# Module-3: Network Layer

Syllabus:The Network layer: What's Inside a Router?: Input Processing, Switching, Output Processing, Where Does Queuing Occur? Routing control plane, IPv6,A Brief foray into IP Security, Routing Algorithms: The Link-State (LS) Routing Algorithm, The Distance-Vector (DV) Routing Algorithm, Hierarchical Routing, Routing in the Internet, Intra-AS Routing in the Internet: RIP, Intra-AS Routing in the Internet: OSPF, Inter/AS Routing: BGP, Broadcast Routing Algorithms and Multicast.

## With a neat diagram, explain the Router structure

- Router is an internetworking device responsible for routing the packets from source to destination. It works at network layer.
- Following figure depicts the major structural components of a router.

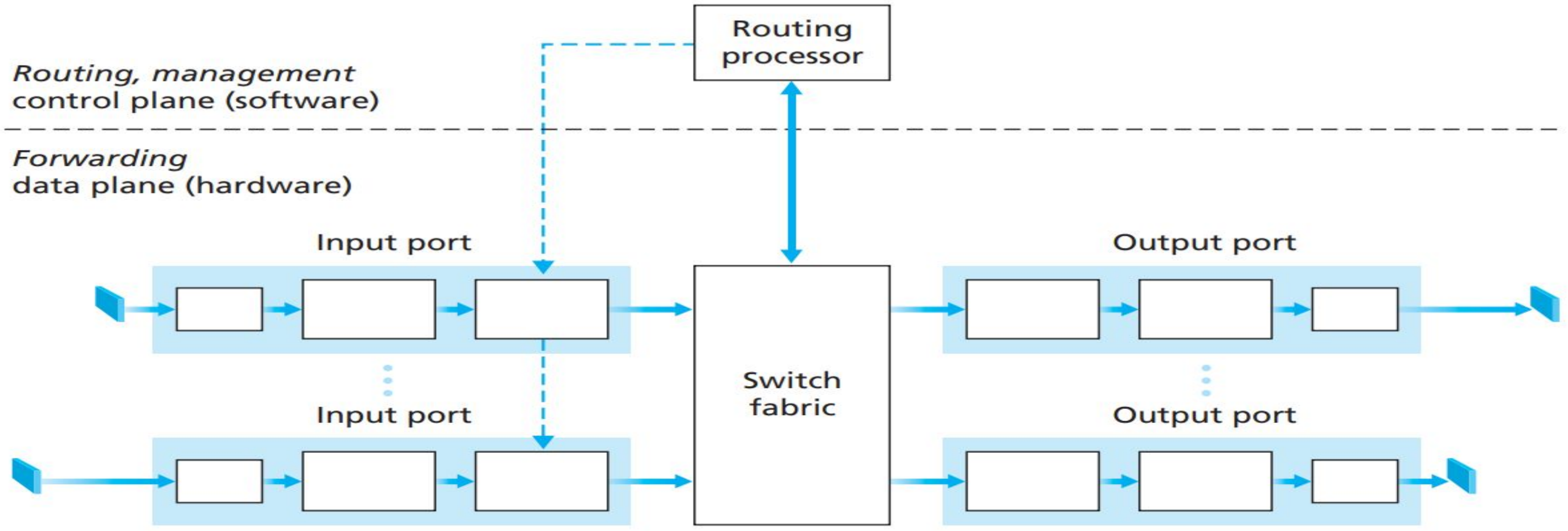


**Figure 1. Structural components of a router**

**Input ports:**

- An input port performs the physical layer function of terminating an incoming physical link at a router.
- An input port also performs link-layer functions needed to interoperate with the link layer at the other side of the incoming link. The lookup function is also performed at the input port.
- Forwarding table is consulted to determine the router output port to which an arriving packet will be forwarded via the switching fabric.

**Switching fabric:**

- The switching fabric connects the router’s input ports to its output ports.
- Switching can be accomplished in a number of ways:

1.Switching via memory.  2.Switching via a bus  3.Switching via an interconnection network

Switching via memory:

- switching between input and output ports being done under direct control of the CPU (routing processor)
- An input port signals the packet to the processor unit.
- The routing processor then extracted the destination address from the header, looked up the appropriate output port in the forwarding table, and copied the packet to the output port buffer. Input and output ports are connected through a common shared memory.

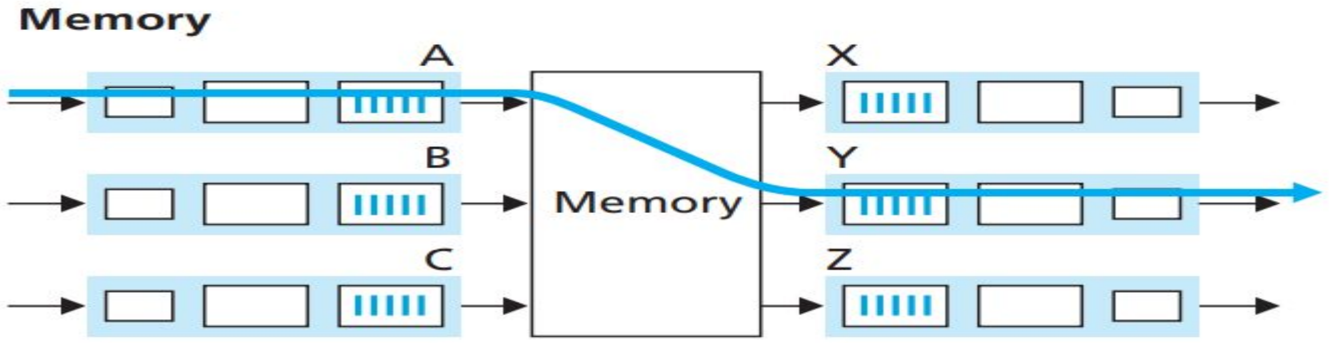


**Figure 2. Switching via memory**

## Switching via a bus:

- In this approach, an input port transfers a packet directly to the output port over a shared bus, without intervention by the routing processor.
- This is typically done by having the input port prepend a switch-internal label (header) to the packet indicating the local output port to which this packet is being transferred and transmitting the packet onto the bus.

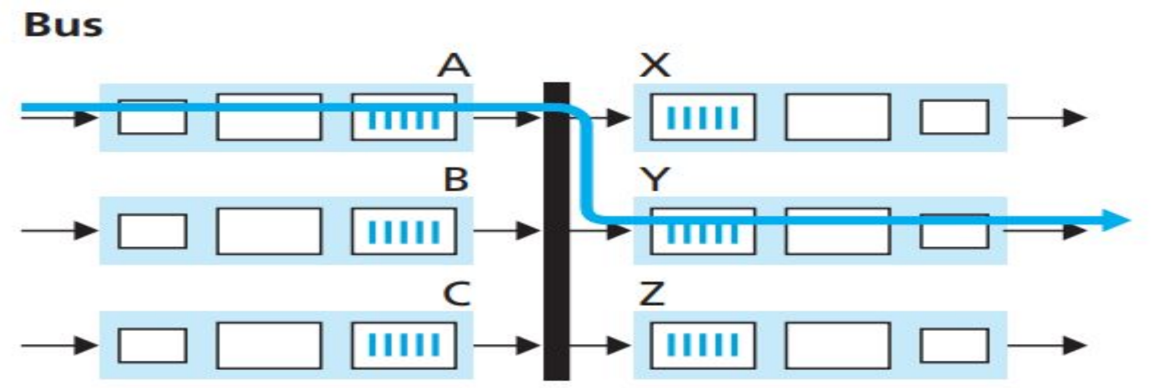


**Figure 3. Switching via Bus**

## Switching via an interconnection network:Uses a crossbar Switch

- A crossbar switch is an interconnection network consisting of 2N buses that connect N input ports to N output ports.
- Each vertical bus intersects each horizontal bus at a crosspoint, which can be opened or closed at any time by the switch fabric.

