Lab Report

Lab Name Switch

Course	Computer Network
Major	Computer Science and Technology
Id	191220129
Name	Shangyu.Xing
Email	191220129@smail.nju.edu.cn
Date	2021.03

Contents

1 Objective

- Learn core functionalities of an Ethernet learning switch;
- learn to implement hardware logic using the Switchyard framework;
- learn to capture network package using wireshark.

2 Requirements

This lab requires to implement the core functionalities of an Ethernet learning switch. Specific tasks are as follows.

- Implement a basic learning switch;
- implement a learning switch with timeout mechanism;
- implement a learning switch with least recently used mechanism;
- implement a learning switch with least traffic volume mechanism;

3 Procedure

In this section, I will explain how I implement the switch with different mechanism in detail.
Although the three mechanisms resemble little with one another, their behavior as a switch remains consistent. The general procedure of a switch processing an incoming packet is listed as follows:

1. Do some update to its forwarding table according to the source address and the specific mechanism;
2. Look its forwarding table for an entry with the destination address;
    - if found, do some update to its forwarding table according to the specific mechanism, and then forward the packet to the destination address;
    - if not found, flood the packet out all ports except the incoming one.

According to the features, we can do some high-level abstraction. I established a python class called ForwardTable to encapsulate the behavior of different mechanisms. The class template is here.

```python
class ForwardTable:
    # a dict containing data of the forwarding table
    data = {}

    # method to update forwarding table
    def update_in(self, mac, intf):
        pass
    def update_out(self, mac, intf):
        pass

    # method for query
    def get(self, mac):
        return None
```

After implementing the above class for every mechanism, we are free from the worry of the different mechanisms. All we have to is to write a uniform main function regardless of the specific mechanism to implement the switch functionality.
The uniform main function is as follows.

```python
def main(net: switchyard.llnetbase.LLNetBase):
    my_interfaces = net.interfaces()
    mymacs = [intf.ethaddr for intf in my_interfaces]
    # init the forwarding table
    table = ForwardTable()

    while True:
        try:
            _, fromIface, packet = net.recv_packet()
        except NoPackets:
            continue
        except Shutdown:
            break

        log_debug (f"In {net.name} received packet {packet} on {
            fromIface}")
        eth = packet.get_header(Ethernet)

        # update the forwarding table upon receiving a packet (
            learn)
        table.update_in(eth.src, fromIface)

        if eth is None:
            log_info("Received a non-Ethernet packet?!")
            return
        if eth.dst in mymacs:
            log_info("Received a packet intended for me")

        else:
```

```
28              # query the forwarding table for interface connected
                    to dst
29              dst_intf = table.get(eth.dst)
30              if dst_intf != None and eth.dst != 'ff:ff:ff:ff:ff:
                    ff':
31                  # update the forwarding table before sending a
                        packet (update status)
32                  table.update_out(eth.dst, dst_intf)
33                  net.send_packet(dst_intf, packet)
34                  log_info(f"Sending packet {packet} to {dst_intf}
                        ")
35              else:
36                  for intf in my_interfaces:
37                      if fromIface!= intf.name:
38                          log_info(f"Flooding packet {packet} to {
                                intf.name}")
39                          net.send_packet(intf, packet)
40
41      net.shutdown()
```

3.1 Basic Switch

There is no status information to maintain, so the implementation of the ForwardTable class is very simple.

```
1  class ForwardTable:
2      data = {} # mac -> intf
3
4      def update_in(self, mac, intf):
5          self.data[mac] = intf
6
7      def update_out(self, mac, intf):
8          pass
9
10     def get(self, mac):
11         return self.data.get(mac)
```

3.2 Timeout

In this part we need to maintain a status information: timestamp. Timestamp is updated when a new packet arrived. If the timestamp of an entry is older than timeout value, the entry is considered invalid, and query of it should return None.

```
1  class ForwardTable:
2      data = {} # mac -> (intf, timestamp)
3      timeout = 10
```

```
4
5       def update_in(self, mac, intf):
6           self.data[mac] = (intf, time())
7
8       def update_out(self, max, intf):
9           pass
10
11      def get(self, mac):
12          t = self.data.get(mac)
13          if t == None or time() - t[1] >= self.timeout:
14              return None
15          return t[0]
```

