UNIT 2 ROUTING ALGORITHMS

Structure **Page Nos.**

2.0 INTRODUCTION

As you have studied earlier, the main function of the network layer is to find the best route from a source to a destination. In routing, the route with the minimum cost is considered to be the best route. For this purpose, a router plays an important role. On the basis of cost of a each link, a router tries to find the optimal route with the help of a good routing algorithm. There are a large number of routing algorithms. These algorithms are a part of the network layer and are responsible for deciding on which output line an incoming packet should be transmitted. Some of these routing algorithms are discussed in this unit.

2.1 OBJECTIVES

After going through this unit, you should be able to:

- understand how the shortest path routing algorithm works;
- draw a spanning tree;
- understand the functioning of distance vector routing and link state routing, and
- understand and implement multicast routing.

2.2 FLOODING

Consider an example of any network topology (VC subnet) in which, there are some link failures or in which, a few routers are not operational. These failures will cause changes in the network typology which have to be communicated to all the nodes in the network. This is called **broadcasting**. These could be many such examples of broadcasting in which the message has to be passed on to the entire network.

A widely used broadcasting method, known as **flooding**, operates in the following manner. The source node sends its information in the form of a packet to those nodes in a subnet to which it is directly connected by a link. These nodes relay it further to their neighbours who are also directly connected, and so on, until the packet reaches

all nodes in the network. For example, in the *Figure 1(a)*, R1 will send its packets to R2 and R3. R2 will send the packet to R5 and R4. Two additional rules are also applied in order to limit the number of packets to be transmitted. First, a node will not relay the packet back to the node from which the packet was obtained. For example, R2 will not send the packet back to R1 if, it has received it from R1. Second, a node will transmit the packet to its neighbours at most once; this can be ensured by including on the packet the ID number of the origin node and a sequence number, which is incremented with each new packet issued by the origin node. By storing the highest sequence number received for each node, and by not relaying packets with sequence numbers that are less than or equal to the one stored, a node can avoid transmitting the same packet more than once, on each of its incident links. On observing these rules, you will notice that, links need not preserve the order of packet transmissions; the sequence numbers can be used to recognise the correct order. The following figure gives an example of flooding and illustrates how, the total number of packet transmission per packet broadcast lies between L and 2L, where L is the number of bi-directional links of the network. In this *Figure 1(a)* Packet broadcasting from router R1 to all other nodes by using flooding [as in *Figure 1(a)*] or a spanning tree [as in *Figure 1(b)*]. Arrows indicate packet transmissions at the time shown. Each packet transmission time is assumed to be one unit. Flooding requires at least as many packet transmissions as the spanning tree method and usually many more. In this example, the time required for the broadcast packet to reach all nodes is the same for the two methods. In general, however, depending on the choice of the spanning tree, the time required for flooding may be less than for the spanning tree method. The spanning tree is used to avoid the looping of packets in the subnet.

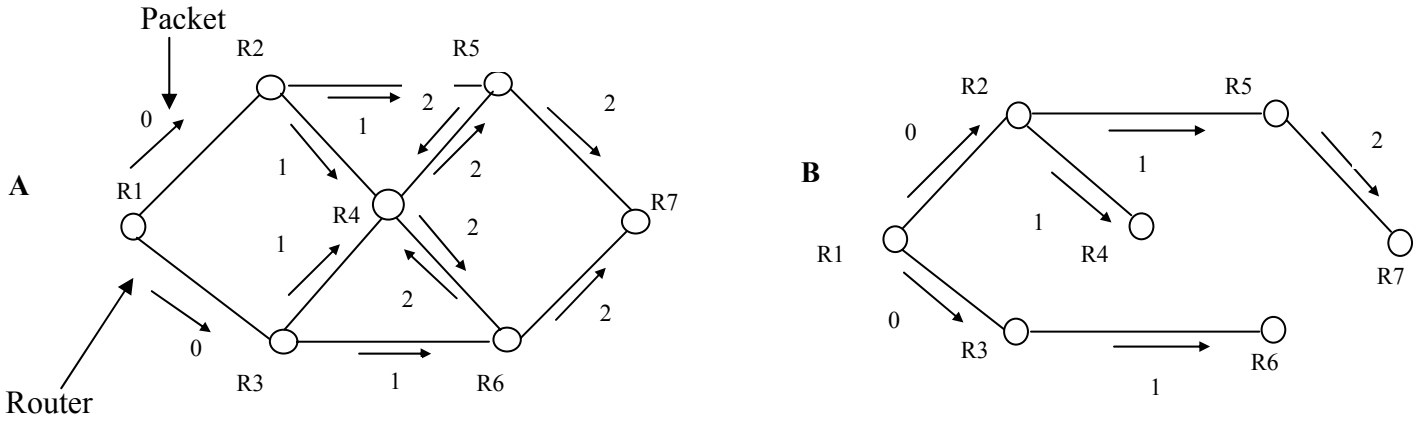


Figure 1(a): Flooding **Figure 1(b): Spanning Tree**

Figure 1: Broadcasting (*Source: Ref. [2]*)

Student should refer to [2] for further reading on this topic.

☞ Check Your Progress 1

1) Following statements are True or False

(i) Dijkstra algorithm divides the node into two sets i.e., tentative and permanent. T ☐ F ☐

(ii) Flooding generates lots of redundant packets. T ☐ F ☐

(iii) Flooding discovers only the optimal routes T ☐ F ☐

2) What is a spanning tree?

……...

……..

……..….

2.3 SHORTEST PATH ROUTHING ALGORITHM

Shortest path routing algorithm is a simple and easy to understand technique. The basic idea of this technique is to build a graph of the subnet, with each node of the graph representing a router and each arc of the graph representing a communication line i.e., link. For finding a route between a given pair of routers, the algorithm just finds the shortest path between them on the graph. The length of a path can be measured in a number of ways as on the basis of the number of hops, or on the basis of geographic distance etc.

