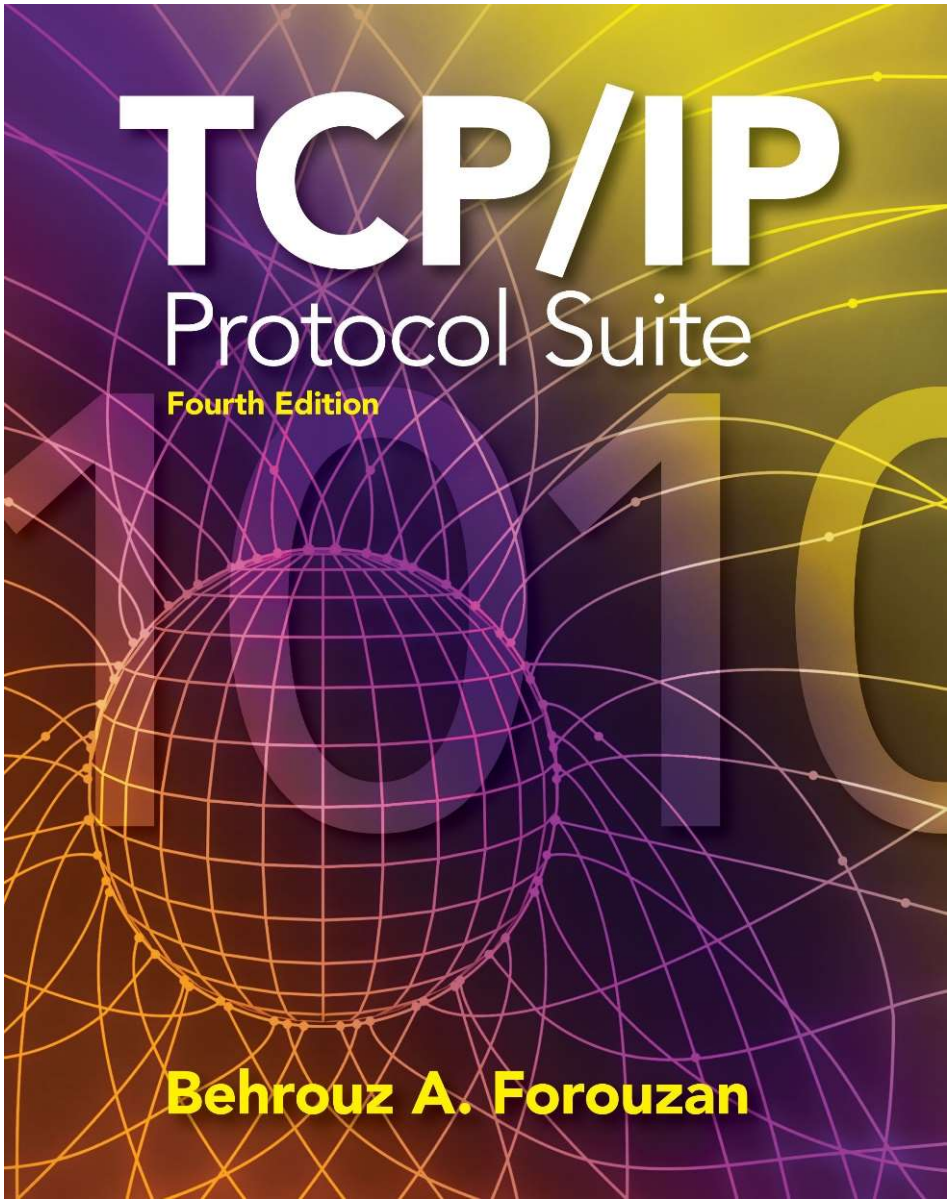
Chapter 5

IPv4 Addresses



OBJECTIVES:

- To introduce the concept of an address space in general and the address space of IPv4 in particular.
- To discuss the classful architecture and the blocks of addresses available in each class.
- To discuss the idea of hierarchical addressing and how it has been implemented in classful addressing.
- To explain subnetting and supernetting for classful architecture.
- To discuss classless addressing, that has been devised to solve the problems in classful addressing.
- To discuss some special blocks and some special addresses in each block.
- To discuss NAT technology and show how it can be used to alleviate of address depletion.

Chapter Outline

5-1 INTRODUCTION

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

Topics Discussed in the Section

- ✓ **Notation**
- ✓ **Range of Addresses**
- ✓ **Operations**

Note

An IPv4 address is 32 bits long.

Note

The IPv4 addresses are unique and universal.

Note

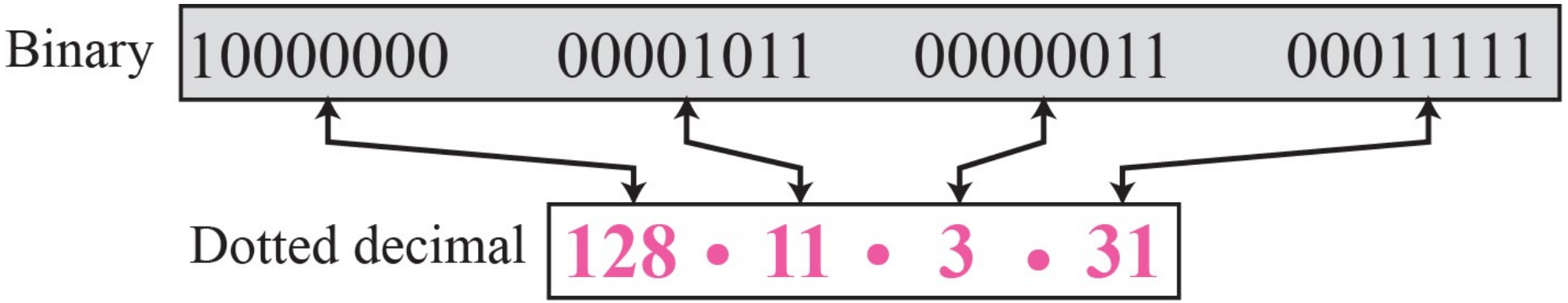
The address space of IPv4 is 2^{32} or 4,294,967,296.

