UNIT-5
Network Layer

Syllabus: **Network Layer**: Design Issues, Store and Forward Packet Switching, Connection Oriented and Connectionless Networks-Routing algorithm: Optimality Principle, shortest path routing, Flooding, Hierarchical routing, Broad cast, Multi cast, distance vector routing, Link State routing, Control to Infinity Problem.

Network layer:

- The network layer is concerned with getting packets from the source all the way to the destination (**End-to-End Delivery**). Which is in contrast with the data link layer which is responsible for **Hop-to-Hop Delivery**.
- The network layer is the lowest layer that deals with end-to-end transmission.
- To achieve its goals, the network layer must know about the topology of the communication subnet (i.e., the set of all routers) and choose appropriate paths through it.
- It must also take care to choose routes to avoid overloading some of the communication lines and routers while leaving others idle.
- When the source and destination are in different networks, new problems occur. It is up to the network layer to deal with them.

Design issues of Network layer:
Network layer has the following design issues

- Store-and-Forward Packet Switching
- Services Provided to the Transport Layer
- Implementation of Connectionless Service
- Implementation of Connection-Oriented Service
- Comparison of Virtual-Circuit and Datagram Networks

Store-and-Forward Packet Switching:

- A host with a packet to send transmits it to the nearest router, either on its own LAN or over a point-to-point link to the ISP. The packet is stored there until it has fully arrived and the link has finished its processing by verifying the checksum.
- Then it is forwarded to the next router along the path until it reaches the destination host, where it is delivered. This mechanism is store-and-forward packet switching.
- The major components of the network are the ISP's equipment (routers connected by transmission lines), shown inside the shaded oval, and the customers' equipment, shown outside the oval.
- Host *H1* is directly connected to one of the ISP's routers, *A*, perhaps as a home computer that is plugged into a DSL modem.
- In contrast, *H2* is on a LAN, which might be an office Ethernet, with a router, *F*, owned and operated by the customer. This router has a leased line to the ISP's equipment. We have shown *F* as being outside the oval because it does not belong to the ISP.

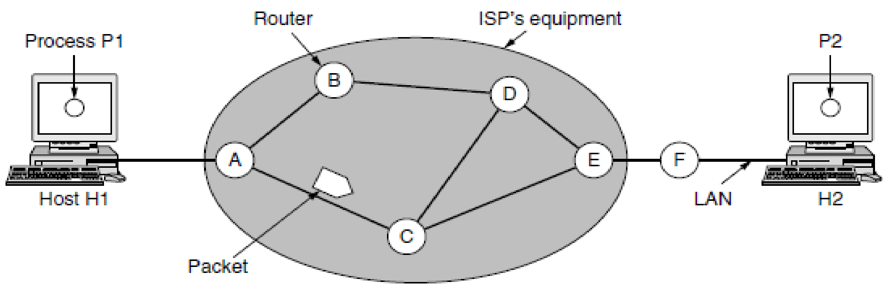


Figure 5-1. The environment of the network layer protocols.

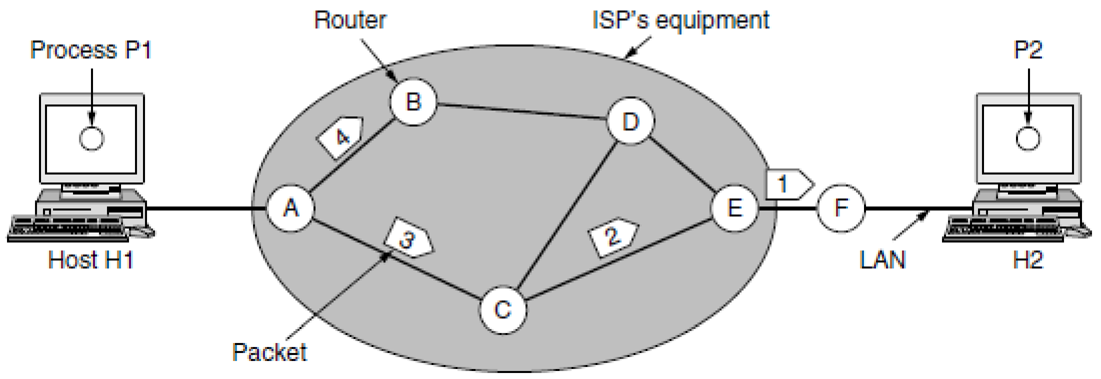
Services Provided to the Transport Layer:

- The network layer provides services to the transport layer at the network layer/transport layer interface. The services need to be carefully designed with the following goals in mind:

  1. The services should be independent of the router technology.
  2. The transport layer should be shielded from the number, type, and topology of the routers present.
  3. The network addresses made available to the transport layer should use a uniform numbering plan, even across LANs and WANs.

Implementation of 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** (in analogy with telegrams) and the network is called a **datagram network**.
- Let us now see how a datagram network works. Suppose that the process *P1* in Figure has a long message for *P2*. It hands the message to the transport layer, with instructions to deliver it to process *P2* on host *H2*.
- The transport layer code runs on *H1*, typically within the operating system. It prepends a transport header to the front of the message and hands the result to the network layer.
- 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 sends each of them in turn to router A using some point-to-point protocol, for example, PPP.



A's table (initially)

Dest.	Line
A	–
B	B
C	C
D	B
E	C
F	C

A's table (later)

A	–
B	B
C	C
D	B
E	B
F	B

C's table

A	A
B	A
C	–
D	E
E	E
F	E

E's table

A	C
B	D
C	C
D	D
E	–
F	F

Figure: Routing within a datagram network

- Every router has an internal table telling it where to send packets for each possible destination.
- 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 the figure, A has only two outgoing lines—to B and C—so every incoming packet must be sent to one of these routers, even if the ultimate destination is some other router. A's initial routing table is shown in the figure under the label ''initially.''
- As they arrived at A, packets 1, 2, and 3 were stored briefly (to verify their checksums). Then each was forwarded to C according to A's table. Packet 1 was then forwarded to E and then to F. When it got to F, it was encapsulated in a data link layer frame and sent to H2 over the LAN. Packets 2 and 3 follow the same route.
- However, something different happened to packet 4. When it got to A it was 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.
- Perhaps it 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.

Implementation of Connection-Oriented Service:

- If connection-oriented service is used, a path from the source router to the destination router must be established before any data packets can be sent. This connection is called a VC (virtual circuit), in analogy with the physical circuits set up by the telephone system, and the subnet is called a virtual-circuit subnet.
- The idea behind virtual circuits is to avoid having to choose a new route for every packet sent. 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.

