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：

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	202220013	姓名	徐简
Email	161200063@smail.nju.edu.cn	开始/完成日期	2021.3.24-2021.4.7

1. 实验名称

Lab 2: Learning Switch

2. 实验目的

1. 了解交换机的功能
2. 了解交换机学习的几种机制
3. 了解并实现一个交换机

3. 实验内容、代码与结果

Step 1: Basic Switch

1. 定义一个字典，用来存储Switch的学习表。

```
table={} # init a learning table
# key:address value:port
```

2. 每次接收到一个包，就将源地址和端口存入学习表中。如果拓扑发生变化，这段代码就可以记录新的拓扑信息，如果没有发生变化，则没有影响。

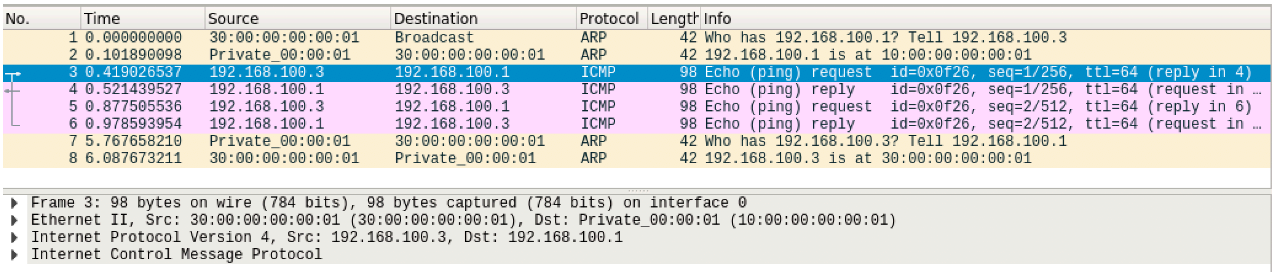
```
table[eth.src]=fromIface
```

3. 交换机的数据转发逻辑如下：如果包的目的地是交换机本身，则do nothing，如果目的地在学习表中，则根据表项进行转发；如果不在表中，则进行洪泛。

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

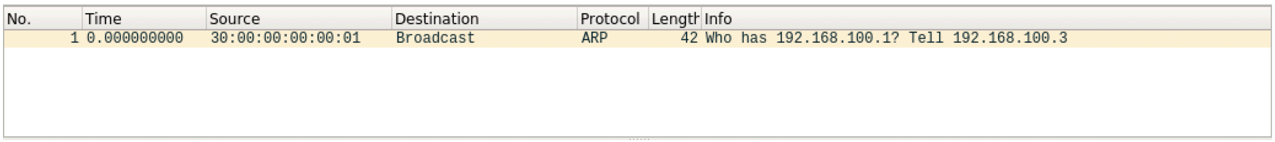
```
if eth.dst in mymacs:
  log_info("Received a packet intended for me")
elif eth.dst in table:
  output_port=table[eth.dst]
  log_debug("Forward packet {} to {}".format(packet,output_port))
  net.send_packet(output_port,packet)
else:
  for intf in my_interfaces:
    if fromIface!= intf.name:
    log_info (f"Flooding packet {packet} to {intf.name}")
    net.send_packet(intf, packet)
```

4. 在Mininet中运行该交换机，并用Wireshark进行数据包抓取，`ping -c 192.168.100.1`。下面两张图分别是server1和server2的抓包结果。



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	0.101890098	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	0.419026537	192.168.100.3	192.168.100.1	ICMP	98	Echo (ping) request id=0x0f26, seq=1/256, ttl=64 (reply in 4)
4	0.521439527	192.168.100.1	192.168.100.3	ICMP	98	Echo (ping) reply id=0x0f26, seq=1/256, ttl=64 (request in …
5	0.877505536	192.168.100.3	192.168.100.1	ICMP	98	Echo (ping) request id=0x0f26, seq=2/512, ttl=64 (reply in 6)
6	0.978593954	192.168.100.1	192.168.100.3	ICMP	98	Echo (ping) reply id=0x0f26, seq=2/512, ttl=64 (request in …
7	5.767658210	Private_00:00:01	30:00:00:00:00:01	ARP	42	Who has 192.168.100.3? Tell 192.168.100.1
8	6.087673211	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 3: 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 (10:00:00:00:00:01)
Internet Protocol Version 4, Src: 192.168.100.3, Dst: 192.168.100.1
Internet Control Message Protocol



