3

IP: Internet Protocol

3.1 Introduction

IP is the workhorse protocol of the TCP/IP protocol suite. All TCP, UDP, ICMP, and IGMP data gets transmitted as IP datagrams (Figure 1.4). A fact that amazes many newcomers to TCP/IP, especially those from an X.25 or SNA background, is that IP provides an unreliable, connectionless datagram delivery service.

By *unreliable* we mean there are no guarantees that an IP datagram successfully gets to its destination. IP provides a best effort service. When something goes wrong, such as a router temporarily running out of buffers, IP has a simple error handling algorithm: throw away the datagram and try to send an ICMP message back to the source. Any required reliability must be provided by the upper layers (e.g., TCP).

The term *connectionless* means that IP does not maintain any state information about successive datagrams. Each datagram is handled independently from all other datagrams. This also means that IP datagrams can get delivered out of order. If a source sends two consecutive datagrams (first A, then B) to the same destination, each is routed independently and can take different routes, with B arriving before A.

In this chapter we take a brief look at the fields in the IP header, describe IP routing, and cover subnetting. We also look at two useful commands: `ifconfig` and `netstat`. We leave a detailed discussion of some of the fields in the IP header for later when we can see exactly how the fields are used. RFC 791 [Postel 1981a] is the official specification of IP.

3.2 IP Header

Figure 3.1 shows the format of an IP datagram. The normal size of the IP header is 20 bytes, unless options are present.

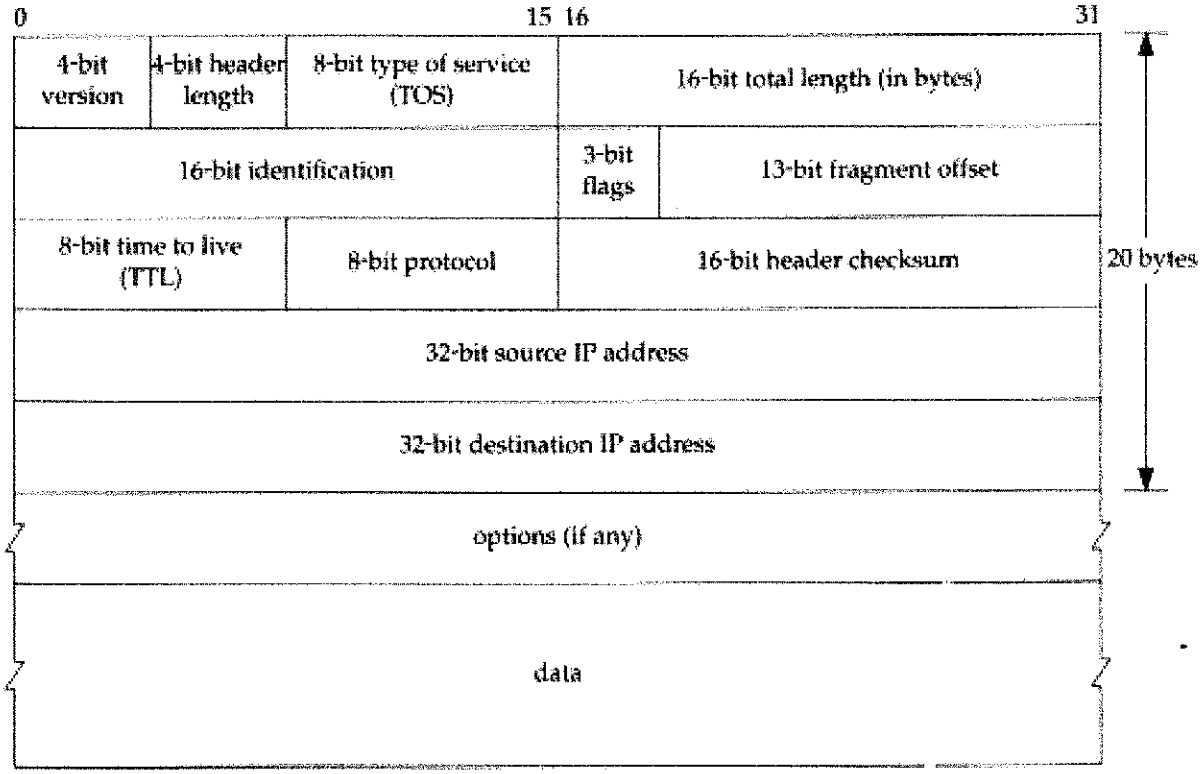

Figure 3.1 IP datagram, showing the fields in the IP header

We will show the pictures of protocol headers in TCP/IP as in Figure 3.1. The most significant bit is numbered 0 at the left, and the least significant bit of a 32-bit value is numbered 31 on the right.

The 4 bytes in the 32-bit value are transmitted in the order: bits 0–7 first, then bits 8–15, then 16–23, and bits 24–31 last. This is called *big endian* byte ordering, which is the byte ordering required for all binary integers in the TCP/IP headers as they traverse a network. This is called the *network byte order*. Machines that store binary integers in other formats, such as the *little endian* format, must convert the header values into the network byte order before transmitting the data.

The current protocol *version* is 4, so IP is sometimes called IPv4. Section 3.10 discusses some proposals for a new version of IP.

The *header length* is the number of 32-bit words in the header, including any options. Since this is a 4-bit field, it limits the header to 60 bytes. In Chapter 8 we'll see that this limitation makes some of the options, such as the record route option, useless today. The normal value of this field (when no options are present) is 5.

The *type-of-service* field (TOS) is composed of a 3-bit precedence field (which is ignored today), 4 TOS bits, and an unused bit that must be 0. The 4 TOS bits are: minimize delay, maximize throughput, maximize reliability, and minimize monetary cost.

Only 1 of these 4 bits can be turned on. If all 4 bits are 0 it implies normal service. RFC 1340 [Reynolds and Postel 1992] specifies how these bits should be set by all the standard applications. RFC 1349 [Almquist 1992] contains some corrections to this RFC, and a more detailed description of the TOS feature.

Figure 3.2 shows the recommended values of the TOS field for various applications. In the final column we show the hexadecimal value, since that's what we'll see in the `tcpdump` output later in the text.

Application	Minimize delay	Maximize throughput	Maximize reliability	Minimize monetary cost	Hex value
Telnet/Rlogin	1	0	0	0	0x10
FTP					
control	1	0	0	0	0x10
data	0	1	0	0	0x08
any bulk data	0	1	0	0	0x08
TFTP	1	0	0	0	0x10
SMTP					
command phase	1	0	0	0	0x10
data phase	0	1	0	0	0x08
DNS					
UDP query	1	0	0	0	0x10
TCP query	0	0	0	0	0x00
zone transfer	0	1	0	0	0x08
ICMP					
error	0	0	0	0	0x00
query	0	0	0	0	0x00
any IGP	0	0	1	0	0x04
SNMP	0	0	1	0	0x04
BOOTP	0	0	0	0	0x00
NNTP	0	0	0	1	0x02

Figure 3.2 Recommended values for type-of-service field.

