HW5: IP Addresses, Prefixes, and Routes - Solutions

COM-208: Computer Networks

Before we start

A few basic rules:

- An IP prefix A/M is the range of IP addresses whose M most significant bits are the same as M's L most significant addresses. E.g., 10.0.0.16 belongs to IP prefix 10.0.0.0/24, because the 24 most significant bits of 10.0.0.16 are the same as the 24 most significant bits of 10.0.0.0.
- This implies that a /M IP prefix contains 2^{32-M} IP addresses. E.g., a /24 IP prefix contains $2^{32-24} = 256$ IP addresses. In other words, we don't have the freedom to create an IP prefixes that contains an arbitrary number of IP addresses, it must always contain a number that is a power of 2.
- Each IP subnet must have its own IP prefix. Hence, IP prefixes allocated to different IP subnets must not overlap.

When assigning IP addresses to network interfaces in an IP subnet, assume that the following IP addresses cannot be assigned to any network interface:

- The first IP address in the subnet's IP prefix (called "network address"). E.g., the first IP address in 10.0.0.0/24 is 10.0.0.0. This address is sometimes reserved for special uses, e.g., a discovery service provided by the subnet.
- The last IP address in the subnet's prefix (called "broadcast address"). E.g., the last IP address in 10.0.0.0/24 is 10.0.0.255. This address is sometimes reserved to be used as the subnet's broadcast address, i.e., as the destination IP address for packets that should be received by all end-systems in a subnet.

Getting familiar with forwarding tables

Basic

For the sake of this exercise only, imagine a world where IP addresses are 8 (not 32) bits long. Suppose a router uses longest-prefix matching and has the following forwarding table:

Dst IP prefix	Output link
00**	0
01**	1
100*	2
otherwise	3

For each of the 4 rows of this forwarding table, identify the range of IP addresses that would match the row. How many IP addresses does each row match?

Destination Address Range	Link Interface
00000000 - 00111111	0
01000000 - 01111111	1
10000000 - 10011111	2
10100000 - 11111111	3

number of addresses for interface 0: $2^6 = 64$
number of addresses for interface 1: $2^6 = 64$
number of addresses for interface 2: $2^5 = 32$
number of addresses for interface 3: $2^5 + 2^6 = 96$

IP prefix allocation

Basic

IP subnets A, B and C contain 10, 5, and 3 network interfaces, respectively. Allocate an IP prefix to each subnet, and assign an IP address to each network interface, from IP prefix 1.2.3.0/27.

Consider three cases (a, b, and c) for allocating prefixes to subnets. In each case, follow the given order:

(a) A, B, C (i.e., largest to smallest)

(b) C, B, A (i.e., smallest to largest)

(c) B, A, C

In the context of this exercise, when we say that allocation "follows a given order," we mean that, if Subnet X comes before Subnet Y in that order, the IP addresses for Subnet X should be arithmetically smaller than the IP addresses for Subnet Y (in the sense that IP address 1.2.3.4 is arithmetically smaller than IP address 1.2.3.5).

Note: Allocating addresses might be infeasible in some cases.

(a) Refer to the answer for (b) for a step-by-step solution (both scenarios follow the same logic).

We can allocate the following prefixes for subnets A, B and C.

A: 1.2.3.0/28
B: 1.2.3.16/29
C: 1.2.3.24/29

(b) Network prefix 1.2.3.0/27 contains $2^{32-27} = 32$ addresses. We need to pick three subnets from the address space 1.2.3.0 - 1.2.3.31, while satisfying the requirements from above.

Subnet C must have at least 5 IP addresses (3 for end-systems, 1 for network address, and 1 for broadcast address). However, since we need to round to to a power of 2 we will allocate 8 addresses: from 1.2.3.0 to 1.2.3.7. Let's represent them in binary format:

0000 0001.0000 0010.0000 0011.0000 0000 = 1.2.3.0

0000 0001.0000 0010.0000 0011.0000 0001 = 1.2.3.1

0000 0001.0000 0010.0000 0011.0000 0010 = 1.2.3.2

0000 0001.0000 0010.0000 0011.0000 0011 = 1.2.3.3

0000 0001.0000 0010.0000 0011.0000 0100 = 1.2.3.4

0000 0001.0000 0010.0000 0011.0000 0101 = 1.2.3.5

0000 0001.0000 0010.0000 0011.0000 0110 = 1.2.3.6

0000 0001.0000 0010.0000 0011.0000 0111 = 1.2.3.7

The prefix is 0000 0001.0000 0010.0000 0011.0000 0 = 1.2.3.0 with length 29, thus Subnet C is 1.2.3.0/29, or 1.2.3.0 - 1.2.3.7.

