Chapter 3

TCP/IP Networks

3.1 Internet Protocol version 4 (IPv4)

Internet Protocol version 4 is the fourth iteration of the Internet Protocol (IP) and it is the first version of the protocol to be widely deployed. IPv4 is the dominant network layer protocol on the Internet and apart from IPv6 it is the only protocol used on the Internet.

It is described in IETF RFC 791 (September 1981) which made obsolete RFC 760 (January 1980). The United States Department of Defense also standardized it as MIL-STD-1777.

IPv4 is a data-oriented protocol to be used on a packet switched internetwork (e.g., Ethernet). It is a best effort protocol in that it doesn't guarantee delivery. It doesn't make any guarantees on the correctness of the data; it may result in duplicated packets and/or packets out-of-order. These aspects are addressed by an upper layer protocol (e.g., TCP, and partly by UDP).

The entire purpose of IP is to provide unique global computer addressing to ensure that two computers communicating over the internet can uniquely identify one another.

3.1.1 IP Addressing

Before discussing IP addressing, we need to say a few words about hosts and routers. A host (also called an end system) has one link into the network. When IP in the host wants to send a datagram, it passes the datagram to its link. The boundary between the host and the link is called the interface. A router is fundamentally different from a host in that it has two or more links that connect to it. When a router forwards a datagram, it forwards the datagram over one of its links. The boundary between the router and any one of its links is also called an interface. Thus, a router has multiple interfaces, one for each of its links. Because every interface (for a host or router) is capable of sending and receiving IP datagrams, IP requires each interface to have an IP address.

Each IP address is 32 bits long (equivalently, four bytes) long. IP addresses are typically written in so-called "dot-decimal notation", whereby each byte of the address is written in its decimal form and is separated by a period. For example, a typical IP address would be 193.32.216.9. The 193 is the decimal equivalent for the first 8 bits of the address; the 32 is the decimal equivalent for the second 8 bits of the address, etc. Thus, the address 193.32.216.9 in binary notation is:
11000001 00100000 11011000 00001001

(A space as been added between the bytes for visual purposes.) Because each IP address is 32 bits long, there are 2^{32} possible IP addresses.

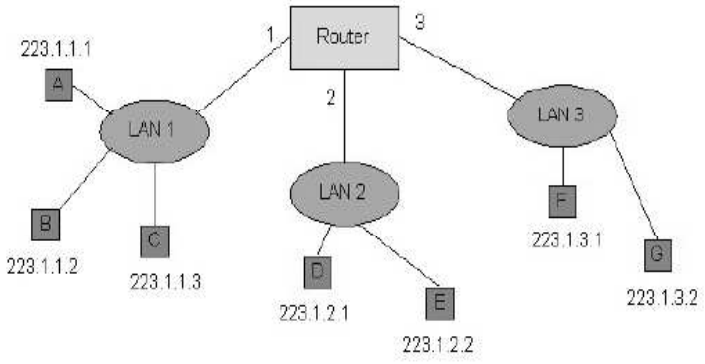

Figure 3.1: LANs are networks in IP jargon

Figure-3.1 provides an example of IP addressing and interfaces. In this figure there is one router which interconnects three LANs. (LANs, also known as local area networks, were briefly discussed in Chapter 1 and will be studied in detail in the next chapter.) In the jargon of IP, each of these LANs is called an IP network or more simply a **"network"**. There are several things to observe from this diagram. First, the router has threes interfaces, labeled 1, 2 and 3. Each of the router interfaces has its own IP address, which are provided in Table-3.1 each host also has its own interface and IP address. Second, all of the interfaces attached to LAN 1, including a router interface, have an IP address of the form 223.1.1.xxx . Similarly, all the interfaces attached to LAN 2 and LAN 3 have IP addresses of the form 223.1.2.xxx and 233.1.3.xxx, respectively. In other words, each address has two parts: the first part (the first three bytes in this example) that specifies the network; and the second part (the last byte in this example) that addresses a specific host on the network.