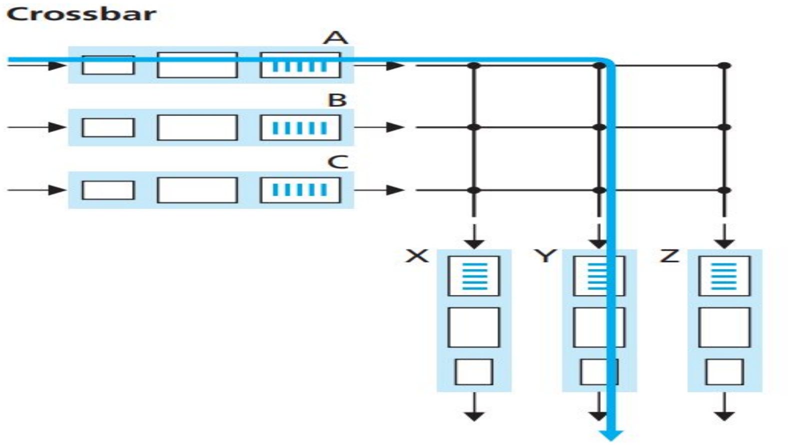


**Figure 4. Switching via crossbar switch**

- crossbar networks are capable of forwarding multiple packets in parallel.

**Output Processing:**

- Output port processing takes packets that have been stored in the output port's memory and transmits them over the output link.
- This includes selecting and de-queuing packets for transmission, and performing the needed link layer and physical-layer transmission functions.

**Routing processor:**

- The routing processor executes the routing protocols maintains routing tables and attached link state information, and computes the forwarding table for the router. It also performs the network management functions.
- Router control plane functions are usually implemented in software and execute on the routing processor

## Queuing:

- Packet queues may form at both the input ports and the output ports.
- The location and extent of queuing will depend on the traffic load, the relative speed of the switching fabric, and the line speed.
- Many packet scheduling algorithms like first-come-first-served (FCFS) scheduling, priority queuing, fair queuing or a more sophisticated scheduling discipline such as weighted fair queuing (WFQ) is available.

## 2. Explain IPv6 datagram format

### IPv6 Datagram Format:

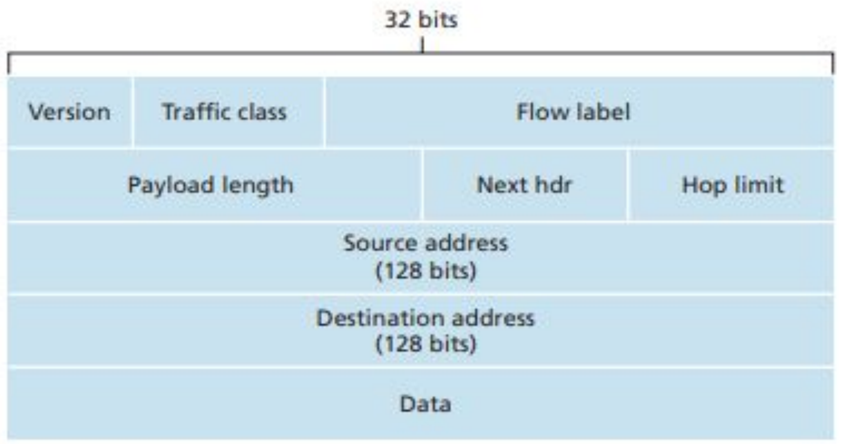


**Figure 5. IPv6 datagram**

- **IPv6 has expanded addressing capabilities:** Increases the size of the IP address from 32 to 128 bits.
- **A streamlined 40-byte header:** 40 bytes of mandatory header is used in IPv6 whereas IPv4 uses 20 bytes of mandatory header.

**Flow labeling and priority:** Flow label refers to labeling of packets belonging to particular flows for which the sender requests special handling, such as a non default quality of service or real-time service. The IPv6 header also has an 8-bit traffic class field. This field, like the TOS field in IPv4, can be used to give priority to certain datagrams within a flow.

**The following fields are defined in IPv6:**

**Version:** This 4-bit field identifies the IP version number.

**Traffic class:** This 8-bit field specify priority.

**Flow label:** this 20-bit field is used to identify a flow of datagrams.

**Payload length:** This 16-bit value is treated as an unsigned integer giving the number of bytes in the IPv6 datagram following the fixed-length, 40-byte datagram header.

**Next header:** This field identifies the next following header.

**Hop limit:** The contents of this field are decremented by one by each router that forwards the datagram. If the hop limit count reaches zero, the datagram is discarded. Source and destination addresses: 128 bit IPv6 address.

**Data:** This is the payload portion of the IPv6 datagram.

## 3. Differentiate between IPv4 and IPv6 datagrams

**The most important changes introduced in IPv6 are:**

**Expanded addressing capabilities:** IPv6 increases the size of the IP address from 32 to 128 bits.

**A streamlined 40-byte header:** 40 bytes of mandatory header is used in IPv6 whereas IPv4 uses 20 bytes of mandatory header.

**Flow labeling and priority:** Flow label refers to labeling of packets belonging to particular flows for which the sender requests special handling, such as a non default quality of service or real-time service. The IPv6 header also has an 8-bit traffic class field. This field, like the TOS field in IPv4, can be used to give priority to certain datagrams within a flow.

**Following fields appearing in the IPv4 datagram are no longer present in the IPv6 datagram**

**Fragmentation/Reassembly:** IPv6 does not allow for fragmentation and reassembly at intermediate routers; these operations can be performed only by the source and destination. If an IPv6 datagram received by a router is too large to be forwarded over the outgoing link, the router simply drops the datagram and sends a "Packet Too Big" ICMP error message (see below) back to the sender. The sender can then resend the data, using a smaller IP datagram size.

**Header checksum:** Because the transport-layer and link-layer protocols in the Internet layers perform check-summing, the designers of IP probably felt that this functionality was sufficiently redundant in the network layer that it could be removed.

**Options:** An options field is no longer a part of the standard IP header. Instead of option field extension headers are used.

## Discuss different approaches for transition from iPv4 to IPv6

**Declaring a flag day:** A given time and date when all Internet machines would be turned off and upgraded from IPv4 to IPv6. But, a flag day involving hundreds of millions of machines and millions of network administrators and users, is even more unthinkable today.

**Dual-stack:** IPv6/IPv4 nodes must have both IPv6 and IPv4 addresses. They must furthermore be able to determine whether another node is IPv6-capable or IPv4-only. In the dual-stack approach, if either the sender or the receiver is only IPv4- capable, an IPv4 datagram must be used. As a result, it is possible that two IPv6- capable nodes can end up, in essence, sending IPv4 datagrams to each other.

**Tunneling:** Tunneling can solve the problem noted above. The basic idea behind tunneling is the following. Suppose two IPv6 nodes want to interoperate using IPv6 datagrams but are connected to each other by intervening IPv4 routers. We refer to the intervening set of IPv4 routers between two IPv6 routers as a tunnel. With tunneling, the IPv6 node on the sending side of the tunnel takes the entire IPv6 datagram and puts it in the data (payload) field of an IPv4 datagram.

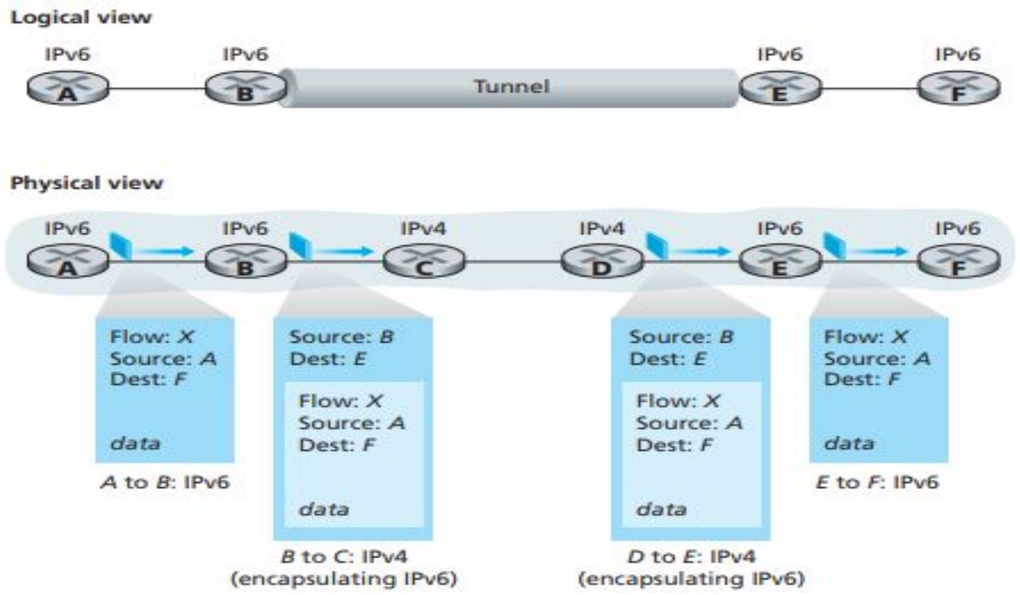


**Figure 6. Tunneling**

IPv4 datagram is then addressed to the IPv6 node on the receiving side of the tunnel (for example, E) and sent to the first node in the tunnel (for example, C).

