

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 助教：方毓楚/于沛文/郑浩/陈伟/李国浩/李睿宸/杨溢...
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：201220096	姓名：钟亚晨
Email: 2991908515@qq.com	开始/完成日期：2022.03.20-2022.03.21

1.实验名称

Learning Switch

完成清单：

- ✧ 利用 switchyard 框架完成了 basic_switch、switch_to、switch_lru、switch_traffic 四种交换机的实现并附加了log_info和关键注释；
- ✧ 所有交换机实现都通过了框架代码所给出的 .srpy 类型的测试用例；
- ✧ 分别为四种交换机实现编写了对应的测试用例并且通过了全部自编写的测试用例，表现为 ./testcases/mytestscenario_switch/_to/_lru/_traffic.py （可选）
- ✧ 完成了总计7个带有 ☒ 的问题，并附在了本实验报告当中（完成了本实验报告）

2.实验目的

- ✧ 掌握基本的switchyard框架使用，熟悉相关API
- ✧ 掌握基本的交换机工作流程，并且由此拓展出使用超时机制、LRU算法以及最小流量机制的交换机，并且使用python实践结合switchyard进行实现
- ✧ 进一步巩固测试方法，包括使用switchyard框架制作testscenario利用框架测试功能进行测试以及在Mininet中，结合xterm以及wireshark等仿真测试，保证组件功能实现的正确性

3.实验内容

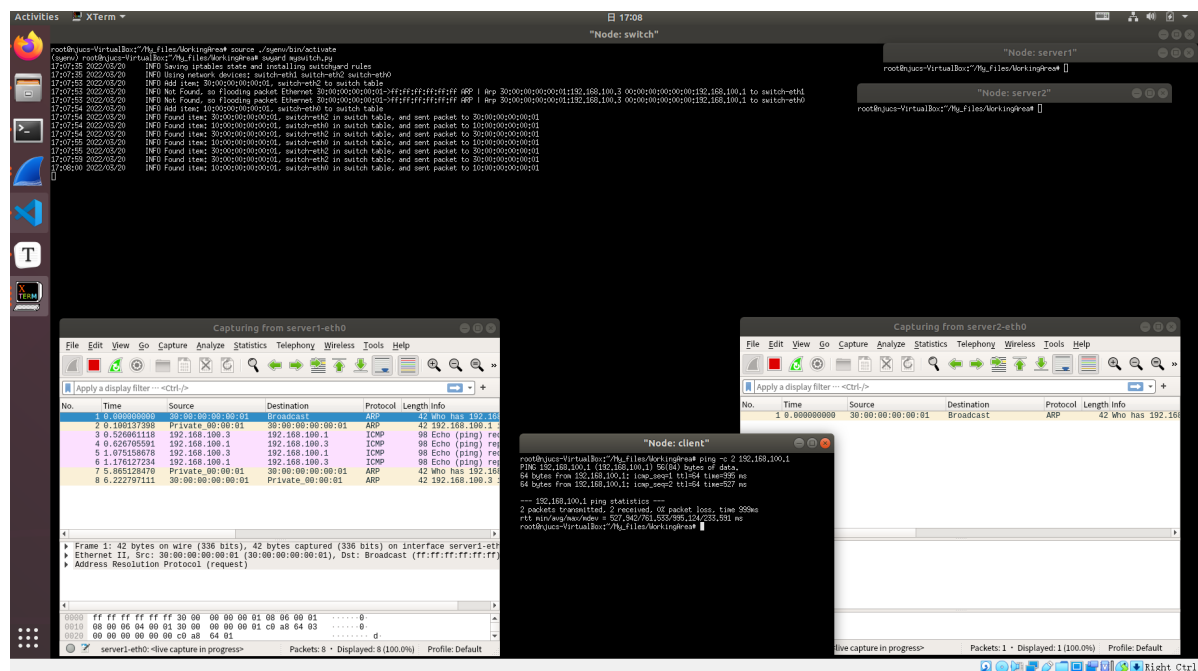
- ✧ 完成基础功能交换机，并且使用mininet进行模拟测试，分析说明测试结果；
- ✧ 完成超时机制交换机，并且通过testscenario的测试，再使用mininet进行真实环境模拟测试，分析说明测试结果；
- ✧ 完成带有LRU算法的交换机，并且通过testscenario的测试，再使用mininet进行真实环境模拟测试，分析说明测试结果；
- ✧ 完成最小流量机制交换机，并且通过testscenario的测试，再使用mininet进行真实环境模拟测试，分析说明测试结果；
- (可选)
- ✧ 自己编写对基础交换机的testscenario，通过测试并阐述测试思路；
- ✧ 自己编写对超时机制交换机的testscenario，通过测试并阐述测试思路；

- ✧自己编写对LRU算法交换机的testscenario，通过测试并阐述测试思路；
- ✧自己编写对最小流量机制交换机的testscenario，通过测试并阐述测试思路；

4.实验结果

简单说明，以下对实验说明中的七个问题进行回答。

0x01 ✓ Task2:Basic Switch-Analyze and state the results of the above process in your report with screenshots. Do not explain how you do step by step but focus on the switch's forwarding logic.



结果分析

可以看到:

节点client利用指令 `ping -c 2 192.168.100.1` 向节点 server2 发送了两个数据包,运行在 server2 上的 wireshark 显示有一对ICMP数据包(request-reply)和一堆ARP数据包,而运行在 server1 上的 wireshark 则仅有一个ARP广播数据包。说明 basic switch 正常运行

另外，我在源代码中使用 `log_info` 对 basic switch 的行为进行了显示，可以看到 basic switch 的终端：

一开始接收到来自client的数据包，便将client地址-端口 添加到交换机表中，这里显示为 Add item开头的info信息，而后由于找不到目的地址对应的表项，便向除接收端口以外的另两个端口广播，这里显示为两个Not Found开头的info信息；

而之后又使用了ARP协议，则交换机会接收到来自目的地址server1的数据包，之后自学习将其地址-对应端口添加到交换机表中，即显示为第二个Add item开头的info信息；

