# UNIT-III

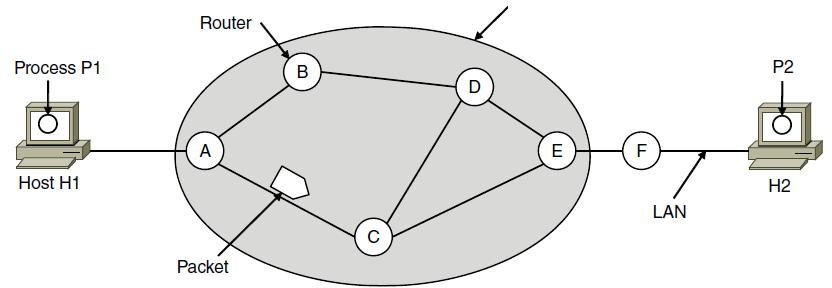
# UNIT –3- NETWORK LAYER

Virtual circuit and Datagram subnets-Routing algorithm shortest path routing, Flooding, Hierarchical routing, Broad cast, Multi cast, distance vector routing. OSPF. IPV4

Network Layer Design Issues

1. Store-and-forward packet switching
2. Services provided to transport layer
3. Implementation of connectionless service
4. Implementation of connection-oriented service
5. Comparison of virtual-circuit and datagram networks

## Store-and-forward packet switching



A host with a packet to send transmits it to the nearest router, either on its own LAN or over a point-to-point link to the ISP. The packet is stored there until it has fully arrived and the link has finished its processing by verifying the checksum. Then it is forwarded to the next router along the path until it reaches the destination host, where it is delivered. This mechanism is store-and-forward packet switching.

## Services provided to transport layer

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

* 1. Services independent of router technology.
  2. Transport layer shielded from number, type, topology of routers.
  3. Network addresses available to transport layer use uniform numbering plan

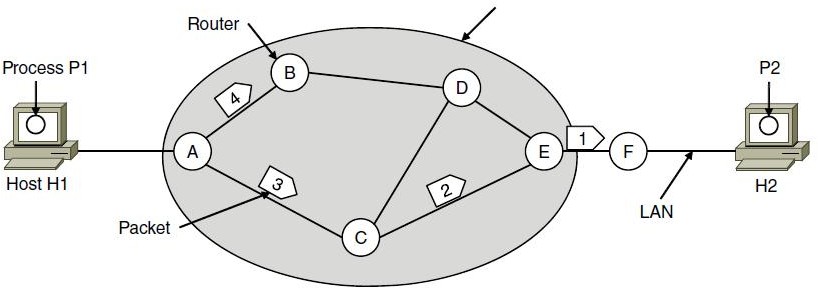
– even across LANs and WANs

## Implementation of connectionless service

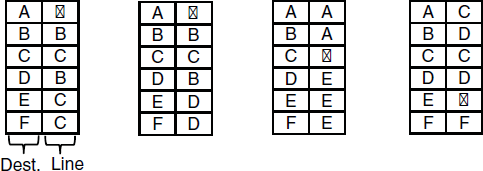
If connectionless service is offered, packets are injected into the network individually and routed independently of each other. No advance setup is needed. In this context, the packets

are frequently called **datagrams** (in analogy with telegrams) and the network is called a

## datagram network.



A’s table (initially) A’s table (later) C’s Table E’s Table



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.*

Every router has an internal table telling it where to send packets for each of the possible destinations. Each table entry is a pair(destination and the outgoing line). Only directly connected lines can be used.

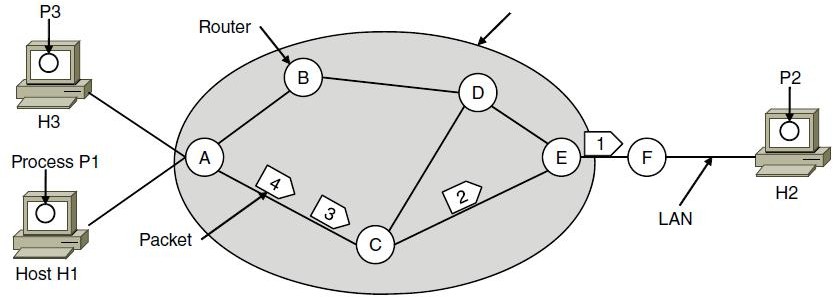
*A*’s initial routing table is shown in the figure under the label ‘‘initially.’’