more than four billion

Note

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

Figure 5.1 *Dotted-decimal notation*



IP addresses can be thought of as a number written with 256 base

IP addresses as 256 base Address

10 base(Decimal) Number System(0-9)		256 Base Number System(0-255)	
Position		Position	
10^0	0000…0009	256^0	0.0.0.0 to 0.0.0.255
10^1	0010…0019…0099	256^1	0.0.1.0 …. 0.0.1.255, 0.0.255.0,… , 0.0.255.225
10^2	0100,…..0999	256^2	0.1.0.0, …,0.1.0.255, 0.1.1.0,…,0.1.255.255,… 0.255.0.0,…, 0.255.255.255
10^3	1000,….9999	256^3	1.0.0.0,..,1.0.0.255, 1.0.1.0,.., 1.0.1.255,…, 1.0.255.0,…, 1.0.255.255, …….. 255.255.255.255

128.11.3.31 =

$128*256^3 + 11*256^2 + 3*256^1 + 31*256^0$

$= 2,147,483,648 + 720,896 + 768 + 31$

$= 2,148,205,343$

We can do 256 base addition, subtraction operations on IP addresses

Example 5.1

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

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

Solution

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

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

Example 5.2

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

a. 111.56.45.78
b. 221.34.7.82
c. 241.8.56.12
d. 75.45.34.78

Solution

We replace each decimal number with its binary equivalent:

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

Example 5.3

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

a. 111.56.045.78

b. 221.34.7.8.20

c. 75.45.301.14

d. 11100010.23.14.67

Solution

a. There should be no leading zeroes (045).

b. We may not have more than 4 bytes in an IPv4 address.

c. Each byte (301) should be less than or equal to 255.

d. A mixture of binary notation and dotted-decimal notation.

Example 5.4

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

a. 10000001 00001011 00001011 11101111

b. 11000001 10000011 00011011 11111111

Solution

We replace each group of 4 bits with its hexadecimal equivalent. Note that 0X (or 0x) is added at the beginning or the subscript 16 at the end.

a. 0X810B0BEF or $810B0BEF_{16}$

b. 0XC1831BFF or $C1831BFF_{16}$

Example 5.5

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

146.102.32.255
- 146.102.29.0
==========
0.0.3.255

Solution :

We can subtract the first address from the last address in base 256 (see Appendix B). The result is 0.0.3.255 in this base.

To find the number of addresses in the range (in decimal), we convert this number to base 10 and add 1 to the result..

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

Example 5.6

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

Solution

We convert the number of addresses (32) minus 1 (i.e. 31) to base 256, which is 0.0.0.31. We then add it to the first address to get the last address. Addition is in base 256.

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

Example

The first address in a range of addresses is 14.11.45.96. If the number of addresses in the range is 160, 161 and 162 , what is the last address in each case?

Solution

We convert the number of addresses(160) minus 1 (i.e. 159) to base 256, which is 0.0.0.159. We then add it to the first address to get the last address. Addition is in base 256.

14.11.45.96	14.11.45.96	14.11.45.96
+ 0. 0. 0.159	+ 0. 0. 0.160	+ 0. 0. 0.161
==========	==========	==========
14.11.45.255	14.11.46.0	14.11.46.1

Example

The first address in a range of addresses is 14.11.45.96. If the number of addresses in the range is 300, 420 what is the last address in each case?

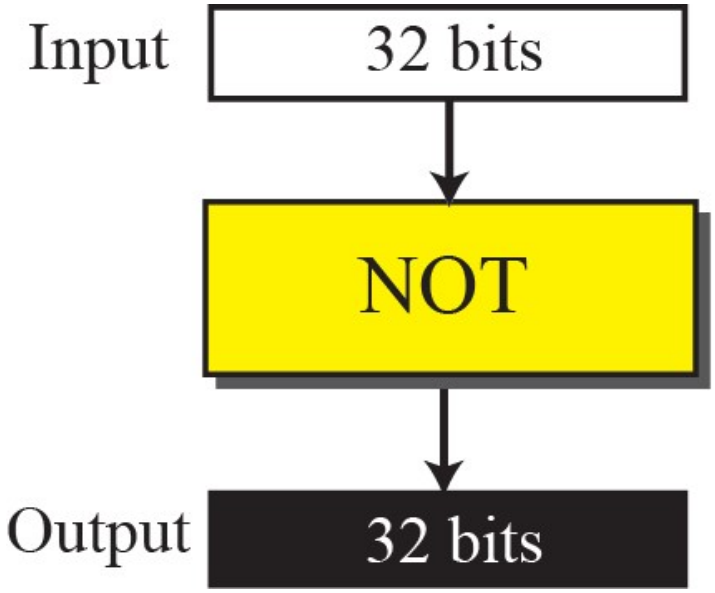
Solution

We convert the number of addresses(300) minus 1 (i.e. 299) to base 256, which is 0.0.1.43 We then add it to the first address to get the last address. Addition is in base 256.

```
  14.11.45.96
+  0.  0. 1.43
==============
  14.11.46.139
```

```
  14.11.45.96
+  0.  0. 1.163
======1======
=   14.11.47.3
```

Figure 5.2 *Bitwise NOT operation*



NOT operation

Input	Output
0	1
1	0

Operation for each bit

Example 5.7

256 base Complement Operation

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

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

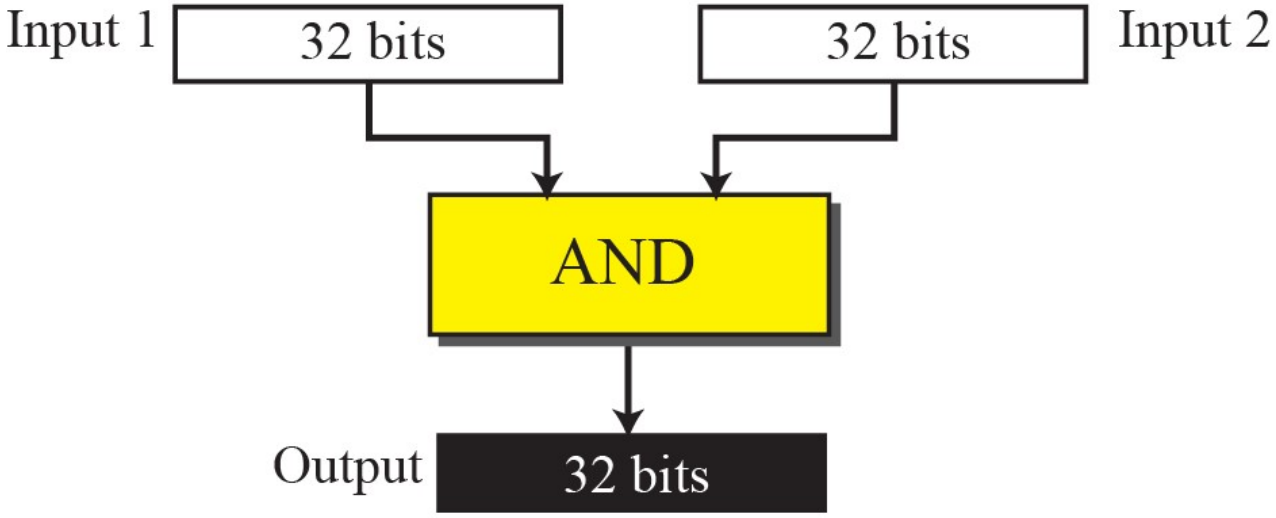
We can use the same operation using the dotted-decimal representation and the short cut.

