Course Plan			
Module	**Contents**	**Hours**	**End Sem. Exam Marks**
III	Network layer – Routing – Shortest path routing, Flooding, Distance Vector Routing, Link State Routing, RIP, OSPF, Routing for mobile hosts.	**07**	**15%**

The Network Layer

The Network Layer:

Responsible for delivering packets between endpoints over multiple links

- Concerned with getting packets from source to destination.
- The network layer must know the topology of the subnet and choose appropriate paths through it.
- When source and destination are in *different networks,* the network layer (IP) must deal with these differences.

Design Goals:

- The services provided by the network layer should be independent of the subnet topology.
- The Transport Layer should be shielded from the number, type and topology of the subnets present.
- The network addresses available to the Transport Layer should use a uniform numbering plan (even across LANs and WANs).

Design issues

1. Store and forward packet switching
2. Services provided to the transport layer
3. Implementation of the connectionless services
4. Implementation of the connection oriented services
5. Comparison of virtual datagram and subnets

1. Store and forward packet switching

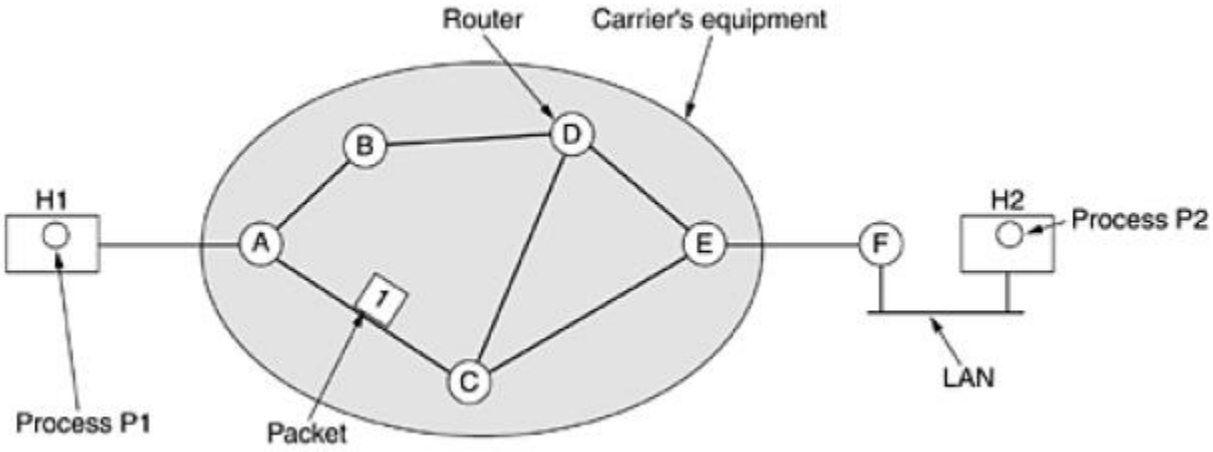


Fig: The environment of the network layer protocol

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 carrier. The packet is stored there until it has fully arrived so the checksum can be verified. 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

2. Services provided to the transport layer

The network layer provides services to the transport layer at the network layer/transport layer interface. The network layer services have been 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.

3. Implementation of the connectionless services

The packets here are frequently called **datagrams** and the subnet is called a **datagram subnet**. 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*. It prepends a transport header to the front of the message and hands the result to the network layer.
Let us assume 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. At this point the carrier takes over. 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.
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**.

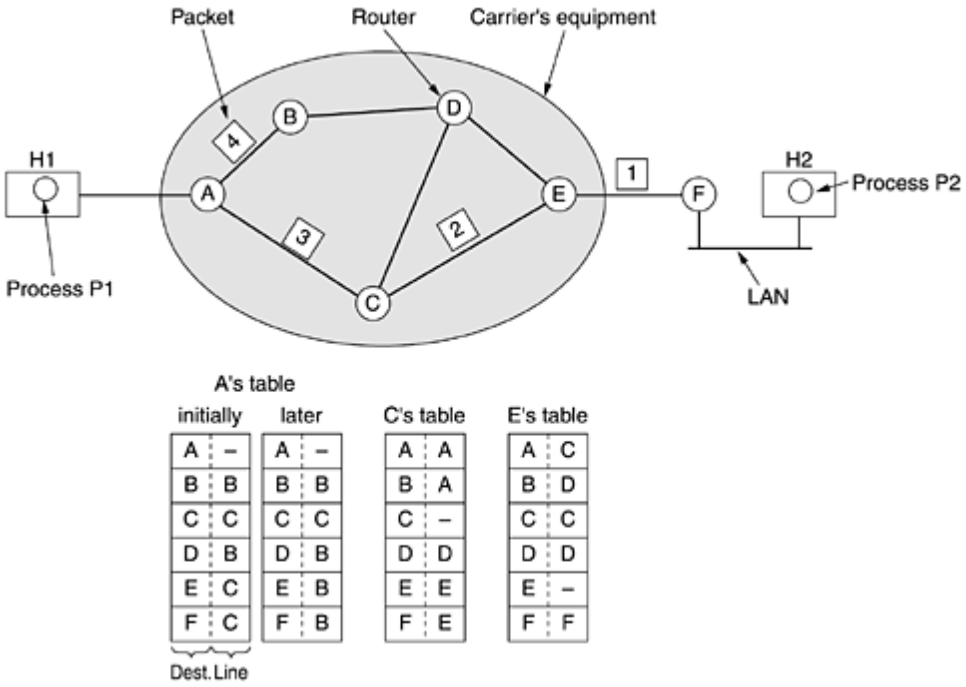


Fig: Routing with a datagram subnet

4. Implementation of the connection oriented services

For connection-oriented service, we need a virtual-circuit subnet. 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 of 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.

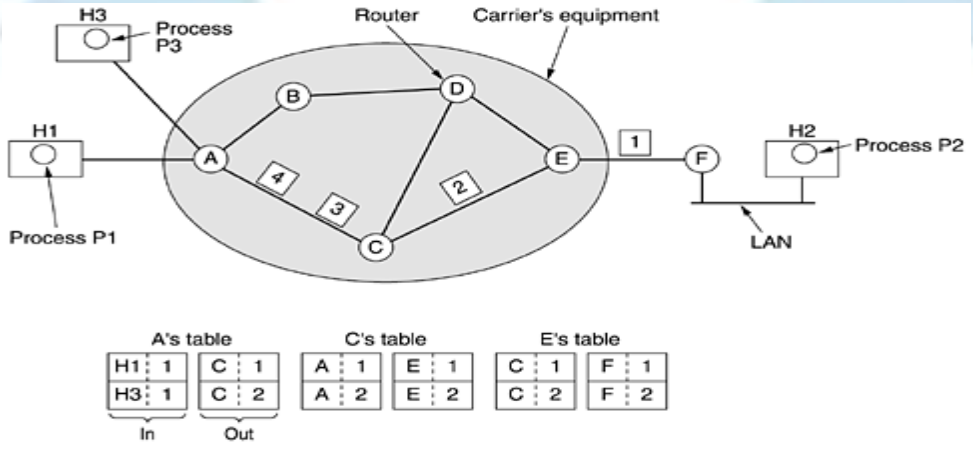


Fig: Routing within a virtual-circuit subnet.

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.

5. Comparison of virtual datagram and subnets

Issue	Datagram subnet	Virtual-circuit subnet
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

Fig. Comparison of datagram and virtual-circuit subnets.

