南京大学计算机网络实验报告

任课教师:田臣

实验一 Switchyard&Mininet

计算机科学与技术系

181860055 **刘国涛**

邮箱: 181860055@smail.nju.edu.cn

2020年3月9日-2020年3月12日

实验目的

1. 配置并熟悉实验环境
2. 熟悉 `switchyard` 的框架
3. 学习使用 `wireshark` 抓包工具
4. 掌握在 `mininet` 下进行测试的方法

实验内容

Step 1 Modify the Mininet topology

Delete `server2` in the topology

`start_mininet.py` 文件对mininet的拓扑进行了初始化,要将server2删去,则在将该文件中删去与server2相关的初始化即可

首先在 `PySwitchTopo` 类中,对初始化方法 `__init__` 进行修改:

```python
class PySwitchTopo(Topo):
    def __init__(self, args):
        # Add default members to class.
        super(PySwitchTopo, self).__init__()

        nodeconfig = {'cpu': -1}
        self.addHost('server1', **nodeconfig)
        #self.addHost('server2', **nodeconfig)
        self.addHost('hub', **nodeconfig)
        self.addHost('client', **nodeconfig)
        nodes = ['server1', 'client']
        for node in nodes:
            # all links are 10Mb/s, 100 millisecond prop delay
            self.addLink(node, 'hub', bw=10, delay='100ms')
```

然后在设置地址的方法注释掉与server2有关的设置

```
def setup_addressing(net):
    reset_macs(net, 'server1', '10:00:00:00:00:{:02x}')
    #reset_macs(net, 'server2', '20:00:00:00:00:{:02x}')
    reset_macs(net, 'client', '30:00:00:00:00:{:02x}')
    reset_macs(net, 'hub', '40:00:00:00:00:{:02x}')
    set_ip(net, 'server1', 'hub', '192.168.100.1/24')
    #set_ip(net, 'server2', 'hub', '192.168.100.2/24')
    set_ip(net, 'client', 'hub', '192.168.100.3/24')
```

Step 2 Modify the logic of a device

要统计packet在hub中的进出情况,则修改 `myhub.py` ,在循环内增加两个计数器 `in_count` 和 `out_count` 进行统计,然后通过 `log_info` 进行输出

```
def main(net):
    ...
    in_count = 0
    out_count = 0
    while True:
        ...
        # count in
        in_count+=1
        ...
        if eth is None:
            log_info("Received a non-Ethernet packet?!")
            continue

        if eth.dst in mymacs:
            log_info ("Received a packet intended for me")

        else:
            # count out
            out_count+=1
            ...

        log_info ("on {} in:{} out:{}".format(timestamp,in_count,out_count))
    net.shutdown()
```

Step 3 Modify the test scenario of a device

在 `hubtest.py` 中构建一个从server1向client1发送的包

```python
reqpkt = mk_pkt("10:00:00:00:00:03", "30:00:00:00:00:02", "192.168.1.100","172.16.42.2")
s.expect(PacketInputEvent("eth0", reqpkt, display=Ethernet), "An Ethernet frame from 10:00:00:00:00:03 to 30:00:00:00:00:02 should arrive on eth0")
s.expect(PacketOutputEvent("eth1", reqpkt, "eth2", reqpkt, display=Ethernet), "Ethernet frame destined for 30:00:00:00:00:02 should be flooded out eth1 and eth2")