The intervening IPv4 routers in the tunnel route this IPv4 datagram among themselves, just as they would any other datagram, blissfully unaware that the IPv4 datagram itself contains a complete IPv6 datagram.

The IPv6 node on the receiving side of the tunnel eventually receives the IPv4 datagram (it is the destination of the IPv4 datagram!), determines that the IPv4 datagram contains an IPv6 datagram, extracts the IPv6 datagram, and then routes the IPv6 datagram exactly as it would if it had received the IPv6 datagram from a directly connected IPv6 neighbor.

## Write and explain Dijkstra’s routing algorithm OR Explain the working of link state routing algorithm with an example

**Dijkstra’s algorithm is a Link-State Routing algorithm named after its inventor.**

Dijkstra’s algorithm computes the least-cost path from one node (the source, which we will refer to as u) to all other nodes in the network.

Dijkstra’s **algorithm is iterative** and has the property that after the $k^{th}$ iteration of the algorithm, the least-cost paths are known to k destination nodes, and among the least-cost paths to all destination nodes, these k paths will have the k smallest costs.

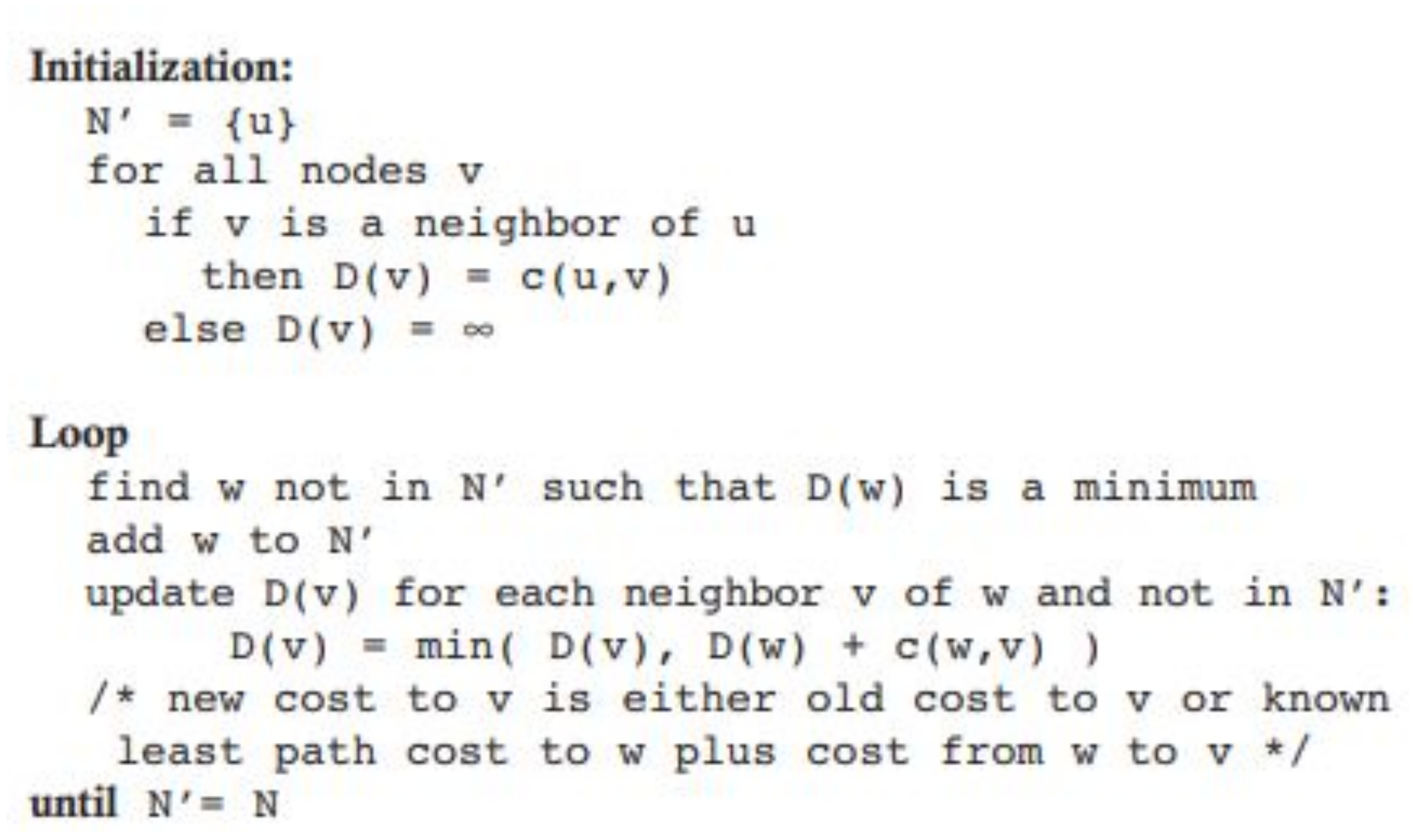
Let us define the following notation:

• D(v): cost of the least-cost path from the source node to destination v as of this iteration of the algorithm.

• p(v): previous node (neighbor of v) along the current least-cost path from the source to v.

• N : subset of nodes; v is in N if the least-cost path from the source to v is definitively known.

## Link-State (LS) Algorithm for Source Node u



```
Initialization:
  N' = {u}
  for all nodes v
    if v is a neighbor of u
      then D(v) = c(u,v)
    else D(v) = ∞

Loop
  find w not in N' such that D(w) is a minimum
  add w to N'
  update D(v) for each neighbor v of w and not in N':
       D(v) = min( D(v), D(w) + c(w,v) )
  /* new cost to v is either old cost to v or known
   least path cost to w plus cost from w to v */
until N'= N
```

Dijkstra’s (Link State) routing algorithm consists of an initialization step followed by a loop. The number of times the loop is executed is equal to the number of nodes in the network. Upon termination, the algorithm will have calculated the shortest paths from the source node u to every other node in the network.