ROUTING ALGORITHMS

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 **correctness, simplicity, robustness, stability, fairness, and optimality** are the features of the routing algorithm.

Robustness:
Once a major network comes on the air, it may be expected to run continuously for years without system wide failures. During that period there will be hardware and software failures of all kinds. Hosts, routers, and lines will fail repeatedly, and the topology will change many times. The routing algorithm should be able to cope with changes in the topology and traffic without requiring all jobs in all hosts to be aborted and the network to be rebooted every time some router crashes.

Fairness and Optimality:
Tradeoff between fairness and optimality affects the delay and throughput.

The optimality Principle

- Optimality is optimal routes without regard to network topology or traffic. There will be a set of optimal routes from all sources to a given destination form a tree rooted at the destination. Such a tree is called a **sink tree.**
- A sink tree is not necessarily unique.
- The goal of all routing algorithms is to discover and use the sink trees for all routers.

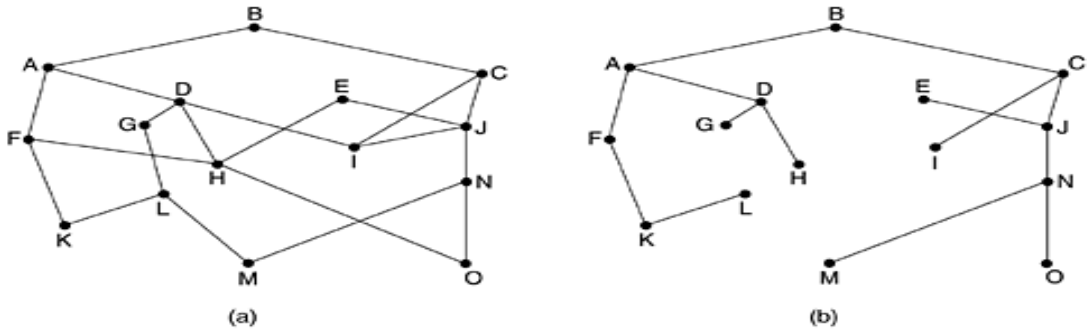


Fig: (a) A subnet. (b) A sink tree for router B.

Shortest Path Routing

The idea 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 (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.

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* in Fig. 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).

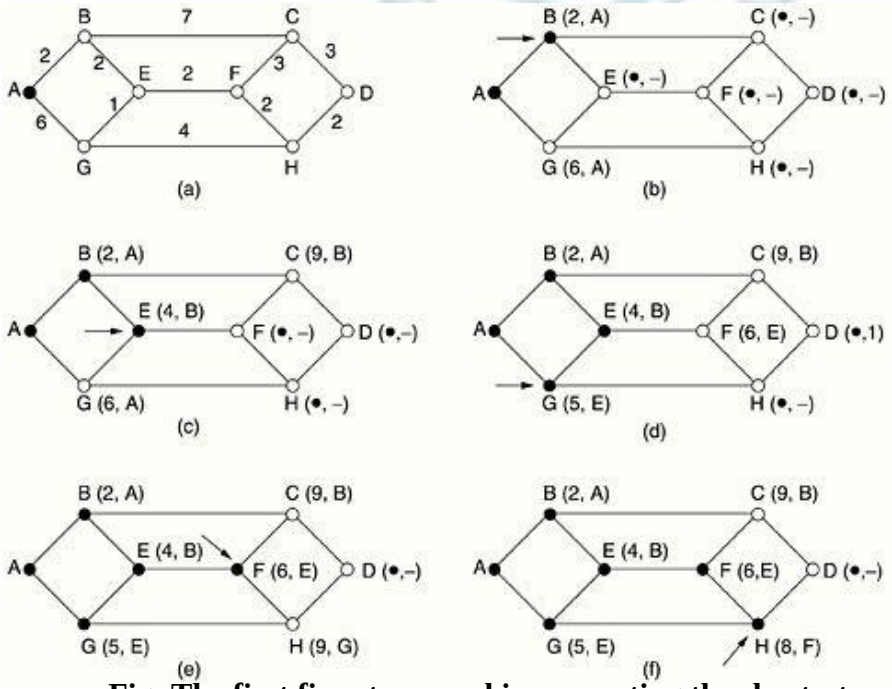


Fig: The first five steps used in computing the shortest path from A to D. The arrows indicate the working node.

Several algorithms for computing the shortest path between two nodes of a graph are known. This one is due to Dijkstra (1959). Each node is labeled (in parentheses) with its distance from the source node along the best known path. Initially, no paths are known, so all nodes a relabeled 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. To illustrate how the labeling algorithm works, look at

the weighted, undirected graph of Fig.(a), where the weights represent, for example, distance. We want to find the shortest path from *A* to *D*. We start out by marking node *A* as permanent, indicated by a filled-in circle. Then we examine, in turn, each of the nodes adjacent to *A*(the working node), relabeling each one with the distance to *A*. Whenever a node is relabeled, we also label it with the node from which the probe was made so that we can reconstruct the final path later. Having examined each of the nodes adjacent to *A*, we examine all the tentatively labeled nodes in the whole graph and make the one with the smallest label permanent, as shown in Fig. (b). This one becomes the new working node. We now start at *B* and examine all nodes adjacent to it. If the sum of the label on *B*and the distance from *B* to the node being considered is less than the label on that node, we have a shorter path, so the node is relabeled.

After all the nodes adjacent to the working node have been inspected and the tentative label s changed if possible, the entire graph is searched for the tentatively-labeled node with the smallest value. This node is made permanent and becomes the working node for the next round. Figure shows the first five steps of the algorithm. To see why the algorithm works, look at Fig.(c). At that point we have just made *E* permanent. Suppose that there were a shorter path than *ABE*, say *AXYZE*. There are two possibilities: either node *Z* has already been made permanent, or it has not been. If it has, then *E* has already been probed (on the round following the one when *Z* was made permanent), so the *AXYZE* path has not escaped our attention and thus cannot be a shorter path. Now consider the case where *Z* is still tentatively labeled. Either the label at *Z* is greater than or equal to that at *E*, in which case *AXYZE* cannot be a shorter path than *ABE*, or it is less than that of *E*, in which case *Z* and not *E* will become permanent first, allowing *E* to be probed from *Z*

Flooding

Flooding, in which every incoming packet is sent out on every outgoing line except the one it arrived on. Flooding obviously generates vast numbers of duplicate packets, in fact, an infinite number unless some measures are taken to damp the process. One such measure is to have a hop counter contained in the header of each packet, which is decremented at each hop, with the packet being discarded when the counter reaches zero. Ideally, the hop counter should be initialized to the length of the path from source to destination. If the sender does not know how long the path is, it can initialize the counter to the worst case, namely, the full diameter of the subnet

Distance vector routing

Here the least cost route between two nodes is the route with minimum distance. Each node maintains a routing table with minimum distances to every node. The table also shows the next node it has to hop to for going to the destination if the destination node is not its immediate neighbour.