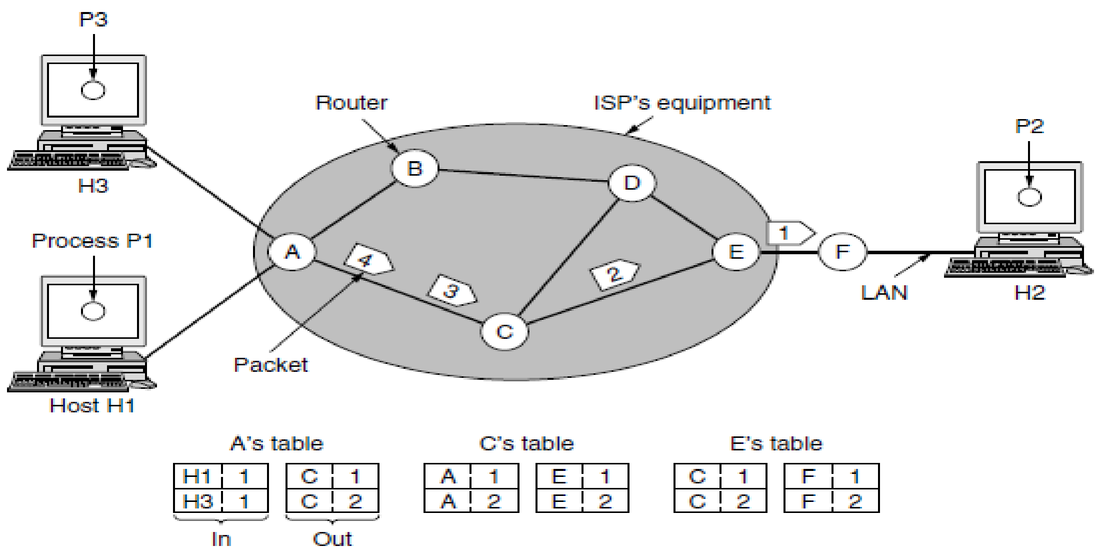


Figure: Routing within a virtual circuit network

- As an example, consider the above figure, Here, host H1 has established connection 1 with host H2. It 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 subnet 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 is called label switching.

Comparison of Virtual-Circuit and Datagram Networks:

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:

- The main function of the network layer is routing packets from the source machine to the destination machine.
- In most subnets, packets will require multiple hops to make the journey. The only notable exception is for broadcast networks, but even here routing is an issue if the source and destination are not on the same network.
- The algorithms that choose the routes and the data structures that they use are a major area of network layer design.
- **The routing algorithm** is that part of the network layer software responsible for deciding which output line an incoming packet should be transmitted on.

- If the subnet uses **datagrams** internally, this decision must be made anew for every arriving data packet since the best route may have changed since last time.
- If the subnet uses **virtual circuits** internally, routing decisions are made only when a new virtual circuit is being set up. Thereafter, data packets just follow the previously-established route. This is sometimes called session routing because a route remains in force for an entire user session.
- Router perform two tasks
    - **Routing:** Making the decision which routes to use.
    - **Forwarding:** Looking up the outgoing line to use for it in the routing tables.
- **Properties of routing algorithms**:
    - Correctness
    - Simplicity
    - Robustness
    - Stability
    - Fairness, and
    - Optimality
- Routing algorithms can be grouped into two major classes:
    1. **Non-adaptive Algorithms:**
       Static and offline. They will not work if there is any failure of link or modification of the subnet. Here the path is calculated prior to the data transmission. Non-adaptive algorithms do not base their routing decisions on measurements or estimates of the current traffic and topology. This procedure is sometimes called static routing.
       Examples: Shortest path routing and Flooding

    2. **Adaptive algorithms:**
       Dynamic and online. These algorithms change their routing decisions to reflect changes in the topology, and usually the traffic as well. Adaptive algorithms differ in where they get their information (e.g., locally, from adjacent routers, or from all routers), when they change the routes (e.g., every ?T sec, when the load changes or when the topology changes), and what metric is used for optimization (e.g., distance, number of hops, or estimated transit time).

- **Optimality principle:** It states that if 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.
- We different types of Routing algorithms as listed below,
    - Shortest path routing
    - Flooding
    - Hierarchical routing
    - Broad cast routing
    - Multi cast routing
    - Distance vector routing
    - Link State routing

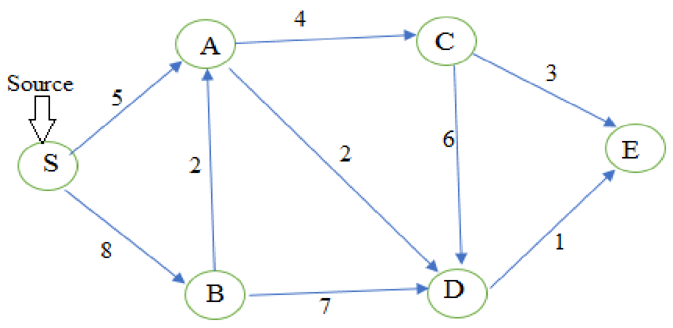
Shortest Path Routing:

- The subnet is considered as a graph, with each node of the graph representing a router and each arc of the graph representing a communication line (often called a link).
- To choose a route between a given pair of routers, the algorithm just finds the shortest path between them on the graph. One way of measuring path length is the number of hops. Another metric is the geographic distance in kilometres.
- The labels on the arcs could be computed as a function of the distance, bandwidth, average traffic, communication cost, mean queue length, measured delay, and other factors. By changing the weighting function, the algorithm would then compute the ''shortest'' path measured according to any one of a number of criteria or to a combination of criteria.

Dijkstra's algorithm:

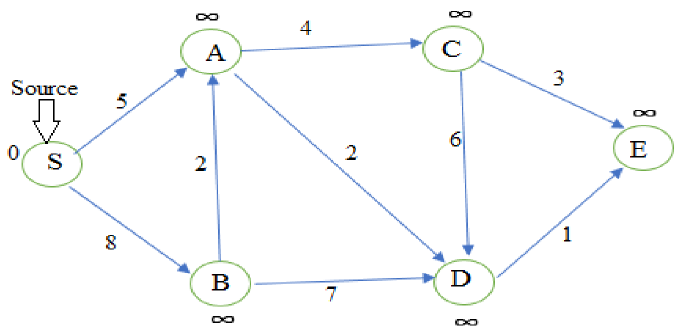
- It is a shortest path routing algorithm in which the shortest path is measured from single source to all destinations in the subnet graph.
- Each node is labelled with its distance from the source node along the best-known path.
- Initially, no paths are known, so all nodes are labelled with infinity. As the algorithm proceeds and paths are found, the labels may change, reflecting better paths.
- A label may be either tentative or permanent. Initially, all labels are tentative. When it is discovered that a label represents the shortest possible path from the source to that node, it is made permanent and never changed thereafter.
- **Algorithm or Procedure:**
  1. Assign to every node a tentative distance value: set it to zero for our initial node and to infinity for all other nodes.
  2. Mark all nodes unvisited. Set the initial node as current. Create a set of the unvisited nodes called the *unvisited set* consisting of all the nodes except the initial node.
  3. For the current node, consider all of its unvisited neighbors and calculate their *tentative* distances. For example, if the current node A is marked with a distance of 6, and the edge connecting it with a neighbor B has length 2, then the distance to B (through A) will be 6+2=8. If this distance is less than the previously recorded tentative distance of B, then overwrite that distance. Even though a neighbor has been examined, it is not marked as "visited" at this time, and it remains in the *unvisited set*.
  4. When we are done considering all of the neighbors of the current node, mark the current node as visited and remove it from the *unvisited set*. A visited node will never be checked again.
  5. If the destination node has been marked visited (when planning a route between two specific nodes) or if the smallest tentative distance among the nodes in the *unvisited set* is infinity (when planning a complete traversal), then stop. The algorithm has finished.
  6. Select the unvisited node that is marked with the smallest tentative distance and set it as the new "current node" then go back to step 3.

- **Example:**



Iteration-0:

From the above graph given node S is the source, so first make distance of 'S' as '0' and for all remaining nodes make distance as '∞'.