Note: to obtain the CIDR notation (that is 1.2.3.0/29), we:

1 Take the binary prefix of the subnet: 0000 0001.0000 0010.0000 0011.0000 0;

2 Append zeros until we obtain 32 bits: 0000 0001.0000 0010.0000 0011.0000 0000;

3 Transform the value into dotted IP notation: 1.2.3.0;

4 Append "/" and the length of the binary prefix: 1.2.3.0/29.

Subnet B needs 7 addresses, so we need to allocate 8: from 1.2.3.8 to 1.2.3.15. We expect that the mask has length 29. Let's represent them in binary:

0000 0001.0000 0010.0000 0011.0000 1000 = 1.2.3.8

0000 0001.0000 0010.0000 0011.0000 1001 = 1.2.3.9

0000 0001.0000 0010.0000 0011.0000 1010 = 1.2.3.10

0000 0001.0000 0010.0000 0011.0000 1011 = 1.2.3.11

0000 0001.0000 0010.0000 0011.0000 1100 = 1.2.3.12

0000 0001.0000 0010.0000 0011.0000 1101 = 1.2.3.13

0000 0001.0000 0010.0000 0011.0000 1110 = 1.2.3.14

0000 0001.0000 0010.0000 0011.0000 1111 = 1.2.3.15

The interval is 1.2.3.8 - 1.2.3.15, and the prefix is 0000 0001.0000 0010.0000 0011.0000 1 with length 29, which corresponds to the block 1.2.3.8/29.

For Subnet A, we need to allocate 12 addresses. Thus we need a block with 16 addresses. We try to allocate starting from 1.2.3.16:

0000 0001.0000 0010.0000 0011.0001 0000 = 1.2.3.16

0000 0001.0000 0010.0000 0011.0001 0001 = 1.2.3.17

0000 0001.0000 0010.0000 0011.0001 0010 = 1.2.3.18

0000 0001.0000 0010.0000 0011.0001 0011 = 1.2.3.19

0000 0001.0000 0010.0000 0011.0001 0100 = 1.2.3.20

0000 0001.0000 0010.0000 0011.0001 0101 = 1.2.3.21

0000 0001.0000 0010.0000 0011.0001 0110 = 1.2.3.22

0000 0001.0000 0010.0000 0011.0001 0111 = 1.2.3.23

0000 0001.0000 0010.0000 0011.0001 1000 = 1.2.3.24

0000 0001.0000 0010.0000 0011.0001 1001 = 1.2.3.25

0000 0001.0000 0010.0000 0011.0001 1010 = 1.2.3.26

0000 0001.0000 0010.0000 0011.0001 1011 = 1.2.3.27

0000 0001.0000 0010.0000 0011.0001 1100 = 1.2.3.28

0000 0001.0000 0010.0000 0011.0001 1101 = 1.2.3.29

0000 0001.0000 0010.0000 0011.0001 1110 = 1.2.3.30

0000 0001.0000 0010.0000 0011.0001 1111 = 1.2.3.31

The interval is 1.2.3.16 - 1.2.3.31, the prefix is 0000 0001.0000 0010.0000 0011.0001 with length 28. Thus the CIDR notation is 1.2.3.16/28.

(c) It is not possible to allocate the addresses in this order.

The reason is that you have only two options in how to allocate addresses for Subnet A (either prefix 1.2.3.0/28 or prefix 1.2.3.16/28).

- If you allocate prefix 1.2.3.0/28 to Subnet A, then there will be no room to allocate smaller addresses for Subnet B.
- Instead, if you allocate prefix 1.2.3.16/28, there will be no room to allocate bigger addresses to Subnet C.

Network configuration

Consider the topology shown in Figure 1. There are three IP subnets (A, B and C) that contain some end-systems, and two IP subnets (D and E) that contain no end-systems. The green boxes (a, b, c, …g) denote network interfaces for routers R1, R2 and R3.

