南京大学本科生实验报告

课程名称：**计算机网络**　　　　任课教师：李文中　　　　助教：

学院	**计算机科学与技术系**	专业（方向）	**计算机科学与技术**
学号	**221220143**	姓名	**陈雨檬**
Email	**221220143@qq.com**	开始/完成日期	**2024.3.20/2024.3.28**

1. 实验名称

Lab2:Learning Switch

2. 实验目的

学习交换机的核心功能，实现交换机的逻辑，实现不同的机制清除交换机转发表中的条目。

3. 实验内容

Task1：Preparation

How：创建好相应的文件和目录即可。

Task2：Basic Switch

How：修改 myswitch.py 代码文件的代码逻辑，添加 maclist 和 interlist 两个列表用于模拟 switch 学习时的表格，记录外部 MAC 地址对应的接口位置。完成后在网络中发送流量进行测试，测试后进行部署并使用 wireshark 进行抓包测试。

Task3：Timeouts

How: 在 Task2 的基础上添加一个 timelist 用来记录每个条目对应上一次出现的时间（time（time）)）。同时在发出包之前，对每个条目进行检查，如果相差超过设定的值 timeout 就清除该条目。再进行样例测试和部署测试。

Task4：Least Recently Used

How: 在 Task2 的基础上添加一个 agelist 用来记录每个条目对应的年龄，同时用 age 作为当前的“年份”（可以避免每一趟都需要更新每个条目的年龄，提高效率)，如果需要更新则找出 agelist 中最小的那一个并更新，当前的目录 agelist 的值更新为当前的 age；同时维护一个列表 stack（伪栈，实际上是伪队列）来按顺序保存应该被弹出的目录的下标，每次需要弹出时都弹出 stack 中的第一个元素，以此来降低时间复杂度，再进行测试和部署。

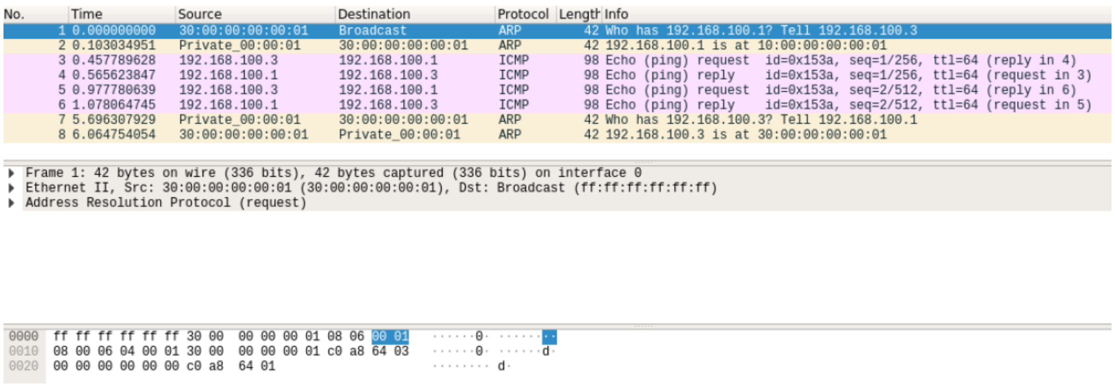
Task5：Least Traffic Volume FAQ

How: 在 Task2 的基础上添加一个 volumelist 用来记录每个条目对应的流量，如果需要更新则找出 volumelist 中最小的条目进行更新，再进行测试和部署。

4. 实验结果

Task1：Task1 无实验结果

Task2：进行部署时的工作后，观察 server1 和 server2 接口抓取到的流量，结果如下

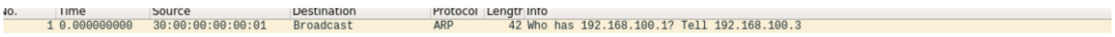


```
No.   Time         Source               Destination          Protocol Length Info
    1 0.000000000  30:00:00:00:00:01    Broadcast            ARP      42 Who has 192.168.100.1? Tell 192.168.100.3
    2 0.103034951  Private_00:00:01     30:00:00:00:00:01    ARP      42 192.168.100.1 is at 10:00:00:00:00:01
    3 0.457789628  192.168.100.3        192.168.100.1        ICMP     98 Echo (ping) request  id=0x153a, seq=1/256, ttl=64 (reply in 4)
    4 0.565623847  192.168.100.1        192.168.100.3        ICMP     98 Echo (ping) reply    id=0x153a, seq=1/256, ttl=64 (request in 3)
    5 0.977780639  192.168.100.3        192.168.100.1        ICMP     98 Echo (ping) request  id=0x153a, seq=2/512, ttl=64 (reply in 6)
    6 1.078064745  192.168.100.1        192.168.100.3        ICMP     98 Echo (ping) reply    id=0x153a, seq=2/512, ttl=64 (request in 5)
    7 5.696307929  Private_00:00:01     30:00:00:00:00:01    ARP      42 Who has 192.168.100.3? Tell 192.168.100.1
    8 6.064754054  30:00:00:00:00:01    Private_00:00:01     ARP      42 192.168.100.3 is at 30:00:00:00:00:01

▶ Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
▶ Ethernet II, Src: 30:00:00:00:00:01 (30:00:00:00:00:01), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
▶ Address Resolution Protocol (request)

0000  ff ff ff ff ff ff 30 00  00 00 00 01 08 06 00 01   ······0· ······..
0010  08 00 06 04 00 01 30 00  00 00 00 01 c0 a8 64 03   ······0· ······d·
0020  00 00 00 00 00 00 c0 a8  64 01                     ········ d·
```