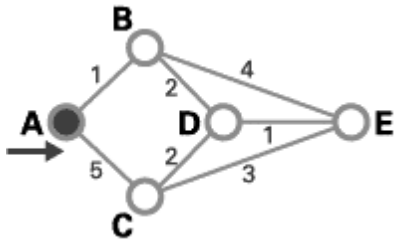
There are a number of algorithms for computing the shortest path between two nodes of a graph. One of the most used algorithm is the **Dijkstra algorithm**. This is explained below:

In this algorithm, each node has a label which represents its distance from the source node along the best known path. On the basis of these labels, the algorithm divides the node into two sets i.e., *tentative* and *permanent*. As in the beginning no paths are known, so all labels are tentative. The Algorithm works in the following manner:

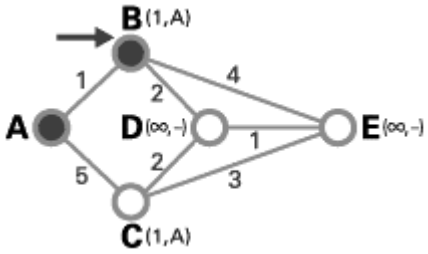
1) First mark source node as current node (T-node)
2) Find all the neighours of the T-node and make them tentative.
3) Now examine all these neigbour.
4) Then among all these node, label one node as permanent (i.e., node with the lowest weight would be labeled as permanent) and mark it as the T-node.
5) If, the desination node is reached or tentative list is empty then stop, else go to step 2.

An example of **Dijkstra** routing algorithm is explained in *Figure 2*: In this example, we want to find the best route between A and E. Here, we will show permanent nodes with filled circles and T-nodes with the --> symbol.

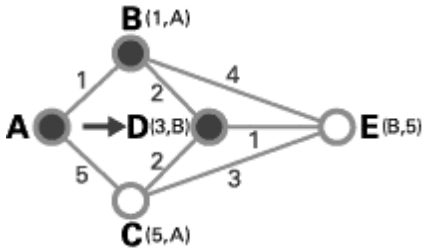
1) As shown in the *Figure* below, the source node (A) has been chosen as T-node, and so its label is permanent.



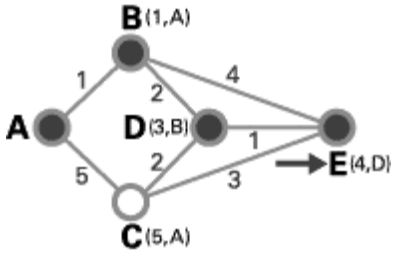
2) In this step, as you see B, C are the tentative nodes directly linked to T-node (A). Among these nodes, since B has less weight, it has been chosen as T-node and its label has changed to permanent.



3) In this step, as you see D, E are the tentative nodes directly linked to T-node(B). Among these nodes, since D has less weight, it has been chosen as T-node and its label has changed to permanent.



4) In this step, as you see C, E are the tentative nodes directly linked to T-node(D). Among these nodes, since E has less weight, it has been chosen as T-node and its label has changed to permanent.



5) E is the destination node. Now, since the destination node (E) has been, reached so, we stop here, and the shortest path is A –B –D –E.

Figure 2: Dijkstra's algorithm to compute the shortest path routing

2.4 DISTANCE VECTOR ROUTNIG

Nowadays, computer networks generally use dynamic routing algorithms rather than the static ones described above because, static algorithms do not take the current network load into account. **Distance vector routing** and **link state routing** are two main dynamic algorithms. In this section, we will go through the distance vector routing algorithm. It is also known as **Belman-Ford routing algorithm**.

The Bellman-Ford algorithm can be stated as follows: Find the shortest paths from a given source node subject keeping in mind the constraint that, the paths contain at most one link; then, find the shortest paths, keeping in mind a contraint of paths of at most two links, and so on. This algorithm also proceeds in stages. The description of the algorighm is given below.

s = source node

$w(i, j)$ = link cost from node i to node j; $w(i, j) = \infty$ if the two nodes are not directly connected; $w(i, j) \geq 0$ if the two nodes are directly connected.

h = maximum number of links in a path at the current stage of the algorithm

$L_h(n)$ = cost of the least-cost path from node s to node n under the constraint of no more than h links

1. [Initialisation]

$L_0(n) = \infty$, for all $n \neq s$
$L_h(s) = 0$, for all h

2. [Update]

For each successive $h \geq 0$:
For each $n \neq s$, compute

$$L_{h+1}(n) = \min_j [L_h(j) = w(j, n)]$$

Connect n with the predecessor node j that achieves the minimum, and eliminate any connection of n with a different predecessor node formed during an earlier iteration. The path from s to n terminates with the link from j to n.

For the iteration of step 2 with $h = K$, and for each destination node n, the algorithm compares potential paths from s to n of length $K + 1$ with the path that existed at the end of the previous iteration. If the previous, shorter path has less cost, then that path is retained. Otherwise a new path with length $K + 1$ is defined from s to n; this path consists of a path of length K from s to some node j, plus a direct hop from node j to node n. In this case, the path from s to j that is used is the K-hop path for j defined in the previous iteration.

Table 1 shows the result of applying this algorithm to a public switched network, using $s = 1$. At each step, the least-cost paths with a maximum number of links equal to h are found. After the final iteration, the least-cost path to each node and the cost of that path has been developed. The same procedure can be used with node 2 as the source node, and so on. Students should apply Dijkstra's algorithm to this subnet and observe that the result will eventually be the same.

