

南京大学本科生实验报告

课程名称： 计算机网络

任课教师： 

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		姓名	
Email		开始/完成日期	

南京大学本科生实验报告

- 1.实验名称
- Lab 1: Switchyard & Mininet
- 2.实验目的
- 3.实验内容
- 4.实验结果
- STEP 1:Modify the Mininet topology
- STEP 2:Modify the logic of a device
- STEP 3:Modify the test scenario of a device
- STEP 4:Run your device in Mininet
- STEP 5:Capture using Wireshark
- 5.核心代码
- 6.总结与感想

1.实验名称

Lab 1: Switchyard & Mininet

2.实验目的

此次实验主要介绍了整个计算机网络实验重所需要的所有的准备工作，并且能够通过学习指导手册来根据要求来对老师提供的一些例子进行一定程度的修改。

3.实验内容

- Preparations for the lab
- Play the Tutorial Again
 - Step 1: Modify the Mininet topology
 - Step 2: Modify the logic of a device
 - Step 3: Modify the test scenario of a device
 - Step 4: Run your device in Mininet
 - Step 5: Capture using Wireshark

4.实验结果

注：为了方便，此部分同时也展开如何进行实验的，以及对于实验中的为何需要这么做的一定的解释。

STEP 1:Modify the Mininet topology

由于我们需要做的事情是delete `server2` in the topology，加上之前的指导手册的提及，由此可知，此项任务的关键是start_mininet.py程序，着眼于main函数：

```
1 def main():
2     topo = PySwitchTopo(args)
3     net = Mininet(controller=None, topo=topo, link=TCLink, cleanup=True)
4     setup_addressing(net)
5     disable_ipv6(net)
6     net.interact()
```

由此可知，我们需要修改的是topo这一变量，但是topo又是类PySwitchTopo的一个实例，这样就追溯到类PySwitchTopo的类定义了，其类定义如下：

```
1 class PySwitchTopo(Topo):
2     def __init__(self, args):
3         # Add default members to class.
4         super(PySwitchTopo, self).__init__()
5         # Host and link configuration
6         #
7         #
8         # server1
9         #         \
10        #             hub----client
11        #             /
12        # server2
13        #
14        nodeconfig = {"cpu": -1}
15        for node in nodes.keys():
16            self.addHost(node, **nodeconfig)
17        for node in nodes.keys():
18            # all links are 10Mb/s, 100 millisecond prop delay
19            if node != "hub":
20                self.addLink(node, "hub", bw=10, delay="100ms")
```

由上述代码可知，类PySwitchTopo是继承于Topo类的，根据实验指导手册上面提供的 [Mininet Walkthrough](#)，我找到了Python API中关于[Topo类的定义](#)。上面代码所使用的api的解释如下：

```
1 class Topo( object ):
2     "Data center network representation for structured multi-trees."
3
4     #.....
5
6     def addHost( self, name, **opts ):
7         """Convenience method: Add host to graph.
8         name: host name
9         opts: host options
10        returns: host name"""
11        if not opts and self.hopts:
12            opts = self.hopts
```

```

13         return self.addNode( name, **opts )
14
15     def addLink( self, node1, node2, port1=None, port2=None,
16                 key=None, **opts ):
17         """node1, node2: nodes to link together
18            port1, port2: ports (optional)
19            opts: link options (optional)
20            returns: link info key"""
21         if not opts and self.lopts:
22             opts = self.lopts
23         port1, port2 = self.addPort( node1, node2, port1, port2 )
24         opts = dict( opts )
25         opts.update( node1=node1, node2=node2, port1=port1, port2=port2 )
26         return self.g.add_edge(node1, node2, key, opts )
27     # ....

```

经过此分析可知，按我们代码中所使用：**addHost**方法就是将**nodes**变量中所有的**host**加入到整张**graph**中；**addLink**方法就是将**nodes**变量中所有除了**hub**的**node**全部与**hub**链接起来。

这样看来，如果要在整个**topology**中将**server2**删除的话，就可以将**nodes**变量中的**server2**节点删除即可满足条件。具体的操作如下：

```

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}",
    }
}

```

这样改动就完成了，利用**python**启动**start_mininet.py**来观察是否删除**server2**成功：

```

(syenv) njucs@njucs-VirtualBox: $ sudo python 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
('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

*** Starting 0 switches

*** Starting CLI:
mininet> nodes
available nodes are:
client hub server1
mininet>

```

由**nodes**命令展示可知，**server2**被成功删除，结果正确。

STEP 2: Modify the logic of a device

正如指导手册中写的这样

In the section [Switchyard](#), we introduced how to program a device. Your task is to count how many packets pass through a hub in and out. You need to log the statistical result every time you receive one packet with the format of each line `in:<ingress packet count> out:<egress packet count>`. For example, if there is a packet that is not addressed to the hub itself, then the hub may log `in:1 out:2`. ✓ Then show the log of your hub when running it in Mininet and how you implement it in your report.