图 1 是 server1 接口抓取到的流量（见 lab_2-1-server1.pcapng）

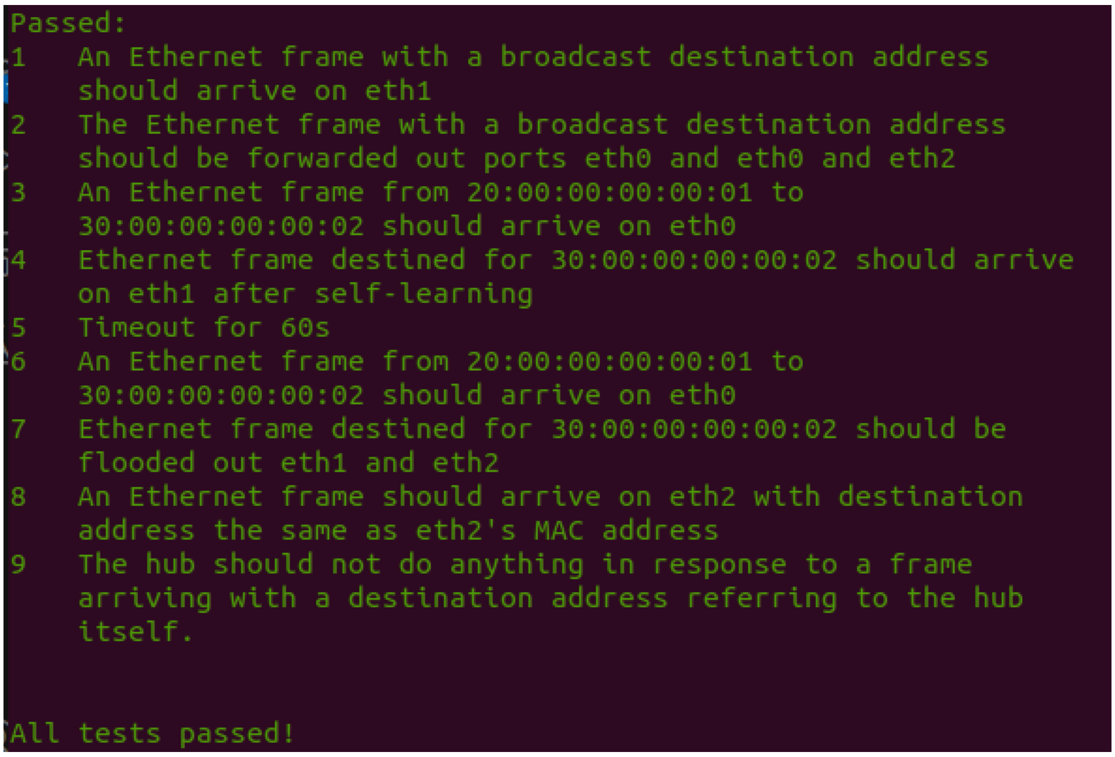


```
No.   Time         Source               Destination          Protocol Length Info
    1 0.000000000  30:00:00:00:00:01    Broadcast            ARP      42 Who has 192.168.100.1? Tell 192.168.100.3
```

图 2 是 server2 接口抓取到的流量（见 lab_2-1-server2.pcapng）

结果分析：首先 client 发送帧， switch 收到帧后发送广播信号，在 table 中添加了 client 的 mac 地址和对应接口信息，然后把帧从自身的另外两个接口发出，server1 和 server2 都接受到了该广播信号，但 server2 不回答，server1 则发送一个回答帧，switch 接受到该帧后记录下了 server1 的 mac 地址和对应接口，同时在 table 中直接找到了回答帧的目的地址，故直接把该帧送往 client。第二次 client 发出帧时，switch 直接在 table 中找到了 server1 的相关信息，故不发出广播信号，因此 server2 没有再收到广播信号。Server1 收到帧后再回答，重复一次上述的回答过程。

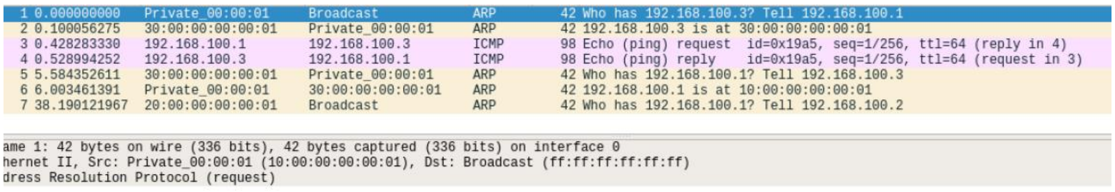
Task3：首先进行样例的测试，测试结果如下：



```
Passed:
1    An Ethernet frame with a broadcast destination address
     should arrive on eth1
2    The Ethernet frame with a broadcast destination address
     should be forwarded out ports eth0 and eth0 and eth2
3    An Ethernet frame from 20:00:00:00:00:01 to
     30:00:00:00:00:02 should arrive on eth0
4    Ethernet frame destined for 30:00:00:00:00:02 should arrive
     on eth1 after self-learning
5    Timeout for 60s
6    An Ethernet frame from 20:00:00:00:00:01 to
     30:00:00:00:00:02 should arrive on eth0
7    Ethernet frame destined for 30:00:00:00:00:02 should be
     flooded out eth1 and eth2
8    An Ethernet frame should arrive on eth2 with destination
     address the same as eth2's MAC address
9    The hub should not do anything in response to a frame
     arriving with a destination address referring to the hub
     itself.


All tests passed!
```

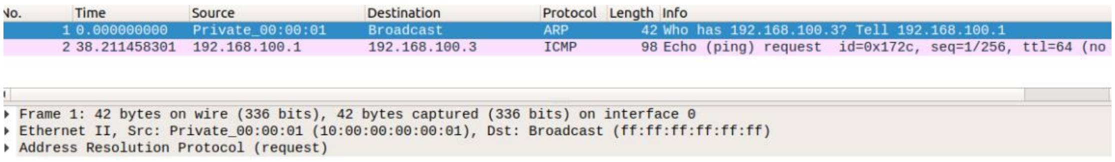
再进行 mininet 的部署后测试，在运行 mininet 后，使用 wireshark 抓取 client-eth0 接口的流量，使 switch 正常运行，首先使 server1 发出一个包给 client，此时 switch 记录 server1 和 client 的接口信息，经过 20s 后使 server1 发出一个包给 client，如果遗忘机制正常，此时 client 应该接受到一个广播信息，同时 server2 也会接收到信息。结果如下



```
  1 0.000000000   Private_00:00:01   Broadcast          ARP    42 Who has 192.168.100.3? Tell 192.168.100.1
  2 0.100056275   30:00:00:00:00:01  Private_00:00:01   ARP    42 192.168.100.3 is at 30:00:00:00:00:01
  3 0.428283330   192.168.100.1      192.168.100.3      ICMP   98 Echo (ping) request  id=0x19a5, seq=1/256, ttl=64 (reply in 4)
  4 0.528994252   192.168.100.3      192.168.100.1      ICMP   98 Echo (ping) reply    id=0x19a5, seq=1/256, ttl=64 (request in 3)
  5 5.584352611   30:00:00:00:00:01  Private_00:00:01   ARP    42 Who has 192.168.100.1? Tell 192.168.100.3
  6 6.003461391   Private_00:00:01   30:00:00:00:00:01  ARP    42 192.168.100.1 is at 10:00:00:00:00:01
  7 38.190121967  20:00:00:00:00:01  Broadcast          ARP    42 Who has 192.168.100.1? Tell 192.168.100.2

ame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
hernet II, Src: Private_00:00:01 (10:00:00:00:00:01), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
dress Resolution Protocol (request)
```