At *A*, packets 1, 2, and 3 are stored briefly, having arrived on the incoming link. 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*.

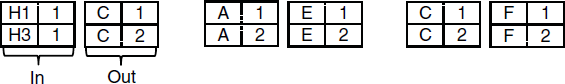
However, something different happens to packet 4. When it gets to *A* it is sent to router *B*, even though it is also destined for *F*. For some reason (traffic jam along ACE path), *A* decided to send packet 4 via a different route than that of the first three packets. Router A updated its routing table, as shown under the label ‘‘later.’’

The algorithm that manages the tables and makes the routing decisions is called the **routing algorithm**.

## Implementation of connection-oriented service



A’s table C’s Table E’s Table



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**), and the network is called a **virtual-circuit network**

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 shown in Figure. Here, 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** (**Multi Protocol Label Switching**).

## Comparison of virtual-circuit and datagram networks

# Virtual Circuit Networks

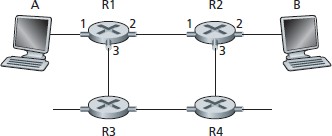
While the internet is a datagram network, many alternative network architectures – including those of ATM (Asynchronous Transfer Mode) and frame relay – are virtual circuit networks and, therefore, use connections at the network layer. These network layer connections are called **virtual circuits (VCs)**. Let’s now consider how a VC service can be implemented in a computer network.

A VC consists of :

* + 1. a path (that is , a series of links and routers) between the source and destination hosts,
    2. VC numbers, one number for each link along the path, and
    3. entries in the forwarding table n each router along the path.

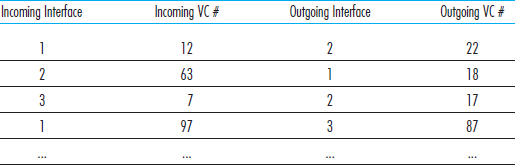
A packet belonging to a virtual circuit will carry a VC number in its header. Because a virtual circuit may have a different VC number on each link, each intervening router must replace the VC number of each traversing packet with a new VC number. The new VC number is obtained from the forwarding table.

To illustrate the concept, consider the network shown in the figure below:



The numbers next to links of R1 in the above figure are the link interface numbers. Suppose now that Host A requests that the network establish a VC between itself and Host B. Suppose also that the network chooses the path A-R1-R2-B and assigns VC numbers 12, 22, and 32 to the three links in this path for this virtual circuit. In this case, when a packet in this VC leaves Host A, the value in the VC number filed in the packet header is 12; when it leaves R1, the value is 22; and when it leaves R2, the value is 32.

How does the router determine the replacement VC number for a packet traversing the router? For a VC network, each router’s forwarding table includes VC number translation; for example the forwarding table in R1 might look something like the table below:



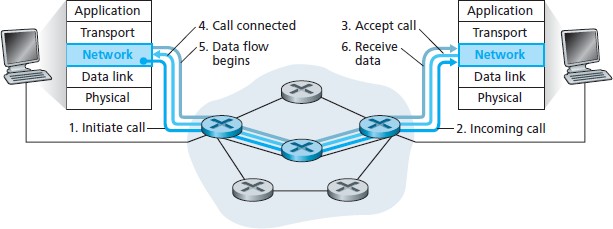
Whenever a new VC is established across a router, an entry is added to the forwarding table. Similarly, whenever a VC terminates, the appropriate entries in each table along its path are removed.

You might be wondering why a packet doesn’t just keep the same VC number on each of the links along its route. The answer is twofold. First, replacing the number from the link reduces the length of the VC field in the packet header. Second, and more importantly, VC setup is considerably simplified by permitting a different VC number at each link along the path of the VC. Specifically, with multiple VC numbers, each link in the path can choose a VC number independently of the VC numbers chosen at other links along the path. If a common VC number were required for all links along the path, the routers would have to exchange and process a substantial number of messages to agree on a common VC number (e.g. one that is not being used by any other existing VC at these routers) to be used for a connection.

