Module 3

Network Layer: Logical Addressing

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.

Topics discussed in this section:

Address Space
Notations
Classful Addressing
Classless Addressing
Network Address Translation (NAT)

Note

An IPv4 address is 32 bits long.

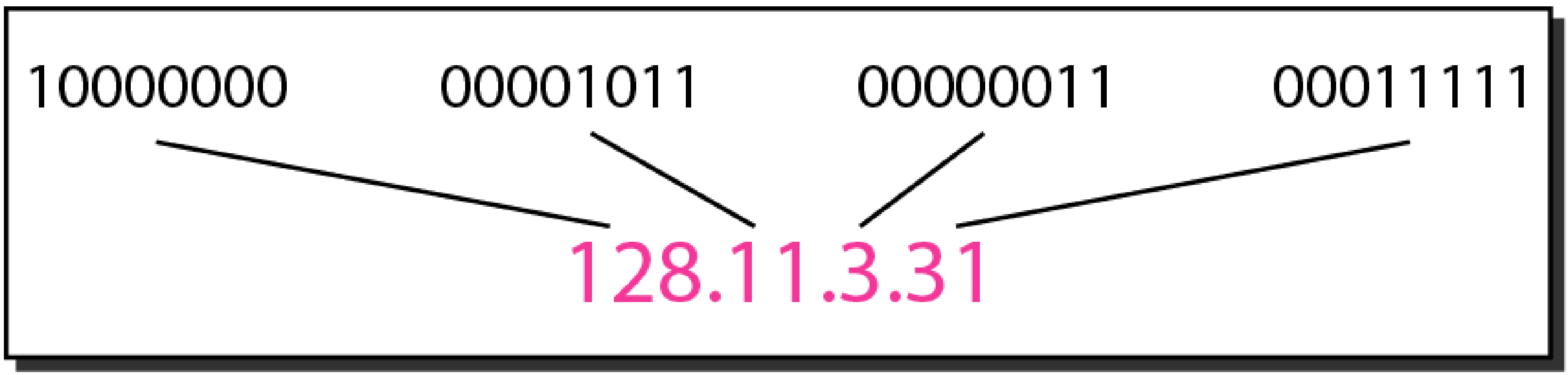
Note

The IPv4 addresses are unique and universal.

Note

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

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



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 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 (8bit).

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.

c. Each number needs to be less than or equal to 255.

d. A mixture of binary notation and dotted-decimal notation is not allowed.

Note

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

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

10101100 172.x.x.x

	First byte	Second byte	Third byte	Fourth byte
Class A	0			
Class B	10			
Class C	110			
Class D	1110			
Class E	1111			

a. Binary notation

	First byte	Second byte	Third byte	Fourth byte
Class A	0–127			
Class B	128–191			
Class C	192–223			
Class D	224–239			
Class E	240–255			

b. Dotted-decimal notation

Figure: Hierarchy in IP Address

Note

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

Example 19.4

Find the class of each address.

a. <u>**0**</u>0000001 00001011 00001011 11101111

b. <u>**110**</u>00001 10000011 00011011 11111111

c. <u>**14**</u>.23.120.8

d. <u>**172**</u>.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; the class is A.

d. The first byte is 172; the class is B.

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

Mask

- It is a 32 bit number in which the 'n' leftmost bits are 1s and 32-n rightmost bits are 0s

Table ***Default masks for classful addressing***

Class	*Binary*	*Dotted-Decimal*	*CIDR*
A	**11111111** 00000000 00000000 00000000	**255**.0.0.0	/8
B	**11111111 11111111** 00000000 00000000	**255.255**.0.0	/16
C	**11111111 11111111 11111111** 00000000	**255.255.255**.0	/24

Note

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

Note

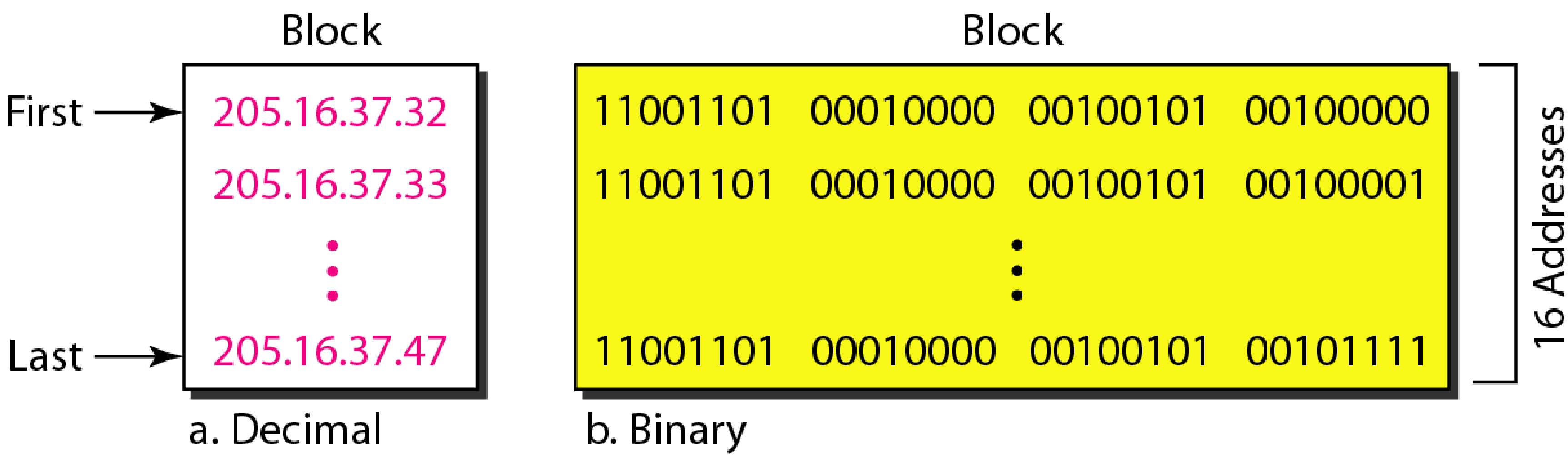
Classful addressing, which is almost obsolete, is replaced with classless addressing.

Sub-netting and Super-netting

- Sub-netting: Dividing the addresses into several contiguous groups and assigning each group to smaller networks, called subnets
  - Increases No. of 1s in the mask
- Super-netting: Combining several (class-C) address blocks to create a larger network; super-net
  - Decreases No. of 1s in the mask

Ex19.5, Fig 19.3 ***A block of 16 addresses granted to a small organization***

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



Example 19.5

Restrictions applied to this block

- ***The addresses must be contiguous***
- ***The number of addresses must be power of 2 (here, 16 = 2^4),***

  and
- ***The first address must be evenly divisible by number of addresses (here, 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.

Note

In IPv4 addressing, a block of addresses can be defined as

x.y.z.t /*n*

in which x.y.z.t defines one of the addresses and the */n* defines the mask.

