南京大学本科生实验报告

课程名称：计算机网络　　任课教师：黄程远　　助教：李世鹏、王宸旭，林哲浩

学院	电子科学与工程学院	专业（方向）	通信工程
学号	238352001	姓名	陈海杰
Email	2596917503@qq.com	开始/完成日期	2023/9/15~2023/9/24

1. 实验名称: Switchyard & Mininet

2. 实验目的

在本次实验中，主要介绍网络实验中所需要的软件，运行环境以及编译语言。在实验过程中，对 Linux、Python、Git 等方面的知识加以学习并应用，同时重点学习 Mininet、Wireshark、Switchyard、VS Code 等软件的运行规则和使用方法。

3. 实验内容

Step 1

要求：通过修改文件 start_mininet.py，实现在拓扑中删除 server2 或者使创建一个包含 6 个节点的不同拓扑。

实现：本阶段我选择实现在现有的拓扑中删除 server2。通过阅读代码中的 main 函数可以得知，文件通过实例化 PySwitchTopo 这个类来实现构建拓扑结构。之后仔细阅读 PySwitchTopo 类的定义，可知要想将拓扑中的 server2 删除，需要将 nodes 中的 server2 删除，包括 IP 地址和 MAC 地址。

```
25  nodes = {
26      "server1": {
27          "mac": "10:00:00:00:00:{:02x}",
28          "ip": "192.168.100.1/24"
29      },
30  #将下面的server2以及其相关的代码删除
31      "server2": {
32          "mac": "20:00:00:00:00:{:02x}",
33          "ip": "192.168.100.2/24"
34      },
35  #到此为止
36      "client": {
37          "mac": "30:00:00:00:00:{:02x}",
38          "ip": "192.168.100.3/24"
39      },
40      "hub": {
41          "mac": "40:00:00:00:00:{:02x}",
42      }
43  }
```

Step 2

要求：通过修改文件 myhub.py，实现每收到一个数据包的时候，都要以 in:<ingress packet count> out:<egress packet count> 格式来输出当前累计收到和发出的数据包数目。

实现：首先分析要求，因为需要记录累计收到和发出的数据包，所以需要先定义两个变量并赋予初值为 0。之后通过阅读代码可知，hub 通过调用函数 net.recv_packet()来实现接受数据包，之后对收到的数据包进行分析，有三种情况：header 为空；header 中的目的地是 hub 本身；需要通过 hub 转发，通过调用 send_packet（ ）实现数据包转发。所以在调用 net.recv_packet()和 send_packet（ ）函数后面，对相应的变量实现自增。在最后一个数据包收发结束后，即 while 循环结束时，调用 log_info 按格式输出收到和发出的数据包总量。

```
def main(net: switchyard.llnetbase.LLNetBase):
    my_interfaces = net.interfaces()
    mymacs = [intf.ethaddr for intf in my_interfaces]
    #设置变量
    incount=0
    outcount=0

    while True:
        try:
            _, fromIface, packet = net.recv_packet()
            incount=incount+1#收到数据包数量+1
        except NoPackets:
            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}")
                    net.send_packet(intf, packet)
                    outcount=outcount+1#发出数据包数量+1
        log_info("in:{} out:{}".format(incount,outcount))#输出收到和发出的数据包数量
```

Step 3

要求：通过修改 testcases/myhub_testscenario.py，实现使用给定函数

new_packet（带不同参数）创建一个测试用例或者使用自己的数据包创建一个测试实例。

实现：本阶段我选择实现使用给定函数创建一个测试用例。首先分析 testcases/myhub_testscenario.py 文件，可以得到以下内容：函数 new_packet 的输入参数分别为发送方 MAC 地址、接收方 MAC 地址、发送方 IP 地址、接收方 IP 地址。实际上测试实例主要在 hub_test 函数中定义，有三种测试类型，分别为广播，一对一通信，以及接收方为 hub 本身。 在这里我选择比较简单的广播形式进行创建测试实例，仿照 case1 创建 testpkt 调用 new_packet 函数，修改发送方的 MAC 地址和 IP 地址，将 MAC 地址修改为“20:00:00:00:00:01”，IP 地址修改为“102.168.1.100”同时将输入端口修改为“eth0”，发出端口修改成“eth1”和“eth2”

```
#test case 4 :a frame with broadcast destination should get sent out
#仿照case 1建立测试实例
testpkt = new_packet(
    "20:00:00:00:00:01",
    "ff:ff:ff:ff:ff:ff",
    "192.168.1.100",
    "255.255.255.255"
)
s.expect(
    PacketInputEvent("eth0", testpkt, display=Ethernet),
    ("An Ethernet frame with a broadcast destination address "
     "should arrive on eth0")
)
s.expect(
    PacketOutputEvent("eth1", testpkt, "eth2", testpkt, display=Ethernet),
    ("The Ethernet frame with a broadcast destination address should be "
     "forwarded out ports eth1 and eth2")
)
return s

scenario = test_hub()
```

