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）	**计算机科学与技术**
学号	**201220102**	姓名	**武雅琛**
Email	**1379067209@qq.com**	开始/完成日期	**20220303~20220309**

1. 实验名称　Switchyard & Mininet & Wireshark

2. 实验目的

a) 了解一些计算机网络实验的准备工作。掌握一些辅助实验的工具，例如 python、Git、Mininet、Wireshark 等等。

b) 学习在 Mininet 构建的拓扑网络中使用 Switchyard 在虚拟环境中运行编写好的网络设备以及利用测试脚本在测试环境中检验编写好的网络设备。

c) 学习使用 Wireshark 捕获数据包来达到分析网络和检查设备的目的。

3. 实验内容

a) **Step 1: 修改 Mininet 网络拓扑**

在拓扑网络中删除 Server 2。

```
62          for node in nodes.keys():
63              self.addHost(node, **nodeconfig)
```

在 start_mininet.py 的 62~63 行代码中，调用 addHost 来向拓扑网络中加入客户端节点，并且节点信息来自 nodes 这个 dict。

```
25  nodes = {
26      "server1": {
27          "mac": "10:00:00:00:00:{:02x}",
28          "ip": "192.168.100.1/24"
29      },
30      "server2": {
31          "mac": "20:00:00:00:00:{:02x}",
32          "ip": "192.168.100.2/24"
33      },
34      "client": {
35          "mac": "30:00:00:00:00:{:02x}",
36          "ip": "192.168.100.3/24"
37      },
38      "hub": {
39          "mac": "40:00:00:00:00:{:02x}",
40      }
41  }
```

因此，我们只需要删除在 dict 中删除 server2 这个元素即可。

b) **Step 2:** 修改设备实现逻辑

统计进出集线器的数据包数目并在执行时显示日志。

```
26          eth = packet.get_header(Ethernet)
27          if eth is None:
28              log_info("Received a non-Ethernet packet?!")
29              return
30          if eth.dst in mymacs:
31              log_info("Received a packet intended for me")
32          else:
33              in_pack_cnt += 1
34              for intf in my_interfaces:
35                  if fromIface!= intf.name:
36                      log_info (f"Flooding packet {packet} to {intf.name}")
37                      net.send_packet(intf, packet)
38                      out_pack_cnt += 1
39              log info(f"in:{in pack cnt} out:{out pack cnt}")
```

可以分析得到，eth 是 hub 接受到的 packethead，如果为空或者目的 mac 地址为 hub 的端口地址则发生异常，输出对应的日志信息。如果收到一个有效数据包，增加一个接受计数，并且向输入端口以外所有的输出端口 flooding 数据包，并且每次发出一个数据包，增加一个转发计数。在结束一次完整的接受和转发后输出日志信息即可。

c) **Step 3: 修改测试场景**

用不同的参数创建一个新的 **packet** 用于测试。

在课程给的测试场景文件中描述了三种情形，接受一个不涉及目的 **mac** 地址的广播数据包并转发，接受一个有目的 **mac** 地址的数据包并转发，接受一个发向 **hub** 端口的数据包并且不响应。

```
#test case 4: modify:a frame with dest address of one of the interfaces should
#result nothing
reqpkt = new_packet(
    "20:00:00:00:00:01",
    "10:00:00:00:00:01",
    '192.168.1.100',
    '172.16.42.2'
)
s.expect(
    PacketInputEvent("eth0", reqpkt, display=Ethernet),
    ("An Ethernet frame should arrive on eth0 with destination address "
     "the same as eth0'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.")
)
```

第一部分构造了一个数据包，包头包括源端口的 **mac** 地址和 **IP** 地址，以及目的端口的 **mac** 地址（即 **hub** 的 **eth0** 的 **mac** 地址）和 **IP** 地址。首先要期望 **eth0** 端口可以接收到这个数据包，之后期望 **hub** 接受数据包解析后对于发向 **hub** 端口的数据包不作响应，正确处理异常。

d) **Step 4: 在 Mininet 中运行我的设备**

首先使用 **Mininet** 构建拓扑网络并且打开 **hub** 终端。激活虚拟环境，利用 **Switchyard** 运行设备。

```
root@njucs-VirtualBox:~/lab-01-Wumaomaomao# source ../switchyard/syenv/bin/acti
vate
(syenv) root@njucs-VirtualBox:~/lab-01-Wumaomaomao# swyard myhub.py
22:25:20 2022/03/09     INFO Saving iptables state and installing switchyard rul
es
22:25:20 2022/03/09     INFO Using network devices: hub-eth0 hub-eth1 hub-eth2
```

