Computer Network* Lab 1

- **徐思源 191220133**
- Department of Computer Science and Technology
- Nanjing University
- 1357307497@qq.com

Step1

- **选择完成任务一：Delete `server2` in the topology**
  - **首先，理解start_mininet.py文件中的代码含义**
    - **开头定义了参数（暂时没有有用的参数）和log的等级，然后定义了一个类PySwitchTopo即我们定义的拓扑网络，之后是一系列实现不同功能的函数，step1中暂时不需要用到**
    - **接下来定义了一系列的nodes，包括host，hub，client等**
    - **在PySwitchTopo初始化中，先按node中的配置新增host，然后对非hub节点和hub建立link**
  - **根据指导文件中的提示：**
    - We build a topology inside the constructor function `__init__` of class `PySwitchTopo`. If you want to change the topology, you should modify it. We setup interfaces in the function `setup_addressing`, you will learn what happens here after knowing Ethernet and IP protocol.
    - **说明需要从拓扑网络只需要对类中的初始化函数进行修改，因为初始化函数都是遍历node中的变量进行初始化，所以直接删除nodes中的server2即可。**
    -

```
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}",
    }
}
```

Step2

- **这里我们需要知道发出的数据包和接收的数据包的数量并且输出**
    - **先对代码进行理解：**

- **根据教程中的内容，我们可以了解到recv函数和send函数的作用就是收发数据包，而针对recv有timeout和shutdown两种未接收到数据包的特殊情况，对于send有non-Ethernet packet 和packet intended for me两种不需要send的数据包，所以我们统计总数的时候，只需要在recv函数下面加上in++的代码，在send函数下加上out++的代码，并且用log输出即可**
  - `recv_packet(timeout=None)`

    Not surprisingly, this method is used to receive at most one packet from any port. The method will *block* until a packet is received, unless a timeout value >=0 is given. The default is to block indefinitely. The method returns a *namedtuple* of length 3, which includes a timestamp for when the packet was received, the name of the input port on which the packet was received, and the packet itself (another example is given below, plus see **`collections.namedtuple`** in the Python library reference).

    The method raises a `Shutdown` exception if the Switchyard framework has been shut down. It can also raise a `NoPackets` exception if no packets are received before the timeout value given to the method expires.

