**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：

|  |  |  |  |
| --- | --- | --- | --- |
| 学院 | **计算机科学与技术系** | 专业（方向） | **计算机** |
| 学号 | **181860077** | 姓名 | **佘帅杰** |
| Email | [**3121416933@qq.com**](mailto:3121416933@qq.com) | 开始/完成日期 | **2020.3.3-2020.3.8** |

1. **实验名称**

实验1

1. **实验目的**

对实验的模块进行学习和尝试使用

1. **实验内容**

**实验内容**：

Task1,Task2,Task3：学习模块和实验所需

Task4：修改代码和应用模块

1. 删除server2或建立一个新拓扑
2. 统计出入包的数量并打印
3. 修改测试样例
4. 在Mininet运行网络保证一切正常
5. 利用wireshark保存包

**理论知识**

基本模块的使用以及python的基本语法

Git的使用

基本局域网广播机制

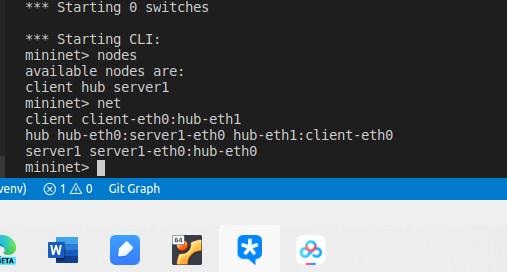
1. **实验结果**

前面的三个task没有啥好分析的，来到task4

**删除server2或建立一个新拓扑**

实现思路：

创建Topo的时候用的类PySwitchTopo。所以直接来到start\_mininet文件，这里是对网络的结构进行配置的地方。可以看到配置了hub，server1，server2还有client三个节点。然后连接三者配置mac地址等工作。想要删除server2，就注释了相关的代码。效果如下，可以看到server2没了，整个网络就是hub连着server1和client了。