Note

The first address in the block can be found by setting the rightmost 32 − *n* bits to 0s

Example 19.6

A block of addresses is granted to a small organization. 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 00100111

If we set 32−28 (i.e 4) rightmost bits to 0, we get

11001101 00010000 00100101 00100000

or

205.16.37.32

This is actually the first address of block shown in Figure 19.3.

Note

The last address in the block can be found by setting the rightmost 32 − n bits to 1s.

Example 19.7

Find the last address for the block if one of the address is 205.16.37.39/28 (Example 19.6).

Solution

The binary representation of the given address is

11001101 00010000 00100101 00100111

If we set 32 − 28 rightmost bits to 1, we get

11001101 00010000 00100101 00101111

or

205.16.37.47

This is actually the last address block shown in Figure 19.3.

Note

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.
Find the number of addresses in the block if one of the address is 205.16.37.39/28

Solution

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

Example 19.9 Alternate Method

Another way to find the first address, the last address, and the number of addresses particularly useful when we are writing a program to find this information.

Methodology

Represent the mask as a 32-bit binary (or 8-digit hexadecimal) number

In Example 19.5 the mask /28 can be represented as

11111111 11111111 11111111 11110000

(twenty-eight 1s and four 0s)

Example 19.9 (continued)

a. ***The first address can be found by ANDing the given addresses with the mask; ANDing here is done bit by bit.***

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

Example 19.9 (continued)

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.***

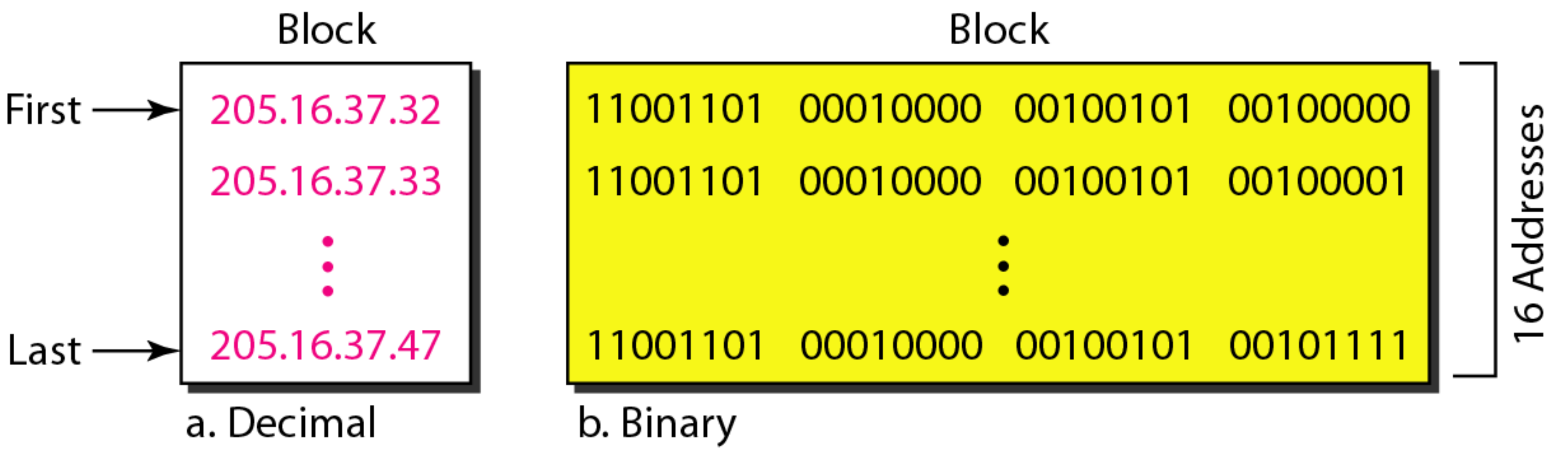
Address:	11001101	00010000	00100101	00100111
Mask complement:	**00000000**	**00000000**	**00000000**	**00001111**
Last address:	11001101	00010000	00100101	00101111

Example 19.9 (continued)

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

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



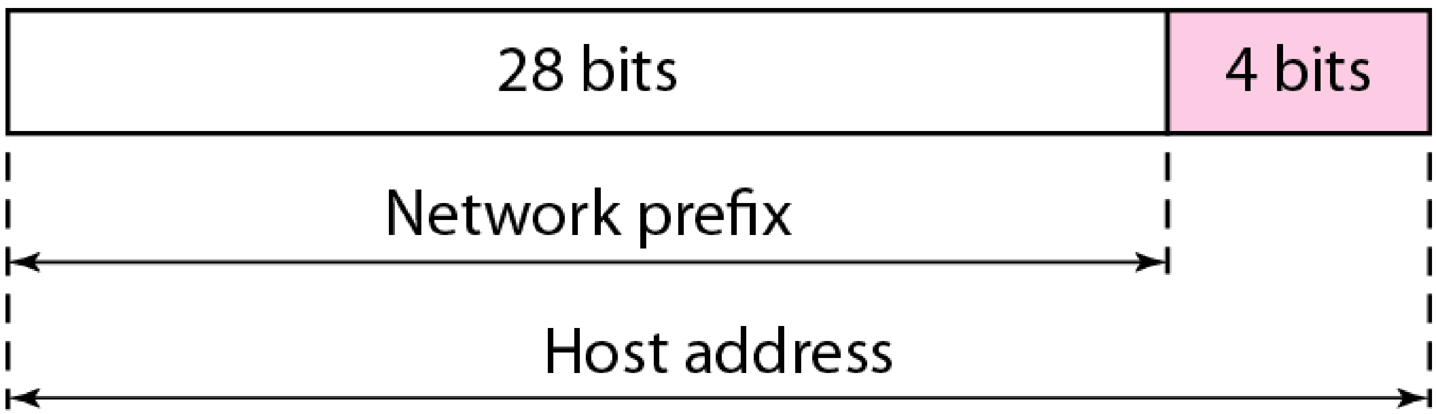
Example: Find first and last IP address for given following address

- 192.168.14.5/ 255.255.255.0
- 11000000 10101000 00001110 00000101
- 11111111 11111111 11111111 00000000
- first IP → 192.168.14.0
- Last address- compliment the mask and perform bit wise OR operation with given IP address
- 0000000 00000000 00000000 11111111
- Last IP add→ 192.168.14.255

Note

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.