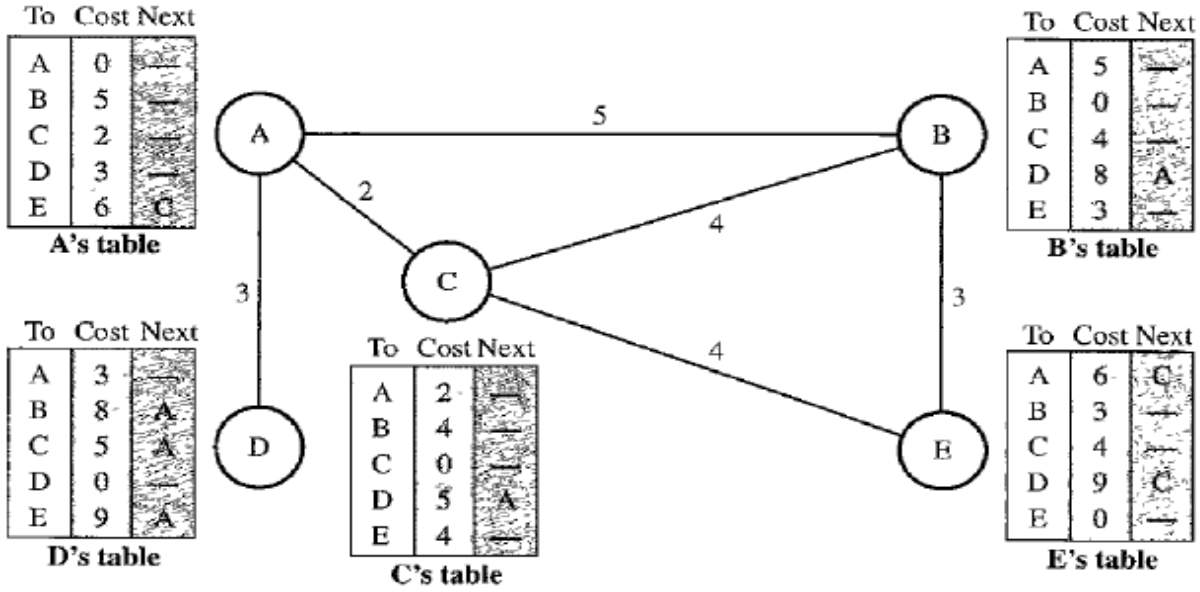


Fig: System with five nodes and their corresponding Distance Vector Routing Tables

For example the table for node A shows how we can reach other nodes in the system. To reach E it has to pass through C and the cost is 6.

Initialization

Initially the nodes will create a table with its immediate neighbours. The other nodes will be entered with cost ∞

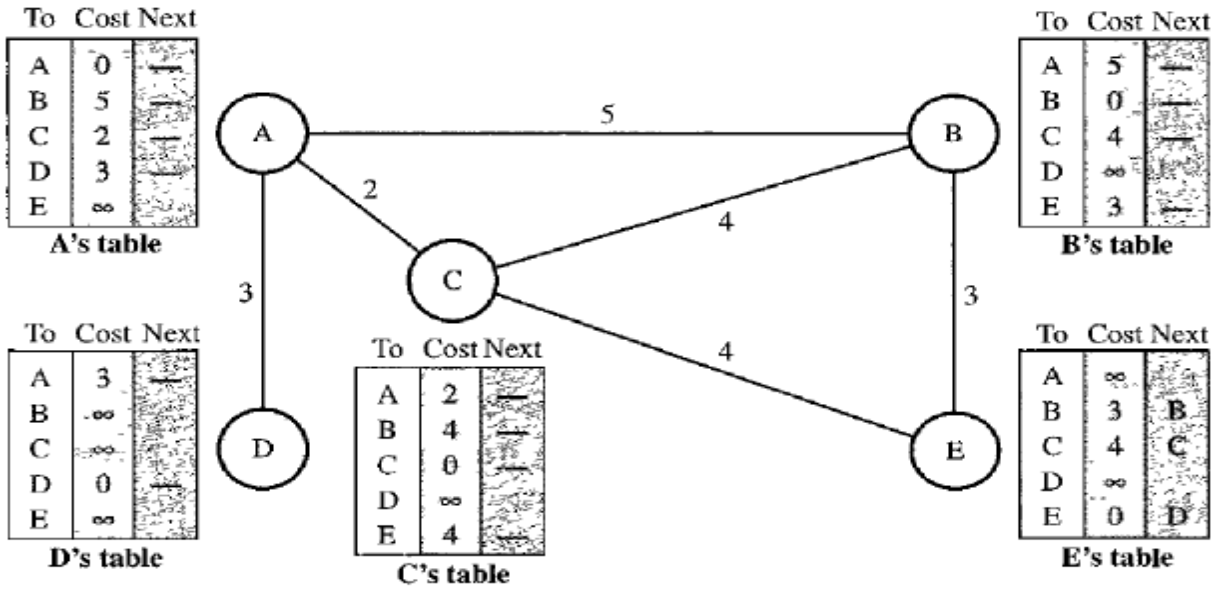


Fig: Initialization of tables in Distance Vector Routing

Sharing

Every node shares the information with their neighbours. Initially node A has no information about E but node C has. Node A updates it by getting information from C.Only the first two columns of the table is shared.

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

Updating

When a node receives the table from the neighbour updating takes place in 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.