(a)

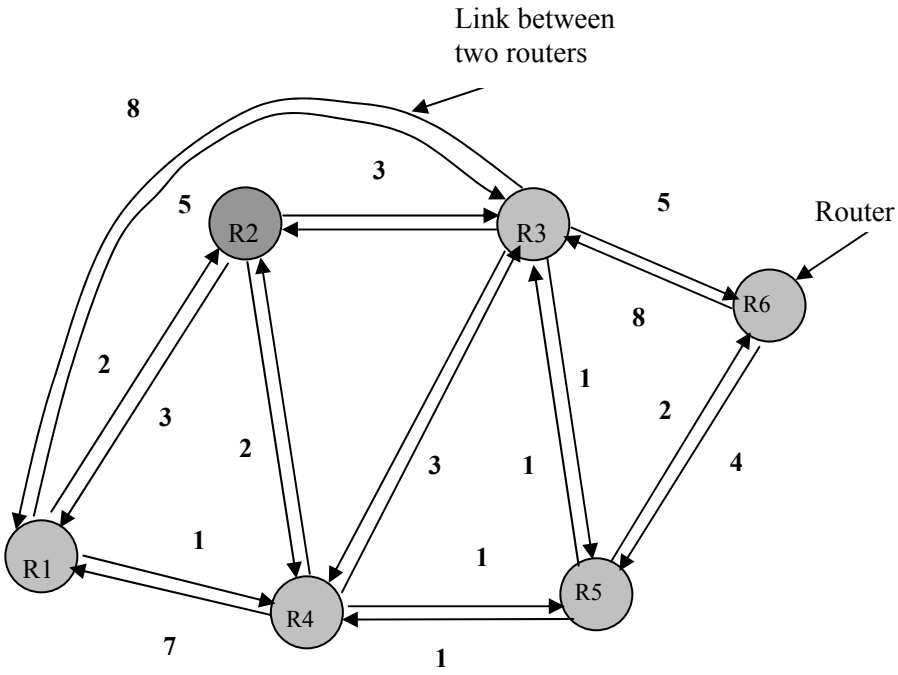


Figure 3: Public switched network

(b) Bellman-Ford Algorithm (s = 1)

Table 1: Link Cost (*Ref. Source [3]*)

H	$L_h(2)$	Path	$L_h(3)$	Path	$L_h(4)$	Path	$L_h(5)$	Path	$L_h(6)$	Path
0	∞	—	∞	—	∞	—	∞	—	∞	—
1	2	1-2	5	1-3	1	1-4	∞	—	∞	—
2	2	1-2	4	1-4-3	1	1-4	2	1-4-5	10	1-3-6
3	2	1-2	3	1-4-5-3	1	1-4	2	1-4-5	4	1-4-5-6
4	2	1-2	3	1-4-5-3	1	1-4	2	1-4-5	4	1-4-5-6

2.4.1 Comparison

Now, let us compare the two algorithms in terms of what information is required by each node to find out the optional path in the Bellman-Ford algorithm. In step 2, the calculation for node *n* involves knowledge of the link cost to all neighboruing nodes to node *n* [i.e., $w(j, n)$ plus the total path cost to each of those neighbouring nodes from a particular source node *s* (i.e., $L_h(j)$]. Each node can maintain a set of costs and associated paths for every other node in the network and, exchange this information with its direct neighbours from time to time. Each node can therefore, use the expression in step 2 of the Bellman-Ford algorithm, based only on information from its neighbours and knowledge of its link costs, to update it costs and paths. On the other hand, consider Dijkstra's algorithm. Step 3, it appears, required that each node have complete topological information about the network. That is, each node must know the link costs of all links in the network. Thus, for this algorithm, information must be exchanged with all other nodes.

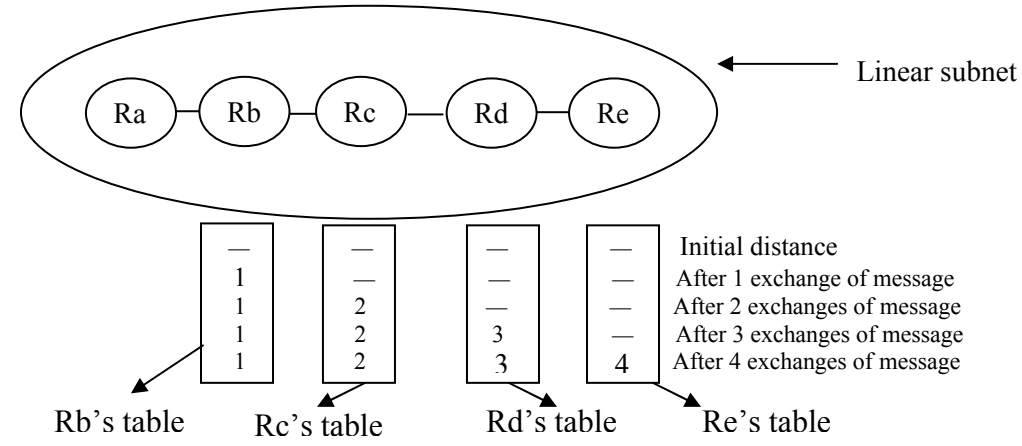
In general, an evaluation of the relative merits of the two algorithms should consider the processing time of the algorithms and the amount of information that must be collected from other nodes in the network or internet. The evaluation will depend on the implementation appraoch and the specific implementation.

A final point: Both algorithm are known to converge under static conditions of topology, and link costs and will converge to the same solution. If the link costs change over time the algorithm will attempt to catch up with these changes. However, if the link cost depends on traffic, which in turn depends on the routes chosen, then a feedback condition exists, that could result in instablities.

