## UNIT III NETWORK LAYER

Packet switching - Routing – Distance Vector and Link State Algorithms – RIP, OSPF and BGP - IPV4 Packet Format and Addressing – Effective IP address management techniques – Subnetting – CIDR – VLSM – DHCP – NAT – ICMP – Need for IPv6 – Addressing methods and types in IPv6 – IPv6 header – Advantages of IPv6 – Transition from IPv4 to IPv6.

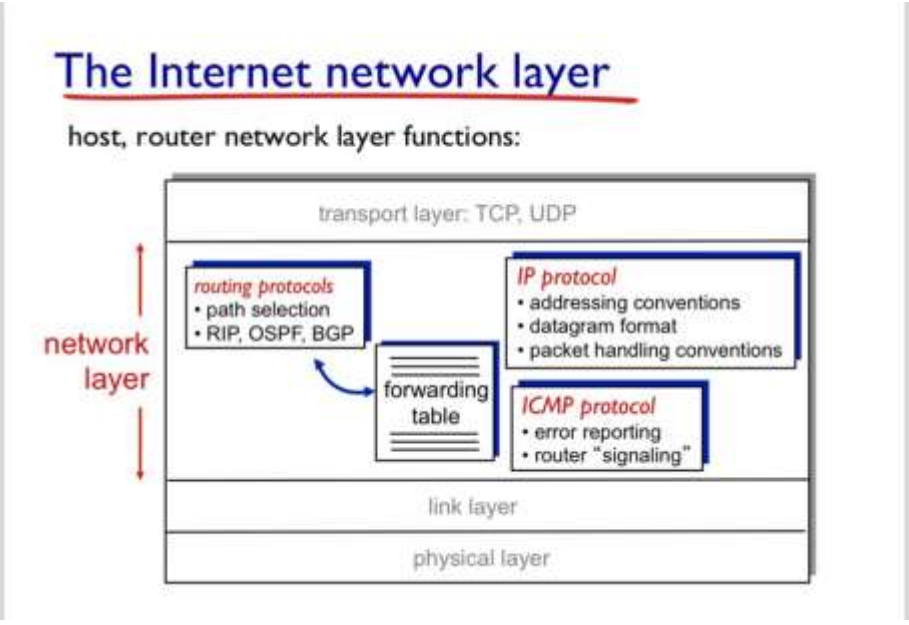
## 1. NETWORK LAYER

- The transport segment from the sending host is encapsulated into datagrams.
- On the receiving side, the segments are delivered to the transport layer by the network layer protocols in every host.
- Routers examine header fields in all IP datagrams passing through them.

**Forwarding**: (Getting through a single interchange) packets from a router's input to the appropriate router output.
**Routing**: (Planning a trip from source to destination) Determines the route taken by packets from source to destination.

The **network layer** is the third layer in the OSI model and plays a critical role in end-to-end communication by providing services such as:

- **Logical addressing**: This includes assigning IP addresses to devices on a network to identify them uniquely and ensure proper packet delivery.
- **Routing**: It involves determining the best path for data packets to travel from source to destination across multiple networks (e.g., the internet).
- **Packet forwarding**: This refers to the movement of packets across network devices (routers and switches) based on the routing table and logical address.
- **Fragmentation and reassembly**: This process breaks large packets into smaller fragments for transmission over networks with different Maximum Transmission Unit (MTU) sizes, then reassembles them at the destination.



## Packet Switching

**Definition**: A method of breaking data into packets for transmission across networks, reassembled at the destination.

**Types**:

1. **Datagram Switching**: (Connectionless) Packets take different paths to the destination (e.g., IP networks).
- **Advantages**: Flexibility, fault tolerance, efficient use of network resources.
- **Disadvantages**: Possibility of packet loss, out-of-order delivery, and delays due to congestion.

2. **Virtual Circuit Switching**: (Connection-Oriented) A fixed path is established before transmission (e.g., MPLS – Multi Protocol Label Switching).
- **Advantages**: More reliable, predictable performance.
- **Disadvantages**: Less flexible, as the path must be predefined and maintained.