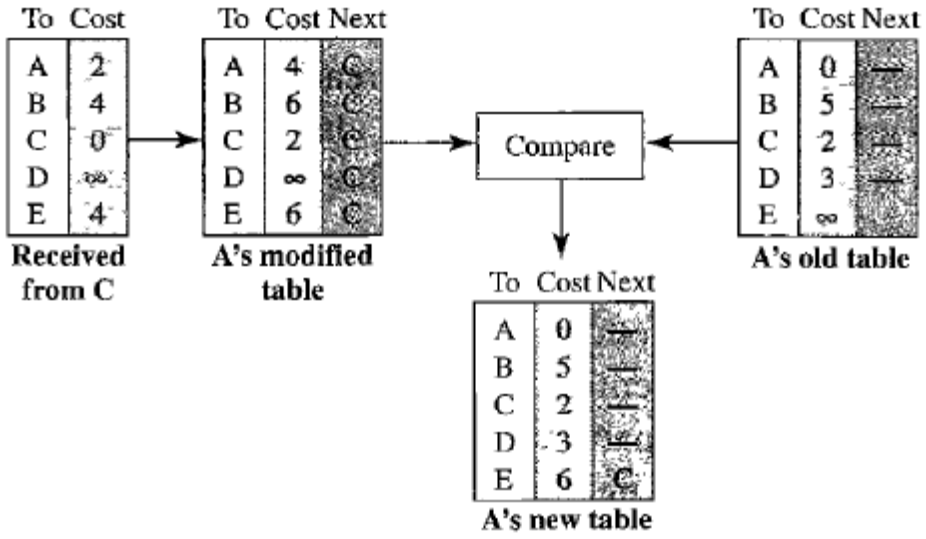


Fig: Updating in Distance Vector Routing

When to share

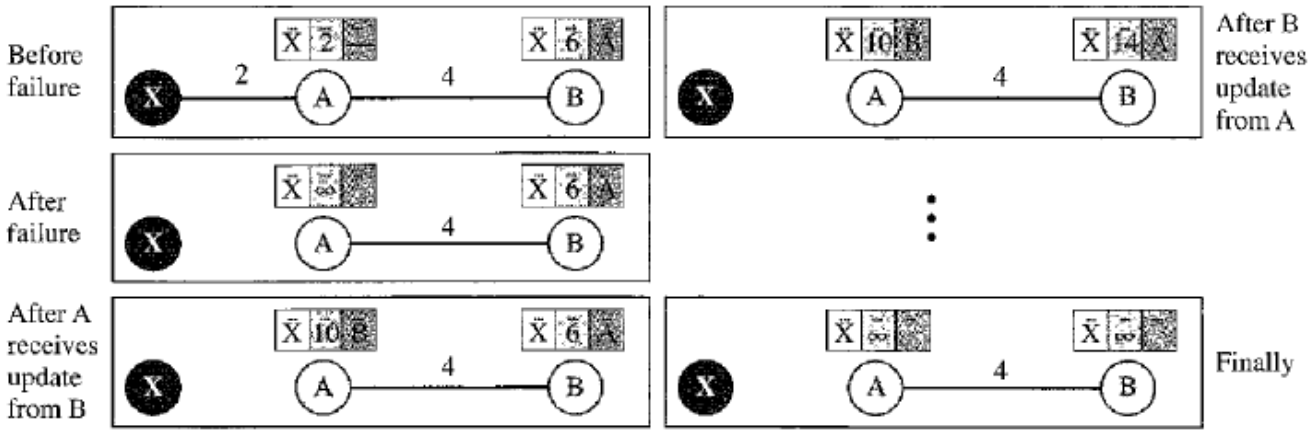
In distance vector routing, each node shares its routing table with its immediate neighbors periodically and when there is a change.

Periodic Update A node sends its routing table, normally every 30 s, in a periodic update. The period depends on the protocol that is using distance vector routing.

Triggered Update A node sends its two-column routing table to its neighbors anytime there is a change in its routing table. This is called a **triggered update.** The change can result from the following.

1. A node receives a table from a neighbor, resulting in changes in its own table after updating.
2. A node detects some failure in the neighboring links which results in a distance change to infinity.

Two Node Instability (Count to Infinity/slow convergence)



A problem with distance vector routing is instability where the network becomes unstable. At beginning both nodes know how to reach node X. But suddenly the link to X from A fails. Node A updates the table. But before the table B updates the new information about the failed link, it sends its data to table A. Node A receives the update and thinks that there is another route to X from B. Based on the triggered update strategy, table A sends the news to B. Now B thinks that something has been changed around A and updates its routing table. The cost of reaching X increases gradually until it reaches infinity. At this moment, both A and B know that X cannot be reached. This time the network is not stable. Packets bounce between A and B creating two node loop problem.

Solutions to this problems are as follows:

- We can define the distance between each node to be 1 and define 16 as infinity.
- Another solution is Split Horizon. Here instead of flooding the table,each node sends only part of the table through each interface. If according to its table, node B thinks that the optimum route to reach X via A, it does not need to advertise this piece of information to A.Later when node A sends its routing table to B,node B also corrects its routing table. The system becomes stable after the first update and B knows X is not reachable.
- Split Horizon and Poison Reverse. Disadvantage of the split horizon is when there is no news about a route for a long time the node deletes that route from its table. In the previous scenario there is a chance of A deleting the route to B due to split horizon. So to avoid this, B sends back an acknowledgement kind of information back to A saying the cost is ∞ about the routes which it has currently received from A. This is Poison Reverse.

- Another solution is to define infinity to be a much smaller value such as 15. Then it does not take too long to become stable.
- Another solution is when a major change occurs advertise the change quickly, but don't accept new routes for a period of time.

Link State Routing

Distance vector routing was used in the ARPANET until 1979, when it was replaced by link state routing.
The idea behind link state routing can be stated as five parts. Each router must do the following:

1. Discover its neighbours and learn their network addresses.
2. Measure the delay or cost to each of its neighbours.
3. Construct a packet telling all it has just learned.
4. Send this packet to all other routers.
5. Compute the shortest path to every other router.

1. Learning about the Neighbours

When a router is booted, its first task is to learn who its neighbours are. So it sends a special **HELLO packet** on each point-to-point line. The router on the other end is expected to send back a reply telling who it is. These names must be globally unique.

2. Measuring Line Cost

The link state routing algorithm requires each router to know the delay to each of its neighbours. To determine this delay it sends 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.
When a router has a choice between two lines with the same bandwidth, one of which is heavily loaded all the time and one of which is not, the router will regard the route over the unloaded line as a shorter path. This choice will result in better performance.

Consider the subnet of figure, which is divided into two parts, East and West, connected by two lines, *CF* and *EI*.

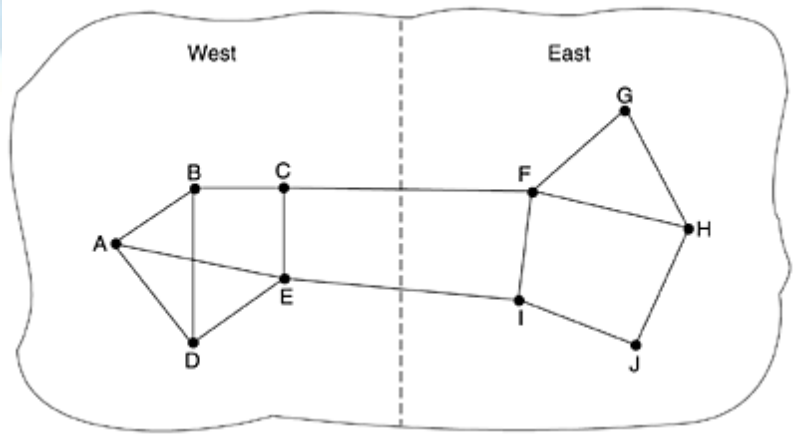


Fig: A subnet in which the East and West parts are connected by two lines.

3. Building Link State Packets

Once the information needed for the exchange has been collected, the next step is for each router to build a packet containing all the data. The 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. An example subnet is given in figure with delays shown as labels on the lines. The corresponding link state packets for all six routers are shown in figure

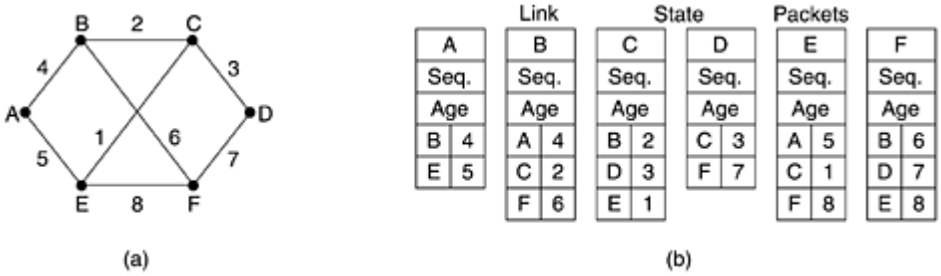


Fig: (a) A subnet. (b) The link state packets for this subnet.

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. Distributing the Link State Packets

The link state packet has to be distributed reliably. The fundamental idea is to use flooding to distribute the link state packets. To keep the flood in check, each packet contains a sequence number that is incremented for each new packet sent. Routers keep track of all the (source router, sequence) pairs they see. When a new link state packet comes in, it is checked against the list of packets already seen. If it is new, it is forwarded on all lines except the one it arrived on. If it is a duplicate, it is discarded. If a packet with a sequence number lower than the highest one seen so far ever arrives, it is rejected as being obsolete since the router has more recent data.

Problems related to numbering of the packets

- If a router ever crashes, it will lose track of its sequence number. If it starts again at 0, the next packet will be rejected as a duplicate.
- If a sequence number is ever corrupted and 65,540 is received instead of 4 (a 1-bit error), packets 5 through 65,540 will be rejected as obsolete, since the current sequence number is thought to be 65,540.

