CHAPTER 18

PRACTICE SET

Questions

Q18-1. Forwarding is delivery to the next node. A router uses its forwarding table to send a packet out of one of its interfaces and to make it to reach to the next node. In other words, forwarding is the decision a router makes to send a packet out of one of its interfaces. Routing, on the other hand, is an end-to-end delivery resulting in a path from the source to the destination for each packet. This means a routing process is a series of forwarding processes. To enable each router to perform its forwarding duty, routing protocols need to be running all of the time to provide updated information for forwarding tables. Although forwarding is something we can see in the foreground, in the background, routing provides help to the routers to do forwarding.

Q18-2. Yes, the prefix length defines the size of the block, but the blocks can belong to different locations in the address space.

Q18-3. The transport layer communication is between two ports; the network layer communication is between two hosts. This means that each layer has a different source/destination address pair; each layer needs a different header to accommodate these pair of addresses. In addition, there are other pieces of information that need to be separately added to the corresponding header.

Q18-4. Routing cannot be done at the transport layer, because the communication at the transport layer is one single logical path between the source port and the destination port. Routing cannot be done at the data-link layer because the communication at the data-link layer is between two nodes (one single path); there is no need for routing. On the other hand, there are several possible paths for a packet between the source host and destination host at the network layer. Routing is the job of selecting one of these paths for the packet.

Q18-5. The throughput is the smallest transmission rate, or 140 Kbps. The bottleneck is now the link between the source host and R1.

Q18-6. Yes. We can find the prefix length.

a. We first find the size of the block as shown below:

N = last address − first address + 1

b. We then find the prefix length.

$n = 32 - \log_2 N$

Q18-7. None of these services are implemented for the IP protocol in order to make it simple.

Q18-8. Four types of delays are transmission delay, propagation delay, processing delay, and queuing delay.

Q18-9. The number of virtual circuits is $2^8 = 256$.

Q18-10. The three phases are setup phase, data transfer, and teardown phase.

Q18-11. Yes. We can find the prefix length using only the block size. The prefix length is directly related to the block size as shown below:

$n = 32 - \log_2 N$

Q18-12.

a. In the datagram approach, the forwarding decision is made based on the destination address in the packet header.

b. In the virtual-circuit approach, the forwarding decision is based on the label in the packet header.

Problems

P18-1.

a. The number of addresses in the ISP block is $N = 2^{32-20} = 4096$. We can add 4095 (which is $N - 1$) to the first address to find the last one (note that the addition can be done in base 256, as described in Appendix B. In base 256, 4095 is (15.255). We have

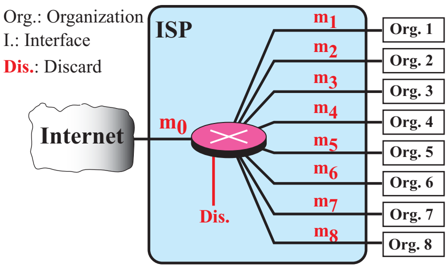
First address: 16.12.64.0**/20** Last address: 16.12.79.255**/20**

The prefix length for each organization is $n_i = 32 - \log_2 256 = 24$. We assume that the addresses are allocated from the beginning of the ISP block with each organization consuming 256 addresses. The following shows how addresses are allocated. Note that the prefix for each block is 24 bits.

Block	First address		Last address	*n*
1	16.12.64.0/**24**	→	16.12.64.255/**24**	**24**
2	16.12.65.0/**24**	→	16.12.65.255/**24**	**24**
3	16.12.66.0/**24**	→	16.12.66.255/**24**	**24**
4	16.12.67.0/**24**	→	16.12.67.255/**24**	**24**
5	16.12.68.0/**24**	→	16.12.68.255/**24**	**24**
6	16.12.69.0/**24**	→	16.12.69.255/**24**	**24**
7	16.12.70.0/**24**	→	16.12.70.255/**24**	**24**
8	16.12.71.0/**24**	→	16.12.71.255/**24**	**24**
Unassigned	16.12.72.0/**21**	→	16.12.79.255/**21**	**21**

The unallocated addresses, which can be reserved for the future use of the ISP, are 16.12.72.0/**21** to 16.12.79.255/**21**, for a total of 2048 addresses.

b. The simplified outline is given below. Note that packets having destination addresses with the last prefix in the figure are discarded until these addresses are assigned.



