

## 实验四——SDN 环境搭建与协议仿真

### 一、实验目的

- 1.1 学习了解软定义网络 SDN
- 1.2 了解 SDN 封包传输的流程
- 1.3 通过观察 OpenFlow 跟传统网络协议的沟通过程
- 1.4 希望能在实体及其的环境下对 SDN 跟传统网络有更好的理解

### 二、实验前的准备

- 2.1 熟悉了解软定义网络 SDN
- 2.2 了解 Mininet 的使用
- 2.3 环境准备：linux 系统【可以安装虚拟机在虚拟机上完成实验，最好安装 Ubuntu16.x 及以上版本进行实验】

### 三、实验内容

#### A. Mininet 使用

```
kwy@kwy-virtual-machine:~/mininet/util$ sudo mn
[sudo] kwy 的密码:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
```

建立默认拓扑

```
mininet> nodes
available nodes are:
c0 h1 h2 s1
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo: s1-eth1:h1-eth0 s1-eth2:h2-eth0
c0
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=21231>
<Host h2: h2-eth0:10.0.0.2 pid=21233>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=21238>
<OVSController c0: 127.0.0.1:6653 pid=21224>
mininet> h1 ping h2
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=36.8 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.398 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.080 ms
^C
--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 0.080/12.439/36.841/17.255 ms
mininet> ping all
*** Unknown command: ping all
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
```

查看节点

显示链接信息

各个节点的信息

h1 ping h2 验证h1 h2的连通性

验证所有主机的连通性

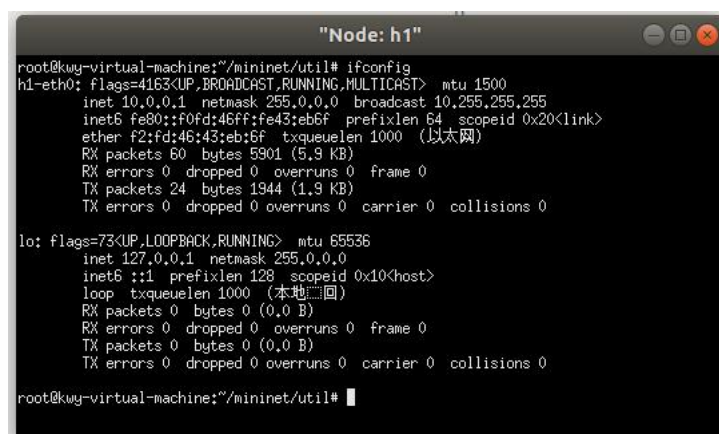
## B. 利用 mininet 创建网络拓扑

使用 `mn -topo single,3` 建立一个具有三个主机，一个交换机，一个控制器的网络。

```
kwy@kwy-virtual-machine:~$ sudo mn --topo single,3
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=21418>
<Host h2: h2-eth0:10.0.0.2 pid=21420>
<Host h3: h3-eth0:10.0.0.3 pid=21422>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None pid=21427>
<OVSController c0: 127.0.0.1:6653 pid=21411>
```

## C. 简易环境测试

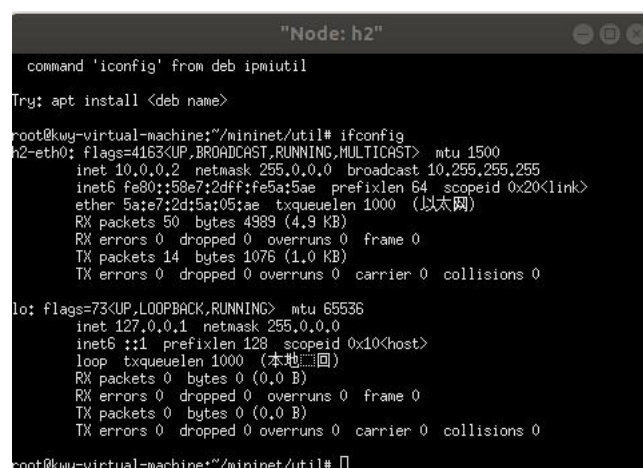
可以从下面实验截图中看到 h1 的 ip 地址是 10.0.0.1, h2 的 ip 地址是 10.0.0.2



```
"Node: h1"
root@kwy-virtual-machine:~/mininet/util# ifconfig
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.0.1 netmask 255.0.0.0 broadcast 10.255.255.255
    inet6 fe80::f0fd:46ff:fe43:eb6f prefixlen 64 scopeid 0x20<link>
    ether f2:fd:46:43:eb:6f txqueuelen 1000 (以太网)
    RX packets 60 bytes 5901 (5.9 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 24 bytes 1944 (1.9 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (本地环回)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@kwy-virtual-machine:~/mininet/util#
```



```
"Node: h2"
command 'ifconfig' from deb ipmiutil
Try: apt install <deb name>

root@kwy-virtual-machine:~/mininet/util# ifconfig
h2-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.0.2 netmask 255.0.0.0 broadcast 10.255.255.255
    inet6 fe80::58e7:2dff:fe5a:5ae prefixlen 64 scopeid 0x20<link>
    ether 5a:e7:2d:5a:05:ae txqueuelen 1000 (以太网)
    RX packets 50 bytes 4989 (4.9 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 14 bytes 1076 (1.0 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (本地环回)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@kwy-virtual-machine:~/mininet/util#
```

