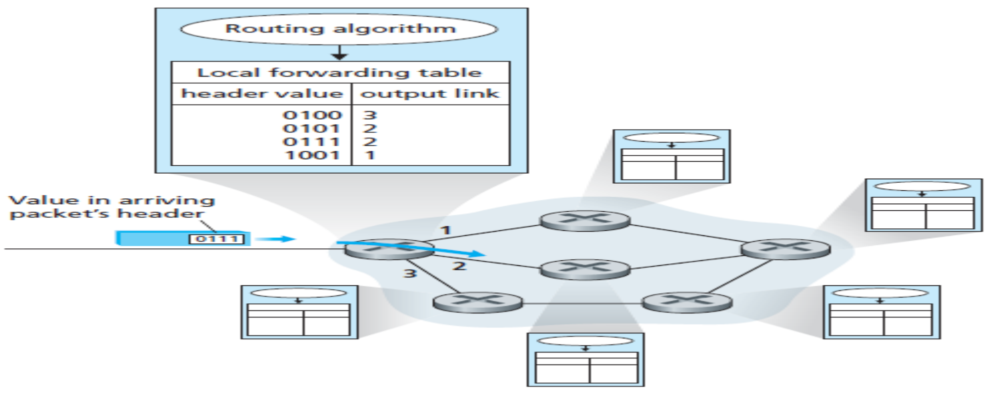
UNIT-III

4.1 Forwarding and Routing in Network Layer: The role of the network layer is to move packets from a sending host to a receiving host. To do so, two important network-layer functions can be identified:

- **Forwarding:** When a packet arrives at a router's input link, the router must move the packet to the appropriate output link. For example, a packet arriving from Host H1 to Router R1 must be forwarded to the next router on a path to H2. Forwarding refers to the router-local action of transferring packet from an input link interface to the appropriate output link interface
- **Routing:** The network layer must determine the route or path taken by packets as they flow from a sender to a receiver. The algorithms that calculate these paths are referred to as routing algorithms. A routing algorithm would determine, for example, the path along which packets flow from H1 to H2. Routing refers to the network-wide process that determines the end-to-end paths that packets take from source to destination.

Every router has a forwarding table. A router forward a packet by examining the value of a field in the arriving packet's header, and then using this header value to index into the router's forwarding table. The value stored in the forwarding table entry for that header indicates the router's outgoing link interface to which the packet is to be forwarded.



In the above figure, a packet with a header field value of 0111 arrives to a router. The router indexes into its forwarding table and determines that the output link interface for this packet is interface 2. The router then internally forwards the packet to interface 2. As shown in the figure above, the routing algorithm determines the values that are inserted into the routers' forwarding tables. The routing algorithm may be centralized (e.g. with an algorithm executing on a central site and downloading routing information to each of the routers) or decentralized (i.e. with a piece of the distributed routing algorithm running in each router). In either case, a router receives routing protocol messages, which are used to configure its forwarding table.

Connection Setup in Network Layer: some network-layer architectures – for example, ATM, frame relay, and MPLS require the routers along the chosen path from source to destination to handshake with each other in order to set up state before network-layer data packets within a given source-to-destination connection can begin to flow. In the network layer, this process is referred to as connection setup.

4.2 Virtual Circuit and Datagram Networks:

Virtual circuit and datagram networks are computer networks that provide connection oriented and connectionless services respectively. A transport layer can offer applications connectionless service or connection-oriented service between two processes. For example, the internet’s transport layer provides each application a choice between two services : UDP, a connectionless service; or TCP, a connection-oriented service between two hosts. Network-layer connection and connectionless service in many ways parallel transport layer connection-oriented and connectionless services.

Although the network layer connection and connectionless services have some parallel with transport-layer connection-oriented and connectionless services, there are crucial differences:

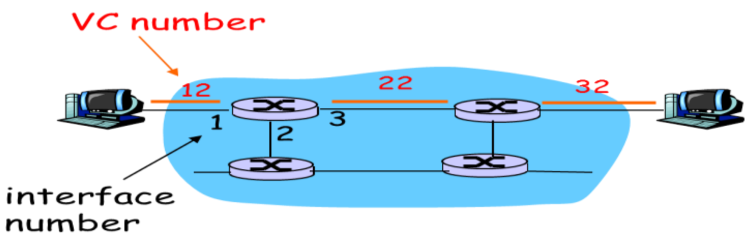
- In the network layer, these services are host-to-host services provided by the network layer for the transport layer. In the transport layer these services are process-to-process services provided by the transport layer for the application layer.
- In all major computer network architectures to date (Internet, ATM, frame relay and so on), the network layer provides either host-to-host connectionless service or host-to-host connection service, but not both. Computer networks that provide only a connection service at the network layer are called virtual circuit (VC) networks ; computer networks that provide only a connectionless service at the network layer are called datagram networks
- The implementations of connection oriented service in the transport layer and the connection service in the network layer are fundamentally different: the transport-layer connection the network-layer connection service is implemented in the routers in the network core as well as in the end systems.-oriented service is implemented at the edge of the network in the end systems.

4.2.1 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). A VC consists of :

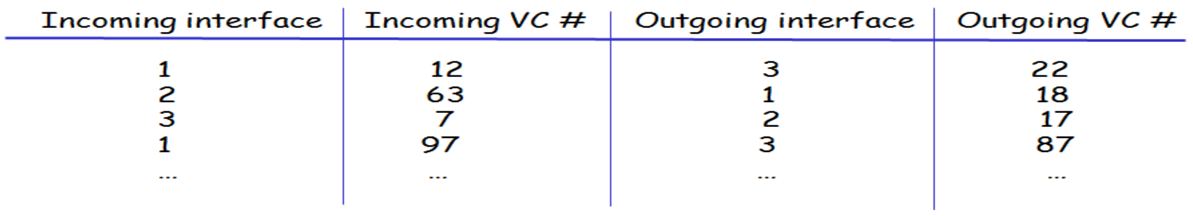
- a path (that is , a series of links and routers) between the source and destination hosts,
- VC numbers, one number for each link along the path, and
- entries in the forwarding table n each router along the path.

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. The numbers next to links of R1 in the above figure are the link interface numbers. For example 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.



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:



Incoming interface	Incoming VC #	Outgoing interface	Outgoing VC #
1	12	3	22
2	63	1	18
3	7	2	17
1	97	3	87
...	...	...	...

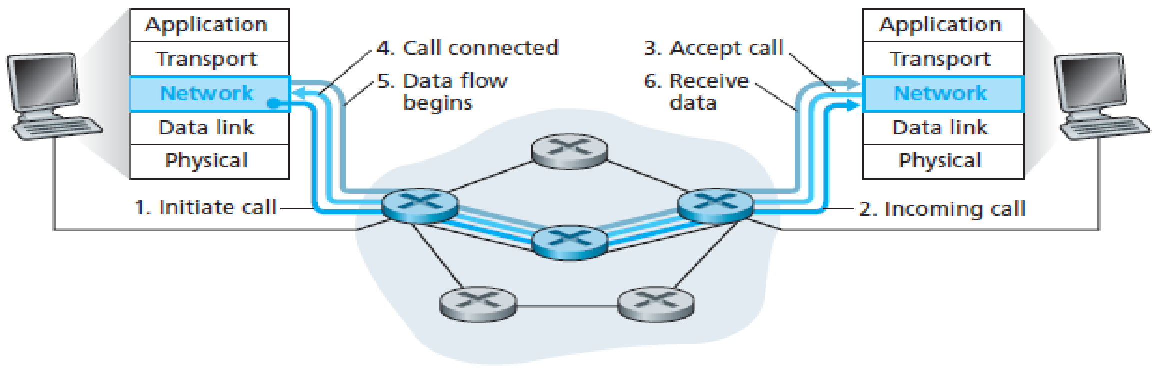
Whenever a new VC is established across a router, an entry is added to the forwarding table. Similarly, whenever a VC terminates, the appropriate entries in each table along its path are removed. In a VC network, the network's routers must maintain connection state information for the on going connections. 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.

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.

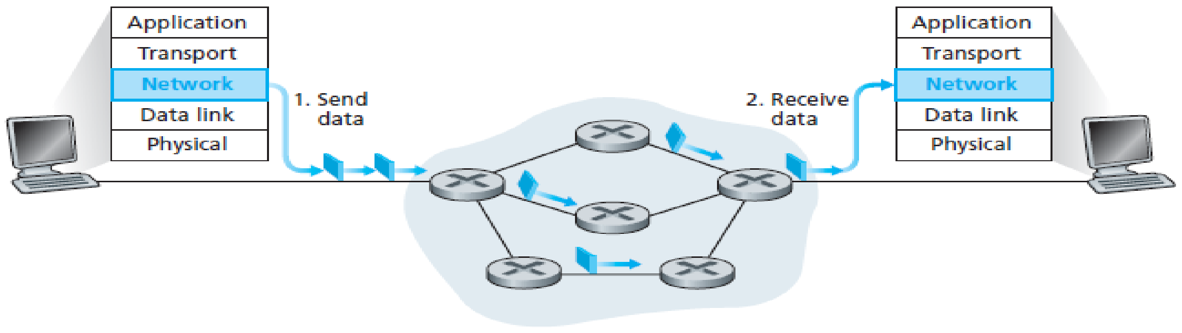
Data Transfer: As shown in the figure below, once the VC has been established, packets can begin to flow along the VC.

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.



