南京大学本科生实验报告

课程名称：**计算机网络**　　任课教师：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	**211220049**	姓名	**石璐**
Email	**211220049@smail.nju.edu.cn**	开始/完成日期	**2023 年 3 月 20 日/2023 年 3 月 20 日**

1. 实验名称：Switchyard & Mininet

2. 实验目的

完成网络实验的主要准备工作，熟悉实验环境，了解基本的工作流程，熟悉各种软件平台的基本操作。

3. 实验内容

Step 1: Modify the Mininet topology

✧ 从 Topology 中删除 Server2：

```
nodes = {
    "server1": {
        "mac": "10:00:00:00:00:{:02x}",
        "ip": "192.168.100.1/24"
    },
    # "server2": {
    #     "mac": "20:00:00:00:00:{:02x}",
    #     "ip": "192.168.100.2/24"
    # },
    "client": {
        "mac": "30:00:00:00:00:{:02x}",
        "ip": "192.168.100.3/24"
    },
    "hub": {
        "mac": "40:00:00:00:00:{:02x}",
    }
}
```

✧ 建立 Topology：加入各个 host，将所有非 hub 的 host 与 hub 建立连接

```
class PySwitchTopo(Topo):

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

        # Host and link configuration
        #
        #
        #   server1
        #          \
        #           hub----client =====> server1----hub----client
        #          /
        #   server2
        #

        nodeconfig = {"cpu": -1}

        for node in nodes.keys():
            self.addHost(node, **nodeconfig)
        for node in nodes.keys():
            # all links are 10Mb/s, 100 millisecond prop delay
            if node != "hub":
                self.addLink(node, "hub", bw=10, delay="100ms")
```

Step 2: Modify the logic of a device

✧ hub 功能：

从某个端口接收 packet，如果 packet 的目标 host 不是 hub 自身，则将该 packet 从各个端口（接收端口除外）发送出去。同时打印日志，记录收发 packet 的数量。

✧ 具体实现：

```
10  def main(net: switchyard.llnetbase.LLNetBase):
11      my_interfaces = net.interfaces()
12      mymacs = [intf.ethaddr for intf in my_interfaces]
13
14      packet_in,packet_out = 0,0
15      while True:
16          try:
17              _, fromIface, packet = net.recv_packet()
18          except NoPackets:
19              continue
20          except Shutdown:
21              break
```

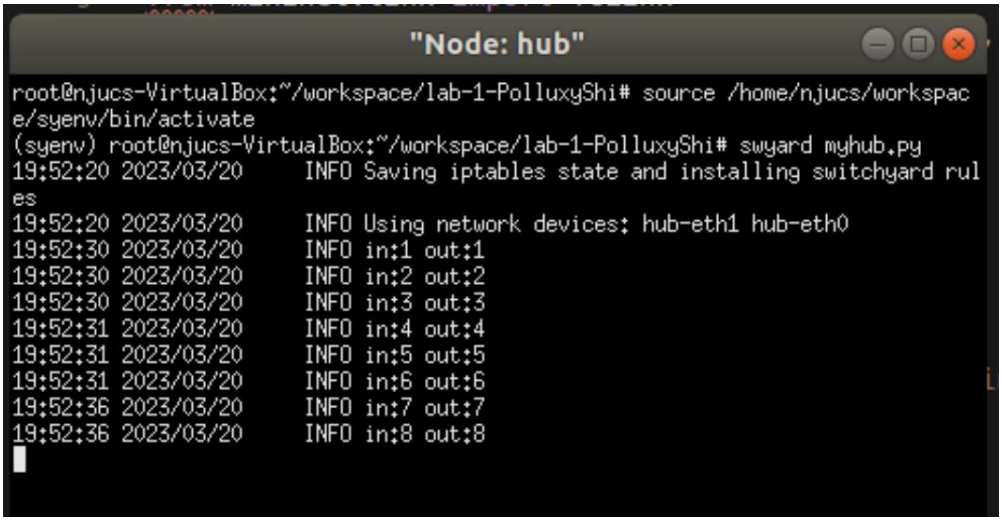
```
23          log_debug (f"In {net.name} received packet {packet} on {fromIface}")
24          eth = packet.get_header(Ethernet)
25          if eth is None:
26              log_info("Received a non-Ethernet packet?!")
27              return
28          packet_in += 1
29          if eth.dst in mymacs:
30              log_info("Received a packet intended for me")
31          else:
32              for intf in my_interfaces:
33                  if fromIface!= intf.name:
34                      # log_info (f"Flooding packet {packet} to {intf.name}")
35                      net.send_packet(intf, packet)
36                      packet_out += 1
37
38          log_info (f"in:{packet_in} out:{packet_out}")
39
40      net.shutdown()
```