In a VC network, the network’s routers must maintain **connection state information** for the ongoing connections. Specifically, each time a new connection is established across a router, a new connection entry must be added to the router’s forwarding table; and each time a connection is released an entry must be removed from the table. Note that even if there is no VC number translation, it is still necessary to maintain connection state information that associates VC numbers with output interface numbers. The issue of whether or not a router maintains connection state information for each ongoing connection is a crucial one.

There are three identifiable phases in a virtual circuit:

* **VC Setup** : During this setup phase, the sending transport layer contacts the network layer, specifies the receiver’s address, and waits for the network to set up the VC. The network layer determines the path between sender and receiver, that is, the series of links and routers through which all packets of the VC will travel. The network layer also determines the VC number for each link along the path. Finally, the network layer adds an entry in the forwarding table in each router along the path. During VC setup, the network layer may also reserve resources (for example, bandwidth) along the path of the VC.
* **Data Transfer** : As shown in the figure below, once the VC has been established, packets can begin to flow along the VC.
* **VC Teardown** : This is initiated when the sender (or receiver) informs the network layer of its desire to terminate the VC. The network layer will then typically inform the end system on the other side of the network of the call termination and update the forwarding table sin each of the packet routers on the path to indicate that the VC no longer exists.

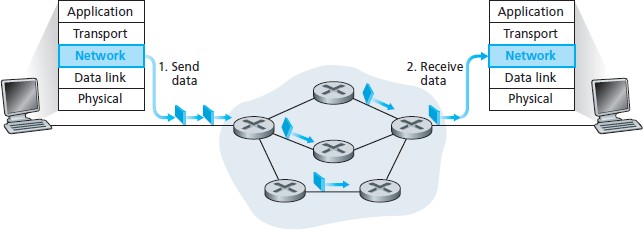


There is subtle but important distinction between VC setup at the network layer and connection setup at the transport layer (for example, the TCP three-way handshake). Connection setup at the transport layer involves only the two end systems. During transport-layer connection setup, the two end systems alone determine the parameters (for example, initial sequence number and flow-control window size) of their transport-layer connection. Although the two end systems are aware of the transport-layer connection, the routers within the network are completely oblivious to it. On the other hand, with a VC network layer, routers along the path between the two end systems are involved in VC setup, and each router is fully aware of all the VCs passing through it.

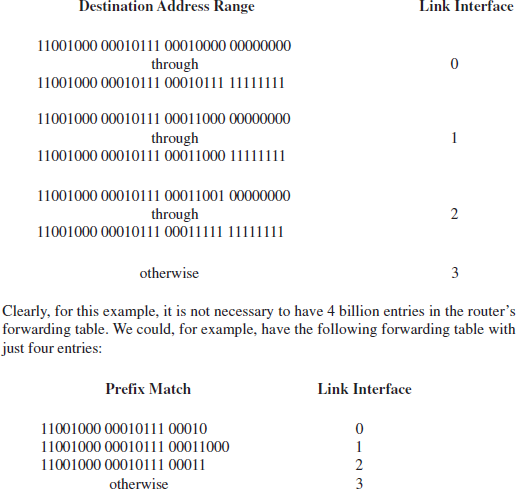
The message that the end systems send into the network to initiate or terminate a VC, and the message passed between the routers to set up the VC (that is, to modify connection state in router tables ) are known as **signalling messages**, and the protocols used to exchange these message are often referred to as **signalling protocols**. VC setup is shown in the figure above

# Datagram Networks

In a datagram network, each time an end system wants to send a packet, it stamps the packet with the address of the destination end system and then pops the packet into the network. As shown in the figure below , there is no VC setup and routers do not maintain any VC state information (because there are no VCs).

As a packet is transmitted from source to destination, it passes through a series of routers. Each of these routers uses the packet’s destination address to forward the packet. Specifically, each router has a forwarding table that maps destination address to link interfaces; when a packet arrives at the router, the router uses the packet’s destination address to look up the appropriate output link interface in the forwarding table. The router then intentionally forwards the packet to that output link interface.

