PART

4

# Network Layer

## Objectives

The network layer is responsible for the source-to-destination delivery of a packet, possibly across multiple networks (links). Whereas the data link layer oversees the delivery of the packet between two systems on the same network (links), the network layer ensures that each packet gets from its point of origin to its final destination.

The network layer is responsible for the delivery of individual packets from the source to the destination host.

The network layer adds a header that includes the logical addresses of the sender and receiver to the packet corning from the upper layer. If a packet travels through the Internet, we need this addressing system to help distinguish the source and destination.

When independent networks or links are connected together to create an internetwork, routers or switches route packets to their final destination. One of the functions of the network layer is to provide a routing mechanism.

In Part 4 of the book, we first discuss logical addressing (referred to as IP addressing in the Internet). We then discuss the main as well as some auxiliary protocols that are responsible for controlling the delivery of a packet from its source to its destination.

Part 4 of the book is devoted to the network layer and the services provided by this layer.

## Chapters

This part consists of four chapters: Chapters 19 to 22.

### *Chapter 19*

Chapter 19 discusses logical or IP addressing. We first discuss the historical classful addressing. We then describe the new classless addressing designed to alleviate some problems inherent in classful addressing. The completely new addressing system, IPv6, which may become prevalent in the near future, is also discussed.

*Chapter 20*

Chapter 20 is devoted to the main protocol at the network layer that supervises and controls the delivery of packets from the source to destination. This protocol is called the Internet Protocol or IP.

*Chapter 21*

Chapter 21 is devoted to some auxiliary protocols defined at the network layer, that help the IP protocol do its job. These protocols perform address mapping (logical to physical or vice versa), error reporting, and facilitate multicast delivery.

*Chapter 22*

Delivery and routing of packets in the Internet is a very delicate and important issue. We devote Chapter 22 to this matter. We first discuss the mechanism of delivery and routing. We then briefly discuss some unicast and multicast routing protocols used in the Internet today.

# CHAPTER 19

# *Network Layer: Logical Addressing*

As we discussed in Chapter 2, communication at the network layer is host-to-host (computer-to-computer); a computer somewhere in the world needs to communicate with another computer somewhere else in the world. Usually, computers communicate through the Internet. The packet transmitted by the sending computer may pass through several LANs or WANs before reaching the destination computer.

For this level of communication, we need a global addressing scheme; we called this logical addressing in Chapter 2. Today, we use the term IP address to mean a logical address in the network layer of the TCP/IP protocol suite.

The Internet addresses are 32 bits in length; this gives us a maximum of $2^{32}$ addresses. These addresses are referred to as IPv4 (IP version 4) addresses or simply IP addresses if there is no confusion.

The need for more addresses, in addition to other concerns about the IP layer, motivated a new design of the IP layer called the new generation of IP or IPv6 (lP version 6). In this version, the Internet uses 128-bit addresses that give much greater flexibility in address allocation. These addresses are referred to as IPv6 (IP version 6) addresses.

In this chapter, we first discuss IPv4 addresses, which are currently being used in the Internet. We then discuss the IPv6 addresses, which may become dominant in the future.

## 19.1 IPv4 ADDRESSES

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

An 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. We will see later that, by using some strategies, an address may be assigned to a device for a time period and then taken away and assigned to another device.

On the other hand, if a device operating at the network layer has *m* connections to the Internet, it needs to have *m* addresses. We will see later that a router is such a device.

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 such as 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 $N$ bits to define an address, the address space is $2^N$ because each bit can have two different values (0 or 1) and $N$ bits can have $2^N$ values.

IPv4 uses 32-bit addresses, which means that the address space is $2^{32}$ or 4,294,967,296 (more than 4 billion). This means that, theoretically, if there were no restrictions, more than 4 billion devices could be connected to the Internet. We will see shortly that the actual number is much less because of the restrictions imposed on the addresses.

---

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

---

## Notations

There are two prevalent notations to show an IPv4 address: binary notation and dotted-decimal notation.

### *Binary Notation*

In binary notation, the IPv4 address is displayed as 32 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 or a 4-byte address. The following is an example of an IPv4 address in binary notation:

01110101 10010101 00011101 00000010

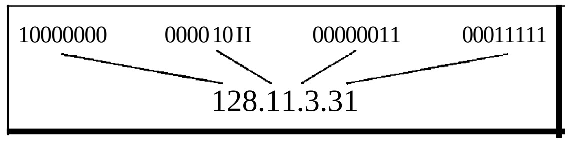
### *Dotted-Decimal Notation*

To make the IPv4 address more compact and easier to read, Internet addresses are usually written in decimal form with a decimal point (dot) separating the bytes. The following is the **dotted-decimal** notation of the above address:

117.149.29.2

Figure 19.1 shows an IPv4 address in both binary and dotted-decimal notation. Note that because each byte (octet) is 8 bits, each number in dotted-decimal notation is a value ranging from 0 to 255.

Figure 19.1 *Dotted-decimal notation and binary notation for an IPv4 address*




Numbering systems are reviewed **in** Appendix B.

### *Example 19.1*

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

a. 10000001 00001011 00001011 11101111

b. 11000001 10000011 00011011 11111111

#### 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

### *Example 19.2*

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

a. 111.56.45.78

b. 221.34.7.82

#### Solution

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

a.• 01101111 00111000 00101101 01001110

b. 11011101 00100010 00000111 01010010

### *Example 19.3*

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 must be no leading zero (045).

b. There can be no more than four numbers in an IPv4 address.

c. Each number needs to 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.

## Classful Addressing

IPv4 addressing, at its inception, used the concept of classes. This architecture is called classful addressing. Although this scheme is becoming obsolete, we briefly discuss it here to show the rationale behind classless addressing.

In classful addressing, the address space is divided into five classes: A, B, C, D, and E. Each class occupies some part of the address space.

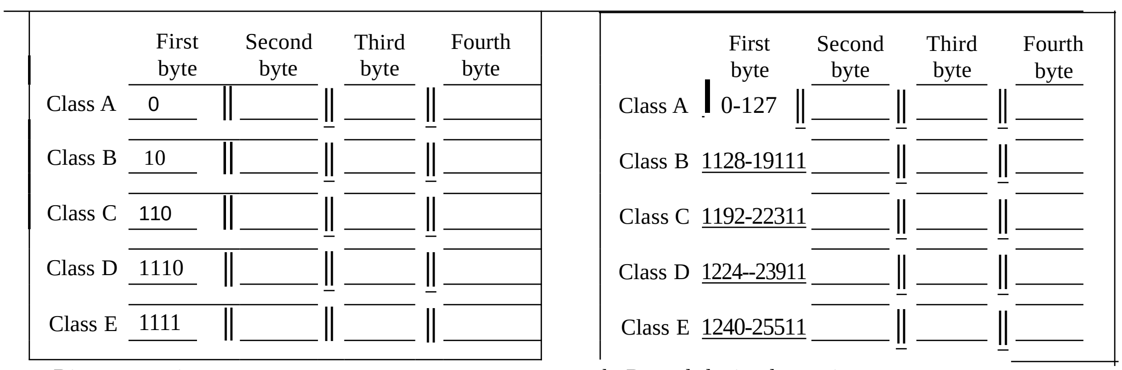
---

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

---

We can find the class of an address when given the address in binary notation or dotted-decimal notation. If the address is given in binary notation, the first few bits can immediately tell us the class of the address. If the address is given in decimal-dotted notation, the first byte defines the class. Both methods are shown in Figure 19.2.

Figure 19.2 *Finding the classes in binary and dotted-decimal notation*




a. Binary notation b. Dotted-decimal notation

### *Example 19.4*

Find the class of each address.

a. 00000001 00001011 00001011 11101111
b. 11000001 10000011 00011011 11111111
c. 14.23.120.8
d. 252.5.15.111

### Solution

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 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.

### 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 as shown in Table 19.1.

Table 19.1 *Number of blocks and block size in classful IPv4 addressing*

| *Class* | *Number of Blocks* | *Block Size* | *Application* |
|---|---|---|---|
| A | 128 | 16,777,216 | Unicast |
| B | 16,384 | 65,536 | Unicast |
| C | 2,097,152 | 256 | Unicast |
| D | 1 | 268,435,456 | Multicast |
| E | 1 | 268,435,456 | Reserved |

Let us examine the table. Previously, when an organization requested a block of addresses, it was granted one in class A, B, or C. Class A addresses were designed for large organizations with a large number of attached hosts or routers. Class B addresses were designed for midsize organizations with tens of thousands of attached hosts or routers. Class C addresses were designed for small organizations with a small number of attached hosts or routers.

We can see the flaw in this design. A block in class A address is too large for almost any organization. This means most of the addresses in class A were wasted and were not used. A block in class B is also very large, probably too large for many of the organizations that received a class B block. A block in class C is probably too small for many organizations. Class D addresses were designed for multicasting as we will see in a later chapter. Each address in this class is used to define one group of hosts on the Internet. The Internet authorities wrongly predicted a need for 268,435,456 groups. This never happened and many addresses were wasted here too. And lastly, the class E addresses were reserved for future use; only a few were used, resulting in another waste of addresses.

In c1assfnl addressing, a large part of the available addresses were wasted.

### Netid and Hostid

In classful addressing, an IP address in class A, B, or C is divided into netid and hostid. These parts are of varying lengths, depending on the class of the address. Figure 19.2 shows some netid and hostid bytes. The netid is in color, the hostid is in white. Note that the concept does not apply to classes D and E.

In class A, one byte defines the netid and three bytes define the hostid. In class B, two bytes define the netid and two bytes define the hostid. In class C, three bytes define the netid and one byte defines the hostid.

### Mask

Although the length of the netid and hostid (in bits) is predetermined in classful addressing, we can also use a mask (also called the default mask), a 32-bit number made of contiguous Is followed by contiguous as. The masks for classes A, B, and C are shown in Table 19.2. The concept does not apply to classes D and E.

Table 19.2 *Default masks for classful addressing*

| *Class* | *Binary* | *Dotted-Decimal* | *CIDR* |
|---|---|---|---|
| A | 11111111 00000000 00000000 00000000 | 255.0.0.0 | *18* |
| B | 11111111 11111111 00000000 00000000 | 255.255.0.0 | *116* |
| C | 11111111 11111111 11111111 00000000 | 255.255.255.0 | *124* |

The mask can help us to find the netid and the hostid. For example, the mask for a class A address has eight 1s, which means the first 8 bits of any address in class A define the netid; the next 24 bits define the hostid.

The last column of Table 19.2 shows the mask in the form *In* where *n* can be 8, 16, or 24 in classful addressing. This notation is also called slash notation or Classless Interdomain Routing (CIDR) notation. The notation is used in classless addressing, which we will discuss later. We introduce it here because it can also be applied to classful addressing. We will show later that classful addressing is a special case of classless addressing.

### *Subnetting*

During the era of classful addressing, subnetting was introduced. If an organization was granted a large block in class A or B, it could divide the addresses into several contiguous groups and assign each group to smaller networks (called subnets) or, in rare cases, share part of the addresses with neighbors. Subnetting increases the number of Is in the mask, as we will see later when we discuss classless addressing.

### *Supernetting*

The time came when most of the class A and class B addresses were depleted; however, there was still a huge demand for midsize blocks. The size of a class C block with a maximum number of 256 addresses did not satisfy the needs of most organizations. Even a midsize organization needed more addresses. 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 super-network or a supemet. An organization can apply for a set of class C blocks instead of just one. For example, an organization that needs 1000 addresses can be granted four contiguous class C blocks. The organization can then use these addresses to create one supernetwork. Supernetting decreases the number of Is in the mask. For example, if an organization is given four class C addresses, the mask changes from /24 to /22. We will see that classless addressing eliminated the need for supernetting.

### *Address Depletion*

The flaws in classful addressing scheme combined with the fast growth of the Internet led to the near depletion of the available addresses. Yet the number of devices on the Internet is much less than the $2^{32}$ address space. We have run out of class A and B addresses, and a class C block is too small for most midsize organizations. One solution that has alleviated the problem is the idea of classless addressing.

---

Classful addressing, which is almost obsolete, is replaced with classless addressing.

---

## Classless Addressing

To overcome address depletion and give more organizations access to the Internet, classless addressing was designed and implemented. In this scheme, there are no classes, but the addresses are still granted in blocks.

### *Address Blocks*

In classless addressing, when an entity, small or large, needs to be connected to the Internet, it is granted a block (range) of addresses. The size of the block (the number of addresses) varies based on the nature and size of the entity. For example, a household may be given only two addresses; a large organization may be given thousands of addresses. An ISP, as the Internet service provider, may be given thousands or hundreds of thousands based on the number of customers it may serve.

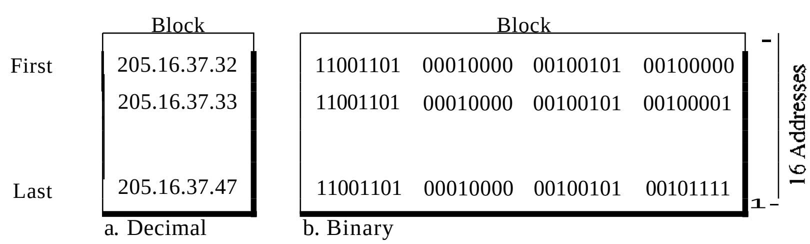
Restriction To simplify the handling of addresses, the Internet authorities impose three restrictions on classless address blocks:

1. The addresses in a block must be contiguous, one after another.
2. The number of addresses in a block must be a power of 2 (I, 2, 4, 8, ... ).
3. The first address must be evenly divisible by the number of addresses.

### *Example 19.5*

Figure 19.3 shows a block of addresses, in both binary and dotted-decimal notation, granted to a small business that needs 16 addresses.

Figure 19.3 *A block of* 16 *addresses granted to a small organization*




We can see that the restrictions are applied to this block. The addresses are contiguous. The number of addresses is a power of 2 ($16 = 2^4$), and the first address is divisible by 16. The first address, when converted to a decimal number, is 3,440,387,360, which when divided by 16 results in 215,024,210. In Appendix B, we show how to find the decimal value of an IP address.

### Mask

A better way to define a block of addresses is to select any address in the block and the mask. As we discussed before, a mask is a 32-bit number in which the $n$ leftmost bits are Is and the 32 - $n$ rightmost bits are Os. However, in classless addressing the mask for a block can take any value from 0 to 32. It is very convenient to give just the value of $n$ preceded by a slash (CIDR notation).

---

In 1Pv4 addressing, a block of addresses can be defined as
x.y.z.tln
in which x.y.z.t defines one of the addresses and the *In* defines the mask.

---

The address and the *In* notation completely define the whole block (the first address, the last address, and the number of addresses).

First Address The first address in the block can be found by setting the 32 - $n$ rightmost bits in the binary notation of the address to Os.

---

The first address in the block can be found by setting the rightmost 32 - $n$ bits to Os.

---

### *Example* 19.6

A block of addresses is granted to a small organization. We know that one of the addresses is 205.16.37.39/28. What is the first address in the block?

#### Solution

The binary representation of the given address is 11001101 00010000 00100101 00100I11. If we set 32 - 28 rightmost bits to 0, we get 11001101 000100000100101 0010000 or 205.16.37.32. This is actually the block shown in Figure 19.3.

Last Address The last address in the block can be found by setting the 32 - $n$ rightmost bits in the binary notation of the address to Is.

---

The last address in the block can be found by setting the rightmost 32 - $n$ bits to Is.

---

### *Example* 19.7

Find the last address for the block in Example 19.6.

#### Solution

The binary representation of the given address is 11001101 000100000010010100100111. If we set 32 - 28 rightmost bits to 1, we get 11001101 00010000 001001010010 1111 or 205.16.37.47. This is actually the block shown in Figure 19.3.

Number of Addresses The number of addresses in the block is the difference between the last and first address. It can easily be found using the formula $2^{32-n}$.

---

The number of addresses in the block can be found by using the formula $2^{32-n}$•

---

### *Example 19.8*

Find the number of addresses in Example 19.6.

#### Solution

The value of *n* is 28, which means that number of addresses is $2^{32-28}$ or 16.

### *Example 19.9*

Another way to find the first address, the last address, and the number of addresses is to represent the mask as a 32-bit binary (or 8-digit hexadecimal) number. This is particularly useful when we are writing a program to find these pieces of information. In Example 19.5 the /28 can be represented as 11111111 11111111 11111111 11110000 (twenty-eight Is and four Os). Find

a. The first address
h. The last address
c. The number of addresses

#### Solution

a. The first address can be found by ANDing the given addresses with the mask. ANDing here is done bit by bit. The result of ANDing 2 bits is 1 if both bits are Is; the result is 0 otherwise.

| | |
|---|---|
| Address: | 11001101 00010000 00100101 00100111 |
| Mask: | 11111111 11111111 11111111 11110000 |
| First address: | 11001101 00010000 00100101 00100000 |

b. The last address can be found by ORing the given addresses with the complement of the mask. ORing here is done bit by bit. The result of ORing 2 bits is 0 if both bits are Os; the result is 1 otherwise. The complement of a number is found by changing each 1 to 0 and each 0 to 1.

| | |
|---|---|
| Address: | 11001101 00010000 00100101 00100111 |
| Mask complement: | 00000000 00000000 00000000 00001111 |
| Last address: | 11001101 00010000 *00100101 00101111* |

c. The number of addresses can be found by complementing the mask, interpreting it as a decimal number, and adding 1 to it.