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

Complement of 17_{256} = 255_{256} - 17_{256} = 238_{256}

Subtracting a given number from 255 gives the 256 base complement of given number

Figure 5.3 *Bitwise AND operation*



AND

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

Operation for each bit

256 base AND Operation

Use two short cuts.

1. When at least one of the numbers is 0 or 255, the AND operation selects the smaller byte (or one of them if equal).

   0 AND X=0 & 255 AND X=X , where X<=255

2. When none of the two bytes is either 0 or 255, we can write each byte as the sum of eight terms, where each term is a power of 2.

   We then select the smaller term in each pair (or one of them if equal) and add them to get the result.

Example 5.8

256 base AND Operation

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

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

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

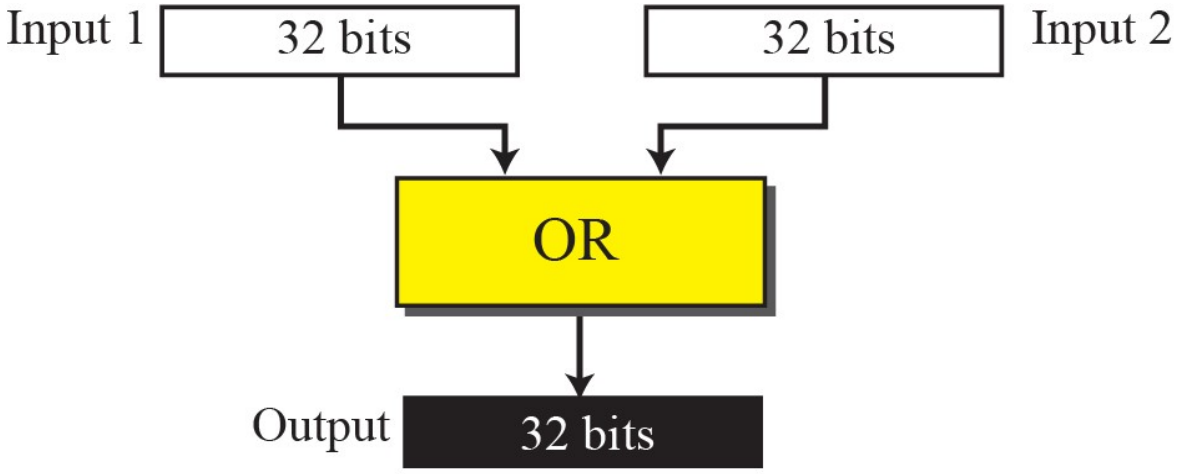
We have applied the first short cut on the first, second, and the fourth byte; we have applied the second short cut on the third byte. We have written 14 and 140 as the sum of terms and selected the smaller term in each pair as shown below.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
0	0	0	0	1	1	1	0	$=14_{(10)}$

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

select the smaller term in each pair (or one of them if equal) and add them to get the result

Figure 5.4 *Bitwise OR operation*



OR

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

Operation for each bit

256 base OR Operation

Use two short cuts.

1. When at least one of the two bytes is 0 or 255, the OR operation selects the larger byte (or one of them if equal).

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

Example 5.9

256 base OR Operation

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

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

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

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

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

OR

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

=142

select the larger term in each pair (or one of them if equal) and add them to get the result

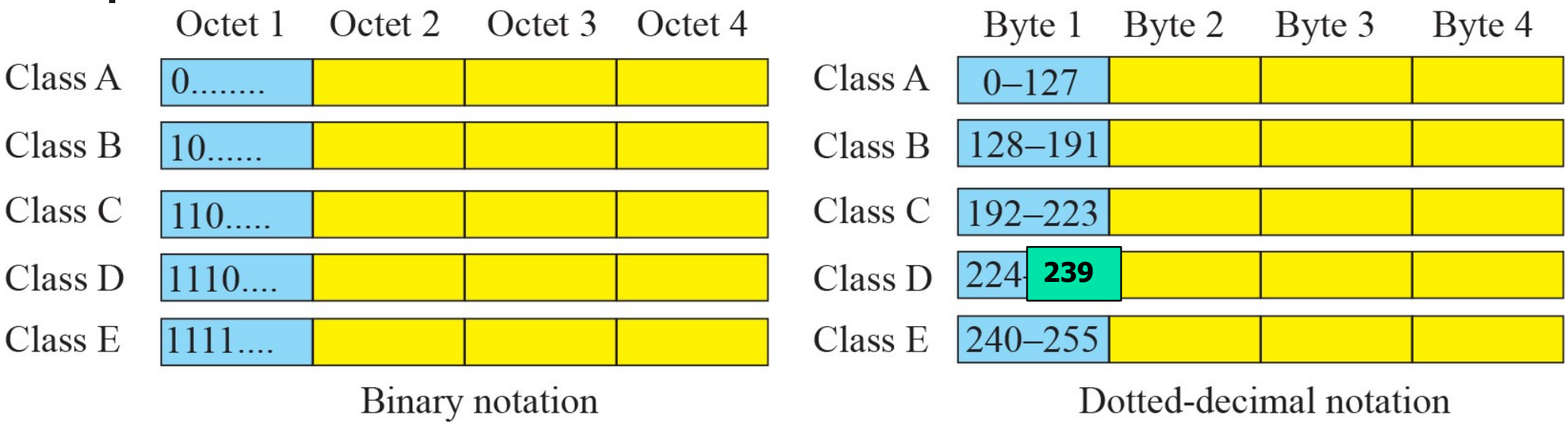
5-2 CLASSFUL ADDRESSING

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

Topics Discussed in the Section

- ✓ **Classes**
- ✓ **Classes and Blocks**
- ✓ **Two-Level Addressing**
- ✓ **Three-Level Addressing: Subnetting**
- ✓ **Supernetting**

