UNIT III

Circuit switching vs. packet switching / Packet switched networks – IP – ARP – RARP – DHCP – ICMP – Queueing discipline – Routing algorithms – RIP – OSPF – Subnetting – CIDR – Interdomain routing – BGP – Ipv6 – Multicasting – Congestion avoidance in network layer

3.1 Circuit switching vs. packet switching / Packet switched networks

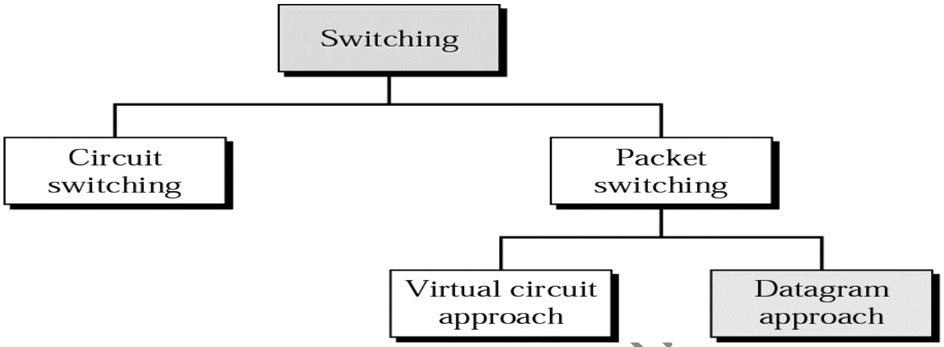


Figure 3.1.1: Switching Techniques

In general switching can be divided into two categories namely circuit switching and packet switching.

In **circuit switching** a dedicated physical link exists between a source and a destination, in this case data is sent as stream of bits.

Packet switching is a communications method in which packets (discrete blocks of data) are routed between nodes over data links shared with other traffic. In each network node, packets are queued or buffered, resulting in variable delay. Packet switching is used to optimize the use of the channel capacity available in digital telecommunication networks such as computer networks, to minimize the transmission latency, and to increase robustness of communication.

The most well-known use of packet switching is the Internet and local area networks. The Internet uses the Internet protocol suite over a variety of data link layer protocols. For example, Ethernet and frame relay are very common. Newer mobile phone technologies also use packet switching.

The following paradigms are available for packet switching

- **Virtual Circuit Switching**: A connection is setup before the packets are transmitted. All the packets follow the same path. The connection could either be permanent (manually setup by network administrator) or switched (dynamically setup through control signals).

- **Datagram Switching**: No connection is setup each packet is forwarded independent of the other.

An initial setup phase is used to set up a route between the intermediate nodes for all the packets passed during the session between the two end nodes. In each intermediate node, an entry is registered in a table to indicate the route for the connection that has been set up. Thus, packets passed through this route, can have short headers, containing only a virtual circuit identifier (VCI), and not their destination. Each intermediate node passes the packets according to the information that was stored in it, in the setup phase.

In this way, packets arrive at the destination in the correct sequence, and it is guaranteed that essentially there will not be errors. This approach is slower than Circuit Switching, since different virtual circuits may compete over the same resources, and an initial setup phase is needed to initiate the circuit. As in Circuit Switching, if an intermediate node fails, all virtual circuits that pass through it are lost.

The most common forms of Virtual Circuit networks are X.25 and Frame Relay, which are commonly used for public data networks (PDN). X.25 is a notable use of packet switching in that, despite being based on packet switching methods, it provided virtual circuits to the user. These virtual circuits carry variable-length packets. X.25 was used to provide the first international and commercial packet switching network, the International Packet Switched Service (IPSS). Asynchronous Transfer Mode (ATM) also is a virtual circuit technology, which uses fixed-length cell relay connection oriented packet switching.

Datagram packet switching is also called connectionless networking because no connections are established. Technologies such as Multiprotocol Label Switching (MPLS) and the Resource Reservation Protocol (RSVP) create virtual circuits on top of datagram networks. Virtual circuits are especially useful in building robust failover mechanisms and allocating bandwidth for delay-sensitive applications. MPLS and its predecessors, as well as ATM, have been called "fast packet" technologies. MPLS, indeed, has been called "ATM without cells". Modern routers, however, do not require these technologies to be able to forward variable-length packets at multigigabit speeds across the network. This approach uses a different, more dynamic scheme, to determine the route through the network links. Each packet is treated as an independent entity, and its header contains full information about the destination of the packet. The intermediate nodes examine the header of the packet, and decide to which node to send the packet so that it will reach its destination. In the decision two factors are taken into account:

- The shortest way to pass the packet to its destination - protocols such as RIP/OSPF is used to determine the shortest path to the destination.
- Finding a free node to pass the packet to - in this way, bottle necks are eliminated, since packets can reach the destination in alternate routes.

Thus, in this method, the packets don't follow a pre-established route, and the intermediate nodes (the routers) don't have pre-defined knowledge of the routes that the packets should be passed through. Packets can follow different routes to the destination, and delivery is not guaranteed (although packets usually do follow the same route, and are reliably sent). Due to the nature of this method, the packets can reach the destination in a different order than they were sent, thus they must be sorted at the destination to form the original message. This approach is time consuming since every router has to decide where to send each packet. The main implementation of Datagram Switching network is the Internet which uses the IP network protocol.

Switching at the network layer in the Internet is done using the datagram approach to packet switching. The communication at the network layer in the Internet is connectionless. The Figure 3.1.2 shows how datagram approach can be used for delivering four packets from station A to station X. It could be viewed that though all the packets belong to the same message they can take different paths to reach their destination.