```

Step 4 Run your device in Mininet && Step 5 Capture using Wireshark

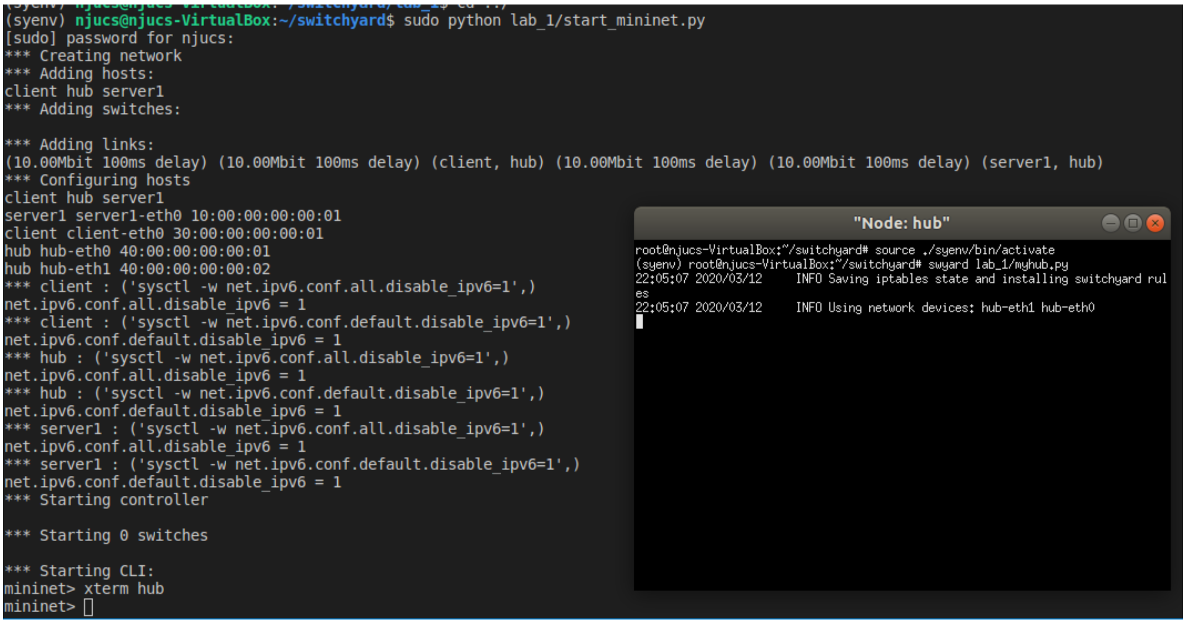
`terminal` 切换到switchyard目录下,键入

```
sudo python lab_1/start_mininet.py
```

运行脚本,启动mininet

然后在mininet中启动hub:

```
#在terminal中键入
xterm hub
#在xterm中键入
source ./syenv/bin/activate  #启动虚拟环境
swyard lab_1/myhub.py        #启动hub
```



```
(syenv) njucs@njucs-VirtualBox:~/switchyard$ sudo python lab_1/start_mininet.py
[sudo] password for njucs:
*** Creating network
*** Adding hosts:
client hub server1
*** Adding switches:

*** Adding links:
(10.00Mbit 100ms delay) (10.00Mbit 100ms delay) (client, hub) (10.00Mbit 100ms delay) (10.00Mbit 100ms delay) (server1, hub)
*** Configuring hosts
client hub server1
server1 server1-eth0 10:00:00:00:00:01
client client-eth0 30:00:00:00:00:01
hub hub-eth0 40:00:00:00:00:01
hub hub-eth1 40:00:00:00:00:02
*** client : ('sysctl -w net.ipv6.conf.all.disable_ipv6=1',)
net.ipv6.conf.all.disable_ipv6 = 1
*** client : ('sysctl -w net.ipv6.conf.default.disable_ipv6=1',)
net.ipv6.conf.default.disable_ipv6 = 1
*** hub : ('sysctl -w net.ipv6.conf.all.disable_ipv6=1',)
net.ipv6.conf.all.disable_ipv6 = 1
*** hub : ('sysctl -w net.ipv6.conf.default.disable_ipv6=1',)
net.ipv6.conf.default.disable_ipv6 = 1
*** server1 : ('sysctl -w net.ipv6.conf.all.disable_ipv6=1',)
net.ipv6.conf.all.disable_ipv6 = 1
*** server1 : ('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 hub
mininet> 
```

