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：洋溢

学院	**计算机**	专业（方向）	**计算机科学与技术**
学号	**221220142**	姓名	**欧阳瑞泽**
Email	**221220142@smail.nju.edu.cn**	开始/完成日期	**2024.3.10**

Lab 1: Switchyard & Mininet

1. 实验目的

配置实验环境，修改实验代码，完成包括修改 Mininet 拓扑结构、统计数据包、修改设备的测试场景、在 Mininet 中运行设备、使用 Wireshark 抓包的五个步骤。

2. 实验内容

Step 1: Modify the Mininet topology

Delete server2 in the topology：

通过阅读手册，得知 start_mininet.py 通过__init__(self, args)函数初始化网络的拓扑结构，而函数中通过循环遍历 nodes 建立拓扑结构，所以只需在遍历至 server2 时 continue，就可以在建图时将 server2 从网络中删除。后续 setup_addressing(net)函数中也需将 server2 去除。

Step 2: Modify the logic of a device：

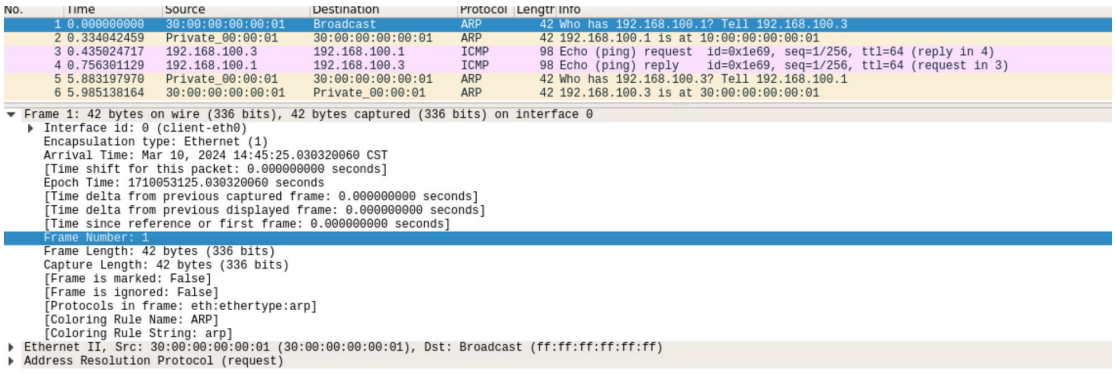
新增两个变量对 packet 进行计数，每当新 packet 进出 hub 时，更新变量，最后输出。值得注意的是，统计发出的 packet 时，发送给 hub 自己的不被统

计。

Step 3: Modify the test scenario of a device
Create one test case by using the given function new_packet with different arguments:
使用 new_packet 函数制作一个新包。仿照 case3，修改 mac 参数，设计一个发送至 hub 的包，由于 hub 不会转发，可知统计包数量的 in 增加，out 不变。

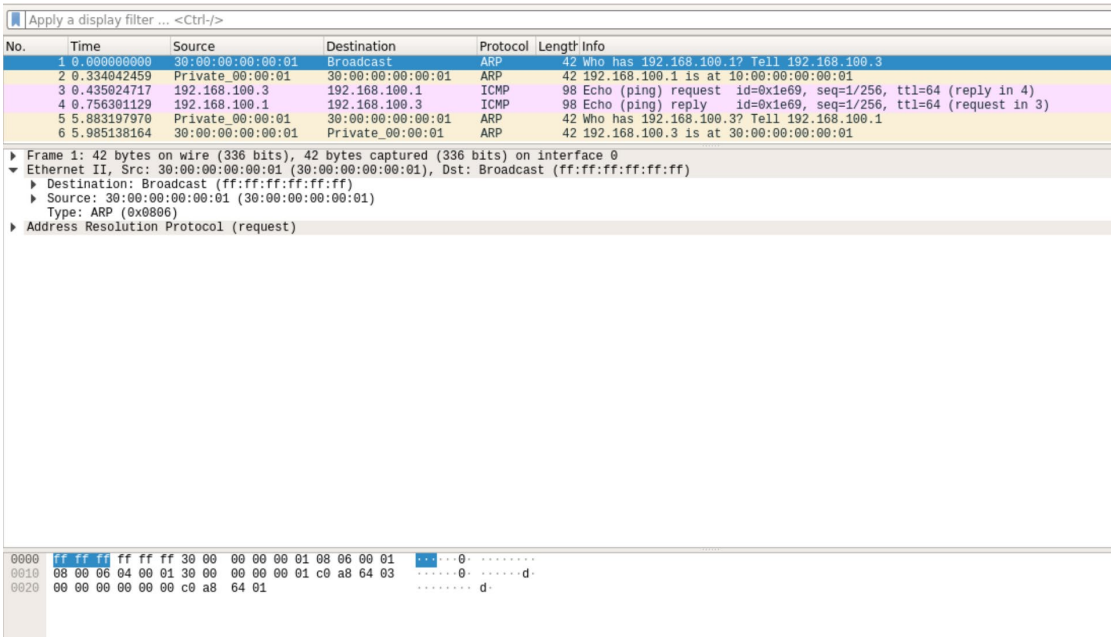
Step 4: Run your device in Mininet
在 mininet 中运行。开启 mininet，打开 xterm，分别激活环境，在 mininet 中测试 pingall，验证网络。

