

# 第 1 小题：简单网络

## 一、实验目的

搭建如图 1.1 所示的简单网络，通过流表操作来实现两台不同主机间的 ping 通与否。

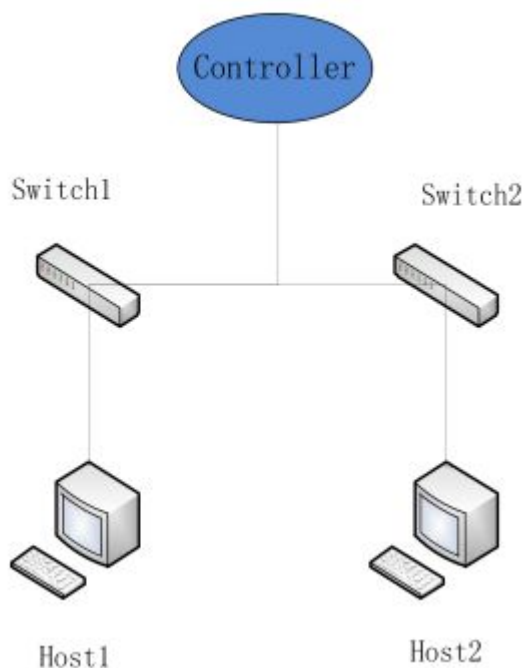


图 1.1：简单网络拓扑

## 二、实验环境搭建

### （一）设计思路

如图 1.1 所示，简单网络由一台 Controller，两台 switch 以及两台 host 组成。我们通过 Controller 添加流表让两台 host 无法 ping 通。

### （二）设备以及平台

我们选择在物理机安装 VMware Workstation 10。下载 SDN Hub ([sdnhub.org](http://sdnhub.org)) 构建的 all-in-one tutorial VM（以下称 SDN 虚拟机）并导入到 VMware。这是一个预装了很多 SDN 相关的软件和工具的 64 位的 Ubuntu 12.10 虚拟机映像。内置软件和工具如下：

- SDN 控制器: Opendaylight, Ryu, Floodlight, Pox 和 Trema
- 示例代码: hub, 2 层学习型交换机和其它应用
- Open vSwitch 1.11 : 支持 Openflow 1.0, 实验性的支持 Openflow 1.2 和 1.3
- Mininet: 创建和运行示例拓扑
- Eclipse 和 Maven
- Wireshark: 协议数据包分析

我们使用的控制器为 floodlight。

### 三、 实验过程及结果

#### (一) 初始环境

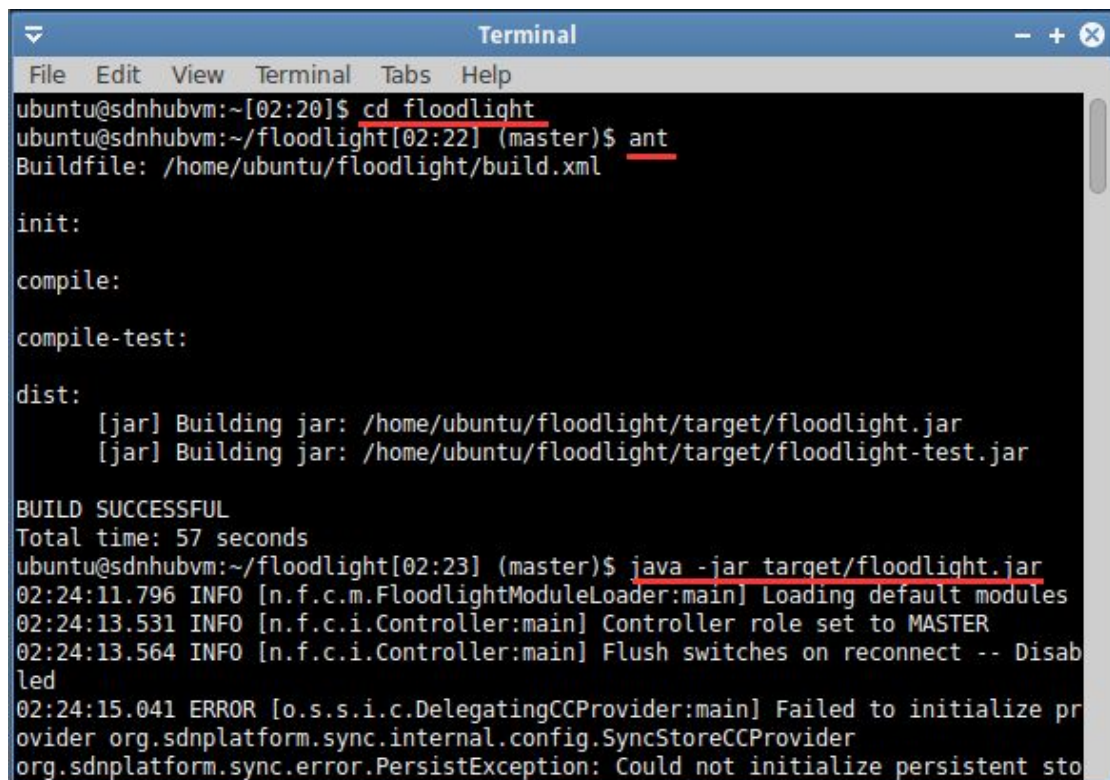
首次进入 SDN 虚拟机, 打开终端, 输入以下命令, 对 floodlight 进行编译及运行:

```
>>cd floodlight
```

```
>>ant
```

```
>>java -jar target/floodlight.jar
```

