Lab2 Report

Name：喻梓浩

Student ID：1900013082

PartA

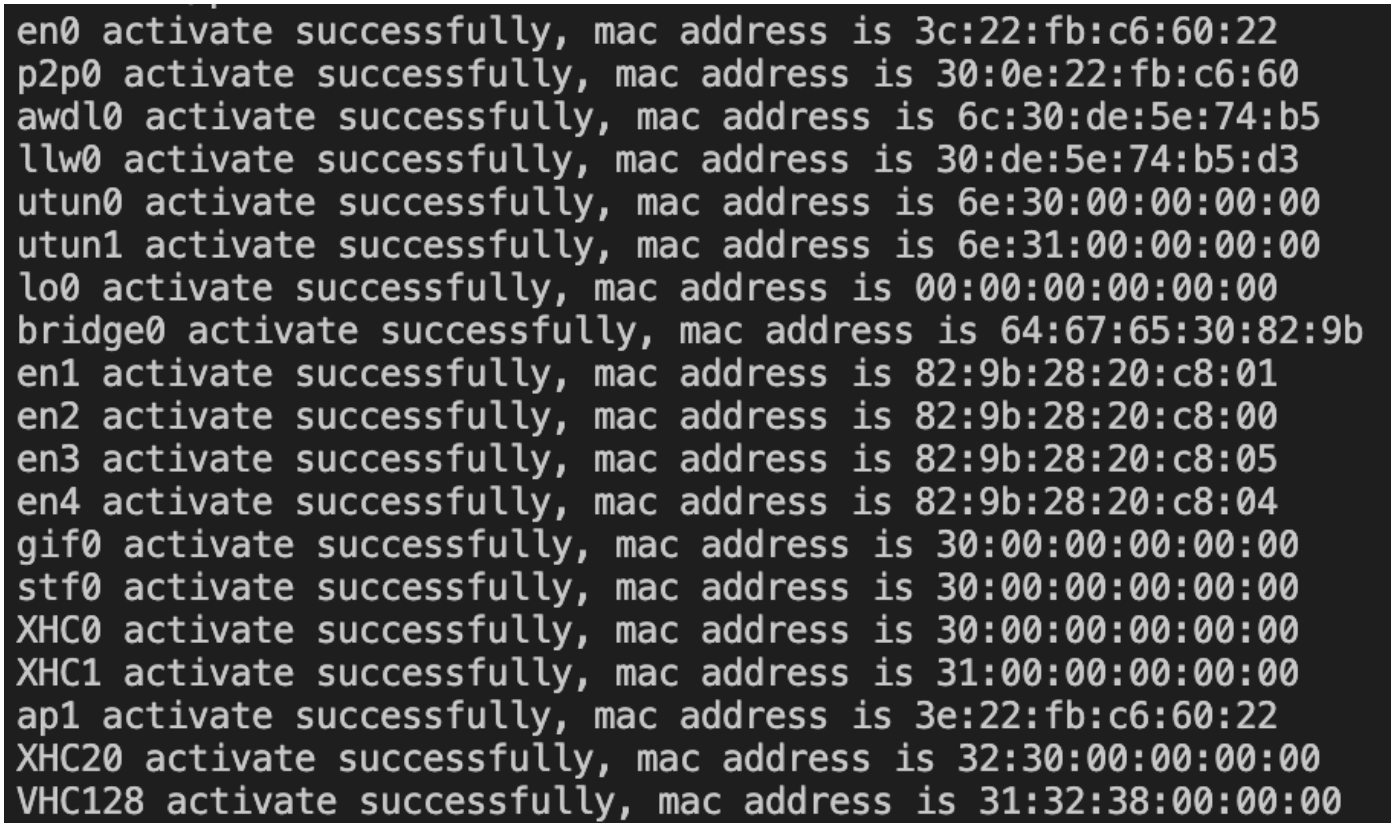
The test code is in parta_checkpoints.cpp which correspond to checkpoint 1,2.