Figure 5.6 *Finding the class of address*

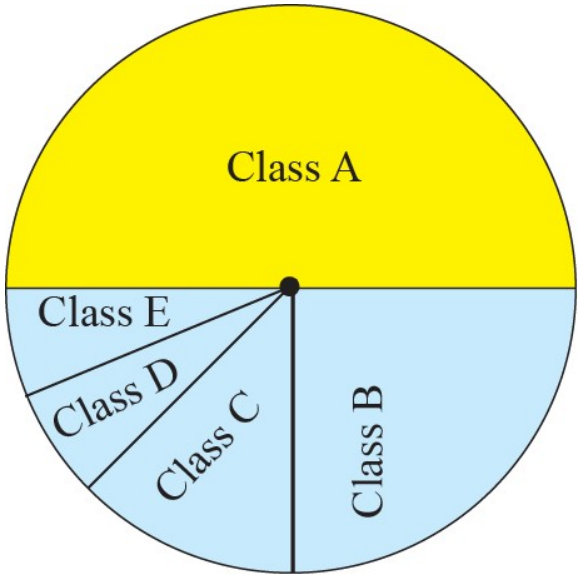


First Octet value range for Classes

Class	Staring Bits	Lower Bound	Upper Bound	Lower Bound	Upper Bound
A	0...	00000000	01111111	0	127
B	10...	10000000	10111111	128	191
C	110..	11000000	11011111	192	223
D	1110..	11100000	11101111	224	239
E	1111...	11110000	11111111	240	255

Figure 5.5 *Occupation of address space*

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



Class A: 2^{31} = 2,147,483,648 addresses, 50%

Class B: 2^{30} = 1,073,741,824 addresses, 25%

Class C: 2^{29} = 536,870,912 addresses, 12.5%

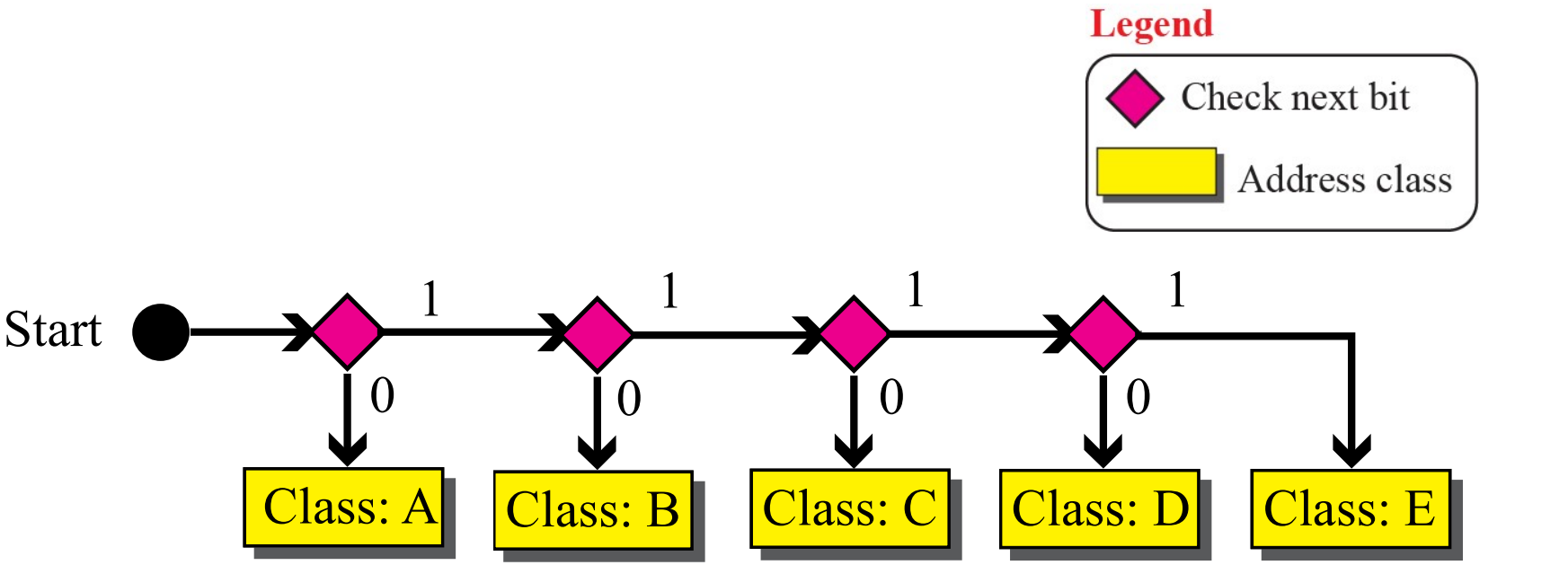
Class D: 2^{28} = 268,435,456 addresses, 6.25%