当hub收到一个packet时，且这个packet不是发送给hub的时候，就会通过两个口转发出去，结合下图可视化这种过程

```
1 server1
2      \
3      hub----client
4      /
5 server2
```

所以就会有 `in:1 out:2` 这种结果的诞生。

现在回到代码的修改部分，因为这里主要涉及的是hub的内容的增加，所以应该涉及的代码改动就是 `myhub.py`了。

利用助教哥哥提供的方法实现了Ctrl+点击能实现方法的跳转后，就可以比较方便的对代码的理解了。

由于我们必须count how many packets pass through a hub in and out.所以就着眼于比较相似于接受packet和发送packet的方法；观察代码可知，net的recv_packet(...)和send_packet(...)的这两个方法都值得重视，跳转到方法的定义后，定义如下：

```
1 class LLNetBase(metaclass=ABCMeta):
2     '''
3     Base class for the low-level networking library in Python.
4     "net" objects are constructed from classes derived from this
5     class.
6     '''
7     def __init__(self, name=None):
8         self._devupdown_callback = None
9         self._devinfo = {} # dict(str -> Interface)
10    # .....
11    @abstractmethod
12    def recv_packet(self, timeout=None):
13        '''
14        Receive a packet on any port/interface.
15        If a non-None timeout is given, the method will block for up
16        to timeout seconds. If no packet is available, the exception
17        NoPackets will be raised. If the Switchyard framework is being
18        shut down, the Shutdown exception will be raised.
19        If a packet is available, the ReceivedPacket named tuple
20        (timestamp, input_port, packet) will be returned.
21        '''
22        raise NoPackets()
23
24    @abstractmethod
25    def send_packet(self, output_port, packet):
26        '''
```

```

27         Send a packet out the given output port/interface.
28         Returns None.
29         '''
30         pass

```

由这些方法的注释可知：

1. `recv_packet`方法就是Receive a packet on any port/interface
2. `send_packet`方法就是Send a packet out the given output port/interface.

所以就可以分别在其后加上对应变量的自增，然后在最后输出**log**即可完成对应的任务，具体的代码改动如下：

```

def main(net: switchyard.llnetbase.LLNetBase):
    my_interfaces = net.interfaces()
    mymacs = [intf.ethaddr for intf in my_interfaces]
    num_in, num_out = 0, 0
    while True:
        try:
            _, fromIface, packet = net.recv_packet()
            num_in = num_in + 1

            except NoPackets:
                continue
            except Shutdown:
                break

            log_debug (f"In {net.name} received packet {packet} on {fromIface}")
            eth = packet.get_header(Ethernet)
            if eth is None:
                log_info("Received a non-Ethernet packet?!")
                return
            if eth.dst in mymacs:
                log_info("Received a packet intended for me")
            else:
                for intf in my_interfaces:
                    if fromIface != intf.name:
                        log_info (f"Flooding packet {packet} to {intf.name}")
                        net.send_packet(intf, packet)
                        num_out = num_out + 1
            log_info (f"in:{num_in} out:{num_out}")
    net.shutdown()