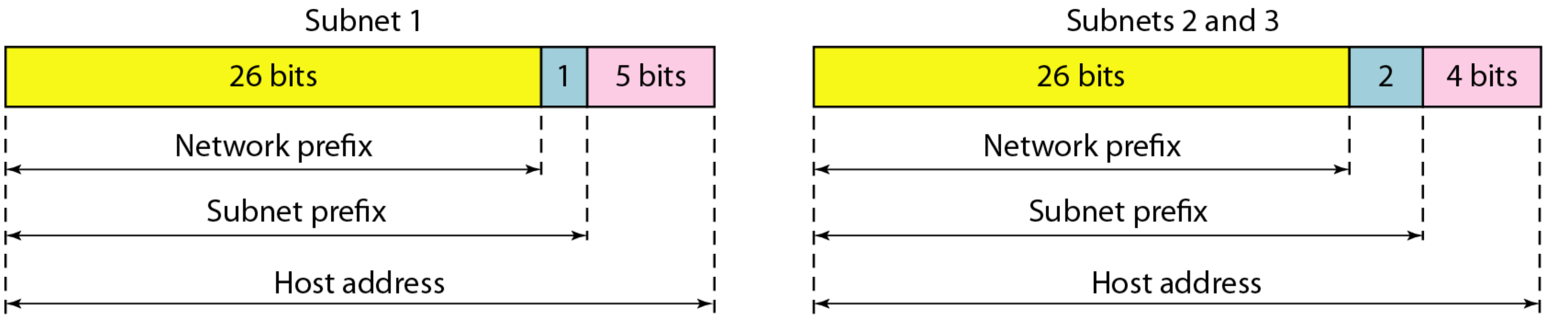
Figure 19.6 *Two Level Hierarchy in IPv4 Address*



Note

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

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



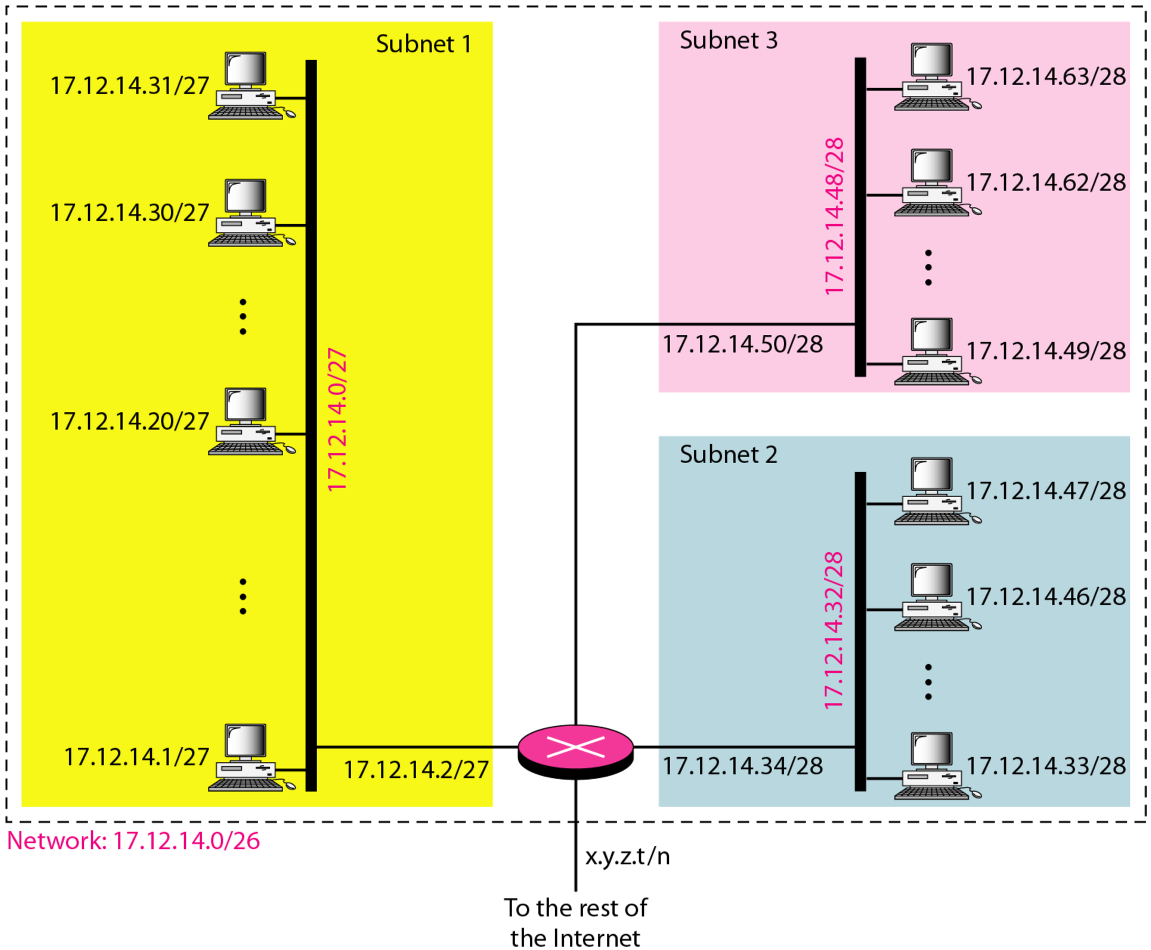
Example 19.10

An organization is given a block of addresses starting with 17.12.40.0/26 (64 addresses). The organization has 3 offices and needs to distribute these addresses into three sub-blocks of 32, 16 and 16 addresses.
Carry out the allocation.

Solution:

- ***Given mask is /26 to define 64 addresses***
- ***Calculate new mask for each sub-block (subnet)***
- ***32 addresses in sub-block 1 → 5 bits reqd ($\log_2 32$)***
  new mask = /27
- ***16 addresses in sub-blocks 2 and 3 → 4bits reqd /28***

Figure 19.7 *Configuration and addresses in a subnetted network*



Example 19.10

An organization is given a block of addresses with one of the address as 172.17.15.15/23
The organization has 3 labs and needs to distribute these addresses into three sub-blocks of 250, 125, and 50 addresses.

IP: ***10101100 00010001 00001111 00001111***
SM: ***11111111 11111111 11111110 00000000***
00000000 00000000 00000001 11111111
Ist address→172.17.14.0
Last adress→172.17.15.255
Solution:Given mask is /23 → 512 addresses

- ***Calculate new mask for each sub-block (subnet)***
- ***250 addresses in sub-block 1→8 bits reqd, new mask = /24***

New masks for the sub-blocks = /24, /25, /26

1st : 172.17.14.0--172.17.14.255
2nd: 172.17.15.0 --172.17.15.127
3rd: 172.17.15.128 – 172.17.15.191

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 sub-blocks and find out how many addresses are still available after these allocations.