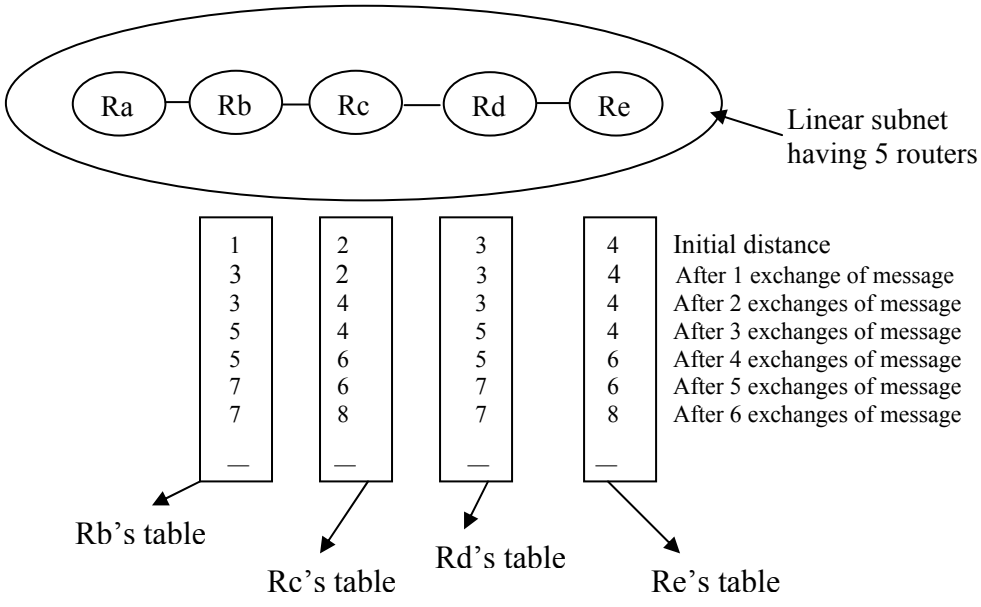
2.4.2 The Count-to-Infinity Problem

One of the serious drawbacks of the Bellman-Food algorithm is that it quickly responds to a path will a shorter delay but, responds slowly to a path with a longer delay. This is also known as count to infinity problem. Consider a subnet in which a router, whose best route to destination *X* is large. If, on the next exchange neighbour, *A* suddenly reports a short delay to *X*, the router just switches over to using line *A* to send traffic to *X*. In one vector exchange, the good news is processed.

To see how fast good news propagates, consider the five-node (linear) subnet of the following figure. (*Figure 4),* where the delay metric is the number of hops. In the *Figure 4 (a)* there are five routers *Ra, Rb, Rc, Rd* and *Re* linked to each other linearly. Suppose, a router Ra is down initially and all the other routers know this. In other words, they have all recorded the delay to *Ra* as infinity.



(a)



(b)

Figure 4: Count to infinity problem

We will describe this problem in the following stages: (i) when router Ra is up, and (ii) when router Ra is down. Now let us take the first stage. When *Ra* is up, the other routers in the subnet learn about it via the information (vector) exchanges. At the time of the first exchange, *Rb* learns that its left neighbour has zero delay to *Ra. Rb* now makes an entry in its routing table that *Ra* is just one hop away to the left. All the other routers still think that *Ra* is down. At this point, the routing table entries for *Ra* are as shown in the second row of *Figure 4(b)*. On the next exchange, *Rc* learns that *Rb* has a path of length 1 to *A*, so it updates its routing table to indicate a path of length 2, but *Rd* and *Re* do not hear the good news until later. Clearly, the good news is spreading at the rate of one hop per exchange. In a subnet whose longest path is of length *N* hops, within *N* exchanges everyone will know about the newly-revived lines and routers.

Now, let us consider the second stage *Figure 4(b)*, in which all the lines and routers are intially up. Routers *Rb,Rc,Rd* and *Re* are at a distance of 1,2,3 and 4 from A. Suddenly, *A* goes down, 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, *Rb* does not hear anything from *Ra*. Fortunately, *Rc* says: Do not worry; I have a path to *A* of length 2. Little does *B* know that *C's* path runs through *Rb* itself. For all *Rb* knows, *Rc* might have ten lines all with separate paths to *Ra* of length 2. As a result, *Rb* thinks it can reach *Ra* via *Rc*, with a path length of 3. *Rd* and *Re* do not update their entries on the first exchange.

On the second exchange, C notices that each of its neighbours are claiming a path to *Ra* of length 3. It picks one of them at random and makes its new distance to *Ra* 4, as shown in the third row of *Figure 4(b)*. Subsequent exchanges produce the history shown in the rest of *Figure 4(b)*.

Form *Figure 4*, it should be clear why bad news travels slowly: no router ever has a value higher than the minimum of all its neighbours. 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 time is 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. Not entirely surprisingly, this problem is known as the count-to-infinity problem. The core of the problem is that when *X* tells *Y* that it has a path somewhere, *Y* has no way of knowing whether it itself is on the path.

2.5 LINK STATE ROUTING

As explained above distance vector routing algorithm has a number of problems like count to infinity problem. For these reasons, it was replaced by a new algorithm, known as the link state routing.

Link state routing protocols are like a road map. A link state router cannot be fooled as easily into making bad routing decisions, because it has a complete picture of the network. The reason is that, unlike approximation approach of distance vector, link state routers have first hand information from all their peer routers. Each router originates information about itself, its directly connected links, and the state of those links. This information is passed around from router to router, each router making a copy of it, but never changing it. Link-state involves each router building up the complete topology of the entire network (or at least of the partition on which the router is situated), thus, each router contains the same information. With this method, routers only send information to of all the other routers when there is a change in the

topology of the network. The ultimate objective is that every router should have identical information about the network, and each router should be able to calculate its own best path independently. Independently calculate its own best paths.