Step 4

要求：在修改后的拓扑结构中运行 myhub.py，并保证其正常运行。

实现：按照实验手册的教程，在终端中输入 sudo python lab_1/start_mininet.py 的指令，进入修改的拓扑结构后，可以输入 xterm hub，打开 Xterm 更好观察输出，输入 source ./syenv/bin/activate 指令，然后执行 swyard lab_1/myhub.py，使 hub 在当前拓扑下工作起来，之后可以输入 ping 等指令进行测试。/

Step 5

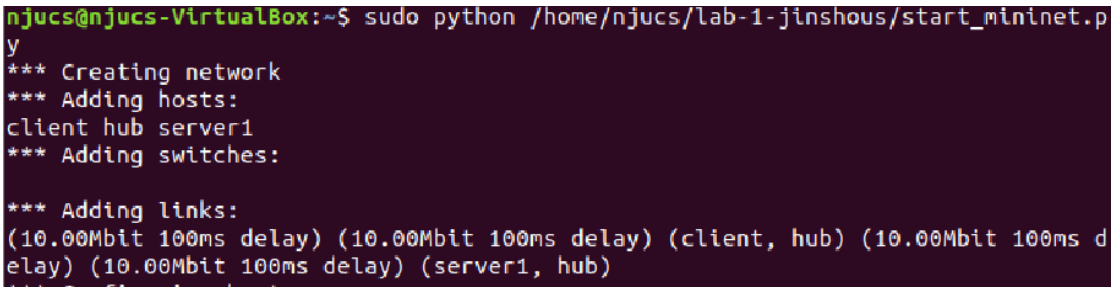
要求：利用 Wireshark 在修改后的拓扑结构中，捕捉除 hub 以外的一个主机通信数据包并进行解释。

实现：按照实验手册的教程，在第四步的基础上，输入 client wireshark &，选择 client-eth0，然后在 mininet 中输入 client ping -c 1 server1，实现对通信数据包的捕捉。

4. 实验结果

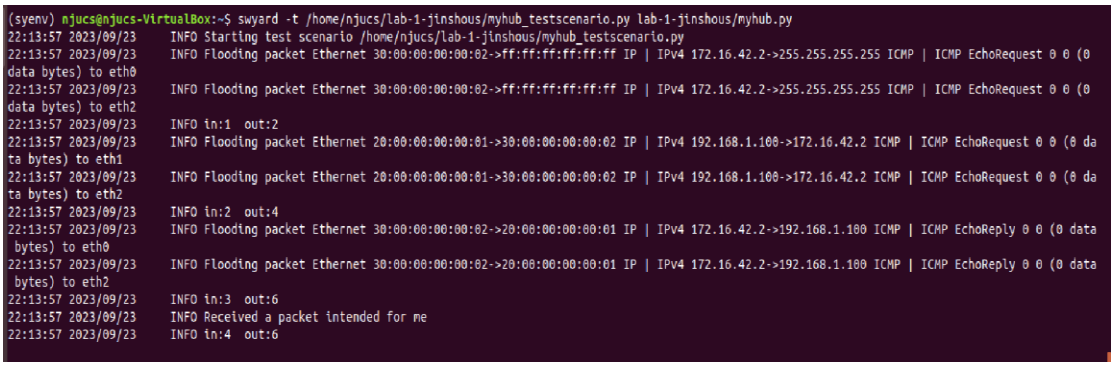
Step 1

删除 server2 后，拓扑结构变为下图所示结果，host 只有 client、hub、server1。



Step 2

在测试环境下运行，结果如下



```
Results for test scenario hub tests: 8 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 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 be
     flooded out eth1 and eth2
5    An Ethernet frame from 30:00:00:00:00:02 to
     20:00:00:00:00:01 should arrive on eth1
6    Ethernet frame destined to 20:00:00:00:00:01 should be
     flooded outeth0 and eth2
7    An Ethernet frame should arrive on eth2 with destination
     address the same as eth2's MAC address
8    The hub should not do anything in response to a frame
     arriving with a destination address referring to the hub
     itself.

All tests passed!
```

