Module 3 Notes

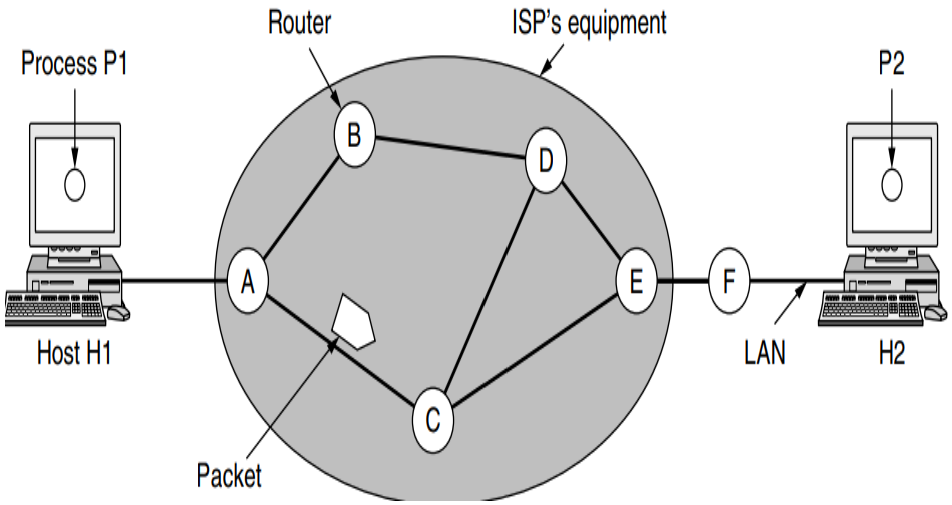
NETWOK LAYER

Network Layer Design Issues

- **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. However routers on customer premises are considered part of the ISP network because they run the same algorithms as the ISP's routers (and our main concern here is algorithms). D C B A E F Packet Process P1 Host H1 Router ISP's equipment LAN H2 P2.

This equipment is used as follows. 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.



- **Services Provide to the transport layer:**

The network layer provides services to the transport layer at the network layer/transport layer interface. An important question is precisely what kind of services the network layer provides to the transport layer.

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.

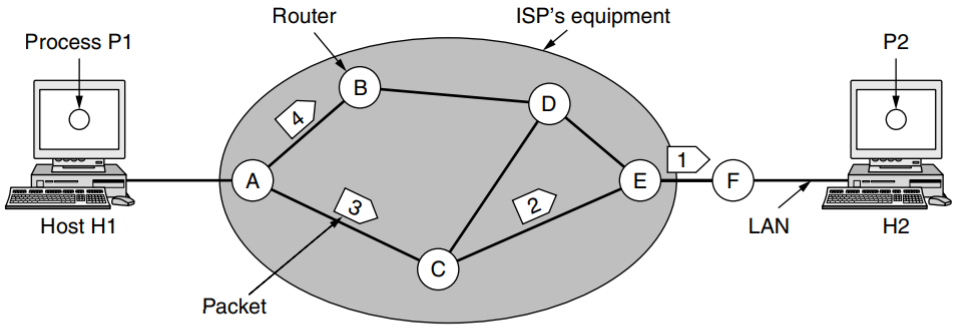
Given these goals, the designers of the network layer have a lot of freedom in writing detailed specifications of the services to be offered to the transport layer. This freedom often degenerates into a raging battle between two warring factions. The discussion centers on whether the network layer should provide connection oriented service or connectionless service. One camp (represented by the Internet community) argues that the routers' job is moving packets around and nothing else. In this view the network is inherently unreliable, no matter how it is designed.

Therefore, the hosts should accept this fact and do error control (i.e., error detection and correction) and flow control themselves. This viewpoint leads to the conclusion that the network service should be connectionless, with primitives SEND PACKET and RECEIVE PACKET and little else. In particular, no packet ordering and flow control should be done, because the hosts are going to do that anyway and there is usually little to be gained by doing it twice. This reasoning is an example of the end-to-end argument, a design principle that has been very influential in shaping the Internet .

Furthermore each packet must carry the full destination address, because each packet sent is carried independently of its predecessors, if any. The other camp (represented by the telephone companies) argues that the network should provide a reliable, connection-oriented service. However, since the days of the ARPANET and the early Internet, connectionless network layers have grown tremendously in popularity. The IP protocol is now an ever-present symbol of success. It was undeterred by a connection-oriented technology called ATM that was developed to overthrow it in the 1980s; instead, it is ATM that is now found in niche uses and IP that is taking over telephone networks. Under the covers, however, the Internet is evolving connection-oriented features as quality of service becomes more important

- **Implementation of Connectionless Service:**

Two different organizations are possible, depending on the type of service offered. 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. If connection-oriented service is used, a path from the source router all the way to the destination router must be established before any data packets can be sent. This connection is called a VC (virtual circuit), in analogy with the physical circuits set up by the telephone system, and the network is called a virtual-circuit network. In this section, we will examine datagram networks; in the next one, we will examine virtual-circuit networks. Let us now see how a datagram network works. Suppose that the process P1 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.



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

Let us assume for this example that the message is four times longer than the maximum packet size, so the network layer has to break it into four packets, 1, 2, 3, and 4, and send each of them in turn to router A using some point-to-point protocol. At this point the ISP takes over. Every router has an internal table telling it where to send packets for each of the possible destinations. Each table entry is a pair consisting of a destination and the outgoing line to use for that destination. Only directly connected lines can be used.

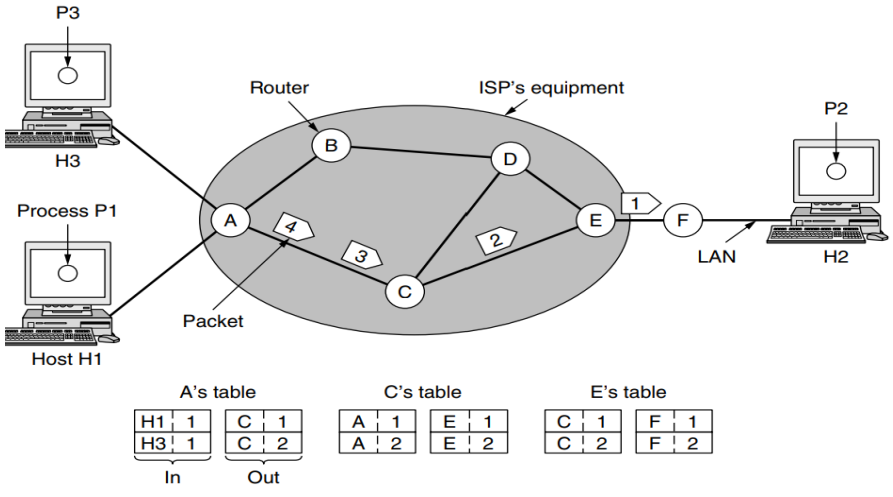
A has only two outgoing lines—to B and to C—so every incoming packet must be sent to one of these routers, even if the ultimate destination is to some other router. A's initial routing table is shown in the figure under the label "initially." At A, packets 1, 2, and 3 are stored briefly, having arrived on the incoming link and had their checksums verified. Then each packet is forwarded

according to A's table, onto the outgoing link to C within a new frame. Packet 1 is then forwarded to E and then to F. When it gets to F, it is sent within a frame over the LAN to H2. Packets 2 and 3 follow the same route. However, something different happens to packet 4.

When it gets to A it is sent to router B, even though it is also destined for F. For some reason, A decided to send packet 4 via a different route than that of the first three packets. Perhaps it has learned of a traffic jam somewhere along the ACE path and updated its routing table, as shown under the label ''later.'' The algorithm that manages the tables and makes the routing decisions is called the routing algorithm. IP (Internet Protocol) which is the basis for the entire Internet, is the dominant example of a connectionless network service. Each packet carries a destination IP address that routers use to individually forward each packet. The addresses are 32 bits in IPv4 packets and 128 bits in IPv6 packets.