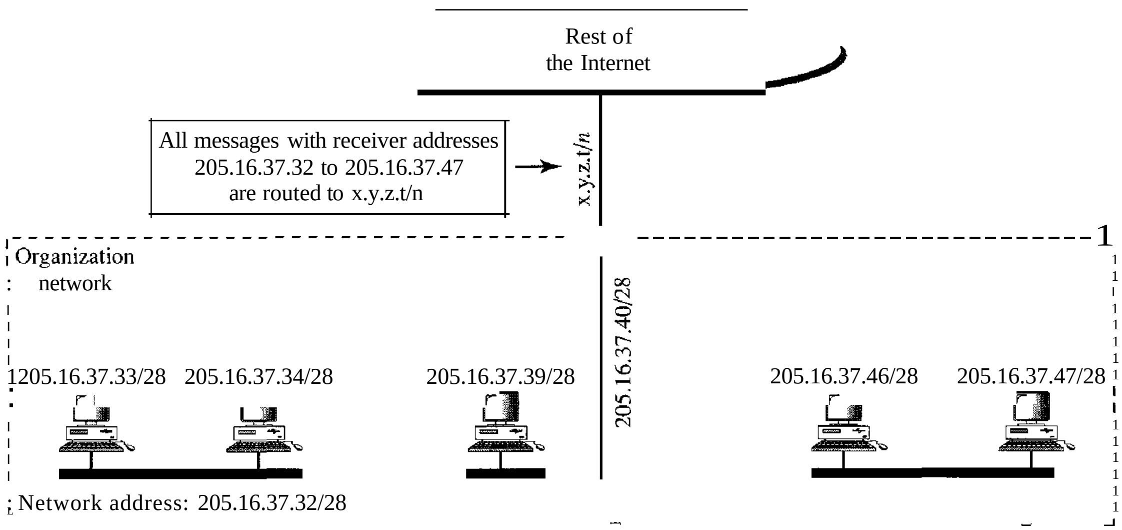
| | |
|---|---|
| Mask complement: | 000000000 00000000 00000000 00001111 |
| Number of addresses: | $15 + 1 = 16$ |

### *Network Addresses*

A very important concept in IP addressing is the network address. When an organization is given a block of addresses, the organization is free to allocate the addresses to the devices that need to be connected to the Internet. The first address in the class, however, is normally (not always) treated as a special address. The first address is called the network address and defines the organization network. It defines the organization itself to the rest of the world. In a later chapter we will see that the first address is the one that is used by routers to direct the message sent to the organization from the outside.

Figure 19.4 shows an organization that is granted a 16-address block.

Figure 19.4 *A network configuration for the block 205.16.37.32/28*



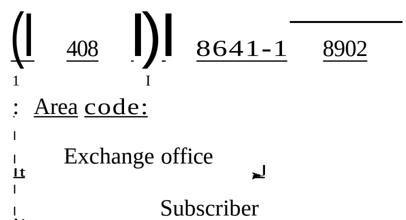

The organization network is connected to the Internet via a router. The router has two addresses. One belongs to the granted block; the other belongs to the network that is at the other side of the router. We call the second address *x.y.z.t/n* because we do not know anything about the network it is connected to at the other side. All messages destined for addresses in the organization block (205.16.37.32 to 205.16.37.47) are sent, directly or indirectly, to *x.y.z.t/n.* We say directly or indirectly because we do not know the structure of the network to which the other side of the router is connected.

The first address in a block is normally not assigned to any device; it is used as the network address that represents the organization to the rest of the world.

### *Hierarchy*

IP addresses, like other addresses or identifiers we encounter these days, have levels of hierarchy. For example, a telephone network in North America has three levels of hierarchy. The leftmost three digits define the area code, the next three digits define the exchange, the last four digits define the connection of the local loop to the central office. Figure 19.5 shows the structure of a hierarchical telephone number.

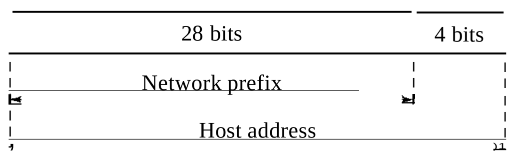
Figure 19.5 *Hierarchy in a telephone network in North America*



*Two-Level Hierarchy: No Subnetting*

An IP address can define only two levels of hierarchy when not subnetted. The *n* leftmost bits of the address *x.y.z.tJn* define the network (organization network); the 32 - *n* rightmost bits define the particular host (computer or router) to the network. The two common terms are prefix and suffix. The part of the address that defines the network is called the prefix; the part that defines the host is called the suffix. Figure 19.6 shows the hierarchical structure of an IPv4 address.

Figure 19.6 *Two levels ofhierarchy in an IPv4 address*




The prefix is common to all addresses in the network; the suffix changes from one device to another.

Each address in the block can be considered as a two-level hierarchical structure:
the leftmost *n* bits (prefix) define the network;
the rightmost 32 - *n* bits define the host.

*Three-Levels ofHierarchy: Subnetting*

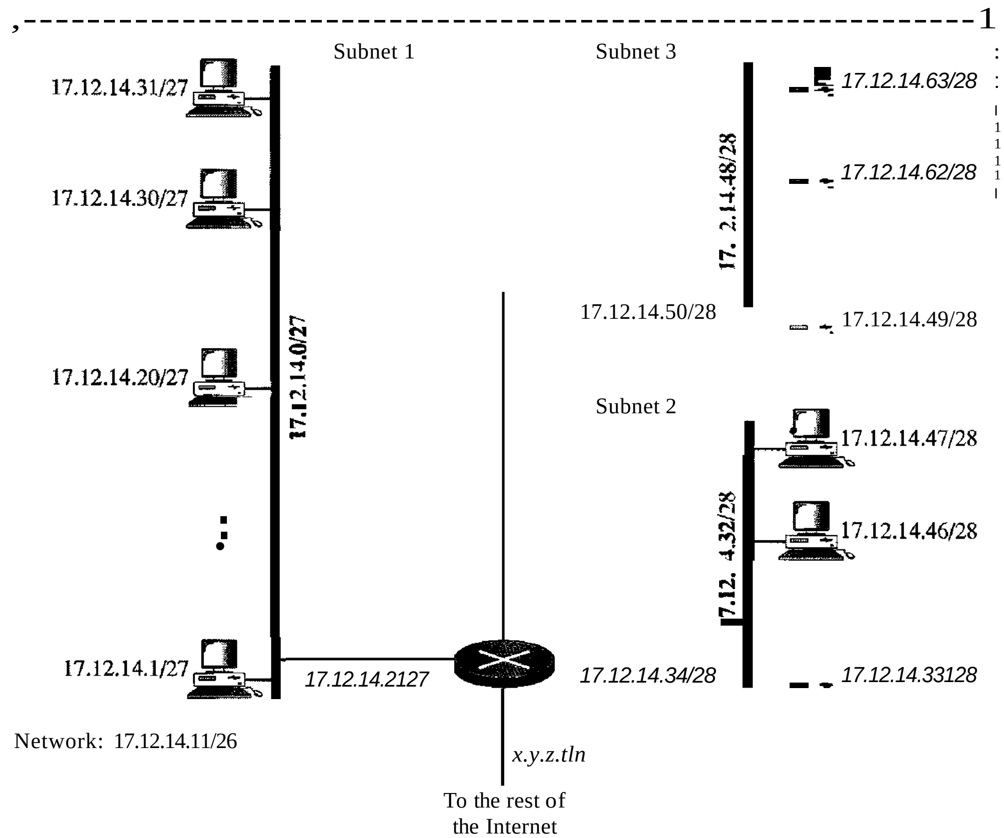
An organization that is granted a large block of addresses may want to create clusters of networks (called subnets) and divide the addresses between the different subnets. The rest of the world still sees the organization as one entity; however, internally there are several subnets. All messages are sent to the router address that connects the organization to the rest of the Internet; the router routes the message to the appropriate subnets. The organization, however, needs to create small subblocks of addresses, each assigned to specific subnets. The organization has its own mask; each subnet must also have its own.

As an example, suppose an organization is given the block 17.12.40.0/26, which contains 64 addresses. The organization has three offices and needs to divide the addresses into three subblocks of 32, 16, and 16 addresses. We can find the new masks by using the following arguments:

1. Suppose the mask for the first subnet is n1, then $2^{32-n1}$ must be 32, which means that n1 = 27.
2. Suppose the mask for the second subnet is n2, then $2^{32-n2}$ must be 16, which means that n2 = 28.
3. Suppose the mask for the third subnet is n3, then $2^{32-n3}$ must be 16, which means that n3 = 28.

This means that we have the masks 27, 28, 28 with the organization mask being 26. Figure 19.7 shows one configuration for the above scenario.

Figure 19.7 *Configuration and addresses in a subnetted network*




Let us check to see if we can find the subnet addresses from one of the addresses in the subnet.

a. In subnet 1, the address 17.12.14.29/27 can give us the subnet address if we use the mask /27 because

| | |
|---|---|
| Host: | 00010001 00001100 00001110 00011101 |
| Mask: | /27 |
| Subnet: | 00010001 00001100 00001110 00000000 .... (17.12.14.0) |

b. In subnet 2, the address 17.12.14.45/28 can give us the subnet address if we use the mask /28 because

| | |
|---|---|
| Host: | 00010001 00001100 00001110 00101101 |
| Mask: | /28 |
| Subnet: | 00010001 00001100 00001110 00100000 .... (17.12.14.32) |

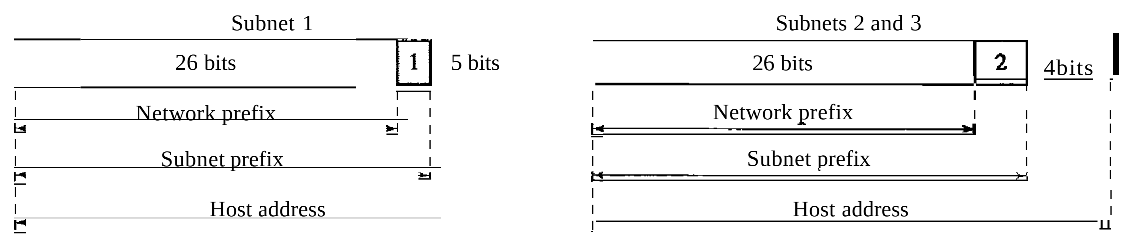
c. In subnet 3, the address 17.12.14.50/28 can give us the subnet address if we use the mask /28 because

| | |
|---|---|
| Host: | 00010001 00001100 00001110 00110010 |
| Mask: | /28 |
| Subnet: | 00010001 00001100 00001110 00110000 .... (17.12.14.48) |

Note that applying the mask of the network, */26,* to any of the addresses gives us the network address 17.12.14.0/26. We leave this proof to the reader.

We can say that through subnetting, we have three levels of hierarchy. Note that in our example, the subnet prefix length can differ for the subnets as shown in Figure 19.8.

Figure 19.8 *Three-level hierarchy in an IPv4 address*




### *More Levels of Hierarchy*

The structure of classless addressing does not restrict the number of hierarchical levels. An organization can divide the granted block of addresses into subblocks. Each subblock can in turn be divided into smaller subblocks. And so on. One example of this is seen in the ISPs. A national ISP can divide a granted large block into smaller blocks and assign each of them to a regional ISP. A regional ISP can divide the block received from the national ISP into smaller blocks and assign each one to a local ISP. A local ISP can divide the block received from the regional ISP into smaller blocks and assign each one to a different organization. Finally, an organization can divide the received block and make several subnets out of it.

### *Address Allocation*

The next issue in classless addressing is address allocation. How are the blocks allocated? The ultimate responsibility of address 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 organizations. It 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. In other words, an ISP receives one large block to be distributed to its Internet users. This is called address aggregation: many blocks of addresses are aggregated in one block and granted to one ISP.

### *Example 19.10*

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:

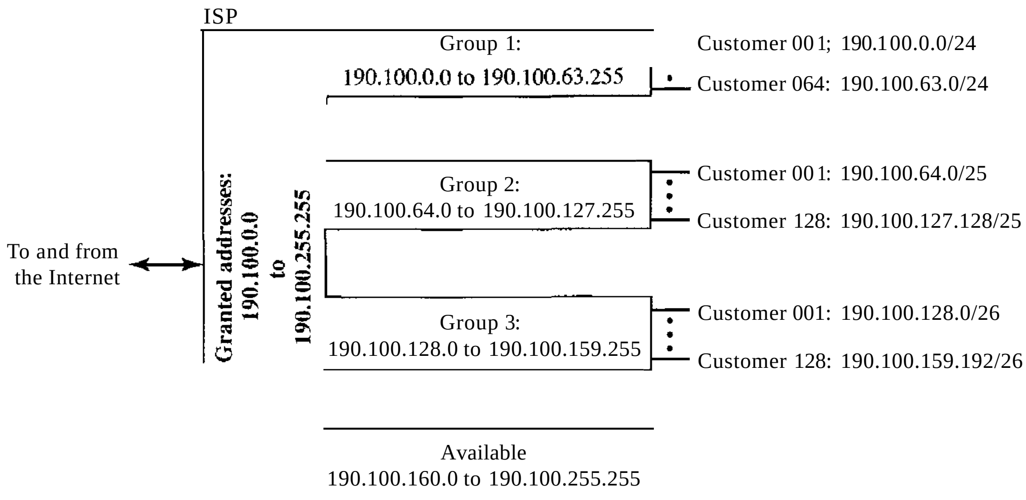
a. The first group has 64 customers; each needs 256 addresses.
b. The second group has 128 customers; each needs 128 addresses.
c. The third group has 128 customers; each needs 64 addresses.

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

### Solution

Figure 19.9 shows the situation.

**Figure 19.9** *An example ofaddress allocation and distribution by an IS?*




1. **Group 1**

   For this group, each customer needs 256 addresses. This means that 8 (log2256) bits are needed to define each host. The prefix length is then 32 - 8 =24. The addresses are

   | | | |
   |---|---|---|
   | *1st Customer:* | *190.100.0.0/24* | *190.100.0.255/24* |
   | *2nd Customer:* | *190.100.1.0/24* | *190.100.1.255/24* |
   | *64th Customer:* | *190.100.63.0/24* | *190.100.63.255/24* |

   *Total* =64 x 256 = 16,384

2. **Group2**

   For this group, each customer needs 128 addresses. This means that 7 (10g2 128) bits are needed to define each host. The prefix length is then 32 – 7 =25. The addresses are

   | | | |
   |---|---|---|
   | *1st Customer:* | *190.100.64.0/25* | *190.100.64.127/25* |
   | *2nd Customer:* | *190.100.64.128/25* | *190.100.64.255/25* |
   | *128th Customer:* | *190.100.127.128/25* | *190.100.127.255/25* |

   *Total* =128 x 128 = 16,384

3. **Group3**

   For this group, each customer needs 64 addresses. This means that 6 (logz 64) bits are needed to each host. The prefix length is then 32 - 6 =26. The addresses are

| | | |
|---|---|---|
| *1st Customer:* | *190.100.128.0/26* | *190.100.128.63/26* |
| *2nd Customer:* | *190.100.128.64/26* | *190.100.128.127/26* |
| | | |
| *128th Customer:* | *190.100.159.192/26* | *190.100.159.255/26* |

*Total* $= 128 \times 64 = 8192$

Number of granted addresses to the ISP: 65,536
Number of allocated addresses by the ISP: 40,960
Number of available addresses: 24,576

## Network Address **Translation** (NAT)

The number of home users and small businesses that want to use the Internet is ever increasing. In the beginning, a user was connected to the Internet with a dial-up line, which means that she was connected for a specific period of time. An ISP with a block of addresses could dynamically assign an address to this user. An address was given to a user when it was needed. But the situation is different today. Home users and small businesses can be connected by an ADSL line or cable modem. In addition, many are not happy with one address; many have created small networks with several hosts and need an IP address for each host. With the shortage of addresses, this is a serious problem.

A quick solution to this problem is called network address translation (NAT). NAT enables a user to have a large set of addresses internally and one address, or a small set of addresses, externally. The traffic inside can use the large set; the traffic outside, the small set.

To separate the addresses used inside the home or business and the ones used for the Internet, the Internet authorities have reserved three sets of addresses as private addresses, shown in Table 19.3.

Table 19.3 *Addresses for private networks*

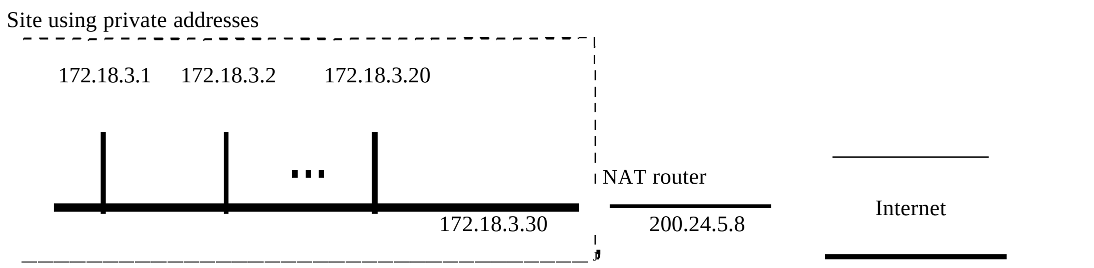
| *Range* | | | *Total* |
|---|---|---|---|
| 10.0.0.0 | to | 10.255.255.255 | $2^{24}$ |
| 172.16.0.0 | to | 172.31.255.255 | $2^{20}$ |
| 192.168.0.0 | to | 192.168.255.255 | $2^{16}$ |

Any organization can use an address out of this set without permission from the Internet authorities. Everyone knows that these reserved addresses are for private networks. They are unique inside the organization, but they are not unique globally. No router will forward a packet that has one of these addresses as the destination address.