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

5. 抓包结果的分析如下：

图一中，在发送第一个包时，client的MAC地址和其对应的端口被存入了学习表中。因此在server1回复时，目的地址直接就是client的MAC地址，实现了准确的发送。同时server1的MAC 地址和对应的端口也被录入了表中，故之后的包都是准确转发的。图二中，可以发现server2只能收到洪泛广播。

Step 2: Timeouts

1. 首先定义一个支持超时机制的学习表table。

```
table={}
# key:address value:[port,timestamp]
```

2. 每次while循环，都要遍历学习表，并删除超时的表项。判断超时通过当前时间戳与记录时间戳的差决定。

```
t=time.time()
eth = packet.get_header(Ethernet)
table[eth.src]=[fromIface,t]
for mac in list(table.keys()):
  if ((t-table[mac][1])>10):
    del table[mac]
```

3. 每接收到一个packet，记录源端口和当前的时间。

```
table[eth.src]=[fromIface,t]
```

4. 转发逻辑与基础交换机一致：如果包的目的地是交换机本身，则do nothing，如果目的地在学习表中，则根据表项进行转发；如果不在表中，则进行洪泛。

```
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")
elif eth.dst in table:
    output_port=table[eth.dst][0]
    log_debug("Forward packet {} to {}".format(packet,output_port))
    net.send_packet(output_port,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)
```

5. `syward -t testcases/myswitch_to_testscenario.srpy myswitch_to.py` 并通过测试提供的测试样例。

```
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!
```

6. 使用自己编写的测试用例进行测试，测试用例模拟的场景如下：

- 发送一个包，将其与端口eth1对应，并存入学习表
- 向之前步骤的源地址，发送一个包，观察是否被正确转发
- 向一个陌生的地址，发送一个包，观察是否被洪泛处理
- 等待20s，测试超时机制
- 向最初记录的地址，发送包，如果被洪泛则说明超时机制起作用

测试结果如图：

```
Results for test scenario switch_to tests: 10 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 be
    forwarded out eth1
5   The switch should not do anything after it sends packet out
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   wait for time out
9   An Ethernet frame from 20:00:00:00:00:01 to
    30:00:00:00:00:02 should arrive on eth0
10  Ethernet frame destined for 30:00:00:00:00:02 should be
    flooded out


All tests passed!
```

6. 在Mininet中运行该交换机，并用Wireshark进行数据包抓取。图片从上到下分别为server1，server2.

   在client上连续两次`ping -c 1 192.168.100.1`等待两分钟后，再次在client上 `ping -c 1 192.168.100.1`

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	0.107356200	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	0.618839832	192.168.100.3	192.168.100.1	ICMP	98	Echo (ping) request id=0x14c5, seq=1/256, ttl=64 (reply in 4)
4	0.719120718	192.168.100.1	192.168.100.3	ICMP	98	Echo (ping) reply id=0x14c5, seq=1/256, ttl=64 (request in …
5	5.971180357	Private_00:00:01	30:00:00:00:00:01	ARP	42	Who has 192.168.100.3? Tell 192.168.100.1
6	6.386427940	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	104.636383814	192.168.100.3	192.168.100.1	ICMP	98	Echo (ping) request id=0x14c8, seq=1/256, ttl=64 (reply in 8)
8	104.741096332	192.168.100.1	192.168.100.3	ICMP	98	Echo (ping) reply id=0x14c8, seq=1/256, ttl=64 (request in …
9	109.819397407	30:00:00:00:00:01	Private_00:00:01	ARP	42	Who has 192.168.100.1? Tell 192.168.100.3
10	109.908511981	Private_00:00:01	30:00:00:00:00:01	ARP	42	Who has 192.168.100.3? Tell 192.168.100.1
11	109.920433501	Private_00:00:01	30:00:00:00:00:01	ARP	42	192.168.100.1 is at 10:00:00:00:00:01
12	110.258774469	30:00:00:00:00:01	Private_00:00:01	ARP	42	192.168.100.3 is at 30:00:00:00:00:01