Forwarding table

Prefix	Interface
Default	m_0
0001000 00001100 01000000	m_1
0001000 00001100 01000001	m_2
0001000 00001100 01000010	m_3
0001000 00001100 01000011	m_4
0001000 00001100 01000100	m_5
0001000 00001100 01000101	m_6
0001000 00001100 01000110	m_7
0001000 00001100 01000111	m_8
0001000 00001100 01001	**Dis.**

P18-2. The total number of addresses is $2^8 = 256$. This means we have 64 addresses for each network. We can divide the whole address space into four blocks (block 0 to block3), each of 64 addresses. The addresses in each block are allocated as (0 to 63), (64 to 127), (128 to 191), and (192 to 255). It can be checked that each block is allocated according to the two restrictions needed for the proper operation of CIDR. First, the number of addresses in each block is a power of 2. Second, the first address in each block is divisible by the number of addresses in the block, as shown below:

Block 0: $0/64 = 0$ **Block 1**: $64/64 = 1$

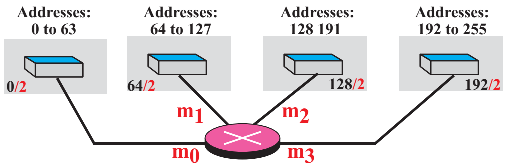
Block 2: $128/64 = 2$ **Block 3**: $192/64 = 3$

The prefix length for each group is $n_i = 8 - \log_2 64 = 2$. We can then write the ranges in binary to find the prefix for each block.

Block	Range	Range in binary	*n*	Prefix
0	0 to 63	**00**000000 to **00**111111	**2**	**00**

1	64 to 127	`01000000` to `01111111`	2	01
2	128 to 191	`10000000` to `10111111`	2	10
3	192 to 255	`11000000` to `11111111`	2	11

The following shows the outline and the forwarding table. Note that each interface can use one of the addresses in the corresponding block.



Forwarding table

Prefix	Interface
00	m_0
01	m_1
10	m_2
11	m_3

P18-3. We can first write the prefix in binary and then change each 8-bit chunk to 3 decimal:

a. `00000000 00000000 00000000 00000000` **mask:** 0.0.0.0

b. `11111111 11111100 00000000 00000000` **mask:** 255.252.0.0

c. `11111111 11111111 11111111 11111100` **mask:** 255.252.255.252

P18-4. The administration can use DHCP to dynamically assign addresses when a host needs to access the Internet. This is possible because, in most organizations, not all of the hosts need to access the Internet at the same time.

P18-5. We can use the formula $N = 2^{32-n}$

a. $N = 2^{32-0} = 4{,}294{,}967{,}296$

b. $N = 2^{32-14} = 262{,}144$

c. $N = 2^{32-32} = 1$

P18-6. The size of the block can be found as $n = 32 - \log_2 N$

a. $n = 32 - \log_2 1 = 32$

b. $n = 32 - \log_2 1024 = 22$

c. $n = 32 - \log_2 2^{32} = 0$

P18-7. Both NAT and DHCP can be used for this purpose. DHCP dynamically assigns one of the assigned addresses when a host needs to access the Internet; NAT permanently assigns a set of private addresses to the host, but maps the private address to the global address when a host needs to use the Internet.

P18-8. The class can be defined by looking at the first byte:

a. Since the first byte is between 128 and 191, the class is B.

b. Since the first byte is between 192 and 223, the class is C.

c. Since the first byte is between 240 and 255, the class is E.

P18-9. The total number of addresses in the organization is $N = 2^{32-16} = 65{,}536$.

a. Each subnet can have $N_{sub} = 65{,}536 / 1024 = 64$ addresses.

b. The subnet prefix for each subnet is $n_{sub} = 32 - \log_2 N_{sub} = 32 - 6 = 26$.

c. Now we can calculate the first and the last address in the first subnet. The first address is the beginning address of the block; the last address is the first address plus 63.

First address: 130.56.0.0/**26** Last address: 130.56.0.63/**26**

d. To find the first address in subnet 1024, we need to add 65,472 (1023 × 64) in base 256 (0.0.255.192) to the first address in subnet 1. The last address can then be found by adding 63 to the first.

First address: 130. 56.255.192/**26** Last address: 130. 56.255.255/**26**

P18-10. The whole block can be represented as 0.0.0.0/0. The first address in the class is 0.0.0.0. The prefix is 0 because no bits define the block; all bits define the address itself. Another test to prove that the prefix is 0 is that the number of addresses in the block can be found as 2^{32-n}. The value of n should be zero to make the number of addresses $N = 2^{32}$.

P18-11. The class can be defined by checking the first few bits. We need to stop checking if we find a 0 bit or four bits have already been checked.

a. Since the first bit is 0, the Class is A.

b. Since the first four bits are 1110, the class is D.

c. Since the first three bits are 110, the class is C.