```

然后是对此结果进行一定的验证：

- ①首先搭建的网络打开

```

1 | sudo python start_mininet.py

```

- ②输入**xterm hub**，然后在其中输入**swyard myhub.py**运行程序代码，得到的结果如下：

```
(syenv) njucs@njucs-VirtualBox: ~$ sudo python 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)
*** 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

*** 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> []
```

```
"Node: hub"
root@njucs-VirtualBox:~/networkLab/ # source ~/switchyard/syenv/bin/activate
(syenv) root@njucs-VirtualBox:~/networkLab/ # swyard nhub.py
21:36:08 2021/03/13 INFO Saving iptables state and installing switchyard rules
21:36:08 2021/03/13 INFO Using network devices: hub-eth0 hub-eth1
21:36:13 2021/03/13 INFO Flooding packet Ethernet 30:00:00:00:00:01->ff:ff:ff:ff:ff:ff ARP | Arp 30:00:00:00:00:00
1:192.168.100.3 00:00:00:00:00:00:192.168.100.1 to hub-eth1
21:36:13 2021/03/13 INFO in1 out:1
21:36:13 2021/03/13 INFO Flooding packet Ethernet 10:00:00:00:00:01->30:00:00:00:00:01 ARP | Arp 10:00:00:00:00:00
1:192.168.100.1 30:00:00:00:00:00:192.168.100.3 to hub-eth0
21:36:13 2021/03/13 INFO in2 out:2
21:36:13 2021/03/13 INFO Flooding packet Ethernet 30:00:00:00:00:01->10:00:00:00:00:01 IP | IPv4 192.168.100.3->1
92.168.100.1 ICMP | ICMP EchoRequest 5851 1 (56 data bytes) to hub-eth1
21:36:13 2021/03/13 INFO in3 out:3
21:36:13 2021/03/13 INFO Flooding packet Ethernet 10:00:00:00:00:01->30:00:00:00:00:01 IP | IPv4 192.168.100.1->1
92.168.100.3 ICMP | ICMP EchoReply 5851 1 (56 data bytes) to hub-eth0
21:36:13 2021/03/13 INFO in4 out:4
21:36:14 2021/03/13 INFO Flooding packet Ethernet 10:00:00:00:00:01->30:00:00:00:00:01 IP | IPv4 192.168.100.1->1
92.168.100.3 ICMP | ICMP EchoRequest 5855 1 (56 data bytes) to hub-eth0
21:36:14 2021/03/13 INFO in5 out:5
21:36:14 2021/03/13 INFO Flooding packet Ethernet 30:00:00:00:00:01->10:00:00:00:00:01 IP | IPv4 192.168.100.3->1
92.168.100.1 ICMP | ICMP EchoReply 5855 1 (56 data bytes) to hub-eth1
21:36:14 2021/03/13 INFO in6 out:6
21:36:18 2021/03/13 INFO Flooding packet Ethernet 10:00:00:00:00:01->30:00:00:00:00:01 ARP | Arp 10:00:00:00:00:00
1:192.168.100.1 00:00:00:00:00:00:192.168.100.3 to hub-eth0
21:36:18 2021/03/13 INFO in7 out:7
21:36:19 2021/03/13 INFO Flooding packet Ethernet 30:00:00:00:00:01->10:00:00:00:00:01 ARP | Arp 30:00:00:00:00:00
1:192.168.100.3 10:00:00:00:00:00:192.168.100.1 to hub-eth1
21:36:19 2021/03/13 INFO in8 out:8
```

由右图可以看到能打印出了log，但是in和out是一样的，这是因为在STEP 1中我们已经将server2的这一个节点删除了，所以就有此现象的发生。

现接着采用testcases/myhub_testscenario.py对其进行测试：

```
(syenv) njucs@njucs-VirtualBox: ~$ swyard -t testcases/myhub_testscenario.py myhub.py
21:42:23 2021/03/13 INFO Starting test scenario testcases/myhub_testscenario.py
21:42:23 2021/03/13 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 e
th0
21:42:23 2021/03/13 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 e
th2
21:42:23 2021/03/13 INFO in:1 out:2
21:42:23 2021/03/13 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 eth
1
21:42:23 2021/03/13 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 eth
2
21:42:23 2021/03/13 INFO in:2 out:4
21:42:23 2021/03/13 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
21:42:23 2021/03/13 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
21:42:23 2021/03/13 INFO Received a packet intended for me
21:42:23 2021/03/13 INFO in:4 out:6
21:42:24 2021/03/13 INFO Flooding packet Ethernet 30:00:00:00:00:01->ff:ff:ff:ff:ff:ff IP | IPv4 192.168.1.100->255.255.255.255 ICMP | ICMP EchoRequest 0 0 (0 data bytes) to
eth1
21:42:24 2021/03/13 INFO Flooding packet Ethernet 30:00:00:00:00:01->ff:ff:ff:ff:ff:ff IP | IPv4 192.168.1.100->255.255.255.255 ICMP | ICMP EchoRequest 0 0 (0 data bytes) to
eth2
21:42:24 2021/03/13 INFO in:5 out:8
Results for test scenario hub tests: 10 passed, 0 failed, 0 pending
```

由于在testcases/myhub_testscenario.py中有专门发送给hub的packet，这样in和out就有了一定的大小差距，也进一步说明了log的正确性。

