

计算机网络实验二

181840135 梁俊凯

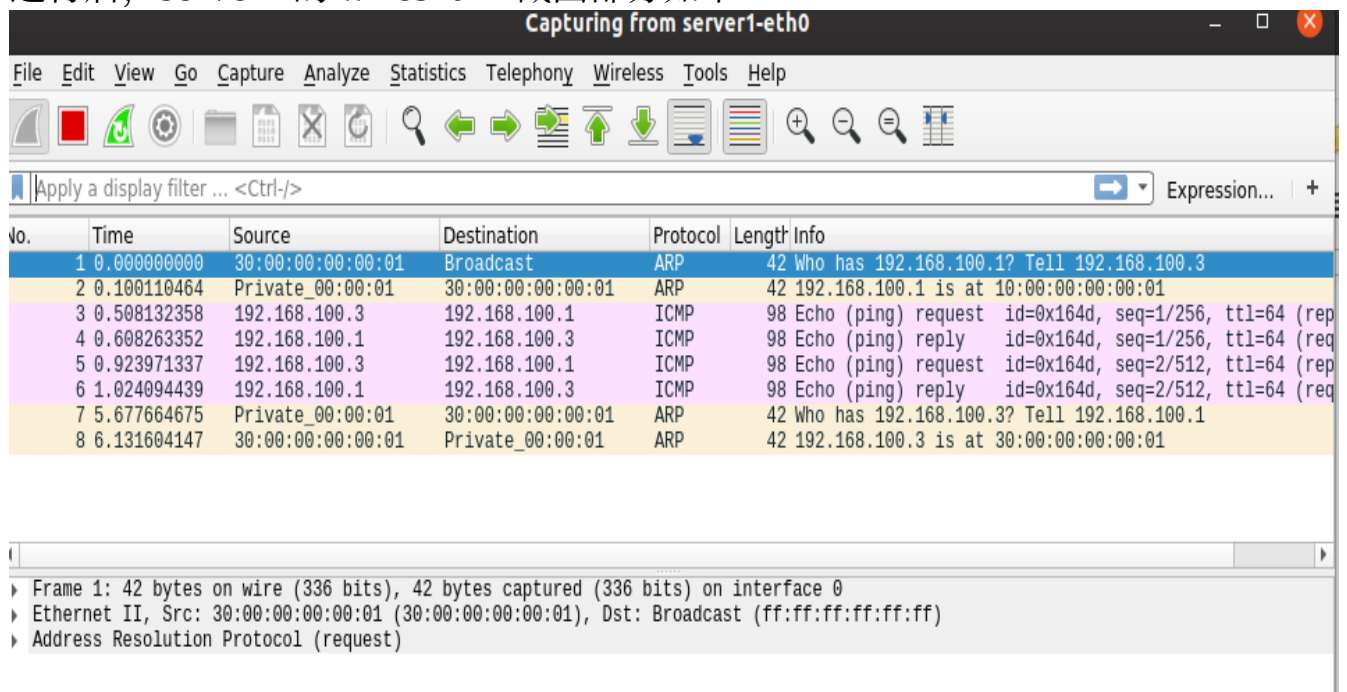
Task1

task1 为进行后续实验前的准备工作，我按照要求复制了文件和 test 文件并进行了 commit

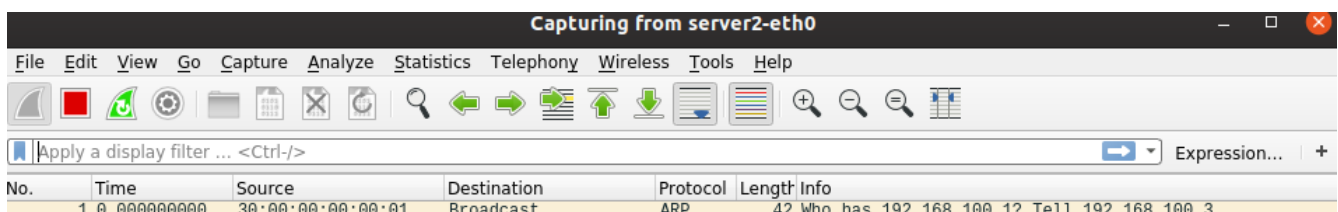
Task2

我实现了交换机初步的自学习代码，当收到包时，将 [包的源 mac 地址：到来的端口]记录在一个字典里，然后根据包的目的 mac 地址，如果在字典中有记录，则仅通过记录的端口号进行转发（Transport），若无记录，则在所有端口进行转发（Flood，除去到来的那个端口）。