在主机终端上输入命令“**pingall**”,可以看出向 **hub** 端口发送数据包发生异常，其余情况正常到达终端。

```
*** Ping: testing ping reachability
client -> X server1 server2
hub -> X X X
server1 -> client X server2
server2 -> client X server1
*** Results: 50% dropped (6/12 received)
mininet>
```

并且，可以在 **hub** 终端上看到运行日志。

为了便于显示运行日志，在终端输入”**server1 ping -c 1 client**”来显示发送一个数据包产生的日志。

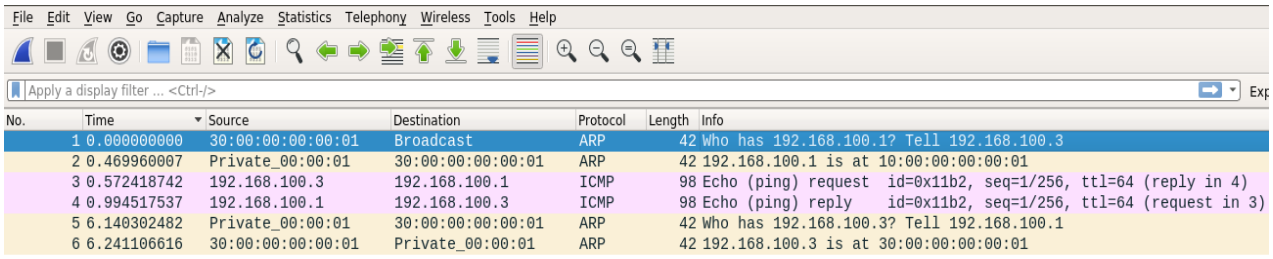
```
mininet> server1 ping -c 1 client
PING 192.168.100.3 (192.168.100.3) 56(84) bytes of data.
64 bytes from 192.168.100.3: icmp_seq=1 ttl=64 time=586 ms

--- 192.168.100.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 586.779/586.779/586.779/0.000 ms
```

```
22:30:01 2022/03/09     INFO Flooding packet Ethernet 10:00:00:00:00:01->30:00:00:00:00:01 IP | IPv4 192.168.100.1->192.168.100.3 ICMP | ICMP EchoRequest 3869 1 (56 data bytes) to hub-eth2
22:30:01 2022/03/09     INFO in:25 out:50
22:30:02 2022/03/09     INFO Flooding packet Ethernet 30:00:00:00:00:01->10:00:00:00:00:01 IP | IPv4 192.168.100.3->192.168.100.1 ICMP | ICMP EchoReply 3869 1 (56 data bytes) to hub-eth1
22:30:02 2022/03/09     INFO Flooding packet Ethernet 30:00:00:00:00:01->10:00:00:00:00:01 IP | IPv4 192.168.100.3->192.168.100.1 ICMP | ICMP EchoReply 3869 1 (56 data bytes) to hub-eth2
22:30:02 2022/03/09     INFO in:26 out:52
22:30:07 2022/03/09     INFO Flooding packet Ethernet 10:00:00:00:00:01->30:00:00:00:00:01 ARP | Arp 10:00:00:00:00:01:192.168.100.1 00:00:00:00:00:00:192.168.100.3 to hub-eth0
22:30:07 2022/03/09     INFO Flooding packet Ethernet 10:00:00:00:00:01->30:00:00:00:00:01 ARP | Arp 10:00:00:00:00:01:192.168.100.1 00:00:00:00:00:00:192.168.100.3 to hub-eth2
22:30:07 2022/03/09     INFO in:27 out:54
22:30:07 2022/03/09     INFO Flooding packet Ethernet 30:00:00:00:00:01->10:00:00:00:00:01 ARP | Arp 30:00:00:00:00:01:192.168.100.3 00:00:00:00:00:00:192.168.100.1 to hub-eth1
22:30:07 2022/03/09     INFO Flooding packet Ethernet 30:00:00:00:00:01->10:00:00:00:00:01 ARP | Arp 30:00:00:00:00:01:192.168.100.3 00:00:00:00:00:00:192.168.100.1 to hub-eth2
22:30:07 2022/03/09     INFO in:28 out:56
22:30:07 2022/03/09     INFO Flooding packet Ethernet 30:00:00:00:00:01->10:00:00:00:00:01 ARP | Arp 30:00:00:00:00:01:192.168.100.3 10:00:00:00:00:01:192.168.100.1 to hub-eth1
22:30:07 2022/03/09     INFO Flooding packet Ethernet 30:00:00:00:00:01->10:00:00:00:00:01 ARP | Arp 30:00:00:00:00:01:192.168.100.3 10:00:00:00:00:01:192.168.100.1 to hub-eth2
22:30:07 2022/03/09     INFO in:29 out:58
22:30:07 2022/03/09     INFO Flooding packet Ethernet 10:00:00:00:00:01->30:00:00:00:00:01 ARP | Arp 10:00:00:00:00:01:192.168.100.1 30:00:00:00:00:01:192.168.100.3 to hub-eth0
22:30:07 2022/03/09     INFO Flooding packet Ethernet 10:00:00:00:00:01->30:00:00:00:00:01 ARP | Arp 10:00:00:00:00:01:192.168.100.1 30:00:00:00:00:01:192.168.100.3 to hub-eth2
22:30:07 2022/03/09     INFO in:30 out:60
```