Input these to test.

```
cd src
make clean
make parta
./parta
```

checkpoint1

By running device_activate_test() in parta_checkpoints.cpp, which print all the device information on the host, showing the result:



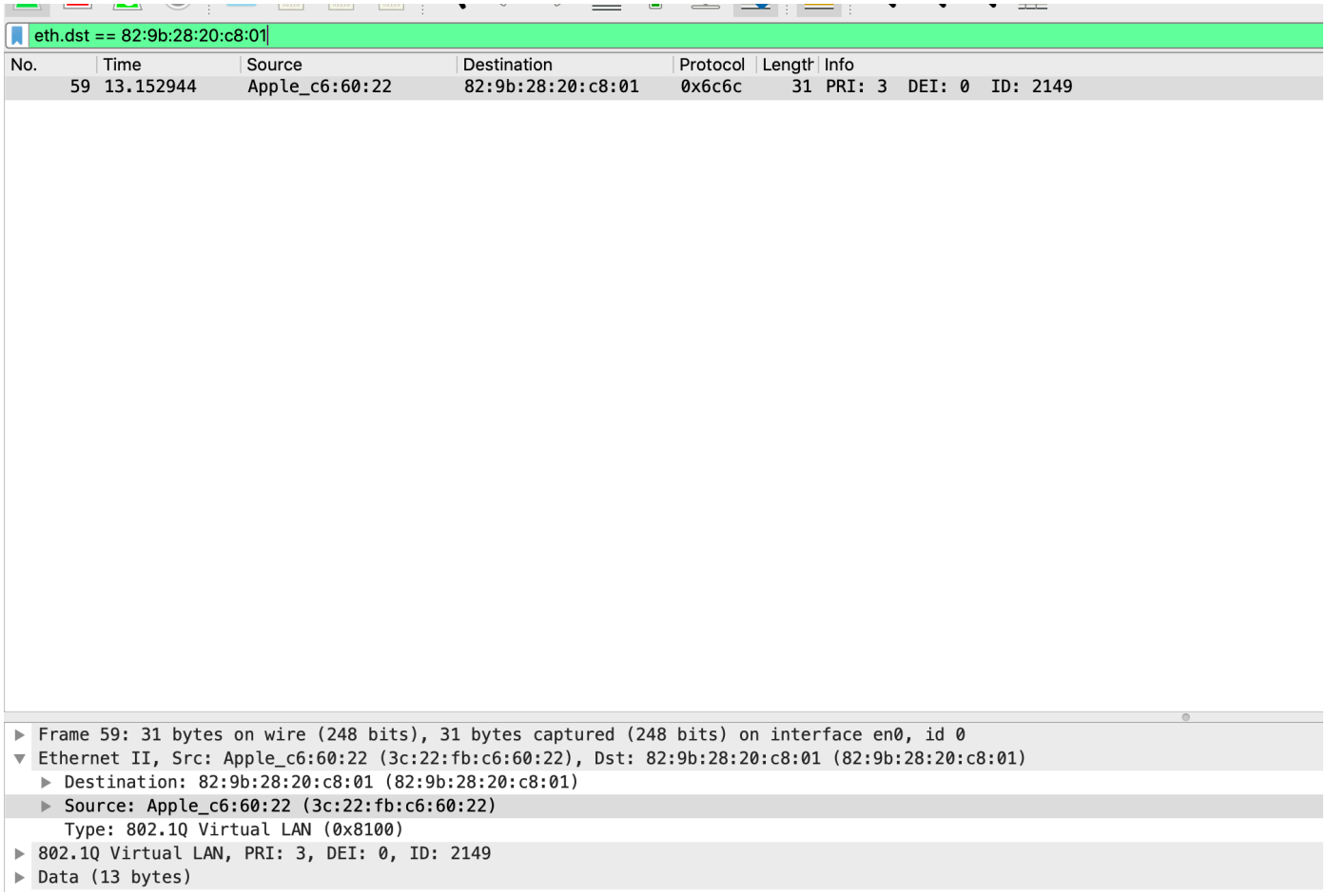
checkpoint2

send

We send frame through device "en0" whose mac address is 3c:22:fb:c6:60:22, the destmac address is 82:9b:28:20:c8:01.

```
successfully add en0 to device pool, id = 0
en0 receiving thread start listening ......
successfully add en1 to device pool, id = 1
en1 receiving thread start listening ......
start sending frame from 3c:22:fb:c6:60:22 to 82:9b:28:20:c8:01
send frame succeeded! frame size is 31
```

By monitor device "en0" by Wireshark, we can find the frame we send.



Receive

After receving thread of "en0" start listening , we visit some websites by Google Chrome, we can see that "en0" successfully receive frames from web service.

```
[pcap_callback]: device id = 0 receive [Caplen: 54] [Len: 54] packet
device's mac address = 3c:22:fb:c6:60:22, frame's destmac = 58:0a:20:8b:3a:80
drop useless packet: frame's destmac address doesn't match

[pcap_callback]: device id = 0 receive [Caplen: 54] [Len: 54] packet
device's mac address = 3c:22:fb:c6:60:22, frame's destmac = 58:0a:20:8b:3a:80
drop useless packet: frame's destmac address doesn't match

[pcap_callback]: device id = 0 receive [Caplen: 60] [Len: 60] packet
device's mac address = 3c:22:fb:c6:60:22, frame's destmac = 3c:22:fb:c6:60:22
[FrameReceivedCallback] of deivce0:
buf len = 42, the first 15 bytes:
45 0 0 28 0 33 40 0 32 6 B2 9A 7A CD 6D

[pcap_callback]: device id = 0 receive [Caplen: 60] [Len: 60] packet
device's mac address = 3c:22:fb:c6:60:22, frame's destmac = 3c:22:fb:c6:60:22
[FrameReceivedCallback] of deivce0:
buf len = 42, the first 15 bytes:
45 0 0 28 0 34 40 0 32 6 B2 99 7A CD 6D
```

PartB

Working environment