运行后，server1 的 wireshark 截图部分如下



可以看到 server1 在开始和结束共进行了四次 ARP，在第一次广播后，switch 已经把 client 和 server1 所在的 mac 地址和相对应的端口记录在表中，理论上以后的信息传递 server2 将不在收到，我们接下来查看 server2 的 wireshark 抓包结果作为比对。



可以看到，server2 仅收到一个开始时的广播地址包，证明关于自学习的交换机初步实现是正确的。

同时我们还应查看 switch 界面输出的信息 (log_info) 观察我们的记录表，在最终状态下应该仅有 client 和 server1 两项。

```
my_memory: {EthAddr('30:00:00:00:00:01'): 'switch-eth2', EthAddr('10:00:00:00:00:01'): 'switch-eth0'}
```

这里的 my_memory 是最后输出的交换机记录表，可以看到 client(30:00:00:00:00:01)和 server1(10:00:00:00:00:01)两项对应的端口被忠实记录了下来，符合我们的预期。

我暂时还未理解使用 ip 地址的意义，在我现有的知识（链路层）看来，仅使用 mac 地址是可以完成包的发送的，因此我未理解 server1 的 wireshark 抓包结果中有关 ip 地址的广播的作用（找寻相对应的 mac 地址？）

本次实验所给出的相关 wireshark 文件均在 lab_2 目录内，命名为 task1_x+后缀

Task3

交换机的转发，自学习的思路与 Task2 类似，但增添了老化期的功能，即在一定时间后（实验中为 10 秒）将删除 10 秒内未经交换机转发的 mac 地址的表项，我使用了 python 的 time 库，通过 time.time()函数获取当前时间戳。我的算法流程如下：

while 循环：

 time_decay()//老化期处理函数

 若收到包：

 自学习函数

 包的转发

这里的老化期处理是无时无刻不在进行的（未收到包时，因为此时 resv_packet 返回 No_packets，所以 while 循环仅仅在不断的读取时间，并根据时间决定是否要删除相应的表项），在这样的思路下，我通过了 switchtests_to.srpy 的测试

```
ljk@ljk-ThinkPad-13: ~/Network_project/assignment-2-AA-sta...
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) ljk@ljk-ThinkPad-13:~/Network_project/assignment-2-AA-stardust/lab_2$
```

接下来，在自己的 mininet 模式下，我又运行了 myswitch_to.py，结果如下

```
14:38:17 2020/10/11      INFO
Transprot log : {EthAddr('30:00:00:00:00:01'): ['switch-eth2', 1602398297.777902
], EthAddr('10:00:00:00:00:01'): ['switch-eth0', 1602398297.8581576]}

14:38:17 2020/10/11      INFO Transport packet Ethernet 10:00:00:00:00:01->30:00:
00:00:00:01 ARP | Arp 10:00:00:00:00:01:192.168.100.1 30:00:00:00:00:01:192.168.
100.3 to switch-eth2
14:38:43 2020/10/11      INFO In ljk-ThinkPad-13 received packet Ethernet 30:00:0
0:00:00:01->10:00:00:00:00:01 IP | IPv4 192.168.100.3->192.168.100.1 ICMP | ICMP
EchoRequest 13857 1 (56 data bytes) on switch-eth2
14:38:43 2020/10/11      INFO
my_momory: {EthAddr('30:00:00:00:00:01'): ['switch-eth2', 1602398323.749945]}
```