（见 lab_2-2-client.pcapng）



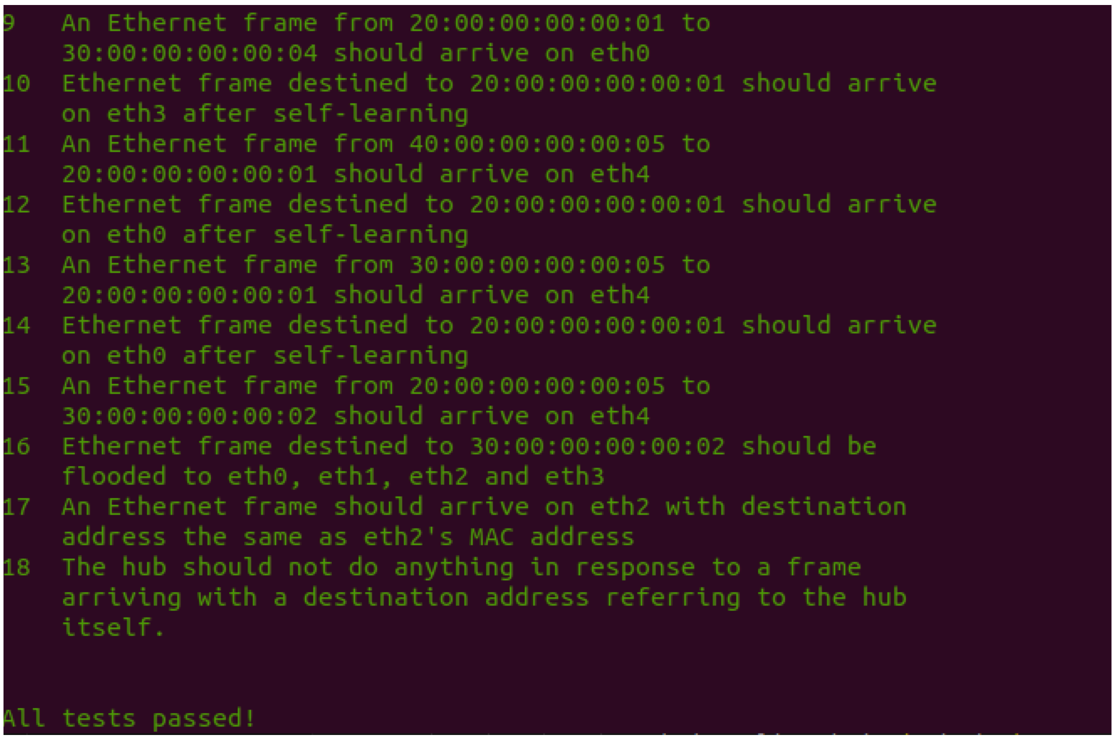
```
No.      Time           Source             Destination        Protocol  Length  Info
       1 0.000000000    Private_00:00:01   Broadcast          ARP           42 Who has 192.168.100.3? Tell 192.168.100.1
       2 38.211458301   192.168.100.1      192.168.100.3      ICMP          98 Echo (ping) request  id=0x172c, seq=1/256, ttl=64 (no

▸ Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
▸ Ethernet II, Src: Private_00:00:01 (10:00:00:00:00:01), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
▸ Address Resolution Protocol (request)
```

（见 lab_2-2-server2.pcapng）

结果符合预期

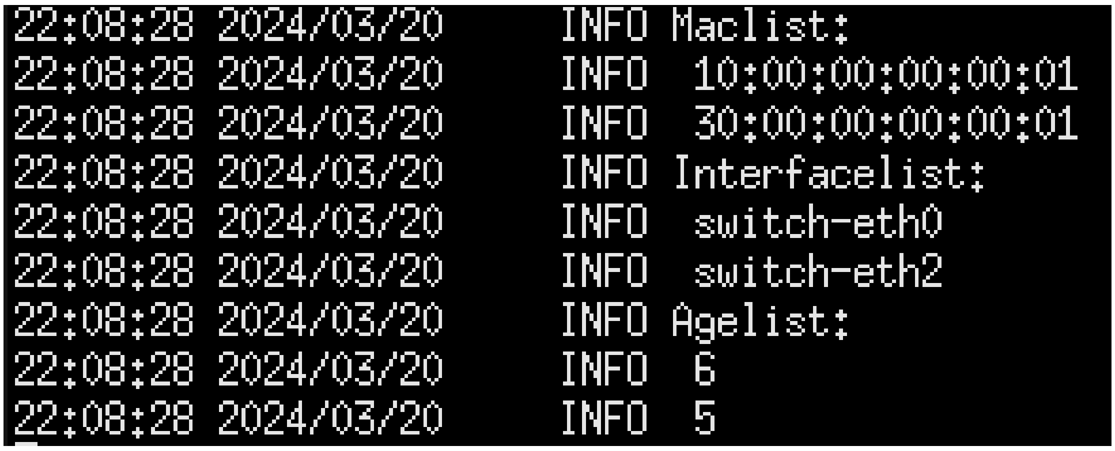
Task4：样例测试结果如下



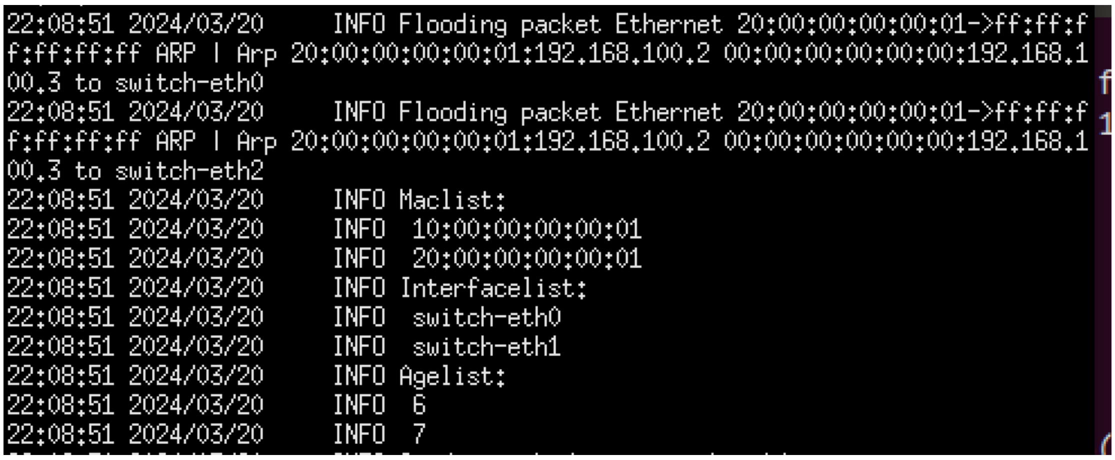
```
9   An Ethernet frame from 20:00:00:00:00:01 to
    30:00:00:00:00:04 should arrive on eth0
10  Ethernet frame destined to 20:00:00:00:00:01 should arrive
    on eth3 after self-learning
11  An Ethernet frame from 40:00:00:00:00:05 to
    20:00:00:00:00:01 should arrive on eth4
12  Ethernet frame destined to 20:00:00:00:00:01 should arrive
    on eth0 after self-learning
13  An Ethernet frame from 30:00:00:00:00:05 to
    20:00:00:00:00:01 should arrive on eth4
14  Ethernet frame destined to 20:00:00:00:00:01 should arrive
    on eth0 after self-learning
15  An Ethernet frame from 20:00:00:00:00:05 to
    30:00:00:00:00:02 should arrive on eth4
16  Ethernet frame destined to 30:00:00:00:00:02 should be
    flooded to eth0, eth1, eth2 and eth3
17  An Ethernet frame should arrive on eth2 with destination
    address the same as eth2's MAC address
18  The hub should not do anything in response to a frame
    arriving with a destination address referring to the hub
    itself.


All tests passed!
```

