CHAPTER

5

IPv4 Addresses

At the network layer, we need to uniquely identify each device on the Internet to allow global communication between all devices. In this chapter, we discuss the addressing mechanism related to the prevalent IPv4 protocol called IPv4 addressing. It is believed that IPv6 protocol will eventually supersede the current protocol, and we need to become aware of IPv6 addressing as well. We discuss IPv6 protocol and its addressing mechanism in Chapters 26 to 28.

OBJECTIVES

This chapter has several objectives:

- To introduce the concept of an address space in general and the address space of IPv4 in particular.
- To discuss the classful architecture, classes in this model, and the blocks of addresses available in each class.
- To discuss the idea of hierarchical addressing and how it has been implemented in classful addressing.
- To explain subnetting and supernetting for classful architecture and show how they were used to overcome the deficiency of classful addressing.
- To discuss the new architecture, classless addressing, that has been devised to solve the problems in classful addressing such as address depletion.
- To show how some ideas borrowed from classful addressing such as subnetting can be easily implemented in classless addressing.
- To discuss some special blocks and some special addresses in each block.
- To discuss NAT technology and show how it can be used to alleviate the shortage in number of addresses in IPv4.

5.1 INTRODUCTION

The identifier used in the IP layer of the TCP/IP protocol suite to identify each device connected to the Internet is called the Internet address or **IP address.** An IPv4 address is a 32-bit address that *uniquely* and *universally* defines the connection of a host or a router to the Internet; an IP address is the address of the interface.

An IPv4 address is 32 bits long.

IPv4 addresses are *unique*. They are unique in the sense that each address defines one, and only one, connection to the Internet. Two devices on the Internet can never have the same address at the same time. However, if a device has two connections to the Internet, via two networks, it has two IPv4 addresses. The IPv4 addresses are *universal* in the sense that the addressing system must be accepted by any host that wants to be connected to the Internet.

The IPv4 addresses are unique and universal.

Address Space

A protocol like IPv4 that defines addresses has an **address space.** An address space is the total number of addresses used by the protocol. If a protocol uses *b* bits to define an address, the address space is 2^b because each bit can have two different values (0 or 1). IPv4 uses 32-bit addresses, which means that the address space is 2^{32} or 4,294,967,296 (more than four billion). Theoretically, if there were no restrictions, more than 4 billion devices could be connected to the Internet.

The address space of IPv4 is 2^{32} or 4,294,967,296.

Notation

There are three common notations to show an IPv4 address: binary notation (base 2), dotted-decimal notation (base 256), and hexadecimal notation (base 16). The most prevalent, however, is base 256. These bases are defined in Appendix B. We also show how to convert a number from one base to another in that appendix. We recommend a review of this appendix before continuing with this chapter.

Numbers in base 2, 16, and 256 are discussed in Appendix B.

Binary Notation: Base 2

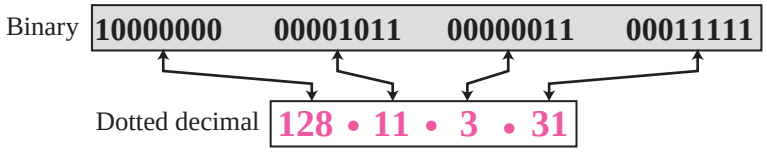
In **binary notation,** an IPv4 address is displayed as 32 bits. To make the address more readable, one or more spaces is usually inserted between each octet (8 bits). Each octet is often referred to as a byte. So it is common to hear an IPv4 address referred to as a 32-bit address, a 4-octet address, or a 4-byte address. The following is an example of an IPv4 address in binary notation:

```
01110101 10010101 00011101 11101010
```

Dotted-Decimal Notation: Base 256

To make the IPv4 address more compact and easier to read, an IPv4 address is usually written in decimal form with a decimal point (dot) separating the bytes. This format is referred to as **dotted-decimal notation.** Figure 5.1 shows an IPv4 address in dotted-decimal notation. Note that because each byte (octet) is only 8 bits, each number in the dotted-decimal notation is between 0 and 255.

Figure 5.1 *Dotted-decimal notation*



Example 5.1

Change the following IPv4 addresses from binary notation to dotted-decimal notation.

a. `10000001 00001011 00001011 11101111`
b. `11000001 10000011 00011011 11111111`
c. `11100111 11011011 10001011 01101111`
d. `11111001 10011011 11111011 00001111`

Solution

We replace each group of 8 bits with its equivalent decimal number (see Appendix B) and add dots for separation:

a. 129.11.11.239
b. 193.131.27.255
c. 231.219.139.111
d. 249.155.251.15

Example 5.2

Change the following IPv4 addresses from dotted-decimal notation to binary notation.

a. 111.56.45.78
b. 221.34.7.82

c. 241.8.56.12
d. 75.45.34.78

Solution
We replace each decimal number with its binary equivalent (see Appendix B):

a. **01101111 00111000 00101101 01001110**
b. **11011101 00100010 00000111 01010010**
c. **11110001 00001000 00111000 00001100**
d. **01001011 00101101 00100010 01001110**

Example 5.3

Find the error, if any, in the following IPv4 addresses:

a. 111.56.045.78
b. 221.34.7.8.20
c. 75.45.301.14
d. 11100010.23.14.67

Solution

a. There should be no leading zeroes in dotted-decimal notation (045).
b. We may not have more than 4 bytes in an IPv4 address.
c. Each byte should be less than or equal to 255; 301 is outside this range.
d. A mixture of binary notation and dotted-decimal notation is not allowed.

Hexadecimal Notation: Base 16

We sometimes see an IPv4 address in **hexadecimal notation.** Each hexadecimal digit is equivalent to four bits. This means that a 32-bit address has 8 hexadecimal digits. This notation is often used in network programming.

Example 5.4

Change the following IPv4 addresses from binary notation to hexadecimal notation.

a. **10000001 00001011 00001011 11101111**
b. **11000001 10000011 00011011 11111111**

Solution
We replace each group of 4 bits with its hexadecimal equivalent (see Appendix B). Note that hexadecimal notation normally has no added spaces or dots; however, 0X (or 0x) is added at the beginning or the subscript 16 at the end to show that the number is in hexadecimal.

a. 0X810B0BEF or $810B0BEF_{16}$
b. 0XC1831BFF or $C1831BFF_{16}$

Range of Addresses

We often need to deal with a range of addresses instead of one single address. We sometimes need to find the number of addresses in a range if the first and last address is given. Other times, we need to find the last address if the first address and the number of addresses in the range are given. In this case, we can perform subtraction or addition

operations in the corresponding base (2, 256, or 16). Alternatively, we can covert the addresses to decimal values (base 10) and perform operations in this base.

Example 5.5

Find the number of addresses in a range if the first address is 146.102.29.0 and the last address is 146.102.32.255.

Solution

We can subtract the first address from the last address in base 256 (see Appendix B). The result is 0.0.3.255 in this base. To find the number of addresses in the range (in decimal), we convert this number to base 10 and add 1 to the result.

$$\textbf{Number of addresses} = (0 \times 256^3 + 0 \times 256^2 + 3 \times 256^1 + 255 \times 256^0) + 1 = 1024$$

Example 5.6

The first address in a range of addresses is 14.11.45.96. If the number of addresses in the range is 32, what is the last address?

Solution

We convert the number of addresses minus 1 to base 256, which is 0.0.0.31. We then add it to the first address to get the last address. Addition is in base 256.

$$\textbf{Last address} = (14.11.45.96 + 0.0.0.31)_{256} = 14.11.45.127$$

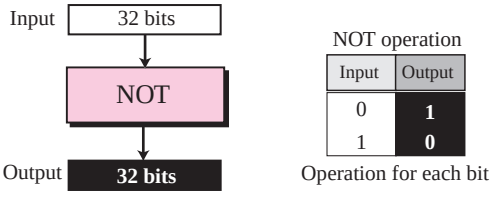
Operations

We often need to apply some operations on 32-bit numbers in binary or dotted-decimal notation. These numbers either represent IPv4 addresses or some entities related to IPv4 addresses (such as a *mask*, which is discussed later). In this section, we introduce three operations that are used later in the chapter: NOT, AND, and OR.

Bitwise NOT Operation

The bitwise NOT operation is a unary operation; it takes one input. When we apply the NOT operation on a number, it is often said that the number is complemented. The NOT operation, when applied to a 32-bit number in binary format, inverts each bit. Every 0 bit is changed to a 1 bit; every 1 bit is changed to a 0 bit. Figure 5.2 shows the NOT operation.

Figure 5.2 *Bitwise NOT operation*



Although we can directly use the NOT operation on a 32-bit number, when the number is represented as a four-byte dotted-decimal notation, we can use a short cut; we can subtract each byte from 255.

Example 5.7

The following shows how we can apply the NOT operation on a 32-bit number in binary.

Original number:	00010001	01111001	00001110	00100011
Complement:	11101110	10000110	11110001	11011100

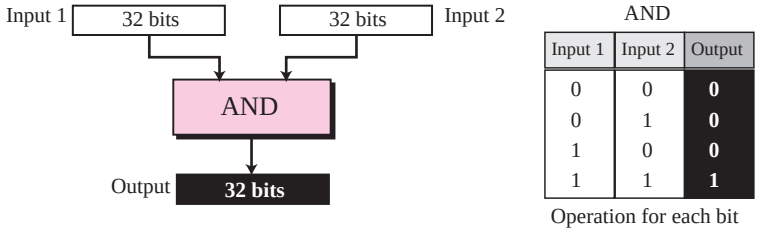
We can use the same operation using the dotted-decimal representation and the short cut.

Original number:	17	.	121	.	14	.	35
Complement:	238	.	134	.	241	.	220

Bitwise AND Operation

The bitwise AND operation is a binary operation; it takes two inputs. The AND operation compares the two corresponding bits in two inputs and selects the smaller bit from the two (or select one of them if the bits are equal). Figure 5.3 shows the AND operation.

Figure 5.3 *Bitwise AND operation*



Although we can directly use the AND operation on the 32-bit binary representation of two numbers, when the numbers are represented in dotted-decimal notation, we can use two short cuts.

1. When at least one of the numbers is 0 or 255, the AND operation selects the smaller byte (or one of them if equal).
2. When none of the two bytes is either 0 or 255, we can write each byte as the sum of eight terms, where each term is a power of 2. We then select the smaller term in each pair (or one of them if equal) and add them to get the result.

Example 5.8

The following shows how we can apply the AND operation on two 32-bit numbers in binary.

First number:	00010001	01111001	00001110	00100011
Second number:	11111111	11111111	10001100	00000000
Result	00010001	01111001	00001100	00000000

We can use the same operation using the dotted-decimal representation and the short cut.

First number:	17	.	121	.	14	.	35
Second number:	255	.	255	.	140	.	0
Result:	17	.	121	.	12	.	0

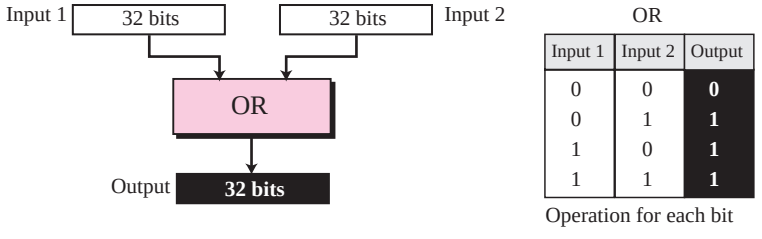
We have applied the first short cut on the first, second, and the fourth byte; we have applied the second short cut on the third byte. We have written 14 and 140 as the sum of terms and selected the smaller term in each pair as shown below.