Example: Consider the following network topology

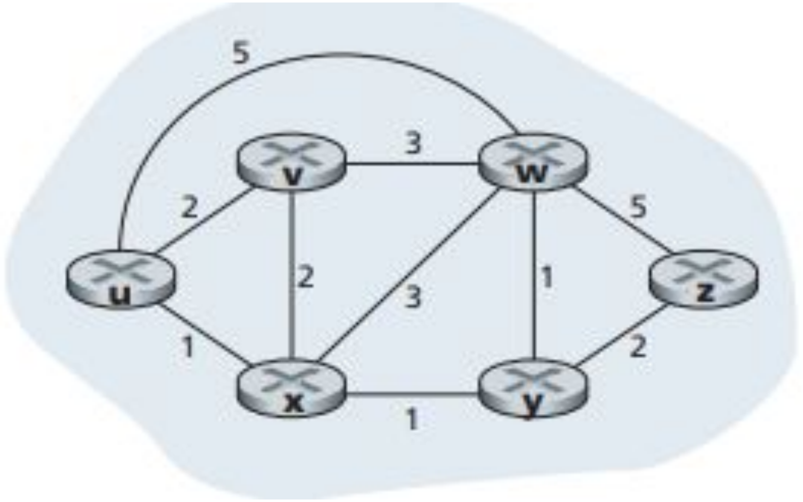
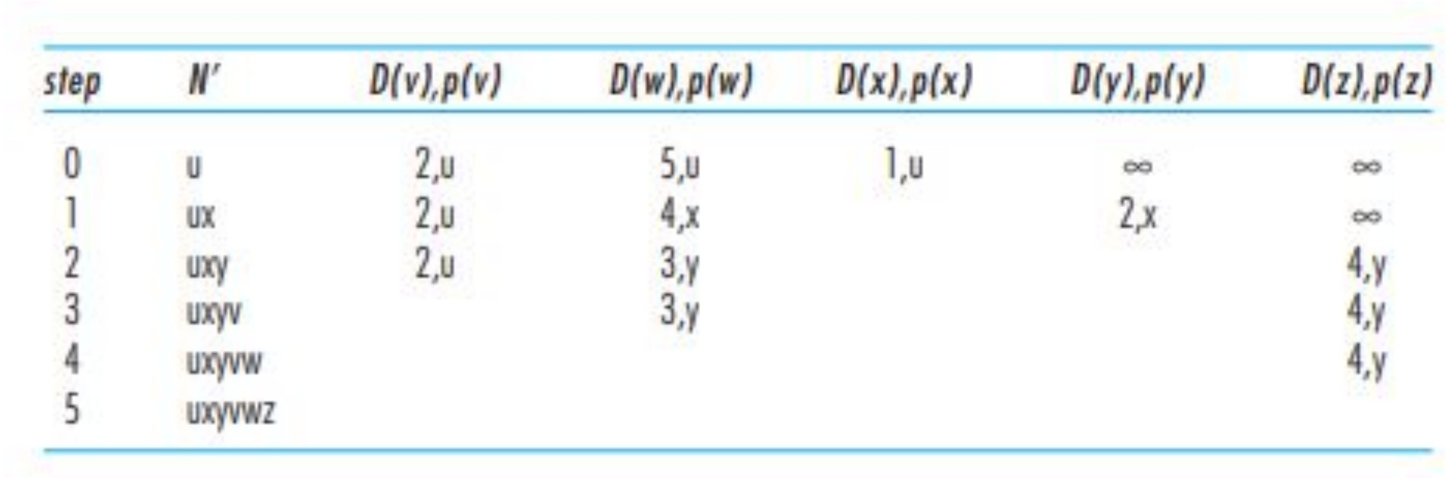


Figure 7. Network topology

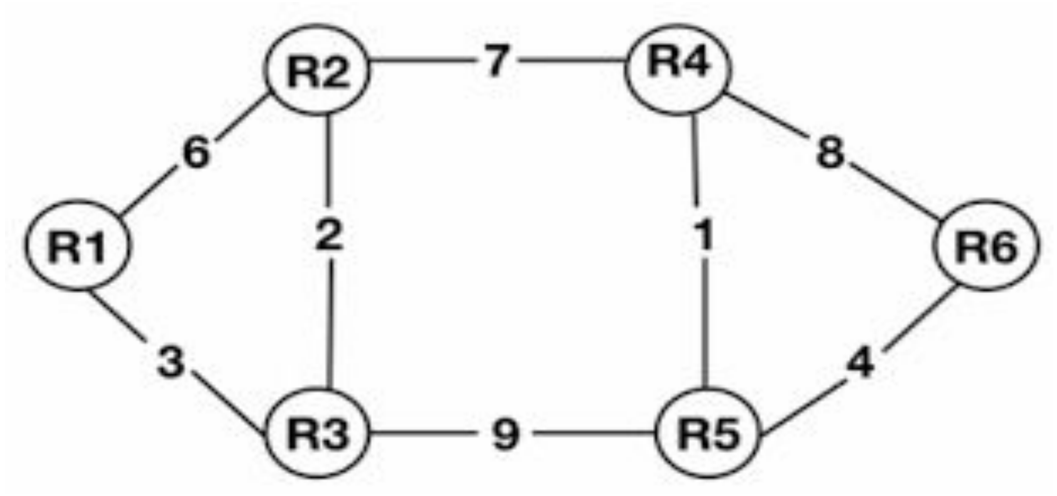
let’s consider the network in Figure 7, and compute the least-cost paths from u to all possible destinations. A tabular summary of the algorithm’s computation is shown in Table given below, where each line in the table gives the values of the algorithm’s variables at the end of the iteration.



| step | N' | D(v),p(v) | D(w),p(w) | D(x),p(x) | D(y),p(y) | D(z),p(z) |
|---|---|---|---|---|---|---|
| 0 | u | 2,u | 5,u | 1,u | ∞ | ∞ |
| 1 | ux | 2,u | 4,x |  | 2,x | ∞ |
| 2 | uxy | 2,u | 3,y |  |  | 4,y |
| 3 | uxyv |  | 3,y |  |  | 4,y |
| 4 | uxyvw |  |  |  |  | 4,y |
| 5 | uxyvwz |  |  |  |  |  |

When the LS algorithm terminates, we have, for each node, its predecessor along the least-cost path from the source node.

