Computer Networking-lab1-Repot

课程名称：**计算机网络** 任课教师：田臣/李文中

学院	Dept. of Computer Science and Technology	专业（方向）	CS
学号	181830044	姓名	董宸郅
Email	pdon#foxmail.com	开始/完成日期	3.1/3.18

实验名称：Switchyard & Mininet Walkthrough

实验目的

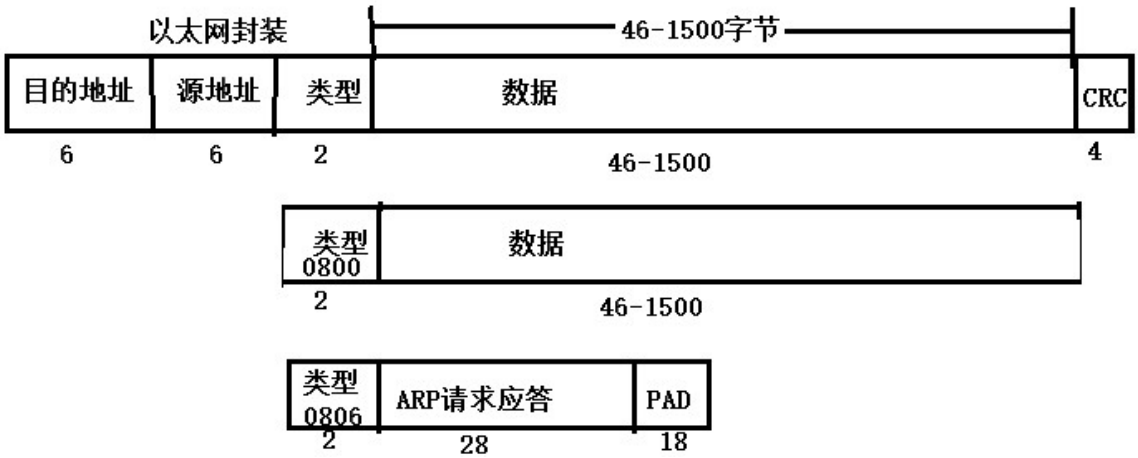
- 安装和配置好必要的实验环境或工具，如LInux, Python, Git, VSC, Mininet, Wireshark, Switchyard
- 熟悉实验流程
- 掌握上述工具的调用、调试方法
- 修改一部分给出的代码以练习工具的使用，加深对数据链路层的理解

实验内容

理论知识

Ethernet：

- 帧格式：



- 工作方式：

  CSMA/CD(载波监听多路访问/冲突检测)

  对于发送端来说，它每发送一个数据时，首先对网络进行监听，当它检测到线路正好有空，便立即发送数据，否则继续检测，直到线路空闲时再发送。 对于接收端来说，对接收到的信号首先进行确认，如果是发给自己的就接收，否则不予理睬。

MAC地址

含义：媒体访问控制，或物理地址，或硬件地址 用途：识别数据链路层中相连的节点。 组成：6个字节，一般用16进制数字加上冒号的形式来表示。在网卡出厂时就确定了，不能修改，mac地址通常是唯一的

Hub、Switch概念辨析

Hub是一种共享设备，本身不能识别目的地址。采用广播的形式传输数据，即向所有端口传送数据。

Switch基于mac地址识别，能完成封装转发数据功能的设备。可以记录mac地址，放在内部地址表中，通过在数据帧的始发者和接收者之间建立临时的交换路径，使数据从源地址到达目的地址。