Powers	2^7		2^6		2^5		2^4		2^3		2^2		2^1		2^0
Byte (14)	0	+	0	+	0	+	0	+	8	+	4	+	2	+	0
Byte (140)	128	+	0	+	0	+	0	+	8	+	4	+	0	+	0
Result (12)	0	+	0	+	0	+	0	+	8	+	4	+	0	+	0

Bitwise OR Operation

The bitwise OR operation is a binary operation; it takes two inputs. The OR operation compares the corresponding bits in the two numbers and selects the larger bit from the two (or one of them if equal). Figure 5.4 shows the OR operation.

Figure 5.4 *Bitwise OR operation*



OR

Input 1	Input 2	Output
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**1**

Operation for each bit

Although we can directly use the OR operation on the 32-bit binary representation of the two numbers, when the numbers are represented in dotted-decimal notation, we can use two short cuts.

1. When at least one of the two bytes is 0 or 255, the OR operation selects the larger byte (or one of them if equal).
2. When none of the two bytes is 0 or 255, we can write each byte as the sum of eight terms, where each term is a power of 2. We then select the larger term in each pair (or one of them if equal) and add them to get the result of OR operation.

Example 5.9

The following shows how we can apply the OR operation on two 32-bit numbers in binary.

First number:	00010001	01111001	00001110	00100011
Second number:	11111111	11111111	10001100	00000000
Result	11111111	11111111	10001110	00100011

We can use the same operation using the dotted-decimal representation and the short cut.

First number:	17	.	121	.	14	.	35
Second number:	255	.	255	.	140	.	0
Result:	255	.	255	.	142	.	35

We have used the first short cut for the first and second bytes and the second short cut for the third byte.

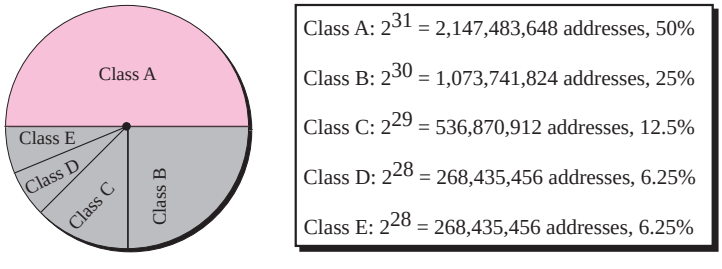
5.2 CLASSFUL ADDRESSING

IP addresses, when started a few decades ago, used the concept of *classes*. This architecture is called **classful addressing.** In the mid-1990s, a new architecture, called **classless addressing,** was introduced that supersedes the original architecture. In this section, we introduce classful addressing because it paves the way for understanding classless addressing and justifies the rationale for moving to the new architecture. Classless addressing is discussed in the next section.

Classes

In classful addressing, the IP address space is divided into five **classes: A, B, C, D,** and **E.** Each class occupies some part of the whole address space. Figure 5.5 shows the class occupation of the address space.

Figure 5.5 *Occupation of the address space*

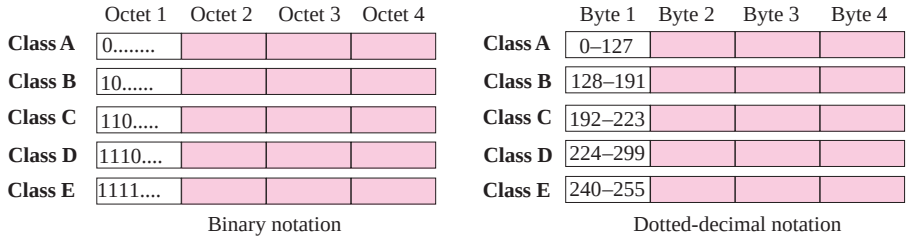


In classful addressing, the address space is divided into five classes: A, B, C, D, and E.

Recognizing Classes

We can find the class of an address when the address is given either in binary or dotted-decimal notation. In the binary notation, the first few bits can immediately tell us the class of the address; in the dotted-decimal notation, the value of the first byte can give the class of an address (Figure 5.6).

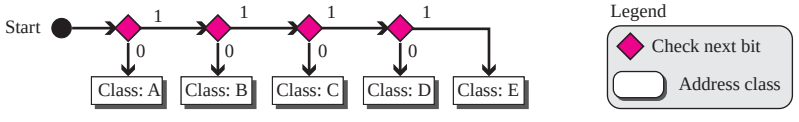
Figure 5.6 *Finding the class of an address*



Note that some special addresses fall in class A or E. We emphasize that these special addresses are exceptions to the classification; they are discussed later in the chapter.

Computers often store IPv4 addresses in binary notation. In this case, it is very convenient to write an algorithm to use a continuous checking process for finding the address as shown in Figure 5.7.

Figure 5.7 *Finding the address class using continuous checking*



Example 5.10

Find the class of each address:

a. **00000001 00001011 00001011 11101111**
b. **11000001 10000011 00011011 11111111**
c. **10100111 11011011 10001011 01101111**
d. **11110011 10011011 11111011 00001111**

Solution

See the procedure in Figure 5.7.

a. The first bit is 0. This is a class A address.
b. The first 2 bits are 1; the third bit is 0. This is a class C address.
c. The first bit is 1; the second bit is 0. This is a class B address.
d. The first 4 bits are 1s. This is a class E address.

Example 5.11

Find the class of each address:

a. 227.12.14.87
b. 193.14.56.22
c. 14.23.120.8
d. 252.5.15.111

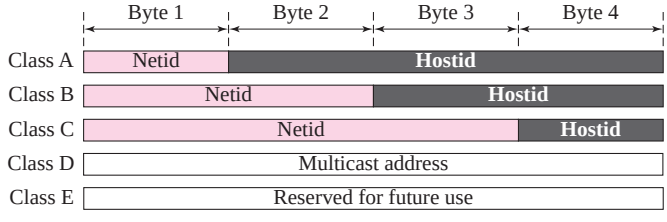
Solution

a. The first byte is 227 (between 224 and 239); the class is D.
b. The first byte is 193 (between 192 and 223); the class is C.
c. The first byte is 14 (between 0 and 127); the class is A.
d. The first byte is 252 (between 240 and 255); the class is E.

Netid and Hostid

In classful addressing, an IP address in classes A, B, and C is divided into **netid** and **hostid.** These parts are of varying lengths, depending on the class of the address. Figure 5.8 shows the netid and hostid bytes. Note that classes D and E are not divided into netid and hostid, for reasons that we will discuss later.

Figure 5.8 *Netid and hostid*



In class A, 1 byte defines the netid and 3 bytes define the hostid. In class B, 2 bytes define the netid and 2 bytes define the hostid. In class C, 3 bytes define the netid and 1 byte defines the hostid.

Classes and Blocks

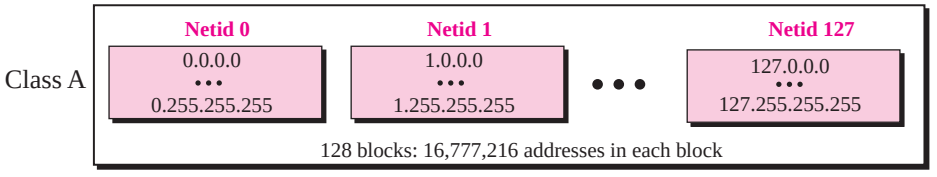
One problem with classful addressing is that each class is divided into a fixed number of blocks with each block having a fixed size. Let us look at each class.

Class A

Since only 1 byte in class A defines the netid and the leftmost bit should be 0, the next 7 bits can be changed to find the number of blocks in this class. Therefore, class A is divided into $2^7 = 128$ blocks that can be assigned to 128 organizations (the number is less because some blocks were reserved as special blocks). However, each block in this class contains 16,777,216 addresses, which means the organization should be a really

large one to use all these addresses. Many addresses are wasted in this class. Figure 5.9 shows the block in class A.

Figure 5.9 *Blocks in class A*

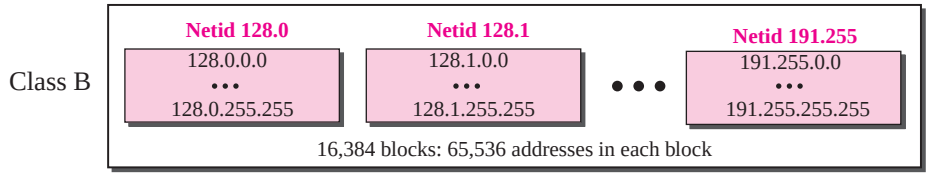


Millions of class A addresses are wasted.

Class B

Since 2 bytes in class B define the class and the two leftmost bit should be 10 (fixed), the next 14 bits can be changed to find the number of blocks in this class. Therefore, class B is divided into $2^{14} = 16{,}384$ blocks that can be assigned to 16,384 organizations (the number is less because some blocks were reserved as special blocks). However, each block in this class contains 65,536 addresses. Not so many organizations can use so many addresses. Many addresses are wasted in this class. Figure 5.10 shows the blocks in class B.

Figure 5.10 *Blocks in class B*

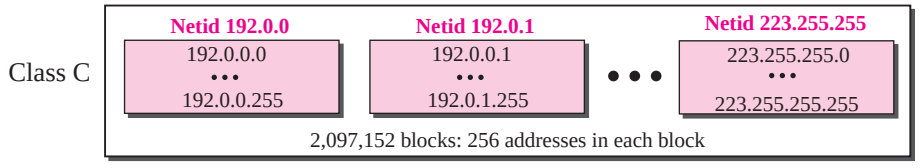


Many class B addresses are wasted.

Class C

Since 3 bytes in class C define the class and the three leftmost bits should be 110 (fixed), the next 21 bits can be changed to find the number of blocks in this class. Therefore, class C is divided into $2^{21} = 2{,}097{,}152$ blocks, in which each block contains 256 addresses, that can be assigned to 2,097,152 organizations (the number is less because some blocks were reserved as special blocks). Each block contains 256 addresses. However, not so many organizations were so small as to be satisfied with a class C block. Figure 5.11 shows the blocks in class C.

Figure 5.11 *Blocks in class C*

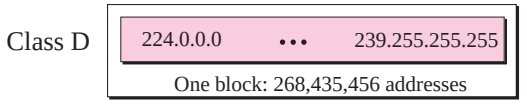


Not so many organizations are so small to have a class C block.

Class D

There is just one block of class D addresses. It is designed for multicasting, as we will see in a later section. Each address in this class is used to define one group of hosts on the Internet. When a group is assigned an address in this class, every host that is a member of this group will have a multicast address in addition to its normal (unicast) address. Figure 5.12 shows the block.

Figure 5.12 *The single block in Class D*

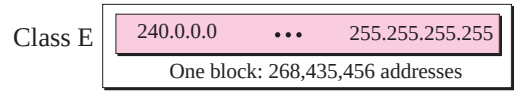


Class D addresses are made of one block, used for multicasting.

Class E

There is just one block of class E addresses. It was designed for use as reserved addresses, as shown in Figure 5.13.

Figure 5.13 *The single block in Class E*



The only block of class E addresses was reserved for future purposes.

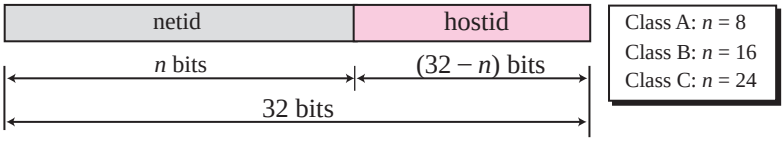
Two-Level Addressing

The whole purpose of IPv4 addressing is to define a destination for an Internet packet (at the network layer). When classful addressing was designed, it was assumed that the whole Internet is divided into many networks and each network connects many hosts. In other words, the Internet was seen as a network of networks. A network was normally created by an organization that wanted to be connected to the Internet. The Internet authorities allocated a block of addresses to the organization (in class A, B, or C).