Example 19.10 (continued)

Solution

Group 1

For this group, each customer needs 256 addresses. This means that 8 ($\log_2 256$) 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 × 256 = 16,384		

Example 19.10 (continued)

Group 2

For this group, each customer needs 128 addresses. This means that 7 ($\log_2$ 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 $= 128 \times 128 = 16{,}384$

Example 19.10 (continued)

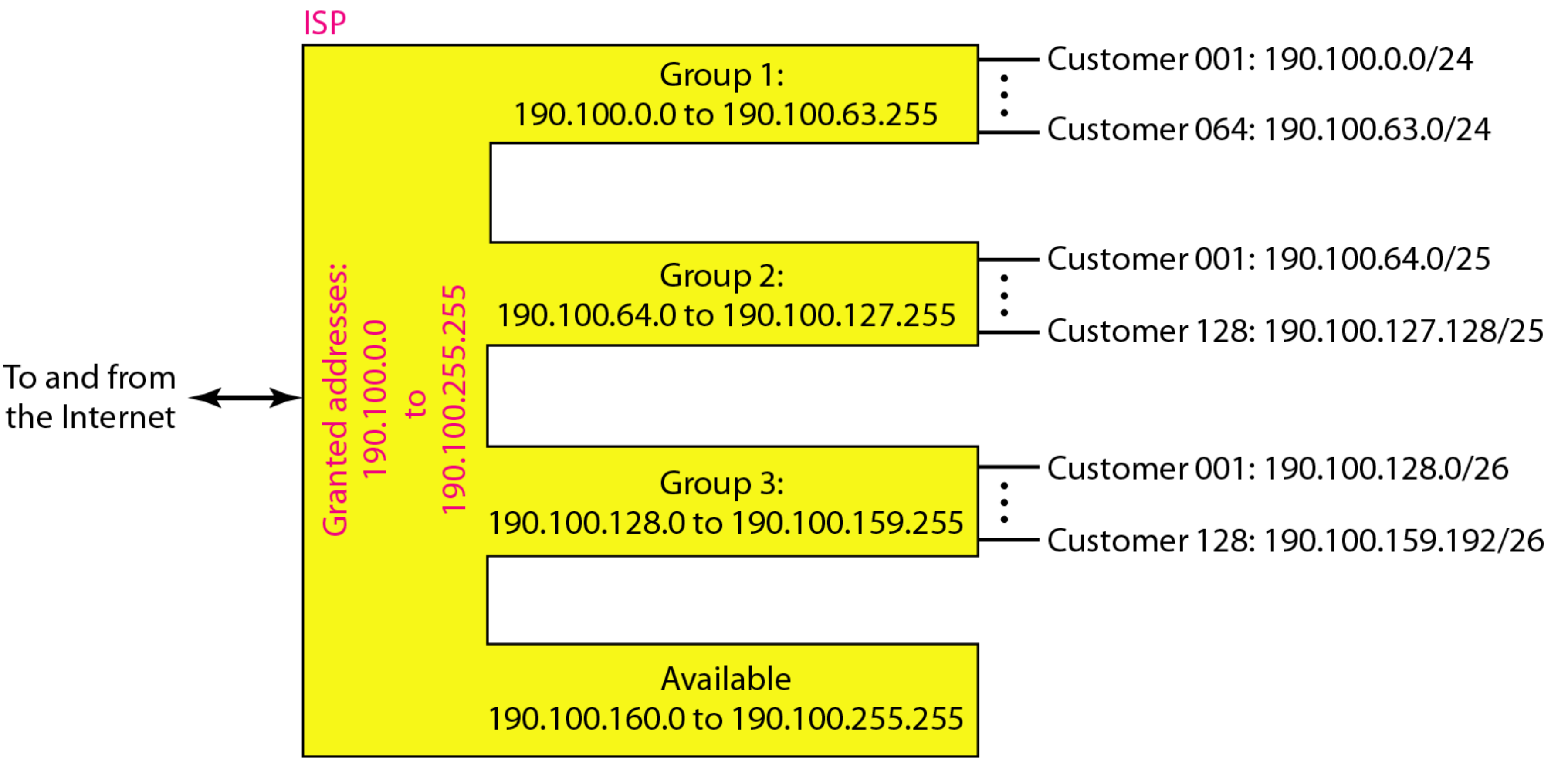
Group 3

For this group, each customer needs 64 addresses. This means that 6 ($\log_2 64$) 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 × 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
Figure 19.9 shows the situation.

Figure 19.9 *An example of address allocation and distribution by an ISP*



Network Address Translation (NAT)

- Shortage of addresses
- NAT enables a user to have a large set of addresses internally (private IP address/es) and one address/ small set of addresses (public IP address/es)externally