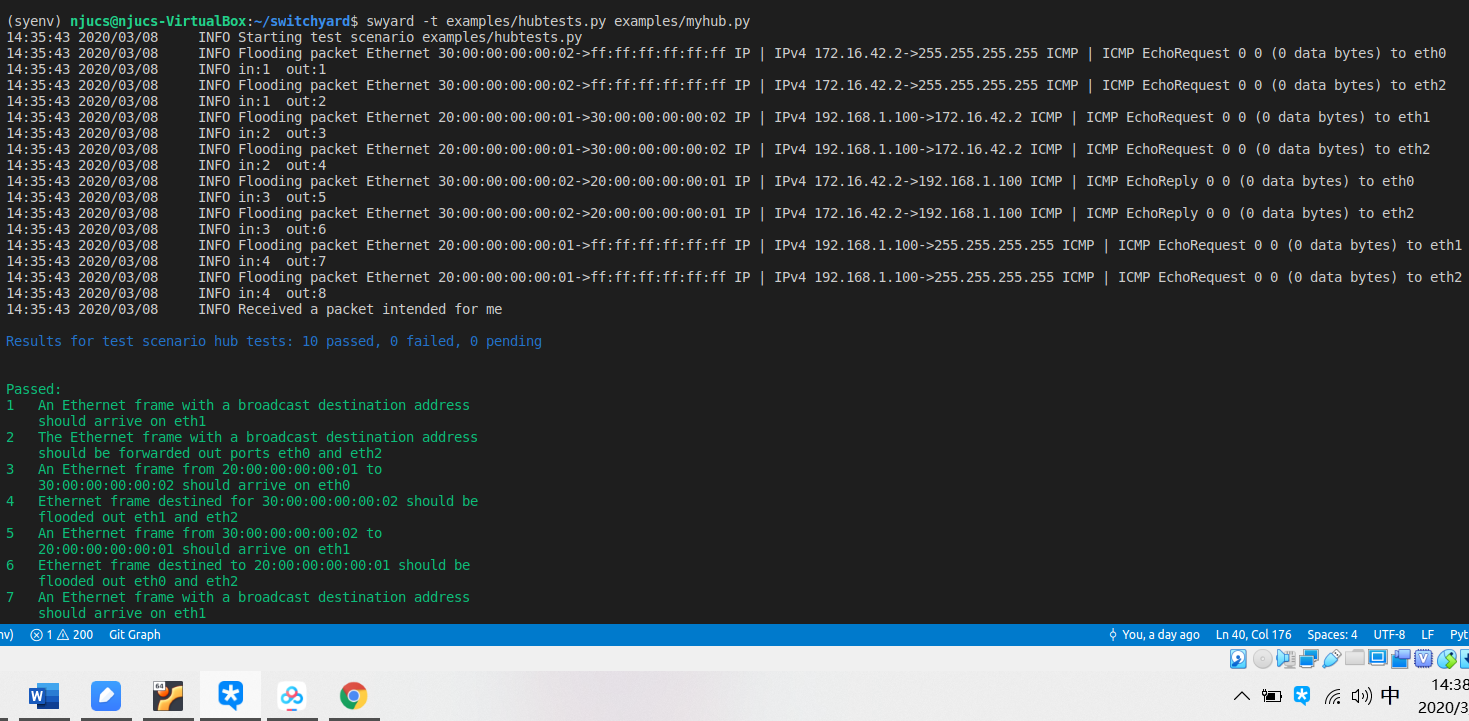


**统计出入包的数量并打印**

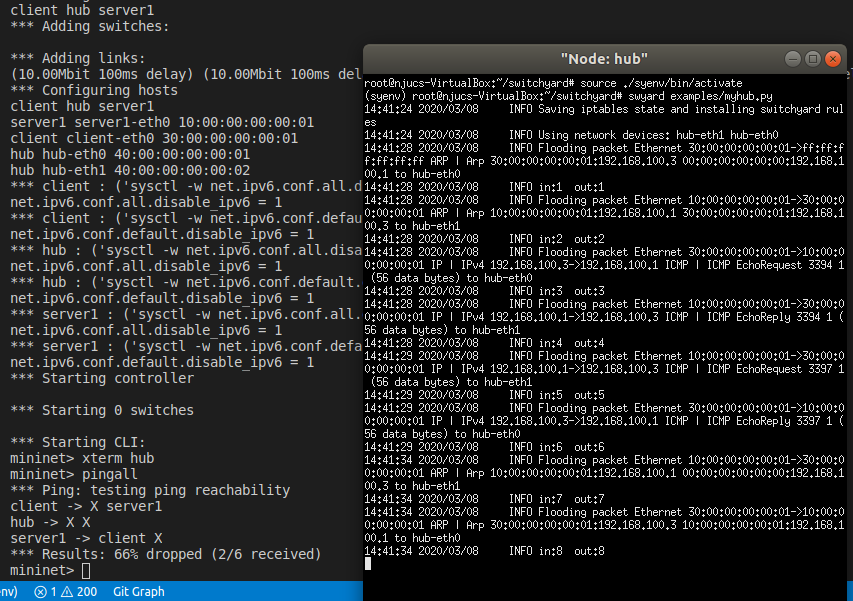
实现思路：出入包的数量，这里是基于hub的，于是来到my\_hub文件，里面是对hub的具体功能的配置，可以看到里面对接受包的情况进行了讨论和处理。

发现在代码里有log函数可以打印出日志信息，还有的发现就是timestamp, dev, packet = net.recv\_packet()。这个函数是收到了包，所以这里可以作为统计收到的包的切入口，于是在这里设置计数变量。继续向下阅读，接下来是对收到的包进行情况讨论。然后在下方的循环里有net.send\_packet(intf, packet)。根据函数名这里应该是发送了包，同样设置计数变量。然后在for循环的末端加上了一个log函数以打印出信息。结果如下：