floodlight 开始监听交换机和 6633 端口 (如图 1.2 和 1.3 所示)。



```
ubuntu@sdnhubvm:~[02:20]$ cd floodlight
ubuntu@sdnhubvm:~/floodlight[02:22] (master)$ ant
Buildfile: /home/ubuntu/floodlight/build.xml

init:

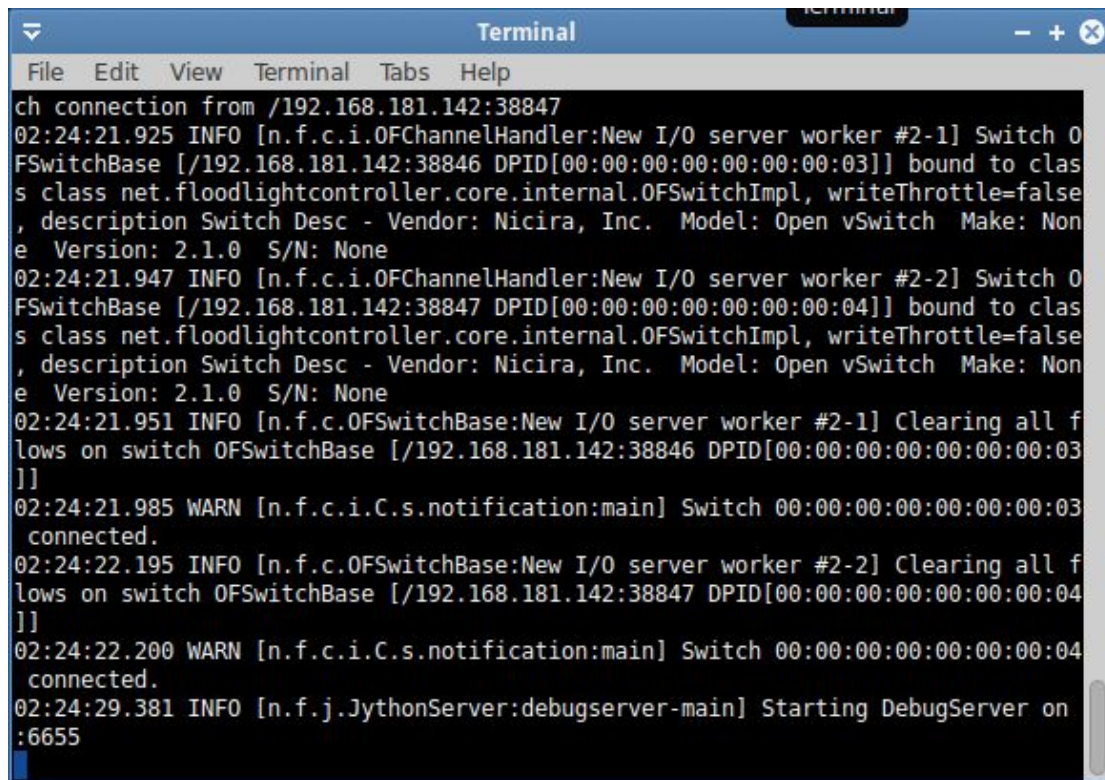
compile:

compile-test:

dist:
[jar] Building jar: /home/ubuntu/floodlight/target/floodlight.jar
[jar] Building jar: /home/ubuntu/floodlight/target/floodlight-test.jar

BUILD SUCCESSFUL
Total time: 57 seconds
ubuntu@sdnhubvm:~/floodlight[02:23] (master)$ java -jar target/floodlight.jar
02:24:11.796 INFO [n.f.c.m.FloodlightModuleLoader:main] Loading default modules
02:24:13.531 INFO [n.f.c.i.Controller:main] Controller role set to MASTER
02:24:13.564 INFO [n.f.c.i.Controller:main] Flush switches on reconnect -- Disabled
02:24:15.041 ERROR [o.s.s.i.c.DelegatingCCProvider:main] Failed to initialize pr
vider org.sdnplatform.sync.internal.config.SyncStoreCCProvider
org.sdnplatform.sync.error.PersistException: Could not initialize persistent sto
```

