南京大学本科生实验报告

课程名称：**计算机网络**　　　　任课教师：田臣/李文中　　　　助教：洋溢

学院	**计算机**	专业（方向）	**计算机科学与技术**
学号	**221220142**	姓名	**欧阳瑞泽**
Email	**221220142@smail.nju.edu.cn**	开始/完成日期	**2024.3.22**

Lab 2: Learning Switch

1. 实验目的

根据代码框架，完成 switchyard 代码，实现 **Timeouts**、**Least Recently Used**、**Least Traffic Volume** 三种不同的转发表维护机制。模拟学习交换机的核心功能。

2. 实验内容

Step 1: Basic Switch

通过阅读手册，得知 switch 的基本转发原理：当受到一个数据包，即可得知与发出地址相连的端口，解析packet的header，得到目标地址的MAC。Switch会维护一个转发表记录地址与对应的发送端口，若 dst 在表中，找到相应端口后发送，否则向所有不同于 src 的端口发送。

Step 2:Timeouts&Deploying

实现超时机制：在 Step1 转发表的基础上多维护一个时间戳，每次收到数据包时，用当前时间与转发表中的时间戳相比较，删除超时的转发表项。然后判断当前 src 的 MAC 与端口映射是否一致，若不一致则更新，并更新时间戳。

Step 3:Least Recently Used&Deploying

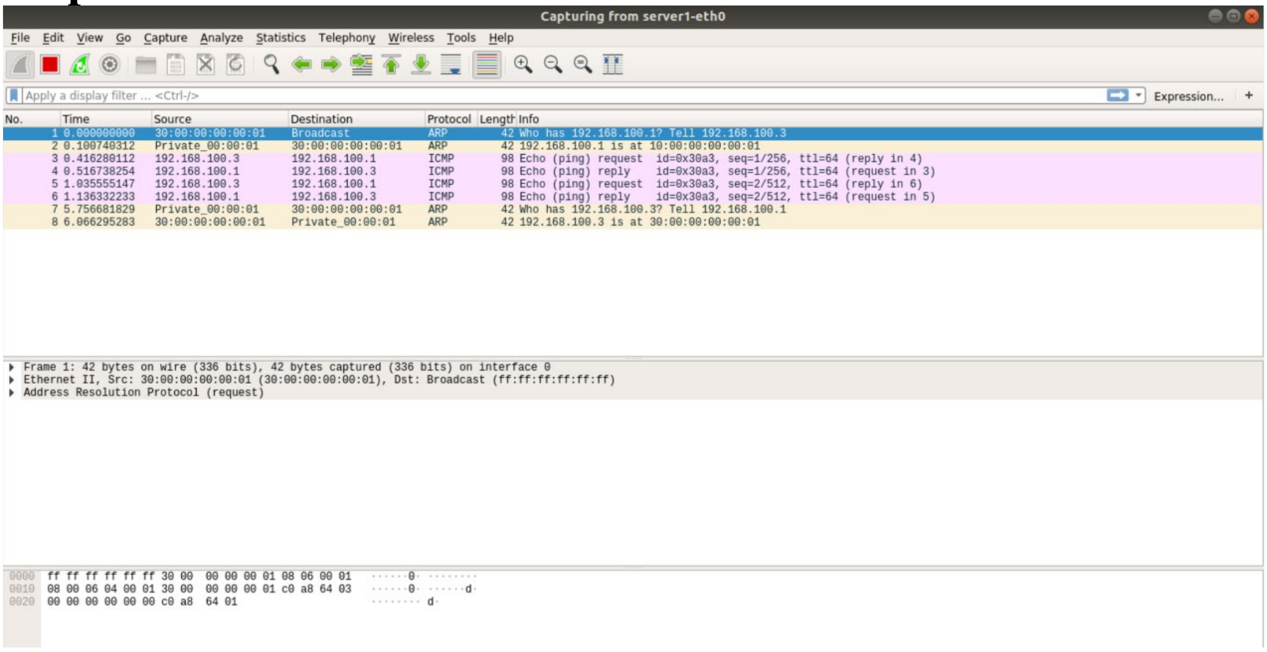
实现 LRU 机制：仍然使用类似 Step2 中的方法，给每个转发表项打上时间戳，每次超过最大容量需要删除表项时，O(n)遍历表项找出时间戳最小，即最早没被访问的表项删除。用链表实现应该可以优化到 O(1)查找和删除。

Step 4: Least Traffic Volume&Deploying

实现机制：使用类似 Step3 中的方法，给每个转发表项加上 volume 变量，每次超过最大容量需要删除表项时，O(n)遍历表项找出 volume 最小的表项删除。用链表实现应该可以优化到 O(1)查找和删除。最后在转发数据包时，若不是广播，就将目的表项的 volume 加一。

3. 实验结果

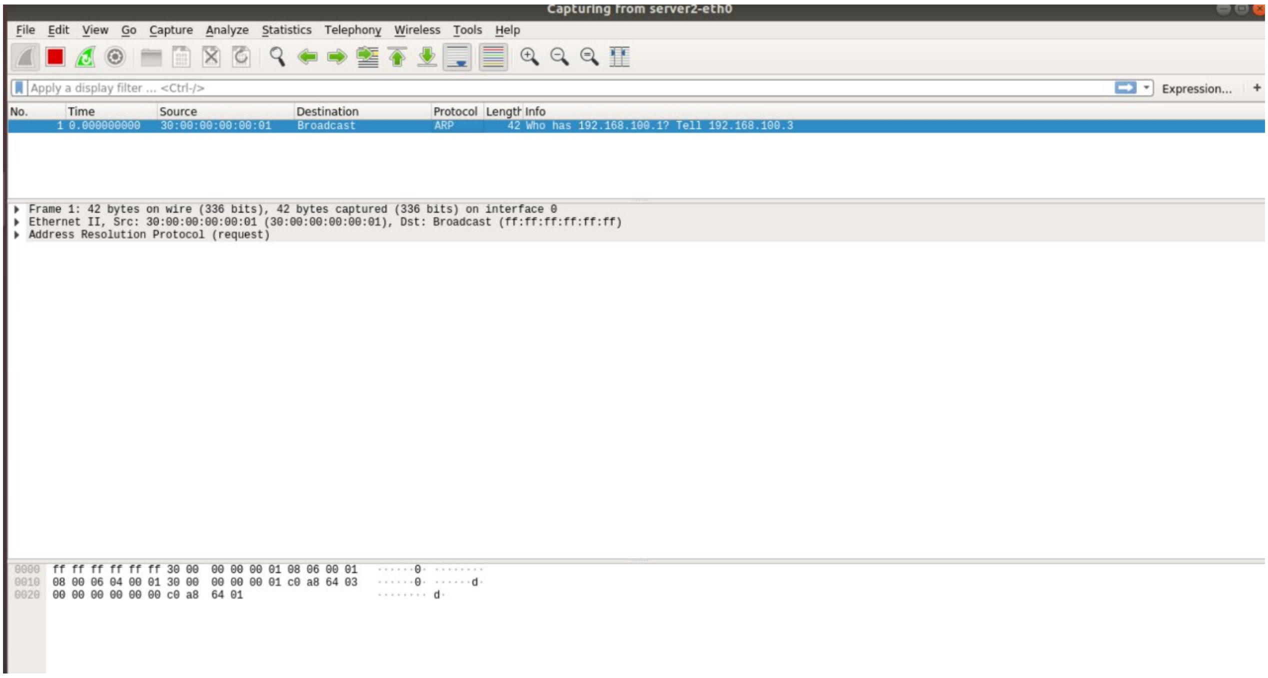
Step 1: Basic Switch



```
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 0.100740312  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 0.416280112  192.168.100.3       192.168.100.1       ICMP       98 Echo (ping) request  id=0x30a3, seq=1/256, ttl=64 (reply in 4)
     4 0.516738254  192.168.100.1       192.168.100.3       ICMP       98 Echo (ping) reply    id=0x30a3, seq=1/256, ttl=64 (request in 3)
     5 1.035555147  192.168.100.3       192.168.100.1       ICMP       98 Echo (ping) request  id=0x30a3, seq=2/512, ttl=64 (reply in 6)
     6 1.136332233  192.168.100.1       192.168.100.3       ICMP       98 Echo (ping) reply    id=0x30a3, seq=2/512, ttl=64 (request in 5)
     7 5.756681829  Private_00:00:01    30:00:00:00:00:01   ARP        42 Who has 192.168.100.3? Tell 192.168.100.1
     8 6.066295283  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)
```