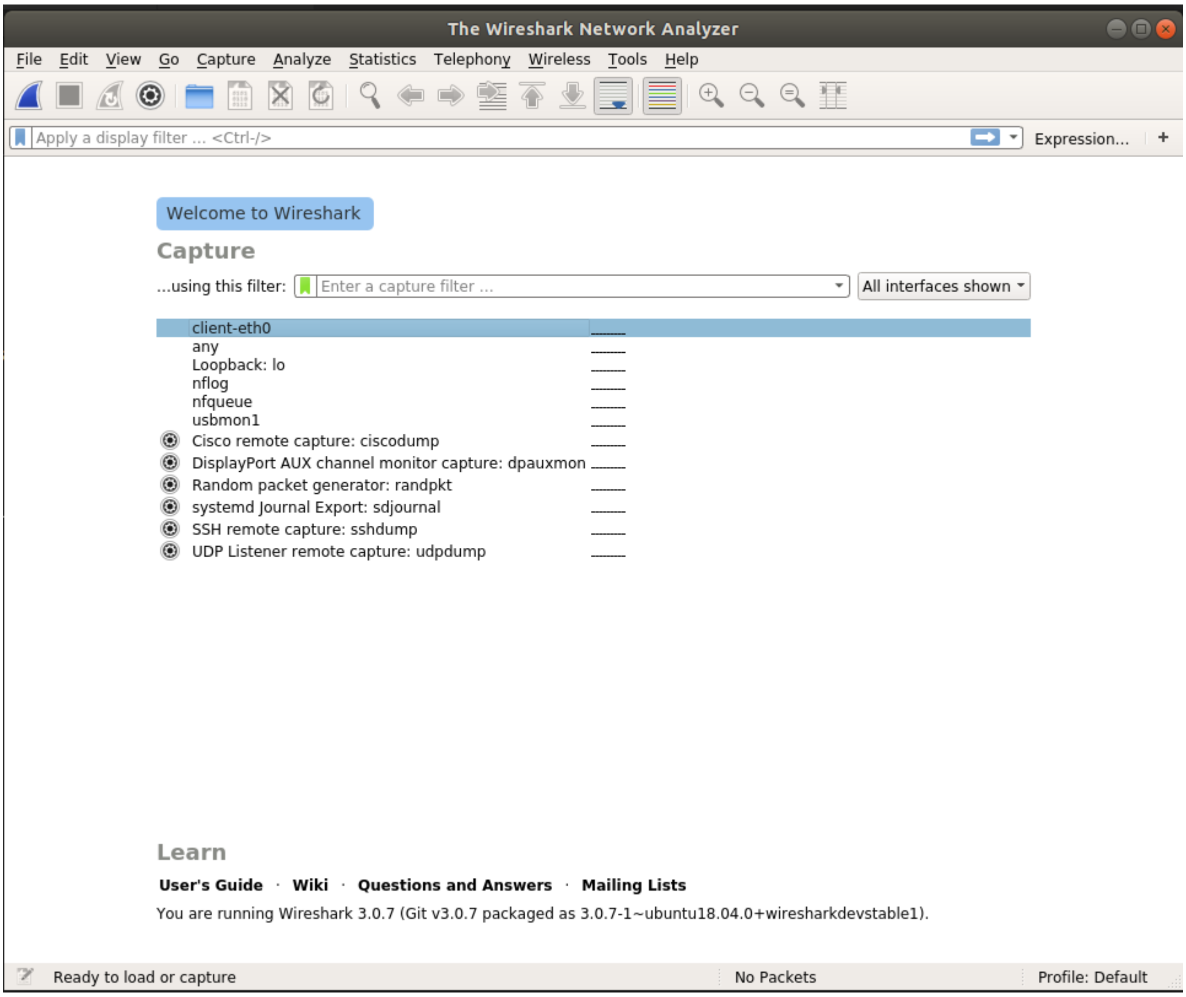
"Node: hub"

```
root@njucs-VirtualBox:~/switchyard# source ./syenv/bin/activate
(syenv) root@njucs-VirtualBox:~/switchyard# swyard lab_1/myhub.py
22:05:07 2020/03/12     INFO Saving iptables state and installing switchyard rul
es
22:05:07 2020/03/12     INFO Using network devices: hub-eth1 hub-eth0
```

然后返回terminal中开启wireshark准备抓包

```
client wireshark &
```

选择对应的网卡:



Capture

...using this filter: Enter a capture filter ... All interfaces shown

client-eth0
any
Loopback: lo
nflog
nfqueue
usbmon1
Cisco remote capture: ciscodump
DisplayPort AUX channel monitor capture: dpauxmon
Random packet generator: randpkt
systemd Journal Export: sdjournal
SSH remote capture: sshdump
UDP Listener remote capture: udpdump