Step 5: Capture using Wireshark
在 mininet 中输入 server1 wireshark &，选择 client-eth0，再让 client ping server1。抓取 ARP 协议的包。最后导出 pcapng 文件。打开包的详细内容，可以得知以下信息：

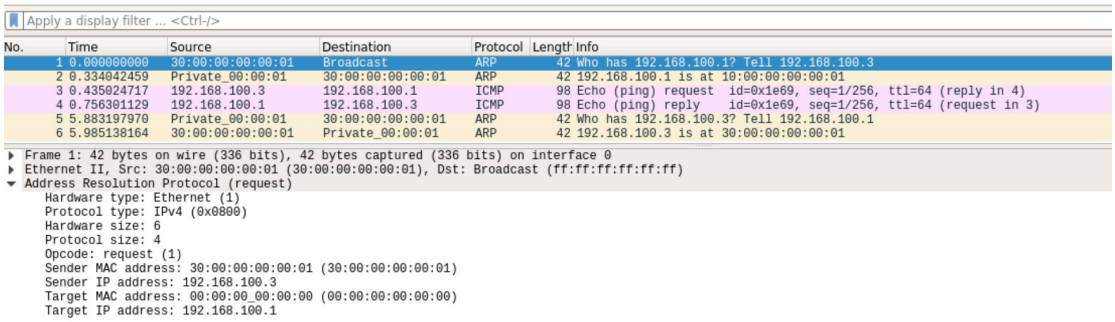


第一栏中
encapsulation type ethernet (1)：表示包的特

定链接层信息
Arrival time: 到达时间
Epoch time: 自1970年1月1日以来的秒数
以及一些时间信息，比如与上一个数据包的时间间隔。
Frame Length: 包的大小
还有包的信息比如协议、sharkwire 中表示的颜色等。
第二栏中



destination: 表示这是一个广播给所有设备的包,所以MAC地址是ff:ff:ff:ff:ff:ff
source: 表示包的源地址是30:00:00:00:00:01
Tyep:表示包的协议是ARP
第三栏中



Hardware Type：所使用的硬件类型
Protocol Type：上层协议的类型。
Hardware Address Length：指定了硬件地址的长度
Protocol Address Length：了网络层地址的长度
Opcode：ARP 请求的类型
Sender Hardware Address：发送 ARP 请求或应答的设备的硬件地址
Sender Protocol Address：发送 ARP 请求或应答的设备的协议地址
Target Hardware Address：ARP 应答中的目标设备的硬件地址。
Target Protocol Address：目标设备的协议地址

3. 实验结果

Step 1: Modify the Mininet topology
Delete server2 in the topology：
删除 server2 后通过 mininet 查看网络拓扑结构，发现并无 server2

```
mininet> net
client client-eth0:hub-eth1
hub hub-eth0:server1-eth0 hub-eth1:client-eth0
server1 server1-eth0:hub-eth0
mininet> nodes
available nodes are:
client hub server1
```

Step 2: Modify the logic of a device：

在控制台输出的 log 如下，由于 Step3 中设计了发送至 hub 的包，所以 in 和 out 的数量可能不同。

```
(syenv) njucs@njucs-VirtualBox:~/CN/Lab1$ swyard -t ./testcases/myhub_testscenario.py myhub.py
14:10:42 2024/03/10     INFO Starting test scenario ./testcases/myhub_testscenario.py
14:10:42 2024/03/10     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
14:10:42 2024/03/10     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
14:10:42 2024/03/10     INFO in:1 out:2
14:10:42 2024/03/10     INFO Flooding packet Ethernet 20:00:00:00:00:01->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
14:10:42 2024/03/10     INFO Flooding packet Ethernet 20:00:00:00:00:01->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
14:10:42 2024/03/10     INFO in:2 out:4
14:10:42 2024/03/10     INFO Flooding packet Ethernet 30:00:00:00:00:02->20:00:00:00:00:01 IP | IPv4 172.16.42.2->192.168.1.100 ICMP | ICMP EchoReply 0 0
 (0 data bytes) to eth0
14:10:42 2024/03/10     INFO Flooding packet Ethernet 30:00:00:00:00:02->20:00:00:00:00:01 IP | IPv4 172.16.42.2->192.168.1.100 ICMP | ICMP EchoReply 0 0
 (0 data bytes) to eth2
14:10:42 2024/03/10     INFO in:3 out:6
14:10:42 2024/03/10     INFO Received a packet intended for me
14:10:42 2024/03/10     INFO in:4 out:6
14:10:43 2024/03/10     INFO Received a packet intended for me
14:10:43 2024/03/10     INFO in:5 out:6
```