The interactive login applications, Telnet and Rlogin, want a minimum delay since they're used interactively by a human for small amounts of data transfer. File transfer by FTP, on the other hand, wants maximum throughput. Maximum reliability is specified for network management (SNMP) and the routing protocols. Usenet news (NNTP) is the only one shown that wants to minimize monetary cost.

The TOS feature is not supported by most TCP/IP implementations today, though newer systems starting with 4.3BSD Reno are setting it. Additionally, new routing protocols such as OSPF and IS–IS are capable of making routing decisions based on this field.

In Section 2.10 we mentioned that SLIP drivers normally provide type-of-service queueing, allowing interactive traffic to be handled before bulk data. Since most implementations don't use the TOS field, this queueing is done ad hoc by SLIP, with the driver looking at the protocol field (to determine whether it's a TCP segment or not) and then checking the source and destination TCP port numbers to see if the port number corresponds to an interactive service. One driver comments that this "disgusting hack" is required since most implementations don't allow the application to set the TOS field.

The *total length* field is the total length of the IP datagram in bytes. Using this field and the header length field, we know where the data portion of the IP datagram starts, and its length. Since this is a 16-bit field, the maximum size of an IP datagram is 65535 bytes. (Recall from Figure 2.5 [p. 30] that a Hyperchannel has an MTU of 65535. This means there really isn't an MTU—it uses the largest IP datagram possible.) This field also changes when a datagram is fragmented, which we describe in Section 11.5.

Although it's possible to send a 65535-byte IP datagram, most link layers will fragment this. Furthermore, a host is not required to receive a datagram larger than 576 bytes. TCP divides the user's data into pieces, so this limit normally doesn't affect TCP. With UDP we'll encounter numerous applications in later chapters (RIP, TFTP, BOOTP, the DNS, and SNMP) that limit themselves to 512 bytes of user data, to stay below this 576-byte limit. Realistically, however, most implementations today (especially those that support the Network File System, NFS) allow for just over 8192-byte IP datagrams.

The total length field is required in the IP header since some data links (e.g., Ethernet) pad small frames to be a minimum length. Even though the minimum Ethernet frame size is 46 bytes (Figure 2.1), an IP datagram can be smaller. If the total length field wasn't provided, the IP layer wouldn't know how much of a 46-byte Ethernet frame was really an IP datagram.

The *identification* field uniquely identifies each datagram sent by a host. It normally increments by one each time a datagram is sent. We return to this field when we look at fragmentation and reassembly in Section 11.5. Similarly, we'll also look at the *flags* field and the *fragmentation offset* field when we talk about fragmentation.

> RFC 791 [Postel 1981a] says that the identification field should be chosen by the upper layer that is having IP send the datagram. This implies that two consecutive IP datagrams, one generated by TCP and one generated by UDP, can have the same identification field. While this is OK (the reassembly algorithm handles this), most Berkeley-derived implementations have the IP layer increment a kernel variable each time an IP datagram is sent, regardless of which layer passed the data to IP to send. This kernel variable is initialized to a value based on the time-of-day when the system is bootstrapped.

The *time-to-live* field, or *TTL*, sets an upper limit on the number of routers through which a datagram can pass. It limits the lifetime of the datagram. It is initialized by the sender to some value (often 32 or 64) and decremented by one by every router that handles the datagram. When this field reaches 0, the datagram is thrown away, and the sender is notified with an ICMP message. This prevents packets from getting caught in routing loops forever. We return to this field in Chapter 8 when we look at the Traceroute program.

We talked about the *protocol* field in Chapter 1 and showed how it is used by IP to demultiplex incoming datagrams in Figure 1.8. It identifies which protocol gave the data for IP to send.

The *header checksum* is calculated over the IP header only. It does *not* cover any data that follows the header. ICMP, IGMP, UDP, and TCP all have a checksum in their own headers to cover their header and data.

To compute the IP checksum for an outgoing datagram, the value of the checksum field is first set to 0. Then the 16-bit one's complement sum of the header is calculated (i.e., the entire header is considered a sequence of 16-bit words). The 16-bit one's

complement of this sum is stored in the checksum field. When an IP datagram is received, the 16-bit one's complement sum of the header is calculated. Since the receiver's calculated checksum contains the checksum stored by the sender, the receiver's checksum is all one bits if nothing in the header was modified. If the result is not all one bits (a checksum error), IP discards the received datagram. No error message is generated. It is up to the higher layers to somehow detect the missing datagram and retransmit.

ICMP, IGMP, UDP, and TCP all use the same checksum algorithm, although TCP and UDP include various fields from the IP header, in addition to their own header and data. RFC 1071 [Braden, Borman, and Partridge 1988] contains implementation techniques for computing the Internet checksum. Since a router often changes only the TTL field (decrementing it by 1), a router can incrementally update the checksum when it forwards a received datagram, instead of calculating the checksum over the entire IP header again. RFC 1141 [Mallory and Kullberg 1990] describes an efficient way to do this.

> The standard BSD implementation, however, does not use this incremental update feature when forwarding a datagram.

Every IP datagram contains the *source IP address* and the *destination IP address*. These are the 32-bit values that we described in Section 1.4.

The final field, the *options*, is a variable-length list of optional information for the datagram. The options currently defined are:

- security and handling restrictions (for military applications, refer to RFC 1108 [Kent 1991] for details),
- record route (have each router record its IP address, Section 7.3),
- timestamp (have each router record its IP address and time, Section 7.4),
- loose source routing (specifying a list of IP addresses that must be traversed by the datagram, Section 8.5), and
- strict source routing (similar to loose source routing but here only the addresses in the list can be traversed, Section 8.5).

These options are rarely used and not all host and routers support all the options.

The options field always ends on a 32-bit boundary. Pad bytes with a value of 0 are added if necessary. This assures that the IP header is always a multiple of 32 bits (as required for the *header length* field).

3.3 IP Routing

Conceptually, IP routing is simple, especially for a host. If the destination is directly connected to the host (e.g., a point-to-point link) or on a shared network (e.g., Ethernet or token ring), then the IP datagram is sent directly to the destination. Otherwise the

host sends the datagram to a default router, and lets the router deliver the datagram to its destination. This simple scheme handles most host configurations.

In this section and in Chapter 9 we'll look at the more general case where the IP layer can be configured to act as a router in addition to acting as a host. Most multiuser systems today, including almost every Unix system, can be configured to act as a router. We can then specify a single routing algorithm that both hosts and routers can use. The fundamental difference is that a host *never* forwards datagrams from one of its interfaces to another, while a router forwards datagrams. A host that contains embedded router functionality should never forward a datagram unless it has been specifically configured to do so. We say more about this configuration option in Section 9.4.