## Find the shortest path from node A to all the other nodes for the given network using Dijkstra's algorithm

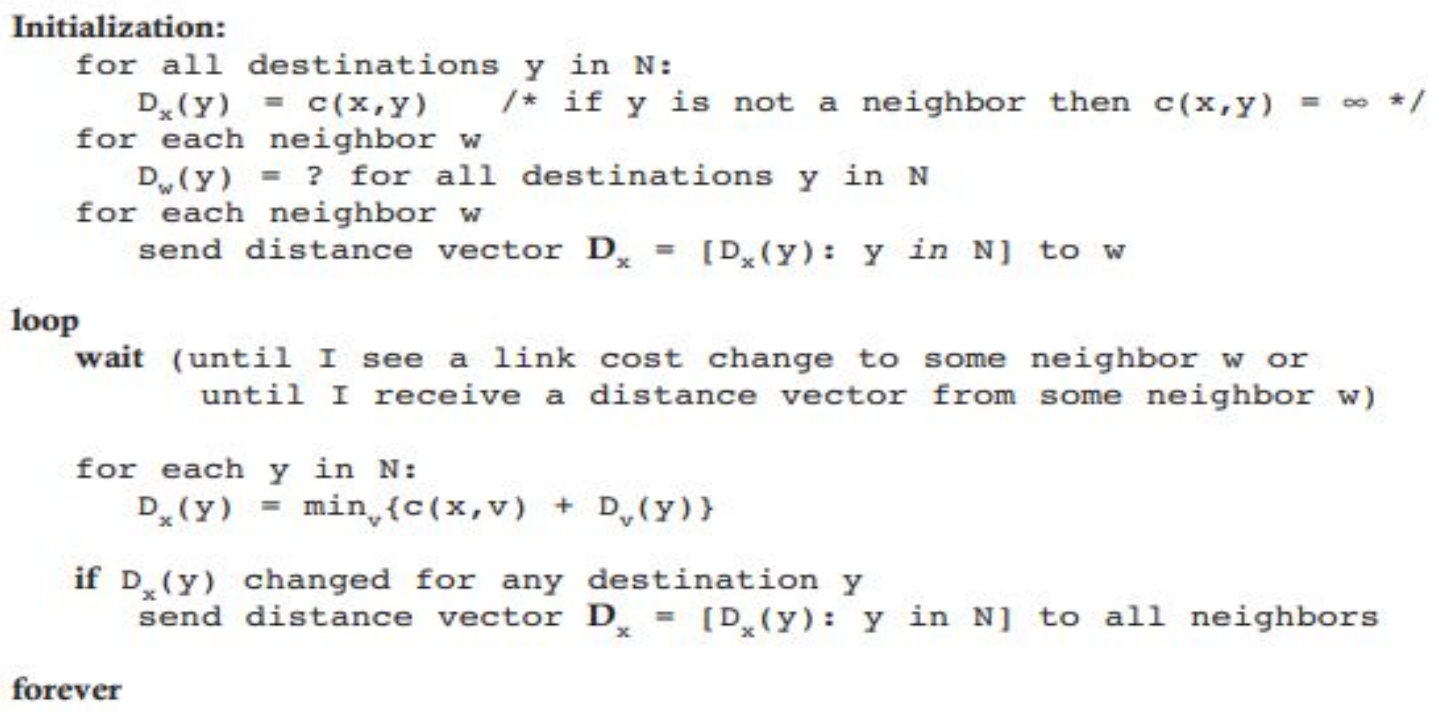


## Write and explain Bellman-Ford routing algorithm 8M OR Explain the working of Distance vector routing algorithm

- Distance vector (DV) algorithm is iterative, asynchronous, and distributed.
- Each node x begins with Dx (y), an estimate of the cost of the least-cost path from itself to node y, for all nodes in N. Let Dx = [Dx (y): y in N] be node x's distance vector, which is the vector of cost estimates from x to all other nodes, y, in N. With the DV algorithm, each node x maintains the following routing information.
- For each neighbor v, the cost c(x,v) from x to directly attached neighbor v,
    - Node x's distance vector, that is, Dx = [Dx (y): y in N], containing x's estimate of its cost to all destinations, y, in N
    - The distance vectors of each of its neighbors, that is, Dv = [Dv (y): y in N] for each neighbor v of x from time to time, each node sends a copy of its distance vector to each of its neighbors.
    - When a node x receives a new distance vector from any of its neighbors v, it saves v's distance vector, and then uses the Bellman-Ford equation to update its own distance vector as follows: Dx (y) minv {c(x,v) + Dv (y)} for each node y in N

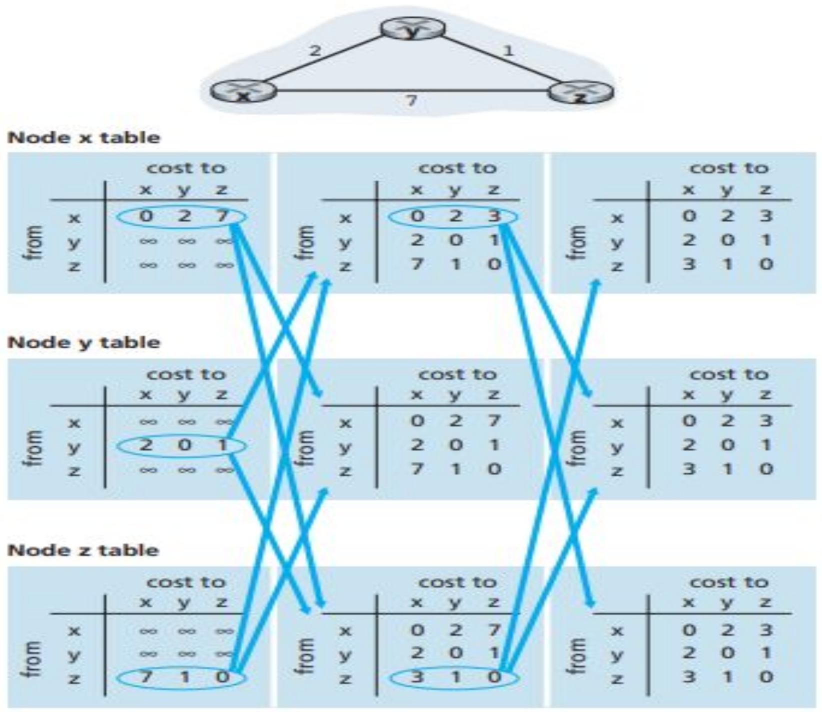
- If node x's distance vector has changed as a result of this update step, node x will then send its updated distance vector to each of its neighbors, which can in turn update their own distance vectors.

Distance-Vector (DV) Algorithm: At each node, x



```
Initialization:
    for all destinations y in N:
        $D_x(y) = c(x,y)$    /* if y is not a neighbor then $c(x,y) = \infty$ */
    for each neighbor w
        $D_w(y) = ?$ for all destinations y in N
    for each neighbor w
        send distance vector $\mathbf{D}_x = [D_x(y): y \textit{ in } N]$ to w

loop
    wait (until I see a link cost change to some neighbor w or
          until I receive a distance vector from some neighbor w)

    for each y in N:
        $D_x(y) = \min_v\{c(x,v) + D_v(y)\}$

    if $D_x(y)$ changed for any destination y
        send distance vector $\mathbf{D}_x = [D_x(y): y$ in $N]$ to all neighbors

forever
```

- In the DV (Bellman Ford) algorithm , a node x updates its distance-vector estimate when it either sees a cost change in one of its directly attached links or receives a distance vector update from some neighbor.
- Figure given below illustrates the operation of the DV (Bellman Ford) algorithm for the simple three node network shown at the top of the figure.



Node x table

| | | cost to | | |
|---|---|---|---|---|
| | | x | y | z |
| from | x | 0 | 2 | 7 |
| | y | ∞ | ∞ | ∞ |
| | z | ∞ | ∞ | ∞ |

| | | cost to | | |
|---|---|---|---|---|
| | | x | y | z |
| from | x | 0 | 2 | 3 |
| | y | 2 | 0 | 1 |
| | z | 7 | 1 | 0 |

| | | cost to | | |
|---|---|---|---|---|
| | | x | y | z |
| from | x | 0 | 2 | 3 |
| | y | 2 | 0 | 1 |
| | z | 3 | 1 | 0 |

Node y table

| | | cost to | | |
|---|---|---|---|---|
| | | x | y | z |
| from | x | ∞ | ∞ | ∞ |
| | y | 2 | 0 | 1 |
| | z | ∞ | ∞ | ∞ |

| | | cost to | | |
|---|---|---|---|---|
| | | x | y | z |
| from | x | 0 | 2 | 7 |
| | y | 2 | 0 | 1 |
| | z | 7 | 1 | 0 |

| | | cost to | | |
|---|---|---|---|---|
| | | x | y | z |
| from | x | 0 | 2 | 3 |
| | y | 2 | 0 | 1 |
| | z | 3 | 1 | 0 |

Node z table

| | | cost to | | |
|---|---|---|---|---|
| | | x | y | z |
| from | x | ∞ | ∞ | ∞ |
| | y | ∞ | ∞ | ∞ |
| | z | 7 | 1 | 0 |

| | | cost to | | |
|---|---|---|---|---|
| | | x | y | z |
| from | x | 0 | 2 | 7 |
| | y | 2 | 0 | 1 |
| | z | 3 | 1 | 0 |

| | | cost to | | |
|---|---|---|---|---|
| | | x | y | z |
| from | x | 0 | 2 | 3 |
| | y | 2 | 0 | 1 |
| | z | 3 | 1 | 0 |

- The operation of the algorithm is illustrated in a synchronous manner, where all nodes simultaneously receive distance vectors from their neighbors, compute their new distance vectors, and inform their neighbors if their distance vectors have changed.
- The leftmost column of the figure displays three initial routing tables for each of the three nodes.
- After initialization, each node sends its distance vector to each of its two neighbors. This is illustrated in above given figure by the arrows from the first column of tables to the second column of tables.
- After receiving the updates, each node recomputes its own distance vector.
- The second column therefore displays, for each node, the node’s new distance vector along with distance vectors just received from its neighbors.
- After the nodes recompute their distance vectors, they again send their updated distance vectors to their neighbors (if there has been a change).

The process of receiving updated distance vectors from neighbors, recomputing routing table entries, and informing neighbors of changed costs of the least-cost path to a destination continues until no update messages are sent. At this point, no further routing table calculations will occur and the algorithm will enter a quiescent state.

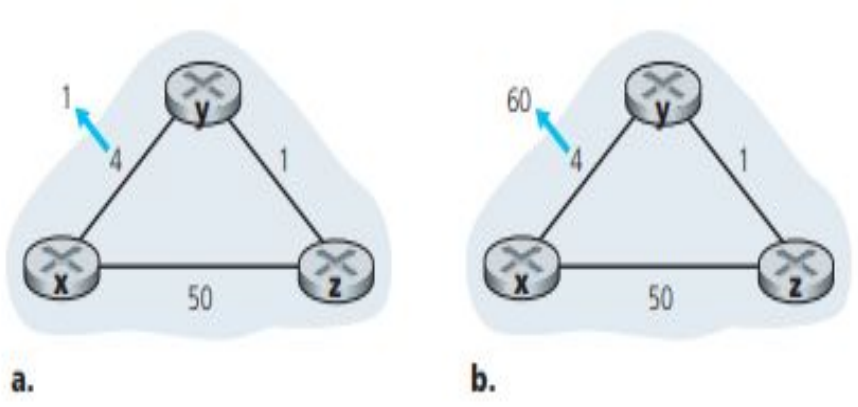
## What is ‘Counting to infinity’ problem in distance vector routing? Discuss about the solution.