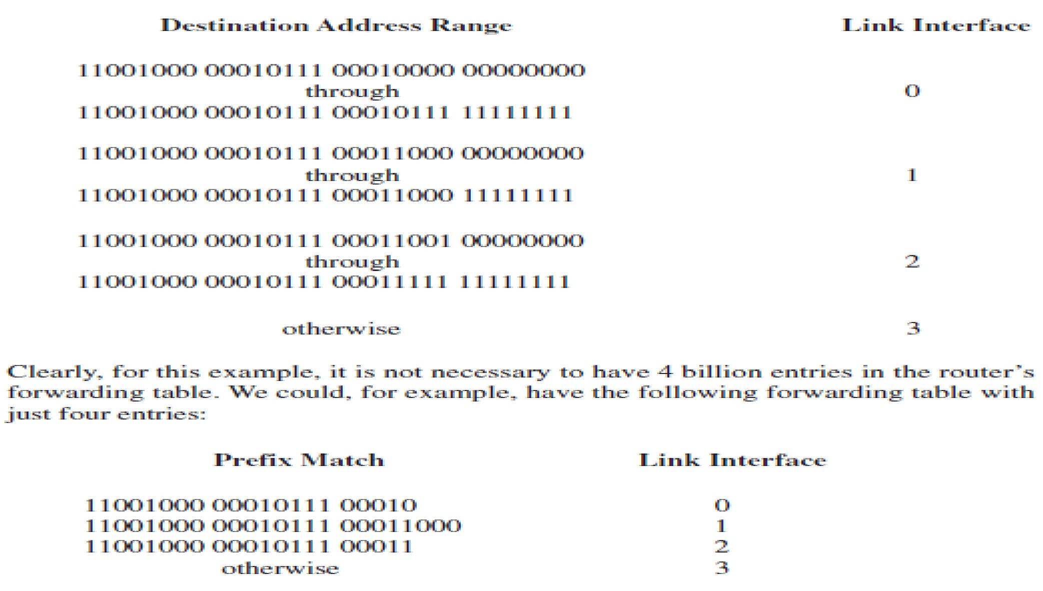
The message that the end systems send into the network to initiate or terminate a VC, and the message passed between the routers to set up the VC (that is, to modify connection state in router tables) are known as signalling messages, and the protocols used to exchange these message are often referred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

4.2.2 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.

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 :



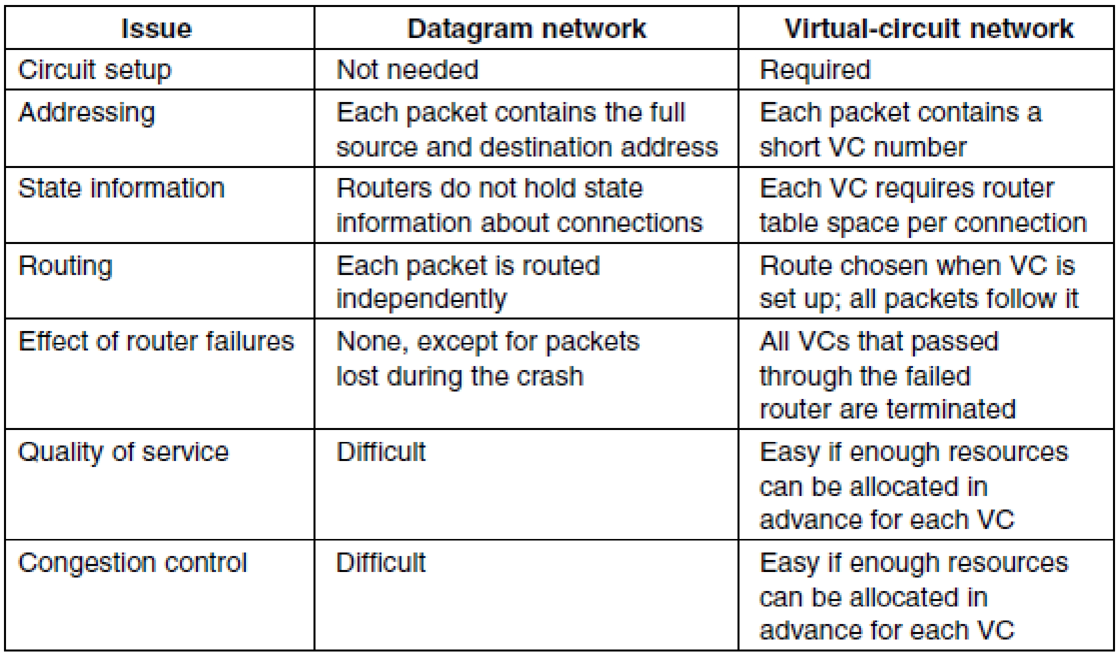
Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
otherwise	3

Clearly, for this example, it is not necessary to have 4 billion entries in the router's forwarding table. We could, for example, have the following forwarding table with just four entries:

Prefix Match	Link Interface
11001000 00010111 00010	0
11001000 00010111 00011000	1
11001000 00010111 00011	2
otherwise	3

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.

Comparison of Virtual Circuit and Datagram:



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

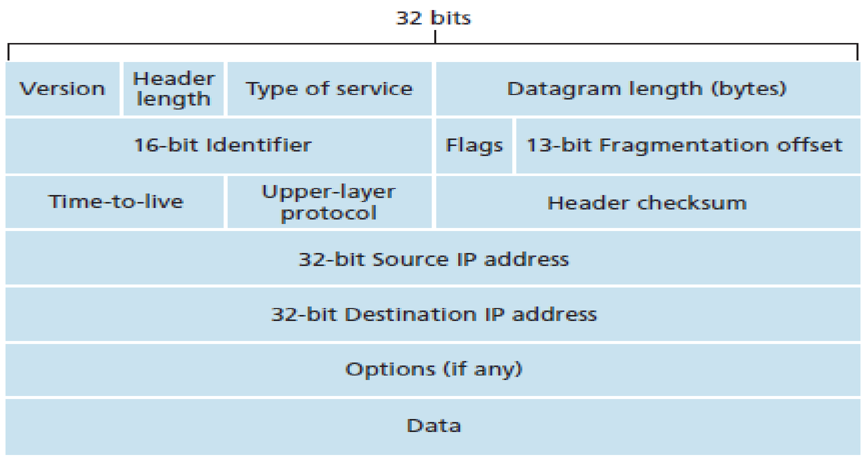
4.4 The Internet Protocol:

Forwarding and Addressing in the Internet: Forwarding and addressing in the internet are important components of the Internet Protocol (IP). As shown in the figure below, the internet's network layer has three major components.

The first component is the IP

- The second major component is the routing component, which determines the path a datagram follows from source to destination
- The final component of the network layer is a facility to report errors in datagrams and respond to requests for certain network-layer information.

4.4.1 IPv4 Datagram Format: A network-layer packet is referred to as a datagram. The IPv4 datagram format is shown in the figure below.



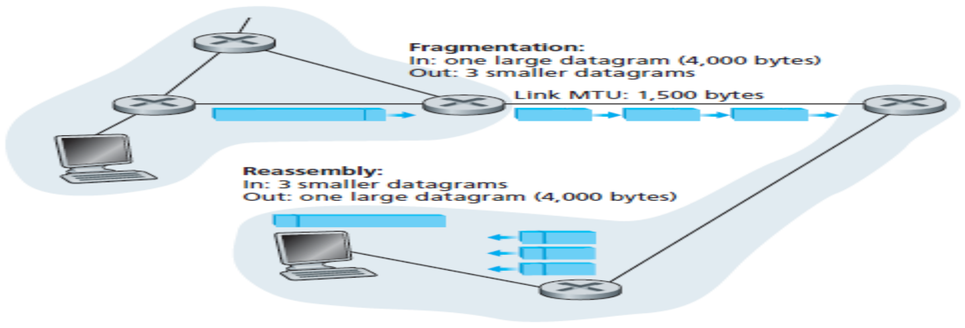
- **Version Number**: These 4 bits specify the IP protocol version of the datagram. By looking at the version number, the router can determine how to interpret the remainder of the IP datagram. Different versions of IP use different datagram formats. The datagram format for the current version of IP, IPv4 is shown in the figure above. The datagram format for the new version of IP is (IPv6)
- **Header Length**: Because an IPv4 datagram can contain a variable number of options (which are included in the IPv4 datagram header), these 4 bits are needed to determine where in the IP datagram the data actually begins. Most IP datagrams do not contain options, so the typical IP datagram has a 20-byte header.
- **Type of Service**: The type of service (TOS) bits were included in the IPv4 header to allow different types of IP datagrams (for example, datagrams particularly requiring low delay, high throughput, or reliability) to be distinguished from each other. For example, it might be useful to distinguish real-time datagrams (such as those used by an IP telephony application) from non-real-time traffic (for example, FTP). The specific level of service to be provided is a policy issue determined by the router's administrator.
- **Datagram Length**: This is the total length of the IP datagram (header plus data), measured in bytes. Since this filed is 16 bits long, the theoretical maximum size of the IP datagram is 65,535 bytes. However, datagrams are rarely larger than 1,500 bytes.

- **Identifier, Flags, Fragmentation Offset**: These three fields have to do with so-called IP fragmentation, a topic we will consider in depth shortly. Interestingly, the new version of IP, IPv6, does not allow fragmentation at routers.
- **Time-to-live**: The time-to-live (TTL) field is included to ensure that datagrams do not circulate forever (due to, for example, a long-lived routing loop) in the network. This field is decremented by one each time the datagram is processed by a router. If the TTL field reaches 0, the datagram must be dropped.
- **Protocol:** This field is used only when an IP datagram reaches its final destination. The value of this field indicates the specific transport-layer protocol to which the data portion of this IP datagram should be passed. For example, a value of 6 indicates that the data portion is passed to TCP, while a value of 17 indicates that the data is passed to UDP.
- **Header Checksum**: The header checksum aids a router in detecting bit errors in a received IP datagram. The header checksum is computed by treating each 2 bytes in the header as a number and summing these numbers using 1s and complement arithmetic.
- **Source and Destination IP Addresses**: When a source creates a datagram, it inserts its IP address into the source IP address field and inserts the address of the ultimate destination into the destination IP address field. Often the source host determines the destination address via a DNS lookup.
- **Options:** The options fields allow an IP header to be extended. Header options were meant to be used rarely – hence the decision to save overhead by not including the information in options fields in every datagram header.
- **Data (Payload):** Finally, we come to the last and most important field. In most circumstances, the data field of the IP datagram contains the transport-layer segment (TCP or UDP) to be delivered to the destination. However, the data field can carry other types of data, such as ICMP messages.