MacOS does not support vnet，try to build linux environment by docker

```
docker build -t lab2-env .
docker run -it --privileged=true -v $(pwd):/home/Lab2 lab2-env
```

Writing Task 1

1. When target IP and source IP are in same subnet, I broadcast a packet(like arp protocol) to all the subnet to find the MAC address of target IP, and record it's MAC address by an arp-map.
2. When target IP and source IP are not in same subnet, I use routing table to find the next hop to the target IP. Routing table is obtained by my own routing algorithm.

Writing Task 2

1. Every node have a thread to broadcast it's routing table, and a thread to update it's routing table.
2. Each item in routing table records the ip_prefix, ip_subnetMast, device to send this packet, MAC address of next-hop, distance from the target IP address and the time it entry to the table.
3. When a router receiving a routing table from it's neighborhood，the router will merge these two tables:
    1. When the item is not exist in table, add the item to table directly.
    2. When the item is exist in table, compare distance to the target IP address of them，accept the closer one and update entry time of this item.
4. Updating thread check the entry time of each item, delete the item who is expired.

There is a situation, when the virtual network is like: ns1-ns2-ns3, they are exchanging it's routing table. When ns3 is disconnected, the information about ns3 in the routing table of ns1 and ns2 will not be deleted, because this information about ns3 will continually delivered between them, the distance to ns3 in this item will be continually increasing. So I use the strategy that deleting the item whose distance to target IP address is greater than or equal to 10.

Checkpoint 3

I cature my own IP packet send from 10.100.1.1 to 10.100.2.2 by tcpdump，the message to send is "hello, world!".