Class E: 2^{28} = 268,435,456 addresses, 6.25%

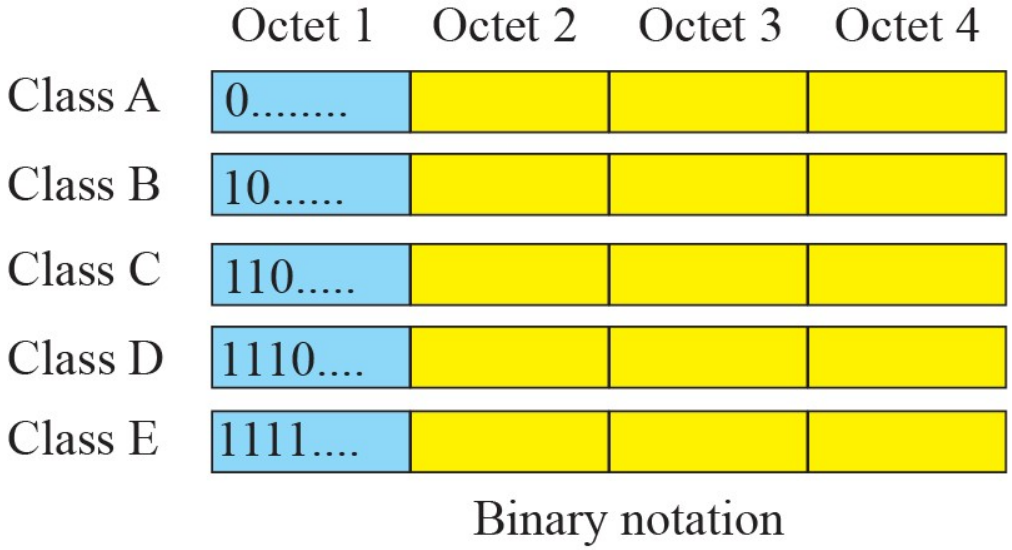
Total possible addresses in IPv4 =2

Class	Starting Bits fixed	Remaining Bits	Possible Number of Addresses	
A	0...	32-1=31	$2^{31}= 2^{32-1}=2^{32}/2^{1}$	half(50%) of 2^{32}
B	10...	32-2=30	$2^{30}= 2^{32-2}=2^{32}/2^{2}$	1/4th(25%) of 2^{32}
C	110..	32-3=29	$2^{29}= 2^{32-3}=2^{32}/2^{3}$	1/8th(12.5%) of 2^{32}
D	1110..	32-4=28	$2^{28}= 2^{32-4}=2^{32}/2^{4}$	1/16th(6.25%) of 2^{32}
E	1111...	32-4=28	$2^{31}= 2^{32-4}=2^{32}/2^{5}$	1/16th(6.25%) of 2^{32}

Figure 5.7 *Finding the class of an address using continuous checking*



Class: A Starts with 0
Class: B Starts with 10
Class: C Starts with 110
Class: D Starts with 1110
Class: E Starts with 1111



Example 5.10

Find the class of each address:

a. 00000001 00001011 00001011 11101111

b. 11000001 10000011 00011011 11111111

c. 10100111 11011011 10001011 01101111

d. 11110011 10011011 11111011 00001111

Solution

See the procedure in Figure 5.7.

a. The first bit is 0. This is a class A address.

b. The first 2 bits are 1; the third bit is 0. This is a class C address.

c. The first bit is 1; the second bit is 0. This is a class B address.

d. The first 4 bits are 1s. This is a class E address.

Example 5.11

Find the class of each address:

a. 227.12.14.87

b. 193.14.56.22

c. 14.23.120.8

d. 252.5.15.111

First Octet value range for Classes

Class	Staring Bits	Lower Bound	Upper Bound	Lower Bound	Upper Bound
A	0...	00000000	01111111	0	127
B	10...	10000000	10111111	128	191
C	110..	11000000	11011111	192	223
D	1110..	11100000	11101111	224	239
E	1111...	11110000	11111111	240	255

Solution

a. The first byte is 227 (between 224 and 239); the class is D.

b. The first byte is 193 (between 192 and 223); the class is C.

c. The first byte is 14 (between 0 and 127); the class is A.

d. The first byte is 252 (between 240 and 255); the class is E.

Figure 5.8 *Netid and hostid*

IP address in classes A, B, and C is divided into netid and hostid

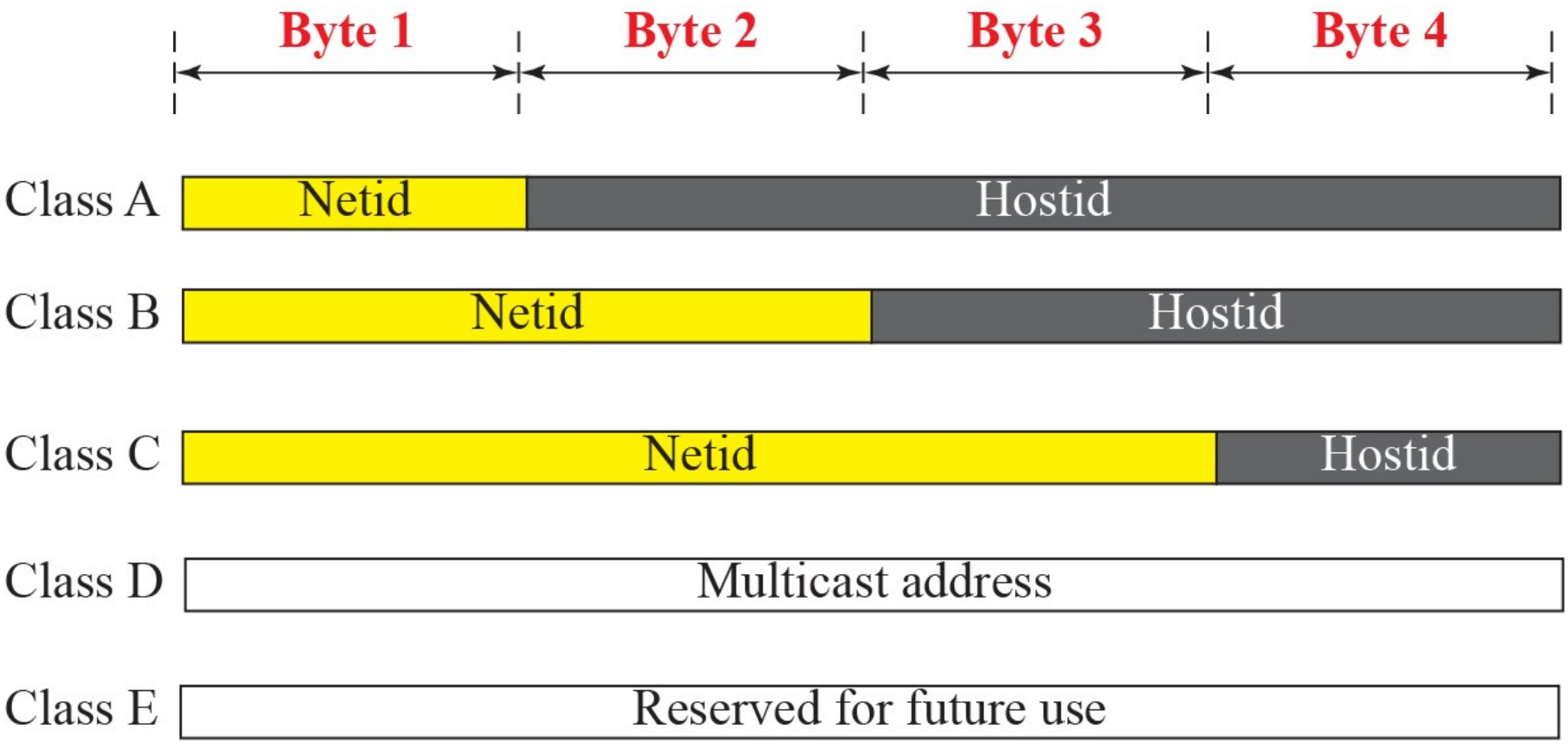
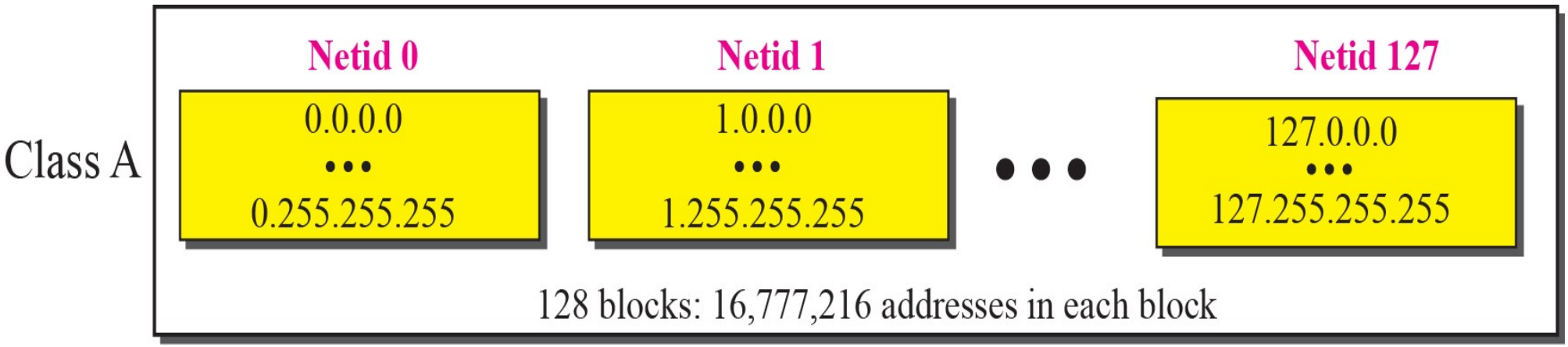


