南京大学本科生实验报告

181860097 计算机科学与技术系 王明骁
Email：1164151483@qq.com
开始日期：2020年3月11日 结束日期：2020年3月13日
lab1

实验名称

Lab 1: Switchyard & Mininet

实验目的

- 学习python的基础语法
- 学习和掌握wireshark，mininet，switchyard的相关使用
- 理解网络拓扑的构建代码，学会更改相关内容达到一些目的

实验内容

- step 1

  step 1中选择了在网络拓扑中删去server2

  由于分布式网络中一个结点的删除并不会直接影响到该网络本身，因此只需要将和server2相关的代码删去即可

  在start_mininet.py中注释了设置server2的ip和mac信息的代码，同时更改

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

  为：for node in ['server1','client']:

- step 2

  首先是对于muhub.py中的hub相关逻辑的理解：

```
    log_debug ("In {} received packet {} on {}".format(net.name, packet, dev))
    eth = packet.get_header(Ethernet)
    cnt_in += 1
    if eth is None:
        log_info("Received a non-Ethernet packet?!")
        continue

    if eth.dst in mymacs:
        log_info ("Received a packet intended for me")

    else:
        for intf in my_interfaces:
            if dev != intf.name:
                log_info ("Flooding packet {} to {}".format(packet, intf.name))
                cnt_out += 1
                net.send_packet(intf, packet)
    log_info("in:{} out:{}".format(cnt_in,cnt_out))
net.shutdown()
```

  从中可以知道hub每收到一个包时，首先判断这个包是不是从以太网收到的，如果不是则不处理，否则判断一下包的目标mac是不是自己，如果是自己的mac，则收下后不转发，否则对除了来源端口之外的端口转发，因此可以知道在收到包时计数可得到收到的总数，在转发时计数可得到转发的总数，最后将其按照格式输出即可

- step 3

  step 3中选择通过传递给mk_pkt函数不同的参数来构造不同的测试样例

```
def mk_pkt(hwsrc, hwdst, ipsrc, ipdst, reply=False):
    ether = Ethernet(src=hwsrc, dst=hwdst, ethertype=EtherType.IP)
    ippkt = IPv4(src=ipsrc, dst=ipdst, protocol=IPProtocol.ICMP, ttl=32)
    icmppkt = ICMP()
    if reply:
        icmppkt.icmptype = ICMPType.EchoReply
    else:
        icmppkt.icmptype = ICMPType.EchoRequest
    return ether + ippkt + icmppkt
```

mk_pkt函数中通过接收发送方物理地址，目标物理地址，发送方ip和目标ip四个参数来构造一个包，因此只需要准备这四个参数传递给mk_pkt函数即可构造好一个包，在准备发送的包后，仿照已有代码的测试样例的发送方式，使用s.expcet来完成一个测试样例的构建

```
# test case 4: a frame created by myself
mytest = mk_pkt("20:00:00:00:00:01", "ff:ff:ff:ff:ff:ff", "192.168.1.100",'172.16.42.2')
s.expect(PacketInputEvent("eth1", mytest, display=Ethernet),"An Ethernet frame with a broadcast destination addre
s.expect(PacketOutputEvent("eth0", mytest, "eth2", mytest, display=Ethernet),"The Ethernet frame with a broadcast
```

具体构建了一个从20 : 00 : 00 : 00 : 00 : 01 到ff : ff : ff : ff: ff : ff : ff的包，该包将会将在网络上进行广播，而hub会进行转发