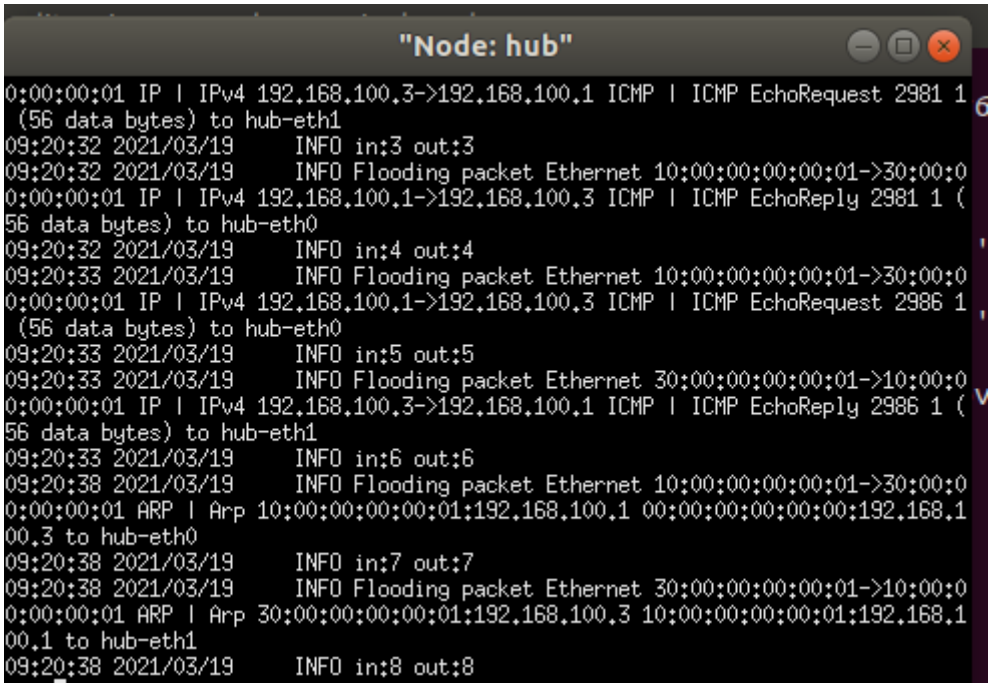
  - `send_packet(output_port, packet)`

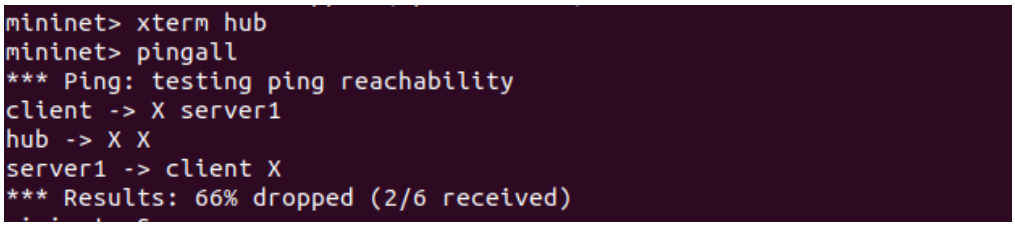
    Again, the meaning of this method call is probably not especially surprising: when called, the given packet will be sent out the given output port. For the `output_port` parameter, the string name of the port can be given, or an `Interface` object may also be supplied (see below for more about Interface objects as well as the Interface and InterfaceType reference).

    This method returns `None`. If the `output_port` or some detail about the given packet is invalid (e.g., something other than a packet is passed as the second parameter), this method raises a `ValueError`.

- ```
  while True:
      try:
          _, fromIface, packet = net.recv_packet()
          in_data=in_data+1
      except NoPackets:
          continue
      except Shutdown:
          break
  ```
- ```
    if fromIface!= intf.name:
        log_info (f"Flooding packet {packet} to {intf.name}")
        net.send_packet(intf, packet)
        out_data=out_data+1
  ```

