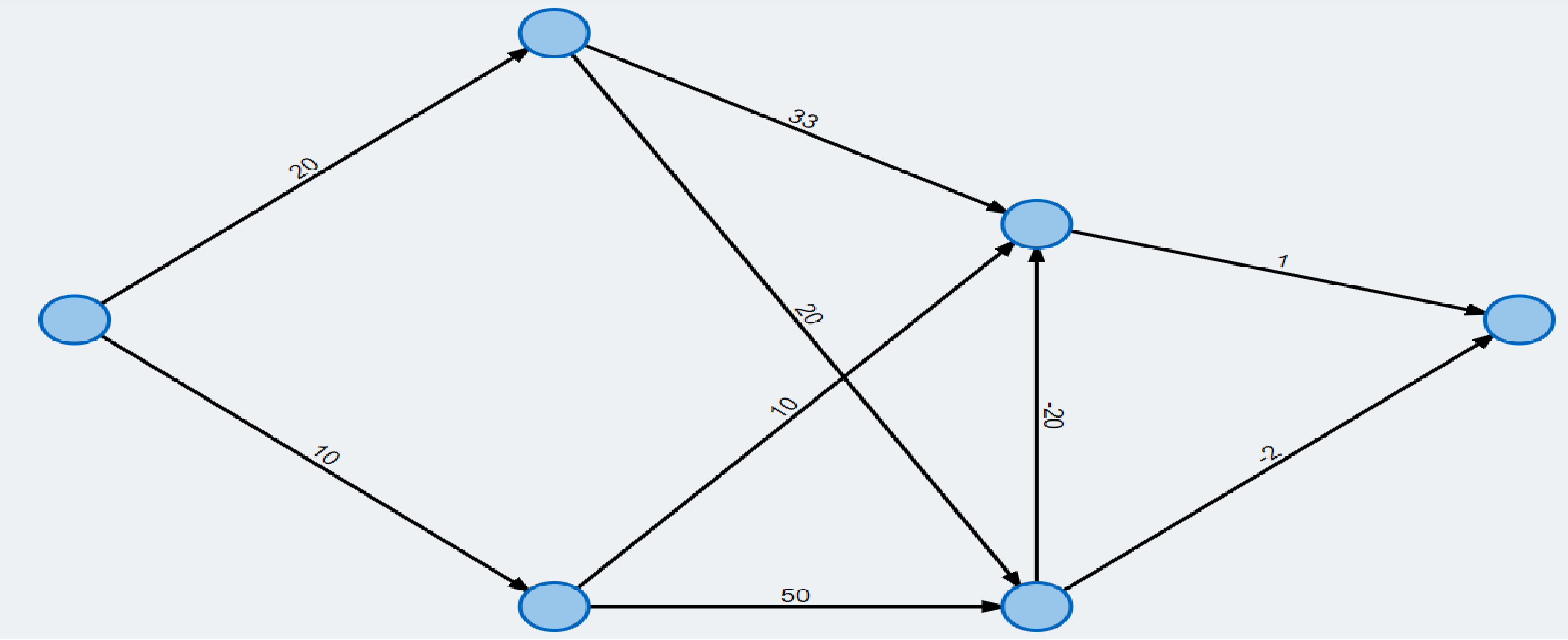
Final Answer



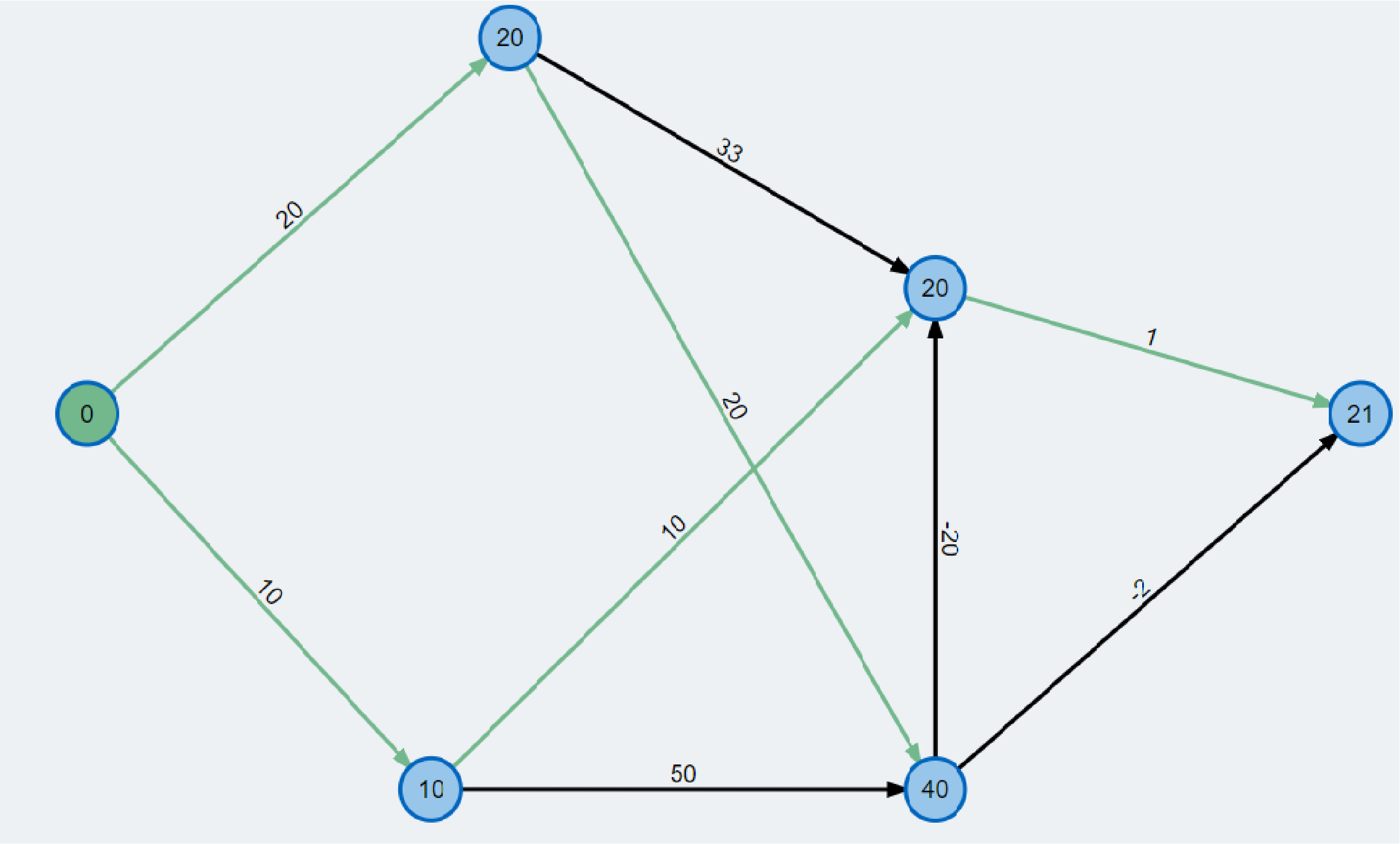
HYU-8: Find Shortest path using Dijkstra’s and Bellman Ford algorithm.



HYU-9: Use Bellman Ford to find shortest path from leftmost node to all others.