P18-12. Router R1 has four interfaces. Let us investigate the possibility of a packet with destination 140.24.7.194 from each of these interfaces and see how it is routed.

a. The packet can arrive from one of the interfaces m0, m1, and m2, because one of the computers in organization 1, 2, or 3 could have sent this packet. The prefix /26 is applied to the address, resulting in the network address 140.24.7.192/26. Since none of the network addresses/masks matches this result, the packet is sent to the default router R2.

b. The packet cannot arrive at router R1 from interface m3 because this means that the packet must have arrived from interface m0 of router R2, which is impossible because if this packet arrives at router R2 (from any interface), the prefix length /26 is applied to the destination address, resulting in the network address/mask of 140.24.7.192/26. The packet is sent out from interface m1 and directed to organization 4 and never reaches router R1.

P18-13. We first write each potential mask in binary notation and then check if it has a contiguous number of 1s from the left followed by 0s.

a. `11111111 11100001 00000000 00000000` Not a mask

b. `11111111 11000000 00000000 00000000` A mask

c. `11111111 11111111 11111111 00000110` Not a mask

P18-14.

a. The number of addresses in the ISP block is $N = 2^{32-21} = 2048$. We can add 2047 (which is $N - 1$) to the first address to find the last one (note that the addition can be done in base 256, as described in Appendix B. In base 256, 2047 is (7.255). We have

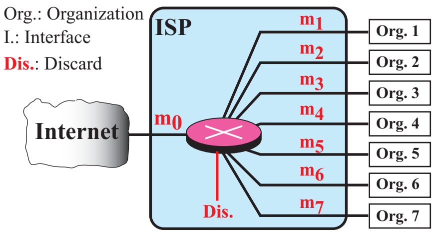
First address: 80.70.56.0/**21** Last address: 80.70.63.255/**21**

b. To make the number of addresses in each block a power of 2 (first CIDR restriction), we assign the following ranges to each organization. The prefix length for each organization is $n_i = 32 - \log_2 N_i$ where N_i is the number of addresses assigned to that organization. Note that the unused addresses cannot fit in a single block (second CIDR restriction).

Block	Size	First address		Last address	n
1	512	80.70.56.0/**23**	→	80.70.57.255/**23**	23
2	512	80.70.58.0/**23**	→	80.70.59.255/**23**	23
3	256	80.70.60.0/**24**	→	80.70.60.255/**24**	24
4	256	80.70.61.0/**24**	→	80.70.61.255/**24**	24
5	64	80.70.62.0/**26**	→	80.70.62.63/**26**	26
6	64	80.70.62.64/**26**	→	80.70.62.127/**26**	26
7	64	80.70.62.128/**26**	→	80.70.62.191/**24**	26
Unused	320	80.70.62.192	→	80.70.63.255	

c. The simplified outline is given below. Note that to make the forwarding table operable, we need to divide the unused addresses into two blocks.

Packets with destination addresses matching the last two prefixes are discarded by the router.



Forwarding table

Prefix	I.
Default	m_0
01010000 01000110 0011100	m_1
01010000 01000110 0011101	m_2
01010000 01000110 00111100	m_3
01010000 01000110 00111101	m_4
01010000 01000110 00111110 00	m_5
01010000 01000110 00111110 01	m_6
01010000 01000110 00111110 10	m_7
01010000 01000110 00111110 11 01010000 01000110 00111111	**Dis.**

P18-15. The size of the address in each case is the base to the power of the number of digits:

a. The size of the address space is $2^{16} = 65{,}536$.

b. The size of the address space is $16^6 = 16{,}777{,}216$.

c. The size of the address space is $8^4 = 4096$.

P18-16. The total number of addresses is $2^9 = 512$. We need, however, to check whether address allocation is done according to the restrictions for CIDR's proper operation. The address allocations to the networks are (N_0: 0 to 63), (N_1: 64 to 255), and (N_2: 256 to 511). Each range is a power of 2, which means that the first restriction is fulfilled. The second restriction (the first address in the block should divide the number of addresses in the block) is fulfilled for N_0 and N_2, but not for N_1:

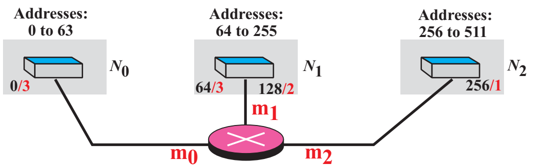
N_0: 0 / 64 = 0	N_1: 64 / 256 = **0.25**	N_2: 256 / 256 = 1