- **Implementation Of Connection-Oriented Service:**

For connection-oriented service, we need a virtual-circuit network.The idea behind virtual circuits is to avoid having to choose a new route for every packet sent.when a connection is established, a route from the source machine to the destination machine is chosen as part of the connection setup and stored in tables inside the routers. That route is used for all traffic flowing over the connection, exactly the same way that the telephone system works. When the connection is released, the virtual circuit is also terminated. With connection-oriented service, each packet carries an identifier telling which virtual circuit it belongs to. As an example, consider the situation where host H1 has established connection 1 with host H2. This connection is remembered as the first entry in each of the routing tables. The first line of A's table says that if a packet bearing connection identifier 1 comes in from H1, it is to be sent to router C and given connection identifier 1. Similarly, the first entry at C routes the packet to E, also with connection



identifier 1.

Now let us consider what happens if H3 also wants to establish a connection to H2. It chooses connection identifier 1 (because it is initiating the connection and this is its only connection) and tells the network to establish the virtual circuit. This leads to the second row in the tables. Note that we have a conflict here because although A can easily distinguish connection 1 packets from H1 from connection 1 packets from H3, C cannot do this. For this reason, A assigns a different connection identifier to the outgoing traffic for the second connection. Avoiding conflicts of this kind is why routers need the ability to replace connection identifiers in outgoing packets. In some contexts, this process is called label switching. An example of a connection-oriented network service is MPLS (MultiProtocol Label Switching). It is used within ISP networks in the Internet, with IP packets wrapped in an MPLS header having a 20-bit connection identifier or label. MPLS is often hidden from customers, with the ISP establishing long-term connections for large amounts of traffic, but it is increasingly being used to help when quality of service is important but also with other ISP traffic management tasks.

- **Comparison of Virtual Circuit and datagram Networks**:

Both virtual circuits and datagrams have their supporters and their detractors. We will now attempt to summarize both sets of arguments. The major issues are listed in although purists could probably find a counterexample for everything in the figure.

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

Inside the network, several trade-offs exist between virtual circuits and data grams. One trade-off is setup time versus address parsing time. Using virtual circuits requires a setup phase, which takes time and consumes resources. However, once this price is paid, figuring out what to do with a data packet in a virtual-circuit network is easy: the router just uses the circuit number to index into a table to find out where the packet goes. In a datagram network, no setup is needed but a

more complicated lookup procedure is required to locate the entry for the destination.

A related issue is that the destination addresses used in datagram networks are longer than circuit numbers used in virtual-circuit networks because they have a global meaning. If the packets tend to be fairly short, including a full destination address in every packet may represent a significant amount of overhead, and hence a waste of bandwidth.

Yet another issue is the amount of table space required in router memory. A datagram network needs to have an entry for every possible destination, whereas a virtual-circuit network just needs an entry for each virtual circuit. However, this advantage is somewhat illusory since connection setup packets have to be routed too, and they use destination addresses, the same as datagrams do.

Virtual circuits have some advantages in guaranteeing quality of service and avoiding congestion within the network because resources (e.g., buffers, bandwidth, and CPU cycles) can be reserved in advance, when the connection is established. Once the packets start arriving, the necessary bandwidth and router capacity will be there. With a datagram network, congestion avoidance is more difficult.

For transaction processing systems (e.g., stores calling up to verify credit card purchases), the overhead required to set up and clear a virtual circuit may easily dwarf the use of the circuit. If the majority of the traffic is expected to be of this kind, the use of virtual circuits inside the network makes little sense. On the other hand, for long-running uses such as VPN traffic between two corporate offices, permanent virtual circuits (that are set up manually and last for months or years) may be useful.

Virtual circuits also have a vulnerability problem. If a router crashes and loses its memory, even if it comes back up a second later, all the virtual circuits passing through it will have to be aborted. In contrast, if a datagram router goes down, only those users whose packets were queued in the router at the time need suffer (and probably not even then since the sender is likely to

retransmit them shortly). The loss of a communication line is fatal to virtual circuits using it, but can easily be compensated for if datagrams are used. Datagrams also allow the routers to balance the traffic throughout the network, since routes can be changed partway through a long sequence of packet transmissions.

Rounting Alorgthims

Routing is the process of establishing the routes that data packets must follow to reach the destination. In this process, a routing table is created which contains information regarding routes that data packets follow. Various routing algorithms are used for the purpose of deciding which route an incoming data packet needs to be transmitted on to reach the destination efficiently.

Classification of Routing Algorithms

The routing algorithms can be classified as follows:

1. **Adaptive Algorithms**
2. **Non-Adaptive Algorithms**

Adaptive Algorithms

These are the algorithms that change their routing decisions whenever network topology or traffic load changes. The changes in routing decisions are reflected in the topology as well as the traffic of the network. Also known as dynamic routing, these make use of dynamic information such as current topology, load, delay, etc. to select routes. Optimization parameters are distance, number of hops, and estimated transit time.

Non-Adaptive Algorithms

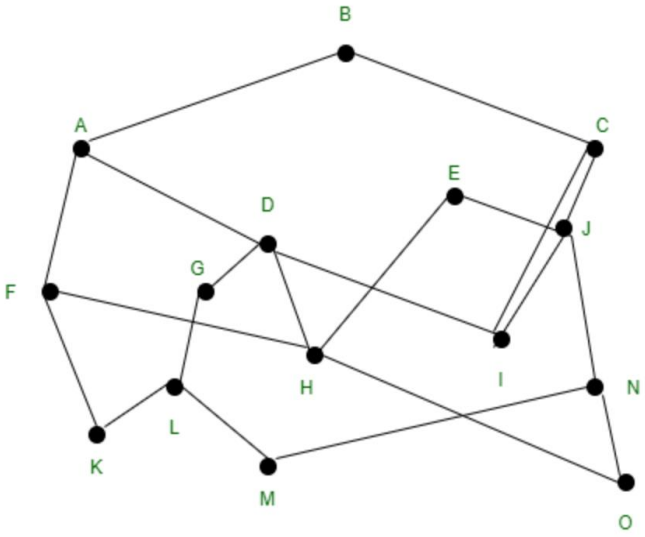
These are the algorithms that do not change their routing decisions once they have been selected. This is also known as static routing as a route to be taken is computed in advance and downloaded to routers when a router is booted.

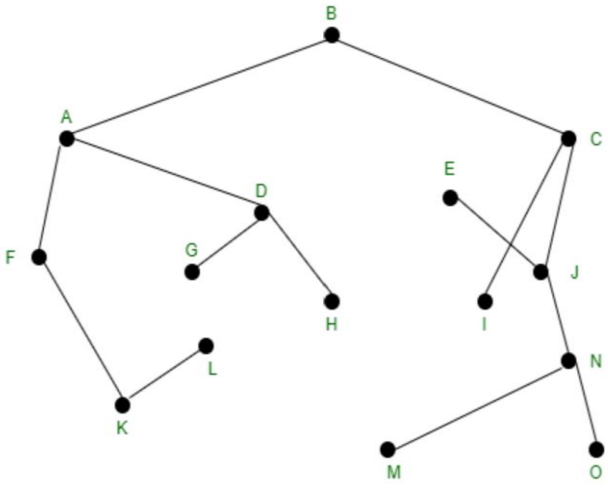
1. **OPTIMALITY PRINCIPLE**

Sink Tree for routers :

We can see that the set of optimal routes from all sources to a given destination from a tree rooted at the destination as a directed consequence of the optimality principle. This tree is called a **sink tree**.

In the given figure the distance metric is the number of hops. Therefore, the goal of all routing algorithms is to discover and use the sink trees for all routers.