再进行 mininet 的部署测试，为了便于测试，我们把 table 能容纳的条目从 5 改成 2.运行 mininet 和 switch 后，我们使 server1 发送一条流量到 client，再使 server2 发送一条流量到 client，switch 的表格内容如下，表明实现的逻辑正确

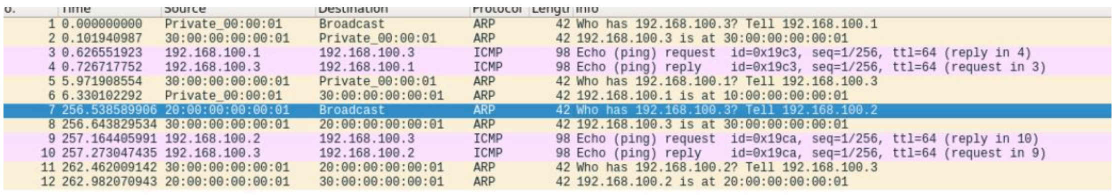


```
22:08:28 2024/03/20     INFO Maclist:
22:08:28 2024/03/20     INFO  10:00:00:00:00:01
22:08:28 2024/03/20     INFO  30:00:00:00:00:01
22:08:28 2024/03/20     INFO Interfacelist:
22:08:28 2024/03/20     INFO  switch-eth0
22:08:28 2024/03/20     INFO  switch-eth2
22:08:28 2024/03/20     INFO Agelist:
22:08:28 2024/03/20     INFO  6
22:08:28 2024/03/20     INFO  5
```



```
22:08:51 2024/03/20     INFO Flooding packet Ethernet 20:00:00:00:00:01->ff:ff:f
f:ff:ff:ff ARP | Arp 20:00:00:00:00:01:192.168.100.2 00:00:00:00:00:00:192.168.1
00.3 to switch-eth0
22:08:51 2024/03/20     INFO Flooding packet Ethernet 20:00:00:00:00:01->ff:ff:f
f:ff:ff:ff ARP | Arp 20:00:00:00:00:01:192.168.100.2 00:00:00:00:00:00:192.168.1
00.3 to switch-eth2
22:08:51 2024/03/20     INFO Maclist:
22:08:51 2024/03/20     INFO  10:00:00:00:00:01
22:08:51 2024/03/20     INFO  20:00:00:00:00:01
22:08:51 2024/03/20     INFO Interfacelist:
22:08:51 2024/03/20     INFO  switch-eth0
22:08:51 2024/03/20     INFO  switch-eth1
22:08:51 2024/03/20     INFO Agelist:
22:08:51 2024/03/20     INFO  6
22:08:51 2024/03/20     INFO  7
```

图一表明在 server1 发送流量到 client 后 switch 的表格内容；因此在 server2 发送流量时，更“老”的 client 的信息被替换成了 server2 的信息，验证成功。

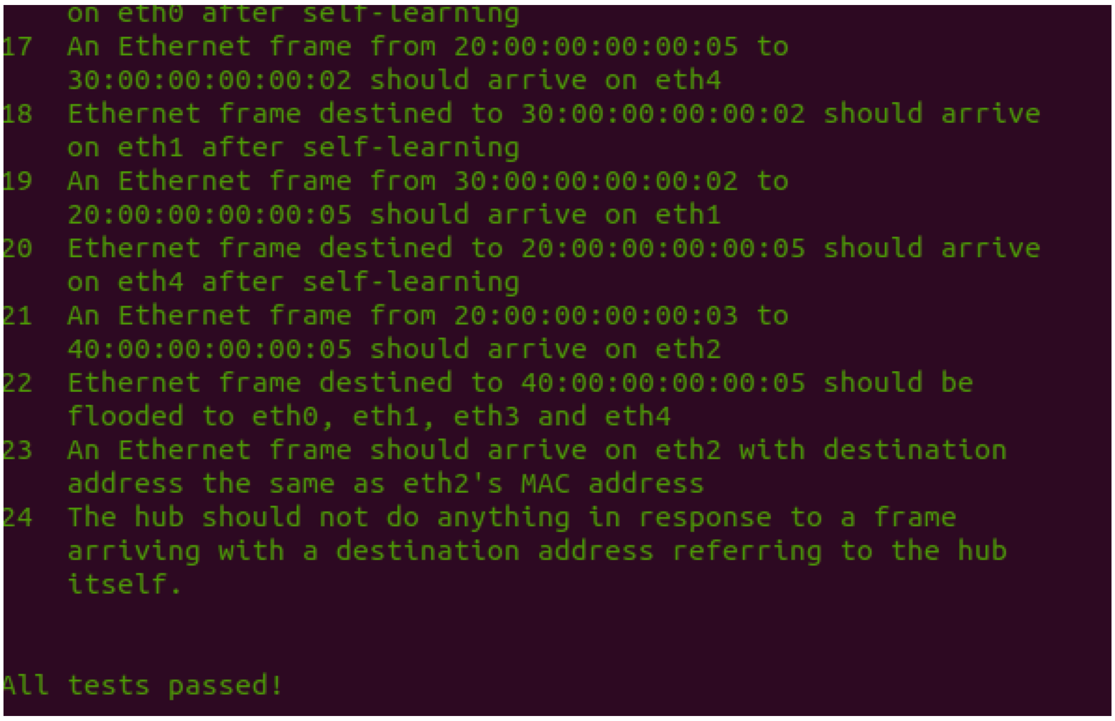


```
o.  Time           Source             Destination        Protocol Length Info
  1 0.000000000    Private_00:00:01   Broadcast          ARP      42 Who has 192.168.100.3? Tell 192.168.100.1
  2 0.101940987    30:00:00:00:00:01  Private_00:00:01   ARP      42 192.168.100.3 is at 30:00:00:00:00:01
  3 0.626551923    192.168.100.1      192.168.100.3      ICMP     98 Echo (ping) request  id=0x19c3, seq=1/256, ttl=64 (reply in 4)
  4 0.726717752    192.168.100.3      192.168.100.1      ICMP     98 Echo (ping) reply    id=0x19c3, seq=1/256, ttl=64 (request in 3)
  5 5.971908554    30:00:00:00:00:01  Private_00:00:01   ARP      42 Who has 192.168.100.1? Tell 192.168.100.3
  6 6.330102292    Private_00:00:01   30:00:00:00:00:01  ARP      42 192.168.100.1 is at 10:00:00:00:00:01
  7 256.538589906  20:00:00:00:00:01  Broadcast          ARP      42 Who has 192.168.100.3? Tell 192.168.100.2
  8 256.643829534  30:00:00:00:00:01  20:00:00:00:00:01  ARP      42 192.168.100.3 is at 30:00:00:00:00:01
  9 257.164405991  192.168.100.2      192.168.100.3      ICMP     98 Echo (ping) request  id=0x19ca, seq=1/256, ttl=64 (reply in 10)
 10 257.273047435  192.168.100.3      192.168.100.2      ICMP     98 Echo (ping) reply    id=0x19ca, seq=1/256, ttl=64 (request in 9)
 11 262.462009142  30:00:00:00:00:01  20:00:00:00:00:01  ARP      42 Who has 192.168.100.2? Tell 192.168.100.3
 12 262.982070943  20:00:00:00:00:01  30:00:00:00:00:01  ARP      42 192.168.100.2 is at 20:00:00:00:00:01
```