One solution would be to think of the addresses in N_1 as the aggregation of two contiguous blocks (64 to 127) and (128 to 256) connected to the same interface. We call these blocks 1-1 and 1-2. The prefixes for blocks are

$$n_0 = 9 - \log_2 64 = 3 \qquad n_{1\text{-}1} = 9 - \log_2 64 = 3$$
$$n_{1\text{-}2} = 9 - \log_2 128 = 2 \qquad n_2 = 9 - \log_2 256 = 1$$

Block	Range	Range in binary	*n*	Prefix
0	0 to 63	000000000 → 000111111	3	000
1-1	64 to 127	001000000 → 001111111	3	001
1-2	128 to 255	010000000 → 011111111	2	01
2	256 to 511	100000000 → 111111111	1	1

Based on the above table, we can show the outline of the internet and addresses and the forwarding table, as shown below. Note that the address aggregation in *N*1 is transparent to the user as long as the router forwards the packet according to its forwarding table. If we need to be fair, we should say that *N*1 actually has two network addresses because it is made of two blocks. The administration can easily divide the block into two subblocks with a router.



Forwarding table

Prefix	Interface
000	m_0
001	m_1
01	m_1
1	m_2

P18-17. We write the address in binary and then keep only the leftmost *n* bits.

a. `10101010 00101000 00001011`

`b. 01101110 00101000 111100`

`c. 01000110 00001110 00`

P18-18. We first write the mask in binary notation and then count the number of left-most 1s.

a. `11111111 11100000 00000000 00000000` *n*: **11**

b. `11111111 11110000 00000000 00000000` *n*: **12**

c. `11111111 11111111 11111111 10000000` *n*: **25**

P18-19. We change each 8-bit section to the corresponding decimal value and insert dots between the bytes.

a. 94.176.117.21

b. 137.142.208.49

c. 87.132.55.15

P18-20. We can write the address in binary. Set the last 32 – *n* bits to 0s to get the first address; set the last 32 – *n* bits to 1s to get the last address. You can use one of the applets at the book website to check the result.

a.

Given:	00001110	00001100	01001000	00001000	14.12.72.8/24
First:	00001110	00001100	01001000	00000000	14.12.72.0/24
Last:	00001110	00001100	01001000	11111111	14.12.72.255/24

b.

Given:	11001000	01101011	00010000	00010001	200.107.16.17/18
First:	11001000	01101011	00000000	00000000	200.107.0.0/18
Last:	11001000	01101011	00111111	11111111	200.107.63.255/18

c.

Given:	01000110	01101110	00010011	00010001	200.107.16.17/18
First:	01000110	01101110	00000000	00000000	200.107.0.0/18
Last:	01000110	01101110	11111111	11111111	200.107.63.255/18

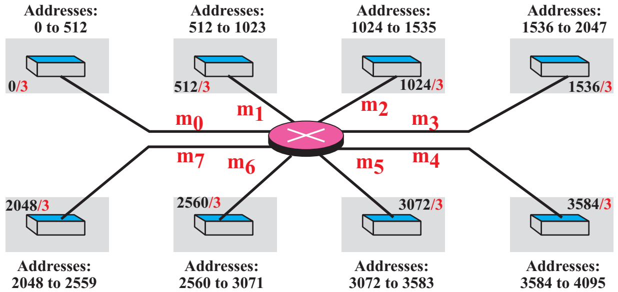
P18-21. The total number of addresses is 2^{12} = 4096. This means that there are 512 addresses for each network. We can divide the whole address space into eight blocks (block 0 to block7), each of 512 addresses. The addresses in each block are allocated as (0 to 511), (512 to 1023), (1024 to 1535), (1536 to 2047), …, (3584 to 4095). It can be checked that each block is allocated according to the two restrictions needed for the proper operation of CIDR. First, the number of addresses in each block is a power of 2. Second, the first address is divisible by the number of addresses as shown below:

Block 0: 0 / 512 = 0 **Block 1**: 512 / 512 = 1 **Block 2**: 1024 / 512 = 2 …

The prefix length for each group is $n_i = 12 - \log_2 512 = 3$. We can then write the ranges in binary to find the prefix for each block.

Block	Range	Range in binary	***n***	Prefix
0	0 to 511	000000000000 to 000111111111	3	000
1	512 to 1023	001000000000 to 001111111111	3	001
2	1024 to 1535	010000000000 to 010111111111	3	010
3	1536 to 2047	011000000000 to 011111111111	3	011
4	2048 to 2559	100000000000 to 100111111111	3	100
5	2560 to 3071	101000000000 to 101111111111	3	101
6	3072 to 3583	110000000000 to 110111111111	3	110
7	3584 to 4095	111000000000 to 111110000000	3	111

The next figure shows the outline and the forwarding table. Note that each interface can use one of the addresses in the corresponding block. The addresses are written in decimal (not dotted-decimal) because of the address space size.