For a general interconnected system of routers and hosts (such as the Internet), we use the following recipe to define the "networks" in the system. We first detach each router interface from its router and each host interface from its host. This creates "islands" of isolated networks, with "interfaces" terminating all the leaves

Router Interface	IP Address
1	223.1.1.4
2	223.1.2.9
3	223.1.3.27

Table 3.1: IP addresses for router interfaces.

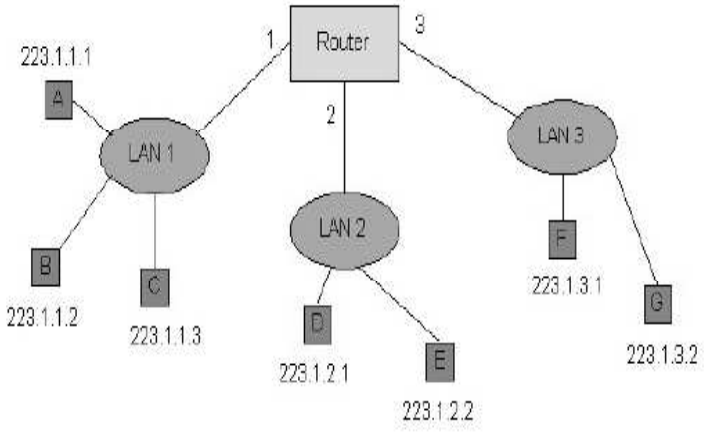

Figure 3.2: An interconnected system consisting of six networks

of the isolated networks . We then call each of these isolated networks a network. Indeed, if we apply this procedure to the interconnected system in Figure-3.2, we get six islands or "networks". The current Internet consists of millions of networks. (In the next chapter we will consider bridges. We mention here that when applying this recipe, we do not detach interfaces from bridges. Thus each bridge lies within the interior of some network.)

Now that we have defined a network, we are ready to discuss IP addressing in more detail. IP addresses are globally unique, that is, no two interfaces in the world have the same IP address. Figure 4.4-3 shows the four possible formats of an IP address. (A fifth address, beginning with 11110, is reserved for future use.) In general, each interface (for a host or router) belongs to a network; the network part of the address identifies the network to which the interface belongs. The host part identifies the specific interface within the network. (We would prefer to use the terminology "interface part of the address" rather than "host part of the address" because IP address is really for an interface and not for a host; but the terminology "host part" is commonly used in practice.) For a class A address, the first 8 bits identify the network, and the last 24 bits identify the interface within that network. Thus with a class A we can have up to 27 networks (the first of the eight bits is fixed as 0) and and 224 interfaces. Note that the interfaces in Figures-3.1 and 3.2 use class A addresses. The class B address space allows for 214 networks, with up to 216 interfaces within each network. A class C address uses 21 bits to identify the network and leaves only 8 bits for the interface identifier. Class D addresses are reserved for so-called multicast addresses. As we will see later that, these addresses do not identify a specific interface but rather provide a mechanism through which multiple hosts can receive a copy of each single packet sent by a sender.

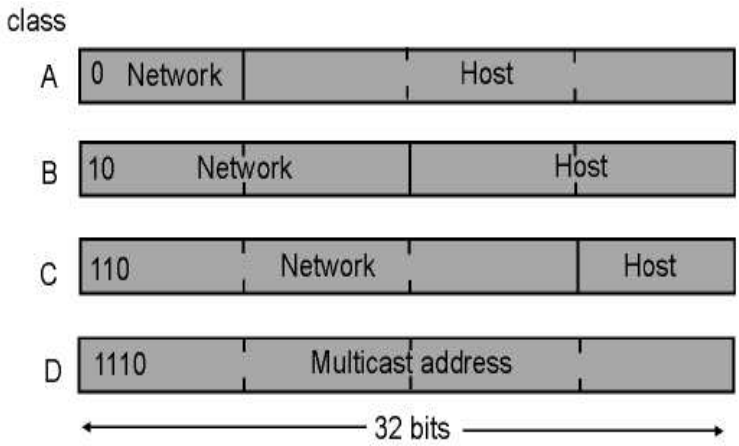

Figure 3.3: IPv4 address formats

Assigning Addresses

Having introduced IP addressing, one question that immediately comes to mind is how does a host get its own IP address? We have just learned that an IP address has two parts, a network part and a host part. The host part of the address can be assigned in several different ways, including:

- **Manual configuration** The IP address is configured into the host (typically in a file) by the system administrator.
- **Dynamic Host Configuration Protocol (DHCP)** (RFC 2131). DHCP is an extension of the BOOTP (RFC 1542) protocol, and is sometimes referred to as Plug and Play. With DHCP, a DHCP server in a network (e.g., in a LAN) receives DHCP requests from a client and in the case of dynamic address allocation, allocates an IP address back to the requesting client. DHCP is used extensively in LANs and in residential Internet access.