- step 4

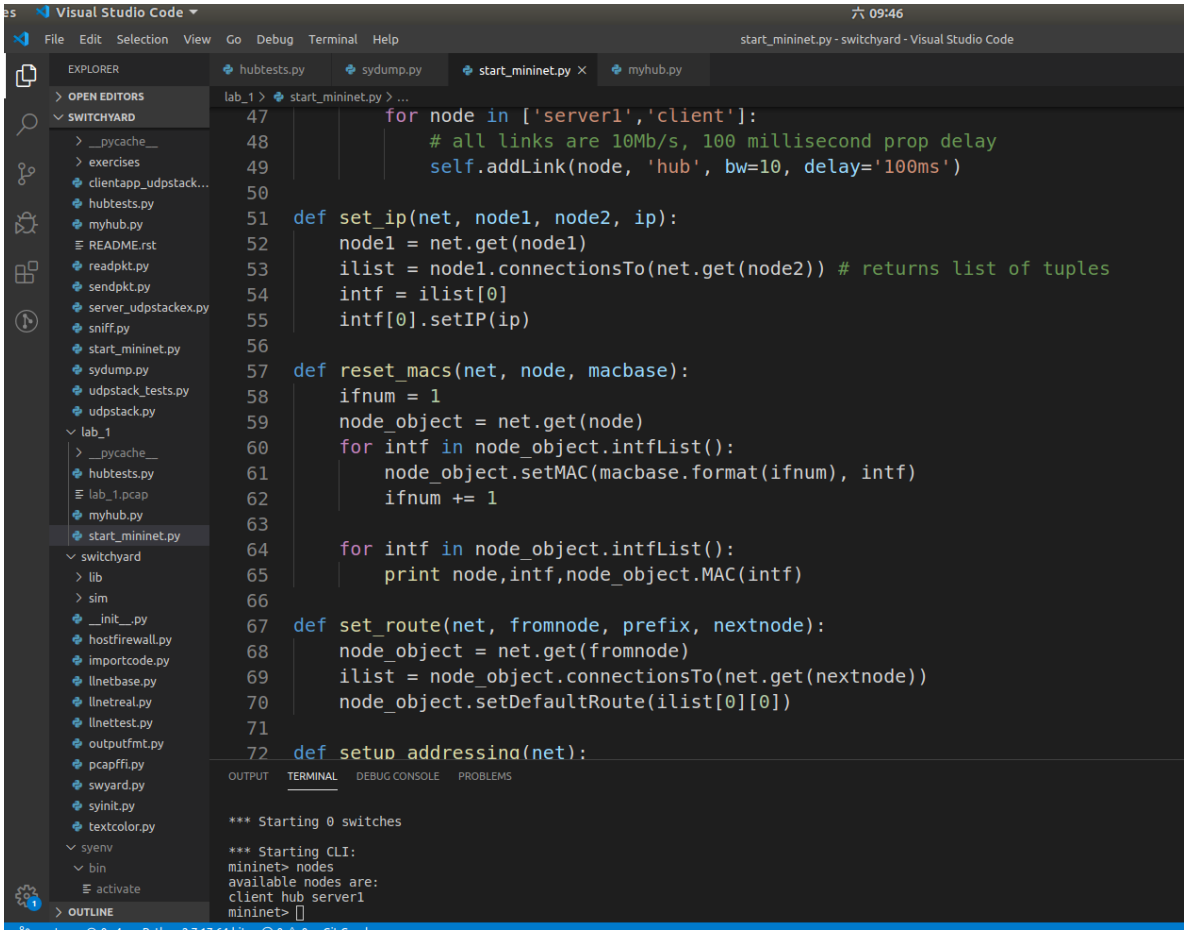
  根据视频教程的讲解可知道，首先运行虚拟环境，然后在虚拟环境下运行sudo python lab_1/start_mininet.py启动mininet，在mininet的cli中运行xterm hub启动xterm，在xterm中运行虚拟环境，然后运行swyard lab_1/muhub.py即可运行我构建的网络拓扑，在网络拓扑中可以运行pingall构造一些流量来看自己的网络构建是否正确，也可以在虚拟环境下运行swyard -t lab_1/hubtest.py lab_1/myhub.py来查看由测试样例构造的流量

- step 5

  根据视频教程中对wireshark的讲解可以知道抓包的具体方法，首先按照step 4 中的步骤运行自己的网络拓扑，然后在mininet中运行server1 wireshark &可在server1上启动wireshark，此时运行指令pingall构造一些流量可以看到wireshark中出现了构造的流量的相关信息，抓取到的包的具体说明在实验结果中列出