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	104.636387200	192.168.100.3	192.168.100.1	ICMP	98	Echo (ping) request id=0x14c8, seq=1/256, ttl=64 (no response…

可以发现在等待一段时间后，server2又收到了消息，说明之前记录的server1的端口信息，已经因为超时而被删除，因此发送给server1的packet又被洪泛转发了。

Step 3: Least Recently Used

1. 定义支持LRU机制的学习表

```
table={}
#key: address value:[port,age]
```

2. 每次循环，将学习表中的每个表项的age加一。

```
for md in table:
  table[md][1]+=1
```

3. 判断packet的源地址在不在表内，并修改更新，更新的时候不改变原来的age。

```
if eth.src in table:
    table[eth.src][0]=fromIface
```

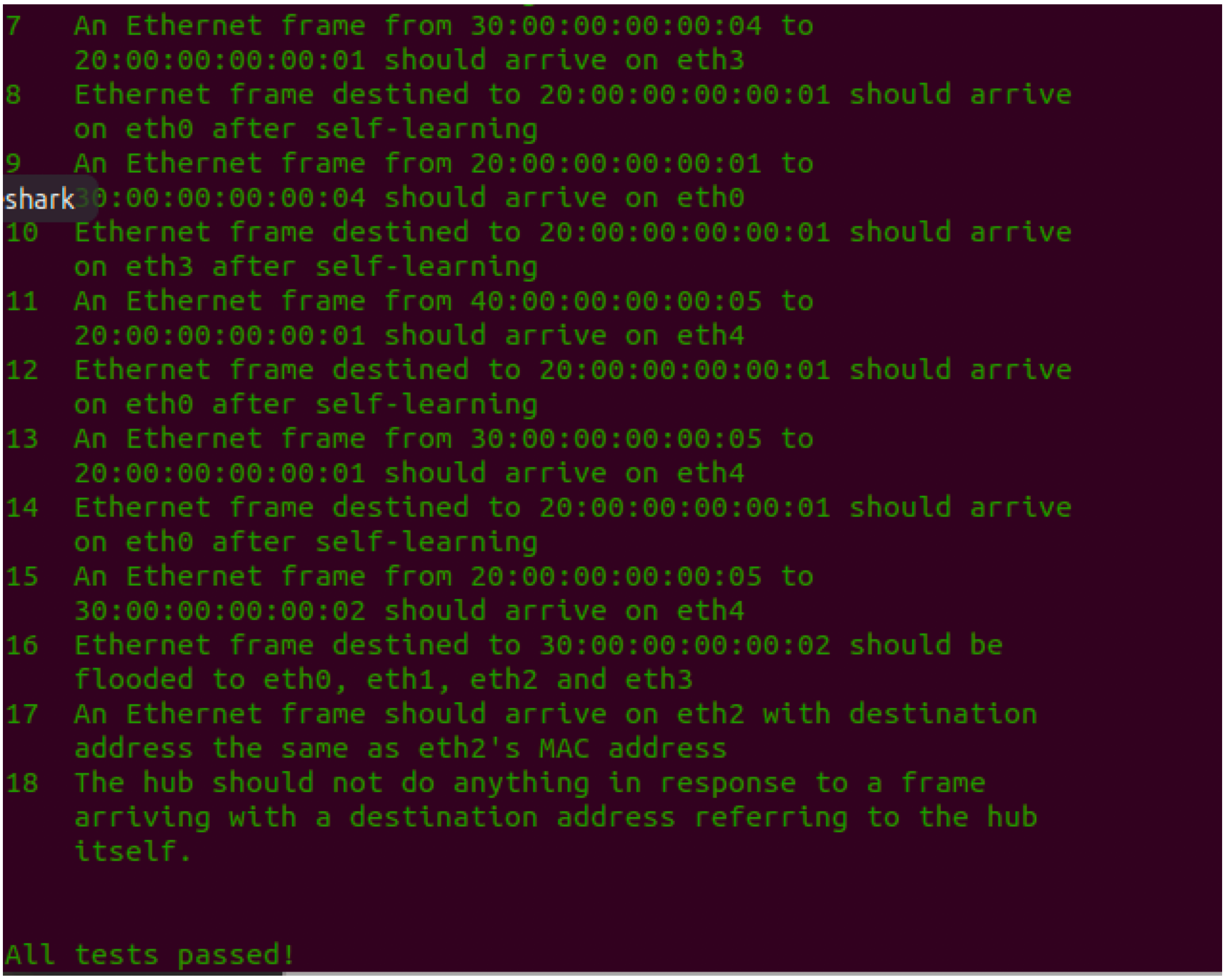
4. 如果不在，根据表是否满进一步处理。如果没有满，直接插入新的表项。如果已经满了，则根据LRU的原则，选择age最大的一项，删除，再插入新的表项，初始age均为0.

```
elif len(table)<Max_num:
    table[eth.src]=[fromIface,0]
else:
    lru_key=list(table.keys())[0]
    for key in table:
        if table[key][1]>table[lru_key][1]:
            lru_key=key
    del table[lru_key]
    table[eth.src]=[fromIface,0]
```