The first 14 bytes are frame header, including dst MAC address(62:5c:58:e3:ab :2e), src MAC address(2a:78:29:b2:e7:84) and ether_type(0800 corresponding to ETHERTYPE_IP). The next 20 bytes are ip header, including IPPROTO(4000 corresponding to IPPROTO_UDP), src IP address(0a64 0101 corresponding to 10.100.1.1), dst IP address(0a64 0202 corresponding to 10.100.2.2) and checksum etc. The remaining bytes are corresponding to the message "hello, world!".

```
05:45:30.498480 IP (tos 0x0, ttl 16, id 0, offset 0, flags [DF], proto UDP (17), length 33)
    26de74c5b225.26725 > 10.100.2.2.27756: [bad udp cksum 0x2077 -> 0xa741!] UDP, bad length 28452 > 5
        0x0000:  625c 58e3 ab2e 2a78 29b2 e784 0800 4500  b\X...*x).....E.
        0x0010:  0021 0000 4000 1011 5302 0a64 0101 0a64  .!..@...S..d...d
        0x0020:  0202 6865 6c6c 6f2c 2077 6f72 6c64 21    ..hello,.world!
```

The second packet I capture is the packet that contain the routing table.

The first 14 bytes are frame header, including dst MAC address(ff:ff:ff:ff:ff :ff), src MAC address(2a:78:29:b2:e7:84) and ether_type(ffff corresponding to my own routing protocol). The remaining 54 bytes are corresponding to the routing table sent by 2a:78:29:b2:e7:84. The table has 3 items, and each item is 18 bytes in size.

```
05:45:30.500397 2a:78:29:b2:e7:84 (oui Unknown) > Broadcast, ethertype Unknown (0xffff), length 68:
        0x0000:  ffff ffff ffff 2a78 29b2 e784 ffff 0a64  ......*x).......d
        0x0010:  0100 ffff ff00 0000 0000 0000 0000 0000  ................
        0x0020:  0a64 0200 ffff ff00 625c 58e3 ab2e 0100  .d......b\X.....
        0x0030:  0000 0a64 0300 ffff ff00 625c 58e3 ab2e  ...d......b\X...
        0x0040:  0200 0000                                ....
```

Checkpoint 4

```
# mynet.txt


4
1 2 10.100.1
2 3 10.100.2
3 4 10.100.3
0 3 10.100.4
```

We can reproduce the experiment by the following steps:

```
# in terminal 1
cd vnetUtils/examples/
bash ./makeVNet < mynet.txt  # build VNet
```

```
cd ../helper
bash ./execNS ns1 bash  # entry ns1
cd ../../src
make clean
make listen #  compile listen.cpp which only route(send routing table and forwarding)
# in terminal 2
bash ./execNS ns2 bash  # entry ns2
cd ../../src
./listen
# in terminal 3
bash ./execNS ns3 bash  # entry ns3
cd ../../src
./listen
# in terminal 4
bash ./execNS ns4 bash  # entry ns4
cd ../../src
./listen

# in terminal 1
./listen

# in terminal 2
ctrl+C
./listen # after 10s seconds
```

Initial routing table of ns1:

```
_____________________________________________________________________________________________
||         ip_prefix||        subnetMask||           send_by||          next_hop||          dist||
||       10.100.1.0||     255.255.255.0|| 2a:78:29:b2:e7:84|| 00:00:00:00:00:00||             0||
||       10.100.2.0||     255.255.255.0|| 2a:78:29:b2:e7:84|| 62:5c:58:e3:ab:2e||             1||
||       10.100.3.0||     255.255.255.0|| 2a:78:29:b2:e7:84|| 62:5c:58:e3:ab:2e||             2||
=============================================================================================
```

Routing table of ns1 after disconnecting ns2 (waiting 10s can see the result):

```
_____________________________________________________________________________________________
||         ip_prefix||        subnetMask||           send_by||          next_hop||          dist||
||       10.100.1.0||     255.255.255.0|| 2a:78:29:b2:e7:84|| 00:00:00:00:00:00||             0||
=============================================================================================
```