H1 ping h2 后查看流表项：

```
root@kwy-virtual-machine:~# ovs-ofctl dump-flows s1 没有ping之前的流表项
cookie=0x0, duration=82.441s, table=0, n_packets=17, n_bytes=1326, priority=0 actions=CONTROLLER:128
root@kwy-virtual-machine:~# ovs-ofctl dump-flows s1
cookie=0x0, duration=14.731s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth2",vlan_tci=0x0000/0x1fff,dl_src=fe:ac:57:ff:3f:9a,dl_dst=b6:b8:64:cb:6b:22,arp_spa=10.0.0.2,arp_tpa=10.0.0.1,arp_op=2 actions=output:"s1-eth1"
cookie=0x0, duration=9.565s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth2",vlan_tci=0x0000/0x1fff,dl_src=fe:ac:57:ff:3f:9a,dl_dst=b6:b8:64:cb:6b:22,arp_spa=10.0.0.2,arp_tpa=10.0.0.1,arp_op=1 actions=output:"s1-eth1"
cookie=0x0, duration=9.550s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth1",vlan_tci=0x0000/0x1fff,dl_src=b6:b8:64:cb:6b:22,dl_dst=fe:ac:57:ff:3f:9a,arp_spa=10.0.0.1,arp_tpa=10.0.0.2,arp_op=2 actions=output:"s1-eth2"
cookie=0x0, duration=14.708s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,icmp,in_port="s1-eth1",vlan_tci=0x0000/0x1fff,dl_src=b6:b8:64:cb:6b:22,dl_dst=fe:ac:57:ff:3f:9a,nw_src=10.0.0.1,nw_dst=10.0.0.2,nw_tos=0,icmp_type=8,icmp_code=0 actions=output:"s1-eth2"
cookie=0x0, duration=14.694s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,icmp,in_port="s1-eth2",vlan_tci=0x0000/0x1fff,dl_src=fe:ac:57:ff:3f:9a,dl_dst=b6:b8:64:cb:6b:22,nw_src=10.0.0.2,nw_dst=10.0.0.1,nw_tos=0,icmp_type=0,icmp_code=0 actions=output:"s1-eth1"
cookie=0x0, duration=156.664s, table=0, n_packets=25, n_bytes=1830, priority=0 actions=CONTROLLER:128
root@kwy-virtual-machine:~#
```