Iteration-1:

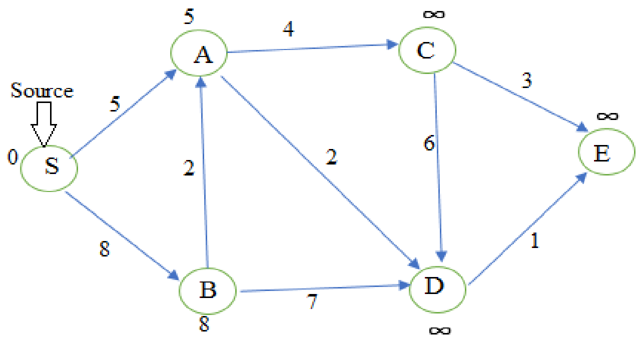
Current node: S

Adjacent nodes of S(Un-visited): A,B

initially=> Dist(A)= ∞ & Dist(B)= ∞

Now update the distances of A, B from S:

- Dist(A)=Dist(S)+Cost(S,A)=0+5=5, now 5<∞ therefore **Dist(A)=5**
- Dist(B)=Dist(S)+Cost(S,B)=0+8=5, now 5<∞ therefore **Dist(B)=8**



Iteration-2:

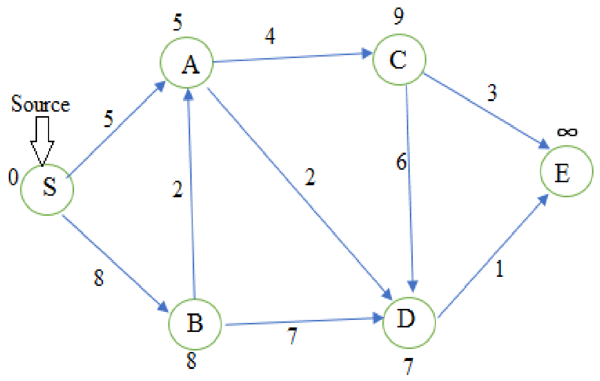
From iteration-1, Between A & B , A is minimum

So choose Current-node: A

Adjacent nodes of A(un-visited): C & D

Now update the distances of C &D from A:

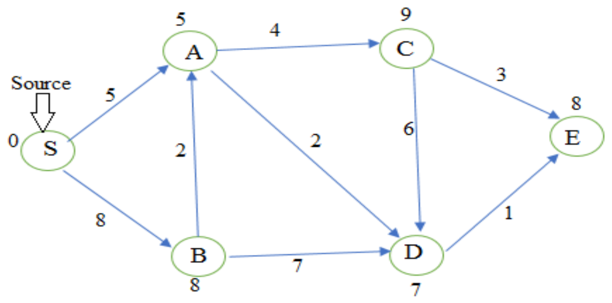
- Dist(C)=Dist(A)+Cost(A,C)=5+4=9, now 9<∞ therefore **Dist(C)=9**
- Dist(D)=Dist(A)+Cost(A,D)=5+2=7, now 7<∞ therefore **Dist(D)=5**



Iteration-3:

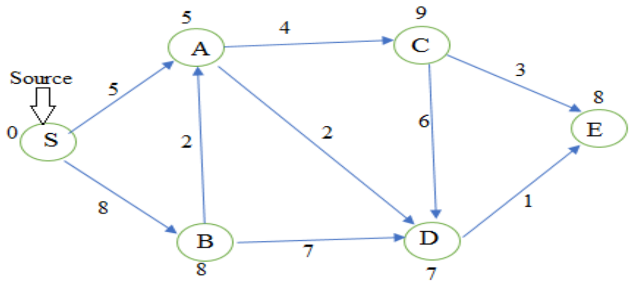
From iteration-2, Among B,C & D, D is minimum
So, choose Current-node: D
Adjacent nodes of D (un-visited): E
Now update the distance of E from D:

- ➢ Dist(E)=Dist(D)+Cost(D,E)=7+1=8, now 8<∞ therefore **Dist(E)=8**



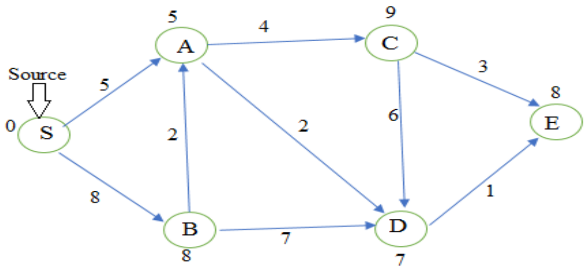
Iteration-4:

From iteration-3, Among B,C & E, B & E minimum
So, choose any one, now Current-node: B
Adjacent nodes of D (un-visited): Empty
No need of updating



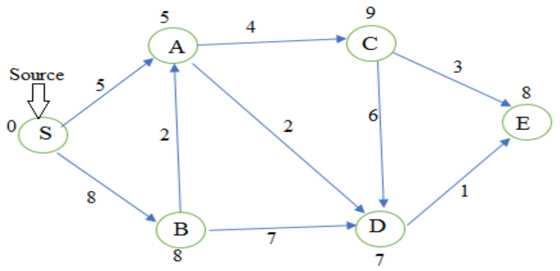
Iteration-5:

From iteration-4, Between C & E, E is minimum
So, choose Current-node: E
Adjacent nodes of E (un-visited): Empty
No need of updating



Iteration-6:

From iteration-5, only C
So, choose Current-node: C
Adjacent nodes of C (un-visited): Empty
No need of updating



Iteration table:

Iteration	Current node	visited	Un-visited	S	A	B	C	D	E
0	Empty	Empty	{S,A,B,C,D,E,}	0	∞	∞	∞	∞	∞
1	S	{S}	{A,B,C,D,E}	0	5	8	∞	∞	∞
2	A	{S,A}	{B,C,D,E}	0	5	8	9	5	∞
3	D	{S,A,D}	{B,C,E}	0	5	8	9	5	8
4	B	{S,A,D,B}	{C,E}	0	5	8	9	5	8
5	E	{S,A,D,B,E}	{C}	0	5	8	9	5	8
6	C	{S,A,D,B,E,C}	Empty	0	5	8	9	5	8

****

Flooding:

- Principle of flooding: a node (or a packet switch) forwards an incoming packet to all ports except to the one it arrived on.
- Each node (a switch) performs the flooding process such that the packet will reach the destination as long as at least one path exists between the source and the destination.
- Flooding is useful
  - ➢ when the information in the routing tables is not available, such as during system startup,
  - ➢ when survivability is required, such as in military networks.
  - ➢ when the source needs to send a packet to all hosts connected to the network (i.e., broadcast delivery).
- Flooding generates vast numbers of duplicate packets

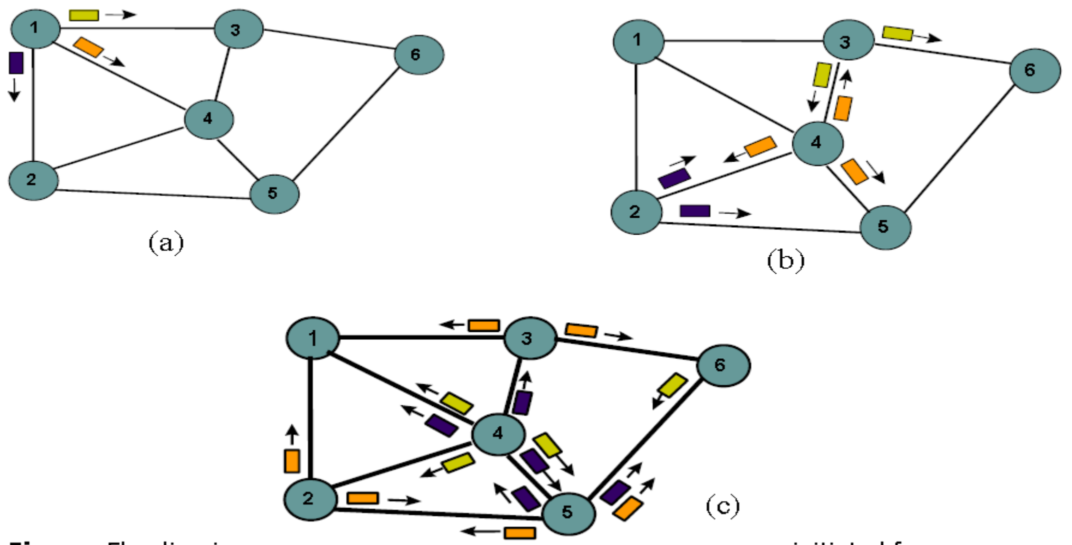


Figure: Flooding is initiated from Node 1: (a) Hop 1 transmissions (b) Hop 2 transmissions (c) Hop 3 transmissions

