CHAPTER 5

IPv4 Addresses

Exercises

1.

a. 2^8 = **256 addresses**

b. 2^{16} = **65536 addresses**

c. 2^{64} = **$1.846744737 \times 10^{19}$ addresses**

3. 3^{10} = **59,049 addresses**

5.

a. **0x72220208**

b. **0x810E0608**

c. **0xD022360C**

d. **0xEE220201**

7.

a. **(8 bits) / (4 bits per hex digits) = 2 hex digits**

b. **(16 bits) / (4 bits per hex digits) = 4 hex digits**

c. **(24 bits) / (4 bits per hex digits) = 6 hex digits**

9. We use Figure 5.6 (the right table) to find the class:

a. The first byte is 208 (between 192 and 223) → **Class C**

b. The first byte is 238 (between 224 and 299) → **Class D**

c. The first byte is 242 (between 240 and 255) → **Class E**

d. The first byte is 129 (between 000 and 127) → **Class A**

11.

a. Class is A → **netid: 114** and **hostid: 34.2.8**

b. Class is B → **netid: 132.56** and **hostid: 8.6**

c. Class is C → **netid: 208.34.54** and **hostid: 12**

d. Class is E → **The address is not divided into netid and hostid.**

13. We first change the number of addresses in the range (minus 1) to base 256

$$2{,}048 - 1 = (0.0.7.255)_{256}$$

We add this number to the first address to find the last address:

First Address:	122	.	12	.	7	.	0
Difference (base 256)	0	.	0	.	7	.	255
Last Address:	122	.	12	.	14	.	255

15.

a. We can apply the first short cut to all bytes here. The result is (**22.14.0.0**).

b. We can apply the first short cut to all bytes here. The result is (**12.0.0.0**).

c. We can apply the first short cut to bytes 1and 4; we need to apply the second short cut to bytes 2 and 3. The result is (**14.72.0.0**).

d. We can apply the first short cut to bytes 1 and 4; we need to apply the second short cut to bytes 2 and 3. The result is (**28.0.32.0**).

17. The first address can be found by ANDing the mask with the IP address as shown below:

IP Address:	25	.	34	.	12	.	56
Mask:	255	.	255	.	0	.	0
First Address	25	.	34	.	0	.	0

The last address can be found by either adding the number of addresses in the subnet $2^{32-n} = 2^{16}$ or by ORing the complement of the mask with the IP address (see the section on classless addressing because classful addressing is a special case of classless addressing) as shown below:

IP Address:	25	.	34	.	12	.	56
Mask Complement:	0	.	0	.	255	.	255
Last Address	25	.	34	.	255	.	255

19. The first address can be found by ANDing the mask with the IP address as shown below:

IP Address:	202	.	44	.	82	.	16
Mask:	255	.	255	.	255	.	192
First Address	202	.	44	.	82	.	0

Note that we use the first short cut on the first three bytes. We use the second short cut on the fourth bytes:

16	→	**0**	+	**0**	+	**0**	+	**16**	+	**0**	+	**0**	+	**0**	+	**0**
192	→	**128**	+	**64**	+	**32**	+	**0**	+	**0**	+	**0**	+	**0**	+	**0**
0	→	**0**	+	**0**	+	**0**	+	**0**	+	**0**	+	**0**	+	**0**	+	**0**

The last address can be found by ORing the complement of the mask with the IP address (see the section on classless addressing because classful addressing is a special case of classless addressing) as shown below:

IP Address:	**202**	.	**44**	.	**82**	.	**16**
Mask Complement:	**0**	.	**0**	.	**0**	.	**63**
Last Address	**202**	.	**44**	.	**82**	.	**127**

Note that we use the first short cut on the first three bytes. We use the second short cut on the fourth byte:

16	→	**0**	+	**64**	+	**32**	+	**16**	+	**0**	+	**0**	+	**0**	+	**0**
63	→	**0**	+	**0**	+	**32**	+	**16**	+	**8**	+	**4**	+	**2**	+	**1**
127	→	**0**	+	**64**	+	**32**	+	**16**	+	**8**	+	**4**	+	**2**	+	**1**

21. With the information given, the first address is found by ANDing the host address with the mask 255.255.0.0 (/16).

Host Address:	**25**	.	**34**	.	**12**	.	**56**
Mask:	**255**	.	**255**	.	**0**	.	**0**
Network Address (First):	**25**	.	**34**	.	**0**	.	**0**

The last address can be found by ORing the host address with the mask complement 0.0.255.255.

Host Address:	**25**	.	**34**	.	**12**	.	**56**
Mask Complement:	**0**	.	**0**	.	**255**	.	**255**
Last Address:	**25**	.	**34**	.	**255**	.	**255**

However, we need to mention that this is the largest possible block with 2^{16} addresses. We can have many small blocks as long as the number of addresses divides this number.

23.