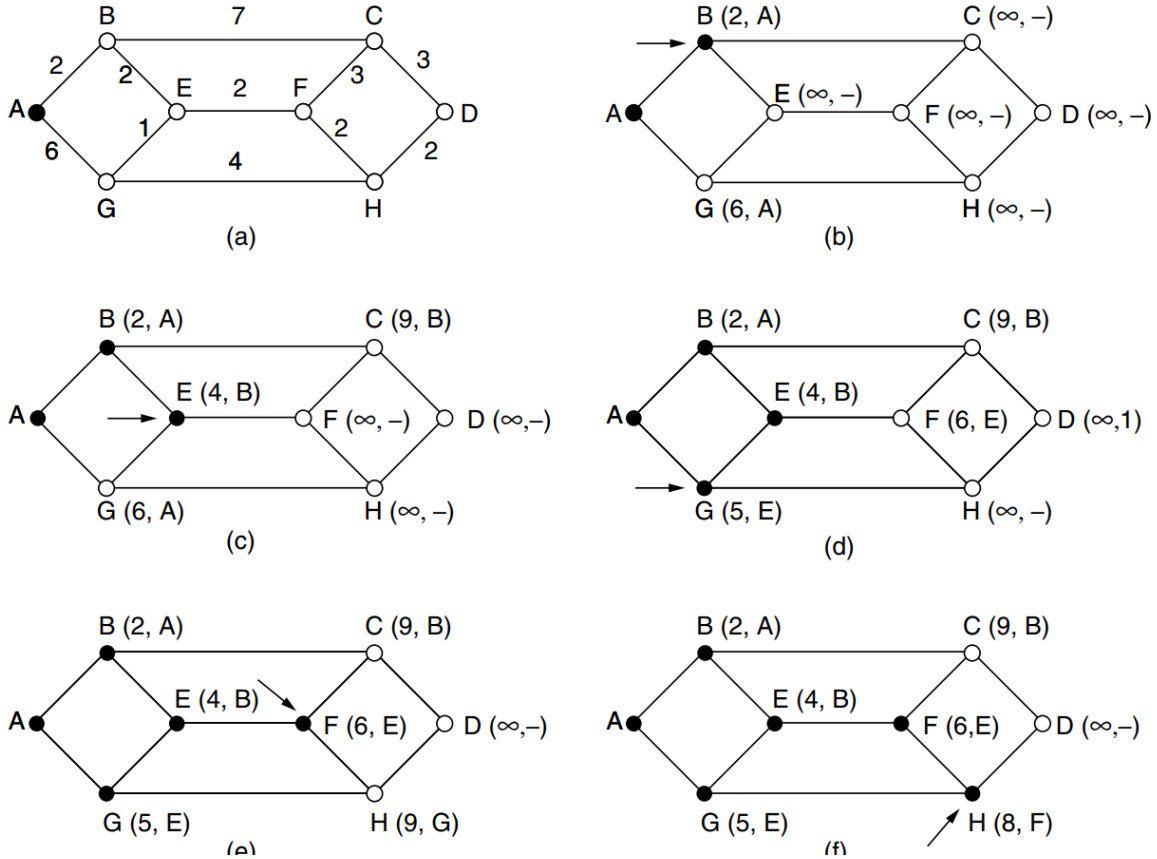


The sink tree is not unique also other trees with the same path lengths may exist. If we allow all of the possible paths to be chosen, the tree becomes a more general structure called a **DAG (Directed Acyclic Graph).** DAGs have no loops. We will use sink trees as a convenient shorthand for both cases. we will take technical assumption for both cases that the paths do not interfere with each other so, for example, a traffic jam on one path will not cause another path to divert.

2. SHORTEST PATH ALGORITHM:

The idea is to build a graph of the network, with each node of the graph representing a router and each edge of the graph representing a communication line, or link. To choose a route between a given pair of routers, the algorithm just finds the shortest path between them on the graph.

The concept of a shortest path deserves some explanation. One way of measuring path length is the number of hops. Using this metric, the paths ABC and ABE are equally long. Another metric is the geographic distance in kilometers, in which case ABC is clearly much longer than ABE (assuming the figure is drawn to scale).



The algorithm maintains a set of visited vertices and a set of unvisited vertices. It starts at the source vertex and iteratively selects the unvisited vertex with the smallest tentative distance from the source. It then visits the neighbors of this vertex and updates their tentative distances if a shorter path is found. This process continues until the destination vertex is reached, or all reachable vertices have been visited.

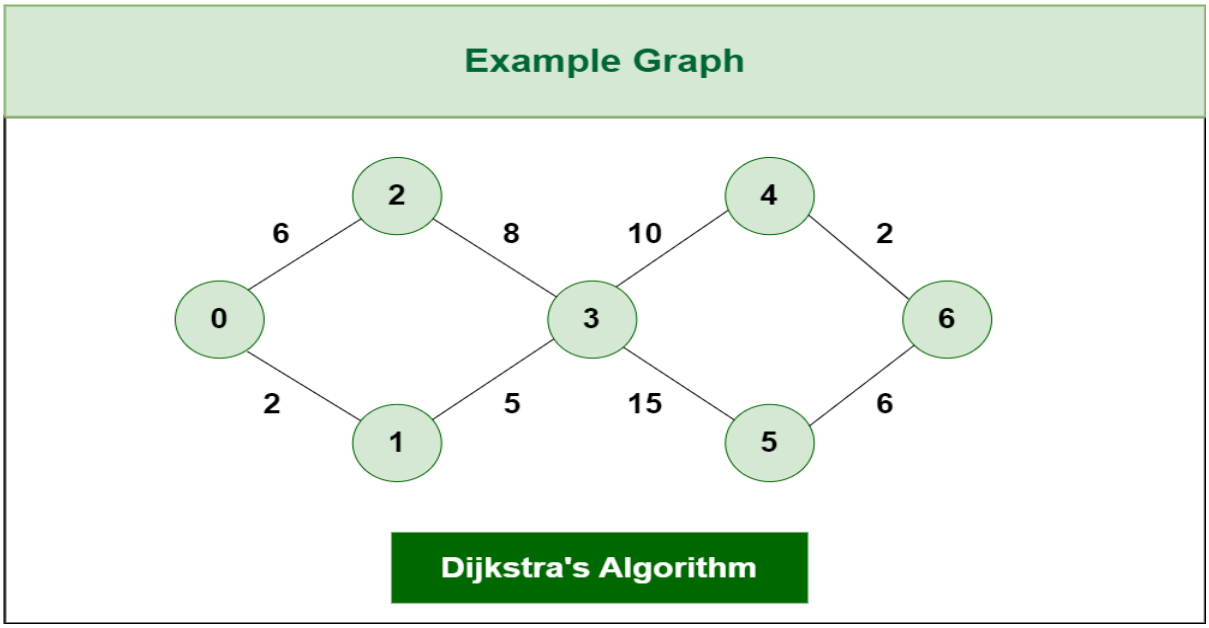
```
#define MAX_NODES 1024                  /* maximum number of nodes */
#define INFINITY 1000000000             /* a number larger than every maximum path */
int n, dist[MAX_NODES][MAX_NODES];      /* dist[i][j] is the distance from i to j */

void shortest_path(int s, int t, int path[])
{ struct state {                        /* the path being worked on */
      int predecessor;                  /* previous node */
      int length;                       /* length from source to this node */
      enum {permanent, tentative} label;/* label state */
  } state[MAX_NODES];

  int i, k, min;
  struct state *p;

  for (p = &state[0]; p < &state[n]; p++) {      /* initialize state */
      p->predecessor = -1;
      p->length = INFINITY;
      p->label = tentative;
  }
  state[t].length = 0;  state[t].label = permanent;
  k = t;                                        /* k is the initial working node */
  do {                                          /* Is there a better path from k? */
      for (i = 0; i < n; i++)                   /* this graph has n nodes */
          if (dist[k][i] != 0 && state[i].label == tentative) {
              if (state[k].length + dist[k][i] < state[i].length) {
                  state[i].predecessor = k;
                  state[i].length = state[k].length + dist[k][i];
              }
          }

      /* Find the tentatively labeled node with the smallest label. */
      k = 0; min = INFINITY;
      for (i = 0; i < n; i++)
          if (state[i].label == tentative && state[i].length < min) {
              min = state[i].length;
              k = i;
          }
      state[k].label = permanent;
  } while (k != s);

  /* Copy the path into the output array. */
  i = 0;  k = s;
  do {path[i++] = k; k = state[k].predecessor; } while (k >= 0);
}
```

How does Dijkstra's Algorithm works?