**Virtual Circuit Networks** and **Datagram Networks**:

| Feature | Virtual Circuit Networks | Datagram Networks |
|---|---|---|
| **Connection Type** | Connection-oriented (path established before communication) | Connectionless (no path established beforehand) |
| **Reliability** | More reliable (due to fixed path and connection) | Less reliable (packets may be lost, duplicated, or unordered) |
| **Routing** | Fixed path for the entire communication session | Independent routing for each packet |
| **Packet Delivery** | Packets follow the same path and are delivered in order | Packets can take different paths and may arrive out of order |
| **Error Handling** | Handled by the network (dedicated path allows easy management) | Handled by upper layers (e.g., transport layer) |
| **Setup Time** | Requires setup time to establish the connection | No setup time required for communication |
| **Example Protocols** | ATM (Asynchronous Transfer Mode), Frame Relay, X.25 | IP (Internet Protocol), UDP |

## Forwarding Table in Virtual Circuits (VC) and Datagram Networks

### *Virtual Circuits (VC)*

**Forwarding Table:** In a Virtual Circuit network, each router maintains a forwarding table that maps incoming VC numbers to outgoing VC numbers and interfaces. This table is essential for directing the packets along the pre-established path.

**Example Forwarding Table**:

**Explanation:**

- **Incoming Interface**: The interface where the packet arrives.
- **Incoming VC**: The virtual circuit number of the incoming packet.
- **Outgoing Interface**: The interface where the packet should be forwarded.
- **Outgoing VC**: The virtual circuit number assigned to the outgoing packet.

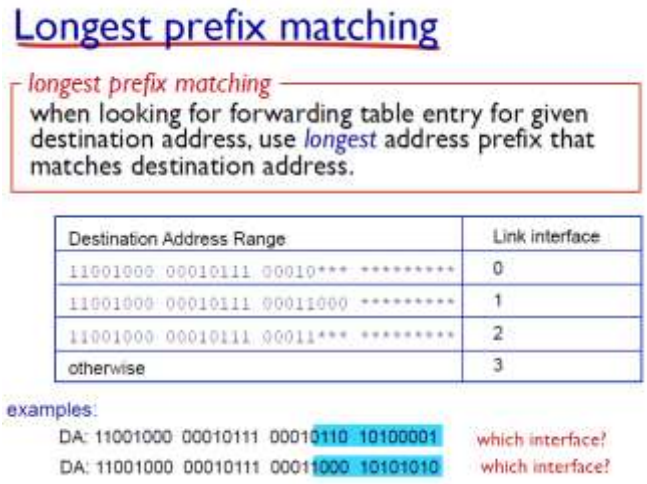

*forwarding table in northwest router:*

| Incoming interface | Incoming VC # | Outgoing interface | Outgoing VC # |
|---|---|---|---|
| 1 | 12 | 3 | 22 |
| 2 | 63 | 1 | 18 |
| 3 | 7 | 2 | 17 |
| 1 | 97 | 3 | 87 |
| ... | ... | ... | ... |

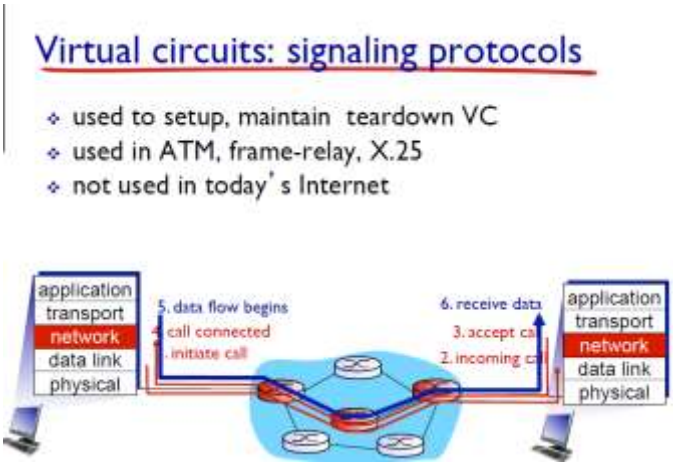
*VC routers maintain connection state information!*

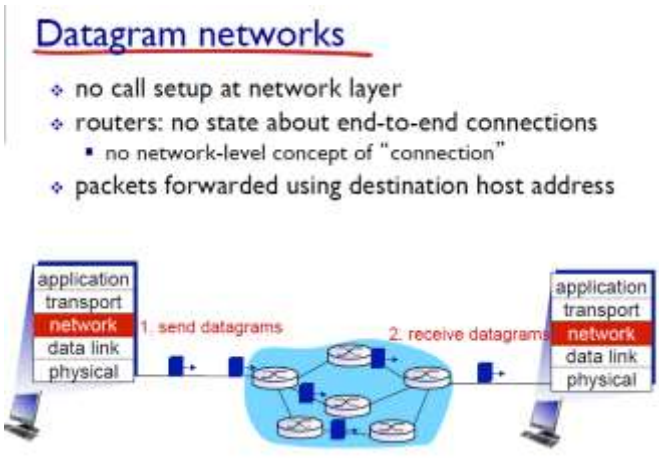
### *Datagram Networks*

**Forwarding Table:** In Datagram Networks, each router maintains a forwarding table that maps IP address prefixes to outgoing interfaces. Each packet is routed independently based on the destination IP address.