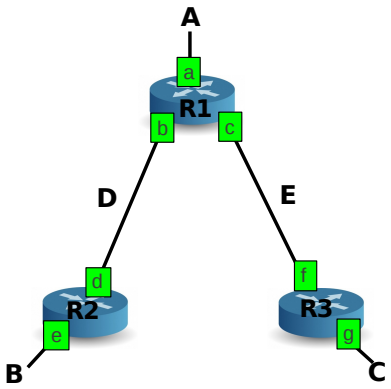

Figure 1: Problem topology.

- (*Basic*) Allocate an IP prefix to each subnets. Your allocation must respect the following constraints:
  - All prefixes must be allocated from 214.97.254/23.
  - Subnet A should have enough addresses to support 250 interfaces.
  - Subnet B should have enough addresses to support 120 interfaces.
  - Subnet C should have enough addresses to support 60 interfaces.
  - Each of subnets D and E should have enough addresses to support 2 interfaces.

> First, you need to round the number of required addresses (number of interfaces + 1 for network address + 1 for broadcast address) up to the nearest power of two.
>
> Then, a good strategy for allocating subnets on the available address space is the following:
>
> - Sort the subnets in descending order according to the (rounded) number of addresses they require
> - Start allocating addresses from one end of the address space (e.g. the beginning)

– For every remaining subnet (following the order in which you sorted them), allocate the address space after the previously allocated prefix. (i.e., do not leave gaps between consecutive subnets)

If you follow this strategy, you can guarantee that every subnet is well-aligned, and that there will be no gaps between allocated address blocks.

Nevertheless, there are multiple possible solutions for this exercise. Here is a possible allocation:

Subnet A: 214.97.254/24 (256 addresses)
Subnet B: 214.97.255.0/25 (128 addresses)
Subnet C: 214.97.255.128/26 (64 addresses)

Subnet D: 214.97.255.192/30 (4 addresses)
Subnet E: 214.97.255.196/30 (4 addresses)

- (*Basic*) Using your previous answer, provide the forwarding tables for each of the three routers (R1, R2, R3). Each table should contain two columns which show (i) the destination IP prefix, and (ii) the corresponding output link.

The simplest way to create a forwarding table is to create a separate rule for every subnet that we need to access.

If we take into account that the address ranges for different subnets do not overlap, we don't have to worry about longest-prefix-length matching.

Router 1:

Longest Prefix Match	Outgoing Interface
11010110 01100001 11111111 110000	Port b
11010110 01100001 11111111 110001	Port c
11010110 01100001 11111111 10	Port c
11010110 01100001 11111111 0	Port b
11010110 01100001 11111110	Port a

Router 2:

Longest Prefix Match	Outgoing Interface
11010110 01100001 11111111 110000	Port d
11010110 01100001 11111111 110001	Port d
11010110 01100001 11111111 10	Port d
11010110 01100001 11111111 0	Port e
11010110 01100001 11111110	Port d

Router 3:

Longest Prefix Match	Outgoing Interface
11010110 01100001 11111111 110000	Port f
11010110 01100001 11111111 110001	Port f
11010110 01100001 11111111 10	Port g
11010110 01100001 11111111 0	Port f
11010110 01100001 11111110	Port f

- (*Advanced*) Can you reduce the number of entries of each forwarding table, i.e., for each table create an equivalent one, which has the same outcome but consists of fewer entries?

We can reduce the number of entries by grouping the Longest prefix matches for the same outgoing interface. This will generate new entries with first column being the longest prefix between all (or some) of those entries and second column being the output port.

However, we need to be careful not to create conflict with another outgoing interface. For example, in the forwarding table of R1, the longest prefix match for Port b is "11010110 01100001 11111111". But if we use this prefix and place it at the first row, then a packet matching "11010110 01100001 11111111 10 " will be forwarded to Port b instead of Port c. Recall that the router scans the forwarding table in order and stops on the first prefix match. Even if we place it after the entries of Port c, packets will be wrongly forwarded to Port c instead of Port b. In this situation, the prefixes of Port b can not be grouped.