ping之后的流表多了四条流表项

分别是 h1 到 h2 的 ARP 包、h2 到 h1 的 ARP 包、h1 到 h2 的 ICMP 包、h2 到 h1 的 ICMP 包。

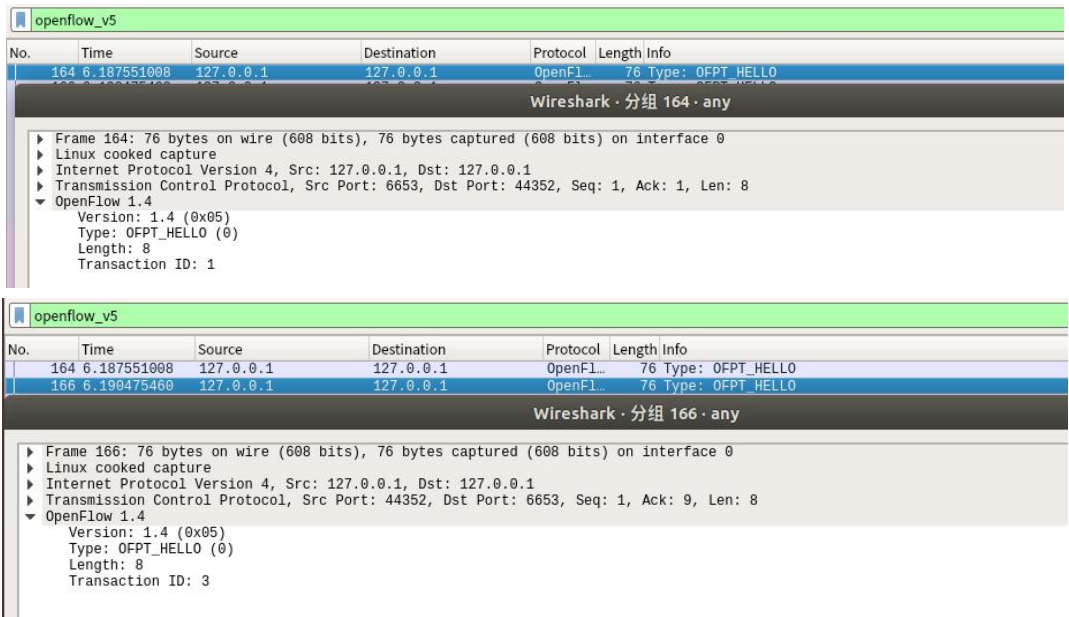
D. Wireshark 抓包

Hello：控制器与交换机建立连接时由其中某一方发起 Hello 消息，双方协调协议版本号。

Version：OpenFlow 版本，低位为版本号。Type：OpenFlow 消息类型

Length：消息总长度，包含头部。

Xid：事件 ID，同一件事件的 ID 号一致。如 feature\_request 和对应的 feature\_reply 就使用同一个 Transaction id，但是两个 hello 消息的 Transaction id 并不相同，不过据实验结果看两个 id 一般是两个相邻的数字。并且 packet\_in 的 transaction id 都为 0。



Features Request 就是控制器问交换机有什么功能。如接口，配置，地址，宽带信息等

No.	Time	Source	Destination	Protocol	Length	Info
164	6.187551008	127.0.0.1	127.0.0.1	OpenFl...	76	Type: OFPT_HELLO
166	6.190475460	127.0.0.1	127.0.0.1	OpenFl...	76	Type: OFPT_HELLO
168	6.190695434	127.0.0.1	127.0.0.1	OpenFl...	76	Type: OFPT_FEATURES_REQUEST
169	6.190728630	127.0.0.1	127.0.0.1	OpenFl...	80	Type: OFPT_SET_CONFIG

Wireshark · 分组 168 · any

▶ Frame 168: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface 0  
▶ Linux cooked capture  
▶ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1  
▶ Transmission Control Protocol, Src Port: 6653, Dst Port: 44352, Seq: 9, Ack: 9, Len: 8  
▼ OpenFlow 1.4  
  Version: 1.4 (0x05)  
  Type: OFPT\_FEATURES\_REQUEST (5)  
  Length: 8  
  Transaction ID: 2

set config 控制器做一个简单的适合交换机的设置

No.	Time	Source	Destination	Protocol	Length	Info
164	6.187551008	127.0.0.1	127.0.0.1	OpenFl...	76	Type: OFPT_HELLO
166	6.190475460	127.0.0.1	127.0.0.1	OpenFl...	76	Type: OFPT_HELLO
168	6.190695434	127.0.0.1	127.0.0.1	OpenFl...	76	Type: OFPT_FEATURES_REQUEST
169	6.190728630	127.0.0.1	127.0.0.1	OpenFl...	80	Type: OFPT_SET_CONFIG
170	6.190866983	127.0.0.1	127.0.0.1	OpenFl...	148	Type: OFPT_FLOW_MOD

Wireshark · 分组 169 · any

▶ Frame 169: 80 bytes on wire (640 bits), 80 bytes captured (640 bits) on interface 0  
▶ Linux cooked capture  
▶ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1  
▶ Transmission Control Protocol, Src Port: 6653, Dst Port: 44352, Seq: 17, Ack: 9, Len: 12  
▼ OpenFlow 1.4  
  Version: 1.4 (0x05)  
  Type: OFPT\_SET\_CONFIG (9)  
  Length: 12  
  Transaction ID: 3  
  Flags: 0x0000  
  Miss send length: 128

向交换机下发默认流表项，优先级是 0，表示收到数据包之后转发到控制器。

168	6.190695434	127.0.0.1	127.0.0.1	OpenFl...	76	Type: OFPT_FEATURES_REQUEST
169	6.190728630	127.0.0.1	127.0.0.1	OpenFl...	80	Type: OFPT_SET_CONFIG
170	6.190866983	127.0.0.1	127.0.0.1	OpenFl...	148	Type: OFPT_FLOW_MOD