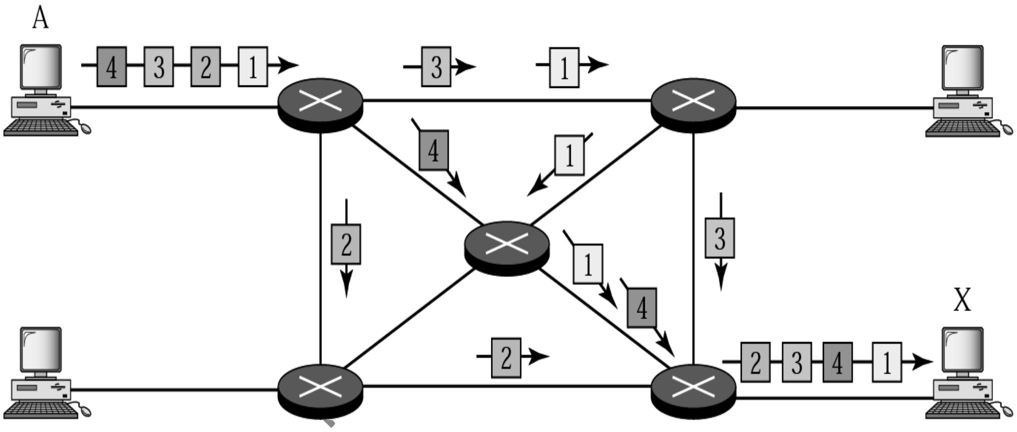


Figure 3.1.2: Datagram Approach

3.2 IP

3.2.1 What Is an Internetwork?

We use the term **"internetwork," or sometimes just "internet"** with a lowercase *i,* to refer to an **arbitrary collection of networks interconnected to provide some sort of host-to-host packet delivery service.** For example, a corporation with many sites might construct a private internetwork by interconnecting the LANs at their different sites with point-to-point links leased from the phone company.

When we are talking about the widely used, global internetwork to which a large percentage of networks are now connected, we call it the **"Internet" with a capital *I*.**

Figure 3.2.1 shows an example internetwork. An internetwork is often referred to as a "network of networks" because it is made up of lots of smaller networks. In this figure, we see Ethernets, an FDDI ring, and a point-to-point link. Each of these is a single-technology network.

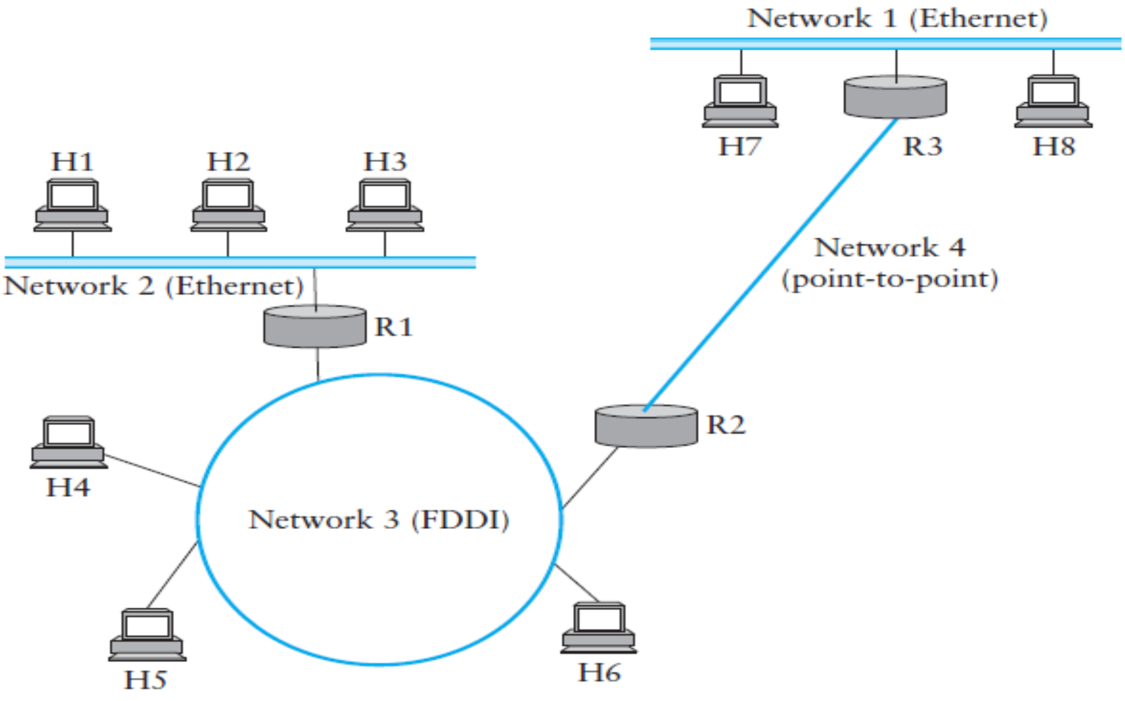


Fig. 3.2.1 A simple internetwork. H*n* = host; R*n* = router.

3.2.2 Service Model

Using IP, we can provide a host-to-host service. The IP service model can be thought of as having two parts: **an addressing scheme, which provides a way to identify all hosts in the internetwork, and a datagram (connectionless) model of data delivery.**

This service model is sometimes called *best effort (unreliable* service) because, although IP makes every effort to deliver datagrams, it makes no guarantees.

