SDN实验---Mininet实验1（玩转流表）

一：实验目的

（一）案例目的

案例目的：

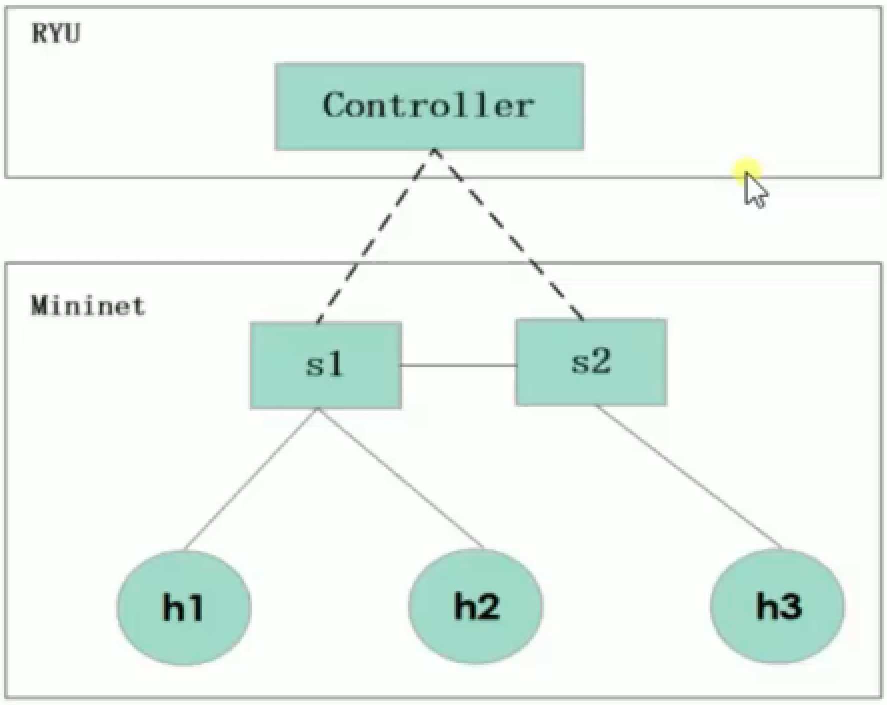
掌握Open vSwitch下发流表操作；

掌握添加、删除流表命令以及设备通信的原理。

（二）实验内容

Mininet创建一个默认树形拓扑并指定Mininet的控制器，进行基本的添加、删除流表操作，使网络实现网络通信和不通信。

（三）网络拓扑结构



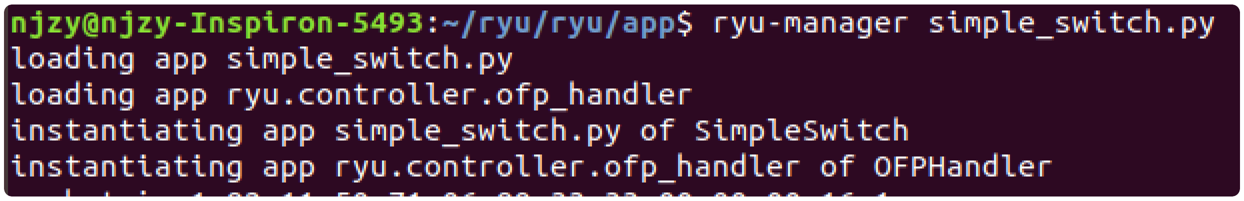
二：OpenFlow流表实验准备

（一）使用Python设置网络拓扑 --- tree_topo.py

```python
from mininet.topo import Topo
from mininet.net import Mininet
from mininet.node import RemoteController
from mininet.link import TCLink
from mininet.util import dumpNodeConnections

class MyTopo(Topo):

    def __init__(self):
        super(MyTopo,self).__init__()

        # add host
        Host1 = self.addHost('h1')
        Host2 = self.addHost('h2')
        Host3 = self.addHost('h3')

        switch1 = self.addSwitch('s1')
        switch2 = self.addSwitch('s2')

        self.addLink(Host1,switch1)
        self.addLink(Host2,switch1)
        self.addLink(Host3,switch2)
        self.addLink(switch1,switch2)

topos = {"mytopo":(lambda:MyTopo())}
```

（二）启动远程Ryu控制器

ryu–manager simple_switch.py　　注意，该控制器py文件在app目录下



```
njzy@njzy-Inspiron-5493:~/ryu/ryu/app$ ryu-manager simple_switch.py
loading app simple_switch.py
loading app ryu.controller.ofp_handler
instantiating app simple_switch.py of SimpleSwitch
instantiating app ryu.controller.ofp_handler of OFPHandler
```

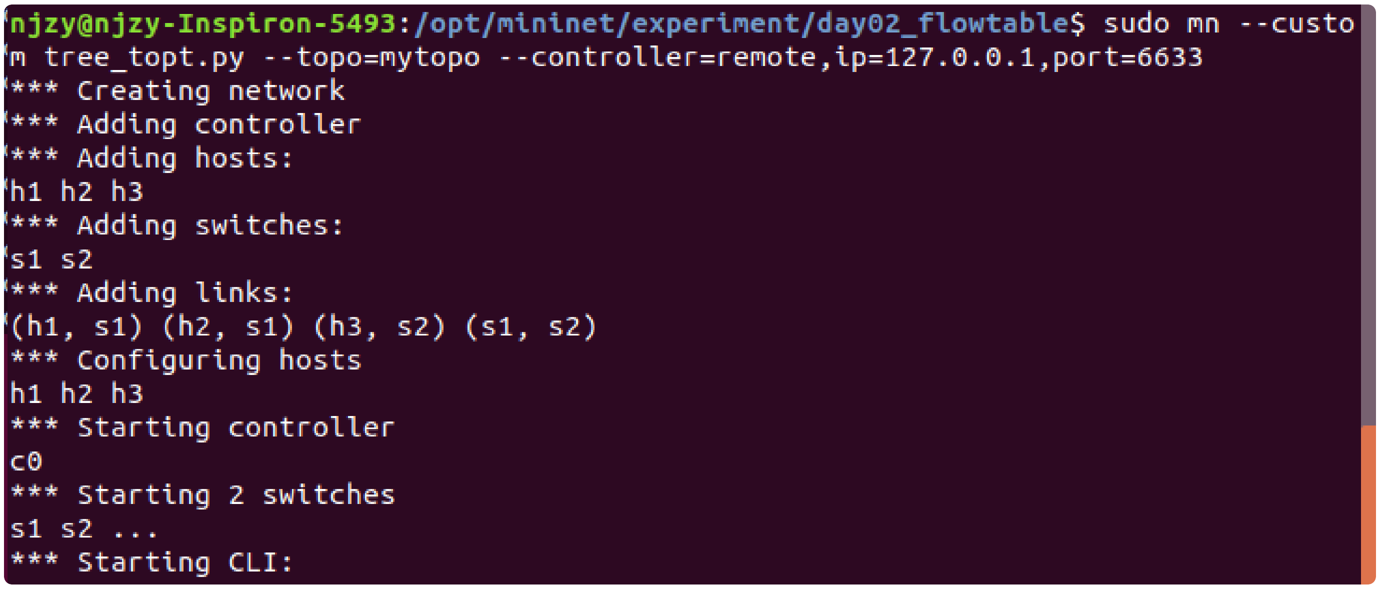
Python 复制代码

```python
# Copyright (C) 2011 Nippon Telegraph and Telephone Corporation.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#    http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
# implied.
# See the License for the specific language governing permissions and
# limitations under the License.

"""
An OpenFlow 1.0 L2 learning switch implementation. 1
"""


from ryu.base import app_manager
from ryu.controller import ofp_event
from ryu.controller.handler import MAIN_DISPATCHER
from ryu.controller.handler import set_ev_cls
from ryu.ofproto import ofproto_v1_0
from ryu.lib.mac import haddr_to_bin
from ryu.lib.packet import packet
from ryu.lib.packet import ethernet
from ryu.lib.packet import ether_types


class SimpleSwitch(app_manager.RyuApp):    不同与之前的Ryu实验，这里面没有在交换机初始连接时下发默认流表...待思考
    OFP_VERSIONS = [ofproto_v1_0.OFP_VERSION]

    def __init__(self, *args, **kwargs):
        super(SimpleSwitch, self).__init__(*args, **kwargs)
        self.mac_to_port = {}

    def add_flow(self, datapath, in_port, dst, src, actions):    下发流表
        ofproto = datapath.ofproto

        match = datapath.ofproto_parser.OFPMatch(
            in_port=in_port,
            dl_dst=haddr_to_bin(dst), dl_src=haddr_to_bin(src))

        mod = datapath.ofproto_parser.OFPFlowMod(
            datapath=datapath, match=match, cookie=0,
            command=ofproto.OFPFC_ADD, idle_timeout=0, hard_timeout=0,
            priority=ofproto.OFP_DEFAULT_PRIORITY,
            flags=ofproto.OFPFF_SEND_FLOW_REM, actions=actions)
        datapath.send_msg(mod)

    @set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
    def _packet_in_handler(self, ev):    交换机向控制器发送数据
        msg = ev.msg
        datapath = msg.datapath
        ofproto = datapath.ofproto

        pkt = packet.Packet(msg.data)
        eth = pkt.get_protocol(ethernet.ethernet)

        if eth.ethertype == ether_types.ETH_TYPE_LLDP:
            # ignore lldp packet
            return
        dst = eth.dst
        src = eth.src

        dpid = datapath.id
        self.mac_to_port.setdefault(dpid, {})

        self.logger.info("packet in %s %s %s %s", dpid, src, dst, msg.in_port)

        # learn a mac address to avoid FLOOD next time.
        self.mac_to_port[dpid][src] = msg.in_port

        if dst in self.mac_to_port[dpid]:
            out_port = self.mac_to_port[dpid][dst]
        else:
            out_port = ofproto.OFPP_FLOOD

        actions = [datapath.ofproto_parser.OFPActionOutput(out_port)]

        # install a flow to avoid packet_in next time
        if out_port != ofproto.OFPP_FLOOD:
            self.add_flow(datapath, msg.in_port, dst, src, actions)

        data = None
        if msg.buffer_id == ofproto.OFP_NO_BUFFER:
            data = msg.data

        out = datapath.ofproto_parser.OFPPacketOut(
                datapath=datapath, buffer_id=msg.buffer_id, in_port=msg.in_port,
                actions=actions, data=data)
        datapath.send_msg(out)

    @set_ev_cls(ofp_event.EventOFPPortStatus, MAIN_DISPATCHER)
    def _port_status_handler(self, ev):
        msg = ev.msg
        reason = msg.reason
        port_no = msg.desc.port_no

        ofproto = msg.datapath.ofproto
        if reason == ofproto.OFPPR_ADD:
            self.logger.info("port added %s", port_no)
        elif reason == ofproto.OFPPR_DELETE:
            self.logger.info("port deleted %s", port_no)
        elif reason == ofproto.OFPPR_MODIFY:
            self.logger.info("port modified %s", port_no)
        else:
            self.logger.info("Illeagal port state %s %s", port_no, reason)
```