Table 19.3 ***Addresses for private networks (Private IP address range)***

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}

Figure 19.10 *A NAT implementation*

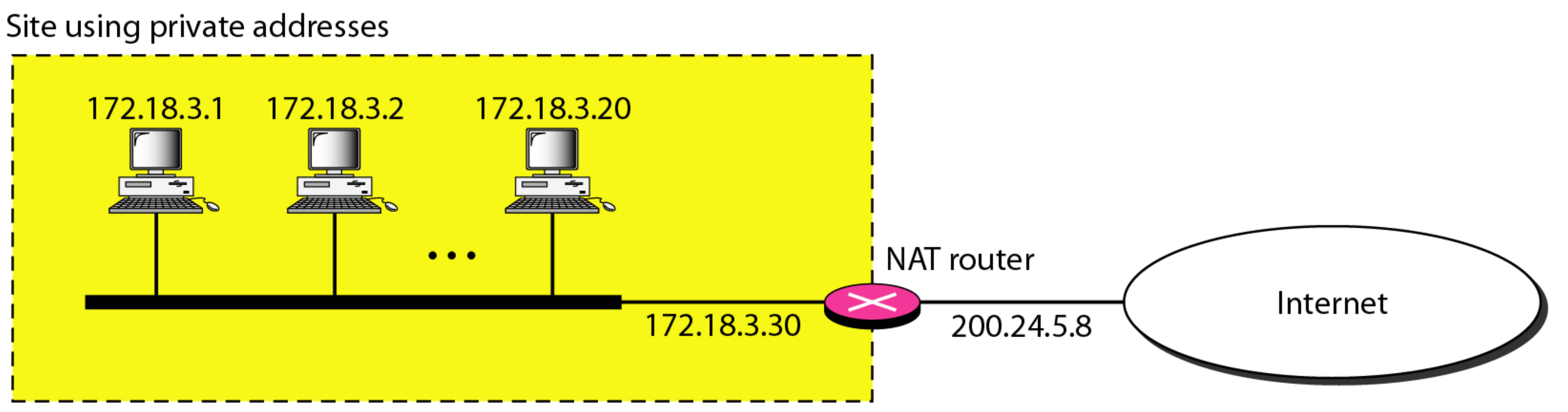


Figure 19.11 *Addresses in a NAT*

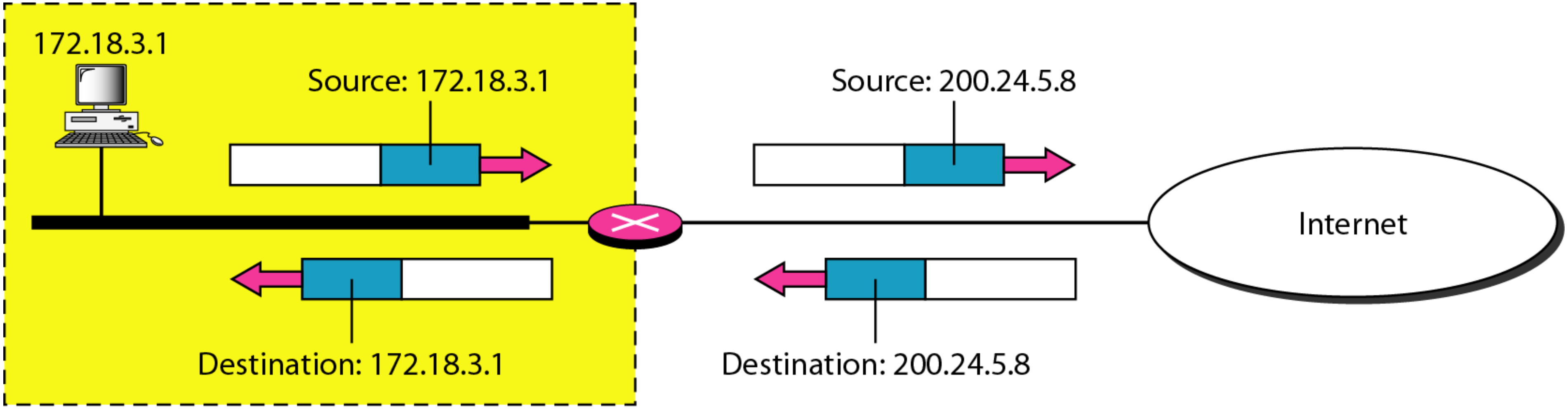
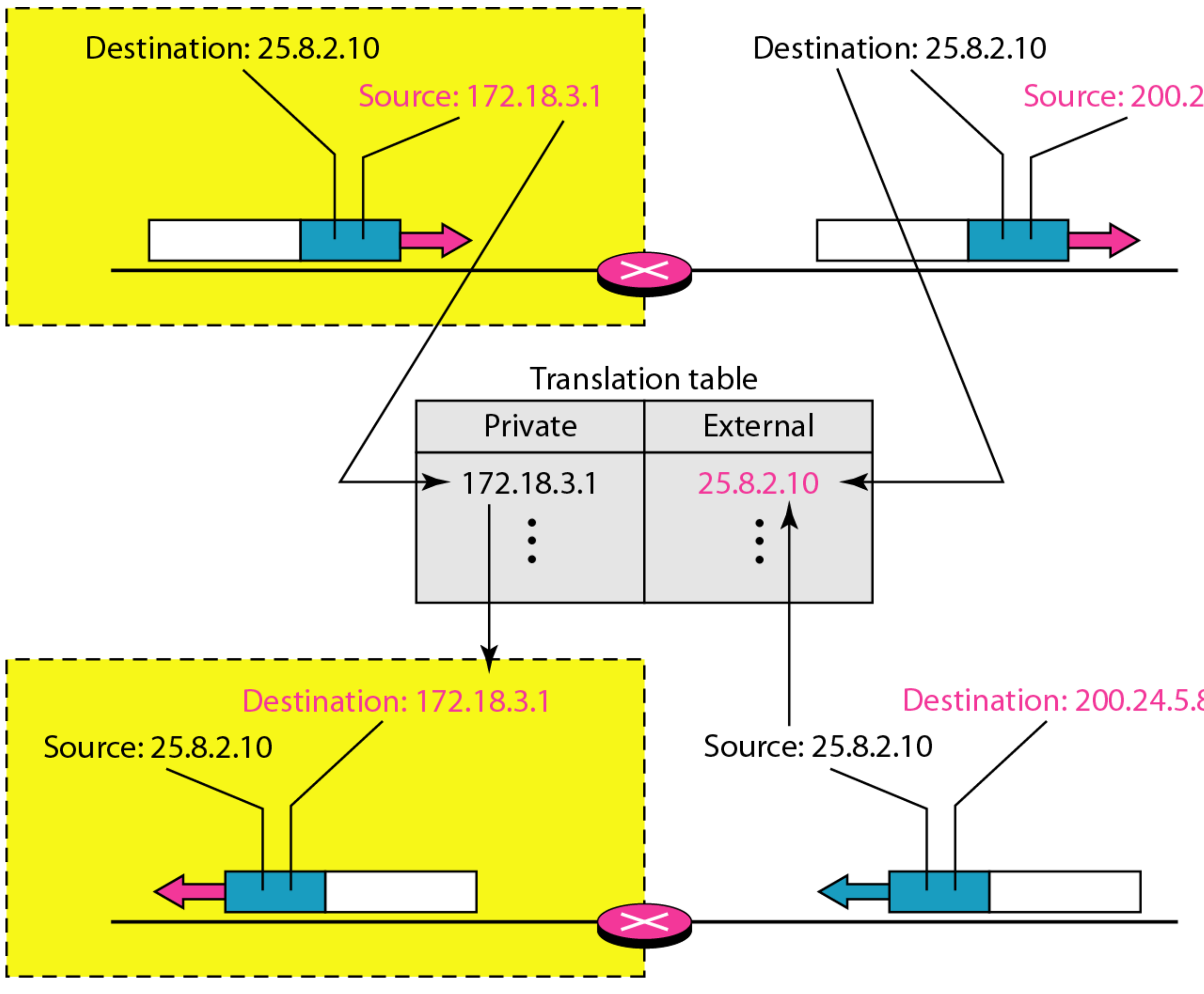