随后，由于client和server1两个主机的地址都已被交换机学习，后续便全部通过查表即可实现转发，显示为一连串的Found item开头的info信息

交换机转发逻辑

具体源代码见 `myswitch.py` 内有详细注释，在此对实现逻辑进行简短说明：

使用一个名为 `switch_table` 的字典作为交换机表，对于每个键值对：键为MAC地址，而值为端口；

当我们接收到发给自己的包时，什么也不做，直接将其丢弃；

当我们接收到了广播时，则直接向接收端口之外的所有端口进行转发广播；

而当我们接收到一个主机-主机的包时，则进行如下操作：

对于发送方：

一、如果在交换机表中查找不到源地址对应的键，那么将“源地址-接收端口”键值添加到交换机表中；

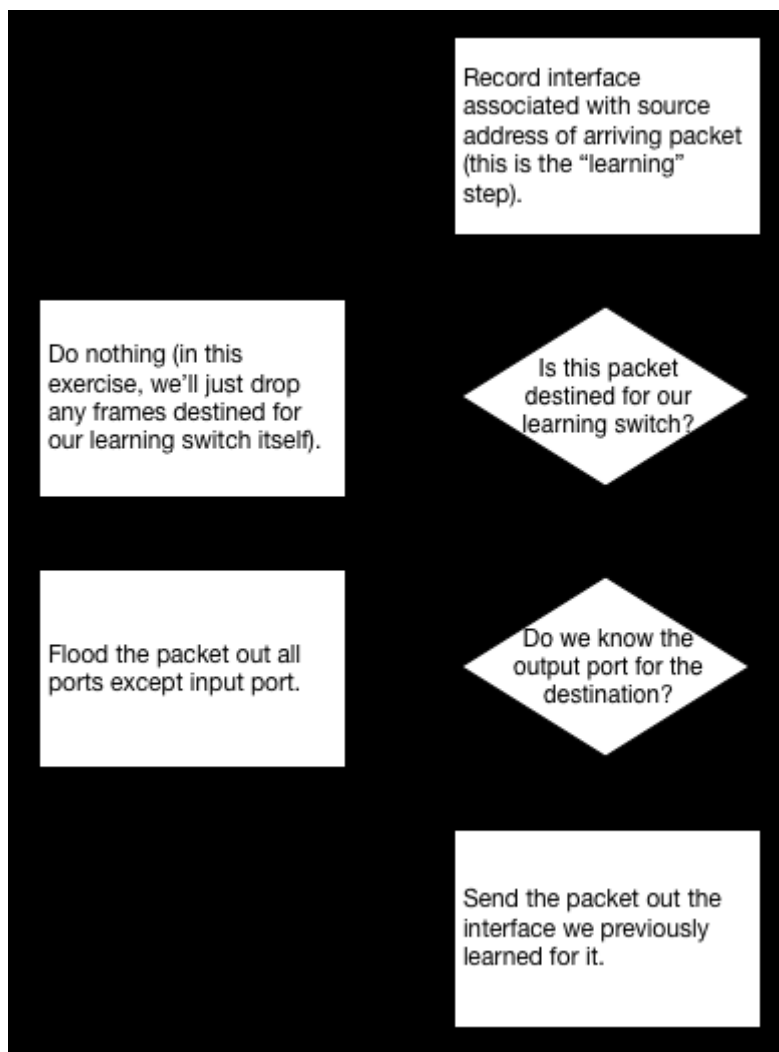
二、如果找到了对应的键，但接收端口变了（这里考虑了主机移动），那么更新对应的键值对；

对于接收方：

一、以目的MAC地址查询交换机表，如果找到了对应表项，从表项中获取端口，将包发送到对应端口；

二、以目的MAC地址查询交换机表，如果找不到对应表项，那么对除接收端口外的所有端口进行广播；

实际上就是按照实验说明中该流程图进行实现：



（补充）编写自己的测试文件，并阐述测试思路：

在 `./testcases/mytestscenario.py` 当中，我编写了对于基础交换机的测试用例，要了解更多细节，可查看对应目录下源代码，有如下几个测试：

test case 1：基础交换机接收到广播，期望其将该报文转发到除接收端口之外的所有其它端口；

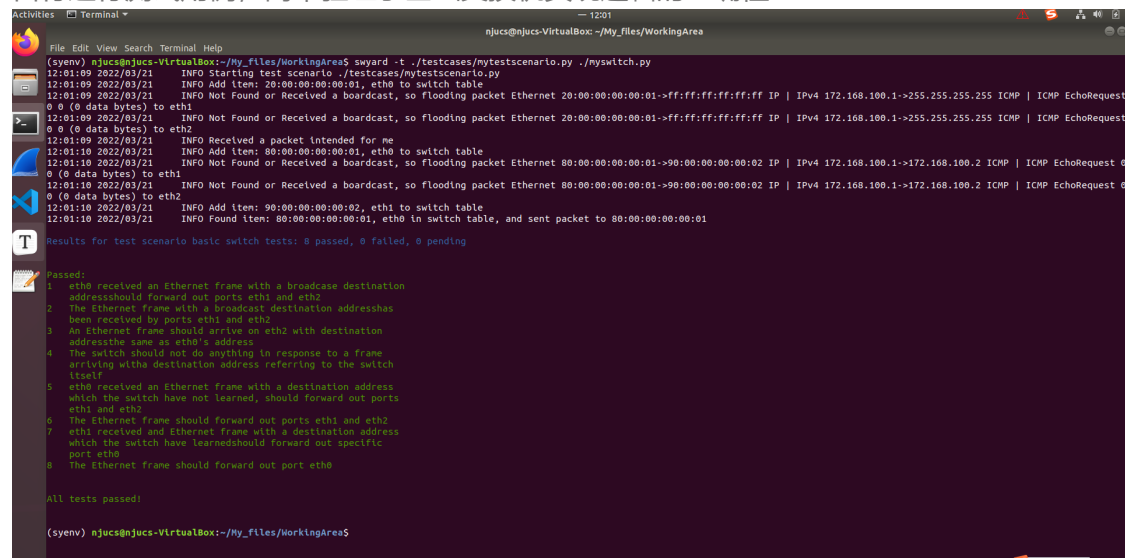
test case 2：基础交换机接收到发送给自己的任意端口的数据包，期望其什么也不做，将该报文丢弃；

test case 3：从eth0接受到一个来自IP：172.168.100.1的数据包，发送到未被交换机学习的eth1的IP：172.168.100.2，期望该交换机进行泛洪，将该数据包转发到除接收端口外的所有其它端口；

此时交换机已经学习到了向IP：172.168.100.1发送数据包需要向eth0端口转发

test case 4：从eth1的IP：172.168.100.2接收到一个发送给IP：172.168.100.1的报文，此时交换机已学习，则该报文只会被转发到eth0端口