The site must have only one single connection to the global Internet through a router that runs the NAT software. Figure 19.10 shows a simple implementation of NAT.

As Figure 19.10 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.

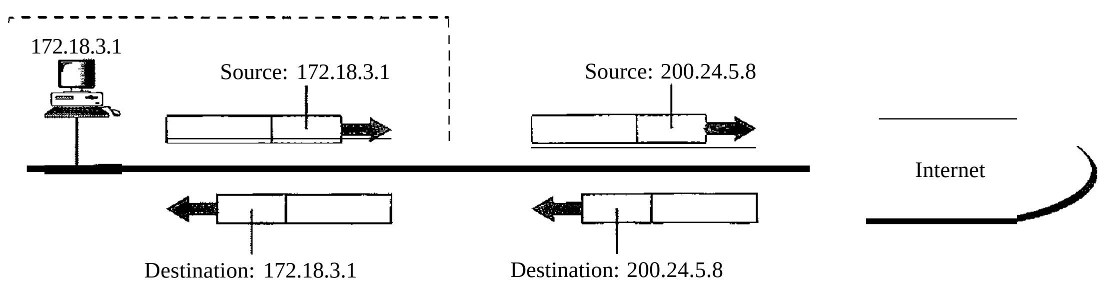
Figure 19.10 *A NAT implementation*




### *Address Translation*

All 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 19.11 shows an example of address translation.

Figure 19.11 *Addresses in a NAT*

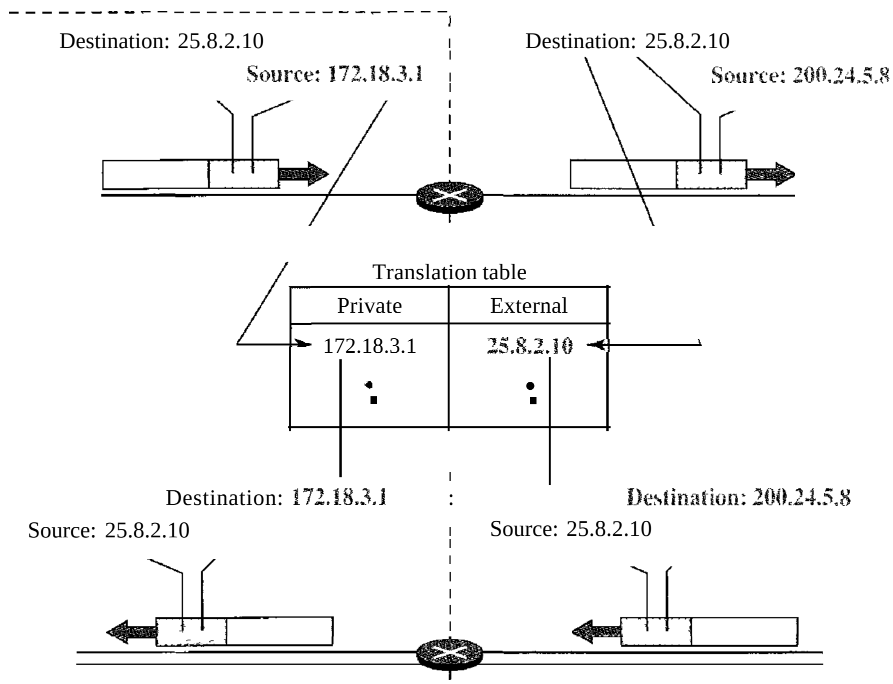



### *Translation Table*

The reader may have noticed that translating the source addresses for outgoing packets 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 fonn, 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 19.12 shows the idea. Note that the addresses that are changed (translated) are shown in color.

Figure 19.12 *NAT address 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 which 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, the e-mail is stored in the mailbox of the. customer until retrieved. 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 Since the NAT router has only one global address, only one private network host can access the same external host. To remove this restriction, the NAT router uses 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. In this example, no more than four connections can be made to the same destination. Also, no private-network host can access two external server programs (e.g., HTTP and FfP) at the same time.

Using Both **IP** Addresses and Port Numbers 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 with addresses 172.18.3.1 and 172.18.3.2 inside a private network 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 numbers of the transport layer protocol, the ambiguity is eliminated. We discuss port numbers in Chapter 23. Table 19.4 shows an example of such a table.

Table 19.4 *Five-column translation table*

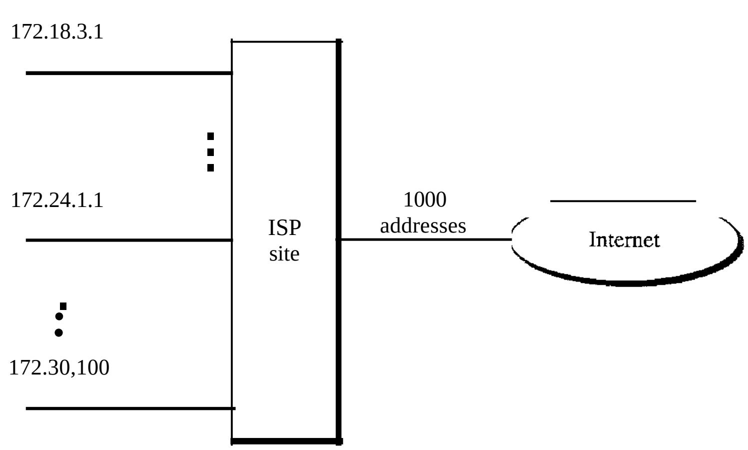
| *Private Address* | *Private Port* | *External Address* | *External Port* | *Transport Protocol* |
|---|---|---|---|---|
| 172.18.3.1 | 1400 | 25.8.3.2 | 80 | TCP |
| 172.18.3.2 | 1401 | 25.8.3.2 | 80 | TCP |
| ... | ... | ... | ... | ... |

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

### *NAT and ISP*

An ISP that serves dial-up customers can use NAT technology to conserve addresses. For example, suppose an ISP is granted 1000 addresses, but has 100,000 customers. Each of the customers is assigned a private network address. The ISP translates each of the 100,000 source addresses in outgoing packets to one of the 1000 global addresses; it translates the global destination address in incoming packets to the corresponding private address. Figure 19.13 shows this concept.

Figure 19.13 *An ISP and NAT*




## 19.2 IPv6 ADDRESSES

Despite all short-term solutions, such as classless addressing, Dynamic Host Configuration Protocol (DHCP), discussed in Chapter 21, and NAT, address depletion is still a long-term problem for the Internet. This and other problems in the IP protocol itself, such as lack of accommodation for real-time audio and video transmission, and encryption and authentication of data for some applications, have been the motivation for IPv6. In this section, we compare the address structure of IPv6 to IPv4. In Chapter 20, we discuss both protocols.

## Structure

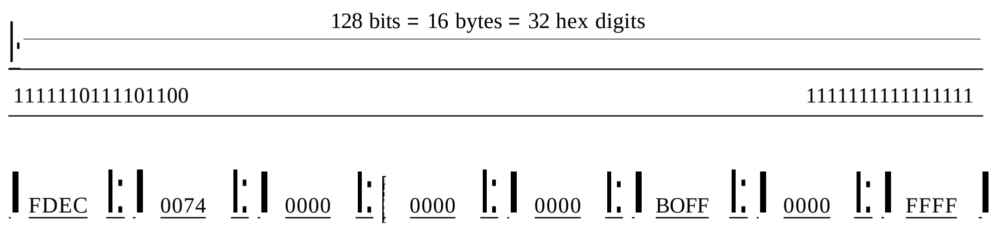
An IPv6 address consists of 16 bytes (octets); it is 128 bits long.

An IPv6 address is 128 bits long.

### *Hexadecimal Colon Notation*

To make addresses more readable, IPv6 specifies hexadecimal colon notation. In this notation, 128 bits is divided into eight sections, each 2 bytes in length. Two bytes in hexadecimal notation requires four hexadecimal digits. Therefore, the address consists of 32 hexadecimal digits, with every four digits separated by a colon, as shown in Figure 19.14.

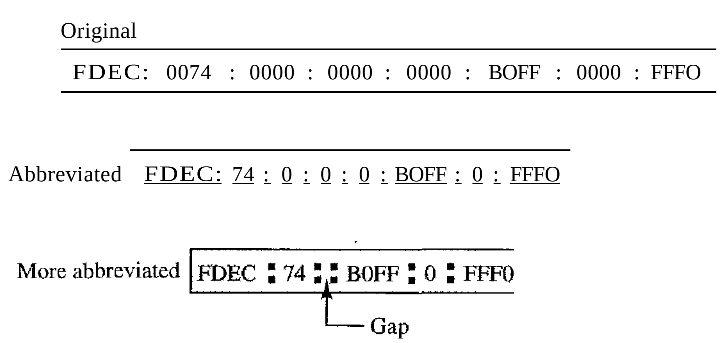
Figure 19.14 *IPv6 address in binary and hexadecimal colon notation*




### *Abbreviation*

Although the IP address, even in hexadecimal format, is very long, many of the digits are zer9s. In this case, we can abbreviate the address. The leading zeros of a section (four digits between two colons) can be omitted. Only the leading zeros can be dropped, not the trailing zeros (see Figure 19.15).

Figure 19.15 *Abbreviated IPv6 addresses*



Using this form of abbreviation, 0074 can be written as 74, OOOF as F, and 0000 as 0. Note that 3210 cannot be abbreviated. Further abbreviations are possible if there are consecutive sections consisting of zeros only. We can remove the zeros altogether and replace them with a double semicolon. Note that this type of abbreviation is allowed only once per address. If there are two runs of zero sections, only one of them can be abbreviated. Reexpansion of the abbreviated address is very simple: Align the unabbreviated portions and insert zeros to get the original expanded address.

*Example 19.11*

Expand the address 0:15::1:12:1213 to its original.

### Solution

We first need to align the left side of the double colon to the left of the original pattern and the right side of the double colon to the right of the original pattern to find now many 0s we need to replace the double colon.

```
xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx
   0:  15:                 l:  12:1213
```

This means that the original address is

0000:0015:0000:0000:0000:0001 :0012: 1213

## Address Space

IPv6 has a much larger address space; $2^{128}$ addresses are available. The designers of IPv6 divided the address into several categories. A few leftmost bits, called the *type prefix,* in each address define its category. The type prefix is variable in length, but it is designed such that no code is identical to the first part of any other code. In this way, there is no ambiguity; when an address is given, the type prefix can easily be determined. Table 19.5 shows the prefix for each type of address. The third column shows the fraction of each type of address relative to the whole address space.

Table 19.5 *Type prefixes for 1Pv6 addresses*

| *Type Prefix* | *Type* | *Fraction* |
|---|---|---|
| 00000000 | Reserved | 1/256 |
| 00000001 | Unassigned | 1/256 |
| 0000001 | ISO network addresses | 1/128 |
| 0000010 | IPX (Novell) network addresses | 1/128 |
| 0000011 | Unassigned | 1/128 |
| 00001 | Unassigned | 1/32 |
| 0001 | Reserved | 1/16 |
| 001 | Reserved | 1/8 |
| 010 | Provider-based unicast addresses | 1/8 |

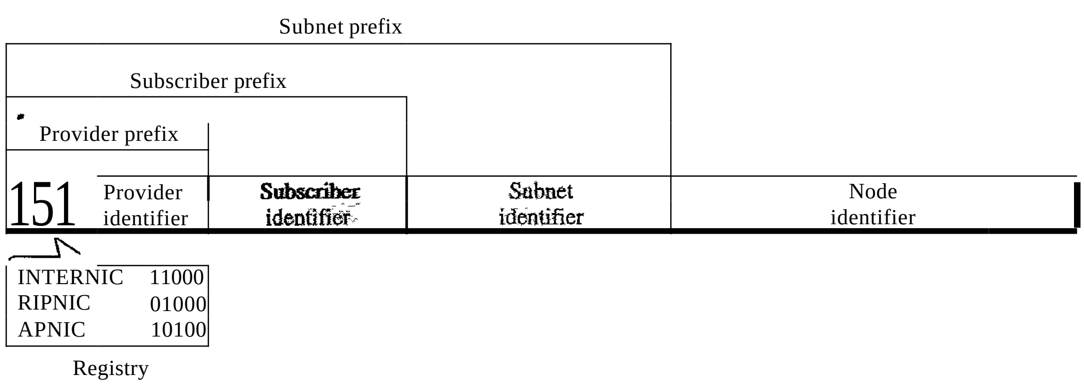
**Table 19.5** *Type prefixes for IPv6 addresses (continued)*

| *Type Prefix* | *Type* | *Fraction* |
|---|---|---|
| 011 | Unassigned | 1/8 |
| 100 | Geographic-based unicast addresses | 1/8 |
| 101 | Unassigned | 1/8 |
| 110 | Unassigned | 1/8 |
| 1110 | Unassigned | 1116 |
| 11110 | Unassigned | 1132 |
| 1111 10 | Unassigned | 1/64 |
| 1111 110 | Unassigned | 1/128 |
| 11111110 a | Unassigned | 1/512 |
| 1111 111010 | Link local addresses | 111024 |
| 1111 1110 11 | Site local addresses | 1/1024 |
| 11111111 | Multicast addresses | 1/256 |

*Unicast Addresses*

A **unicast address** defines a single computer. The packet sent to a unicast address must be delivered to that specific computer. IPv6 defines two types of unicast addresses: geographically based and provider-based. We discuss the second type here; the first type is left for future definition. The provider-based address is generally used by a normal host as a unicast address. The address format is shown in Figure 19.16.

**Figure 19.16** *Prefixes for provider-based unicast address*




Fields for the provider-based address are as follows:

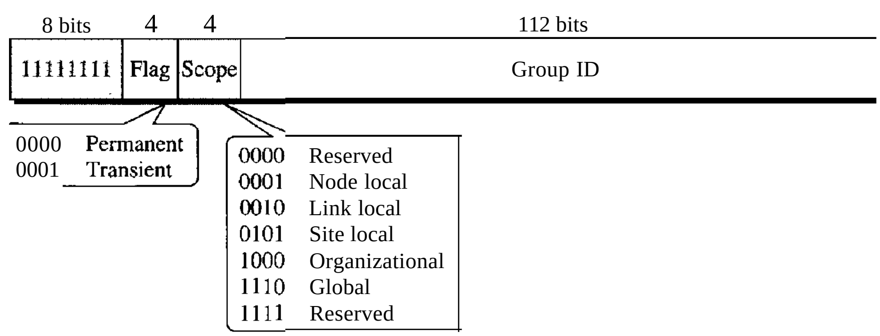
- **Type identifier.** This 3-bit field defines the address as a provider-base.d address.
- **Registry identifier.** This 5-bit field indicates the agency that has registered the address. Currently three registry centers have been defined. INTERNIC (code 11000) is the center for North America; RIPNIC (code 01000) is the center for European registration; and APNIC (code 10100) is for Asian and Pacific countries.

- **Provider identifier.** This variable-length field identifies the provider for Internet access (such as an ISP). A 16-bit length is recommended for this field.
- **Subscriber identifier.** When an organization subscribes to the Internet through a provider, it is assigned a subscriber identification. A 24-bit length is recommended for this field.
- **Subnet identifier.** Each subscriber can have many different subnetworks, and each subnetwork can have an identifier. The subnet identifier defines a specific subnetwork under the territory of the subscriber. A 32-bit length is recommended for this field.
- **Node identifier.** The last field defines the identity of the node connected to a subnet. A length of 48 bits is recommended for this field to make it compatible with the 48-bit link (physical) address used by Ethernet. In the future, this link address will probably be the same as the node physical address.

### *Multicast Addresses*

Multicast addresses are used to define a group of hosts instead of just one. A packet sent to a multicast address must be delivered to each member of the group. Figure 19.17 shows the format of a multicast address.

Figure 19.17 *Multicast address in IPv6*




The second field is a flag that defines the group address as either permanent or transient. A permanent group address is defined by the Internet authorities and can be accessed at all times. A transient group address, on the other hand, is used only temporarily. Systems engaged in a teleconference, for example, can use a transient group address. The third field defines the scope of the group address. Many different scopes have been defined, as shown in Figure 19.17.