Routing table of ns1 after ns2 connected again:

```
_____________________________________________________________________________________________
||         ip_prefix||        subnetMask||           send_by||          next_hop||          dist||
||       10.100.1.0||     255.255.255.0|| 2a:78:29:b2:e7:84|| 00:00:00:00:00:00||             0||
||       10.100.2.0||     255.255.255.0|| 2a:78:29:b2:e7:84|| 62:5c:58:e3:ab:2e||             1||
||       10.100.3.0||     255.255.255.0|| 2a:78:29:b2:e7:84|| 62:5c:58:e3:ab:2e||             2||
=============================================================================================
```

Checkpoint 5

```
# mynet2.txt which correspond to the network structure

6
1 2 10.100.1
2 3 10.100.2
3 4 10.100.3
2 5 10.100.4
5 6 10.100.5
3 6 10.100.6

1 default 2
2 default 3
3 default 4
2 default 5
5 default 6
3 default 6
```

The routing tables of ns1,ns2,...,ns6 are as follows:

```
_____________________________________________________________________________________________________
||         ip_prefix||        subnetMask||           send_by||          next_hop||              dist||
||        10.100.1.0||     255.255.255.0|| 92:be:e2:a5:4e:00|| 00:00:00:00:00:00||                 0||
||        10.100.2.0||     255.255.255.0|| 92:be:e2:a5:4e:00|| 46:e5:d0:4d:cc:df||                 1||
||        10.100.3.0||     255.255.255.0|| 92:be:e2:a5:4e:00|| 46:e5:d0:4d:cc:df||                 2||
||        10.100.4.0||     255.255.255.0|| 92:be:e2:a5:4e:00|| 46:e5:d0:4d:cc:df||                 1||
||        10.100.5.0||     255.255.255.0|| 92:be:e2:a5:4e:00|| 46:e5:d0:4d:cc:df||                 2||
||        10.100.6.0||     255.255.255.0|| 92:be:e2:a5:4e:00|| 46:e5:d0:4d:cc:df||                 2||
=====================================================================================================
```

```
_____________________________________________________________________________________________________
||         ip_prefix||        subnetMask||           send_by||          next_hop||              dist||
||        10.100.1.0||     255.255.255.0|| 46:e5:d0:4d:cc:df|| 00:00:00:00:00:00||                 0||
||        10.100.2.0||     255.255.255.0|| da:a8:ef:fc:b0:a3|| 00:00:00:00:00:00||                 0||
||        10.100.3.0||     255.255.255.0|| da:a8:ef:fc:b0:a3|| 32:2d:84:59:c4:74||                 1||
||        10.100.4.0||     255.255.255.0|| ea:c1:01:78:cd:b2|| 00:00:00:00:00:00||                 0||
||        10.100.5.0||     255.255.255.0|| ea:c1:01:78:cd:b2|| 56:a6:60:a9:dd:a3||                 1||
||        10.100.6.0||     255.255.255.0|| da:a8:ef:fc:b0:a3|| 32:2d:84:59:c4:74||                 1||
=====================================================================================================
```

```
_____________________________________________________________________________________________________
||         ip_prefix||        subnetMask||           send_by||          next_hop||              dist||
||        10.100.1.0||     255.255.255.0|| 32:2d:84:59:c4:74|| da:a8:ef:fc:b0:a3||                 1||
||        10.100.2.0||     255.255.255.0|| 32:2d:84:59:c4:74|| 00:00:00:00:00:00||                 0||
||        10.100.3.0||     255.255.255.0|| 06:99:83:aa:1b:6c|| 00:00:00:00:00:00||                 0||
||        10.100.4.0||     255.255.255.0|| 32:2d:84:59:c4:74|| da:a8:ef:fc:b0:a3||                 1||
||        10.100.5.0||     255.255.255.0|| da:21:7a:c9:18:b0|| 6a:4c:88:7c:57:94||                 1||
||        10.100.6.0||     255.255.255.0|| da:21:7a:c9:18:b0|| 00:00:00:00:00:00||                 0||
=====================================================================================================
```