## Signaling Protocols in Virtual Circuits (VC) and Datagram Networks

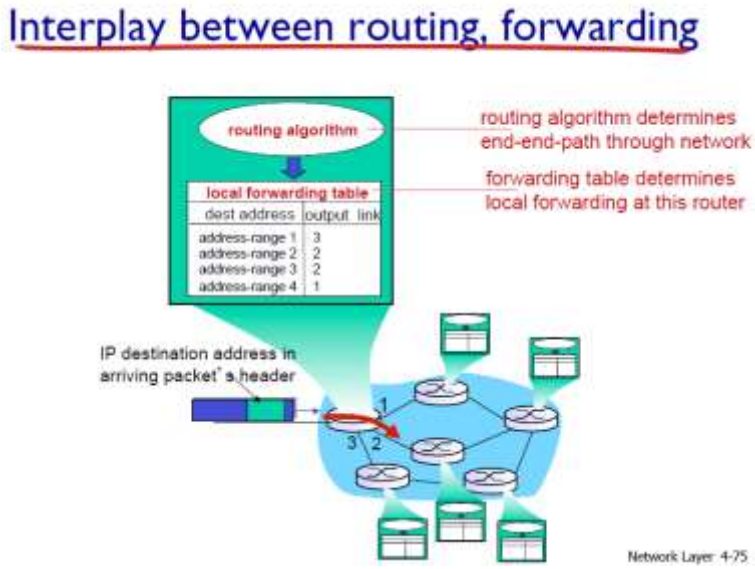


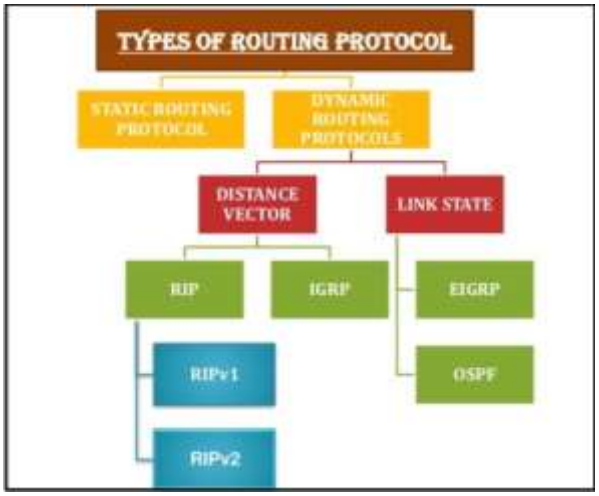
## Routing and Forwarding:

- **Routing** is about selecting the best path to take across the network.
- **Forwarding** is about sending a packet along that path based on the destination address.

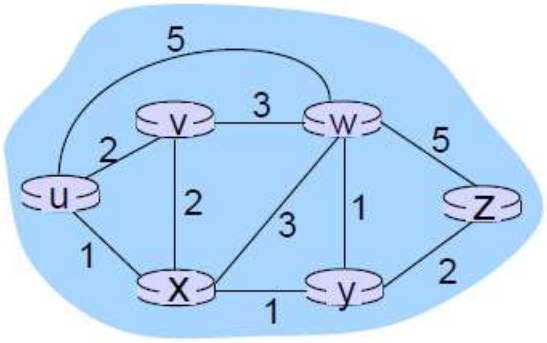
| Aspect | Routing | Forwarding |
|---|---|---|
| **Definition** | The process of determining the best path for packets to take. | The process of moving a packet from input to output based on the destination. |
| **Performed by** | Routers use routing protocols and algorithms to decide on paths. | Routers or network devices forward packets using forwarding tables. |
| **Timing** | Occurs when a router learns the network topology and updates routing tables. | Happens continuously when a packet arrives at the router. |
| **Scope** | Involves choosing the overall route from source to destination. | Involves deciding the next hop for a specific packet. |
| **Example Protocols** | OSPF, BGP, RIP | IP forwarding (in routers), MPLS forwarding |




**Flowchart Structure – Routing**



## Graph Extraction:




graph: G = (N,E)

N = set of routers = { u, v, w, x, y, z }

E = set of links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

c(x,x') = cost of link (x,x')

e.g., c(w,z) = 5

cost could always be 1, or inversely related to bandwidth, or inversely related to congestion

cost of path (x1, x2, x3,..., xp) = c(x1,x2) + c(x2,x3) + ... + c(xp-1,xp)

# Routing algorithm classification

## Based on Path Calculation Method:

(i) **Static Routing Algorithms**:

- The paths are preconfigured by the network administrator and do not change unless manually modified.
- **Example**: **Manual configuration** of routes in a router.
- **Pros**: Simple, predictable.
- **Cons**: Not adaptive to network changes, requires manual intervention.