自行运行测试用例，简单验证了基础交换机实现逻辑的正确性：



```
(syenv) njucs@njucs-VirtualBox:~/My_files/WorkingArea$ swyard -t ./testcases/mytestscenario.py ./nyswitch.py
12:01:09 2022/03/21 INFO Starting test scenario ./testcases/mytestscenario.py
12:01:09 2022/03/21 INFO Add item: 20:00:00:00:00:01, eth0 to switch table
12:01:09 2022/03/21 INFO Not Found or Received a boardcast, so flooding packet Ethernet 20:00:00:00:00:01->ff:ff:ff:ff:ff:ff IP | IPv4 172.168.100.1->255.255.255.255 ICMP | ICMP EchoRequest 0
0 0 (0 data bytes) to eth1
12:01:09 2022/03/21 INFO Not Found or Received a boardcast, so flooding packet Ethernet 20:00:00:00:00:01->ff:ff:ff:ff:ff:ff IP | IPv4 172.168.100.1->255.255.255.255 ICMP | ICMP EchoRequest 0
0 0 (0 data bytes) to eth2
12:01:09 2022/03/21 INFO Received a packet intended for me
12:01:10 2022/03/21 INFO Add item: 80:00:00:00:00:01, eth0 to switch table
12:01:10 2022/03/21 INFO Not Found or Received a boardcast, so flooding packet Ethernet 80:00:00:00:00:01->90:00:00:00:00:02 IP | IPv4 172.168.100.1->172.168.100.2 ICMP | ICMP EchoRequest 0
0 0 (0 data bytes) to eth1
12:01:10 2022/03/21 INFO Not Found or Received a boardcast, so flooding packet Ethernet 80:00:00:00:00:01->90:00:00:00:00:02 IP | IPv4 172.168.100.1->172.168.100.2 ICMP | ICMP EchoRequest 0
0 0 (0 data bytes) to eth2
12:01:10 2022/03/21 INFO Add item: 90:00:00:00:00:02, eth1 to switch table
12:01:10 2022/03/21 INFO Found item: 80:00:00:00:00:01, eth0 in switch table, and sent packet to 80:00:00:00:00:01
Results for test scenario basic switch tests: 0 passed, 0 failed, 0 pending

Passed:
1 eth0 received an Ethernet frame with a broadcast destination address should forward out ports eth1 and eth2
2 The Ethernet frame with a broadcast destination address has been received by ports eth1 and eth2
3 An Ethernet frame should arrive on eth2 with destination address the same as eth0's address
4 The switch should not do anything in response to a frame arriving with a destination address referring to the switch itself
5 eth0 received an Ethernet frame with a destination address which the switch have not learned, should forward out ports eth1 and eth2
6 The Ethernet frame should forward out ports eth1 and eth2
7 eth1 received an Ethernet frame with a destination address which the switch have learned should forward out specific port eth0
8 The Ethernet frame should forward out port eth0

All tests passed!

(syenv) njucs@njucs-VirtualBox:~/My_files/WorkingArea$
```

0x02 Task3:Timeouts-In the report, show the test result of your switch. (Optional) If you have written the test files yourself, show how you test the timeout mechanism.

可以看到通过了所给的所有测试用例

```
Activities Terminal
njucs@njucs-VirtualBox: ~/My_files/WorkingArea
(syenv) njucs@njucs-VirtualBox:~/My_files/WorkingArea$ sward -t ./testcases/myswitch_to_testscenario.py
20:15:26 2022/03/20 INFO Starting test scenario ./testcases/myswitch_to_testscenario.py
20:15:26 2022/03/20 INFO Add item: 30:00:00:00:00:02, eth1, 1647778526.839 to switch table
20:15:26 2022/03/20 INFO Not Found or Received a broadcast, so 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
20:15:26 2022/03/20 INFO Not Found or Received a broadcast, so 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
20:15:26 2022/03/20 INFO Add item: 20:00:00:00:00:01, eth0, 1647778526.8398883 to switch table
20:15:26 2022/03/20 INFO Found item: 30:00:00:00:00:02, eth1 in switch table, and sent packet to 30:00:00:00:00:02
20:15:46 2022/03/20 INFO The item: 30:00:00:00:00:02, eth1, 1647778526.8398883 expired, now is 1647778546.856227, so delete it!
20:15:46 2022/03/20 INFO Add item: 20:00:00:00:00:01, eth0, 1647778526.8399417 to switch table
20:15:46 2022/03/20 INFO Add item: 20:00:00:00:00:01, eth0, 1647778546.8573575 to switch table
20:15:46 2022/03/20 INFO Not Found or Received a broadcast, so 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
20:15:46 2022/03/20 INFO Not Found or Received a broadcast, so 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
20:15:46 2022/03/20 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 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 20s
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!
(syenv) njucs@njucs-VirtualBox:~/My_files/WorkingArea$
```

(选做) 编写自己的测试文件，并阐述测试思路：

自己编写了测试文件 `./testcases/mytestscenario_to.py`，该测试由 `./testcases/mytestscenario.py` 进行拓展得来，和超时机制交换机由基础交换机源代码拓展实现而来类似，编写了如下测试用例：

test case 1：接收到广播，期望将该报文进行泛洪；

test case 2：接收到发送给自己的包，期望什么也不做将其丢弃；

test case 3：接收到目的地为未学习的包，期望对其进行泛洪，且此时已将源地址-端口进行学习，对应表项记为item1；

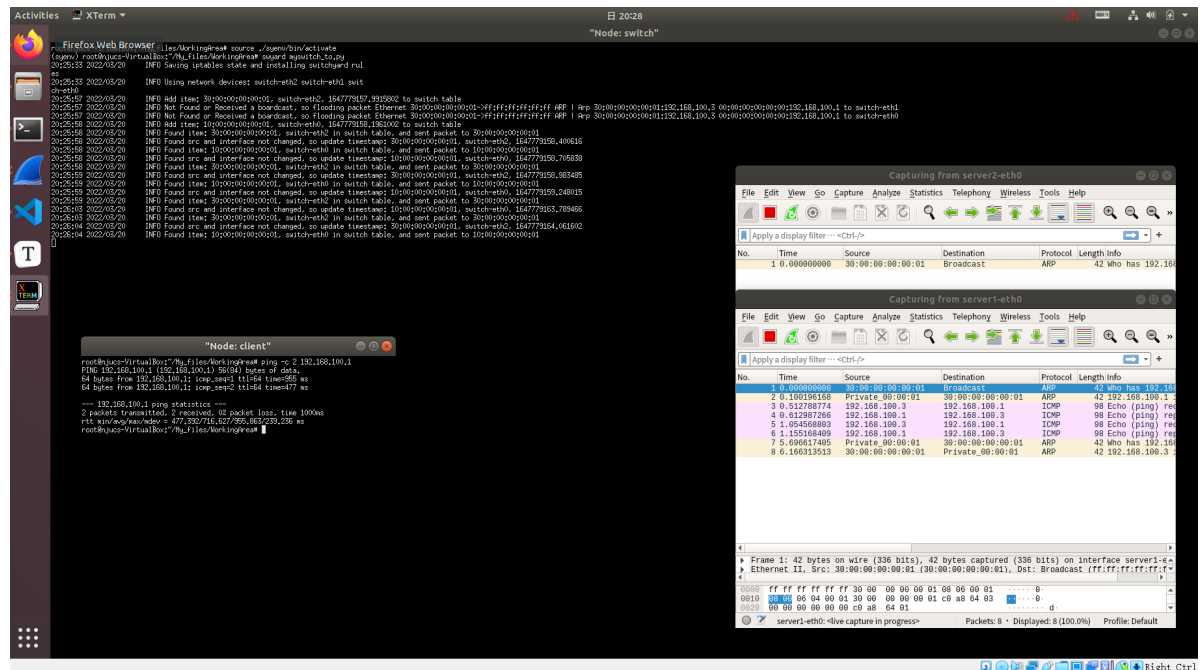
test case 4：立即向已学习的地址（item1）发送报文，期望仅有其对应的端口接收到报文；