5. 转发逻辑与之前类似，需要注意的是，要将目的地的age更新为0.

```
elif eth.dst in table:
    out_port = table[eth.dst][0]
    table[eth.dst][1]=0
    log_debug("Forward packet {} to {}".format(packet,out_port))
    net.send_packet(out_port,packet)
```

6. `syward -t testcases/myswitch_lru_testscenario.srpy myswitch_lru.py` 并通过测试提供的测试样例。



7. 使用自己编写的测试用例进行测试，测试用例模拟的场景如下：

- 分别从5个不同地址发送包，将学习表填满
- 从表中某地址发送一个包，目的地也为表中地址，观察是否精确转发
- 从一个新的地址，向表中age最大的地址发送包，由于表满，age最大的会被删除，观察是否被洪泛转发

测试结果如图：

```
3   An Ethernet frame from 3:00:00:00:00:02 to 40:00:00:00:00:01
    should arrive on eth0
4   Ethernet frame destined for 40:00:00:00:00:01 should be
    flooded out eth1 and eth2
5   An Ethernet frame from 30:00:00:00:00:03 to
    40:00:00:00:00:01 should arrive on eth1
6   Ethernet frame destined for 40:00:00:00:00:01 should be
    flooded out eth0 and eth2
7   An Ethernet frame from 30:00:00:00:00:04 to
    40:00:00:00:00:01 should arrive on eth1
8   Ethernet frame destined for 40:00:00:00:00:01 should be
    flooded out eth0 and eth2
9   An Ethernet frame from 30:00:00:00:00:05 to
    40:00:00:00:00:01 should arrive on eth1
10  Ethernet frame destined for 40:00:00:00:00:01 should be
    flooded out eth0 and eth1
11  An Ethernet frame from 30:00:00:00:00:03 to
    30:00:00:00:00:01 should arrive on eth1
12  Ethernet frame destined for 40:00:00:00:00:01 should be
    forwarded out eth0
13  The switch should not do anything after it sends packet out
    frome eth0
14  An Ethernet frame from 30:00:00:00:00:06 to
    30:00:00:00:00:02 should arrive on eth2
15  Ethernet frame destined for 30:00:00:00:00:02 should be
    flooded out


All tests passed!
```

8. 在Mininet中运行该交换机，并用Wireshark进行数据包抓取。图片从上到下分别为client，server1，server2.为了简化测试，将学习表的容量暂时设置为2
    - 在client上键入指令 ping -c 1 192.168.100.1
    - 在server2上键入指令 ping -c 1 192.168.100.3
    - 在server2上键入指令 ping -c 1 192.168.100.1