(ii) **Dynamic Routing Algorithms**:

- The paths are automatically determined and adjusted based on network topology changes and traffic conditions. Routers exchange routing information to adapt to network changes.
- **Example**: **RIP**, **OSPF**, **BGP**.
- **Pros**: Adaptive to changes, can handle network failures and congestion.
- **Cons**: More complex, requires more resources (e.g., processing power and memory).

| Aspect | Static Routing | Dynamic Routing |
|---|---|---|
| **Configuration** | Manually set by the admin. | Automatically updated by routers. |
| **Adaptability** | No automatic updates. | Updates automatically with network changes. |
| **Scalability** | Best for small networks. | Best for large networks. |
| **Overhead** | Low; no need for updates. | Higher due to constant updates. |
| **Fault Tolerance** | Limited; manual fixes needed. | High; reroutes automatically. |
| **Examples** | Manual routes, Static IP routing. | RIP, OSPF, EIGRP, BGP. |
| **Pros** | Simple, low overhead. | Flexible, adapts to changes. |
| **Cons** | Inflexible, hard to manage large networks. | More complex, higher overhead. |

# ROUTING PROTOCOL

Routing protocols are vital for ensuring efficient data transmission across networks. This assignment delves into three critical routing protocols: Routing Information Protocol (RIP), Open Shortest Path First (OSPF), and Border Gateway Protocol (BGP). While RIP and OSPF are primarily intra-domain protocols, BGP operates at the inter-domain level, managing routing across Autonomous Systems (AS).

## 1. Routing Information Protocol (RIP)

### *Overview*

RIP, one of the earliest intra-AS routing protocols, is still utilized in smaller networks due to its simplicity. Defined in:

- **RIP Version 1**: [RFC 1058]
- **RIP Version 2**: [RFC 2453] (adds backward compatibility and route aggregation).

It is a distance-vector protocol based on hop count as the cost metric.

### *Key Mechanisms*

1. **Hop Count Metric**
    a. Each link has a cost of 1 hop.
    b. Maximum allowable cost: 15 hops, limiting RIP to smaller ASs.
2. **Routing Table**

Each router maintains a table with:

    a. Destination subnet.
    b. Next hop router.
    c. Number of hops to destination.

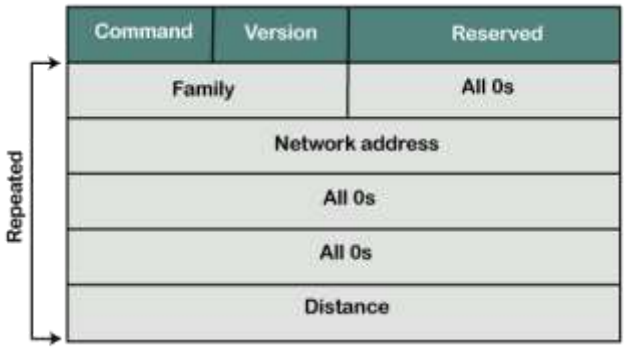
3. **Routing Updates**
    a. Routers exchange distance vectors (RIP advertisements) every 30 seconds.
    b. Advertisements include up to 25 subnets.
4. **Failure Detection**
    a. No update within 180 seconds marks the neighbor unreachable.
5. **Implementation**
    a. Application-layer protocol using UDP on port 520.
    b. UNIX uses a `routed` process for routing updates.

**RIP Message Format**

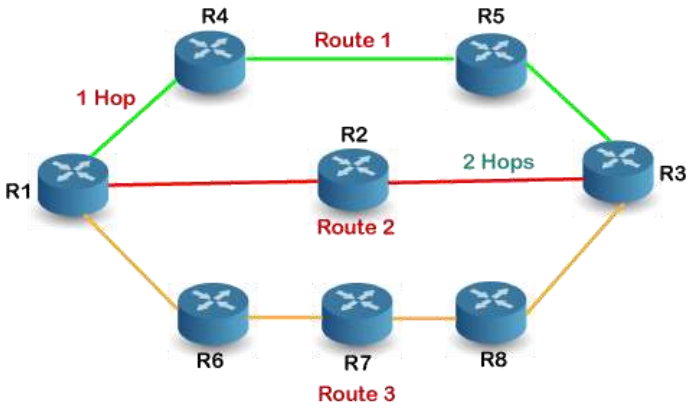
The message format is used to share information among different routers. The RIP contains the following fields in a message:

- Command: It is an 8-bit field that is used for request or reply. The value of the request is 1, and the value of the reply is 2.
- Version: Here, version means that which version of the protocol we are using. Suppose we are using the protocol of version1, then we put the 1 in this field.
- Reserved: This is a reserved field, so it is filled with zeroes.
- Family: It is a 16-bit field. As we are using the TCP/IP family, so we put 2 value in this field.