Figure 5.9 *Blocks in Class A*

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



Each of 128 blocks are containing large number of addresses.
Means every netid has large number of hostids (IPs for hosts) which is very large for any network. So many go waste.

Note

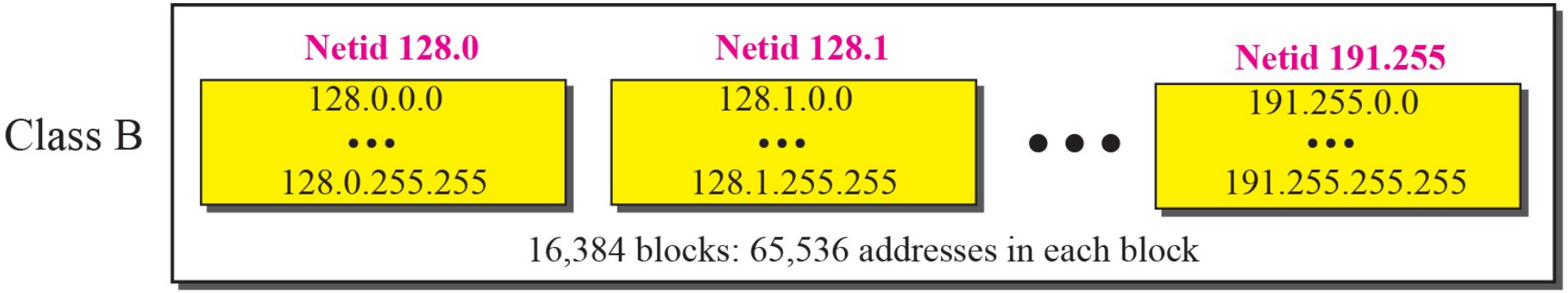
Millions of class A addresses are wasted.

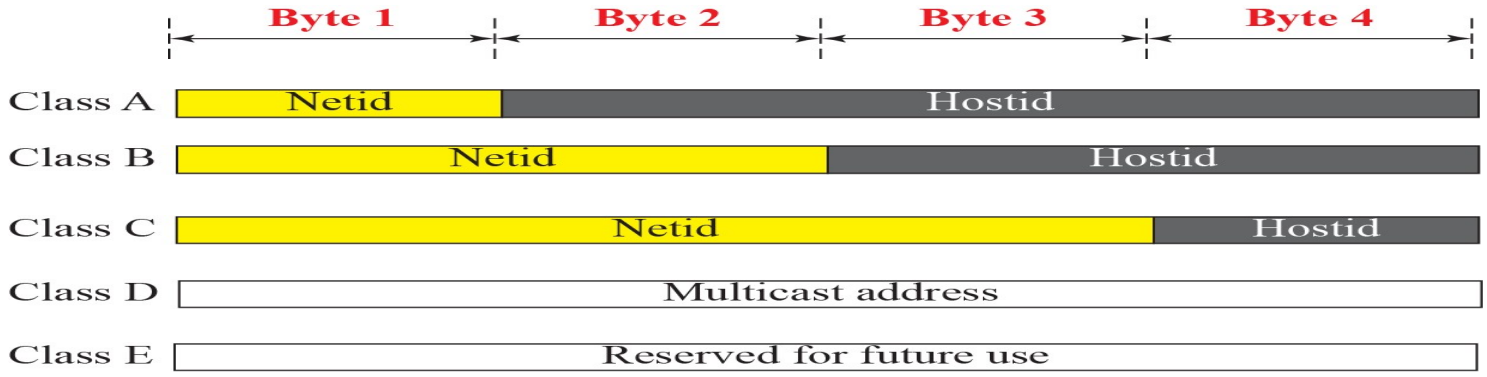
Figure 5.10 *Blocks in Class B*

1st Octet value range for Classes

Net ids's 2 bytes(16 bits),
Leftmost 2 bits fixed 10
16-2=14 bits for netids
2^{14}= 16384 networks

