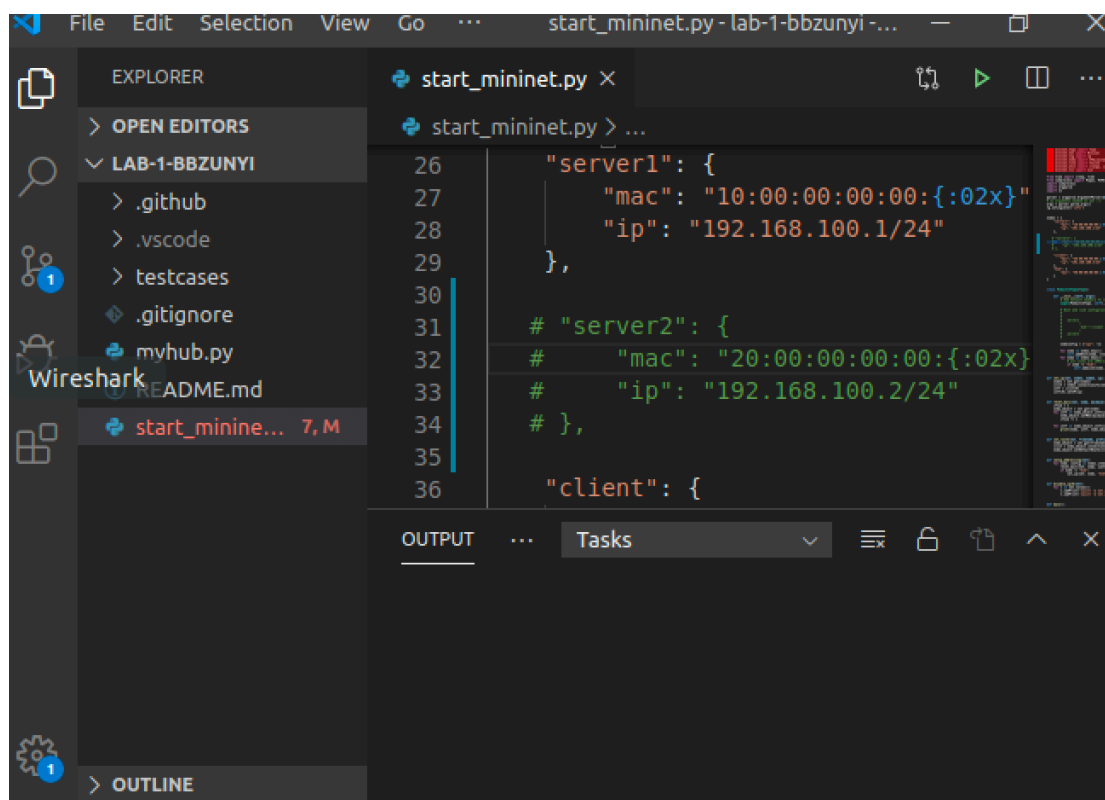


计网 lab1 Switchyard&Mininet 实验报告

张洋彬 191220169

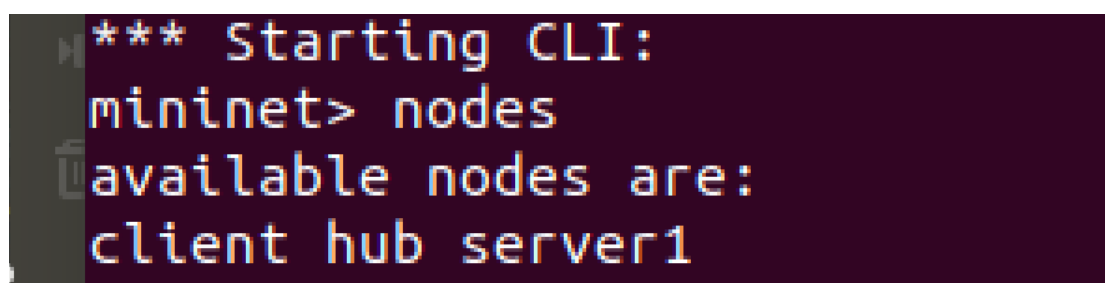
Step1：修改 mininet 的拓扑结构

在这个环节，我选择了在拓扑图中删除 server2，在 start_mininet.py 中，将本段代码注释掉，即可完成删除 server2 的操作



```
26     "server1": {
27         "mac": "10:00:00:00:00:{:02x}"
28         "ip": "192.168.100.1/24"
29     },
30
31     # "server2": {
32     #     "mac": "20:00:00:00:00:{:02x}"
33     #     "ip": "192.168.100.2/24"
34     # },
35
36     "client": {
```