In contrast to the distance-vector routing protocol, which works by sharing its knowledge of the entire network with its neighbours, link-state routing works by having the routers inform every router in the network about its nearest neighbours. The entire routing table is not distributed any router but, the part of the table containing its neighbours is:

Link-state is also known as **shortest path first**.

Link State Packet

When a router floods the network with information about its neighbourhood, it is said to be advertising. The basis of this advertising is a short packet called a link state packet (**LSP**). An LSP usually contains four fields: the ID of the advertiser, the ID of the destination network, the cost, and the ID of the neighbour router. The structure of a LSP is shown in *Table 2*.

Table 2: Link state packet (LSP)

Advertiser DD	Network DD	Cost	Neighbour DD
……	……	……	……
……	……	……	……

How Link State Routing Operates

The idea behind link state routing is simple and can be stated in five parts as suggested by Tanenbaum [Ref.1]. Each router must do the following:

1) Neighbour discovery

The Router has to discover its neighbours and learn their network addresses. As a router is booted, its first task is to learn who its neighbours are.

The Router does this by sending a special HELLO packet on each point-to-point line. The router on the other end is expected to send a reply disclosing its identity. These names must be globally unique. If two or more routers are connected by a LAN, the situation becomes slightly more complicated. One way of modeling the LAN is to consider it as a node itself. Please see reference [1] for further explanation through a diagram.

2) Measure delay

Another job that a router needs to perform is to measure the delay or cost to each of its neighbours. The most common way to determine this delay is to send over the line a special ECHO packet that the other side is required to send back immediately. By measuring the round-trip time and dividing it by two, the sending router can get a reasonable estimate of the delay. For even better results, the test can be conducted several times and the average used.

This method implicitly assumes that delays are symmetric, which may not always be the case.

3) Building link state packets

After collecting the information needed for the exchange, the next step for each router is to build a link state packet containing all the data. This packet starts with the

identity of the sender, followed by a sequence number and age, and a list of neighbours. For each neighbour, the delay to that neighbour is given.

As an example, let's consider the subnet given in *Figure 5* with delays shown as labels on the lines. For this network, the corresponding link state packets for all six routers are shown in the *Table 3*.

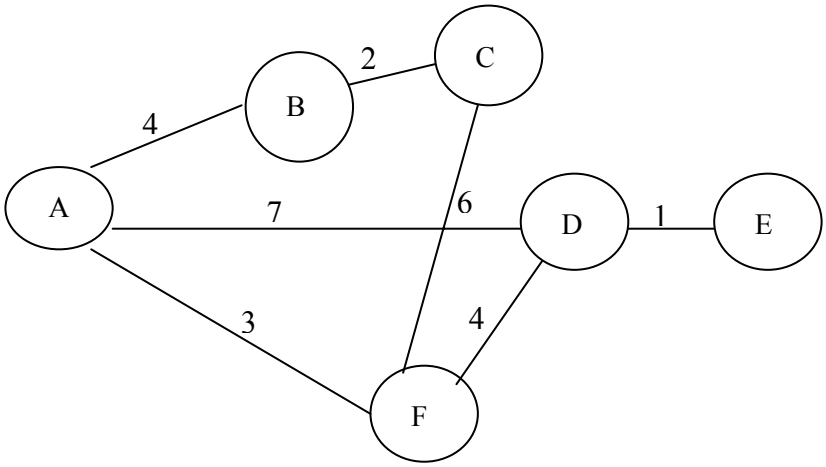


Figure 5: A subnet for link state routing

Table 3: The link state packets (LSPs) for the subnet in figure

A	
Seq.	
Age	
B	4
D	7
F	3

B	
Seq.	
Age	
C	2
A	4

C	
Seq.	
Age	
B	2
F	6

D	
Seq.	
Age	
A	7
E	1
F	4

E	
Seq.	
Age	
D	1

F	
Seq.	
Age	
A	3
C	4
D	6

Building the link state packets is easy. The hard part is determining when to build them. One possibility, is to build them periodically, that is, at regular intervals. Another possibility is to build them when some significant event occurs, such as a line or neighbour going down or coming back up again or changing its properties appreciably.

4) Distribute the packets

Let us describe the basic algorithm in distributing the link state packet. The fundamental concept here is flooding to distribute the packets. But to keep the number of packets flowing in the subnet under control, each packet contains a sequence number that is incremented for each new packet delivered. When a new link state packet arrives, it is checked against the list of packets already scene by a router. It is discarded in case the packet is old; otherwise it is forwarded on all lines except the one it arrived on. A router discards an obsolete packet (i.e., with a lower sequence) in case it has seen the packet with a highest sequence number.

The age of a data packet is used to prevent corruption of the sequence number from causing valid data to be ignored. The age field is decremented once per second by the routers which forward the packet. When it hits zero it is discarded.
How often should data be exchanged?

5) Compute shortest path tree

After accumulating all link state packets, a router can construct the entire subnet graph because every link is represented. In fact, every link is represented twice, once for each direction. The two values can be averaged or used separately.

Now, an algorithm like Dijkstra's algorithm can be run locally to construct the shortest path to all possible destinations. The results of this algorithm can be installed in the routing tables, and normal operation resumed.

Problems in Link State Routing

- In link state protocol, the memory required to store the data is proportional to $k * n$, for n routers each with k neighbors and the time required to compute can also be large.

- In it bad data e.g., data from routers in error will corrupt the computation.

☞ Check Your Progress 2

1) Which of the following statements are True or False.

   (i) Distance vector routing is a static routing algorithm. T ☐ F ☐

   (ii) Dijkstra's algorithms can be run locally in link state routing to construct the shortest path. T ☐ F ☐

2) Answer the following questions briefly.