IP Datagram Fragmentation with Example: Not all link-layer protocols can carry network-layer packets of the same size. Some protocols can carry big datagrams, whereas other protocols can carry only little packets. The maximum amount of data that a link-layer frame can carry is called the maximum transmission unit (MTU). Because each IP datagram is encapsulated within the link-layer frame for transport from one router to the next router, the MTU of the link-layer protocol places a hard limit on the length of an IP datagram. The solution is to fragment the data in the IP datagram into two or smaller IP datagrams encapsulate each of these smaller IP datagram in a separate link-layer frame; and send these frames over the outgoing link. Each of these smaller datagrams is referred to as a fragment .Fragments needs to be reassembled before they reach the transport layer at the destination. To allow the destination host to perform these reassembly tasks, the designers of IP (version 4) put identification flag, and fragmentation offset fields in the IP datagram header. When a datagram is created, the sending host stamps the datagram with an identification number as well as source and destination address. When a router needs to fragment a datagram, each resulting datagram (that is, fragment) is stamped with the source address, destination address, and identification number of the original datagram. When the destination receives a series of datagrams from the same sending host, it can examine the identification numbers of the datagrams to determine which of the datagrams actually fragments of the same larger datagram are.

Because IP is an unreliable service, one or more of the fragments may never arrive at the destination. For this reason, in order for the destination host to be absolutely sure it has received the last fragment of the original datagram , the last fragment has a flag bit set to 0, whereas all the other fragments have flag bit set to 1. Also, in order for the destination host to determine whether a fragment is missing (and also to be able to reassemble the fragments in their proper order), the offset field is used to specify where the fragment fits within the original IP datagram.

Example:



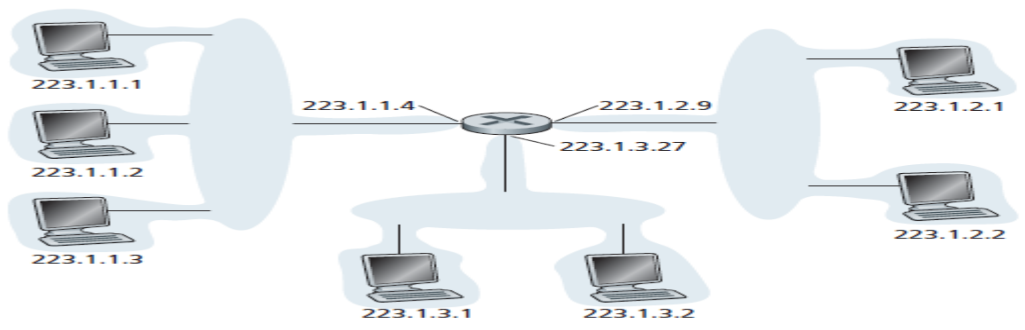
A datagram of 4,000 bytes (20 bytes of IP header plus 3,980 bytes of IP payload) arrives at a router and must be forwarded to a link with an MTU of 1,500 bytes. This implies that the 3,980 data bytes in the original datagram must be allocated to three separate fragments (each of which is also an IP datagram).Suppose that the original datagram is stamped with an identification number of 777.The characteristics of three fragments are shown in the table below.

Fragment	Bytes	ID	Offset	Flag
1st fragment	1,480 bytes in the data field of the IP datagram	identification = 777	offset = 0 (meaning the data should be inserted beginning at byte 0)	flag = 1 (meaning there is more)
2nd fragment	1,480 bytes of data	identification = 777	offset = 185 (meaning the data should be inserted beginning at byte 1,480. Note that 185 · 8 = 1,480)	flag = 1 (meaning there is more)
3rd fragment	1,020 bytes (= 3,980–1,480–1,480) of data	identification = 777	offset = 370 (meaning the data should be inserted beginning at byte 2,960. Note that 370 · 8 = 2,960)	flag = 0 (meaning this is the last fragment)

The values in the above table reflect the requirement that the amount of original payload data in all but the last fragment be a multiple of 8 bytes, and that the offset value be specified in units of 8-byte chunks. At the destination, the payload of the datagram is passed to the transport layer only after the IP layer has fully reconstructed the IP datagram. If one or more of the fragments does not arrive at the destination, the incomplete datagram is discarded and not passed to the transport layer.

4.4.2 IP Addressing and Subnetting: A host typically has only a single link into the network; when the IP in the host wants to send a datagram, it does so over this link. The boundary between the host and the physical link is called an interface. A router thus has multiple interfaces, one for each of its links. The boundary between the router and any one of its links is also called an interface. Because every host and router is capable of sending and receiving IP datagrams, IP requires each host and router interface to have its own IP address. **Thus, an IP address is technically associated with an interface, rather than with the host or router containing that interface.** Each IP address is 32 bits long, these addresses are typically written in so-called dotted-decimal notation, in which each byte of the address is written in its decimal form and is separated by a period (dot) from other bytes in the address. Thus, the address 193.32.216.9 in binary notation is: 11000001 00100000 11011000 00001001. Each interface on every host and router in the global internet must have an IP address that is globally unique.

The figure below provides an example of IP addressing and interfaces



The three hosts in the upper-left portion of the above figure, and the router interface to which they are connected, all have an IP address of the form 223.1.1.xxx. That is, they all have the same leftmost 24 bits in their IP address**.** This network interconnecting three host interfaces and one router interface forms a subnet. IP addressing assigns an address to this subnet: 223.1.1.0/24, where the /24 notation, sometimes known as a subnet mask, indicates that the leftmost 24 bits of the 32 bit quantity define the subnet address.

Subnet (subnetwork): A subnet is a logical partition of an IP network into multiple, smaller network segments. It is typically used to subdivide large networks into smaller, more efficient subnet works**.**

CIDR: The internet's address assignment strategy is known as **Classless Inter domain Routing**. CIDR generalizes the notion of subnet addressing. CIDR generalizes the notion of subnet addressing. As with subnet addressing, the 32-bit IP address is divided into two parts and again has the dotted-decimal form a.d.c.d/x, where x indicates the number of bits in the first part of the address. The x most significant bits of an address of the form a.b.c.d/x constitute the network portion of the IP address, and are often referred to as the prefix (or network prefix) of the address.

Obtaining a Host Address: the Dynamic Host Configuration Protocol: Once an organization has obtained a block of addresses, it can assign individual IP addresses to the host and router interfaces in its organization. A system administrator will typically manually configure the IP addresses into the router .Host addresses can also be configured manually, but more often this task is now done using the Dynamic Host Configuration Protocol (DHCP). DHCP allows a host to obtain (be allocated) an IP address automatically. A network administrator can configure DHCP so that a given host receives the same IP address each time it connects to the network, or a host may be assigned a temporary IP address that will be different each time the host connects to the network. Because of DHCP’s ability to automate the network-related aspects of connecting a host into a network, it is often referred to as a plug-and-play protocol.

DHCP is a client-server protocol. A client is typically a newly arriving host wanting to obtain network configuration information, including an IP address for itself. In the simplest case, each subnet (in the addressing sense of Figure 4.17) will have a DHCP server. If no server is present on the subnet, a DHCP relay agent (typically a router) that knows the address of a DHCP server for that network is needed. For a newly arriving host, the DHCP protocol is a four-step process, as shown in Figure 4.21

The four steps are:

- **DHCP server discovery**. The first task of a newly arriving host is to find a DHCP server with which to interact. This is done using a DHCP discover message, which a client sends within a UDP packet to port 67. The UDP packet is encapsulated in an IP datagram
- **DHCP server offer(s).** A DHCP server receiving a DHCP discover message responds to the client with a DHCP offer message that is broadcast to all nodes on the subnet, again using the IP broadcast address of 255.255.255.255Each server offer message contains the transaction ID of the received discover message, the proposed IP address for the client, the network mask, and an IP address lease time—the amount of time for which the IP address will be valid.
- **DHCP request**. The newly arriving client will choose from among one or more server offers and respond to its selected offer with a DHCP request message, echoing back the configuration parameters.
- **DHCP ACK**. The server responds to the DHCP request message with a DHCP ACK message, confirming the requested parameters.

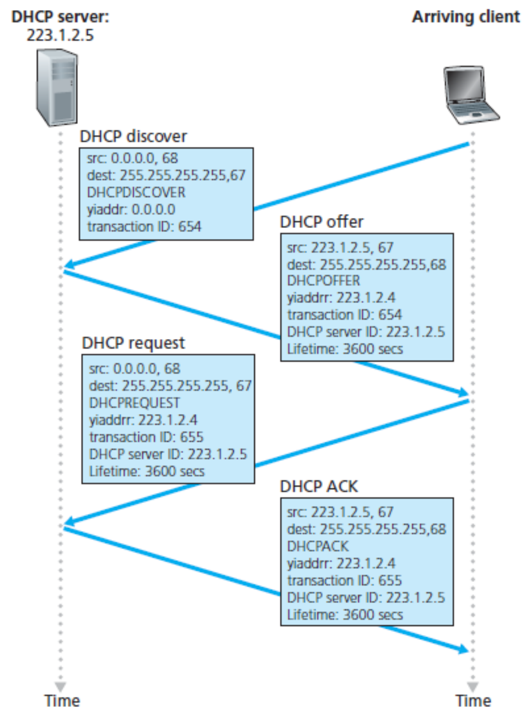


Figure 4.21 ♦ DHCP client-server interaction

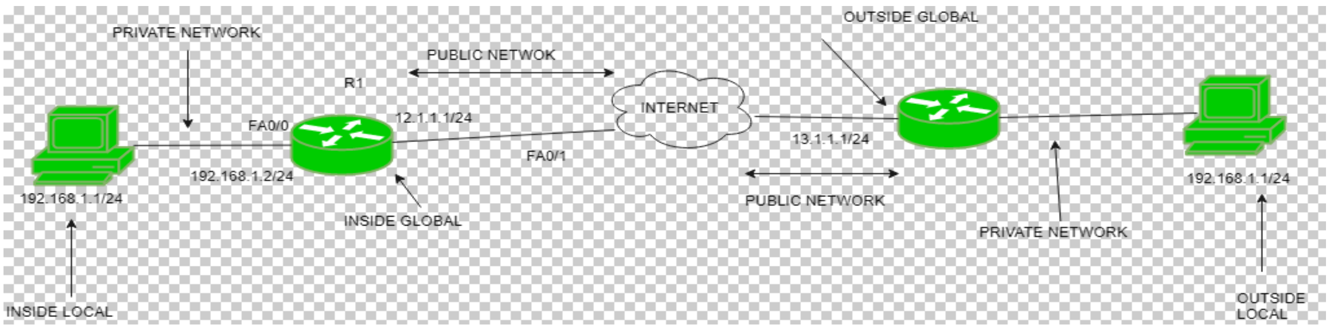
Network Address Translation (NAT): Network Address Translation (NAT) is designed for IP address conservation. It enables private IP networks that use unregistered IP addresses to connect to the Internet. NAT operates on a router, usually connecting two networks together, and translates the private (not globally unique) addresses in the internal network into legal addresses, before packets are forwarded to another network. NAT generally operates on router or firewall.