3.3 Least Recently Used

The status information here is age. Age of all entries is incremented by 1 whenever a new packet arrives. If the number of entries is larger than max_size, the oldest entry is removed.

```
1  class ForwardTable:
2      data = {} # mac -> (intf, age)
3      max_size = 5
4
5      def update_in(self, mac, intf):
6          for key in self.data.keys():
7              self.data[key] = (self.data[key][0], self.data[key
                   ][1] + 1)
8          if self.data.get(mac) != None:
9              self.data[mac] = (intf, self.data[mac][1])
10         else:
11             self.data[mac] = (intf, 0)
12             if len(self.data) > self.max_size:
13                 del self.data[max(self.data, key=lambda x: self.
                       data[x][1])]
14
15     def update_out(self, mac, intf):
16         if self.data.get(mac) != None:
17             self.data[mac] = (self.data[mac][0], 0)
18
19     def get(self, mac):
20         t = self.data.get(mac)
21         if t == None:
22             return None
23         return t[0]
```

3.4 Least Traffic Volume

Here we need to record traffic volume for each entry. Traffic of the corresponding entry is increased by 1 when a new packet arrives. If the number of entries is larger than max_size, the entry with least traffic volume is removed.

```
class ForwardTable:
    data = {} # mac -> (intf, traffic)
    max_size = 5

    def update_in(self, mac, intf):
        if self.data.get(mac) != None:
            self.data[mac] = (intf, self.data[mac][1])
        else:
            if len(self.data) >= self.max_size:
                del self.data[min(self.data, key=lambda x: self.
                    data[x][1])]
            self.data[mac] = (intf, 0)

    def update_out(self, mac, intf):
        if self.data.get(mac) != None:
            self.data[mac] = (intf, self.data[mac][1] + 1)

    def get(self, mac):
        t = self.data.get(mac)
        if t == None:
            return None
        return t[0]
```

4 Result

4.1 Basic Switch

The procedure of the switch's forwarding is as follows:

1. The client broadcast arp packet, and the switch learned that the client is related to a certain interface;
2. the server reply arp packet to the client, and the switch learned another entry;
3. the client send echo request to the server, and the switch forward the packet according to its forwarding table;
4. the server broadcast arp packet;

5. the client reply arp packet, and the switch forward the packet according to its forwarding table;

6. the server send echo reply to the client, and the switch forward the packet according to its forwarding table;

7. repeat procedure 3, 6.

As a result, wireshark can capture echo packet in server1 but cannot capture echo packet in server2.

4.2 Timeout

Switchyard test result:

```
xsy@ASUS-VivoBook:/media/xsy/SSD/MEGA/assignments/network/lab-2-xingshangyu$ sudo swyard -t testcases/myswitch_to_testscenario.srpy myswitch_to.py
[sudo] password for xsy:
16:54:58 2021/03/24     INFO Starting test scenario testcases/myswitch_to_testscenario.srpy
16:54:58 2021/03/24     INFO Flooding packet Ethernet 30:00:00:00:00:02->ff:ff:ff:ff:ff:ff IP | IPv4 172.16.42.2->255.255.255.255 ICMP | ICMP EchoRequest 0 0 (0
 data bytes) to eth0
16:54:58 2021/03/24     INFO Flooding packet Ethernet 30:00:00:00:00:02->ff:ff:ff:ff:ff:ff IP | IPv4 172.16.42.2->255.255.255.255 ICMP | ICMP EchoRequest 0 0 (0
 data bytes) to eth2
16:54:58 2021/03/24     INFO Sending packet Ethernet 20:00:00:00:00:01->30:00:00:00:00:02 IP | IPv4 192.168.1.100->172.16.42.2 ICMP | ICMP EchoRequest 0 0 (0 da
ta bytes) to eth1
16:55:18 2021/03/24     INFO Flooding packet Ethernet 20:00:00:00:00:01->30:00:00:00:00:02 IP | IPv4 192.168.1.100->172.16.42.2 ICMP | ICMP EchoRequest 0 0 (0 d
ata bytes) to eth1
16:55:18 2021/03/24     INFO Flooding packet Ethernet 20:00:00:00:00:01->30:00:00:00:00:02 IP | IPv4 192.168.1.100->172.16.42.2 ICMP | ICMP EchoRequest 0 0 (0 d
ata bytes) to eth2
16:55:18 2021/03/24     INFO Received a packet intended for me

Results for test scenario switch tests: 9 passed, 0 failed, 0 pending
```