3.2.2.1 Datagram Delivery

The IP datagram is fundamental to the Internet Protocol. A datagram is a type of packet that happens to be sent in a connectionless manner over a network. Every datagram carries enough information to let the network forward the packet to its correct destination; there is no need for any advance setup mechanism to tell the network what to do when the packet arrives. You just send it, and the network makes its best effort to get it to the desired destination. The “best-effort” part means that if something goes wrong and the packet gets lost, corrupted, misdelivered, or in any way fails to reach its intended destination, the network does nothing—it made its best effort, and that is all it has to do. It does not make any attempt to recover from the failure. This is sometimes called an *unreliable* service.

Best-effort delivery does not just mean that packets can get lost. Sometimes they can get delivered out of order, and sometimes the same packet can get delivered more than once. The higher-level protocols or applications that run above IP need to be aware of all these possible failure modes.

Packet Format

The IP datagram, like most packets, consists of a header followed by a number of bytes of data. The format of the header is shown in Figure 3.2.2

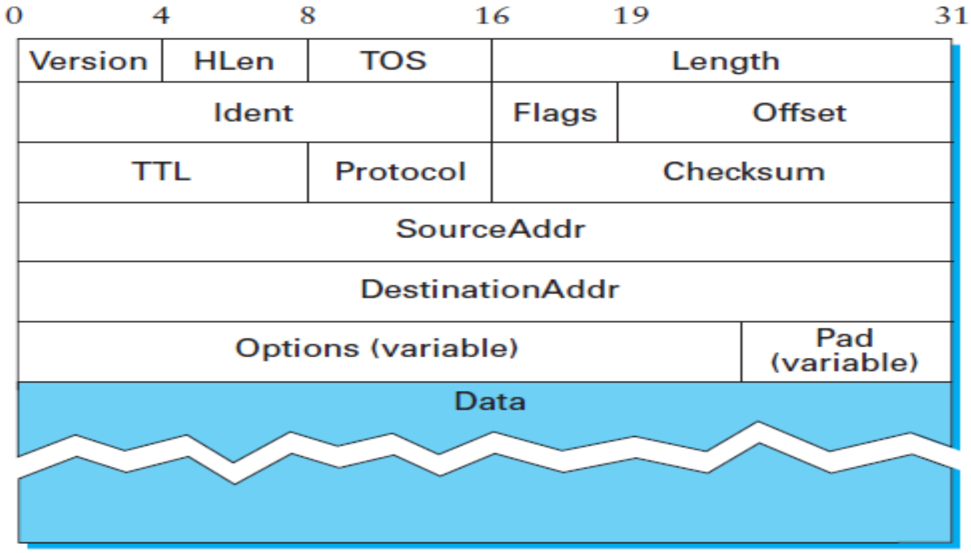


Fig. 3.2.2 IPv4 Packet Header

Version: The **Version field** specifies the version of IP. The current version of IP is 4, and it is sometimes called IPv4.

HLen : The next field, **HLen,** specifies the length of the header in 32-bit words. When there are no options, which is most of the time, the header is 5 words (20 bytes) long.

TOS : The 8-bit **TOS (**type of service) field has had a number of different definitions over the years, but its basic function is to allow packets to be treated differently based on application needs. For example, the TOS value might determine whether or not a packet should be placed in a special queue that receives low delay.

Length : The next 16 bits of the header contain the **Length** of the datagram, including the header. Unlike the HLen field, the Length field counts bytes rather than words. Thus, the maximum size of an IP datagram is 65,535 bytes. The physical network over which IP is running, however, may not support such long packets. For this reason, IP supports a **fragmentation and reassembly process**. The second word of the header contains information about fragmentation, and the details of its use are presented under "Fragmentation and Reassembly" below.

TTL : Moving on to the third word of the header, the next byte is the **TTL (time to live)** field. Its name reflects its historical meaning rather than the way it is commonly used today. The intent of the field is to catch packets that have been going around in routing loops and discard them, rather than let them consume resources indefinitely. Originally, TTL was set to a specific number of seconds that the packet would be allowed to live, and routers along the path would decrement this field until it reached 0. However, since it was rare for a packet to sit for as long as 1 second in a router, and routers did not all have access to a common clock, most routers just decremented the TTL by 1 as they forwarded the packet. Thus, it became more of a hop count than a timer, which is still a perfectly good way to catch packets that are stuck in routing loops. One subtlety is in the initial setting of this field by the sending host: Set it too high and packets could circulate rather a lot before getting dropped; set it too low and they may not reach their destination. The value 64 is the current default.

Ident : It allows the destination host to determine which datagram a newly arrived fragment belongs to. All the fragment of a datagram contain the same identification value

Flags : DF – Don't fragment, MF- More fragment

Offset : Max 8192 fragment per datagram

Protocol : This field is simply a demultiplexing key that identifies the higher-level protocol to which this IP packet should be passed. There are values defined for TCP (6), UDP (17), and many other protocols that may sit above IP in the protocol graph.

Checksum : This field is calculated by considering the entire IP header as a sequence of 16-bit words, adding them up using ones complement arithmetic, and taking the ones complement of the result. Thus, if any bit in the header is corrupted in transit, the checksum will not contain the correct value upon receipt of the packet. Since a corrupted header may contain an error in the destination address—and, as a result, may have been misdelivered—it makes sense to discard any packet that fails the checksum. It should be noted that this type of checksum does not have the same strong error detection properties as a CRC, but it is much easier to calculate in software.