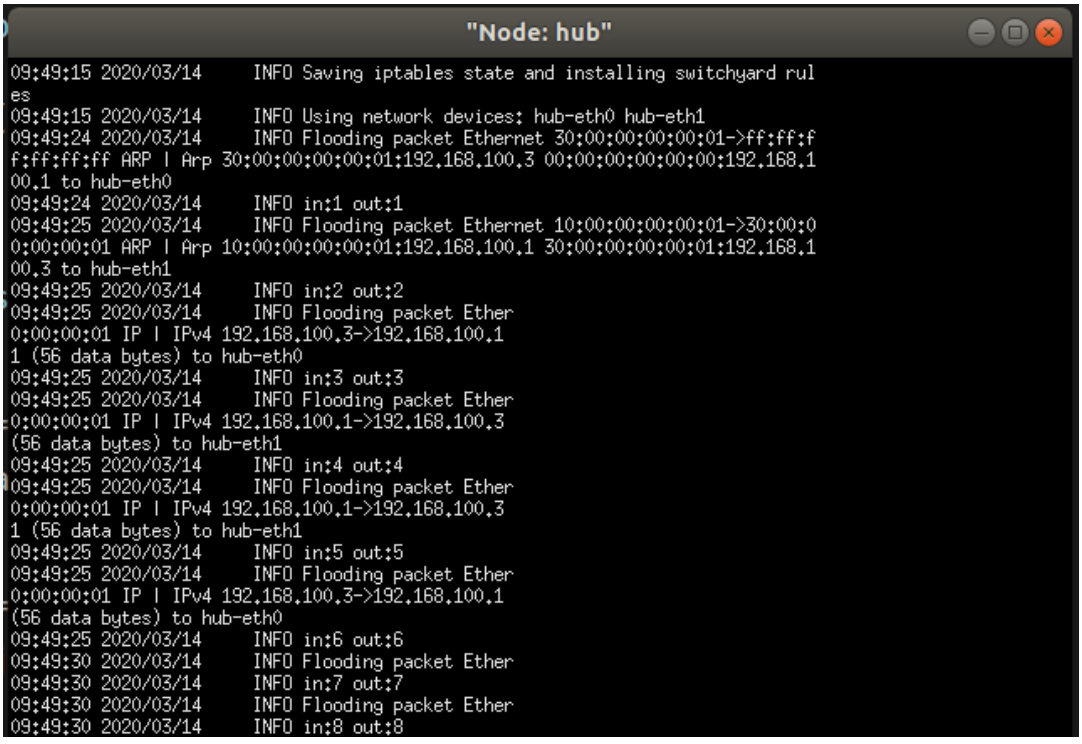
实验结果

- step 1



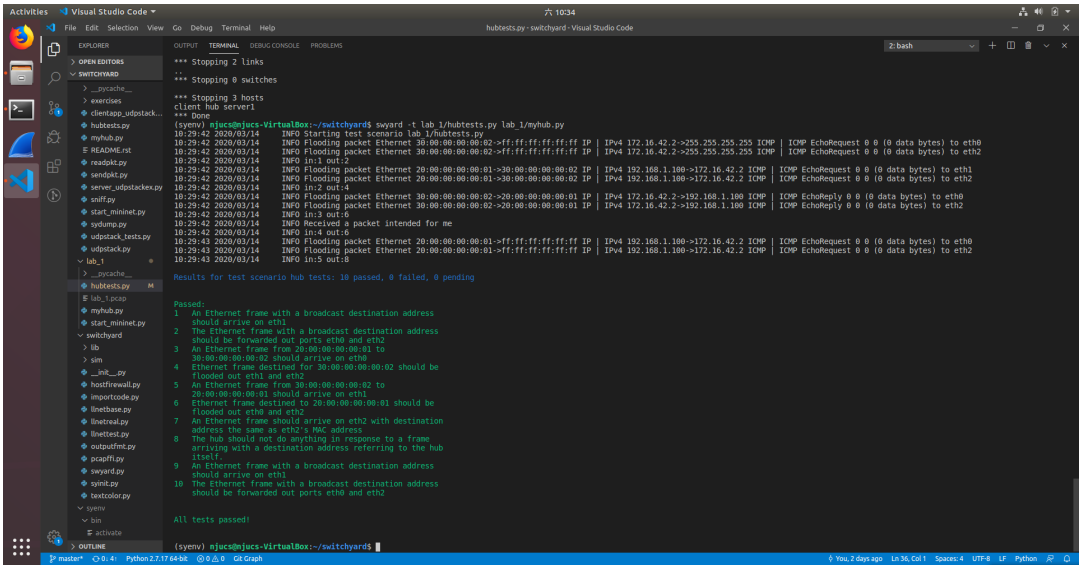
在mininet中运行nodes指令可以看到server2确实已经被删除

- step 2



在mininet中运行pingall指令构造一些流量，在xterm中可以看到输出信息（运行测试样例的流量在step 3 的实验结果中展示，同样可以看到输出信息）

- step 3



运行swyard -t lab_1/hubtest.py lab_1/myhub.py可以看到测试样例成功构建，且能被正确执行

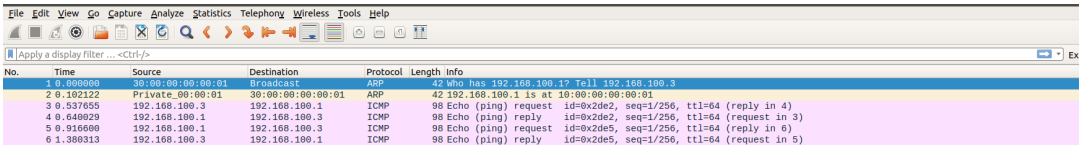
- step 4

```
(syenv) njucs@njucs-VirtualBox:~/switchyard$ sudo python lab_1/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
server1 server1-eth0 10:00:00:00:00:01
client client-eth0 30:00:00:00:00:01
hub hub-eth0 40:00:00:00:00:01
hub 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> xterm hub
mininet> pingall
*** Ping: testing ping reachability
client -> X server1
hub -> X X
server1 -> client X
*** Results: 66% dropped (2/6 received)
mininet>
```

在mininet中运行该网络，并构造一些流量，可以看到该网络正常

- step 5

  运行pingall得到相关的流量被wireshark抓取



在wireshark中可以看到抓取到的包，首先是ARP request 请求192.168.100.1的MAC地址，然后是ARP reply 回复物理地址为10 : 00 : 00 : 00 : 00 : 01

然后pingall所构造的流量，从192.168.100.3发送到192.168.100.1，然后回复，然后从192.168.100.1发送到192.168.100.3，然后回复

核心代码

- step 1

  注释了对server2的相关设置

```
def setup_addressing(net):
    reset_macs(net, 'server1', '10:00:00:00:00:{:02x}')
  #  reset_macs(net, 'server2', '20:00:00:00:00:{:02x}')
    reset_macs(net, 'client', '30:00:00:00:00:{:02x}')
    reset_macs(net, 'hub', '40:00:00:00:00:{:02x}')
    set_ip(net, 'server1','hub','192.168.100.1/24')
  #  set_ip(net, 'server2','hub','192.168.100.2/24')
    set_ip(net, 'client','hub','192.168.100.3/24')