Step 3: Modify the test scenario of a device

case3&case4

```
14:10:42 2024/03/10     INFO Received a packet intended for me
14:10:42 2024/03/10     INFO in:4 out:6
14:10:43 2024/03/10     INFO Received a packet intended for me
14:10:43 2024/03/10     INFO in:5 out:6
```

Step 4: Run your device in Mininet

```
(syenv) njucs@njucs-VirtualBox:~/CN/Lab1$ 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')
('hub', <TCIntf hub-eth0>, '40:00:00:00:00:01')
('hub', <TCIntf hub-eth1>, '40:00:00:00:00:02')
('server1', <TCIntf server1-eth0>, '10:00:00:00:00:01')
*** 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> pingall
*** Ping: testing ping reachability
client -> X server1
hub -> X X
server1 -> client X
*** Results: 66% dropped (2/6 received)
```

"Node: hub"

```
0:00:00:01 IP | IPv4 192.168.100.3->192.168.100.1 ICMP | ICMP EchoRequest 7503 1
 (56 data bytes) to hub-eth1
14:36:40 2024/03/10     INFO in:3 out:3
14:36:40 2024/03/10     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 7503 1 (
56 data bytes) to hub-eth0
14:36:40 2024/03/10     INFO in:4 out:4
14:36:41 2024/03/10     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 7506 1
 (56 data bytes) to hub-eth0
14:36:41 2024/03/10     INFO in:5 out:5
14:36:41 2024/03/10     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 7506 1 (
56 data bytes) to hub-eth1
14:36:41 2024/03/10     INFO in:6 out:6
14:36:46 2024/03/10     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
14:36:46 2024/03/10     INFO in:7 out:7
14:36:46 2024/03/10     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
14:36:46 2024/03/10     INFO in:8 out:8
```

Step 5: Capture using Wireshark

```
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> 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=856 ms

--- 192.168.100.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 856.615/856.615/856.615/0.000 ms
```

"Node: hub"

```
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-eth1
14:45:25 2024/03/10     INFO in:1 out:1
14:45:25 2024/03/10     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-eth0
14:45:25 2024/03/10     INFO in:2 out:2
14:45:25 2024/03/10     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 7785 1
 (56 data bytes) to hub-eth1
14:45:25 2024/03/10     INFO in:3 out:3
14:45:25 2024/03/10     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 7785 1 (
56 data bytes) to hub-eth0
14:45:25 2024/03/10     INFO in:4 out:4
14:45:30 2024/03/10     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
14:45:30 2024/03/10     INFO in:5 out:5
14:45:31 2024/03/10     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
14:45:31 2024/03/10     INFO in:6 out:6
```

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	0.334042459	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	0.435024717	192.168.100.3	192.168.100.1	ICMP	98	Echo (ping) request id=0x1e69, seq=1/256, ttl=64 (reply in 4)
4	0.756301129	192.168.100.1	192.168.100.3	ICMP	98	Echo (ping) reply id=0x1e69, seq=1/256, ttl=64 (request in 3)
5	5.883197970	Private_00:00:01	30:00:00:00:00:01	ARP	42	Who has 192.168.100.3? Tell 192.168.100.1
6	5.985138164	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·
```

client-eth0: <live capture in progress> Packets: 6 · Displayed: 6 (100.0%) Profile: Default

4. 核心代码

Step 1: Modify the Mininet topology

Delete server2 in the topology：

```
    for node in nodes.keys():
        if node == "server2":
            continue
        self.addHost(node, **nodeconfig)
    for node in nodes.keys():
        # all links are 10Mb/s, 100 millisecond prop delay
        if node == "server2":
            continue
        if node != "hub":
            self.addLink(node, "hub", bw=10, delay="100ms")

def setup_addressing(net):
    for node, config in nodes.items():
        if node== "server2":
                continue
        reset_macs(net, node, config["mac"])
```

Step 2: Modify the logic of a device：

```
while True:
    try:
        _, fromIface, packet = net.recv_packet()
    except NoPackets:
        continue
    except Shutdown:
        break

    sum_in += 1
    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}")
                net.send_packet(intf, packet)
                sum_out += 1
    log_info(f"in:{sum_in} out:{sum_out}")
```

Step 3: Modify the test scenario of a device

```
# test case 4: a frame with dest address of one of the interfaces should
# result in nothing happening
reqpkt = new_packet(
    "20:00:00:00:00:02",
    "10:00:00:00:00:03",
    '192.168.1.100',
    '172.16.42.2'
)
s.expect(
    PacketInputEvent("eth2", reqpkt, display=Ethernet),
    ("An Ethernet frame should arrive on eth2 with destination address "
     "the same as eth2'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.")
)
return s
```

5. 总结与感想

实验还是有一定难度，但通过手册结合 gpt 基本都能得到解答。在 mininet 中进行 pingall 操作时，不理解为什么 hub（集线器）会导致有的包 drop，询问 gpt 得知是由于 ICMP 数据包具有特定的目标地址，在集线器上接收到这些数据包时，由于集线器无法根据目标 IP 地址过滤数据包，集线器会将它们转发到所有连接的端口，包括源主机和目标主机，最终就导致了目标 IP 地址不匹配而 drop。