The last two required fields in the header are the **SourceAddr and the DestinationAddr for the packet.** The latter is the key to datagram delivery: Every packet contains a full address for its intended destination so that forwarding decisions can be made at each router. The source address is required to allow recipients to decide if they want to accept the packet and to enable them to reply.

Finally, there may be a number of **options** at the end of the header. The presence or absence of options may be determined by examining the header length (HLen) field. While options are used fairly rarely, a complete IP implementation must handle them all.

Fragmentation and Reassembly

One of the problems of providing a uniform host-to-host service model over a heterogeneous collection of networks is that each network technology tends to have its own idea of how large a packet can be. For example, an Ethernet can accept packets up to 1500 bytes long, while FDDI packets may be 4500 bytes long. This leaves two choices for the IP service model: make sure that all IP datagrams are small enough to fit inside one packet on any network technology, or provide a means by which packets can be fragmented and reassembled when they are too big to go over a given network technology.

The central idea here is that every network type has a ***maximum transmission unit*** **(MTU),** which is the largest IP datagram that it can carry in a frame. When a host sends an IP datagram, therefore, it can choose any size that it wants. A reasonable choice is the MTU of the network to which the host is directly attached. Then fragmentation will only be necessary if the path to the destination includes a network with a smaller MTU. Should the transport protocol that sits on top of IP give IP a packet larger than the local MTU, however, then the source host must fragment it.

Fragmentation typically occurs in a router when it receives a datagram that it wants to forward over a network that has an MTU that is smaller than the received datagram. To enable these fragments to be reassembled at the receiving host, they all carry the same identifier in the **Ident field.** This identifier is chosen by the sending host and is intended to be unique among all the datagrams that might arrive at the destination from this source over some reasonable time period. Since all fragments of the original datagram contain this identifier, the reassembling host will be able to recognize those fragments that go together. Should all the fragments not arrive at the receiving host, the host gives up on the reassembly process and discards the fragments that did arrive. IP does not attempt to recover from missing fragments.

3.2.2.2 IP addressing methods

An IPv4 address is a 32-bit address that *uniquely* and *universally* defines the connection of a device (for example, a computer or a router) to the Internet.

An IP address is a 32 bit binary number usually represented as 4 decimal values, each representing 8 bits, in the range 0 to 255 (known as octets) separated by decimal points. This is known as "dotted decimal" notation. Figure 3.2.3 shows the **binary and dotted decimal** representation of the IP address ***128.11.3.31***.

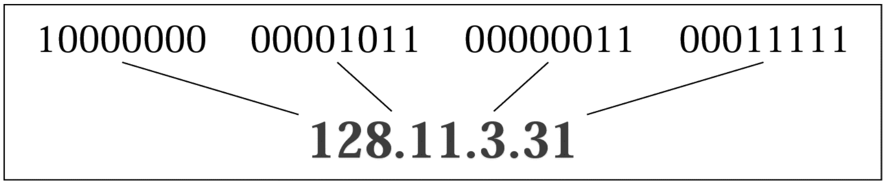


Figure 3.2.3: Binary and dotted decimal notation of IP address

There are three modes of addressing namely, Uni-cast, Multi-cast and Broad-cast addresses**.** ***Uni-Cast*** address is simply a **send-to one** addressing mode. ***Broad-Cast*** address is simply a **send-to all** addressing mode and cannot be used as source address. ***Multi-Cast*** is simply a **send-to group** addressing mode and can never be used as a source address.

Every IP address consists of two parts, one identifying the network and the other identifying the node. The Class of the address and the subnet mask determine which part belongs to the network address and which part belongs to the node address.

There are 5 different address classes namely A, B, C, D and E. It is possible to determine to which class a given IP address belong to by examining the most significant four bits of the IP address.

- **Class A** addresses begin with **0xxx**, or **1 to 127** decimal.
- **Class B** addresses begin with **10xx**, or **128 to 191** decimal.
- **Class C** addresses begin with **110x**, or **192 to 223** decimal.
- **Class D** addresses begin with **1110**, or **224 to 239** decimal.
- **Class E** addresses begin with **1111**, or **240 to 255** decimal.

The class of the IP address determines the default number of bits used for the network identification and host identification within the network. The netid and the hostid bytes for all the classes are shown in the Figure 3.2.4.