调用 recv_packet()接收 packet，27 行判断没有收到以太网 packet 则直接返回，若执行到 28 行，说明收到了一个以太网 packet，则记录收到 packet 数量的变量 packet_in += 1；

接下来判断 packet 的目标是否为 hub 自身，如果不是则需将该 packet 从其他端口再发送出去，并将累计的发送 packet 数量记录在变量 packet_out 中。由于 hub 只连接 Server1 和 client，因此每收到一个非发送给 hub 的 packet，就会向另一个端口发送该 packet。

✧ 日志：执行 pingall

```
OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS 7

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



Step 3: Modify the test scenario of a device

测试样例：hub 从 eth0 端口收到来自 30:00:00:00:00:02 的 packet，目标为 hub 自身，因此不需要向外再发送该 packet。

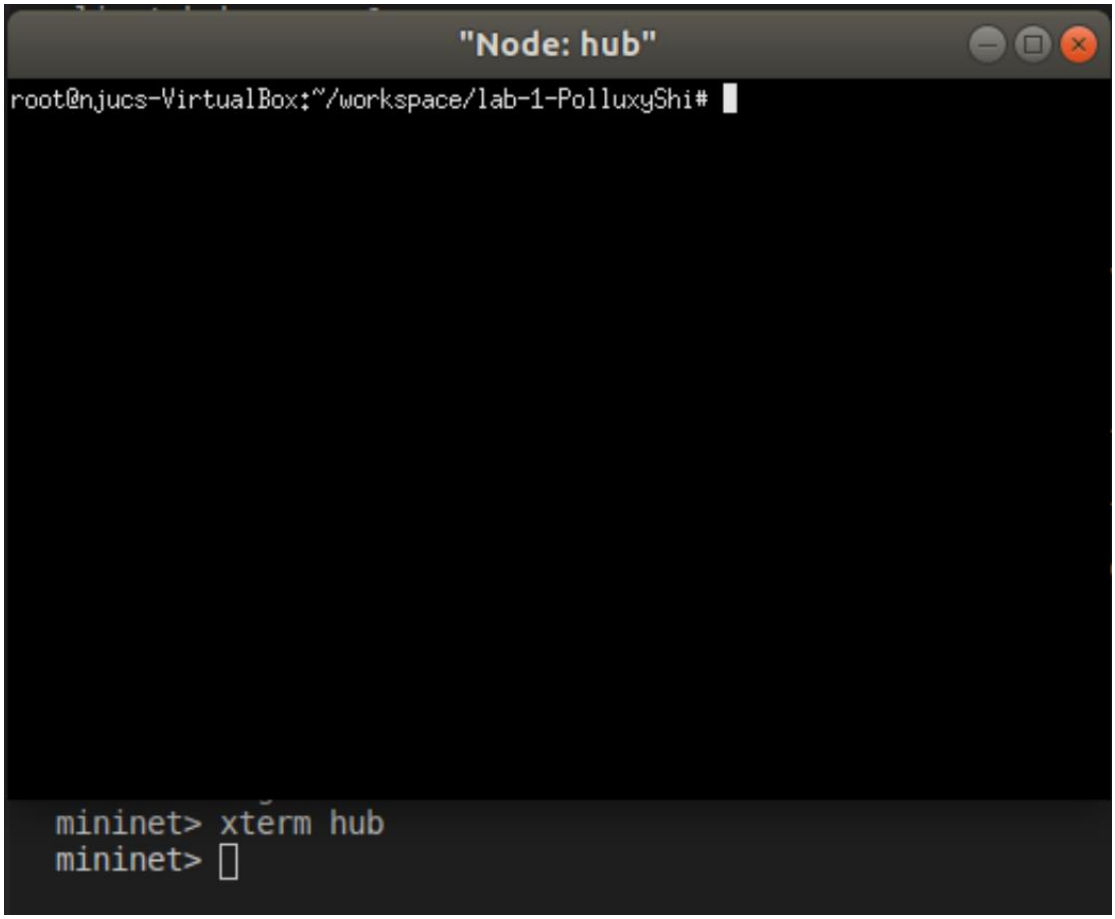
```
# my test case:
mypkt = new_packet(
    "30:00:00:00:00:02",
    "10:00:00:00:00:01",
    "172.16.42.2",
    '192.168.1.100'
)
s.expect(
    PacketInputEvent("eth0", mypkt, display=Ethernet),
    ("An Ethernet frame should arrive on eth0 with destination address "
     "the same as eth0's MAC address")
)
s.expect(
    PacketInputTimeoutEvent(1.0),
    ("The hub should not do anything in response to a frame arriving with"
     " a destination address referring to the hub itself.")
)
```

Step 4: Run your device in Mininet

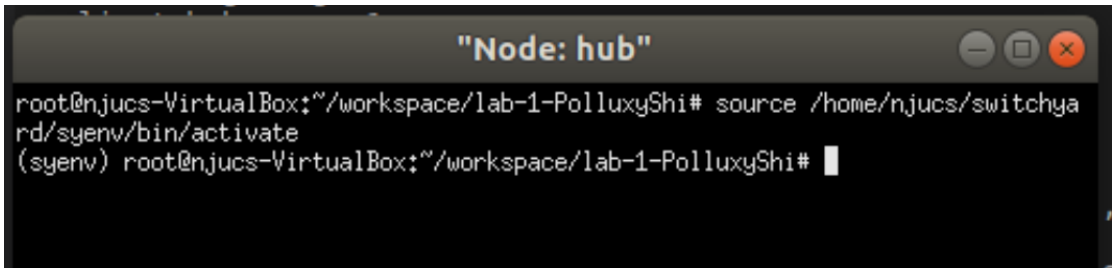
✧ Start the topology