STEP 3:Modify the test scenario of a device

这一STEP就是需要创造自己的testcase，我选择的是Create one test case by using the given function new_packet with different arguments，也就是通过传入不同的参数给写好的new_packet函数来创造一个新的testcase

首先先进行对于代码中给出的三个testcase进行一定的分析

1. testcase 1:包从30:00:00:00:00:02 被广播之后, eth1 模拟包到达, 然后模拟包从其他两个端口发出。
2. testcase 2:除了指定给hub接口的地址外, 任何单播地址的帧都应该被发送到除入口之外的所有端口
3. testcase 3:因为这是一个发送给hub自己本身的包, 所以应该无事发生

以下就是我新创造的testcase，这是仿照testcase1所设计的。

```
1 mytestpkt = new_packet(
2     "30:00:00:00:00:01",
3     "ff:ff:ff:ff:ff:ff",
4     "192.168.1.100",
5     "255.255.255.255"
```

```

6   )
7   s.expect(
8       PacketInputEvent("eth0", mytestpkt, display=Ethernet),
9       ("An Ethernet frame with a broadcast destination address "
10        "should arrive on eth0")
11   )
12   s.expect(
13       PacketOutputEvent("eth1", mytestpkt, "eth2", mytestpkt,
14        display=Ethernet),
15       ("The Ethernet frame with a broadcast destination address should be "
16        "forwarded out ports eth1 and eth2")
17   )

```

具体的含义就是：

包从30:00:00:00:00:01 被广播之后, eth0 模拟包到达, 模拟包从其他两个端口发出。

产生的结果如下面红色框内的内容：

```

*** Done
(syenv) njucs@njucs-VirtualBox: $ swyard -t testcases/myhub_testscenario.py myhub.py
21:42:23 2021/03/13 INFO Starting test scenario testcases/myhub_testscenario.py
21:42:23 2021/03/13 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
21:42:23 2021/03/13 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
21:42:23 2021/03/13 INFO in:1 out:2
21:42:23 2021/03/13 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
21:42:23 2021/03/13 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
21:42:23 2021/03/13 INFO in:2 out:4
21:42:23 2021/03/13 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
21:42:23 2021/03/13 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
21:42:23 2021/03/13 INFO in:3 out:6
21:42:23 2021/03/13 INFO Received a packet intended for me
21:42:23 2021/03/13 INFO in:4 out:6
21:42:24 2021/03/13 INFO Flooding packet Ethernet 30:00:00:00:00:01->ff:ff:ff:ff:ff:ff IP IPv4 192.168.1.100->255.255.255.255 ICMP | ICMP EchoRequest 0 0 (0 data bytes) to eth1
21:42:24 2021/03/13 INFO Flooding packet Ethernet 30:00:00:00:00:01->ff:ff:ff:ff:ff:ff IP IPv4 192.168.1.100->255.255.255.255 ICMP | ICMP EchoRequest 0 0 (0 data bytes) to eth2
21:42:24 2021/03/13 INFO in:5 out:8

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

```

STEP 4:Run your device in Mininet

①首先搭建的网络打开：

```
1 | sudo python start_mininet.py
```

②输入**xterm hub**，然后在其中输入**swyard myhub.py**运行程序代码，然后**CIL**中输入**pingall**得到的结果如下：

```

(syenv) njucs@njucs-VirtualBox: $ sudo python 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)
*** 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

*** 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>

```

这样就完成了在mininet上面成功地run my device。

STEP 5: Capture using Wireshark

此STEP就可以开始实现抓取数据包了，根据手册中的 [Wireshark](#) 和 [Switchyard](#)，就可以得知该如何获取数据包，同样的在CLI输入对应指令

```
1 | server1 wireshark &
```