- **输出结果如下：**

- 

- **注意pingall的时候，出现如下的情况：**
  - 

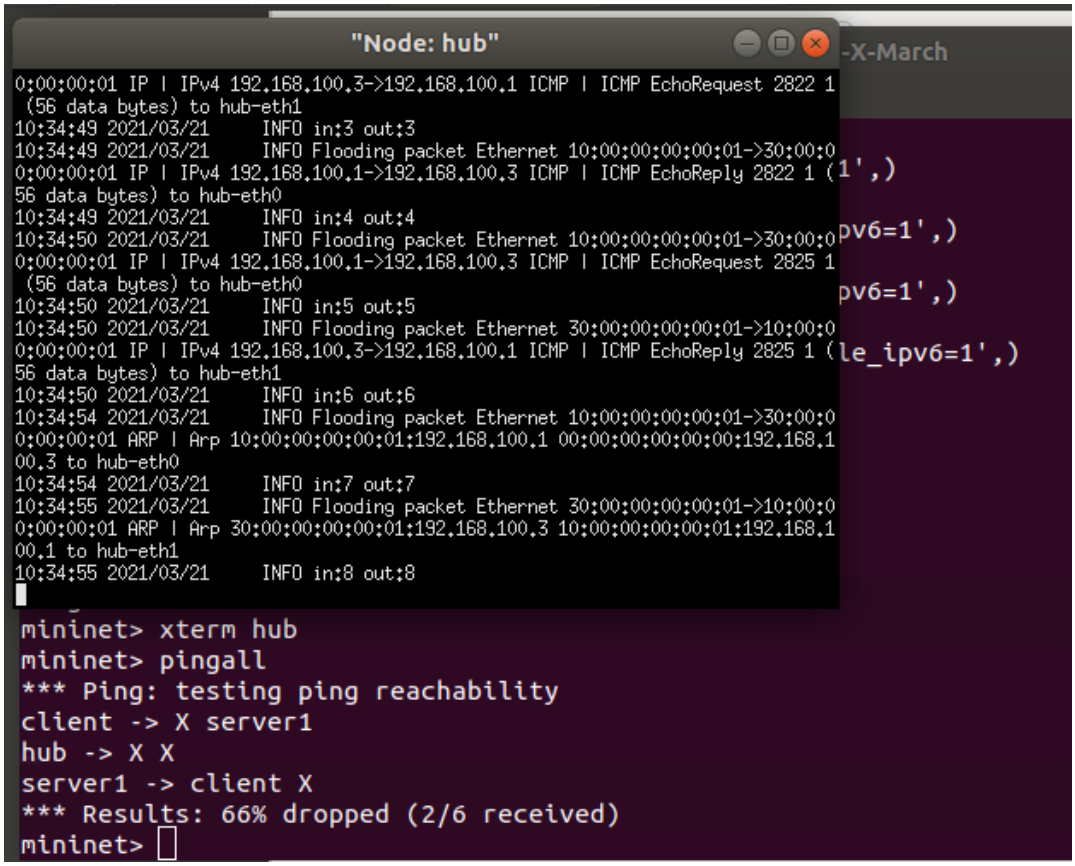
  - **由于step1中我们选择删除了server2，所以这里只有client，hub，server1三个node，其中每个node都尝试ping了其他两个node，但是只有client和server1ping通了，查阅资料可以知道hub在实际情况下其实是一个物理层设备，并不属于主机等节点类，不具有对应的ip地址，实验中我们只是为了方便起见把他当成和client以及server一样的节点，而我们如果没有打开hub则都不能ping通**
    - 端口上。从Hub的工作方式可以看出，它在网络中只起到信号放大和重发作用，其目的是扩大网络的传输范围，而不具备信号的定向传送能力，是一个标准的共享式设备。因此有
  - **所以出现上述情况的原因是hub在实际情况中其实并不参与ping的过程，所以只有client和server1连通了。**