图 1.2: 编译并运行 floodlight

A screenshot of a terminal window titled "Terminal". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The terminal output shows logs for floodlight. It starts with a connection from 192.168.181.142:38847. Then, two INFO messages show the creation of OFSwitchBase instances for DPIDs 00:00:00:00:00:00:03 and 00:00:00:00:00:00:04. These are followed by WARN messages indicating that switches with these DPIDs are connected. Finally, an INFO message shows the starting of a DebugServer on port 6655.

```
ch connection from /192.168.181.142:38847
02:24:21.925 INFO [n.f.c.i.OFChannelHandler:New I/O server worker #2-1] Switch 0
FSwitchBase [/192.168.181.142:38846 DPID[00:00:00:00:00:00:03]] bound to clas
s class net.floodlightcontroller.core.internal.OFSwitchImpl, writeThrottle=false
, description Switch Desc - Vendor: Nicira, Inc. Model: Open vSwitch Make: Non
e Version: 2.1.0 S/N: None
02:24:21.947 INFO [n.f.c.i.OFChannelHandler:New I/O server worker #2-2] Switch 0
FSwitchBase [/192.168.181.142:38847 DPID[00:00:00:00:00:00:04]] bound to clas
s class net.floodlightcontroller.core.internal.OFSwitchImpl, writeThrottle=false
, description Switch Desc - Vendor: Nicira, Inc. Model: Open vSwitch Make: Non
e Version: 2.1.0 S/N: None
02:24:21.951 INFO [n.f.c.OFSwitchBase:New I/O server worker #2-1] Clearing all f
lows on switch OFSwitchBase [/192.168.181.142:38846 DPID[00:00:00:00:00:00:03
]]
02:24:21.985 WARN [n.f.c.i.C.s.notification:main] Switch 00:00:00:00:00:00:03
connected.
02:24:22.195 INFO [n.f.c.OFSwitchBase:New I/O server worker #2-2] Clearing all f
lows on switch OFSwitchBase [/192.168.181.142:38847 DPID[00:00:00:00:00:00:04
]]
02:24:22.200 WARN [n.f.c.i.C.s.notification:main] Switch 00:00:00:00:00:00:04
connected.
02:24:29.381 INFO [n.f.j.JythonServer:debugserver-main] Starting DebugServer on
:6655
```

图 1.3: 已运行的 floodlight

## (二) mininet 创建拓扑

在终端中输入以下命令创建拓扑:

```
>>sudo mn --custom /home/ubuntu/mininet/custom/topo-2sw-2host.py --topo mytopo
--switch ovsk --controller=remote,ip=192.168.181.142,port=6633
```