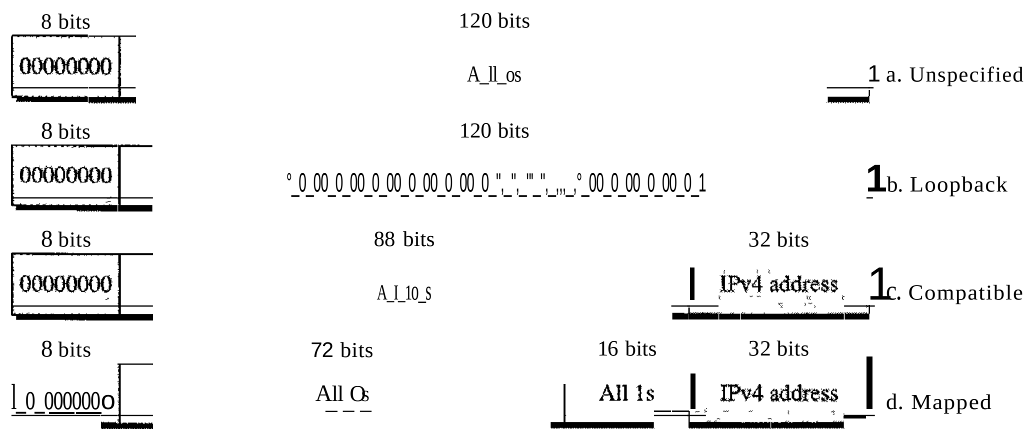
### *Allycast Addresses*

IPv6 also defines anycast addresses. An anycast address, like a multicast address, also defines a group of nodes. However, a packet destined for an anycast address is delivered to only one of the members of the anycast group, the nearest one (the one with the shortest route). Although the definition of an anycast address is still debatable, one possible use is to assign an anycast address to all routers of an ISP that covers a large logical area in the Internet. The routers outside the ISP deliver a packet destined for the ISP to the nearest ISP router. No block is assigned for anycast addresses.

*Reserved Addresses*

Another category in the address space is the reserved address. These addresses start with eight 0s (type prefix is 00000000). A few subcategories are defined in this category, as shown in Figure 19.18.

Figure 19.18 *Reserved addresses in IPv6*

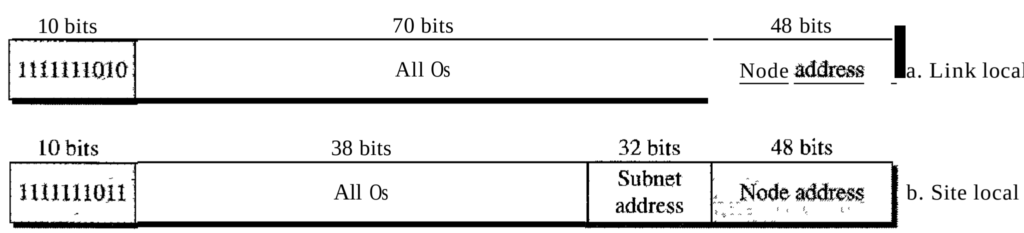



An unspecified address is used when a host does not know its own address and sends an inquiry to find its address. A loopback address is used by a host to test itself without going into the network. A compatible address is used during the transition from IPv4 to IPv6 (see Chapter 20). It is used when a computer using IPv6 wants to send a message to another computer using IPv6, but the message needs to pass through a part of the network that still operates in IPv4. A mapped address is also used during transition. However, it is used when a computer that has migrated to IPv6 wants to send a packet to a computer still using IPv4.

*Local Addresses*

These addresses are used when an organization wants to use IPv6 protocol without being connected to the global Internet. In other words, they provide addressing for private networks. Nobody outside the organization can send a message to the nodes using these addresses. Two types of addresses are defined for this purpose, as shown in Figure 19.19.

Figure 19.19 *Local addresses in IPv6*



A link local address is used in an isolated subnet; a site local address is used in an isolated site with several subnets.

## 19.3 RECOMMENDED READING

For more details about subjects discussed in this chapter, we recommend the following books and sites. The items in brackets [...] refer to the reference list at the end of the text.

### Books

IPv4 addresses are discussed in Chapters 4 and 5 of [For06], Chapter 3 of [Ste94], Section 4.1 of [PD03], Chapter 18 of [Sta04], and Section 5.6 of [Tan03]. IPv6 addresses are discussed in Section 27.1 of [For06] and Chapter 8 of [Los04]. A good discussion of NAT can be found in [DutOl].

### Sites

- www.ietf.org/rfc.html Information about RFCs

### RFCs

A discussion of IPv4 addresses can be found in most of the RFCs related to the IPv4 protocol:

760,781,791,815,1025,1063,1071,1141,1190, 1191, 1624,2113

A discussion of 1Pv6 addresses can be found in most of the RFCs related to IPv6 protocol:

1365,1550,1678,1680,1682,1683,1686,1688,1726, 1752, 1826, 1883, 1884,1886,1887, 1955,2080,2373,2452,2463,2465,2466,2472,2492,2545,2590

A discussion of NAT can be found in

1361,2663,2694

## 19.4 KEY TERMS

address aggregation
address space
anycast address
binary notation
class A address
class B address
class C address
class D address
class E address
classful addressing
classless addressing
classless interdomain routing (CIDR)

compatible address
dotted-decimal notation
default mask
hexadecimal colon notation
hostid
IP address
IPv4 address
IPv6 address
link local address
mapped address
mask
multicast address
netid
network address
network address translation (NAT)
prefix
reserved address
site local address
subnet
subnet mask
subnetting
suffix
supernet
supernet mask
supernetting
unicast address
unspecified address

## 19.5 SUMMARY

- At the network layer, a global identification system that uniquely identifies every host and router is necessary for delivery of a packet from host to host.
- An IPv4 address is 32 bits long and uniquely and universally defines a host or router on the Internet.
- In classful addressing, the portion of the IP address that identifies the network is called the netid.
- In classful addressing, the portion of the IP address that identifies the host or router on the network is called the hostid.
- An IP address defines a device's connection to a network.
- There are five classes in IPv4 addresses. Classes A, B, and C differ in the number of hosts allowed per network. Class D is for multicasting and Class E is reserved.
- The class of an address is easily determined by examination of the first byte.
- Addresses in classes A, B, or C are mostly used for unicast communication.
- Addresses in class D are used for multicast communication.
- Subnetting divides one large network into several smaller ones, adding an intermediate level of hierarchy in IP addressing.
- Supernetting combines several networks into one large one.
- In classless addressing, we can divide the address space into variable-length blocks.
- There are three restrictions in classless addressing:
  a. The number of addresses needs to be a power of 2.
  b. The mask needs to be included in the address to define the block.
  c. The starting address must be divisible by the number of addresses in the block.
- The mask in classless addressing is expressed as the prefix length *(In)* in CIDR notation.

- To find the first address in a block, we set the rightmost 32 - $n$ bits to 0.
- To find the number of addresses in the block, we calculate $2^{32-n}$, where $n$ is the prefix length.
- To find the last address in the block, we set the rightmost 32 - $n$ bits to 0.
- Subnetting increases the value of $n$.
- The global authority for address allocation is ICANN. ICANN normally grants large blocks of addresses to ISPs, which in tum grant small subblocks to individual customers.
- IPv6 addresses use hexadecimal colon notation with abbreviation methods available.
- There are three types of addresses in IPv6: unicast, anycast, and multicast.
- In an IPv6 address, the variable type prefix field defines the address type or purpose.

---

# 19.6 PRACTICE SET

## Review Questions

1. What is the number of bits in an IPv4 address? What is the number of bits in an IPv6 address?
2. What is dotted decimal notation in IPv4 addressing? What is the number of bytes in an IPv4 address represented in dotted decimal notation? What is hexadecimal notation in IPv6 addressing? What is the number of digits in an IPv6 address represented in hexadecimal notation?
3. What are the differences between classful addressing and classless addressing in IPv4?
4. List the classes in classful addressing and define the application of each class (unicast, multicast, broadcast, or reserve).
5. Explain why most of the addresses in class A are wasted. Explain why a medium-size or large-size corporation does not want a block of class C addresses.
6. What is a mask in IPv4 addressing? What is a default mask in IPv4 addressing?
7. What is the network address in a block of addresses? How can we find the network address if one of the addresses in a block is given?
8. Briefly define subnetting and supemetting. How do the subnet mask and supemet mask differ from a default mask in classful addressing?
9. How can we distinguish a multicast address in IPv4 addressing? How can we do so in IPv6 addressing?
10. What is NAT? How can NAT help in address depletion?

## Exercises

II. 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

12. An address space has a total of 1024 addresses. How many bits are needed to represent an address?
13. An address space uses the 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?
14. 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
15. 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
16. Find the class of the following IP addresses.
    a. 208.34.54.12
    b. 238.34.2.1
    c. 114.34.2.8
    d. 129.14.6.8
17. 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
18. 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
19. In a block of addresses, we know the IP address of one host is 25.34.12.56/16. What are the first address (network address) and the last address (limited broadcast address) in this block?
20. In a block of addresses, we know the IP address of one host is 182.44.82.16/26. What are the first address (network address) and the last address in this block?
21. 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 last addresses in subnet 1.
    d. Find the first and last addresses in subnet 500.

22. 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 last addresses in subnet 1.
    d. Find the first and last addresses in subnet 1024.
23. 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 last addresses in subnet 1.
    d. Find the first and last addresses in subnet 32.
24. Write the following masks in slash notation *(In)*.
    a. 255.255.255.0
    b. 255.0.0.0
    c. 255.255.224.0
    d. 255.255.240.0
25. 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
26. 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 128 addresses.
    b. The second group has 400 small businesses; each needs 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.
27. An ISP is granted a block of addresses starting with 120.60.4.0/22. The ISP wants to distribute these blocks to 100 organizations with each organization receiving just eight addresses. Design the subblocks and give the slash notation for each subblock. Find out how many addresses are still available after these allocations.
28. An ISP has a block of 1024 addresses. It needs to divide the addresses among 1024 customers. Does it need subnetting? Explain your answer.
29. Show the shortest form of the following addresses.
    a. 2340: lABC: 119A:AOOO:0000:0000:0000:0000
    b. OOOO:OOAA:OOOO:OOOO:OOOO:OOOO: 119A:A231
    c. 2340:0000:0000:0000:0000: 119A:AOO1:0000
    d. 0000:0000:0000:2340:0000:0000:0000:0000

30. Show the original (unabbreviated) form of the following addresses.
    a. 0::0
    b. O:AA::O
    c. 0: 1234::3
    d. 123::1:2
31. What is the type of each of the following addresses?
    a. FE80::12
    b. FECO: :24A2
    c. FF02::0
    d. 0::01
32. What is the type of each of the following addresses?
    a. 0::0
    b. 0: :FFFF:O:O
    c. 582F:1234::2222
    d. 4821::14:22
    e. 54EF::A234:2
33. Show the provider prefix (in hexadecimal colon notation) of an address assigned to a subscriber if it is registered in the United States with ABC1 as the provider identification.
34. Show in hexadecimal colon notation the IPv6 address
    a. Compatible to the IPv4 address 129.6.12.34
    b. Mapped to the IPv4 address 129.6.12.34
35. Show in hexadecimal colon notation
    a. The link local address in which the node identifier is 0:: 123/48
    b. The site local address in which the node identifier is 0:: 123/48
36. Show in hexadecimal colon notation the permanent multicast address used in a link local scope.
37. A host has the address 581E: 1456:2314:ABCD:: 1211. If the node identification is 48 bits, find the address of the subnet to which the host is attached.
38. A site with 200 subnets has the class B address of 132.45.0.0. The site recently migrated to IPv6 with the subscriber prefix 581E:1456:2314::ABCD/80. Design the subnets and define the subnet addresses, using a subnet identifier of 32 bits.

## Research Activities

39. Find the block of addresses assigned to your organization or institution.
40. If you are using an ISP to connect from your home to the Internet, find the name of the ISP and the block of addresses assigned to it.
41. Some people argue that we can consider the whole address space as one single block in which each range of addresses is a subblock to this single block. Elaborate on this idea. What happens to subnetting if we accept this concept?
42. Is your school or organization using a classful address? If so, find out the class of the address.

CHAPTER 20

# *Network Layer: Internet Protocol*

In the Internet model, the main network protocol is the **Internet Protocol** (IP). In this chapter, we first discuss internetworking and issues related to the network layer protocol in general.

We then discuss the current version of the Internet Protocol, version 4, or IPv4. This leads us to the next generation of this protocol, or IPv6, which may become the dominant protocol in the near future.

Finally, we discuss the transition strategies from IPv4 to IPv6. Some readers may note the absence of IPv5. IPv5 is an experimental protocol, based mostly on the OSI model that never materialized.

## 20.1 INTERNETWORKING

The physical and data link layers of a network operate locally. These two layers are jointly responsible for data delivery on the network from one node to the next, as shown in Figure 20.1.

This internetwork is made of five networks: four LANs and one WAN. If host A needs to send a data packet to host D, the packet needs to go first from A to R1 (a switch or router), then from R1 to R3, and finally from R3 to host D. We say that the data packet passes through three links. In each link, two physical and two data link layers are involved.

However, there is a big problem here. When data arrive at interface f1 of R1, how does R1 know that interface f3 is the outgoing interface? There is no provision in the data link (or physical) layer to help R1 make the right decision. The frame does not carry any routing information either. The frame contains the MAC address of A as the source and the MAC address of R1 as the destination. For a LAN or a WAN, delivery means carrying the frame through one link, and not beyond.

### Need for Network Layer

To solve the problem of delivery through several links, the network layer (or the internetwork layer, as it is sometimes called) was designed. The network layer is responsible for host-to-host delivery and for routing the packets through the routers or switches. Figure 20.2 shows the same internetwork with a network layer added.

Figure 20.1 *Links between two hosts*

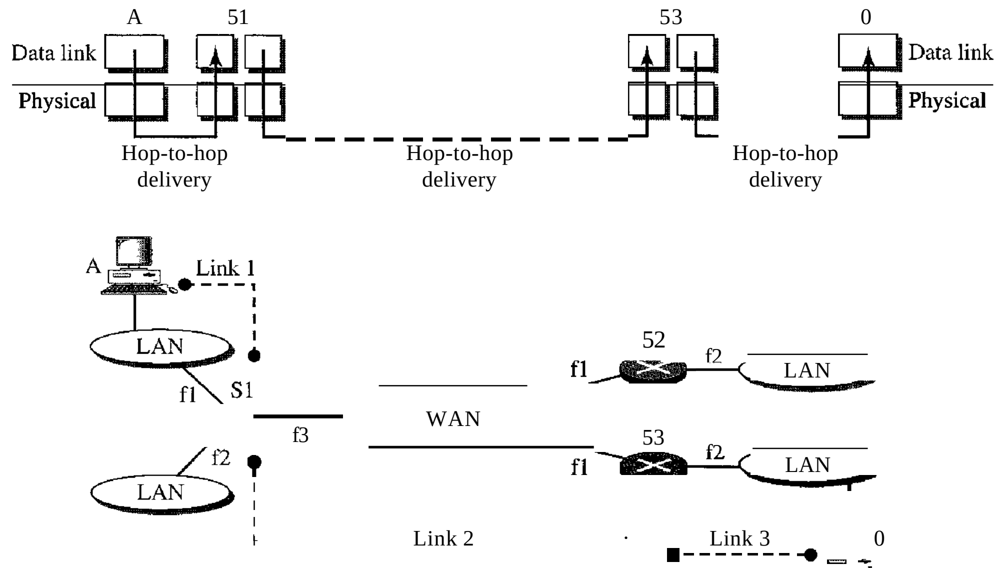



Figure 20.2 *Network layer in an internetwork*

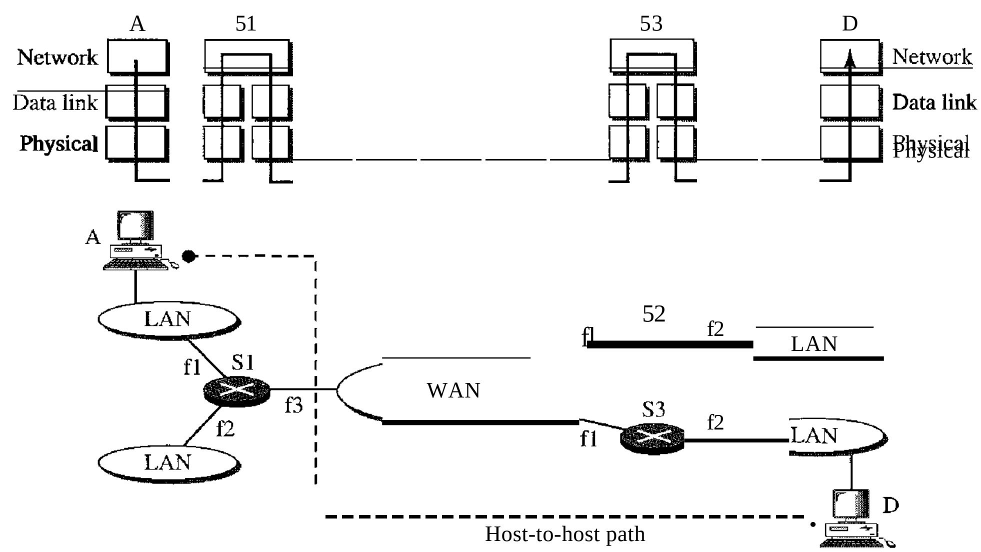
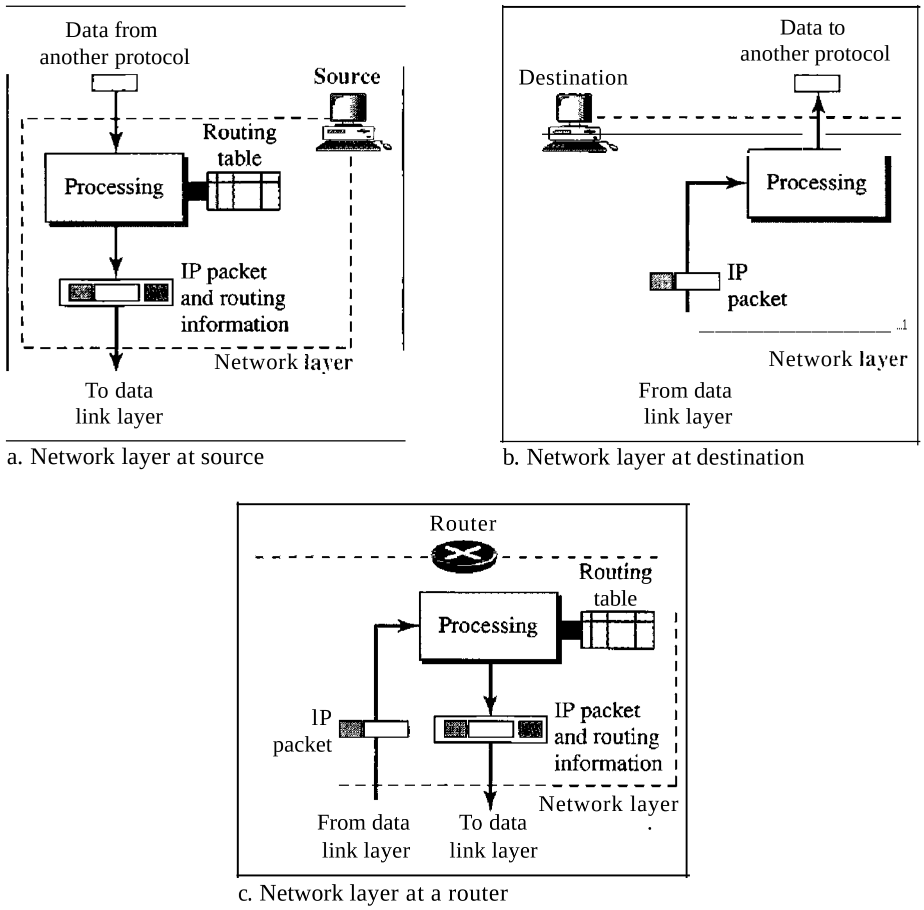