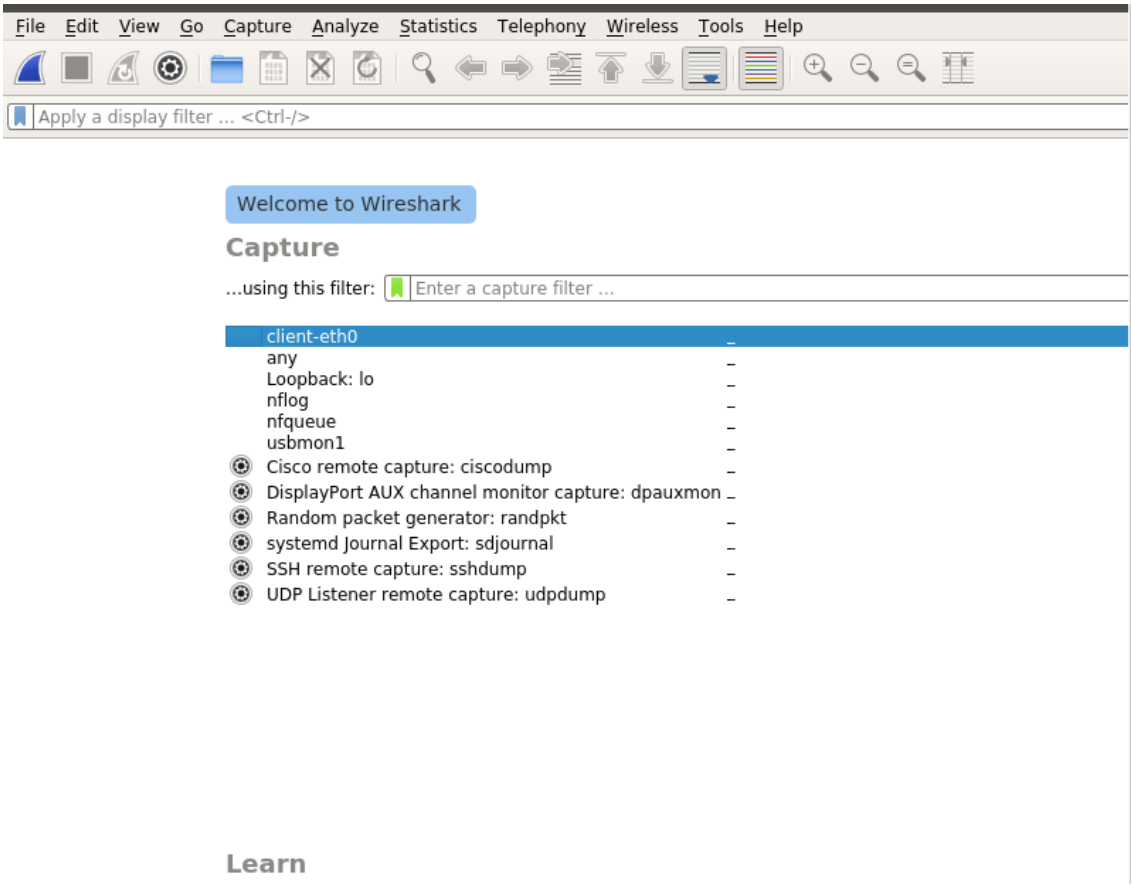
step3

- **首先根据switchyard文件中的说明，知道testcase有三种类型，如下**
  - that a particular packet should arrive on a particular interface/port,
    that a particular packet should be emitted out one or more ports, and
    that the user program should *time out* when calling `recv_packet` because no packets are available.

- **所给的testcase文件中已经提供了这三种样例的一个例子，我们根据所给的文件分析testcase的设置方式**
  - **首先，add_interface函数用于添加和配置接口，参数信息如下：**
    - device, you will need to add *at least* two interfaces. Arguments to the `add_interface` method are used to specify the interface's name (e.g., `en0`), its hardware Ethernet (MAC) address, and its (optional) IP address and netmask.
  - **expect函数用于添加预期事件并且打印，下面对所给文件中的几个testcase进行说明**
    - ```
      # test case 1: a frame with broadcast destination should get sent out
      # all ports except ingress
      testpkt = new_packet(
          "30:00:00:00:00:02",
          "ff:ff:ff:ff:ff:ff",
          "172.16.42.2",
          "255.255.255.255"
      )
      s.expect(
          PacketInputEvent("eth1", testpkt, display=Ethernet),
          ("An Ethernet frame with a broadcast destination address "
           "should arrive on eth1")
      )
      s.expect(
          PacketOutputEvent("eth0", testpkt, "eth2", testpkt, display=Ethernet),
          ("The Ethernet frame with a broadcast destination address should be "
           "forwarded out ports eth0 and eth2")
      )
      ```
    - client广播一个数据包被接口eth1接收，然后将从两个端口eth0,eth2发出
    - ```
      # test case 2: a frame with any unicast address except one assigned to hub
      # interface should be sent out all ports except ingress
      reqpkt = new_packet(
          "20:00:00:00:00:01",
          "30:00:00:00:00:02",
          '192.168.1.100',
          '172.16.42.2'
      )
      s.expect(
          PacketInputEvent("eth0", reqpkt, display=Ethernet),
          ("An Ethernet frame from 20:00:00:00:00:01 to 30:00:00:00:00:02 "
           "should arrive on eth0")
      )
      s.expect(
          PacketOutputEvent("eth1", reqpkt, "eth2", reqpkt, display=Ethernet),
          ("Ethernet frame destined for 30:00:00:00:00:02 should be flooded out"
           " eth1 and eth2")
      )
      ```
    - server1向client发送数据包，数据包到达接口eth0，并被从eth1和eth2两个端口发出
    - ```
      # test case 3: a frame with dest address of one of the interfaces should
      # result in nothing happening
      reqpkt = new_packet(
          "20:00:00:00:00:01",
          "10:00:00:00:00:03",
          '192.168.1.100',
          '172.16.42.2'
      )
      s.expect(
          PacketInputEvent("eth2", reqpkt, display=Ethernet),
          ("An Ethernet frame should arrive on eth2 with destination address "
           "the same as eth2'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.")
      )s
      ```
    - server1向client发送一个数据包特定的接口eth2，而hub不需要做出转发
    - 自己添加构造一个第三类型的testcase，如下
      ```

```
reqpkt = new_packet(
    "20:00:00:00:00:01",
    "10:00:00:00:00:02",
    '192.168.1.100',
    '172.16.42.2'
)
s.expect(
    PacketInputEvent("eth1", reqpkt, display=Ethernet),
    ("An Ethernet frame should arrive on eth1 with destination address "
     "the same as eth1'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.")
)
```