test case 5：等待11秒后，再次向已学习的地址（item1）发送报文，期望进行泛洪（此时先前已学习的item1根据超时机制已经过期失效）

自运行测试用例，通过了所有测试用例，可以验证超时机制交换机实现的正确性

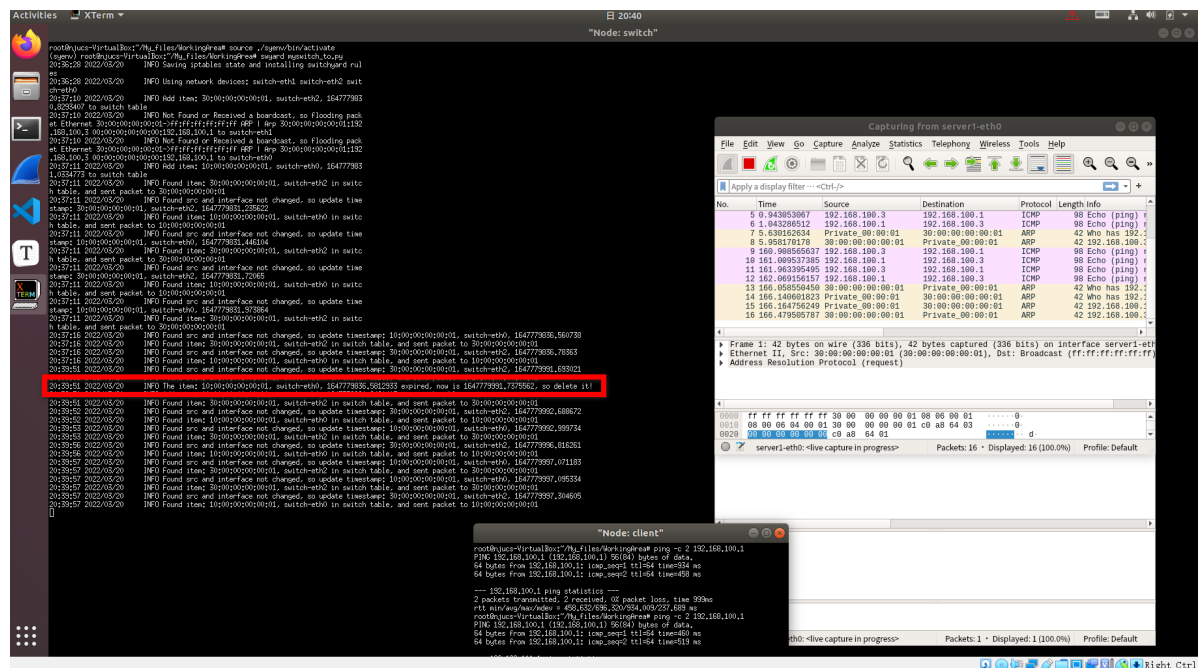
```
File Edit View Search Terminal Help
njucs@njucs-VirtualBox:~/My_files/WorkingArea$ sward -t ./testcases/mytestscenario_to.py ./myswitch_to.py
12:30:09 2022/03/21 INFO Starting test scenario ./testcases/mytestscenario_to.py
12:30:09 2022/03/21 INFO Add item: 20:00:00:00:00:01, eth0, 164737905.4439087 to switch table
12:30:09 2022/03/21 INFO Not Found or Received a broadcast, so flooding packet Ethernet 20:00:00:00:00:01->ff:ff:ff:ff:ff:ff IP | IPv4 172.168.100.1->255.255.255.255 ICMP | ICMP EchoRequest 0
0 (0 data bytes) to eth1
12:30:09 2022/03/21 INFO Not Found or Received a broadcast, so flooding packet Ethernet 20:00:00:00:00:01->ff:ff:ff:ff:ff:ff IP | IPv4 172.168.100.1->255.255.255.255 ICMP | ICMP EchoRequest 0
0 (0 data bytes) to eth2
12:30:09 2022/03/21 INFO Received a packet intended for me
12:30:09 2022/03/21 INFO Add item: 80:00:00:00:00:01, eth0, 164737906.4473922 to switch table
12:30:09 2022/03/21 INFO Not Found or Received a broadcast, so flooding packet Ethernet 80:00:00:00:00:01->90:00:00:00:00:02 IP | IPv4 172.168.100.1->172.168.100.2 ICMP | ICMP EchoRequest 0
0 (0 data bytes) to eth1
12:30:09 2022/03/21 INFO Not Found or Received a broadcast, so flooding packet Ethernet 80:00:00:00:00:01->90:00:00:00:00:02 IP | IPv4 172.168.100.1->172.168.100.2 ICMP | ICMP EchoRequest 0
0 (0 data bytes) to eth2
12:30:09 2022/03/21 INFO Add item: 80:00:00:00:00:01, eth0, 164737906.448494 to switch table
12:30:09 2022/03/21 INFO Found item: 80:00:00:00:00:01, eth0 in switch table, and sent packet to 80:00:00:00:00:01
12:30:17 2022/03/21 INFO The item: 20:00:00:00:00:01, eth0, 164737906.4488857 expired, now is 164737907.4539802, so delete it!
12:30:17 2022/03/21 INFO The item: 80:00:00:00:00:01, eth0, 164737906.4473281 expired, now is 164737907.4541647, so delete it!
12:30:17 2022/03/21 INFO The item: 90:00:00:00:00:02, eth1, 164737906.448357 expired, now is 164737907.4541647, so delete it!
12:30:17 2022/03/21 INFO Add item: 90:00:00:00:00:02, eth1, 164737907.454272 to switch table
12:30:17 2022/03/21 INFO Not Found or Received a broadcast, so flooding packet Ethernet 90:00:00:00:00:02->80:00:00:00:00:01 IP | IPv4 172.168.100.2->172.168.100.1 ICMP | ICMP EchoRequest 0
0 (0 data bytes) to eth0
12:30:17 2022/03/21 INFO Not Found or Received a broadcast, so flooding packet Ethernet 90:00:00:00:00:02->80:00:00:00:00:01 IP | IPv4 172.168.100.2->172.168.100.1 ICMP | ICMP EchoRequest 0
0 (0 data bytes) to eth2
Results for test scenario basic switch tests: 11 passed, 0 failed, 0 pending
Passed:
1 eth0 received an Ethernet frame with a broadcast destination
address should forward out ports eth1 and eth2
2 The Ethernet frame with a broadcast destination address has
been received by ports eth1 and eth2
3 An Ethernet frame should arrive on eth2 with destination
address the same as eth0's address
4 The switch should not do anything in response to a frame
arriving with destination address referring to the switch
itself
5 eth0 received an Ethernet frame with a destination address
which the switch have not learned, should forward out ports
eth1 and eth2
6 The Ethernet frame should forward out ports eth1 and eth2
7 eth1 received an Ethernet frame with a destination address
which the switch have learned should forward out specific
port eth0
8 The Ethernet frame should forward out port eth0
9 waiting for 1s
10 eth1 received an Ethernet frame with a destination address
which the switch have learned should forward out specific
port eth0
11 The Ethernet frame should forward out port eth0
All tests passed!
njucs@njucs-VirtualBox:~/My_files/WorkingArea$
```

0x03 Task3:Timeouts-In Mininet, test your timeout mechanism. Prove that the timeout mechanism works with your testing procedure in the report.