The range of addresses allocated to an organization in classful addressing was a block of addresses in Class A, B, or C.

Since all addresses in a network belonged to a single block, each address in classful addressing contains two parts: netid and hostid. The netid defines the network; the hostid defines a particular host connected to that network. Figure 5.14 shows an IPv4 address in classful addressing. If *n* bits in the class defines the net, then 32 – *n* bits defines the host. However, the value of *n* depends on the class the block belongs to. The value of *n* can be 8, 16 or 24 corresponding to classes A, B, and C respectively.

Figure 5.14 *Two-level addressing in classful addressing*



Example 5.12

Two-level addressing can be found in other communication systems. For example, a telephone system inside the United States can be thought of as two parts: area code and local part. The area code defines the area, the local part defines a particular telephone subscriber in that area.

(626) 3581301

The area code, 626, can be compared with the netid, the local part, 3581301, can be compared to the hostid.

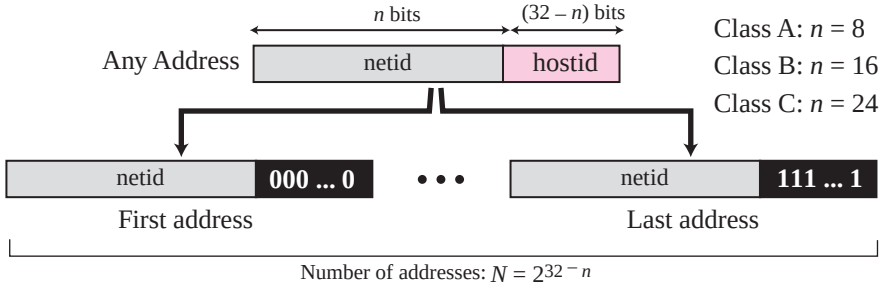
Extracting Information in a Block

A block is a range of addresses. Given any address in the block, we normally like to know three pieces of information about the block: the number of addresses, the first address, and the last address. Before we can extract these pieces of information, we need to know the class of the address, which we showed how to find in the previous section. After the class of the block is found, we know the value of *n*, the length of netid in bits. We can now find these three pieces of information as shown in Figure 5.15.

1. The number of addresses in the block, *N*, can be found using $N = 2^{32-n}$.

2. To find the first address, we keep the *n* leftmost bits and set the (32 – *n*) rightmost bits all to 0s.
3. To find the last address, we keep the *n* leftmost bits and set the (32 – *n*) rightmost bits all to 1s.

Figure 5.15 *Information extraction in classful addressing*



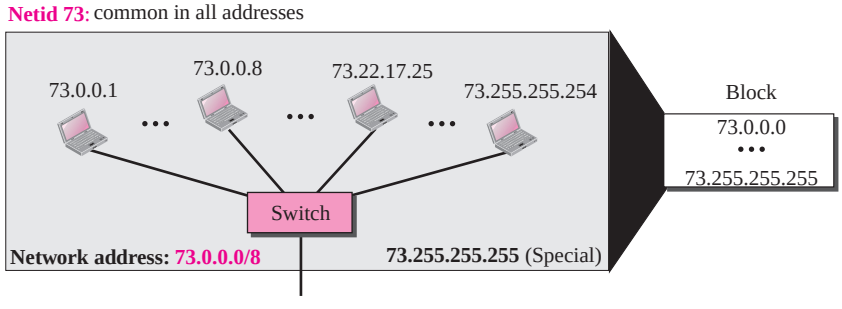
Example 5.13

An address in a block is given as 73.22.17.25. Find the number of addresses in the block, the first address, and the last address.

Solution

Since 73 is between 0 and 127, the class of the address is A. The value of *n* for class A is 8. Figure 5.16 shows a possible configuration of the network that uses this block. Note that we show the value of *n* in the network address after a slash. Although this was not a common practice in classful addressing, it helps to make it a habit in classless addressing in the next section.

Figure 5.16 *Solution to Example 5.13*



1. The number of addresses in this block is $N = 2^{32-n} = 2^{24} = 16{,}777{,}216$.
2. To find the first address, we keep the leftmost 8 bits and set the rightmost 24 bits all to 0s. The first address is 73.0.0.0/8 in which 8 is the value of *n*. The first address is called the *network address* and is not assigned to any host. It is used to define the network.

3. To find the last address, we keep the leftmost 8 bits and set the rightmost 24 bits all to 1s. The last address is 73.255.255.255. The last address is normally used for a special purpose, as discussed later in the chapter.

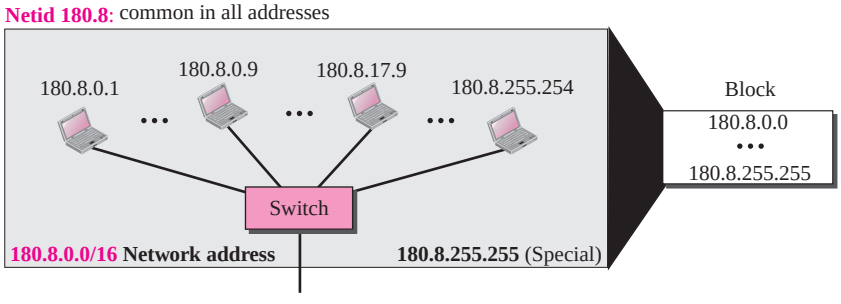
Example 5.14

An address in a block is given as 180.8.17.9. Find the number of addresses in the block, the first address, and the last address.

Solution

Since 180 is between 128 and 191, the class of the address is B. The value of *n* for class B is 16. Figure 5.17 shows a possible configuration of the network that uses this block.

Figure 5.17 *Solution to Example 5.14*



1. The number of addresses in this block is $N = 2^{32-n} = 2^{16} = 65{,}536$.
2. To find the first address, we keep the leftmost 16 bits and set the rightmost 16 bits all to 0s. The first address (network address) is 18.8.0.0/16, in which 16 is the value of *n*.
3. To find the last address, we keep the leftmost 16 bits and set the rightmost 16 bits all to 1s. The last address is 18.8.255.255.

Example 5.15

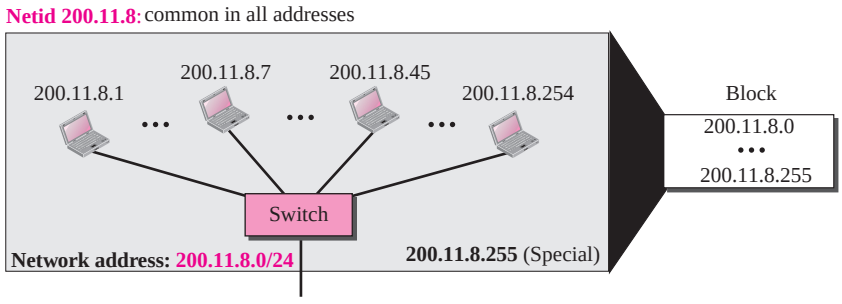
An address in a block is given as 200.11.8.45. Find the number of addresses in the block, the first address, and the last address.

Solution

Since 200 is between 192 and 223, the class of the address is C. The value of *n* for class C is 24. Figure 5.18 shows a possible configuration of the network that uses this block.

1. The number of addresses in this block is $N = 2^{32-n} = 2^{8} = 256$.
2. To find the first address, we keep the leftmost 24 bits and set the rightmost 8 bits all to 0s. The first address is 200.11.8.0/24. The first address is called the network address.
3. To find the last address, we keep the leftmost 24 bits and set the rightmost 8 bits all to 1s. The last address is 200.11.8.255.

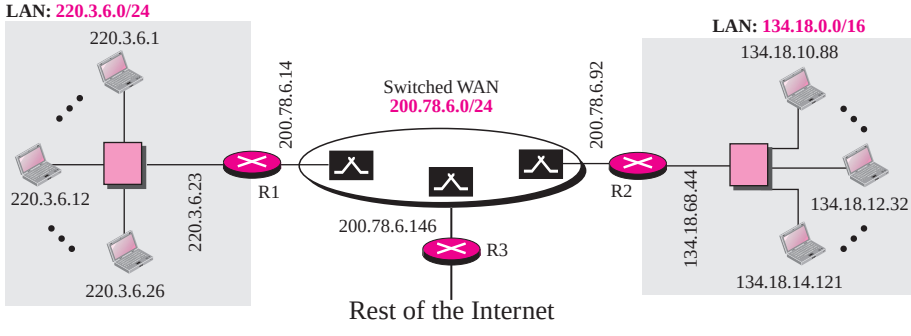
Figure 5.18 *Solution to Example 5.15*



An Example

Figure 5.19 shows a hypothetical part of an internet with three networks.

Figure 5.19 *Sample internet*



We have

1. A LAN with the network address 220.3.6.0 (class C).
2. A LAN with the network address 134.18.0.0 (class B).
3. A switched WAN (class C), such as Frame Relay or ATM, that can be connected to many routers. We have shown three. One router connects the WAN to the left LAN, one connects the WAN to the right LAN, and one connects the WAN to the rest of the internet.

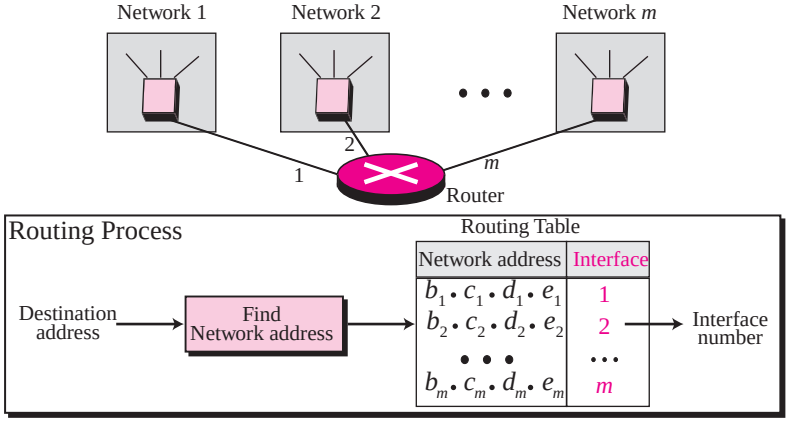
Network Address

The above three examples show that, given any address, we can find all information about the block. The first address, **network address,** is particularly important because it is used in routing a packet to its destination network. For the moment, let us assume that an internet is made of *m* networks and a router with *m* interfaces. When a packet arrives at the router from any source host, the router needs to know to which network the packet

should be sent; the router needs to know from which interface the packet should be sent out. When the packet arrives at the network, it reaches its destination host using another strategy that we discuss in later chapters. Figure 5.20 shows the idea. After the network address has been found, the router consults its routing table to find the corresponding interface from which the packet should be sent out. The network address is actually the identifier of the network; each network is identified by its network address.

The network address is the identifier of a network.

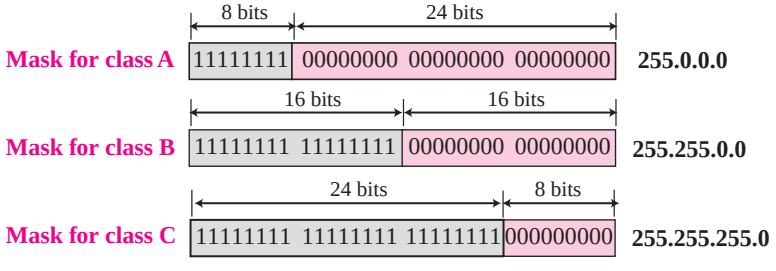
Figure 5.20 *Network address*



Network Mask

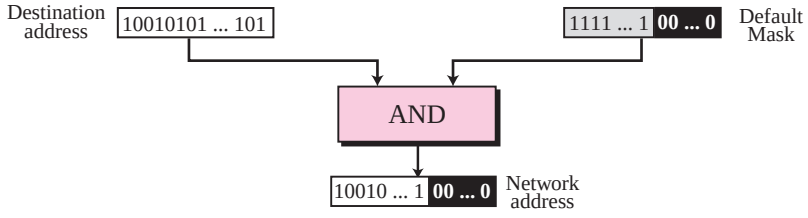
The methods we described previously for extracting the network address are mostly used to show the concept. The routers in the Internet normally use an algorithm to extract the network address from the destination address of a packet. To do this, we need a network mask. A **network mask** or a **default mask** in classful addressing is a 32-bit number with n leftmost bits all set to 1s and $(32 - n)$ rightmost bits all set to 0s. Since n is different for each class in classful addressing, we have three default masks in classful addressing as shown in Figure 5.21.