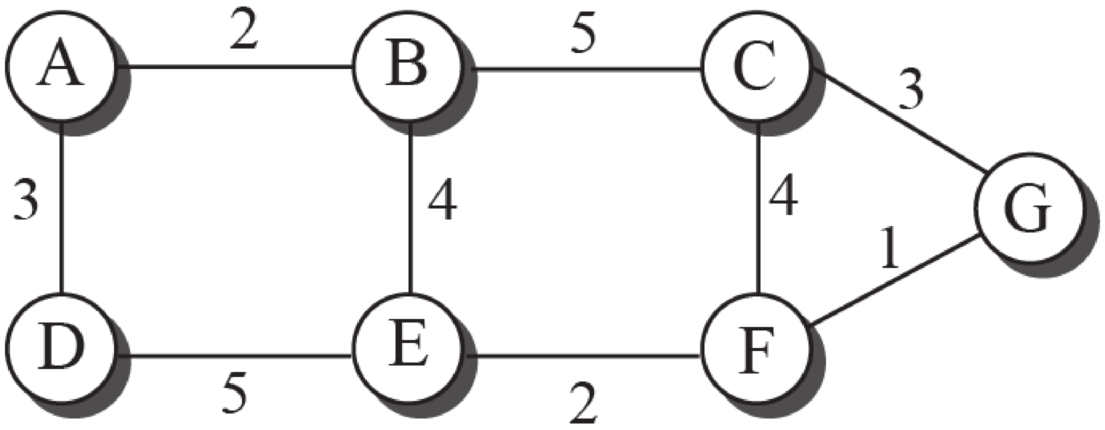
Final Answer



A	0
B	2
C	∞
D	3
E	∞
F	∞
G	∞

A	2
B	0
C	5
D	∞
E	4
F	∞
G	∞

A	∞
B	5
C	0
D	∞
E	∞
F	4
G	3

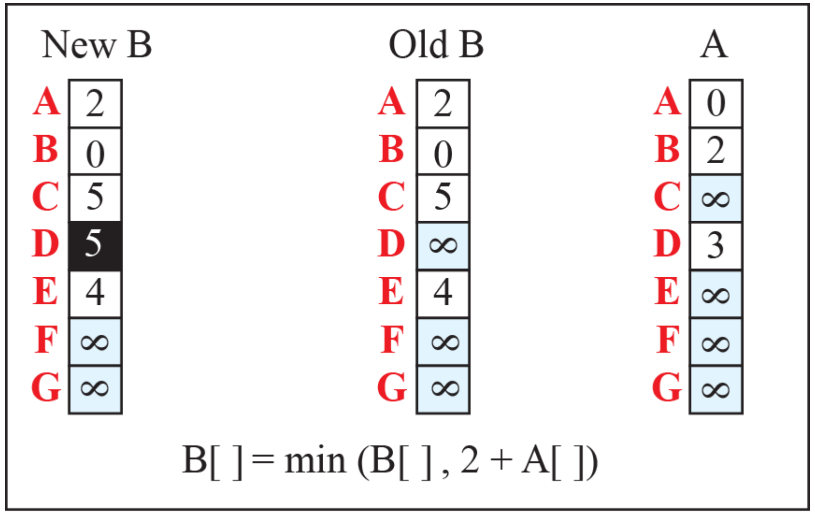


A	∞
B	∞
C	3
D	∞
E	∞
F	1
G	0

A	3
B	∞
C	∞
D	0
E	5
F	∞
G	∞

A	∞
B	4
C	∞
D	5
E	0
F	2
G	∞

A	∞
B	∞
C	4
D	∞
E	2
F	0
G	1

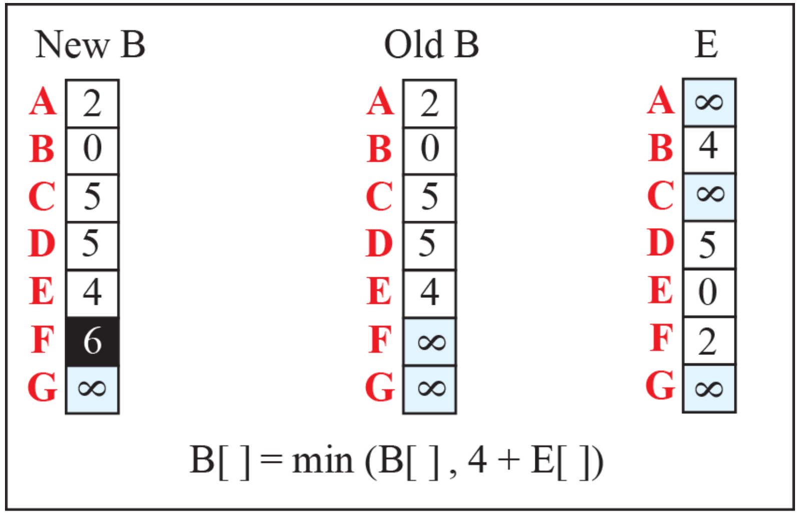


	New B	Old B	A
A	2	2	0
B	0	0	2
C	5	5	∞
D	5	∞	3
E	4	4	∞
F	∞	∞	∞
G	∞	∞	∞

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

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

Note:
X[ ]: the whole vector

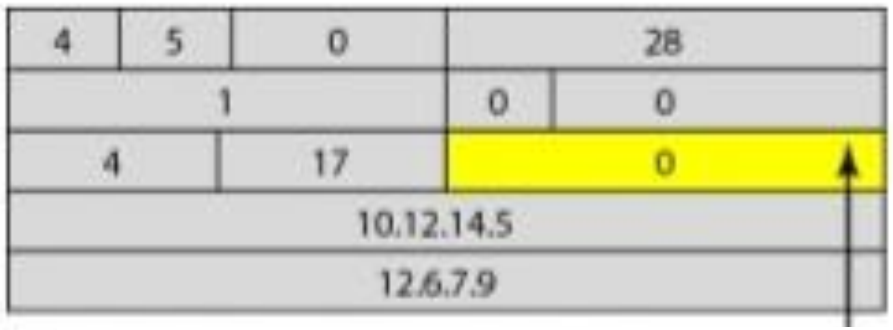


	New B	Old B	E
A	2	2	∞
B	0	0	4
C	5	5	∞
D	5	5	5
E	4	4	0
F	6	∞	2
G	∞	∞	∞

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

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