直接进行测试



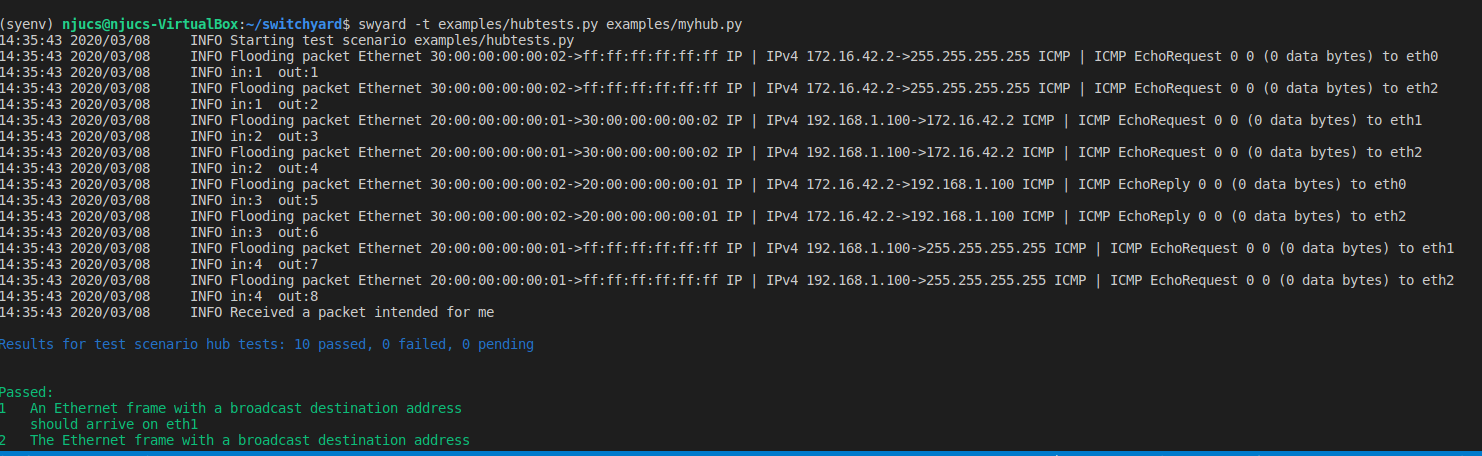
使用xtermhub 和 pingall进行尝试



**修改测试样例**

测试的样例是在hubtest文件里，打开文件阅读代码之后仿写一个

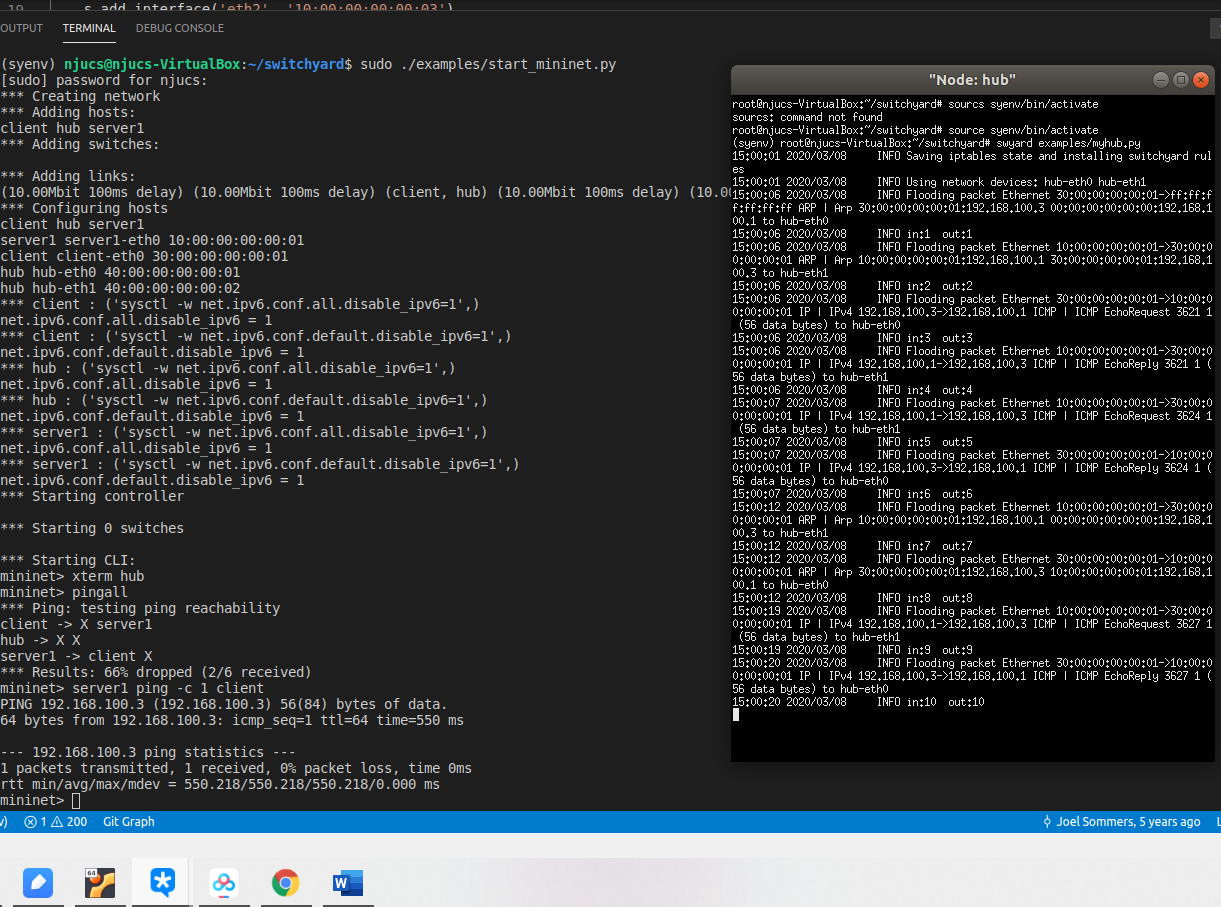
样例具体就是广播模式，20:00:00:00:00:01发出，广播的目的地址。从eth0接受，在eth1和eth2发送出去的测试。然后测试的结果就如下（最后一个是新建的）



**在Mininet运行网络保证一切正常**

上面Problem2其实就已经在mininet运行了新的Topo了。

再运行一次如下



总的流程就是首先是start\_mininet文件启动，配置了三个节点server1，hub和client。然后连接节点以及配置地址等。随后转入Myhub，对这个中间节点进行配置，写好了对应收到包以及如何处理到最后发出包的工作。Xterm hub启动独立的窗口，打开虚拟环境和运行myhub，这个时候hub就启动了，有了一个左右中间节点的能力（在这之前直接pingall的话会100%loss，没有hub就没有连接）。然后在mininet里使用pingall指令，节点就开始互相发数据。Myhub把数据转发，log函数就打印出了对应的信息。总的工作流程就是这样。