- In figure, initially one packet arriving at node 1 triggers three packets to nodes 2, 3, and 4. In the second phase nodes 2, 3, and 4 send two, two, and three packets respectively. These packets arrive at nodes 2 through 6. In the third phase 15 more packets are generated giving a total of 25 packets after three phases. The flooding needs to be controlled so that packets are not generated excessively.

How to control flooding?

There are three methods to reduce the resource consumption in the network

1) Use a time-to-live (TTL) field in each packet.

- ➢ When the source sends a packet, the time-to-live field is initially set to some small number.
- ➢ Each node decrements the field by one before flooding the packet. If the value reaches zero, the switch discards the packet.
- ➢ To avoid unnecessary waste of bandwidth, the time-to-live should ideally be set to the minimum hop number between two furthest nodes (called the diameter of the network).

2) Add an identifier before flooding

- ➢ Every node adds an identifier before flooding
- ➢ When a node identifies a packet that contains the identifier of the switch, it discards the packet.
- ➢ This method effectively prevents a packet from going around a loop.

3) Have a unique sequence number

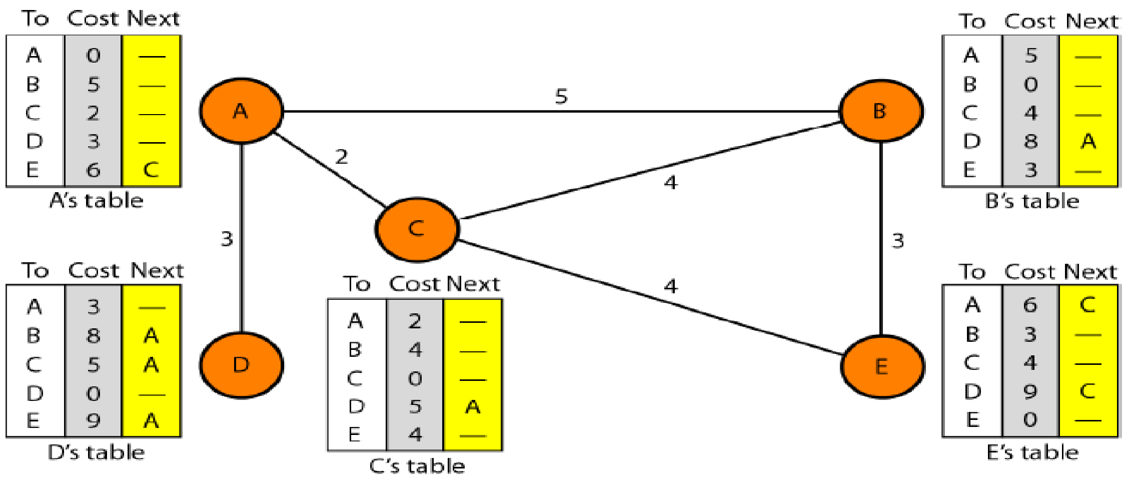
- ➢ Each packet from the given source is uniquely identified with a sequence number
- ➢ When a node receives a packet, it records the source address and the sequence number

- If node discovers that packet has already visited the node, it will discard the packet

Distance vector routing:

In **distance vector routing,** the least-cost route between any two nodes is the route with minimum distance. In this protocol, as the name implies, each node maintains a vector (table) of minimum distances to every node. The table at each node also guides the packets to the desired node by showing the next stop in the route (next-hop routing).

Figure *Distance vector routing tables*



To	Cost	Next
A	0	—
B	5	—
C	2	—
D	3	—
E	6	C

To	Cost	Next
A	5	—
B	0	—
C	4	—
D	8	A
E	3	—

To	Cost	Next
A	3	—
B	8	A
C	5	A
D	0	—
E	9	A

To	Cost	Next
A	2	—
B	4	—
C	0	—
D	5	A
E	4	—

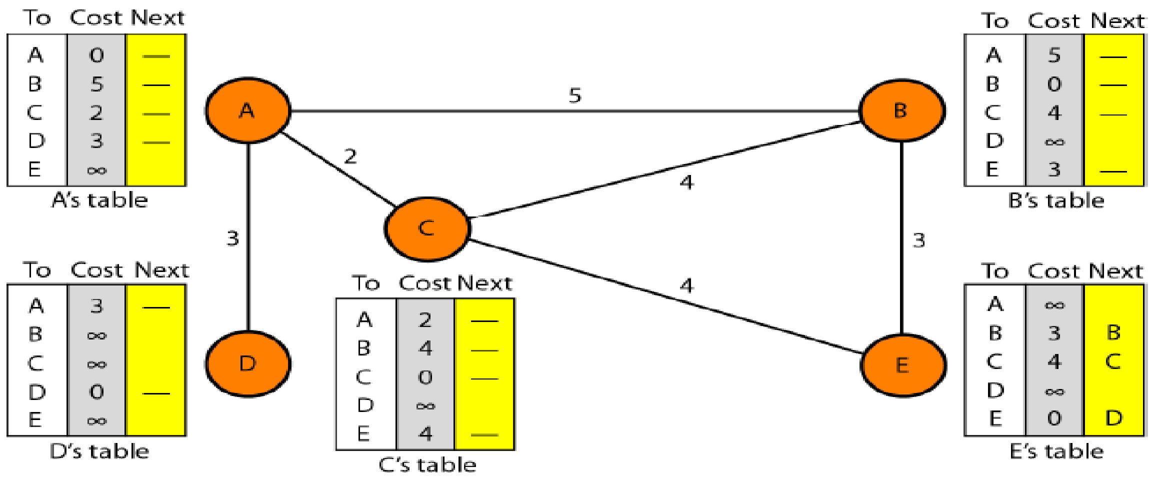
To	Cost	Next
A	6	C
B	3	—
C	4	—
D	9	C
E	0	—

Initialization of routing table:

Initialization

The tables in Figure are stable; each node knows how to reach any other node and the cost. At the beginning, however, this is not the case. Each node can know only the distance between itself and its **immediate neighbors,** those directly connected to it. So for the moment, we assume that each node can send a message to the immediate neighbors and find the distance between itself and these neighbors. Below figure shows the initial tables for each node. The distance for any entry that is not a neighbor is marked as infinite (unreachable).