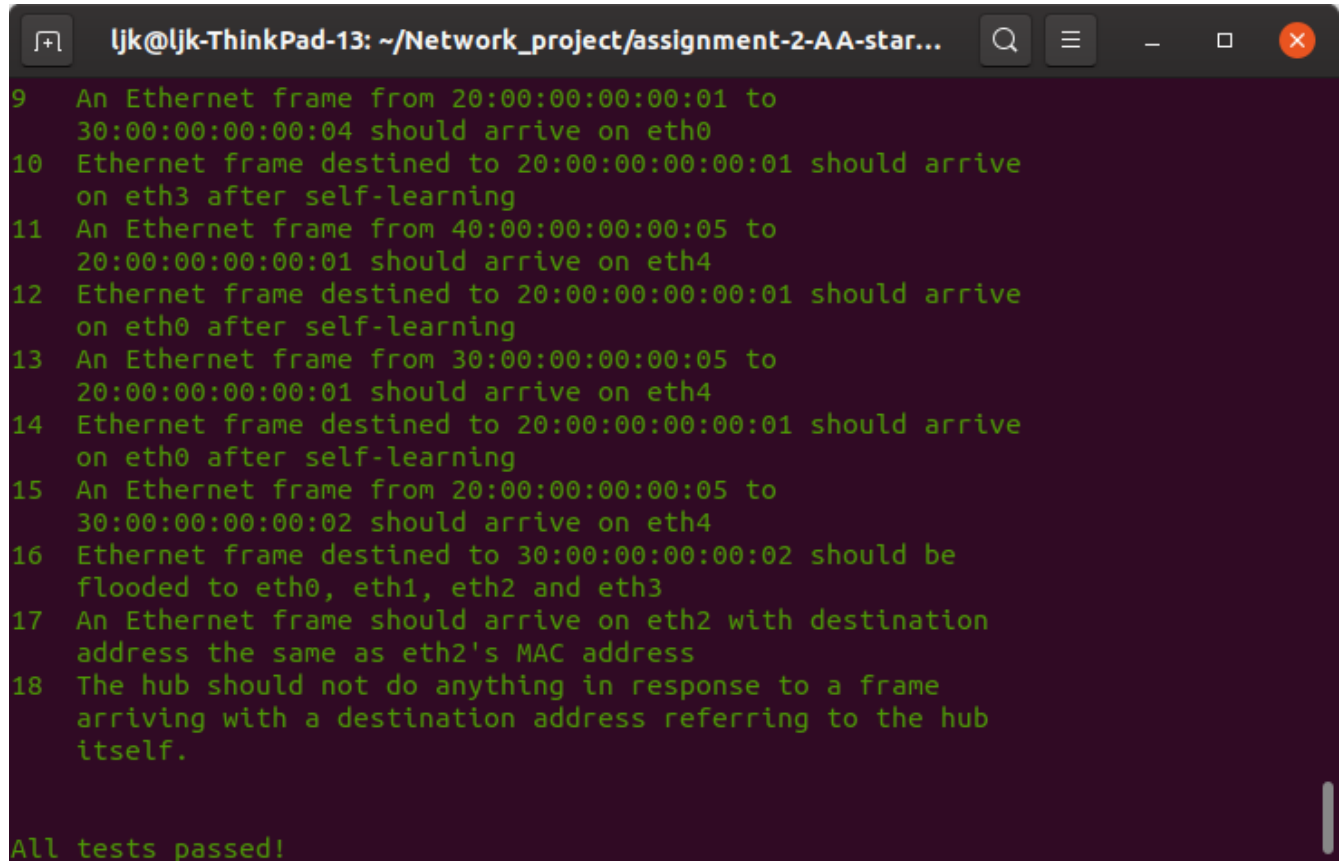
可以稍作分析一下，初始时我在 client 端执行 `ping -c 2 192.168.100.1`（比 `pingall` 包少一些，输出的关键信息更易找到）在 14:38:17 时我进行等待，此时 Transprot log 中打印出了转发表的内容，有两项。在 14:38:43 后我继续 ping 此时在转发表自学习函数打印了 memory 的内容，仅为刚刚学习的那一项，证明转发表在刚刚的过程中完成老化，我的实现是基本正确的

Task4

本次我完成了 lru 替换的 switch，我没有使用 age 那个信息，因为在 FAQ 中我看到教授使用了一种更简洁的表示法，即一个先进先出的队列，当某表项被使用时，仅需将其置于队列头即可，当需要删除表项时，仅需将队列尾部的表项删除，拓

扑结构更新时，保持每个表项的位置不变，将相应 mac 地址的表项对应的端口信息更新即可，在这里，转发表每个表项的位置信息是相应的 lru 信息，所以我们不能再用无序的字典进行存储了，需要使用有序列表。

test 文件测试通过：

A terminal window with a dark background and green text. The window title is 'ljk@ljk-ThinkPad-13: ~/Network_project/assignment-2-AA-star...'. It displays 18 numbered test cases. Cases 9 through 18 describe various Ethernet frame scenarios, including self-learning, flooding, and destination matching. The final line of the terminal output is 'All tests passed!' in green.

```
ljk@ljk-ThinkPad-13: ~/Network_project/assignment-2-AA-star...
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!
```