Wireshark · 分组 170 · any

▶ Frame 170: 148 bytes on wire (1184 bits), 148 bytes captured (1184 bits) on interface 0  
▶ Linux cooked capture  
▶ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1  
▶ Transmission Control Protocol, Src Port: 6653, Dst Port: 44352, Seq: 29, Ack: 9, Len: 80  
▼ OpenFlow 1.4  
  Version: 1.4 (0x05)  
  Type: OFPT\_FLOW\_MOD (14)  
  Length: 80  
  Transaction ID: 4  
  Cookie: 0x0000000000000000  
  Cookie mask: 0x0000000000000000  
  Table ID: 0  
  Command: OFPFC\_ADD (0)  
  Idle timeout: 0  
  Hard timeout: 0  
  Priority: 0  
  Buffer ID: OFP\_NO\_BUFFER (4294967295)  
  Out port: OFPP\_ANY (4294967295)  
  Out group: OFPG\_ANY (4294967295)  
  Flags: 0x0000  
    ... ..0 = Send flow removed: False  
    ... ..0. = Check overlap: False  
    ... ..0.. = Reset counts: False  
    ... ..0... = Don't count packets: False  
    ... ..0.... = Don't count bytes: False  
  Importance: 0  
  ▼ Match  
    Type: OFPMT\_OXM (1)  
    Length: 4  
    Pad: 00000000  
  ▼ Instruction  
    Type: OFPIT\_APPLY\_ACTIONS (4)  
    Length: 24  
    Pad: 00000000  
    ▼ Action  
      Type: OFPAT\_OUTPUT (0)  
      Length: 16  
      Port: OFPP\_CONTROLLER (4294967293)  
      Max length: 128  
      Pad: 000000000000



Features Reply 交换机向控制器回复有什么功能。

No.	Time	Source	Destination	Protocol	Length	Info
164	6.187551008	127.0.0.1	127.0.0.1	OpenFl...	76	Type: OFPT_HELLO
166	6.190475460	127.0.0.1	127.0.0.1	OpenFl...	76	Type: OFPT_HELLO
168	6.190695434	127.0.0.1	127.0.0.1	OpenFl...	76	Type: OFPT_FEATURES_REQUEST
169	6.190728630	127.0.0.1	127.0.0.1	OpenFl...	80	Type: OFPT_SET_CONFIG
170	6.190866983	127.0.0.1	127.0.0.1	OpenFl...	148	Type: OFPT_FLOW_MOD
171	6.192679941	127.0.0.1	127.0.0.1	OpenFl...	100	Type: OFPT_FEATURES_REPLY
173	6.238104452	127.0.0.1	127.0.0.1	OpenFl...	156	Type: OFPT_PORT_STATUS

Port status 交换机和控制器连接后，控制器会不断发送 stats 消息询问交换机的状态，维持网络视图的实时更新。

171	6.432679941	127.0.0.1	127.0.0.1	OpenFl...	170	Type: OFPT_FEATURE_REPLY
173	6.238104452	127.0.0.1	127.0.0.1	OpenFl...	156	Type: OFPT_PORT_STATUS
175	6.238450162	127.0.0.1	127.0.0.1	OpenFl...	156	Type: OFPT_PORT_STATUS