In our general scheme, IP can receive a datagram from TCP, UDP, ICMP, or IGMP (that is, a locally generated datagram) to send, or one that has been received from a network interface (a datagram to forward). The IP layer has a routing table in memory that it searches each time it receives a datagram to send. When a datagram is received from a network interface, IP first checks if the destination IP address is one of its own IP addresses or an IP broadcast address. If so, the datagram is delivered to the protocol module specified by the protocol field in the IP header. If the datagram is not destined for this IP layer, then (1) if the IP layer was configured to act as a router the packet is forwarded (that is, handled as an outgoing datagram as described below), else (2) the datagram is silently discarded.

Each entry in the routing table contains the following information:

- Destination IP address. This can be either a complete *host address* or a *network address*, as specified by the flag field (described below) for this entry. A host address has a nonzero host ID (Figure 1.5) and identifies one particular host, while a network address has a host ID of 0 and identifies all the hosts on that network (e.g., Ethernet, token ring).
- IP address of a *next-hop router*, or the IP address of a directly connected network. A next-hop router is one that is on a directly connected network to which we can send datagrams for delivery. The next-hop router is not the final destination, but it takes the datagrams we send it and forwards them to the final destination.
- Flags. One flag specifies whether the destination IP address is the address of a network or the address of a host. Another flag says whether the next-hop router field is really a next-hop router or a directly connected interface. (We describe each of these flags in Section 9.2.)
- Specification of which network interface the datagram should be passed to for transmission.

IP routing is done on a hop-by-hop basis. As we can see from this routing table information, IP does not know the complete route to any destination (except, of course, those destinations that are directly connected to the sending host). All that IP routing provides is the IP address of the next-hop router to which the datagram is sent. It is assumed that the next-hop router is really "closer" to the destination than the sending host is, and that the next-hop router is directly connected to the sending host.

IP routing performs the following actions:

1. Search the routing table for an entry that matches the complete destination IP address (matching network ID and host ID). If found, send the packet to the indicated next-hop router or to the directly connected interface (depending on the flags field). Point-to-point links are found here, for example, since the other end of such a link is the other host's complete IP address.
2. Search the routing table for an entry that matches just the destination network ID. If found, send the packet to the indicated next-hop router or to the directly connected interface (depending on the flags field). All the hosts on the destination network can be handled with this single routing table entry. All the hosts on a local Ethernet, for example, are handled with a routing table entry of this type.

   This check for a network match must take into account a possible subnet mask, which we describe in the next section.
3. Search the routing table for an entry labeled "default." If found, send the packet to the indicated next-hop router.

If none of the steps works, the datagram is undeliverable. If the undeliverable datagram was generated on this host, a "host unreachable" or "network unreachable" error is normally returned to the application that generated the datagram.

A complete matching host address is searched for before a matching network ID. Only if both of these fail is a default route used. Default routes, along with the ICMP redirect message sent by a next-hop router (if we chose the wrong default for a datagram), are powerful features of IP routing that we'll come back to in Chapter 9.

The ability to specify a route to a network, and not have to specify a route to every host, is another fundamental feature of IP routing. Doing this allows the routers on the Internet, for example, to have a routing table with thousands of entries, instead of a routing table with more than one million entries.

Examples

First consider a simple example: our host `bsdi` has an IP datagram to send to our host `sun`. Both hosts are on the same Ethernet (see inside front cover). Figure 3.3 shows the delivery of the datagram.

When IP receives the datagram from one of the upper layers it searches its routing table and finds that the destination IP address (140.252.13.33) is on a directly connected network (the Ethernet 140.252.13.0). A matching network address is found in the routing table. (In the next section we'll see that because of subnetting the network address of this Ethernet is really 140.252.13.32, but that doesn't affect this discussion of routing.)

The datagram is passed to the Ethernet device driver, and sent to `sun` as an Ethernet frame (Figure 2.1). The destination address in the IP datagram is Sun's IP address (140.252.13.33) and the destination address in the link-layer header is the 48-bit Ethernet address of `sun`'s Ethernet interface. This 48-bit Ethernet address is obtained using ARP, as we describe in the next chapter.

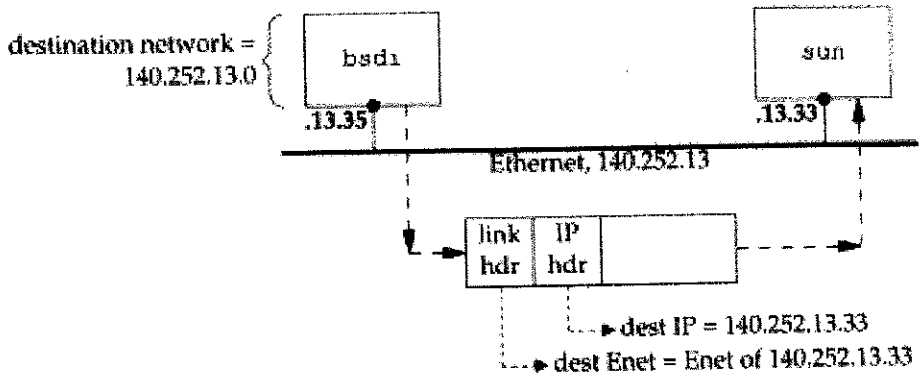

Figure 3.3 Delivery of IP datagram from `bsdi` to `sun`.

Now consider another example: `bsdi` has an IP datagram to send to the host `ftp.uu.net`, whose IP address is 192.48.96.9. Figure 3.4 shows the path of the datagram through the first three routers. First `bsdi` searches its routing table but doesn't find a matching host entry or a matching network entry. It uses its default entry, which tells it to send datagrams to `sun`, the next-hop router. When the datagram travels from `bsdi` to `sun` the destination IP address is the final destination (192.48.96.9) but the link-layer address is the 48-bit Ethernet address of `sun`'s Ethernet interface. Compare this datagram with the one in Figure 3.3, where the destination IP address and the destination link-layer address specified the same host (`sun`).

When `sun` receives the datagram it realizes that the datagram's destination IP address is not one of its own, and `sun` is configured to act as a router, so it forwards the datagram. Its routing table is searched and the default entry is used. The default entry on `sun` tells it to send datagrams to the next-hop router `netb`, whose IP address is 140.252.1.183. The datagram is sent across the point-to-point SLIP link, using the minimal encapsulation we showed in Figure 2.2. We don't show a link-layer header, as we do on the Ethernets, because there isn't one on a SLIP link.

When `netb` receives the datagram it goes through the same steps that `sun` just did: the datagram is not destined for one of its own IP addresses, and `netb` is configured to act as a router, so the datagram is forwarded. The default routing table entry is used, sending the datagram to the next-hop router `gateway` (140.252.1.4). ARP is used by `netb` on the Ethernet 140.252.1 to obtain the 48-bit Ethernet address corresponding to 140.252.1.4, and that Ethernet address is the destination address in the link-layer header.