Figure 20.3 shows the general idea of the functionality of the network layer at a source, at a router, and at the destination. The network layer at the source is responsible for creating a packet from the data coming from another protocol (such as a transport layer protocol or a routing protocol). The header of the packet contains, among other information, the logical addresses of the source and destination. The network layer is responsible for checking its routing table to find the routing information (such as the outgoing interface of the packet or the physical address of the next node). If the packet is too large, the packet is fragmented (fragmentation is discussed later in this chapter).

Figure 20.3 *Network layer at the source, router, and destination*




The network layer at the switch or router is responsible for routing the packet. When a packet arrives, the router or switch consults its routing table and finds the interface from which the packet must be sent. The packet, after some changes in the header, with the routing infonnation is passed to the data link layer again.

The network layer at the destination is responsible for address verification; it makes sure that the destination address on the packet is the same as the address of the host. If the packet is a fragment, the network layer waits until all fragments have arrived, and then reassembles them and delivers the reassembled packet to the transport layer.

## **Internet** as a Datagram Network

The Internet, at the network layer, is a packet-switched network. We discussed switching in Chapter 8. We said that, in general, switching can be divided into three broad categories: circuit switching, packet switching, and message switching. Packet switching uses either the virtual circuit approach or the datagram approach.

The Internet has chosen the datagram approach to switching in the network layer. It uses the universal addresses defined in the network layer to route packets from the source to the destination.

Switching at the network layer in the Internet uses the datagram approach to packet switching.

### Internet as a Connectionless Network

Delivery of a packet can be accomplished by using either a connection-oriented or a connectionless network service. In a connection-oriented service, the source first makes a connection with the destination before sending a packet. When the connection is established, a sequence of packets from the same source to the same destination can be sent one after another. In this case, there is a relationship between packets. They are sent on the same path in sequential order. A packet is logically connected to the packet traveling before it and to the packet traveling after it. When all packets of a message have been delivered, the connection is terminated.

In a connection-oriented protocol, the decision about the route of a sequence of packets with the same source and destination addresses can be made only once, when the connection is established. Switches do not recalculate the route for each individual packet. This type of service is used in a virtual-circuit approach to packet switching such as in Frame Relay and ATM.

In conneetionless service, the network layer protocol treats each packet independently, with each packet having no relationship to any other packet. The packets in a message mayor may not travel the same path to their destination. This type of service is used in the datagram approach to packet switching. The Internet has chosen this type of service at the network layer.

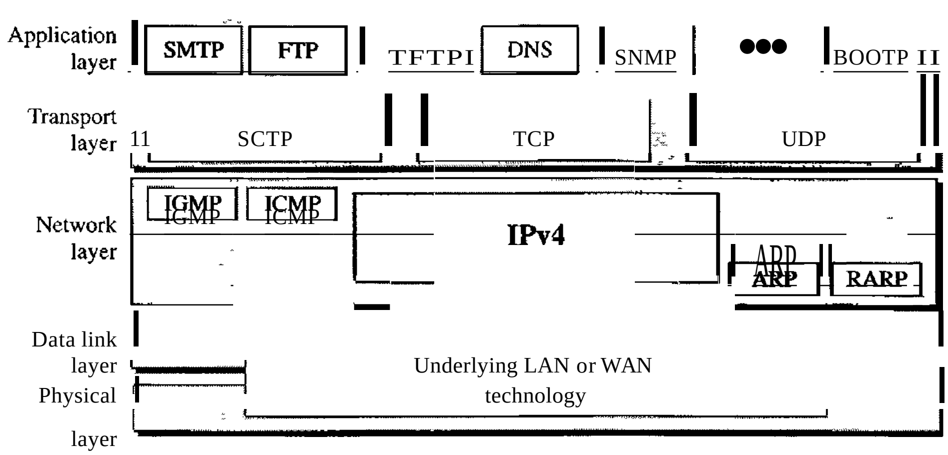
The reason for this decision is that the Internet is made of so many heterogeneous networks that it is almost impossible to create a connection from the source to the destination without knowing the nature of the networks in advance.

Communication at the network layer in the Internet is connectionless.

## 20.2 IPv4

The Internet Protocol version 4 (IPv4) is the delivery mechanism used by the TCP/IP protocols. Figure 20.4 shows the position of IPv4 in the suite.

Figure 20.4 *Position ofIPv4 in TCPIIP protocol suite*



IPv4 is an unreliable and connectionless datagram protocol-a best-effort delivery service. The term *best-effort* means that IPv4 provides no error control or flow control (except for error detection on the header). IPv4 assumes the unreliability of the underlying layers and does its best to get a transmission through to its destination, but with no guarantees.

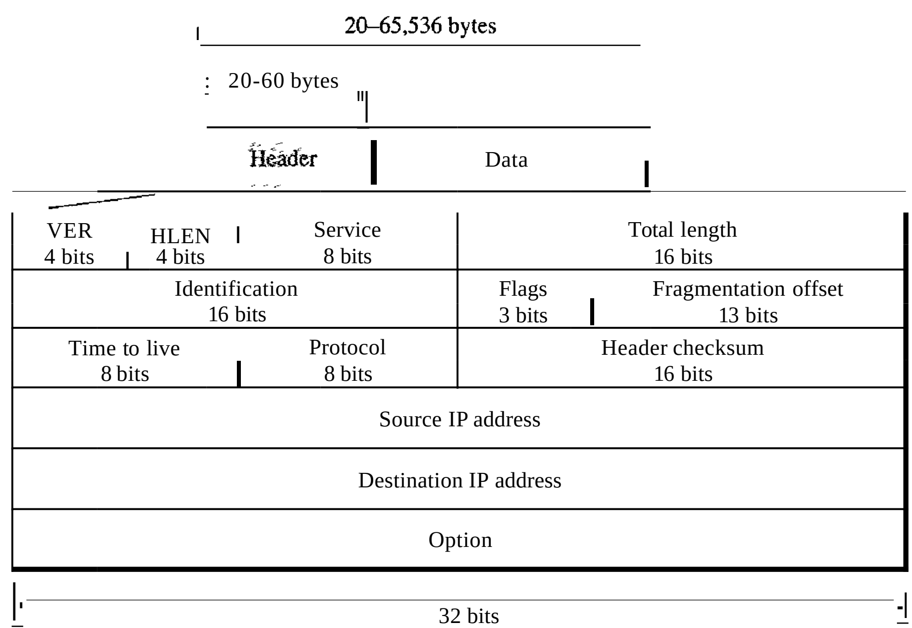
If reliability is important, IPv4 must be paired with a reliable protocol such as TCP. An example of a more commonly understood best-effort delivery service is the post office. The post office does its best to deliver the mail but does not always succeed. If an unregistered letter is lost, it is up to the sender or would-be recipient to discover the loss and rectify the problem. The post office itself does not keep track of every letter and cannot notify a sender of loss or damage.

IPv4 is also a connectionless protocol for a packet-switching network that uses the datagram approach (see Chapter 8). This means that each datagram is handled independently, and each datagram can follow a different route to the destination. This implies that datagrams sent by the same source to the same destination could arrive out of order. Also, some could be lost or corrupted during transmission. Again, IPv4 relies on a higher-level protocol to take care of all these problems.

## Datagram

Packets in the IPv4 layer are called datagrams. Figure 20.5 shows the IPv4 datagram format.

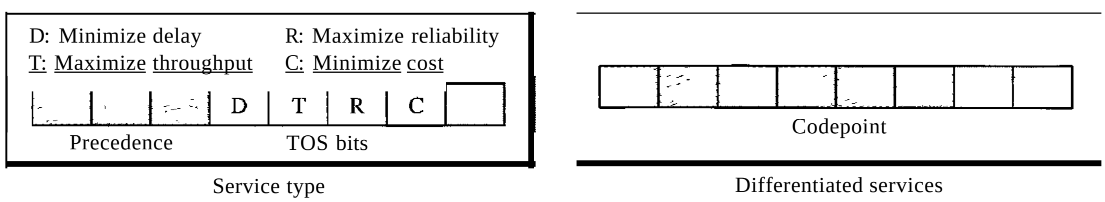
Figure 20.5 *IPv4 datagramfannat*




A datagram is a variable-length packet consisting of two parts: header and data. The header is 20 to 60 bytes in length and contains information essential to routing and delivery. It is customary in *TCP/IP* to show the header in 4-byte sections. A brief description of each field is in order.

- Version (VER). This 4-bit field defines the version of the IPv4 protocol. Currently the version is 4. However, version 6 (or IPng) may totally replace version 4 in the future. This field tells the IPv4 software running in the processing machine that the datagram has the format of version 4. All fields must be interpreted as specified in the fourth version of the protocol. If the machine is using some other version of IPv4, the datagram is discarded rather than interpreted incorrectly.
- Header length (HLEN). This 4-bit field defines the total length of the datagram header in 4-byte words. This field is needed because the length of the header is variable (between 20 and 60 bytes). When there are no options, the header length is 20 bytes, and the value of this field is 5 (5 x 4 = 20). When the option field is at its maximum size, the value of this field is 15 (15 x 4 = 60).
- Services. IETF has changed the interpretation and name of this 8-bit field. This field, previously called service type, is now called differentiated services. We show both interpretations in Figure 20.6.

Figure 20.6 *Service type or differentiated services*




1. Service Type
   In this interpretation, the first 3 bits are called precedence bits. The next 4 bits are called type of service (TOS) bits, and the last bit is not used.
   a. Precedence is a 3-bit subfield ranging from 0 (000 in binary) to 7 (111 in binary). The precedence defines the priority of the datagram in issues such as congestion. If a router is congested and needs to discard some datagrams, those datagrams with lowest precedence are discarded first. Some datagrams in the Internet are more important than others. For example, a datagram used for network management is much more urgent and important than a datagram containing optional information for a group.

The precedence subfield was part of version **4,** but never used.

   b. TOS bits is a 4-bit subfield with each bit having a special meaning. Although a bit can be either 0 or 1, one and only one of the bits can have the value of 1 in each datagram. The bit patterns and their interpretations are given in Table 20.1. With only 1 bit set at a time, we can have five different types of services.

Table 20.1 *Types of service*

| *TOS Bits* | *Description* |
|---|---|
| 0000 | Normal (default) |
| 0001 | Minimize cost |
| 0010 | Maximize reliability |
| 0100 | Maximize throughput |
| 1000 | Minimize delay |

Application programs can request a specific type of service. The defaults for some applications are shown in Table 20.2.

Table 20.2 *Default types of service*

| *Protocol* | *TOS Bits* | *Description* |
|---|---|---|
| ICMP | 0000 | Normal |
| BOOTP | 0000 | Normal |
| NNTP | 0001 | Minimize cost |
| IGP | 0010 | Maximize reliability |
| SNMP | 0010 | Maximize reliability |
| TELNET | 1000 | Minimize delay |
| FTP (data) | 0100 | Maximize throughput |
| FTP (control) | 1000 | Minimize delay |
| TFTP | 1000 | Minimize delay |
| SMTP (command) | 1000 | Minimize delay |
| SMTP (data) | 0100 | Maximize throughput |
| DNS (UDP query) | 1000 | Minimize delay |
| DNS (TCP query) | 0000 | Normal |
| DNS (zone) | 0100 | Maximize throughput |

It is clear from Table 20.2 that interactive activities, activities requiring immediate attention, and activities requiring immediate response need minimum delay. Those activities that send bulk data require maximum throughput. Management activities need maximum reliability. Background activities need minimum cost.

2. Differentiated Services

   In this interpretation, the first 6 bits make up the codepoint subfield, and the last 2 bits are not used. The codepoint subfield can be used in two different ways.

   a. When the 3 rightmost bits are 0s, the 3 leftmost bits are interpreted the same as the precedence bits in the service type interpretation. In other words, it is compatible with the old interpretation.

b. When the 3 rightmost bits are not all 0s, the 6 bits define 64 services based on the priority assignment by the Internet or local authorities according to Table 20.3. The first category contains 32 service types; the second and the third each contain 16. The first category (numbers 0, 2,4, ... ,62) is assigned by the Internet authorities (IETF). The second category (3, 7, 11, 15, , 63) can be used by local authorities (organizations). The third category (1, 5, 9, ,61) is temporary and can be used for experimental purposes. Note that the numbers are not contiguous. If they were, the first category would range from 0 to 31, the second from 32 to 47, and the third from 48 to 63. This would be incompatible with the TOS interpretation because XXXOOO (which includes 0, 8, 16, 24, 32, 40, 48, and 56) would fall into all three categories. Instead, in this assignment method all these services belong to category 1. Note that these assignments have not yet been finalized.

Table 20.3 *Values for codepoints*

| *Category* | *Codepoint* | *Assigning Authority* |
|---|---|---|
| 1 | XXXXXO | Internet |
| 2 | XXXXl1 | Local |
| 3 | XXXXOI | Temporary or experimental |

O Total length. This is a In-bit field that defines the total length (header plus data) of the IPv4 datagram in bytes. To find the length of the data coming from the upper layer, subtract the header length from the total length. The header length can be found by multiplying the value in the HLEN field by 4.

Length of data = total length - header length

Since the field length is 16 bits, the total length of the IPv4 datagram is limited to 65,535 ($2^{16}$ - 1) bytes, of which 20 to 60 bytes are the header and the rest is data from the upper layer.

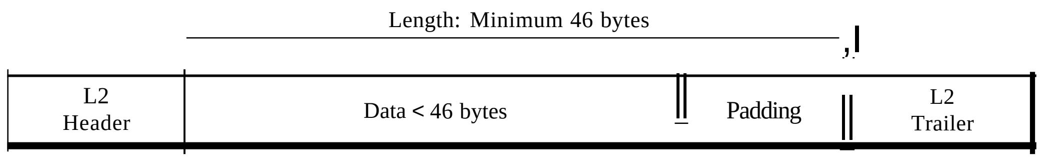
The total length field defines the total length of the datagram including the header.

Though a size of 65,535 bytes might seem large, the size of the IPv4 datagram may increase in the near future as the underlying technologies allow even more throughput (greater bandwidth).

When we discuss fragmentation in the next section, we will see that some physical networks are not able to encapsulate a datagram of 65,535 bytes in their frames. The datagram must be fragmented to be able to pass through those networks.

One may ask why we need this field anyway. When a machine (router or host) receives a frame, it drops the header and the trailer, leaving the datagram. Why include an extra field that is not needed? The answer is that in many cases we really do not need the value in this field. However, there are occasions in which the datagram is not the only thing encapsulated in a frame; it may be that padding has been added. For example, the Ethernet protocol has a minimum and maximum restriction on the size of data that can be encapsulated in a frame (46 to 1500 bytes). If the size of an IPv4 datagram is less than 46 bytes, some padding will be added to meet this requirement. In this case, when a machine decapsulates the datagram, it needs to check the total length field to determine how much is really data and how much is padding (see Figure 20.7).