（见 lab_2-3-client.pcapng）

对 client 进行抓包，发现在 server2 call client 时仍受到了广播信号，符合预期。

Task5：首先进行测试样例的测试，结果如下

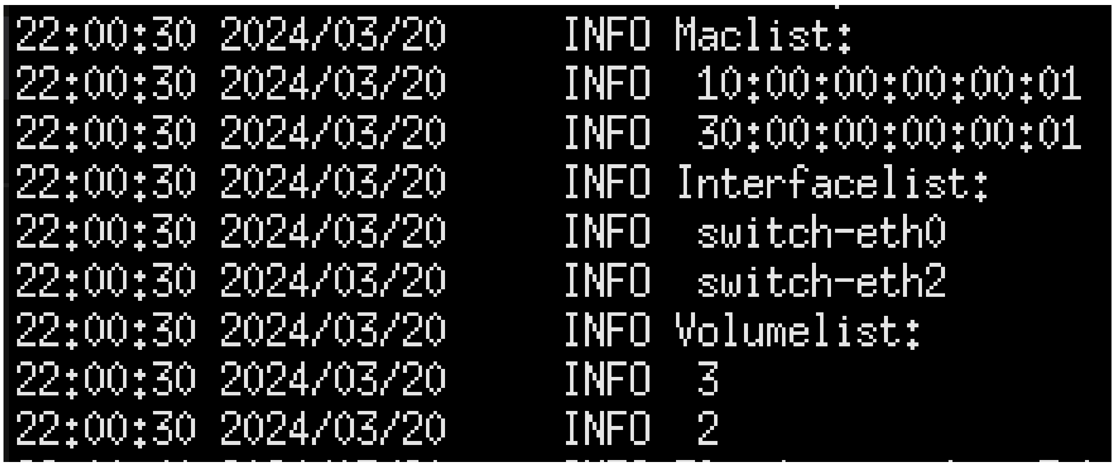


```
    on eth0 after self-learning
17  An Ethernet frame from 20:00:00:00:00:05 to
    30:00:00:00:00:02 should arrive on eth4
18  Ethernet frame destined to 30:00:00:00:00:02 should arrive
    on eth1 after self-learning
19  An Ethernet frame from 30:00:00:00:00:02 to
    20:00:00:00:00:05 should arrive on eth1
20  Ethernet frame destined to 20:00:00:00:00:05 should arrive
    on eth4 after self-learning
21  An Ethernet frame from 20:00:00:00:00:03 to
    40:00:00:00:00:05 should arrive on eth2
22  Ethernet frame destined to 40:00:00:00:00:05 should be
    flooded to eth0, eth1, eth3 and eth4
23  An Ethernet frame should arrive on eth2 with destination
    address the same as eth2's MAC address
24  The hub should not do anything in response to a frame
    arriving with a destination address referring to the hub
    itself.


All tests passed!
```

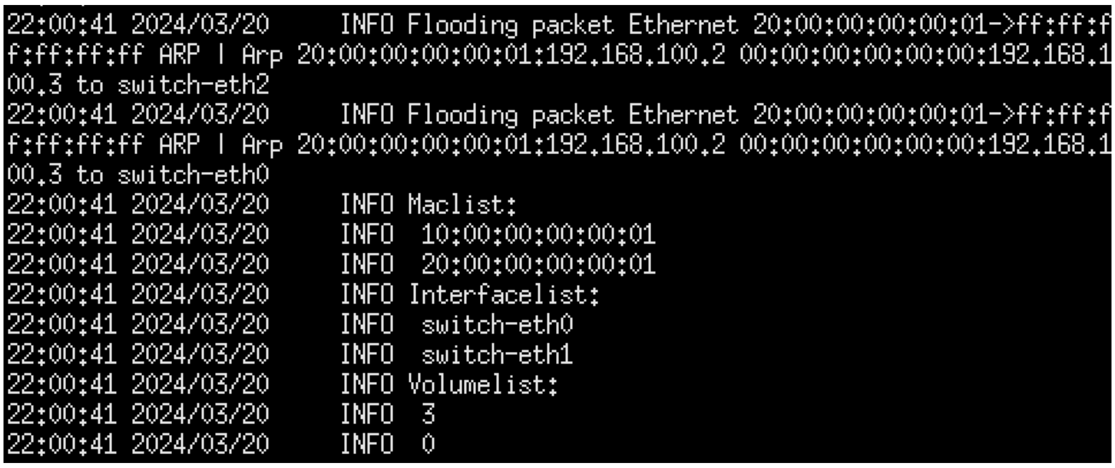
然后部署 mininet 进行测试，同样的为了便于测试，我们把最大条目数量改成 2.使 mininet 和 switch 运行，先使 server1 向 client 发送流量，再使 server2 向 client 发生流量。