来打开wireshark并选择对应的server1-eth0，进行数据包的抓取：

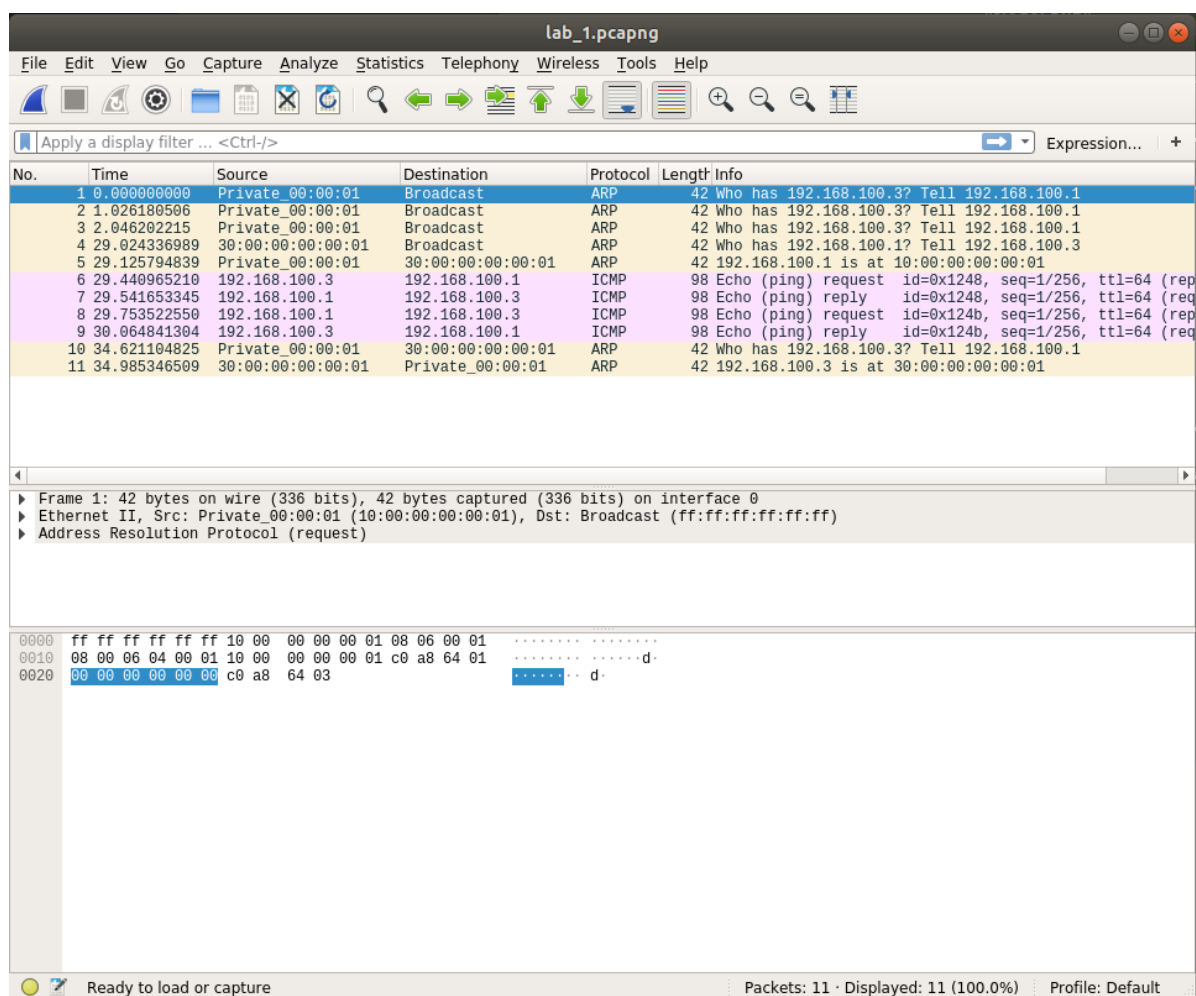
①首先在mininet中输入pingall，发现是100% dropped

②然后在"Node :hub"中输入swyard myhub.py

就能够成功地创造流量了。

```
start_mininet.py x myhub.py myhub_testscenario.py
start_mininet.py ...
16 import argparse
17 import os
18
19 parser = argparse.ArgumentParser(description="Mininet pyswitch topology")
20 # no arguments needed as yet :- )
21 args = parser.parse_args()
22 lg.setLevel('info')
23
24
25 nodes = {}
26
27 "server1": {
28     "mac": "10:00:00:00:00:{:02x}"
29     "ip": "192.168.100.1/24"
30 },
31 # "server2": {
32 #     "mac": "20:00:00:00:00:{:02x}"
33 #     "ip": "192.168.100.2/24"
34 # },
35 "client": {
36     "mac": "30:00:00:00:00:{:02x}"
37     "ip": "192.168.100.3/24"
38 },
39 "hub": {
40     "mac": "40:00:00:00:00:{:02x}"
41 }
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551

```

①第一部分：

帧**Frame 1** 指的是要发送的数据块，其中，所抓帧的序号为**0**，捕获字节数等于传送字节数也即**42bytes**。接下来还有许多关于这一部分的细节参数：

1. **Encapsulation type**：Ethernet (1)，这代表的是一个**Wireshark-internal**值，表示所讨论的包的特定链接层报头类型。
2. **Arrival time**：表示的是数据包的**到达的时间**
3. **Epoch time**：表示的是接受的时间自**1970年1月1日**以来的秒数。
4. **Time delta**：由于这是第一个接受到的包，所以均为 **0 seconds**
5. **Frame**：这里展示的是**Frame**的部分属性，包括**Frame Number**，**Frame Length**，协议等信息。

lab_1.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/> Expression...