Figure *Initialization of tables in distance vector routing*



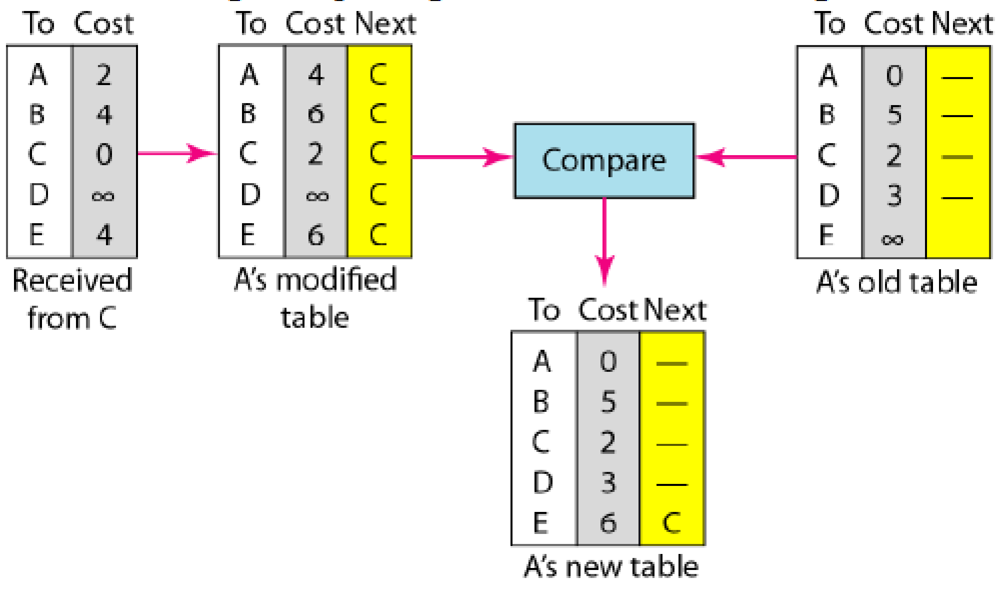
- In distance vector routing, each node shares its routing table with its immediate neighbours periodically and when there is a change.

Updating

When a node receives a two-column table from a neighbor, it needs to update its routing table. Updating takes three steps:

1. The receiving node needs to add the cost between itself and the sending node to each value in the second column. The logic is clear. If node C claims that its distance to a destination is x mi, and the distance between A and C is y mi, then the distance between A and that destination, via C, is $x + y$ mi.
2. The receiving node needs to add the name of the sending node to each row as the third column if the receiving node uses information from any row. The sending node is the next node in the route.
3. The receiving node needs to compare each row of its old table with the corresponding row of the modified version of the received table.
   a. If the next-node entry is different, the receiving node chooses the row with the smaller cost. If there is a tie, the old one is kept.
   b. If the next-node entry is the same, the receiving node chooses the new row. For example, suppose node C has previously advertised a route to node X with distance 3. Suppose that now there is no path between C and X; node C now advertises this route with a distance of infinity. Node A must not ignore this value even though its old entry is smaller. The old route does not exist any more. The new route has a distance of infinity.

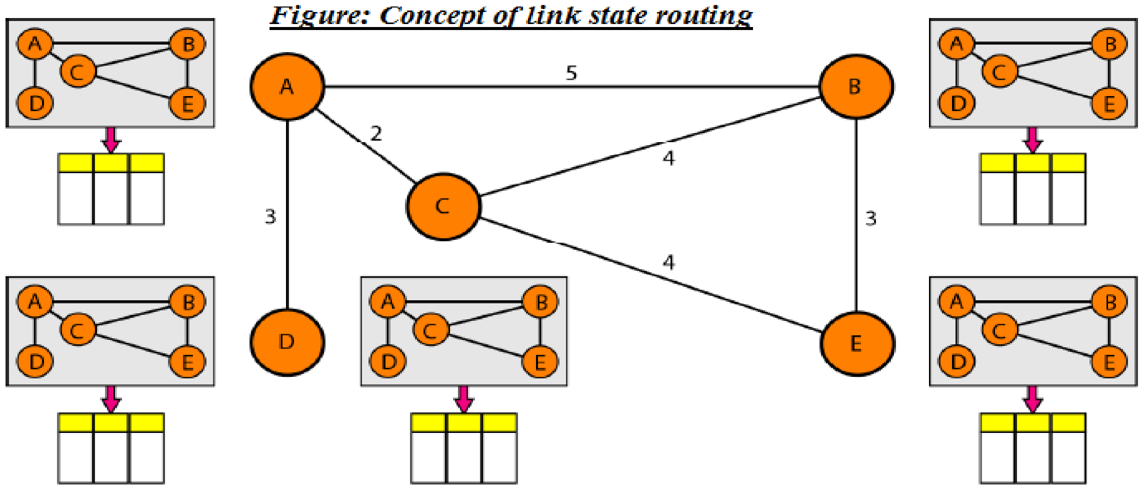
Figure: Updating in distance vector routing



Link State routing:

Link state routing has a different philosophy from that of distance vector routing. In link state routing, if each node in the domain has the entire topology of the domain—the list of nodes and links, how they are connected including the type, cost (metric), and condition of the links (up or down)—the node can use **Dijkstra's algorithm** to build a routing table. Below figure shows the concept.

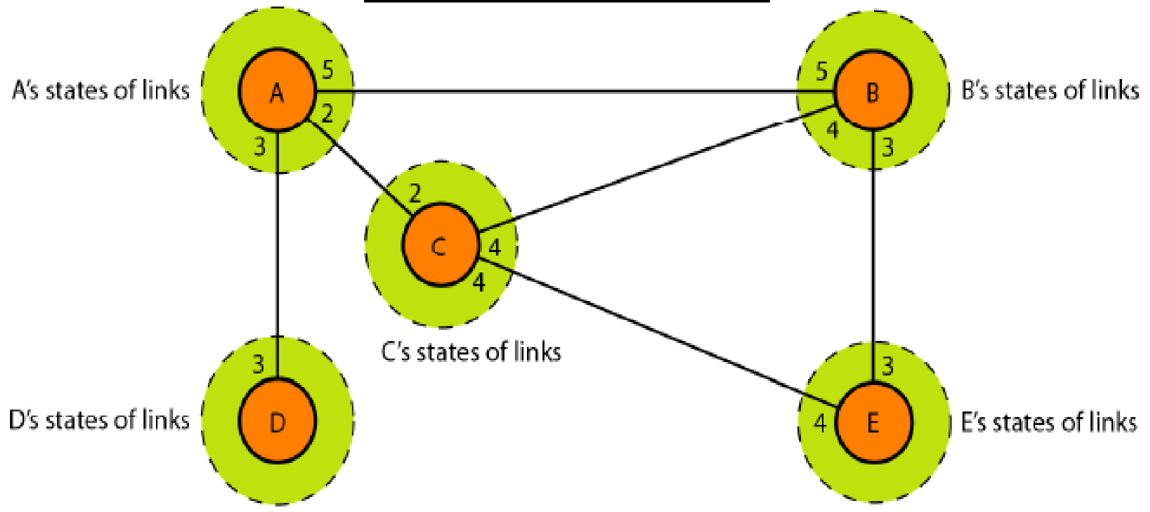
Figure: Concept of link state routing



The figure shows a simple domain with five nodes. Each node uses the same topology to create a routing table, but the routing table for each node is unique because the calculations are based on different interpretations of the topology. This is analogous to a city map. While each person may have the same map, each needs to take a different route to reach her specific destination.

The topology must be dynamic, representing the latest state of each node and each link. If there are changes in any point in the network (a link is down, for example), the topology must be updated for each node.