- When a node running the DV(Bellman Ford) algorithm, detects a change in the link cost from itself to a neighbor, it updates its distance vector and, if there’s a change in the cost of the least-cost path, informs its neighbors of its new distance vector.
- Decreased cost between any set of nodes propagates quickly through the network. The algorithm converges quickly. But when the link cost increases than the cost of link before, algorithm may continue through iterations to find the new shortest path as per the logic of distance vector routing. But after certain number of iterations, it may form routing loops and algorithm never gets converged. This problem is called as counting to infinity problem.

Example:

## Example for counting to infinity Problem:

- Suppose that the link cost between nodes x and y in the below given figure, increases from 4 to 60, as shown in Figure(b). Before the link cost changes, D y (x) = 4, D y (z) = 1, Dz (y) = 1, and Dz (x) = 5.
- At time t 0, y detects the link-cost change (the cost has changed from 4 to 60). y computes its new minimum-cost path to x to have a cost of D y (x) = min{c(y,x) + Dx (x), c(y,z) + Dz (x)} = min{60 + 0, 1 + 5} = 6



- Of course, with our global view of the network, we can see that this new cost via z is wrong. But the only information node y has is that its direct cost to x is 60 and that z has last told y that z could get to x with a cost of 5.
- So in order to get to x, y would now route through z, fully expecting that z will be able to get to x with a cost of 5.
- As of $t_1$ we have a routing loop in order to get to x, y routes through z, and z routes through y.
- A routing loop is like a black hole a packet destined for x arriving at y or z as of $t_1$ will bounce back and forth between these two nodes forever (or until the forwarding tables are changed).
- Since node y has computed a new minimum cost to x, it informs z of its new distance vector at time $t_1$.
- Sometime after $t_1$ , z receives y's new distance vector, which indicates that y's minimum cost to x is 6.

- z knows it can get to y with a cost of 1 and hence computes a new least cost to x of Dz (x) = min{50 + 0,1 + 6} = 7.
- Since z’s least cost to x has increased, it then informs y of its new distance vector at $t_2$.
- In a similar manner, after receiving z’s new distance vector, y determines D y (x) = 8 and sends z its distance vector. z then determines Dz (x) = 9 and sends y its distance vector, and so on.
- This problem of ending with routing loops is called Counting to infinity problem.(As this looping may continue forever if there is a link break, which is the only link connecting the node to the network.)

**Solution to counting to infinity problem:**

The specific looping scenario just described can be avoided using a technique known as Split horizon poisoned reverse. When the algorithm goes through the iterations beyond the number of nodes in the network, the nodes update their default state as initial state by updating the distance to all the nodes as infinity. The shortest path computation begins from the initial state.

## Differentiate between Link State and Distance vector routing algorithms

**The DV and LS algorithms take complementary approaches towards computing routing.**

**Message complexity:**

- LS requires each node to know the cost of each link in the network. This requires O(|N| |E|) messages to be sent. Also, whenever a link cost changes, the new link cost must be sent to all nodes
- The DV algorithm requires message exchanges between directly connected neighbors at each iteration.  When link costs change, the DV algorithm will propagate the results of the changed link cost only if the new link cost results in a changed least-cost path for one of the nodes attached to that link.

**Speed of convergence:**

- Implementation of LS is an O(|N|$^2$) algorithm requiring O(|N| |E|)) messages.
- The DV algorithm can converge slowly and can have routing loops while the algorithm is converging. DV also suffers from the count-to-infinity problem.

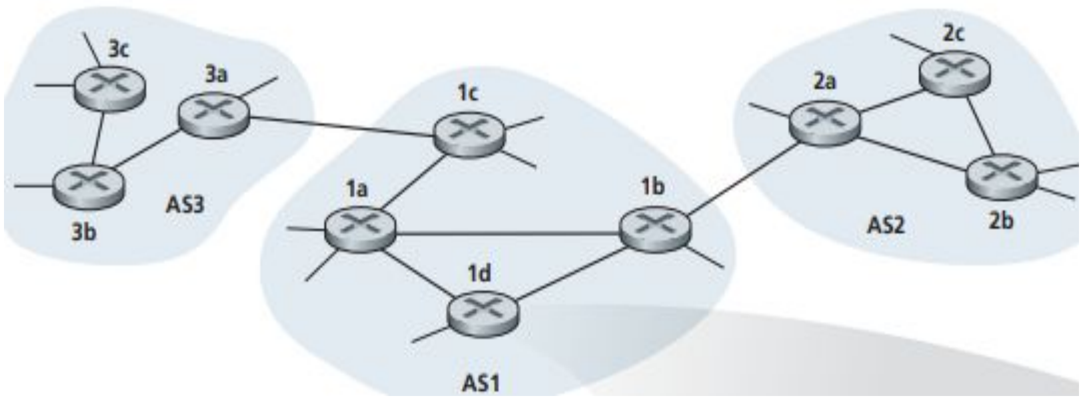
**Robustness:**

- If a router fails, misbehaves, or is sabotaged, under LS, a router could broadcast an incorrect cost for one of its attached links (but no others). An LS node is computing only its own forwarding tables; other nodes are performing similar calculations for themselves. This means route calculations are somewhat separated under LS, providing a degree of robustness.
- Under DV, a node can advertise incorrect least-cost paths to any or all destinations.

## Write a short note on Hierarchical routing

- Today's public Internet consists of hundreds of millions of hosts. Storing routing information at each of these hosts would clearly require enormous amounts of memory.This would result in two major problems. **Scaling of networks** and Maintaining **autonomy of the network** operations.
- Both of these problems can be solved by organizing routers into autonomous systems (ASs), with each AS consisting of a group of routers that are typically under the same administrative control. Routers within the same AS all run the same routing algorithm (for example, an LS or DV algorithm) and have information about each other exactly as was the case in our idealized model in the preceding section.
- The routing algorithm running within an autonomous system is called an intra autonomous system routing protocol. It will be necessary, of course, to connect ASs to each other, and these routers which connects the different ASs are called gateway routers.

- Forwarding table is configured by both intra- and inter-AS routing algorithm
- Intra-AS sets entries for internal dests
- Inter-AS & Intra-As sets entries for external dests



**Inter-AS tasks:**

Suppose router in AS1 receives datagram for which dest is outside of AS1 Router should forward packet towards on of the gateway routers, but which one? to learn which destinations are reachable through AS2 and which through AS3 to propagate this reachability info to all routers in AS1

- The problems of scale and administrative authority are solved by defining autonomous systems. Within an AS, all routers run the same intra-AS routing protocol. Among themselves, the ASs run the same inter-AS routing protocol. The problem of scale is solved because an intra-AS router need only know about routers within its AS.
- The problem of administrative authority is solved since an organization can run whatever intra-AS routing protocol it chooses; however, each pair of connected ASs needs to run the same inter-AS routing protocol to exchange reachability information. In the following section

## Explain the working of OSPF

**OSPF routing is widely used for intra-AS routing in the Internet.**

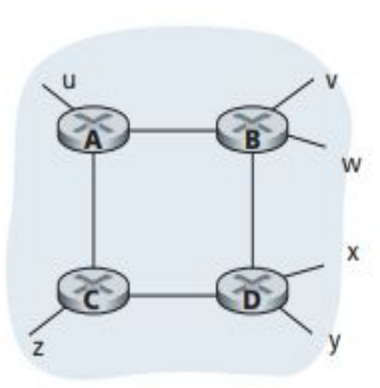
- OSPF is a link-state protocol that uses flooding of link-state information and a Dijkstra least-cost path algorithm.
- With OSPF, a router constructs a complete topological map (that is, a graph) of the entire autonomous system.
- The router then locally runs Dijkstra's shortest-path algorithm to determine a shortest-path tree to all subnets, with itself as the root node.
- Individual link costs are configured by the network administrator (see Principles and Practice: Setting OSPF Weights). The administrator might choose to set all link costs to 1, thus achieving minimum-hop routing, or might choose to set the link weights to be inversely proportional to link capacity in order to discourage traffic from using low-bandwidth links.
- OSPF does not mandate a policy for how link weights are set (that is the job of the network administrator), but instead provides the mechanisms (protocol) for determining least-cost path routing for the given set of link weights