Learn

User's Guide · Wiki · Questions and Answers · Mailing Lists

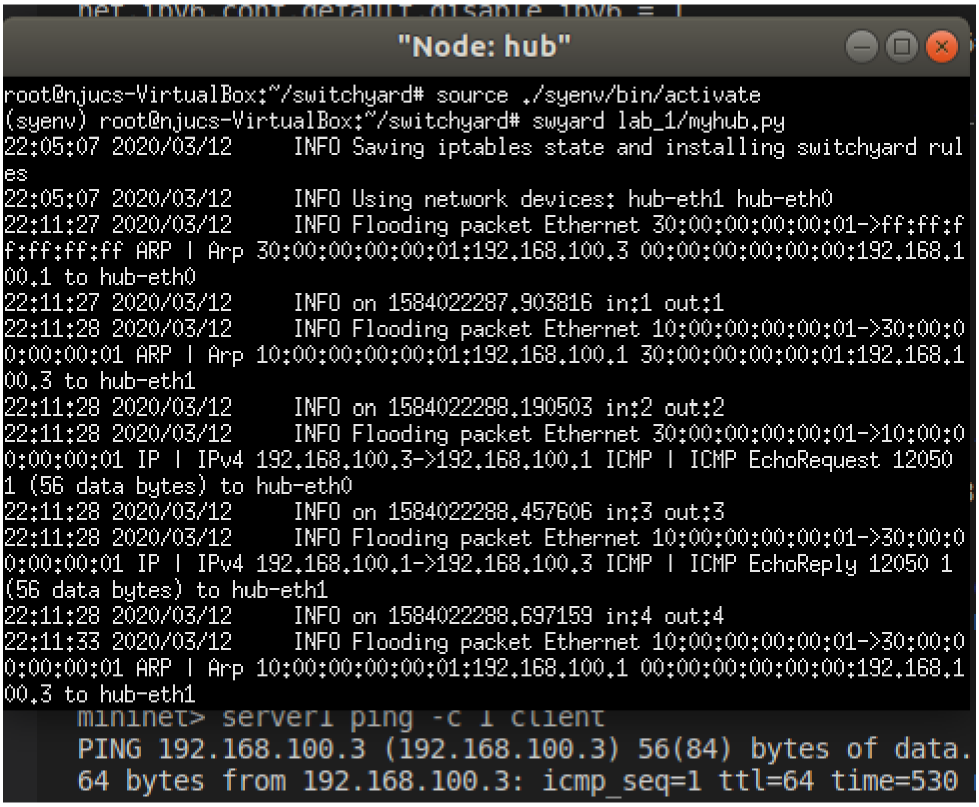
You are running Wireshark 3.0.7 (Git v3.0.7 packaged as 3.0.7-1~ubuntu18.04.0+wiresharkdevstable1).

Ready to load or capture | No Packets | Profile: Default

然后开始构造一些流量:

```
client ping -c 1 server1 # client  -> server1
server1 ping -c 1 client # server1 -> client
```

查看hub的log:

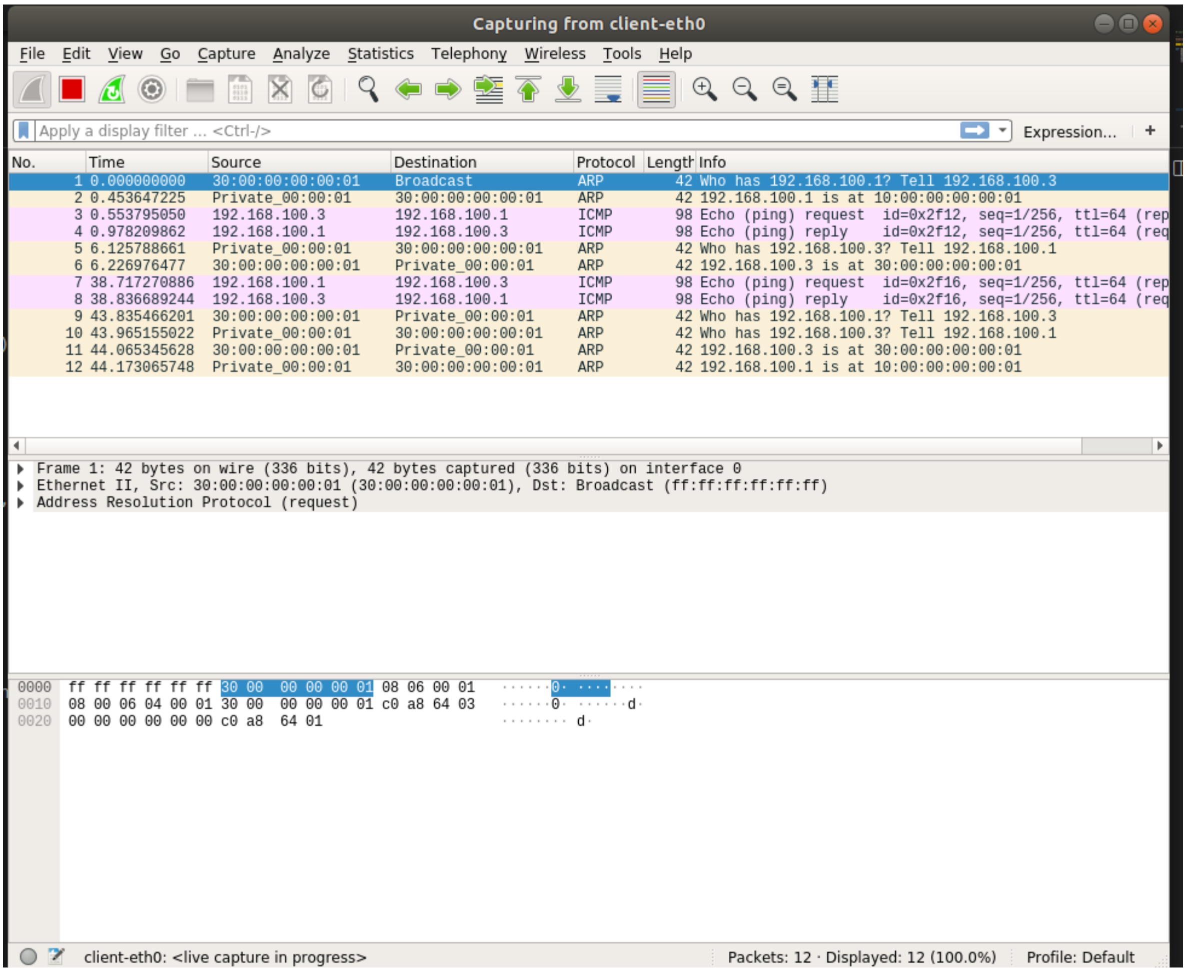


```
net.ipv6.conf.default.disable_ipv6 = 1
```

```
root@njucs-VirtualBox:~/switchyard# source ./syenv/bin/activate
(syenv) root@njucs-VirtualBox:~/switchyard# swyard lab_1/myhub.py
22:05:07 2020/03/12     INFO Saving iptables state and installing switchyard rul
es
22:05:07 2020/03/12     INFO Using network devices: hub-eth1 hub-eth0
22:11:27 2020/03/12     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 hub-eth0
22:11:27 2020/03/12     INFO on 1584022287.903816 in:1 out:1
22:11:28 2020/03/12     INFO Flooding packet Ethernet 10:00:00:00:00:01->30:00:0
0:00:00:01 ARP | Arp 10:00:00:00:00:01:192.168.100.1 30:00:00:00:00:01:192.168.1
00.3 to hub-eth1
22:11:28 2020/03/12     INFO on 1584022288.190503 in:2 out:2
22:11:28 2020/03/12     INFO Flooding packet Ethernet 30:00:00:00:00:01->10:00:0
0:00:00:01 IP | IPv4 192.168.100.3->192.168.100.1 ICMP | ICMP EchoRequest 12050
1 (56 data bytes) to hub-eth0
22:11:28 2020/03/12     INFO on 1584022288.457606 in:3 out:3
22:11:28 2020/03/12     INFO Flooding packet Ethernet 10:00:00:00:00:01->30:00:0
0:00:00:01 IP | IPv4 192.168.100.1->192.168.100.3 ICMP | ICMP EchoReply 12050 1
(56 data bytes) to hub-eth1
22:11:28 2020/03/12     INFO on 1584022288.697159 in:4 out:4
22:11:33 2020/03/12     INFO Flooding packet Ethernet 10:00:00:00:00:01->30:00:0
0:00:00:01 ARP | Arp 10:00:00:00:00:01:192.168.100.1 00:00:00:00:00:00:192.168.1
00.3 to hub-eth1
```

```
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=530
```

查看wireshark的抓包情况:

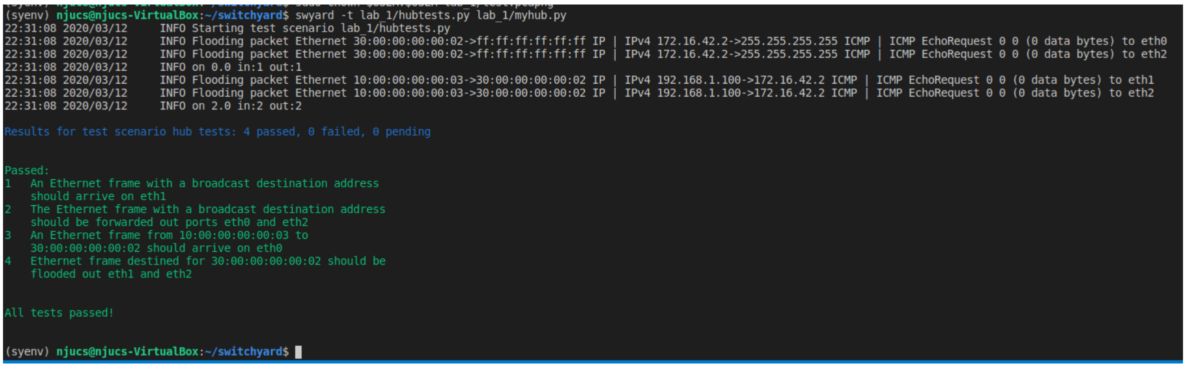

Capturing from client-eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
Expression…
+

```
No.  Time          Source             Destination        Protocol Length Info
   1 0.000000000   30:00:00:00:00:01  Broadcast          ARP       42 Who has 192.168.100.1? Tell 192.168.100.3
   2 0.453647225   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 0.553795050   192.168.100.3      192.168.100.1      ICMP      98 Echo (ping) request  id=0x2f12, seq=1/256, ttl=64 (rep
   4 0.978209862   192.168.100.1      192.168.100.3      ICMP      98 Echo (ping) reply    id=0x2f12, seq=1/256, ttl=64 (req
   5 6.125788661   Private_00:00:01   30:00:00:00:00:01  ARP       42 Who has 192.168.100.3? Tell 192.168.100.1
   6 6.226976477   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 38.717270886  192.168.100.1      192.168.100.3      ICMP      98 Echo (ping) request  id=0x2f16, seq=1/256, ttl=64 (rep
   8 38.836689244  192.168.100.3      192.168.100.1      ICMP      98 Echo (ping) reply    id=0x2f16, seq=1/256, ttl=64 (req
   9 43.835466201  30:00:00:00:00:01  Private_00:00:01   ARP       42 Who has 192.168.100.1? Tell 192.168.100.3
  10 43.965155022  Private_00:00:01   30:00:00:00:00:01  ARP       42 Who has 192.168.100.3? Tell 192.168.100.1
  11 44.065345628  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 44.173065748  Private_00:00:01   30:00:00:00:00:01  ARP       42 192.168.100.1 is at 10:00:00:00:00:01

▸ 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·

client-eth0: <live capture in progress>        Packets: 12 · Displayed: 12 (100.0%)      Profile: Default
```