See below. The number of created subnets are equal to or greater than required.

a. $\log_2 2 = 1$ Number of 1's = **1** Number of created subnets: **2**
b. $\log_2 62 = 5.95$ Number of 1's = **6** Number of created subnets: **64**

c. $\log_2 122 = 6.93$ Number of 1's = **7** Number of created subnets: **128**
d. $\log_2 250 = 7.96$ Number of 1's = **8** Number of created subnets: **256**

25.

a. $\log_2 1024 = 10$ Extra 1s = **10** Possible subnets: **1024** Mask: **/26**

b. $2^{32-26} = 64$ addresses in each subnet

c.

First subnet:
The first address is the beginning address of the block.

first address in subnet 1:	**130**	.	**56**	.	**0**	.	**0**

To find the last address, we need to write 63 (one less than the number of addresses in each subnet) in base 256 (0.0.0.63) and add it to the first address (in base 256).

first address in subnet 1:	**130**	.	**56**	.	**0**	.	**0**
number of addresses:	**0**	.	**0**	.	**0**	.	**63**
last address in subnet 1:	**130**	.	**56**	.	**0**	.	**63**

d.

Last subnet (Subnet 1024):
To find the first address in subnet 1024, we need to add 65,472 (1023×64) in base 256 (0.0.255.92) to the first address in subnet 1.

first address in subnet 1	**130**	.	**56**	.	**0**	.	**0**
number of addresses:	**0**	.	**0**	.	**255**	.	**192**
first address in subnet 1024:	**130**	.	**56**	.	**255**	.	**192**

Now we can calculate the last address in subnet 1024 as we did for the first address.

first address in subnet 500:	**130**	.	**56**	.	**255**	.	**192**
number of addresses:	**0**	.	**0**	.	**0**	.	**63**
last address in subnet 500:	**130**	.	**56**	.	**255**	.	**255**

27. We first change the mask to binary to find the number of 1's:

a. `11111111 11111111 11111111 00000000` → **/24**
b. `11111111 00000000 00000000 00000000` → **/8**
c. `11111111 11111111 11100000 00000000` → **/19**
d. `11111111 11111111 11110000 00000000` → **/20**

29. If the first and the last addresses are known, the block is fully defined. We can first find the number of addresses in the block. We can then use the relation

$$N = 2^{32} - n \quad \rightarrow \quad n = 32 - \log_2 N$$

to find the prefix length. For example, if the first address is **17.24.12.64** and the last address is **17.24.12.127**, then the number of addresses in the block is 64. We can find the prefix length as

$$n = 32 - \log_2 N = 32 - \log_2 64 = \mathbf{26}$$

The block is then **72.24.12.64/26**.

31. Many blocks can have the same prefix length. The prefix length only determines the number of addresses in the block, not the block itself. Two blocks can have the same prefix length but start in two different point in the address space. For example, the following two blocks

127.15.12.32/27	**174.18.19.64/27**

have the same prefix length, but they are definitely two different blocks. The length of the blocks are the same, but the blocks are different.

33.

Group 1

For this group, each customer needs 128 addresses. This means the suffix length is $\log_2 128 = 7$). The prefix length is then $32 - 7 = \mathbf{25}$. The range of addresses are given for the first, second, and the last customer. The range of addresses for other customers can be easily found:

1st customer:	**150.80.0.0/25**	**to**	**150.80.0.127/25**
2nd customer:	**150.80.0.128/25**	**to**	**150.80.0.255/25**
...	…		…
200th customer:	**150.80.99.128/25**	**to**	**150.80.99.255/25**

Total addresses for group 1 = $200 \times 128 = 25{,}600$ addresses

Group 2

For this group, each customer needs 16 addresses. This means the suffix length is $\log_2 16 = 4$. The prefix length is then $32 - 4 = \mathbf{28}$. The addresses are:

1st customer:	**150.80.100.0/28**	**to**	**150.80.100.15/28**
2nd customer:	**150.80.100.16/28**	**to**	**150.80.100.31/28**
...	…		…
400th customer:	**150.80.124.240/28**	**to**	**150.80.124.255/28**

Total addresses for group 2 = $400 \times 16 = 6400$ addresses

Group 3

For this group, each customer needs 4 addresses. This means the suffix length is $\log_2 4 = 2$. The prefix length is then $32 - 2 = \mathbf{30}$. The addresses are:

1st customer:	**150.80.125.0/30**	**to**	**150.80.125.3/30**
2nd customer:	**150.80.125.4/30**	**to**	**150.80.100.7/30**
...	…		…
64th customer:	**150.80.125.252/30**	**to**	**150.80.125.255/30**
65th customer:	**150.80.126.0/30**	**to**	**150.80.126.3/30**
...	…		…
2048th customer:	**150.80.156.252/30**	**to**	**150.80.156.255/30**

Total addresses for group 3 = $2048 \times 4 = 8192$ addresses

Number of allocated addresses: **40,192**

Number of available addresses: **25,344**

35. There are actually two choices. If the ISP wants to use subnetting (a router with 32 output ports) then the prefix length for each customer is $n_{sub} = 32$. However, there is no need for a router and subnetting. Each customer can be directly connected to the ISP server. In this case, the whole set of the customer can be taught of addresses in one single block with the prefix length *n* (the prefix length assigned to the ISP).