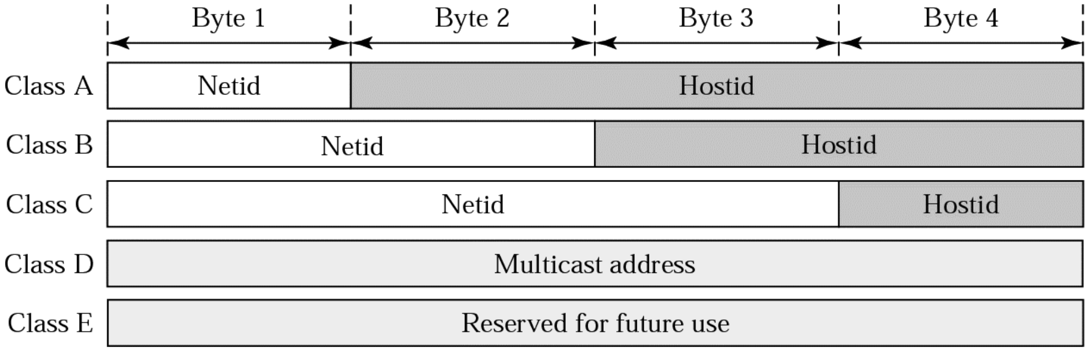


Figure 3.2.4: netid and hostid bits

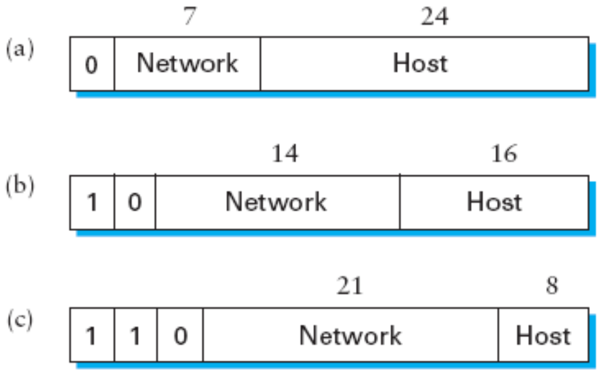


Figure 3.2.5: IP addresses : (a) Class A ; (b) Class B ; (c) Class C.

Class A networks have 7 bits for the network part and 24 bits for the host part.There can be only 126 (0 and 127 are reserved) Class A networks. Each of them can accommodate 2^{24} -2 hosts. Class A addresses were designed for large organizations with large number of hosts or routers attached to their network. Class B networks have 14 bits for the network part and16 bits for the host part. Class B networks can accommodate 65,534 hosts. Class B addresses were designed for midsize organizations that may have tens of thousands of hosts or routers attached to their networks. Class C networks have 8 bits for the network part and 21 bits for the host part. Class C networks can accommodate only 256 hosts. Class C addresses were designed for small organizations with a small number of hosts or routers attached to their network. There is just one block of Class D addresses, which is designed for multicasting. There is just one block of Class E addresses, which is designed for use as reserved addresses.

3.3 ADDRESS MAPPING

An internet is made of a combination of physical networks connected by internetworking devices such as routers. A packet starting from a source host may pass through several different physical networks before finally reaching the destination host. The hosts and routers are recognized at the network level by their logical (IP) addresses. However, packets pass through physical networks to reach these hosts and routers. At the physical level, the hosts and routers are recognized by their physical addresses.

A physical address is a local address. Its jurisdiction is a local network. It must be unique locally, but is not necessarily unique universally. It is called a *physical* address because it is usually (but not always) implemented in hardware. An example of a physical address is the 48-bit MAC address in the Ethernet protocol, which is imprinted on the NIC installed in the host or router.

The physical address and the logical address are two different identifiers. We need both because a physical network such as Ethernet can have two different protocols at the network layer such as IP and IPX (Novell) at the same time. Likewise, a packet at a network layer such as IP may pass through different physical networks such as Ethernet and LocalTalk (Apple). This means that delivery of a packet to a host or a router requires two levels of addressing: logical and physical. We need to be able to **map a logical address to its corresponding physical address and vice versa.** These can be done by using either **static or dynamic mapping**.

Static mapping involves in the creation of a table that associates a logical address with a physical address. This table is stored in each machine on the network. Each machine that knows, for example, the IP address of another machine but not its physical address can look it up in the table. This has some limitations because physical addresses may change in the following ways:

1. A machine could change its NIC, resulting in a new physical address.
2. In some LANs, such as LocalTalk, the physical address changes every time the computer is turned on.
3. A mobile computer can move from one physical network to another, resulting in a change in its physical address.

To implement these changes, a static mapping table must be updated periodically. This overhead could affect network performance.

In dynamic mapping each time a machine knows one of the two addresses (logical or physical), it can use a protocol to find the other one.

3.3.1 ARP- Address Resolution Protocol

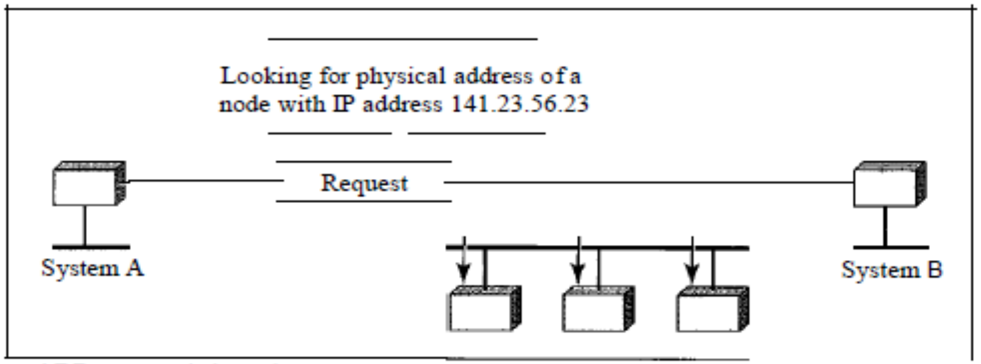
Mapping Logical to Physical Address: ARP

Anytime a host or a router has an IP datagram to send to another host or router, it has the logical (IP) address of the receiver. The logical (IP) address is obtained from the DNS if the sender is the host or it is found in a routing table if the sender is a router. But the IP datagram must be encapsulated in a frame to be able to pass through the physical network. This means that the sender needs the physical address of the receiver.