ip_prefix	subnetMask	send_by	next_hop	dist
10.100.1.0	255.255.255.0	e6:11:3e:e7:03:ee	06:99:83:aa:1b:6c	2
10.100.2.0	255.255.255.0	e6:11:3e:e7:03:ee	06:99:83:aa:1b:6c	1
10.100.3.0	255.255.255.0	e6:11:3e:e7:03:ee	00:00:00:00:00:00	0
10.100.4.0	255.255.255.0	e6:11:3e:e7:03:ee	06:99:83:aa:1b:6c	2
10.100.5.0	255.255.255.0	e6:11:3e:e7:03:ee	06:99:83:aa:1b:6c	2
10.100.6.0	255.255.255.0	e6:11:3e:e7:03:ee	06:99:83:aa:1b:6c	1

ip_prefix	subnetMask	send_by	next_hop	dist
10.100.1.0	255.255.255.0	56:a6:60:a9:dd:a3	ea:c1:01:78:cd:b2	1
10.100.2.0	255.255.255.0	56:a6:60:a9:dd:a3	ea:c1:01:78:cd:b2	1
10.100.3.0	255.255.255.0	f2:10:da:06:38:af	32:b7:38:f1:b4:7d	2
10.100.4.0	255.255.255.0	56:a6:60:a9:dd:a3	00:00:00:00:00:00	0
10.100.5.0	255.255.255.0	f2:10:da:06:38:af	00:00:00:00:00:00	0
10.100.6.0	255.255.255.0	f2:10:da:06:38:af	32:b7:38:f1:b4:7d	1

ip_prefix	subnetMask	send_by	next_hop	dist
10.100.1.0	255.255.255.0	32:b7:38:f1:b4:7d	f2:10:da:06:38:af	2
10.100.2.0	255.255.255.0	6a:4c:88:7c:57:94	da:21:7a:c9:18:b0	1
10.100.3.0	255.255.255.0	6a:4c:88:7c:57:94	da:21:7a:c9:18:b0	1
10.100.4.0	255.255.255.0	32:b7:38:f1:b4:7d	f2:10:da:06:38:af	1
10.100.5.0	255.255.255.0	32:b7:38:f1:b4:7d	00:00:00:00:00:00	0
10.100.6.0	255.255.255.0	6a:4c:88:7c:57:94	00:00:00:00:00:00	0

After disconnect ns5, the routing tables of ns1,...,ns4,ns6 are as follows:

ip_prefix	subnetMask	send_by	next_hop	dist
10.100.1.0	255.255.255.0	92:be:e2:a5:4e:00	00:00:00:00:00:00	0
10.100.2.0	255.255.255.0	92:be:e2:a5:4e:00	46:e5:d0:4d:cc:df	1
10.100.3.0	255.255.255.0	92:be:e2:a5:4e:00	46:e5:d0:4d:cc:df	2
10.100.4.0	255.255.255.0	92:be:e2:a5:4e:00	46:e5:d0:4d:cc:df	1
10.100.5.0	255.255.255.0	92:be:e2:a5:4e:00	46:e5:d0:4d:cc:df	3
10.100.6.0	255.255.255.0	92:be:e2:a5:4e:00	46:e5:d0:4d:cc:df	2

ip_prefix	subnetMask	send_by	next_hop	dist
10.100.1.0	255.255.255.0	46:e5:d0:4d:cc:df	00:00:00:00:00:00	0
10.100.2.0	255.255.255.0	da:a8:ef:fc:b0:a3	00:00:00:00:00:00	0
10.100.3.0	255.255.255.0	da:a8:ef:fc:b0:a3	32:2d:84:59:c4:74	1
10.100.4.0	255.255.255.0	ea:c1:01:78:cd:b2	00:00:00:00:00:00	0
10.100.5.0	255.255.255.0	da:a8:ef:fc:b0:a3	32:2d:84:59:c4:74	2
10.100.6.0	255.255.255.0	da:a8:ef:fc:b0:a3	32:2d:84:59:c4:74	1