The solution to all these problems is to include the age of each packet after the sequence number and decrement it once per second. When the age hits zero, the information from that router is discarded. The *Age* field is also decremented by each router during the initial flooding process, to make sure no packet can get lost and live for an indefinite period of time (a packet whose age is zero is discarded).

Some refinements to this algorithm make it more robust. When a link state packet comes in to a router for flooding, it is not queued for transmission immediately. Instead it is first put in a holding area to wait a short while. If another link state packet from the same source comes in before the first packet is transmitted, their sequence numbers are compared. If they are equal, the duplicate is discarded. If they are different, the older one is thrown out. To guard against errors on the router-router lines, all link state packets are acknowledged. When a line goes idle, the holding area is scanned in round-robin order to select a packet or acknowledgement to send.

The data structure used by router *B* for the subnet shown in figure above is depicted in figure . Each row here corresponds to a recently-arrived, but as yet not fully-processed, link state packet. The table records where the packet originated, its sequence number and age, and the data. In addition, there are send and acknowledgement flags for each of *B*'s three lines (to *A*,*C*, and *F*,

respectively). The send flags mean that the packet must be sent on the indicated line. The acknowledgement flags mean that it must be acknowledged there.

			Send flags			ACK flags			
Source	Seq.	Age	A	C	F	A	C	F	Data
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

Fig: The packet buffer for router B

5. Computing the New Routes

Once a router has accumulated a full set of link state packets, it can construct the entire subnet graph.Now 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.

RIP(Routing Information Protocol)

- The **Routing Information Protocol (RIP)** is one of the most commonly used Interior Gateway Protocol which helps a router dynamically adapt to changes of network connections .
- RIP was first developed in 1969 as a part of ARAPNET.
- It is a distance-vector protocol, which employs **Hop Count** as the metric.
- Hop count along the path refers to the routers the datagram passes through while going from source to destination. The maximum number of hops allowed with RIP is 15.
- It runs above Network layer of the Internet protocol suite, using **UDP port 520** to carry its data.
- RIP uses a distributed version of **Bellman-Ford algorithm.** *Bellman-Ford algorithm* computes single-source shortest paths in a weighted graph (where some of the edge weights may be negative).

The algorithm is distributed because it involves a number of nodes (routers) within an Autonomous system. It consists of the following steps:

- Each node calculates the distances between itself and all other nodes within the AS and stores this information as a table.
- Each node sends its table to all neighboring nodes.
- When a node receives distance tables from its neighbors, it calculates the shortest routes to all other nodes and updates its own table to reflect any changes.
- The main disadvantages of Bellman-Ford algorithm in this setting are
  - Does not scale well
  - Changes in network topology are not reflected quickly since updates are spread node-by-node.
  - Counting to infinity

- RIP partitions participants (node within the AS) into ***active*** **and** ***passive*** (silent) nodes. Active routers advertise their routes to others; passive node just listens and updates their routes based on the advertisements. Passive nodes do not advertise.

Routing Table Format

Each entry in a RIP routing table provides a variety of information, including the ultimate destination, the next hop on the way to that destination, and a metric. The metric indicates the distance in number of hops to the destination.

RIP Timers

Periodic timer:

- controls the advertising of regular update messages
- Protocol specifies it to be 30s but working model is 25-35s
- Prevents synchronization and therefore overload
- It counts down and when it reaches zero, update message is sent

Expiration timer:

- governs the validity of a route. When a
- router receives update, it sets timer to 180s.
- No update within 180s? Hop count of route set to 16, which means dest. unreachable.

Garbage collection timer:

- If route information is invalid, it is not immediately purged from the table
- The route is advertised with a metric value 16
- Also timer set to 120s after route set to 16.
- When timer expires, then toss route info.

Hop-Count Limit

RIP prevents routing loops from continuing indefinitely by implementing a limit on the number of hops allowed in a path from the source to a destination. The maximum number of hops in a path is 15. If a router receives a routing update that contains a new or changed entry, and if increasing the metric value by 1 causes the metric to be infinity (that is, 16), the network destination is considered unreachable. The downside of this stability feature is that it limits the maximum diameter of a RIP network to less than 16.

Count to infinity problem in RIP

In *Count-to Infinity problem* as discussed in distance vector routing*(explain as in distance vector routing), i.e. bad news travel slowly through the network and to advertise a bad news throughout the entire network will take a long time. This problem is also called as *slow convergence problem*.

Solutions to count to infinity are hold-downs, split horizons, poison reverse updates and triggered updates.

Hold-Downs

Hold downs tell routers to hold on to any changes that might affect recently removed routes, for a certain period of time to update the entire network with a route change

Split Horizons

Here the router does not propagates its information about that route back to its sender.

Poison Reverse Updates

In Poison Reverse Updates an infinite cost is included in the broadcast.

Triggered Updates

Triggered updates force a router to send an immediate broadcast when receiving bad news, instead of waiting for the next periodic broadcast. By sending an update immediately, a router minimizes the time it is vulnerable to believing in good news.

RIP Message Format

RIP messages can be broadly classified into two types: routing information messages and messages used to request information. Both uses same format, which consists of fixed header information followed by optional list of network and distance pairs. Figure illustrates the IP RIP packet format.

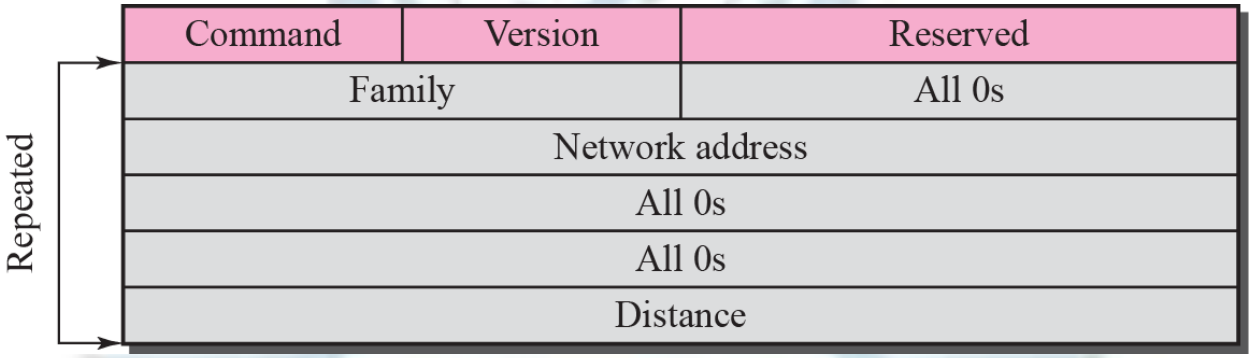
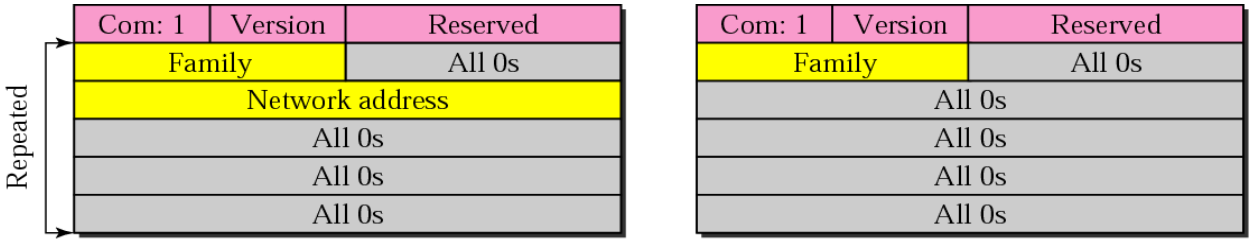


Fig:RIP Message

- Command(8 bit) – specifies type of message: request(1),response(2)
- Version(8 bit) – defines the version
- Family (16 bit) – defines the family of protocol. TCP/IP(2)
- Network address (14 bytes) – address of destination network.
- Distance (32 bit) – defines hop count from advertising router to destination network.