（三）Mininet开始启动网络拓扑

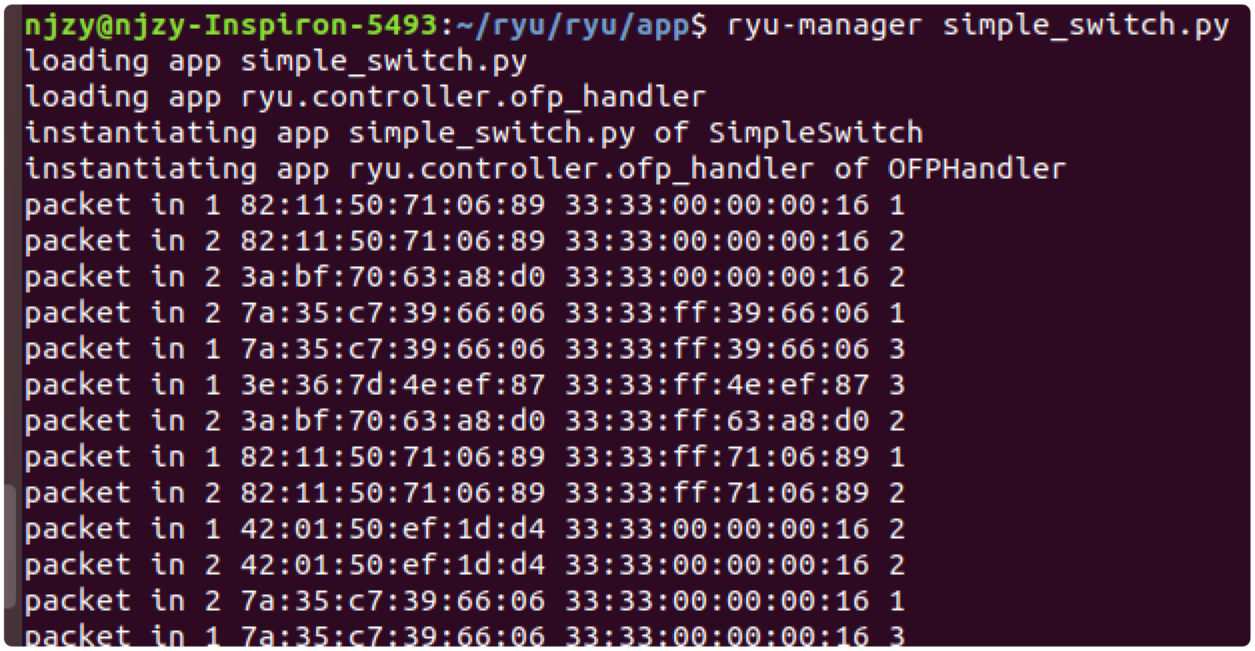
Shell 复制代码

```
sudo mn --custom tree_topt.py --topo=mytopo --controller=remote,ip=127.0.0.1,port=6633
```

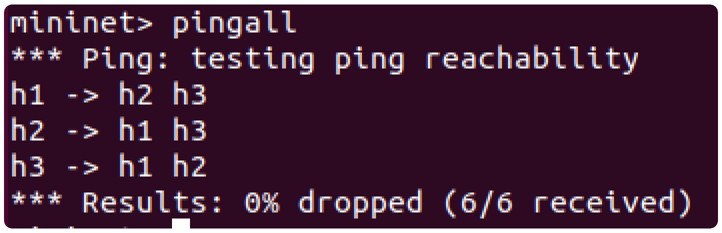


```
njzy@njzy-Inspiron-5493:/opt/mininet/experiment/day02_flowtable$ sudo mn --custom tree_topt.py --topo=mytopo --controller=remote,ip=127.0.0.1,port=6633
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1 s2
*** Adding links:
(h1, s1) (h2, s1) (h3, s2) (s1, s2)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...
*** Starting CLI:
```

注意：应该是主机连接发送了数据，导致控制器对网络进行了拓扑收集，问题同上：SDN实验---Ryu的应用开发（二）Learning Switch



```
njzy@njzy-Inspiron-5493:~/ryu/ryu/app$ ryu-manager simple_switch.py
loading app simple_switch.py
loading app ryu.controller.ofp_handler
instantiating app simple_switch.py of SimpleSwitch
instantiating app ryu.controller.ofp_handler of OFPHandler
packet in 1 82:11:50:71:06:89 33:33:00:00:00:16 1
packet in 2 82:11:50:71:06:89 33:33:00:00:00:16 2
packet in 2 3a:bf:70:63:a8:d0 33:33:00:00:00:16 2
packet in 2 7a:35:c7:39:66:06 33:33:ff:39:66:06 1
packet in 1 7a:35:c7:39:66:06 33:33:ff:39:66:06 3
packet in 1 3e:36:7d:4e:ef:87 33:33:ff:4e:ef:87 3
packet in 2 3a:bf:70:63:a8:d0 33:33:ff:63:a8:d0 2
packet in 1 82:11:50:71:06:89 33:33:ff:71:06:89 1
packet in 2 82:11:50:71:06:89 33:33:ff:71:06:89 2
packet in 1 42:01:50:ef:1d:d4 33:33:00:00:00:16 2
packet in 2 42:01:50:ef:1d:d4 33:33:00:00:00:16 2
packet in 2 7a:35:c7:39:66:06 33:33:00:00:00:16 1
packet in 1 7a:35:c7:39:66:06 33:33:00:00:00:16 3
```