Class	Staring Bits	Lower Bound	Upper Bound	Lower Bound	Upper Bound
A	0...	00000000	01111111	0	127
B	10...	10000000	10111111	128	191
C	110..	11000000	11011111	192	223
D	1110..	11100000	11101111	224	239
E	1111...	11110000	11111111	240	255





Note

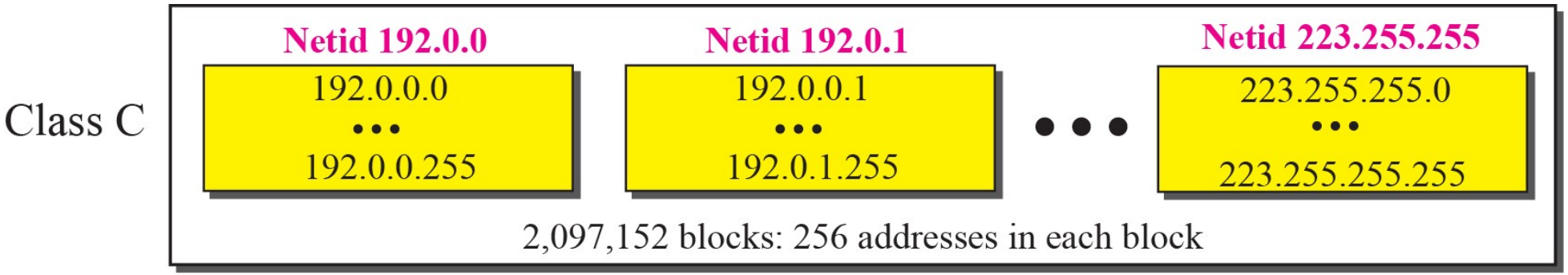
Many class B addresses are wasted.

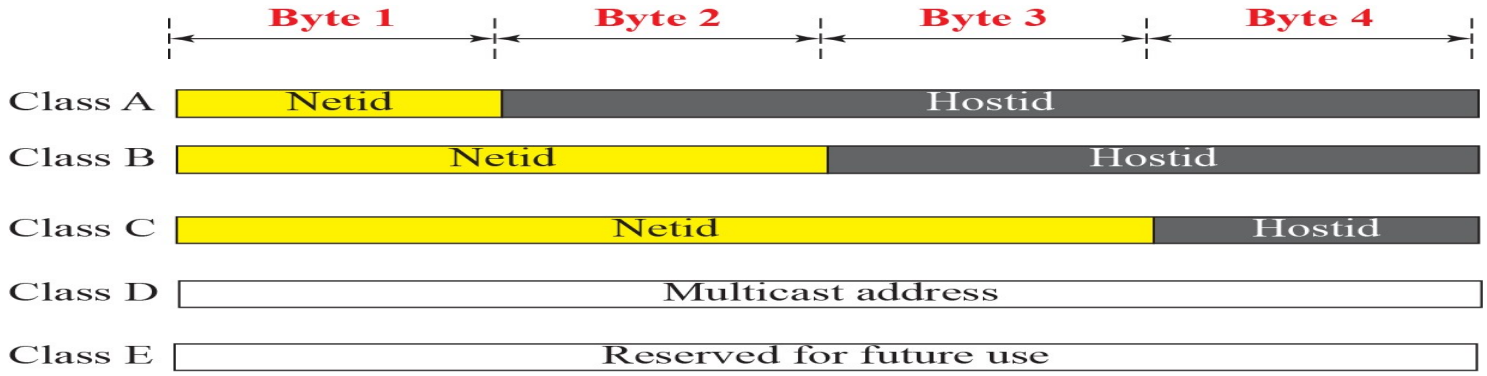
Figure 5.11 *Blocks in Class C*

1st Octet value range for Classes

Net ids's 3 bytes(24 bits),
Leftmost 3 bits fixed 110
24-3=21 bits for netids
2^{21}= 20,97,152 networks

Class	Staring Bits	Lower Bound	Upper Bound	Lower Bound	Upper Bound
A	0...	00000000	01111111	0	127
B	10...	10000000	10111111	128	191
C	110..	11000000	11011111	192	223
D	1110..	11100000	11101111	224	239
E	1111...	11110000	11111111	240	255





Note

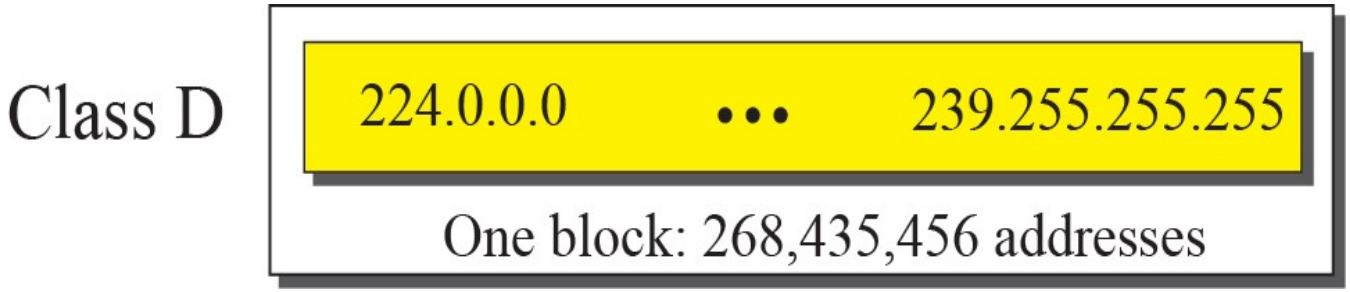
Not so many organizations are so small to have a class C block.