   (a) What are the problems with distance vector routing algorithm? .

   ……

   ……

   ……

   (b) What is LSP?

   ……

   ……

   ……

2.6 HIERARCHICAL ROUTING

As you see, in both link state and distance vector algorithms, every router has to save some information about other routers. When the network size grows, the number of routers in the network increases. Consequently, the size of routing tables increases, as well, and routers cannot handle network traffic as efficiently. We use *hierarchical routing* to overcome this problem. Let's examine this subject with an example:

We use distance vector algorithms to find best routers between nodes. In the situation depicted below in *Figure 6,* every node of the network has to save a routing table with 17 records.

Here is a typical graph and routing table (*Table 4*) for A:

Table 4: A's Routing Table

Destination	Line	Weight
A	---	---
B	B	1
C	C	1
D	B	2
E	B	3
F	B	3
G	B	4
H	B	5
I	C	5
J	C	6
K	C	5
L	C	4
M	C	4
N	C	3
O	C	4
P	C	2
Q	C	3

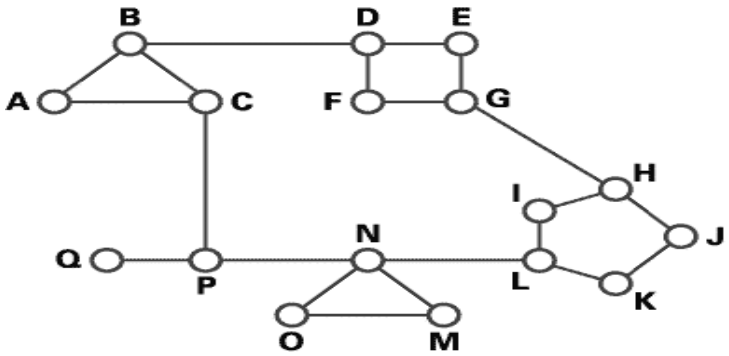


Figure 6: Network graph

In hierarchical routing, routers are classified in groups known as ***regions*** *(Figure 7)*. Each router has only the information about the routers in its own region and has no information about routers in other regions. So routers just save one record in their table for every other region. In this example, we have classified our network into five regions as shown below.

Table 5: A's Routing table for Hierarchical routing

Destination	Line	Weight
A	---	---
B	B	1
C	C	1
Region 2	B	2
Region 3	C	2
Region 4	C	3
Region 5	C	4

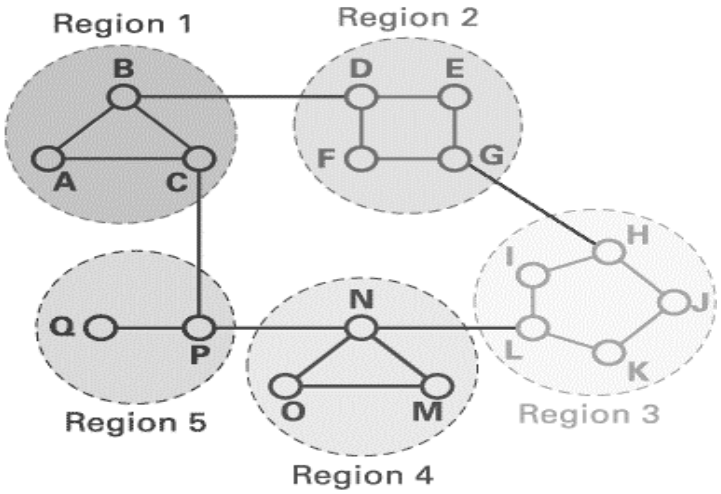


Figure 7: Hierarchical routing

If A wants to send packets to any router in region 2 (D, E, F or G), it sends them to B, and so on. As you can see, in this type of routing, the tables sizes are reduced so, network efficiency improves. The above example shows two-level hierarchical routing. We can also use three or four level hierarchical routing as well.

In three-level hierarchical routing, the network is classified into a number of *clusters*. Each cluster is made up of a number of regions, and each region contains a number of routers. Hierarchical routing is widely used in Internet routing and makes use of several routing protocols.

2.7 BROADCAST ROUTING

Up to now we were discussing about sending message from a source to a destination. Sometimes a host needs to send messages all other hosts. This type of transmission i.e., to send a message to all destinations simultaneously is known as *broadcasting*. There are a no. of methods for broadcasting. These are:

- **Send a distinct packet to each destination**

This is a very simple method, in which a source sends a distinct packet to each destination. Major disadvantages of this method are:

a) It wastes bandwidth.
b) In this method source needs to have a complete list of all destinations.

Because of this reason this method is the least desirable of the other methods.

- **Flooding**

This is also a very simple method of broadcasting. In this method every incoming packet is sent out on every outgoing line except the line from which it arrived. This algorithm is very simple to implement. But the major disadvantage of this algorithm is that it generates lots of redundant packets, thus consumes too much bandwidth.

- **Multidestination routing**

In this method each packet contains either a list of destinations or a bit map indicating the desired destinations. When a packet arrives at a router, router determines the set of output lines that would be required by checking all the destinations. Router only chooses those output lines which are the best route to at least one of the destinations. After this router creates a new copy of the packet for each output line to be used and

in each packet it includes only those destinations that are to use the line. Therefore, the destination set is partitioned among the output lines. In this, after a sufficient number of hops, each packet will carry only one destination and can be treated as a normal packet.

- **Using a spanning tree**

A spanning tree is a subset of graph that includes all the nodes (of graph) but contains no loops. This method uses the spanning tree, therefore, each router knows which of its lines belong to the spanning tree. When a packet arrives at a router, it copies onto all the spanning tree lines except the one it arrived on.