1	0.000000000	30:00:00:00:00:01	Broadcast	ARP	42	Who has 192.168.100.1? Tell 192.168.100.3
2	0.477738467	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	0.580687437	192.168.100.3	192.168.100.1	ICMP	98	Echo (ping) request id=0x17f2, seq=1/256, ttl=64 (reply in 4)
4	0.893673909	192.168.100.1	192.168.100.3	ICMP	98	Echo (ping) reply id=0x17f2, seq=1/256, ttl=64 (request in …
5	6.066136235	Private_00:00:01	30:00:00:00:00:01	ARP	42	Who has 192.168.100.3? Tell 192.168.100.1
6	6.166684356	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	55.992292885	20:00:00:00:00:01	Broadcast	ARP	42	Who has 192.168.100.3? Tell 192.168.100.2
8	56.116973764	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	56.551062758	192.168.100.2	192.168.100.3	ICMP	98	Echo (ping) request id=0x17f4, seq=1/256, ttl=64 (reply in 10)
10	56.652332866	192.168.100.3	192.168.100.2	ICMP	98	Echo (ping) reply id=0x17f4, seq=1/256, ttl=64 (request in …
11	61.905523404	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	62.387563453	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
13	64.064691456	20:00:00:00:00:01	Broadcast	ARP	42	Who has 192.168.100.1? Tell 192.168.100.2
14	64.270767947	Private_00:00:01	20:00:00:00:00:01	ARP	42	192.168.100.1 is at 10:00:00:00:00:01

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	0.120607208	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	0.533317717	192.168.100.3	192.168.100.1	ICMP	98	Echo (ping) request id=0x17f2, seq=1/256, ttl=64 (reply in 4)
4	0.633422727	192.168.100.1	192.168.100.3	ICMP	98	Echo (ping) reply id=0x17f2, seq=1/256, ttl=64 (request in …
5	5.688247197	Private_00:00:01	30:00:00:00:00:01	ARP	42	Who has 192.168.100.3? Tell 192.168.100.1
6	6.129487970	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	55.840354182	20:00:00:00:00:01	Broadcast	ARP	42	Who has 192.168.100.3? Tell 192.168.100.2
8	63.912749018	20:00:00:00:00:01	Broadcast	ARP	42	Who has 192.168.100.1? Tell 192.168.100.2
9	64.014830242	Private_00:00:01	20:00:00:00:00:01	ARP	42	192.168.100.1 is at 10:00:00:00:00:01
10	64.338543039	192.168.100.2	192.168.100.1	ICMP	98	Echo (ping) request id=0x17f7, seq=1/256, ttl=64 (reply in 11)
11	64.438867837	192.168.100.1	192.168.100.2	ICMP	98	Echo (ping) reply id=0x17f7, seq=1/256, ttl=64 (request in …
12	69.689995139	Private_00:00:01	20:00:00:00:00:01	ARP	42	Who has 192.168.100.2? Tell 192.168.100.1
13	70.165019572	20:00:00:00:00:01	Private_00:00:01	ARP	42	192.168.100.2 is at 20:00:00:00:00:01

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	55.724331100	20:00:00:00:00:01	Broadcast	ARP	42	Who has 192.168.100.3? Tell 192.168.100.2
3	56.177568312	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
4	56.278032314	192.168.100.2	192.168.100.3	ICMP	98	Echo (ping) request id=0x17f4, seq=1/256, ttl=64 (reply in 5)
5	56.709907064	192.168.100.3	192.168.100.2	ICMP	98	Echo (ping) reply id=0x17f4, seq=1/256, ttl=64 (request in …
6	61.913853505	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
7	62.026504079	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
8	63.752019864	20:00:00:00:00:01	Broadcast	ARP	42	Who has 192.168.100.1? Tell 192.168.100.2
9	64.118607183	Private_00:00:01	20:00:00:00:00:01	ARP	42	192.168.100.1 is at 10:00:00:00:00:01
10	64.219598408	192.168.100.2	192.168.100.1	ICMP	98	Echo (ping) request id=0x17f7, seq=1/256, ttl=64 (reply in 11)
11	64.552084472	192.168.100.1	192.168.100.2	ICMP	98	Echo (ping) reply id=0x17f7, seq=1/256, ttl=64 (request in …
12	69.852425324	Private_00:00:01	20:00:00:00:00:01	ARP	42	Who has 192.168.100.2? Tell 192.168.100.1
13	69.956948239	20:00:00:00:00:01	Private_00:00:01	ARP	42	192.168.100.2 is at 20:00:00:00:00:01

