LAB1实验报告

专业	学号	姓名	开始/结束日期
计算机科学与技术系	191220156	张桓	3.14~3.16

实验名称

switchyard & mininet

实验目的

为网络实验做好准备工作，熟悉掌握实验环境。通过编写Python代码在Switchyard中实现具有各种功能的设备。然后在Mininet中运行设备并使用Wireshark捕获数据包。 使用Git来管理项目并提交。使用Linux虚拟机并且用Visual Studio Code作为编辑器。

实验内容

修改mininet拓扑

> 修改`start_mininet.py`代码，在拓扑中删除`server2`结点。

直接在`strat_mininet.py`中将`server2`结点的代码注释掉即可。

- 核心代码

```
nodes = {
    "server1": {
        ...
    },
    #"server2": {
    #   "mac": "20:00:00:00:00:{:02x}",
    #    "ip": "192.168.100.2/24"
    #},
    "client": {
        ...
    },
    "hub": {
       ...
    }
}
```

- 实验结果

```
mininet> nodes
available nodes are:
client hub server1
mininet> dump
<Host client: client-eth0:192.168.100.3 pid=2776>
<Host hub: hub-eth0:10.0.0.2,hub-eth1:None pid=2778>
<Host server1: server1-eth0:192.168.100.1 pid=2780>
mininet>
```

可以看到`server2`已经被成功删除。

打印包的数量信息

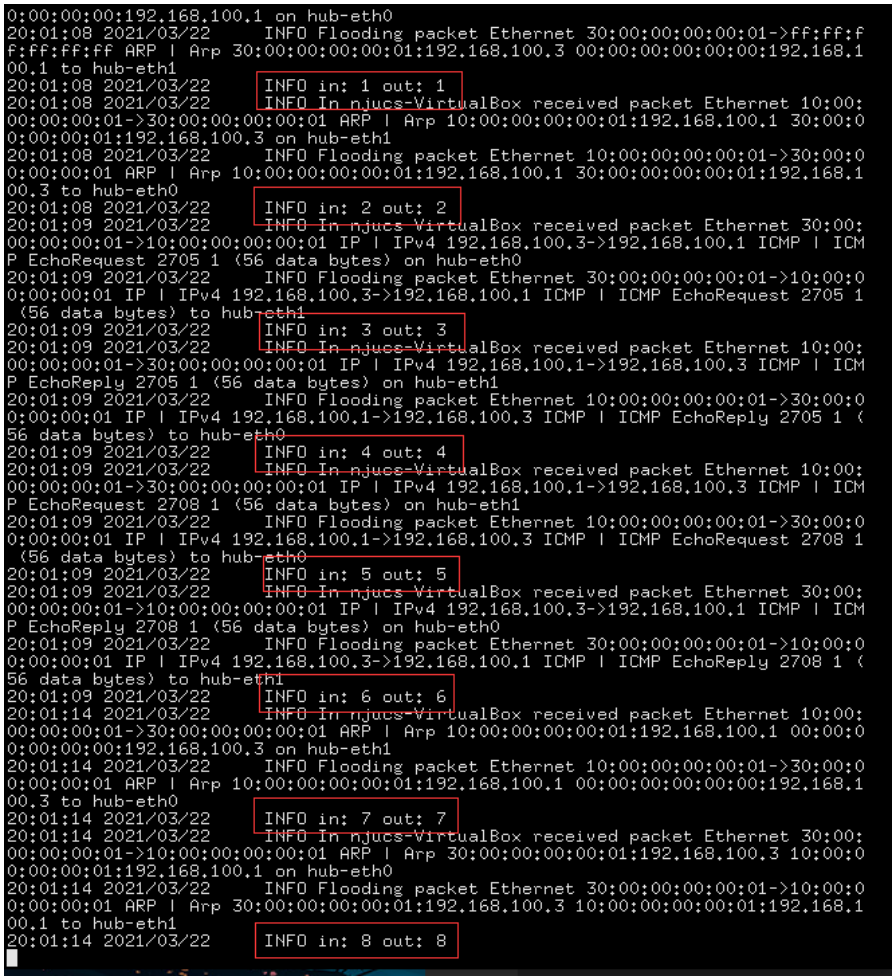
> 修改`myhub.py`代码，计算有多少数据包经过`hub`进出，每次收到一个数据包时按照`in: <ingress packet count> out:<egress packet count>`的格式记录结果。