Requests and Responses

- Two types of messages
  - Request
  - Response
- A request message is sent by a router that has just come up or by a router that has some time-out entries.



a. Request for some b. Request for all

- A response message is sent in answer to a request (solicited response, or simply every 30 seconds (unsolicited).
- Response is also called update packet.

RIP version 2

- overcomes the shortcomings of version 1
- New fields
  - Route tag - carries information like AS number. Enables RIP to receive information from an interdomain protocol
  - Subnet mask - 4 byte field that carries subnet mask. RIP2 supports classless addressing and CIDR
  - Next-hop Address - gives address of next hop

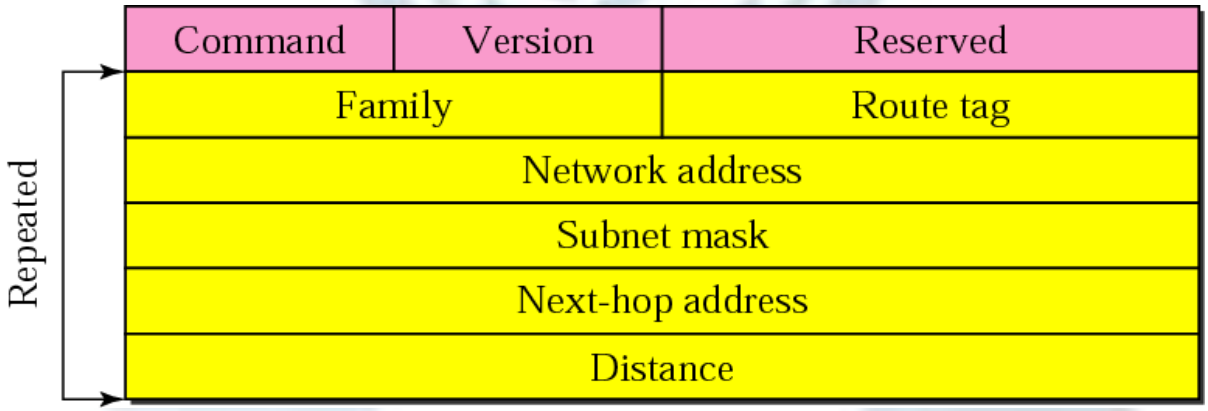


Fig: RIP Version 2

OSPF(Open Shortest Path First)

- The Open Shortest Path First (OSPF) protocol is an intra-domain routing protocol based on link state routing.
- Its domain is also an autonomous system
- Areas
  - OSPF divides an AS into areas for efficient and timely routing
  - Area is a collection of networks, hosts and routers within an AS
  - All networks inside an area must be connected
- Routers inside an area flood the area with routing information
- **Area border routers** summarize the information about the area and send it to other areas

- **Backbone area and backbone routers**
  - All of the areas inside an AS must be connected to the backbone
  - Backbone serves as a primary area, other areas are secondary areas
  - Routers inside the backbone are called backbone routers
- **Virtual link**
  - If the connectivity between a backbone and an area is broken, a virtual link between routers must be created by the administration to allow continuity of the functions of the backbone as the primary area
  - Each area has an area identification. It is zero for the backbone

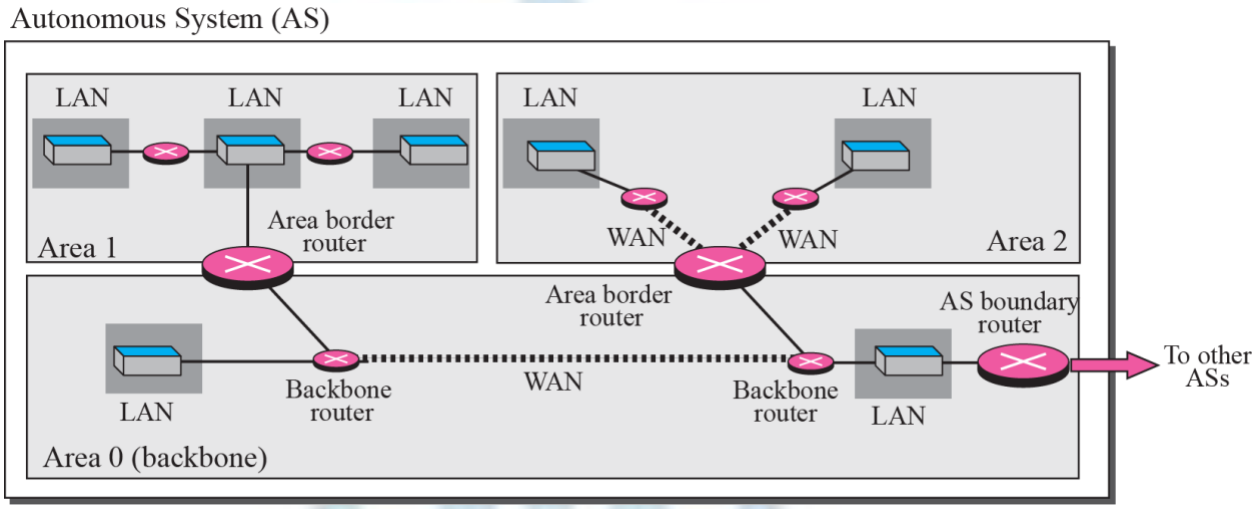


Fig: Areas in Autonomous Systems

- **Metric**
  - OSPF protocol allows the administrator to assign a cost to each route called metric
  - Metric can be based on minimum delay, maximum throughput etc.
- **Types of links**
  - **A connection is called a link**

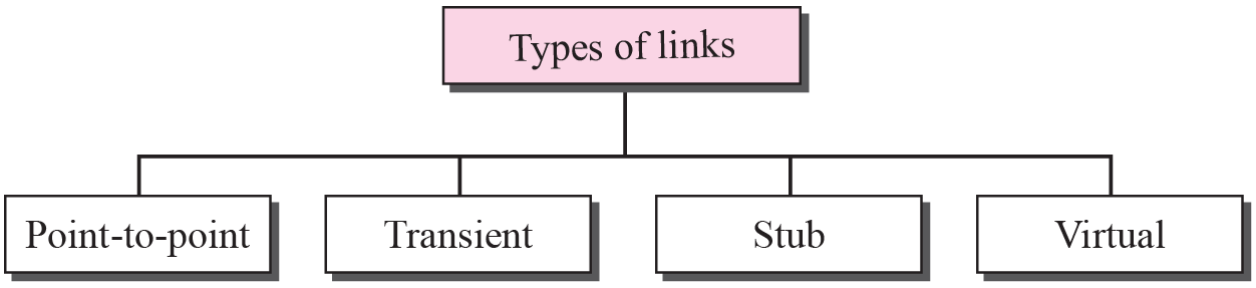
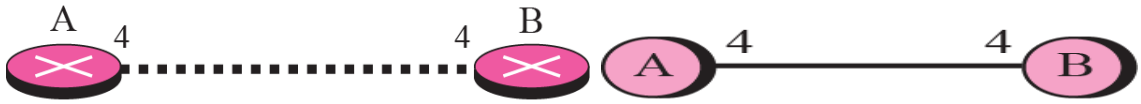