（解释详见**实验结果**）

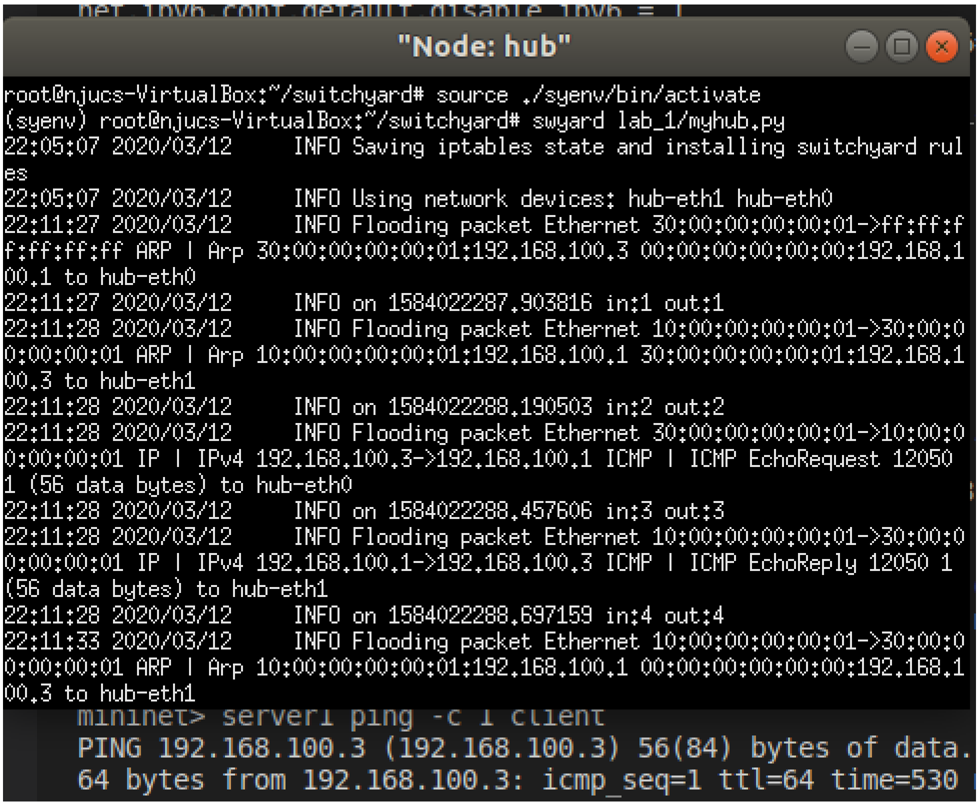
实验结果

运行 `hubtest.py` 结果:



```
(syenv) njucs@njucs-VirtualBox:~/switchyard$ swyard -t lab_1/hubtests.py lab_1/myhub.py
22:31:08 2020/03/12     INFO Starting test scenario lab_1/hubtests.py
22:31:08 2020/03/12     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
22:31:08 2020/03/12     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
22:31:08 2020/03/12     INFO on 0.0 in:1 out:1
22:31:08 2020/03/12     INFO Flooding packet Ethernet 10:00:00:00:00:03->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
22:31:08 2020/03/12     INFO Flooding packet Ethernet 10:00:00:00:00:03->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
22:31:08 2020/03/12     INFO on 2.0 in:2 out:2

Results for test scenario hub tests: 4 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 10:00:00:00:00:03 to
    30:00:00:00:00:02 should arrive on eth0
4   Ethernet frame destined for 30:00:00:00:00:02 should be
    flooded out eth1 and eth2


All tests passed!


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

在mininet中运行结果:

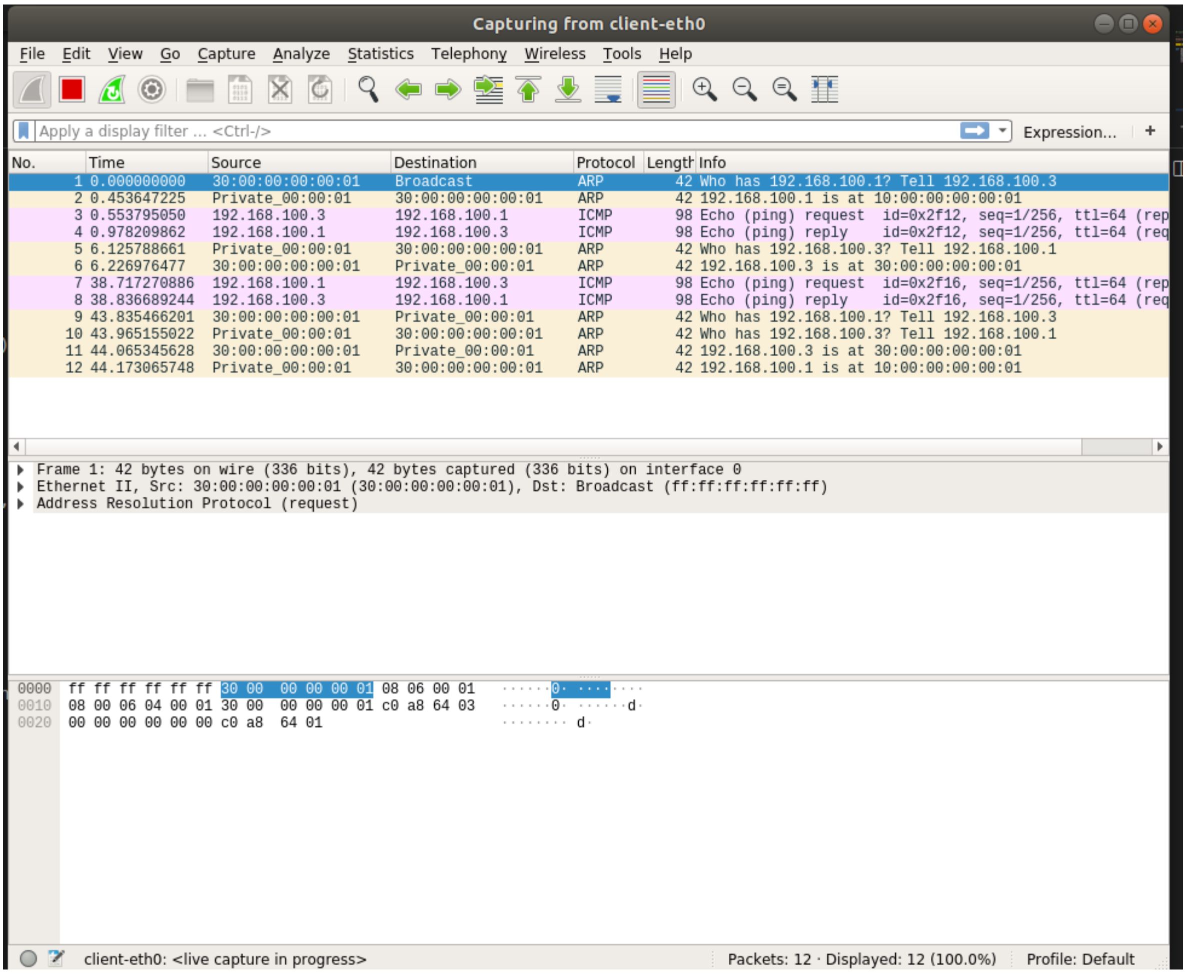