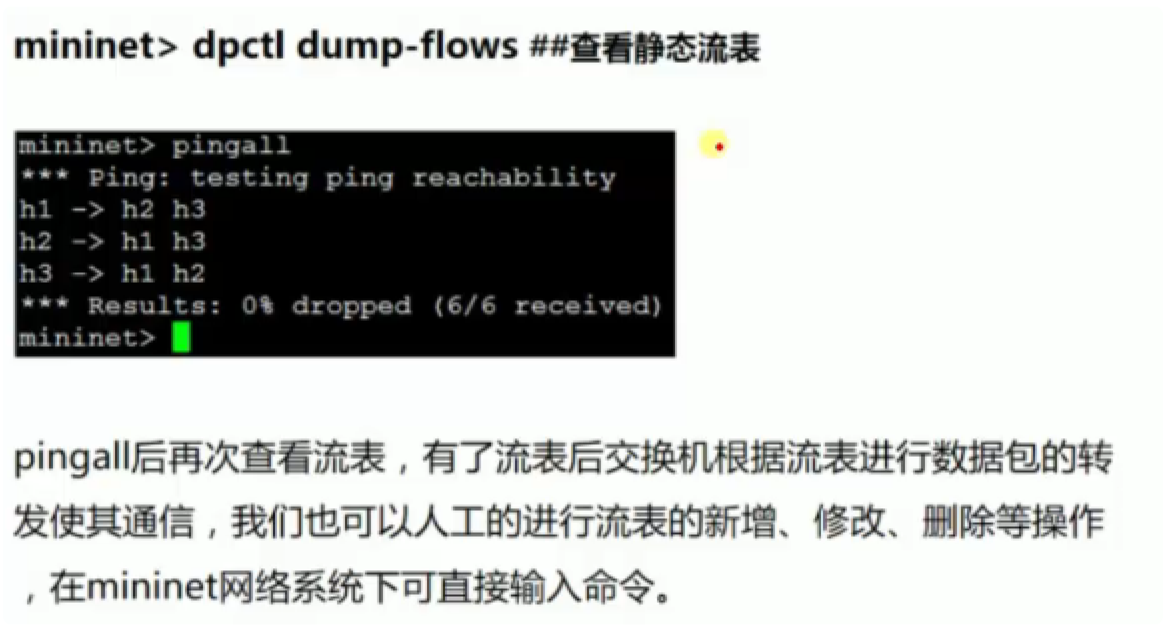


```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
```

三：进行OpenFlow流表分析

（一）主要流表操作命令

dpctl dump-flows　　查看静态流表

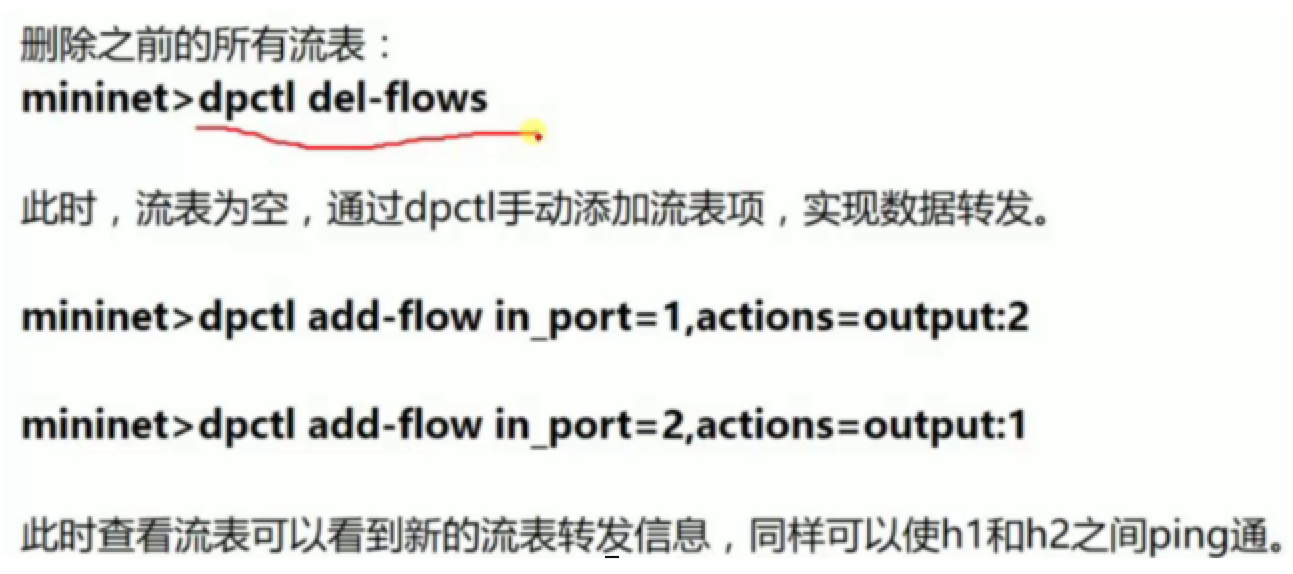


mininet> dpctl dump-flows ##查看静态流表

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet> 
```

pingall后再次查看流表，有了流表后交换机根据流表进行数据包的转发使其通信，我们也可以人工的进行流表的新增、修改、删除等操作，在mininet网络系统下可直接输入命令。

```shell
dpctl del-flows     删除所有交换机中的流表
dpctl add-flow in_port=1,actions=output:2    添加流表项到所有交换机，注意：一般是成对添加，实现双方通信
```



删除之前的所有流表：

mininet>dpctl del-flows

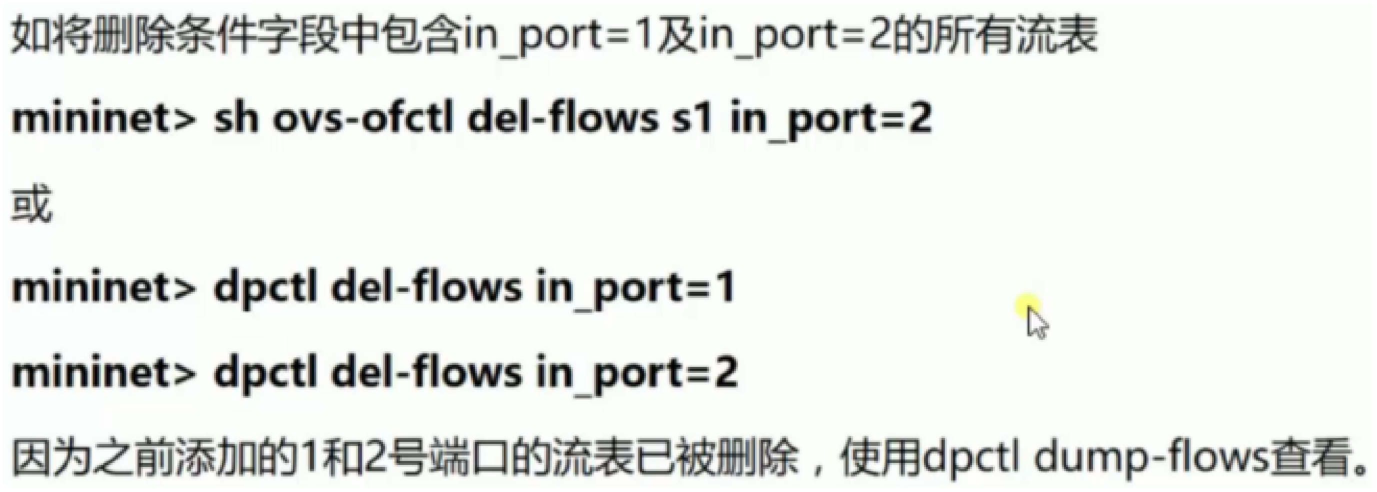
此时，流表为空，通过dpctl手动添加流表项，实现数据转发。

mininet>dpctl add-flow in_port=1,actions=output:2

mininet>dpctl add-flow in_port=2,actions=output:1

此时查看流表可以看到新的流表转发信息，同样可以使h1和h2之间ping通。