Figure 1: Timeout test result

Mininet test(using command:
client ping -c 1 server1
(after 5s) repeat
(after 15s) repeat
)

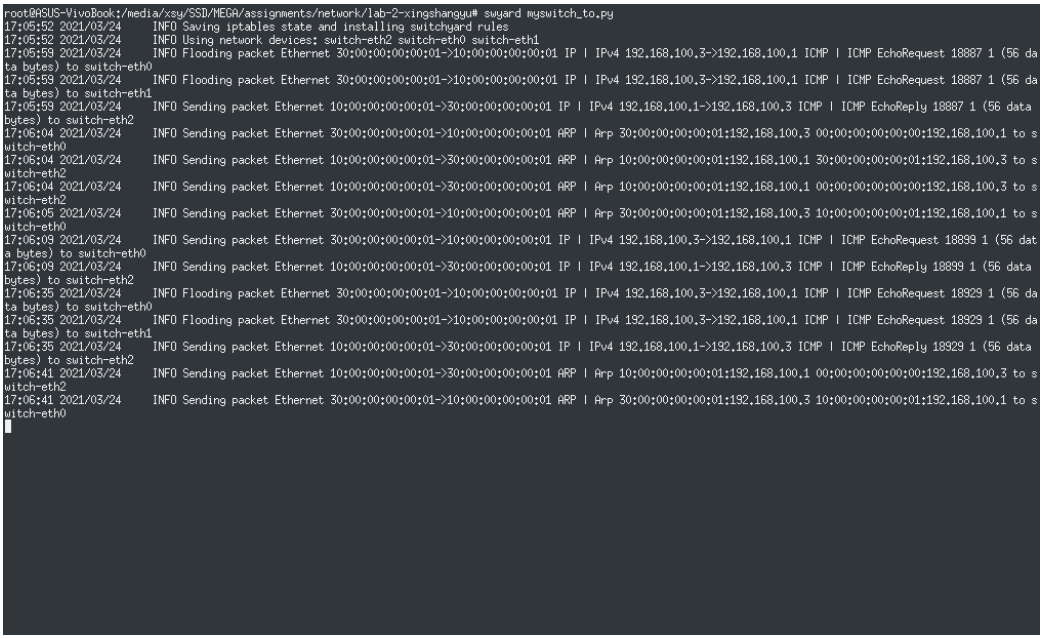

Figure 2: Switch log

As can be seen in the picture above, the entries in forwarding table did not timeout in 5s but timeout after 15s.