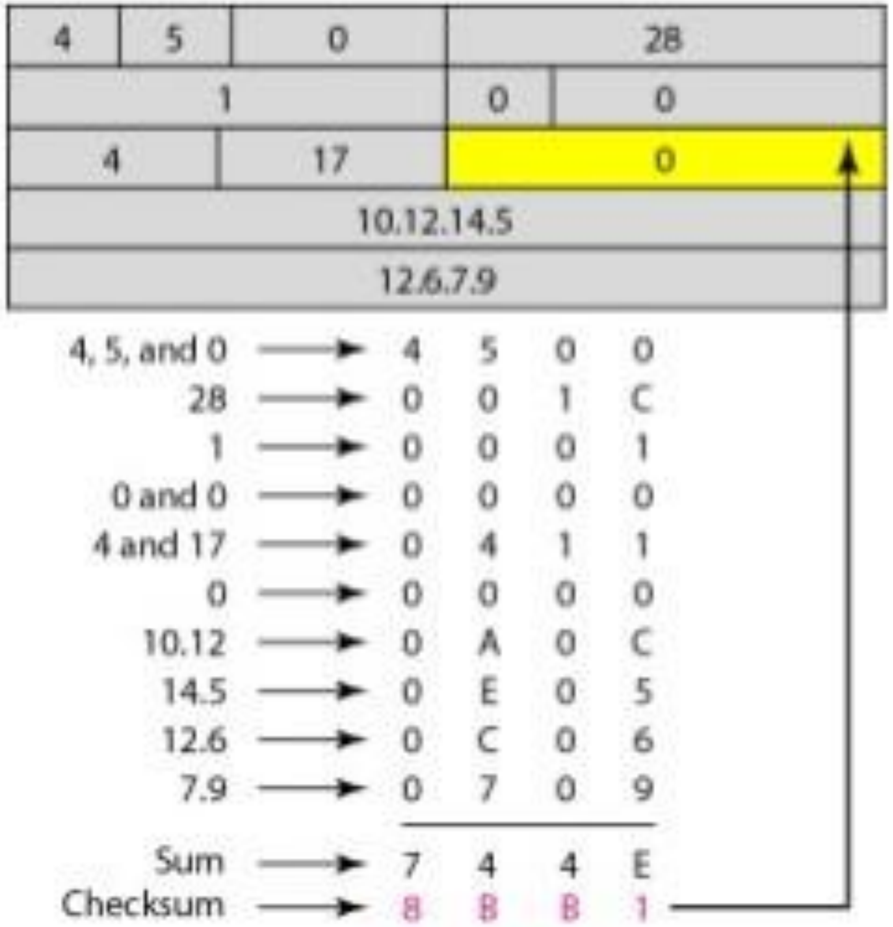
Example 5: Calculate the checksum for IPv4 header shown below:



<table>
<tr><td>4</td><td colspan="2">5</td><td>0</td><td colspan="2">28</td></tr>
<tr><td colspan="4">1</td><td>0</td><td>0</td></tr>
<tr><td colspan="2">4</td><td colspan="2">17</td><td colspan="2">0</td></tr>
<tr><td colspan="6">10.12.14.5</td></tr>
<tr><td colspan="6">12.6.7.9</td></tr>
</table>

Checksum ?

Problem: Calculate the checksum for IPv4 header shown below:



4	5	0	28
1		0	0
4	17	0	
10.12.14.5			
12.6.7.9			

4, 5, and 0	→	4	5	0	0
28	→	0	0	1	C
1	→	0	0	0	1
0 and 0	→	0	0	0	0
4 and 17	→	0	4	1	1
0	→	0	0	0	0
10.12	→	0	A	0	C
14.5	→	0	E	0	5
12.6	→	0	C	0	6
7.9	→	0	7	0	9
Sum	→	7	4	4	E
Checksum	→	8	B	B	1

End of Module 4