```shell
sh ovs-ofctl del-flows s1 in_port=2    删除指定交换机的，匹配in_port=2的流表
dpctl del-flows in_port=1    删除所有交换机中符合in_port=1的流表
```



如将删除条件字段中包含in_port=1及in_port=2的所有流表

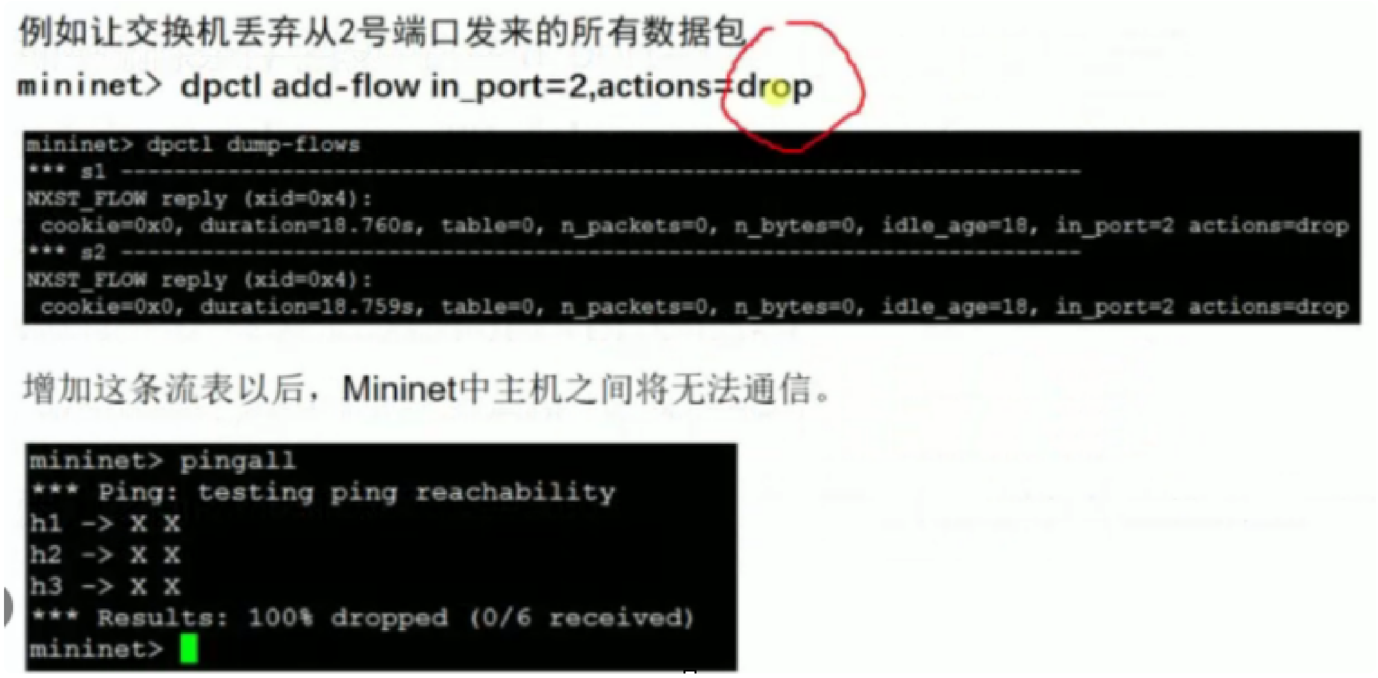
```
mininet> sh ovs-ofctl del-flows s1 in_port=2
```

或

```
mininet> dpctl del-flows in_port=1
mininet> dpctl del-flows in_port=2
```

因为之前添加的1和2号端口的流表已被删除，使用dpctl dump-flows查看。

dpctl add-flow in_port=2,actions=drop　　　添加丢弃数据包的流表项



例如让交换机丢弃从2号端口发来的所有数据包

```
mininet> dpctl add-flow in_port=2,actions=drop
```

```
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=18.760s, table=0, n_packets=0, n_bytes=0, idle_age=18, in_port=2 actions=drop
*** s2 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=18.759s, table=0, n_packets=0, n_bytes=0, idle_age=18, in_port=2 actions=drop
```

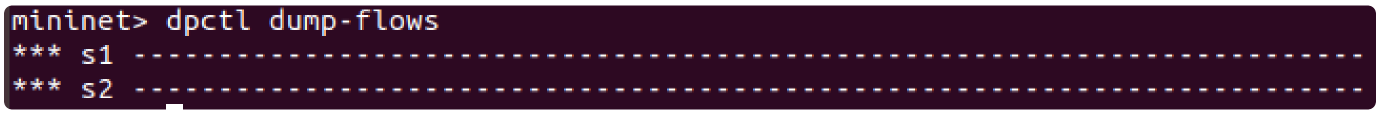
增加这条流表以后，Mininet中主机之间将无法通信。

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X
h2 -> X X
h3 -> X X
*** Results: 100% dropped (0/6 received)
mininet>
```

（二）先解决上面问题，是不是启动Mininet后进行了数据包发送，导致控制器下发流表

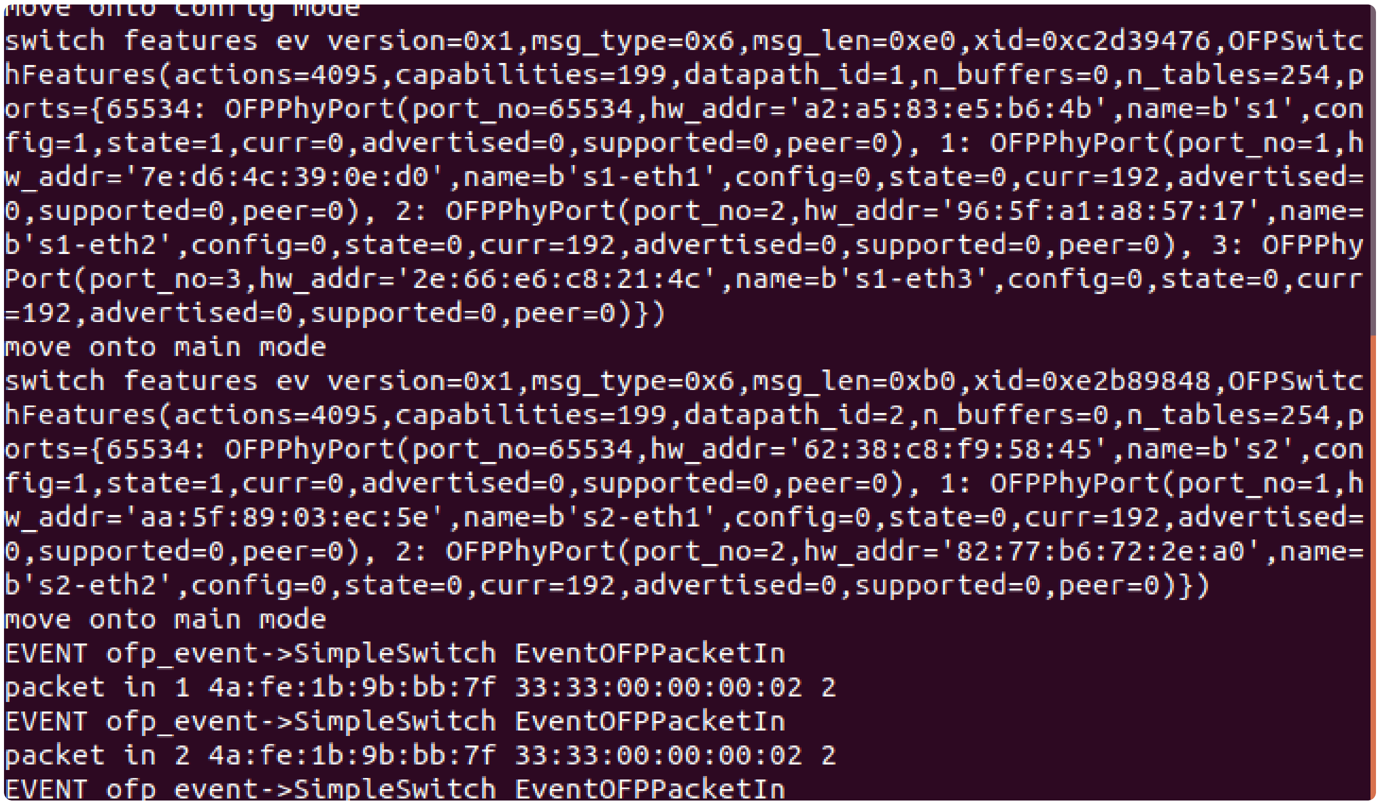
重新启动Ryu和Mininet，直接查看交换机中是否有流表。

1.先启动交换机，查看流表，为空

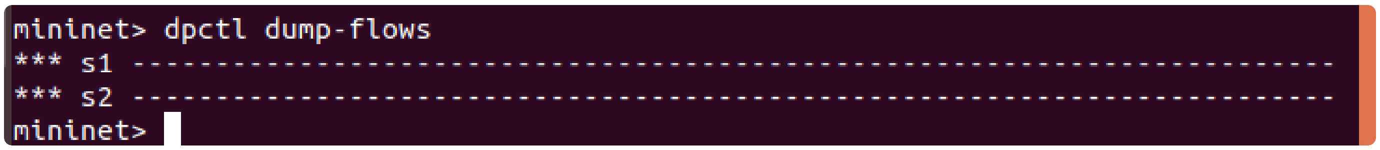


```
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
*** s2 ------------------------------------------------------------------------
```