The network part of the address is the same for all the hosts in the network. To obtain the network part of the address for a network, the network administrator might

first contact the network's ISP, which would provide addresses from a larger block of addressees that have already been allocated to the ISP. But how does an ISP get a block of addresses? IP addresses are managed under the authority of the Internet Assigned Numbers Authority (IANA), under the guidelines set forth in [RFC 2050]. The actual assignment of addresses is now managed by regional Internet registries. As of mid-1998, there are three such regional registries: the American Registry for Internet Number (ARIN, which handles registrations for North and South America, as well as parts of Africa. ARIN has recently taken over a number of the functions previously provided by Network Solutions), the Reseaux IP Europeans (RIPE, which covers Europe and nearby countries), and the Asia Pacific Network Information Center (APNIC).

Before leaving our discussion of addressing, we want to mention that mobile hosts may change the network to which they are attached, either dynamically while in motion or on a longer time scale. Because routing is to a network first, and then to a host within the network, this means that the mobile host's IP address must change when the host changes networks. Techniques for handling such issues are now under development within the IETF and the research community (RFC2002) (RFC2131).

Datagram Format

The IPv4 datagram format is shown in Figure-3.4

- **Version Number** These 4 bits specify the IP protocol version of the datagram. By looking at the version number, the router can then 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 Figure-3.4. The datagram format for the "new" version

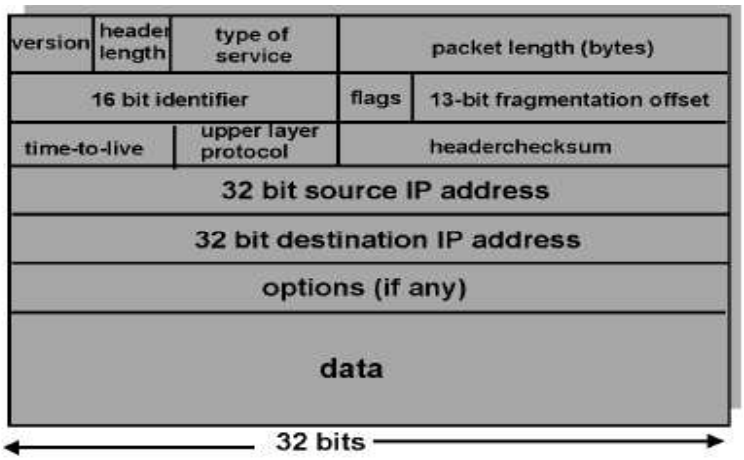

Figure 3.4: IPv4 datagram format

of IP (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.

- **TOS** The type of service (TOS) bits were included in the IPv4 header to allow different "types" of IP datagrams to be distinguished from each other, presumably so that they could be handled differently in times of overload. When the network is overloaded, for example, it would be useful to be able to distinguish network control datagrams from datagrams carrying data (e.g., HTTP messages). It would also be useful to distinguish real-time datagrams (e.g., used by an IP telephony application) from non-real-time traffic (e.g., FTP). More recently, one major routing vendor (Cisco) interprets the first three ToS bits as defining differential levels of service that can be provided by the router. The specific level of service to be provided is a policy issue determined by the

router's administrator. We shall explore the topic of differentiated service in detail in Chapter 6.

- **Datagram Length** This is the total length of the IP datagram (header plus data) measured in bytes. Since this field is 16 bits long, the theoretical maximum size of the IP datagram to 65,535 bytes. However, datagrams are rarely greater than 1500 bytes, and are often limited in size to 576 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, simply does not allow for fragmentation.