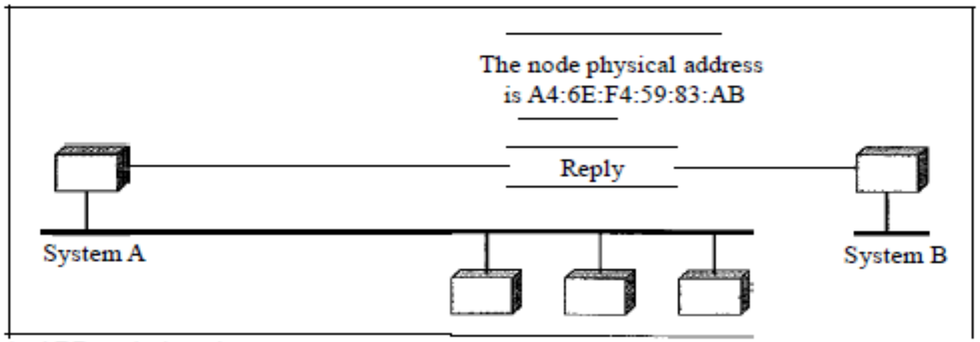
The host or the router sends an ARP query packet. The packet includes the physical and IP addresses of the sender and the IP address of the receiver. Because the sender does not know the physical address of the receiver, the query is broadcast over the network (see Figure 3.3.1). Every host or router on the network receives and processes the ARP query packet, but only the intended recipient recognizes its IP address and sends back an ARP response packet. The response packet contains the recipient's IP and physical addresses. The packet is unicast directly to the inquirer by using the physical address received in the query packet.

In Figure 3.3.1 a, the system on the left (A) has a packet that needs to be delivered to another system (B) with IP address 141.23.56.23. System A needs to pass the packet to its data link layer for the actual delivery, but it does not know the physical address of the recipient. It uses the services of ARP by asking the ARP protocol to send a broadcast ARP request packet to ask for the physical address of a system with an IP address of 141.23.56.23.

This packet is received by every system on the physical network, but only system B will answer it, as shown in Figure 3.3.1 b. System B sends an ARP reply packet that includes its physical address. Now system A can send all the packets it has for this destination by using the physical address it received.


Looking for physical address of a node with IP address 141.23.56.23
Request
System A
System B

a. ARP request is broadcast



b. ARP reply is unicast

Figure 3.3.1 *ARP operation*

Cache Memory

Using ARP is inefficient if system A needs to broadcast an ARP request for each IP packet it needs to send to system B. It could have broadcast the IP packet itself. ARP can be useful if the ARP reply is cached (kept in cache memory for a while) because a system normally sends several packets to the same destination. A system that receives an ARP reply stores the mapping in the cache memory and keeps it for 20 to 30 minutes unless the space in the cache is exhausted. Before sending an ARP request, the system first checks its cache to see if it can find the mapping.

Packet Format

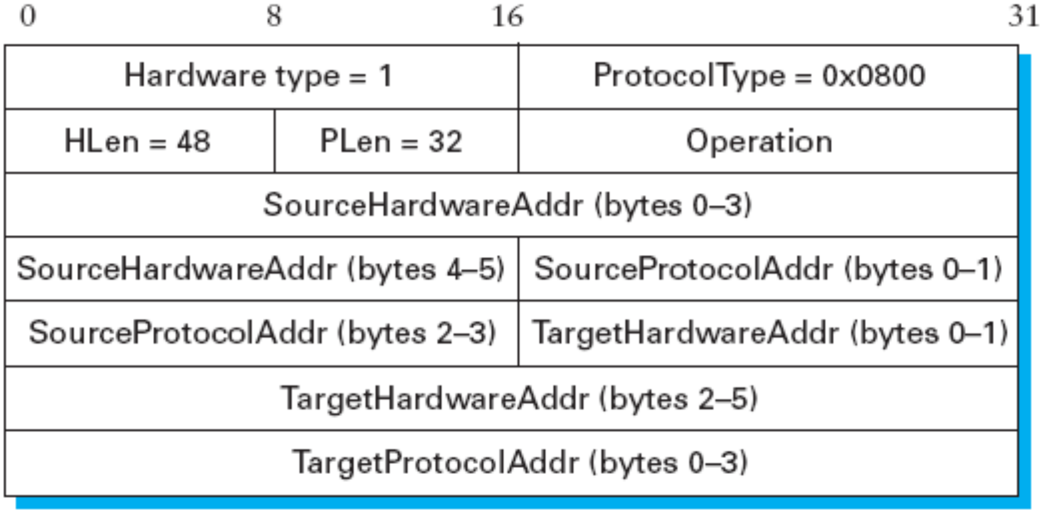

0
8
16
31
Hardware type = 1
ProtocolType = 0x0800
HLen = 48
PLen = 32
Operation
SourceHardwareAddr (bytes 0–3)
SourceHardwareAddr (bytes 4–5)
SourceProtocolAddr (bytes 0–1)
SourceProtocolAddr (bytes 2–3)
TargetHardwareAddr (bytes 0–1)
TargetHardwareAddr (bytes 2–5)
TargetProtocolAddr (bytes 0–3)

Figure 3.3.2 shows the format of an ARP packet for mapping IP address into Ethernet addresses.

The fields are as follows:

- **Hardware type :** This is a 16-bit field defining the type of the network on which ARP is running. Each LAN has been assigned an integer based on its type. For example, Ethernet is given type 1. ARP can be used on any physical network.
- **Protocol type :** This is a 16-bit field defining the protocol. For example, the value of this field for the IPv4 protocol is 080016, ARP can be used with any higher-level protocol.
- **HLen(“ hardware” address length) and PLen(“ protocol” address length):** These fields specify the length of the link-layer address and higher-layer protocol address respectively.
- **Operation**. This field specifies whether this is a request or a response.
- **The Source and Target hardware(Ethernet) and protocol(IP) addresses**.