4.3 Least Recently Used

Switchyard test result:

```
xsy@ASUS-VivoBook:/media/xsy/SSD/MEGA/assignments/network/lab-2-xingshangyu$ sudo swyard -t testcases/myswitch_lru_testscenario.srpy myswitch_lru.py
17:27:27 2021/03/24     INFO Starting test scenario testcases/myswitch_lru_testscenario.srpy
17:27:27 2021/03/24     INFO Flooding packet Ethernet 30:00:00:00:00:02->ff:ff:ff:ff:ff:ff IP | IPv4 172.16.42.2->255.255.255.255 ICMP | ICMP EchoRequest 0 0 (0
 data bytes) to eth0
17:27:27 2021/03/24     INFO Flooding packet Ethernet 30:00:00:00:00:02->ff:ff:ff:ff:ff:ff IP | IPv4 172.16.42.2->255.255.255.255 ICMP | ICMP EchoRequest 0 0 (0
 data bytes) to eth2
17:27:27 2021/03/24     INFO Flooding packet Ethernet 30:00:00:00:00:02->ff:ff:ff:ff:ff:ff IP | IPv4 172.16.42.2->255.255.255.255 ICMP | ICMP EchoRequest 0 0 (0
 data bytes) to eth3
17:27:27 2021/03/24     INFO Flooding packet Ethernet 30:00:00:00:00:02->ff:ff:ff:ff:ff:ff IP | IPv4 172.16.42.2->255.255.255.255 ICMP | ICMP EchoRequest 0 0 (0
 data bytes) to eth4
17:27:27 2021/03/24     INFO Sending packet Ethernet 20:00:00:00:00:01->30:00:00:00:00:02 IP | IPv4 192.168.1.100->172.16.42.2 ICMP | ICMP EchoRequest 0 0 (0 da
ta bytes) to eth1
17:27:27 2021/03/24     INFO Sending packet Ethernet 20:00:00:00:00:03->30:00:00:00:00:02 IP | IPv4 192.168.1.102->172.16.42.2 ICMP | ICMP EchoRequest 0 0 (0 da
ta bytes) to eth1
17:27:27 2021/03/24     INFO Sending packet Ethernet 30:00:00:00:00:04->20:00:00:00:00:01 IP | IPv4 172.16.42.4->192.168.1.100 ICMP | ICMP EchoRequest 0 0 (0 da
ta bytes) to eth0
17:27:27 2021/03/24     INFO Sending packet Ethernet 20:00:00:00:00:01->30:00:00:00:00:04 IP | IPv4 192.168.1.100->172.16.42.4 ICMP | ICMP EchoReply 0 0 (0 data
 bytes) to eth3
17:27:27 2021/03/24     INFO Sending packet Ethernet 40:00:00:00:00:05->20:00:00:00:00:01 IP | IPv4 128.16.42.4->192.168.1.100 ICMP | ICMP EchoRequest 0 0 (0 da
ta bytes) to eth0
17:27:27 2021/03/24     INFO Sending packet Ethernet 30:00:00:00:00:05->20:00:00:00:00:01 IP | IPv4 172.16.42.5->192.168.1.100 ICMP | ICMP EchoRequest 0 0 (0 da
ta bytes) to eth0
17:27:27 2021/03/24     INFO Flooding packet Ethernet 20:00:00:00:00:05->30:00:00:00:00:02 IP | IPv4 192.16.42.4->172.16.42.2 ICMP | ICMP EchoRequest 0 0 (0 dat
a bytes) to eth0
17:27:27 2021/03/24     INFO Flooding packet Ethernet 20:00:00:00:00:05->30:00:00:00:00:02 IP | IPv4 192.16.42.4->172.16.42.2 ICMP | ICMP EchoRequest 0 0 (0 dat
a bytes) to eth1
17:27:27 2021/03/24     INFO Flooding packet Ethernet 20:00:00:00:00:05->30:00:00:00:00:02 IP | IPv4 192.16.42.4->172.16.42.2 ICMP | ICMP EchoRequest 0 0 (0 dat
a bytes) to eth2
17:27:27 2021/03/24     INFO Flooding packet Ethernet 20:00:00:00:00:05->30:00:00:00:00:02 IP | IPv4 192.16.42.4->172.16.42.2 ICMP | ICMP EchoRequest 0 0 (0 dat
a bytes) to eth3
17:27:27 2021/03/24     INFO Received a packet intended for me

Results for test scenario switch tests: 18 passed, 0 failed, 0 pending
```

Figure 3: Least recently used test result

Mininet test(setting table capacity to 2, using command:
client ping -c 1 server1;
client ping -c 1 server2;
client ping -c 1 server1
)