- 运行结果如下

```
Results for test scenario hub tests: 10 passed, 0 failed, 0 pending

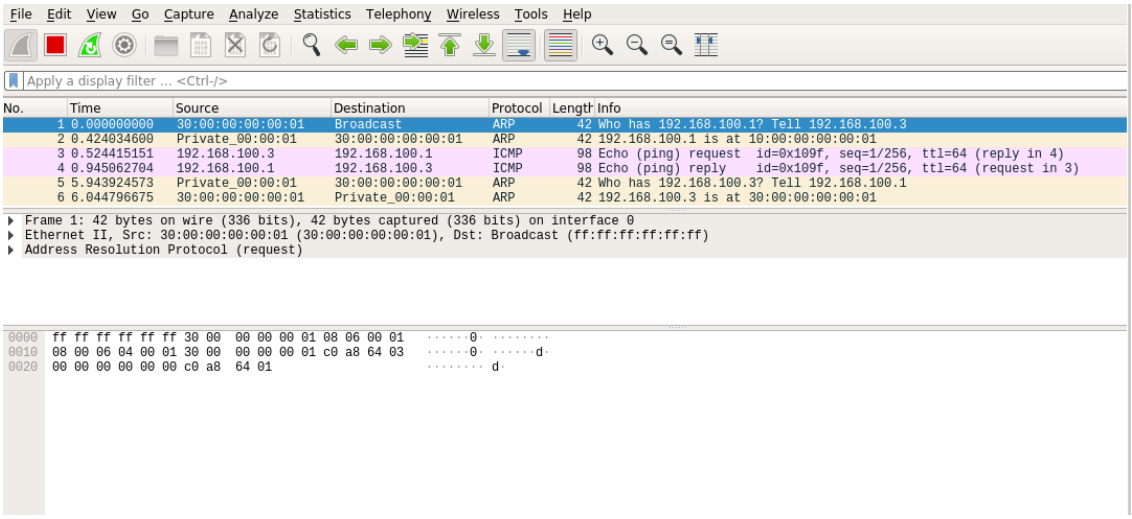

Passed:
1   An Ethernet frame with a broadcast destination address
    should arrive on eth1
2   The Ethernet frame with a broadcast destination address
    should be forwarded out ports eth0 and eth2
3   An Ethernet frame from 20:00:00:00:00:01 to
    30:00:00:00:00:02 should arrive on eth0
4   Ethernet 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 eth1 with destination
    address the same as eth1'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!
```