2.启动控制器，之后再查看交换机中流表信息，依旧为空

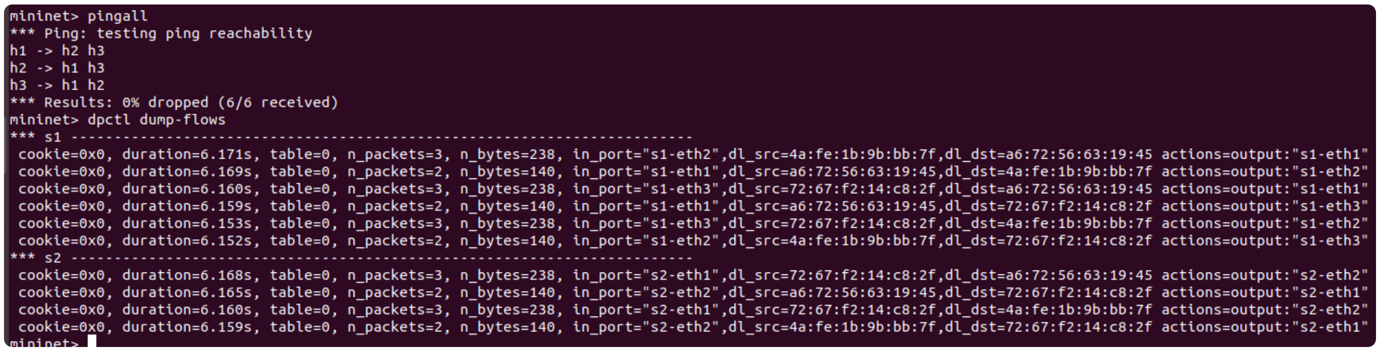


```
move onto config mode
switch features ev version=0x1,msg_type=0x6,msg_len=0xe0,xid=0xc2d39476,OFPSwitc
hFeatures(actions=4095,capabilities=199,datapath_id=1,n_buffers=0,n_tables=254,p
orts={65534: OFPPhyPort(port_no=65534,hw_addr='a2:a5:83:e5:b6:4b',name=b's1',con
fig=1,state=1,curr=0,advertised=0,supported=0,peer=0), 1: OFPPhyPort(port_no=1,h
w_addr='7e:d6:4c:39:0e:d0',name=b's1-eth1',config=0,state=0,curr=192,advertised=
0,supported=0,peer=0), 2: OFPPhyPort(port_no=2,hw_addr='96:5f:a1:a8:57:17',name=
b's1-eth2',config=0,state=0,curr=192,advertised=0,supported=0,peer=0), 3: OFPPhy
Port(port_no=3,hw_addr='2e:66:e6:c8:21:4c',name=b's1-eth3',config=0,state=0,curr
=192,advertised=0,supported=0,peer=0)})
move onto main mode
switch features ev version=0x1,msg_type=0x6,msg_len=0xb0,xid=0xe2b89848,OFPSwitc
hFeatures(actions=4095,capabilities=199,datapath_id=2,n_buffers=0,n_tables=254,p
orts={65534: OFPPhyPort(port_no=65534,hw_addr='62:38:c8:f9:58:45',name=b's2',con
fig=1,state=1,curr=0,advertised=0,supported=0,peer=0), 1: OFPPhyPort(port_no=1,h
w_addr='aa:5f:89:03:ec:5e',name=b's2-eth1',config=0,state=0,curr=192,advertised=
0,supported=0,peer=0), 2: OFPPhyPort(port_no=2,hw_addr='82:77:b6:72:2e:a0',name=
b's2-eth2',config=0,state=0,curr=192,advertised=0,supported=0,peer=0)})
move onto main mode
EVENT ofp_event->SimpleSwitch EventOFPPacketIn
packet in 1 4a:fe:1b:9b:bb:7f 33:33:00:00:00:02 2
EVENT ofp_event->SimpleSwitch EventOFPPacketIn
packet in 2 4a:fe:1b:9b:bb:7f 33:33:00:00:00:02 2
EVENT ofp_event->SimpleSwitch EventOFPPacketIn
```



```
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
*** s2 ------------------------------------------------------------------------
mininet>
```

3.主机使用pingall命令后，查看流表，发生变化



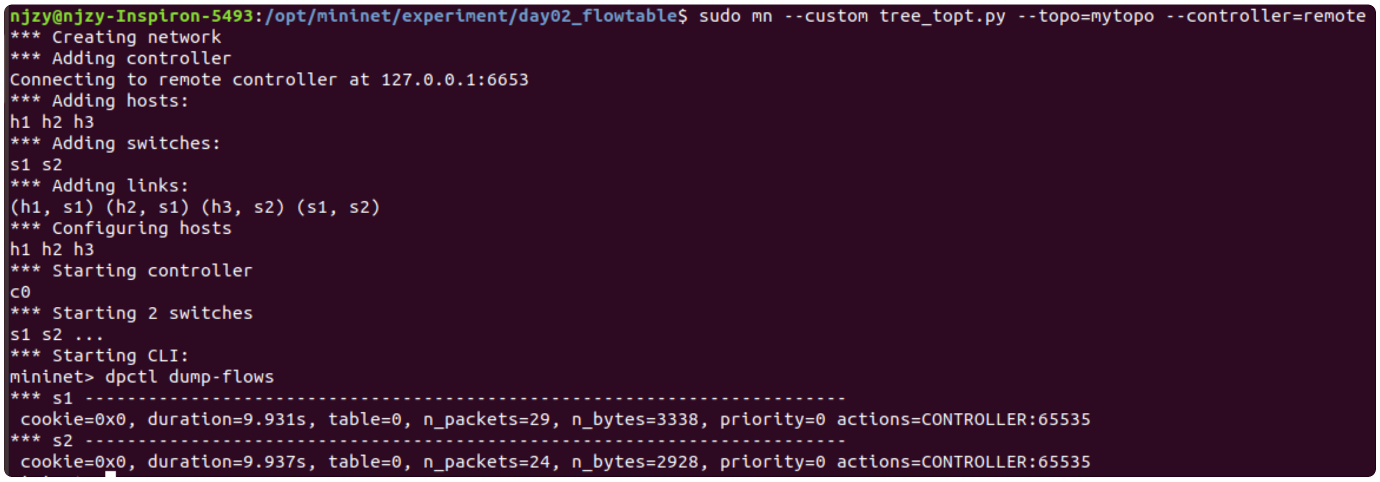
```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
 cookie=0x0, duration=6.171s, table=0, n_packets=3, n_bytes=238, in_port="s1-eth2",dl_src=4a:fe:1b:9b:bb:7f,dl_dst=a6:72:56:63:19:45 actions=output:"s1-eth1"
 cookie=0x0, duration=6.169s, table=0, n_packets=2, n_bytes=140, in_port="s1-eth1",dl_src=a6:72:56:63:19:45,dl_dst=4a:fe:1b:9b:bb:7f actions=output:"s1-eth2"
 cookie=0x0, duration=6.160s, table=0, n_packets=3, n_bytes=238, in_port="s1-eth3",dl_src=72:67:f2:14:c8:2f,dl_dst=a6:72:56:63:19:45 actions=output:"s1-eth1"
 cookie=0x0, duration=6.159s, table=0, n_packets=2, n_bytes=140, in_port="s1-eth1",dl_src=a6:72:56:63:19:45,dl_dst=72:67:f2:14:c8:2f actions=output:"s1-eth3"
 cookie=0x0, duration=6.153s, table=0, n_packets=3, n_bytes=238, in_port="s1-eth3",dl_src=72:67:f2:14:c8:2f,dl_dst=4a:fe:1b:9b:bb:7f actions=output:"s1-eth2"
 cookie=0x0, duration=6.152s, table=0, n_packets=2, n_bytes=140, in_port="s1-eth2",dl_src=4a:fe:1b:9b:bb:7f,dl_dst=72:67:f2:14:c8:2f actions=output:"s1-eth3"
*** s2 ------------------------------------------------------------------------
 cookie=0x0, duration=6.168s, table=0, n_packets=3, n_bytes=238, in_port="s2-eth1",dl_src=72:67:f2:14:c8:2f,dl_dst=a6:72:56:63:19:45 actions=output:"s2-eth2"
 cookie=0x0, duration=6.165s, table=0, n_packets=2, n_bytes=140, in_port="s2-eth2",dl_src=a6:72:56:63:19:45,dl_dst=72:67:f2:14:c8:2f actions=output:"s2-eth1"
 cookie=0x0, duration=6.160s, table=0, n_packets=3, n_bytes=238, in_port="s2-eth1",dl_src=72:67:f2:14:c8:2f,dl_dst=4a:fe:1b:9b:bb:7f actions=output:"s2-eth2"
 cookie=0x0, duration=6.159s, table=0, n_packets=2, n_bytes=140, in_port="s2-eth2",dl_src=4a:fe:1b:9b:bb:7f,dl_dst=72:67:f2:14:c8:2f actions=output:"s2-eth1"
mininet>
```