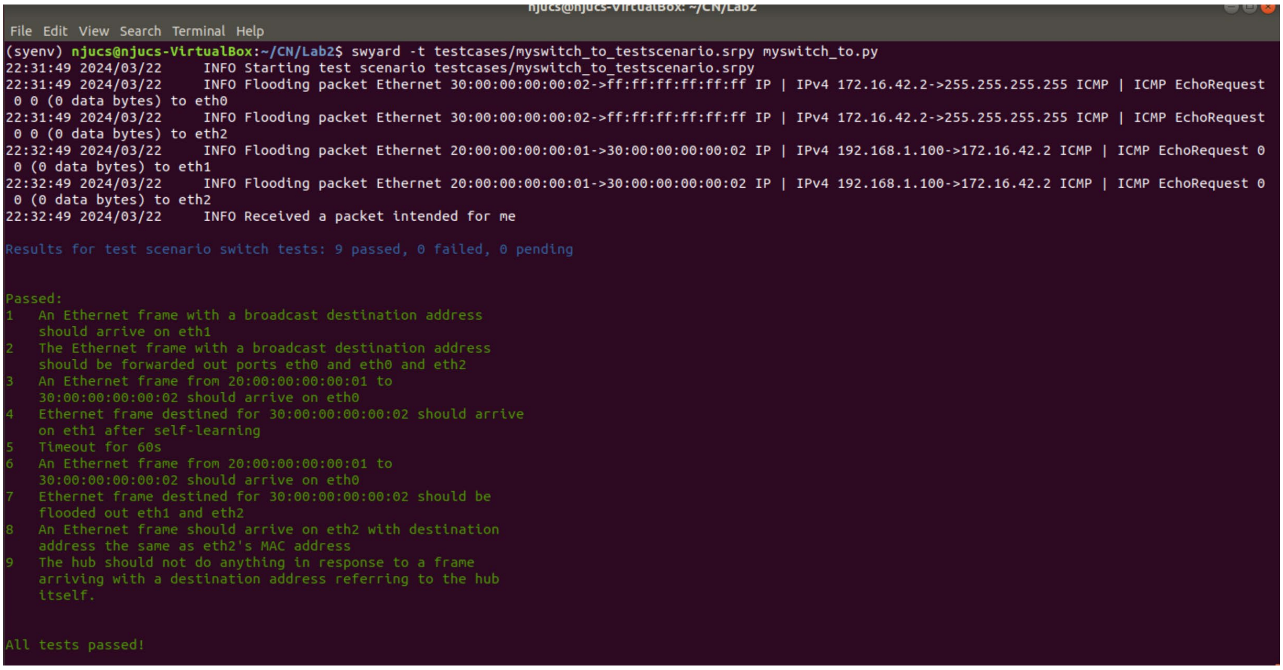
```
0000  ff ff ff ff ff ff 30 00  00 00 00 01 08 06 00 01   ······0· ········
0010  08 00 06 04 00 01 30 00  00 00 00 01 c0 a8 64 03   ······0· ······d·
0020  00 00 00 00 00 00 c0 a8  64 01                     ········ d·
```



```
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

▸ 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·
```

图一为 server1 的记录，图二为 server2 的记录，可见 client 在输入 ping server1 指令后，由于不知道 server1 的 MAC 地址，所以先在局域网中广播，switch 知道了 client 相连的端口，同时 server1 和 server2 收到了 ARP 类型的数据包，server1 回应时，switch 知道与 server1 相连的端口，同时向 client 转发，因此 server2 不会收到第二个数据包。之后两次 echo request 和 reply 以及一次广播，由于 switch 知道 client 与 server1 的相连端口，所以 server2 同样不会收到数据包。

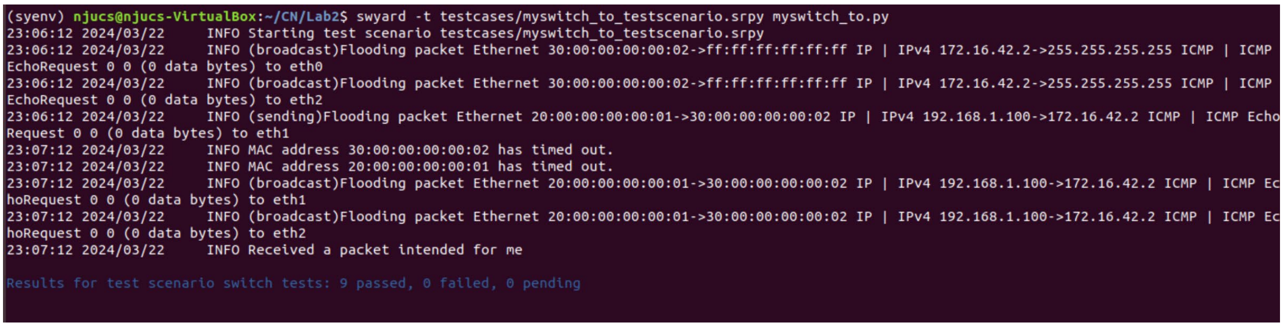
Step 2:Timeouts&Deploying



njucs@njucs-VirtualBox: ~/CN/Lab2
File Edit View Search Terminal Help

```
(syenv) njucs@njucs-VirtualBox:~/CN/Lab2$ swyard -t testcases/myswitch_to_testscenario.srpy myswitch_to.py
22:31:49 2024/03/22     INFO Starting test scenario testcases/myswitch_to_testscenario.srpy
22:31:49 2024/03/22     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
22:31:49 2024/03/22     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
22:32:49 2024/03/22     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 data bytes) to eth1
22:32:49 2024/03/22     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 data bytes) to eth2
22:32:49 2024/03/22     INFO Received a packet intended for me

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


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!
```

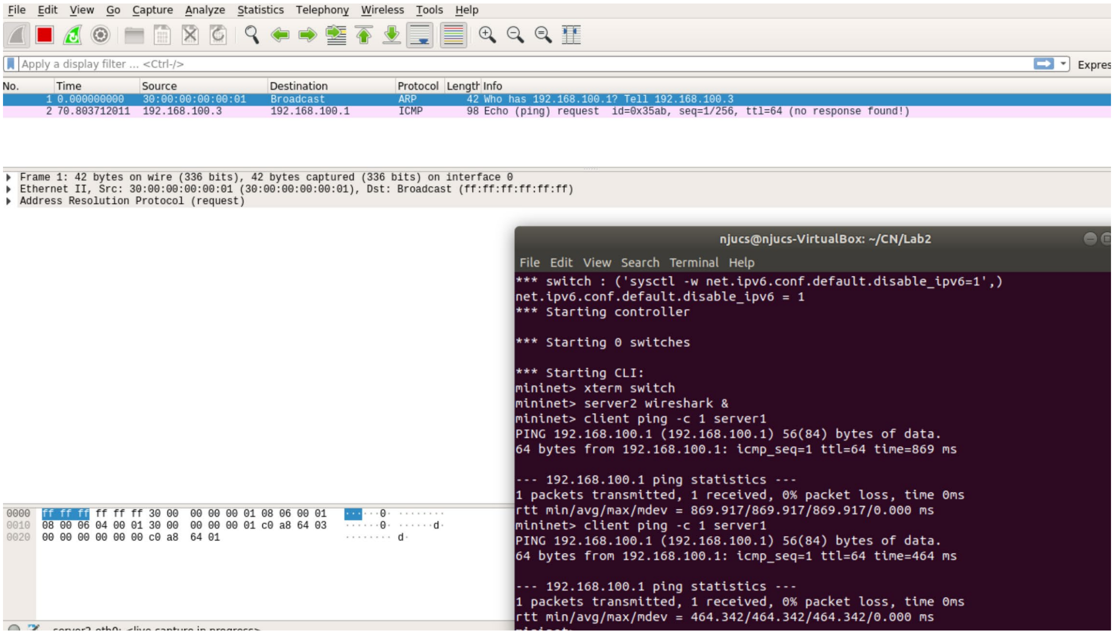
修改了输出日志后：



```
(syenv) njucs@njucs-VirtualBox:~/CN/Lab2$ swyard -t testcases/myswitch_to_testscenario.srpy myswitch_to.py
23:06:12 2024/03/22     INFO Starting test scenario testcases/myswitch_to_testscenario.srpy
23:06:12 2024/03/22     INFO (broadcast)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
23:06:12 2024/03/22     INFO (broadcast)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
23:06:12 2024/03/22     INFO (sending)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 Echo
Request 0 0 (0 data bytes) to eth1
23:07:12 2024/03/22     INFO MAC address 30:00:00:00:00:02 has timed out.
23:07:12 2024/03/22     INFO MAC address 20:00:00:00:00:01 has timed out.
23:07:12 2024/03/22     INFO (broadcast)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 Ec
hoRequest 0 0 (0 data bytes) to eth1
23:07:12 2024/03/22     INFO (broadcast)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 Ec
hoRequest 0 0 (0 data bytes) to eth2
23:07:12 2024/03/22     INFO Received a packet intended for me

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

可见在 23：06 等待 60s 后，23：07 时 switch 清除了超时的两个转发表项，并将之后的转发变成了广播。

接着在 mininet 中运行 switch 以验证超时机制。


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
Expres