在client ping server1后，学习表中储存了二者的MAC地址和对应端口。下一步在server2 ping，要将server2的对应信息要存入表中，因此删除了age最大的client，因此这个包被洪泛转发。之后server2 ping server1时，在请求时由于表中没有server1，故这个请求消息也被洪泛处理，server1收到请求，后回复server2，但由于表满，server2 age大， 故server2被踢出表，server1发送的回复被洪泛处理，client也能收到。

Step 4: Least Traffic Volume

1. 定义支持LTV的学习表

```
table={}
#key:address value:port,traffic volume
```

2. 判断packet的源地址在不在表内，并修改更新，更新的时候不改变原来的traffic volume。

```
if eth.src in table:
  table[eth.src][0]=fromIface
```

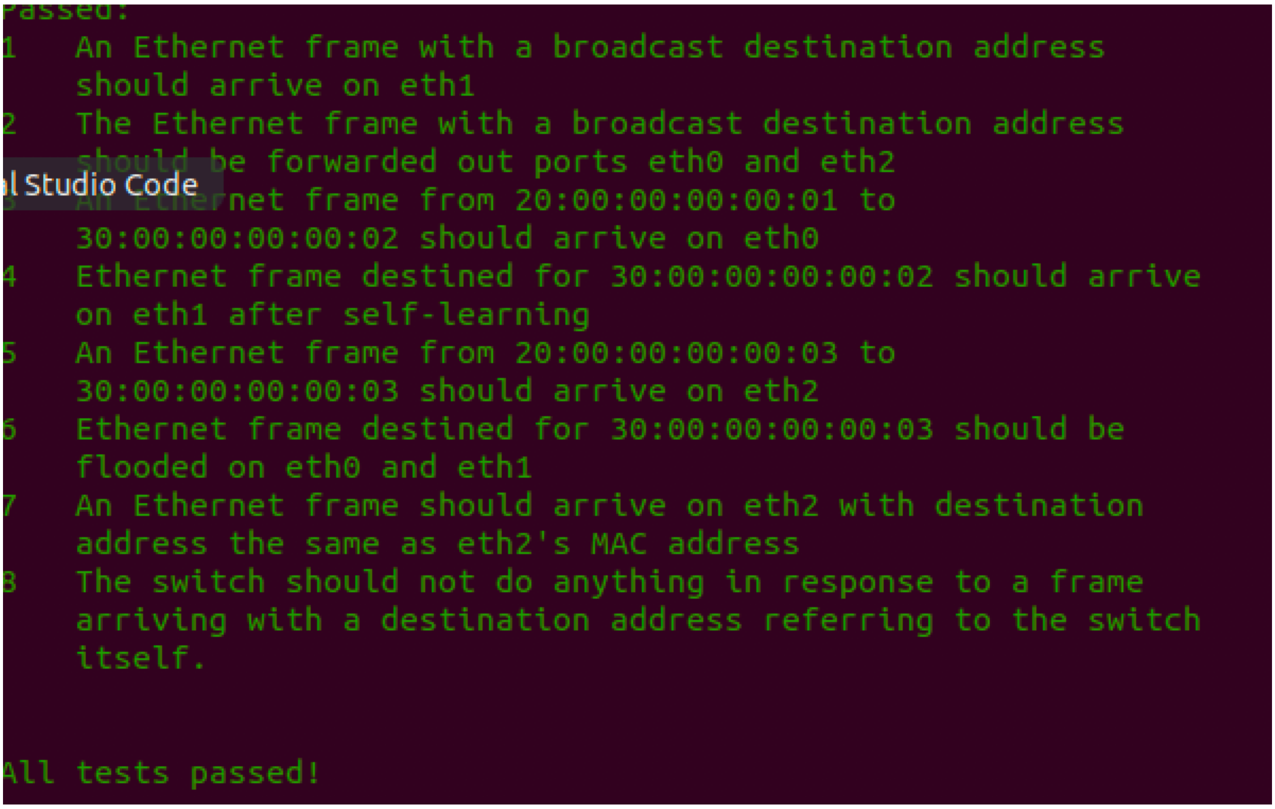
3. 如果不在，根据表是否满进一步处理。如果没有满，直接插入新的表项。如果已经满了，则根据LRU的原则，选择traffic volume最小的一项，删除，再插入新的表项，初始age均为0.

```
elif len(table) <Max_num:
    table[eth.src]=[fromIface,0]
else:
    lt_key=list(table.keys())[0]
    for key in table:
        if table[key][1]<table[lt_key][1]:
            lt_key=key
    del table[lt_key]
    table[eth.src]=[fromIface,0]
```