Figure 19.12 *NAT address translation*



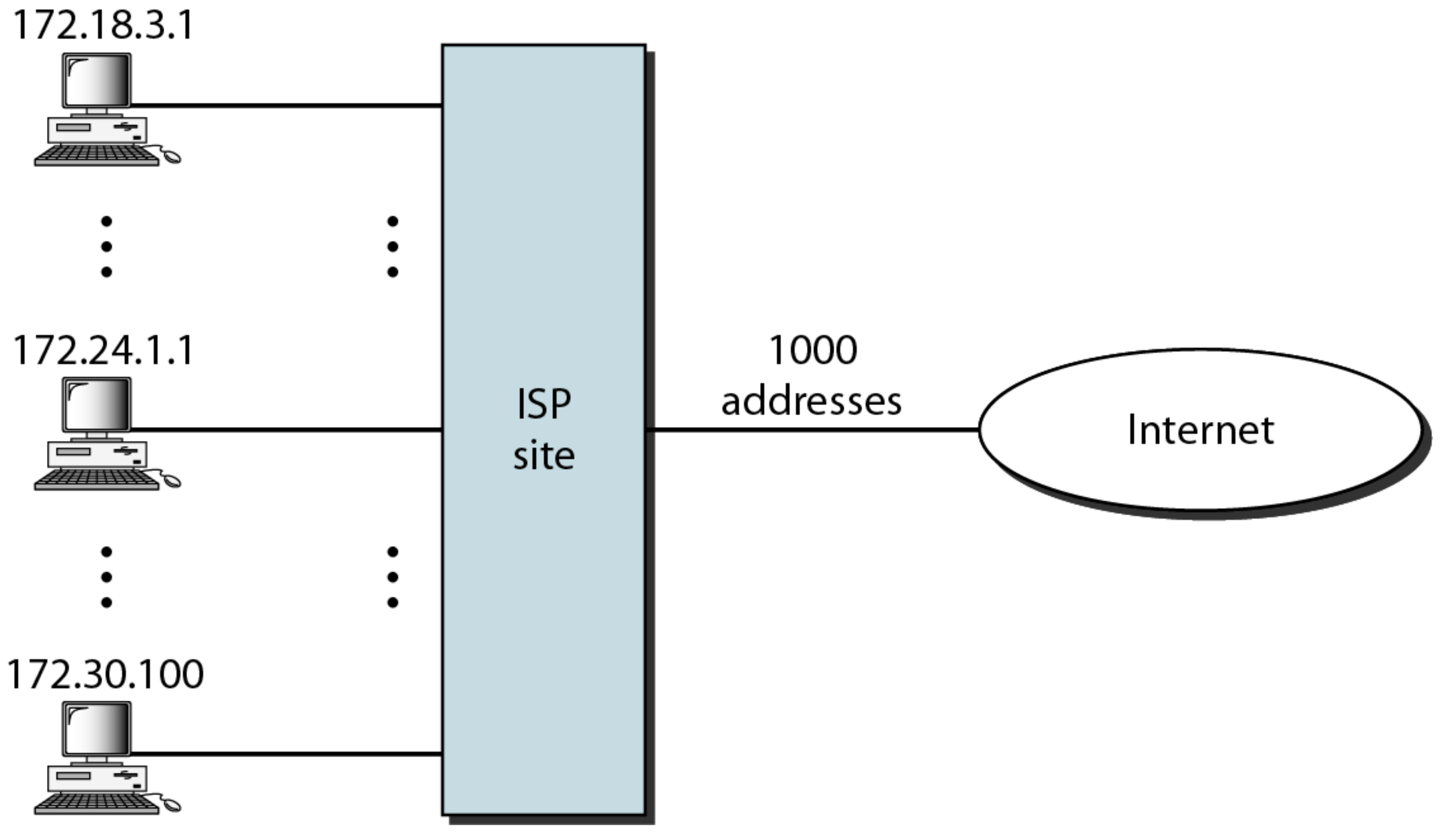
NAT…

- If NAT router has only 1 global address, only 1 private network host can access same external host
- To allow many to many relationship, more information reqd in translation table
- Combination of source address and port number defines the private network host

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
. . .	. . .	. . .	. . .	. . .

Figure 19.13 *An ISP and NAT*



19-2 IPv6 ADDRESSES

Despite all short-term solutions, address depletion is still a long-term problem for the Internet.
→ This and other problems in the IP protocol itself have been the motivation for IPv6.

Topics discussed in this section:

Structure
Address Space

Note

An IPv6 address is 128 bits long.

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

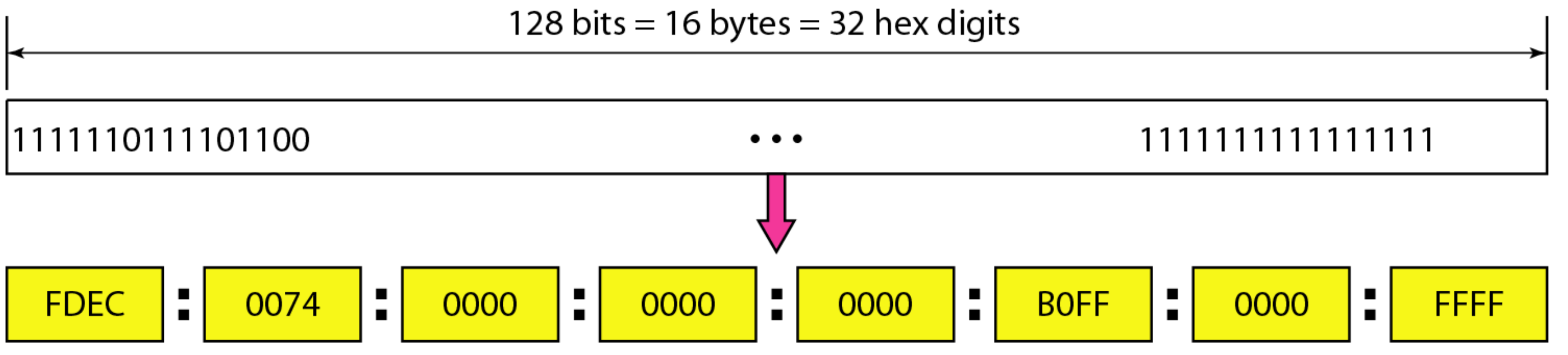
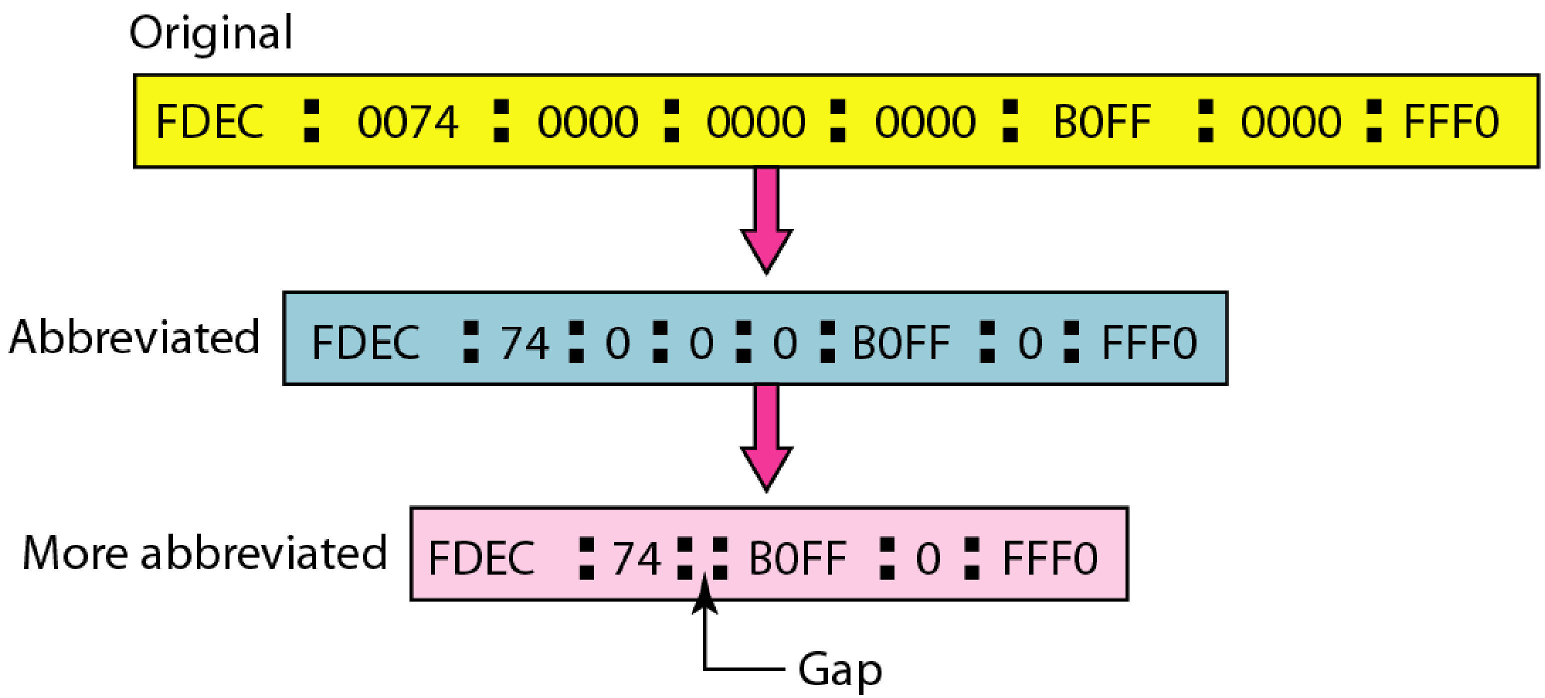