Advantage of this method is that it makes excellent use of bandwidth and generates only the minimum number of packets required to do the job.

In this method each router must have knowledge of some spanning tree. Sometimes this information is available (e.g., with link state routing) but sometimes it is not (e.g., with distance vector routing), this is the major disadvantage of this method.

- **Reverse path forwarding**

Our last broadcast algorithm is an attempt to approximate the behaviour of the previous one, even when the routers do not know anything at all about spanning trees. The idea, called reverse path forwarding, is remarkably simple once it has been pointed out.

In this method, when a broadcast packet arrives at a router, the router checks whether the packet arrived on the line that is normally used for sending packets to the source of the broadcast or not.

If the packet arrived on the line that is normally used for sending packets to the source of the broadcast then

Router forwards copies of it onto all lines except the one it arrived on.

Else (i.e., packet arrived on a line other than the preferred one for reaching the source)

Router discards the packet as a likely duplicate.

2.8 MULTICAST ROUTING

In many cases, you need to send same data to multiple clients at the same time. In this case, if, we use unicasting then the server will connect to each of its clients again and again, but each time it will send an identical data stream to each client. This is a waste of both server and network capacity. If, we use broadcasting in this case, it would be inefficient because sometimes receivers are not interested in the message but they receive it nonetheless, or sometimes they are interested but are not supposed to see the message.

In such cases i.e., for sending a message to a group of users (clients), we use another technique known as multicasting. The routing algorithm used for multicasting, is called multicast routing.

Group management is the heart of multicasting. For group management, we require some methods to create and destroy a group and to allow processes to join and leave a group. When a router joins a group, it informs its host of this fact. For routing, routers

mainly want to know which of their hosts belong to which group. For this, either the host must inform their router about changes in the group membership, or routers must query their hosts periodically. On receiving this information, routers tell their neighbours, so the informations propagated through the subnet.

Now, we will learn the working of multicasting through an example. In our example (as shown in *Figure 8*), we have taken a network containing two groups i.e., group 1 and 2. Here, some routers are attached to hosts that belong to only one of these groups and some routers are attached to hosts that belong to both of these groups.

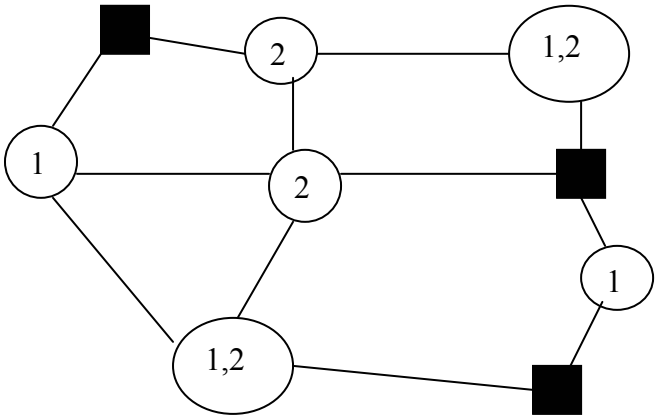


Figure 8: Network containing two groups i.e., 1 and 2

To do multicast routing, first, each router computes a *spanning tree* covering all other routers. For example, *Figure 9* shows spanning tree for the leftmost router.

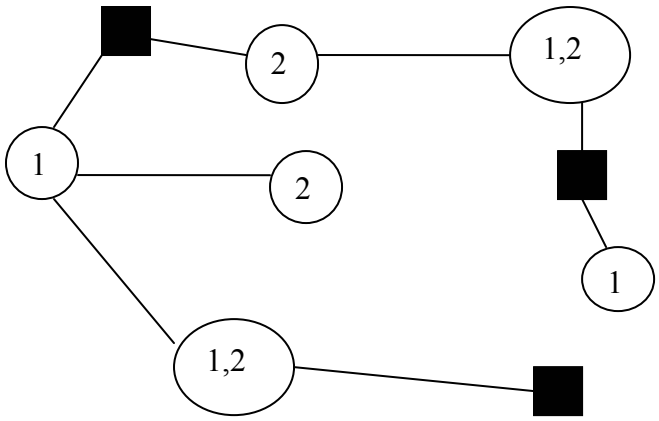


Figure 9: Spanning tree for the leftmost router

Now, when a process sends a multicast packet to a group, the first router examines its spanning tree and prunes it. *Pruning* is the task of removing all lines that do not lead to hosts that are members of the group. For example, *Fig. 10* shows the pruned spanning tree for group 1 and *Fig. 11* shows the pruned spanning tree for group 2. There are a number of ways of pruning the spanning tree. One of the simplest ones that can be used, if link state routing is used and each router is aware of the complete topology, including the hosts that belong to those groups. Then, the spanning tree can be pruned, starting at the end of each path, working toward the root, and removing all routers that do not belong to the group under consideration. With distance vector routing, a different pruning strategy can be followed. The basic algorithm is reverse path forwarding. However, whenever a router with no hosts interested in a particular group and no connections to other routers, receives a multicast message for that group,

it responds with a PRUNE message, thus, telling the sender not to send it any more multicasts for that group. When a router with no group members among its own hosts has received such a message on its lines, it, too, can respond with a PRUNE message. In this way, the subnet is recursively pruned.