`gateway` goes through the same steps as the previous two routers and its default routing table entry specifies 140.252.104.2 as the next-hop router. (We'll verify that this is the next-hop router for `gateway` using Traceroute in Figure 8.4.)

A few key points come out in this example.

1. All the hosts and routers in this example used a default route. Indeed, most hosts and some routers can use a default route for everything other than destinations on local networks.

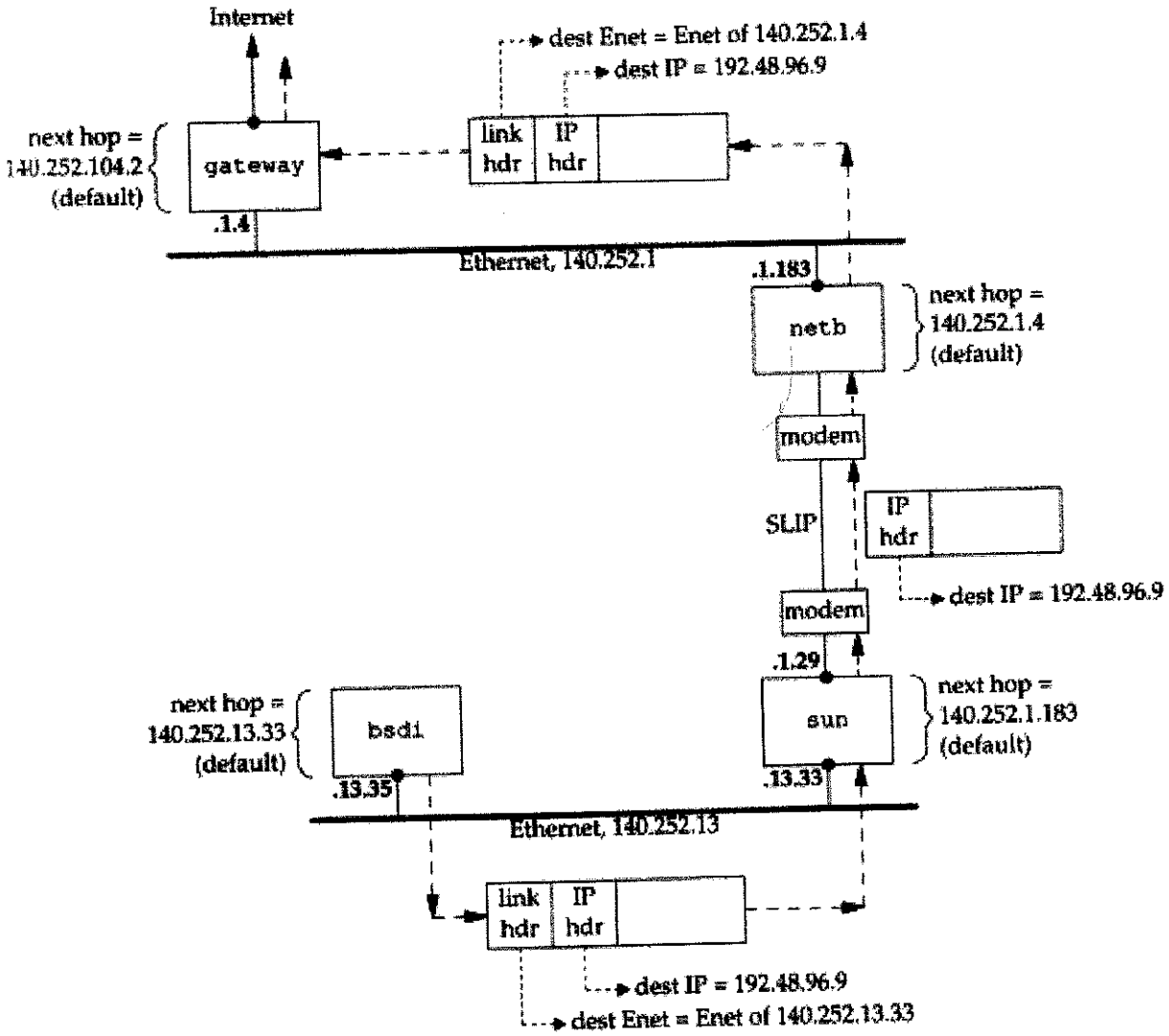

Figure 3.4 Initial path of datagram from `bsdi` to `ftp.uu.net` (192.48.96.9).

2. The destination IP address in the datagram never changes. (In Section 8.5 we'll see that this is not true only if source routing is used, which is rare.) All the routing decisions are based on this destination address.
3. A different link-layer header can be used on each link, and the link-layer destination address (if present) always contains the link-layer address of the next hop. In our example both Ethernets encapsulated a link-layer header containing the next-hop's Ethernet address, but the SLIP link did not. The Ethernet addresses are normally obtained using ARP.

In Chapter 9 we'll look at IP routing again, after describing ICMP. We'll also look at some sample routing tables and how they're used for routing decisions.

3.4 Subnet Addressing

All hosts are now required to support subnet addressing (RFC 950 [Mogul and Postel 1985]). Instead of considering an IP address as just a network ID and host ID, the host ID portion is divided into a subnet ID and a host ID.

This makes sense because class A and class B addresses have too many bits allocated for the host ID: $2^{24} - 2$ and $2^{16} - 2$, respectively. People don't attach that many hosts to a single network. (Figure 1.5 [p. 8] shows the format of the different classes of IP addresses.) We subtract 2 in these expressions because host IDs of all zero bits or all one bits are invalid.

After obtaining an IP network ID of a certain class from the InterNIC, it is up to the local system administrator whether to subnet or not, and if so, how many bits to allocate to the subnet ID and host ID. For example, the internet used in this text has a class B network address (140.252) and of the remaining 16 bits, 8 are for the subnet ID and 8 for the host ID. This is shown in Figure 3.5.

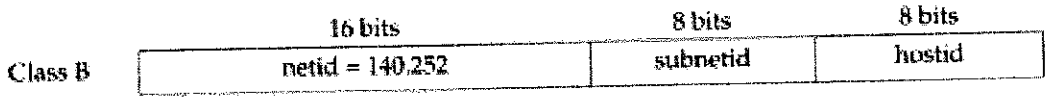

Figure 3.5 Subnetting a class B address.

This division allows 254 subnets, with 254 hosts per subnet.

Many administrators use the natural 8-bit boundary in the 16 bits of a class B host ID as the subnet boundary. This makes it easier to determine the subnet ID from a dotted-decimal number, but there is no requirement that the subnet boundary for a class A or class B address be on a byte boundary.

Most examples of subnetting describe it using a class B address. Subnetting is also allowed for a class C address, but there are fewer bits to work with. Subnetting is rarely shown with a class A address because there are so few class A addresses. (Most class A addresses are, however, subnetted.)