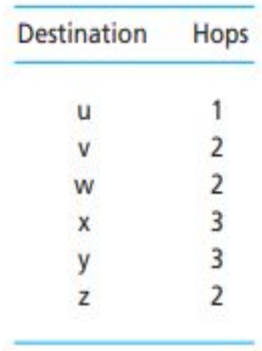
- With OSPF, a router broadcasts routing information to all other routers in the autonomous system, not just to its neighboring routers.
- The OSPF protocol must itself implement functionality such as reliable message transfer and link-state broadcast. The OSPF protocol also checks that links are operational (via a HELLO message that is sent to an attached neighbor) and allows an OSPF router to obtain a neighboring router's database of network-wide link state.
- Exchanges between OSPF routers (for example, link-state updates) can be authenticated. With authentication, only trusted routers can participate in the OSPF protocol within an AS, thus preventing malicious intruders

- Two types of authentication can be configured simple and MD5. With simple authentication, the same password is configured on each router. When a router sends an OSPF packet, it includes the password in plaintext.
- Multiple same-cost paths. When multiple paths to a destination have the same cost, OSPF allows multiple paths to be used (that is, a single path need not be chosen for carrying all traffic when multiple equal-cost paths exist).
- Integrated support for unicast and multicast routing. Multicast OSPF (MOSPF) [RFC 1584] provides simple extensions to OSPF to provide for multicast routing
- Support for hierarchy within a single routing domain. Perhaps the most significant advance in OSPF is the ability to structure an autonomous system hierarchically.

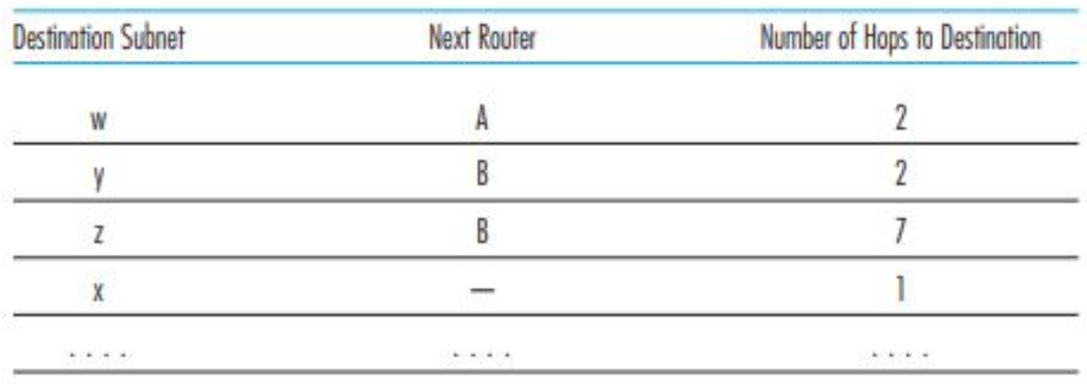
## Explain RIP protocol

- RIP (Routing Information Protocol): RIP was one of the earliest intra-AS Internet routing protocols and is still in widespread use today.
- RIP is a distance-vector protocol that operates in a manner very close to the idealized DV protocol(Bellman Ford algorithm)
- In RIP (and also in OSPF), costs are actually from source router to a destination subnet. RIP uses the term hop, which is the number of subnets traversed along the shortest path from source router to destination subnet, including the destination subnet.





| Destination | Hops |
|---|---|
| u | 1 |
| v | 2 |
| w | 2 |
| x | 3 |
| y | 3 |
| z | 2 |



| Destination Subnet | Next Router | Number of Hops to Destination |
|---|---|---|
| w | A | 2 |
| y | B | 2 |
| z | B | 7 |
| x | — | 1 |
| .... | .... | .... |

- Note that the routing table has three columns. The first column is for the destination subnet, the second column indicates the identity of the next router along the shortest path to the destination subnet, and the third column indicates the number of hops (that is, the number of subnets that have to be traversed, including the destination subnet) to get to the destination subnet along the shortest path.
- Distance vectors: exchanged among neighbors every 30 sec via Response Message (also called advertisement)
- Each advertisement: list of up to 25 destination nets within AS.

## RIP Link failure and recovery:

If no advertisement heard after 180 sec --> neighbor/link declared dead

- routes via neighbor invalidated
- new advertisements sent to neighbors
- neighbors in turn send out new advertisements (if tables changed)
- link failure info quickly propagates to entire net
- poison reverse used to prevent ping-pong loops (infinite distance = 16 hops)

## Explain Broadcast routing algorithms

There are three different approaches in broadcast algorithms.

1. **N-wayunicast approach**

- Given N destination nodes, the source node simply makes N copies of the packet, addresses each copy to a different destination, and then transmits the N copies to the N destinations using unicast routing

**Drawbacks:**

- The first drawback is its inefficiency. If the source node is connected to the rest of the network via a single link, then N separate copies of the (same) packet will traverse this single link
- An implicit assumption of N-way-unicast is that broadcast recipients, and their addresses, are known to the sender. This would add more overhead and, importantly, additional complexity to a protocol

- A final drawback of N-way-unicast relates to the purposes for which broadcast is to be used

**2. Uncontrolled Flooding:**

- The most obvious technique for achieving broadcast is a flooding approach in which the source node sends a copy of the packet to all of its neighbors. When a node receives a broadcast packet, it duplicates the packet and forwards it to all of its neighbors (except the neighbor from which it received the packet).

**Drawback:**

- If the graph has cycles, then one or more copies of each broadcast packet will cycle indefinitely.
- When a node is connected to more than two other nodes, it will create and forward multiple copies of the broadcast packet, each of which will create multiple copies of itself (at other nodes with more than two neighbors), and so on creating a broadcast storm.

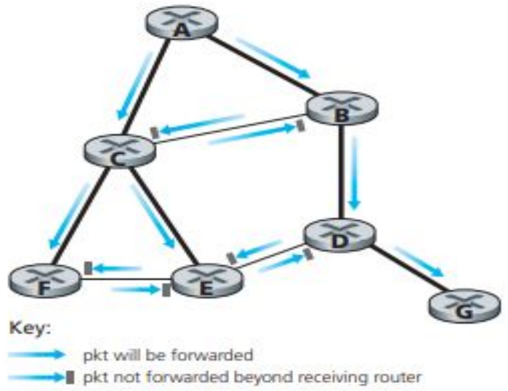
## 3. Controlled Flooding:

a. **Sequence-number-controlled flooding:**

- A source node puts its address (or other unique identifier) as well as a broadcast sequence number into a broadcast packet, then sends the packet to all of its neighbors.
- Each node maintains a list of the source address and sequence number of each broadcast packet it has already received, duplicated, and forwarded.
- When a node receives a broadcast packet, it first checks whether the packet is in this list. If so, the packet is dropped; if not, the packet is duplicated and forwarded to all the node's neighbors (except the node from which the packet has just been received).

## b. Reverse Path Forwarding (RPF):

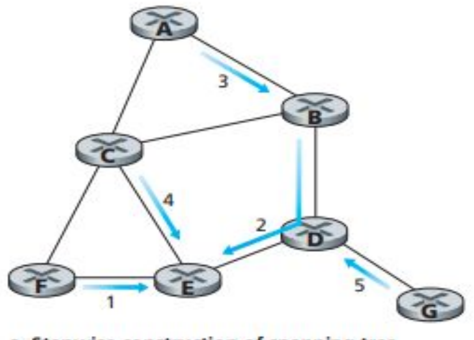
- When a router receives a broadcast packet with a given source address, it transmits the packet on all of its outgoing links (except the one on which it was received) only if the packet arrived on the link that is on its own shortest unicast path back to the source. Otherwise, the router simply discards the incoming packet without forwarding it on any of its outgoing links.
- RPF need only know the next neighbor on its unicast shortest path to the sender; it uses this neighbor's identity only to determine whether or not to flood a received broadcast packet.