Server1 向 client 发送流量完成后，switch 表格的记录如下



```
22:00:30 2024/03/20     INFO Maclist:
22:00:30 2024/03/20     INFO  10:00:00:00:00:01
22:00:30 2024/03/20     INFO  30:00:00:00:00:01
22:00:30 2024/03/20     INFO Interfacelist:
22:00:30 2024/03/20     INFO  switch-eth0
22:00:30 2024/03/20     INFO  switch-eth2
22:00:30 2024/03/20     INFO Volumelist:
22:00:30 2024/03/20     INFO  3
22:00:30 2024/03/20     INFO  2
```

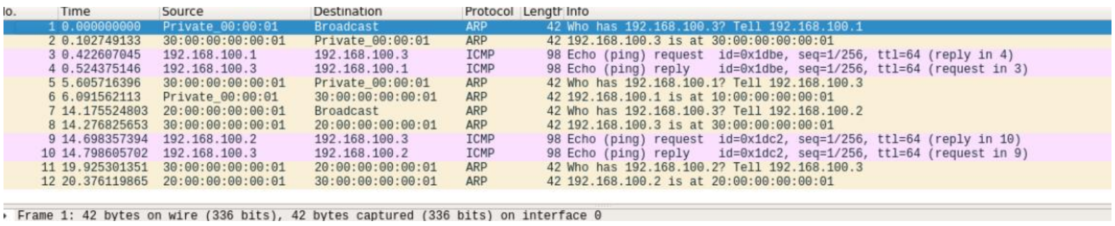
Server2 向 client 发送流量时



```
22:00:41 2024/03/20     INFO Flooding packet Ethernet 20:00:00:00:00:01->ff:ff:f
f:ff:ff:ff ARP | Arp 20:00:00:00:00:01:192.168.100.2 00:00:00:00:00:00:192.168.1
00.3 to switch-eth2
22:00:41 2024/03/20     INFO Flooding packet Ethernet 20:00:00:00:00:01->ff:ff:f
f:ff:ff:ff ARP | Arp 20:00:00:00:00:01:192.168.100.2 00:00:00:00:00:00:192.168.1
00.3 to switch-eth0
22:00:41 2024/03/20     INFO Maclist:
22:00:41 2024/03/20     INFO  10:00:00:00:00:01
22:00:41 2024/03/20     INFO  20:00:00:00:00:01
22:00:41 2024/03/20     INFO Interfacelist:
22:00:41 2024/03/20     INFO  switch-eth0
22:00:41 2024/03/20     INFO  switch-eth1
22:00:41 2024/03/20     INFO Volumelist:
22:00:41 2024/03/20     INFO  3
22:00:41 2024/03/20     INFO  0
```

可以看到关于 client 的记录已经被清除掉（因为流量数量更小），

被替换成了 server2 的信息，因此验证成功。



```
lo.  Time         Source             Destination        Protocol Length Info
   1 0.000000000  Private_00:00:01   Broadcast          ARP      42 Who has 192.168.100.3? Tell 192.168.100.1
   2 0.102749133  30:00:00:00:00:01  Private_00:00:01   ARP      42 192.168.100.3 is at 30:00:00:00:00:01
   3 0.422607045  192.168.100.1      192.168.100.3      ICMP     98 Echo (ping) request  id=0x1dbe, seq=1/256, ttl=64 (reply in 4)
   4 0.524375146  192.168.100.3      192.168.100.1      ICMP     98 Echo (ping) reply    id=0x1dbe, seq=1/256, ttl=64 (request in 3)
   5 5.605716396  30:00:00:00:00:01  Private_00:00:01   ARP      42 Who has 192.168.100.1? Tell 192.168.100.3
   6 6.091562113  Private_00:00:01   30:00:00:00:00:01  ARP      42 192.168.100.1 is at 10:00:00:00:00:01
   7 14.175524803 20:00:00:00:00:01  Broadcast          ARP      42 Who has 192.168.100.3? Tell 192.168.100.2
   8 14.276825653 30:00:00:00:00:01  20:00:00:00:00:01  ARP      42 192.168.100.3 is at 30:00:00:00:00:01
   9 14.698357394 192.168.100.2      192.168.100.3      ICMP     98 Echo (ping) request  id=0x1dc2, seq=1/256, ttl=64 (reply in 10)
  10 14.798605702 192.168.100.3      192.168.100.2      ICMP     98 Echo (ping) reply    id=0x1dc2, seq=1/256, ttl=64 (request in 9)
  11 19.925301351 30:00:00:00:00:01  20:00:00:00:00:01  ARP      42 Who has 192.168.100.2? Tell 192.168.100.3
  12 20.376119865 20:00:00:00:00:01  30:00:00:00:00:01  ARP      42 192.168.100.2 is at 20:00:00:00:00:01

▸ Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
```

（见 lab_2-4-client.pcapng）

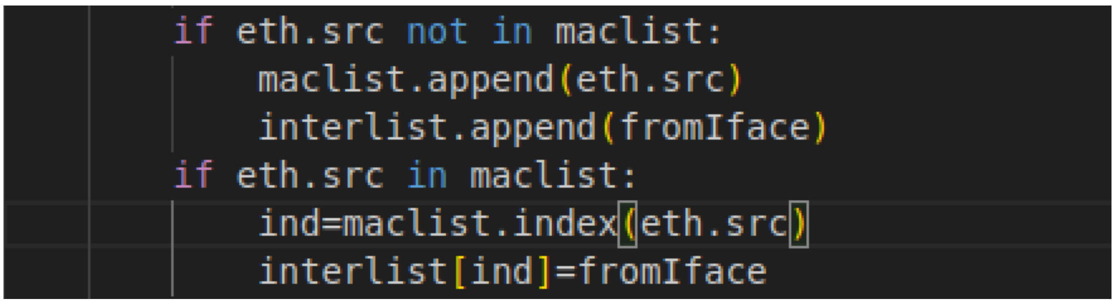
Client 抓包结果如下，也收到了两个 broadcast 信号，说明符合预期。