Let's see how Dijkstra's Algorithm works with an example given below:

Dijkstra's Algorithm will generate the shortest path from Node 0 to all other Nodes in the graph.

Consider the below graph:



The algorithm will generate the shortest path from node 0 to all the other nodes in the graph.

For this graph, we will assume that the weight of the edges represents the distance between two nodes.

As, we can see we have the shortest path from,

Node 0 to Node 1, from

Node 0 to Node 2, from

Node 0 to Node 3, from

Node 0 to Node 4, from

Node 0 to Node 6.

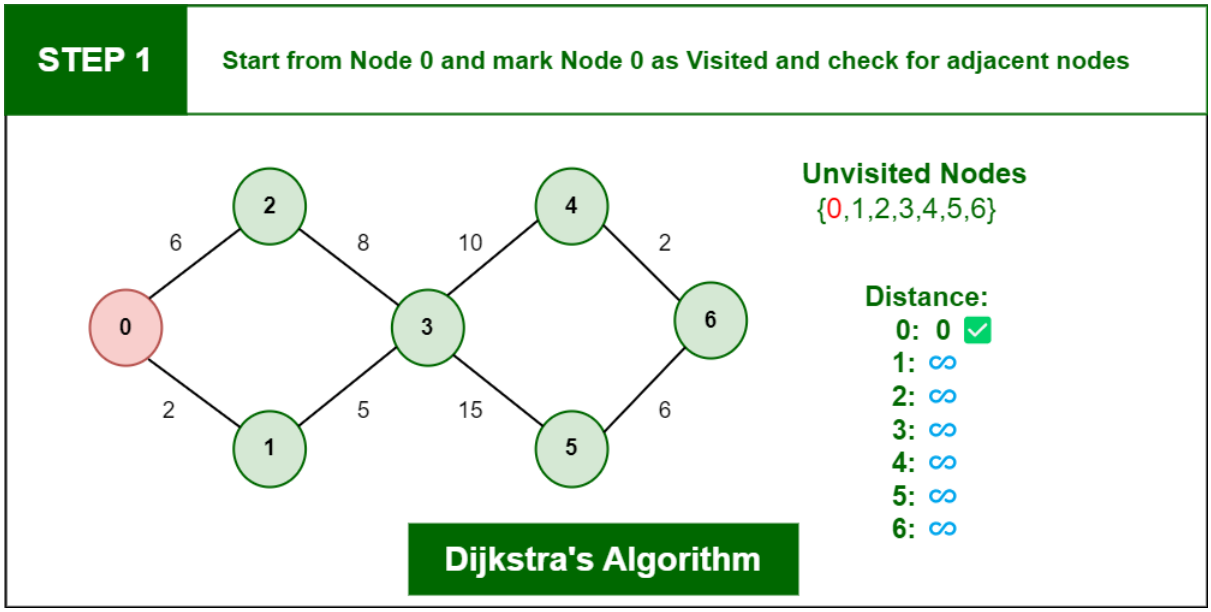
Initially we have a set of resources given below :

- *The Distance from the source node to itself is 0. In this example the source node is 0.*
- *The distance from the source node to all other node is unknown so we mark all of them as infinity.*

Example: 0 -> 0, 1-> ∞,2-> ∞,3-> ∞,4-> ∞,5-> ∞,6-> ∞.

- *we'll also have an array of unvisited elements that will keep track of unvisited or unmarked Nodes.*
- *Algorithm will complete when all the nodes marked as visited and the distance between them added to the path.* ***Unvisited Nodes:- 0 1 2 3 4 5 6.***

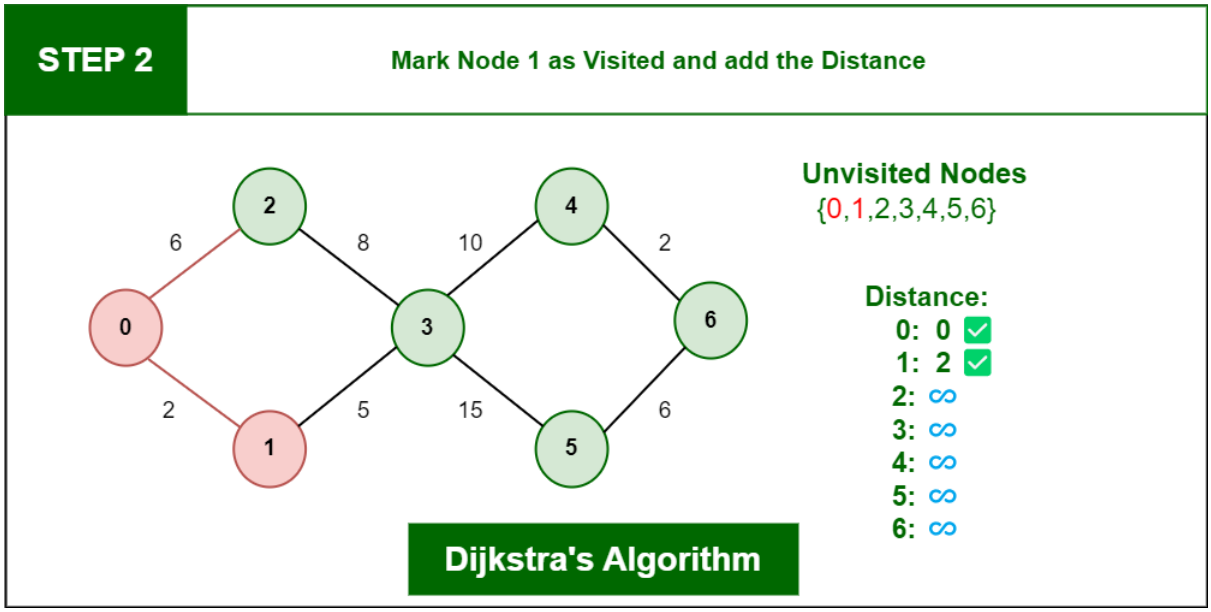
Step 1: *Start from Node 0 and mark Node as visited as you can check in below image visited Node is marked red.*



Dijkstra's Algorithm

Step 2: *Check for adjacent Nodes, Now we have to choices (Either choose Node1 with distance 2 or either choose Node 2 with distance 6) and choose Node with minimum distance. In this step* ***Node 1*** *is Minimum distance adjacent Node, so marked it as visited and add up the distance.*