4. 转发逻辑与之前类似，需要注意的是，要将目的地的traffic volume加一.

```
out_port=table[eth.dst][0]
table[eth.dst][1]+=1
log_debug("Forward packet {} to {}".format(packet,out_port))
net.send_packet(out_port,packet
```

5. `syward -t testcases/myswitch_traffic_testscenario.srpy myswitch_traffic.py` 并通过测试提供的测试样例。



5. 使用自己编写的测试用例进行测试，测试用例模拟的场景如下：

- 分别从5个不同地址发送包，将学习表填满
- 发送目的地为表中地址的四个包，将表项中四项的traffic volume变成2
- 从一个新的地址，向表中traffic volume最小的地址发送包，由于表满，traffic volume最小的会被删除，观察是否被洪泛转发

测试结果如图：

```
12  Ethernet frame destined for 30:00:00:00:00:01 should be
    forwarded out eth0
13  The switch should not do anything after it sends packet out
    frome eth0
14  An Ethernet frame from 30:00:00:00:00:03 to
    30:00:00:00:00:02 should arrive on eth1
15  Ethernet frame destined for 30:00:00:00:00:02 should be
    forwarded out eth0
16  The switch should not do anything after it sends packet out
    frome eth0
17  An Ethernet frame from 30:00:00:00:00:01 to
    30:00:00:00:00:03 should arrive on eth0
18  Ethernet frame destined for 30:00:00:00:00:03 should be
    forwarded out eth1
19  The switch should not do anything after it sends packet out
    frome eth1
20  An Ethernet frame from 30:00:00:00:00:01 to
    30:00:00:00:00:04 should arrive on eth0
21  Ethernet frame destined for 30:00:00:00:00:04 should be
    forwarded out eth1
22  The switch should not do anything after it sends packet out
    frome eth1
23  An Ethernet frame from 30:00:00:00:00:06 to
    30:00:00:00:00:05 should arrive on eth0
24  Ethernet frame destined for 30:00:00:00:00:05 should be
    flooded out


All tests passed!
```

7. 在Mininet中运行该交换机，并用Wireshark进行数据包抓取。为了简化，暂时设置表容量为2。图片从上到下分别为client，server1，server2。

- 在server1上 ping -c 1 192.168.100.3
- 在server2上 ping -c 1 192.168.100.3
- 在server1上 ping -c 1 192.168.100.2

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	0.100270725	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	0.420896754	192.168.100.1	192.168.100.3	ICMP	98	Echo (ping) request id=0x1afc, seq=1/256, ttl=64 (reply in 4)
4	0.524737685	192.168.100.3	192.168.100.1	ICMP	98	Echo (ping) reply id=0x1afc, seq=1/256, ttl=64 (request in …
5	5.656232590	30:00:00:00:00:01	Private_00:00:01	ARP	42	Who has 192.168.100.1? Tell 192.168.100.3
6	5.992644899	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	[illegible]	20:00:00:00:00:01	Broadcast	ARP	42	Who has 192.168.100.3? Tell 192.168.100.2
8	9.940122832	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	10.472900789	192.168.100.2	192.168.100.3	ICMP	98	Echo (ping) request id=0x1afe, seq=1/256, ttl=64 (reply in 10)
10	10.572971924	192.168.100.3	192.168.100.2	ICMP	98	Echo (ping) reply id=0x1afe, seq=1/256, ttl=64 (request in …
11	15.630671377	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	16.054180518	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
13	25.785367375	Private_00:00:01	Broadcast	ARP	42	Who has 192.168.100.2? Tell 192.168.100.1

