

UNIT-3

Network Layer

B SAI BABA,M.Tech(Ph.D),VIT,Bhimavaram

Syllabus

- ★ **Network Layer:**

- **Network layer design issues**

- ★ **Routing Algorithms**

- ★ **Congestion Control Algorithms**

- ★ **Internet working**

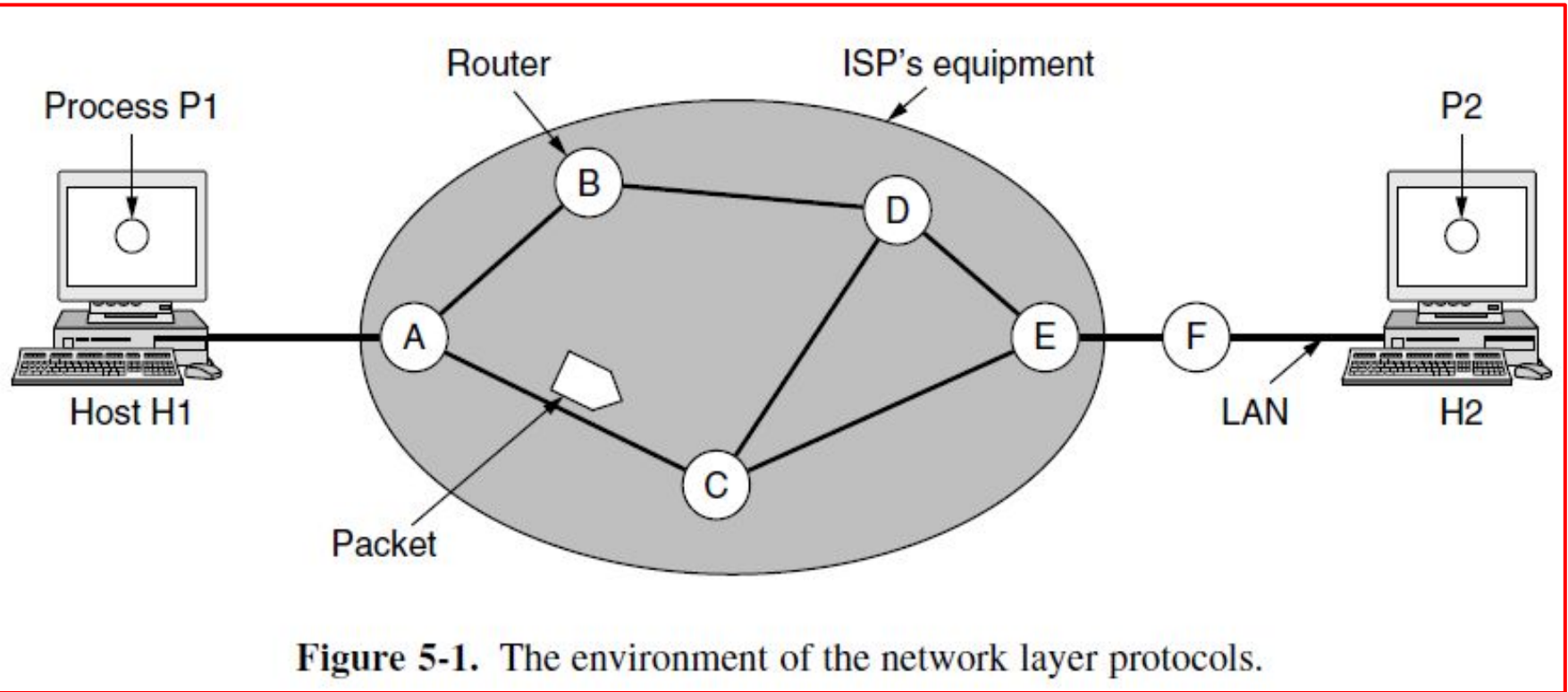
- ★ **The Network layer in the internet (IPv4 and IPv6)**

- ★ **Quality of Service**

Network Layer Design Issues

- ★ **The Network Layer or Layer 3** of the OSI (Open Systems Interconnection) model.
- ★ The design issues can be elaborated as—
 1. **Store – and – Forward Packet Switching**
 2. **Services to Transport Layer**
 - a. **Providing Connection Oriented Service**
 - b. **Providing Connectionless Service**
 3. **Routing**
 4. **Congestion Control**
 5. **Logical Addressing(IP Address)**

Store – and – Forward Packet Switching



Store – and – Forward Packet Switching

- ★ The network layer operates in an environment that uses store and forward packet switching.
- ★ The node which has a packet to send, **delivers it to the nearest router.**
- ★ The packet is stored in the router until it has fully arrived and its checksum is verified for error detection.
- ★ Once, this is done, **the packet is forwarded to the next router.**
- ★ Since, each router needs to store the entire packet before it can forward it to the next hop, the mechanism is called **store – and – forward switching.**

Services to Transport Layer

- ★ The Network Layer provides service to its immediate upper layer, namely **Transport Layer**, through the network – transport layer interface.
- ★ The two types of services provided are –

1. Connection – Oriented Service :

In this service, **a path is setup between the source and the destination**, and all the data packets belonging to a message are routed along this path.

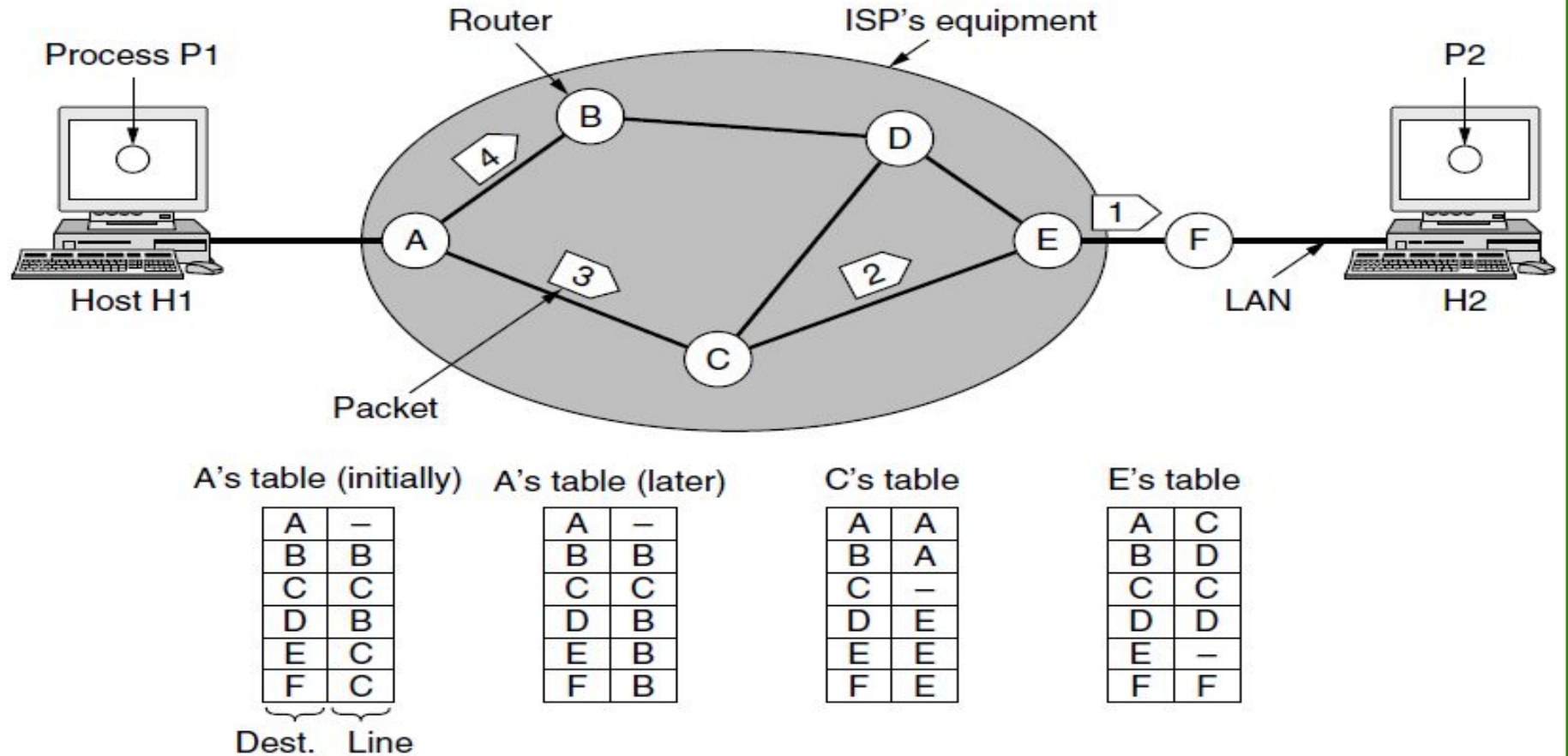
2. Connectionless Service :

In this service, each packet of the message is considered as **an independent entity** and is **individually routed from the source to the destination**.

Providing Connectionless Service

- ★ If connectionless service is offered, packets are injected into the network individually and routed independently of each other. **No advance setup is needed.**
- ★ In this context, the packets are frequently called **datagrams** and the network is called **a datagram network**.
- ★ In this section, we will examine datagram networks :

Figure 5-2. Routing within a datagram network



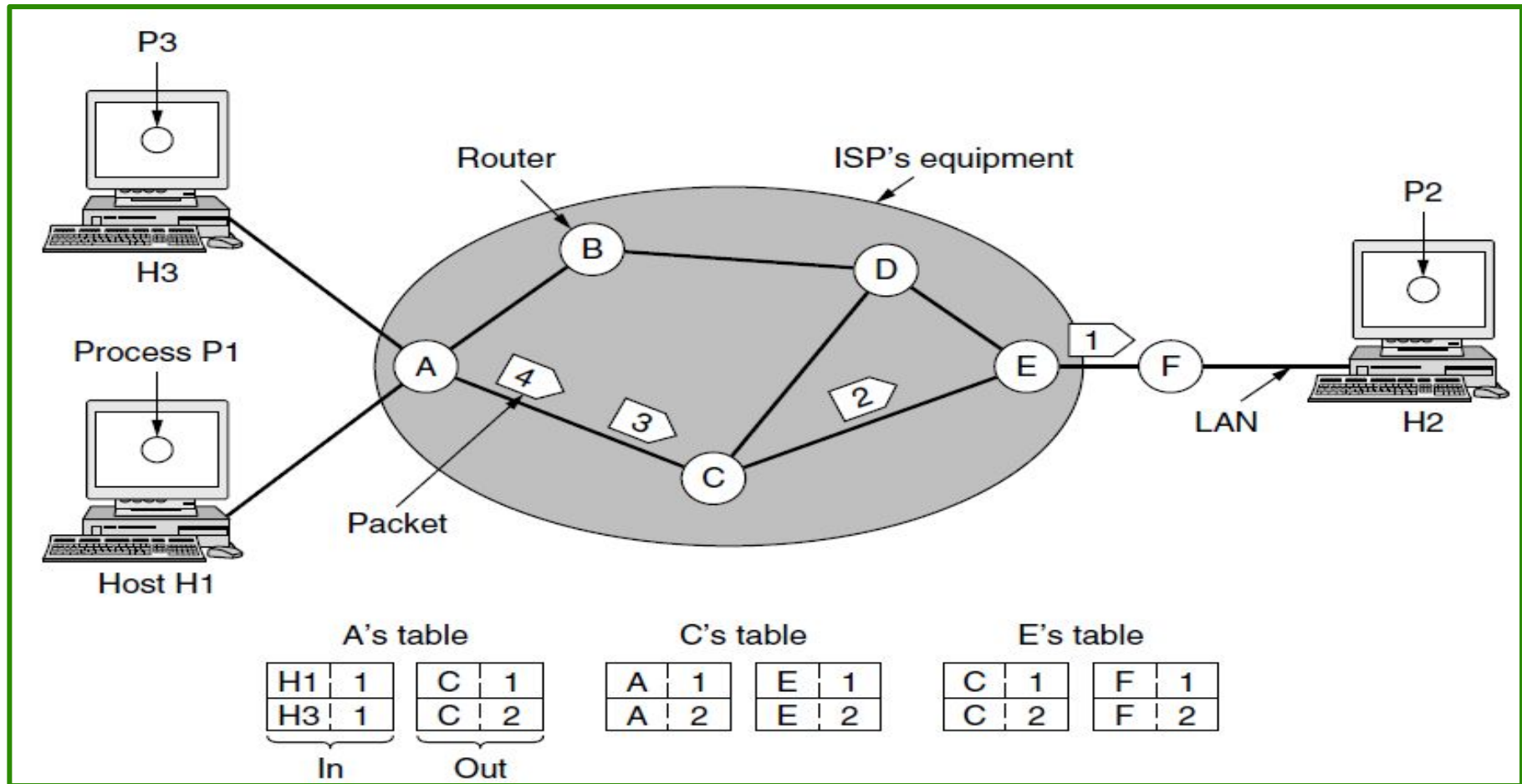
- ★ Let us assume for this example that the message is four times longer than the maximum packet size, so the network layer has to break it into four packets, 1, 2, 3, and 4, and send each of them in turn to router A using some point-to-point protocol.
- ★ At this point the ISP takes over. Every **router** has **an internal table** telling it where to send packets for each of the possible destinations.
- ★ **Each table entry is a pair** consisting of **a destination** and **the outgoing line to use for that destination**. Only directly connected lines can be used.
- ★ For example, in Fig. 5-2, **A has only two outgoing lines—to B and to C**—so every incoming packet must be sent to one of these routers, even if the ultimate destination is to some other router.
- ★ A's initial routing table is shown in the figure under the label “initially.”

- ★ At A, packets 1, 2, and 3 are stored briefly, having arrived on the incoming link and had their checksums verified. Then each packet is forwarded according to A's table, onto the outgoing link to C within a new frame.
- ★ Packet 1 is then forwarded to E and then to F. When it gets to F, **it is sent within a frame** over the LAN to H2. **Packets 2 and 3** follow the same route.
- ★ However, something different happens to **packet 4**.
- ★ When it gets to A it is sent to router B, even though it is also destined for F.
- ★ For some reason, A decided to send packet 4 via a different route than that of the first three packets. Perhaps it has learned of **a traffic jam** somewhere along **the ACE path** and updated its routing table, as shown under the label "later."
- ★ The algorithm that manages the tables and makes the routing decisions is called **the routing algorithm**.

Providing Connection Oriented Service

- ★ If connection-oriented service is used, a **path** from the source router all the way to the destination router **must be established before any data packets can be sent.**
- ★ This connection is called a **VC (virtual circuit),**
- ★ The network is called **a virtual-circuit network.**

Figure 5-3. Routing within a virtual-circuit network



- ★ The idea behind virtual circuits is **to avoid having to choose a new route** for every packet sent, as in Fig. 5-2.
- ★ Instead, when a connection is established,a route from the source machine to the destination machine is chosen as part of the connection setup and stored in tables inside the routers.
- ★ That route is used for all traffic flowing over the connection, exactly the same way that the telephone system works.
- ★ When the connection is released, the virtual circuit is also terminated.
- ★ **With connection-oriented service, each packet carries an identifier** telling which virtual circuit it belongs to.
- ★ As an example, consider the situation shown in Fig. 5-3. Here, host H1 has established connection 1 with host H2. This connection is remembered as the first entry in each of the routing tables.

- ★ The first line of A's table says that if a packet bearing connection identifier 1 comes in from H1, it is to be sent to router C and given connection identifier 1.
- ★ Similarly, the first entry at C routes the packet to E, also with connection identifier 1.
- ★ Now let us consider what happens if H3 also wants to establish a connection to H2.
- ★ It chooses connection identifier 1 (because it is initiating the connection and this is its only connection) and tells the network to establish the virtual circuit. This leads to the second row in the tables.
- ★ Note that we have a conflict here because although A can easily distinguish connection 1 packets from H1 from connection 1 packets from H3, C cannot do this. For this reason, A assigns a different connection identifier to the outgoing traffic for the second connection.
- ★ Avoiding conflicts of this kind is why routers need the ability to replace connection identifiers in outgoing packets. In some contexts, this process is called **label switching**.

Comparison of Virtual Circuit and Datagram Network

| Issue | Datagram network | Virtual-circuit network |
|---------------------------|--|--|
| Circuit setup | Not needed | Required |
| Addressing | Each packet contains the full source and destination address | Each packet contains a short VC number |
| State information | Routers do not hold state information about connections | Each VC requires router table space per connection |
| Routing | Each packet is routed independently | Route chosen when VC is set up; all packets follow it |
| Effect of router failures | None, except for packets lost during the crash | All VCs that passed through the failed router are terminated |
| Quality of service | Difficult | Easy if enough resources can be allocated in advance for each VC |
| Congestion control | Difficult | Easy if enough resources can be allocated in advance for each VC |

Routing Algorithms

Routing Algorithm

- ★ In order to transfer the packets from source to the destination, the network layer must determine **the best route** through which packets can be transmitted.
- ★ Whether the network layer provides **datagram service or virtual circuit service**, the main job of the network layer is to provide **the best route**. The routing protocol provides this job.
- ★ The routing protocol is a routing algorithm that provides the best path from the source to the destination. The best path is the path that has the **"least-cost path"** from source to the destination.
- ★ Routing is the process of forwarding the packets from source to the destination but **the best route to send the packets is determined by the routing algorithm.**

Optimality Principle

Introduction :

- ★ A general statement is made about optimal routes without regard to network topology or traffic. This statement is known as the optimality principle(Bellman,1975).