**利用wireshark保存包**

如实验手册指导，输入server1 wireshark &唤起wireshark

然后server1 ping -c 1 client制造流量得到了如下结果

然后任意选一个为例进行主要参数的简单分析。（图在下页，word插入图片好像不太友好，，，会跑到奇怪的地方，试了很久只能这样了）

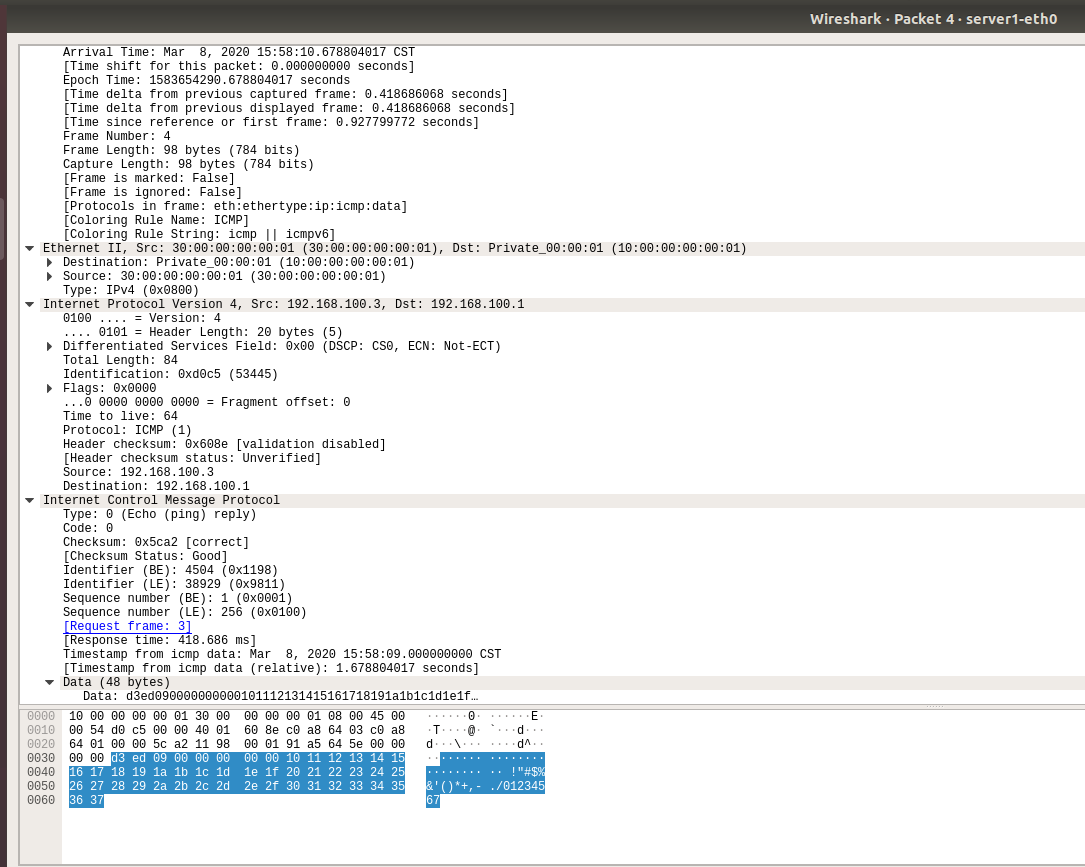
第一行，帧Frame 1 指的是要发送的数据块，其中，所抓帧的序号为1，捕获字节数等于传送字节数98字节；里面还有arrive\_time 等等参数

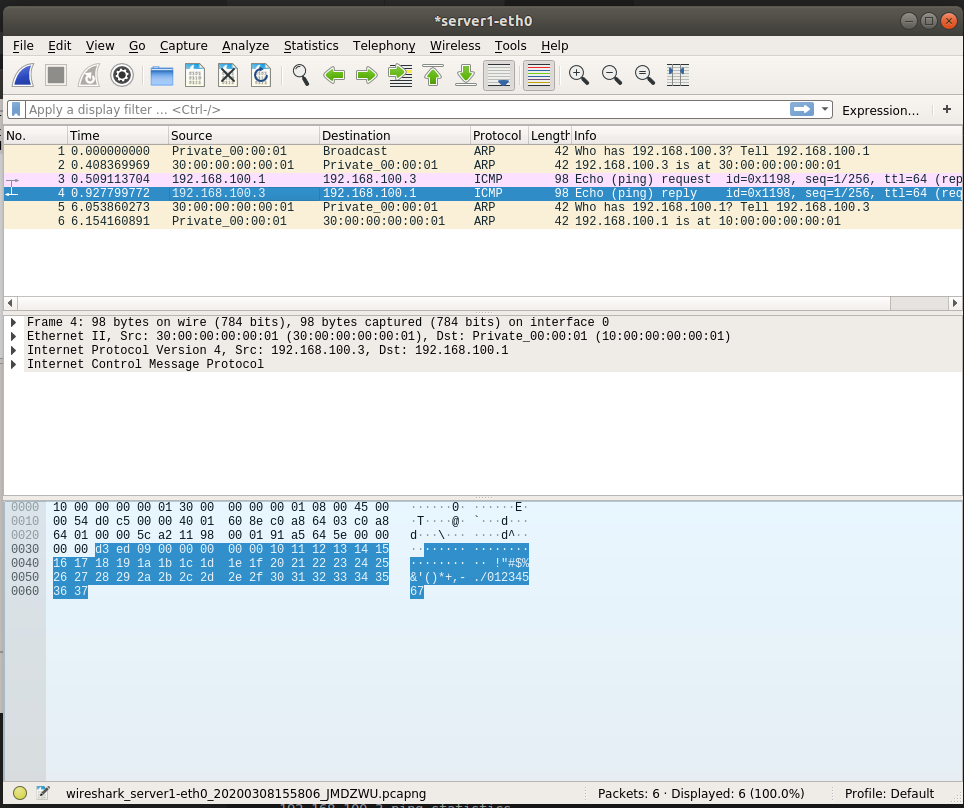
第二行，以太网，有线局域网技术，是数据链路层。源Mac地址为30：00：00：00：00：01；目标Mac地址为10：00：00：00：01；type是0x800也就是表明是IP数据包