在我的 start_mininet.py 中我也进行了测试，由于该文件中 switch 仅有三个端口，于是我把转发表的最大长度设置为 2，在终端中输入 pingall 指令，通过观察相应的输出调试信息来证明我的算法运行良好，由于信息复杂，我选择了几个算法运行的侧面来进行验证：

1>插入新表项时，需要将其置于队列头（先进先出，后进后出），信息如下：

```

my_memory: [[EthAddr('30:00:00:00:00:01'), 'switch-eth2']]

20:49:53 2020/10/13      INFO Flooding packet Ethernet 30:00:00:00:00:01->ff:ff:f
f:ff:ff:ff ARP | Arp 30:00:00:00:00:01:192.168.100.3 00:00:00:00:00:00:192.168.1
00.1 to switch-eth1
20:49:53 2020/10/13      INFO Flooding packet Ethernet 30:00:00:00:00:01->ff:ff:f
f:ff:ff:ff ARP | Arp 30:00:00:00:00:01:192.168.100.3 00:00:00:00:00:00:192.168.1
00.1 to switch-eth0
20:49:53 2020/10/13      INFO In ljk-ThinkPad-13 received packet Ethernet 10:00:0
0:00:00:01->30:00:00:00:00:01 ARP | Arp 10:00:00:00:00:01:192.168.100.1 30:00:00
:00:00:01:192.168.100.3 on switch-eth0
20:49:53 2020/10/13      INFO
my_memory: [[EthAddr('10:00:00:00:00:01'), 'switch-eth0'], [EthAddr('30:00:00:00
:00:01'), 'switch-eth2']]

```

可以看到，插入新 mac 地址 10:00:00:00:00:01 时，它插入到了表项的头部。

2>当通过转发表定点转发时，该使用到的表项将移动到表头

```

my_memory: [[EthAddr('10:00:00:00:00:01'), 'switch-eth0'], [EthAddr('30:00:00:00
:00:01'), 'switch-eth2']]

20:49:53 2020/10/13      INFO Transport packet Ethernet 10:00:00:00:00:01->30:00:
00:00:00:01 ARP | Arp 10:00:00:00:00:01:192.168.100.1 30:00:00:00:00:01:192.168.
100.3 to 30:00:00:00:00:01
20:49:53 2020/10/13      INFO memory after transport: [[EthAddr('30:00:00:00:00:0
1'), 'switch-eth2'], [EthAddr('10:00:00:00:00:01'), 'switch-eth0']]

```

可以看到当 Transport 完成后 mac 地址 30....01 对应的表项由队列尾部进入到了队列头部。

3>表满插入，删除尾部的表项，并将新表项插入到头部


```

20:49:54 2020/10/13      INFO memory after transport: [[EthAddr('30:00:00:00:00:0
1'), 'switch-eth2'], [EthAddr('10:00:00:00:00:01'), 'switch-eth0']]
20:49:54 2020/10/13      INFO In ljk-ThinkPad-13 received packet Ethernet 30:00:0
0:00:00:01->ff:ff:ff:ff:ff:ff ARP | Arp 30:00:00:00:00:01:192.168.100.3 00:00:00
:00:00:00:192.168.100.2 on switch-eth2
20:49:54 2020/10/13      INFO Flooding packet Ethernet 30:00:00:00:00:01->ff:ff:f
f:ff:ff:ff ARP | Arp 30:00:00:00:00:01:192.168.100.3 00:00:00:00:00:00:192.168.1
00.2 to switch-eth1
20:49:54 2020/10/13      INFO Flooding packet Ethernet 30:00:00:00:00:01->ff:ff:f
f:ff:ff:ff ARP | Arp 30:00:00:00:00:01:192.168.100.3 00:00:00:00:00:00:192.168.1
00.2 to switch-eth0
20:49:54 2020/10/13      INFO In ljk-ThinkPad-13 received packet Ethernet 20:00:0
0:00:00:01->30:00:00:00:00:01 ARP | Arp 20:00:00:00:00:01:192.168.100.2 30:00:00
:00:00:01:192.168.100.3 on switch-eth1
20:49:54 2020/10/13      INFO
my_momory: [[EthAddr('20:00:00:00:00:01'), 'switch-eth1'], [EthAddr('30:00:00:00
:00:01'), 'switch-eth2']]