Encapsulation

An ARP packet is encapsulated directly into a data link frame. For example, an ARP packet can be encapsulated in an Ethernet frame. Note that the type field indicates that the data carried by the frame are an ARP packet.

Operation

Let us see how ARP functions on a typical internet. First we describe the steps involved.
Then we discuss the four cases in which a host or router needs to use ARP. These are the
steps involved in an ARP process:

1. The sender knows the IP address of the target. We will see how the sender obtains this shortly.
2. IP asks ARP to create an ARP request message, filling in the sender physical address, the sender IP address, and the target IP address. The target physical address field is filled with Os.
3. The message is passed to the data link layer where it is encapsulated in a frame by using the physical address of the sender as the source address and the physical broadcast address as the destination address.
4. Every host or router receives the frame. Because the frame contains a broadcast destination address, all stations remove the message and pass it to ARP. All machines except the one targeted drop the packet. The target machine recognizes its IP address.
5. The target machine replies with an ARP reply message that contains its physical address. The message is unicast.
6. The sender receives the reply message. It now knows the physical address of the target machine.
7. The IP datagram, which carries data for the target machine, is now encapsulated in a frame and is unicast to the destination.

ProxyARP

A technique called *proxy* ARP is used to create a subnetting effect. A proxy ARP is an ARP that acts on behalf of a set of hosts. Whenever a router running a proxy ARP receives an ARP request looking for the IP address of one of these hosts, the router sends an ARP reply announcing its own hardware (physical) address. After the router receives the actual IP packet, it sends the packet to the appropriate host or router.

3.4 Mapping Physical to Logical Address: RARP, BOOTP, and DHCP .

There are occasions in which a host knows its physical address, but needs to know its logical address. This may happen in two cases:

1. A diskless station is just booted. The station can find its physical address by checking its interface, but it does not know its IP address.
2. An organization does not have enough IP addresses to assign to each station; it needs to assign IP addresses on demand. The station can send its physical address and ask for a short time lease.

3.4.1 RARP

Reverse Address Resolution Protocol (RARP) finds the logical address for a machine that knows only its physical address. Each host or router is assigned one or more logical (IP) addresses, which are unique and independent of the physical (hardware) address of the machine. To create an IP datagram, a host or a router needs to know its own IP address or addresses. The IP address of a machine is usually read from its configuration file stored on a disk file.

However, a diskless machine is usually booted from ROM, which has minimum booting information. The ROM is installed by the manufacturer. It cannot include the IP address because the IP addresses on a network are assigned by the network administrator. The machine can get its physical address (by reading its NIC, for example), which is unique locally. It can then use the physical address to get the logical address by using the RARP protocol. A RARP request is created and broadcast on the local network. Another machine on the local network that knows all the IP addresses will respond with a RARP reply. The requesting machine must be running a RARP client program; the responding machine must be running a RARP server program.

There is a serious problem with RARP: Broadcasting is done at the data link layer. The physical broadcast address, all is in the case of Ethernet, does not pass the boundaries of a network. This means that if an administrator has several networks or several subnets, it needs to assign a RARP server for each network or subnet. This is the reason that RARP is almost obsolete. Protocols like BOOTP and DHCP, are replacing RARP.

BOOTP

The Bootstrap Protocol (BOOTP) is a client/server protocol designed to provide physical address to logical address mapping. BOOTP is an application layer protocol. The administrator may put the client and the server on the same network or on different networks .BOOTP messages are encapsulated in a UDP packet, and the UDP packet itself is encapsulated in an IP packet.

3.5 DHCP- Dynamic Host Configuration Protocol

DHCP dynamically assigns IP addresses to hosts. That is DHCP allows addresses to be “leased” for some period of time. Once the lease expires, the server is free to return that address to its pool.

DHCP relies on the existence of a DHCP server that is responsible for providing configuration information to hosts. Since the goal of DHCP is to minimize the amount of manual configuration required for a host to function, it would rather defeat the purpose if each host had to be configured with the address of a DHCP server. Thus, the first problem faced by DHCP is that of server discovery.

To contact a DHCP server, a newly booted or attached host sends a DHCPDISCOVER message to a special IP address (255.255.255.255) that is an IP broadcast address. This means it will be received by all hosts and routers on that network. (Routers do not forward such packets onto other networks, preventing broadcast to the entire Internet.) In the simplest case, one of these nodes is the DHCP server for the network. The server would then reply to the host that generated the discovery message (all the other nodes would ignore it). However, it is not really desirable to require one DHCP server on every network because this still creates a potentially large number of servers that need to be correctly and consistently configured.

Thus, DHCP uses the concept of a *relay agent*. There is at least one relay agent on each network, and it is configured with just one piece of information: the IP address of the DHCP server. When a relay agent receives a DHCPDISCOVER message, it unicasts it to the DHCP server and awaits the response, which it will then send back to the requesting client. The process of relaying a message from a host to a remote DHCP server is shown in Figure 3.5.1.