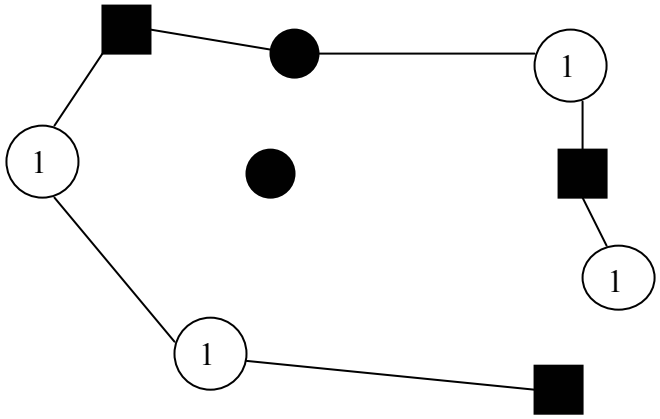


Figure 10: A pruned spanning multicast tree for group 1

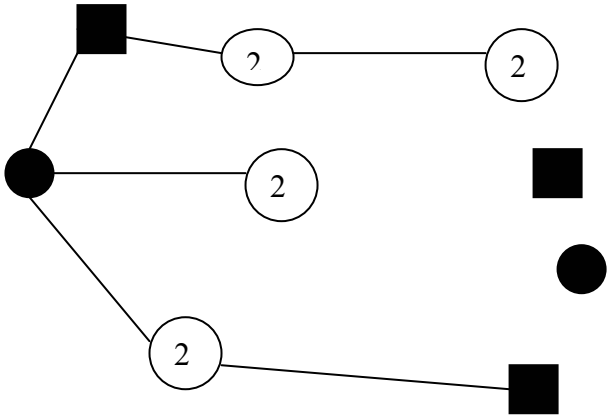


Figure 11: A pruned spanning multicast tree for group 2

After pruning, multicast packets are forwarded only along the appropriate spanning tree. This algorithm needs to store separate pruned spanning tree for each member of each group. Therefore, this would not be good for large networks.

☞ Check Your Progress 3

1) Which of the following statements are True or False.

   (a) In hierarchical routing, each router has no information about routers in other regions. T ☐ F ☐

   (b) A spanning tree is a subset of a graph that includes some of the nodes of that graph. T ☐ F ☐

   (c) Sending a message to a group of users in a network is known as broadcasting. T ☐ F ☐

2) Answer the following questions in brief.

a) Explain reverse path forwarding in brief.

…….

……

……

b) What is Pruning?

……

……

……

2.9 SUMMARY

In this unit, we first studied different routing algorithms. First, we looked at finding the route between a given pair of routers. The algorithm finds the shortest path between them on the graph. A number of algorithms for computing the shortest path between two nodes of a graph are known. Here, we have studied the Dijkstra algorithm.

Next, we studied flooding. In flooding, every incoming packet is sent out on every outgoing line except the line from which it arrived. This algorithm is very simple to implement, but it generates lots of redundant packets. It discovers all routes, including the optimal one, therefore this is robust and gives high performance.

Next, we studied the Belman-Ford routing algorithm. In this algorithm each host maintains a routing table. This routing table has an entry for every other router in the subnet. These tables are updated by exchanging information with the neighbours.

Next, we studied the link state routing algorithm. In this algorithm, each router originates information about itself, its directly connected links, and the state of those links. This information is passed around from router to router, each router making a copy of it, but never changing it. The ultimate objective is that every router has identical information about the network, and each router will independently calculate its own best paths.

Next, we discussed hierarchical routing algorithm. In hierarchical routing, routers are classified in groups known as regions. Each router has only the information about the routers in its own region and has no information about routers in other regions.

Next, we studied broadcasting i.e., to send a message to all destinations in a network simultaneously. There are a number of methods for broadcasting such as, flooding, muitidestination routing, reverse path forwarding etc. In this unit we discussed all these methods in brief.

Finally, we discussed multicasting i.e., to send a message to a group of users in a network. In multicasting, each router first computes a spanning tree covering all other router. Now, when a process sends a multicast packet to a group, the first router examines its spanning tree and prunes it. Then, packets are forwarded only along the appropriate spanning tree.

2.10 SOLUTIONS/ANSWERS

Check Your Progress 1

1) (i) True
(ii) True
(iii) False

2) (a) A variation of flooding that is selective flooding. In this algorithm the routers do not send every incoming packet *out* on every line, only *on* those lines that are moving approximately in the right direction.

Check Your Progress 2

1) (i) False
(ii) True

2) (a) Problems with distance vector routing algorithm are: it uses only approximation, neighbours, slowly increase their path length to a dead node, and the condition of being dead (infinite distance) is reached by counting to infinity, one at a time.

(b) In link state routing, a router floods the network with information about 3 of its neighbours, by using a short packet, known as link state packet (LSP). An LSP usually contains four fields: the ID of the advertiser, the ID of the destination network, the cost, and the ID of the neighbour router.

Advertiser	Network	Cost	Neighbour
…… ……	…… ……	…… ……	…… ……

Link State Packet (LSP)

Check Your Progress 3

1) (a) True
(b) False
(c) False

2) (a) Reverse path forwarding is a method of broadcasting. In this method, when a broadcast packet arrives at a router, the router checks whether the packet has arrived on the line that is normally used for sending packets to the source of the broadcast or not. *If* the packet has arrived on the line that is normally used for sending packets to the source of the broadcast then, the Router forwards copies of it onto all lines except the one it has arrived on. *Else (i.e.,* packet arrived on a line other than the preferred one for reaching the source). The router discards the packet as a likely duplicate.

(b) In multicasting, pruning is the task of removing all lines from the spanning tree of a router that do not lead to hosts that are members of a particular group.

2.11 FURTHER READINGS

1) *Computer Network*, S. Tanenbaum, 4th edition, Prentice Hall of India, New Delhi 2002.

2) *Data Network,* Drnitri Berteskas and Robert Galleger, Second edition, Prentice Hall of India, 1997, New Delhi.

3) *Data and Computer Communication,* William Stalling, Pearson Education, 2nd Edition, Delhi.