实验步骤（含结果与关键代码）

Step 1: Modify the Mininet topology

将server2在原拓扑中删除，前后对比图如下所示

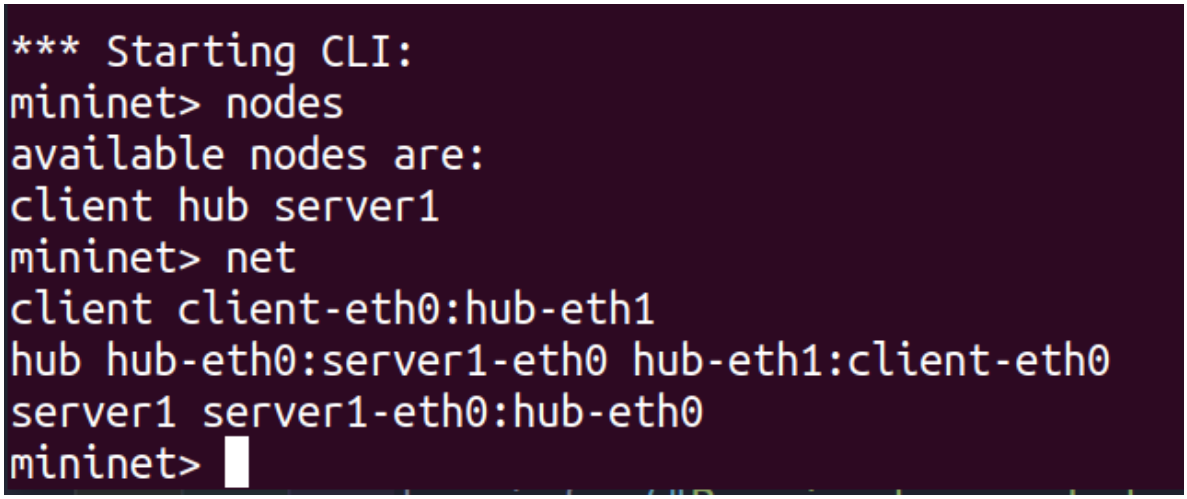
代码：

```
nodeconfig = {'cpu':-1}
        self.addHost('server1', **nodeconfig)
#        self.addHost('server2', **nodeconfig)
        self.addHost('hub', **nodeconfig)
        self.addHost('client', **nodeconfig)

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

```

效果：



Step 2: Modify the logic of a device

每次记录hub接收和发送packet的当前总和，调用switchyard提供的log_info()输出

代码：

```
log_debug ("In {} received packet {} on {}".format(net.name, packet, dev))
        eth = packet.get_header(Ethernet)
        # in_cnt should +1 when another packet received
        in_cnt+=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")
            log_info ("in:{} out:{}".format(in_cnt, out_cnt))
        else:
            # add the exact number of interfaces in my_interface except the receiving             port itself to out_cnt
            out_cnt+=len(my_interfaces)-1
            # add log_info as required
            log_info ("in:{} out:{}".format(in_cnt, out_cnt))
            for intf in my_interfaces:
                if dev != intf.name:
                    log_info ("Flooding packet {} to {}".format(packet, intf.name))
                    net.send_packet(intf, packet)
    net.shutdown()
```

效果：

```
pdon@ubuntu ~/switchyard % swyard -t lab_1/hubtests.py lab_1/myhub.py
15:39:17 2020/03/18     INFO Starting test scenario lab_1/hubtests.py
15:39:17 2020/03/18     INFO in:1 out:2
15:39:17 2020/03/18     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
15:39:17 2020/03/18     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
15:39:17 2020/03/18     INFO in:2 out:4
15:39:17 2020/03/18     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
15:39:17 2020/03/18     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
15:39:17 2020/03/18     INFO in:3 out:6
15:39:17 2020/03/18     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
15:39:17 2020/03/18     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
15:39:17 2020/03/18     INFO Received a packet intended for me
15:39:17 2020/03/18     INFO in:4 out:6
15:39:18 2020/03/18     INFO in:5 out:8
15:39:18 2020/03/18     INFO Flooding packet Ethernet 00:11:22:33:44:55->66:55:44:33:22:11 IP | IPv4 1.1.1.1->2.2.2.2 ICMP | ICMP EchoRequest 0 0 (0 data bytes) to eth1
15:39:18 2020/03/18     INFO Flooding packet Ethernet 00:11:22:33:44:55->66:55:44:33:22:11 IP | IPv4 1.1.1.1->2.2.2.2 ICMP | ICMP EchoRequest 0 0 (0 data bytes) to eth2
```