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	0.335647341	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	0.436359120	192.168.100.1	192.168.100.3	ICMP	98	Echo (ping) request id=0x1afc, seq=1/256, ttl=64 (reply in 4)
4	0.758775629	192.168.100.3	192.168.100.1	ICMP	98	Echo (ping) reply id=0x1afc, seq=1/256, ttl=64 (request in …
5	5.906317216	30:00:00:00:00:01	Private_00:00:01	ARP	42	Who has 192.168.100.1? Tell 192.168.100.3
6	6.007234299	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	9.939635884	20:00:00:00:00:01	Broadcast	ARP	42	Who has 192.168.100.3? Tell 192.168.100.2
8	10.281280815	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	10.592272719	192.168.100.2	192.168.100.3	ICMP	98	Echo (ping) request id=0x1afe, seq=1/256, ttl=64 (reply in 10)
10	10.810465283	192.168.100.3	192.168.100.2	ICMP	98	Echo (ping) reply id=0x1afe, seq=1/256, ttl=64 (request in …
11	[illegible]	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	[illegible]	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
13	25.742922776	Private_00:00:01	Broadcast	ARP	42	Who has 192.168.100.2? Tell 192.168.100.1
14	26.110735485	20:00:00:00:00:01	Private_00:00:01	ARP	42	192.168.100.2 is at 20:00:00:00:00:01
15	26.210824788	192.168.100.1	192.168.100.2	ICMP	98	Echo (ping) request id=0x1b00, seq=1/256, ttl=64 (reply in 16)
16	26.526857075	192.168.100.2	192.168.100.1	ICMP	98	Echo (ping) reply id=0x1b00, seq=1/256, ttl=64 (request in …
17	31.744642168	20:00:00:00:00:01	Private_00:00:01	ARP	42	Who has 192.168.100.1? Tell 192.168.100.2
18	31.846147339	Private_00:00:01	20:00:00:00:00:01	ARP	42	192.168.100.1 is at 10:00:00:00:00:01

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	9.598167712	20:00:00:00:00:01	Broadcast	ARP	42	Who has 192.168.100.3? Tell 192.168.100.2
3	10.161905425	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
4	10.262687677	192.168.100.2	192.168.100.3	ICMP	98	Echo (ping) request id=0x1afe, seq=1/256, ttl=64 (reply in 5)
5	10.691090205	192.168.100.3	192.168.100.2	ICMP	98	Echo (ping) reply id=0x1afe, seq=1/256, ttl=64 (request in …
6	15.837286168	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
7	15.937738142	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
8	25.785274989	Private_00:00:01	Broadcast	ARP	42	Who has 192.168.100.2? Tell 192.168.100.1
9	25.889871482	20:00:00:00:00:01	Private_00:00:01	ARP	42	192.168.100.2 is at 20:00:00:00:00:01
10	26.200410312	192.168.100.1	192.168.100.2	ICMP	98	Echo (ping) request id=0x1b00, seq=1/256, ttl=64 (reply in 11)
11	26.301911132	192.168.100.2	192.168.100.1	ICMP	98	Echo (ping) reply id=0x1b00, seq=1/256, ttl=64 (request in …
12	31.498340231	20:00:00:00:00:01	Private_00:00:01	ARP	42	Who has 192.168.100.1? Tell 192.168.100.2
13	31.941638460	Private_00:00:01	20:00:00:00:00:01	ARP	42	192.168.100.1 is at 10:00:00:00:00:01

server1 ping client后，两者对应信息被存入学习表，因此除了第一条被广播，其他的都被准确转发。当server2 ping client时，流量低的client被删除，添加server2，故这些消息被洪泛处理，client回复的时候要加入表中，再把流量低的server2删除，故此次回复又被洪泛处理。之后的ARP包的发送使表中存储的是server2和client的对应信息。 故server1 ping server2时，server2在表中，request被准确转发，server2回复的时候由于server1已经进来把client替换了，故回复消息也被准确转发。

4. 总结与感想

本次实验的switch逻辑实现部分其实并不困难，按照手册的流程图可以顺利的实现。其中比较有意思的是广播逻辑的实现，还是很巧妙的。因为不会从广播发出packet，所以当目的地为全F时，转发逻辑会自动进入最后一个else，从而实现向每个端口的广播。这样避免了广播的特殊处理，值得思考。

此次实验主要难点在于利用 Mininet进行测试这一环节。如何设计测试流程，以及如何解读Wireshark里的繁杂抓包条目都是比较困难的。但是从条目中，观察验证自己代码的正确性，还是很有趣的，也加深了对于互相通信的过程的理解。