删除结果如下所示：



```
*** Starting CLI:
mininet> nodes
available nodes are:
client hub server1
```

Step2：修改设备的逻辑

```
def main(net: switchyard.llnetbase.LLNetBase):
    my_interfaces = net.interfaces()
    mymacs = [intf.ethaddr for intf in my_interfaces]
    in_count=0
    out_count=0
    while True:
        try:
            _, fromIface, packet = net.recv_packet()
            in_count+=1
        except KeyboardInterrupt:
            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 eth.dst in mymacs:
            log_info("Received a packet intended for me")
        else:
            for intf in my_interfaces:
                if fromIface!= intf.name:
                    log_info (f"Flooding packet {packet} to {intf.name}")
                    out_count+=1
                    net.send_packet(intf, packet)
            log_info(f"in:{in_count} out:{out_count}")
    net.shutdown()
```

- 1、定义 in_count&out_count 并赋予初值;
- 2、当有包传输的时候, in_count+1;
- 3、当 packet 被发送走的时候, out_count+1;
- 4、每次循环都打印出 in : in_count out:out_count(结果如下图所示)

```
"Node: hub"
0:00:00:01 IP | IPv4 192.168.100.3->192.168.100.1 ICMP | ICMP EchoRequest 2601 1
(56 data bytes) to hub-eth1
18:05:59 2021/03/21      INFO in:3 out:3
18:05:59 2021/03/21      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 2601 1 (
56 data bytes) to hub-eth0
18:05:59 2021/03/21      INFO in:4 out:4
18:05:59 2021/03/21      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 EchoRequest 2604 1
(56 data bytes) to hub-eth0
18:05:59 2021/03/21      INFO in:5 out:5
18:06:00 2021/03/21      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 EchoReply 2604 1 (
56 data bytes) to hub-eth1
18:06:00 2021/03/21      INFO in:6 out:6
18:06:04 2021/03/21      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-eth0
18:06:04 2021/03/21      INFO in:7 out:7
18:06:04 2021/03/21      INFO Flooding packet Ethernet 30:00:00:00:00:01->10:00:0
0:00:00:01 ARP | Arp 30:00:00:00:00:01:192.168.100.3 10:00:00:00:00:01:192.168.1
00.1 to hub-eth1
18:06:04 2021/03/21      INFO in:8 out:8
```

Step3 修改测试文件

- 1、选择第一个任务：利用 `new_packet` 创建一个新的测试

```
reqpkt = new_packet([
    "30:00:00:00:00:02",
    "10:00:00:00:00:02",
    '172.16.42.2',
    '192.168.1.100'
])
s.expect(
    PacketInputEvent("eth1", reqpkt, display=Ethernet)
    ("An Ethernet frame should arrive on eth1 with des
    "the same as eth1's MAC address")
)

s.expect(
    PacketInputTimeoutEvent(1.0),
    ("The hub should not do anything in response to a
    " a destination address referring to the hub itse
    )
)
```