```
OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS                                  1: sudo
(syenv) njucs@njucs-VirtualBox:~/workspace/lab-1-PolluxyShi$ sudo python 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
('client', <TCIntf client-eth0>, '30:00:00:00:00:01')
('server1', <TCIntf server1-eth0>, '10:00:00:00:00:01')
('hub', <TCIntf hub-eth0>, '40:00:00:00:00:01')
('hub', <TCIntf 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>
```

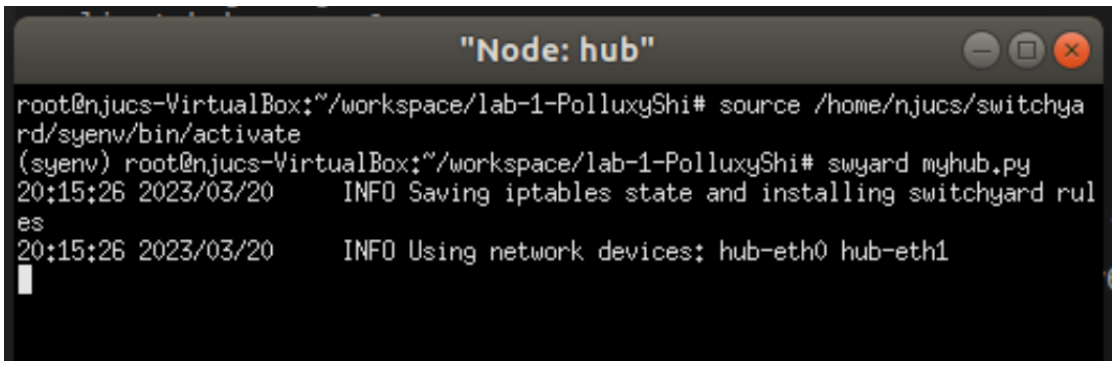
✧ Run the hub code on the device named “hub” through Xterm



✧ Activate the virtual environment



✧ Go to the switchyard



✧ Test connectivity between hosts