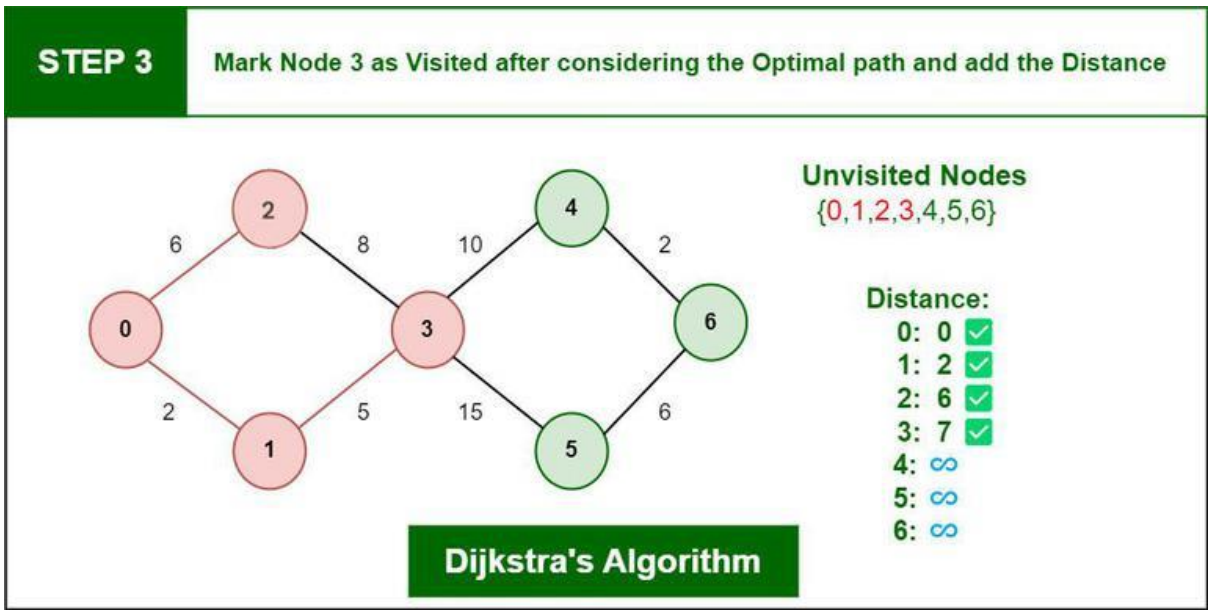
Distance: Node 0 -> Node 1 = 2



Dijkstra's Algorithm

Step 3: *Then Move Forward and check for adjacent Node which is Node 3, so marked it as visited and add up the distance, Now the distance will be:*

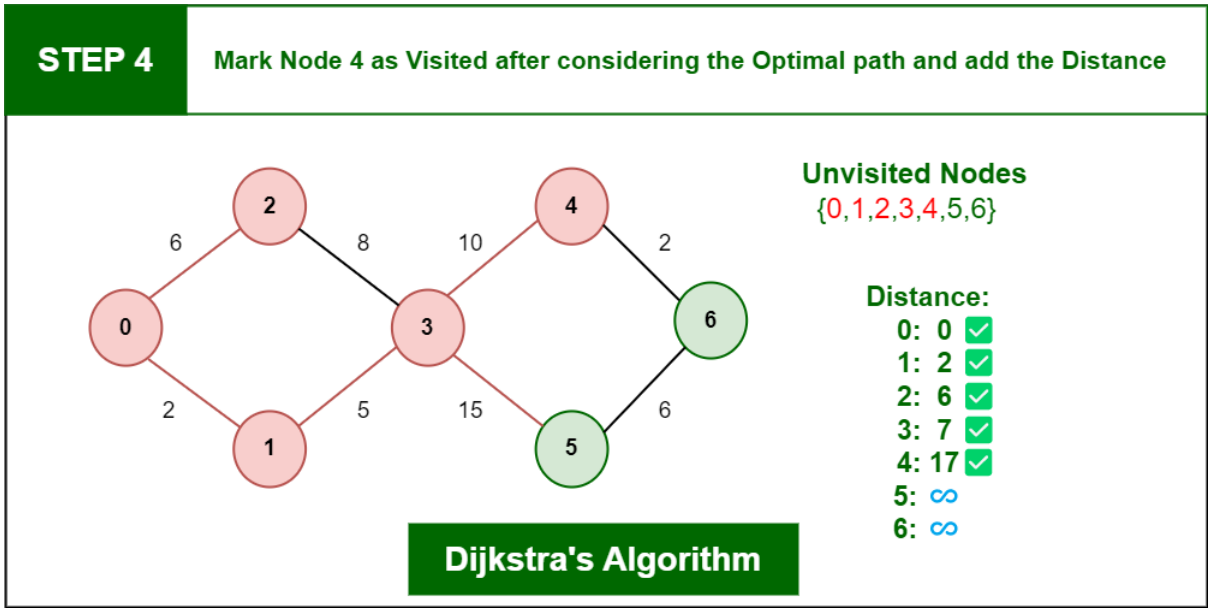
Distance: Node 0 -> Node 1 -> Node 3 = 2 + 5 = 7



Dijkstra's Algorithm

Step 4: *Again we have two choices for adjacent Nodes (Either we can choose Node 4 with distance 10 or either we can choose Node 5 with distance 15) so choose Node with minimum distance. In this step* ***Node 4*** *is Minimum distance adjacent Node, so marked it as visited and add up the distance.*

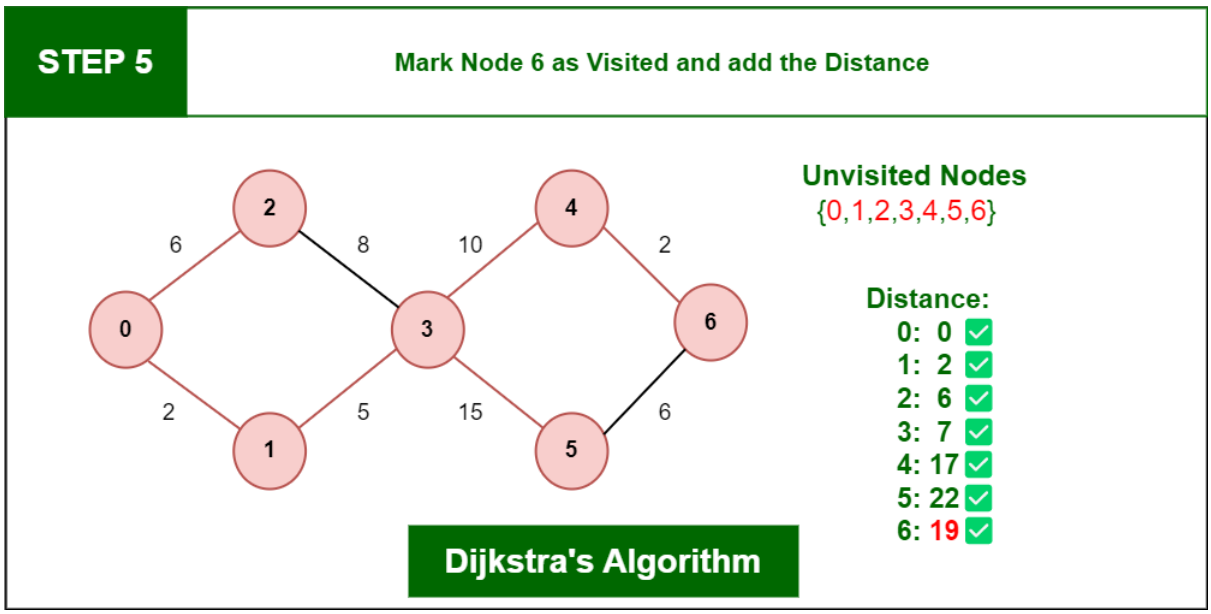
Distance: Node 0 -> Node 1 -> Node 3 -> Node 4 = 2 + 5 + 10 = 17



Dijkstra's Algorithm

Step 5: *Again, Move Forward and check for adjacent Node which is* ***Node 6****, so marked it as visited and add up the distance, Now the distance will be:*

Distance: Node 0 -> Node 1 -> Node 3 -> Node 4 -> Node 6 = 2 + 5 + 10 + 2 = 19



Dijkstra's Algorithm

So, the Shortest Distance from the Source Vertex is 19 which is optimal one

3. FLOODING

- Requires no network information like topology, load condition, cost of diff. paths
- Every incoming packet to a node is sent out on every outgoing like except the one it arrived on.
- For Example in the above figure
    - An incoming packet to (1) is sent out to (2),(3)
    - from (2) is sent to (6),(4), and from (3) it is sent to (4),(5)
    - from (4) it is sent to (6),(5),(3), from (6) it is sent to (2),(4),(5), from (5) it is sent to (4),(3)

Characteristics –

- All possible routes between Source and Destination are tried. A packet will always get through if the path exists
- As all routes are tried, there will be at least one route which is the shortest
- All nodes directly or indirectly connected are visited

Limitations –

- looding generates a vast number of duplicate packets
- Suitable damping mechanism must be used

4. DISTANCE VECTOR ROUTING ALGORITHM

• Distance vector (DV) algorithm is 1) iterative, 2) asynchronous, and 3) distributed.