To get some further insight into the lookup operation, let’s look at a specific example. Suppose that all destination addresses are 32 bits (which just happens to be the length of the destination address in an IP datagram). A brute-force implementation of the forwarding table would have one entry for every possible destination address. Since there are more than 4 billion possible addresses, this option is totally out of the question.

Now, let’s further suppose that our router has four links, numbered 0 through 3, and the packets are to be forwarded to the link interfaces as follows :

With this style of forwarding table, the router matches a prefix of the packet’s destination address with the entries in the table; if there’s a match, the router forwards the packet to a link associated with the match.

For example , suppose the packet’s destination address is 11001000 00010110 10100001; because the 21- bit prefix of this address matches the first entry in the table, the router forwards the packet to link interface 0. If a prefix doesn’t match any of the first three entries, then the router forwards the packet to interface 3.

Although this sounds simple enough, there’s an important subtlety here. You may have noticed that it is possible for a destination address to match more than one entry. For example, the first 24 bits of the address 11001000 00010111 00011000 10101010 match the second entry in the table, and the first 21 bits of the address match the third entry in the table. When there are multiple matches, the router uses the **longest prefix matching rule**; that is, it finds the longest matching entry in the table and forwards the packet to the link interface associated with the longest prefix match.

Although routers in datagram networks maintain no connection state information, they nevertheless maintain forwarding state information in their forwarding tables. However, the time scale at which this forwarding information changes is relatively slow. Indeed, in a datagram network the forwarding tables are modified by routing algorithms, which typically update a forwarding table every one-to-five minutes or so. In a VC network, a forwarding table in a router is modified whenever a new connection is set up through the router or whenever an existing connection through the router is torn down. This could easily happen at microsecond timescale in a backbone, tier-1 router.

Because forwarding tables in datagram networks can be modified at any time, a series of packets sent from one end system to another may follow different paths through the network and may arrive out of order.

**Routing Algorithms**

The main function of NL (Network Layer) is routing packets from the source machine to the destination machine.

There are two processes inside router:

1. One of them handles each packet as it arrives, looking up the outgoing line to use for it in the routing table. This process is forwarding.
2. The other process is responsible for filling in and updating the routing tables. That is where the routing algorithm comes into play. This process is routing.

Regardless of whether routes are chosen independently for each packet or only when new connections are established, certain properties are desirable in a routing algorithm **correctness, simplicity, robustness, stability, fairness, optimality**

Routing algorithms can be grouped into two major classes:

* 1. nonadaptive (Static Routing)
  2. adaptive. (Dynamic Routing)

Nonadaptive algorithm do not base their routing decisions on measurements or estimates of the current traffic and topology. Instead, the choice of the route to use to get from I to J is computed in advance, off line, and downloaded to the routers when the network is booted. This procedure is sometimes called static routing.

Adaptive algorithm, in contrast, change their routing decisions to reflect changes in the topology, and usually the traffic as well.

Adaptive algorithms differ in

1. Where they get their information (e.g., locally, from adjacent routers, or from all routers),
2. When they change the routes (e.g., every ∆T sec, when the load changes or when the

topology changes), and

1. What metric is used for optimization (e.g., distance, number of hops, or estimated transit time).

This procedure is called dynamic routing

Different Routing Algorithms

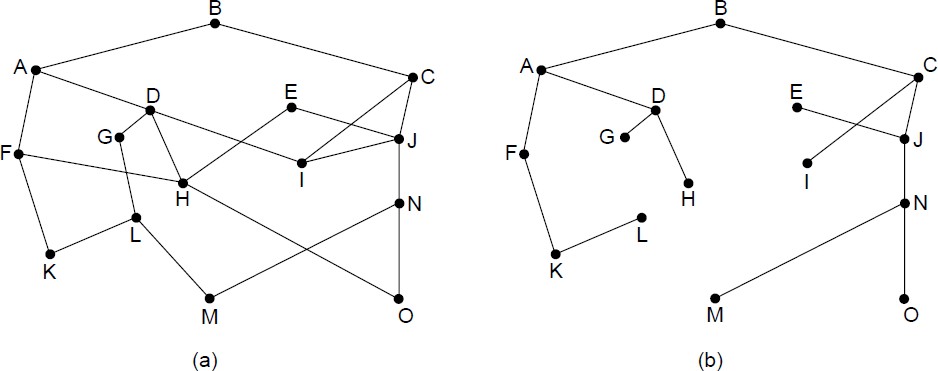
* + Optimality principle
  + Shortest path algorithm
  + Flooding
  + Distance vector routing
  + Link state routing
  + Hierarchical Routing