```
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 70.803712011 192.168.100.3      192.168.100.1    ICMP     98 Echo (ping) request  id=0x35ab, seq=1/256, ttl=64 (no response found!)
```

```
▸ 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·
```

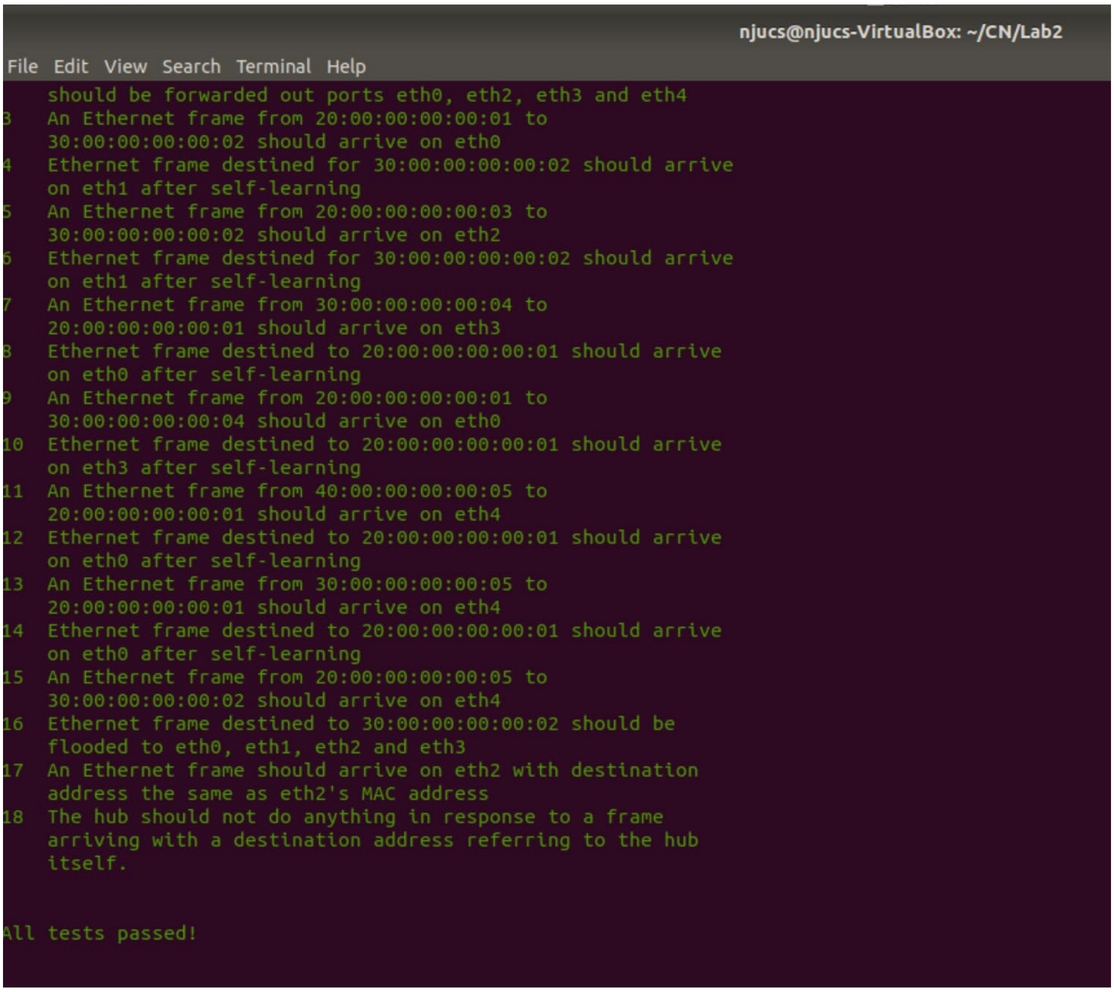
server2-eth0: <live capture in progress>

njucs@njucs-VirtualBox: ~/CN/Lab2

```
*** switch : ('sysctl -w net.ipv6.conf.default.disable_ipv6=1',)
net.ipv6.conf.default.disable_ipv6 = 1
*** Starting controller

*** Starting 0 switches

*** Starting CLI:
mininet> xterm switch
mininet> server2 wireshark &
mininet> client ping -c 1 server1
PING 192.168.100.1 (192.168.100.1) 56(84) bytes of data.
64 bytes from 192.168.100.1: icmp_seq=1 ttl=64 time=869 ms

--- 192.168.100.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 869.917/869.917/869.917/0.000 ms
mininet> client ping -c 1 server1
PING 192.168.100.1 (192.168.100.1) 56(84) bytes of data.
64 bytes from 192.168.100.1: icmp_seq=1 ttl=64 time=464 ms

--- 192.168.100.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 464.342/464.342/464.342/0.000 ms
```

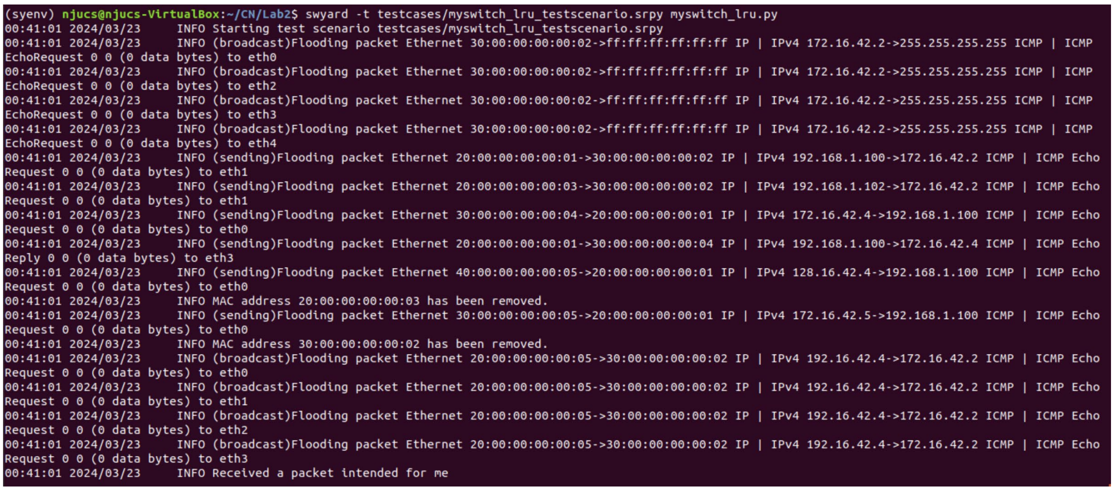
用 wireshark 监测 server2。Client 两次 ping server1，间隔大于 10s。可见在第二次 ping 时，由于 switch 清除了超时的 server1 表项，导致 server2 收到了被 switch 广播的本该发给 server1 的 ICMP 数据包。可以验证 switch_to 正确。

Step 3:Least Recently Used&Deploying


njucs@njucs-VirtualBox: ~/CN/Lab2
File Edit View Search Terminal Help

```
   should be forwarded out ports eth0, eth2, eth3 and eth4
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  An Ethernet frame from 20:00:00:00:00:03 to
   30:00:00:00:00:02 should arrive on eth2
6  Ethernet frame destined for 30:00:00:00:00:02 should arrive
   on eth1 after self-learning
7  An Ethernet frame from 30:00:00:00:00:04 to
   20:00:00:00:00:01 should arrive on eth3
8  Ethernet frame destined to 20:00:00:00:00:01 should arrive
   on eth0 after self-learning
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!
```

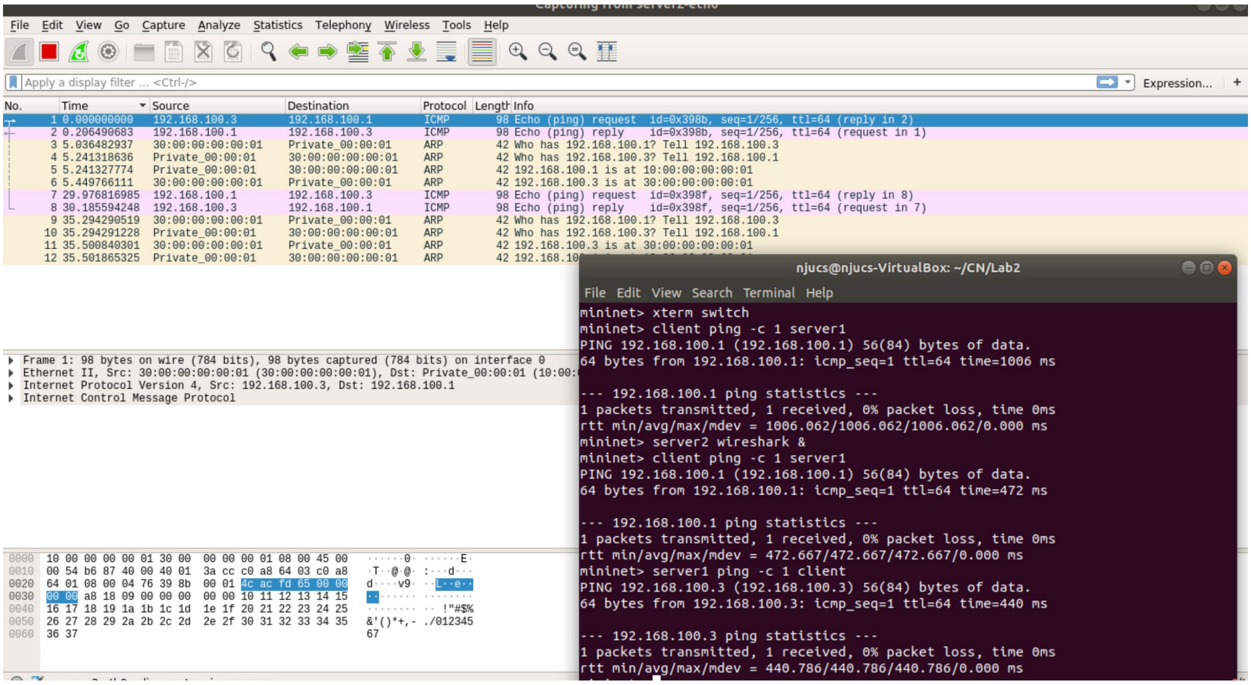
修改了输出日志后：