- **Time-to-live** The time-to-live (TTL) field is included to insure that datagrams do not circulate forever (due to, for example, a long lived router 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 only used when an IP datagram reaches its final destination. The value of this field indicates the transport-layer protocol at the destination to which the data portion of this IP datagram will 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. For a listing of all possible numbers, see [RFC 1700]. Note that the the protocol number in the IP datagram has a role that is fully analogous to the role of the port number field in the transport-layer segment. The protocol number is the "glue" that holds the network and transport layers together, whereas port number is the "glue" that holds the transport and application layers together. We will see in Chapter 5 that the link layer frame also has a special field which glues the link layer to the network layer.

- **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 1's complement arithmetic. As discussed in Section 3.3, the 1's complement of this sum, known as the Internet checksum, is stored in the checksum field. A router computes the Internet checksum for each received IP datagram and detects an error condition if the checksum carried in the datagram does not equal the computed checksum. Routers typically discard datagrams for which an error has been detected. Note that the checksum must be recomputed and restored at each router, as the TTL field, and possibly options fields as well, may change. An interesting discussion of fast algorithms for computing the Internet checksum is [1071]. A question often asked at this point is, why does TCP/IP perform error checking at both the transport and network layers? There are many reasons for this. First, routers are not required to perform error checking, so the transport layer cannot count on the network layer to do the job. Second, TCP/ UDP and IP do not necessarily have to both belong to the same protocol stack. TCP can, in principle, run over a different protocol (e.g., ATM) and IP can carry data without passing through TCP/UDP (e.g., RIP data).

- **Source and Destination IP Address** These fields carry the 32 bit IP address of the source and final destination for this IP datagram. The use and importance of the destination address is clear. The source IP address (along with the source and destination port numbers) is used at the destination host to direct the application data in the proper socket.

- **Options** The optional options fields allows 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. However, the mere existence of options does complicate matters – since datagram headers can be of variable length, one can not determine a priori where the data field will start. Also, since some datagrams may require options processing and others may not, the amount of time needed to process a IP datagram can vary greatly. These considerations become particularly important for IP processing in high performance routers and hosts. For these reasons and others, IP options were dropped in the IPv6 header.

- **Data (payload)** Finally, we come to the last, and most important field - the for the datagram in the first place! 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 ICMP messages

Note that IP datagram has a total of 20 bytes of header (assuming it has no options). If the IP datagram carries a TCP segment, then each (non-fragmented) datagram carries a total of 40 bytes of header (20 IP bytes and 20 TCP bytes) along with the application-layer data.

3.1.2 IP Fragmentation and Reassembly

Some protocols can carry "big" packets whereas other protocols can only carry "little" packets. For example, Ethernet packets can carry no more than 1500 bytes of data, whereas packets for many wide-area links can carry no more than 576 bytes. The maximum amount of data that a link-layer packet can carry is called the MTU (maximum transfer unit). Because each IP datagram is encapsulated within the link-layer packet 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. Having a hard limit on the size of an IP datagram is not much of a problem. What is a

problem is that each of the links along the route between sender and destination can use different link-layer protocols, and each of these protocols can have different MTUs.

To understand the problem better, imagine that you are a router that interconnects several links, each running different link layer protocols with different MTUs. Suppose you receive an IP datagram from one link, you check your routing table to determine the outgoing link, and this outgoing link has an MTU that is smaller than the length of the IP datagram. Time to panic – how are you going to squeeze this over sized IP packet into the payload field of the link-layer packet? The solution to this problem is to "fragment" the data in the IP datagram among two or more smaller IP datagrams, and then send these smaller datagrams over the outgoing link. Each of these smaller datagrams is referred to as a fragment.

Fragments need to be reassembled before they reach the transport layer at the destination. Indeed, both TCP and UDP are expecting to receive from the network layer complete, un-fragmented segments. The designers of IPv4 felt that reassembling (and possibly re-fragmenting) datagrams in the routers would introduce significant complication into the protocol and put a damper on router performance. (If you were a router, would you want to be reassembling fragments on top of everything else you have to do?) Sticking to end-to-end principle for the Internet, the designers of IPv4 decided to put the job of datagram reassembly in the end systems rather than in the network interior.