Subnetting hides the details of internal network organization (within a company or campus) to external routers. Using our example network, all IP addresses have the class B network ID of 140.252. But there are more than 30 subnets and more than 400 hosts distributed over those subnets. A single router provides the connection to the Internet, as shown in Figure 3.6.

In this figure we have labeled most of the routers as R*n*, where *n* is the subnet number. We show the routers that connect these subnets, along with the nine systems from the figure on the inside front cover. The Ethernets are shown as thicker lines, and the point-to-point links as dashed lines. We do *not* show all the hosts on the various subnets. For example, there are more than 50 hosts on the 140.252.3 subnet, and more than 100 on the 140.252.1 subnet.

The advantage to using a single class B address with 30 subnets, compared to 30 class C addresses, is that subnetting reduces the size of the Internet's routing tables. The fact that the class B address 140.252 is subnetted is transparent to all Internet routers other than the ones within the 140.252 subnet. To reach any host whose IP

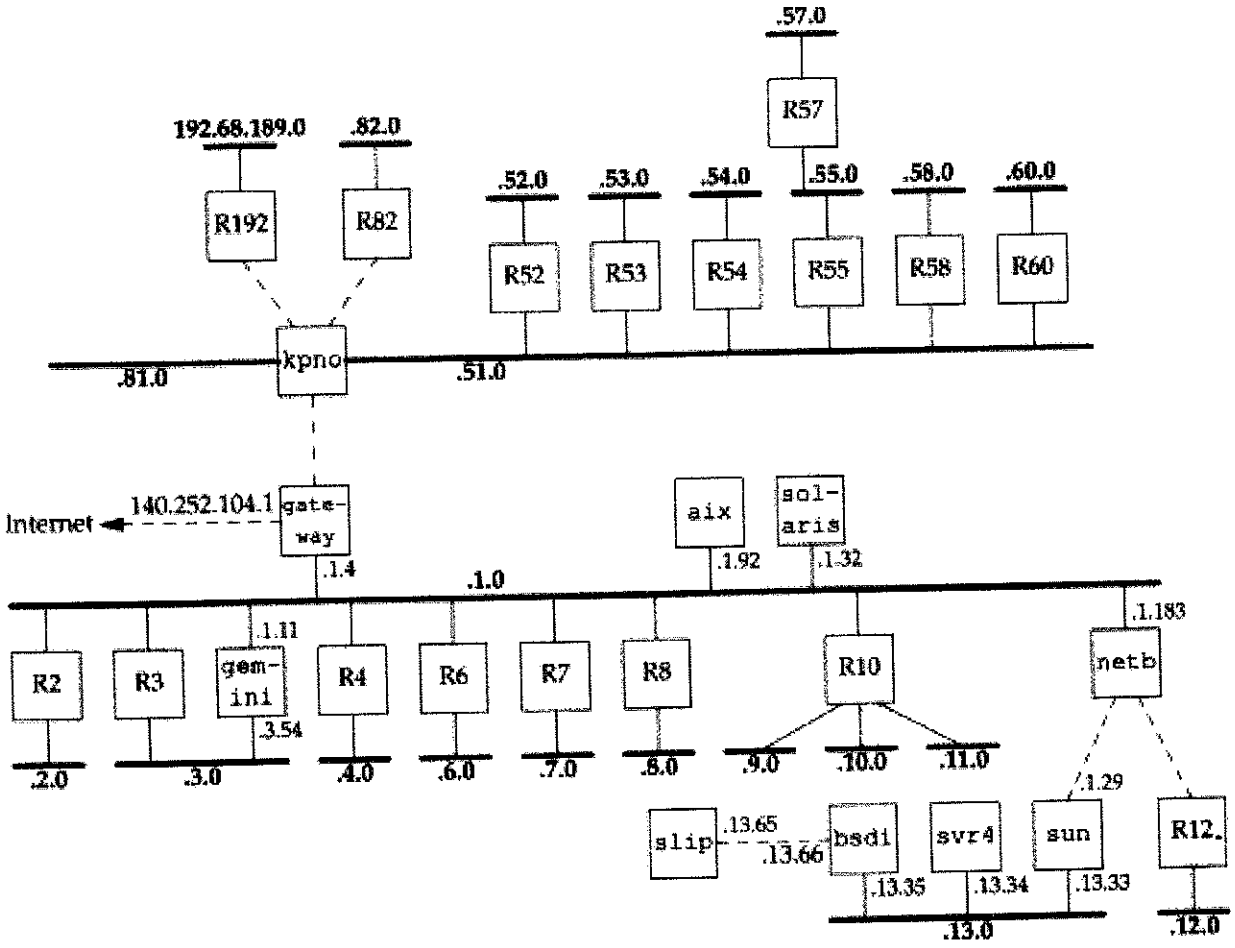

Figure 3.6 Arrangement of most of the noao.edu 140.252 subnets.

address begins with 140.252, the external routers only need to know the path to the IP address 140.252.104.1. This means that only one routing table entry is needed for all the 140.252 networks, instead of 30 entries if 30 class C addresses were used. Subnetting, therefore, reduces the size of routing tables. (In Section 10.8 we'll look at a new technique that helps reduce the size of routing tables even if class C addresses are used.)

To show that subnetting is not transparent to routers within the subnet, assume in Figure 3.6 that a datagram arrives at `gateway` from the Internet with a destination address of 140.252.57.1. The router `gateway` needs to know that the subnet number is 57, and that datagrams for this subnet are sent to `kpno`. Similarly `kpno` must send the datagram to R55, who then sends it to R57.

3.5 Subnet Mask

Part of the configuration of any host that takes place at bootstrap time is the specification of the host's IP address. Most systems have this stored in a disk file that's read at bootstrap time, and we'll see in Chapter 5 how a diskless system can also find out its IP address when it's bootstrapped.

In addition to the IP address, a host also needs to know how many bits are to be used for the subnet ID and how many bits are for the host ID. This is also specified at bootstrap time using a *subnet mask*. This mask is a 32-bit value containing one bits for the network ID and subnet ID, and zero bits for the host ID. Figure 3.7 shows the formation of the subnet mask for two different partitions of a class B address. The top example is the partitioning used at `noao.edu`, shown in Figure 3.5, where the subnet ID and host ID are both 8 bits wide. The lower example shows a class B address partitioned for a 10-bit subnet ID and a 6-bit host ID.