Step4

- 只需要按照步骤进行运行即可

- 

  S

Step5

- **开启wireshark开始抓包**
- ```
  mininet> client wireshark &
  ```
- **选择网卡**
-
  ```



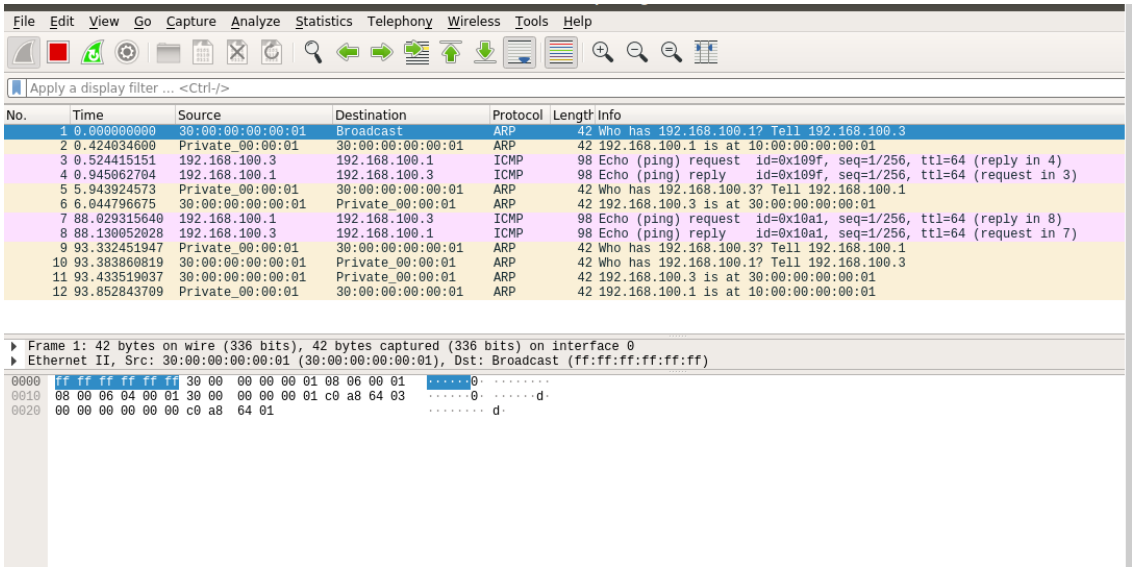
- **构造流量，**
-

```
mininet> client ping -c 1 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=1045 ms

--- 192.168.100.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1045.837/1045.837/1045.837/0.000 ms
```

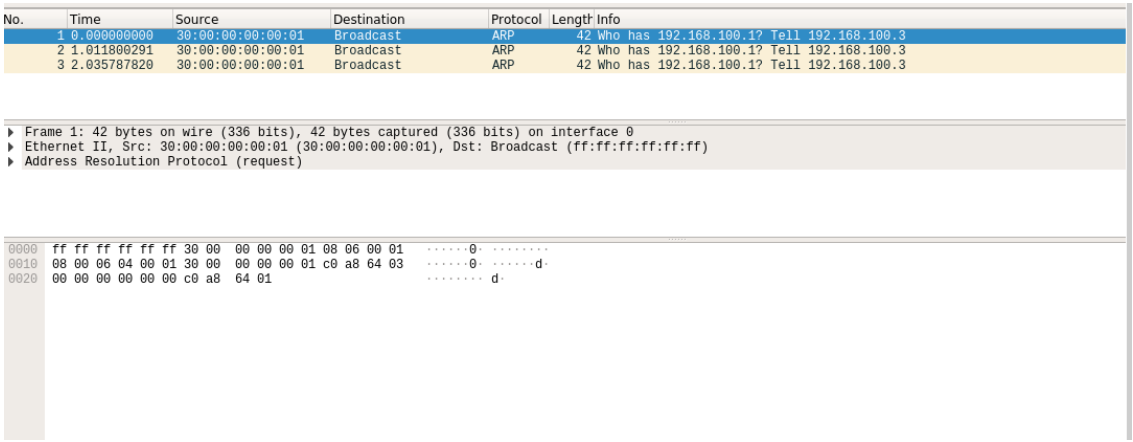
- **抓包结果如下**
- 

- **添加srvere1对client尝试ping**
-

```
mininet> server1 ping -c 1 client
PING 192.168.100.3 (192.168.100.3) 56(84) bytes of data.
From 192.168.100.1 icmp_seq=1 Destination Host Unreachable

--- 192.168.100.3 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms
```

- 

- **对抓包结果进行分析**
  - 根据结果我们可以看到，当client尝试ping通server1的时候，首先要询问server1的mac地址，然后接收到回复mac地址为10:00:00:00:00:01，之后client对server1发出ping的request，server1接收到信息后进行reply，完成ping，server1对client进行ping通的时候也是一样的过程
- **问题分析，一开始进行抓包的时候忘记了打开hub，之后client对server1进行ping的时候，抓包情况如下：**

- 

- **可以看到，hub未打开的时候client发出询问server1mac地址并未得到回复，ping没有成功，通过这个情况我们可以更加深刻的了解到hub集线器的作用，也就是起到将信号扩大和转发的目的，hub未打开的时候，host之间是不可以相互ping通的**