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Private_00:00:01	Broadcast	ARP	42	Who has 192.168.100.3? Tell 192.168.100.1
2	1.026180506	Private_00:00:01	Broadcast	ARP	42	Who has 192.168.100.3? Tell 192.168.100.1
3	2.046202215	Private_00:00:01	Broadcast	ARP	42	Who has 192.168.100.3? Tell 192.168.100.1
4	29.024336989	30:00:00:00:00:01	Broadcast	ARP	42	Who has 192.168.100.1? Tell 192.168.100.3
5	29.125794839	Private_00:00:01	30:00:00:00:00:01	ARP	42	192.168.100.1 is at 10:00:00:00:00:01
6	29.440965210	192.168.100.3	192.168.100.1	ICMP	98	Echo (ping) request id=0x1248, seq=1/256, ttl=64
7	29.541653345	192.168.100.1	192.168.100.3	ICMP	98	Echo (ping) reply id=0x1248, seq=1/256, ttl=64
8	29.753522550	192.168.100.1	192.168.100.3	ICMP	98	Echo (ping) request id=0x124b, seq=1/256, ttl=64
9	30.064841304	192.168.100.3	192.168.100.1	ICMP	98	Echo (ping) reply id=0x124b, seq=1/256, ttl=64
10	34.621104825	Private_00:00:01	30:00:00:00:00:01	ARP	42	Who has 192.168.100.3? Tell 192.168.100.1
11	34.085346500	30:00:00:00:00:01	Private_00:00:01	ARP	42	192.168.100.3 is at 30:00:00:00:00:01

Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0

- Interface id: 0 (server1-eth0)
 - Encapsulation type: Ethernet (1)
 - Arrival Time: Mar 13, 2021 19:02:55.209601343 CST
 - [Time shift for this packet: 0.000000000 seconds]
 - Epoch Time: 1615633375.209601343 seconds
 - [Time delta from previous captured frame: 0.000000000 seconds]
 - [Time delta from previous displayed frame: 0.000000000 seconds]
 - [Time since reference or first frame: 0.000000000 seconds]
 - Frame Number: 1
 - Frame Length: 42 bytes (336 bits)
 - Capture Length: 42 bytes (336 bits)
 - [Frame is marked: False]
 - [Frame is ignored: False]
 - [Protocols in frame: eth:ethertype:arp]
 - [Coloring Rule Name: ARP]
 - [Coloring Rule String: arp]
 - Ethernet II, Src: Private_00:00:01 (10:00:00:00:00:01), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
 - Address Resolution Protocol (request)

```

0000  ff ff ff ff ff ff 10 00 00 00 00 01 08 06 00 01  .....d.
0010  08 00 06 04 00 01 10 00 00 00 00 01 c0 a8 64 01  .....d.
0020  00 00 00 00 00 00 c0 a8 64 03  .....d.