Figure 5.21 *Network mask*



To extract the network address from the destination address of a packet, a router uses the AND operation described in the previous section. When the destination address (or any address in the block) is ANDed with the default mask, the result is the network address (Figure 5.22). The router applies the AND operation on the binary (or hexadecimal representation) of the address and the mask, but when we show an example, we use the short cut discussed before and apply the mask on the dotted-decimal notation. The default mask can also be used to find the number of addresses in the block and the last address in the block, but we discuss these applications in classless addressing.

Figure 5.22 *Finding a network address using the default mask*



Example 5.16

A router receives a packet with the destination address 201.24.67.32. Show how the router finds the network address of the packet.

Solution

We assume that the router first finds the class of the address and then uses the corresponding default mask on the destination address, but we need to know that a router uses another strategy as we will discuss in the next chapter. Since the class of the address is B, we assume that the router applies the default mask for class B, 255.255.0.0 to find the network address.

Destination address	→	201	.	24	.	67	.	32
Default mask	→	255	.	255	.	0	.	0
Network address	→	201	.	24	.	0	.	0

We have used the first short cut as described in the previous section. The network address is 201.24.0.0 as expected.

Three-Level Addressing: Subnetting

As we discussed before, the IP addresses were originally designed with two levels of addressing. To reach a host on the Internet, we must first reach the network and then the host. It soon became clear that we need more than two hierarchical levels, for two reasons. First, an organization that was granted a block in class A or B needed to divide its large network into several subnetworks for better security and management. Second, since the blocks in class A and B were almost depleted and the blocks in class C were smaller than the needs of most organizations, an organization that has been granted a block in class A or B could divide the block into smaller subblocks and share them with

other organizations. The idea of splitting a block to smaller blocks is referred to as subnetting. In **subnetting,** a network is divided into several smaller subnetworks (subnets) with each subnetwork having its own subnetwork address.

Example 5.17

Three-level addressing can be found in the telephone system if we think about the local part of a telephone number as an exchange and a subscriber connection:

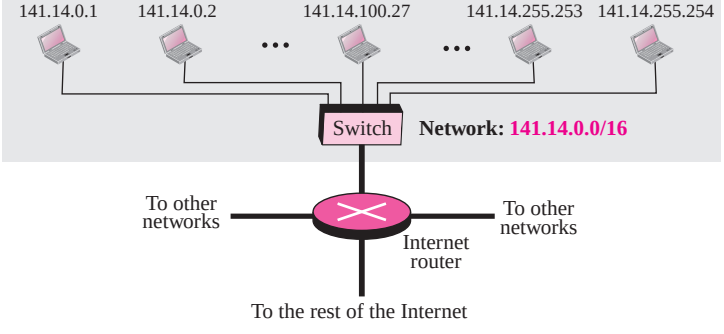
(626) 358 - 1301

in which 626 is the area code, 358 is the exchange, and 1301 is the subscriber connection.

Example 5.18

Figure 5.23 shows a network using class B addresses before subnetting. We have just one network with almost 2^{16} hosts. The whole network is connected, through one single connection, to one of the routers in the Internet. Note that we have shown /16 to show the length of the netid (class B).

Figure 5.23 *Example 5.18*



Example 5.19

Figure 5.24 shows the same network in Figure 5.23 after subnetting. The whole network is still connected to the Internet through the same router. However, the network has used a private router to divide the network into four subnetworks. The rest of the Internet still sees only one network; internally the network is made of four subnetworks. Each subnetwork can now have almost 2^{14} hosts. The network can belong to a university campus with four different schools (buildings). After subnetting, each school has its own subnetworks, but still the whole campus is one network for the rest of the Internet. Note that /16 and /18 show the length of the netid and subnetids.

Subnet Mask

We discussed the network mask (default mask) before. The network mask is used when a network is not subnetted. When we divide a network to several subnetworks, we need to create a subnetwork mask (or subnet mask) for each subnetwork. A subnetwork has subnetid and hostid as shown in Figure 5.25.

Figure 5.24 *Example 5.19*

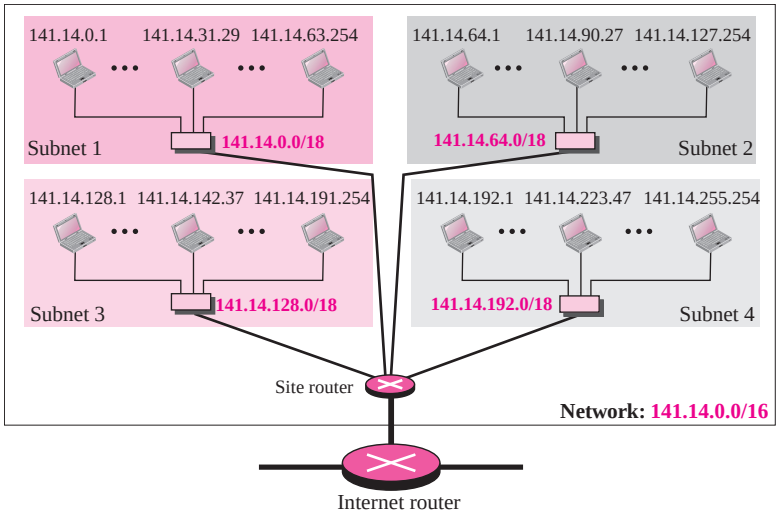
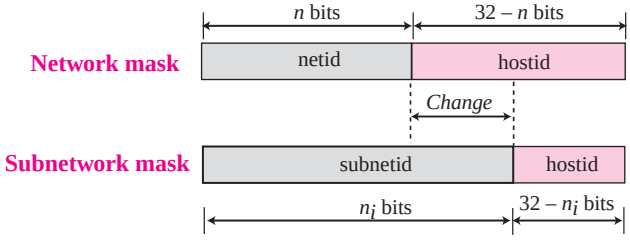


Figure 5.25 *Network mask and subnetwork mask*



Subnetting increases the length of the netid and decreases the length of hostid. When we divide a network to s number of subnetworks, each of equal numbers of hosts, we can calculate the subnetid for each subnetwork as

$$n_{sub} = n + \log_2 s$$

in which n is the length of netid, n_{sub} is the length of each subnetid, and s is the number of subnets which must be a power of 2.

Example 5.20

In Example 5.19, we divided a class B network into four subnetworks. The value of $n = 16$ and the value of $n_1 = n_2 = n_3 = n_4 = 16 + \log_2 4 = 18$. This means that the subnet mask has eighteen 1s and fourteen 0s. In other words, the subnet mask is 255.255.192.0 which is different from the network mask for class B (255.255.0.0).

Subnet Address

When a network is subnetted, the first address in the subnet is the identifier of the subnet and is used by the router to route the packets destined for that subnetwork. Given any address in the subnet, the router can find the subnet mask using the same procedure we discussed to find the network mask: ANDing the given address with the subnet mask. The short cuts we discussed in the previous section can be used to find the subnet address.

Example 5.21

In Example 5.19, we show that a network is divided into four subnets. Since one of the addresses in subnet 2 is 141.14.120.77, we can find the subnet address as:

Address	→	141	.	14	.	120	.	77
Mask	→	255	.	255	.	192	.	0
Subnet Address	→	141	.	14	.	64	.	0

The values of the first, second, and fourth bytes are calculated using the first short cut for AND operation. The value of the third byte is calculated using the second short cut for the AND operation.

Address (120)	0	+	64	+	32	+	16	+	8	+	0	+	0	+	0
Mask (192)	128	+	64	+	0	+	0	+	0	+	0	+	0	+	0
Result (64)	0	+	64	+	0	+	0	+	0	+	0	+	0	+	0

Designing Subnets

We show how to design a subnet when we discuss classless addressing. Since classful addressing is a special case of classless addressing, what is discussed later can also be applied to classful addressing.

Supernetting

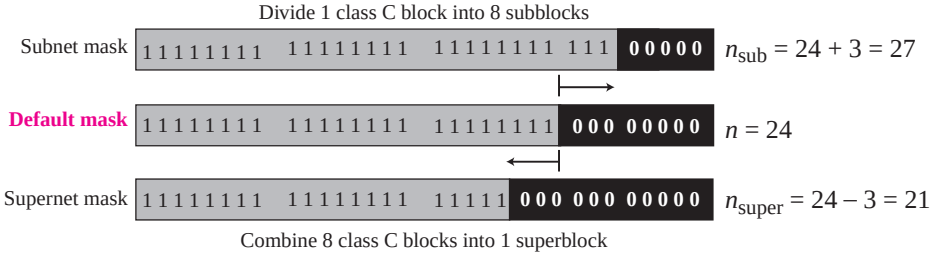
Subnetting could not completely solve address depletion problems in classful addressing because most organizations did not want to share their granted blocks with others. Since class C blocks were still available but the size of the block did not meet the requirement of new organizations that wanted to join the Internet, one solution was **supernetting.** In supernetting, an organization can combine several class C blocks to create a larger range of addresses. In other words, several networks are combined to create a supernetwork. By doing this, an organization can apply for several class C blocks instead of just one. For example, an organization that needs 1000 addresses can be granted four class C blocks.

Supernet Mask

A **supernet mask** is the reverse of a subnet mask. A subnet mask for class C has more 1s than the default mask for this class. A supernet mask for class C has less 1s than the default mask for this class.

Figure 5.26 shows the difference between a subnet mask and a supernet mask. A subnet mask that divides a block into eight subblocks has three more 1s ($2^3 = 8$) than the default mask; a supernet mask that combines eight blocks into one superblock has three less 1s than the default mask.

Figure 5.26 *Comparison of subnet, default, and supernet masks*



In supernetting, the number of class C addresses that can be combined to make a supernet needs to be a power of 2. The length of the supernetid can be found using the formula

$$n_{super} = n - \log_2 c$$

in which n_{super} defines the length of the supernetid in bits and c defines the number of class C blocks that are combined.

Unfortunately, supernetting provided two new problems: First, the number of blocks to combine needs to be a power of 2, which means an organization that needed seven blocks should be granted at least eight blocks (address wasting). Second, supernetting and subnetting really complicated the routing of packets in the Internet.

5.3 CLASSLESS ADDRESSING

Subnetting and supernetting in classful addressing did not really solve the address depletion problem and made the distribution of addresses and the routing process more difficult. With the growth of the Internet, it was clear that a larger address space was needed as a long-term solution. The larger address space, however, requires that the length of IP addresses to be increased, which means the format of the IP packets needs to be changed. Although the long-range solution has already been devised and is called IPv6 (see Chapters 26 to 28), a short-term solution was also devised to use the same address space but to change the distribution of addresses to provide a fair share to each organization. The short-term solution still uses IPv4 addresses, but it is called *classless* addressing. In other words, the class privilege was removed from the distribution to compensate for the address depletion.