Step3

在调整过的 myhub_testscenario.py 的测试环境下，运行 myhub.py 文件，结果如下

```
(syenv) njucs@njucs-VirtualBox:~$ swyard -t /home/njucs/lab-1-jinshous/myhub_testscenario.py lab-1-jinshous/myhub.py
22:18:53 2023/09/23     INFO Starting test scenario /home/njucs/lab-1-jinshous/myhub_testscenario.py
22:18:53 2023/09/23     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
22:18:53 2023/09/23     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
22:18:53 2023/09/23     INFO in:1  out:2
22:18:53 2023/09/23     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 (8 da
ta bytes) to eth1
22:18:53 2023/09/23     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 (8 da
ta bytes) to eth2
22:18:53 2023/09/23     INFO in:2  out:4
22:18:53 2023/09/23     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
22:18:53 2023/09/23     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
22:18:53 2023/09/23     INFO in:3  out:6
22:18:53 2023/09/23     INFO Received a packet intended for me
22:18:53 2023/09/23     INFO in:4  out:6

22:18:54 2023/09/23     INFO Flooding packet Ethernet 20:00:00:00:00:01->ff:ff:ff:ff:ff:ff IP | IPv4 192.168.1.100->255.255.255.255 ICMP | ICMP EchoRequest 0 0 (
0 data bytes) to eth1
22:18:54 2023/09/23     INFO Flooding packet Ethernet 20:00:00:00:00:01->ff:ff:ff:ff:ff:ff IP | IPv4 192.168.1.100->255.255.255.255 ICMP | ICMP EchoRequest 0 0 (
0 data bytes) to eth2
22:18:54 2023/09/23     INFO in:5  out:8
```

```
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 be
    flooded out eth1 and eth2
5   An Ethernet frame from 30:00:00:00:00:02 to
    20:00:00:00:00:01 should arrive on eth1
6   Ethernet frame destined to 20:00:00:00:00:01 should be
    flooded outeth0 and eth2
7   An Ethernet frame should arrive on eth2 with destination
    address the same as eth2's MAC address
8   The hub should not do anything in response to a frame
    arriving with a destination address referring to the hub
    itself.
9   An Ethernet frame with a broadcast destination address
    should arrive on eth0
10  The Ethernet frame with a broadcast destination address
    should be forwarded out ports eth1 and eth2


All tests passed!
```

Step 4

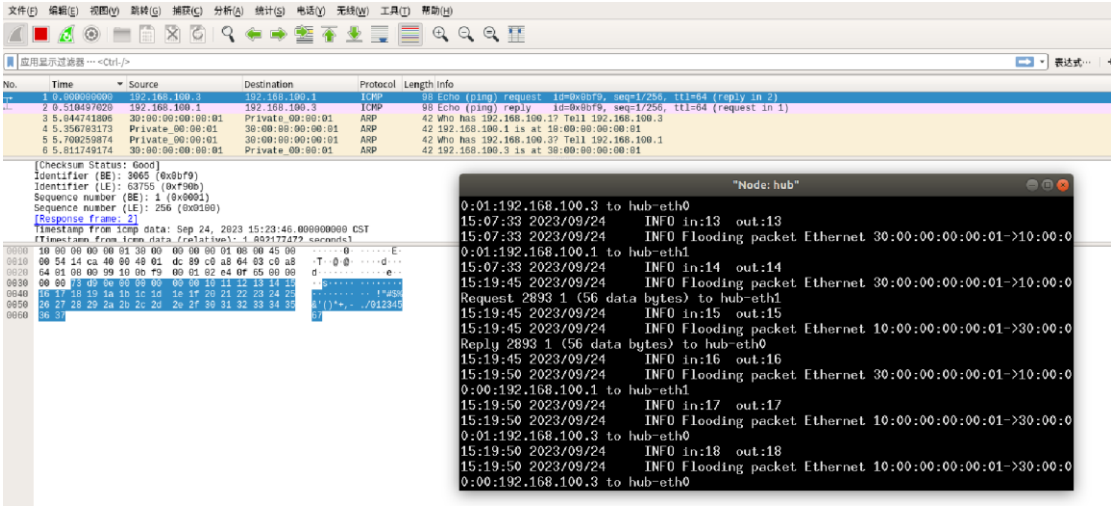
在 mininet 下运行，依次输入 pingall 与 client ping -c 1 server1hub 后的窗口输出结果分别为

```
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
15:02:52 2023/09/24      INFO in:1  out:1
15:02:52 2023/09/24      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
15:02:52 2023/09/24      INFO in:2  out:2
15:02:53 2023/09/24      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 2799 1
 (56 data bytes) to hub-eth1
15:02:53 2023/09/24      INFO in:3  out:3
15:02:53 2023/09/24      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 2799 1 (
56 data bytes) to hub-eth0
15:02:53 2023/09/24      INFO in:4  out:4
15:02:53 2023/09/24      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 2802 1
 (56 data bytes) to hub-eth0
15:02:53 2023/09/24      INFO in:5  out:5
15:02:53 2023/09/24      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 2802 1 (
56 data bytes) to hub-eth1
15:02:53 2023/09/24      INFO in:6  out:6
15:02:58 2023/09/24      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
15:02:58 2023/09/24      INFO in:7  out:7
15:02:58 2023/09/24      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
15:02:58 2023/09/24      INFO in:8  out:8
```

```
mininet> pingall
*** Ping: testing ping reachability
client -> *** Error: could not parse ping output: connect: 网络不可达

X server1
hub -> *** Error: could not parse ping output: connect: 网络不可达

X *** Error: could not parse ping output: connect: 网络不可达

X
server1 -> client *** Error: could not parse ping output: connect: 网络不可达

X
*** Results: 66% dropped (2/6 received)
mininet>
```