已解决。但是交换机是如何设置默认流表当不知道packet如何处理的时候发生给控制器？如果这是默认动作，那么我们之前Ryu实验中为何要实现

@set_ev_cls(ofp_event.EventOFPSwitchFeatures,CONFIG_DISPATCHER)

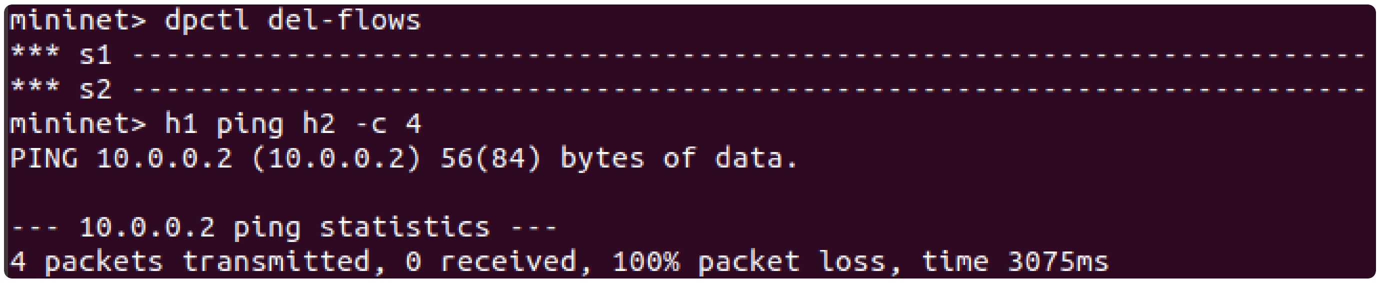
def switch_features_handler(self,ev):　　　　？？？？？

经过启动hub.py在控制器上，进行测试，发现会进入switch_features_handler，并且会下发默认流表---所以说，我们可以不用设置这个默认流表也可以，但是这个函数中，我们可以设置一些其他的流表进行控制---所以说还是比较有用的


```
njzy@njzy-Inspiron-5493:/opt/mininet/experiment/day02_flowtable$ sudo mn --custom tree_topt.py --topo=mytopo --controller=remote
*** Creating network
*** Adding controller
Connecting to remote controller at 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1 s2
*** Adding links:
(h1, s1) (h2, s1) (h3, s2) (s1, s2)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...
*** Starting CLI:
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
 cookie=0x0, duration=9.931s, table=0, n_packets=29, n_bytes=3338, priority=0 actions=CONTROLLER:65535
*** s2 ------------------------------------------------------------------------
 cookie=0x0, duration=9.937s, table=0, n_packets=24, n_bytes=2928, priority=0 actions=CONTROLLER:65535
```

注意从（三）开始的实验我们需要关闭控制器Ryu进行

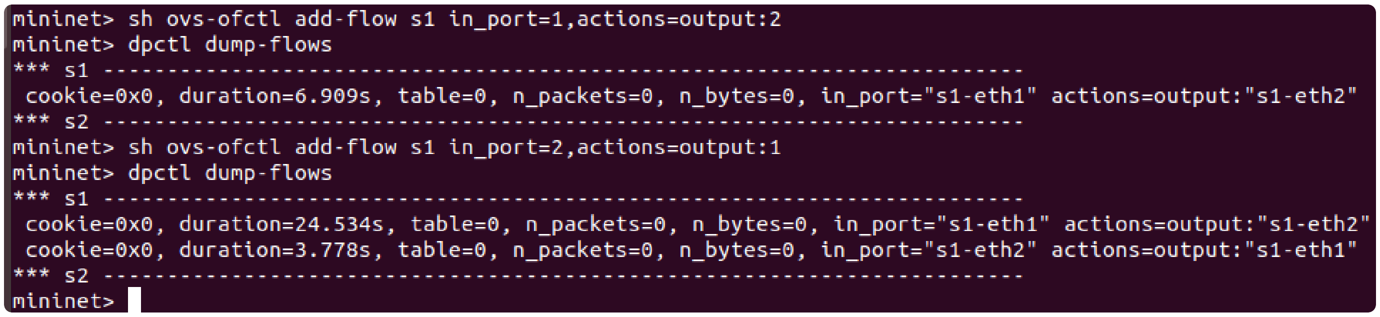
（三）删除所有流表


```
mininet> dpctl del-flows
*** s1 ------------------------------------------------------------------------
*** s2 ------------------------------------------------------------------------
mininet> h1 ping h2 -c 4
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.

--- 10.0.0.2 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3075ms
```

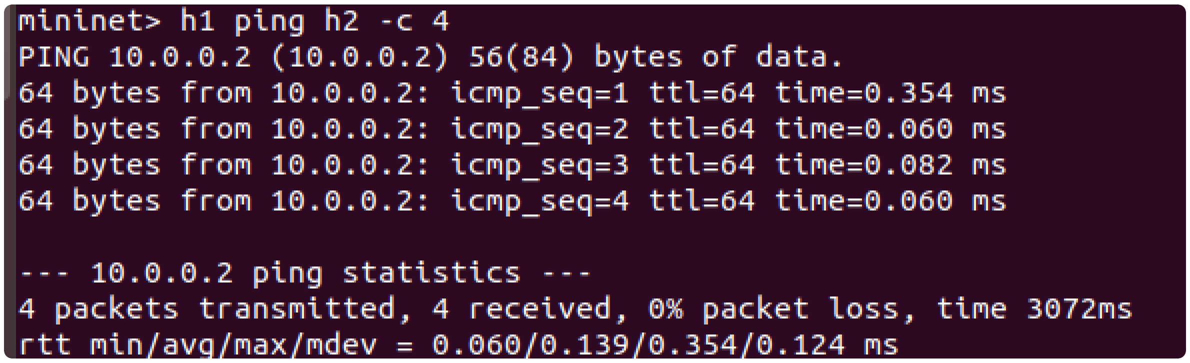
由于没有流表，所有ping操作不可达

（四）添加h1与和h2之间的流表转发

1.单个交换机操作

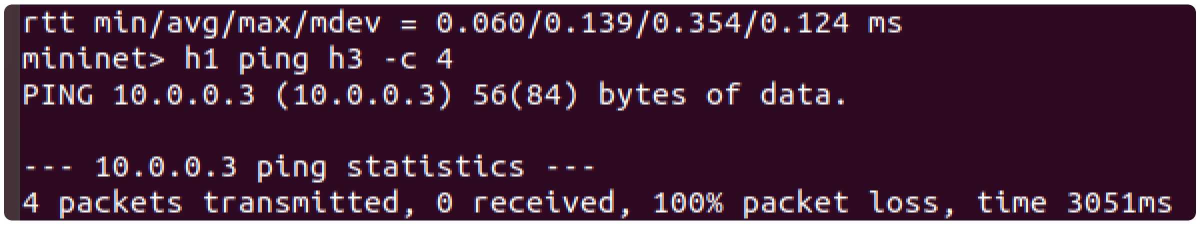

```
mininet> sh ovs-ofctl add-flow s1 in_port=1,actions=output:2
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
 cookie=0x0, duration=6.909s, table=0, n_packets=0, n_bytes=0, in_port="s1-eth1" actions=output:"s1-eth2"
*** s2 ------------------------------------------------------------------------
mininet> sh ovs-ofctl add-flow s1 in_port=2,actions=output:1
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
 cookie=0x0, duration=24.534s, table=0, n_packets=0, n_bytes=0, in_port="s1-eth1" actions=output:"s1-eth2"
 cookie=0x0, duration=3.778s, table=0, n_packets=0, n_bytes=0, in_port="s1-eth2" actions=output:"s1-eth1"
*** s2 ------------------------------------------------------------------------
mininet>
```