```
(syenv) njucs@njucs-VirtualBox:~/CN/Lab2$ swyard -t testcases/myswitch_lru_testscenario.srpy myswitch_lru.py
00:41:01 2024/03/23     INFO Starting test scenario testcases/myswitch_lru_testscenario.srpy
00:41:01 2024/03/23     INFO (broadcast)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
00:41:01 2024/03/23     INFO (broadcast)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
00:41:01 2024/03/23     INFO (broadcast)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
00:41:01 2024/03/23     INFO (broadcast)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
00:41:01 2024/03/23     INFO (sending)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 Echo
Request 0 0 (0 data bytes) to eth1
00:41:01 2024/03/23     INFO (sending)Flooding 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 Echo
Request 0 0 (0 data bytes) to eth1
00:41:01 2024/03/23     INFO (sending)Flooding 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 Echo
Request 0 0 (0 data bytes) to eth0
00:41:01 2024/03/23     INFO (sending)Flooding 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 Echo
Reply 0 0 (0 data bytes) to eth3
00:41:01 2024/03/23     INFO (sending)Flooding 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 Echo
Request 0 0 (0 data bytes) to eth0
00:41:01 2024/03/23     INFO MAC address 20:00:00:00:00:03 has been removed.
00:41:01 2024/03/23     INFO (sending)Flooding 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 Echo
Request 0 0 (0 data bytes) to eth0
00:41:01 2024/03/23     INFO MAC address 30:00:00:00:00:02 has been removed.
00:41:01 2024/03/23     INFO (broadcast)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 Echo
Request 0 0 (0 data bytes) to eth0
00:41:01 2024/03/23     INFO (broadcast)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 Echo
Request 0 0 (0 data bytes) to eth1
00:41:01 2024/03/23     INFO (broadcast)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 Echo
Request 0 0 (0 data bytes) to eth2
00:41:01 2024/03/23     INFO (broadcast)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 Echo
Request 0 0 (0 data bytes) to eth3
00:41:01 2024/03/23     INFO Received a packet intended for me
```

可见在储存了 5 个表项后，日志输出 00:41:01 2024/03/23 INFO MAC address 20:00:00:00:00:03 has been removed. 随后 30:00:00:00:00:05 加入。下一次同理。

接着在 mininet 中运行 switch 以验证 LRU 机制。由于拓扑结构以及为了检验方便，将表项最大个数改为 1。


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...

```
No.  Time          Source             Destination        Protocol Length Info
   1 0.000000000   192.168.100.3      192.168.100.1      ICMP      98 Echo (ping) request  id=0x398b, seq=1/256, ttl=64 (reply in 2)
   2 0.206490683   192.168.100.1      192.168.100.3      ICMP      98 Echo (ping) reply    id=0x398b, seq=1/256, ttl=64 (request in 1)
   3 5.036482937   30:00:00:00:00:01  Private_00:00:01   ARP       42 Who has 192.168.100.1? Tell 192.168.100.3
   4 5.241318636   Private_00:00:01   30:00:00:00:00:01  ARP       42 Who has 192.168.100.3? Tell 192.168.100.1
   5 5.241327774   Private_00:00:01   30:00:00:00:00:01  ARP       42 192.168.100.1 is at 10:00:00:00:00:01
   6 5.449766111   30:00:00:00:00:01  Private_00:00:01   ARP       42 192.168.100.3 is at 30:00:00:00:00:01
   7 29.976816985  192.168.100.1      192.168.100.3      ICMP      98 Echo (ping) request  id=0x398f, seq=1/256, ttl=64 (reply in 8)
   8 30.185594248  192.168.100.3      192.168.100.1      ICMP      98 Echo (ping) reply    id=0x398f, seq=1/256, ttl=64 (request in 7)
   9 35.294290519  30:00:00:00:00:01  Private_00:00:01   ARP       42 Who has 192.168.100.1? Tell 192.168.100.3
  10 35.294291228  Private_00:00:01   30:00:00:00:00:01  ARP       42 Who has 192.168.100.3? Tell 192.168.100.1
  11 35.500840301  30:00:00:00:00:01  Private_00:00:01   ARP       42 192.168.100.3 is at 30:00:00:00:00:01
  12 35.501865325  Private_00:00:01   30:00:00:00:00:01  ARP       42 192.168.10
```

```
▸ Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
▸ Ethernet II, Src: 30:00:00:00:00:01 (30:00:00:00:00:01), Dst: Private_00:00:01 (10:00:
▸ Internet Protocol Version 4, Src: 192.168.100.3, Dst: 192.168.100.1
▸ Internet Control Message Protocol
```

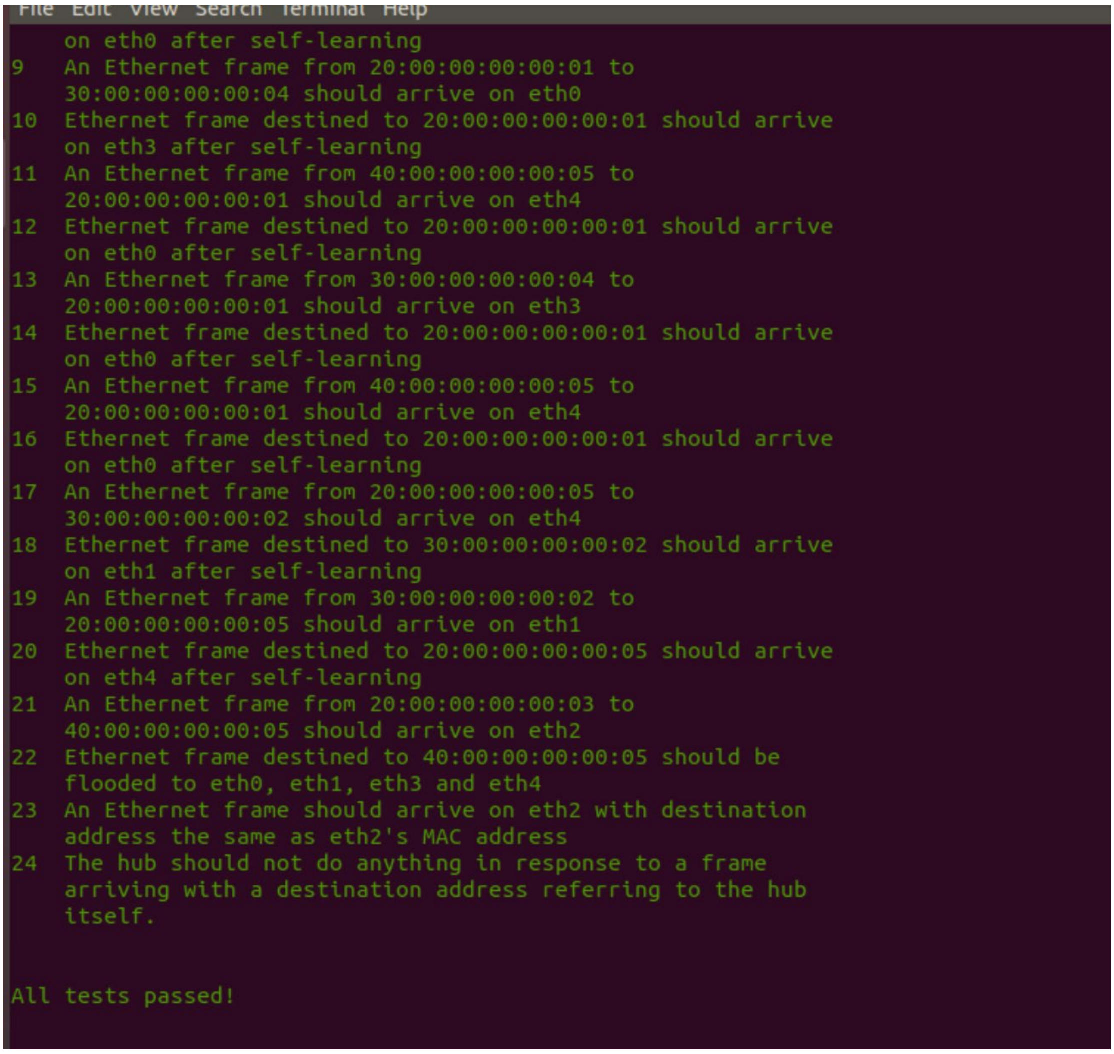
```
0000  10 00 00 00 00 01 30 00  00 00 00 01 08 00 45 00   ······0· ······E·
0010  00 54 b6 87 40 00 40 01  3a cc c0 a8 64 03 c0 a8   ·T··@·@· :···d···
0020  64 01 08 00 04 76 39 8b  00 01 4c ac fd 65 00 00   d····v9· ··L··e··
0030  00 00 a8 18 09 00 00 00  00 00 10 11 12 13 14 15   ········ ········
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ········ ·· !"#$%
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35   &'()*+,- ./012345
0060  36 37                                              67
```

njucs@njucs-VirtualBox: ~/CN/Lab2