5. 核心代码

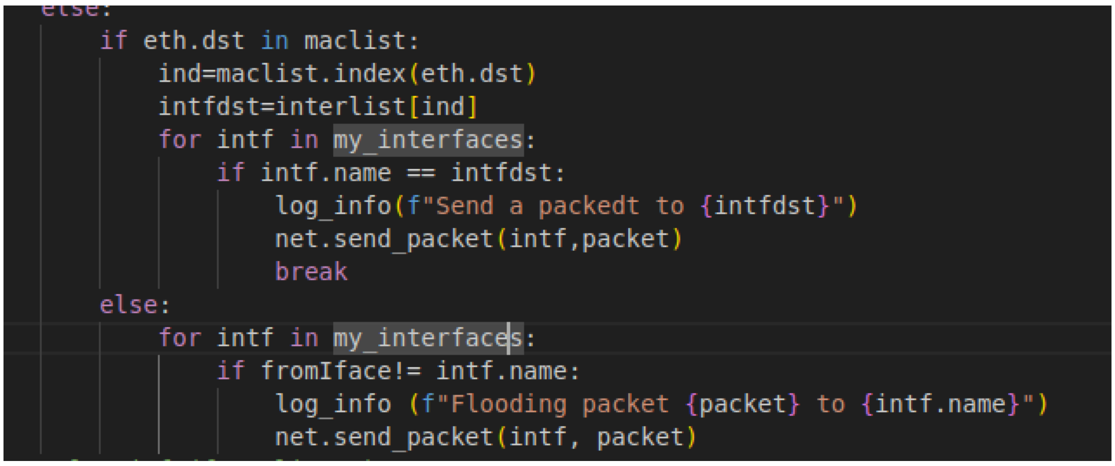
Task1 Task1 无核心代码

Task2 核心代码

修改 myswitch.py 如下

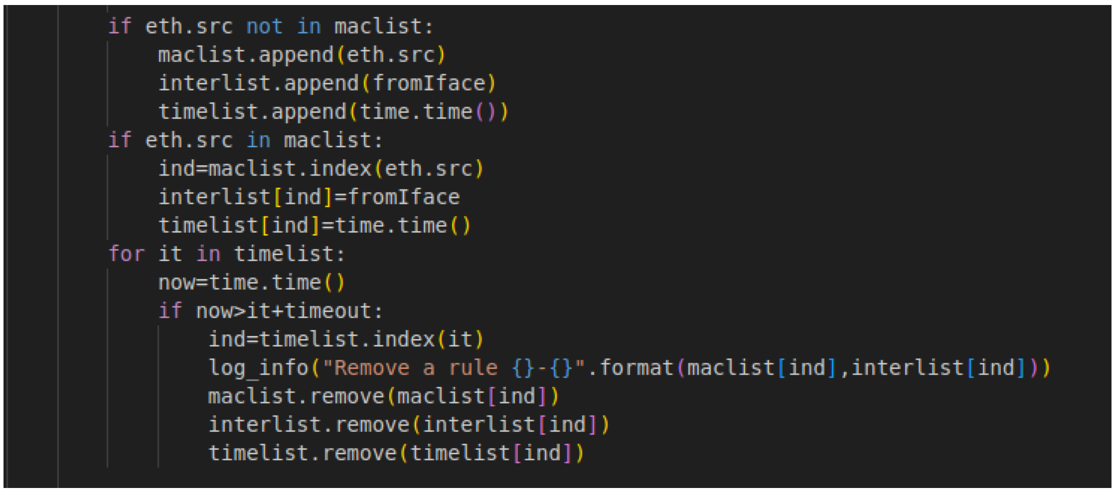


```python
        if eth.src not in maclist:
            maclist.append(eth.src)
            interlist.append(fromIface)
        if eth.src in maclist:
            ind=maclist.index(eth.src)
            interlist[ind]=fromIface
```



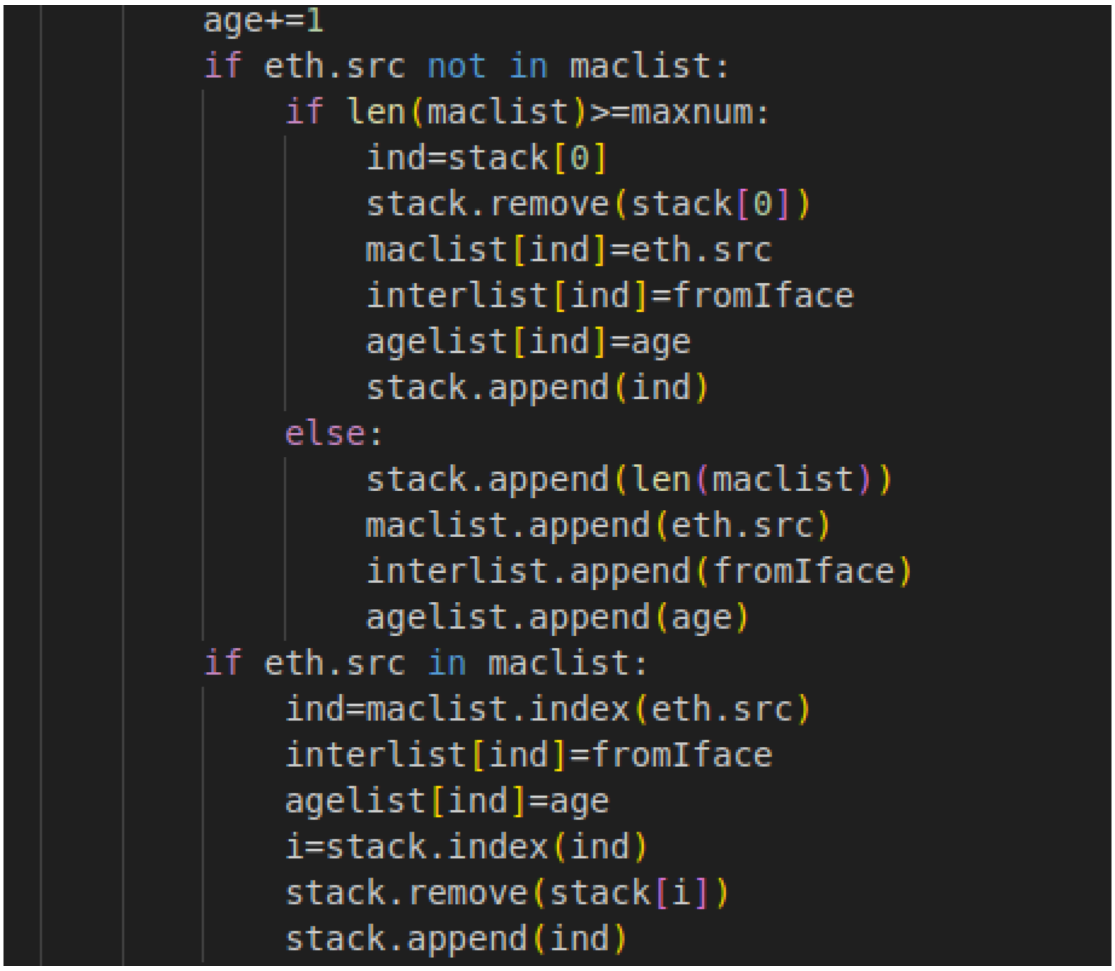
```
else:
    if eth.dst in maclist:
        ind=maclist.index(eth.dst)
        intfdst=interlist[ind]
        for intf in my_interfaces:
            if intf.name == intfdst:
                log_info(f"Send a packedt to {intfdst}")
                net.send_packet(intf,packet)
                break
    else:
        for intf in my_interfaces:
            if fromIface!= intf.name:
                log_info (f"Flooding packet {packet} to {intf.name}")
                net.send_packet(intf, packet)
```