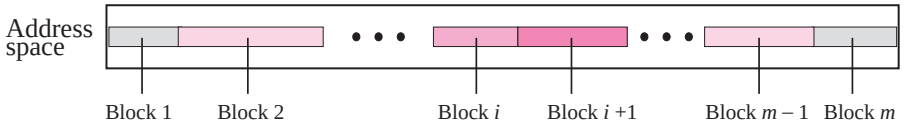
There was another motivation for classless addressing. During the 1990s, Internet service providers (ISPs) came into prominence. An ISP is an organization that provides Internet access for individuals, small businesses, and midsize organizations that do not want to create an Internet site and become involved in providing Internet services (such as e-mail services) for their employees. An ISP can provide these services. An ISP is granted a large range of addresses and then subdivides the addresses (in groups of 1, 2, 4, 8, 16, and so on), giving a range of addresses to a household or a small business. The customers are connected via a dial-up modem, DSL, or cable modem to the ISP. However, each customer needs some IPv4 addresses.

In 1996, the Internet authorities announced a new architecture called classless addressing. In classless addressing, variable-length blocks are used that belong to no classes. We can have a block of 1 address, 2 addresses, 4 addresses, 128 addresses, and so on.

Variable-Length Blocks

In classful addressing the whole address space was divided into five classes. Although each organization was granted one block in class A, B, or C, the size of the blocks was predefined; the organization needed to choose one of the three block sizes. The only block in class D and the only block in class E were reserved for a special purpose. In classless addressing, the whole address space is divided into variable length blocks. Theoretically, we can have a block of $2^0, 2^1, 2^2, \ldots, 2^{32}$ addresses. The only restriction, as we discuss later, is that the number of addresses in a block needs to be a power of 2. An organization can be granted one block of addresses. Figure 5.27 shows the division of the whole address space into nonoverlapping blocks.

Figure 5.27 *Variable-length blocks in classless addressing*

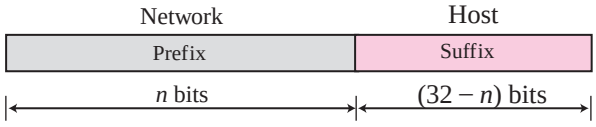


Two-Level Addressing

In classful addressing, two-level addressing was provided by dividing an address into *netid* and *hostid*. The netid defined the network; the hostid defined the host in the network. The same idea can be applied in classless addressing. When an organization is granted a block of addresses, the block is actually divided into two parts, the **prefix** and the **suffix.** The prefix plays the same role as the netid; the suffix plays the same role as the hostid. All addresses in the block have the same prefix; each address has a different suffix. Figure 5.28 shows the prefix and suffix in a classless block.

In classless addressing, the prefix defines the network and the suffix defines the host.

Figure 5.28 *Prefix and suffix*



In classful addressing, the length of the netid, *n*, depends on the class of the address; it can be only 8, 16, or 24. In classless addressing, the length of the prefix, *n*, depends on the size of the block; it can be 0, 1, 2, 3, . . . , 32. In classless addressing, the value of *n* is referred to as **prefix length;** the value of 32 – *n* is referred to as **suffix length.**

The prefix length in classless addressing can be 1 to 32.

Example 5.22

What is the prefix length and suffix length if the whole Internet is considered as one single block with 4,294,967,296 addresses?

Solution

In this case, the prefix length is 0 and the suffix length is 32. All 32 bits vary to define 2^{32} = 4,294,967,296 hosts in this single block.

Example 5.23

What is the prefix length and suffix length if the Internet is divided into 4,294,967,296 blocks and each block has one single address?

Solution

In this case, the prefix length for each block is 32 and the suffix length is 0. All 32 bits are needed to define 2^{32} = 4,294,967,296 blocks. The only address in each block is defined by the block itself.

Example 5.24

The number of addresses in a block is inversely related to the value of the prefix length, *n*. A small *n* means a larger block; a large *n* means a small block.

Slash Notation

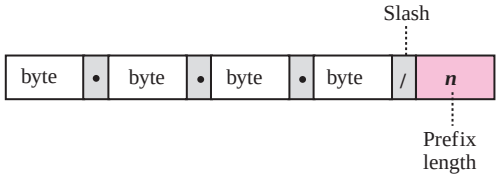
The netid length in classful addressing or the prefix length in classless addressing play a very important role when we need to extract the information about the block from a given address in the block. However, there is a difference here in classful and classless addressing.

- In classful addressing, the netid length is inherent in the address. Given an address, we know the class of the address that allows us to find the netid length (8, 16, or 24).

- In classless addressing, the prefix length cannot be found if we are given only an address in the block. The given address can belong to a block with any prefix length.

In classless addressing, we need to include the prefix length to each address if we need to find the block of the address. In this case, the prefix length, *n*, is added to the address separated by a slash. The notation is informally referred to as **slash notation.** An address in classless addressing can then be represented as shown in Figure 5.29.

Figure 5.29 *Slash notation*



The slash notation is formally referred to as **classless interdomain routing** or **CIDR** (pronounced cider) notation.

In classless addressing, we need to know one of the addresses in the block and the prefix length to define the block.

Example 5.25

In classless addressing, an address cannot per se define the block the address belongs to. For example, the address 230.8.24.56 can belong to many blocks some of them are shown below with the value of the prefix associated with that block:

Prefix length:16	→	Block:	230.8.0.0	to	230.8.255.255
Prefix length:20	→	Block:	230.8.16.0	to	230.8.31.255
Prefix length:26	→	Block:	230.8.24.0	to	230.8.24.63
Prefix length:27	→	Block:	230.8.24.32	to	230.8.24.63
Prefix length:29	→	Block:	230.8.24.56	to	230.8.24.63
Prefix length:31	→	Block:	230.8.24.56	to	230.8.24.57

Network Mask

The idea of network mask in classless addressing is the same as the one in classful addressing. A network mask is a 32-bit number with the *n* leftmost bits all set to 0s and the rest of the bits all set to 1s.

Example 5.26

The following addresses are defined using slash notations.

a. In the address 12.23.24.78/8, the network mask is 255.0.0.0. The mask has eight 1s and twenty-four 0s. The prefix length is 8; the suffix length is 24.

b. In the address 130.11.232.156/16, the network mask is 255.255.0.0. The mask has sixteen 1s and sixteen 0s.The prefix length is 16; the suffix length is 16.

c. In the address 167.199.170.82/27, the network mask is 255.255.255.224. The mask has twenty-seven 1s and five 0s. The prefix length is 27; the suffix length is 5.

Extracting Block Information

An address in slash notation (CIDR) contains all information we need about the block: the first address (network address), the number of addresses, and the last address. These three pieces of information can be found as follows:

- The number of addresses in the block can be found as:

$$N = 2^{32-n}$$

in which n is the prefix length and N is the number of addresses in the block.

- The first address (network address) in the block can be found by ANDing the address with the network mask:

First address = (any address) AND (network mask)

Alternatively, we can keep the n leftmost bits of any address in the block and set the 32– n bits to 0s to find the first address.

- The last address in the block can be found by either adding the first address with the number of addresses or, directly, by ORing the address with the complement (NOTing) of the network mask:

Last address = (any address) OR [NOT (network mask)]

Alternatively, we can keep the n leftmost bits of any address in the block and set the 32 – n bits to 1s to find the last address.

Example 5.27

One of the addresses in a block is 167.199.170.82/27. Find the number of addresses in the network, the first address, and the last address.

Solution

The value of n is 27. The network mask has twenty-seven 1s and five 0s. It is 255.255.255.240.

a. The number of addresses in the network is $2^{32-n} = 2^{32-n} = 2^5 = 32$.

b. We use the AND operation to find the first address (network address).The first address is 167.199.170.64/27.

Address in binary:	**10100111 11000111 10101010 01010010**
Network mask:	**11111111 11111111 11111111 11100000**
First address:	**10100111 11000111 10101010 01000000**

c. To find the last address, we first find the complement of the network mask and then OR it with the given address: The last address is 167.199.170.95/27.

Address in binary:	10100111	11000111	10101010	01010010
Complement of network mask:	00000000	00000000	00000000	00011111
Last address:	10100111	11000111	10101010	01011111

Example 5.28

One of the addresses in a block is 17.63.110.114/24. Find the number of addresses, the first address, and the last address in the block.

Solution

The network mask is 255.255.255.0.

a. The number of addresses in the network is $2^{32-24} = 256$.

b. To find the first address, we use the short cut methods discussed early in the chapter.

Address:	17	63	110	114
Network mask:	255	255	255	0
First address (AND):	17	63	110	0

The first address is 17.63.110.0/24.

c. To find the last address, we use the complement of the network mask and the first short cut method we discussed before. The last address is 17.63.110.255/24.

Address:	17	63	110	114
Complement of the mask (NOT):	0	0	0	255
Last address (OR):	17	63	110	255

Example 5.29

One of the addresses in a block is 110.23.120.14/20. Find the number of addresses, the first address, and the last address in the block.

Solution

The network mask is 255.255.240.0.

a. The number of addresses in the network is $2^{32-20} = 4096$.

b. To find the first address, we apply the first short cut to bytes 1, 2, and 4 and the second short cut to byte 3. The first address is 110.23.112.0/20.

Address:	110	23	120	14
Network mask:	255	255	240	0
First address (AND):	110	23	112	0

c. To find the last address, we apply the first short cut to bytes 1, 2, and 4 and the second short cut to byte 3. The OR operation is applied to the complement of the mask. The last address is 110.23.127.255/20.

Address:	110	.	23	.	120	.	14
Network mask:	0	.	0	.	15	.	255
Last address (OR):	110	.	23	.	127	.	255

Block Allocation

The next issue in classless addressing is block allocation. How are the blocks allocated? The ultimate responsibility of block allocation is given to a global authority called the Internet Corporation for Assigned Names and Addresses (ICANN). However, ICANN does not normally allocate addresses to individual Internet users. It assigns a large block of addresses to an ISP (or a larger organization that is considered an ISP in this case). For the proper operation of the CIDR, three restrictions need to be applied to the allocated block.

1. The number of requested addresses, N, needs to be a power of 2. This is needed to provide an integer value for the prefix length, n (see the second restriction). The number of addresses can be 1, 2, 4, 8, 16, and so on.
2. The value of prefix length can be found from the number of addresses in the block. Since $N = 2^{32 - n}$, then $n = \log_2 (2^{32}/N) = 32 - \log_2 N$. That is the reason why N needs to be a power of 2.
3. The requested block needs to be allocated where there are a contiguous number of unallocated addresses in the address space. However, there is a restriction on choosing the beginning addresses of the block. The beginning address needs to be divisible by the number of addresses in the block. To see this restriction, we can show that the beginning address can be calculated as $X \times 2^{n - 32}$ in which X is the decimal value of the prefix. In other words, the beginning address is $X \times N$.

Example 5.30

An ISP has requested a block of 1000 addresses. The following block is granted.

a. Since 1000 is not a power of 2, 1024 addresses are granted ($1024 = 2^{10}$).
b. The prefix length for the block is calculated as $n = 32 - \log_2 1024 = 22$.
c. The beginning address is chosen as 18.14.12.0 (which is divisible by 1024).

The granted block is 18.14.12.0/22. The first address is 18.14.12.0/22 and the last address is 18.14.15.255/22.

Relation to Classful Addressing

All issues discussed for classless addressing can be applied to classful addressing. As a matter of fact, classful addressing is a special case of the classless addressing in which the blocks in class A, B, and C have the prefix length $n_A = 8$, $n_B = 16$, and $n_C = 24$. A block in classful addressing can be easily changed to a block in class addressing if we use the prefix length defined in Table 5.1.

Table 5.1 *Prefix length for classful addressing*

Class	*Prefix length*	*Class*	*Prefix length*
A	/8	D	/4
B	/16	E	/4
C	/24		

Example 5.31

Assume an organization has given a class A block as 73.0.0.0 in the past. If the block is not revoked by the authority, the classless architecture assumes that the organization has a block 73.0.0.0/8 in classless addressing.