Figure 20.7 *Encapsulation of a small datagram in an Ethernet frame*

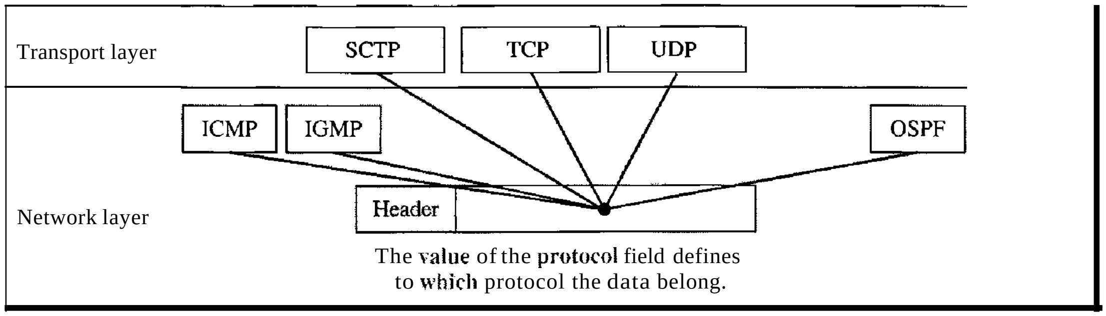



- Identification. This field is used in fragmentation (discussed in the next section).
- Flags. This field is used in fragmentation (discussed in the next section).
- Fragmentation offset. This field is used in fragmentation (discussed in the next section).
- Time to live. A datagram has a limited lifetime in its travel through an internet. This field was originally designed to hold a timestamp, which was decremented by each visited router. The datagram was discarded when the value became zero. However, for this scheme, all the machines must have synchronized clocks and must know how long it takes for a datagram to go from one machine to another. Today, this field is used mostly to control the maximum number of hops (routers) visited by the datagram. When a source host sends the datagram, it stores a number in this field. This value is approximately 2 times the maximum number of routes between any two hosts. Each router that processes the datagram decrements this number by 1. If this value, after being decremented, is zero, the router discards the datagram.

  This field is needed because routing tables in the Internet can become corrupted. A datagram may travel between two or more routers for a long time without ever getting delivered to the destination host. This field limits the lifetime of a datagram.

  Another use of this field is to intentionally limit the journey of the packet. For example, if the source wants to confine the packet to the local network, it can store 1 in this field. When the packet arrives at the first router, this value is decremented to 0, and the datagram is discarded.
- Protocol. This 8-bit field defines the higher-level protocol that uses the services of the IPv4 layer. An IPv4 datagram can encapsulate data from several higher-level protocols such as TCP, UDP, ICMP, and IGMP. This field specifies the final destination protocol to which the IPv4 datagram is delivered. In other words, since the IPv4 protocol carries data from different other protocols, the value of this field helps the receiving network layer know to which protocol the data belong (see Figure 20.8).

Figure 20.8 *Protocol field and encapsulated data*




The value of this field for each higher-level protocol is shown in Table 20.4.

Table 20.4 *Protocol values*

| *Value* | *Protocol* |
|---|---|
| 1 | ICMP |
| 2 | IGMP |
| 6 | TCP |
| 17 | UDP |
| 89 | OSPF |

- Checksum. The checksum concept and its calculation are discussed later in this chapter.
- Source address. This 32-bit field defines the IPv4 address of the source. This field must remain unchanged during the time the IPv4 datagram travels from the source host to the destination host.
- Destination address. This 32-bit field defines the IPv4 address of the destination. This field must remain unchanged during the time the IPv4 datagram travels from the source host to the destination host.

*Example 20.1*

An IPv4 packet has arrived with the first 8 bits as shown:

01000010

The receiver discards the packet. Why?

Solution

There is an elTOr in this packet. The 4 leftmost bits (0100) show the version, which is correct. The next 4 bits (0010) show an invalid header length ($2 \times 4 = 8$). The minimum number of bytes in the header must be 20. The packet has been corrupted in transmission.

*Example 20.2*

In an IPv4 packet, the value of HLEN is 1000 in binary. How many bytes of options are being carried by this packet?

**Solution**

The HLEN value is 8, which means the total number of bytes in the header is 8 x 4, or 32 bytes. The first 20 bytes are the base header, the next 12 bytes are the options.

*Example 20.3*

In an IPv4 packet, the value of HLEN is 5, and the value of the total length field is Ox0028. How many bytes of data are being carried by this packet?

**Solution**

The HLEN value is 5, which means the total number of bytes in the header is 5 x 4, or 20 bytes (no options). The total length is 40 bytes, which means the packet is carrying 20 bytes of data (40- 20).

*Example 20.4*

An IPv4 packet has arrived with the first few hexadecimal digits as shown.

Ox45000028000100000102 ...

How many hops can this packet travel before being dropped? The data belong to what upper-layer protocol?

**Solution**

To find the time-to-live field, we skip 8 bytes (16 hexadecimal digits). The time-to-live field is the ninth byte, which is 01. This means the packet can travel only one hop. The protocol field is the next byte (02), which means that the upper-layer protocol is IGMP (see Table 20.4).

## Fragmentation

A datagram can travel through different networks. Each router decapsulates the IPv4 datagram from the frame it receives, processes it, and then encapsulates it in another frame. The format and size of the received frame depend on the protocol used by the physical network through which the frame has just traveled. The format and size of the sent frame depend on the protocol used by the physical network through which the frame is going to travel. For example, if a router connects a LAN to a WAN, it receives a frame in the LAN format and sends a frame in the WAN format.

*Maximum Transfer Unit (MTU)*

Each data link layer protocol has its own frame format in most protocols. One of the fields defined in the format is the maximum size of the data field. In other words, when a datagram is encapsulated in a frame, the total size of the datagram must be less than this maximum size, which is defined by the restrictions imposed by the hardware and software used in the network (see Figure 20.9).

The value of the MTU depends on the physical network protocol. Table 20.5 shows the values for some protocols.

Figure 20.9 *Maximum transfer unit (MTU)*

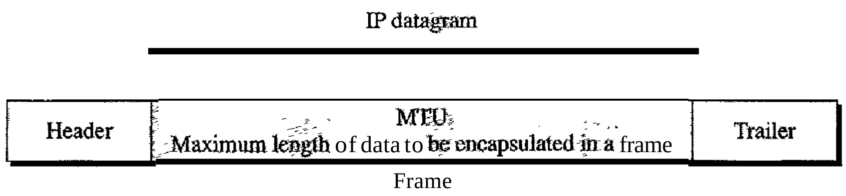



Table 20.5 *MTUs for some networks*

| *Protocol* | *MTU* |
|---|---|
| Hyperchannel | 65,535 |
| Token Ring (16 Mbps) | 17,914 |
| Token Ring (4 Mbps) | 4,464 |
| FDDI | 4,352 |
| Ethernet | 1,500 |
| X.25 | 576 |
| PPP | 296 |

To make the IPv4 protocol independent of the physical network, the designers decided to make the maximum length of the IPv4 datagram equal to 65,535 bytes. This makes transmission more efficient if we use a protocol with an MTU of this size. However, for other physical networks, we must divide the datagram to make it possible to pass through these networks. This is called fragmentation.

The source usually does not fragment the IPv4 packet. The transport layer will instead segment the data into a size that can be accommodated by IPv4 and the data link layer in use.

When a datagram is fragmented, each fragment has its own header with most of the fields repeated, but with some changed. A fragmented datagram may itself be fragmented if it encounters a network with an even smaller MTU. In other words, a datagram can be fragmented several times before it reaches the final destination.

In IPv4, a datagram can be fragmented by the source host or any router in the path although there is a tendency to limit fragmentation only at the source. The reassembly of the datagram, however, is done only by the destination host because each fragment becomes an independent datagram. Whereas the fragmented datagram can travel through different routes, and we can never control or guarantee which route a fragmented datagram may take, all the fragments belonging to the same datagram should finally arrive at the destination host. So it is logical to do the reassembly at the final destination. An even stronger objection to reassembling packets during the transmission is the loss of efficiency it incurs.

When a datagram is fragmented, required parts of the header must be copied by all fragments. The option field mayor may not be copied, as we will see in the next section. The host or router that fragments a datagram must change the values of three fields:

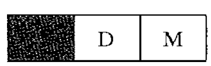
flags, fragmentation offset, and total length. The rest of the fields must be copied. Of course, the value of the checksum must be recalculated regardless of fragmentation.

### *Fields Related to Fragmentation*

The fields that are related to fragmentation and reassembly of an IPv4 datagram are the identification, flags, and fragmentation offset fields.

- **Identification.** This 16-bit field identifies a datagram originating from the source host. The combination of the identification and source IPv4 address must uniquely define a datagram as it leaves the source host. To guarantee uniqueness, the IPv4 protocol uses a counter to label the datagrams. The counter is initialized to a positive number. When the IPv4 protocol sends a datagram, it copies the current value of the counter to the identification field and increments the counter by 1. As long as the counter is kept in the main memory, uniqueness is guaranteed. When a datagram is fragmented, the value in the identification field is copied to all fragments. In other words, all fragments have the same identification number, the same as the original datagram. The identification number helps the destination in reassembling the datagram. It knows that all fragments having the same identification value must be assembled into one datagram.
- **Flags.** This is a 3-bit field. The first bit is reserved. The second bit is called the *do notfragment* bit. If its value is 1, the machine must not fragment the datagram. If it cannot pass the datagram through any available physical network, it discards the datagram and sends an ICMP error message to the source host (see Chapter 21). If its value is 0, the datagram can be fragmented if necessary. The third bit is called the *more fragment* bit. If its value is 1, it means the datagram is not the last fragment; there are more fragments after this one. If its value is 0, it means this is the last or only fragment (see Figure 20.10).

Figure 20.10 *Flags used in fragmentation*



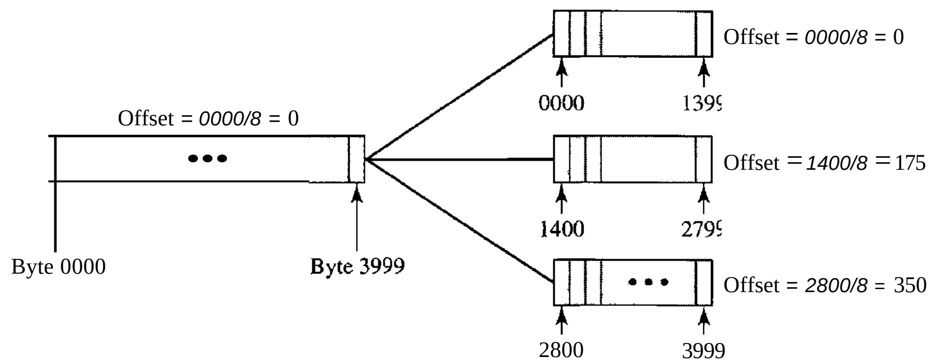

D: Donat fragment
M: More fragments

- **Fragmentation offset.** This 13-bit field shows the relative position of this fragment with respect to the whole datagram. It is the offset of the data in the original datagram measured in units of 8 bytes. Figure 20.11 shows a datagram with a data size of 4000 bytes fragmented into three fragments.

  The bytes in the original datagram are numbered 0 to 3999. The first fragment carries bytes 0 to 1399. The offset for this datagram is 0/8 = 0. The second fragment carries bytes 1400 to 2799; the offset value for this fragment is 1400/8 = 175. Finally, the third fragment carries bytes 2800 to 3999. The offset value for this fragment is 2800/8 = 350.

  Remember that the value of the offset is measured in units of 8 bytes. This is done because the length of the offset field is only 13 bits and cannot represent a

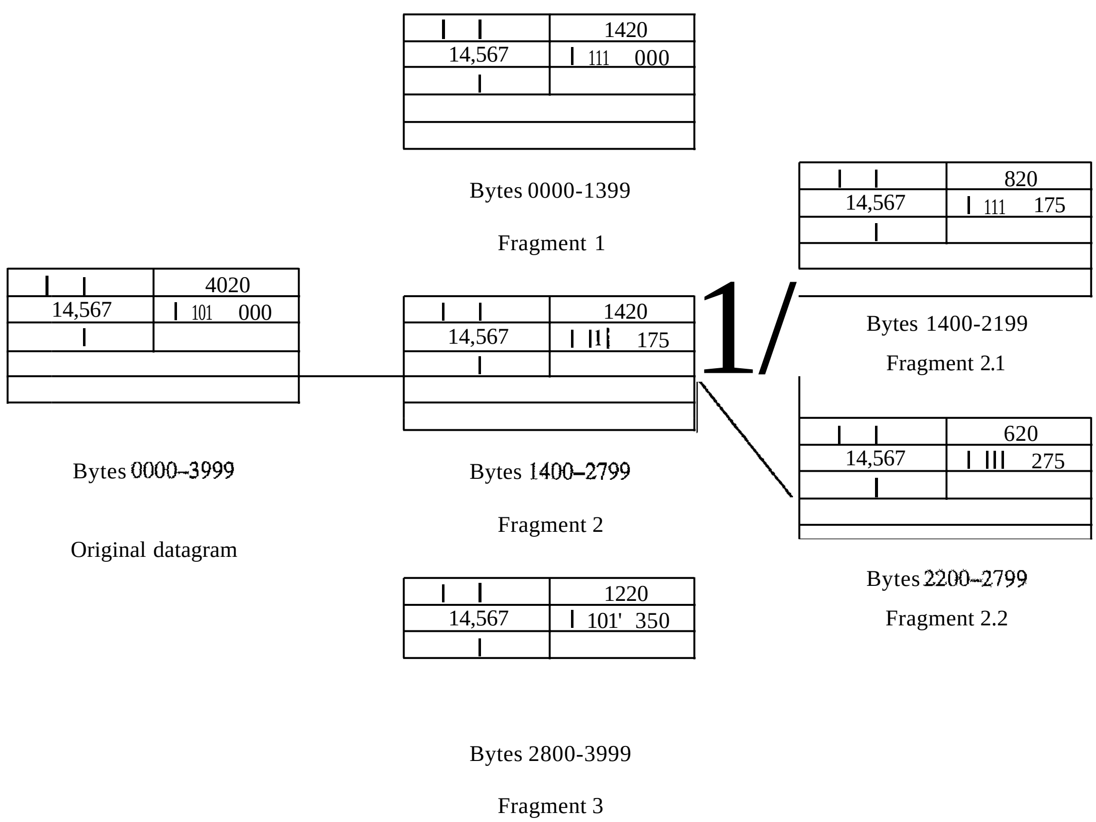
**Figure 20.11** *Fragmentation example*




sequence of bytes greater than 8191. This forces hosts or routers that fragment datagrams to choose a fragment size so that the first byte number is divisible by 8.

Figure 20.12 shows an expanded view of the fragments in Figure 20.11. Notice the value of the identification field is the same in all fragments. Notice the value of the flags field with the *more* bit set for all fragments except the last. Also, the value of the offset field for each fragment is shown.

**Figure 20.12** *Detailedfragmentation example*



The figure also shows what happens if a fragment itself is fragmented. In this case the value of the offset field is always relative to the original datagram. For example, in the figure, the second fragment is itself fragmented later to two fragments of 800 bytes and 600 bytes, but the offset shows the relative position of the fragments to the original data.

It is obvious that even if each fragment follows a different path and arrives out of order, the final destination host can reassemble the original datagram from the fragments received (if none of them is lost) by using the following strategy:

1. The first fragment has an offset field value of zero.
2. Divide the length of the first fragment by 8. The second fragment has an offset value equal to that result.
3. Divide the total length of the first and second fragments by 8. The third fragment has an offset value equal to that result.
4. Continue the process. The last fragment has a *more* bit value of 0.

*Example 20.5*

A packet has arrived with an *M* bit value of 0. Is this the first fragment, the last fragment, or a middle fragment? Do we know if the packet was fragmented?

**Solution**

If the *M* bit is 0, it means that there are no more fragments; the fragment is the last one. However, we cannot say if the original packet was fragmented or not. A nonfragmented packet is considered the last fragment.

*Example 20.6*

A packet has arrived with an *M* bit value of 1. Is this the first fragment, the last fragment, or a middle fragment? Do we know if the packet was fragmented?

**Solution**

If the *M* bit is 1, it means that there is at least one more fragment. This fragment can be the first one or a middle one, but not the last one. We don't know if it is the first one or a middle one; we need more information (the value of the fragmentation offset). See Example 20.7.

*Example 20.7*

A packet has arrived with an *M* bit value of 1 and a fragmentation offset value of 0. Is this the first fragment, the last fragment, Or a middle fragment?

**Solution**

Because the *M* bit is 1, it is either the first fragment or a middle one. Because the offset value is 0, it is the first fragment.

*Example 20.8*

A packet has arrived in which the offset value is 100. What is the number of the first byte? Do we know the number of the last byte?

**Solution**

To find the number of the first byte, we multiply the offset value by 8. This means that the first byte number is 800. We cannot determine the number of the last byte unless we know the length of the data.

*Example 20.9*

A packet has arrived in which the offset value is 100, the value of HLEN is 5, and the value of the tota1length field is 100. What are the numbers of the first byte and the last byte?

**Solution**

The first byte number is 100 x 8 = 800. The total length is 100 bytes, and the header length is 20 bytes (5 x 4), which means that there are 80 bytes in this datagram. If the first byte number is 800, the last byte number must be 879.

## Checksum

We discussed the general idea behind the checksum and how it is calculated in Chapter 10. The implementation of the checksum in the IPv4 packet follows the same principles. First, the value of the checksum field is set to 0. Then the entire header is divided into 16-bit sections and added together. The result (sum) is complemented and inserted into the checksum field.