When a destination host receives a series of datagrams from the same source, it needs to determine if any of these datagrams are fragments of some "original" bigger datagram. If it does determine that some datagrams are fragments, it must

further determine when it has received the last fragment and how the fragments it has received should be pieced back together to form the original datagram. To allow the destination host to perform these reassembly tasks, the designers of of IP (version 4) put identification, flag and fragmentation fields in the IP datagram. When a datagram is created, the sending host stamps the datagram with an identification number as well as a source and destination address. The sending host increments the identification number for each datagram it sends. When a router needs to fragment a datagram, each resulting datagram (i.e., "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 are actually fragments of the same bigger datagram. 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 this flag bit set to 1. Also, in order for the destination host to determine if a fragment is missing (and also to be able to reassemble the fragments in the proper order), the offset field is used to specify where the fragment fits within the original IP datagram. This bit is set to 1 in all except the last fragment.

Figure-3.5 illustrates an example. A datagram 4,000 bytes arrives to a router, and this datagram must be forwarded to a link with a MTU of 1500 bytes. These implies that the 3,980 data bytes in the original datagram must be allocated to three separate fragments (each of which are also IP datagrams). Suppose that the original datagram is stamped with an identification number of 777. Then the characteristics of the three fragments are as follows:

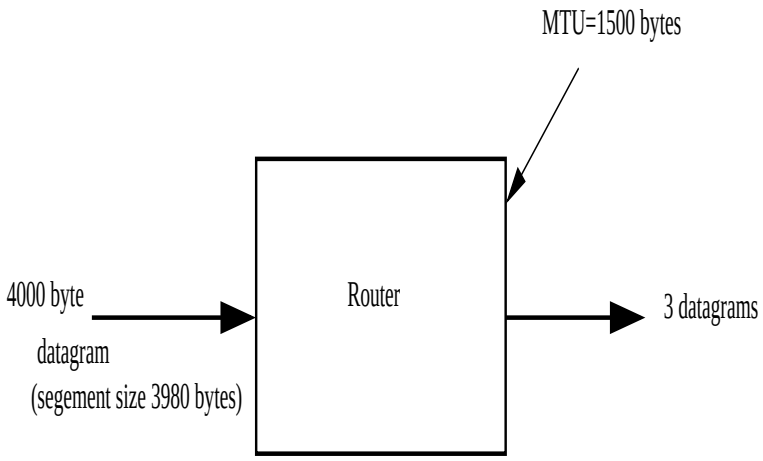

Figure 3.5: IP fragmentation

- 1st fragment
  - 1480 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
  - 1480 byte information field
  - identification = 777
  - offset = 1,480 (meaning the data should be inserted beginning at byte 1,480
  - flag = 1 (meaning there is more)
- 3rd fragment
  - 1020 byte (=3980-1480-1480) information field

- identification = 777
- offset = 2,960 (meaning the data should be inserted beginning at byte 2,960)
- flag = 0 (meaning this is the last fragment)

The payload of the datagram is only passed to the transport layer once the IP layer has fully reconstructed the original IP datagram. If one or more of the fragments does not arrive to the destination, the datagram is "lost" and not passed to the transport layer. But, as we learned in the previous chapter, if TCP is being used at the transport layer, then TCP will recover from this loss by having the source retransmit the data in the original datagram.

Fragmentation and reassembly puts an additional burden on Internet routers (the additional effort to create fragments out of a datagram) and on the destination hosts (the additional effort to reassembly fragments). For this reason it is desirable to keep fragmentation to a minimum. This is often done by limiting the TCP and UDP segments to a relatively small size, so that the fragmentation of the corresponding datagrams is unlikely. Because all data link protocols supported by IP are supposed to have MTUs of at least 576 bytes, fragmentation can be entirely eliminated by using a MSS of 536 bytes, 20 bytes of TCP segment header and 20 bytes of IP datagram header. This is why most TCP segments for bulk data transfer (such as with HTTP) are 512- 536 bytes long. (You may have noticed while surfing the Web that 500 or so bytes of data often arrive at a time.)

3.1.3 Routing

In order to transfer packets from a sending host to the destination host, the network layer must determine the path or route that the packets are to follow. Whether the