Subnetting

Three levels of hierarchy can be created using subnetting. An organization (or an ISP) that is granted a range of addresses may divide the range into several subranges and assign each subrange to a **subnetwork** (or **subnet**). The concept is the same as we discussed for classful addressing. Note that nothing stops the organization from creating more levels. A subnetwork can be divided into several sub-subnetworks. A sub-subnetwork can be divided into several sub-sub-subnetworks. And so on.

Designing Subnets

The subnetworks in a network should be carefully designed to enable the routing of packets. We assume the total number of addresses granted to the organization is N, the prefix length is n, the assigned number of addresses to each subnetwork is N_{sub}, the prefix length for each subnetwork is n_{sub}, and the total number of subnetworks is s. Then, the following steps need to be carefully followed to guarantee the proper operation of the subnetworks.

1. The number of addresses in each subnetwork should be a power of 2.
2. The prefix length for each subnetwork should be found using the following formula:

$$n_{sub} = n + \log_2 (N/N_{sub})$$

3. The starting address in each subnetwork should be divisible by the number of addresses in that subnetwork. This can be achieved if we first assign addresses to larger networks.

The restrictions applied in allocating addresses for a subnetwork are parallel to the ones used to allocate addresses for a network.

Finding Information about Each Subnetwork

After designing the subnetworks, the information about each subnetwork, such as first and last address, can be found using the process we described to find the information about each network in the Internet.

Example 5.32

An organization is granted the block 130.34.12.64/26. The organization needs four subnetworks, each with an equal number of hosts. Design the subnetworks and find the information about each network.

Solution

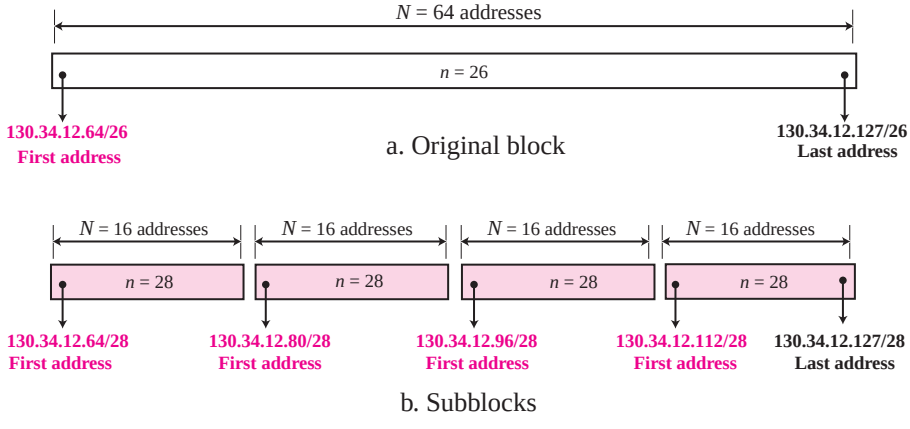
The number of addresses for the whole network can be found as $N = 2^{32 - 26} = 64$. Using the process described in the previous section, the first address in the network is 130.34.12.64/26 and the last address is 130.34.12.127/26. We now design the subnetworks:

1. We grant 16 addresses for each subnetwork to meet the first requirement (64/16 is a power of 2).
2. The **subnetwork** mask for each subnetwork is:

$$n_1 = n_2 = n_3 = n_4 = n + \log_2 (N/N_i) = 26 + \log_2 4 = 28$$

3. We grant 16 addresses to each subnet starting from the first available address. Figure 5.30 shows the subblock for each subnet. Note that the starting address in each subnetwork is divisible by the number of addresses in that subnetwork.

Figure 5.30 *Solution to Example 5.32*



Example 5.33

An organization is granted a block of addresses with the beginning address 14.24.74.0/24. The organization needs to have 3 subblocks of addresses to use in its three subnets as shown below:

- One subblock of 120 addresses.
- One subblock of 60 addresses.
- One subblock of 10 addresses.

Solution

There are $2^{32 - 24} = 256$ addresses in this block. The first address is 14.24.74.0/24; the last address is 14.24.74.255/24.

a. The number of addresses in the first subblock is not a power of 2. We allocate 128 addresses. The first can be used as network address and the last as the special address. There are still 126 addresses available. The subnet mask for this subnet can be found as $n_1 = 24 + \log_2 (256/128) = 25$. The first address in this block is 14.24.74.0/25; the last address is 14.24.74.127/25.

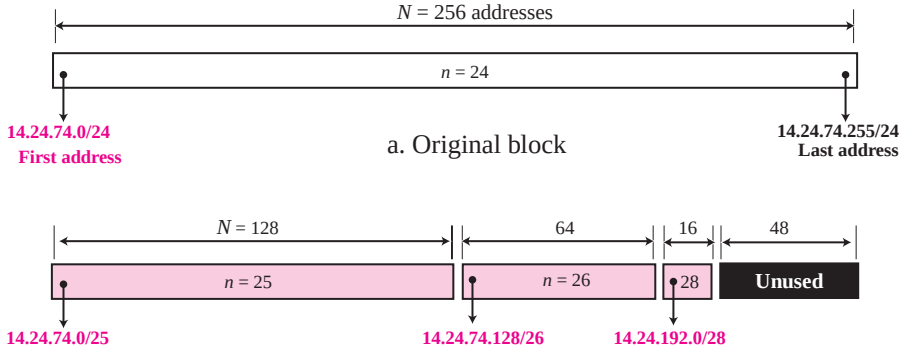
b. The number of addresses in the second subblock is not a power of 2 either. We allocate 64 addresses. The first can be used as network address and the last as the special address. There are still 62 addresses available. The subnet mask for this subnet can be found as $n_1 = 24 + \log_2 (256/64) = 26$. The first address in this block is 14.24.74.128/26; the last address is 14.24.74.191/26.

c. The number of addresses in the third subblock is not a power of 2 either. We allocate 16 addresses. The first can be used as network address and the last as the special address. There are still 14 addresses available. The subnet mask for this subnet can be found as $n_1 = 24 + \log_2 (256/16) = 28$. The first address in this block is 14.24.74.192/28; the last address is 14.24.74.207/28.

d. If we add all addresses in the previous subblocks, the result is 208 addresses, which means 48 addresses are left in reserve. The first address in this range is 14.24.74.209. The last address is 14.24.74.255. We don't know about the prefix length yet.

e. Figure 5.31 shows the configuration of blocks. We have shown the first address in each block.

Figure 5.31 *Solution to Example 5.33*



a. Original block

b. Subblocks

Example 5.34

Assume a company has three offices: Central, East, and West. The Central office is connected to the East and West offices via private, point-to-point WAN lines. The company is granted a block of 64 addresses with the beginning address 70.12.100.128/26. The management has decided to allocate 32 addresses for the Central office and divides the rest of addresses between the two other offices.

1. The number of addresses are assigned as follows:

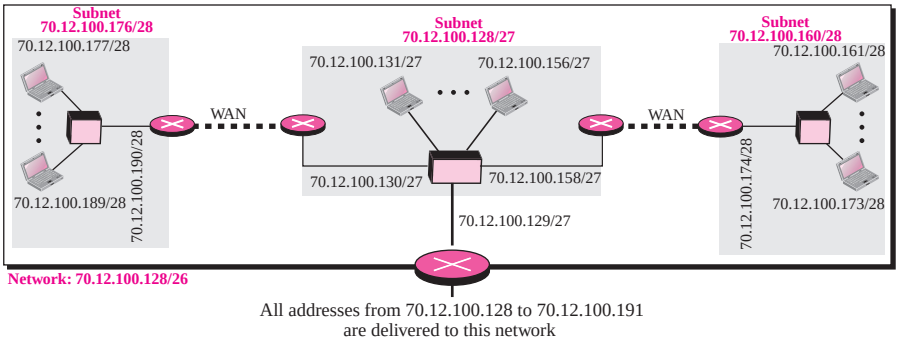
Central office $N_c = 32$	East office $N_e = 16$	West office $N_w = 16$

2. We can find the prefix length for each subnetwork:

$n_c = n + \log_2(64/32) = 27$	$n_e = n + \log_2(64/16) = 28$	$n_w = n + \log_2(64/16) = 28$

3. Figure 5.32 shows the configuration designed by the management. The Central office uses addresses 70.12.100.128/27 to 70.12.100.159/27. The company has used three of these addresses for the routers and has reserved the last address in the subblock. The East office uses the addresses 70.12.100.160/28 to 70.12.100.175/28. One of these addresses is used for the router and the company has reserved the last address in the subblock. The West office uses the addresses 70.12.100.160/28 to 70.12.100.175/28. One of these addresses is used for the router and the company has reserved the last address in the subblock. The company uses no address for the point-to-point connections in WANs.

Figure 5.32 *Example 14*



All addresses from 70.12.100.128 to 70.12.100.191 are delivered to this network

Address Aggregation

One of the advantages of CIDR architecture is **address aggregation.** ICANN assigns a large **block of addresses** to an ISP. Each ISP in turn divides its assigned block into smaller subblocks and grants the subblocks to its customers; many blocks of addresses are aggregated in one block and granted to one ISP.

Example 5.35

An ISP is granted a block of addresses starting with 190.100.0.0/16 (65,536 addresses). The ISP needs to distribute these addresses to three groups of customers as follows:

- The first group has 64 customers; each needs approximately 256 addresses.
- The second group has 128 customers; each needs approximately 128 addresses.
- The third group has 128 customers; each needs approximately 64 addresses.

We design the subblocks and find out how many addresses are still available after these allocations.

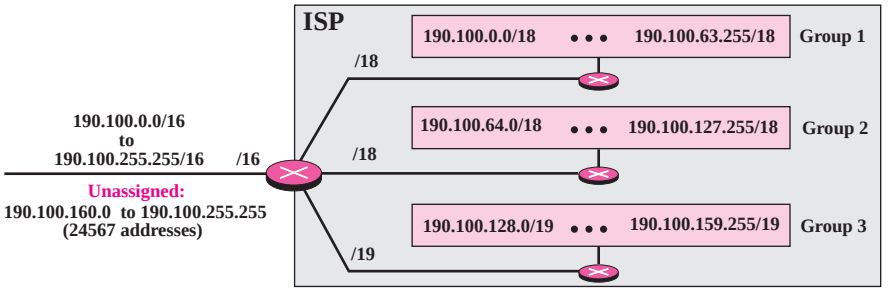
Solution

Let us solve the problem in two steps. In the first step, we allocate a subblock of addresses to each group. The total number of addresses allocated to each group and the prefix length for each subblock can found as

Group 1: $64 \times 256 = 16{,}384$	$n_1 = 16 + \log_2 (65536/16384) = 18$
Group 2: $128 \times 128 = 16{,}384$	$n_2 = 16 + \log_2 (65536/16384) = 18$
Group 3: $128 \times 64 = 8192$	$n_3 = 16 + \log_2 (65536/8192) = 19$

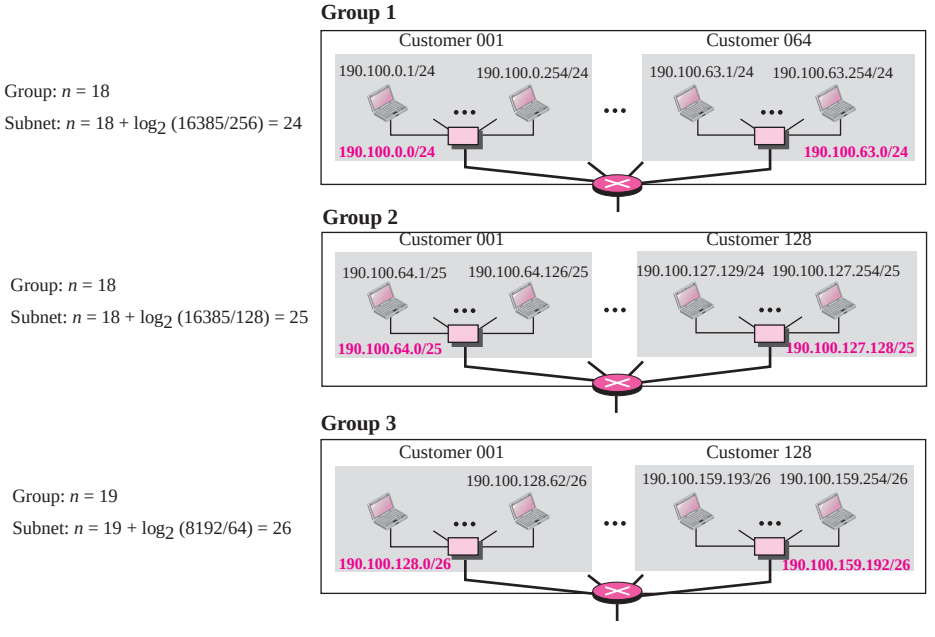
Figure 5.33 shows the design for the first hierarchical level.