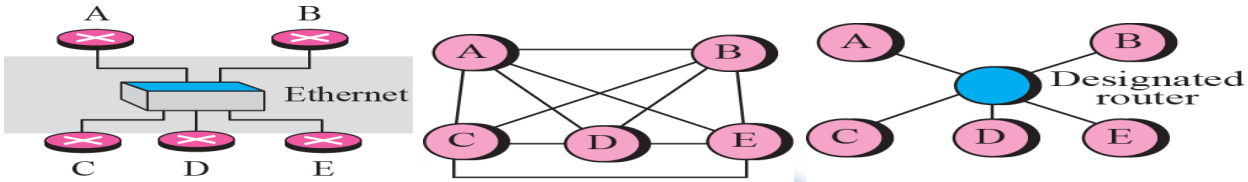
Fig: Types of Links

- **Point-to-point link**
  - Connects two routers without any other host or router in between
  - Ex: two routers connected by a telephone line or a T-line
  - No need to assign n/w address



a. Point-to-point network b. Representation

- **Transient link**
  - n/w with several routers attached to it
  - Data can enter and leave through any router
  - All LANs and WANs with two or more routers are of this type
  - Each router has many neighbors



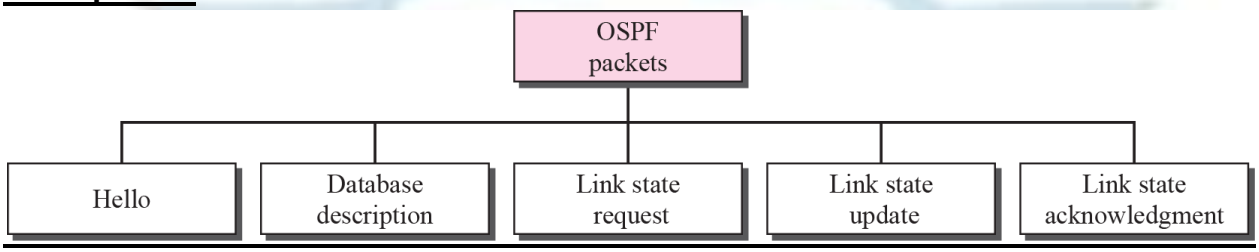
a. Transient network b. Unrealistic c. Realistic

- **Stub link**
  - n/w that is connected to only one router
  - Data packets enter and leave the n/w through the same router
  - Special case of transient n/w

a. Stub network b. Representation

- **Virtual link**
  - When the link between two routers is broken, administration may create a virtual link between them using a longer path that goes through several routers

OSPF packets



1. Router link
2. Network link
3. Summary link to network
4. Summary link to AS boundary router
5. External link

- All OSPF packets have fixed header, which is common to all messages.
- There are various variable part, different for different messages used in OSPF.

- **Fixed Header:** All OSPF packets begin with a 24-byte header, as illustrated in figure. Summary of the functions of different fields are given below:
- **Version number**—identifies the OSPF version used.

- **Type**—Identifies the OSPF packet type as one of the following:

1. **Hello**—Establishes and maintains neighbor relationships.

2. **Database description**—Describes the contents of the topological database. These messages are exchanged when an adjacency is initialized.
3. **Link-state request**—Requests pieces of the topological database from neighbor routers. These messages are exchanged after a router discovers (by examining database-description packets) that parts of its topological database are outdated.
4. **Link-state update**—Responds to a link-state request packet. These messages also are used for the regular dispersal of LSAs. Several LSAs can be included within a single link-state update packet.
5. **Link-state acknowledgment**—Acknowledges link-state updates packets.
   - **Message length**—Specifies the packet length, including the OSPF header, in bytes.
   - **Source Router IP address**—Identifies the source of the packet.
   - **Area ID**—Identifies the area to which the packet belongs. All OSPF packets are associated with a single area.
   - **Checksum**—Checks the entire packet contents for any damage suffered in transit.
   - **Authentication type**—Contains the authentication type. All OSPF protocol exchanges are authenticated. The authentication type is configurable on per-area basis.
   - **Authentication**—Contains authentication information.

   - **Data**—Contains encapsulated upper-layer information.

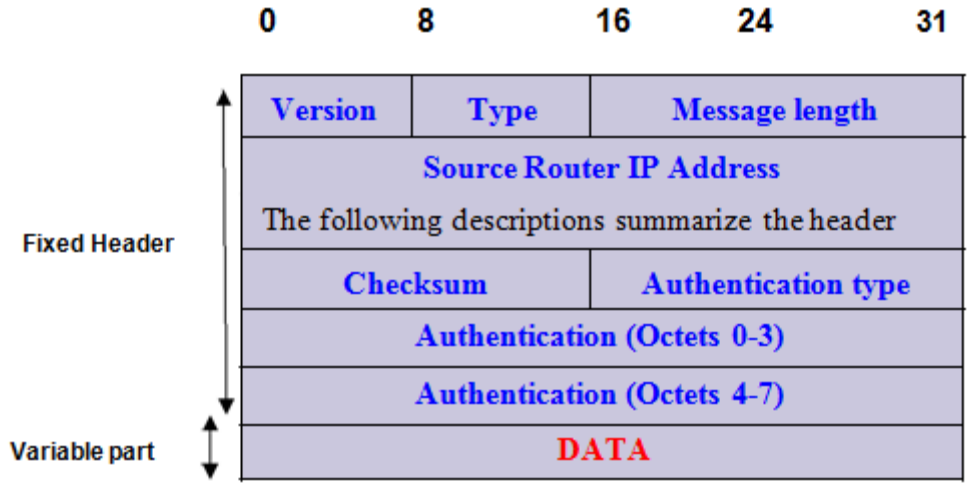


Fig: OSPF Message Header

Hello Message Format: OSPF sends Hello (greeting) messages on each link periodically to establish and test neighbor reachability. The format of this message is shown in Figure. Functions of the header fields are briefly explained below.

- **Fixed Header**: as discussed in previous section and Figure

- **Network mask**: contains mask for the network over which the message is to be send.

- **Dead Timer**: gives time in seconds after which a non-responding neighbor is considered dead.

- **Hello Interval**: means Hello Interval, it is the normal period, in seconds, between hello messages.
- **Gway Prio**: means gateway priority, it is the interior priority of this router, and is used in selecting the backup designated router.

- **Designated Router**: IP address of the router, which is the designated router for the network as viewed by the sender.
- **Backup Designated Router**: IP address of the router, which is the Backup designated router for the network as viewed by the sender.

- **Neighbor IP Address**: IP address of all the neighbors from which the sender has recently received Hello Messages.