1) It is distributed. This is because each node

→ receives some information from one or more of its directly attached neighbors

→ performs the calculation and

→ distributes then the results of the calculation back to the neighbors.

2) It is iterative. This is because

→ the process continues on until no more info is exchanged b/w neighbors.

3) It is asynchronous. This is because

→ the process does not require all of the nodes to operate in lockstep with each other.

The basic idea is as follows:

1) Let us define the following notation:

→**Dx(y) =** cost of the least-cost path from node x to node y, for all nodes in N.

→**Dx = [Dx(y): y in N] be** node x's distance vector of cost estimates from x to all other nodes y in N.

2) Each node **x** maintains the following routing information:

i) For each neighbor v, the cost c(x,v) from node x to directly attached neighbor v

ii) 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.

iii) The distance vectors of each of its neighbors, that is, Dv = [Dv(y): y in N] for each neighbor v of x.

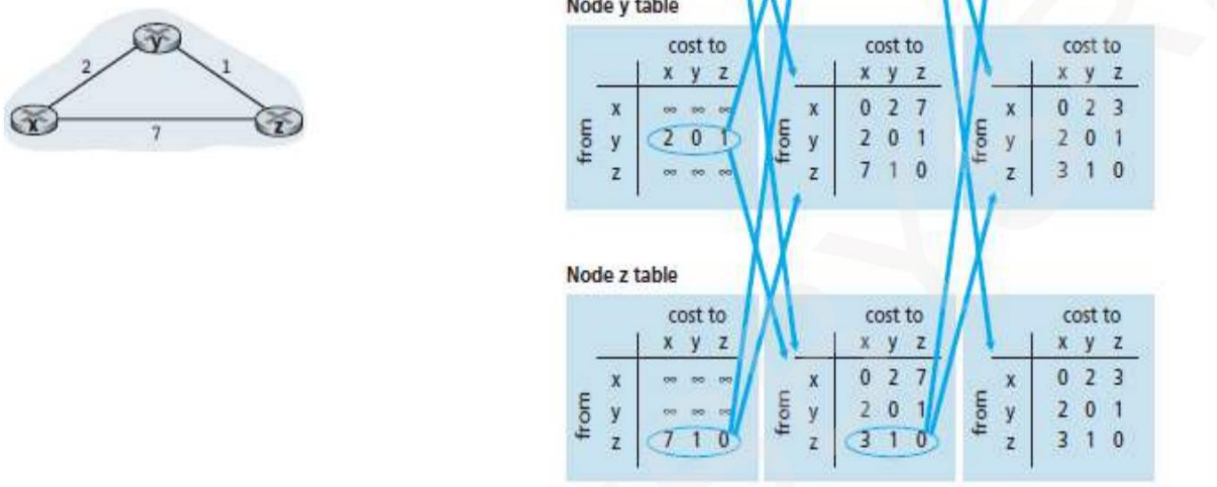
3) From time to time, each node sends a copy of its distance vector to each of its neighbors.

4) The least costs are computed by the Bellman-Ford equation:

Dx(y) = minv{c(x,v) + Dv(y)} for each node y in N

5) 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.

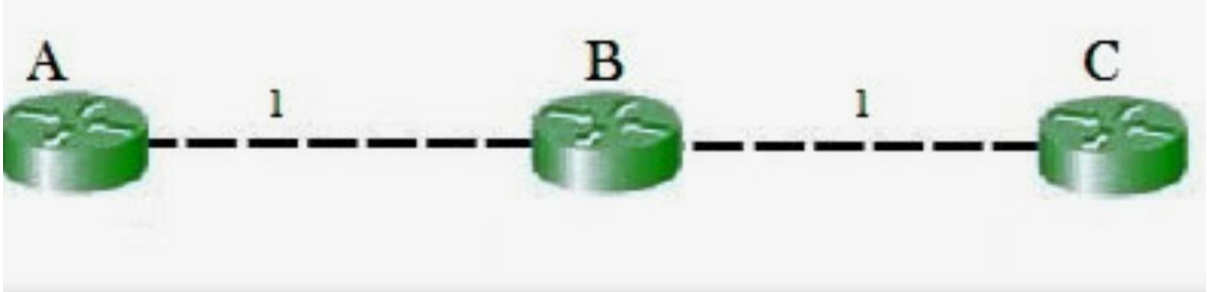
Illustrates the operation of the DV algorithm for the simple three node network.



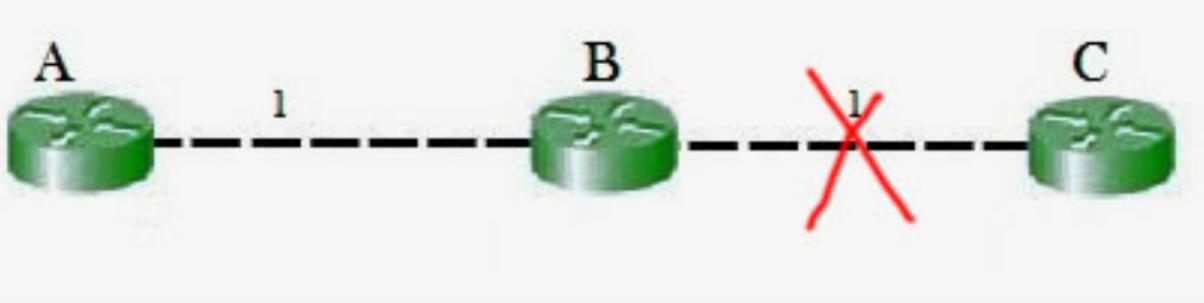
The Count-to-Infinity Problem

The main issue with **D**istance Vector **R**outing (DVR) protocols is Routing Loops since Bellman-Ford Algorithm cannot prevent loops. This routing loop in the DVR network causes the Count to Infinity Problem. Routing loops usually occur when an interface goes down or two routers send updates at the same time.

Counting to infinity problem:



So in this example, the Bellman-Ford algorithm will converge for each router, they will have entries for each other. B will know that it can get to C at a cost of 1, and A will know that it can get to C via B at a cost of 2.



If the link between B and C is disconnected, then B will know that it can no longer get to C via that link and will remove it from its table. Before it can send any updates it's possible that it will receive an update from A which will be advertising that it can get to C at a cost of 2. B can get to A at a cost of 1, so it will update a route to C via A at a cost of 3. A will then receive updates from B later and update its cost to 4. They will then go on feeding each other bad information toward infinity which is called as **Count to Infinity problem**.