Task3 修改 myswitch_to.py 如下



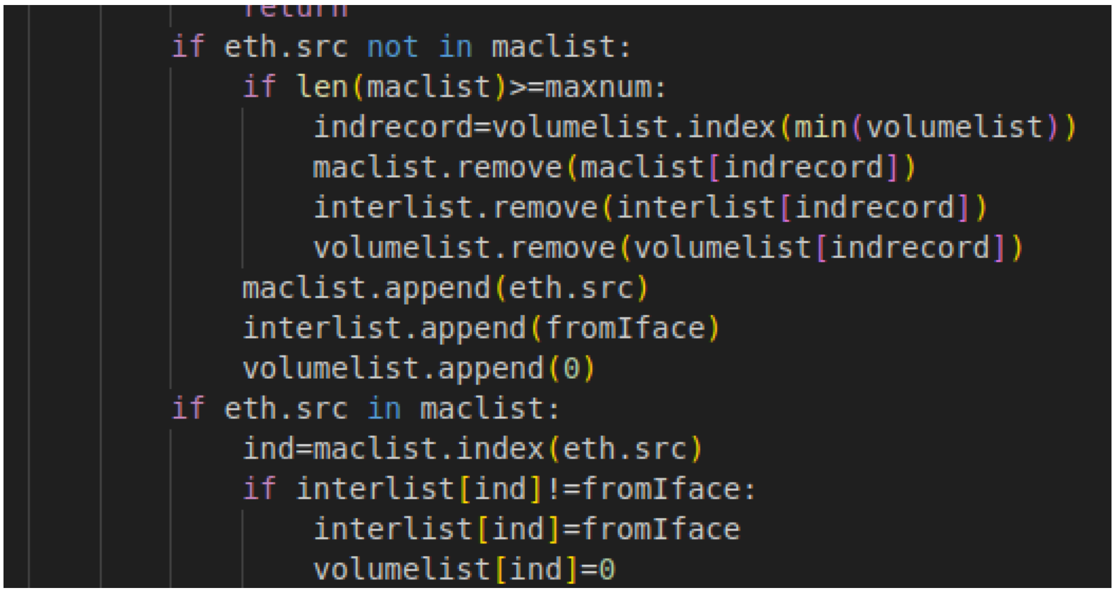
```
if eth.src not in maclist:
    maclist.append(eth.src)
    interlist.append(fromIface)
    timelist.append(time.time())
if eth.src in maclist:
    ind=maclist.index(eth.src)
    interlist[ind]=fromIface
    timelist[ind]=time.time()
for it in timelist:
    now=time.time()
    if now>it+timeout:
        ind=timelist.index(it)
        log_info("Remove a rule {}-{}".format(maclist[ind],interlist[ind]))
        maclist.remove(maclist[ind])
        interlist.remove(interlist[ind])
        timelist.remove(timelist[ind])
```

Task4 修改 myswitch_lru.py 如下

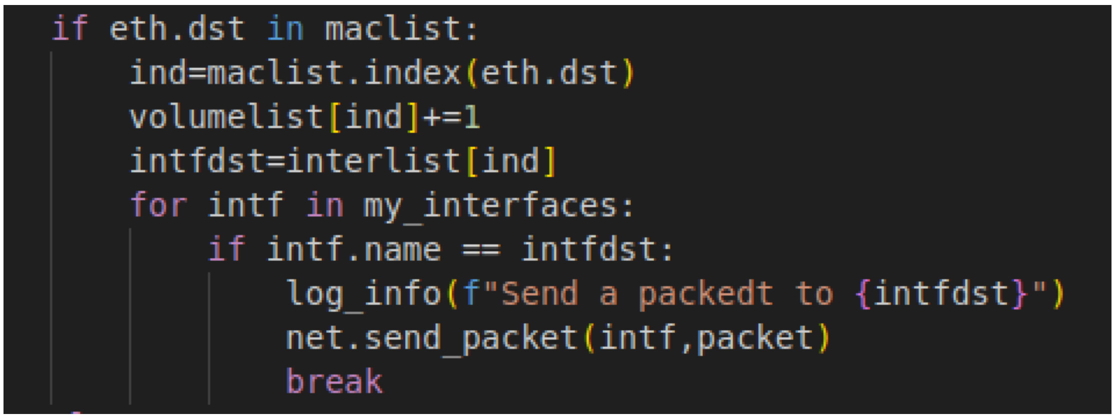


```python
            age+=1
            if eth.src not in maclist:
                if len(maclist)>=maxnum:
                    ind=stack[0]
                    stack.remove(stack[0])
                    maclist[ind]=eth.src
                    interlist[ind]=fromIface
                    agelist[ind]=age
                    stack.append(ind)
                else:
                    stack.append(len(maclist))
                    maclist.append(eth.src)
                    interlist.append(fromIface)
                    agelist.append(age)
            if eth.src in maclist:
                ind=maclist.index(eth.src)
                interlist[ind]=fromIface
                agelist[ind]=age
                i=stack.index(ind)
                stack.remove(stack[i])
                stack.append(ind)
```

Task5 修改 myswitch_traffic.py 如下



```python
                return
            if eth.src not in maclist:
                if len(maclist)>=maxnum:
                    indrecord=volumelist.index(min(volumelist))
                    maclist.remove(maclist[indrecord])
                    interlist.remove(interlist[indrecord])
                    volumelist.remove(volumelist[indrecord])
                maclist.append(eth.src)
                interlist.append(fromIface)
                volumelist.append(0)
            if eth.src in maclist:
                ind=maclist.index(eth.src)
                if interlist[ind]!=fromIface:
                    interlist[ind]=fromIface
                    volumelist[ind]=0
```



```python
if eth.dst in maclist:
    ind=maclist.index(eth.dst)
    volumelist[ind]+=1
    intfdst=interlist[ind]
    for intf in my_interfaces:
        if intf.name == intfdst:
            log_info(f"Send a packedt to {intfdst}")
            net.send_packet(intf,packet)
            break
```

6. 思考与感受

这一次实验为我们更详细地介绍了 switch 的学习机制，让我们对交换机的工作原理有了更深刻的认识。