Forwarding table

Prefix	Interface
000	m_0
001	m_1
010	m_2
011	m_3
000	m_4
000	m_5
000	m_6
000	m_7

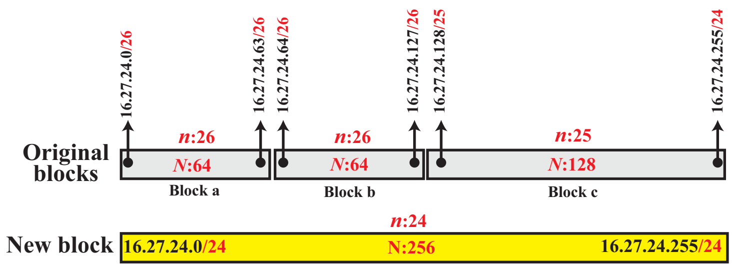
P18-22. We change each byte to the corresponding binary representation:

```
a. 01101110 00001011 00000101 01011000
b. 00001100 01001010 00010000 00010010
c. 11001001 00011000 00101100 00100000
```

P18-23. One way to do this is to first find the size of each block. We can then add the size to the first address in the block to find the last address. Next, we can put the blocks together to find whether they can be combined into a larger block.

Block	Size	First address		Last address
a	$N = 2^{32-26} = 64$	16.27.24.0/26	→	16.27.24.63/26
b	$N = 2^{32-26} = 64$	16.27.24.64/26	→	16.27.24.127/26
c	$N = 2^{32-25} = 128$	16.27.24.128/25	→	16.27.24.255/26

Since the blocks are contiguous, we can combine the three blocks into a larger one. The new block has 256 addresses and $n = 32 - \log_2 256 = 24$.



P18-24. The organization is granted $2^{32-21} = 2^{11} = 2048$ addresses. The medium-size company has $2^{32-22} = 2^{10} = 1024$ addresses. Each small organization has $2^{32-23} = 2^9 = 512$ addresses. We can plot the range of addresses for each organization as shown below:

Large organization:		12.44.184.0/**21**	to	12.44.191.255/**21**
Medium organization:		12.44.184.0/**22**	to	12.44.187.255/**22**
Small organization 1:		12.44.188.0/**23**	to	12.44.189.255/**23**
Small organization 2:		12.44.190.0/**23**	to	12.44.191.255/**23**

The company install a router whose forwarding table is based on the longest-prefix match first principle as shown below.

Network address /mask	Next hop	Interface
`00001100 00101100 1011110`	…	Small organization 1
`00001100 00101100 1011111`	…	Small organization 2
`00001100 00101100 101110`	…	Medium organization

Let us use three cases to show that the packets are forwarded correctly.

a. Assume a packet with the destination address 12.44.185.110 is arrived. The router first extracts the first 23 bits (00001100 00101100 1011100) and check to see if it matches with the first row of the table, which does not. It then checks with the second row, which does not match either. The router next extracts the first 22 bits (00001100 00101100 101110), which matches with the last entry. The packet is correctly forwarded to the interface of the medium organization.

b. Assume a packet with the destination address 12.44.190.25 is arrived. The router first extracts the first 23 bits (00001100 00101100 1011111) and check to see if it matches with the first row of the table, which does not. It then checks with the second row, which does. The packet is correctly forwarded to the interface of second small organization.

c. Assume a packet with the destination address 12.44.189.24 is arrived. The router first extracts the first 23 bits (00001100 00101100 1011110) and check to see if it matches with the first row of the table, which does The packet is correctly forwarded to the interface of first small organization.

P18-25. The packet is sent to router R1 and eventually to organization 1 as shown below:

a. Router R2 applies the mask /26 to the address (or it extracts the leftmost 26 bits) resulting in the network address/mask of 140.24.7.0/26, which does not match with the first entry in the forwarding table.

b. Router R2 applies the mask /24 to the address (or it extracts the leftmost 24 bits) resulting in the network address/mask of 140.24.7.0/24, which matches with the second entry in the forwarding table. The packet is sent out from interface m0 to router R1.

c. Router R1 applies the mask /26 to the address (or it extracts the leftmost 26 bits) resulting in the network address/mask of 140.24.7.0/26, which matches with the first entry in the forwarding table. The packet is sent out from interface m0 to organization 1.