Figure: Linkstate Knowledge

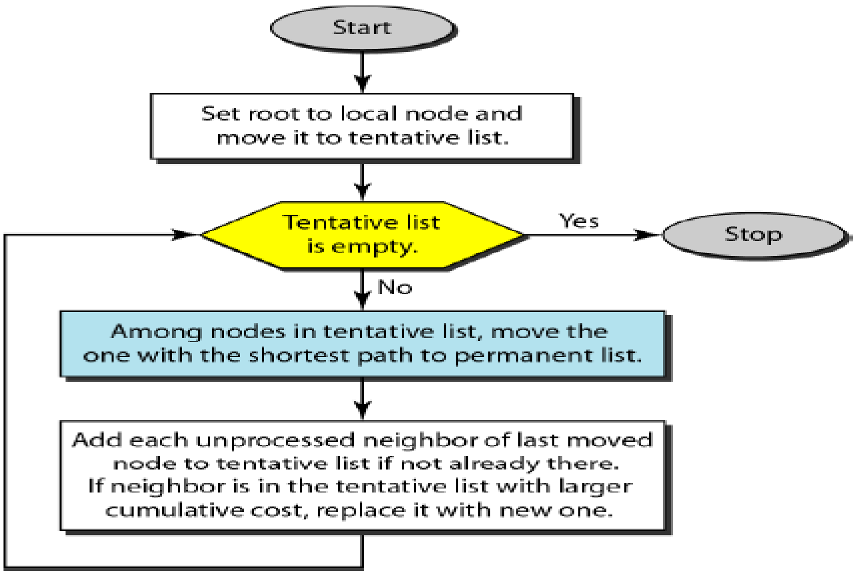


Building Routing Tables

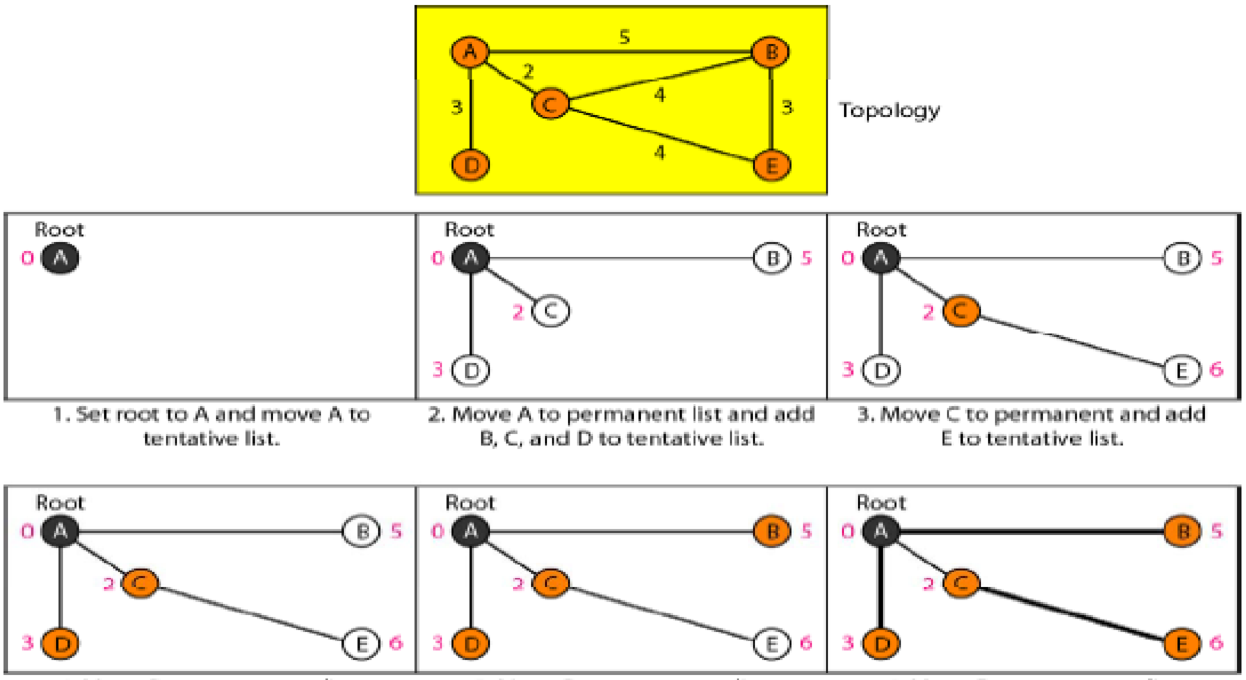
In **link state routing,** four sets of actions are required to ensure that each node has the routing table showing the least-cost node to every other node.

1. Creation of the states of the links by each node, called the link state packet (LSP).
2. Dissemination of LSPs to every other router, called **flooding,** in an efficient and reliable way.
3. Formation of a shortest path tree for each node.
4. Calculation of a routing table based on the shortest path tree.

Formation of shortest path using Dijkstra's:


Start
Set root to local node and move it to tentative list.
Tentative list is empty.
Yes
Stop
No
Among nodes in tentative list, move the one with the shortest path to permanent list.
Add each unprocessed neighbor of last moved node to tentative list if not already there. If neighbor is in the tentative list with larger cumulative cost, replace it with new one.

Example for formation of shortest path tree:



1. Set root to A and move A to tentative list.
2. Move A to permanent list and add B, C, and D to tentative list.
3. Move C to permanent and add E to tentative list.
4. Move D to permanent list.
5. Move B to permanent list.
6. Move E to permanent list (tentative list is empty).

Calculation of Routing Table from Shortest Path Tree Each node uses the shortest path tree protocol to construct its routing table. The routing table shows the cost of reaching each node from the root. Table shows the routing table for node A.

Table *Routing table for node A*

Node	*Cost*	*Next Router*
A	0	—
B	5	—
C	2	—
D	3	—
E	6	C

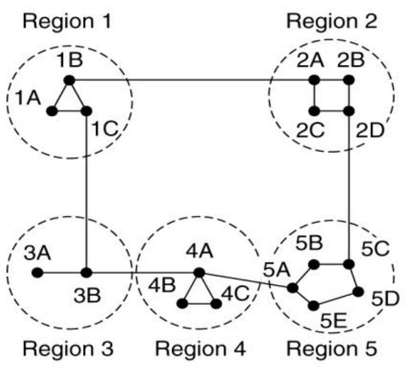
****

Hierarchical routing:

- As networks grow in size, the router routing tables grow proportionally.
- There are some problems with the increase in network size
  - Router memory consumed by increasing tables
  - CPU time is needed to scan routing table and
  - More bandwidth is needed to send status reports about routing table.

so, the routing will have to be done hierarchically, as it is in the telephone network.

- When hierarchical routing is used, the routers are divided into regions, with each router knowing all the details about how to route packets to destinations within its own region but knowing nothing about the internal structure of other regions.
- When different networks are interconnected, it is natural to regard each one as a separate region in order to free the routers in one network from having to know the topological structure of the other ones.
- 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, until we run out of names for aggregations.
- Example:
  - The below figure (a) gives an example of routing in a two-level hierarchy with five regions.
  - The full routing table for router 1A has 17 entries, as shown in Figure (b)
  - When routing is done hierarchically, as in Figure (c) there are entries for all the local routers as before, but all other regions have been 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.



(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)

Figure: (a) Subnet (b) Full table of 1A (c) Hierarchical table of 1A

- For example, consider a subnet with 720 routers.
  If there is no hierarchy, each router needs 720 routing table entries.
  If the subnet is partitioned into 24 regions of 30 routers each, each router needs 30 local entries plus 23 remote entries for a total of 53 entries.
  If a three-level hierarchy is chosen, with eight clusters, each containing 9 regions of 10 routers, each router needs 10 entries for local routers, 8 entries for routing to other regions within its own cluster, and 7 entries for distant clusters, for a total of 25 entries.
- Kamoun and Kleinrock (1979) discovered that the optimal number of levels for an N router subnet is **ln N**, requiring a total of e ln N entries per router.