The checksum in the IPv4 packet covers only the header, not the data. There are two good reasons for this. First, all higher-level protocols that encapsulate data in the IPv4 datagram have a checksum field that covers the whole packet. Therefore, the checksum for the IPv4 datagram does not have to check the encapsulated data. Second, the header of the IPv4 packet changes with each visited router, but the data do not. So the checksum includes only the part that has changed. If the data were included, each router must recalculate the checksum for the whole packet, which means an increase in processing time.

*Example 20.10*

Figure 20.13 shows an example of a checksum calculation for an IPv4 header without options. The header is divided into 16-bit sections. All the sections are added and the sum is complemented. The result is inserted in the checksum field.

## Options

The header of the IPv4 datagram is made of two parts: a fixed palt and a variable part. The fixed part is 20 bytes long and was discussed in the previous section. The variable part comprises the options that can be a maximum of 40 bytes.

Options, as the name implies, are not required for a datagram. They can be used for network testing and debugging. Although options are not a required part of the IPv4 header, option processing is required of the IPv4 software. This means that all implementations must be able to handle options if they are present in the header.

The detailed discussion of each option is beyond the scope of this book. We give the taxonomy of options in Figure 20.14 and briefly explain the purpose of each.

*No Operation*

A **no-operation option** is a I-byte option used as a filler between options.

Figure 20.13 *Example of checksum calculation in IPv4*

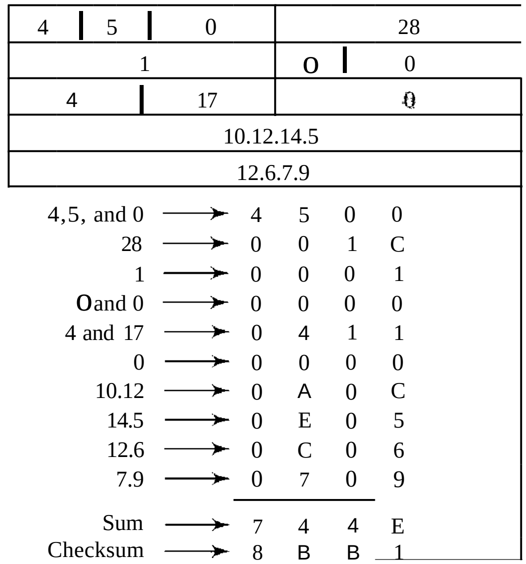
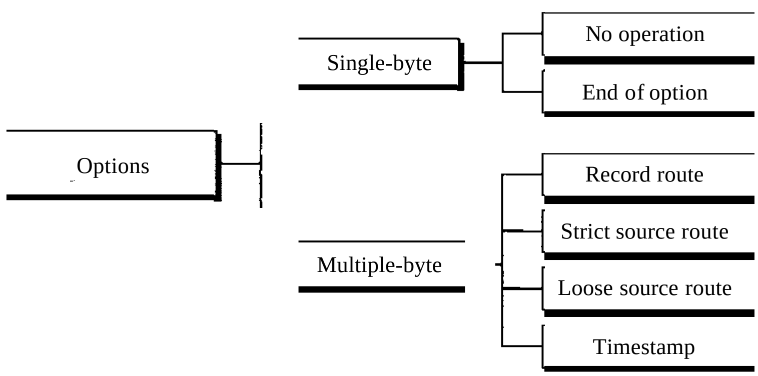



Figure 20.14 *Taxonomy of options in IPv4*




### *End of Option*

An end-of-option option is a I-byte option used for padding at the end of the option field. It, however, can only be used as the last option.

### *Record Route*

A record route option is used to record the Internet routers that handle the datagram. It can list up to nine router addresses. It can be used for debugging and management purposes.

### *Strict Source Route*

A strict source route option is used by the source to predetermine a route for the datagram as it travels through the Internet. Dictation of a route by the source can be useful for several purposes. The sender can choose a route with a specific type of service, such as minimum delay or maximum throughput. Alternatively, it may choose a route that is safer or more reliable for the sender's purpose. For example, a sender can choose a route so that its datagram does not travel through a competitor's network.

If a datagram specifies a strict source route, all the routers defined in the option must be visited by the datagram. A router must not be visited if its IPv4 address is not listed in the datagram. If the datagram visits a router that is not on the list, the datagram is discarded and an error message is issued. If the datagram arrives at the destination and some of the entries were not visited, it will also be discarded and an error message issued.

*Loose Source Route*

A loose source route option is similar to the strict source route, but it is less rigid. Each router in the list must be visited, but the datagram can visit other routers as well.

*Timestamp*

A timestamp option is used to record the time of datagram processing by a router. The time is expressed in milliseconds from midnight, Universal time or Greenwich mean time. Knowing the time a datagram is processed can help users and managers track the behavior of the routers in the Internet. We can estimate the time it takes for a datagram to go from one router to another. We say *estimate* because, although all routers may use Universal time, their local clocks may not be synchronized.

## 20.3 IPv6

The network layer protocol in the TCPIIP protocol suite is currently IPv4 (Internetworking Protocol, version 4). IPv4 provides the host-to-host communication between systems in the Internet. Although IPv4 is well designed, data communication has evolved since the inception of IPv4 in the 1970s. IPv4 has some deficiencies (listed below) that make it unsuitable for the fast-growing Internet.

- Despite all short-term solutions, such as subnetting, classless addressing, and NAT, address depletion is still a long-term problem in the Internet.
- The Internet must accommodate real-time audio and video transmission. This type of transmission requires minimum delay strategies and reservation of resources not provided in the IPv4 design.
- The Internet must accommodate encryption and authentication of data for some applications. No encryption or authentication is provided by IPv4.

To overcome these deficiencies, IPv6 (Internetworking Protocol, version 6), also known as IPng (Internetworking Protocol, next generation), was proposed and is now a standard. In IPv6, the Internet protocol was extensively modified to accommodate the unforeseen growth of the Internet. The format and the length of the IP address were changed along with the packet format. Related protocols, such as ICMP, were also modified. Other protocols in the network layer, such as ARP, RARP, and IGMP, were either deleted or included in the ICMPv6 protocol (see Chapter 21). Routing protocols, such as RIP and OSPF (see Chapter 22), were also slightly modified to accommodate these changes. Communications experts predict that IPv6 and its related protocols will soon replace the current IP version. In this section first we discuss IPv6. Then we explore the strategies used for the transition from version 4 to version 6.

The adoption of IPv6 has been slow. The reason is that the original motivation for its development, depletion of IPv4 addresses, has been remedied by short-term strategies such as classless addressing and NAT. However, the fast-spreading use of the Internet, and new services such as mobile IP, IP telephony, and IP-capable mobile telephony, may eventually require the total replacement of IPv4 with IPv6.

## Advantages

The next-generation IP, or IPv6, has some advantages over IPv4 that can be summarized as follows:

- Larger address space. An IPv6 address is 128 bits long, as we discussed in Chapter 19. Compared with the 32-bit address of IPv4, this is a huge ($2^{96}$) increase in the address space.
- Better header format. IPv6 uses a new header format in which options are separated from the base header and inserted, when needed, between the base header and the upper-layer data. This simplifies and speeds up the routing process because most of the options do not need to be checked by routers.
- New options. IPv6 has new options to allow for additional functionalities.
- Allowance for extension. IPv6 is designed to allow the extension of the protocol if required by new technologies or applications.
- Support for resource allocation. In IPv6, the type-of-service field has been removed, but a mechanism (calledjlow *label)* has been added to enable the source to request special handling of the packet. This mechanism can be used to support traffic such as real-time audio and video.
- Support for more security. The encryption and authentication options in IPv6 provide confidentiality and integrity of the packet.

## Packet Format

The IPv6 packet is shown in Figure 20.15. Each packet is composed of a mandatory base header followed by the payload. The payload consists of two parts: optional extension headers and data from an upper layer. The base header occupies 40 bytes, whereas the extension headers and data from the upper layer contain up to 65,535 bytes of information.

### *Base Header*

Figure 20.16 shows the base header with its eight fields.

These fields are as follows:

- Version. This 4-bit field defines the version number of the IP. For IPv6, the value is 6.
- Priority. The 4-bit priority field defines the priority of the packet with respect to traffic congestion. We will discuss this field later.

Figure 20.15 *IPv6 datagram header and payload*

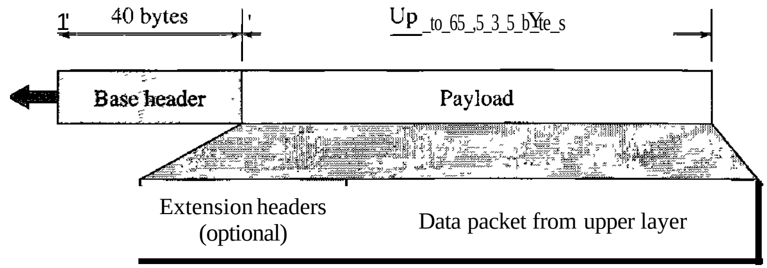
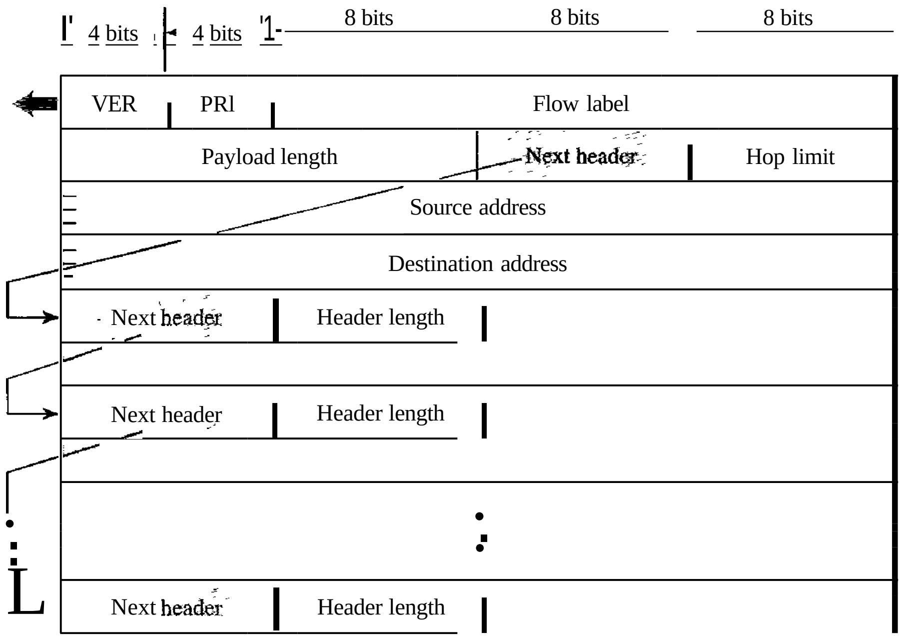



Figure 20.16 *Format of an IPv6 datagram*




- **Flow label.** The flow label is a 3-byte (24-bit) field that is designed to provide special handling for a particular flow of data. We will discuss this field later.
- **Payload length.** The 2-byte payload length field defines the length of the IP datagram excluding the base header.
- **Next header.** The next header is an 8-bit field defining the header that follows the base header in the datagram. The next header is either one of the optional extension headers used by IP or the header of an encapsulated packet such as UDP or TCP. Each extension header also contains this field. Table 20.6 shows the values of next headers. Note that this field in version 4 is called the *protocol.*
- **Hop limit.** This 8-bit hop limit field serves the same purpose as the TTL field in IPv4.
- **Source address.** The source address field is a 16-byte (128-bit) Internet address that identifies the original source of the datagram.

**Table 20.6** *Next header codes for IPv6*

| *Code* | *Next Header* |
|---|---|
| 0 | Hop-by-hop option |
| 2 | ICMP |
| 6 | TCP |
| 17 | UDP |
| 43 | Source routing |
| 44 | Fragmentation |
| 50 | Encrypted security payload |
| 51 | Authentication |
| 59 | Null (no next header) |
| 60 | Destination option |

- **Destination address.** The destination address field is a 16-byte (128-bit) Internet address that usually identifies the final destination of the datagram. However, if source routing is used, this field contains the address of the next router.

### *Priority*

The priority field of the IPv6 packet defines the priority of each packet with respect to other packets from the same source. For example, if one of two consecutive datagrams must be discarded due to congestion, the datagram with the lower **packet priority** will be discarded. IPv6 divides traffic into two broad categories: congestion-controlled and noncongestion-controlled.

**Congestion-Controlled Traffic** If a source adapts itself to traffic slowdown when there is congestion, the traffic is referred to as **congestion-controlled traffic.** For example, TCP, which uses the sliding window protocol, can easily respond to traffic. In congestion-controlled traffic, it is understood that packets may arrive delayed, lost, or out of order. Congestion-controlled data are assigned priorities from 0 to 7, as listed in Table 20.7. A priority of 0 is the lowest; a priority of 7 is the highest.

**Table 20.7** *Priorities for congestion-controlled traffic*

| *Priority* | *Meaning* |
|---|---|
| 0 | No specific traffic |
| 1 | Background data |
| 2 | Unattended data traffic |
| 3 | Reserved |
| 4 | Attended bulk data traffic |
| 5 | Reserved |
| 6 | Interactive traffic |
| 7 | Control traffic |

The priority descriptions are as follows:

- No specific traffic. A priority of 0 is assigned to a packet when the process does not define a priority.
- Background data. This group (priority 1) defines data that are usually delivered in the background. Delivery of the news is a good example.
- Unattended data traffic. If the user is not waiting (attending) for the data to be received, the packet will be given a priority of 2. E-mail belongs to this group. The recipient of an e-mail does not know when a message has arrived. In addition, an e-mail is usually stored before it is forwarded. A little bit of delay is of little consequence.
- Attended bulk data traffic. A protocol that transfers data while the user is waiting (attending) to receive the data (possibly with delay) is given a priority of 4. FTP and HTTP belong to this group.
- Interactive traffic. Protocols such as TELNET that need user interaction are assigned the second-highest priority (6) in this group.
- Control traffic. Control traffic is given the highest priority (7). Routing protocols such as OSPF and RIP and management protocols such as SNMP have this priority.

Noncongestion-Controlled Traffic This refers to a type of traffic that expects minimum delay. Discarding of packets is not desirable. Retransmission in most cases is impossible. In other words, the source does not adapt itself to congestion. Real-time audio and video are examples of this type of traffic.

Priority numbers from 8 to 15 are assigned to noncongestion-controlled traffic. Although there are not yet any particular standard assignments for this type of data, the priorities are usually based on how much the quality of received data is affected by the discarding of packets. Data containing less redundancy (such as low-fidelity audio or video) can be given a higher priority (15). Data containing more redundancy (such as high-fidelity audio or video) are given a lower priority (8). See Table 20.8.

Table 20.8 *Priorities for noncongestion-controlled traffic*

| *Priority* | *Meaning* |
|---|---|
| 8 | Data with greatest redundancy |
| ... | ... |
| 15 | Data with least redundancy |

### *Flow Label*

A sequence of packets, sent from a particular source to a particular destination, that needs special handling by routers is called a *flow* of packets. The combination of the source address and the value of the *flow label* uniquely defines a flow of packets.

To a router, a flow is a sequence of packets that share the same characteristics, such as traveling the same path, using the same resources, having the same kind of security, and so on. A router that supports the handling of flow labels has a flow label table. The table has an entry for each active flow label; each entry defines the services required by the corresponding flow label. When the router receives a packet, it consults its flow label table to find the corresponding entry for the flow label value defined in the packet. It then provides the packet with the services mentioned in the entry. However, note that the flow label itself does not provide the information for the entries of the flow label table; the information is provided by other means such as the hop-by-hop options or other protocols.

In its simplest form, a flow label can be used to speed up the processing of a packet by a router. When a router receives a packet, instead of consulting the routing table and going through a routing algorithm to define the address of the next hop, it can easily look in a flow label table for the next hop.

In its more sophisticated form, a flow label can be used to support the transmission of real-time audio and video. Real-time audio or video, particularly in digital form, requires resources such as high bandwidth, large buffers, long processing time, and so on. A process can make a reservation for these resources beforehand to guarantee that real-time data will not be delayed due to a lack of resources. The use of real-time data and the reservation of these resources require other protocols such as Real-Time Protocol (RTP) and Resource Reservation Protocol (RSVP) in addition to IPv6.

To allow the effective use of flow labels, three rules have been defined:

1. The flow label is assigned to a packet by the source host. The label is a random number between 1 and $2^{24} - 1$. A source must not reuse a flow label for a new flow while the existing flow is still active.
2. If a host does not support the flow label, it sets this field to zero. If a router does not support the flow label, it simply ignores it.
3. All packets belonging to the same flow have the same source, same destination, same priority, and same options.

### *Comparison Between IPv4 and IPv6 Headers*

Table 20.9 compares IPv4 and IPv6 headers.