Figure 5.33 *Solution to Example 5.35: first step*



Now we can think about each group. The prefix length changes for the networks in each group depending on the number of addresses used in each network. Figure 5.34 shows the second level of the hierarchy. Note that we have used the first address for each customer as the subnet address and have reserved the last address as a special address.

Figure 5.34 *Solution to Example 5.35: second step*



5.4 SPECIAL ADDRESSES

In classful addressing some addresses were reserved for special purposes. The classless addressing scheme inherits some of these special addresses from classful addressing.

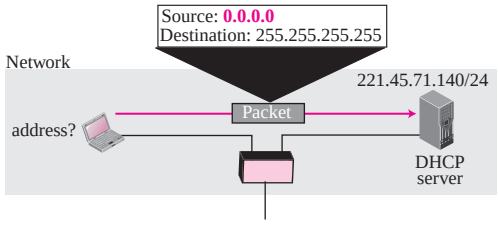
Special Blocks

Some blocks of addresses are reserved for special purposes.

All-Zeros Address

The block 0.0.0.0/32, which contains only one single address, is reserved for communication when a host needs to send an IPv4 packet but it does not know its own address. This is normally used by a host at bootstrap time when it does not know its IPv4 address. The host sends an IPv4 packet to a bootstrap server (called DHCP server as discussed in Chapter 18) using this address as the source address and a **limited broadcast address** as the destination address to find its own address (see Figure 5.35).

Figure 5.35 *Examples of using the all-zeros address*



All-Ones Address: Limited Broadcast Address

The block 255.255.255.255/32, which contains one single address, is reserved for limited broadcast address in the current network. A host that wants to send a message to every other host can use this address as a destination address in an IPv4 packet. However, a router will block a packet having this type of address to confine the broadcasting to the local network. In Figure 5.36, a host sends a datagram using a destination IPv4 address consisting of all 1s. All devices on this network receive and process this datagram.

Loopback Addresses

The block 127.0.0.0/8 is used for the **loopback address,** which is an address used to test the software on a machine. When this address is used, a packet never leaves the machine; it simply returns to the protocol software. It can be used to test the IPv4 software. For example, an application such as "ping" can send a packet with a loopback address as the destination address to see if the IPv4 software is capable of receiving and processing a packet. As another example, the loopback address can be used by a *client process* (a running application program) to send a message to a server process on the same machine. Note that this can be used only as a destination address in an IPv4 packet (see Figure 5.37).

Figure 5.36 *Example of limited broadcast address*

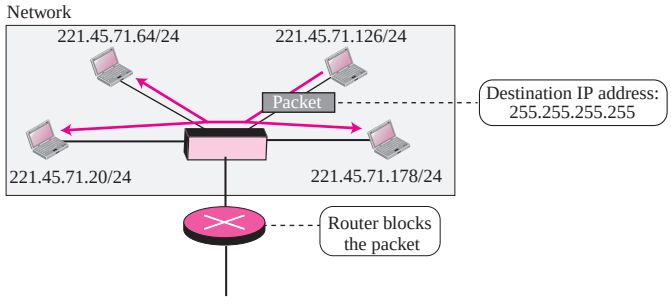
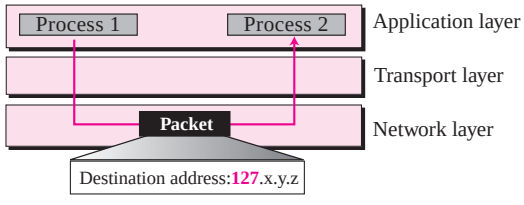


Figure 5.37 *Example of loopback address*



Private Addresses

A number of blocks are assigned for private use. They are not recognized globally. These blocks are depicted in Table 5.2. These addresses are used either in isolation or in connection with network address translation techniques (see NAT section later in this chapter).

Table 5.2 *Addresses for private networks*

Block	*Number of addresses*	*Block*	*Number of addresses*
10.0.0.0/8	16,777,216	192.168.0.0/16	65,536
172.16.0.0/12	1,047,584	169.254.0.0/16	65,536

Multicast Addresses

The block 224.0.0.0/4 is reserved for multicast communication. We discuss multicasting in Chapter 12 in detail.

Special Addresses in Each block

It is not mandatory, but it is recommended, that some addresses in a block be used for special addresses. These addresses are not assigned to any host. However, if a block (or subblock) is so small, we cannot afford to use part of the addresses as special addresses.

Network Address

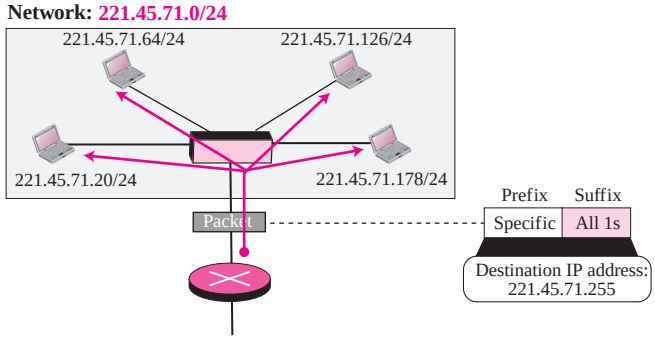
We have already discussed network addresses. The first address (with the suffix set all to 0s) in a block defines the network address. It actually defines the network itself

(cabling) and not any host in the network. Of course, the first address in a subnetwork is called the subnetwork address and plays the same role.

Direct Broadcast Address

The last address in a block or subblock (with the suffix set all to 1s) can be used as a **direct broadcast address.** This address is usually used by a router to send a packet to all hosts in a specific network. All hosts will accept a packet having this type of destination address. Note that this address can be used only as a destination address in an IPv4 packet. In Figure 5.38, the router sends a datagram using a destination IPv4 address with a suffix of all 1s. All devices on this network receive and process the datagram.

Figure 5.38 *Example of a direct broadcast address*



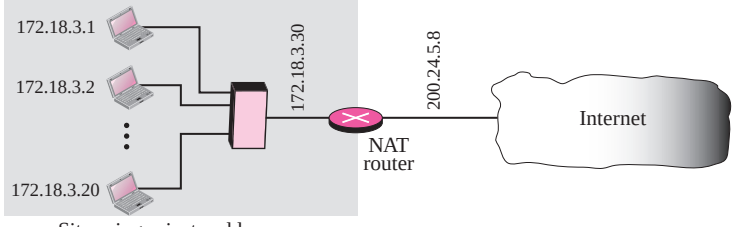
5.5 NAT

The distribution of addresses through ISPs has created a new problem. Assume that an ISP has granted a small range of addresses to a small business or a household. If the business grows or the household needs a larger range, the ISP may not be able to grant the demand because the addresses before and after the range may have already been allocated to other networks. In most situations, however, only a portion of computers in a small network need access to the Internet simultaneously. This means that the number of allocated addresses does not have to match the number of computers in the network. For example, assume a small business with 20 computers in which the maximum number of computers that access the Internet simultaneously is only 5. Most of the computers are either doing some task that does not need Internet access or communicating with each other. This small business can use the TCP/IP protocol for both internal and universal communication. The business can use 20 (or 25) addresses from the private block addresses discussed before for internal communication; five addresses for universal communication can be assigned by the ISP.

A technology that can provide the mapping between the private and universal addresses, and at the same time, support virtual private networks that we discuss in Chapter 30, is **network address translation (NAT).** The technology allows a site to use a set of private addresses for internal communication and a set of global Internet addresses

(at least one) for communication with the rest of the world. The site must have only one single connection to the global Internet through a NAT-capable router that runs NAT software. Figure 5.39 shows a simple implementation of NAT.

Figure 5.39 *NAT*

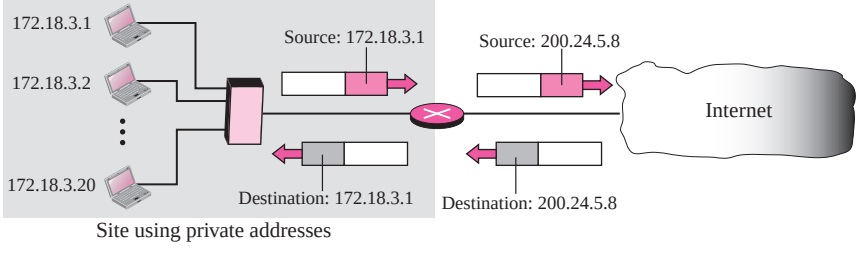


As the figure shows, the private network uses private addresses. The router that connects the network to the global address uses one private address and one global address. The private network is transparent to the rest of the Internet; the rest of the Internet sees only the NAT router with the address 200.24.5.8.

Address Translation

All of the outgoing packets go through the NAT router, which replaces the *source address* in the packet with the global NAT address. All incoming packets also pass through the NAT router, which replaces the *destination address* in the packet (the NAT router global address) with the appropriate private address. Figure 5.40 shows an example of address translation.

Figure 5.40 *Address translation*



Translation Table

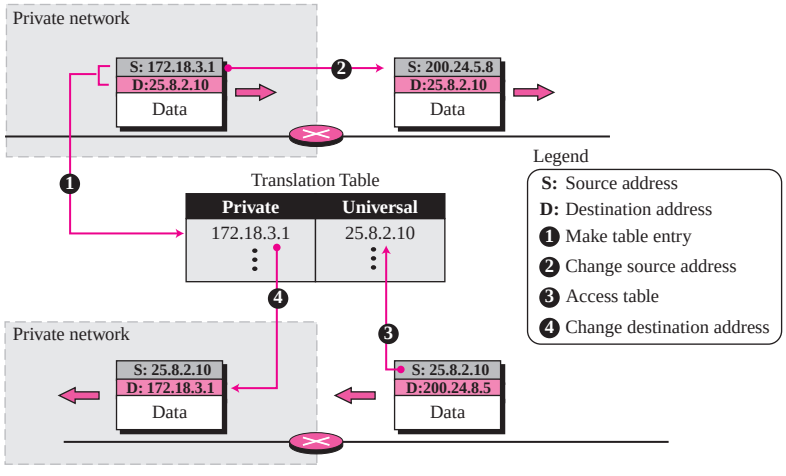
The reader may have noticed that translating the source addresses for an outgoing packet is straightforward. But how does the NAT router know the destination address for a packet coming from the Internet? There may be tens or hundreds of private IP

addresses, each belonging to one specific host. The problem is solved if the NAT router has a **translation table.**

Using One IP Address

In its simplest form, a translation table has only two columns: the private address and the external address (destination address of the packet). When the router translates the source address of the outgoing packet, it also makes note of the destination address—where the packet is going. When the response comes back from the destination, the router uses the source address of the packet (as the external address) to find the private address of the packet. Figure 5.41 shows the idea.