## The Optimality Principle

One can make a general statement about optimal routes without regard to network topology or traffic. This statement is known as the optimality principle.

It states that if router J is on the optimal path from router I to router K, then the optimal path from J to K also falls along the same

As a direct consequence of the optimality principle, we can see that the 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**. The goal of all routing algorithms is to discover and use the sink trees for all routers



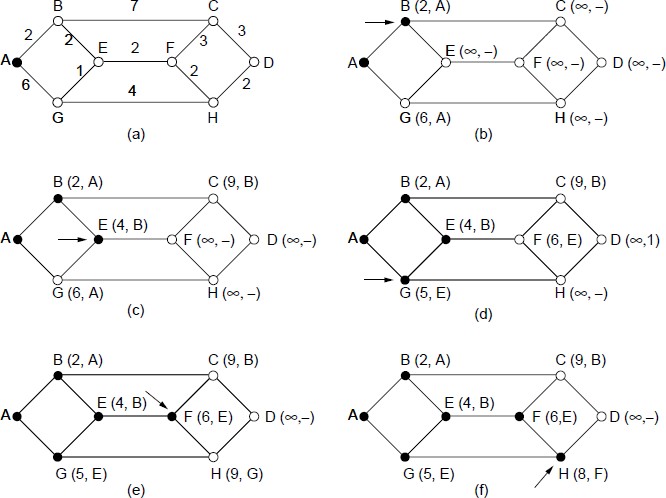
(a) A network. (b) A sink tree for router *B.*

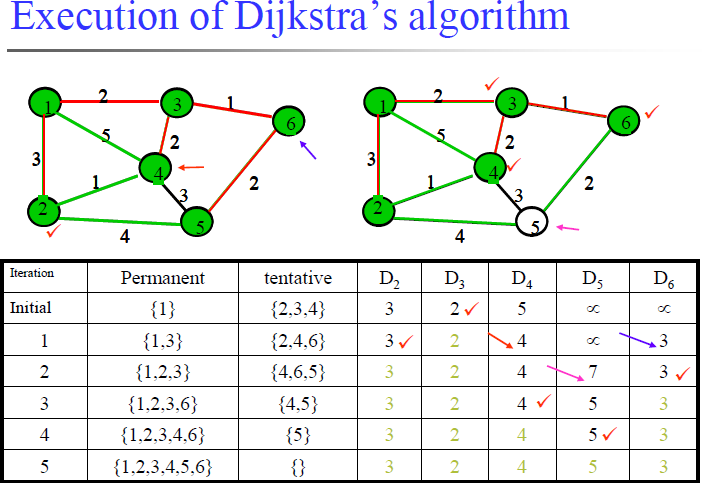
## Shortest Path Routing (Dijkstra’s)

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 or link.

To choose a route between a given pair of routers, the algorithm just finds the shortest path between them on the graph

1. Start with the local node (router) as the root of the tree. Assign a cost of 0 to this node and make it the first permanent node.
2. Examine each neighbor of the node that was the last permanent node.
3. Assign a cumulative cost to each node and make it tentative
4. Among the list of tentative nodes
   1. Find the node with the smallest cost and make it Permanent
   2. If a node can be reached from more than one route then select the route with the shortest cumulative cost.
5. Repeat steps 2 to 4 until every node becomes permanent





## Flooding

* Another static algorithm is 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.
* A variation of flooding that is slightly more practical is **selective flooding**. In this algorithm the routers do not send every incoming packet out on every line, only on those lines that are going approximately in the right direction.
* Flooding is not practical in most applications.