****

Broadcast Routing:

- **Broadcasting:** Sending a packet to all destinations simultaneously is called broadcasting. The algorithms used for broadcasting are called the " Broadcast Routing algorithms".
- For example, a service distributing weather reports, stock market updates, or live radio programs might work best by broadcasting to all machines and letting those that are interested read the data.
- There are various methods for broadcast routing:
  1. Distinct point-to-point routing
  2. Flooding
  3. Multi Destination routing
  4. Spanning Tree Routing
  5. Reverse path Forwarding
- **Distinct point-to-point routing:**
  - The source simply sends a distinct packet to each destination.
  - **Drawbacks:**
    - This method wasteful of bandwidth
    - It also requires the source to have a complete list of all destinations.
- **Flooding**:
  - Flooding is the forwarding of a packet from any node to every other node attached to the router except the node from which the packet arrived.

- Flooding is a way to distribute routing information updates quickly to every node in a large network.
- **Drawback:** It generates too many packets and consumes too much bandwidth.

- **Multi-destination routing:**
  - If this method is used, each packet contains either a list of destinations or a bit map indicating the desired destinations.
  - When a packet arrives at a router,
    - The router checks all the destinations to determine the set of output lines that will be needed
    - The router generates a new copy of the packet for each output line to be used and includes in each packet only those destinations that are to use the line.
    - In effect, the destination set is partitioned among the output lines.
    - After a sufficient number of hops, each packet will carry only one destination and can be treated as a normal packet.
    - Multi-destination routing is like separately addressed packets, except that when several packets must follow the same route, one of them pays full fare and the rest ride free.
- **Spanning tree:**
  - A spanning tree is a subset of the subnet that includes all the routers but contains no loops.
  - If each router knows which of its lines belong to the spanning tree, it can copy an incoming broadcast packet onto all the spanning tree lines except the one it arrived on.
  - This method makes excellent use of bandwidth, generating the absolute minimum number of packets necessary to do the job.
  - **Drawback:** The each router must have knowledge of some spanning tree for the method to be applicable. Sometimes this information is available (e.g., with link state routing) but sometimes it is not (e.g., with distance vector routing).

Example:

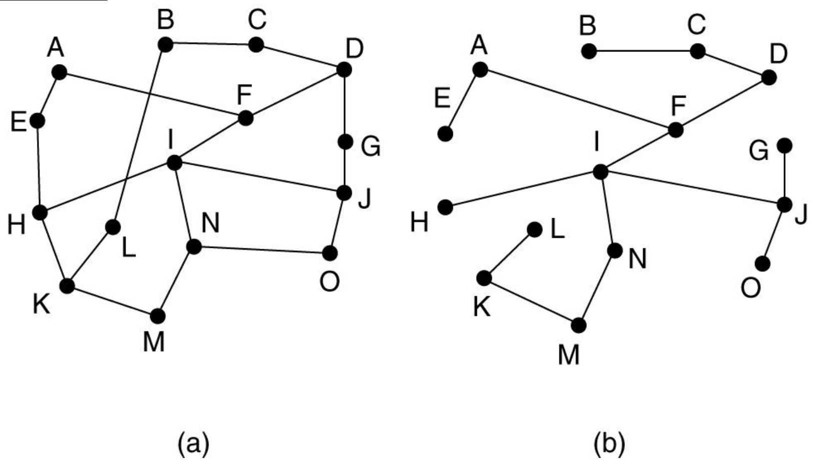


Figure (a): Subnet (b): The Spanning tree for Subnet in Figure A

- **Reverse Path Forwarding**:
  - This algorithm works even when the routers do not know anything at all about spanning trees.
  - When a broadcast packet arrives at a router,
    - The router checks to see if the packet arrived on the line that is normally used for sending packets to the source of the broadcast.

- If so, the broadcast packet itself followed the best route from the router and is therefore the first copy to arrive at the router. Then, the router forwards copies of it onto all lines except the one it arrived on.
- If, however, the broadcast packet arrived on a line other than the preferred one for reaching the source, the packet is discarded as a likely duplicate.

Example:

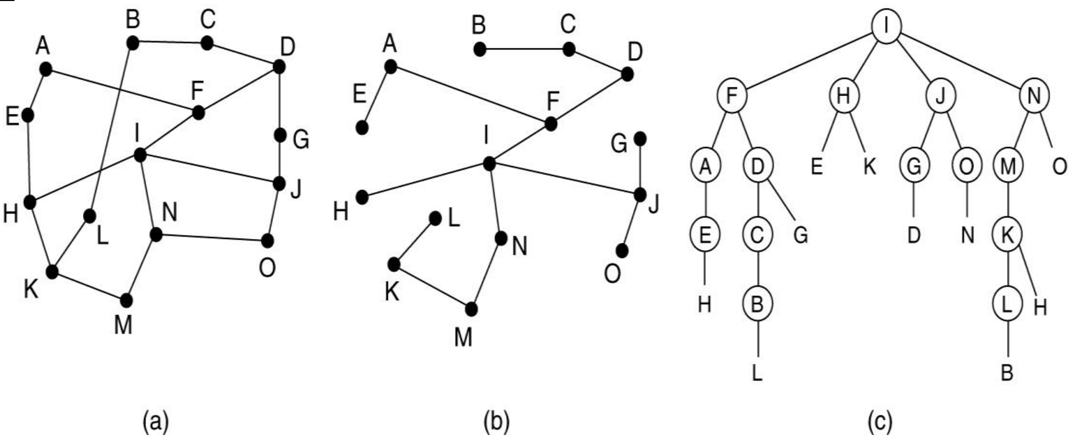


Figure (a): Subnet Figure (b) Spanning Tree for (a) (c) : Reverse path tree for (a)

On the first hop, I send packets to F, H, J, and N, as indicated by the second row of the tree. Each of these packets arrives on the preferred path to and is so indicated by a circle around the letter. On the second hop, eight packets are generated, two by each of the routers that received a packet on the first hop. As it turns out, all eight of these arrive at previously unvisited routers, and five of these arrive along the preferred line. Of the six packets generated on the third hop, only three arrive on the preferred path (at C, E, and K); the others are duplicates. After five hops and 24 packets, the broadcasting terminates, compared with four hops and 14 packets had the sink tree been followed exactly.

Advantages of reverse path forwarding:

1. Reasonably efficient and easy to implement.
2. It does not require routers to know about spanning trees
3. It does have the overhead of a destination list or bit map in each broadcast packet as does multi-destination addressing.
4. It does not require any special mechanism to stop the process, as flooding does.

Multicast Routing

- **Multicasting:** Sending a message to a group is called multicasting, and its routing algorithm is called multicast routing.
- Multicasting requires group management. Some way is needed to create and destroy groups, and to allow processes to join and leave groups.
- It is important that routers know which of their hosts belong to which groups.
- Either hosts must inform their routers about changes in group membership, or routers must query their hosts periodically.
- Routers learn about which of their hosts are in which groups.
- Routers tell their neighbours, so the information propagates through the subnet.

- To do multicast routing, each router computes a spanning tree covering all other routers.