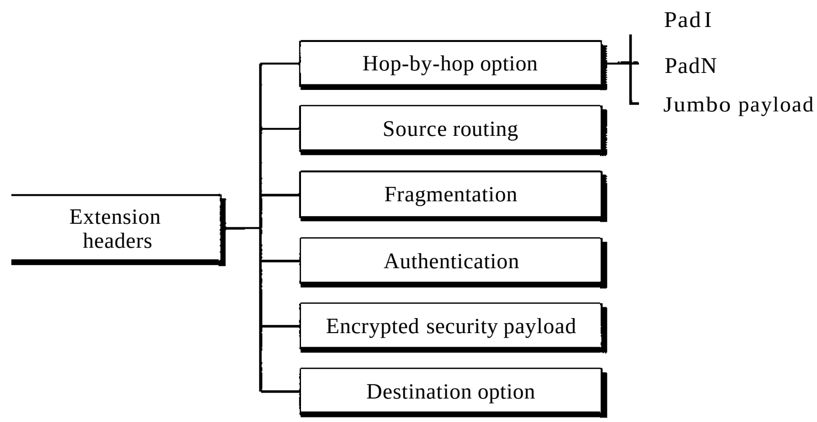
Table 20.9 *Comparison between IPv4 and IPv6 packet headers*

| *Comparison* |
|---|
| 1. The header length field is eliminated in IPv6 because the length of the header is fixed in this version. |
| 2. The service type field is eliminated in IPv6. The priority and flow label fields together take over the function of the service type field. |
| 3. The total length field is eliminated in IPv6 and replaced by the payload length field. |
| 4. The identification, flag, and offset fields are eliminated from the base header in IPv6. They are included in the fragmentation extension header. |
| 5. The TTL field is called hop limit in IPv6. |
| 6. The protocol field is replaced by the next header field. |
| 7. The header checksum is eliminated because the checksum is provided by upper-layer protocols; it is therefore not needed at this level. |
| 8. The option fields in IPv4 are implemented as extension headers in IPv6. |

## Extension Headers

The length of the base header is fixed at 40 bytes. However, to give greater functionality to the IP datagram, the base header can be followed by up to six extension headers. Many of these headers are options in IPv4. Six types of extension headers have been defined, as shown in Figure 20.17.

Figure 20.17 *Extension header types*




### *Hop-by-Hop Option*

The hop-by-hop option is used when the source needs to pass information to all routers visited by the datagram. So far, only three options have been defined: Padl, PadN, and jumbo payload. The Padl option is 1 byte long and is designed for alignment purposes. PadN is similar in concept to Padi. The difference is that PadN is used when 2 or more bytes is needed for alignment. The jumbo payload option is used to define a payload longer than 65,535 bytes.

Source Routing  The source routing extension header combines the concepts of the strict source route and the loose source route options of IPv4.

### *Fragmentation*

The concept of fragmentation is the same as that in IPv4. However, the place where fragmentation occurs differs. In IPv4, the source or a router is required to fragment if the size of the datagram is larger than the MTU of the network over which the datagram travels. In IPv6, only the original source can fragment. A source must use a path MTU discovery technique to find the smallest MTU supported by any network on the path. The source then fragments using this knowledge.

### *Authentication*

The authentication extension header has a dual purpose: it validates the message sender and ensures the integrity of data. We discuss this extension header when we discuss network security in Chapter 31.

*Encrypted Security Payload*

The encrypted security payload (ESP) is an extension that provides confidentiality and guards against eavesdropping. We discuss this extension header in Chapter 31.

Destination Option The destination option is used when the source needs to pass information to the destination only. Intermediate routers are not permitted access to this information.

### *Comparison Between IPv4 Options and IPv6 Extension Headers*

Table 20.10 compares the options in IPv4 with the extension headers in IPv6.

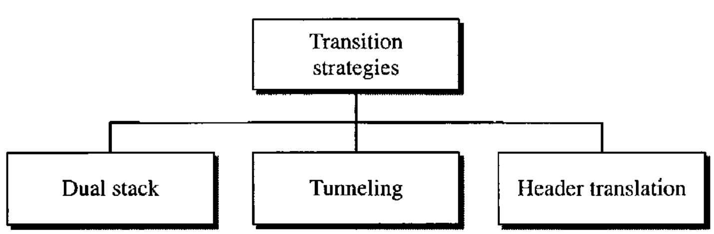
Table 20.10 *Comparison between IPv4 options and IPv6 extension headers*

| *Comparison* |
|---|
| 1. The no-operation and end-of-option options in IPv4 are replaced by Padl and PadN options in IPv6. |
| 2. The record route option is not implemented in IPv6 because it was not used. |
| 3. The timestamp option is not implemented because it was not used. |
| 4. The source route option is called the source route extension header in IPv6. |
| 5. The fragmentation fields in the base header section of IPv4 have moved to the fragmentation extension header in IPv6. |
| 6. The authentication extension header is new in IPv6. |
| 7. The encrypted security payload extension header is new in IPv6. |

## 20.4 TRANSITION FROM IPv4 TO IPv6

Because of the huge number of systems on the Internet, the transition from IPv4 to IPv6 cannot happen suddenly. It takes a considerable amount of time before every system in the Internet can move from IPv4 to IPv6. The transition must be smooth to prevent any problems between IPv4 and IPv6 systems. Three strategies have been devised by the !E1F to help the transition (see Figure 20.18).

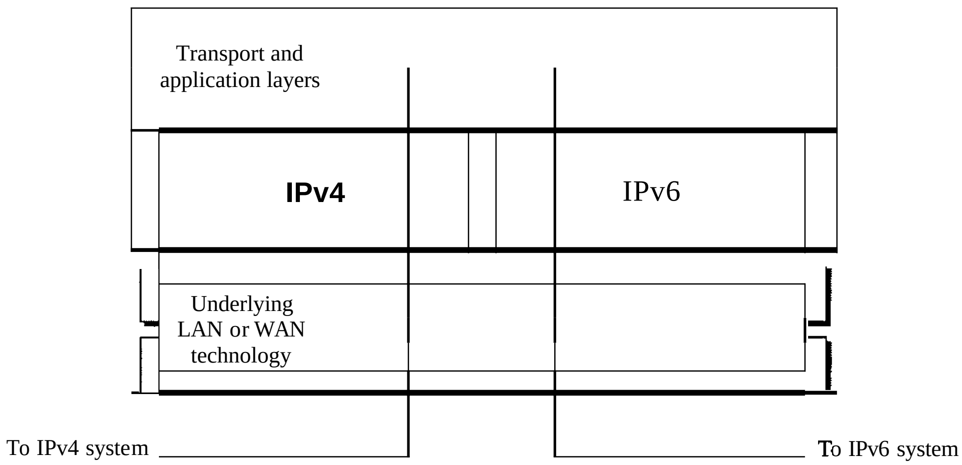
Figure 20.18 *Three transition strategies*



## Dual Stack

It is recommended that all hosts, before migrating completely to version 6, have a **dual** stack of protocols. In other words, a station must run IPv4 and IPv6 simultaneously until all the Internet uses IPv6. See Figure 20.19 for the layout of a dual-stack configuration.

Figure 20.19 *Dual stack*

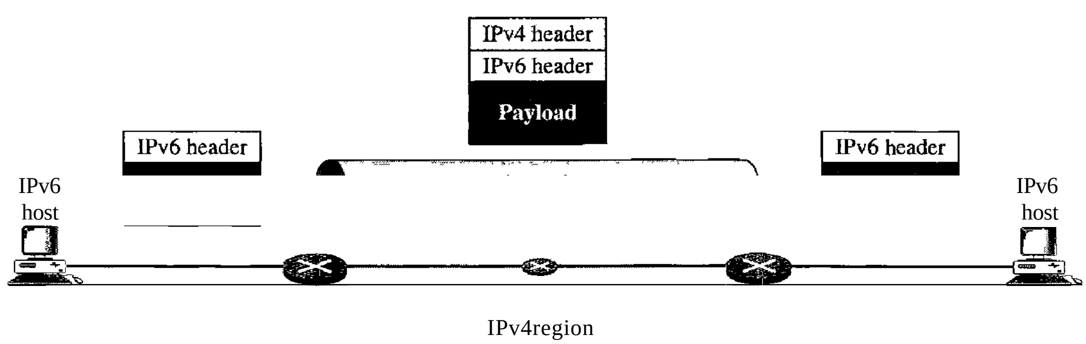



To determine which version to use when sending a packet to a destination, the source host queries the DNS. If the DNS returns an IPv4 address, the source host sends an IPv4 packet. If the DNS returns an IPv6 address, the source host sends an IPv6 packet.

## Tunneling

**Thnneling** is a strategy used when two computers using IPv6 want to communicate with each other and the packet must pass through a region that uses IPv4. To pass through this region, the packet must have an IPv4 address. So the IPv6 packet is encapsulated in an IPv4 packet when it enters the region, and it leaves its capsule when it exits the region. It seems as if the IPv6 packet goes through a tunnel at one end and emerges at the other end. To make it clear that the IPv4 packet is carrying an IPv6 packet as data, the protocol value is set to 41. Tunneling is shown in Figure 20.20.

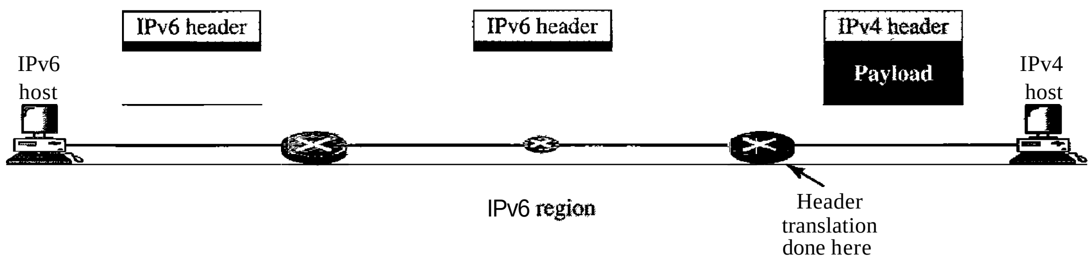
Figure 20.20 *Tunneling strategy*



### Header Translation

Header translation is necessary when the majority of the Internet has moved to IPv6 but some systems still use IPv4. The sender wants to use IPv6, but the receiver does not understand IPv6. Tunneling does not work in this situation because the packet must be in the IPv4 format to be understood by the receiver. In this case, the header format must be totally changed through header translation. The header of the IPv6 packet is converted to an IPv4 header (see Figure 20.21).

Figure 20.21 *Header translation strategy*




Header translation uses the mapped address to translate an IPv6 address to an IPv4 address. Table 20.11 lists some rules used in transforming an IPv6 packet header to an IPv4 packet header.

Table 20.11 *Header translation*

| *Header Translation Procedure* |
|---|
| 1. The IPv6 mapped address is changed to an IPv4 address by extracting the rightmost 32 bits. |
| 2. The value of the IPv6 priority field is discarded. |
| 3. The type of service field in IPv4 is set to zero. |
| 4. The checksum for IPv4 is calculated and inserted in the corresponding field. |
| 5. The IPv6 flow label is ignored. |
| 6. Compatible extension headers are converted to options and inserted in the IPv4 header. Some may have to be dropped. |
| 7. The length of IPv4 header is calculated and inserted into the corresponding field. |
| 8. The total length of the IPv4 packet is calculated and inserted in the corresponding field. |

## 20.5 RECOMMENDED READING

For more details about subjects discussed in this chapter, we recommend the following books and sites. The items in brackets [...] refer to the reference list at the end of the text.

### Books

IPv4 is discussed in Chapter 8 of [For06], Chapter 3 of [Ste94J, Section 4.1 of [PD03], Chapter 18 of [Sta04], and Section 5.6 of [Tan03]. IPv6 is discussed in Chapter 27 of [For06] and [Los04].

### Sites

O www.ietforg/rfc.html Information about RFCs

### RFCs

A discussion of IPv4 can be found in following RFCs:

760,781,791,815,1025,1063,1071,1141,1190, 1191, 1624,2113

A discussion of IPv6 can be found in the following RFCs:

1365,1550,1678,1680,1682,1683,1686,1688,1726, 1752, 1826, 1883, 1884, 1886, 1887, 1955,2080,2373,2452,2463,2465,2466,2472,2492,2545,2590

## 20.6 KEY TERMS

authentication
base header
best-effort delivery
codepoint
connectionless service
connection-oriented service
datagram
destination address
destination option
differentiated services
dual stack
encrypted security payload (ESP)
end-of-option option
extension header
flow label
fragmentation
fragmentation offset
header length
header translation
hop limit
hop-by-hop option
Internet Protocol (IP)
Internet Protocol, next generation (IPng)
Internet Protocol version 4 (IPv4)
Internet Protocol version 6 (IPv6)
jumbo payload
loose source route option
maximum transfer unit (MTU)
next header
noncongestion-controlled traffic
no-operation option
packet priority
Padl
PadN
path MTU discovery technique
precedence

| | |
|---|---|
| record route option | time to live |
| service type | timestamp option |
| source address | tunneling |
| strict source route option | type of service (TOS) |

## 20.7 SUMMARY

- IPv4 is an unreliable connectionless protocol responsible for source-to-destination delivery.
- Packets in the IPv4 layer are called datagrams. A datagram consists of a header (20 to 60 bytes) and data. The maximum length of a datagram is 65,535 bytes.
- The MTU is the maximum number of bytes that a data link protocol can encapsulate. MTUs vary from protocol to protocol.
- Fragmentation is the division of a datagram into smaller units to accommodate the MTU of a data link protocol.
- The IPv4 datagram header consists of a fixed, 20-byte section and a variable options section with a maximum of 40 bytes.
- The options section of the IPv4 header is used for network testing and debugging.
- The six IPv4 options each have a specific function. They are as follows: filler between options for alignment purposes, padding, recording the route the datagram takes, selection of a mandatory route by the sender, selection of certain routers that must be visited, and recording of processing times at routers.
- IPv6, the latest version of the Internet Protocol, has a 128-bit address space, a revised header format, new options, an allowance for extension, support for resource allocation, and increased security measures.
- An IPv6 datagram is composed of a base header and a payload.
- Extension headers add functionality to the IPv6 datagram.
- Three strategies used to handle the transition from version 4 to version 6 are dual stack, tunneling, and header translation.

## 20.8 PRACTICE SET

### Review Questions

1. What is the difference between the delivery of a frame in the data link layer and the delivery of a packet in the network layer?
2. What is the difference between connectionless and connection-oriented services? Which type of service is provided by IPv4? Which type of service is provided by IPv6?
3. Define fragmentation and explain why the IPv4 and IPv6 protocols need to fragment some packets. Is there any difference between the two protocols in this matter?

4. Explain the procedure for checksum calculation and verification in the IPv4 protocol. What part of an IPv4 packet is covered in the checksum calculation? Why? Are options, if present, included in the calculation?
5. Explain the need for options in IPv4 and list the options mentioned in this chapter with a brief description of each.
6. Compare and contrast the fields in the main headers of IPv4 and IPv6. Make a table that shows the presence or absence of each field.
7. Both IPv4 and IPv6 assume that packets may have different priorities or precedences. Explain how each protocol handles this issue.
8. Compare and contrast the options in IPv4 and the extension headers in IPv6. Make a table that shows the presence or absence of each.
9. Explain the reason for the elimination of the checksum in the IPv6 header.
10. List three transition strategies to move from IPv4 to IPv6. Explain the difference between tunneling and dual stack strategies during the transition period. When is each strategy used?

## Exercises

11. Which fields of the IPv4 header change from router to router?
12. Calculate the HLEN (in IPv4) value if the total length is 1200 bytes, 1176 of which is data from the upper layer.
13. Table 20.5 lists the MTUs for many different protocols. The MTUs range from 296 to 65,535. What would be the advantages of having a large MTU? What would be the advantages of having a small MTU?
14. Given a fragmented datagram (in IPv4) with an offset of 120, how can you determine the first and last byte numbers?
15. Can the value of the header length in an IPv4 packet be less than 5? When is it exactly 5?
16. The value of HLEN in an IPv4 datagram is 7. How many option bytes are present?
17. The size of the option field of an IPv4 datagram is 20 bytes. What is the value of HLEN? What is the value in binary?
18. The value of the total length field in an IPv4 datagram is 36, and the value of the header length field is 5. How many bytes of data is the packet carrying?
19. An IPv4 datagram is carrying 1024 bytes of data. If there is no option information, what is the value of the header length field? What is the value of the total length field?
20. A host is sending 100 datagrams to another host. If the identification number of the first datagram is 1024, what is the identification number of the last (in IPv4)?
21. An IPv4 datagram arrives with fragmentation offset of 0 and an *M* bit (*more* fragment bit) of 0. Is this a first fragment, middle fragment, or last fragment?
22. An IPv4 fragment has arrived with an offset value of 100. How many bytes of data were originally sent by the source before the data in this fragment?

23. An IPv4 datagram has arrived with the following information **in** the header (in hexadecimal):

    Ox45 00 00 54 00 03 58 50 20 06 00 00 7C 4E 03 02 B4 OE OF 02

    a. Is the packet corrupted?
    b. Are there any options?
    c. Is the packet fragmented?
    d. What is the size of the data?
    e. How many more routers can the packet travel to?
    f. What is the identification number of the packet?
    g. What is the type of service?

24. In an IPv4 datagram, the *M* bit is 0, the value of HLEN is 5, the value of total length is 200, and the offset value is 200. What is the number of the first byte and number of the last byte **in** this datagram? Is this the last fragment, the first fragment, or a middle fragment?

## Research Activities

25. Find out why there are two security protocols (AH and ESP) **in** IPv6.