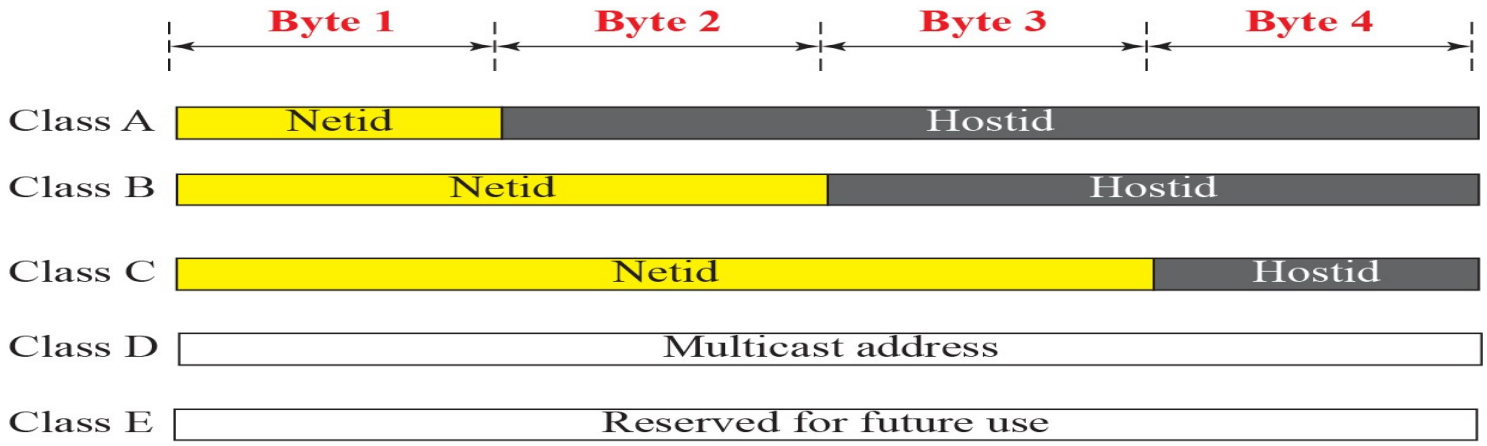
Figure 5.12 *The single block in Class D*

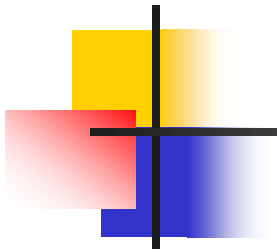
1st Octet value range for Classes

Class	Staring Bits	Lower Bound	Upper Bound	Lower Bound	Upper Bound
A	0...	00000000	01111111	0	127
B	10...	10000000	10111111	128	191
C	110..	11000000	11011111	192	223
D	1110..	11100000	11101111	224	239
E	1111...	11110000	11111111	240	255

One single block
Leftmost 4 bits fixed 1110
32-4=28 bits for addresses
2^{28}= 26,84,35,456 addresses







Note

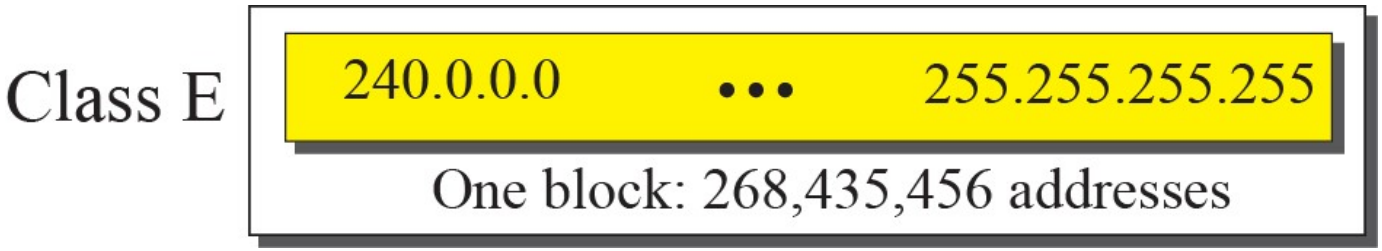
Class D addresses are made of one block, used for multicasting.

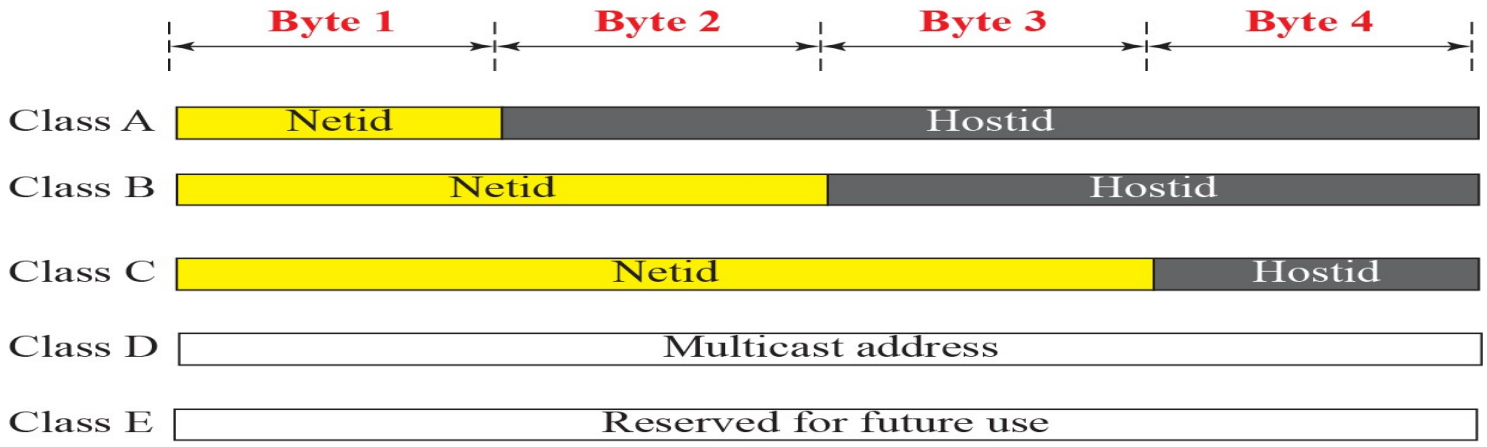
Figure 5.13 *The single block in Class E*

1st Octet value range for Classes

One single block
Leftmost 4 bits fixed 1111
32-4=28 bits for addresses
2^{28}= 26,84,35,456 addresses

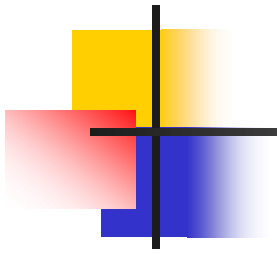
Class	Staring Bits	Lower Bound	Upper Bound	Lower Bound	Upper Bound
A	0...	00000000	01111111	0	127
B	10...	10000000	10111111	128	191
C	110..	11000000	11011111	192	223
D	1110..	11100000	11101111	224	239
E	1111...	11110000	11111111	240	255





Note

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



Note

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