首先我们按照基础交换机的测试方法进行测试，可以看见使用超时机制的交换机同样可以正常工作，即基础的交换机功能实现无误；

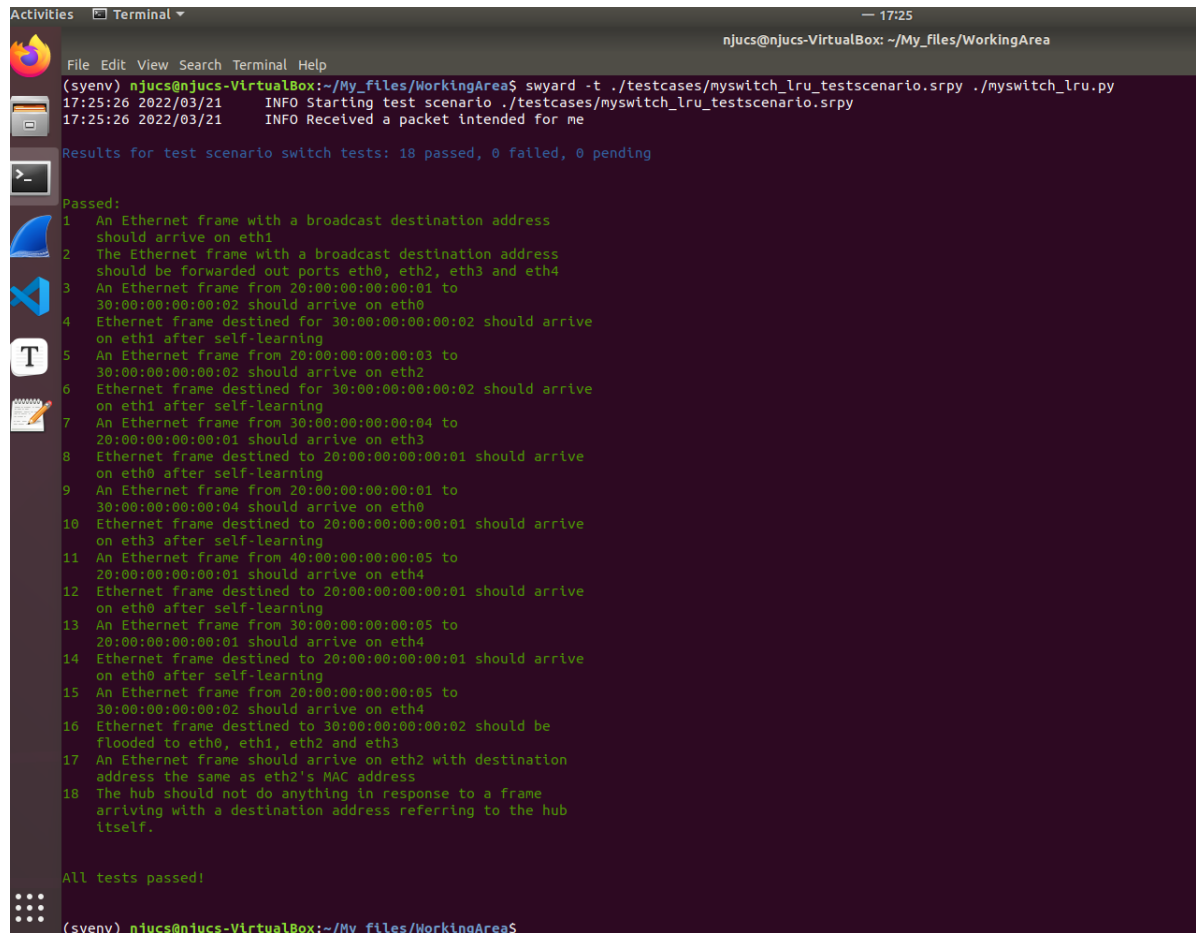
在等待一段时间（大于10s）后，我们再次由 client 向 server1 发送请求，可以看到经过一段时间后原本存在于交换机表中的表项过期而被删除，再次向对应地址发送请求时又找不到了，从而需要进行泛洪



在Mininet进行仿真测试，这说明所实现的交换机是成功实现了超时机制的。

0x04 Task4:Least Recently Used-In the report, show the test result of your switch. (Optional) If you have written the test files yourself, show how you test the LRU algorithm.

可以看到通过了所有所给的测试用例：



```
(syenv) njucs@njucs-VirtualBox:~/My_files/WorkingArea$ swyard -t ./testcases/myswitch_lru_testscenario.srpy ./myswitch_lru.py
17:25:26 2022/03/21 INFO Starting test scenario ./testcases/myswitch_lru_testscenario.srpy
17:25:26 2022/03/21 INFO Received a packet intended for me

Results for test scenario switch tests: 18 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, 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!

(syenv) njucs@njucs-VirtualBox:~/My_files/WorkingArea$
```

(选做) 编写自己的测试文件，并阐述测试思路：

自己编写了测试文件 `./testcases/mytestscenario_lru.py`，该测试由 `./testcases/mytestscenario.py` 进行拓展得来，和基于LRU算法的交换机由基础交换机源代码拓展实现而来类似，编写了如下测试用例：

前4个test cases，直接基于 `myswitchscenario.py`，用于测试该基于LRU算法的交换机是否具备交换机的基本功能；

而后一个test case，由指定的172.168.100.101->172.168.100.110发送数据包，此时交换机未学习到，应当对其它所有端口进行泛洪；

而后有5个test case，全部都会为交换机添加不同的新表项，以期望将上一步的表项冲刷；

最后一个testcase，重复第五个测试用例，原本被学习的表项被LRU算法所删除，此时期望进行广播，即表现为该表项已被删除，交换机尚未学习到。

自运行测试用例，通过了所有测试用例，可以验证基于LRU算法的交换机功能实现的正确性

```
Activities Terminal 17:51 njucs@njucs-VirtualBox: ~/My_files/WorkingArea

File Edit View Search Terminal Help

njucs@njucs-VirtualBox:~/My_files/WorkingArea$ swyard -t ./testcases/mytestscenario_lru.py ./myswitch_lru.py
17:51:18 2022/03/21 INFO Starting test scenario ./testcases/mytestscenario_lru.py
17:51:18 2022/03/21 INFO Received a packet intended for me

Results for test scenario basic switch tests: 22 passed, 0 failed, 0 pending

Passed:
1 eth0 received an Ethernet frame with a broadcast destination address should forward out ports eth1 and eth2
2 The Ethernet frame with a broadcast destination address has been received by ports eth1 and eth2
3 An Ethernet frame should arrive on eth2 with destination address the same as eth0's address
4 The switch should not do anything in response to a frame arriving with a destination address referring to the switch itself
5 eth0 received an Ethernet frame with a destination address which the switch has not learned, should forward out ports eth1 and eth2
6 The Ethernet frame should forward out ports eth1 and eth2
7 eth1 received an Ethernet frame with a destination address which the switch has learned should forward out specific port eth0
8 The Ethernet frame should forward out port eth0
9 LRU test case 1: 172.168.100.101 -> 172.168.100.110
10 forward to other ports eth1 and eth2
11 the 1st new packet has been switched
12 forward to other ports eth0 and eth2
13 the 1st new packet has been switched
14 forward to other ports eth0 and eth2
15 the 1st new packet has been switched
16 forward to other ports eth0 and eth2
17 the 1st new packet has been switched
18 forward to other ports eth0 and eth2
19 the 1st new packet has been switched
20 forward to other ports eth0 and eth2
21 LRU test case 1: 172.168.100.101 -> 172.168.100.110
22 forward to other ports eth1 and eth2

All tests passed!

njucs@njucs-VirtualBox:~/My_files/WorkingArea$
```