2.h1 ping h2,信息可达（因为有流表进行指导）



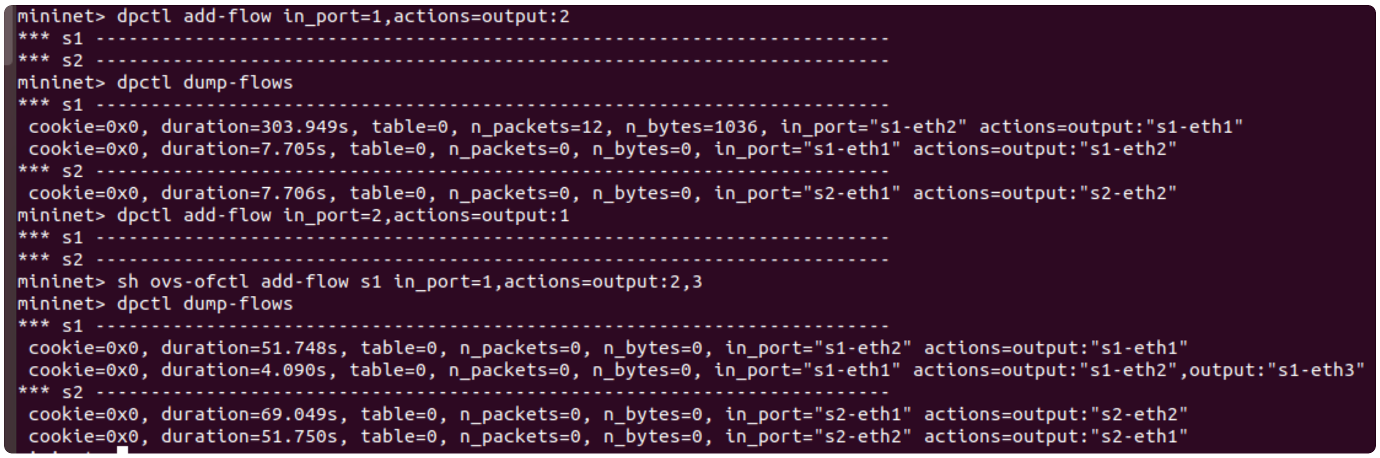
```
mininet> h1 ping h2 -c 4
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.354 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.060 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.082 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.060 ms

--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3072ms
rtt min/avg/max/mdev = 0.060/0.139/0.354/0.124 ms
```

3.h1 ping h3,消息不可达（因为交换机2中没有流表项，并且交换机1也没有配置到port3的动作



```
rtt min/avg/max/mdev = 0.060/0.139/0.354/0.124 ms
mininet> h1 ping h3 -c 4
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.

--- 10.0.0.3 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3051ms
```

4.实现所有网络所有主机互通（先清空流表）



```
mininet> dpctl add-flow in_port=1,actions=output:2
*** s1 ------------------------------------------------------------------------
*** s2 ------------------------------------------------------------------------
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
 cookie=0x0, duration=303.949s, table=0, n_packets=12, n_bytes=1036, in_port="s1-eth2" actions=output:"s1-eth1"
 cookie=0x0, duration=7.705s, table=0, n_packets=0, n_bytes=0, in_port="s1-eth1" actions=output:"s1-eth2"
*** s2 ------------------------------------------------------------------------
 cookie=0x0, duration=7.706s, table=0, n_packets=0, n_bytes=0, in_port="s2-eth1" actions=output:"s2-eth2"
mininet> dpctl add-flow in_port=2,actions=output:1
*** s1 ------------------------------------------------------------------------
*** s2 ------------------------------------------------------------------------
mininet> sh ovs-ofctl add-flow s1 in_port=1,actions=output:2,3
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
 cookie=0x0, duration=51.748s, table=0, n_packets=0, n_bytes=0, in_port="s1-eth2" actions=output:"s1-eth1"
 cookie=0x0, duration=4.090s, table=0, n_packets=0, n_bytes=0, in_port="s1-eth1" actions=output:"s1-eth2",output:"s1-eth3"
*** s2 ------------------------------------------------------------------------
 cookie=0x0, duration=69.049s, table=0, n_packets=0, n_bytes=0, in_port="s2-eth1" actions=output:"s2-eth2"
 cookie=0x0, duration=51.750s, table=0, n_packets=0, n_bytes=0, in_port="s2-eth2" actions=output:"s2-eth1"
```

为所有交换机添加端口1和端口2的操作---两个交换机公共操作

Shell 复制代码

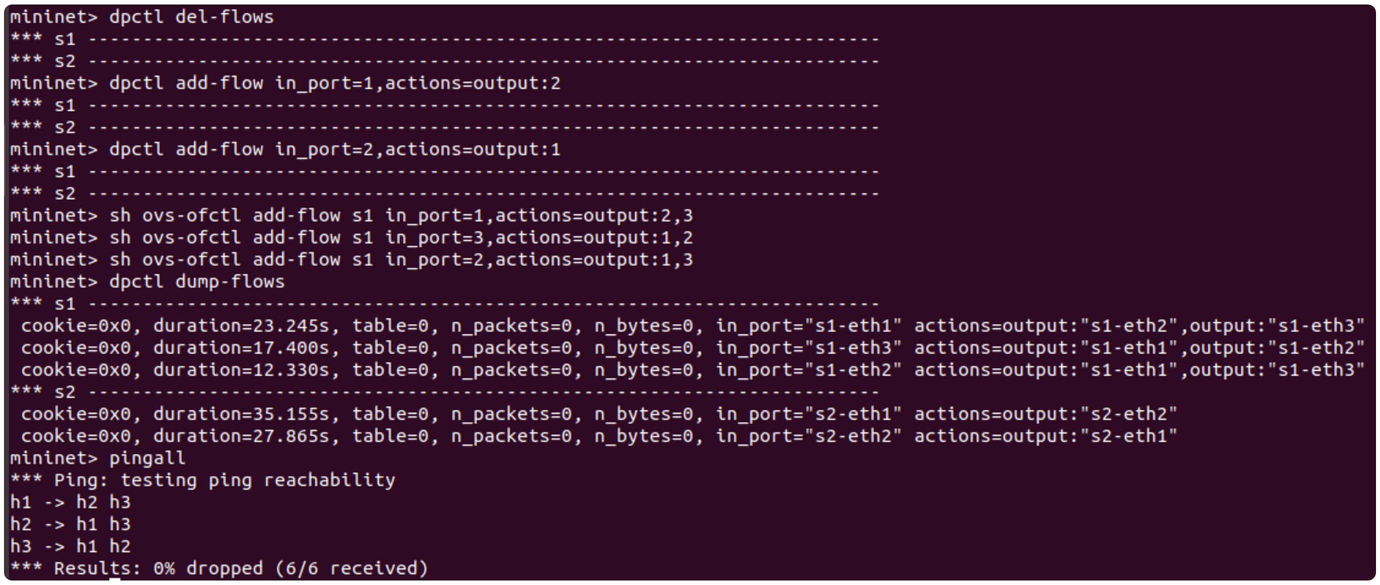
```shell
dpctl add-flow in_port=1,actions=output:2
dpctl add-flow in_port=2,actions=output:1
```

为交换机之间端口提供交互---只操作s1(因为只有s1有端口3)

Shell 复制代码

```shell
sh ovs-ofctl add-flow s1 in_port=1,actions=output:2,3
sh ovs-ofctl add-flow s1 in_port=3,actions=output:1,2
sh ovs-ofctl add-flow s1 in_port=2,actions=output:1,3
```