设置两个变量`ingress, outgress = 0, 0`分别统计接收到的和发送出去的包的数量，每次接受到一个包`ingress`加一，发出去一个包`outgress`加一，`hub`发出去的包**不包含**发给`hub`自己的。每次循环接受包并且判断发出包之后，`log_info`一次数量信息即可。在`mininet`上测试时，用`sudo python start_mininet.py`命令打开拓扑，然后在`mininet CRL`下输入`xterm hub`命令在`hub`上打开`xterm`，然后在上面输入`source switchyard/syenv/bin/activate`激活`python`虚拟环境，再用`swyard myhub.py`执行`hub`的代码，然后`pingall`构造一些流量，可以得到实验结果如下图。

- 核心代码

```
ingress, outgress = 0,0 #TO DO
    while True:
        try:
            ...
        ingress = ingress + 1 #TO DO
            ...
            ...
        else:
            for intf in my_interfaces:
                if fromIface!= intf.name:
                    outgress = outgress + 1 # TO DO
                    log_info (f"Flooding packet {packet} to {intf.name}")
                    net.send_packet(intf, packet)
        log_info(f"in: {ingress} out: {outgress}") #TO DO
```

- 实验结果



可以看到正确打印出了数量信息。

增加测试用例

> 修改`testcases/myhub_testscenario.py`代码，创建一个新的数据包作为测试用例，可以使用`new_packet`函数。

用`new_packet`函数构造一个包，我构造的包的目的MAC地址是`hub`的`eth1`端口，因此`hub`应该在`eth1`处收到这个包但不发出。

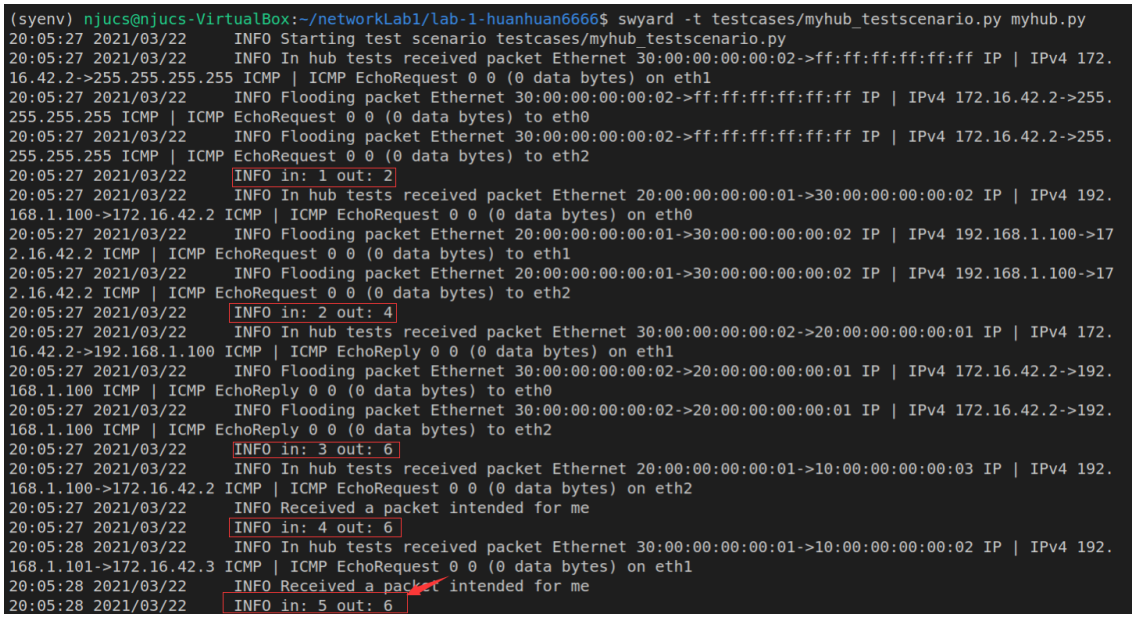
- 核心代码

```
#test case 4: another frame for hub's port eth1
    mypkt = new_packet(
        "30:00:00:00:00:01",
        "10:00:00:00:00:02", #目的MAC为eth1端口的MAC地址
        '192.168.1.101',
        '172.16.42.3'
    )
    s.expect(
        PacketInputEvent("eth1", mypkt, display=Ethernet),
        ("An Ethernet frame should arrive on eth1 with destination address "
         "the same as eth1'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.")
    )
```

- 实验结果

  用`swyard -t testcases/myhub_testscenario.py myhub.py`执行测试用例得到结果如下，`hub`接受到包但不转发，箭头标注的地方为构造的测试用例，可以看到此输出及之前测试用例的输出都是正确的。