```

可以看到，表尾 10...01 对应的表项被删去了，而新表项被插入到了表头。

这是一个笨的调试方法，但正如计算机发明初期是一个笨拙的大家伙一样，这是我的一个探索的过程，“通过观察算法运行的截面（即某个行为）”是我的核心思路，后期一定会被优化为更好的方法。

Task5

本实验与 Task4 很类似，仅仅是更新的信息方式不同，这次我使用了一个 traffic_volume 位来记录，每当一个表项被使用一次后，就将起对应的 traffic_volume+1 所以我的转发表如下：

[mac, port, traffic_volume]

按照流程图的步骤，当遇到包时，若在转发表中，就查看拓扑结构有无变化，做出相应的改变，若不在转发表中，则增添此

[src_mac, input_port, traffic_volume=0]表项，特殊地若此时表满，则删除 traffic_volume 最小的表项。在转发时若是根据转发表进行转发，则需将对应表项的 traffic_volume+1 并记录。我通过了给出的 test，展示如下：

```
ljk@ljk-ThinkPad-13: ~/Network_project/assignment-2-AA-star...  
ljk@ljk-ThinkPad-13: ~/Network_proje... x ljk@ljk-ThinkPad-13: ~/Network_proje... x  
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 An Ethernet frame from 20:00:00:00:00:03 to  
  30:00:00:00:00:03 should arrive on eth2  
6 Ethernet frame destined for 30:00:00:00:00:03 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) ljk@ljk-ThinkPad-13:~/Network_project/assignment-2-AA-stardust/lab_2$
```

在自己的 mininet 上运行如下，我继续采用 Task4 中使用的算法截面调试法

1>每当使用转发表“精准”转发时，相应表项的 traffic_volume+1

```
21:20:04 2020/10/13      INFO Transport packet Ethernet 30:00:00:00:00:01->10:00:00:00:00:01 ARP | Arp 30:00:00:00:00:01:192.168.100.3 10:00:00:00:00:01:192.168.100.1 to 10:00:00:00:00:01  
21:20:04 2020/10/13      INFO memory after transport: [[EthAddr('30:00:00:00:00:01'), 'switch-eth2', 0], [EthAddr('10:00:00:00:00:01'), 'switch-eth0', 1]]
```

可以看到，10...01 对应的 traffic_volume 从 0 变为 1

2>表满时，删除 traffic_volume 较小的一项

```
my_momory: [[EthAddr('10:00:00:00:00:01'), 'switch-eth0', 1], [EthAddr('20:00:00:00:00:01'), 'switch-eth1', 0]]

21:20:04 2020/10/13      INFO Flooding packet Ethernet 20:00:00:00:00:01->30:00:00:00:00:01 ARP | Arp 20:00:00:00:00:01:192.168.100.2 30:00:00:00:00:01:192.168.100.3 to switch-eth2
21:20:04 2020/10/13      INFO Flooding packet Ethernet 20:00:00:00:00:01->30:00:00:00:00:01 ARP | Arp 20:00:00:00:00:01:192.168.100.2 30:00:00:00:00:01:192.168.100.3 to switch-eth0
21:20:05 2020/10/13      INFO In ljk-ThinkPad-13 received packet Ethernet 30:00:00:00:00:01->20:00:00:00:00:01 IP | IPv4 192.168.100.3->192.168.100.2 ICMP | ICMP EchoRequest 10275 1 (56 data bytes) on switch-eth2
21:20:05 2020/10/13      INFO
my_momory: [[EthAddr('10:00:00:00:00:01'), 'switch-eth0', 1], [EthAddr('30:00:00:00:00:01'), 'switch-eth2', 0]]
```

可以看到，由于 20...01 对应的 traffic_volume 为 0，是最小的，所以更新时它被删去了。

通过实验提供的 srpy 文件进行测试，并佐以自己的观察，使我们对程序正确性的把握增大了（但在极端情况下仍可能出现 bug!!）

本次实验报告结束，感谢阅读