```

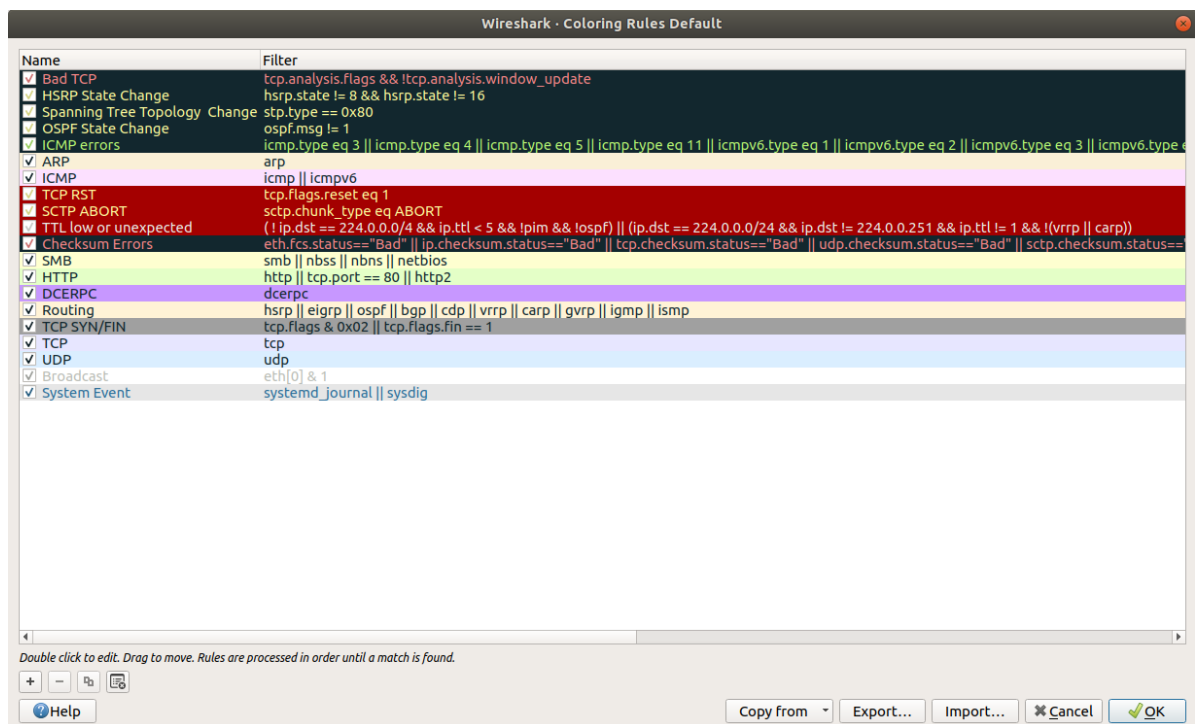
Ready to load or capture Packets: 11 · Displayed: 11 (100.0%) Profile: Default

②第二部分:

Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0

- Ethernet II, Src: Private_00:00:01 (10:00:00:00:00:01), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
 - Destination: Broadcast (ff:ff:ff:ff:ff:ff)
 - Source: Private_00:00:01 (10:00:00:00:00:01)
 - Type: ARP (0x0806)
- Address Resolution Protocol (request)
 - Hardware type: Ethernet (1)
 - Protocol type: IPv4 (0x0800)
 - Hardware size: 6
 - Protocol size: 4
 - Opcode: request (1)
 - Sender MAC address: Private_00:00:01 (10:00:00:00:00:01)
 - Sender IP address: 192.168.100.1
 - Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
 - Target IP address: 192.168.100.3

(注：每种不同颜色的流量包含含义如下：)



接下来来看看其他类型的数据包：

③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10:00:00:00:00:01；

type是0x0800表明了此数据包是IP数据包

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Private_00:00:01	Broadcast	ARP	42	Who has 192.168.100.3? Tell 192.168.100.1
2	1.026180506	Private_00:00:01	Broadcast	ARP	42	Who has 192.168.100.3? Tell 192.168.100.1
3	2.046202215	Private_00:00:01	Broadcast	ARP	42	Who has 192.168.100.3? Tell 192.168.100.1
4	29.024336989	30:00:00:00:00:01	Broadcast	ARP	42	Who has 192.168.100.1? Tell 192.168.100.3
5	29.125794639	Private_00:00:01	30:00:00:00:00:01	ARP	42	192.168.100.1 is at 10:00:00:00:00:01
6	29.440965210	192.168.100.3	192.168.100.1	ICMP	98	Echo (ping) request id=0x1248, seq=1/256, ttl=64 (reply in 7)
7	29.541653345	192.168.100.1	192.168.100.3	ICMP	98	Echo (ping) reply id=0x1248, seq=1/256, ttl=64 (request in 6)
8	29.753522550	192.168.100.1	192.168.100.3	ICMP	98	Echo (ping) request id=0x124b, seq=1/256, ttl=64 (reply in 9)
9	30.064841304	192.168.100.3	192.168.100.1	ICMP	98	Echo (ping) reply id=0x124b, seq=1/256, ttl=64 (request in 8)
10	34.621104825	Private_00:00:01	30:00:00:00:00:01	ARP	42	Who has 192.168.100.3? Tell 192.168.100.1
11	34.985346509	30:00:00:00:00:01	Private_00:00:01	ARP	42	192.168.100.3 is at 30:00:00:00:00:01

▶ Frame 9: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0

▶ Ethernet II, Src: 30:00:00:00:00:01 (30:00:00:00:00:01), Dst: Private_00:00:01 (10:00:00:00:00:01)