0x05 ✓ Task4: Least Recently Used-In Mininet, test the LRU algorithm. Prove that the LRU algorithm works with your testing procedure in the report.

```
Activities Terminal 18:27 njucs@njucs-VirtualBox: ~/My_files/WorkingArea

File Edit View Search Terminal Help

root@njucs-VirtualBox:~/My_files/WorkingArea$ source ./myenv/bin/activate
root@njucs-VirtualBox:~/My_files/WorkingArea$ mininet
mininet> xterm switch
mininet> server1 ping -c 1 192.168.100.3
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=1127 ms

--- 192.168.100.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1127.966/1127.966/1127.966/0.000 ms
mininet> server2 ping -c 1 192.168.100.3
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=935 ms

--- 192.168.100.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 935.041/935.041/935.041/0.000 ms
mininet> server3 ping -c 1 192.168.100.3
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=967 ms

--- 192.168.100.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 967.197/967.197/967.197/0.000 ms
mininet> server4 ping -c 1 192.168.100.3
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=919 ms

--- 192.168.100.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 919.088/919.088/919.088/0.000 ms
mininet> server5 ping -c 1 192.168.100.3
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=849 ms

--- 192.168.100.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 849.432/849.432/849.432/0.000 ms
mininet> client ping -c 1 192.168.100.1
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=451 ms

--- 192.168.100.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 451.033/451.033/451.033/0.000 ms
mininet>
```

在这一个使用Mininet进行模拟测试的过程中，由于主机不够，因而我自行修改了./start_mininet.py当中的拓扑结构，为其添加了编号末3-6的4个server，从而有足够的主机来创建超过5个的互不相同的表项，从而验证我们的LRU机制实现的正确性，具体更改已在./start_mininet.py当中以注释给出。

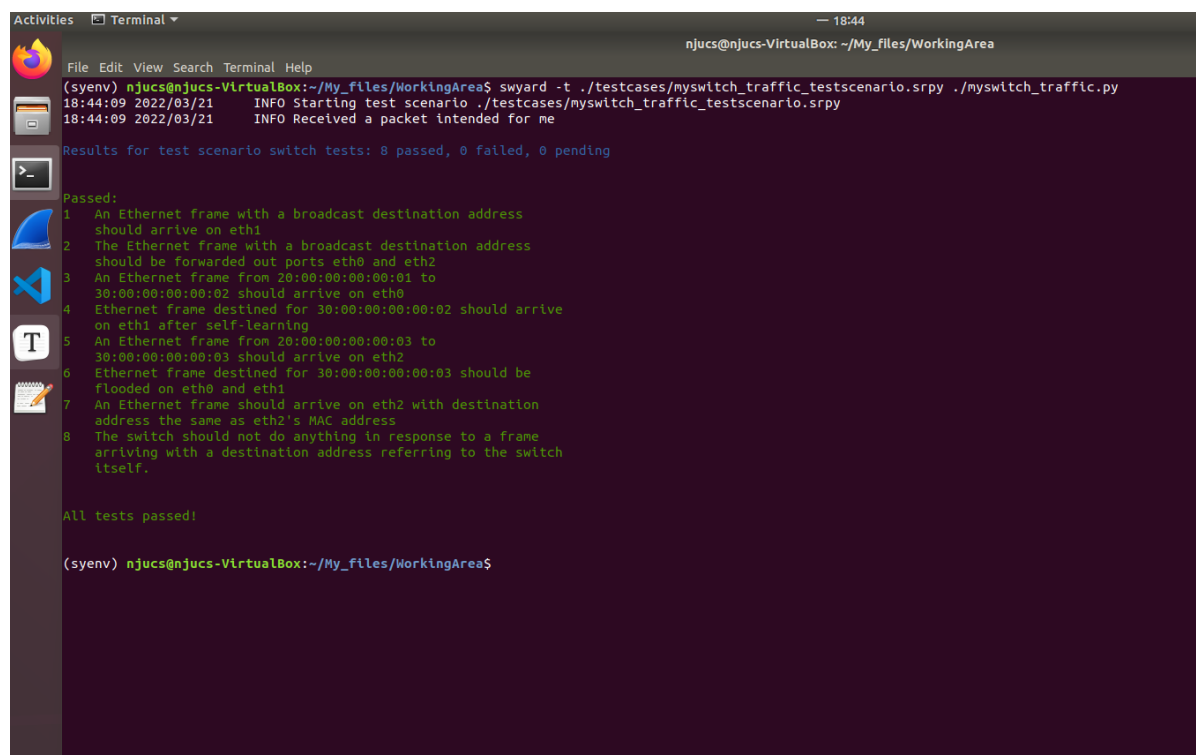
在这个模拟测试当中，①我们首先由 server1 向 client 发送数据包，而后交换机学习到 server1 对应的地址-端口表项，而后我们依次通过 server2 到 server6 五个不同的主机向 client 发送数据包，依次向交换机表中添加五个不同的表项，②此时原先记录着 server1 地址-端口MAC地址的表项根据LRU机制就已经被删除了，③那么随后我们通过 client 向 server1 发送数据包，此时交换机表中并不存在 server1 的表项，从而要对除接收端口以外的其它端口进行泛洪。

以上三个步骤依次对应于图片当中三个红框中通过 log_info 打印出的信息。

通过上述步骤，我们在Mininet的仿真模拟环境下进行了测试，可以基本验证所实现的基于LRU算法的交换机的功能正确性。

0x06 In the report, show the test result of your switch.(Optional) If you have written the test files yourself, show how you test the least traffic volume algorithm.

可以看到通过了所有所给的测试用例：



```
Activities  Terminal  — 18:44
njucs@njucs-VirtualBox: ~/My_files/WorkingArea
(syenv) njucs@njucs-VirtualBox:~/My_files/WorkingArea$ swyard -t ./testcases/myswitch_traffic_testscenario.srpy ./myswitch_traffic.py
18:44:09 2022/03/21 INFO Starting test scenario ./testcases/myswitch_traffic_testscenario.srpy
18:44:09 2022/03/21 INFO Received a packet intended for me

Results for test scenario switch tests: 8 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 eth2
3 An Ethernet frame from 20:00:00:00:01 to
  30:00:00:00:00:02 should arrive on eth0
4 Ethernet frame destined for 30:00:00:00:02 should arrive
  on eth1 after self-learning
5 An Ethernet frame from 20:00:00:00:03 to
  30:00:00:00:00:03 should arrive on eth2
6 Ethernet frame destined for 30:00:00:00:03 should be
  flooded on eth0 and eth1
7 An Ethernet frame should arrive on eth2 with destination
  address the same as eth2's MAC address
8 The switch should not do anything in response to a frame
  arriving with a destination address referring to the switch
  itself.

All tests passed!

(syenv) njucs@njucs-VirtualBox:~/My_files/WorkingArea$
```