A Comparison of LS and DV Routing-algorithms

Distance Vector Protocol	Link State Protocol
Entire routing-table is sent as an update	Updates are incremental & entire routing-table is not sent as update
Distance vector protocol send periodic update at every 30 or 90 second	Updates are triggered not periodic
Updates are broadcasted	Updates are multicasted
Updates are sent to directly connected neighbour only	Update are sent to entire network & to just directly connected neighbour
Routers don't have end to end visibility of entire network.	Routers have visibility of entire network of that area only.
Prone to routing loops	No routing loops
Each node talks to only its directly connected neighbors	Each node talks with all other nodes (via broadcast)

5. LINK STATE ROUTING

Link state routing is the second family of routing protocols. While distance-vector routers use a distributed algorithm to compute their routing tables, link-state routing uses link-state routers to exchange messages that allow each router to learn the entire network topology. Based on this learned topology, each router is then able to compute its routing table by using the shortest path computation.

Link state routing is a technique in which each router shares the knowledge of its neighborhood with every other router i.e. the internet work. The three keys to understand the link state routing algorithm.

1. **Knowledge about the neighborhood**: Instead of sending its routing table, a router sends the information about its neighborhood only. A router broadcast its identities and cost of the directly attached links to other routers.
2. **Flooding:** Each router sends the information to every other router on the internetwork except its neighbors. This process is known as flooding. Every router that receives the packet sends the copies to all the neighbors. Finally each and every router receives a copy of the same information.
3. **Information Sharing**: A router send the information to every other router only when the change occurs in the information.

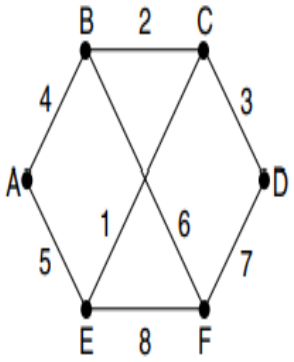
Link state routing has two phase:

1. **Reliable Flooding: Initial state**– Each node knows the cost of its neighbors. Final state- Each node knows the entire graph.
2. **Route Calculation**: Each node uses Dijkstra' s algorithm on the graph to calculate the optimal routes to all nodes. The link state routing algorithm is also known as Dijkstra's algorithm which is used to find the shortest path from one node to every other node in the network.

Features of Link State Routing Protocols

- **Link State Packet:** A small packet that contains routing information.
- **Link-State Database:** A collection of information gathered from the link-state packet.
- **Shortest Path First Algorithm (Dijkstra algorithm):** A calculation performed on the database results in the shortest path
- **Routing Table:** A list of known paths and interfaces.

Illustration:

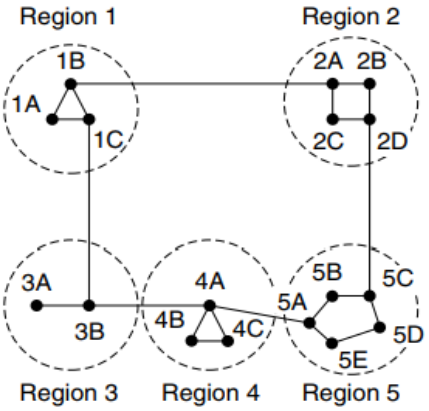


Link State Packets

A	B	C	D	E	F
Seq.	Seq.	Seq.	Seq.	Seq.	Seq.
Age	Age	Age	Age	Age	Age
B 4	A 4	B 2	C 3	A 5	B 6
E 5	C 2	D 3	F 7	C 1	D 7
	F 6	E 1		F 8	E 8

6. HIERARCHICAL ROUTING:

Hierarchical routing protocols consist of a hierarchical topology to organize the network and routing information. Multiple layers and levels are introduced in a network. Each layer may be assigned a different responsibility like forwarding packets, maintaining routing tables, etc. HRPs are valuable for large networks, as they provide the capability of organizing network information and reducing the amount of routing information that should be exchanged between nodes. Hence, HRPs demonstrate significant scalability and fault tolerance. This is attributed to their hierarchical structure, which provides redundancy and facilitates the efficient distribution of routing data throughout the network.



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

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

7. BROADCAST ROUTING

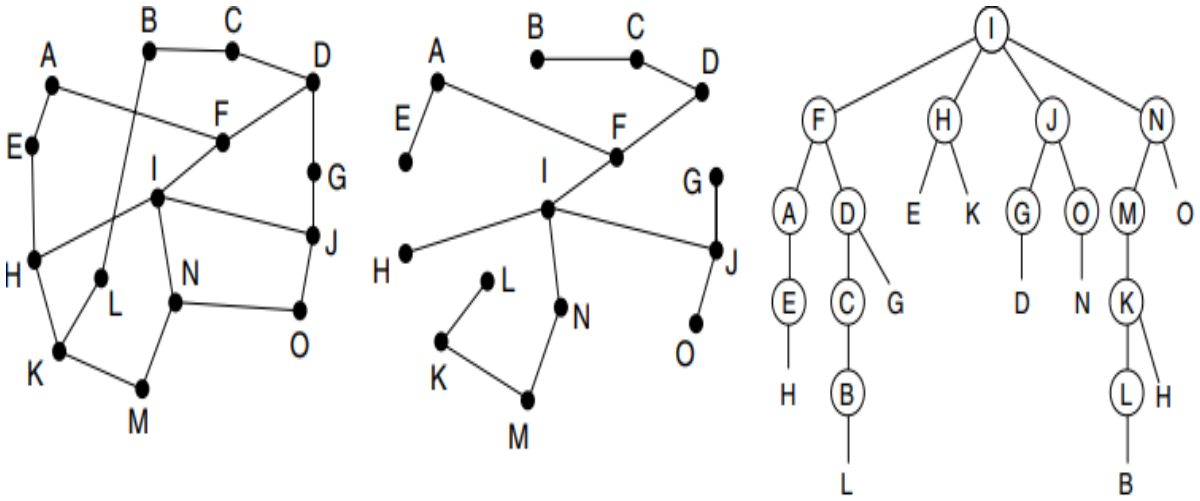
It plays a role, in computer networking and telecommunications. It involves transmitting data, messages, or signals from one source to destinations within a network. Unlike routing (one-to-one communication) or multicast routing (one-to-many communication) broadcast routing ensures that information reaches all devices or nodes within the network.

Mechanisms for Broadcast Routing

The mechanisms and protocols are employed to efficiently distribute data to multiple recipients through broadcast routing. Here are some important methods:

- **Flooding:** Flooding is an approach to broadcast routing. In this method, the sender broadcasts the message to all connected devices, which then forwards it to their connected devices and so on. This continues until the message reaches all intended recipients or a predefined maximum number of hops is reached. However flooding can lead to network congestion and inefficiency.
- **Spanning Tree Protocol (STP):** STP is utilized in Ethernet networks to prevent loops and ensure broadcast routing. It establishes a tree structure that connects all devices, in the network while avoiding paths. Reducing network congestion and avoiding broadcast messages are the benefits of implementing this approach.

- **The Internet Group Management Protocol (IGMP):** It is a communication protocol utilized in IP networks to facilitate the management of group memberships. Its purpose is to enable hosts to join or leave groups ensuring that only interested recipients receive the multicast traffic. This not enhances network efficiency. Also prevents unnecessary data transmission.
- **Broadcast Domains:** Segmenting a network into broadcast domains also known as subnetting is a way to manage and control the scope of broadcast messages. By dividing a network into segments we can contain the impact of broadcast traffic within each segment minimizing its overall effect, on the entire network.



the idea for **reverse path forwarding** is elegant and remarkably simple once it has been pointed out (Dalal and Metcalfe, 1978). When a broadcast packet arrives at a router, the router checks to see if the packet arrived on the link that is normally used for sending packets toward the source of the broadcast. If so, there is an excellent chance that the broadcast packet itself followed the best route from the router and is therefore the first copy to arrive at the router. This being the case, the router forwards copies of it onto all links except the one it arrived on. If, however, the broadcast packet arrived on a link other than the preferred one for reaching the source, the packet is discarded as a likely duplicate.