▶ Destination: Private_00:00:01 (10:00:00:00:00:01)

▶ Source: 30:00:00:00:00:01 (30:00:00:00:00:01)

▶ Type: IPv4 (0x0800)

▶ Internet Protocol Version 4, Src: 192.168.100.3, Dst: 192.168.100.1

▶ Internet Control Message Protocol

④IPV4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。

Time to live (TTL)为64表明包经过64路由器没到达会被丢弃。

Protocol: ICMP说明采用的协议是ICMP。

Headchecksum：头部校验和0x608e。

⑤ICMP协议

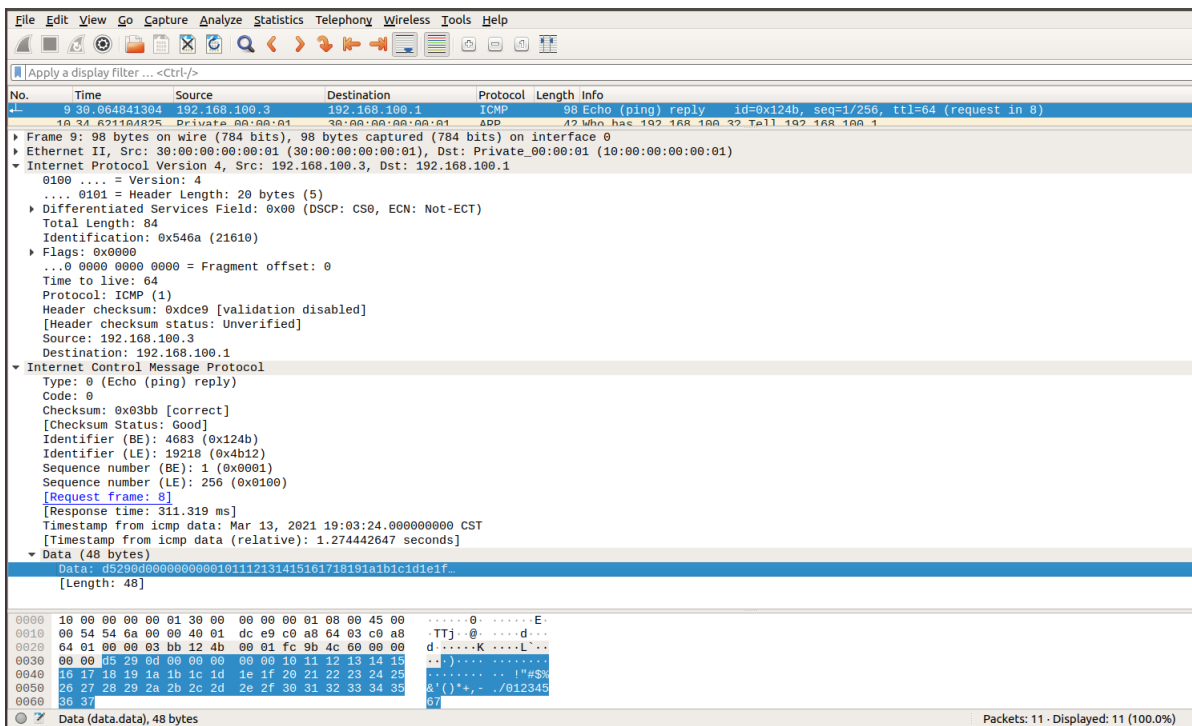
Type是0对应了ping reply；

checksum校验和的状态是correct正确的；

Identifier后面的括号里，BE和LE代表大端和小端，根据生成标识符和序列号的操作系统，来更容易地检查丢失的序列。

以及最后的时间戳Timestamp。

在包的最后我们也可以看到包中所包含的data，足有48 bytes。



5.核心代码

具体的所需要改动的实验核心代码在上一部分**实验结果**中全部提及了，在此部分就不再过多赘述了。

6.总结与感想

总的来说，对于第一次的计算机网络实验，都是一些对于各种软件和api的熟练下进行一定的准备工作，包括但不限于对于**swyard**，**mininet**等工具的熟悉。但是还是有些不是很懂的东西，毕竟还有许多没有教学到的内容，后面我会一点一点将这些内容全部弄明白的。

感想：因为首次面对这种全英文的指导手册，而且各种帮助文档也都是英文的，所以读起来是非常费劲的其实。但是需要通过时间的积累来减少阅读英文的时间，况且以后也都是像类似于这样的英文的帮助文档的，所以需要长时间的锻炼的。