Statement of the optimality principle :

- ★ It states that if the router J is on the optimal path from router I to router K, then the optimal path from J to K also falls along the same route. Call the route from I to J, r_1 and the rest of the route r_2 . it could be concatenated with r_1 to improve the route from I to K, contradicting our statement that r_1r_2 is optimal only if a route better than r_2 existed from J to K.

Introduction :

- ★ A general statement is made about optimal routes without regard to network topology or traffic. This statement is known as the optimality principle(Bellman,1975).

Explanation:

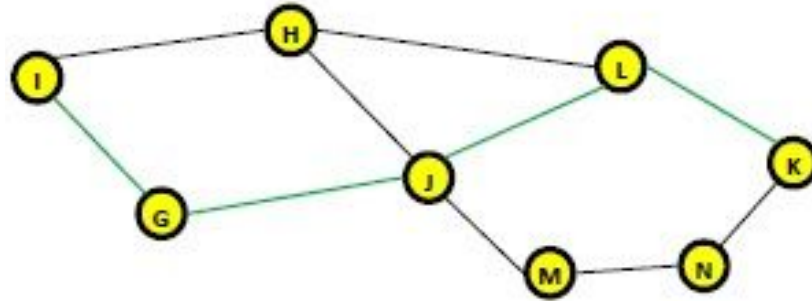
- ★ The purpose of a routing algorithm at a router is to decide which output line an incoming packet should go.
- ★ **The optimal path** from a particular router to another may be the **least cost path, the least distance path, the least time path, the least hops path** or a combination of any of the above.

The optimality principle can be logically proved as follows –

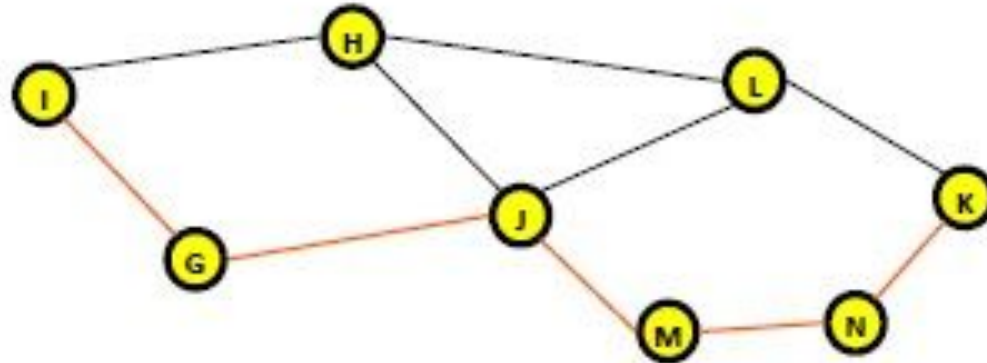
If the router J is on the optimal path from router I to router K, If a better route could be found between router J and router K, the path from router I to router K via J would be updated via this route. Thus, the optimal path from J to K will again lie on the optimal path from I to K.

Example

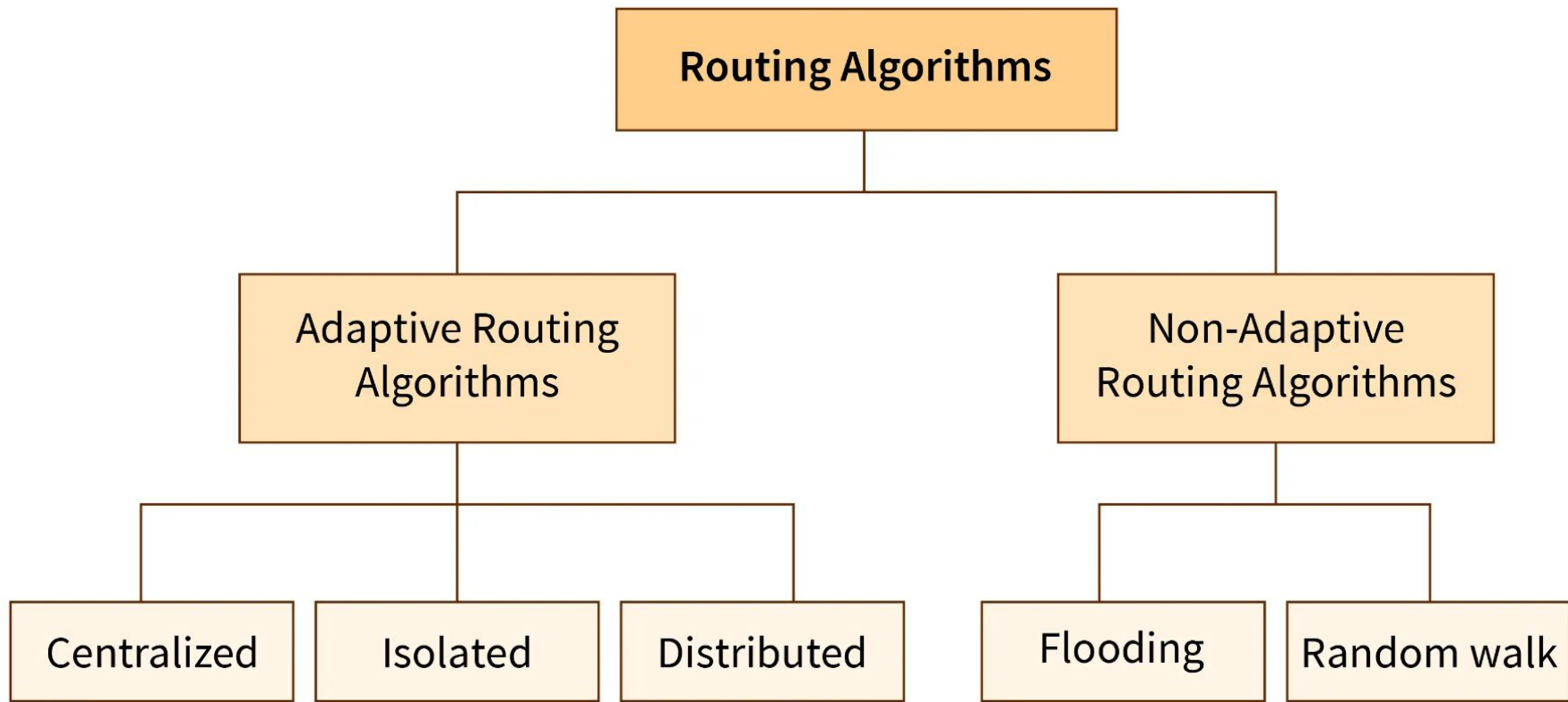
Consider a network of routers, $\{G, H, I, J, K, L, M, N\}$ as shown in the figure. Let the optimal route from I to K be as shown via the green path, i.e. via the route I-G-J-L-K. According to the optimality principle, the optimal path from J to K will be along the same route, i.e. J-L-K.



Now, suppose we find **a better route from J to K is found**, say along J-M-N-K. Consequently, we will also need to update the optimal route from I to K as I-GJ- M-N-K, since the previous route ceases to be optimal in this situation. This new optimal path is shown line orange lines in the following figure –



Types of Routing Algorithm



Adaptive Routing Algorithm

- ★ Adaptive routing algorithm is also called **a dynamic routing algorithm**.
- ★ In this algorithm, **the routing decisions** are made based on **network traffic and topology**.
- ★ The parameters which are used in adaptive routing algorithms are **distance, hop, estimated transit time**.
- ★ The adaptive routing algorithm is of **three types** –
 - **Centralized algorithm**
 - **Isolation algorithm**
 - **Distributed algorithm**

Centralized algorithm:

- ★ In centralized routing, **one centralized node** has the total network information and takes the routing decisions.
- ★ It finds the least-cost path between source and destination nodes by using **global knowledge about the network**. So, it is also known as global routing algorithm.
- ★ The advantage of this routing is that only the central node is required to store network information and so the resource requirement of the other nodes may be less.
- ★ Eg: **Link state routing algorithm**

Isolated algorithm:

- ★ This algorithm procures the routing information by using **local information** instead of gathering information from other nodes.

Distributed algorithm:

- ★ This is a **decentralized algorithm** where each node receives information from its neighbouring nodes and takes the decision based upon the received information.
- ★ The least-cost path between source and destination is computed **iteratively in a distributed manner**.
- ★ An advantage is that each node can **dynamically change routing decisions** based upon the changes in the network.
- ★ Example : **distance vector routing algorithm**.

Non-adaptive Routing Algorithm

- ★ Non-adaptive routing algorithm is also called **a static routing algorithm**.
- ★ In a non-adaptive routing algorithm, the routing decisions are **not made based on network traffic and topology**.
- ★ This algorithm is used by static routing.
- ★ Non-adaptive routing algorithms are **simple** as compared to Adaptive routing algorithms in terms of **complexity**.
- ★ The non-adaptive routing algorithm is of **two types** –
 - **Flooding**
 - **Random walks**

Shortest Path Algorithm

Introduction:

- ★ **Dijkstra's algorithm** and **the Bellman-Ford algorithm** are two different algorithms used to find the shortest path in a weighted graph.

| Dijkstra's algorithm | Bellman-Ford algorithm |
|--|---|
| It works only for graphs with non-negative edge weights . | It works for graphs with both non-negative and negative edge weights . |

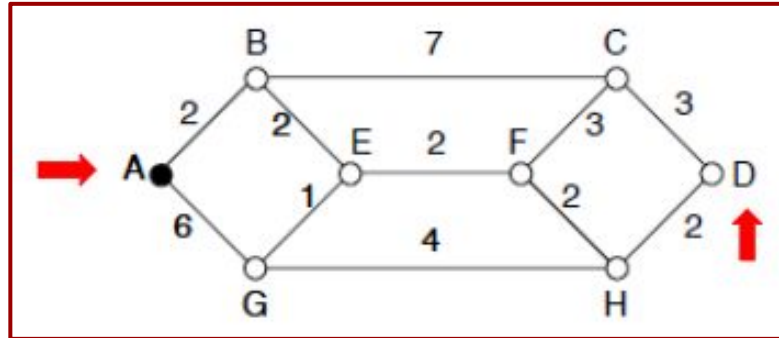
Dijkstra's algorithm:

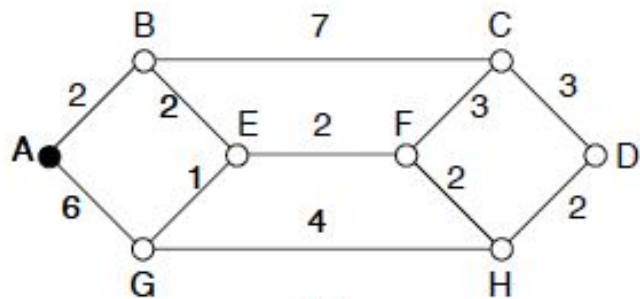
- ★ The shortest path algorithm was first introduced by **Edsger W. Dijkstra in 1956.**
- ★ This algorithm, commonly known as **Dijkstra's algorithm**, is used to find the shortest path between nodes in a graph, particularly in **weighted graphs** where each **edge has a non-negative weight.**
- ★ Dijkstra's algorithm is widely used in various applications, including **computer networking, transportation systems**, and more, to find the shortest path from a source node to all other nodes in the graph.

Procedure of Shortest Path Algorithm

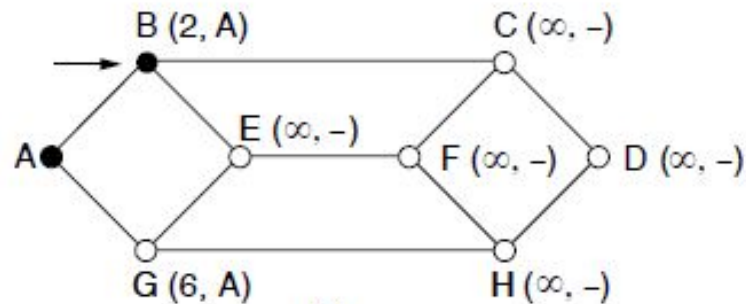
- ★ Initially mark all nodes (except source) with infinite distance.
 - working node = source node
 - Sink node = destination node
- ★ While the working node is not equal to the sink
 1. Mark the working node as permanent.
 2. Examine all adjacent nodes in turn
 - If the sum of label on working node plus distance from working node to adjacent node is less than current labeled distance on the adjacent node, this implies a shorter path. Re label the distance on the adjacent node and label it with the node from which the probe was made.
 - 3. Examine all tentative nodes (not just adjacent nodes) and mark the node with the smallest labeled value as permanent. This node becomes the new working node.
- ★ Reconstruct the path backwards from sink to source

- The idea is to build a graph of the network, with each node of the graph representing a router and each edge of the graph representing a communication line, or link.
- To choose a route between a given pair of routers, the algorithm just finds the shortest path between them on the graph.
- The concept of a shortest path deserves some explanation.
 - One way of measuring path length is the **number of hops**. Using this metric, the paths ABC and ABE in Fig. are equally long.
 - Another metric is **the geographic distance in kilometers**, in which case ABC is clearly much longer than ABE

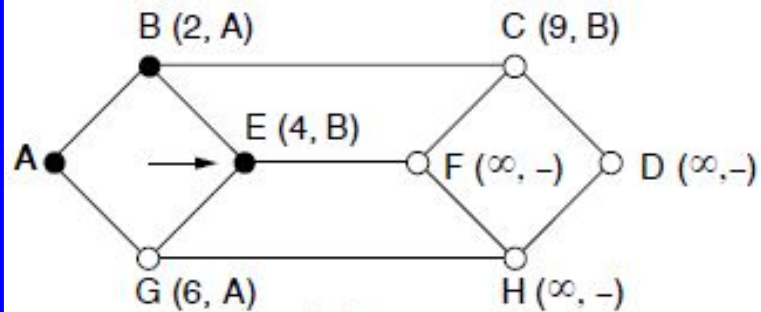




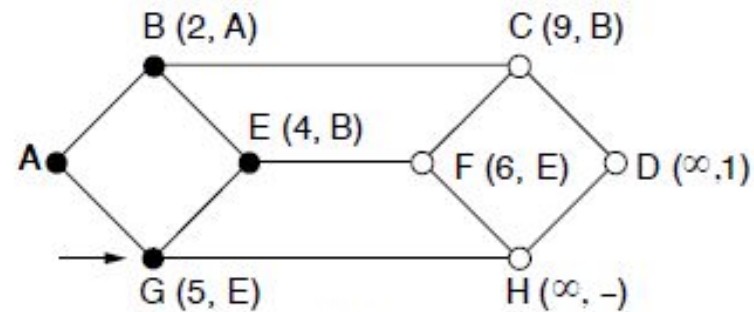
(a)



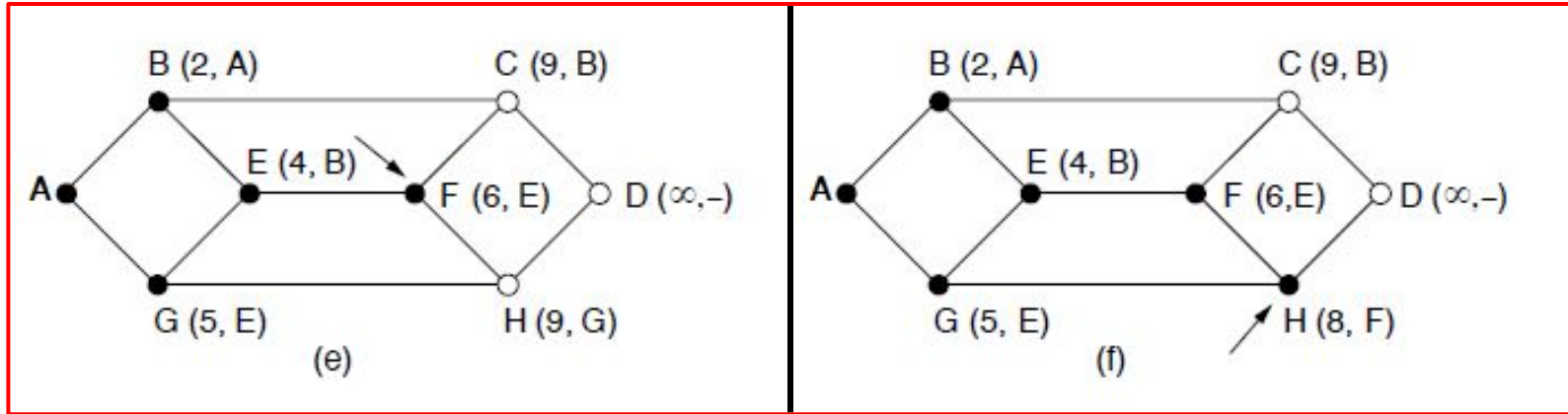
(b)



(c)



(d)

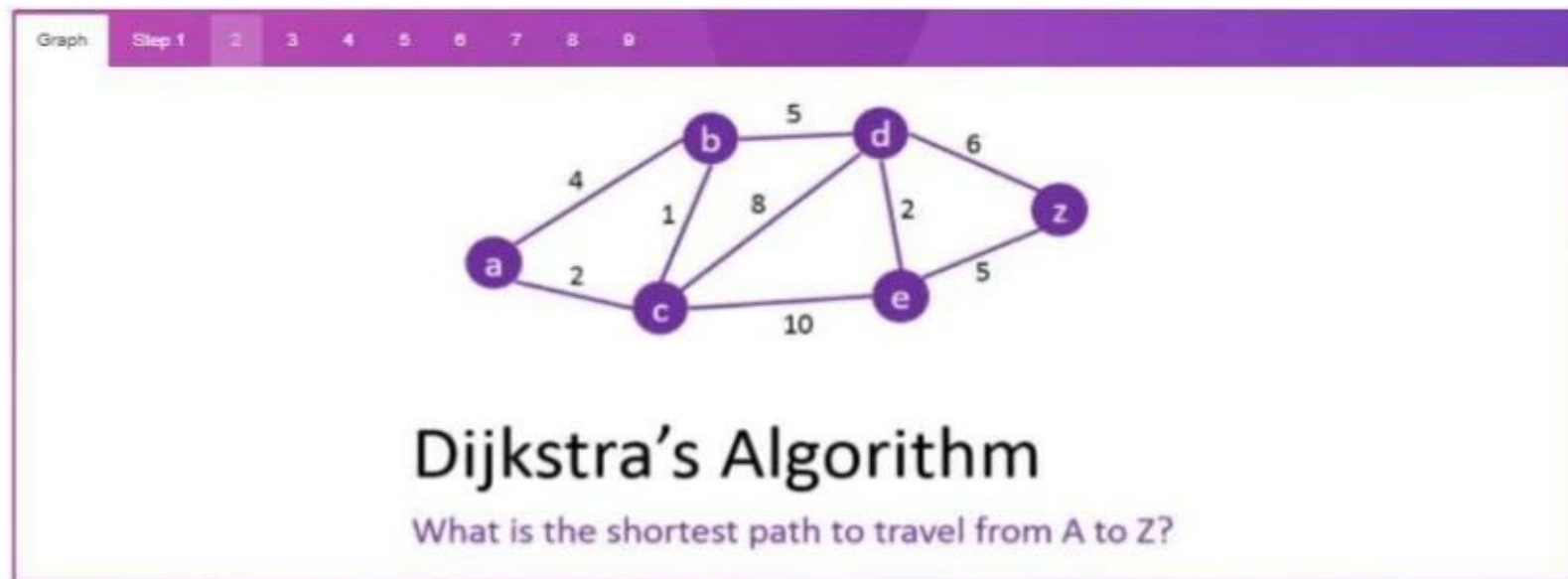


- The first six steps used in computing the shortest path from A to D.
- The arrows indicate the working node.