第三行，IPV4协议，也称网际协议，是网络层；源IP地址为192.168.100.3；目标IP地址为192.168.100.1；version4表明IPV4协议，还有Time to live TTL数字64表明包经过64路由器没到达会被丢弃。Protocol：ICMP采用的协议是ICMP。Headchecksum：头部校验和0x608e

第四行，ICMP协议的具体

Type可以看到是0对应了ping reply，以及checksum校验和的状态：good correct正确，以及最后的时间戳。最后是48字节数据，占了整个数据包的一大半





1. **核心代码**

下面的代码就都是节选出修改部分

Start\_mininet

基本如下注释server2相关，修改连接代码，下面设置ip，mac的函数也一样，没有创建自然就没有server2节点了

nodeconfig = {'cpu':-1}

self.addHost('server1', \*\*nodeconfig)

#self.addHost('server2', \*\*nodeconfig)

self.addHost('hub', \*\*nodeconfig)

self.addHost('client', \*\*nodeconfig)

'''

for node in ['server1','server2','client']:

# all links are 10Mb/s, 100 millisecond prop delay

self.addLink(node, 'hub', bw=10, delay='100ms')

'''

for node in ['server1','client']:

# all links are 10Mb/s, 100 millisecond prop delay

self.addLink(node, 'hub', bw=10, delay='100ms')

myhub

基本思路就是加上计数代码，下面就是接受的异常处理程序，在rec加上1，如果是没有包就没有加。同理在send位置加上，最后打印。

try:

timestamp, dev, packet = net.recv\_packet()

in\_num+=1

except NoPackets:

continue

except Shutdown:

return

for intf in my\_interfaces:

#print("testcase1:",intf.name)

#print("testcase2:",dev)

if dev != intf.name:

log\_info("Flooding packet {} to {}".format(packet, intf.name))

net.send\_packet(intf, packet)

out\_num+=1

log\_info("in:{} out:{}".format(in\_num, out\_num))

hubtest

创建的思路上面已经解释过了

#mytestcase

testpkt=mk\_pkt("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")

1. **总结与感想**

总结：主要就是熟悉了相关的模块使用方法，还有就是配置swyard环境等。然后就是对实验的流程比较熟悉了一些。

感想：一开始做的太蒙了，视频看了不知道怎么用到任务上，英文看到略有吃力。不知道要怎么做，不过群里大家讨论之后稍微好了一些（但还是不确定，感觉自己做的不太对。。。）

建议：实验报告可以用markdown（或者交Pdf）交吗，不然贴图和粘贴代码不太友好···

**附件：（源代码）**