这里所做的操作是创建一个 eth1 的包，目的地址是 eth1 的 Mac 地址，所以并不会经过 hub 结点，所以说 hub 不会做任何事。

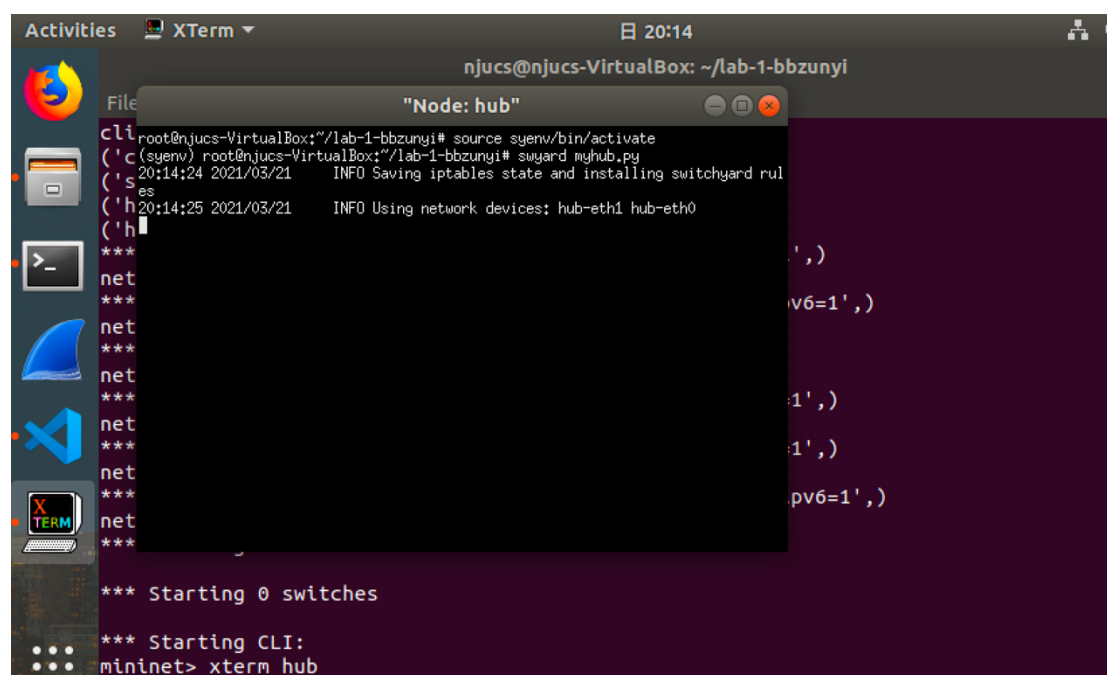
通过测试后，输出第九条和第十条信息。

```
3 An Ethernet frame from 20:00:00:00:00:01 to
4 30:00:00:00:00:02 should arrive on eth0
5 Ethernet frame destined for 30:00:00:00:00:02 should be
6 flooded out eth1 and eth2
7 An Ethernet frame from 30:00:00:00:00:02 to
8 20:00:00:00:00:01 should arrive on eth1
9 Ethernet frame destined to 20:00:00:00:00:01 should be
10 flooded out eth0 and eth2
11 An Ethernet frame should arrive on eth2 with destination
12 address the same as eth2's MAC address
13 The hub should not do anything in response to a frame
14 arriving with a destination address referring to the hub
15 itself.
16 An Ethernet frame should arrive on eth1 with destination
17 address the same as eth1's MAC address
18 The hub should not do anything in response to a frame
19 arriving with a destination address referring to the hub
20 itself.

All tests passed!
```

Step4：在 mininet 中运行设备

- 1、输入 `sudo python start_mininet.py`，打开 mininet，创建拓扑结构
- 2、输入 xterm 打开 hub 节点，并在 xterm 中激活虚拟环境，在用 switchyard 运行 myhub.py



```
Activities XTerm 20:14
njucs@njucs-VirtualBox: ~/lab-1-bbzunyi

"Node: hub"
cli root@njucs-VirtualBox:~/lab-1-bbzunyi# source ~/.env/bin/activate
('c(syenv) root@njucs-VirtualBox:~/lab-1-bbzunyi# switchyard myhub.py
('s 20:14:24 2021/03/21 INFO Saving iptables state and installing switchyard rul
('h 20:14:25 2021/03/21 INFO Using network devices: hub-eth1 hub-eth0
('h
***
net
***
net
***
net
***
net
***
net
***
net
***
net
***
net
***

*** Starting 0 switches
*** Starting CLI:
mininet> xterm hub
```