```
mininet> xterm switch
mininet> client ping -c 1 server1
PING 192.168.100.1 (192.168.100.1) 56(84) bytes of data.
64 bytes from 192.168.100.1: icmp_seq=1 ttl=64 time=1006 ms

--- 192.168.100.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1006.062/1006.062/1006.062/0.000 ms
mininet> server2 wireshark &
mininet> client ping -c 1 server1
PING 192.168.100.1 (192.168.100.1) 56(84) bytes of data.
64 bytes from 192.168.100.1: icmp_seq=1 ttl=64 time=472 ms

--- 192.168.100.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 472.667/472.667/472.667/0.000 ms
mininet> server1 ping -c 1 client
PING 192.168.100.3 (192.168.100.3) 56(84) bytes of data.
64 bytes from 192.168.100.3: icmp_seq=1 ttl=64 time=440 ms

--- 192.168.100.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 440.786/440.786/440.786/0.000 ms
```

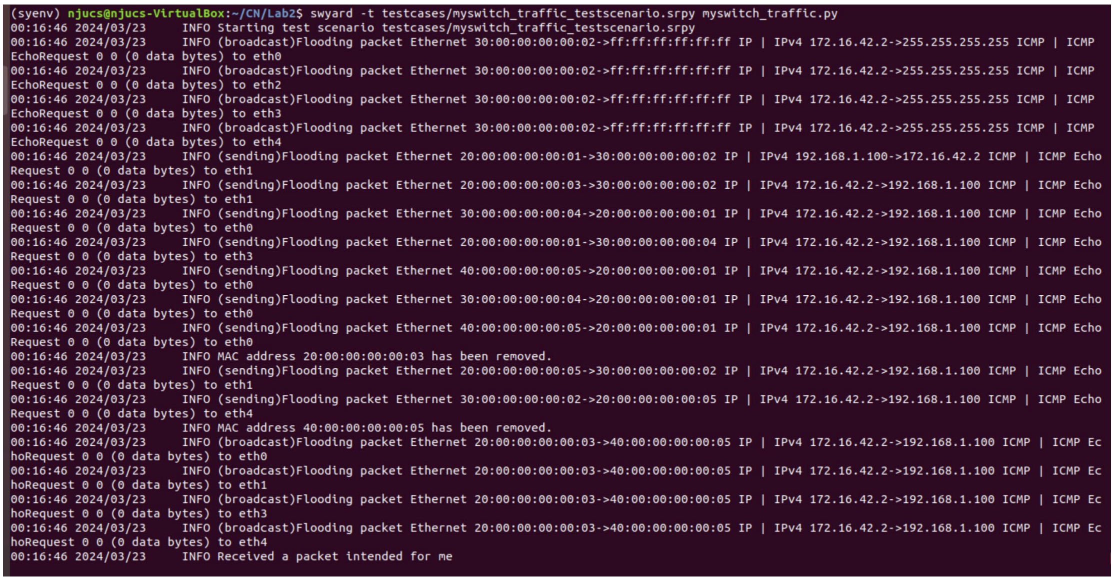
用 wireshark 监测 server2。先用 client ping server1，再用 server1 ping client。可见第二次 ping 时 switch 清除了 client 的表项，数据包变为广播，导致 server2 收到数据包。可以验证 switch_lru 正确。

Step 4: Least Traffic Volume&Deploying



```
     on eth0 after self-learning
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 30:00:00:00:00:04 to
     20:00:00:00:00:01 should arrive on eth3
14   Ethernet frame destined to 20:00:00:00:00:01 should arrive
     on eth0 after self-learning
15   An Ethernet frame from 40:00:00:00:00:05 to
     20:00:00:00:00:01 should arrive on eth4
16   Ethernet frame destined to 20:00:00:00:00:01 should arrive
     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!
```

修改了输出日志后：

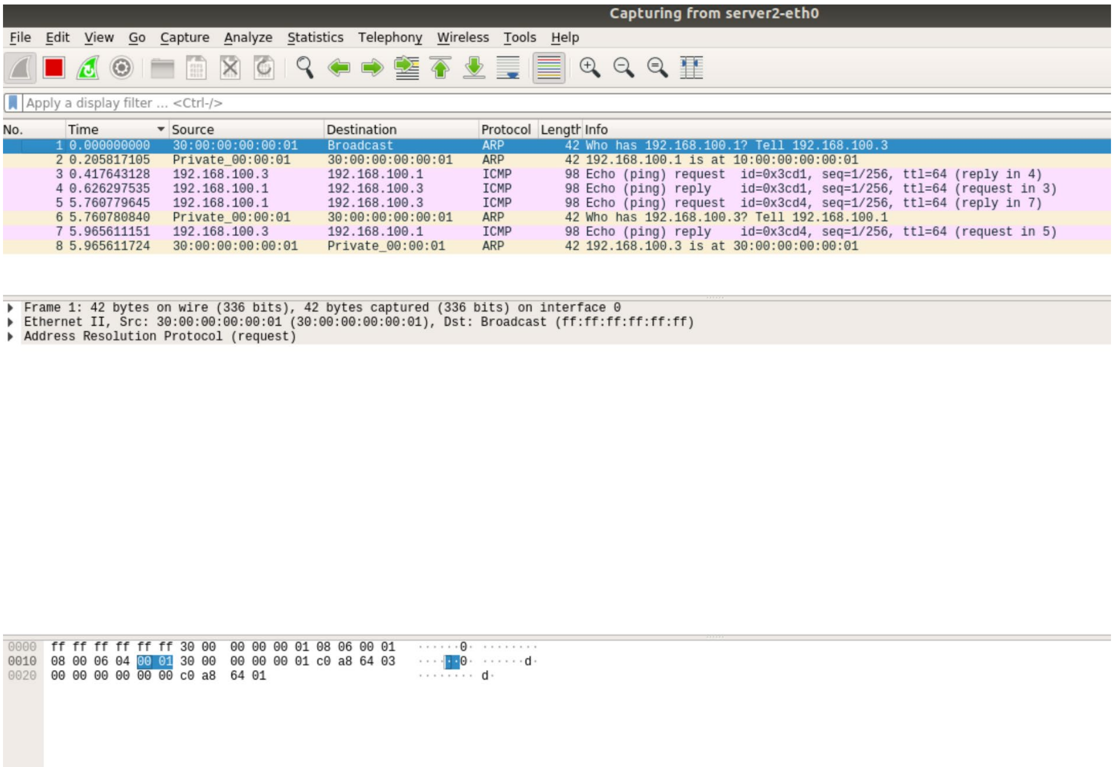