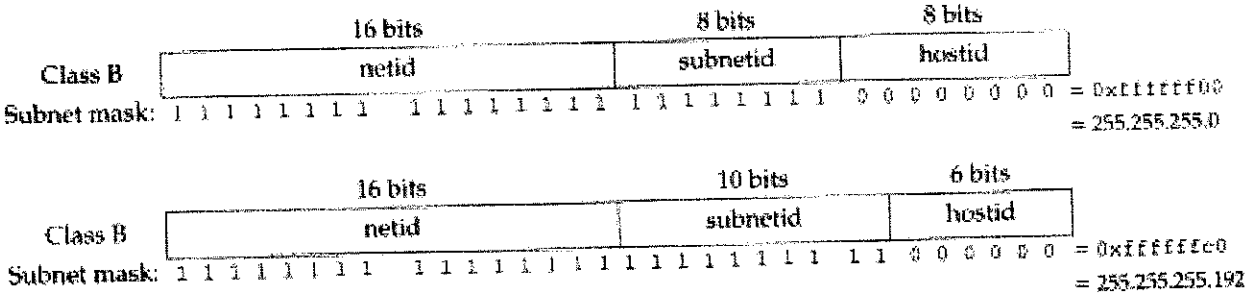

Figure 3.7 Example subnet masks for two different class B subnet arrangements.

Although IP addresses are normally written in dotted-decimal notation, subnet masks are often written in hexadecimal, especially if the boundary is not a byte boundary, since the subnet mask is a bit mask.

Given its own IP address and its subnet mask, a host can determine if an IP datagram is destined for (1) a host on its own subnet, (2) a host on a different subnet on its own network, or (3) a host on a different network. Knowing your own IP address tells you whether you have a class A, B, or C address (from the high-order bits), which tells you where the boundary is between the network ID and the subnet ID. The subnet mask then tells you where the boundary is between the subnet ID and the host ID.

Example

Assume our host address is 140.252.1.1 (a class B address) and our subnet mask is 255.255.255.0 (8 bits for the subnet ID and 8 bits for the host ID).

- If a destination IP address is 140.252.4.5, we know that the class B network IDs are the same (140.252), but the subnet IDs are different (1 and 4). Figure 3.8 shows how this comparison of two IP addresses is done, using the subnet mask.
- If the destination IP address is 140.252.1.22, the class B network IDs are the same (140.252), and the subnet IDs are the same (1). The host IDs, however, are different.
- If the destination IP address is 192.43.235.6 (a class C address), the network IDs are different. No further comparisons can be made against this address.

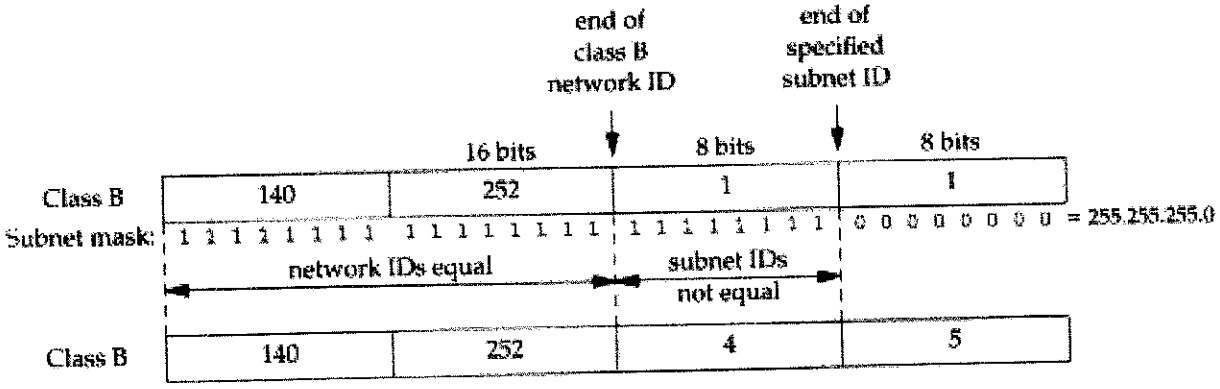

Figure 3.8 Comparison of two class B addresses using a subnet mask.

The IP routing function makes comparisons like this all the time, given two IP addresses and a subnet mask.

3.6 Special Case IP Addresses

Having described subnetting we now show the seven special case IP addresses in Figure 3.9. In this figure, 0 means a field of all zero bits, –1 means a field of all one bits, and *netid*, *subnetid*, and *hostid* mean the corresponding field that is neither all zero bits nor all one bits. A blank subnet ID column means the address is not subnetted.

IP address			Can appear as		Description
net ID	subnet ID	host ID	source?	destination?	
0		0	OK	never	this host on this net (see restrictions below)
0		*hostid*	OK	never	specified host on this net (see restrictions below)
127		*anything*	OK	OK	loopback address (Section 2.7)
–1		–1	never	OK	limited broadcast (never forwarded)
netid		–1	never	OK	net-directed broadcast to *netid*
netid	*subnetid*	–1	never	OK	subnet-directed broadcast to *netid*, *subnetid*
netid	–1	–1	never	OK	all-subnets-directed broadcast to *netid*

Figure 3.9 Special case IP addresses.

We have divided this table into three sections. The first two entries are special case source addresses, the next one is the special loopback address, and the final four are the broadcast addresses.

The first two entries in the table, with a network ID of 0, can only appear as the source address as part of an initialization procedure when a host is determining its own IP address, for example, when the BOOTP protocol is being used (Chapter 16).

In Section 12.2 we'll examine the four types of broadcast addresses in more detail.

3.7 A Subnet Example

This example shows the subnet used in the text, and how two different subnet masks are used. Figure 3.10 shows the arrangement.

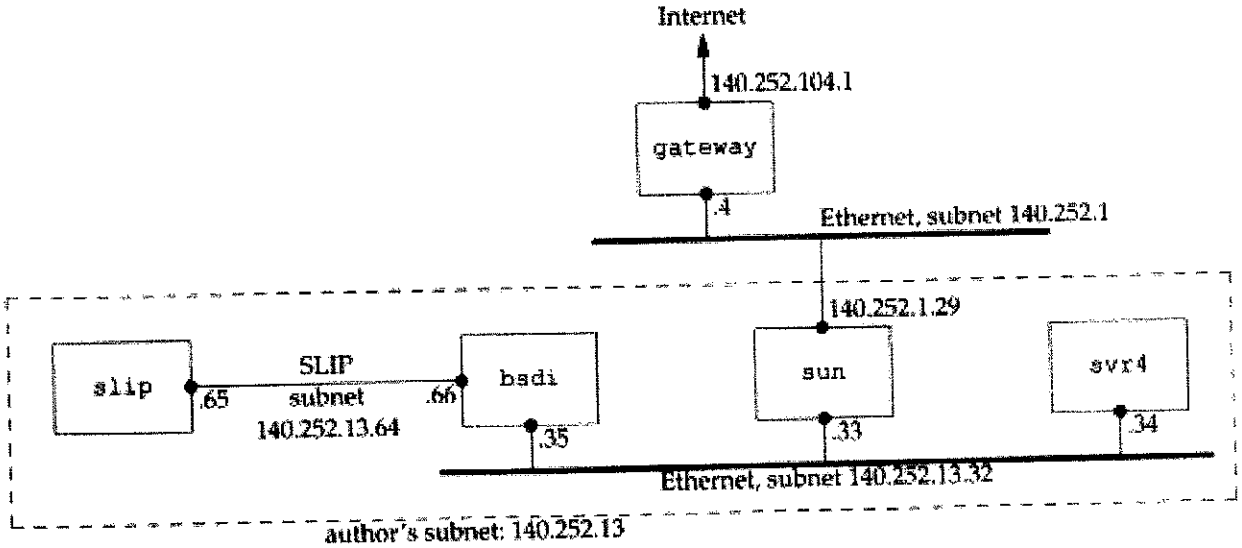

Figure 3.10 Arrangement of hosts and networks for author's subnet.