Example :2

Dijkstra Algorithm (Shortest Path Algorithm)

- How Routers Decide Shortest Path Using dijkstra Algorithm ?



Step 1

Graph

Step 1

2

3

4

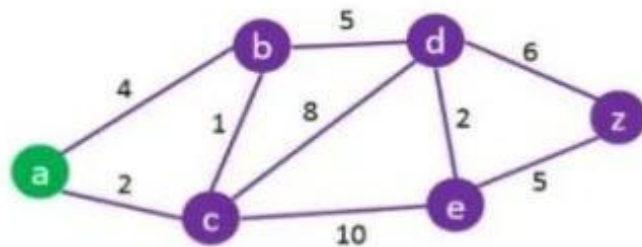
5

6

7

8

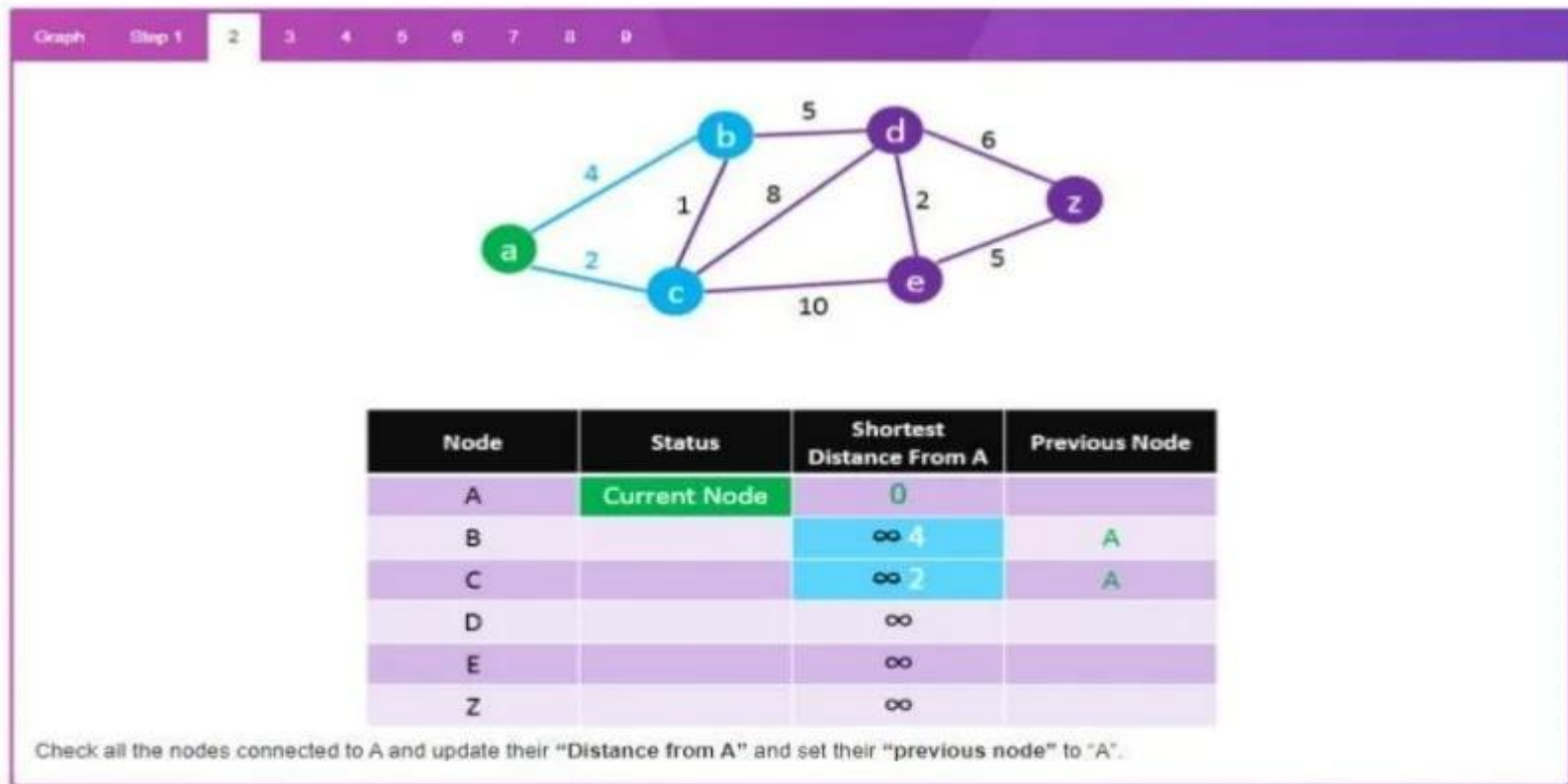
9



| Node | Status | Shortest Distance From A | Previous Node |
|------|--------------|--------------------------|---------------|
| A | Current Node | 0 | |
| B | | ∞ | |
| C | | ∞ | |
| D | | ∞ | |
| E | | ∞ | |
| Z | | ∞ | |

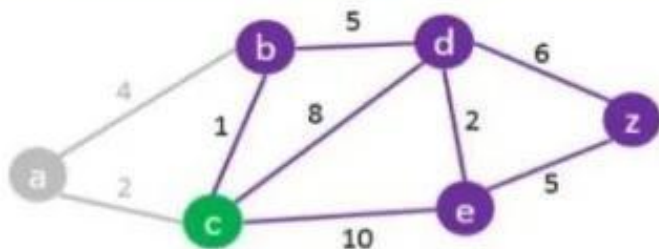
Start by setting the starting node (A) as the current node.

Step 2



Step 3

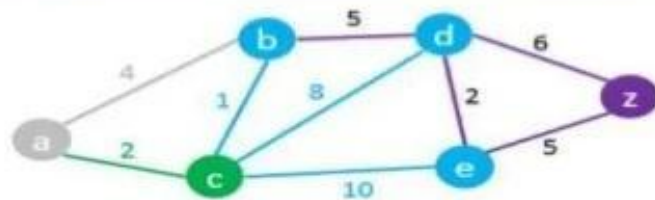
Graph Step 1 2 3 4 5 6 7 8 9



| Node | Status | Shortest Distance From A | Previous Node |
|------|--------------|--------------------------|---------------|
| A | Visited Node | 0 | |
| B | | ∞ 4 | A |
| C | Current Node | ∞ 2 | A |
| D | | ∞ | |
| E | | ∞ | |
| Z | | ∞ | |

Set the current node (A) to "visited" and use the closest unvisited node to A as the current node (e.g. in this case: Node C).

Step 4



| Node | Status | Shortest Distance From A | Previous Node |
|------|--------------|---|---------------|
| A | Visited Node | 0 | |
| B | | 4 $2+1=3$ | C |
| C | Current Node | 2 | A |
| D | | ∞ $2+8=10$ | C |
| E | | ∞ $2+10=12$ | C |
| Z | | ∞ | |

Check all unvisited nodes connected to the current node and add the distance from A to C to all distances from the connected nodes. Replace their values only if the new distance is lower than the previous one.

C \rightarrow B: $2 + 1 = 3 < 4$ – Change Node B

C \rightarrow D: $2 + 8 = 10 < \infty$ – Change Node D

C \rightarrow E: $2 + 10 = 12 < \infty$ – Change Node E

Step 5

Graph

Step 1

2

3

4

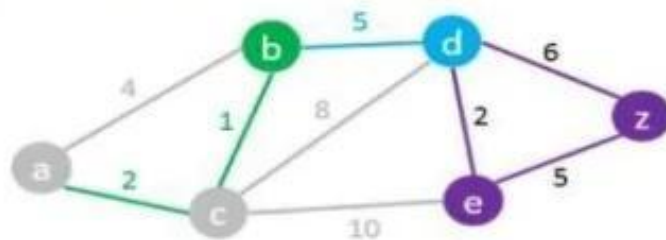
5

6

7

8

9



| Node | Status | Shortest Distance From A | Previous Node |
|------|--------------|--------------------------|---------------|
| A | Visited Node | 0 | |
| B | Current Node | 3 | C |
| C | Visited Node | 2 | A |
| D | | 10 | C |
| E | | 12 | C |
| Z | | ∞ | |

Set the current node C status to Visited.

We then repeat the same process always picking the closest unvisited node to A as the current node.

In this case node B becomes the current node.

Step 6

Graph

Step 1

2

3

4

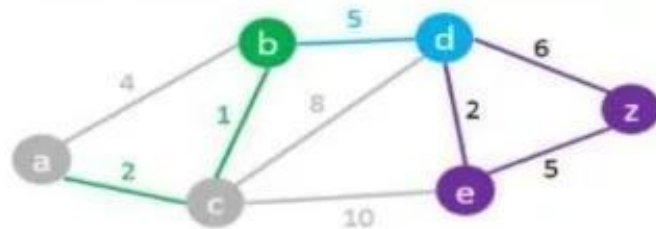
5

6

7

8

9



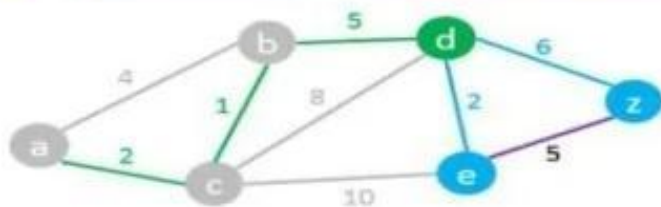
| Node | Status | Shortest Distance From A | Previous Node |
|------|--------------|--------------------------|---------------|
| A | Visited Node | 0 | |
| B | Current Node | 3 | C |
| C | Visited Node | 2 | A |
| D | | 10 $3+5=8$ | B |
| E | | 12 | C |
| Z | | ∞ | |

B \rightarrow D $3+5 = 8 < 10$ – Change Node D

Next "Current Node" will be D as it has the shortest distance from A amongst all unvisited nodes.

Step 7

Graph Step 1 2 3 4 5 6 7 8 9



| Node | Status | Shortest Distance From A | Previous Node |
|------|--------------|-------------------------------|---------------|
| A | Visited Node | 0 | |
| B | Visited Node | 3 | C |
| C | Visited Node | 2 | A |
| D | Current Node | 8 | B |
| E | | 12 $8 + 2 = 10$ | D |
| Z | | ∞ $8 + 6 = 14$ | D |

D \rightarrow E $8 + 2 = 10 < 12$ – Change Node E

D \rightarrow Z $8 + 6 = 14 < \infty$ – Change Node Z

We found a path from A to Z but it may not be the shortest one yet. So we need to carry on the process.

Next "Current Node": E

Step 8

Graph

Step 1

2

3

4

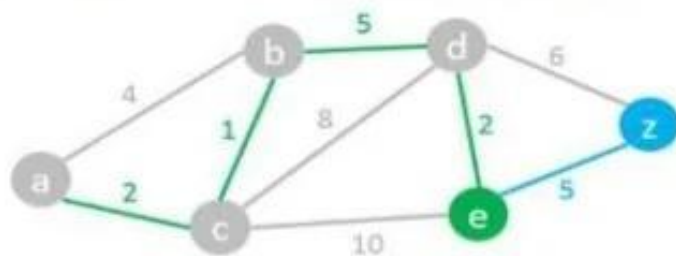
5

6

7

8

9



| Node | Status | Shortest Distance From A | Previous Node |
|------|--------------|------------------------------|---------------|
| A | Visited Node | 0 | |
| B | Visited Node | 3 | C |
| C | Visited Node | 2 | A |
| D | Visited Node | 8 | B |
| E | Current Node | 10 | D |
| Z | | 14 10 + 5 = 15 | D |

E → Z $10 + 5 = 15 > 14$ – We do not change node Z.

Step 9

Graph

Step 1

2

3

4

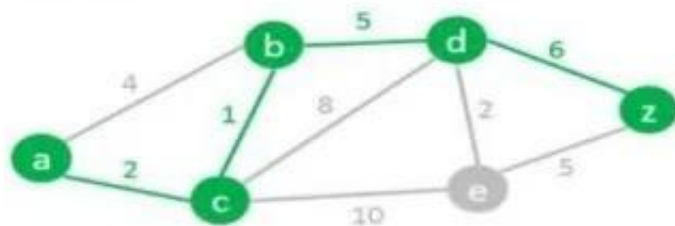
5

6

7

8

9



| Node | Status | Shortest Distance From A | Previous Node |
|------|--------------|--------------------------|---------------|
| A | Visited Node | 0 | |
| B | Visited Node | 3 | C |
| C | Visited Node | 2 | A |
| D | Visited Node | 8 | B |
| E | Visited Node | 10 | D |
| Z | Current Node | 14 | D |

We found the shortest path from A to Z.

Read the path from Z to A using the previous node column:

Z > D > B > C > A

So the Shortest Path is:

A - C - B - D - Z with a length of 14

Distance Vector Routing Algorithm

“Distant vector routing algorithm also called as **Bellman-Ford algorithm** or **Ford Fulkerson algorithm** used to calculate the shortest path in the network.”

Distance Vector Routing Algorithm

- The Distance vector algorithm is a dynamic algorithm.
- The term distance vector refers to the fact that the protocol manipulates vectors of distances to other nodes in the network.
- It is mainly used in Advanced Research Projects Agency Network (ARPANET), and Routing Information Protocol (RIP).
- Each router maintains a distance table known as **Vector**.

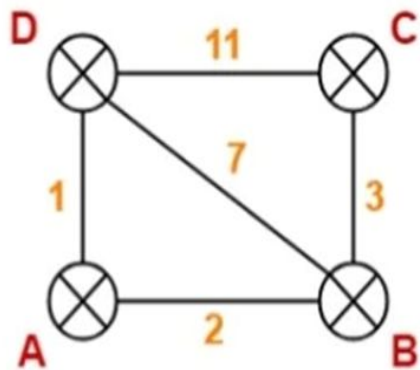


Distance Vector Routing Algorithm

- The Distance vector algorithm is iterative, asynchronous and distributed.
- **Distributed:** It is distributed in that each node receives information from one or more of its directly attached neighbors, performs calculation and then distributes the result back to its neighbors.
- **Iterative:** It is iterative in that its process continues until no more information is available to be exchanged between neighbors.
- **Asynchronous :**An asynchronous algorithm is one in which nodes operate independently and don't necessarily follow a synchronized clock or timing

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



A Routing Table

| Net Id | Cost | Next Hop |
|--------|----------|----------|
| A | 0 | A |
| B | 2 | B |
| C | ∞ | - |
| D | 1 | D |

B Routing Table

| Net Id | Cost | Next Hop |
|--------|------|----------|
| A | 2 | A |
| B | 0 | B |
| C | 3 | C |
| D | 7 | D |

C Routing Table

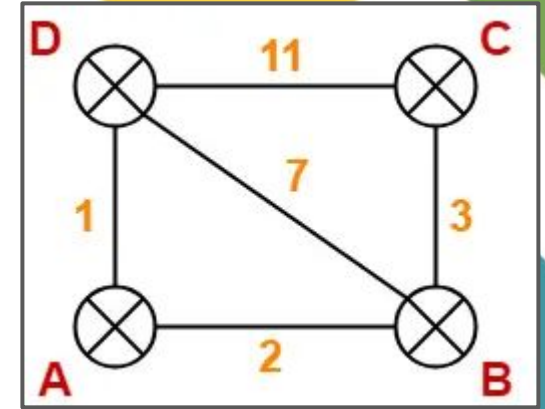
| Net Id | Cost | Next Hop |
|--------|----------|----------|
| A | ∞ | - |
| B | 3 | B |
| C | 0 | C |
| D | 11 | D |

D Routing Table

| Net Id | Cost | Next Hop |
|--------|------|----------|
| A | 1 | A |
| B | 7 | B |
| C | 11 | C |
| D | 0 | D |

Updating Routing Table

1. Each router exchanges its distance vector with its neighbors.
2. After exchanging the distance vectors, each router prepares a new routing table.
3. Router A receives distance vectors from its neighbors B and D.
4. Router A prepares a new routing table.