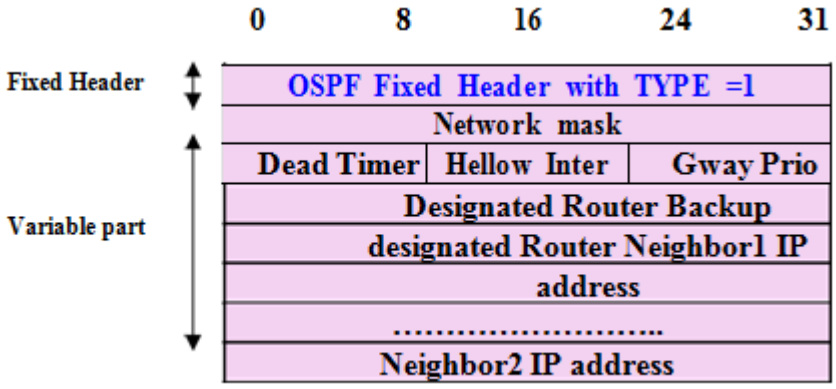


Fig: OSPF Hello Message Format

Database Description message Format: These messages are exchanged by routers to initialize their network topology database. In this exchange one router serves as a master, while other as a slave. The slave acknowledges each database description message with a response. This message is further divided into several messages using **I** and **M** bits. The functions of different fields, as shown in Figure, are summarized below:

- **Fixed Header**: as discussed in previous section and Figure

- **I, M, S bits**: Bit **I** is set to **1** if additional message follows. Bit **S** indicates whether a message was sent by a **master (1)** or a **slave (0).**

- **Database Sequence Number**: this is used to sequence the messages so that the receiver can detect if any of the message is missing. Initial message contains a random sequence number **R**; subsequent messages contain sequential integers starting from **R**.

- **Link Type**: describes one link in network topology; it is repeated for each link. Different possible values for Link Type is as follows:

Link Type	Meaning
1	Router Link
2	Network Link
3	Summary Link (IP Network)
4	Summary Link (Link to Border Router)
5	External Link (Link to another site)

- **Link ID:** gives an identification of the Link, generally an IP address.
- **Advertising Router**: specifics the router which is advertising this link.
- **Link sequence Number**: integer to ensure that messages are not mixed or received out of order.
- **Link Checksum**: Checksum to ensure that the information has not been corrupted.

Link Age: Helps order messages, gives the time (in seconds) since link was established.

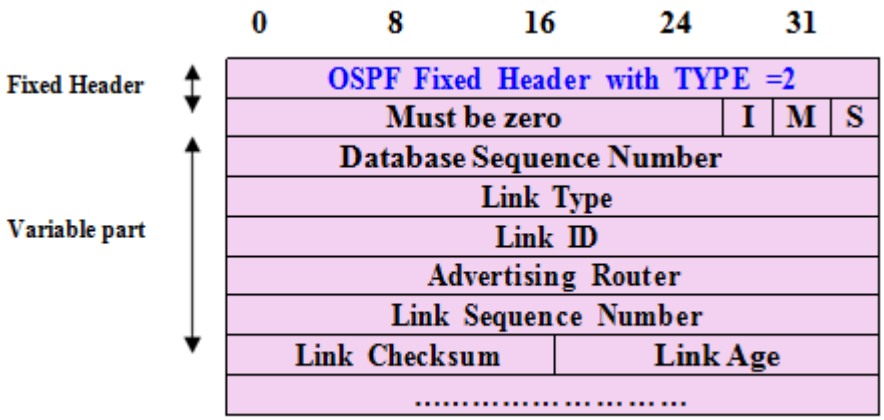


Fig: OSPF Database Description Message Format

Link Status Request Message: After exchanging Database Description message, router may discover that some part of its database is outdated. Link Status message is used to request the neighbor to supply the updated information. The message lists specific links, as shown in Figure 7.3.11. The neighbor responds with the most current information it has about those links. The three fields as shown are repeated for each link, about which status is requested. More than one request message is required if list is long. All the fields have usual meaning as discussed in previous message format.

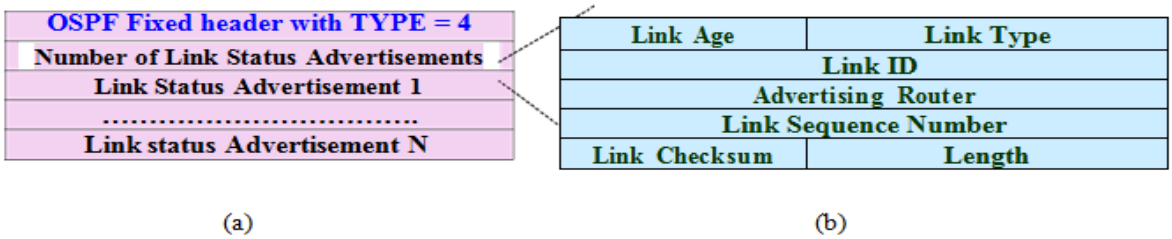


Fig:(a) Link Status Update Message, (b) Format of each Link Advertisement

Routing for Mobile Hosts:

Hosts that never move are said to be stationary. They are connected to the network by copper wires or fiber optics. In contrast, we can distinguish two other kinds of hosts. Migratory hosts are basically stationary hosts who move from one fixed site to another from time to time but use the network only when they are physically connected to it. Roaming hosts actually compute on the run and want to maintain their connections as they move around. We will use the term **mobile hosts** to mean either of the latter two categories, that is, all hosts that areaway from home and still want to be connected. All hosts are assumed to have a permanent **home location** that never changes. Hosts also have a permanent home address that can be used to determine their home locations, analogous to the way the telephone number 1-212-5551212 indicates the United States(country code 1) and Manhattan (212). The routing goal in systems with mobile hosts is to make it possible to send packets to mobile hosts using their home addresses and have the packets efficiently reach them wherever they may be.

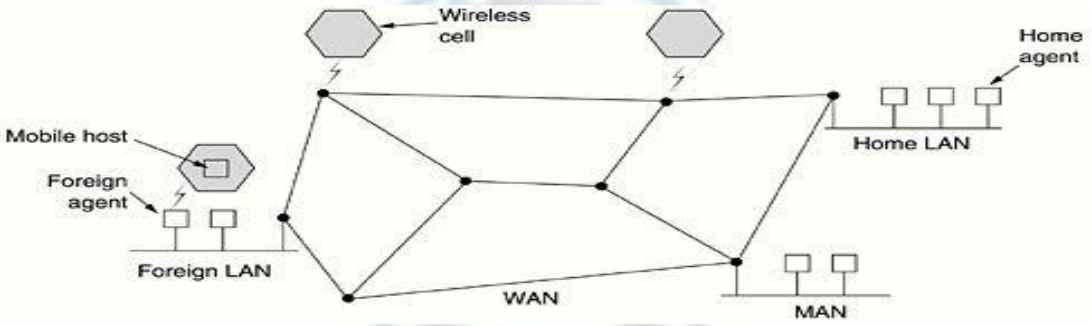


Fig: A WAN to which LANs, MANs, and wireless cells are attached.

Registration Procedure:

1. Periodically, each foreign agent broadcasts a packet announcing its existence and address. A newly-arrived mobile host may wait for one of these messages, but if none arrives quickly enough, the mobile host can broadcast a packet saying: Are there any foreign agents around?
2. The mobile host registers with the foreign agent, giving its home address, current data link layer address, and some security information
3. The foreign agent contacts the mobile host's home agent and says: One of your hosts is over here. The message from the foreign agent to the home agent contains the foreign agent's network address. It also includes the security information to convince the home agent that the mobile host is really there.
4. The home agent examines the security information, which contains a timestamp, to prove that it was generated within the past few seconds. If it is happy, it tells the foreign agent to proceed.
5. When the foreign agent gets the acknowledgement from the home agent, it makes an entry in its tables and informs the mobile host that it is now registered.