topo-2sw-2host.py 文件是该系统中已有的文件, 可直接使用, 且拓扑与题目要求相同。文件内容如下:

```
"""Custom topology example
Two directly connected switches plus a host for each switch:
   host --- switch --- switch --- host
Adding the 'topos' dict with a key/value pair to generate our newly defined
topology enables one to pass in '--topo=mytopo' from the command line.
"""

from mininet.topo import Topo
class MyTopo( Topo ):
    "Simple topology example."

    def __init__( self ):
        "Create custom topo."
```

```

# Initialize topology
Topo.__init__( self )

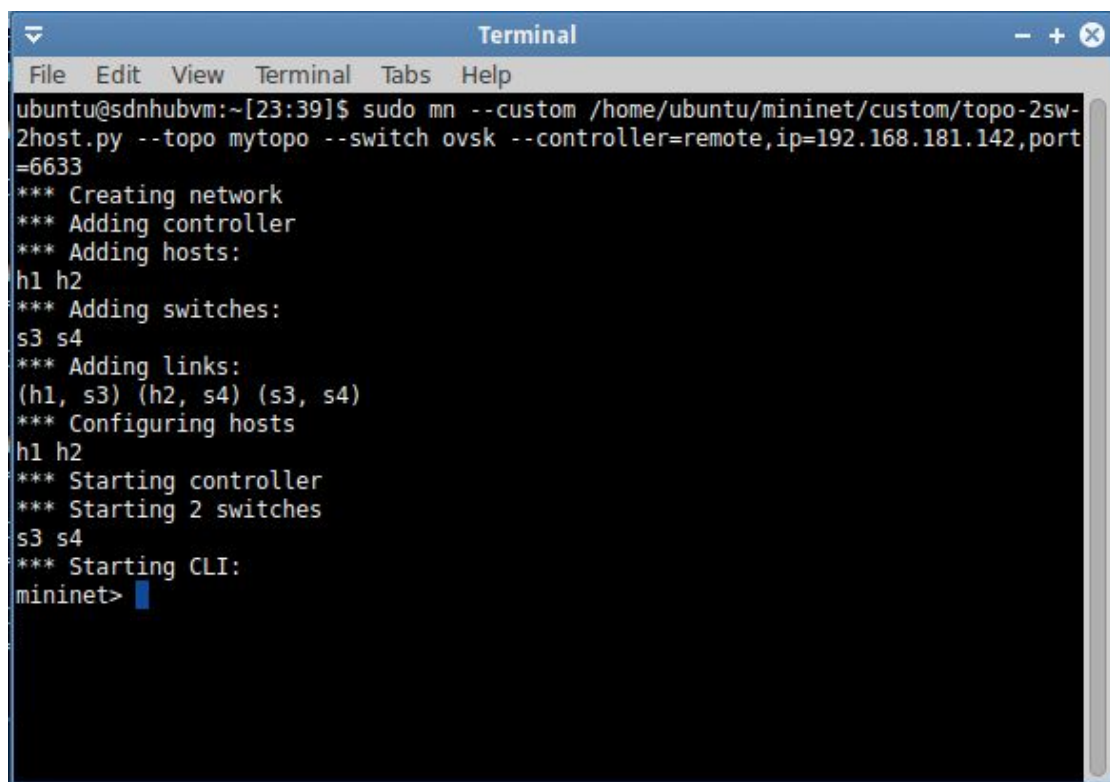
# Add hosts and switches
leftHost = self.addHost( 'h1' )
rightHost = self.addHost( 'h2' )
leftSwitch = self.addSwitch( 's3' )
rightSwitch = self.addSwitch( 's4' )

# Add links
self.addLink( leftHost, leftSwitch )
self.addLink( leftSwitch, rightSwitch )
self.addLink( rightSwitch, rightHost )

topos = { 'mytopo': ( lambda: MyTopo() ) }

```

结果如图 1.4 所示，此拓扑由两台主机 h1,h2 和两台交换机 s3,s4 组成，拓扑图如图 1.1 所示。



```

Terminal
File Edit View Terminal Tabs Help
ubuntu@sdnhubvm:~[23:39]$ sudo mn --custom /home/ubuntu/mininet/custom/topo-2sw-
2host.py --topo mytopo --switch ovsk --controller=remote,ip=192.168.181.142,port
=6633
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s3 s4
*** Adding links:
(h1, s3) (h2, s4) (s3, s4)
*** Configuring hosts
h1 h2
*** Starting controller
*** Starting 2 switches
s3 s4
*** Starting CLI:
mininet>

```

图 1.4: mininet 建立拓扑

打开浏览器输入 <http://localhost:8080/ui/index.html>，进入图形化的 mininet 可  
视界面（如图 1.5 和 1.6 所示）。



[Dashboard](#)
[Topology](#)
[Switches](#)
[Hosts](#)

☒ Live updates

---

Uptime:

7519 s

JVM memory bloat:

60792904 free out of 104333312

Modules loaded:

n.f.topology.TopologyManager, n.f.flowcache.FlowReconcileManager, n.f.devicemanager.internal.DefaultEntityClassifier,  
 n.f.storage.memory.MemoryStorageSource, n.f.debugcounter.DebugCounter, n.f.counter.CounterStore, n.f.restserver.RestApiServer,  
 org.sdnplatform.sync.internal.SyncManager, n.f.firewall.Firewall, n.f.perfmon.PktInProcessingTime,  
 n.f.devicemanager.internal.DeviceManagerImpl, n.f.linkdiscovery.internal.LinkDiscoveryManager, n.f.threadpool.ThreadPool,  
 n.f.staticflowentry.StaticFlowEntryPusher, n.f.core.internal.FloodlightProvider, n.f.loadbalancer.LoadBalancer,  
 n.f.debugevent.DebugEvent,

## Switches (2)

DPID	IP Address	Vendor	Packets	Bytes	Flows	Connected Since
00:00:00:00:00:00:03	/192.168.181.142:39007	Nicira, Inc.	0	0	0	5/23/2015, 11:40:03 PM
00:00:00:00:00:00:04	/192.168.181.142:39009	Nicira, Inc.	0	0	0	5/23/2015, 11:40:04 PM

## Hosts (2)

MAC Address	IP Address	Switch Port	Last Seen
d6:62:fa:a2:1f:4f	10.0.0.2	00:00:00:00:00:00:04-1	5/23/2015, 11:55:30 PM
26:05:bf:b2:0e:3b	10.0.0.1	00:00:00:00:00:00:03-2	5/23/2015, 11:55:30 PM