```
(syenv) njucs@njucs-VirtualBox:~/CN/Lab2$ swyard -t testcases/myswitch_traffic_testscenario.srpy myswitch_traffic.py
00:16:46 2024/03/23     INFO Starting test scenario testcases/myswitch_traffic_testscenario.srpy
00:16:46 2024/03/23     INFO (broadcast)Flooding packet Ethernet 30:00:00:00:00:02->ff:ff:ff:ff:ff:ff IP | IPv4 172.16.42.2->255.255.255.255 ICMP | ICMP
EchoRequest 0 0 (0 data bytes) to eth0
00:16:46 2024/03/23     INFO (broadcast)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
00:16:46 2024/03/23     INFO (broadcast)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
00:16:46 2024/03/23     INFO (broadcast)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
00:16:46 2024/03/23     INFO (sending)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 Echo
Request 0 0 (0 data bytes) to eth1
00:16:46 2024/03/23     INFO (sending)Flooding packet Ethernet 20:00:00:00:00:03->30:00:00:00:00:02 IP | IPv4 172.16.42.2->192.168.1.100 ICMP | ICMP Echo
Request 0 0 (0 data bytes) to eth1
00:16:46 2024/03/23     INFO (sending)Flooding packet Ethernet 30:00:00:00:00:04->20:00:00:00:00:01 IP | IPv4 172.16.42.2->192.168.1.100 ICMP | ICMP Echo
Request 0 0 (0 data bytes) to eth0
00:16:46 2024/03/23     INFO (sending)Flooding packet Ethernet 20:00:00:00:00:01->30:00:00:00:00:04 IP | IPv4 172.16.42.2->192.168.1.100 ICMP | ICMP Echo
Request 0 0 (0 data bytes) to eth3
00:16:46 2024/03/23     INFO (sending)Flooding packet Ethernet 40:00:00:00:00:05->20:00:00:00:00:01 IP | IPv4 172.16.42.2->192.168.1.100 ICMP | ICMP Echo
Request 0 0 (0 data bytes) to eth0
00:16:46 2024/03/23     INFO (sending)Flooding packet Ethernet 30:00:00:00:00:04->20:00:00:00:00:01 IP | IPv4 172.16.42.2->192.168.1.100 ICMP | ICMP Echo
Request 0 0 (0 data bytes) to eth0
00:16:46 2024/03/23     INFO (sending)Flooding packet Ethernet 40:00:00:00:00:05->20:00:00:00:00:01 IP | IPv4 172.16.42.2->192.168.1.100 ICMP | ICMP Echo
Request 0 0 (0 data bytes) to eth0
00:16:46 2024/03/23     INFO MAC address 20:00:00:00:00:03 has been removed.
00:16:46 2024/03/23     INFO (sending)Flooding packet Ethernet 20:00:00:00:00:05->30:00:00:00:00:02 IP | IPv4 172.16.42.2->192.168.1.100 ICMP | ICMP Echo
Request 0 0 (0 data bytes) to eth1
00:16:46 2024/03/23     INFO (sending)Flooding packet Ethernet 30:00:00:00:00:02->20:00:00:00:00:05 IP | IPv4 172.16.42.2->192.168.1.100 ICMP | ICMP Echo
Request 0 0 (0 data bytes) to eth4
00:16:46 2024/03/23     INFO MAC address 40:00:00:00:00:05 has been removed.
00:16:46 2024/03/23     INFO (broadcast)Flooding packet Ethernet 20:00:00:00:00:03->40:00:00:00:00:05 IP | IPv4 172.16.42.2->192.168.1.100 ICMP | ICMP Ec
hoRequest 0 0 (0 data bytes) to eth0
00:16:46 2024/03/23     INFO (broadcast)Flooding packet Ethernet 20:00:00:00:00:03->40:00:00:00:00:05 IP | IPv4 172.16.42.2->192.168.1.100 ICMP | ICMP Ec
hoRequest 0 0 (0 data bytes) to eth1
00:16:46 2024/03/23     INFO (broadcast)Flooding packet Ethernet 20:00:00:00:00:03->40:00:00:00:00:05 IP | IPv4 172.16.42.2->192.168.1.100 ICMP | ICMP Ec
hoRequest 0 0 (0 data bytes) to eth3
00:16:46 2024/03/23     INFO (broadcast)Flooding packet Ethernet 20:00:00:00:00:03->40:00:00:00:00:05 IP | IPv4 172.16.42.2->192.168.1.100 ICMP | ICMP Ec
hoRequest 0 0 (0 data bytes) to eth4
00:16:46 2024/03/23     INFO Received a packet intended for me
```

可见在储存了 5 个表项过后，日志输出 00:16:46 2024/03/23 INFO MAC address

20:00:00:00:00:03 has been removed. 删除了volume 最小的 20:00:00:00:00:03。下一次同理。

接着在 mininet 中运行 switch 验证。
由于拓扑结构以及为了检验方便，将表项最大个数改为 1。



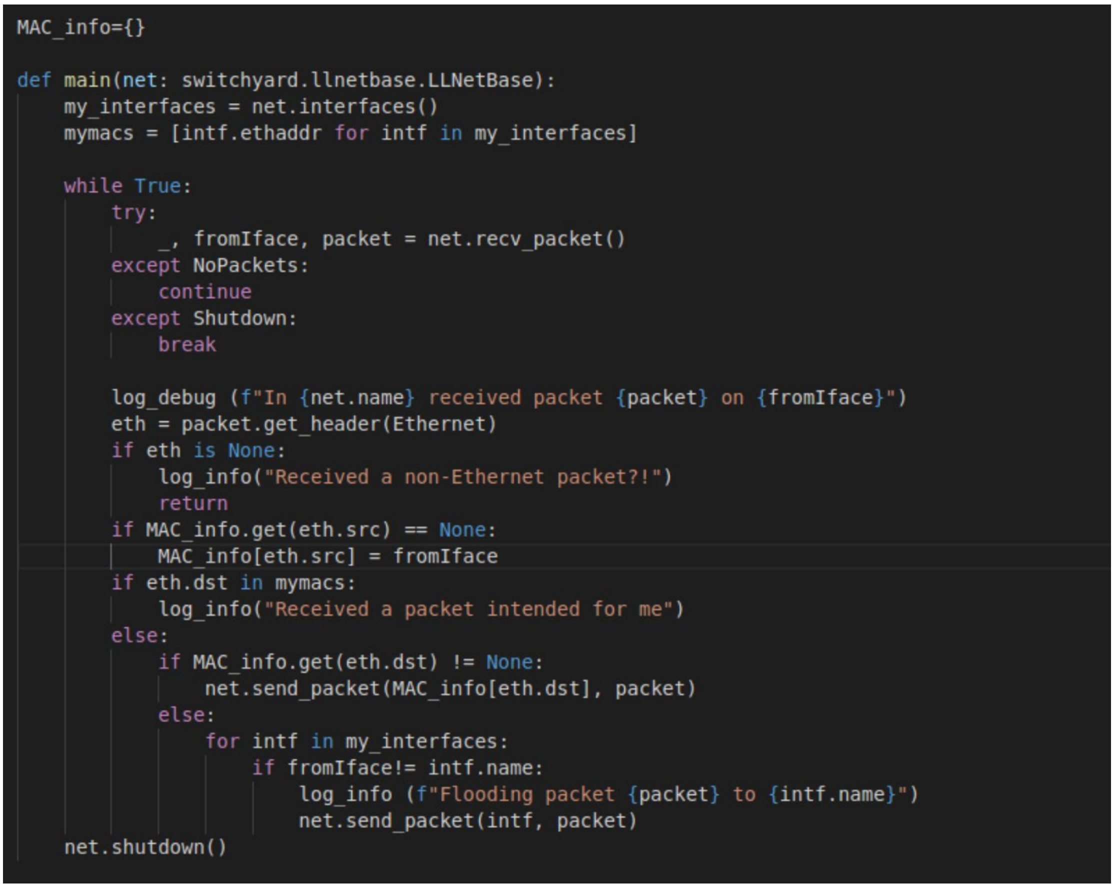
Capturing from server2-eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
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 0.205817105   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 0.417643128   192.168.100.3       192.168.100.1       ICMP       98 Echo (ping) request  id=0x3cd1, seq=1/256, ttl=64 (reply in 4)
    4 0.626297535   192.168.100.1       192.168.100.3       ICMP       98 Echo (ping) reply    id=0x3cd1, seq=1/256, ttl=64 (request in 3)
    5 5.760779645   192.168.100.1       192.168.100.3       ICMP       98 Echo (ping) request  id=0x3cd4, seq=1/256, ttl=64 (reply in 7)
    6 5.760780840   Private_00:00:01    30:00:00:00:00:01   ARP        42 Who has 192.168.100.3? Tell 192.168.100.1
    7 5.965611151   192.168.100.3       192.168.100.1       ICMP       98 Echo (ping) reply    id=0x3cd4, seq=1/256, ttl=64 (request in 5)
    8 5.965611724   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·
```

操作过程同Step3，可见类似Step3，server2 在第二次 server1 ping client 时，由于已经 switch 删除 client 的表项，所以 server2 收到了被广播的数据包。可以验证 switch_traffic 正确。