As part of this capability, NAT can be configured to advertise only one address for the entire network to the outside world. This provides additional security by effectively hiding the entire internal network behind that address. NAT offers the dual functions of security and address conservation and is typically implemented in remote-access environments

Network Address Translation (NAT) working –Generally, the border router is configured for NAT i.e the router which has one interface in local (inside) network and one interface in global (outside) network. When a packet traverse outside the local (inside) network, then NAT converts that local (private) IP address to a global (public) IP address. When a packet enters the local network, the global (public) IP address is converted to local (private) IP address. If NAT run out of addresses, i.e., no address is left in the pool configured then the packets will be dropped and an Internet Control Message Protocol (ICMP) host unreachable packet to the destination is send.

NAT inside and outside addresses –

Inside refers to the addresses which must be translated. Outside refers to the addresses which are not in control of an organisation. These are the network Addresses in which the translation of the addresses will be done



Advantages of NAT –

- NAT conserves legally registered IP addresses.
- It provides privacy as the device IP address, sending and receiving the traffic, will be hidden.
- Eliminates address renumbering when a network evolves.

Disadvantage of NAT –

- Translation results in switching path delays.
- Certain applications will not function while NAT is enabled.
- Complicates tunnelling protocols such as IPsec.

IPv6 Datagram Format

The format of the IPv6 datagram is shown in Figure 4.24. The most important changes introduced in IPv6 are evident in the datagram format:

- **Expanded addressing capabilities**. IPv6 increases the size of the IP address from 32 to 128 bits. In addition to unicast and multicast addresses, IPv6 has introduced a new type of address, called an Anycast address, which allows a datagram to be delivered to any one of a group of hosts
- **A streamlined 40-byte header**. A number of IPv4 fields have been dropped or made optional. The resulting 40-byte fixed-length header allows for faster processing of the IP datagram. A new encoding of options allows for more flexible options processing.
- **Flow labeling and priority**. IPv6 has an elusive definition of a flow labeling of packets, they belonging to particular flows for which the sender requests special handling, such as a nondefault quality of service or real-time service. The IPv6 header also has an 8-bit traffic class field. This field, like the TOS field in IPv4, can be used to give priority to certain datagrams within a flow, or it can be used to give priority to datagrams from certain applications

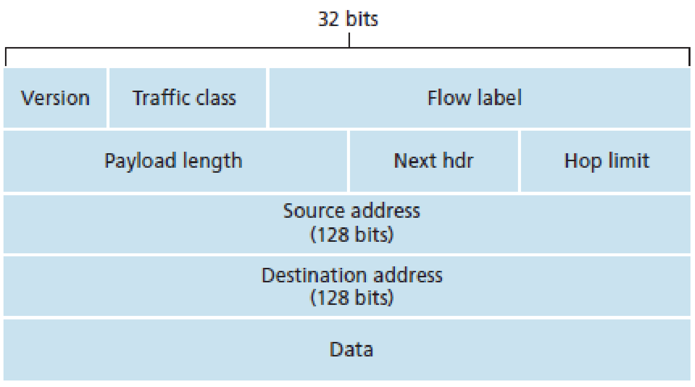


Figure 4.24 ♦ IPv6 datagram format

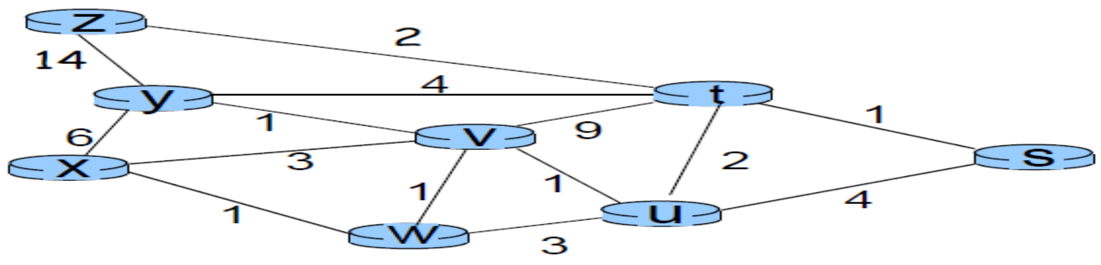
The following fields are defined in IPv6:

- **Version.** This 4-bit field identifies the IP version number. Not surprisingly, IPv6 carries a value of 6 in this field. Note that putting a 4 in this field does not create a valid IPv4 datagram.
- **Traffic class**. This 8-bit field is similar in spirit to the TOS field we saw in IPv4.
- **Flow label**. This 20-bit field is used to identify a flow of datagrams.
- **Payload length**. This 16-bit value is treated as an unsigned integer giving the number of bytes in the IPv6 datagram following the fixed-length, 40-byte datagram header.
- **Next header**. This field identifies the protocol to which the contents (data field) of this datagram will be delivered (for example, to TCP or UDP). The field uses the same values as the protocol field in the IPv4 header.
- **Hop limit**. The contents of this field are decremented by one by each router that forwards the datagram. If the hop limit count reaches zero, the datagram is discarded.
- **Source and destination addresses**. The various formats of the IPv6 128-bit address are described in RFC 4291.
- **Data**. This is the payload portion of the IPv6 datagram. When the datagram reaches its destination, the payload will be removed from the IP datagram and passed on to the protocol specified in the next header field.

The fields appearing in the IPv4 datagram are no longer present in the IPv6 datagram:

- **Fragmentation/Reassembly**. IPv6 does not allow for fragmentation and reassembly at intermediate routers; these operations can be performed only by the source and destination. If an IPv6 datagram received by a router is too large to be forwarded over the outgoing link, the router simply drops the datagram and sends a “Packet Too Big” ICMP error message (see below) back to the sender
- **Header checksum**. Because the transport-layer (for example, TCP and UDP) and link-layer (for example, Ethernet) protocols in the Internet layers perform checksumming, the designers of IP probably felt that this functionality was sufficiently redundant in the network layer that it could be removed.
- **Options.** An options field is no longer a part of the standard IP header Instead, the options field is one of the possible next headers pointed to from within the IPv6 header

4.5 Routing Algorithms: The purpose of a routing algorithm is simple: given a set of routers, with links connecting the routers, a routing algorithm finds a "good" path from source to destination. Typically, a "good" path is one which has "least cost".



Given the graph abstraction, the problem of finding the least cost path from a source to a destination requires identifying a series of links such that:

Graph: G = (N,E)

N = set of routers = { u, v, w, x, y, z }

E = set of links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

Cost of path (x1, x2, x3,…, xp) = c(x1,x2) + c(x2,x3) + … + c(xp-1,xp)

Classification of Routing Algorithms: Broadly, **one way** in which we can classify routing algorithms is according to whether they are centralized (global) or decentralized:

A global routing algorithm computes the least cost path between a source and destination using complete, global knowledge about the network. That is, the algorithm takes the connectivity between all nodes and all links costs as inputs. This then requires that the algorithm somehow obtain this information before actually performing the calculation. The calculation itself can be run at one site (a centralized global routing algorithm) or replicated at multiple sites. In practice, algorithms with global state information are often referred to as link state algorithms.

Decentralized routing algorithm, the calculation of the least cost path is carried out in an iterative, distributed manner. No node has complete information about the costs of all network links. Instead, each node begins with only knowledge of the costs of its own directly attached links and then through an iterative process of calculation and exchange of information with its neighbouring nodes (i.e., nodes which are at the "other end" of links to which it itself is attached) gradually calculates the least cost path to a destination, or set of destinations. The decentralized routing algorithm is known as a distance vector algorithm.

A **second** broad way to classify routing algorithms is according to whether they are **static or dynamic**. In static routing algorithms, routes change very slowly over time, often as a result of human intervention (e.g., a human manually editing a router's forwarding table). Dynamic routing algorithms change the routing paths as the network traffic loads (and the resulting delays experienced by traffic) or topology change.

A third way to classify routing algorithms is according to weather they are **load sensitive or load-insensitive algorithm**. In load sensitive algorithm link costs vary dynamically to reflect the current level of congestion in the underlying link. In load-insensitive algorithm a link's cost does not explicitly reflect its current level of congestion.