Figure 19.15 *Abbreviated IPv6 addresses*



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 how many 0s we need to replace the double colon.

```
xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx
   0:  15:                 :   1:  12:1213
```

This means that the original address is-

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

IPv6 Address Categories

Table 19.5 ***Type prefixes for IPv6 addresses***

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	1/16
1111 0	Unassigned	1/32
1111 10	Unassigned	1/64
1111 110	Unassigned	1/128
1111 1110 0	Unassigned	1/512
1111 1110 10	Link local addresses	1/1024
1111 1110 11	Site local addresses	1/1024
1111 1111	Multicast addresses	1/256

Figure 19.16 *Prefixes for provider-based unicast address*

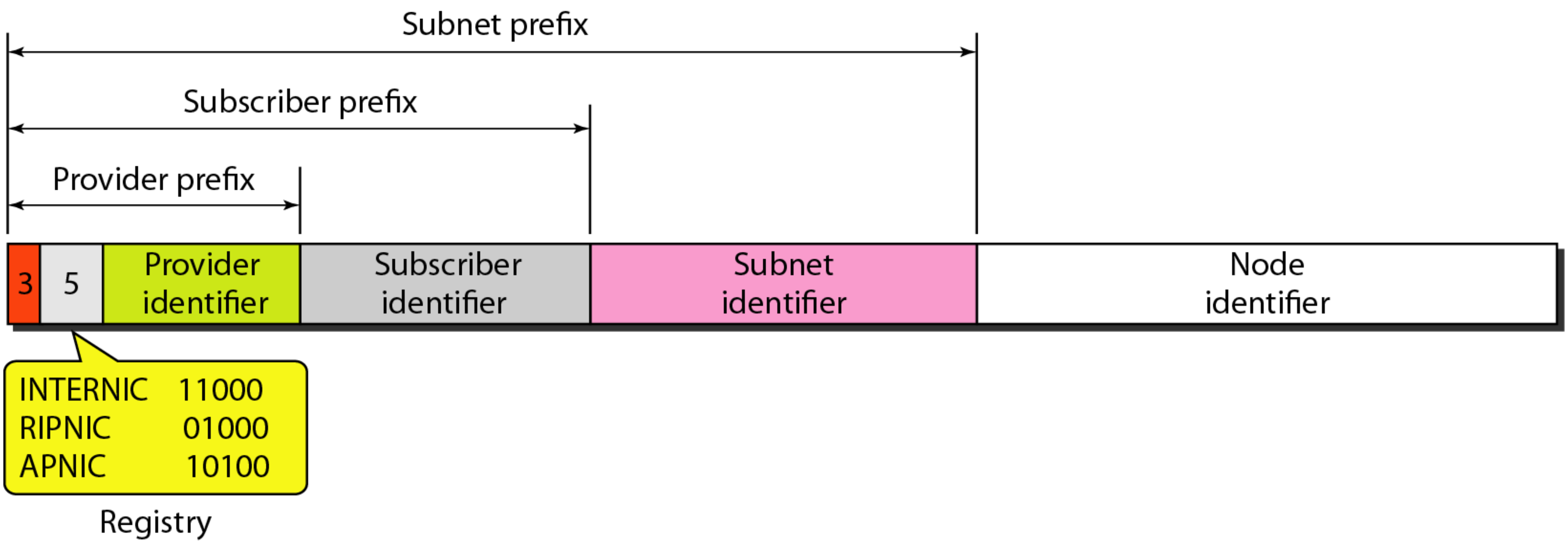


Figure 19.17 *Multicast address and Anycast IPv6*

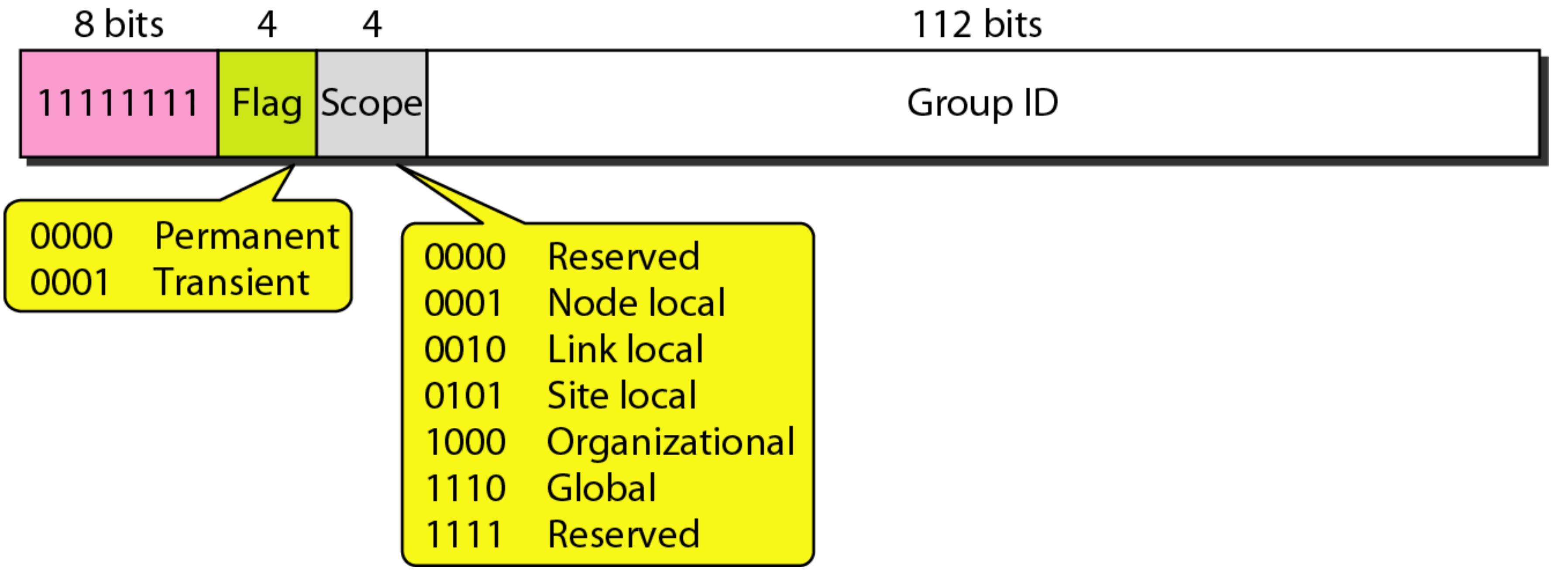


Figure 19.18 *Reserved addresses in IPv6*

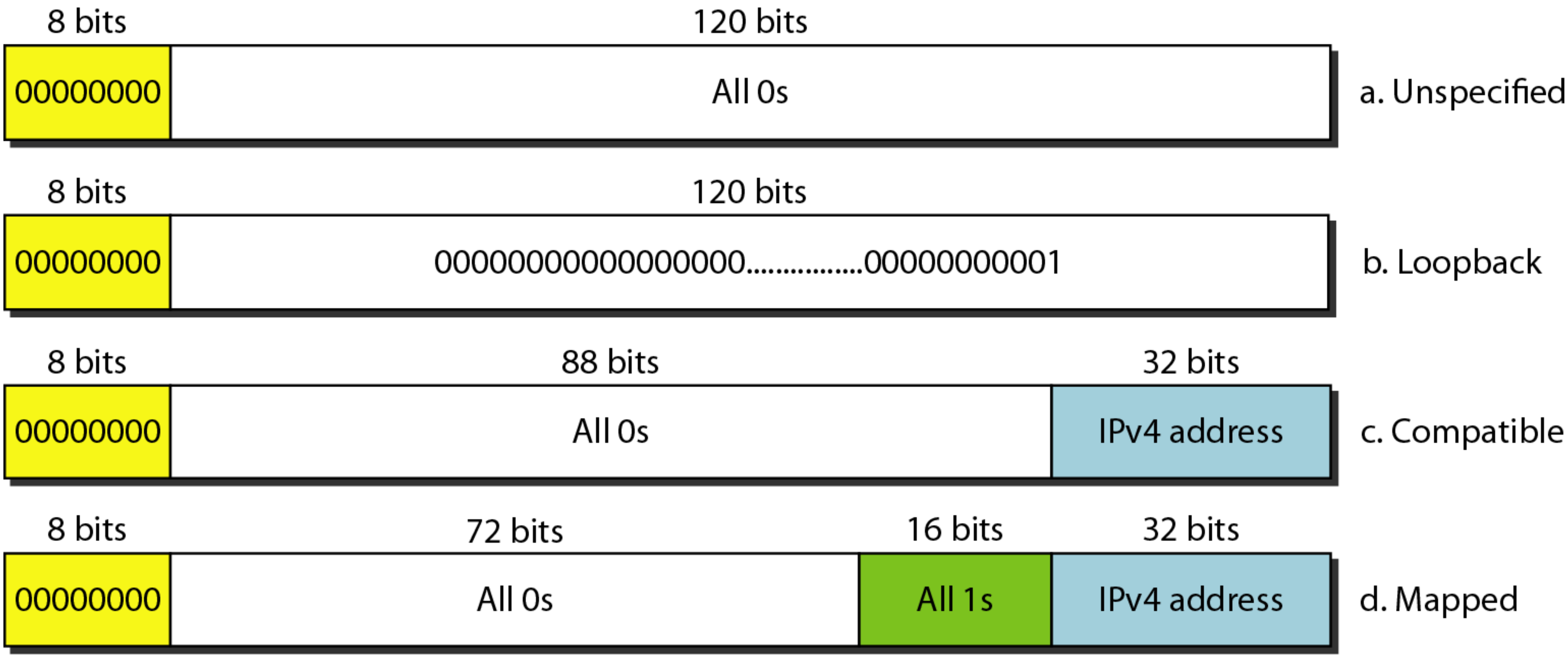
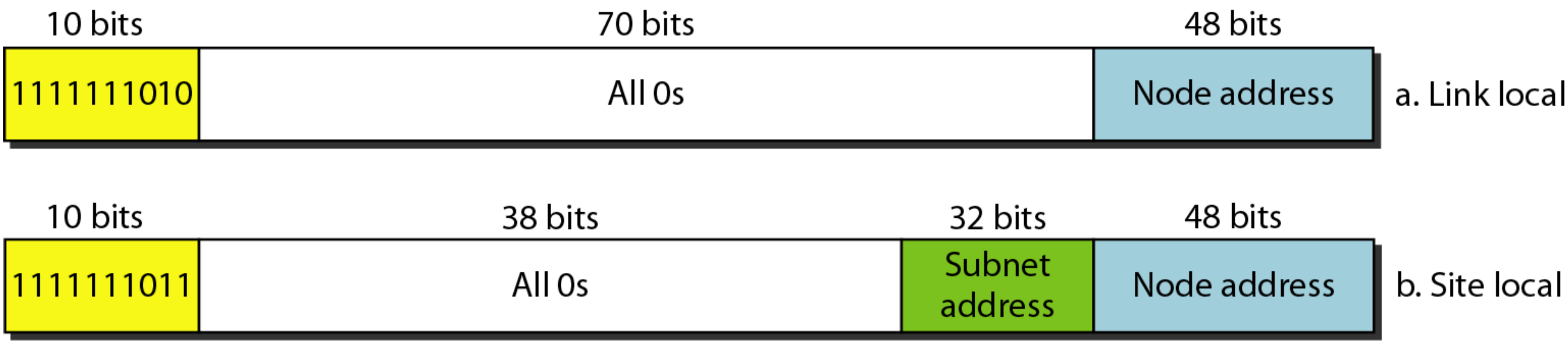


Figure 19.19 *Local addresses in IPv6: (private addresses)*



- IPv4 Datagram Header