```

同时更改for

```
        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']:
        for node in ['server1','client']:
            # all links are 10Mb/s, 100 millisecond prop delay
            self.addLink(node, 'hub', bw=10, delay='100ms')
```

- step 2

```
    log_debug ("In {} received packet {} on {}".format(net.name, packet, dev))
    eth = packet.get_header(Ethernet)
    cnt_in += 1
    if eth is None:
        log_info("Received a non-Ethernet packet?!")
        continue

    if eth.dst in mymacs:
        log_info ("Received a packet intended for me")

    else:
        for intf in my_interfaces:
            if dev != intf.name:
                log_info ("Flooding packet {} to {}".format(packet, intf.name))
                cnt_out += 1
                net.send_packet(intf, packet)
    log_info("in:{} out:{}".format(cnt_in,cnt_out))
net.shutdown()
```

在hub收到包时对收到的包进行计数，在hub发出包后对发出的包进行计数

- step 3

```
# test case 4: a frame created by myself
mytest = mk_pkt("20:00:00:00:00:01", "ff:ff:ff:ff:ff:ff", "192.168.1.100",'172.16.42.2')
s.expect(PacketInputEvent("eth1", mytest, display=Ethernet),"An Ethernet frame with a broadcast destination addre
s.expect(PacketOutputEvent("eth0", mytest, "eth2", mytest, display=Ethernet),"The Ethernet frame with a broadcast
```

提供给mk_pkt函数相关参数构造包，并发送来构造测试样例

- step 4和step 5未涉及代码相关部分

总结与感想

本次实验是从0到1的突破，我学会了网络实验的相关工具的使用，同时也对网络相关的代码有了初步的了解，虽然完成了本次实验，但是框架代码中的一些细节的地方还不是太清楚，相信通过之后的学习可以更深入的理解网络相关的知识和网络相关的代码的编写