Step 5

利用 Wireshark 捕获的结果如下图



结果解析：已知 client 的 MAC 地址为'30:00:00:00:00:01'，IP 地址为'192.168.100.3'；server1 的 MAC 地址为'10:00:00:00:00:01'，IP 地址为'192.168.100.1'，指令为 client ping -c 1 server1。因为 client 开始并不知道 server1 的 MAC 地址，所以先以广播形式发送寻找 server1 的消息，server1 收到消息后，与自己的 IP 地址匹配后，就向 client 发送应答消息，client 收到消息后更新自己的 APR 表，之后 server1 为了更新自己的 APR 表，也广播发送寻找 client，client 收到后，与自己的 IP 地址匹配后，就向 server1 发送应答消息。

5. 核心代码地址

Step 1

```
  },
#将下面的server2以及其相关的代码删除
  "server2":{
    "mac": "20:00:00:00:00:{:02x}",
    "ip": "192.168.100.2/24"
  },
#到此为止
```

Step 2

```
def main(net: switchyard.llnetbase.LLNetBase):
    my_interfaces = net.interfaces()
    mymacs = [intf.ethaddr for intf in my_interfaces]
    #设置变量
    incount=0
    outcount=0

    while True:
        try:
            _, fromIface, packet = net.recv_packet()
            incount=incount+1#收到数据包数量+1
        except NoPackets:
            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}")
                    net.send_packet(intf, packet)
                    outcount=outcount+1#发出数据包数量+1
        log_info("in:{} out:{}".format(incount,outcount))#输出收到和发出的数据包数量
```

Step 3

```
    #test case 4 :a frame with broadcast destination should get sent out
    #仿照case 1建立测试实例
    testpkt = new_packet(
        "20:00:00:00:00:01",
        "ff:ff:ff:ff:ff:ff",
        "192.168.1.100",
        "255.255.255.255"
    )
    s.expect(
        PacketInputEvent("eth0", testpkt, display=Ethernet),
        ("An Ethernet frame with a broadcast destination address "
         "should arrive on eth0")
    )
    s.expect(
        PacketOutputEvent("eth1", testpkt, "eth2", testpkt, display=Ethernet),
        ("The Ethernet frame with a broadcast destination address should be "
         "forwarded out ports eth1 and eth2")
    )
    return s

scenario = test_hub()
```

6. 总结与感想

通过本次实验，我了解了 linux，git，python 的基本知识，同时我还学会 wireshark、xterm、switchyard 的使用方法，同时这次实验还让我对许多知识有了直观的感受，例如在分析 wireshark 捕获的数据包时，我就对主机间的数据通信有了更加深刻的印象与理解，为我之后的学习提供了帮助。