Figure 5.41 *Translation*



In this strategy, communication must always be initiated by the private network. The NAT mechanism described requires that the private network start the communication. As we will see, NAT is used mostly by ISPs that assign one single address to a customer. The customer, however, may be a member of a private network that has many private addresses. In this case, communication with the Internet is always initiated from the customer site, using a client program such as HTTP, TELNET, or FTP to access the corresponding server program. For example, when e-mail that originates from a non-customer site is received by the ISP e-mail server, it is stored in the mailbox of the customer until retrieved with a protocol such as POP.

A private network cannot run a server program for clients outside of its network if it is using NAT technology.

Using a Pool of IP Addresses

Using only one global address by the NAT router allows only one private-network host to access the same external host. To remove this restriction, the NAT router can use a

pool of global addresses. For example, instead of using only one global address (200.24.5.8), the NAT router can use four addresses (200.24.5.8, 200.24.5.9, 200.24.5.10, and 200.24.5.11). In this case, four private-network hosts can communicate with the same external host at the same time because each pair of addresses defines a connection. However, there are still some drawbacks. No more than four connections can be made to the same destination. No private-network host can access two external server programs (e.g., HTTP and TELNET) at the same time. And, likewise, two private-network hosts cannot access the same external server program (e.g., HTTP or TELNET) at the same time.

Using Both IP Addresses and Port Addresses

To allow a many-to-many relationship between private-network hosts and external server programs, we need more information in the translation table. For example, suppose two hosts inside a private network with addresses 172.18.3.1 and 172.18.3.2 need to access the HTTP server on external host 25.8.3.2. If the translation table has five columns, instead of two, that include the source and destination port addresses and the transport layer protocol, the ambiguity is eliminated. Table 5.3 shows an example of such a table.

Table 5.3 *Five-column translation table*

Private Address	*Private Port*	*External Address*	*External Port*	*Transport Protocol*
172.18.3.1	1400	25.8.3.2	80	TCP
172.18.3.2	1401	25.8.3.2	80	TCP
…	…	…	…	…

Note that when the response from HTTP comes back, the combination of source address (25.8.3.2) and destination port address (1400) defines the private network host to which the response should be directed. Note also that for this translation to work, the ephemeral port addresses (1400 and 1401) must be unique.

5.6 FURTHER READING

For more details about subjects discussed in this chapter, we recommend the following books and RFCs. The items enclosed in brackets refer to the reference list at the end of the book.

Books

Several books give thorough coverage of materials discussed in this chapter. We recommend [Com 06], [Tan 03], [Koz 05], [Ste 95].

RFCs

Several RFCs deal with IPv4 addressing including RFC 917, RFC 927, RFC 930, RFC 932, RFC 940, RFC 950, RFC 1122, and RFC 1519.

5.7 KEY TERMS

address aggregation
address space
binary notation
block of addresses
class A address
class B address
class C address
class D address
class E address
classful addressing
classless addressing
classless interdomain routing (CIDR)
default mask
direct broadcast address
dotted-decimal notation
hexadecimal notation
hostid
IP address
limited broadcast address
loopback address
netid
network address
Network Address Translation (NAT)
network mask
prefix
prefix length
slash notation
subnet
subnet mask
subnetting
subnetwork
suffix
suffix length
supernet mask
supernetting
translation table

5.8 SUMMARY

- The identifier used in the IP layer of the TCP/IP protocol suite is called the Internet address or IP address. An IPv4 address is 32 bits long. An address space is the total number of addresses used by the protocol. The address space of IPv4 is 2^{32} or 4,294,967,296.
- In classful addressing, the IPv4 address space is divided into five classes: A, B, C, D, and E. An organization is granted a block in one of the three classes, A, B, or C. Classes D and E is reserved for special purposes. An IP address in classes A, B, and C is divided into netid and hostid.
- In classful addressing, the first address in the block is called the network address. It defines the network to which an address belongs. The network address is used in routing a packet to its destination network.
- A network mask or a default mask in classful addressing is a 32-bit number with n leftmost bits all set to 1s and $(32 - n)$ rightmost bits all set to 0s. It is used by a router to find the network address from the destination address of a packet.
- The idea of splitting a network into smaller subnetworks is called subnetting. A subnetwork mask, like a network mask, is used to find the subnetwork address when a destination IP address is given. In supernetting, an organization can combine several class C blocks to create a larger range of addresses.
- In 1996, the Internet authorities announced a new architecture called classless addressing or CIDR that allows an organization to have a block of addresses of any size as long as the size of the block is a power of two.

- The address in classless addressing is also divided into two parts: the prefix and the suffix. The prefix plays the same role as the netid; the suffix plays the same role as the hostid. All addresses in the block have the same prefix; each address has a different suffix.
- Some of the blocks in IPv4 are reserved for special purposes. In addition, some addresses in a block are traditionally used for special addresses. These addresses are not assigned to any host.
- To improve the distribution of addresses, NAT technology has been created to allow the separation of private addresses in a network from the global addresses used in the Internet. A translation table can translate the private addresses, selected from the blocks allocated for this purpose, to global addresses. The translation table also translates the IP addresses as well as the port number for mapping from the private to global addresses and vice versa.

5.9 PRACTICE SET

Exercises

1. What is the address space in each of the following systems?
   a. a system with 8-bit addresses
   b. a system with 16-bit addresses
   c. a system with 64-bit addresses
2. An address space has a total of 1,024 addresses. How many bits are needed to represent an address?
3. An address space uses three symbols: 0, 1, and 2 to represent addresses. If each address is made of 10 symbols, how many addresses are available in this system?
4. Change the following IP addresses from dotted-decimal notation to binary notation:
   a. 114.34.2.8
   b. 129.14.6.8
   c. 208.34.54.12
   d. 238.34.2.1
5. Change the following IP addresses from dotted-decimal notation to hexadecimal notation:
   a. 114.34.2.8
   b. 129.14.6.8
   c. 208.34.54.12
   d. 238.34.2.1
6. Change the following IP addresses from hexadecimal notation to binary notation:
   a. 0x1347FEAB
   b. 0xAB234102

c. 0x0123A2BE

d. 0x00001111

7. How many hexadecimal digits are needed to define the netid in each of the following classes?

a. Class A

b. Class B

c. Class C

8. Change the following IP addresses from binary notation to dotted-decimal notation:

a. **01111111 11110000 01100111 01111101**

b. **10101111 11000000 11111000 00011101**

c. **11011111 10110000 00011111 01011101**

d. **11101111 11110111 11000111 00011101**

9. Find the class of the following IP addresses:

a. 208.34.54.12

b. 238.34.2.1

c. 242.34.2.8

d. 129.14.6.8

10. Find the class of the following IP addresses:

a. **11110111 11110011 10000111 11011101**

b. **10101111 11000000 11110000 00011101**

c. **11011111 10110000 00011111 01011101**

d. **11101111 11110111 11000111 00011101**

11. Find the netid and the hostid of the following IP addresses:

a. 114.34.2.8

b. 132.56.8.6

c. 208.34.54.12

d. 251.34.98.5

12. Find the number of addresses in the range if the first address is 14.7.24.0 and the last address is 14.14.34.255.

13. If the first address in a range is 122.12.7.0 and there are 2048 addresses in the range, what is the last address?

14. Find the result of each operation:

a. NOT (22.14.70.34)

b. NOT (145.36.12.20)

c. NOT (200.7.2.0)

d. NOT (11.20.255.255)

15. Find the result of each operation:

a. (22.14.70.34) AND (255.255.0.0)

b. (12.11.60.12) AND (255.0.0.0)

c. (14.110.160.12) AND (255.200.140.0)

d. (28.14.40.100) AND (255.128.100.0)

16. Find the result of each operation:

a. (22.14.70.34) OR (255.255.0.0)

b. (12.11.60.12) OR (255.0.0.0)

c. (14.110.160.12) OR (255.200.140.0)

d. (28.14.40.100) OR (255.128.100.0)

17. In a class A subnet, we know the IP address of one of the hosts and the subnet mask as given below:

IP Address: 25.34.12.56	Subnet mask: 255.255.0.0

What is the first address (subnet address)? What is the last address?

18. In a class B subnet, we know the IP address of one of the hosts and the subnet mask as given below:

IP Address: 131.134.112.66	Subnet mask: 255.255.224.0

What is the first address (subnet address)? What is the last address?

19. In a class C subnet, we know the IP address of one of the hosts and the subnet mask as given below:

IP Address: 202.44.82.16	Subnet mask: 255.255.255.192

What is the first address (subnet address)? What is the last address?

20. Find the subnet mask in each case:

a. 1024 subnets in class A

b. 256 subnets in class B

c. 32 subnets in class C

d. 4 subnets in class C

21. In a block of addresses, we know the IP address of one host is 25.34.12.56/16. What is the first address (network address) and the last address (limited broadcast address) in this block?

22. In a block of addresses, we know the IP address of one host is 182.44.82.16/26. What is the first address (network address) and the last address (limited broadcast address) in this block?

23. In fixed-length subnetting, find the number of 1s that must be added to the mask if the number of desired subnets is _____.

a. 2

b. 62

c. 122

d. 250

24. An organization is granted the block 16.0.0.0/8. The administrator wants to create 500 fixed-length subnets.

a. Find the subnet mask.

b. Find the number of addresses in each subnet.

c. Find the first and the last address in the first subnet.

d. Find the first and the last address in the last subnet (subnet 500).

25. An organization is granted the block 130.56.0.0/16. The administrator wants to create 1024 subnets.

a. Find the subnet mask.

b. Find the number of addresses in each subnet.

c. Find the first and the last address in the first subnet.

d. Find the first and the last address in the last subnet (subnet 1024).

26. An organization is granted the block 211.17.180.0/24. The administrator wants to create 32 subnets.

a. Find the subnet mask.

b. Find the number of addresses in each subnet.

c. Find the first and the last address in the first subnet.

d. Find the first and the last address in the last subnet (subnet 32).

27. Write the following mask in slash notation (/*n*):

a. 255.255.255.0

b. 255.0.0.0

c. 255.255.224.0

d. 255.255.240.0

28. Find the range of addresses in the following blocks:

a. 123.56.77.32/29

b. 200.17.21.128/27

c. 17.34.16.0/23

d. 180.34.64.64/30

29. In classless addressing, we know the first and the last address in the block. Can we find the prefix length? If the answer is yes, show the process and give an example.

30. In classless addressing, we know the first address and the number of addresses in the block. Can we find the prefix length? If the answer is yes, show the process and give an example.

31. In classless addressing, can two blocks have the same prefix length? Explain.

32. In classless addressing, we know the first address and one of the addresses in the block (not necessarily the last address). Can we find the prefix length? Explain.

33. An ISP is granted a block of addresses starting with 150.80.0.0/16. The ISP wants to distribute these blocks to 2600 customers as follows:

a. The first group has 200 medium-size businesses; each needs approximately 128 addresses.

b. The second group has 400 small businesses; each needs approximately 16 addresses.

c. The third group has 2000 households; each needs 4 addresses.

Design the subblocks and give the slash notation for each subblock. Find out how many addresses are still available after these allocations.

34. An ISP is granted a block of addresses starting with 120.60.4.0/20. The ISP wants to distribute these blocks to 100 organizations with each organization receiving 8 addresses only. Design the subblocks and give the slash notation for each subblock. Find out how many addresses are still available after these allocations.

35. An ISP has a block of 1024 addresses. It needs to divide the addresses to 1024 customers. Does it need subnetting? Explain your answer.