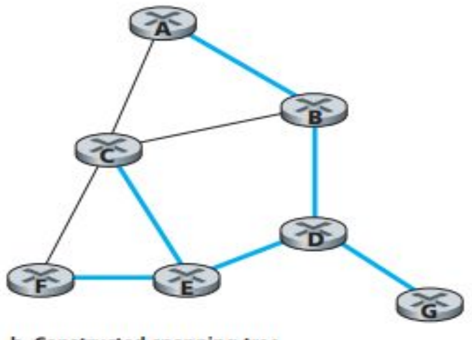
- Suppose that the links drawn with thick lines represent the least-cost paths from the receivers to the source (A).
- Node A initially broadcasts a source-A packet to nodes C and B. Node B will forward the source-A packet it has received from A (since A is on its least-cost path to A) to both C and D.
- B will ignore (drop, without forwarding) any source-A packets it receives from any other nodes (for example, from routers C or D).
- Let us now consider node C, which will receive a source-A packet directly from A as well as from B.
- Since B is not on C's own shortest path back to A, C will ignore any source-A packets it receives from B. On the other hand, when C receives a source-A packet directly from A, it will forward the packet to nodes B, E, and F.

## C. Spanning-Tree Broadcast:

- While sequence-number-controlled flooding and RPF avoid broadcast storms, they do not completely avoid the transmission of redundant broadcast packets.Ideally, every node should receive only one copy of the broadcast packet.
- This approach first constructs a spanning tree. When a source node wants to send a broadcast packet, it sends the packet out on all of the incident links that belong to the spanning tree. A node receiving a broadcast packet then forwards the packet to all its neighbors in the spanning tree (except the neighbor from which it received the packet).



a. Stepwise construction of spanning tree



b. Constructed spanning tree

- Not only does spanning tree eliminate redundant broadcast packets, but once in place, the spanning tree can be used by any node to begin a broadcast, as shown in the above given figures (a) and (b). Note that a node need not be aware of the entire tree; it simply needs to know which of its neighbors in G are spanning-tree neighbors.

## Explain the working of Multicast routing algorithms

- In multicast service, a multicast packet is delivered to only a subset of network nodes.
- A multicast packet is addressed using address indirection. That is, a single identifier is used for the group of receivers, and a copy of the packet that is addressed to the group using this single identifier is delivered to all of the multicast receivers associated with that group.
- In the Internet, the single identifier that represents a group of receivers is a class D multicast IP address. The group of receivers associated with a class D address is referred to as a multicast group.

Two approaches have been adopted for determining the multicast routing tree,

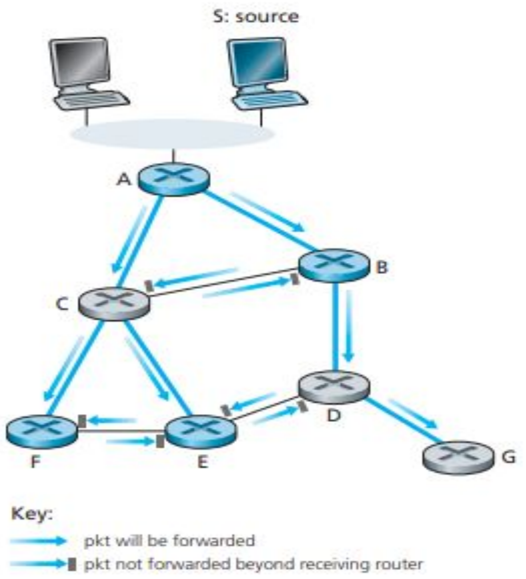
1. Multicast routing using a group-shared tree
2. Multicast routing using a source-based tree

**1. Multicast routing using a group-shared tree:**

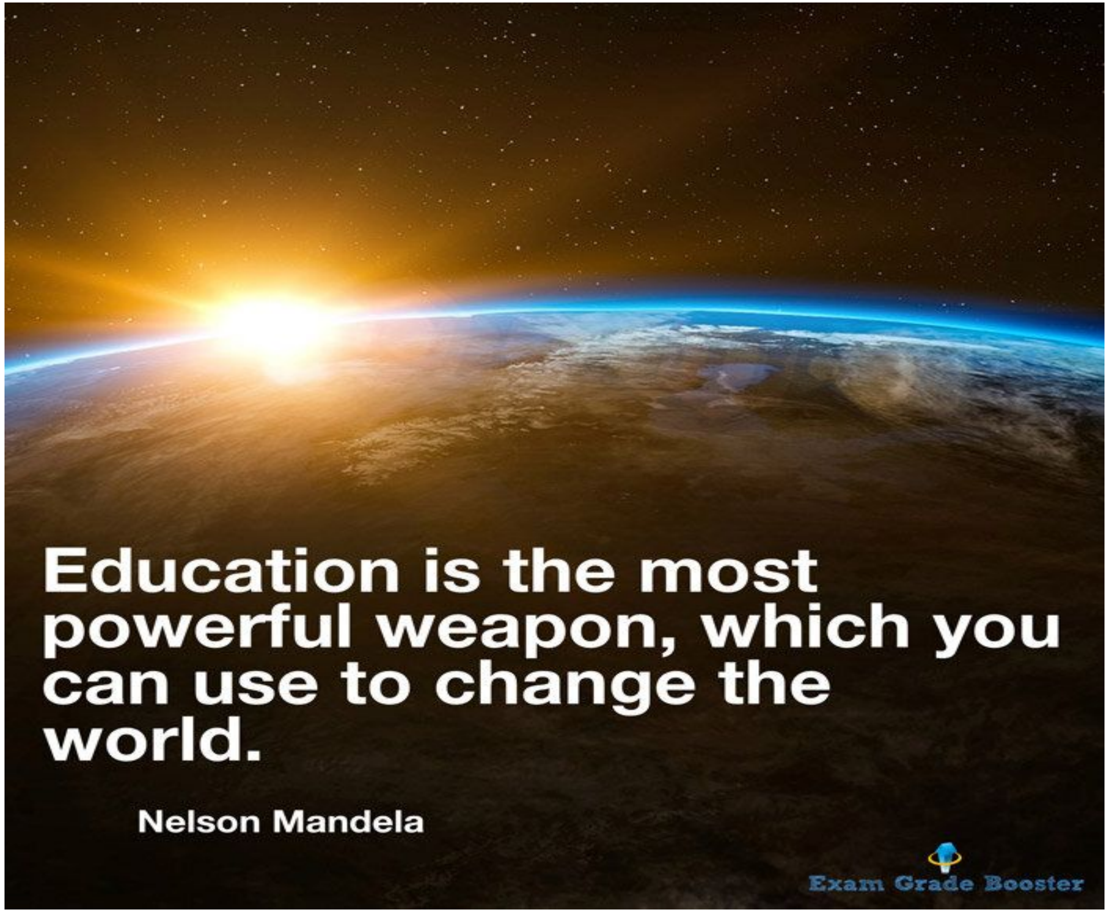
- Multicast routing over a group-shared tree is based on building a tree that includes all edge routers with attached hosts belonging to the multicast group.
- In practice, a center-based approach is used to construct the multicast routing tree, with edge routers with attached hosts belonging to the multicast group sending(via unicast) join messages addressed to the center node.
- A join message is forwarded using unicast routing toward the center until it either arrives at a router that already belongs to the multicast tree or arrives at the center.
- All routers along the path that the join message follows will then forward received multicast packets to the edge router that initiated the multicast join.

## 2. Multicast routing using a source-based tree:

- Approach constructs a multicast routing tree for each source in the multicast group.
- In practice, an RPF algorithm (with source node x) is used to construct a multicast forwarding tree for multicast datagrams originating at source x.



- In order to avoid unnecessary forward of the group messages to the routers which does not have any group member connected to it, this method makes use of pruning concept.
- A multicast router that receives multicast packets and has no attached hosts joined to that group will send a prune message to its upstream router. If a router receives prune messages from each of its downstream routers, then it can forward a prune message upstream.


Education is the most powerful weapon, which you can use to change the world.
Nelson Mandela
Exam Grade Booster