- 3、在 mininet 中输入 `pingall` 指令，检查各个节点的连通性，hub 工作在网络层之下，没有 ip 地址，所以无法与其他节点连接。

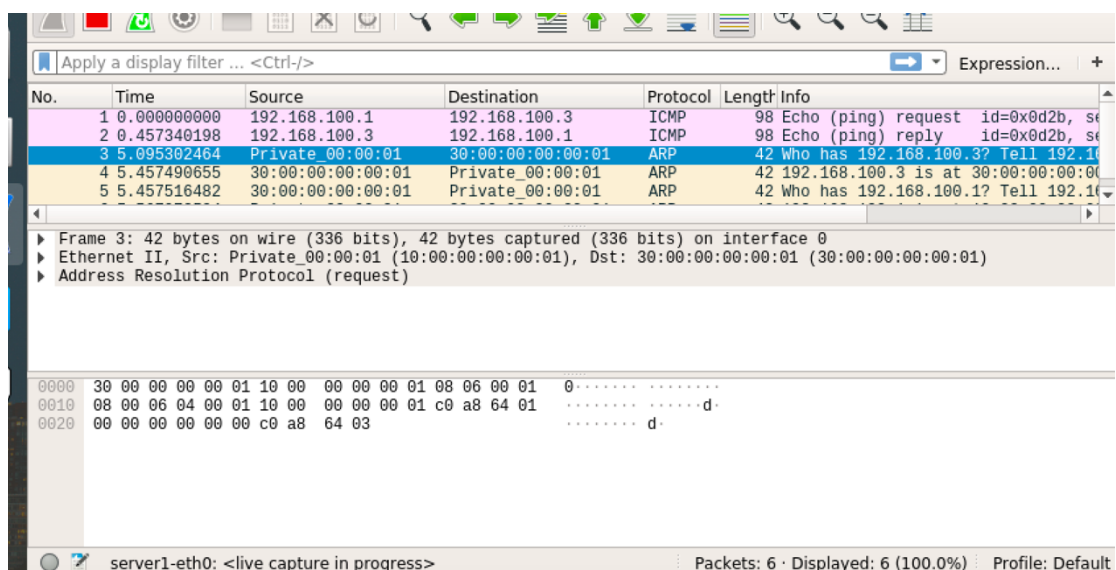
```
mininet> pingall
*** Ping: testing ping reachability
client -> X server1
hub -> X X
server1 -> client X
*** Results: 66% dropped (2/6 received)
```

Step5 用 wireshark 抓包

- 1、同 step4, 先用 myhub.py 进行 hub 实例化。

[illegible]

- 2、在 mininet 中输入 `server1 wireshark &` 指令，对 server 抓包
- 3、输入 `server1 ping -c 1 client` 让 server1 和 client 进行通信



如这两张图所示, server1 和 client 之间先通过 ICMP 协议通信一次, server1 先向 client 发送一次请求, 然后 client 收到后给予回复 server1。下图中的四个包, 他们之间以问答的形式, 询问 IP 地址对应的 MAC 地址并得到了地址。

5302464	Private_00:00:01	30:00:00:00:00:01	ARP	42 Who has 192.168.100.3? Tell 192.168.100.1
7490655	30:00:00:00:00:01	Private_00:00:01	ARP	42 192.168.100.3 is at 30:00:00:00:00:01
7516482	30:00:00:00:00:01	Private_00:00:01	ARP	42 Who has 192.168.100.1? Tell 192.168.100.3
7973504	Private_00:00:01	30:00:00:00:00:01	ARP	42 192.168.100.1 is at 10:00:00:00:00:01

感想与总结

本次实验内容比较简单, 主要是基础知识的学习, 在做实验的过程中学习到了很多知识, 比如之前了解比较少的 git 和 python, 现在基本了解了 git 的全部内容, 比如删除分支、合并分支之类的操作, 以及多人一起在 github 上工作的各种技巧, python 也学习了一些基础的语言操作。

感觉助教在问题网站上给予的回答有很大的参考性, 基本了解了本次实验中之前会迷惑的内容: 比如 hub 为什么与其他几个点不能传包, mininet 和 switchyard 的概念等等。