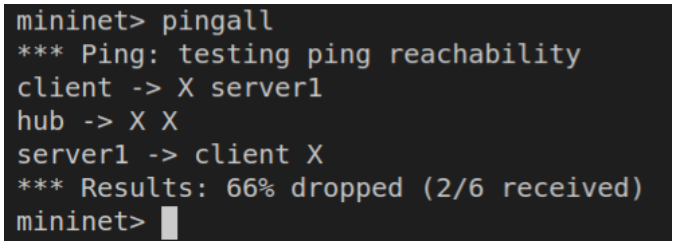
在mininet上运行设备

> 在mininet上运行修改拓扑后的`hub`，并且确保可以正常工作。

首先用`sudo python start_mininet.py`命令打开拓扑，然后在`mininet CRL`下输入`xterm hub`命令在`hub`上打开`xterm`，然后在上面输入`source switchyard/syenv/bin/activate`激活`python`虚拟环境，再用`swyard myhub.py`执行`hub`的代码。

- 实验结果

  用`pingall`构造一些流量，可以看到正常工作了



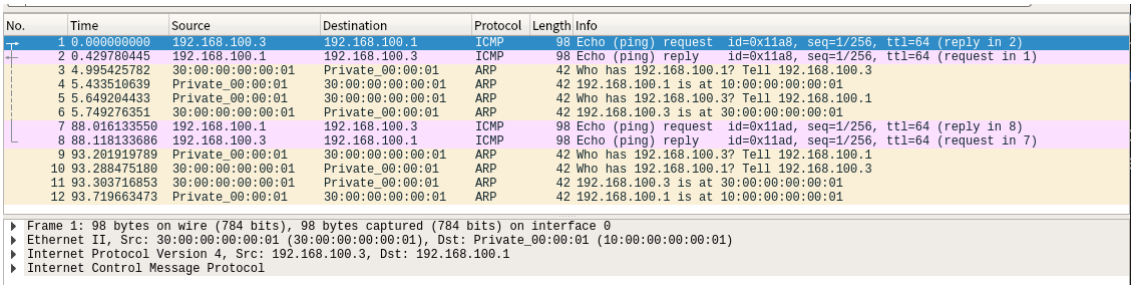
用wireshark捕获数据包

> 创造一些流量并在主机上捕获数据包，不能在`hub`上，并将捕获的结果保存。

选择在`client`上捕获流量，输入命令`client wireshark &`，即在`client`上运行`wireshark`，双击`client-eth0`开始监控`eth0`端口流量。然后构造一些流量，我用了两个命令`client ping -c1 server1`和`server1 ping -c1 client`即`client`和`server1`互相`ping`。捕获到如下结果：

- 实验结果

  捕获到的文件



  当`client ping -c1 server1`时，`client`发出的`ARP`广播请求数据包寻找`server1`的`IP`对应的`MAC`地址，并且收到对方的回应，同样`server1 ping -c1 client`时，`server1`也发出`ARP`广播数据包寻找`client`的`MAC`地址，并且收到回应，实验成功了。

总结与感想

这次实验的的最大收获是大概理清了`mininet switchyard wireshark`的各自的功能和关系。其中`mininet`是一个虚拟网络管理器，创建多个相互隔离的虚拟网络空间和虚拟网络接口，这些虚拟网络空间其实就是实验中的主机`client server`等，参照`start_mininet.py`的代码，我们可以知道`mininet`自定义了多个`nodes`并且定义了每个`nodes`的`IP`和`MAC`等，并且将结点以某种拓扑连接起来搭建成一个形式上完整的虚拟网络。

但是`mininet`创造的结点只有**普通主机**的功能，可以作为`client`和`server`，可是实验中的`hub`需要的功能显然不止这些，`hub`要有接收分析转发的功能。这时候就需要`switchyard`了，我们编写好`switchyard`程序即实验中的`myhub.py`。然后在`mininet`中打开`hub`的`xterm`，在上面运行`muhub.py`，`myhub.py`会绑定`hub`的网络接口，参照其代码我们会发现它实现了接收分析转发的功能。因此`switchyard`是在`mininet`的基础上，用自己的`python`代码实现某个特定网络设备的功能，比如交换机，本实验中的`hub`等。

`wireshark`相对前两者独立一点，它是用来捕获指定接口上流过的数据包的。我们用`mininet`构建好一个网络拓扑，然后用`switchyard`实现所谓`hub`结点的功能，这个网络实际上已经完成了，用`wireshark`去对某个特定接口抓包，可以用来检测网络是否正确。