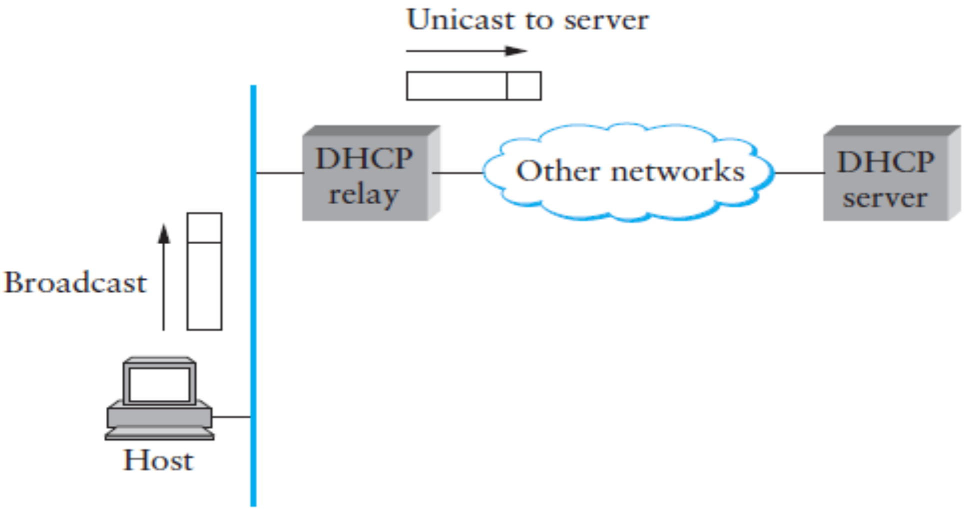
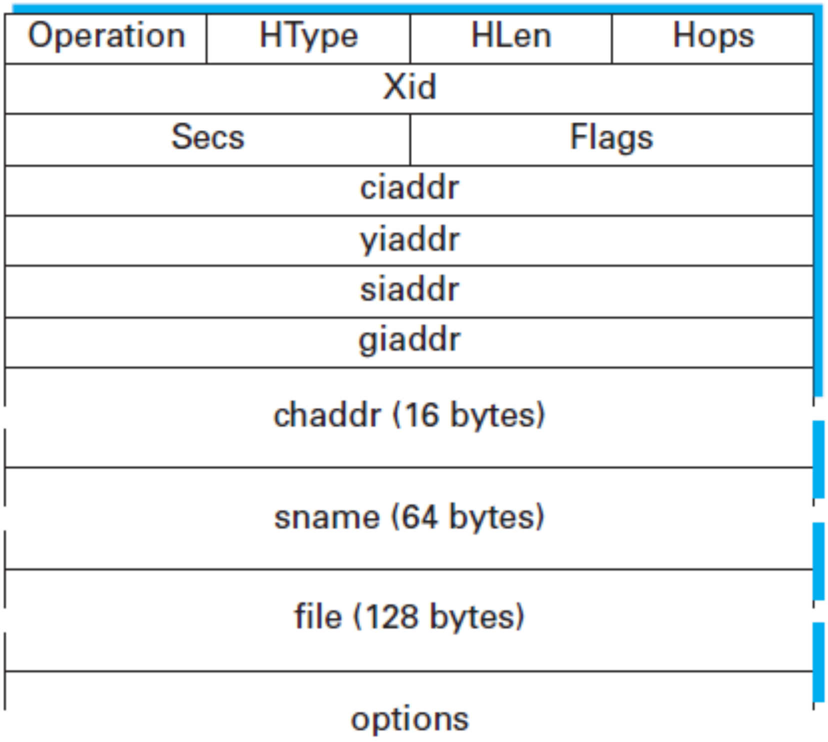


Fig 3.5.1 A DHCP relay agent receives a broadcast DHCPDISCOVER message from a host and sends a unicast DHCPDISCOVER to the DHCP server



Operation	HType	HLen	Hops
Xid			
Secs		Flags	
ciaddr			
yiaddr			
siaddr			
giaddr			
chaddr (16 bytes)			
sname (64 bytes)			
file (128 bytes)			
options			

Fig 3.5.2 DHCP Packet Format

Figure 3.5.2 shows the format of a DHCP message. The message is actually sent using a protocol called UDP (the User Datagram Protocol) that runs over IP. DHCP is derived from an earlier protocol called BOOTP, and some of the packet fields are thus not strictly relevant to host configuration. When trying to obtain configuration information, the client puts its hardware address (e.g., its Ethernet address) in the chaddr field. The DHCP server replies by filling in the yiaddr ("your" IP address) field and sending it to the client. Other information such as the default router to be used by this client can be included in the options field.

In the case where DHCP dynamically assigns IP addresses to hosts, it is clear that hosts cannot keep addresses indefinitely, as this would eventually cause the server to exhaust its address pool. At the same time, a host cannot be depended upon to give back its address, since it might have crashed, been unplugged from the network, or been turned off. Thus, DHCP allows addresses to be "leased" for some period of time. Once the lease expires, the server is free to return that address to its pool. A host with a leased address clearly needs to renew the lease periodically if in fact it is still connected to the network and functioning correctly. Note that DHCP may also introduce some more complexity into network management, since it makes the binding between physical hosts and IP addresses much more dynamic.

3.6 ICMP- Internet Control Message Protocol

When a router does not know how to forward the datagram or when one fragment of a datagram fails to arrive at the destination—it does not necessarily fail silently. IP is always configured with a companion protocol, known as the Internet Control Message Protocol (ICMP).

ICMP defines a collection of error messages that are sent back to the source host whenever a router or host is unable to process an IP datagram successfully. For example, ICMP defines error messages indicating that the destination host is unreachable (perhaps due to a link failure), that the reassembly process failed, that the TTL had reached 0, that the IP header checksum failed, and so on.

ICMP also defines a handful of control messages that a router can send back to a source host. One of the most useful control messages, called an ICMP-Redirect, tells the source host that there is a better route to the destination.