4.5.1 A Link-State Routing Algorithm: The link state algorithm is known as Dijkstra's algorithm, named after its inventor. It computes the least cost path from one node to all other nodes in the network. Dijkstra's algorithm is iterative and has the property that after the kth iteration of the algorithm, the least cost paths are known to k destination nodes, and among the least cost paths to all destination nodes, these k path will have the k smallest costs. The notations are as follows:

c(x,y): link cost from node x to y; set to ∞ if a and y are not direct neighbours

D(v): current value of cost of path from source to destination. v

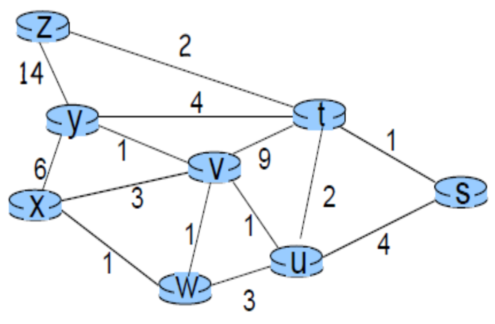
p(v): v's predecessor node along path from source to v

N': set of nodes whose least cost path is definitively known

Link- state Algorithm

```
1  Initialization (u = source node):
2  N' = {u} /* path to self is all we know */
3  for all nodes v
4  if v adjacent to u
5  then D(v) = c(u,v) /* assign link cost to neighbours */
6  else D(v) = ∞
7
8  Loop
9      find w not in N' such that D(w) is a minimum
10     add w to N'
11     update D(v) for all v adjacent to w and not in N' :
12     D(v) = min( D(v), D(w) + c(w,v) )
13    /* new cost to v is either old cost to v or known
14    shortest path cost to w plus cost from w to v */
15 until all nodes in N'
```

Example :



Step	N'	D(s),p(s)	D(t),p(t)	D(u),p(u)	D(v),p(v)	D(w),p(w)	D(y),p(y)	D(z),p(z)
0	x	∞	∞	∞	3,x	1,x	6,x	∞

Initialization:
- Store source node x in N'
- Assign link cost to neighbours (v,w,y)
- Keep track of predecessor to destination node

The X’s forwarding table for the graph is as follows:

Step	N’	D(s),p(s)	D(t),p(t)	D(u),p(u)	D(v),p(v)	D(w),p(w)	D(y),p(y)	D(z),p(z)
0	x	∞	∞	∞	3,x	1,x	6,x	∞
1	xw	∞	∞	4,w	2,w		6,x	∞
2	xwv	∞	11,v	3,v			3,v	∞
3	xwvu	7,u	5,u				3,v	∞
4	xwvuy	7,u	5,u					17,y
5	xwvuyt	6,t						7,t
6	xwvuyts							7,t

Algorithm complexity: n nodes

- each iteration: need to check all nodes, w, not in N
- n(n+1)/2 comparisons: O(n2)
- more efficient implementations possible: O(nlogn)

4.5.2 A Distance Vector Routing Algorithm: The distance vector (DV) algorithm is iterative, asynchronous, and distributed. It is distributed in that each node receives some information from one or more of its directly attached neighbours, performs a calculation, and may then distribute the results of its calculation back to its neighbours. It is iterative in that this process continues on until no more information is exchanged between neighbours. The algorithm is asynchronous in that it does not require all of the nodes to operate in lock step with each other. The principal data structure in the DV algorithm is the distance table maintained at each node. Each node's distance table has a row for each destination in the network and a column for each of its directly attached neighbours. Each node x begins with Dx(y), an estimate of the cost of the least-cost path from itself to node y, for all nodes in N. Let Dx = [Dx(y): yin N] be node x’s distance vector, which is the vector of cost estimates from x to all other nodes, y, in N. With the DV algorithm, each node x maintains the following routing information:

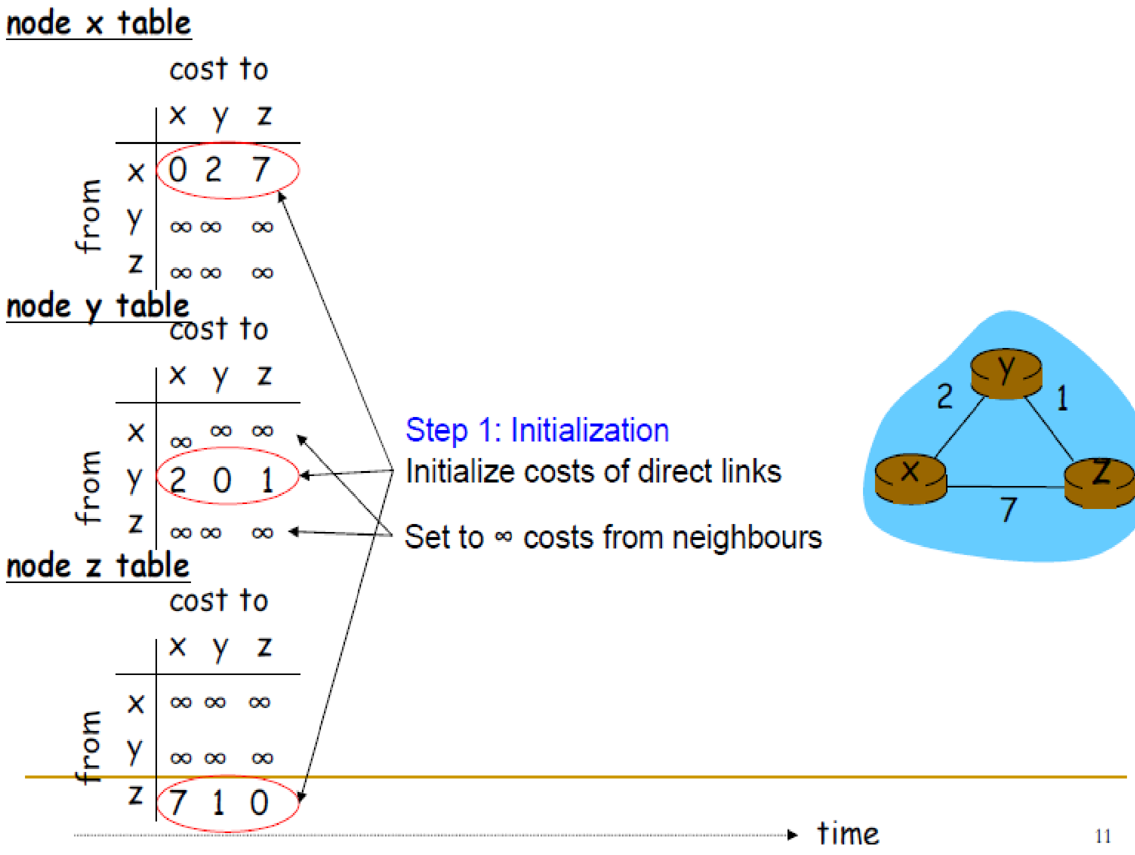
- For each neighbour v, the cost c(x,v) from x to directly attached neighbour, v
- Node x’s distance vector, that is, Dx = [Dx(y): y in N], containing x’s estimate of its cost to all destinations, y, in N
- The distance vectors of each of its neighbours, that is, Dv = [Dv (y): y in N] for each neighbour v of x

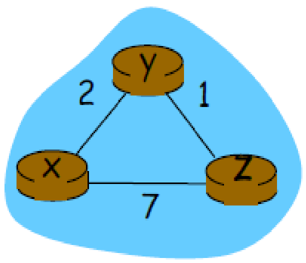
In the distributed, asynchronous algorithm, from time to time, each node sends a copy of its distance vector to each of its neighbours. When a node x receives a new distance vector from any of its neighbour’s v, it saves v’s distance vector, and then uses the Bellman-Ford equation to update its own distance vector as follows:

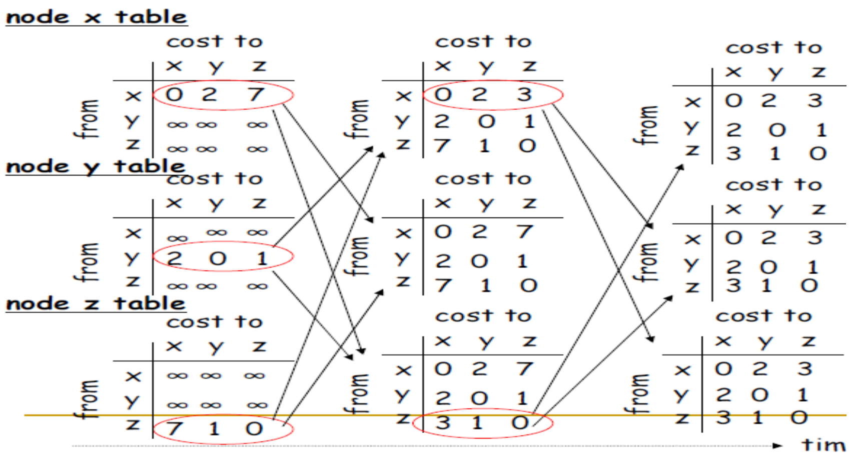
Dx(y) minv{c(x,v) + Dv(y)} for each node y in N

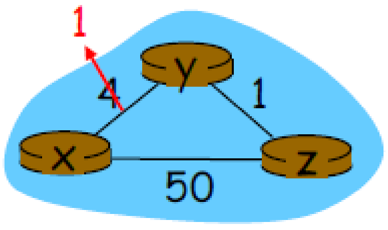
If node x’s distance vector has changed as a result of this update step, node x will then send its updated distance vector to each of its neighbours, which can in turn update their own distance vectors.

Example:



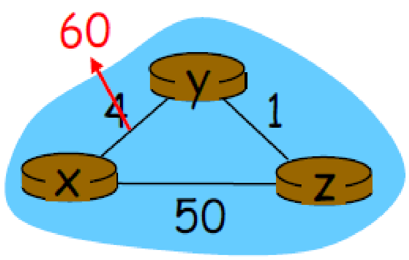






Distance Vector: link cost changes (decreases)

- node detects local link cost change
- updates routing info, recalculates distance vector
- if DV changes, notify neighbours



At time t0, y detects the link-cost change, updates its DV, and informs its neighbours.

At time t1, z receives the update from y and updates its table. It computes a new least cost to x and sends neighbours its DV.

At time t2, y receives z’s update and updates its distance table. y’s least costs do not change and hence y does not send any message to z.

Distance Vector: link cost changes (increases)

- Bad news travels slow - “count to infinity” problem!
- When y detects cost change to 60, it will update its DV using the z’s cost to x, which is 5 (via y), to obtain an incorrect new cost to x of 6, over the path yzyx that has a loop
- 44 iterations before algorithm stabilizes, while y and z exchange updates

Distance-Vector Algorithm: Adding Poisoned Reverse

Count to infinity can be avoided using a technique known as poisoned reverse. The idea is simple—if z routes through y to get to destination x, then z will advertise to y that its distance to x is infinity, that is, z will advertise to y that $D_z(x) = \infty$ (even though z knows $D_z(x) = 5$ in truth). z will continue telling this little white lie to y as long as it routes to x via y. Since y believes that z has no path to x, y will never attempt to route to x via z, as long as z continues to route to x via y (and lies about doing so)

Distance Vector Routing Algorithms

```
Initialization:
for all destinations y in N:
$D_x(y) = c(x,y)$ /* if y is not a neighbour then $c(x,y) = \infty$ */
for each neighbour w
$D_w(y) = ?$ for all destinations y in N
for each neighbour w
send distance vector $D_x = [D_x(y): y \text{ in } N]$ to w

 loop
   wait (until I see a link cost change to some neighbour w or
           until I receive a distance vector from some neighbour w)

  for each y in N:
        $D_x(y) = \min_v\{c(x,v) + D_v(y)\}$

  if $D_x(y)$ changed for any destination y
    send distance vector $D_x = [D_x(y): y \text{ in } N]$ to all neighbours

  forever
```

A Comparison of LS and DV Routing Algorithms: In the DV algorithm, each node talks to only its directly connected neighbours, but it provides its neighbours with least-cost estimates from itself to all the nodes (that it knows about) in the network**.** In the LS algorithm, each node talks with all other nodes (via broadcast), but it tells them only the costs of its directly connected links.