Figure shows a network, a sink tree for router I of that network, and how the reverse path algorithm works. On the first hop, I sends 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 I (assuming that the preferred path falls along the sink tree) 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.

The principal advantage of reverse path forwarding is that it is efficient while being easy to implement. It sends the broadcast packet over each link only once in each direction, just as in flooding, yet it requires only that routers know how to reach all destinations, without needing to

remember sequence numbers (or use other mechanisms to stop the flood) or list all destinations in the packet.

Our last broadcast algorithm improves on the behavior of reverse path forwarding. It makes explicit use of the sink tree—or any other convenient spanning tree—for the router initiating the broadcast. A **spanning tree** is a subset of the network that includes all the routers but contains no loops. Sink trees are spanning trees. 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 for example, when the sink tree of part (b) is used as the spanning tree, the broadcast packet is sent with the minimum 14 packets. The only problem is that each router must have knowledge of some spanning tree for the method to be applicable. Sometimes this information is available.

Multicast Routing

Some applications, such as a multiplayer game or live video of a sports event streamed to many viewing locations, send packets to multiple receivers. Unless the group is very small, sending a distinct packet to each receiver is expensive. On the other hand, broadcasting a packet is wasteful if the group consists of, say, 1000 machines on a million-node network, so that most receivers are not interested in the message Thus, we need a way to send messages to well-defined groups that are numerically large in size but small compared to the network as a whole. Sending a message to such a group is called multicasting, and the routing al gorithm used is called multicast routing. All multicasting schemes require some way to create and destroy groups and to identify which routers are members of a group.

Multicast routing schemes build on the broadcast routing schemes we have already studied, sending packets along spanning trees to deliver the packets to the members of the group while making efficient use of bandwidth. However, the best spanning tree to use depends on whether the group is dense, with receivers scattered over most of the network, or sparse.

MOSPF (Multicast OSPF) is an example of a link state protocol. DVMRP (Distance Vector Multicast Routing Protocol) is an example of a multicast routing protocol that works this way

Routing for Mobile Hosts

- Millions of people use mobile hosts for communications while they are moving from one location to other location.
- All mobile hosts assumed to have permanent home location. Where they will get registered. Each hosts also has a permanent home address that can be used to determine its home location, analogous to the way the telephone number 1-212-5551212
- When they are moved to new Internet locations, mobile host acquire new network addresses.
- The basic idea used for mobile routing in the Internet and cellular networks is for the mobile host to tell a host at the home location where it is now. This host, which acts on behalf of the mobile host, is called the home agent. Once it knows where the mobile host is currently located, it can forward packets so that they are delivered
- Fig. 5-19 shows mobile routing in action. A sender in the northwest city of Seattle wants to send a packet to a host normally located across the United States in New York. The case of interest to us is when the mobile host is not at home. Instead, it is temporarily in San Diego. The mobile host in San Diego must acquire a local network address before it can use the network. This happens in the normal way that hosts obtain network addresses; we will cover how this works for the Internet later in this chapter. The local address is called a care of address. Once the mobile host has this address, it can tell its home agent where it is now. It does this by sending a registration message to the home agent (step 1) with the care of address. The message is shown with a dashed line in Fig. 5-19 to indicate that it is a control message, not a data message. Next, the sender sends a data packet to the mobile host using its permanent address (step 2). This packet is routed by the network to the host's home location because that is where the home address belongs. In New York, the home agent intercepts this packet because the mobile host is away from home. It then wraps or encapsulates the packet with a new header and sends this bundle to the care of address (step 3). This mechanism is called tunnelling.

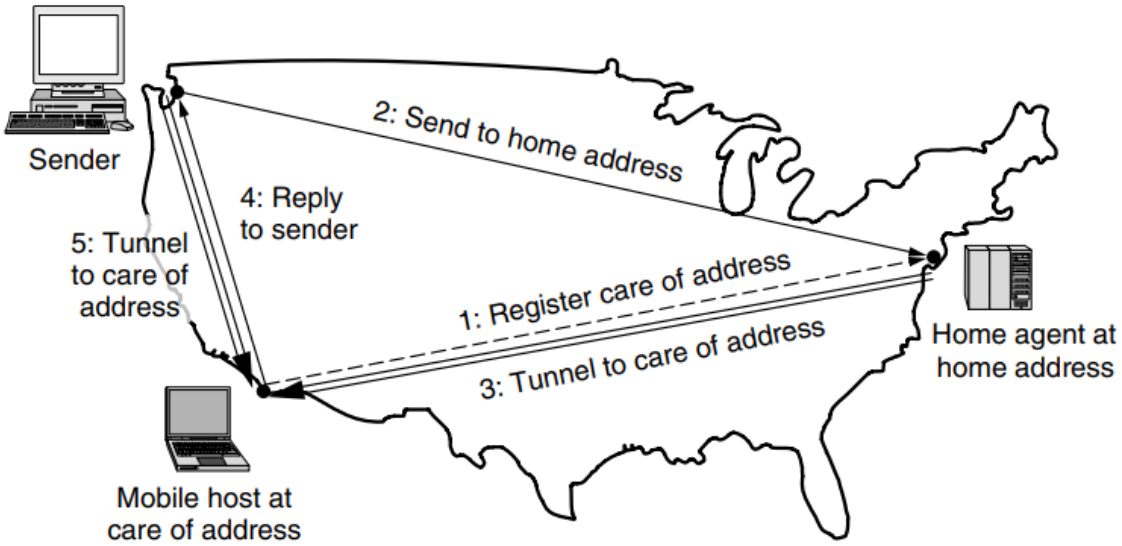


Figure 5-19. Packet routing for mobile hosts.

When the encapsulated packet arrives at the care of address, the mobile host unwraps it and retrieves the packet from the sender. The mobile host then sends its reply packet directly to the sender (step 4). The overall route is called triangle routing because it may be circuitous if the remote location is far from the home location. As part of step 4, the sender may learn the current care of address. Sub sequent packets can be routed directly to the mobile host by tunneling them to the care of address (step 5), bypassing the home location entirely. If connectivity is lost for any reason as the mobile moves, the home address can always be used to reach the mobile.

Routing in Ad hoc Network

Let us consider the possibilities are emergency workers at an earthquake site, military vehicles on a battlefield, a fleet of ships at sea, or a gathering of people with laptop computers in an area lacking 802.11.

In all these cases, and others, each node communicates wirelessly and acts as both a host and a router. Networks of nodes that just happen to be near each other are called ad hoc networks or MANETs (Mobile Ad hoc NETworks).

What makes ad hoc networks different from wired networks is that the topo logy is suddenly tossed out the window. Nodes can come and go or appear in new places at the drop of a bit. With a wired network, if a router has a valid path to some destination, that path continues to be valid

barring failures, which are hope fully rare. With an ad hoc network, the topology may be changing all the time, so the desirability and even the validity of paths can change spontaneously without warning.

As an example, we will look at one of the most popular routing algorithms, AODV (Ad hoc On-demand Distance Vector)

Route Discovery

In AODV, routes to a destination are discovered on demand, that is, only when a somebody wants to send a packet to that destination. This saves much work that would otherwise be wasted when the topology changes before the route is used.

Route Maintenance.

Because nodes can move or be switched off, the topology can change spontaneously. The algorithm needs to be able to deal with this. Periodically, each node broadcasts a Hello message. Each of its neighbours is expected to respond to it. If no response is forthcoming, the broadcaster knows that that neighbour has moved out of range or failed and is no longer connected to it. Similarly, if it tries to send a packet to a neighbour that does not respond, it learns that the neighbour is no longer available.

At this stage, the invalid routes have been purged from the network, and send ers can find new, valid routes by using the discovery mechanism that we described. However, there is a complication. Recall that distance vector protocols can suffer from slow convergence or count-to-infinity problems after a topology change in which they confuse old, invalid routes with new, valid routes.

****