```
OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS 7

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

```
"Node: hub"

root@njucs-VirtualBox:~/workspace/lab-1-PolluxyShi# source /home/njucs/workspac
e/syenv/bin/activate
(syenv) root@njucs-VirtualBox:~/workspace/lab-1-PolluxyShi# swyard myhub.py
19:52:20 2023/03/20     INFO Saving iptables state and installing switchyard rul
es
19:52:20 2023/03/20     INFO Using network devices: hub-eth1 hub-eth0
19:52:30 2023/03/20     INFO in:1 out:1
19:52:30 2023/03/20     INFO in:2 out:2
19:52:30 2023/03/20     INFO in:3 out:3
19:52:31 2023/03/20     INFO in:4 out:4
19:52:31 2023/03/20     INFO in:5 out:5
19:52:31 2023/03/20     INFO in:6 out:6
19:52:36 2023/03/20     INFO in:7 out:7
19:52:36 2023/03/20     INFO in:8 out:8
```

Step 5: Capture using Wireshark

✧ 执行 $ client ping -c1 server1

```
mininet> client wireshark &
mininet> client ping -c1 server1
QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
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=1114 ms

--- 192.168.100.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1114.558/1114.558/1114.558/0.000 ms
mininet>
```

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

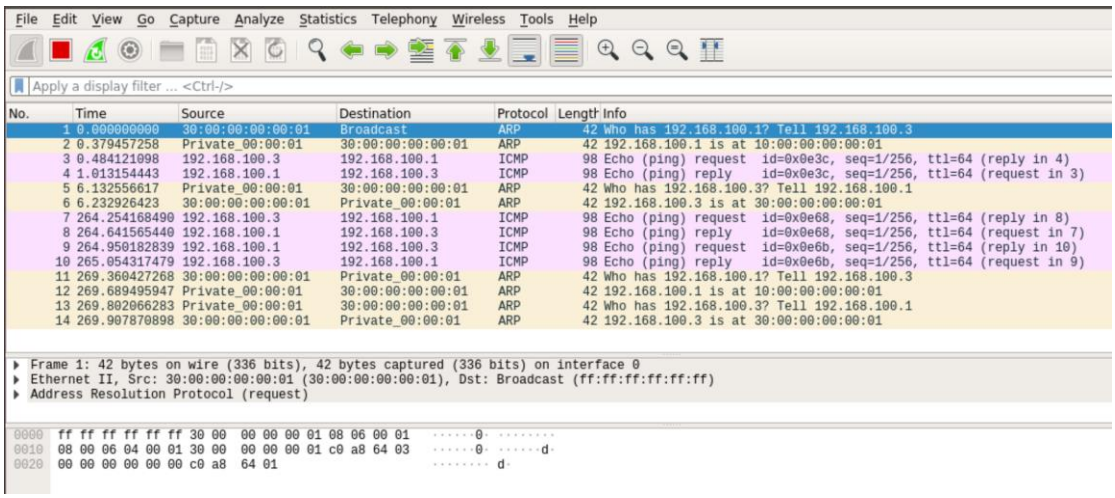
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	30:00:00:00:00:01	Broadcast	ARP	42	Who has 192.168.100.1? Tell 192.168.100.3
2	0.379457258	Private_00:00:01	30:00:00:00:00:01	ARP	42	192.168.100.1 is at 10:00:00:00:00:01
3	0.484121098	192.168.100.3	192.168.100.1	ICMP	98	Echo (ping) request id=0x0e3c, seq=1/256, ttl=64 (reply in 4)
4	1.013154443	192.168.100.1	192.168.100.3	ICMP	98	Echo (ping) reply id=0x0e3c, seq=1/256, ttl=64 (request in 3)
5	6.132556617	Private_00:00:01	30:00:00:00:00:01	ARP	42	Who has 192.168.100.3? Tell 192.168.100.1
6	6.232926423	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 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·
```

✧ 执行 $ pingall

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



1-6：

在执行“$ client ping -c1 server1”时捕获的报文，前两条分别为 client 通过广播询问 Server1 的 MAC 地址，以及 Server1 响应 client 查询的报文，均为 ARP 分组；中间两条分别为 client 向 Server1 发送的 ICMP 报文相应的响应报文；后两条则为 Server1 查询 client 的 MAC 地址的 ARP 报文即响应报文。

7-14：

同理，在 pingall 过程中分别有 client 向 Server1 发送报文，Server1 向 client 发送报文，其中的 ARP（地址解析协议）报文用于查询目标 MAC 地址，ICMP 报文则携带发送的信息。