图 1.5: Mininet 图形化界面

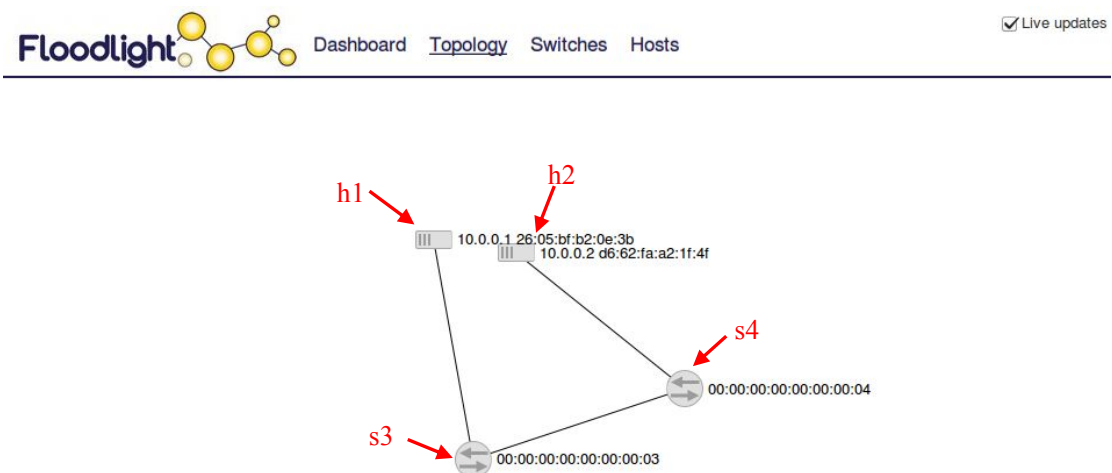


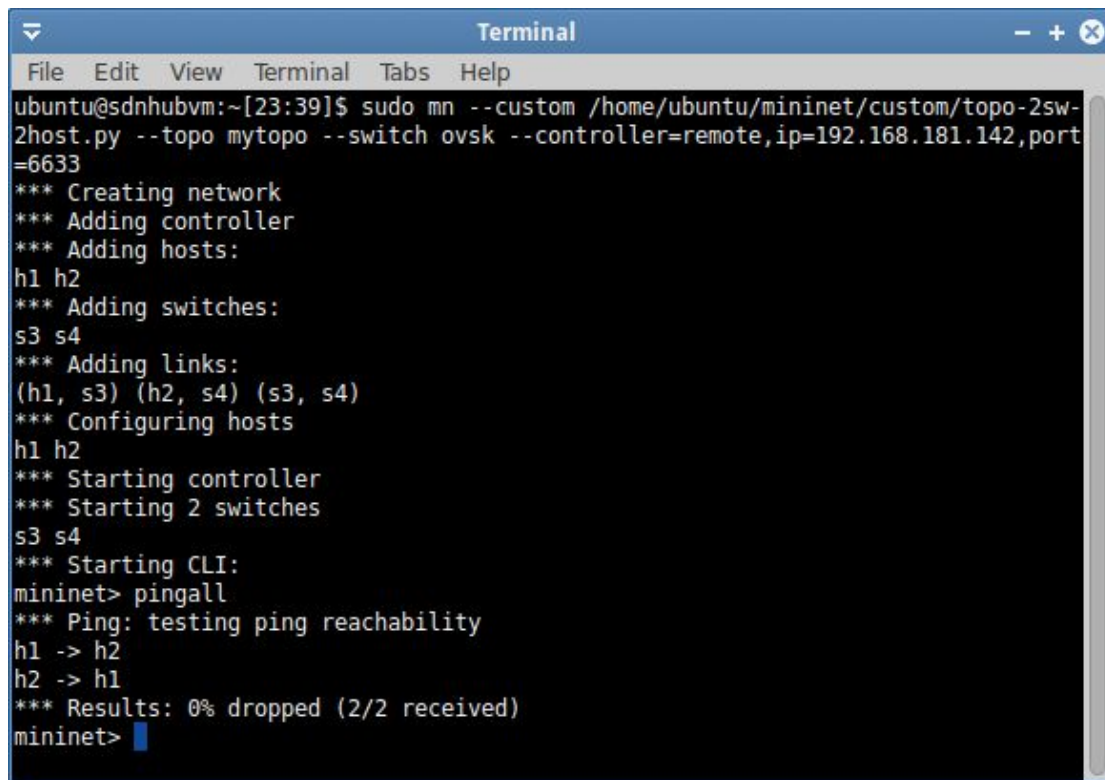
图 1.6: Mininet 图形化拓扑结构界面

### (三) 测试主机是否 ping 通

在 mininet 中输入以下命令，测试 h1 与 h2 能否 ping 通（如图 1.7 所示）：

```
>>pingall
```





```
ubuntu@sdnhubvm:~[23:39]$ sudo mn --custom /home/ubuntu/mininet/custom/topo-2sw-2host.py --topo mytopo --switch ovsk --controller=remote,ip=192.168.181.142,port=6633
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s3 s4
*** Adding links:
(h1, s3) (h2, s4) (s3, s4)
*** Configuring hosts
h1 h2
*** Starting controller
*** Starting 2 switches
s3 s4
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet>
```

图 1.7: 测试 h1 和 h2 的连接

此时丢包率为 0%，即两台主机 h1 与 h2 可以 ping 通。

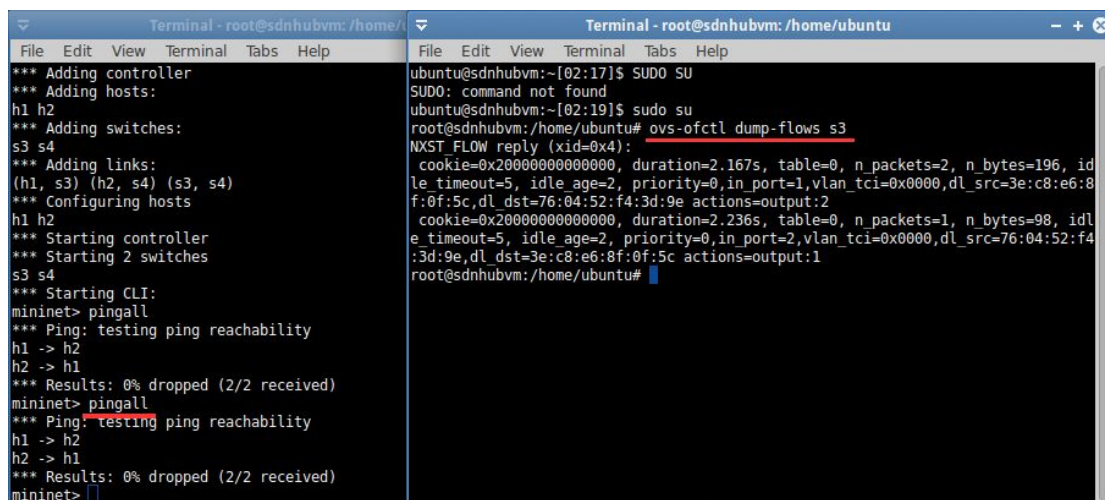
#### （四）添加流表使主机间无法 ping 通

在终端中输入以下命令：

```
>>ovs-ofctl dump-flows s3
```

```
>>ovs-ofctl add-flow s3 priority=1,in_port=2,actions=drop
```