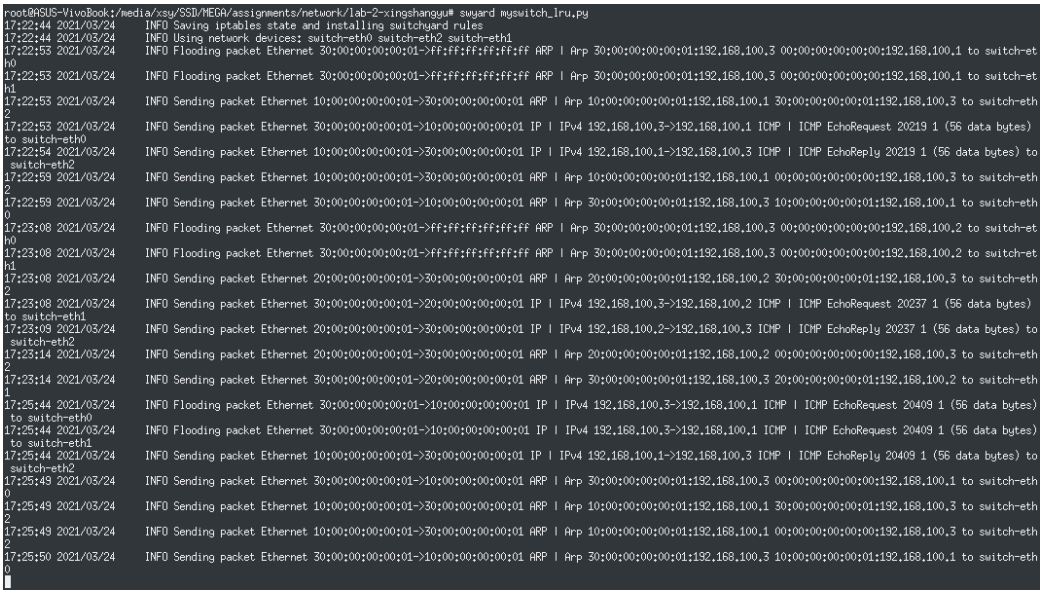

Figure 4: Switch log

As can be seen in the picture above, the least recently used entry(10:00:00:00:00:01) in forwarding table is removed when the second command is executed.

4.4 Least Traffic Volume

Switchyard test result:

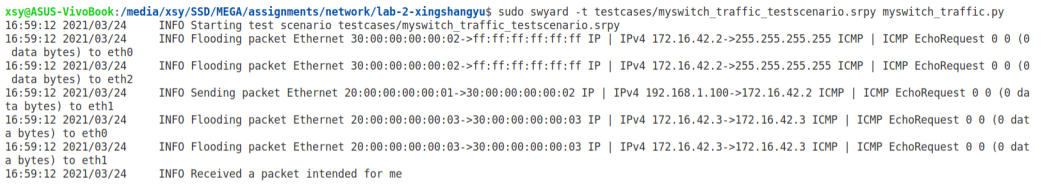

Figure 5: Least Traffic Volume test result

Mininet test(setting table capacity to 2, using command:
client ping -c 1 server1;
client ping -c 1 server2;
client ping -c 1 server1
)

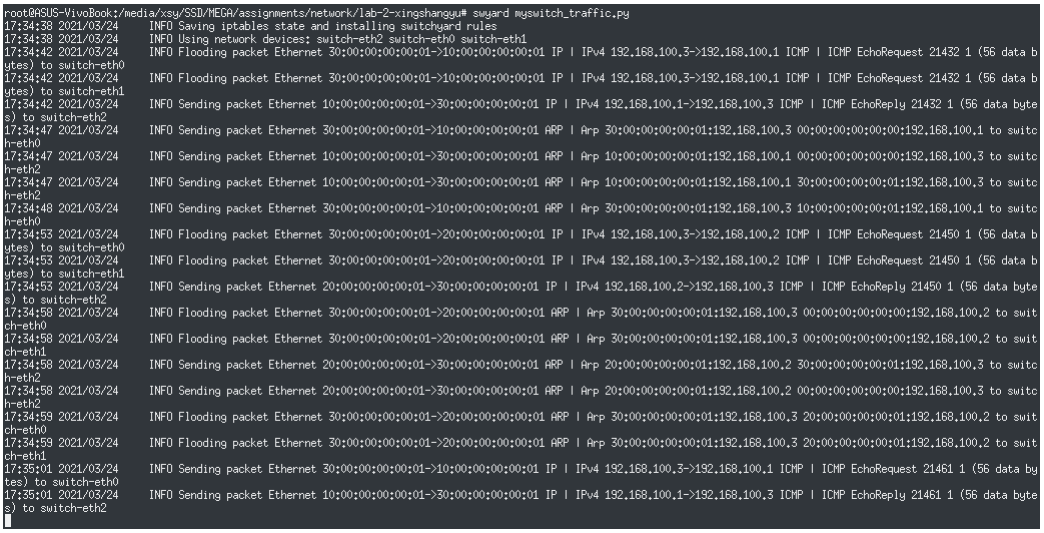

Figure 6: Switch log

As can be seen in the picture above, the entry woth least traffic volume(20:00:00:00:00:01) in forwarding table is removed when the second command is executed.

5 Summary

- Knowing how to use tools effectively will greatly enhance working efficiency;
- English reading and writing skills are important.