If you compare this figure with the one on the inside front cover, you'll notice that we've omitted the detail that the connection from the router sun to the top Ethernet in Figure 3.10 is really a dialup SLIP connection. This detail doesn't affect our description of subnetting in this section. We'll return to this detail in Section 4.6 when we describe proxy ARP.

The problem is that we have two separate networks within subnet 13: an Ethernet and a point-to-point link (the hardwired SLIP link). (Point-to-point links always cause problems since each end normally requires an IP address.) There could be more hosts and networks in the future, but not enough hosts across the different networks to justify using another subnet number. Our solution is to extend the subnet ID from 8 to 11 bits, and decrease the host ID from 8 to 5 bits. This is called *variable-length subnets* since most networks within the 140.252 network use an 8-bit subnet mask while our network uses an 11-bit subnet mask.

> RFC 1009 [Braden and Postel 1987] allows a subnetted network to use more than one subnet mask. The new Router Requirements RFC [Almquist 1993] requires support for this.
>
> The problem, however, is that not all routing protocols exchange the subnet mask along with the destination network ID. We'll see in Chapter 10 that RIP does not support variable-length subnets, while RIP Version 2 and OSPF do. We don't have a problem with our example, since RIP isn't required on the author's subnet.

Figure 3.11 shows the IP address structure used within the author's subnet. The first 8 bits of the 11-bit subnet ID are always 13 within the author's subnet. For the remaining 3 bits of the subnet ID, we use binary 001 for the Ethernet, and binary 010 for

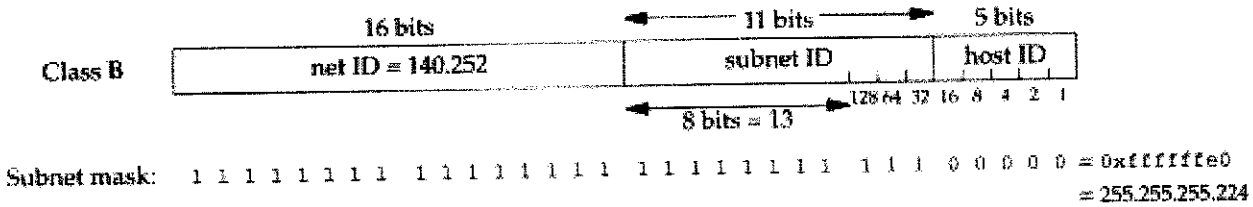

Figure 3.11 Using variable-length subnets.

the point-to-point SLIP link. This variable-length subnet mask does not cause a problem for other hosts and routers in the 140.252 network—as long as all datagrams destined for the subnet 140.252.13 are sent to the router `sun` (IP address 140.252.1.29) in Figure 3.10, and if `sun` knows about the 11-bit subnet ID for the hosts on its subnet 13, everything is fine.

The subnet mask for all the interfaces on the 140.252.13 subnet is 255.255.255.224, or `0xffffffe0`. This indicates that the rightmost 5 bits are for the host ID, and the 27 bits to the left are the network ID and subnet ID.

Figure 3.12 shows the allocation of IP addresses and subnet masks for the interfaces shown in Figure 3.10.

Host	IP address	Subnet mask	Net ID/Subnet ID	Host ID	Comment
`sun`	140.252.1.29	255.255.255.0	140.252.1	29	on subnet 1
	140.252.13.33	255.255.255.224	140.252.13.32	1	on author's Ethernet
`svr4`	140.252.13.34	255.255.255.224	140.252.13.32	2	
`bsdi`	140.252.13.35	255.255.255.224	140.252.13.32	3	on Ethernet
	140.252.13.66	255.255.255.224	140.252.13.64	2	point-to-point
`slip`	140.252.13.65	255.255.255.224	140.252.13.64	1	point-to-point
	140.252.13.63	255.255.255.224	140.252.13.32	31	broadcast addr on Ethernet

Figure 3.12 IP addresses on author's subnet.

The first column is labeled "Host," but both `sun` and `bsdi` also act as routers, since they are multihomed and route packets from one interface to another.

The final row in this table notes that the broadcast address for the bottom Ethernet in Figure 3.10 is 140.252.13.63: it is formed from the subnet ID of the Ethernet (140.252.13.32) and the low-order 5 bits in Figure 3.11 set to 1 (16 + 8 + 4 + 2 + 1 = 31). (We'll see in Chapter 12 that this address is called the subnet-directed broadcast address.)

3.8 `ifconfig` Command

Now that we've described the link layer and the IP layer we can show the command used to configure or query a network interface for use by TCP/IP. The `ifconfig`(8) command is normally run at bootstrap time to configure each interface on a host.

For dialup interfaces that may go up and down (such as SLIP links), `ifconfig` must be run (somehow) each time the line is brought up or down. How this is done each time the SLIP link is brought up or down depends on the SLIP software being used.

The following output shows the values for the author's subnet. Compare these values with the values in Figure 3.12.

```
sun % /usr/etc/ifconfig -a          SunOS -a option says report on all interfaces
le0: flags=63<UP,BROADCAST,NOTRAILERS,RUNNING>
        inet 140.252.13.33 netmask ffffffe0 broadcast 140.252.13.63
sl0: flags=1051<UP,POINTOPOINT,RUNNING,LINK0>
        inet 140.252.1.29 --> 140.252.1.183 netmask ffffff00
lo0: flags=49<UP,LOOPBACK,RUNNING>
        inet 127.0.0.1 netmask ff000000
```

The loopback interface (Section 2.7) is considered a network interface. Its class A address is not subnetted.

Other things to notice are that trailer encapsulation (Section 2.3) is not used on the Ethernet, and that the Ethernet is capable of broadcasting, while the SLIP link is a point-to-point link.

The flag `LINK0` for the SLIP interface is the configuration option that enables compressed slip (CSLIP, Section 2.5). Other possible options are `LINK1`, which enables CSLIP if a compressed packet is received from the other end, and `LINK2`, which causes all outgoing ICMP packets to be thrown away. We'll look at the destination address of this SLIP link in Section 4.6.

> A comment in the installation instructions gives the reason for this last option: "This shouldn't have to be set, but some cretin pinging you can drive your throughput to zero."