- Network Address: It is defined as 14 bytes field. If we use the IPv4 version, then we use 4 bytes, and the other 10 bytes are all zeroes.
- Distance: The distance field specifies the hop count, i.e., the number of hops used to reach the destination.




How does the RIP work?




If there are 8 routers in a network where Router 1 wants to send the data to Router 3. If the network is configured with RIP, it will choose the route which has the least number of hops. There are three routes in the above network, i.e., Route 1, Route 2, and Route 3. The Route 2 contains the least number of hops, i.e., 2 where Route 1 contains 3 hops, and Route 3 contains 4 hops, so RIP will choose Route 2.

### *Advantages and Disadvantages*

| Advantages | Disadvantages |
|---|---|
| Simplicity, easy setup | Limited to 15-hop networks |
| Widely supported | Slow convergence (30-second updates) |
| Decentralized operations | Vulnerable to count-to-infinity issue |

### *Comparison with OSPF*

- **Routing Type**: RIP uses distance-vector; OSPF uses link-state.
- **Scalability**: RIP is limited by 15 hops, while OSPF is more scalable.
- **Convergence**: OSPF converges faster than RIP.

## 2. Open Shortest Path First (OSPF)

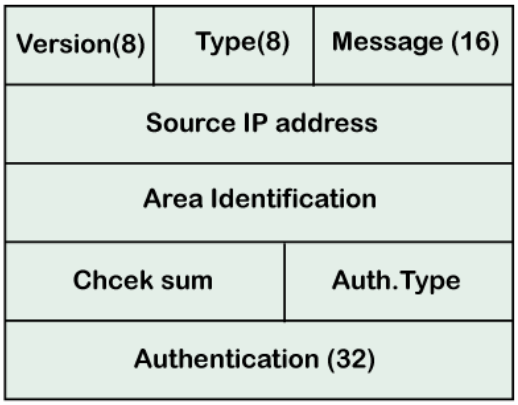
### *Overview*

OSPF is a robust and scalable intra-AS link-state routing protocol designed to replace RIP. Defined in **RFC 2328**, it uses Dijkstra's shortest-path algorithm.

### *Key Mechanisms*

1. **Link-State Flooding**
    a. Broadcasts link-state updates to all routers within an AS.
    b. Updates are triggered by topology changes or every 30 minutes.
2. **Routing Table Calculation**
    a. Builds a complete AS topology map.
    b. Shortest-path tree computed using Dijkstra's algorithm.
3. **Link Costs**
    a. Configurable by administrators to reflect hop count, bandwidth, etc.
4. **Integration with IP**
    a. OSPF uses IP protocol 89 and manages its own reliability and authentication.

**OSPF Message Format**
**The following are the fields in an OSPF message format:**




- **Version:** It is an 8-bit field that specifies the OSPF protocol version.
- **Type:** It is an 8-bit field. It specifies the type of the OSPF packet.
- **Message:** It is a 16-bit field that defines the total length of the message, including the header. Therefore, the total length is equal to the sum of the length of the message and header.
- **Source IP address:** It defines the address from which the packets are sent. It is a sending routing IP address.
- **Area identification:** It defines the area within which the routing takes place.
- **Checksum:** It is used for error correction and error detection.
- **Authentication type:** There are two types of authentication, i.e., 0 and 1. Here, 0 means for none that specifies no authentication is available and 1 means for pwd that specifies the password-based authentication.
- **Authentication:** It is a 32-bit field that contains the actual value of the authentication data.

### *Advanced Features*

- **Security**: MD5-based authentication ensures message authenticity.
- **Multiple Same-Cost Paths**: Enables load balancing.
- **Hierarchical Routing**: Divides AS into areas connected by Area Border Routers (ABRs).