Bellman-Ford algorithm:

The Distance Vector calculation is based on minimizing the cost to each destination

$D_x(y)$ = Estimate of least cost from x to y

$C(x,v)$ = Node x knows cost to each neighbor v

$$D_x(y) = \min \{ C(x,v) + D_v(y) \}$$

Source

Destination

Intermediate Node

At Router A :

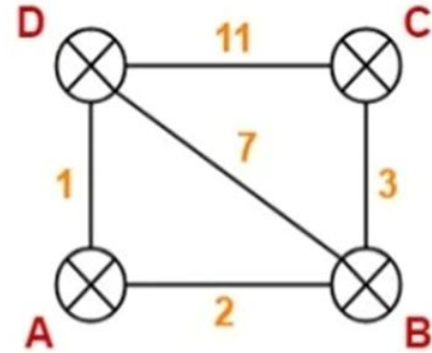
| From B | | From D | |
|--------|---|--------|----|
| A | 2 | A | 1 |
| B | 0 | B | 7 |
| C | 3 | C | 11 |
| D | 7 | D | 0 |

Cost(A→B) = 2

Cost(A→D) = 1

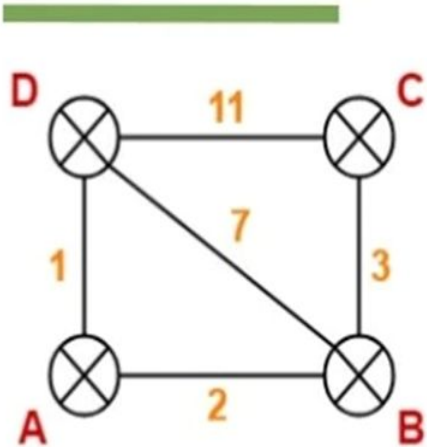
| Destination | Distance | Next hop |
|-------------|----------|----------|
| A | 0 | A |
| B | | |
| C | | |
| D | | |

New Routing Table at Router A



- 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.

Updating "A Routing Table"



A Routing Table

| Net Id | Cost | Next Hop |
|--------|----------|----------|
| A | 0 | A |
| B | 2 | B |
| C | ∞ | - |
| D | 1 | D |

New A Routing Table

| Net Id | Cost | 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

From A

| | |
|---|----------|
| A | 0 |
| B | 2 |
| C | ∞ |
| D | 1 |

From C

| | |
|---|----------|
| A | ∞ |
| B | 3 |
| C | 0 |
| D | 11 |

From D

| | |
|---|----|
| A | 1 |
| B | 7 |
| C | 11 |
| D | 0 |

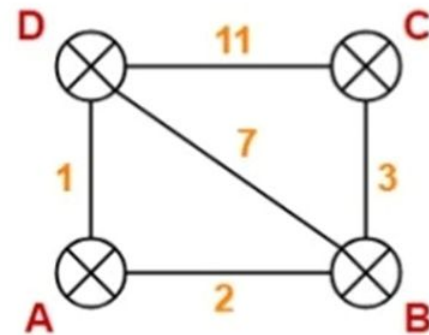
Cost (B→A) = 2

Cost (B→C) = 3

Cost (B→D) = 7

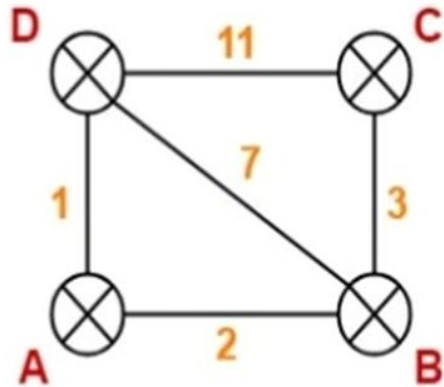
| Destination | Distance | Next hop |
|-------------|----------|----------|
| A | | |
| B | 0 | B |
| C | | |
| D | | |

New Routing Table at Router B



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

Updating "B Routing Table"



B Routing Table

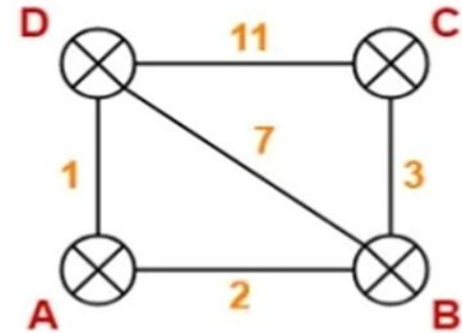
| Net Id | Cost | Next Hop |
|--------|------|----------|
| A | 2 | A |
| B | 0 | B |
| C | 3 | C |
| D | 7 | D |

**New B
Routing Table**

| Net Id | Cost | 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



| From B | | From D | |
|--------|---|--------|----|
| A | 2 | A | 1 |
| B | 0 | B | 7 |
| C | 3 | C | 11 |
| D | 7 | D | 0 |

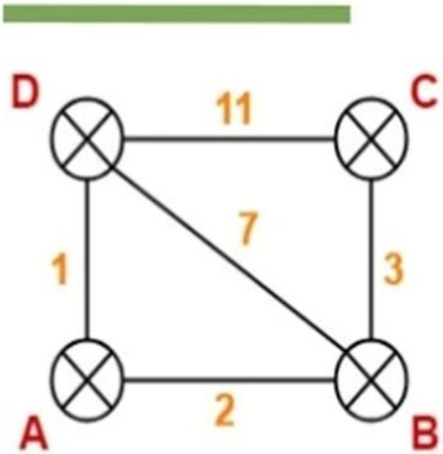
Cost (C→B) = 3 Cost (C→D) = 11

| Destination | Distance | Next hop |
|-------------|----------|----------|
| A | | |
| B | | |
| C | 0 | C |
| D | | |

New Routing Table at Router C

- 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.

Updating "C Routing Table"



C Routing Table

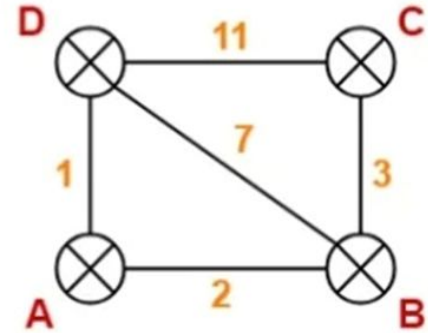
| Net Id | Cost | Next Hop |
|--------|----------|----------|
| A | ∞ | - |
| B | 3 | B |
| C | 0 | C |
| D | 11 | D |

New C Routing Table

| Net Id | Cost | 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 D.
- Router D prepares a new routing table



| From A | | From B | | From C | |
|--------|----------|--------|---|--------|----------|
| A | 0 | A | 2 | A | ∞ |
| B | 2 | B | 0 | B | 3 |
| C | ∞ | C | 3 | C | 0 |
| D | 1 | D | 7 | D | 11 |

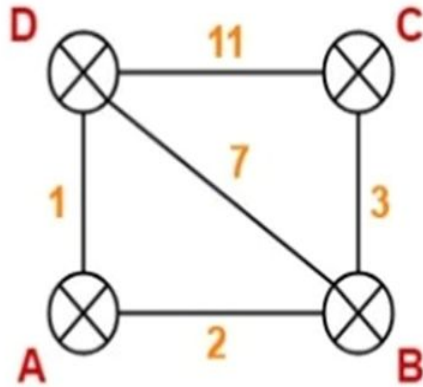
Cost (D→A) = 1 Cost (D→B) = 7 Cost (D→C) = 11

| Destination | Distance | Next hop |
|-------------|----------|----------|
| A | | |
| B | | |
| C | | |
| D | 0 | D |

New Routing Table at Router D

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

Updating "D Routing Table"



D Routing Table

| Net Id | Cost | Next Hop |
|--------|------|----------|
| A | 1 | A |
| B | 7 | B |
| C | 11 | C |
| D | 0 | D |

New D Routing Table

| Net Id | Cost | Next Hop |
|--------|------|----------|
| A | 1 | A |
| B | 3 | A |
| C | 10 | B |
| D | 0 | D |

New Routing Tables:

Router A

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

Router B

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

Router C

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

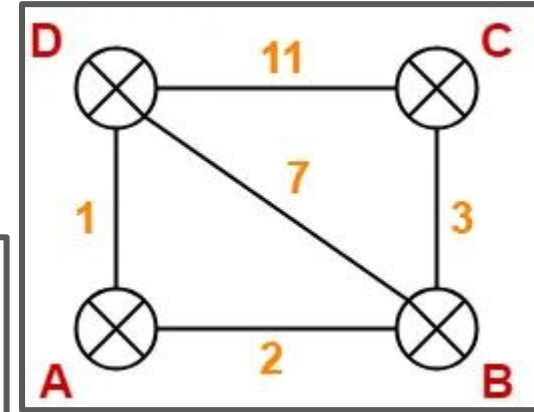
Router D

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

"Still, the routing tables don't produce an optimal path. So, repeat the process, i.e., each router exchanges its tables until the table gives an optimal path."

At Router A-

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



| From B | | From D | | | | |
|--------|---|--------|----|-------------|----------|----------|
| A | 2 | A | 1 | Destination | Distance | Next hop |
| B | 0 | B | 3 | A | 0 | A |
| C | 3 | C | 10 | B | | |
| D | 3 | D | 0 | C | | |
| | | | | D | | |

Cost(A→B) = 2 Cost(A→D) = 1

New Routing Table at Router A

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

The new routing table **at Router A** is-

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

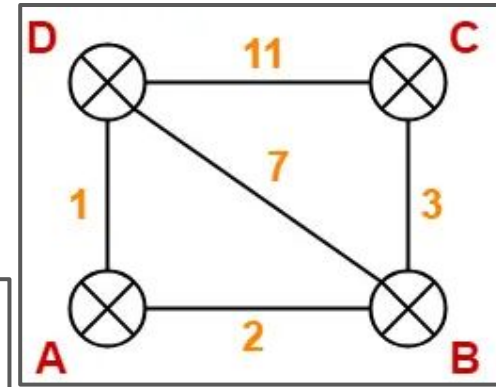
At Router B-

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

| From A | | From C | | From D | | | | |
|----------------|---|----------------|----|----------------|----|--|--|--|
| A | 0 | A | 5 | A | 1 | | | |
| B | 2 | B | 3 | B | 3 | | | |
| C | 5 | C | 0 | C | 10 | | | |
| D | 1 | D | 10 | D | 0 | | | |
| Cost (B→A) = 2 | | Cost (B→C) = 3 | | Cost (B→D) = 3 | | | | |

| Destination | Distance | Next hop |
|-------------|----------|----------|
| A | | |
| B | 0 | B |
| C | | |
| D | | |

New Routing Table at Router B



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

The new routing table **at Router B** is-

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

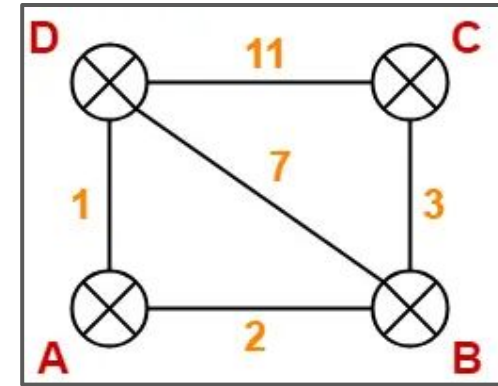
At Router C-

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

| From B | | From D | | | | |
|--------|---|--------|----|-------------|----------|----------|
| A | 2 | A | 1 | Destination | Distance | Next hop |
| B | 0 | B | 3 | A | | |
| C | 3 | C | 10 | B | | |
| D | 3 | D | 0 | C | 0 | C |
| | | | | D | | |

Cost (C→B) = 3 Cost (C→D) = 10

New Routing Table at Router C



- Cost of reaching destination A from router C = $\min \{ 3+2, 10+1 \} = 5$ via B.
- Cost of reaching destination B from router C = $\min \{ 3+0, 10+3 \} = 3$ via B.
- Cost of reaching destination D from router C = $\min \{ 3+3, 10+0 \} = 6$ via B.

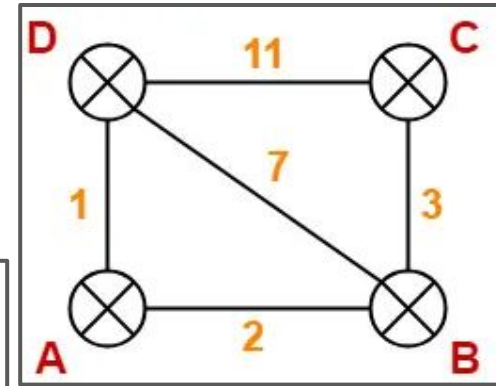
The new routing table **at Router C** is-

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

At Router D-

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

| From A | | From B | | From C | | | | |
|----------------|---|----------------|---|-----------------|----|-------------------------------|----------|----------|
| A | 0 | A | 2 | A | 5 | Destination | Distance | Next hop |
| B | 2 | B | 0 | B | 3 | A | | |
| C | 5 | C | 3 | C | 0 | B | | |
| D | 1 | D | 3 | D | 10 | C | | |
| | | | | | | D | 0 | D |
| Cost (D→A) = 1 | | Cost (D→B) = 3 | | Cost (D→C) = 10 | | New Routing Table at Router D | | |



- Cost of reaching destination A from router D = $\min \{ 1+0, 3+2, 10+5 \} = 1$ via A.
- Cost of reaching destination B from router D = $\min \{ 1+2, 3+0, 10+3 \} = 3$ via A.
- Cost of reaching destination C from router D = $\min \{ 1+5, 3+3, 10+0 \} = 6$ via A.

The new routing table **at Router D** is-

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

These will be the final routing tables at each router.

Router A

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

Router B

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

Router C

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

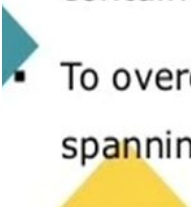
Router D

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

Non- Adaptive Routing Algorithm



Flooding

- In case of flooding, every incoming packet is sent to all the outgoing links except the one from it has been reached.
 - The disadvantage of flooding is that node may contain several copies of a particular packet.
 - To overcome this problem, sequence numbers, spanning tree & hop count are used.
- 

Random Walks



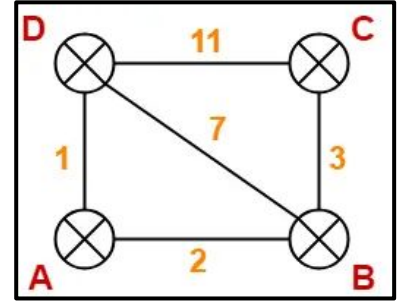
- In this type of algorithm, data packets are transmitted through the node by node or host by host randomly to one of its neighbors.
- This method is extremely strong which is frequently executed by transmitting data packets over the network link which is queued least.

Flooding

“Flooding is a **simple broadcasting technique** where a network node sends a message to all of its neighbors *without considering the topology or routing paths.*”

Flooding:

- Flooding requires **no information** like **topology, load condition, cost of different paths**.
- Every incoming packet to a node is sent out on every outgoing line except the one it arrived on.
- For example in the above figure
 - Incoming packet to [A] is sent out to [B], [D]
 - From [B] is sent to [C],[D] and from [D] it is sent to [A], [C].



Limitations:

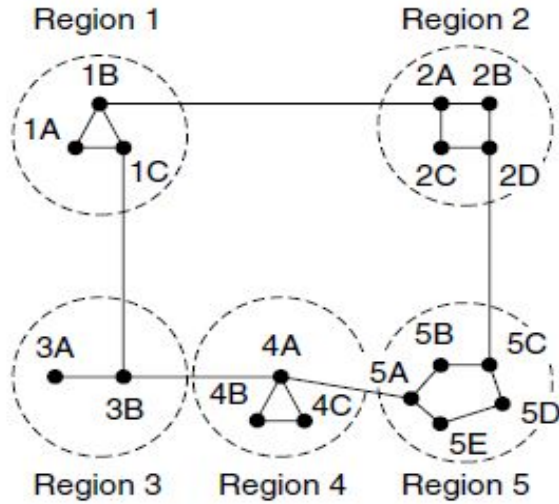
- Flooding generates **vast number of duplicate packets**.
- **Network Congestion:** Flooding causes a lot of unnecessary traffic since messages are sent to all nodes in the network, reduces overall network performance.

Advantages:

- **Reliability:** Flooding ensures that a message reaches all nodes in the network, making it highly reliable.
- **Simple Implementation:** Flooding is straightforward to implement and **does not require complex routing tables or algorithms.**
- **Fault Tolerance:** Because flooding sends messages to all nodes, it is inherently fault-tolerant. **Even if some nodes or links in the network fail, the message can still reach other nodes.**
- **Limited Network Size:** Flooding can be efficient in **very small networks** where the number of nodes is manageable.