4. 核心代码

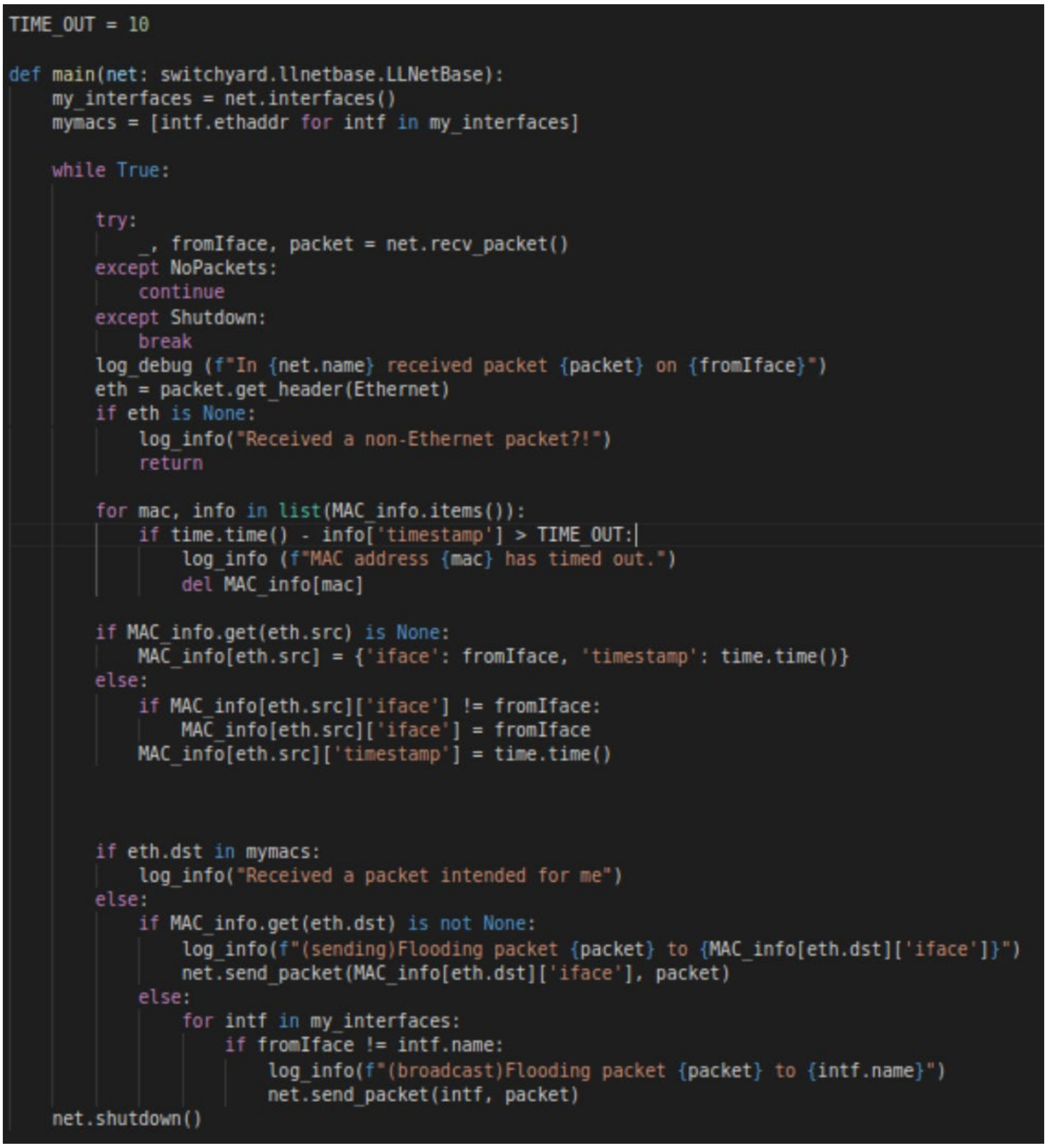
Step 1: Basic Switch



```
MAC_info={}

def main(net: switchyard.llnetbase.LLNetBase):
    my_interfaces = net.interfaces()
    mymacs = [intf.ethaddr for intf in my_interfaces]

    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)
        if eth is None:
            log_info("Received a non-Ethernet packet?!")
            return
        if MAC_info.get(eth.src) == None:
            MAC_info[eth.src] = fromIface
        if eth.dst in mymacs:
            log_info("Received a packet intended for me")
        else:
            if MAC_info.get(eth.dst) != None:
                net.send_packet(MAC_info[eth.dst], packet)
            else:
                for intf in my_interfaces:
                    if fromIface!= intf.name:
                        log_info (f"Flooding packet {packet} to {intf.name}")
                        net.send_packet(intf, packet)
    net.shutdown()
```

新建字典用于储存MAC地址与端口的映射关系。解析 header 后，首先判断来源的地址是否在字典内，然后判断目的地址。

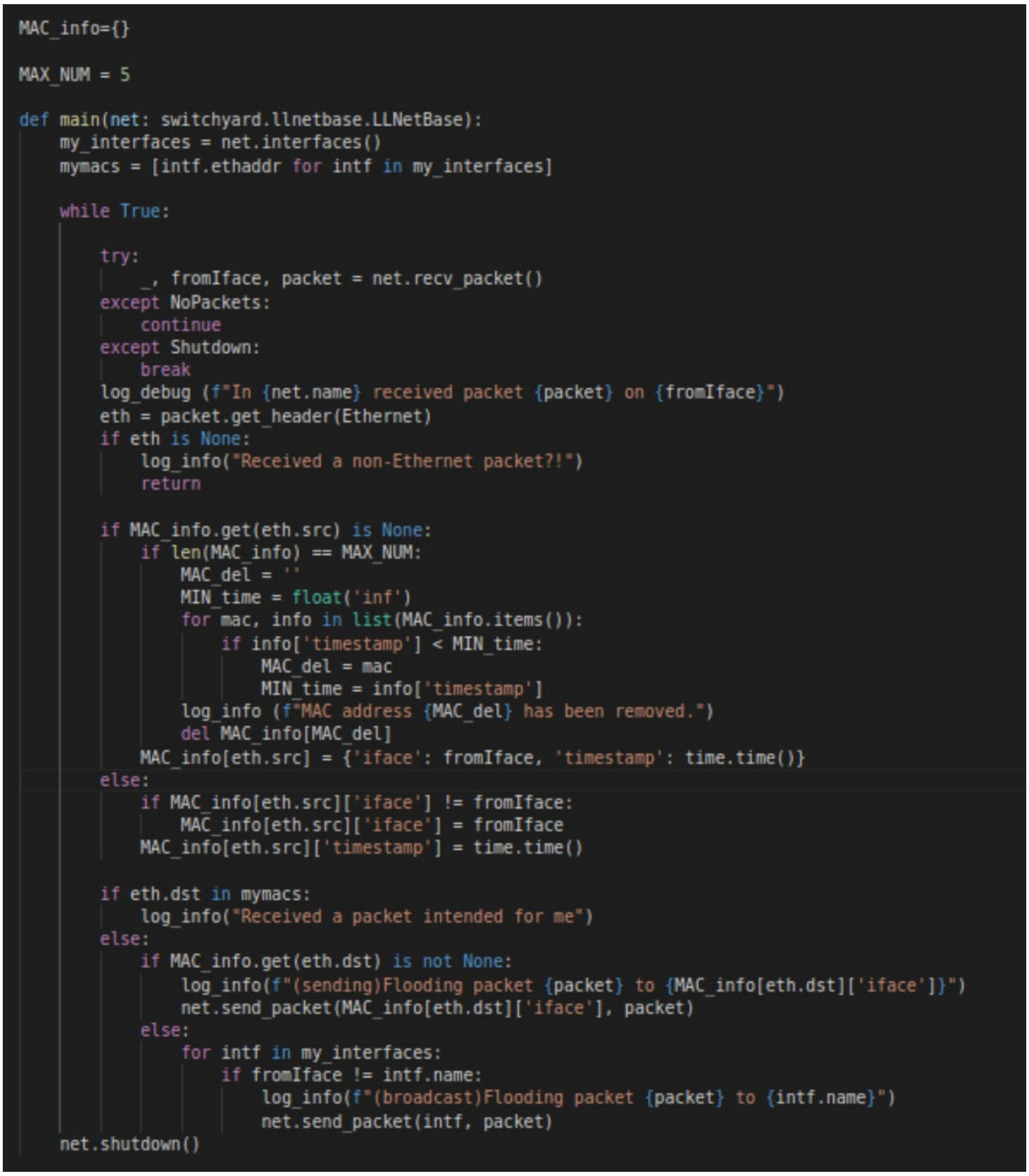
Step 2:Timeouts&Deploying



```python
TIME_OUT = 10

def main(net: switchyard.llnetbase.LLNetBase):
    my_interfaces = net.interfaces()
    mymacs = [intf.ethaddr for intf in my_interfaces]

    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)
        if eth is None:
            log_info("Received a non-Ethernet packet?!")
            return

        for mac, info in list(MAC_info.items()):
            if time.time() - info['timestamp'] > TIME_OUT:
                log_info (f"MAC address {mac} has timed out.")
                del MAC_info[mac]

        if MAC_info.get(eth.src) is None:
            MAC_info[eth.src] = {'iface': fromIface, 'timestamp': time.time()}
        else:
            if MAC_info[eth.src]['iface'] != fromIface:
                MAC_info[eth.src]['iface'] = fromIface
            MAC_info[eth.src]['timestamp'] = time.time()



        if eth.dst in mymacs:
            log_info("Received a packet intended for me")
        else:
            if MAC_info.get(eth.dst) is not None:
                log_info(f"(sending)Flooding packet {packet} to {MAC_info[eth.dst]['iface']}")
                net.send_packet(MAC_info[eth.dst]['iface'], packet)
            else:
                for intf in my_interfaces:
                    if fromIface != intf.name:
                        log_info(f"(broadcast)Flooding packet {packet} to {intf.name}")
                        net.send_packet(intf, packet)
    net.shutdown()
```