- **Message complexity**
  - ♦ **LS**: with n nodes, E links, O(nE) msgs sent each
  - ♦ **DV**: exchange between neighbours only, convergence time varies
- **Speed of Convergence**
  - ♦ **LS**: O(n2) algorithm requires O(nE) msgs
    - ➢ may have oscillations

- ♦ **DV**: convergence time varies
    - ➢ may be routing loops
    - ➢ count-to-infinity problem
- **Robustness: what happens if router malfunctions?**
  - ♦ **LS**:
    - ➢ node can advertise incorrect link cost
    - ➢ each node computes only its own table
  - ♦ **DV**:
    - ➢ DV node can advertise incorrect path cost
    - ➢ each node's table used by others(error propagate thru network)

5.1 Link Layer: Introduction: Any device that runs a link layer protocol is a node. Nodes include hosts, routers, switches, and WiFi access points. In order for a datagram to be transferred from source host to destination host, it must be move over each of the individual links in the end-to-end path.

5.1.1 Link Layer Services: Link layer services provided by a link layer protocol are the following

- Framing
- Link access
- Reliable delivery
- Error detection and correction
- Half-Duplex and Full-Duplex

Framing: Almost all link layer protocols encapsulate each network layer datagram with a link layer frame before transmission over the link. A frame consists of a data field, in which the network layer datagram is inserted, and a number of header fields. The structure of the frame is specified by the link layer protocol.

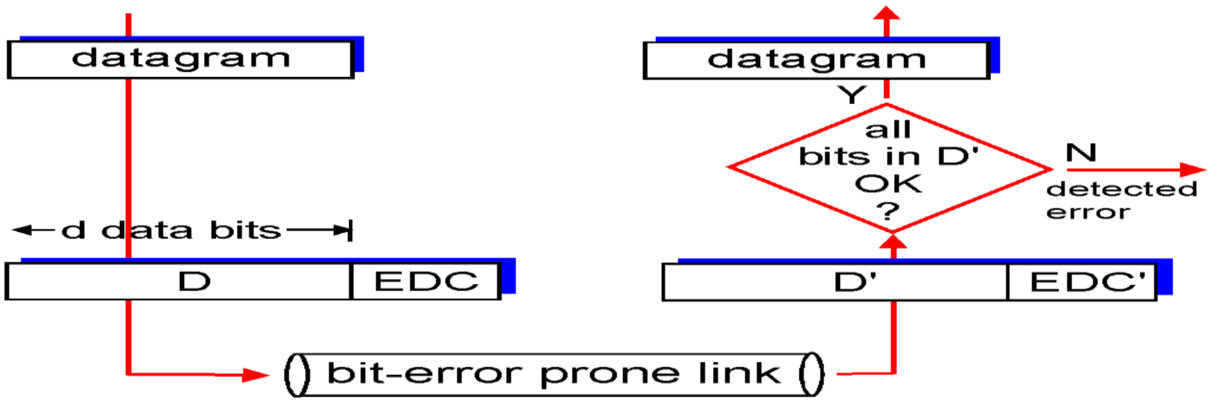
Link access: A medium access control (MAC) protocol specifies the rules by which a frame is transmitted onto the link. For point-to-point links that have single sender at one end of the link and a single receiver at the other end of the link, the MAC protocol is simple (or non-existent) – the sender can send a frame whenever the link is idle. The more interesting case is when multiple nodes share a single broadcast link – the so-called multiple access problem. Here, the MAC protocol serves to coordinate the frame transmission of the many nodes.

Reliable delivery: When a link layer protocol provides reliable delivery service, it guarantees to move each network layer datagram across the link without error. Similar to a transport layer reliable delivery service, a link layer reliable delivery service can be achieved with acknowledgements and retransmissions. A link layer reliable delivery service is often used for links that are prone to high error rates, such as a wireless link, with the goal of correcting an error locally – on the link where the link error occurs – rather than forcing an end-to-end retransmission of the data by a transport- or application layer protocol.

Error Detection & Correction: Error detection in the link layer is usually more sophisticated and is implemented in hardware. Error correction is similar to error detection, except that a receiver not only detects when bit errors have occurred in the frame but also determines exactly where in the frame the errors have occurred (and then corrects these errors).

Half-Duplex and Full-Duplex: With full-duplex transmission, both nodes at the ends of a link may transmit packets at the same time. With half-duplex transmission, a node cannot both transmit and receive at the same time

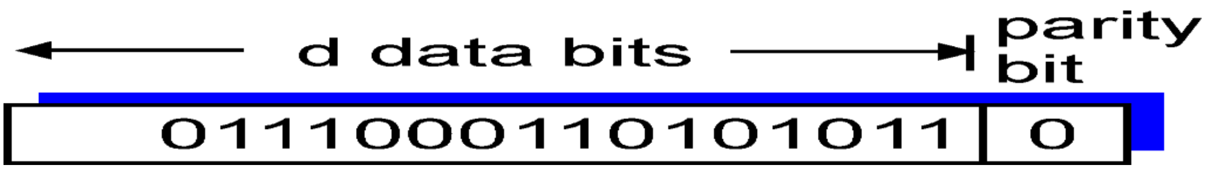
5.2 Error Detection and Correction Techniques: Detecting and correcting the corruption of bits in a link-layer frame sent from one node to another physically connected neighbouring node—are two services often provided by the link layer. Figure below illustrates the Error-detection and -correction scenario. At the sending node, data, D, to be protected against bit errors is augmented with error-detection and -correction bits (EDC). Both D and EDC are sent to the receiving node in a link-level frame. At the receiving node, a sequence of bits, D^1 and EDC^1 is received. The receiver's challenge is to determine whether or not D^1 is the same as the original D, given that it has only received D^1 and EDC^1



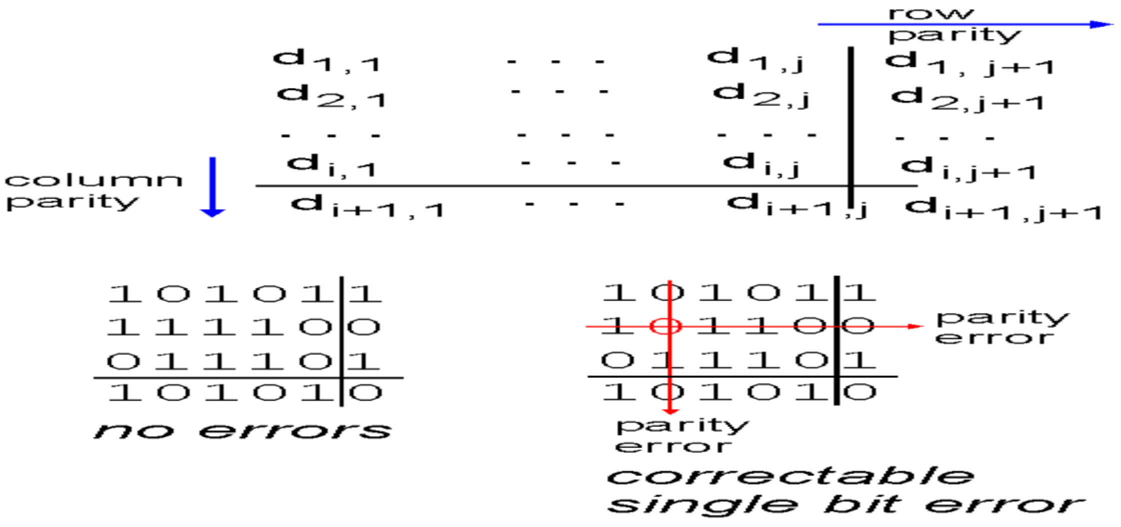
Techniques for error detection are:

1. Simple Parity check
2. Two-dimensional Parity check
3. Checksum
4. Cyclic redundancy check

5.2.1 Parity Checks: Perhaps the simplest form of error detection is the use of a single parity bit. Suppose that the information to be sent, D, has d bits. In an even parity scheme, the sender simply includes one additional bit and chooses its value such that the total number of 1s in the d + 1 bits (the original information plus a parity bit) is even. For odd parity schemes, the parity bit value is chosen such that there is an odd number of 1s. Figure below illustrates an even parity scheme, with the single parity bit being stored in a separate field. Receiver operation is also simple with a single parity bit. The receiver need only count the number of 1s in the received d + 1 bits. If an odd number of 1- valued bits are found with an even parity scheme, the receiver knows that at least one bit error has occurred. More precisely, it knows that some odd number of bit errors has occurred.