Hierarchical Routing Algorithm

- **As networks grow in size, the router routing tables grow proportionally.**
- Not only is router memory consumed by ever-increasing tables, but more CPU time is needed to scan them and more bandwidth is needed to send status reports about them.
- At a certain point, the network may grow to the point where it is no longer feasible for every router to have an entry for every other router, so the routing will have to be done hierarchically.
- **When hierarchical routing is used**, the routers are divided into what we will call **regions**.
- Each router knows all the details about how to route packets to destinations within its own region but knows nothing about the internal structure of other regions.
- For huge networks, a two-level hierarchy may be insufficient; it may be necessary to group the **regions into clusters, the clusters into zones, the zones into groups**, and so on.



(a)

Full table for 1A

| Dest. | Line | Hops |
|-------|------|------|
| 1A | — | — |
| 1B | 1B | 1 |
| 1C | 1C | 1 |
| 2A | 1B | 2 |
| 2B | 1B | 3 |
| 2C | 1B | 3 |
| 2D | 1B | 4 |
| 3A | 1C | 3 |
| 3B | 1C | 2 |
| 4A | 1C | 3 |
| 4B | 1C | 4 |
| 4C | 1C | 4 |
| 5A | 1C | 4 |
| 5B | 1C | 5 |
| 5C | 1B | 5 |
| 5D | 1C | 6 |
| 5E | 1C | 5 |

(b)

Hierarchical table for 1A

| Dest. | Line | Hops |
|-------|------|------|
| 1A | — | — |
| 1B | 1B | 1 |
| 1C | 1C | 1 |
| 2 | 1B | 2 |
| 3 | 1C | 2 |
| 4 | 1C | 3 |
| 5 | 1C | 4 |

(c)

*Autonomous System (AS) ?

* Gateway Router ?

Figure 5-14. Hierarchical routing.

- Figure 5-14 gives a quantitative example of routing in **a two-level hierarchy** with **five regions**.
- The full routing table for router **1A has 17 entries**, as shown in Fig. 5-14(b).
- When routing is done hierarchically, as in Fig. 5-14(c), there are entries for all the local routers, as before, but all other regions are condensed into a single router, so all traffic for region 2 goes via the 1B-2A line, but the rest of the remote traffic goes via the 1C-3B line.
- **Hierarchical routing has reduced the table from 17 to 7 entries.**
- As the ratio of the number of regions to the number of routers per region grows, the savings in table space increase.

- An **Autonomous System (AS)** is a collection of routers whose prefixes and routing policies are **under common administrative control**. This could be a network service provider, a large company, a university.
- All the routers which present in the **Autonomous System** will follow the same routing algorithm.
- Autonomous routing protocols can be categorized into **two main types** based on their **scope and purpose** within an Autonomous System (AS) or between ASes.
 - **Intra-AS Routing Protocol (Interior Routing Protocols)**
 - **Inter-AS Routing Protocol (Exterior Routing Protocols (EGPs))**

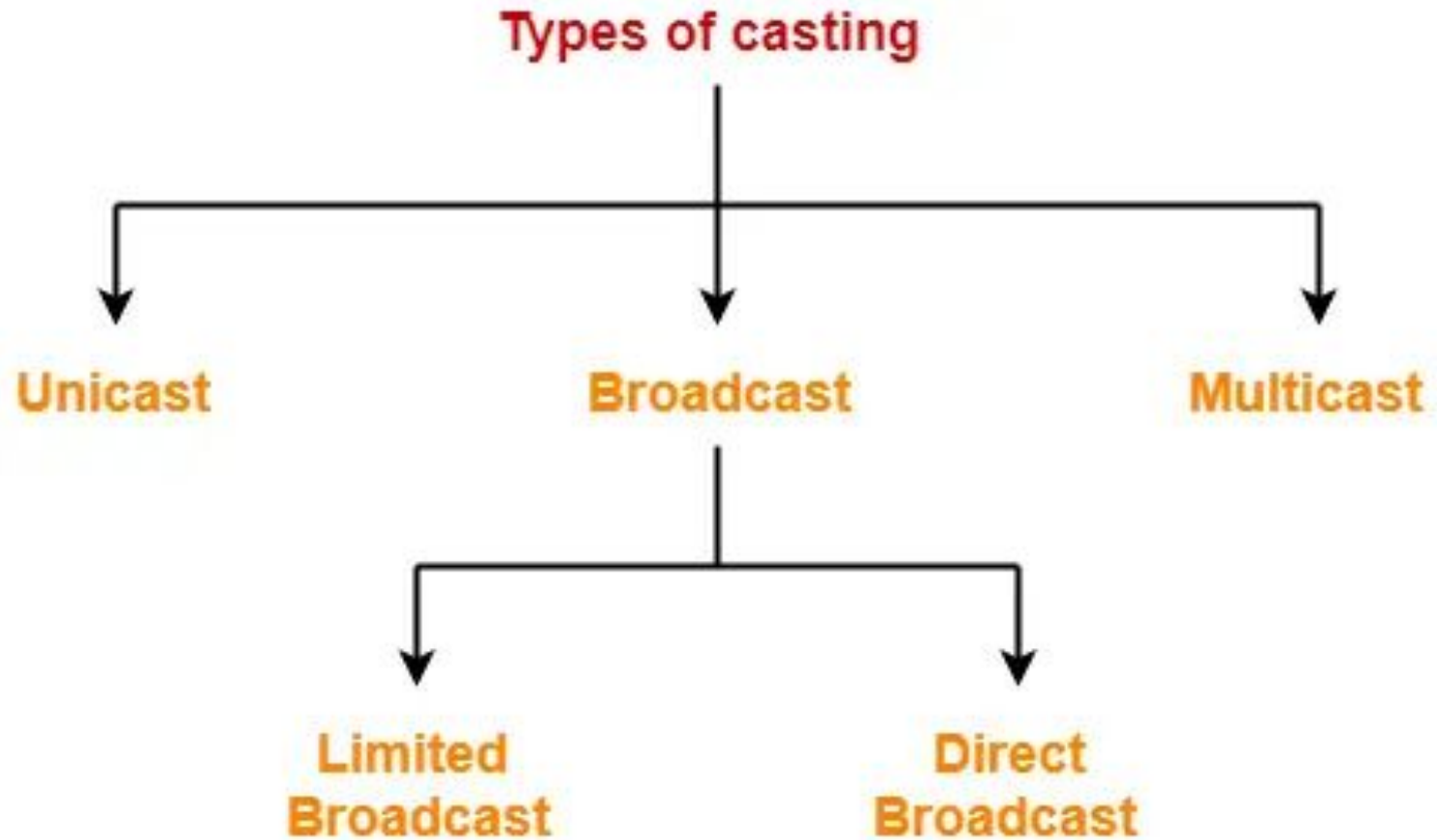
Intra-AS Routing Protocol (Interior Routing Protocols):

- Interior Routing Protocols are used for **routing within a single Autonomous System (AS)**.
- They determine how traffic is routed within the boundaries of the AS.
- **Eg: Routing Information Protocol (RIP), Open Shortest Path First (OSPF)**

Inter-AS Routing Protocol (Exterior Routing Protocols (EGPs) :

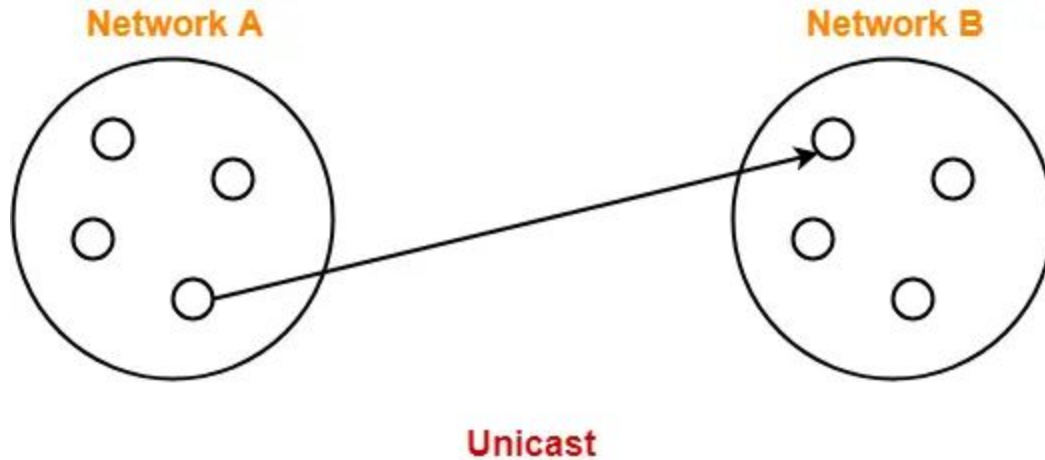
- Exterior Routing Protocols are used for **routing between different Autonomous Systems (ASes)**.
- They are responsible for inter-domain routing, facilitating the exchange of routing information .
- **Eg: Border Gateway Protocol (BGP) , Exterior Gateway Protocol (EGP)**

Broadcast and Multicast Routing



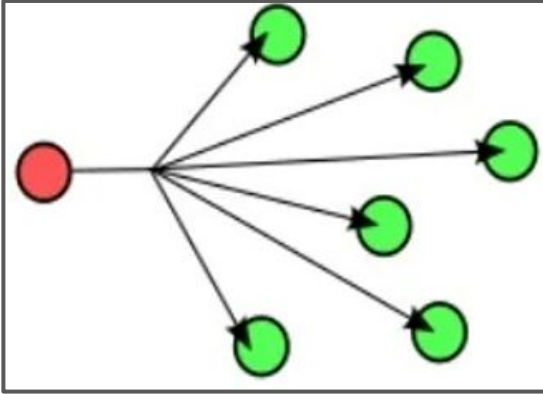
Unicast:

- Transmitting data from **one source host** to **one destination host** is called as unicast.
- It is a **one to one transmission**

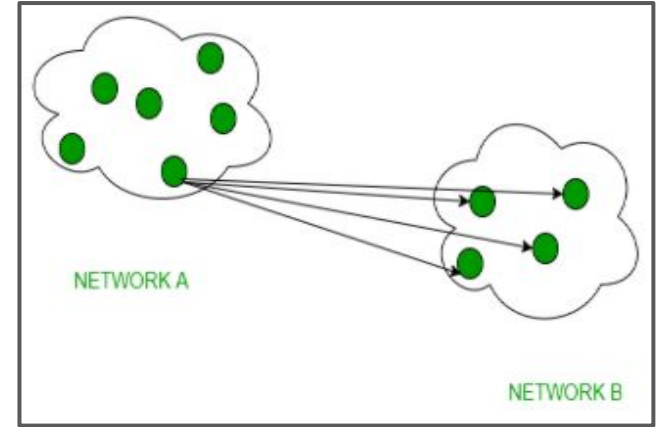


Broadcast Routing:

[1]



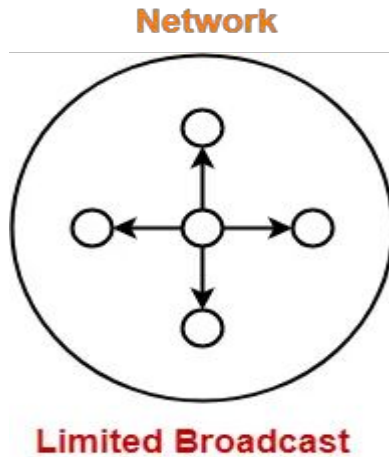
[2]



- In broadcast routing, **packets are sent to all nodes even if they do not want it.**
- Broadcast transfer uses **one-to-all** technique and can be classified into two types :
 - **Limited Broadcasting**
 - **Direct Broadcasting**
- In broadcasting mode, transmission happens from one host to all the other hosts connected on the LAN.

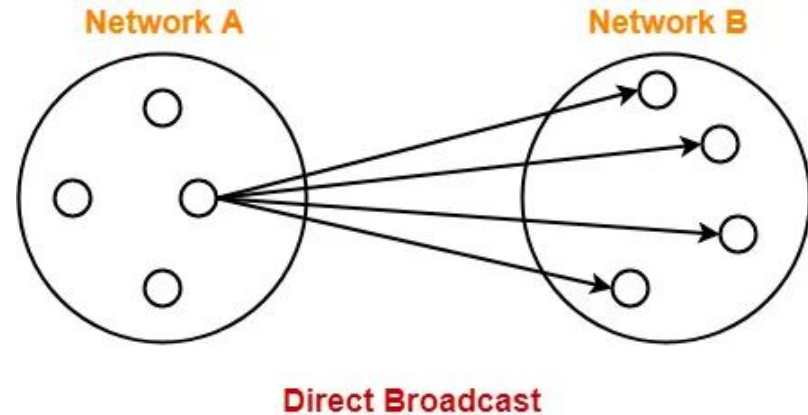
Limited Broadcasting:

- Transmitting data from one source host to all other hosts **residing in the same network** is called as limited broadcast.



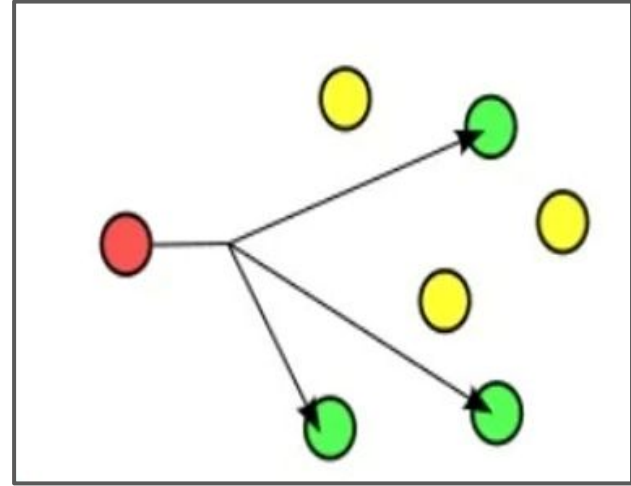
Directed Broadcasting:

- Transmitting data from one source host to all other hosts **residing in some other network** is called as direct broadcast.



Multicast Routing

- In Multicast routing, **the data is sent to only nodes which wants to receive the packets.**
- Transmitting data from one source host to a particular group of hosts having interest in receiving the data is called as multicast.
- Multicast transfer uses **one-to-many** technique.



Broadcast Routing vs Multicast Routing

| Broadcast | Multicast |
|---|--|
| It scale well across large networks. | It does not scale well across large networks. |
| Its bandwidth is wasted. | It utilizes bandwidth efficiently. |
| It has one-to-all mapping. | It has one-to-many mapping. |
| Hub is an example of a broadcast device. | Switch is an example of a multicast device. |
| It works on star and bus topology | It works on star, mesh, tree and hybrid topology. |

Congestion Control Algorithms

- Congestion in a computer network refers to **a situation in which there is an excessive amount of data traffic on a network than it can effectively handle**, which leads to
 - Performance degradation
 - Packet loss
 - Reduced Quality of Service
 - Network instability
- It's a common problem in networks, particularly in situations where **the demand for network resources exceeds their availability or capacity**.
- Congestion can occur in both **wired and wireless networks** and can have various causes and consequences.

Congestion-The situation in which too many packets are present in the subnet.