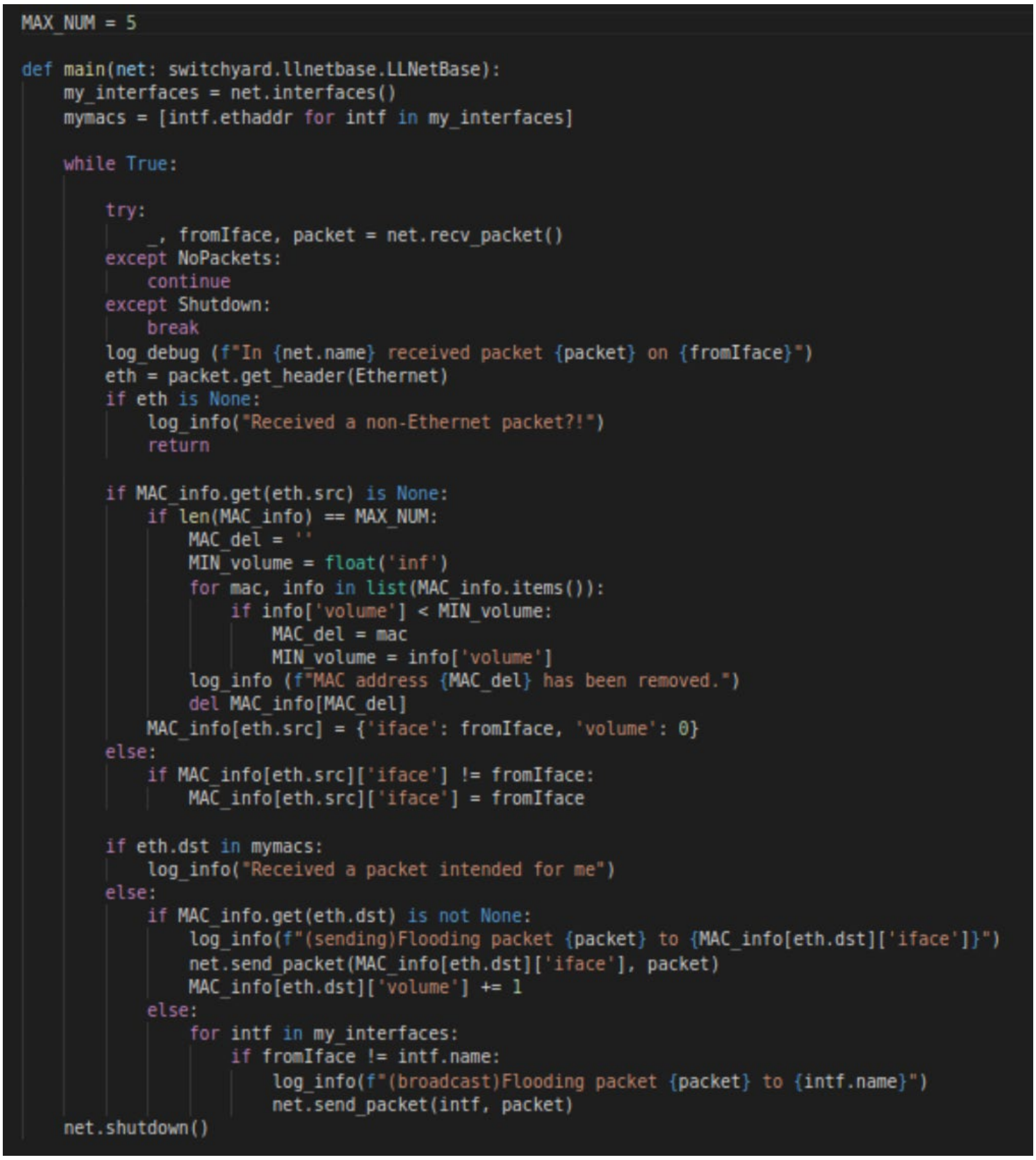
字典套字典，第一关键字和第二关键字即位 Step1 中的 MAC 地址和端口，第三关键字记录时间戳。

Step 3:Least Recently Used&Deploying



```python
MAC_info={}

MAX_NUM = 5

def main(net: switchyard.llnetbase.LLNetBase):
    my_interfaces = net.interfaces()
    mymacs = [intf.ethaddr for intf in my_interfaces]

    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)
        if eth is None:
            log_info("Received a non-Ethernet packet?!")
            return

        if MAC_info.get(eth.src) is None:
            if len(MAC_info) == MAX_NUM:
                MAC_del = ''
                MIN_time = float('inf')
                for mac, info in list(MAC_info.items()):
                    if info['timestamp'] < MIN_time:
                        MAC_del = mac
                        MIN_time = info['timestamp']
                log_info (f"MAC address {MAC_del} has been removed.")
                del MAC_info[MAC_del]
            MAC_info[eth.src] = {'iface': fromIface, 'timestamp': time.time()}
        else:
            if MAC_info[eth.src]['iface'] != fromIface:
                MAC_info[eth.src]['iface'] = fromIface
            MAC_info[eth.src]['timestamp'] = time.time()

        if eth.dst in mymacs:
            log_info("Received a packet intended for me")
        else:
            if MAC_info.get(eth.dst) is not None:
                log_info(f"(sending)Flooding packet {packet} to {MAC_info[eth.dst]['iface']}")
                net.send_packet(MAC_info[eth.dst]['iface'], packet)
            else:
                for intf in my_interfaces:
                    if fromIface != intf.name:
                        log_info(f"(broadcast)Flooding packet {packet} to {intf.name}")
                        net.send_packet(intf, packet)
    net.shutdown()
```

Step 4: Least Traffic Volume&Deploying



```python
MAX_NUM = 5

def main(net: switchyard.llnetbase.LLNetBase):
    my_interfaces = net.interfaces()
    mymacs = [intf.ethaddr for intf in my_interfaces]

    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)
        if eth is None:
            log_info("Received a non-Ethernet packet?!")
            return

        if MAC_info.get(eth.src) is None:
            if len(MAC_info) == MAX_NUM:
                MAC_del = ''
                MIN_volume = float('inf')
                for mac, info in list(MAC_info.items()):
                    if info['volume'] < MIN_volume:
                        MAC_del = mac
                        MIN_volume = info['volume']
                log_info (f"MAC address {MAC_del} has been removed.")
                del MAC_info[MAC_del]
            MAC_info[eth.src] = {'iface': fromIface, 'volume': 0}
        else:
            if MAC_info[eth.src]['iface'] != fromIface:
                MAC_info[eth.src]['iface'] = fromIface

        if eth.dst in mymacs:
            log_info("Received a packet intended for me")
        else:
            if MAC_info.get(eth.dst) is not None:
                log_info(f"(sending)Flooding packet {packet} to {MAC_info[eth.dst]['iface']}")
                net.send_packet(MAC_info[eth.dst]['iface'], packet)
                MAC_info[eth.dst]['volume'] += 1
            else:
                for intf in my_interfaces:
                    if fromIface != intf.name:
                        log_info(f"(broadcast)Flooding packet {packet} to {intf.name}")
                        net.send_packet(intf, packet)
    net.shutdown()
```

大致是将Step3中的timestamp时间戳改为volume。

5. 总结与感想

三种机制的转发表都使用 python 中的字典实现，为了有选择的删除，所以都在Base Case的基础上加了一个比较关键字，每次 O(n)查找需要删除的表项，不过应该可以用链表优化成O(1)。最后在mininet中测试时，为了方便所以修改了最大的表项数，通过输出日志以及抓包可以验证正确性。