<u>Example:</u>

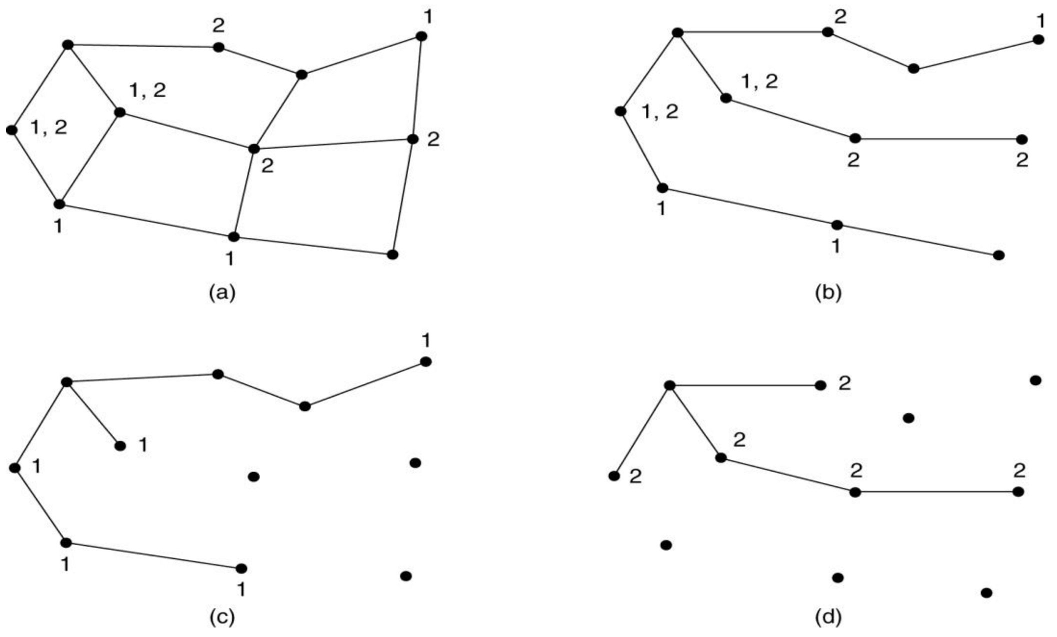


Figure: (a) A network. (b) A spanning tree for the leftmost router. (c) A multicast tree for group 1. (d) A multicast tree for group 2.

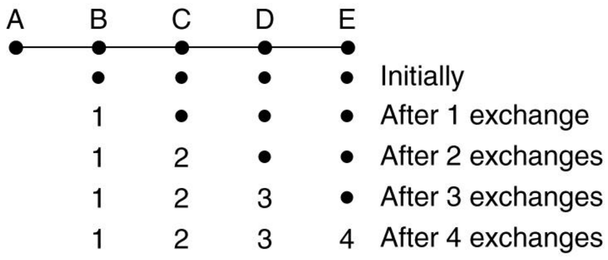
- In The we have two groups, 1 and 2. Some routers are attached to hosts that belong to one or both of these groups, as indicated in the figure.
- A spanning tree for the leftmost router is shown in Figure (b).
- When a process sends a multicast packet to a group, the first router examines its spanning tree and prunes it, removing all lines that do not lead to hosts that are members of the group.
- In our example, Figure (c) shows the pruned spanning tree for group 1.
- Similarly, Figure (d) shows the pruned spanning tree for group 2. Multicast packets are forwarded only along the appropriate spanning tree.

Disadvantage: This algorithm is that it scales poorly to large networks. Suppose that a network has **n** groups, each with an average of **m** members. For each group, m pruned spanning trees must be stored, for a total of **mn** trees. When many large groups exist, considerable storage is needed to store all the trees.

- **Core-based trees** is an alternative for multicast routing. Here, a single spanning tree per group is computed, with the root (the core) near the middle of the group. To send a multicast message, a host sends it to the core, which then does the multicast along the spanning tree. Although this tree will not be optimal for all sources, the reduction in storage costs from m trees to one tree per group is a major saving.

Count-to-Infinity Problem:

- Distance vector routing works in theory but has a serious drawback in practice: although it converges to the correct answer, it may do so slowly. In particular, it reacts rapidly to good news, but leisurely to bad news.
- Consider a router whose best route to destination X is large. If on the next exchange neighbor, A suddenly reports a short delay to X, the router just switches over to using the line to A to send traffic to X. In one vector exchange, the good news is processed.
- To see how fast good news (No Link Failure) propagates,
- Consider the five-node (linear) subnet of figure below, where the delay metric is the number of hops.
    - Suppose A is down initially and all the other routers know this. In other words, they have all recorded the delay to A as infinity.
    - When A comes up, the other routers learn about it via the vector exchanges.
    - At the time of the first exchange, B learns that its left neighbor has zero delay to A. B now makes an entry in its routing table that A is one hop away to the left.
    - All the other routers still think that A is down. At this point, the routing table entries for A are as shown in the second row of figure below. On the next exchange, C learns that B has a path of length 1 to A, so it updates its routing table to indicate a path of length 2, but D and E do not hear the good news until later. Clearly, the good news is spreading at the rate of one hop per exchange.



A	B	C	D	E	
	●	●	●	●	Initially
	1	●	●	●	After 1 exchange
	1	2	●	●	After 2 exchanges
	1	2	3	●	After 3 exchanges
	1	2	3	4	After 4 exchanges

Figure: Propagation of good news

- In a subnet whose longest path is of length N hops, within N exchanges everyone will know about newly-revived lines and routers.

- Bad News Propagation:
    - Now let us consider the situation of, in which all the lines and routers are initially up. Routers B, C, D, and E have distances to A of 1, 2, 3, and 4, respectively.
    - Suddenly A goes down (Link failure), or alternatively, the line between A and B is cut, which is effectively the same thing from B's point of view.
    - At the first packet exchange, B does not hear anything from A.
    - Fortunately, C says: Do not worry; I have a path to A of length 2. Little does B know that C's path runs through B itself. For all B knows, C might have ten lines all with separate paths to A of length 2. As a result, B thinks it can reach A via C, with a path length of 3. D and E do not update their entries for A on the first exchange.
    - On the second exchange, C notices that each of its neighbors claims to have a path to A of length 3. It picks one of the them at random and makes its new distance to A 4, as shown in the third row of Figure below. Subsequent exchanges produce the history shown in the rest of.

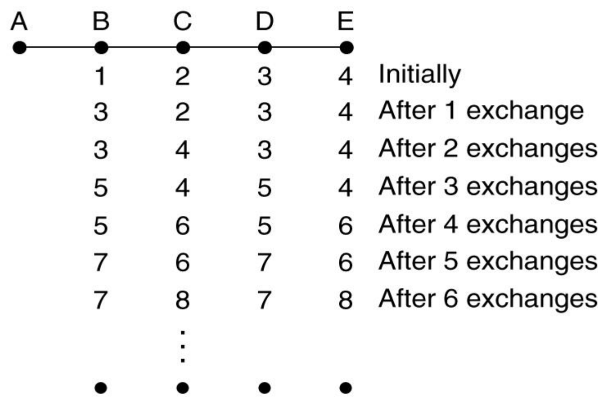

A B C D E
1 2 3 4 Initially
3 2 3 4 After 1 exchange
3 4 3 4 After 2 exchanges
5 4 5 4 After 3 exchanges
5 6 5 6 After 4 exchanges
7 6 7 6 After 5 exchanges
7 8 7 8 After 6 exchanges

Figure: Bad news Propagation

- From this figure, it should be clear why bad news travels slowly. Gradually, all routers work their way up to infinity, but the number of exchanges required depends on the numerical value used for infinity.
- For this reason, it is wise to set infinity to the longest path plus 1. If the metric is time delay, there is no well-defined upper bound, so a high value is needed to prevent a path with a long delay from being treated as down.
- This problem is known as the count-to-infinity problem.

****