### *Advantages and Disadvantages*

| Advantages | Disadvantages |
|---|---|
| Efficient convergence | More complex configuration |
| Suitable for large networks | Higher memory and CPU usage |
| Customizable link costs | |

## 3. Border Gateway Protocol (BGP)

### *Overview*

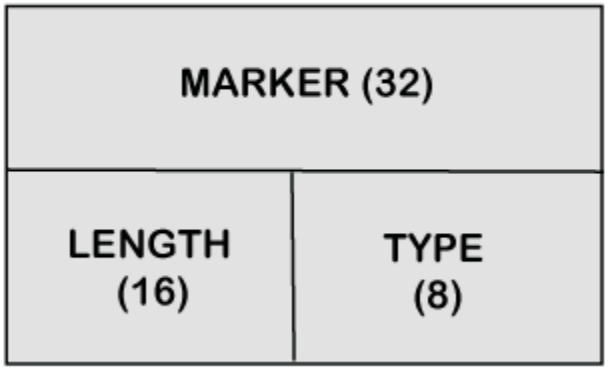
BGP is an inter-domain routing protocol that manages routing between ASs. It is defined in **RFC 4271** and ensures efficient and policy-driven routing on the global Internet.

### *Key Features*

1. **Path Vector Protocol**
    a. Maintains the path information (AS Path) for each destination.
    b. Avoids routing loops by rejecting routes with its own AS in the path.
2. **Policy-Based Routing**
    a. Allows network operators to define routing policies based on business or technical requirements.
3. **Message Types**
    a. **OPEN**: Establishes peer connections.
    b. **UPDATE**: Communicates route additions/removals.
    c. **KEEPALIVE**: Ensures session liveness.
    d. **NOTIFICATION**: Indicates errors or session termination.
4. **Scalability**
    a. Suitable for large-scale networks with thousands of ASs.
    b. Supports CIDR for efficient route aggregation.

**BGP Packet Format**
**Now we will see the format in which the packet travels. The following are the fields in a BGP packet format:**

BGP Packet Format




1. **Marker:** It is a 32-bit field which is used for the authentication purpose.
2. **Length:** It is a 16-bit field that defines the total length of the message, including the header.
3. **Type:** It is an 8-bit field that defines the type of the packe

***Advantages and Disadvantages***

| Advantages | Disadvantages |
|---|---|
| Scalable for large networks | Configuration complexity |
| Policy-driven routing | Vulnerable to misconfigurations |
| Avoids loops with AS Path | |

## Based on Routing Information Exchange:

### (i) Link-State Routing Algorithm: (GLOBAL):

- Routers maintain a complete map of the network topology and use this information to calculate the best path. Every router exchanges information about its directly connected links with all other routers.
- **Example**: **OSPF (Open Shortest Path First)**, **IS-IS (Intermediate System to Intermediate System)**.
- **Working**: Routers send link-state advertisements (LSAs) to all other routers, which then use algorithms (like Dijkstra's algorithm) to calculate the shortest paths.
- **Pros**: Faster convergence, avoids routing loops.
- **Cons**: Requires more resources (memory, CPU) for maintaining the topology map.

## Dijkstra's Algorithm

Dijkstra's algorithm is a fundamental algorithm used in **link-state routing** protocols like **Open Shortest Path First (OSPF)**. It finds the shortest path from a source node to all other nodes in a weighted graph.

## Steps of Dijkstra's Algorithm

1. **Initialization**:
    a. Set the distance to the source node as 0.
    b. Set the distance to all other nodes as infinity (∞).
    c. Mark all nodes as unvisited.

2. **Select Node**:
    a. Pick the unvisited node with the smallest tentative distance (initially the source node).
3. **Update Neighbors**:
    a. For each unvisited neighbor of the current node:
        i. Calculate the tentative distance: Tentative Distance=Current Distance+Edge Weight
        ii. If the calculated distance is smaller than the known distance, update it.
4. **Mark Visited**:
    a. Once all neighbors are considered, mark the current node as visited.
5. **Repeat**:
    a. Repeat steps 2–4 until all nodes are visited or the shortest path to the destination is found.

## Notation Used in Dijkstra's Algorithm

- G(V,E): A graph with vertices V and edges E.
- d[u]: The tentative distance from the source node s to node u.
- w(u,v): The weight of the edge between nodes u and v.
- : A priority queue (min-heap) of unvisited nodes, sorted by tentative distance.

## Example of Dijkstra's Algorithm

### *Graph Details:*

- **Nodes**: A,B,C,D,E,F
- **Edges and Weights**:
    - A→B: 4
    - A→C: 2
    - B→C: 5
    - B→D: 10
    - C→E: 3
    - E→D: 4
    - E→F: 6
    - D→F: 1

### *Initial State:*

| Node | Distance from A | Previous Node |
|---|---|---|
| A | 0 | - |
| B | ∞ | - |
| C | ∞ | - |
| D | ∞ | - |
| E | ∞ | - |
| F | ∞ | - |

### *Execution:*

1. **Start at A:**
    a. Tentative distances: d[B]=4,d[C]=2.
    b. Mark A as visited.
2. **Move to C (smallest tentative distance):**

    a. Tentative distances: d[E]=5 (via C).
    b. Mark C as visited.
3. **Move to B:**
    a. Tentative distances: d[D]=14 (via B).
    b. Mark B as visited.
4. **Move to E:**
    a. Tentative distances: d[D]=9,d[F]=11.
    b. Mark E as visited.
5. **Move to D:**
    a. Tentative distances: d[F]=10 (via D).
    b. Mark D as visited.
6. **Move to F:**
    a. All nodes are visited.