Step 3: Modify the test scenario of a device

我选择在`mk_pkt`函数中，以不同的参数创建新的packet，作为test case 4进行测试

代码：

```
# test case 4:
    new_pkt = mk_pkt("00:11:22:33:44:55", "66:55:44:33:22:11", "1.1.1.1",
"2.2.2.2")
    s.expect(PacketInputEvent('eth0', new_pkt, display=Ethernet), "The frame
should arrive on eth0.")
    s.expect(PacketOutputEvent('eth1', new_pkt, "eth2", new_pkt,
display=Ethernet), "The frame should be flooded out eth1.")
```

效果：

```
Results for test scenario hub tests: 10 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 out eth0 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   The frame should arrive on eth0.
10  The frame should be flooded out eth1.


All tests passed!
```

Step 4: Run your device in Mininet

```
# 启动mininet
sudo python lab_1/start_mininet.py
```

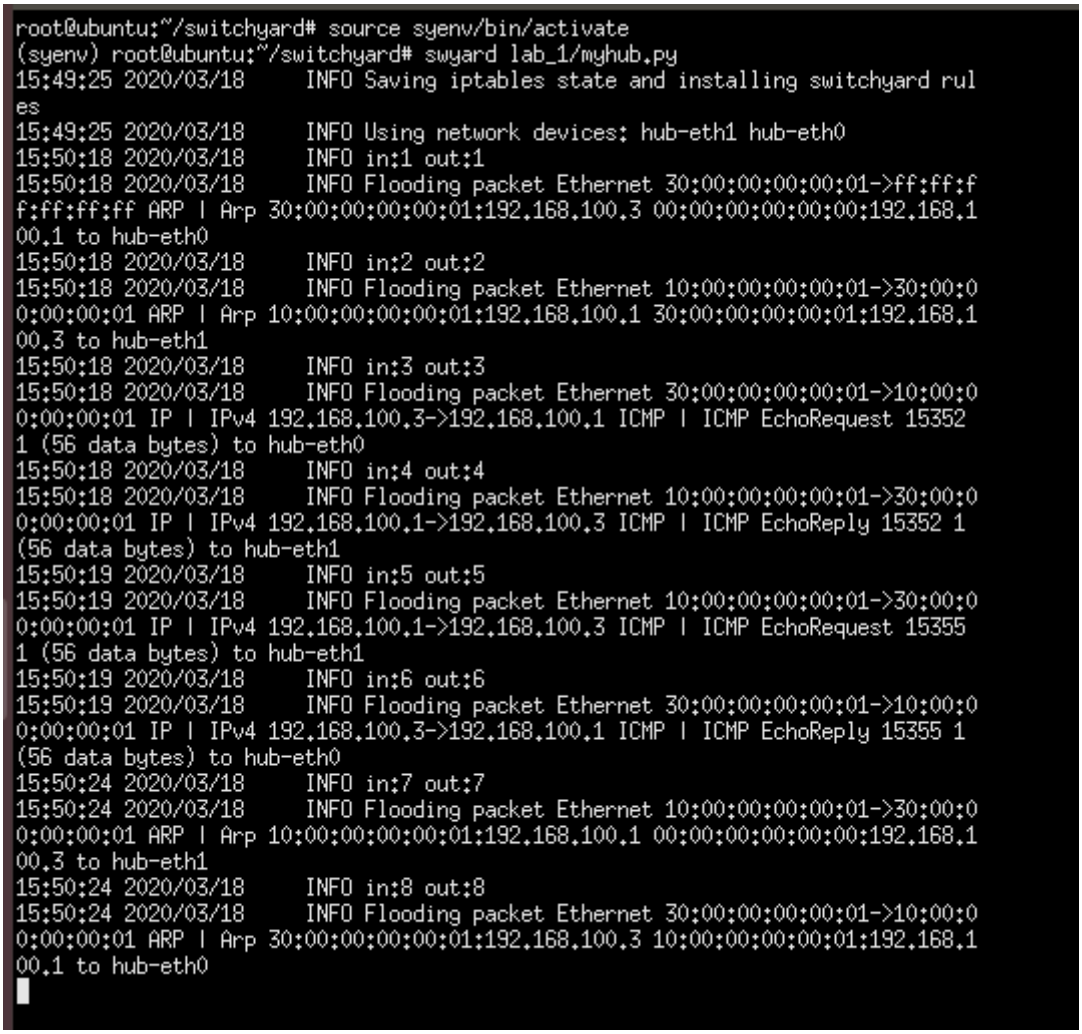
```
# 在hub上后台启动xterm
mininet> hub xterm &
```

```
# 在hub上启动python虚拟环境
source ./syenv/bin/activate
```

```
# 在hub上使用swyard命令加载myhub.py
swyard examples/myhub.py
```

```
# 回到mininet CLI，制造一些流量
mininet> pingall
```