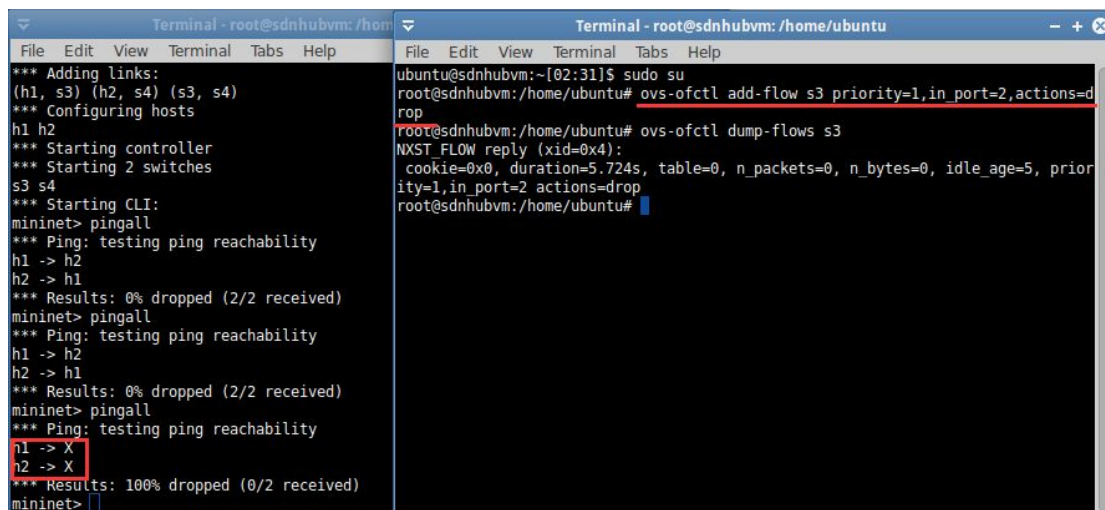
如图 1.8 和图 1.9 所示。图 1.8 中显示的流表为 Controller 添加的临时流表，5s 后会自动失效。



```
Terminal - root@sdnhubvm: /home/
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s3 s4
*** Adding links:
(h1, s3) (h2, s4) (s3, s4)
*** Configuring hosts
h1 h2
*** Starting controller
*** Starting 2 switches
s3 s4
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet>

Terminal - root@sdnhubvm: /home/ubuntu
ubuntu@sdnhubvm:~[02:17]$ SUDO SU
SUDO: command not found
ubuntu@sdnhubvm:~[02:19]$ sudo su
root@sdnhubvm:/home/ubuntu# ovs-ofctl dump-flows s3
NXST_FLOW reply (xid=0x4):
  cookie=0x2000000000000000, duration=2.167s, table=0, n_packets=2, n_bytes=196, idle timeout=5, idle age=2, priority=0, in_port=1, vlan_tci=0x0000, dl_src=3e:c8:e6:8f:0f:5c, dl_dst=76:04:52:f4:3d:9e actions=output:2
  cookie=0x2000000000000000, duration=2.236s, table=0, n_packets=1, n_bytes=98, idle timeout=5, idle age=2, priority=0, in_port=2, vlan_tci=0x0000, dl_src=76:04:52:f4:3d:9e, dl_dst=3e:c8:e6:8f:0f:5c actions=output:1
root@sdnhubvm:/home/ubuntu#
```