实验结果显示

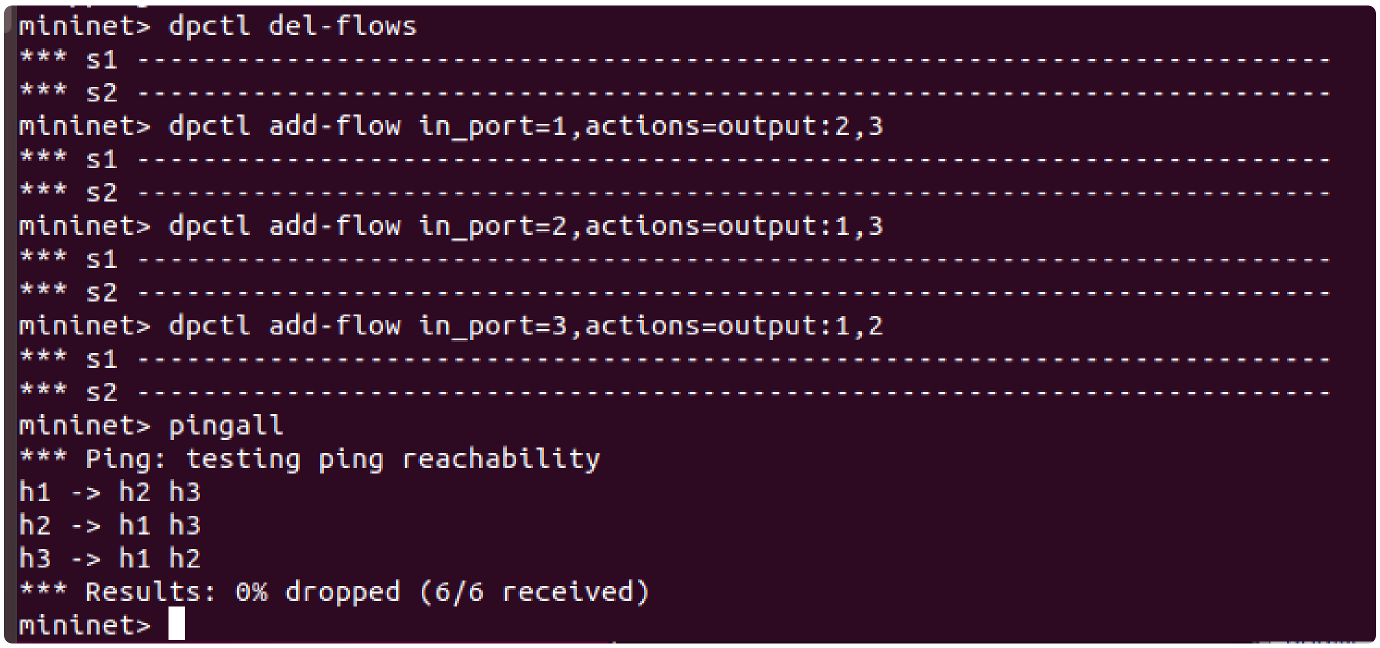


```
mininet> dpctl del-flows
*** s1 ------------------------------------------------------------------------
*** s2 ------------------------------------------------------------------------
mininet> dpctl add-flow in_port=1,actions=output:2
*** s1 ------------------------------------------------------------------------
*** s2 ------------------------------------------------------------------------
mininet> dpctl add-flow in_port=2,actions=output:1
*** s1 ------------------------------------------------------------------------
*** s2 ------------------------------------------------------------------------
mininet> sh ovs-ofctl add-flow s1 in_port=1,actions=output:2,3
mininet> sh ovs-ofctl add-flow s1 in_port=3,actions=output:1,2
mininet> sh ovs-ofctl add-flow s1 in_port=2,actions=output:1,3
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
 cookie=0x0, duration=23.245s, table=0, n_packets=0, n_bytes=0, in_port="s1-eth1" actions=output:"s1-eth2",output:"s1-eth3"
 cookie=0x0, duration=17.400s, table=0, n_packets=0, n_bytes=0, in_port="s1-eth3" actions=output:"s1-eth1",output:"s1-eth2"
 cookie=0x0, duration=12.330s, table=0, n_packets=0, n_bytes=0, in_port="s1-eth2" actions=output:"s1-eth1",output:"s1-eth3"
*** s2 ------------------------------------------------------------------------
 cookie=0x0, duration=35.155s, table=0, n_packets=0, n_bytes=0, in_port="s2-eth1" actions=output:"s2-eth2"
 cookie=0x0, duration=27.865s, table=0, n_packets=0, n_bytes=0, in_port="s2-eth2" actions=output:"s2-eth1"
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
```

或者：我们直接添加下面流表也可以实现上面操作

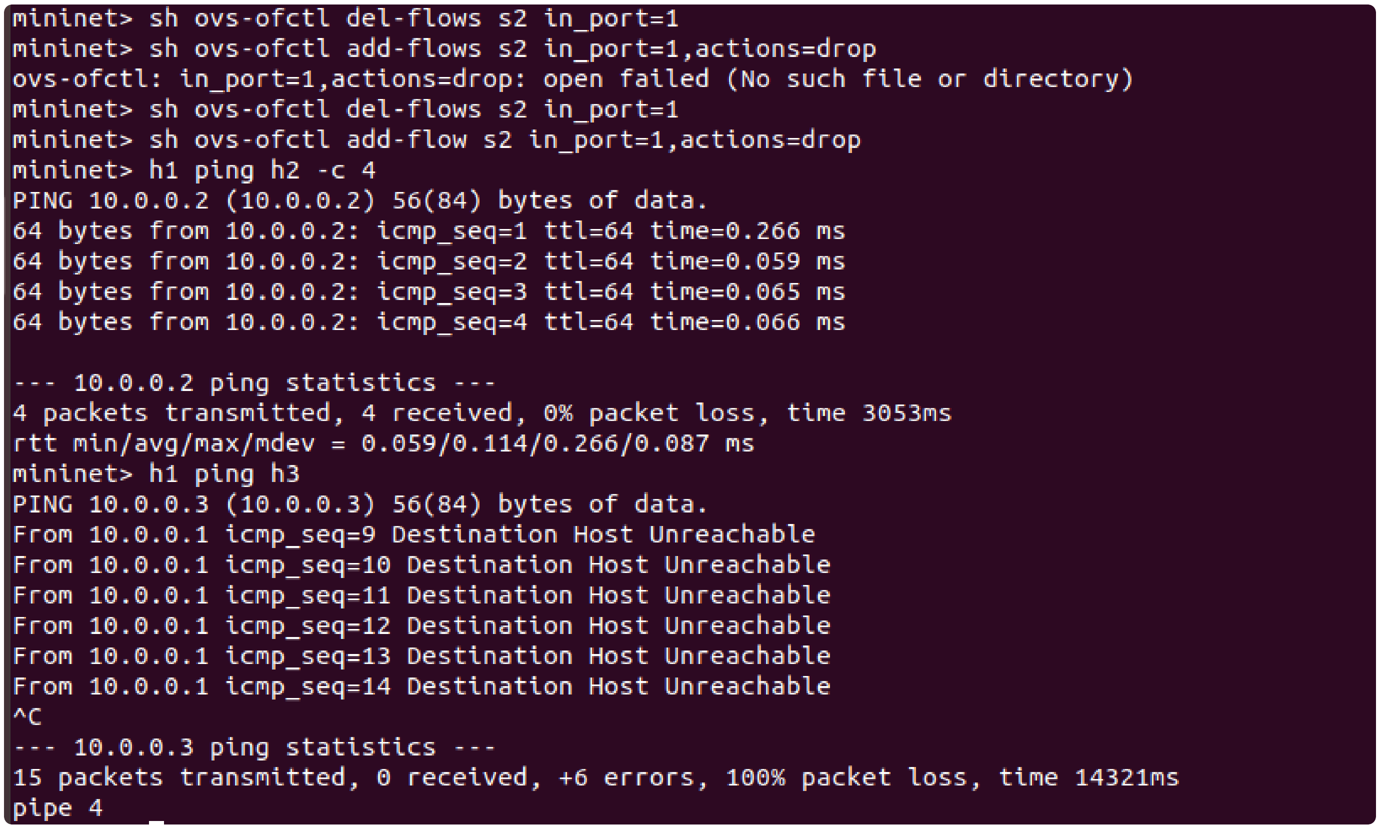
Shell 复制代码

```shell
mininet> dpctl add-flow in_port=1,actions=output:2,3
mininet> dpctl add-flow in_port=2,actions=output:1,3
mininet> dpctl add-flow in_port=3,actions=output:1,2
```


```
mininet> dpctl del-flows
*** s1 ------------------------------------------------------------------------
*** s2 ------------------------------------------------------------------------
mininet> dpctl add-flow in_port=1,actions=output:2,3
*** s1 ------------------------------------------------------------------------
*** s2 ------------------------------------------------------------------------
mininet> dpctl add-flow in_port=2,actions=output:1,3
*** s1 ------------------------------------------------------------------------
*** s2 ------------------------------------------------------------------------
mininet> dpctl add-flow in_port=3,actions=output:1,2
*** s1 ------------------------------------------------------------------------
*** s2 ------------------------------------------------------------------------
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet> 
```

5.为交换机2添加丢弃流表，使得两个交换机不可通信（在前面互通基础上实现）

```shell
mininet> sh ovs-ofctl del-flows s2 in_port=1    删除原有流表
mininet> sh ovs-ofctl add-flow s2 in_port=1,actions=drop    添加丢弃流表
```



```
mininet> sh ovs-ofctl del-flows s2 in_port=1
mininet> sh ovs-ofctl add-flows s2 in_port=1,actions=drop
ovs-ofctl: in_port=1,actions=drop: open failed (No such file or directory)
mininet> sh ovs-ofctl del-flows s2 in_port=1
mininet> sh ovs-ofctl add-flow s2 in_port=1,actions=drop
mininet> h1 ping h2 -c 4
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.266 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.059 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.065 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.066 ms

--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3053ms
rtt min/avg/max/mdev = 0.059/0.114/0.266/0.087 ms
mininet> h1 ping h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=9 Destination Host Unreachable
From 10.0.0.1 icmp_seq=10 Destination Host Unreachable
From 10.0.0.1 icmp_seq=11 Destination Host Unreachable
From 10.0.0.1 icmp_seq=12 Destination Host Unreachable
From 10.0.0.1 icmp_seq=13 Destination Host Unreachable
From 10.0.0.1 icmp_seq=14 Destination Host Unreachable
^C
--- 10.0.0.3 ping statistics ---
15 packets transmitted, 0 received, +6 errors, 100% packet loss, time 14321ms
pipe 4
```