(选做) 编写自己的测试文件，并阐述测试思路：

这里我们分为两个测试文件来进行测试：

分为两个部分是因为在测试交换机基本功能时便会产生一定流量，增加后续编写对流量控制机制测试用例的难度，耦合性过高，因此将两者分离开分别进行测试：

一个是利用基本的交换机测试用例 `myscenario.py`，由于其条目数小于等于五，因此可以用于测试我们所实现的基于流量机制的交换机作为交换机的基本功能实现的正确性：

```
Activities Terminal 19:04
njucs@njucs-VirtualBox: ~/My_files/WorkingArea
File Edit View Search Terminal Help
njucs@njucs-VirtualBox:~/My_files/WorkingArea$ source ./syenv/bin/activate
(syenv) njucs@njucs-VirtualBox:~/My_files/WorkingArea$ swyard -t ./testcases/mytestscenario.py ./myswitch_traffic.py
19:04:17 2022/03/21 INFO Starting test scenario ./testcases/mytestscenario.py
19:04:17 2022/03/21 INFO Received a packet intended for me

Results for test scenario basic switch tests: 8 passed, 0 failed, 0 pending

Passed:
1 eth0 received an Ethernet frame with a broadcast destination
addressshould forward out ports eth1 and eth2
2 The Ethernet frame with a broadcast destination addresshas
been received by ports eth1 and eth2
3 An Ethernet frame should arrive on eth2 with destination
addressthe same as eth0's address
4 The switch should not do anything in response to a frame
arriving witha destination address referring to the switch
itself
5 eth0 received an Ethernet frame with a destination address
which the switch have not learned, should forward out ports
eth1 and eth2
6 The Ethernet frame should forward out ports eth1 and eth2
7 eth1 received an Ethernet frame with a destination address
which the switch have learnedshould forward out specific
port eth0
8 The Ethernet frame should forward out port eth0

All tests passed!

(syenv) njucs@njucs-VirtualBox:~/My_files/WorkingArea$
```