e) **Step 5:** 在 **Wireshark** 中捕获数据包并分析

在主机命令行输入“**client ping -c server1**” 得到结果。



在观察 hub-eth0 端口，捕获了以上六个数据包，根据数据包行为分为三组：

I. 依据 ARP 协议。Client 只知道 server1 的 IP 地址但是不知道它的 mac 地址。所以通过广播询问 IP 地址 192.168.100.1 的 mac 地址。经过 hub 转发后，server1 返回了一个包含了 mac 地址的数据包。

II. 依据 ICMP 协议。因为我们使用了基于 ICMP 协议的工具 ping 来验证从 client 发出数据包是否可以到达 server1。

III. 依据 ARP 协议。这是 Linux 操作系统的确认机制。server1 发出一个数据包验证 client 的 mac 地址是否正确。

4. 实验结果

Step 1: 可以看到，修改后 Mininet 创建的拓扑网络已经不包括 server2。

```
*** 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
('client', <TCIntf client-eth0>, '30:00:00:00:00:01')
('server1', <TCIntf server1-eth0>, '10:00:00:00:00:01')
('hub', <TCIntf hub-eth0>, '40:00:00:00:00:01')
('hub', <TCIntf 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
```

Step 2: 在上一部分已经展示了修改后输出日志的结果。

Step 3：

```
23:45:25 2022/03/09      INFO in:3 out:6
23:45:26 2022/03/09      INFO Received a packet intended for me

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 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 should arrive on eth0 with destination
    address the same as eth0's MAC address
10  The hub should not do anything in response to a frame
    arriving with a destination address referring to the hub
    itself.


All tests passed!
```

Step 5: 上一部分已提及。

5. 核心代码

a) **Step 1:**

```
nodes = {
    "server1": {
        "mac": "10:00:00:00:00:{:02x}",
        "ip": "192.168.100.1/24"
    },
    #"server2": {
        #"mac": "20:00:00:00:00:{:02x}",
        #"ip": "192.168.100.2/24"
    #},
    "client": {
        "mac": "30:00:00:00:00:{:02x}",
        "ip": "192.168.100.3/24"
    },
    "hub": {
        "mac": "40:00:00:00:00:{:02x}",
    }
}
```

b) **Step 2:**

```
26          eth = packet.get_header(Ethernet)
27          if eth is None:
28              log_info("Received a non-Ethernet packet?!")
29              return
30          if eth.dst in mymacs:
31              log_info("Received a packet intended for me")
32          else:
33              in_pack_cnt += 1
34              for intf in my_interfaces:
35                  if fromIface!= intf.name:
36                      log_info (f"Flooding packet {packet} to {intf.name}")
37                      net.send_packet(intf, packet)
38                      out_pack_cnt += 1
39              log info(f"in:{in pack cnt} out:{out pack cnt}")
```

c) **Step 3:**

```
#test case 4: modify:a frame with dest address of one of the interfaces should
#result nothing
reqpkt = new_packet(
    "20:00:00:00:00:01",
    "10:00:00:00:00:01",
    '192.168.1.100',
    '172.16.42.2'
)
s.expect(
    PacketInputEvent("eth0", reqpkt, display=Ethernet),
    ("An Ethernet frame should arrive on eth0 with destination address "
     "the same as eth0'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.")
)
```

6. 总结与感想

a) 实验手册确实写得非常贴心，回头看几乎顾及到了每一个点，如果遇到困难大多数情况下是看的时候忽略了细节。但是一周阅读这么多资料而且还是英文版的确实是痛苦的。

b) 总之还是感谢，属于倒逼我学了很多东西。