Router 1:

Longest Prefix Match	Outgoing Interface
11010110 01100001 11111111 110000	Port b
11010110 01100001 11111111 1	Port c
11010110 01100001 11111111 0	Port b
11010110 01100001 11111110	Port a

Router 2:

Longest Prefix Match	Outgoing Interface
11010110 01100001 11111111 0	Port e
11010110 01100001 1111111	Port d

Router 3:

Longest Prefix Match	Outgoing Interface
11010110 01100001 11111111 10	Port g
11010110 01100001 1111111	Port f

Network Address Translation

Intermediate

Figure 2 illustrates how Network Address Translation (NAT) works. Consider a similar topology, but suppose that the NAT gateway has external IP address 24.34.112.235, while the private IP address space is 192.168.1.0/24.

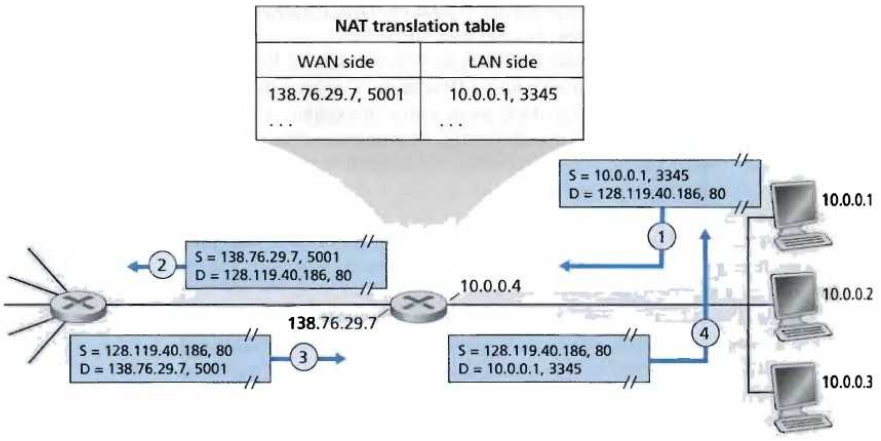

Figure 2: Network Address Translation (NAT) Process

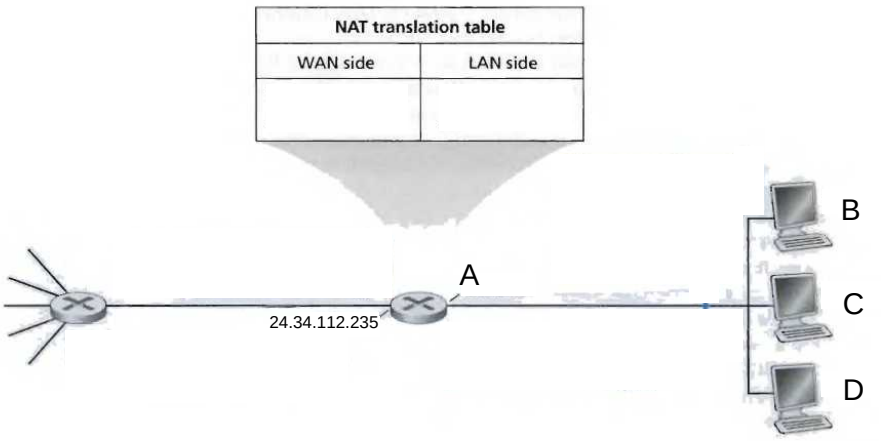

Figure 3: Problem topology.

- Complete Figure 3 by assigning IP addresses to all interfaces (labels A, B, C, and D) in the internal (private) network.

Router interface: 192.168.1.1
Home device addresses: 192.168.1.2, 192.168.1.3, 192.168.1.4

- Suppose that each end-system has two ongoing TCP connections, all to IP address 128.119.40.86, port 80. Provide the six corresponding entries in the NAT translation table.

Here is an example of a NAT translation table. Multiple solutions exist.

WAN Side	LAN Side
24.34.112.235, 4000	192.168.1.2, 3345
24.34.112.235, 4001	192.168.1.2, 3346
24.34.112.235, 4002	192.168.1.3, 3445
24.34.112.235, 4003	192.168.1.3, 3446
24.34.112.235, 4004	192.168.1.4, 3545
24.34.112.235, 4005	192.168.1.4, 3546