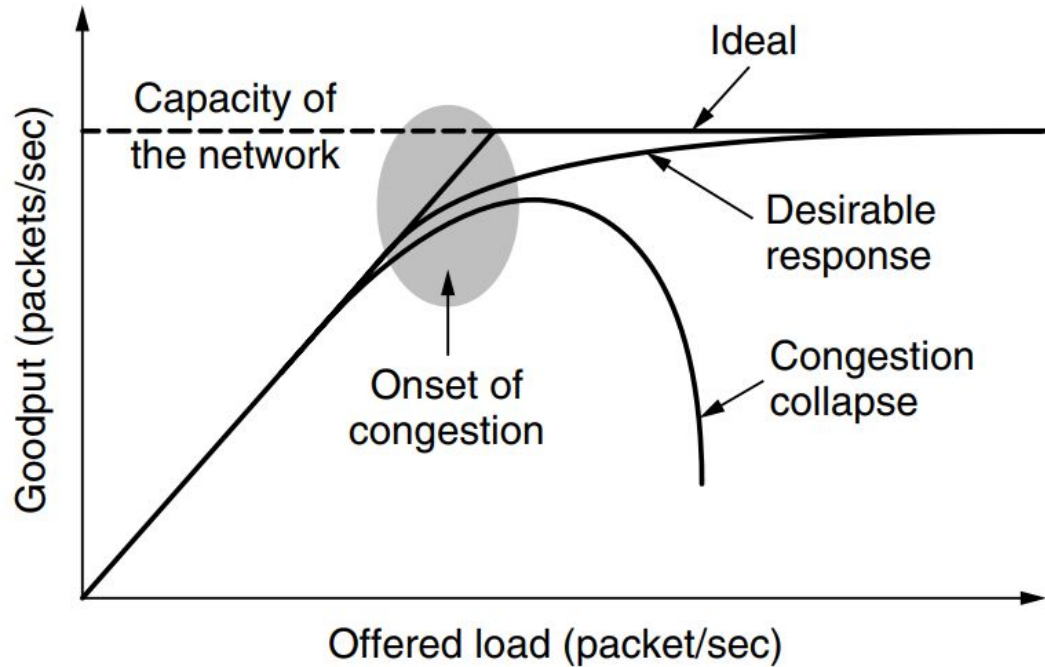


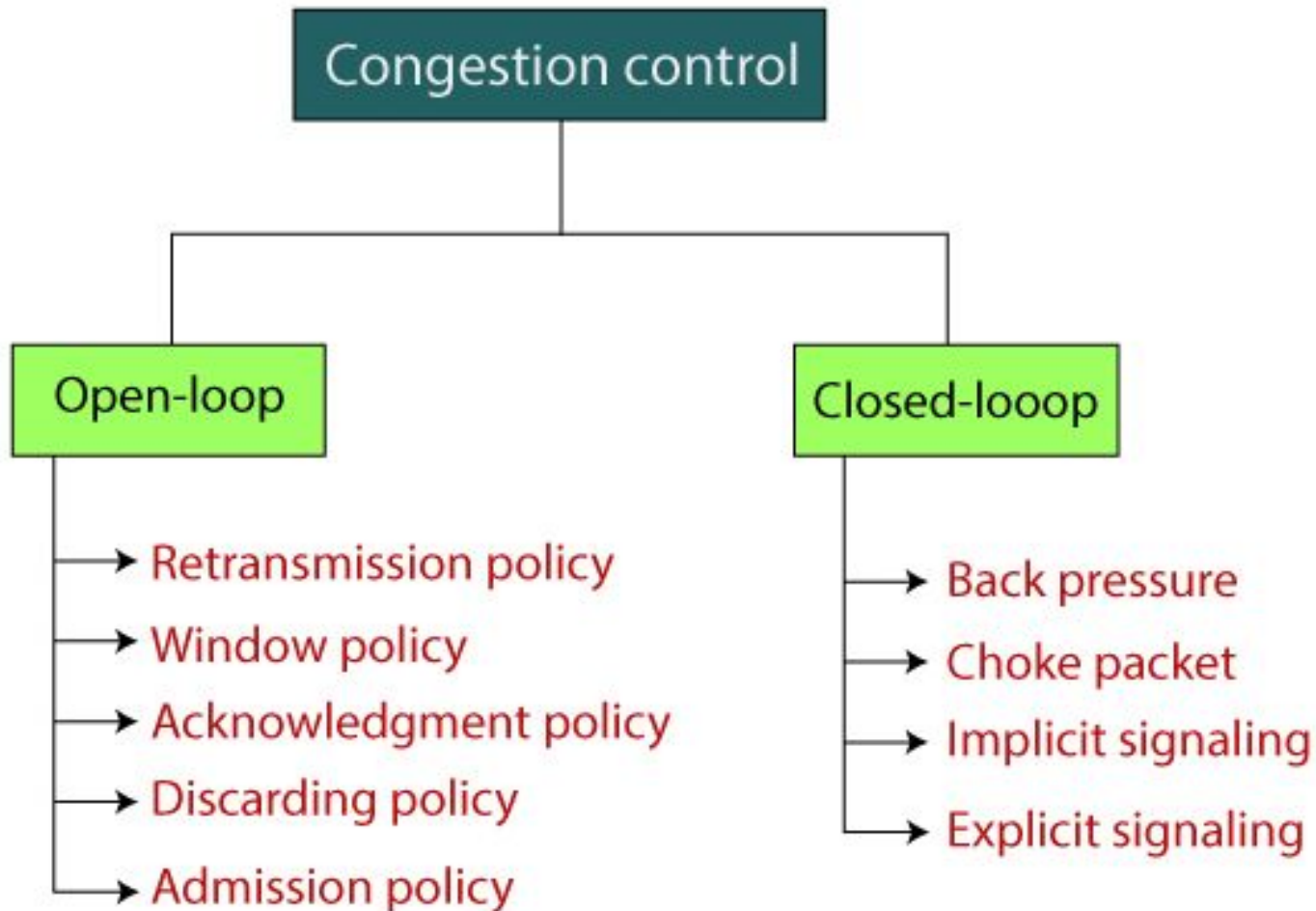
Figure 5-21. With too much traffic, performance drops sharply.

- Figure 5-21 depicts the onset of congestion.
- When the number of packets hosts send into the network is well **within its carrying capacity**, **the number delivered is proportional to the number sent**.
- If twice as many are sent, twice as many are delivered.
- However, as **the offered load approaches the carrying capacity**, bursts of traffic occasionally fill up the buffers inside routers and **some packets are lost**.
- These lost packets consume some of the capacity, so the number of delivered packets falls below the ideal curve. The network is now **congested**.

Congestion Control Techniques

- Congestion control refers to techniques and mechanisms that can **either prevent congestion**, before it happens, or **remove congestion**, after it has happened.
- In general, we can divide congestion control mechanisms into two broad categories:
 - **Open-loop congestion control (Prevention)**
 - **Closed-loop congestion control (Removal)**

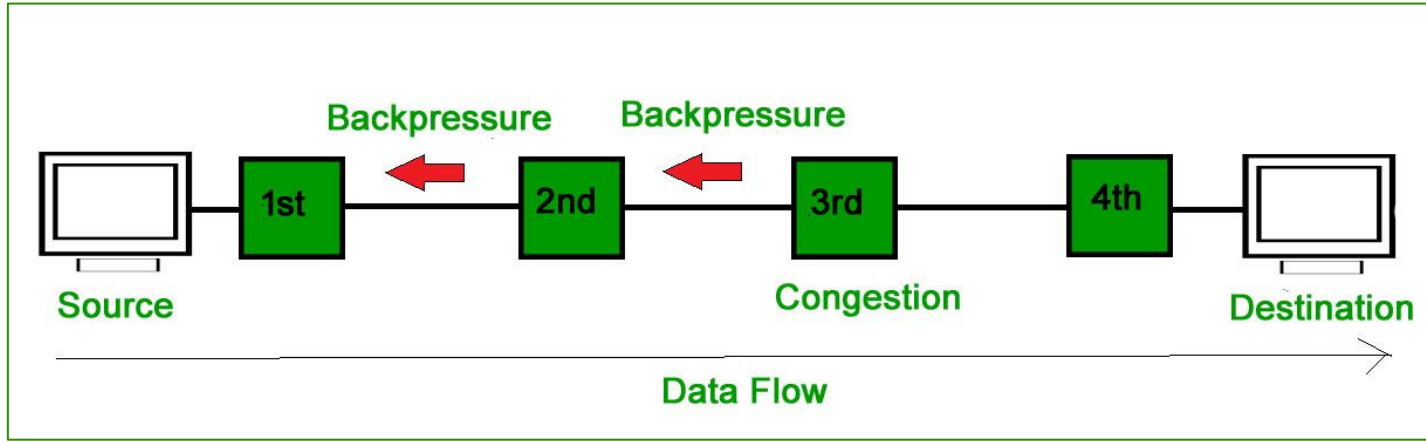
| Open-Loop Congestion Control (Prevention) | Closed-Loop Congestion Control (Removal) |
|---|---|
| <ul style="list-style-type: none"><li data-bbox="253 230 966 601">● Open-loop congestion control focuses on preventing congestion from occurring in the first place by adjusting the traffic before it enters the network.<li data-bbox="253 639 966 934">● This approach is proactive and relies on various techniques to manage the flow of traffic to avoid congestion. | <ul style="list-style-type: none"><li data-bbox="1020 230 1731 601">● Closed-loop congestion control, also known as feedback-based congestion control, focuses on detecting and reacting to congestion after it has occurred.<li data-bbox="1020 639 1731 852">● It dynamically adjusts the network traffic based on real-time feedback to alleviate congestion. |



Warning Bit (or) Back Pressure

- ★ Backpressure is a technique in which **a congested node stops receiving packets from upstream node.**
- ★ This may cause the upstream node or nodes to become congested and reject receiving data from above nodes.
- ★ Backpressure is a node-to-node congestion control technique that propagate in the opposite direction of data flow
- ★ **A special bit in the packet header is set by the router to warn the source when congestion is detected.**
- ★ The bit is copied and piggy-backed on the ACK and sent to the sender.
- ★ The sender monitors the number of ACK packets it receives with the warning bit set and adjusts its transmission rate accordingly.

Warning Bit (or) Back Pressure

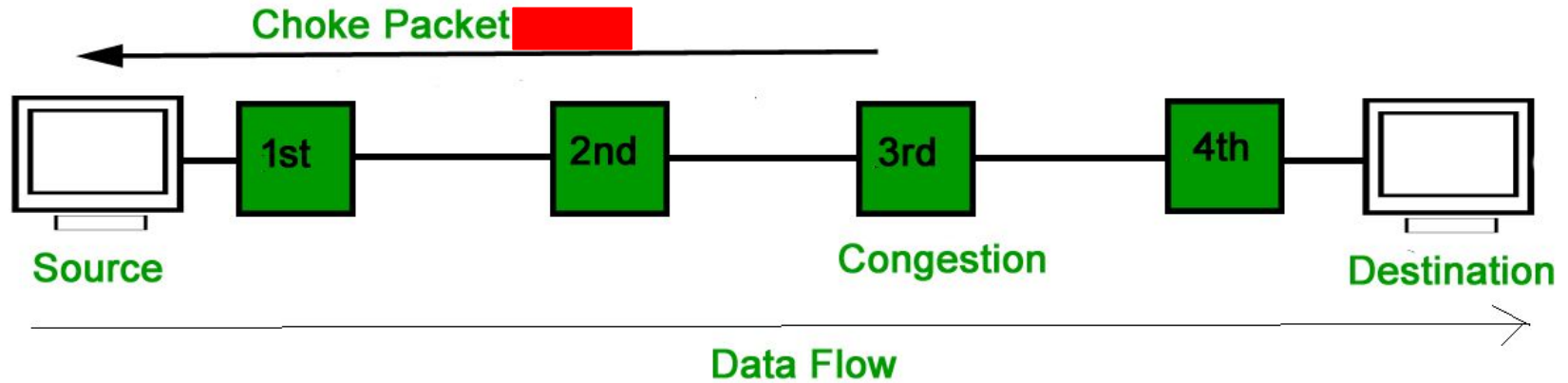


In above diagram the 3rd node is congested and stops receiving packets as a result 2nd node may be get congested due to slowing down of the output data flow. Similarly 1st node may get congested and inform the source to slow down.

Choke Packet Technique

- ★ Choke packet technique is applicable to both **virtual networks as well as datagram subnets**.
- ★ A choke packet is a packet sent by a node to the source to inform it of congestion.
- ★ Each router monitors its resources and the utilization at each of its output lines.
- ★ Whenever the resource utilization exceeds the threshold value which is set by the administrator, the router directly sends a choke packet to the source giving it a feedback to reduce the traffic.
- ★ The intermediate nodes through which the packets has traveled are not warned about congestion.

Choke Packet Technique



Implicit Signaling

- ★ In implicit signaling, **there is no communication between the congested node or nodes and the source.**
- ★ The source guesses that there is congestion somewhere in the network from other symptoms. **For example**, when a source sends several packets and there is **no acknowledgment for a while**, one assumption is that the network is congested.
- ★ The delay in receiving an acknowledgment is interpreted as congestion in the network; **the source should slow down.**

Explicit Signaling

- ★ In explicit signaling, if a node experiences congestion **it can explicitly sends a packet to the source or destination to inform about congestion.**
- ★ In the choke packet method, a separate packet is used for this purpose; in the explicit signaling method, the signal is included in the packets that carry data.

Explicit signaling can occur in either **forward or backward direction.**

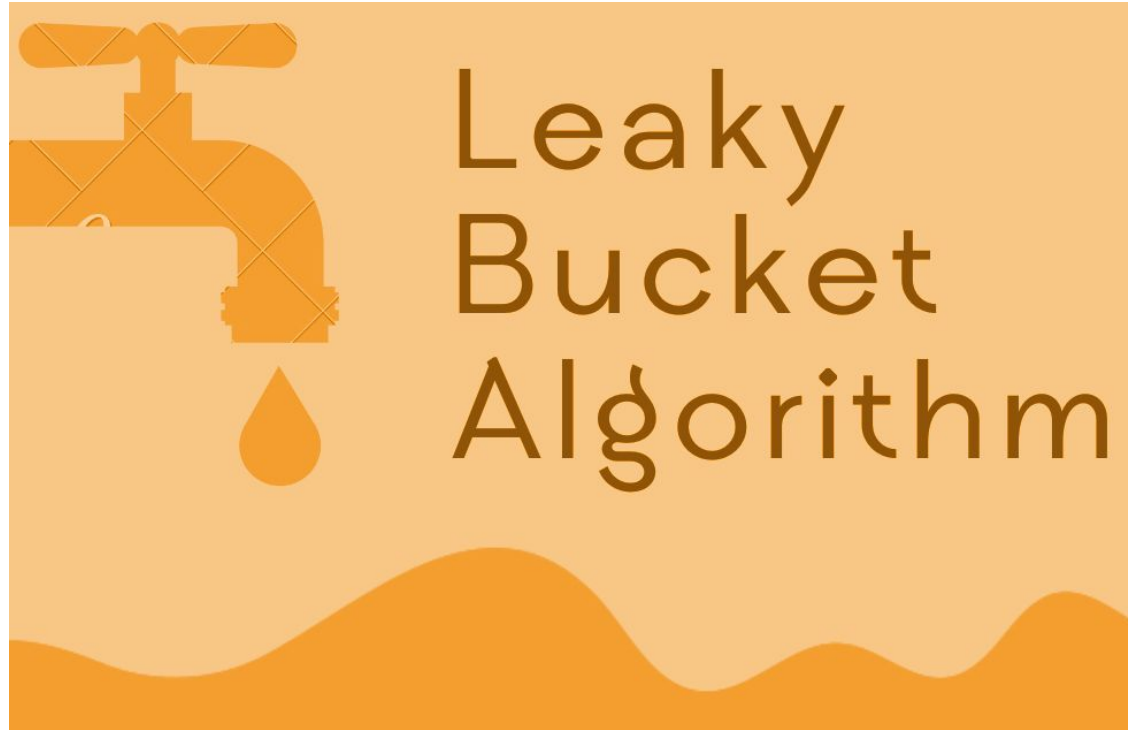
- ★ **Forward Signaling** : In forward signaling, a signal is sent in the direction of the congestion. **The destination is warned about congestion.** The receiver in this case adopt policies to prevent further congestion.
- ★ **Backward Signaling** : In backward signaling, a signal is sent in the opposite direction of the congestion. **The source is warned about congestion** and it needs to slow down.

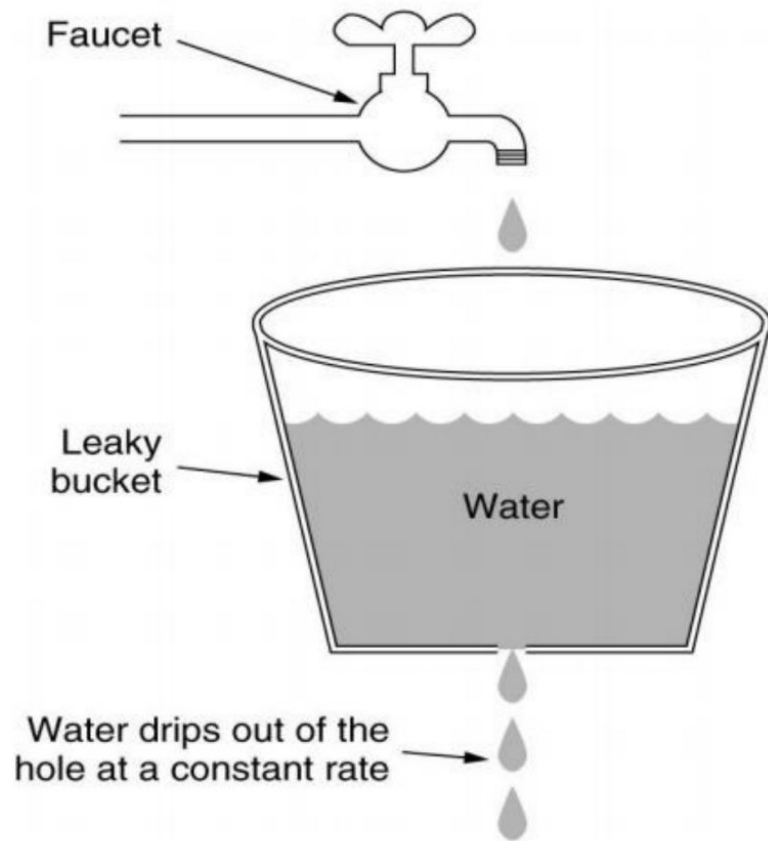
Traffic Shaping

- ★ Another method of congestion control is **to shape the traffic before it enters the network.**
- ★ In the network layer, before the network can make Quality of service guarantees, it must know what traffic is being guaranteed.
- ★ One of the main causes of congestion is that traffic is often bursty.
- ★ **Traffic Shaping** is a mechanism to *control the rate of traffic sent to the network.*
- ★ Traffic shaping **helps to regulate the rate of data transmission** and **reduces congestion.**