***Final Result:***

| Node | Distance from A | Previous Node |
|---|---|---|
| A | 0 | - |
| B | 4 | A |
| C | 2 | A |
| D | 9 | E |
| E | 5 | C |
| F | 10 | D |

## Discussion on Dijkstra's Algorithm

### *Advantages:*

1. **Guaranteed Shortest Path**: Always finds the shortest path in graphs with non-negative edge weights.
2. **Efficient**: Runs in O(V2) for dense graphs or O((V+E)logV) with a priority queue.
3. **Widely Used**: Employed in real-world routing protocols like OSPF.

### *Limitations:*

1. **Non-Negative Weights Only**: Does not work correctly with graphs containing negative weight edges.
2. **Not Suitable for Frequent Updates**: In networks with highly dynamic topology, repeated recalculations can be computationally expensive.

### *Applications:*

- Used in routing protocols (e.g., OSPF) to compute the shortest path.
- Traffic engineering and navigation systems.
- Network optimization and resource allocation.

**(ii) Distance-Vector Routing Algorithms (DECENTRALIZED):**

- Routers send updates about their routing tables to their neighbors periodically. The distance to a destination is typically measured by the number of hops.
- **Example**: **RIP (Routing Information Protocol)**, **BGP (Border Gateway Protocol)** (in some modes).
- **Working**: Routers exchange routing tables, and each router updates its table based on information from neighbors. They calculate the shortest path based on distance metrics (like hop count).
- **Pros**: Simple to implement.
- **Cons**: Slow to converge, prone to routing loops (especially in large networks).

## Distance vector algorithm

### Distance vector algorithm

- $D_x(y)$ = estimate of least cost from x to y
  - x maintains distance vector $\mathbf{D}_x = [D_x(y): y \in N]$
- node x:
  - knows cost to each neighbor v: c(x,v)
  - maintains its neighbors' distance vectors. For each neighbor v, x maintains $\mathbf{D}_v = [D_v(y): y \in N]$

*key idea:*

- from time-to-time, each node sends its own distance vector estimate to neighbors
- when x receives new DV estimate from neighbor, it updates its own DV using B-F equation:

$$D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\} \text{ for each node } y \in N$$

- under minor, natural conditions, the estimate $D_x(y)$ converge to the actual least cost $d_x(y)$

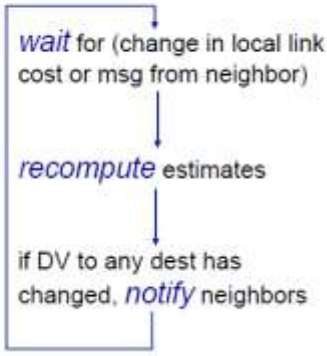
*iterative, asynchronous:* each local iteration caused by:

- local link cost change
- DV update message from neighbor

*distributed:*

- each node notifies neighbors *only* when its DV changes
  - neighbors then notify their neighbors if necessary

*each node:*




## Bellman-Ford Algorithm

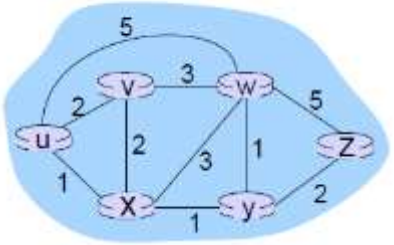
clearly, dv(z) = 5, dx(z) = 3, dw(z) = 3B-F equation says:

du(z) = min { c(u,v) + dv(z),

c(u,x) + dx(z),

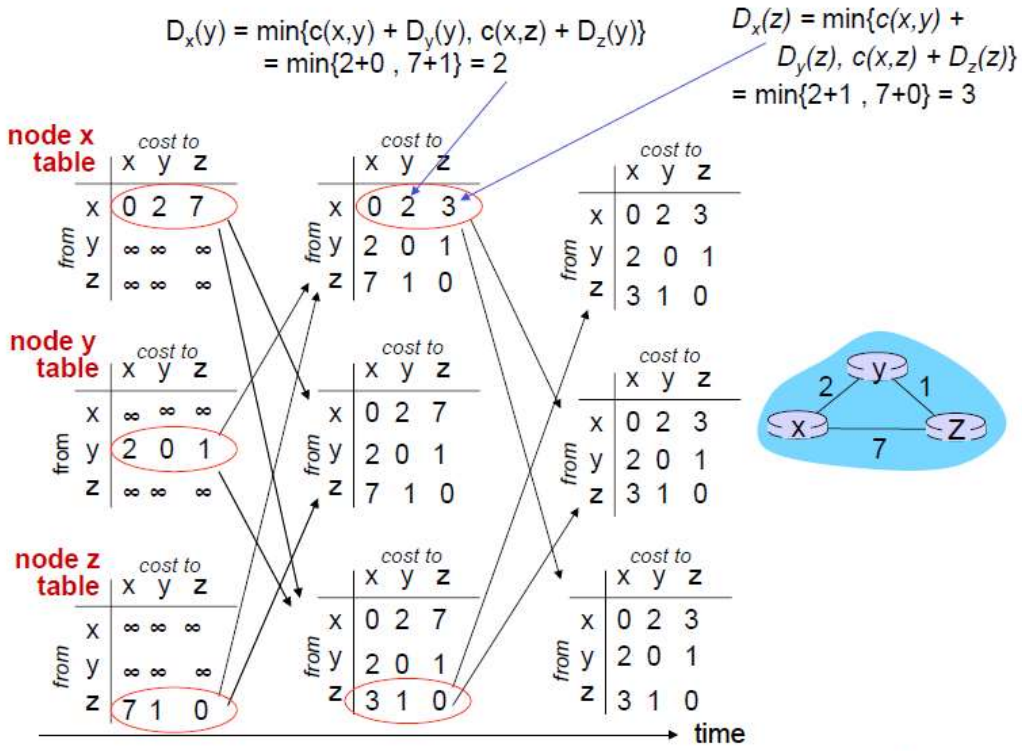
c(u,w) + dw(z) }

= min {2 + 5,1 + 3, 5 + 3} = 4




node achieving minimum is next

hop in shortest path, used in forwarding table

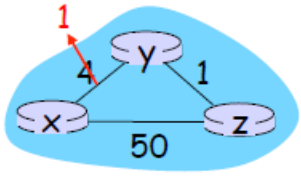





# Distance vector: link cost changes

*link cost changes:*

- node detects local link cost change
- updates routing info, recalculates distance vector
- if DV changes, notify neighbors




"good news travels fast"

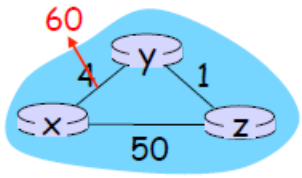
$t_0$ : *y* detects link-cost change, updates its DV, informs its neighbors.

$t_1$ : *z* receives update from *y*, updates its table, computes new least cost to *x* , sends its neighbors its DV.

$t_2$ : *y* receives *z*' s update, updates its distance table. *y*' s least costs do *not* change, so *y* does *not* send a message to *z*.

*link cost changes:*

- node detects local link cost change
- *bad news travels slow* - "count to infinity" problem!
- 44 iterations before algorithm stabilizes: see text




*poisoned reverse:*

- If Z routes through Y to get to X :
  - Z tells Y its (Z' s) distance to X is infinite (so Y won' t route to X via Z)
- will this completely solve count to infinity problem?

## Comparison Table

| Aspect | Link State (LS) | Distance Vector (DV) |
|---|---|---|
| **Message Complexity** | - Each node floods link-state packets (LSPs) to all other nodes.<br>- With n nodes and E links: O(nE) messages are sent. | - Only exchanges information with direct neighbors.<br>- Message complexity depends on convergence time and varies with network conditions. |
| **Speed of Convergence** | - Dijkstra's algorithm runs in O(n2) time (or O(nlogn) with optimized implementations).<br>- Requires O(nE) messages for global topology updates.<br>- May exhibit oscillations under certain conditions. | - Convergence time is variable and depends on network size and changes.<br>- Susceptible to issues like **routing loops** and **count-to-infinity** problems, which slow down convergence. |
| **Robustness (Fault Tolerance)** | - A malfunctioning node can advertise incorrect link costs.<br>- Each node computes its own table independently, so errors are localized. | - A faulty node can advertise incorrect path costs.<br>- Errors can propagate through the network as other nodes rely on shared distance vectors. |
| **Issues** | - Oscillations may occur during frequent updates or rapid topology changes. | - Vulnerable to routing loops, especially in the absence of mechanisms like **split horizon** or **poison reverse**.<br>- Count-to-infinity problem leads to slow error correction. |
| **Error Propagation** | - Errors remain localized since each node computes its table based on its own topology knowledge. | - Errors spread across the network because each node's table is used by others to compute paths. |