可以看到通过了所有的测试用例，因此可以验证所实现的基于流量机制的交换机的交换机基本功能的正确性；

另一个则是针对流量机制的测试用例 `myscenario_traffic.py`，对流量机制进行测试，该测试集的实现思路是：

首先初始化七个端口eth0-6：

首先由eth0向eth6发送数据包，以将eth0添加到交换机表中，此时eth6并不在交换器表中，期望交换机将报文泛洪到除接收端口以外的其它所有端口；

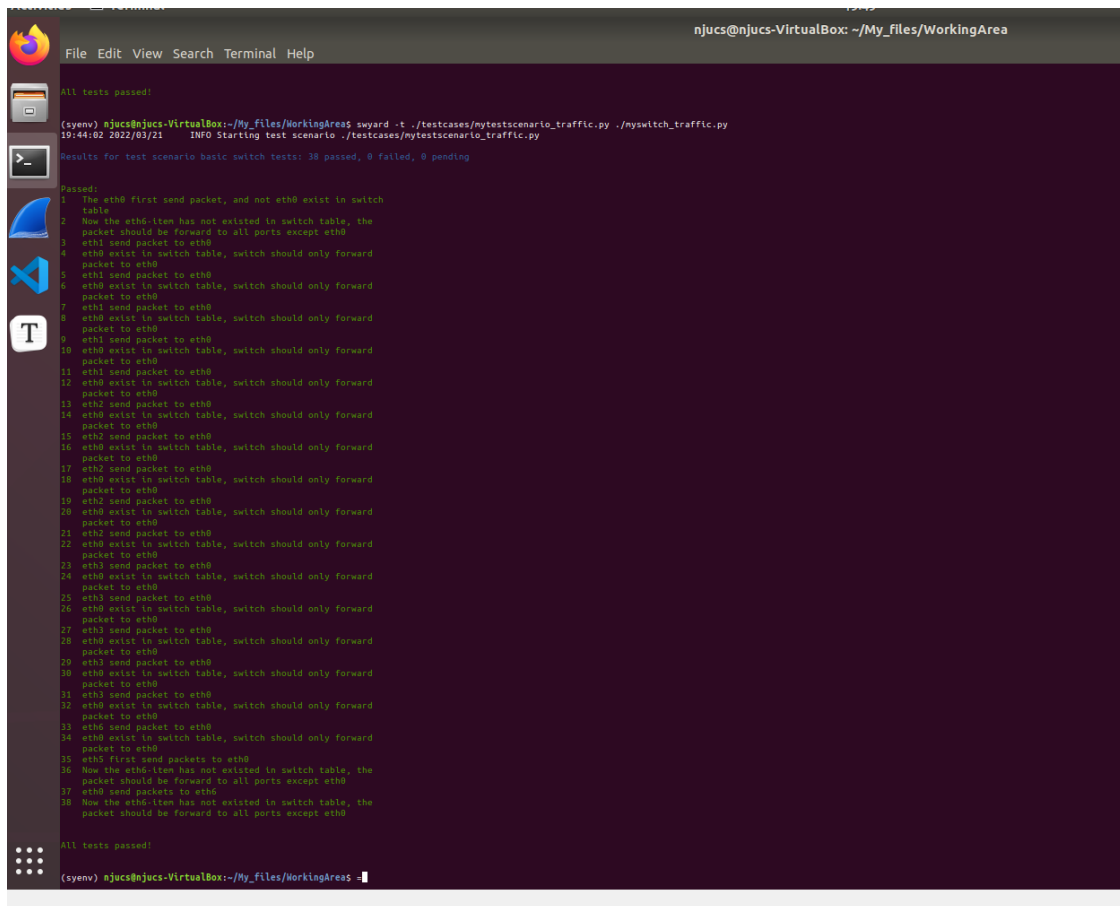
而后由eth1-3分别连续向eth0发送5个报文，制造流量；

再由eth6向eth0发送一个报文，此时交换机表已满，而eth6对应表项流量最小；

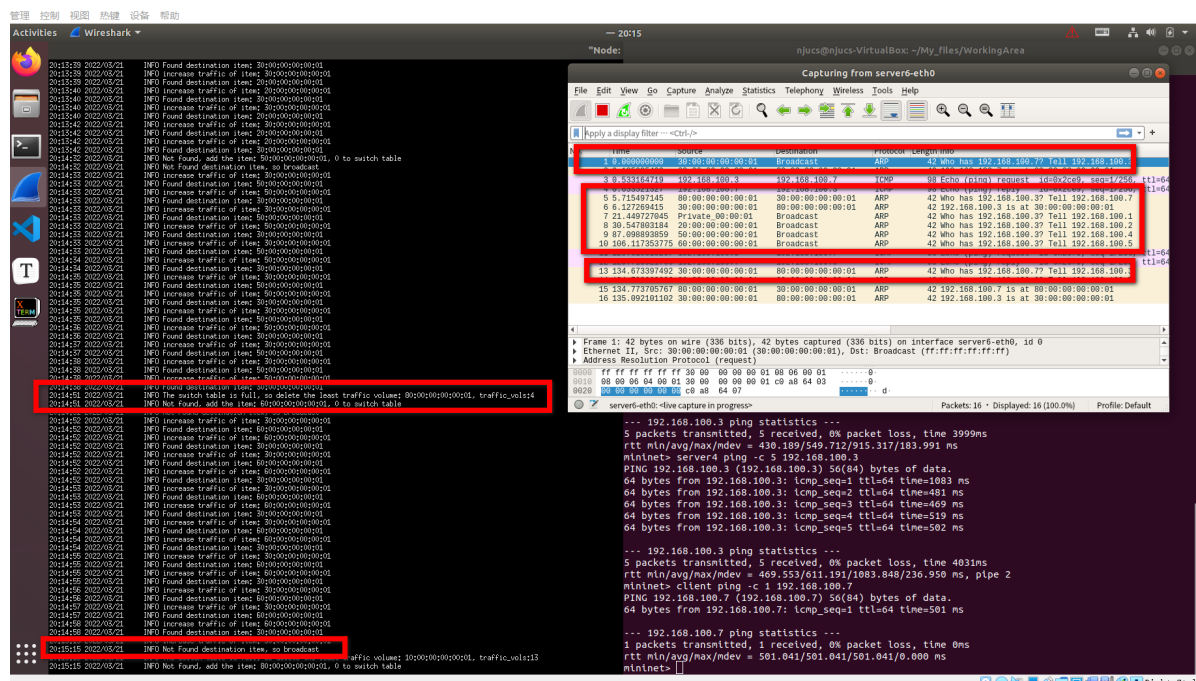
之后再由eth5向eth0发送报文，这时根据流量最小机制会将eth6表项删除；

再由eth0向eth6发送报文，如果实现正确，此时交换机表中已经不存在eth6表项了，期望交换机向除接收端口外的其它端口泛洪；

运行测试用例，通过了所有测试，此时我们可以基本检验判定基于流量机制的交换机实现的正确性



0x07 ✓ In Mininet, test the least traffic volume algorithm. Prove that the least traffic volume algorithm works with your testing procedure in the report.



在这一个使用Mininet进行模拟测试的过程中，由于主机不够，因而我自行修改了 `./start_mininet.py` 当中的拓扑结构，为其添加了编号末3-6的4个server，从而有足够的本机来创建超过5个的互不相同的表项，从而验证我们的LRU机制实现的正确性，具体更改已在 `./start_mininet.py` 当中以注释给出。（所使用的拓扑同 `switch_lru` 测试时使用的拓扑）

在Mininet的测试当中，我首先令 client 向 server6 发送1次ping请求，而后依次由 server1 到 server3 向 client 发送5次ping请求，此时交换机表中存有表项： client ， server1 到 server3 以及 server6 总计6个表项，并且我们不难知道 server6 的流量是最少的，随后我们用 server4 向 client 发送ping请求，不出意料，正像左侧从上至下第一个红框的log_info信息所示，将 server6 对应表项删除了，而后我们再用 client 向 server6 发送ping请求，如期望的一样，在交换机表中找不到表项，只能进行广播。

另外，在运行于 server6 的wireshark当中，我们也能很明显地看到首先是 client 请求与 server6 建立连接的广播，而后的一连串广播向交换机表中追加了4个新表项，而client表项在第一次建立连接时就被添加到交换机表中，最后一个则是 client 再一次向 server6 发送ping请求，由于 server6 表项已被删除，从而进行的广播

5.核心代码

```
break

log_debug (f"In {net.name} received packet {packet} on {fromIface}")
eth = packet.get_header(Ethernet)
# 接收到非因特网包 - 直接退出
if eth is None:
    log_info("Received a non-Ethernet packet?!")
    return
# 接收到发给自己的包 - 直接丢弃 (本实验中)
if eth.dst in mymacs:
    log_info("Received a packet intended for me")
# 接收到指定地址的包
else:
    # 查询，为源地址添加表项
    if eth.src not in switch_table.keys(): # 不在交换机表中
        log_info(f"Add item: {eth.src}, {fromIface} to switch table")
        switch_table[eth.src] = fromIface
    elif (eth.src in switch_table.keys()) and (switch_table[eth.src] != fromIface): # 在交换机表中，但端口改变了
        log_info(f"Update item: {eth.src}, {fromIface} in switch table")
        switch_table[eth.src] = fromIface
    # 查询交换机表中有无目的地址表项，没有则泛洪
    if (eth.dst not in switch_table.keys() or eth.dst == "ff:ff:ff:ff:ff:ff"): # 未找到表项/接收到广播帧，向接收端口之外的所有端口进行泛洪
        for intf in my_interfaces:
            if fromIface != intf.name:
                log_info(f"Not Found or Received a boardcast, so flooding packet {packet} to {intf.name}")
                net.send_packet(intf, packet)
        else:
            # 找到了对应表项且不为广播帧，直接查询对应表项转发到相应端口
            log_info(f"Found item: {eth.dst}, {switch_table[eth.dst]} in switch table, and sent packet to {eth.dst}")
            net.send_packet(switch_table[eth.dst], packet)
net.shutdown()
```

所有交换机实现的核心代码都来源于 myswitch.py，无论是基于LRU算法还是超时机制亦或是流量机制，都是在最基本的 basic switch 基础上稍加拓展进行实现的。

6.总结与感想

总结：

- 🧠掌握了基本的交换机工作原理，并利用代码进行了实际实现，加深了理论课上有关交换机自学习的理解和记忆；
- 🧠熟悉并且掌握了三种拓展的交换机在空间限制下的处理机制，接触了最为常用的超时机制，并且对其进行了一一实现；
- 🧠通过任务驱动，通读和精读了 switchyard 框架文档，熟悉了常用API和相关知识，对于Lab1中陌生的代码有了不少新的理解，并且也掌握了 switchyard 框架的基本使用；
- 🧠通过完成选做任务，自己编写了不少的 testscenario 对自实现器件进行了正确性检验，对于 testsecnario 的制作有了更深的熟悉和掌握；
- 🧠本次实验无论是实验设计还是制作，都有了一个完整带的流程：逻辑实现->正确性测试->报告撰写，对于计算机网络中交换机相关的知识有了更深刻的理解，并且结合ARP协议，对于不同层次之间协同工作有了新的体会；

感想：

- 👉 用惯了C++静态类型语言，对于Python这类动态的、包装了一大堆方法的编程语言使用起来并算不上顺手，近期还需要抽时间复习和学习Python语言；
- 👉 对 `switchyard` 框架有了不少新的认识，尤其是其集成的测试功能的便捷，让人不禁感慨其方便，并且无论是利用 `switchyard` 实现功能逻辑还是进行测试，两者都是很重要的；
- 👉 跟着实验说明一步一步走，仔细阅读总没有错，遇到问题了首先翻读实验说明；
- 👉 交换机自学习机制看起来非常强大，但实际上其基本逻辑又是简洁明了的，有一种简洁美（就像一些伟大的数学公式😄）
- 👉 写测试用例时自己添加题示信息的那一堆堆字符串写得人麻了x