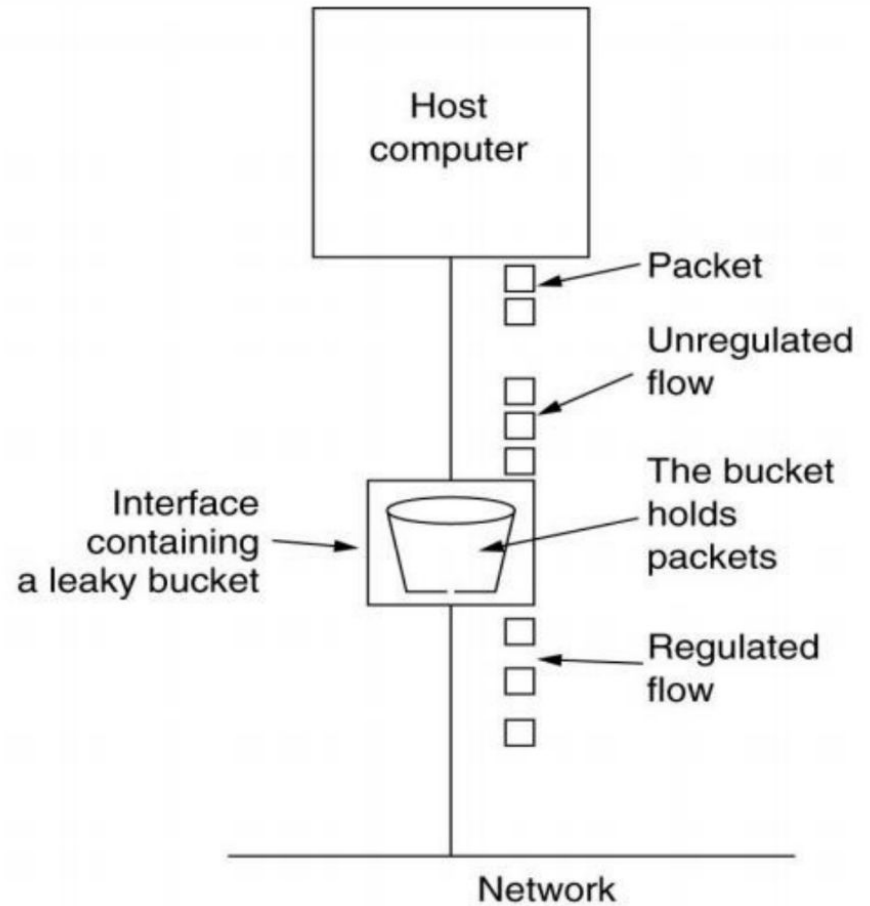
There are 2 types of traffic shaping algorithms:

- **Leaky Bucket**
- **Token Bucket**





(a)



(b)

- ★ The leaky bucket algorithm is ideal for **smoothing out bursty traffic**.
- ★ Just like a hole at the bottom of a water bucket leaks water out at a fixed rate, the leaky bucket algorithm does the same with network traffic.
- ★ Bursty chunks of traffic are stored in a "bucket" with a "hole" and **sent out at a controlled, average rate**.
- ★ The **hole** represents **the network's commitment** to a particular bandwidth.
- ★ The leaky bucket shapes the incoming traffic to ensure it conforms to the commitment.
- ★ Thus, regardless of how much data traffic enters the bucket, it always leaves at a constant output rate (the commitment).
- ★ This mechanism regulates the packet flow in the network and helps to prevent congestion that leads to performance deterioration and traffic delays.

- ★ Suppose data enters the network from various sources at different speeds.
- ★ Consider **one bursty source** that sends data at **20 Mbps for 2 seconds** for a total of **40 Mbps**.
- ★ Then it is silent, sending **no data for 5 seconds**. Then it again transmits data at a rate of **10 Mbps for 3 seconds**, thus sending a total of **30 Mbps**.
- ★ So, in a **time span of 10 seconds** the source sends **70 Mb data**.
- ★ *However, the network has only committed a bandwidth of 5 Mbps for this source.*
- ★ Therefore, it uses **the leaky bucket algorithm** to output traffic at the rate of **5 Mbps** during the same time period of 10 seconds, which smooths out the network traffic.
- ★ Without the leaky bucket algorithm in place, the initial burst of 20 Mbps would have consumed a lot more bandwidth than the network had reserved (committed) for the source, which would have caused **congestion and a slowdown in the network**.

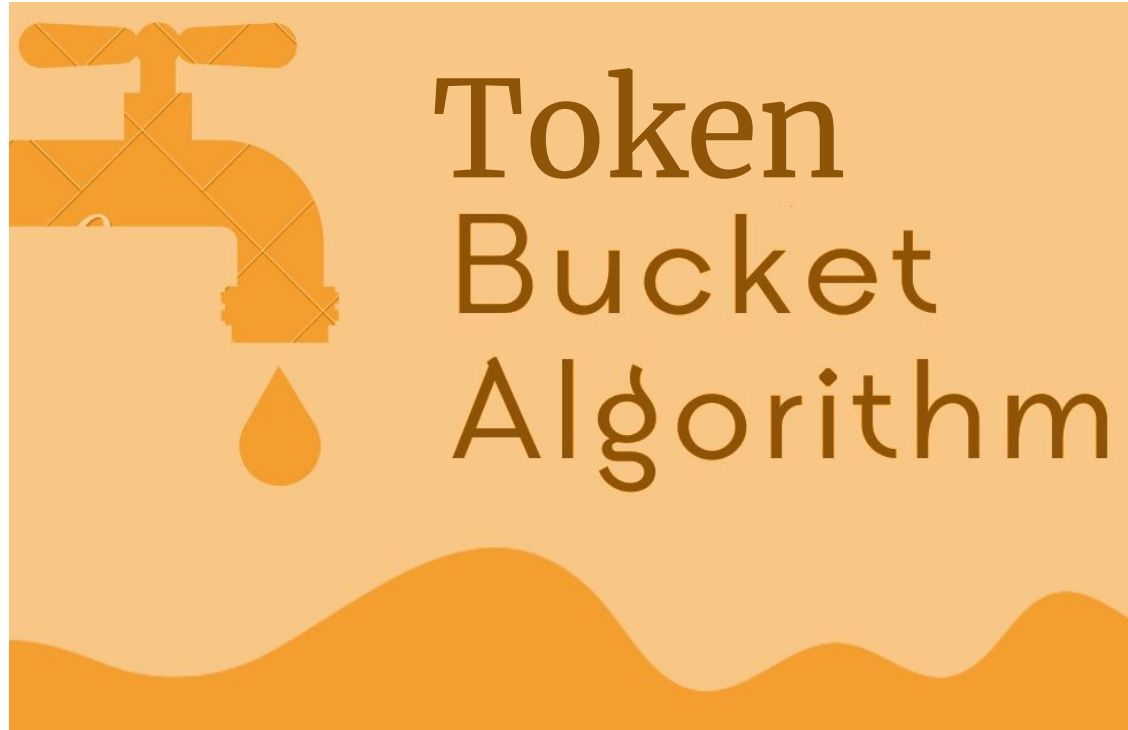
Problems in Leaky bucket Algorithm

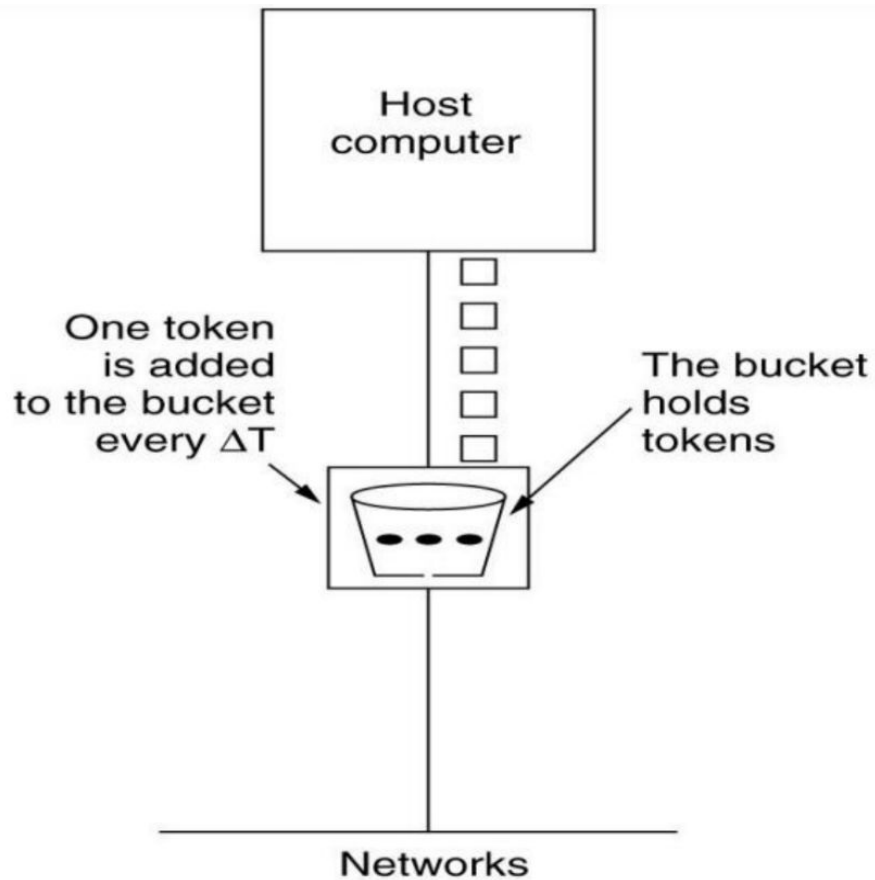
[1] Fixed Bucket Size:

- ★ The leaky bucket algorithm is **primarily designed to regulate the average data rate over time.**
- ★ **It does not handle bursts of traffic well.**
- ★ If traffic arrives in bursts that **exceed the bucket's capacity**, it may result in **packet loss.**

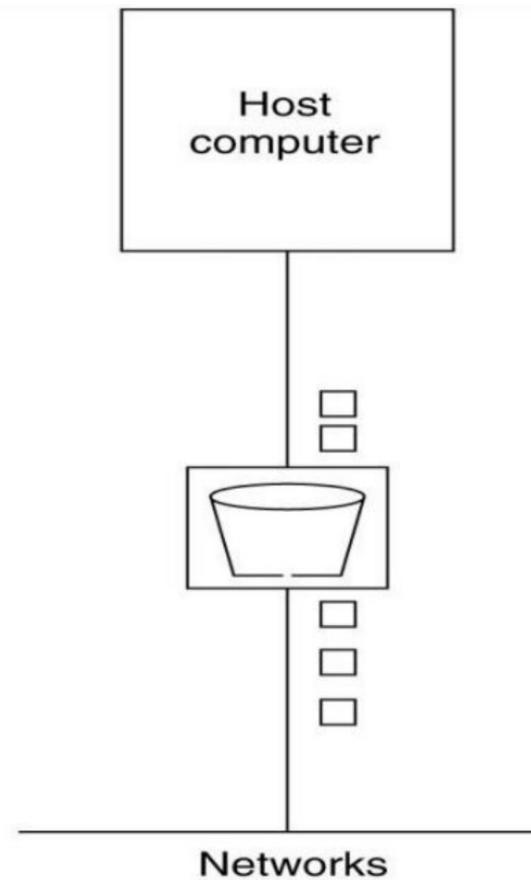
[2] Not Suitable for Real-Time Traffic:

- ★ For real-time applications like **voice and video streaming**, the fixed-rate nature of the leaky bucket algorithm may not be ideal. These applications often require strict Quality of Service (QoS) guarantees, which the leaky bucket cannot provide on its own.





(a)



(b)

Token bucket Algorithm

The token bucket algorithm works as follows:

- ★ **Token Generation:** A bucket is used to store tokens, which represent the permission to transmit a packet or request. Tokens are generated at a fixed rate and are added to the bucket at regular intervals. The rate at which tokens are added to the bucket is often referred to as the "token generation rate" or "token arrival rate."
- ★ **Token Consumption:** When a packet or request needs to be transmitted, it must first obtain a token from the bucket. If there are tokens available in the bucket, it is allowed to proceed. If there are no tokens available, the packet or request must wait until tokens become available.
- ★ **Bucket Capacity:** The bucket has a maximum capacity, and tokens cannot accumulate beyond this capacity. If the bucket is already full and new tokens arrive, they are simply discarded.

IP Address

IP address:

- ★ An IP address is **a unique address** that identifies a device on the internet or a local network.
- ★ An IP address, or Internet Protocol address, is **a numerical label** assigned to each device connected to a computer network that uses the Internet Protocol for communication.
- ★ **It serves two main purposes:**
 - *Identifying the host*
 - *Providing the location of the host in the network*
- ★ Computers that communicate over the internet or via local networks share information to a specific location using IP addresses.
- ★ The internet needs a way to differentiate between different computers, routers, and websites.

Here's a simplified explanation of how IP addresses work with a real-time example:

Device Connection:

- ★ When you connect your device (such as a computer, smartphone, or tablet) to the internet, it is assigned an IP address. This can happen through various means, such as a local router assigning a private IP address or your Internet Service Provider (ISP) assigning a public IP address.

Sending Data:

- ★ When you want to send or receive data over the internet, your device breaks down the data into packets. Each packet is a unit of data that includes the sender's IP address, the recipient's IP address, and the actual data being sent.

Here's a simplified explanation of how IP addresses work with a real-time example:

Routing:

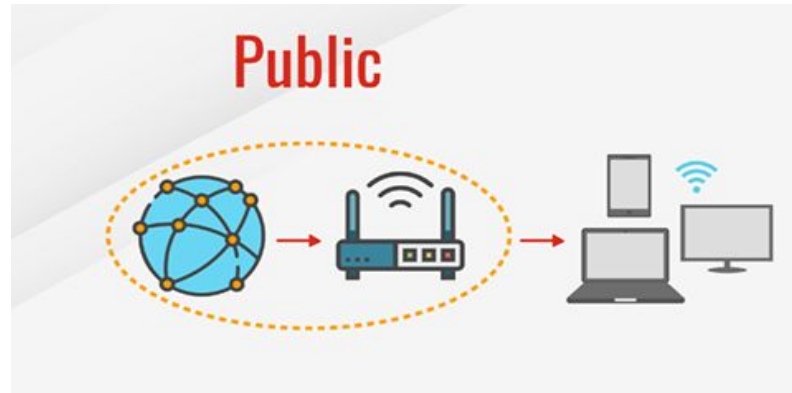
- ★ The packets travel through various routers and switches on the internet to reach their destination. Routers use the IP addresses to determine the best path for each packet to take to reach its destination.

Receiving Data:

- ★ When the packets arrive at their destination, the recipient's device uses its IP address to reassemble the packets into the original data.

Public IP Address:

- ★ A public IP address is an address assigned to your device by your **Internet Service Provider (ISP)** and is **visible** on the internet.
- ★ It uniquely identifies your device on **the global network**.
- ★ Public IP addresses are used for communication between devices on the internet.

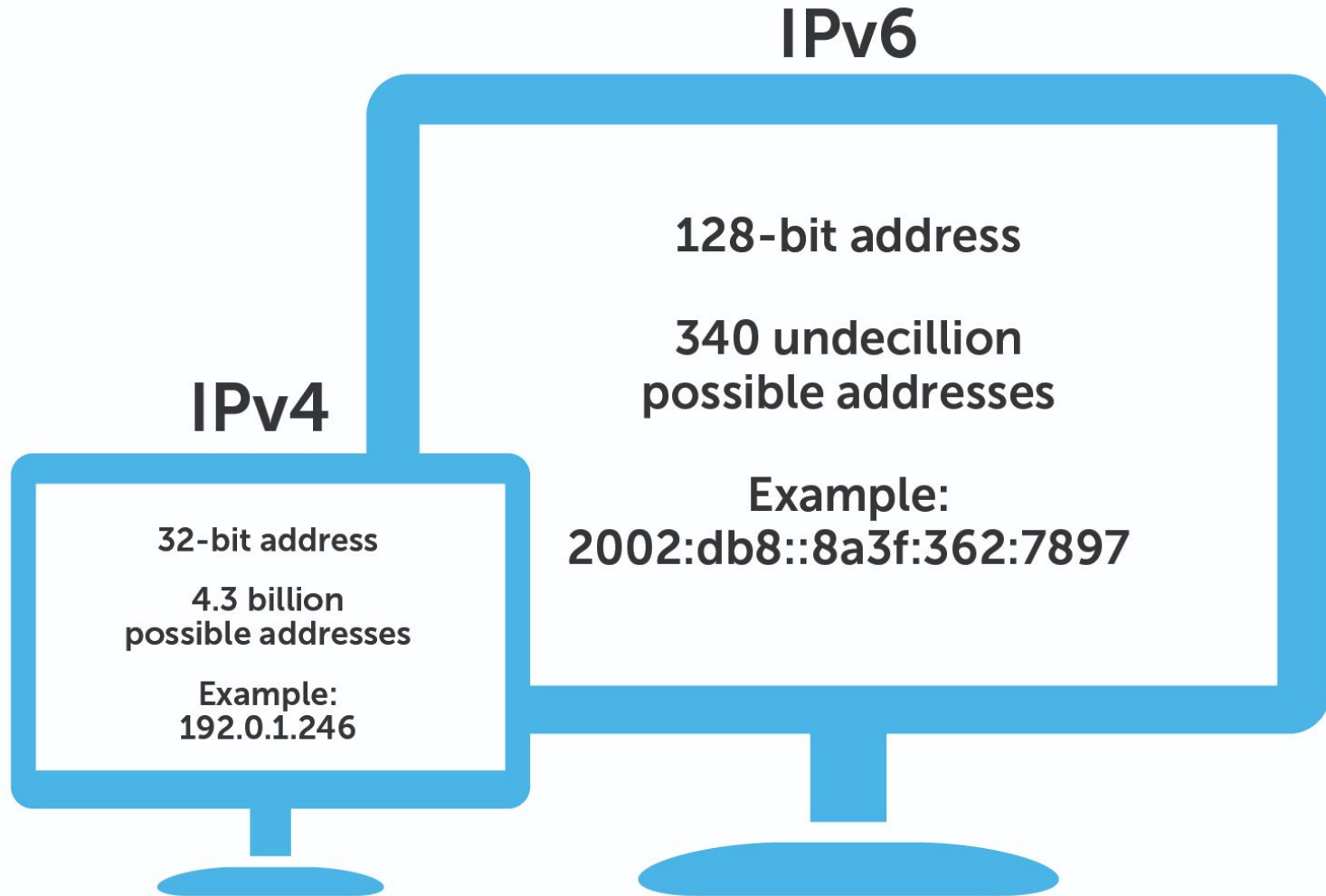


Private IP Address:

- ★ Devices within your home network are assigned private IP addresses. These addresses are not directly accessible from the internet.
- ★ Example: Your computer might have a private IP address like 192.168.1.2, your smartphone might be assigned 192.168.1.3, and so on.
- ★ Devices communicate with each other using these private IP addresses within the local network, but when they access the internet, the router translates their requests into the public IP address.