```
net.ipv6.conf.default.disable_ipv6 = 1
```

"Node: hub"

```
root@njucs-VirtualBox:~/switchyard# source ./syenv/bin/activate
(syenv) root@njucs-VirtualBox:~/switchyard# swyard lab_1/myhub.py
22:05:07 2020/03/12     INFO Saving iptables state and installing switchyard rul
es
22:05:07 2020/03/12     INFO Using network devices: hub-eth1 hub-eth0
22:11:27 2020/03/12     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 hub-eth0
22:11:27 2020/03/12     INFO on 1584022287.903816 in:1 out:1
22:11:28 2020/03/12     INFO Flooding packet Ethernet 10:00:00:00:00:01->30:00:0
0:00:00:01 ARP | Arp 10:00:00:00:00:01:192.168.100.1 30:00:00:00:00:01:192.168.1
00.3 to hub-eth1
22:11:28 2020/03/12     INFO on 1584022288.190503 in:2 out:2
22:11:28 2020/03/12     INFO Flooding packet Ethernet 30:00:00:00:00:01->10:00:0
0:00:00:01 IP | IPv4 192.168.100.3->192.168.100.1 ICMP | ICMP EchoRequest 12050
1 (56 data bytes) to hub-eth0
22:11:28 2020/03/12     INFO on 1584022288.457606 in:3 out:3
22:11:28 2020/03/12     INFO Flooding packet Ethernet 10:00:00:00:00:01->30:00:0
0:00:00:01 IP | IPv4 192.168.100.1->192.168.100.3 ICMP | ICMP EchoReply 12050 1
(56 data bytes) to hub-eth1
22:11:28 2020/03/12     INFO on 1584022288.697159 in:4 out:4
22:11:33 2020/03/12     INFO Flooding packet Ethernet 10:00:00:00:00:01->30:00:0
0:00:00:01 ARP | Arp 10:00:00:00:00:01:192.168.100.1 00:00:00:00:00:00:192.168.1
00.3 to hub-eth1
```

```
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=530
```



Capturing from client-eth0

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    30:00:00:00:00:01   Broadcast           ARP      42 Who has 192.168.100.1? Tell 192.168.100.3
   2 0.453647225    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 0.553795050    192.168.100.3       192.168.100.1       ICMP     98 Echo (ping) request  id=0x2f12, seq=1/256, ttl=64 (rep
   4 0.978209862    192.168.100.1       192.168.100.3       ICMP     98 Echo (ping) reply    id=0x2f12, seq=1/256, ttl=64 (req
   5 6.125788661    Private_00:00:01    30:00:00:00:00:01   ARP      42 Who has 192.168.100.3? Tell 192.168.100.1
   6 6.226976477    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 38.717270886   192.168.100.1       192.168.100.3       ICMP     98 Echo (ping) request  id=0x2f16, seq=1/256, ttl=64 (rep
   8 38.836689244   192.168.100.3       192.168.100.1       ICMP     98 Echo (ping) reply    id=0x2f16, seq=1/256, ttl=64 (req
   9 43.835466201   30:00:00:00:00:01   Private_00:00:01    ARP      42 Who has 192.168.100.1? Tell 192.168.100.3
  10 43.965155022   Private_00:00:01    30:00:00:00:00:01   ARP      42 Who has 192.168.100.3? Tell 192.168.100.1
  11 44.065345628   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 44.173065748   Private_00:00:01    30:00:00:00:00:01   ARP      42 192.168.100.1 is at 10:00:00:00:00:01
```

```
▸ 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·
```

client-eth0: <live capture in progress>   Packets: 12 · Displayed: 12 (100.0%)   Profile: Default

分析:

在mininet中构造了两种流量:

```
client ping -c 1 server1 # client  -> server1
server1 ping -c 1 client # server1 -> client
```

从wireshark中可以看到,在client向server1发包时,有以下流程:

1. 广播询问 `192.168.100.1` (server1)的MAC地址
2. 接收到server1的MAC地址 `10:00:00:00:00:00:01`
3. client向server1发送一个request
4. server1接收后向clinet1发回一个包

在server1向client发包时也是同样的流程

总结与感想

本次实验了解了mininet和switchyard的使用方法