`bsdi` is the other router. Since the `-a` option is a SunOS feature, we have to execute `ifconfig` multiple times, specifying the interface name as an argument:

```
bsdi % /sbin/ifconfig we0
we0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX>
        inet 140.252.13.35 netmask ffffffe0 broadcast 140.252.13.63
bsdi % /sbin/ifconfig sl0
sl0: flags=1011<UP,POINTOPOINT,LINK0>
        inet 140.252.13.66 --> 140.252.13.65 netmask ffffffe0
```

Here we see a new option for the Ethernet interface (`we0`): `SIMPLEX`. This 4.4BSD flag specifies that the interface can't hear its own transmissions. It is set in BSD/386 for all the Ethernet interfaces. When set, if the interface is sending a frame to the broadcast address, a copy is made for the local host and sent to the loopback address. (We show an example of this feature in Section 6.3.)

On the host `slip` the configuration of the SLIP interface is nearly identical to the output shown above on `bsdi`, with the exception that the IP addresses of the two ends are swapped:

```
slip % /sbin/ifconfig sl0
sl0: flags=1011<UP,POINTOPOINT,LINK0>
        inet 140.252.13.65 --> 140.252.13.66 netmask ffffffe0
```

The final interface is the Ethernet interface on the host `svr4`. It is similar to the Ethernet output shown earlier, except that SVR4's version of `ifconfig` doesn't print the `RUNNING` flag:

```
svr4 % /usr/sbin/ifconfig emd0
emd0: flags=23<UP,BROADCAST,NOTRAILERS>
        inet 140.252.13.34 netmask ffffffe0 broadcast 140.252.13.63
```

The `ifconfig` command normally supports other protocol families (other than TCP/IP) and has numerous additional options. Check your system's manual for these details.

3.9 `netstat` Command

The `netstat`(1) command also provides information about the interfaces on a system. The `-i` flag prints the interface information, and the `-n` flag prints IP addresses instead of hostnames.

```
sun % netstat -in
Name  Mtu  Net/Dest      Address        Ipkts  Ierrs Opkts  Oerrs Collis Queue
le0   1500 140.252.13.32 140.252.13.33  67719  0     92133  0     1      0
sl0   552  140.252.1.183 140.252.1.29   48035  0     54963  0     0      0
lo0   1536 127.0.0.0     127.0.0.1      15548  0     15548  0     0      0
```

This command prints the MTU of each interface, the number of input packets, input errors, output packets, output errors, collisions, and the current size of the output queue.

We'll return to the `netstat` command in Chapter 9 when we use it to examine the routing table, and in Chapter 13 when we use a modified version to see active multicast groups.

3.10 IP Futures

There are three problems with IP. They are a result of the phenomenal growth of the Internet over the past few years. (See Exercise 1.2 also.)

1. Over half of all class B addresses have already been allocated. Current estimates predict exhaustion of the class B address space around 1995, if they continue to be allocated as they have been in the past.
2. 32-bit IP addresses in general are inadequate for the predicted long-term growth of the Internet.
3. The current routing structure is not hierarchical, but flat, requiring one routing table entry per network. As the number of networks grows, amplified by the allocation of multiple class C addresses to a site with multiple networks, instead of a single class B address, the size of the routing tables grows.

CIDR (Classless Interdomain Routing) proposes a fix to the third problem that will extend the usefulness of the current version of IP (IP version 4) into the next century. We discuss it in more detail in Section 10.8.

Four proposals have been made for a new version of IP, often called *IPng*, for the next generation of IP. The May 1993 issue of *IEEE Network* (vol. 7, no. 3) contains overviews of the first three proposals, along with an article on CIDR. RFC 1454 [Dixon 1993] also compares the first three proposals.

1. SIP, the Simple Internet Protocol. It proposes a minimal set of changes to IP that uses 64-bit addresses and a different header format. (The first 4 bits of the header still contain the version number, with a value other than 4.)
2. PIP. This proposal also uses larger, variable-length, hierarchical addresses with a different header format.
3. TUBA, which stands for "TCP and UDP with Bigger Addresses," is based on the OSI CLNP (Connectionless Network Protocol), an OSI protocol similar to IP. It provides much larger addresses: variable length, up to 20 bytes. Since CLNP is an existing protocol, whereas SIP and PIP are just proposals, documentation already exists on CLNP. RFC 1347 [Callon 1992] provides details on TUBA. Chapter 7 of [Perlman 1992] contains a comparison of IPv4 and CLNP. Many routers already support CLNP, but few hosts do.
4. TP/IX, which is described in RFC 1475 [Ullmann 1993]. As with SIP, it uses 64 bits for IP addresses, but it also changes the TCP and UDP headers: 32-bit port number for both protocols, along with 64-bit sequence numbers, 64-bit acknowledgment numbers, and 32-bit windows for TCP.

The first three proposals use basically the same versions of TCP and UDP as the transport layers.

Since only one of these four proposals will be chosen as the successor to IPv4, and since the decision may have been made by the time you read this, we won't say any more about them. With the forthcoming implementation of CIDR to handle the short-term problem, it will take many years to implement the successor to IPv4.

3.11 Summary

We started this chapter with a description of the IP header and briefly described all the fields in this header. We also gave an introduction to IP routing, and saw that host routing can be simple: the destination is either on a directly connected network, in which case the datagram is sent directly to the destination, or a default router is chosen.

Hosts and routers have a routing table that is used for all routing decisions. There are three types of routes in the table: host specific, network specific, and optional default routes. There is a priority to the entries in a routing table. A host route will be chosen over a network router, and a default route is used only when no other route exists to the destination.

IP routing is done on a hop-by-hop basis. The destination IP address never changes as the datagram proceeds through all the hops, but the encapsulation and destination link-layer address can change on each hop. Most hosts and many routers use a default next-hop router for all nonlocal traffic.

Class A and B addresses are normally subnetted. The number of bits used for the subnet ID is specified by the subnet mask. We gave a detailed example of this, using the author's subnet, and introduced variable-length subnets. The use of subnetting reduces the size of the Internet routing tables, since many networks can often be accessed through a single point. Information on the interfaces and networks is available through the `ifconfig` and `netstat` commands. This includes the IP address of the interface, its subnet mask, broadcast address, and MTU.

We finished the chapter with a discussion of potential changes to the Internet protocol suite–the next generation of IP.

Exercises

3.1 Must the loopback address be 127.0.0.1?

3.2 Identify the routers in Figure 3.6 with more than two network interfaces.

3.3 What's the difference in the subnet mask for a class A address with 16 bits for the subnet ID and a class B address with 8 bits for the subnet ID?

3.4 Read RFC 1219 [Tsuchiya 1991] for a recommended technique for assigning subnet IDs and host IDs.

3.5 Is the subnet mask 255.255.0.255 valid for a class A address?

3.6 Why do you think the MTU of the loopback interface printed in Section 3.9 is set to 1536?

3.7 The TCP/IP protocol suite is built on a datagram network technology, the IP layer. Other protocol suites are built on a connection-oriented network technology. Read [Clark 1988] to discover the three advantages the datagram network layer provides.