Types of IP Address

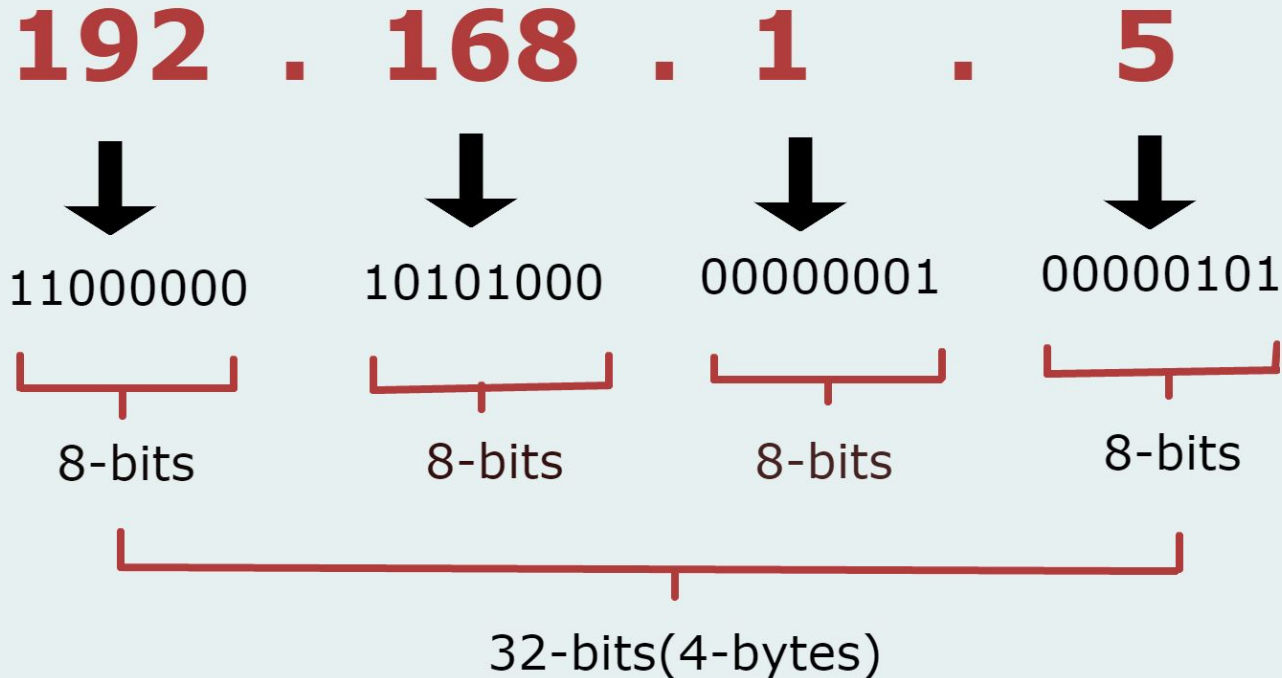


IPV4 Address

- ★ IPv4 stands for **Internet Protocol version four**, It is **a current version** and the most commonly used IP address.
- ★ It was introduced in 1981 by DARPA and was the first deployed version in 1982 for production on SATNET and on the ARPANET in January 1983.
- ★ IPv4 addresses are **32-bit integers** that have to be expressed in **Dotted Decimal Notation**.
- ★ It is represented by **4 numbers** separated by **dots** in the range of **0-255**, which have to be converted to 0 and 1, to be understood by Computers.
 - For Example, An IPv4 Address can be written as **189.123.123.90**.
- ★ **Each number in an octet** is in the range from **0-255**. This address can produce 4,294,967,296 possible unique addresses.i.e **4.29×10^9 address space**

IPV4 Address Format

IPV4 address represented in dotted-decimal notation



IPV6 Address

- ★ IPv6 is based on IPv6 and stands for **Internet Protocol version 6**.
- ★ It was first introduced in **December 1995** by **Internet Engineering Task Force**.
- ★ IP version 6 is the new version of Internet Protocol, which is way better than IP version 4 in terms of **complexity and efficiency**.
- ★ IPv6 is a **128-bit hexadecimal address**, which is written as a group of **8 hexadecimal numbers separated by colon (:)**.
- ★ This **hexadecimal address** contains both **numbers and alphabets**.
- ★ Due to the usage of both the numbers and alphabets, IPv6 is capable of producing over **340 undecillion (3.4×10^{38}) addresses**.
- ★ IPv6 provides a **large address space**, and it contains a **simple header** as compared to IPv4.

IPV6 Address Format

- IPv6 is a **128-bit hexadecimal address** made up of **8 sets of 16 bits each**, and these 8 sets are separated by a colon.
- In IPv6, each hexadecimal character represents 4 bits. So, we need to convert 4 bits to a hexadecimal number at a time.

2001:23ab:7612:0000:0000:aaaa:ac61:fde2

16 bit 16 bit 16 bit 16 bit 16 bit 16 bit 16 bit 16 bit

Difference Between IPv4 and IPv6

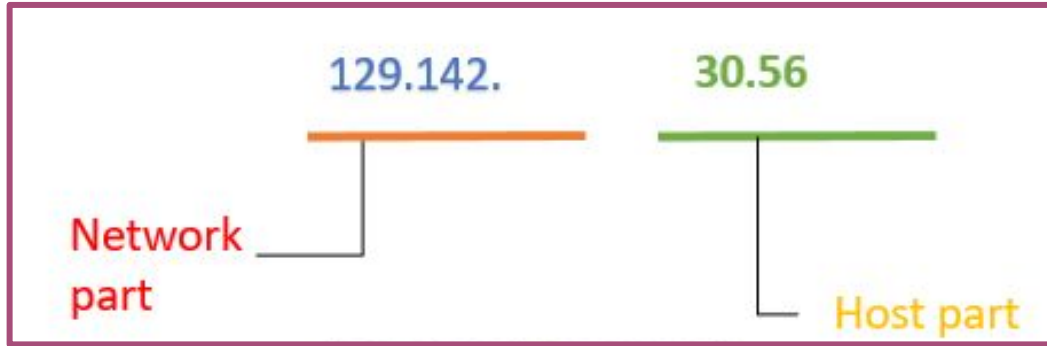
| IPV4 | IPV6 |
|--|--|
| IPv4 has a 32-bit address length | IPv6 has a 128-bit address length |
| It Supports Manual and DHCP address configuration | It supports Auto and renumbering address configuration |
| It can generate 4.29×10^9 address space | The address space of IPv6 is quite large it can produce 3.4×10^{38} address space |
| Address representation of IPv4 is in decimal | Address Representation of IPv6 is in hexadecimal |
| The Security feature is dependent on the application | IPSEC is an inbuilt security feature in the IPv6 protocol |
| In IPv4 Encryption and Authentication facility not provided | In IPv6 Encryption and Authentication are provided |
| IPv4's IP addresses are divided into five different classes. Class A , Class B, Class C, Class D , Class E. | IPv6 does not have any classes of the IP addresses |

Classes of IP Address

IPv4's IP addresses are divided into five different classes. Class A , Class B, Class C, Class D , Class E.

- An IP (Internet Protocol) address is a numerical label assigned to the devices connected to a computer network that uses the IP for communication.
- IP address act as an identifier for a specific machine on a particular network.
- An IP address is a 32-bit unique address.
- The 32-bit IP address is divided into **five sub-classes**
 1. **Class A**
 2. **Class B**
 3. **Class C**
 4. **Class D**
 5. **Class E**

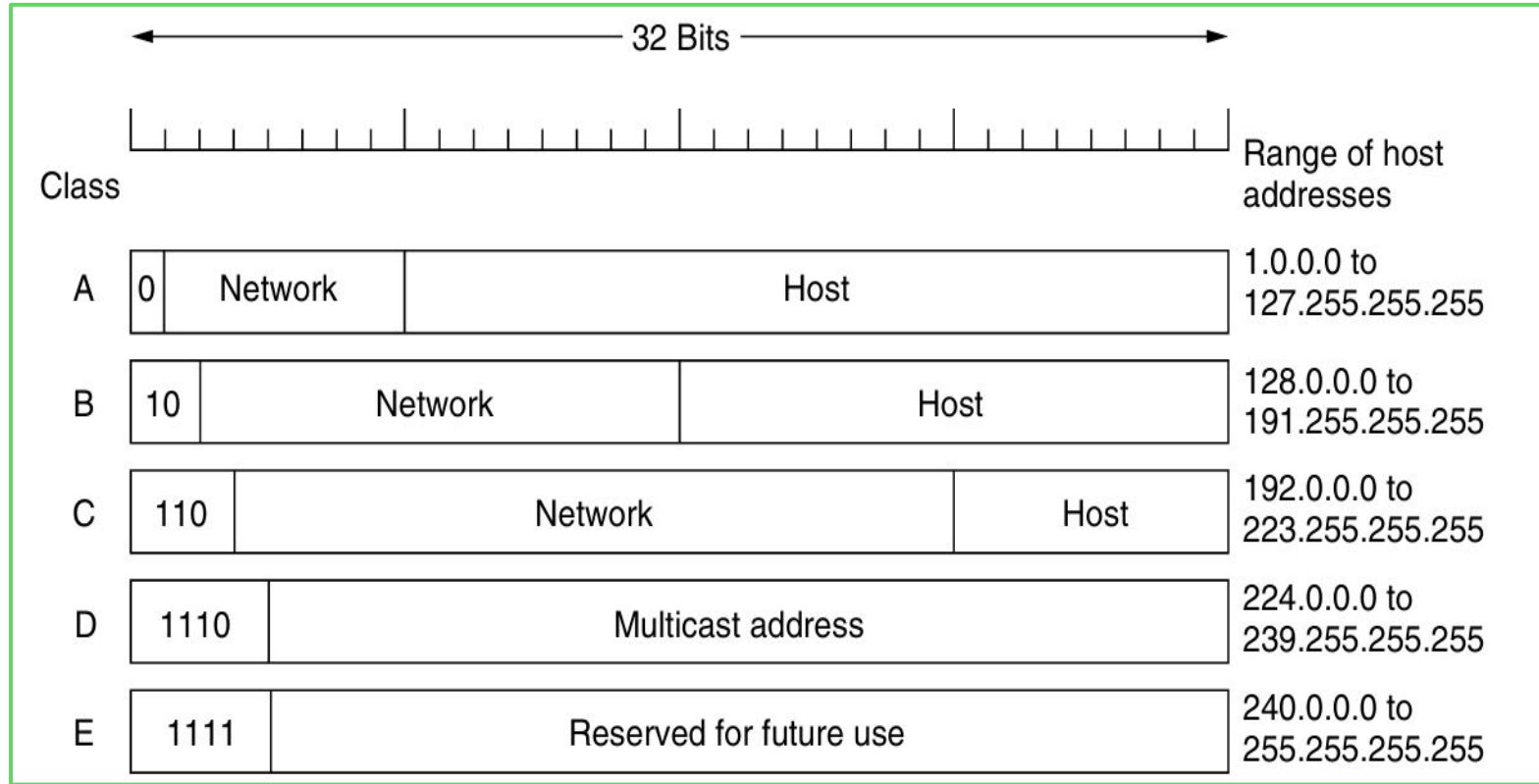
Parts of IP Address



IP Address is divided into two parts:

- **Prefix:** The prefix part of IP address *identifies the physical network to which the computer is attached*. Prefix is also known as **a network address**.
- **Suffix:** The suffix part *identifies the individual computer on the network*. The suffix is also called **the host address**

IP Address Formats



Classes A, B, C offers addresses for networks of three distinct network sizes. Class D is only used for multicast, and class E reserved exclusively for experimental purposes.

Class A Address

- IP addresses belonging to class A are assigned to the networks that contain a large number of hosts.
 - **The network ID is 8 bits long.**
 - **The host ID is 24 bits long.**
- In a Class A type of network,
 - The **higher-order bit** of the first octet in class A is always set to **0**.
 - The **remaining 7 bits** in the first octet are used to **determine network ID**.
 - The **remaining have 24 bits** used to determine the **host in the network**.
- An example of a Class A address is 102.168.212.226. Here, “102” helps you identify the network and 168.212.226 identify the host.

Class B Address

- IP address belonging to class B is assigned to networks that range from medium-sized to large-sized networks.
 - **The network ID is 16 bits long.**
 - **The host ID is 16 bits long.**
- In a Class B type of network,
 - The **higher-order bit** of the first octet in class B is always set to **10**.
 - The **remaining 14 bits** in the first octet are used to **determine network ID**.
 - The **remaining have 16 bits** used to determine the **host in the network**.
- An example of Class B IP address is 168.212.226.204, where "168 212" identifies the network and "226.204" helps you identify the Host.

Class C Address

- IP addresses belonging to class C are assigned to small-sized networks such as home or small business networks
 - **The network ID is 24 bits long.**
 - **The host ID is 8 bits long.**
- In a Class C type of network,
 - The **higher-order bit** of the first octet in class C is always set to **110**.
 - The **remaining 21 bits** in the first octet are used to **determine network ID**.
 - The **remaining have 8 bits** used to determine the **host in the network**.
- An example of Class C IP address is 192.168.178.1, where "192.168.178." identifies the network and "1" helps you identify the Host.

Class D Address

- The **higher-order bits** of the first octet of IP addresses belonging to class D is always set to **1110**.
- Class D address range from 224.0.0.0 to 239.255.255.255.
- **Reserved for multicast groups and is not used for unicast communication.**
- Multicast addresses are used to **send data to multiple hosts simultaneously**.

Class E Address

- The **higher-order bits** of the first octet of class E are always set to **1111**.
- Class E address range from 240.0.0.0 to 255.255.255.255)
- Reserved for **experimental and research purposes**.
- **Not used for general IP communication.**

Activity Time:

GATE

Q1

Identify the **valid and invalid IP address** in the following set,**If invalid** write the reason

- A. 24.25.26.8
- B. 10.3.156.256
- C. 0.0.0.0
- D. 255.255.255.255
- E. 100.2.3.345.456
- F. 16.2e.54.67
- G. 111.064.25.4
- H. 10111010.2.24.36

Activity Time:

Q2

The Dotted Decimal Notation(DDN) format for the given Hexadecimal Notation(HDN) C22F1582 is

- A. 194.50.21.145
- B. 194.47.21.130
- C. 194.45.21.120
- D. 194.47.20.130

Activity Time:

Q2

The Dotted Decimal Notation(DDN) format for the given Hexadecimal Notation(HDN) C22F1582 is

- A. 194.50.21.145
- B. 194.47.21.130**
- C. 194.45.21.120
- D. 194.47.20.130

Activity Time:

Q3

Identify the no.of Networks, no.of Hosts per network in Class B IP addressing format.

- A. $2^{16}, 2^{16}$
- B. $2^{14}, 2^{16}$
- C. $2^{16}, 2^{14}$
- D. $2^{14}, 2^{16}-2$

Activity Time:

Q3

Identify the no.of Networks, no.of Hosts per network in Class B IP addressing format.

- A. $2^{16}, 2^{16}$
- B. $2^{14}, 2^{16}$
- C. $2^{16}, 2^{14}$
- D. $2^{14}, 2^{16}-2$**

Activity Time:

Q4

In IPV4 addressing format, the number of networks allowed under Class C is ?

- A. 2^{14}
- B. 2^7
- C. 2^{21}
- D. 2^{24}

Activity Time:

Q4

In IPV4 addressing format, the number of networks allowed under Class C is ?

- A. 2^{14}
- B. 2^7
- C. 2^{21}
- D. 2^{24}

Activity Time:

Q5

Suppose, instead of using 16 bits for network part of a Class B, 20 bits had been used. Then the number of Networks and Hosts per Network are

- A. $2^{10}, 2^{12}$
- B. $2^{18}, 2^{12}$
- C. $2^{18}, 2^{12} - 2$
- D. $2^{10}, 2^{12}-2$

Activity Time:

Q5

Suppose, instead of using 16 bits for network part of a Class B, 20 bits had been used. Then the number of Networks and Hosts per Network are

- A. $2^{10}, 2^{12}$
- B. $2^{18}, 2^{12}$
- C. $2^{18}, 2^{12} - 2$
- D. $2^{10}, 2^{12}-2$