效果：


root@ubuntu:~/switchyard# source syenv/bin/activate
(syenv) root@ubuntu:~/switchyard# swyard lab_1/myhub.py
15:49:25 2020/03/18 INFO Saving iptables state and installing switchyard rul
es
15:49:25 2020/03/18 INFO Using network devices: hub-eth1 hub-eth0
15:50:18 2020/03/18 INFO in:1 out:1
15:50:18 2020/03/18 INFO Flooding packet Ethernet 30:00:00:00:00:01->ff:ff:f
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-eth0
15:50:18 2020/03/18 INFO in:2 out:2
15:50:18 2020/03/18 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-eth1
15:50:18 2020/03/18 INFO in:3 out:3
15:50:18 2020/03/18 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 15352
1 (56 data bytes) to hub-eth0
15:50:18 2020/03/18 INFO in:4 out:4
15:50:18 2020/03/18 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 15352 1
(56 data bytes) to hub-eth1
15:50:19 2020/03/18 INFO in:5 out:5
15:50:19 2020/03/18 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 15355
1 (56 data bytes) to hub-eth1
15:50:19 2020/03/18 INFO in:6 out:6
15:50:19 2020/03/18 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 15355 1
(56 data bytes) to hub-eth0
15:50:24 2020/03/18 INFO in:7 out:7
15:50:24 2020/03/18 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-eth1
15:50:24 2020/03/18 INFO in:8 out:8
15:50:24 2020/03/18 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-eth0

```
*** Starting CLI:
mininet> hub xterm &
mininet> pingall
*** Ping: testing ping reachability
client -> X server1
hub -> X X
server1 -> client X
*** Results: 66% dropped (2/6 received)
mininet> exit
*** Stopping 0 controllers

*** Stopping 2 links
..
*** Stopping 0 switches

*** Stopping 3 hosts
client hub server1
*** Done
```

Step 5: Capture using Wireshark

使用Wireshark抓包并保存

```
mininet>
mininet> hub xterm &
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=859 ms

--- 192.168.100.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 859.378/859.378/859.378/0.000 ms
mininet>
```

共抓到6个包，其中1、2、5、6为ARP协议，3、4为ICMP协议

在Source和Destination中可以看出收发地址

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	0.341249278	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	0.443724925	192.168.100.3	192.168.100.1	ICMP	98	Echo (ping) request id=0x2c67, seq=1/256, ttl=64 (reply in 4)
4	0.758305197	192.168.100.1	192.168.100.3	ICMP	98	Echo (ping) reply id=0x2c67, seq=1/256, ttl=64 (request in 3)
5	6.102812398	Private_00:00:01	30:00:00:00:00:01	ARP	42	Who has 192.168.100.3? Tell 192.168.100.1
6	6.203263361	30:00:00:00:00:01	Private_00:00:01	ARP	42	192.168.100.3 is at 30:00:00:00:00:01

总结与感想

第一次网络实验，比预想的要困难不少。前期看文档的时候不觉得很tricky，实际上手才发现有些细节只有亲身经历才能体会得深刻。

希望（或许是个不情之请。。）之后的实验文档里，助教可以在必要时提供当前task、step所需要的理论知识。因为刚接触新知识的时候，可能难以和当前的具体实验情景相联系，有时候不知道要用什么知识点解决当前问题。适当点拨有助于加深对相应知识点的理解，从而加快实验进度。