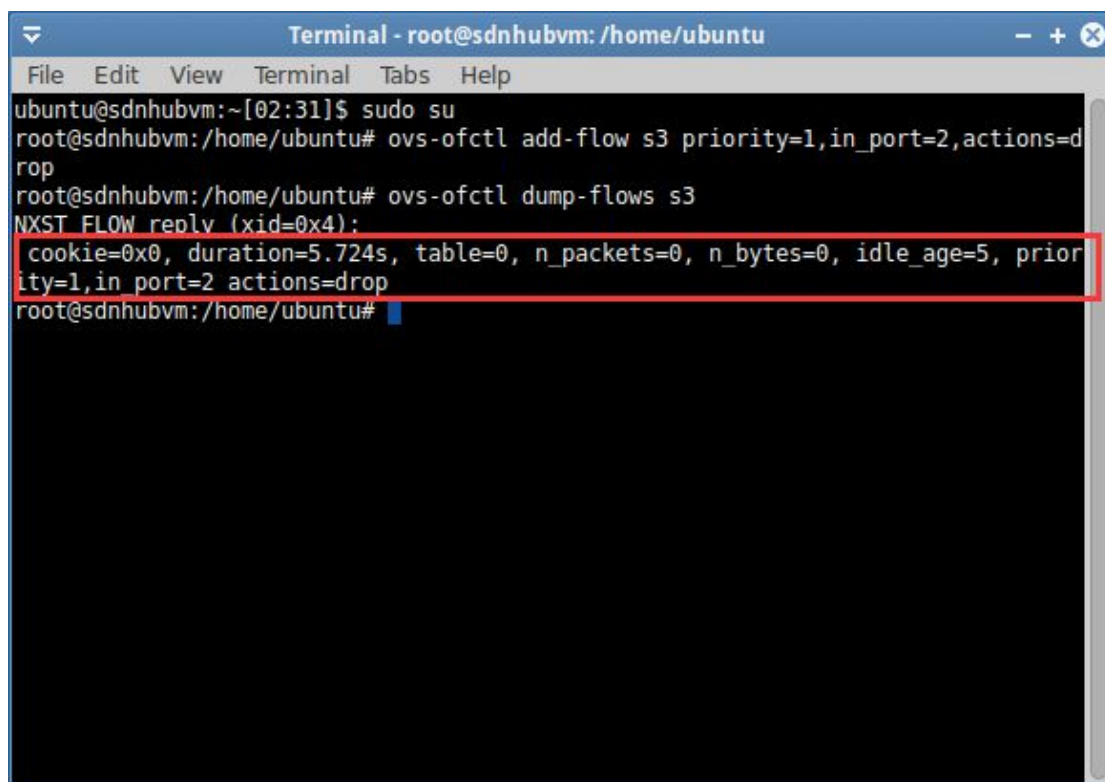
图 1.8: 查看交换机 s3 的流表



```
Terminal - root@sdnhubvm: /home/
File Edit View Terminal Tabs Help
*** Adding links:
(h1, s3) (h2, s4) (s3, s4)
*** Configuring hosts
h1 h2
*** Starting controller
*** Starting 2 switches
s3 s4
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet> pingall
*** Ping: testing ping reachability
h1 -> X
h2 -> X
*** Results: 100% dropped (0/2 received)
mininet>

Terminal - root@sdnhubvm: /home/ubuntu
File Edit View Terminal Tabs Help
ubuntu@sdnhubvm:~[02:31]$ sudo su
root@sdnhubvm:/home/ubuntu# ovs-ofctl add-flow s3 priority=1,in_port=2,actions=drop
root@sdnhubvm:/home/ubuntu# ovs-ofctl dump-flows s3
NXST_FLOW reply (xid=0x4):
cookie=0x0, duration=5.724s, table=0, n_packets=0, n_bytes=0, idle_age=5, priority=1,in_port=2 actions=drop
root@sdnhubvm:/home/ubuntu#
```

图 1.9: 添加流表使 h1, h2 不能连通



```
Terminal - root@sdnhubvm: /home/ubuntu
File Edit View Terminal Tabs Help
ubuntu@sdnhubvm:~[02:31]$ sudo su
root@sdnhubvm:/home/ubuntu# ovs-ofctl add-flow s3 priority=1,in_port=2,actions=drop
root@sdnhubvm:/home/ubuntu# ovs-ofctl dump-flows s3
NXST_FLOW reply (xid=0x4):
cookie=0x0, duration=5.724s, table=0, n_packets=0, n_bytes=0, idle_age=5, priority=1,in port=2 actions=drop
root@sdnhubvm:/home/ubuntu#
```

图 1.10: 添加的流表信息

由图 1.10 显示，在添加了该流表后，输入 pingall 命令，丢包率达 100%，主机 h1 和 h2 无法 ping 通，实现题目要求。