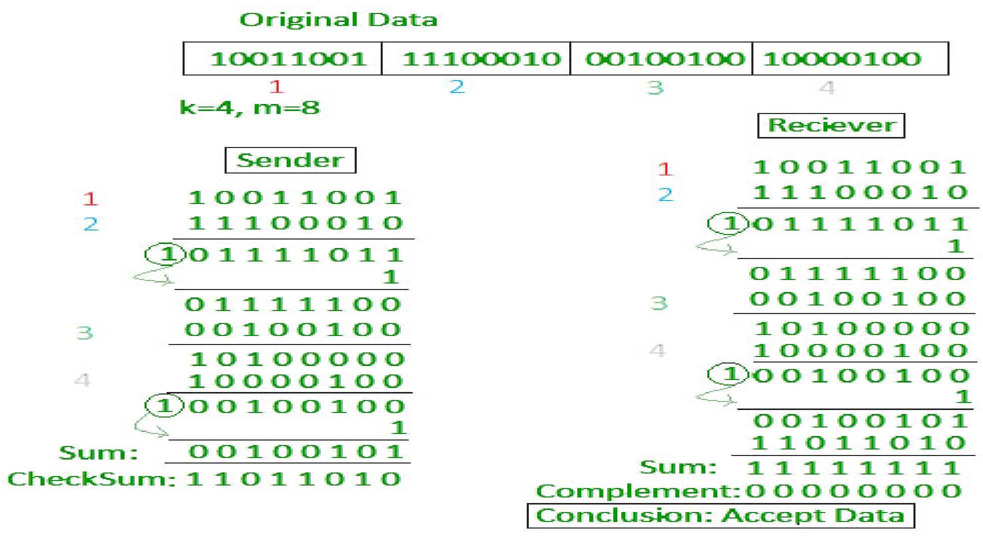
Two-dimensional Parity check: Here, the d bits in D are divided into i rows and j columns. A parity value is computed for each row and for each column. The resulting i + j + 1 parity bits comprise the link-layer frame's error-detection bits. Suppose now that a single bit error occurs in the original d bits of information. With this two-dimensional parity scheme, the parity of both the column and the row containing the flipped bit will be in error. The receiver can thus not only detect the fact that a single bit error has occurred, but can use the column and row indices of the column and row with parity errors to actually identify the bit that was corrupted and correct that error.


row parity
column parity
$d_{1,1}$ ... $d_{1,j}$ $d_{1,j+1}$
$d_{2,1}$ ... $d_{2,j}$ $d_{2,j+1}$
$d_{i,1}$ ... $d_{i,j}$ $d_{i,j+1}$
$d_{i+1,1}$ ... $d_{i+1,j}$ $d_{i+1,j+1}$
101011
111100
011101
101010
no errors
101011
101100
011101
101010
parity error
parity error
correctable single bit error

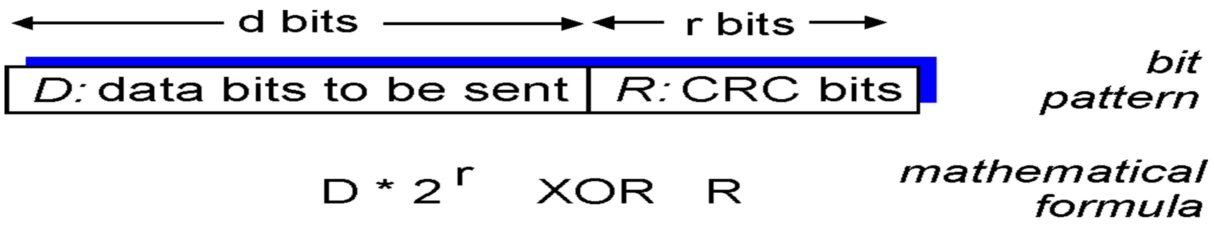
Two-dimensional even parity

5.2.2 Check summing Methods:

- In checksum error detection scheme, the data is divided into k segments each of m bits.
- In the sender's end the segments are added using 1's complement arithmetic to get the sum. The sum is complemented to get the checksum.
- The checksum segment is sent along with the data segments.
- At the receiver's end, all received segments are added using 1's complement arithmetic to get the sum. The sum is complemented.
- If the result is zero, the received data is accepted; otherwise discarded.



5.2.3 Cyclic Redundancy Check (CRC): An error-detection technique used widely in today's computer networks is based on cyclic redundancy check (CRC) codes. CRC codes are also known as polynomial codes, since it is possible to view the bit string to be sent as a polynomial whose coefficients are the 0 and 1 values in the bit string, with operations on the bit string interpreted as polynomial arithmetic.



CRC codes operate as follows. Consider the d-bit piece of data, D, that the sending node wants to send to the receiving node. The sender and receiver must first agree on an r + 1 bit pattern, known as a generator, which we will denote as G. We will require that the most significant (leftmost) bit of G be a 1. For a given piece of data, D, the sender will choose r additional bits, R, and append them to D such that the resulting d + r bit pattern (interpreted as a binary number) is exactly divisible by G (i.e., has no remainder) using modulo-2 arithmetic. The process of error checking with CRCs is thus simple: The receiver divides the d + r received bits by G. If the remainder is nonzero, the receiver knows that an error has occurred; otherwise the data is accepted as being correct.

All CRC calculations are done in modulo-2 arithmetic without carries in addition or borrows in subtraction. This means that addition and subtraction are identical, and both are equivalent to the bitwise exclusive-or (XOR) of the operands. Thus, for example,

1011 XOR 0101 = 1110

1001 XOR 1101 = 0100

Also, we similarly have

1011 – 0101 = 1110

1001 – 1101 = 0100

Multiplication and division are the same as in base-2 arithmetic, except that any required addition or subtraction is done without carries or borrows. Let us now turn to the crucial question of how the sender computes R. Recall that we want to find R such that there is an n such that

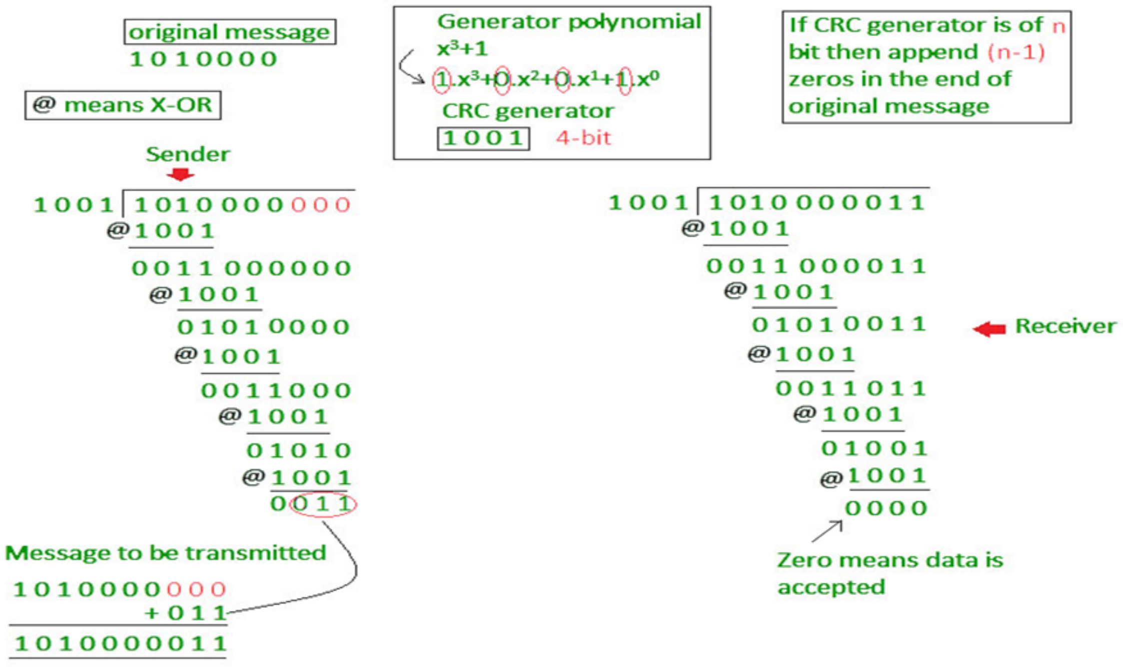
$$D*2^r \text{XOR } R = nG$$

That is, we want to choose R such that G divides into $D*2^r$XOR R without remainder. If we XOR (that is, add modulo-2, without carry) R to both sides of the above equation, we get

$$D*2^r = nG \text{ XOR } R$$

This equation tells us that if we divide $D*2^r$ by G, the value of the remainder is precisely R. In other words, we can calculate R as

$$R = \text{remainder} (D*2^r / G)$$

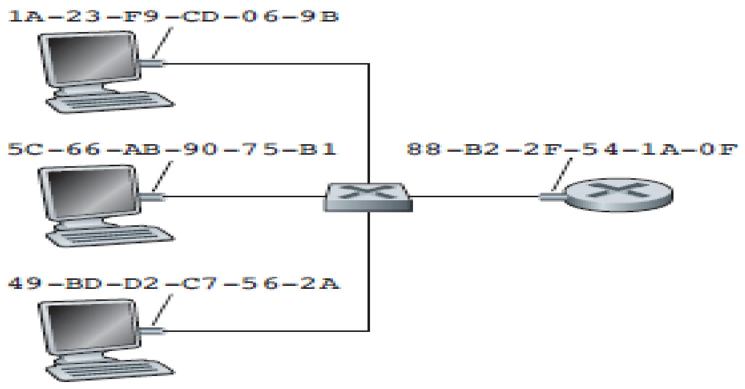


5.4 Link-Layer Addressing: Hosts and routers have link-layer addresses. Hosts and routers have network-layer addresses as well**.** Why in the world do we need to have addresses at both the network and link layers?