ip_prefix	subnetMask	send_by	next_hop	dist
10.100.1.0	255.255.255.0	32:2d:84:59:c4:74	da:a8:ef:fc:b0:a3	1
10.100.2.0	255.255.255.0	32:2d:84:59:c4:74	00:00:00:00:00:00	0
10.100.3.0	255.255.255.0	06:99:83:aa:1b:6c	00:00:00:00:00:00	0
10.100.4.0	255.255.255.0	32:2d:84:59:c4:74	da:a8:ef:fc:b0:a3	1
10.100.5.0	255.255.255.0	da:21:7a:c9:18:b0	6a:4c:88:7c:57:94	1
10.100.6.0	255.255.255.0	da:21:7a:c9:18:b0	00:00:00:00:00:00	0

```
||         ip_prefix||        subnetMask||           send_by||          next_hop||               dist||
||       10.100.1.0||     255.255.255.0|| e6:11:3e:e7:03:ee|| 06:99:83:aa:1b:6c||                  2||
||       10.100.2.0||     255.255.255.0|| e6:11:3e:e7:03:ee|| 06:99:83:aa:1b:6c||                  1||
||       10.100.3.0||     255.255.255.0|| e6:11:3e:e7:03:ee|| 00:00:00:00:00:00||                  0||
||       10.100.4.0||     255.255.255.0|| e6:11:3e:e7:03:ee|| 06:99:83:aa:1b:6c||                  2||
||       10.100.5.0||     255.255.255.0|| e6:11:3e:e7:03:ee|| 06:99:83:aa:1b:6c||                  2||
||       10.100.6.0||     255.255.255.0|| e6:11:3e:e7:03:ee|| 06:99:83:aa:1b:6c||                  1||
```

```
||         ip_prefix||        subnetMask||           send_by||          next_hop||               dist||
||       10.100.1.0||     255.255.255.0|| 6a:4c:88:7c:57:94|| da:21:7a:c9:18:b0||                  2||
||       10.100.2.0||     255.255.255.0|| 6a:4c:88:7c:57:94|| da:21:7a:c9:18:b0||                  1||
||       10.100.3.0||     255.255.255.0|| 6a:4c:88:7c:57:94|| da:21:7a:c9:18:b0||                  1||
||       10.100.4.0||     255.255.255.0|| 6a:4c:88:7c:57:94|| da:21:7a:c9:18:b0||                  2||
||       10.100.5.0||     255.255.255.0|| 32:b7:38:f1:b4:7d|| 00:00:00:00:00:00||                  0||
||       10.100.6.0||     255.255.255.0|| 6a:4c:88:7c:57:94|| 00:00:00:00:00:00||                  0||
```

We can see that the routing tables of ns3 and ns4 have not changed, and the routing tables of ns1, ns2 and ns6 have changed.

Checkpoint 6

My routing table is sorted according to the subnet mask, and the item who has longer prefix length will be listed in front of the routing table. When looking up the routing table, search in order and return after finding a matched item, so the item with the longest prefix will be obtained first.

```
bool RouterItem::operator < (const RouterItem& item) const {
    if (subnetMask.s_addr != item.subnetMask.s_addr)
        return subnetMask.s_addr > item.subnetMask.s_addr;
    return ip_prefix.s_addr < item.ip_prefix.s_addr;
}
```

We can check its correctness through a unit test:

```
cd src
make prefix
./prefix
```

The test insert two items with different ip_prefix, and find the next hop of 10.100.1.1, showing the following result:

```
||         ip_prefix||        subnetMask||           send_by||          next_hop||               dist||
||       10.100.1.0||     255.255.255.0|| aa:aa:aa:aa:aa:aa||                 b||                  2||
||       10.100.0.0||       255.255.0.0|| aa:aa:aa:aa:aa:aa||                 a||                  2||
The next hop of [10.100.1.1] is [b]
```