5.4.1 MAC Addresses: In truth, it is not hosts and routers that have link-layer addresses but rather their adapters that have link-layer addresses. A link-layer address is variously called a LAN address, a physical address, or a MAC address. MAC address is 6 bytes long, giving

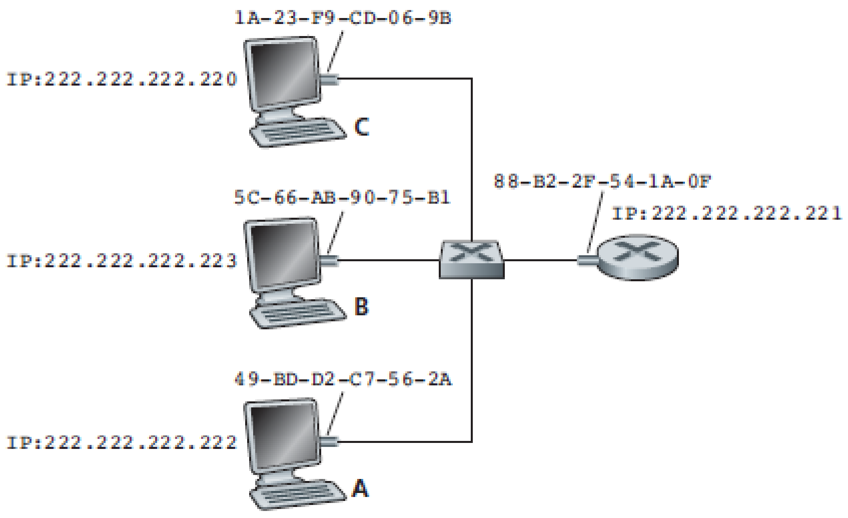
248 possible MAC addresses, these 6-byte addresses are typically expressed in hexadecimal notation, with each byte of the address expressed as a pair of hexadecimal numbers. One interesting property of MAC addresses is that no two adapters have the same address. An adapter's MAC address has a flat structure (as opposed to a hierarchical structure) and doesn't change no matter where the adapter goes. When an adapter wants to send a frame to some destination adapter, the sending adapter inserts the destination adapter's MAC address into the frame and then sends the frame into the LAN. Thus, an adapter may receive a frame that isn't addressed to it. Thus, when an adapter receives a frame, it will check to see whether the destination MAC address in the frame matches its own MAC address. If there is a match, the adapter extracts the enclosed datagram and passes the datagram up the protocol stack. If there isn't a match, the adapter discards the frame, without passing the network-layer datagram up.

However, sometimes a sending adapter does want all the other adapters on the LAN to receive and process the frame it is about to send. In this case, the sending adapter inserts a special MAC broadcast address into the destination address field of the frame. For LANs that use 6-byte addresses (such as Ethernet and 802.11), the broadcast address is a string of 48 consecutive 1s (that is, FF-FF-FF-FF-FFFF in hexadecimal notation).



Each interface connected to a LAN has a unique MAC address

5.4.2 Address Resolution Protocol (ARP): Because there are both network-layer addresses (for example, Internet IP addresses) and link-layer addresses (that is, MAC addresses), there is a need to translate between them. For the Internet, this is the job of the Address Resolution Protocol. In this example below, each host and router has a single IP address and single MAC address



Now suppose that the host with IP address 222.222.222.220 wants to send an IP datagram to host 222.222.222.222. In this example, both the source and destination are in the same subnet. To send a datagram, the source must give its adapter not only the IP datagram but also the MAC address for destination 222.222.222.222. ARP does this work. The sending adapter will then construct a link layer frame containing the destination's MAC address and send the frame into the LAN. An ARP module in the sending host takes any IP address on the same LAN as input, and returns the corresponding MAC address. In the example at hand, sending host 222.222.222.220 provides its ARP module the IP address 222.222.222.222, and the ARP module returns the corresponding MAC address 49-BD-D2-C7-56-2A.

Working of ARP: Each host and router has an ARP table in its memory, which contains mappings of IP addresses to MAC addresses. Figure 5.18 shows what an ARP table in host 222.222.222.220 might look like. The ARP table also contains a time-to-live (TTL) value, which indicates when each mapping will be deleted from the table. Note that a table does not necessarily contain an entry for every host and router on the subnet; some may have never been entered into the table, and others may have expired. A typical expiration time for an entry is 20 minutes from when an entry is placed in an ARP table.

IP Address	MAC Address	TTL
222.222.222.221	88-B2-2F-54-1A-0F	13:45:00
222.222.222.223	5C-66-AB-90-75-B1	13:52:00

Figure 5.18 ♦ A possible ARP table in 222.222.222.220

If the ARP table doesn't currently have an entry for the destination, First, the sender constructs a special packet called an ARP packet. The purpose of the ARP query packet is to query all the other hosts and routers on the subnet to determine the MAC address corresponding to the IP address that is being resolved. Both ARP query and response packets have the same format

Returning to our example, 222.222.222.220 passes an ARP query packet to the adapter along with an indication that the adapter should send the packet to the MAC broadcast address, namely, FF-FF-FF-FF-FF-FF. The adapter encapsulates the ARP packet in a link-layer frame, uses the broadcast address for the frame's destination address, and transmits the frame into the subnet. The frame containing the ARP query is received by all the other adapters on the subnet, and (because of the broadcast address) each adapter passes the ARP packet within the frame up to its ARP module. Each of these ARP modules checks to see if its IP address matches the destination IP address in the ARP packet. The one with a match sends back to the querying host a response ARP packet with the desired mapping. The querying host 222.222.222.220 can then update its ARP table and send its IP datagram, encapsulated in a link-layer frame whose destination MAC is that of the host or router responding to the earlier ARP query.

Sending a Datagram off the Subnet: ARP operates when a host wants to send a datagram to another host on the same subnet. When a host on a subnet wants to send a network-layer datagram to a host off the subnet (that is, across a router onto another subnet) the procedure is as follows:

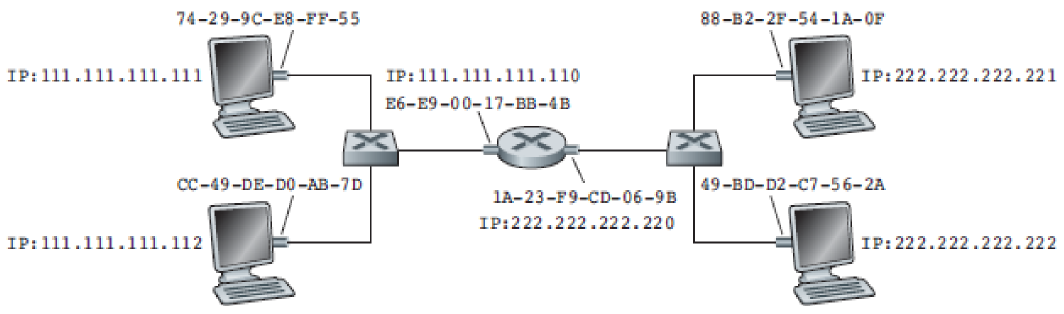


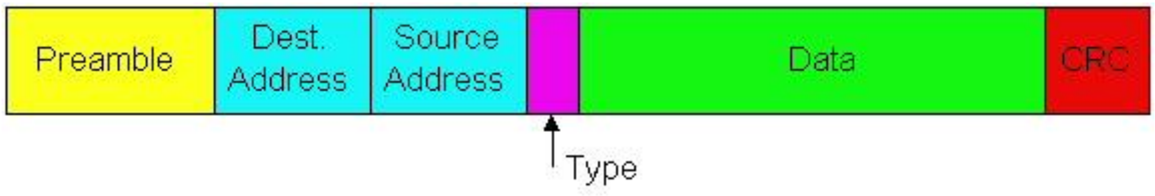
Figure 5.19 ♦ Two subnets interconnected by a router

Each host has exactly one IP address and one adapter. A router has an IP address for each of its interfaces. For each router interface there is also an ARP module (in the router) and an adapter. Specifically, suppose that host 111.111.111.111 wants to send an IP datagram to a host 222.222.222.222. In order for a datagram to go from 111.111.111.111 to a host on Subnet 2, the datagram must first be sent to the router interface 111.111.111.110, which is the IP address of the first-hop router on the path to the final destination. Thus, the appropriate MAC address for the frame is the address of the adapter for router interface 111.111.111.110, namely, E6-E9-00-17-BB-4B. Once the sending adapter has this MAC address, it creates a frame (containing the datagram addressed to 222.222.222.222) and sends the frame into Subnet 1. The router adapter on Subnet 1 sees that the link-layer frame is addressed to it, and therefore passes the frame to the network layer of the router. The router now has this is done by consulting a forwarding table in the router. The forwarding table tells the router that the datagram is to be forwarded via router interface 222.222.222.220. This interface then passes the datagram to its adapter, which encapsulates the datagram in a new frame and sends the frame into Subnet 2. This time, the destination MAC address of the frame is indeed the MAC address of the ultimate destination to determine the correct interface on which the datagram is to be forwarded.

5.5 Ethernet: Ethernet is by far the most prevalent wired LAN technology; there are many reasons for Ethernet's success.

- First, Ethernet was the first widely deployed high-speed LAN
- Second, token ring, FDDI, and ATM were more complex and expensive than Ethernet,
- Third, Ethernet producing versions that operated at equal data rates or higher than others
- Ethernet hardware (in particular, adapters and switches) has become a commodity and is remarkably cheap.
- Ethernet is easy to understand, implement, maintain and allows low cost network implementation. Also, Ethernet offers flexibility in terms of topologies which are allowed. Ethernet operates in two layers of OSI model, Physical Layer and Data Link Layer.

The Ethernet frame structure is as follows:



Preamble – Ethernet frame starts with 7-Bytes Preamble. This is pattern of alternative 0's and 1's which indicates starting of the frame and allows sender and receiver to establish bit synchronization. Initially, PRE (Preamble) was introduced to allow for the loss of few bits due to signal delays. But todays high-speed Ethernet don't need Preamble to protect the frame bits. PRE (Preamble) indicates the receiver that frame is coming and allow the receiver to lock onto the data stream before the actual frame begins.

Destination Address – This is 6-Byte field which contains the MAC address of machine for which data is destined.

Source Address – This is a 6-Byte field which contains the MAC address of source machine. As Source Address is always an individual address (Unicast), the least significant bit of first byte is always 0

Data – This is the place where actual data is inserted, also known as Payload. Both IP header and data will be inserted here, if Internet Protocol is used over Ethernet. The maximum data present may be as long as 1500 Bytes. In case data length is less than minimum length i.e. 46 bytes, then padding 0’s is added to meet the minimum possible length.

Cyclic Redundancy Check (CRC) – CRC is 4 Byte field. This field contains 32-bits hash code of data, which is generated over Destination Address, Source Address, Length and Data field. If the checksum computed by destination is not same as sent checksum value, data received is corrupted

Type - The type field permits Ethernet to "multiplex" network-layer protocols. A hosts can use other network-layer protocols besides IP. In fact, a given host may support multiple network layer protocols, and use different protocols for different applications. For this reason, when the Ethernet frame arrives at adapter B, adapter B needs to know to which network-layer protocol it should pass the contents of the data field. IP and other data-link layer protocols (e.g., Novell IPX or AppleTalk) each have their own, standardized type number.