734	16.359217076	b6:a2:24:f8:3a:18	127.0.0.1	ARP	44 Who has 10.0.0.3? Tell 10.0.0.1
735	16.359425877	127.0.0.1	127.0.0.1	OpenFl_	152 Type: OFPT_PACKET_IN
736	16.359597099	127.0.0.1	127.0.0.1	OpenFl_	150 Type: OFPT_PACKET_OUT
737	16.351086759	b6:a2:24:f8:3a:18	127.0.0.1	ARP	44 Who has 10.0.0.3? Tell 10.0.0.1
738	16.351089419	b6:a2:24:f8:3a:18	127.0.0.1	ARP	44 Who has 10.0.0.3? Tell 10.0.0.1
739	16.351101599	12:19:d6:93:ea:60	127.0.0.1	ARP	44 10.0.0.3 is at 12:19:d6:93:ea:60
740	16.351223631	127.0.0.1	127.0.0.1	OpenFl_	152 Type: OFPT_PACKET_IN
741	16.351335774	127.0.0.1	127.0.0.1	OpenFl_	212 Type: OFPT_FLOW_MOD
742	16.351359873	127.0.0.1	127.0.0.1	OpenFl_	150 Type: OFPT_PACKET_OUT
743	16.352248134	127.0.0.1	127.0.0.1	TCP	68 44758 -> 6653 [ACK] Seq=2829 Ack=2389 Win=54784 Len=0 TSval=2965534448 TSecr=2965534447
744	16.352272157	12:19:d6:93:ea:60	127.0.0.1	ARP	44 10.0.0.3 is at 12:19:d6:93:ea:60
745	16.352305969	10.0.0.1	10.0.0.3	ICMP	100 Echo (ping) request id=0x567f, seq=1/256, ttl=64 (no response found!)
746	16.355327892	127.0.0.1	127.0.0.1	OpenFl_	208 Type: OFPT_PACKET_IN
747	16.355449796	127.0.0.1	127.0.0.1	OpenFl_	228 Type: OFPT_FLOW_MOD
748	16.355472435	127.0.0.1	127.0.0.1	OpenFl_	206 Type: OFPT_PACKET_OUT
749	16.355858175	127.0.0.1	127.0.0.1	TCP	68 44758 -> 6653 [ACK] Seq=2969 Ack=2679 Win=56832 Len=0 TSval=2965534452 TSecr=2965534451
750	16.355885456	10.0.0.1	10.0.0.3	ICMP	100 Echo (ping) request id=0x567f, seq=1/256, ttl=64 (reply in 751)
751	16.355909924	10.0.0.3	10.0.0.1	ICMP	100 Echo (ping) reply id=0x567f, seq=1/256, ttl=64 (request in 750)
752	16.356195432	127.0.0.1	127.0.0.1	OpenFl_	208 Type: OFPT_PACKET_IN
753	16.356198612	127.0.0.1	127.0.0.1	OpenFl_	228 Type: OFPT_FLOW_MOD
754	16.356211165	127.0.0.1	127.0.0.1	OpenFl_	206 Type: OFPT_PACKET_OUT
755	16.356607917	127.0.0.1	127.0.0.1	TCP	68 44758 -> 6653 [ACK] Seq=3109 Ack=2969 Win=58880 Len=0 TSval=2965534452 TSecr=2965534452
756	16.356622773	10.0.0.3	10.0.0.1	ICMP	100 Echo (ping) reply id=0x567f, seq=1/256, ttl=64
757	16.455701354	172.16.115.128	224.0.0.251	MDNS	121 Standard query 0x0000 PTR _apple-mobdev._tcp.local, "QM" question PTR 4c32057a._sub._apple-mobdev._tcp.local TXT TXT _apple-flush-SDU
758	16.461987370	172.16.106.17	224.0.0.251	MDNS	121 Standard query response 0x0000 PTR _apple-mobdev._tcp.local, "QM" question PTR 4c32057a._sub._apple-mobdev._tcp.local TXT TXT _apple-flush-SDU

针对以上实验截图解释 H1 ping h3 的过程：

**734 行：**首先 H1 发送一个 APR Request 数据包给交换机，想获取 H3 的 MAC 地址。

**735 行：**当交换机 S1 收到 ARP Request 数据包时，对初始的流表项（当控制器链接交换机后，会对交换机下发初始流表，且此流表优先级为 0 最低，当数据包没有对应的流表项进行匹配时，匹配此流表，将其发送至控制器）进行匹配，并通过 Packet-in 方式发送给控制器。

**736 行：**控制器对接收到的 packet-in 数据包在 mac-to-port 中查找是否存在对应的 M A C 地址和端口。发现不存在，发送 packet-out,对除源端口之外的所有端口进行泛洪处理，并将 H 1 的对应 M A C 地址和端口信息存储到 mac-to-port 表中，这个过程控制器并不下发流表。

**739 行：**当 H3 收到泛洪信息后，将会回复一条 ARP Reply 给交换机，

**740 行：**交换机中没有可以匹配的流表项，所以 ARP Reply 也会执行 packet-in 发送到控制器。

**742 行：**在控制器的 mac-to-port 中存在了 H1 的信息，所以控制器会通过 packet-out 直接发送到端口 1。

**741 行：**控制器下发关于入端口 3，目的地址 H1，输出端口 1 的 ARP 流表项，并且记录 H3 相应的信息到 mac-to-port 表中。

**745 行：**H1 收到 H3 的回应后，发送 ICMP 报文给交换机。

**746 行：**交换机中并不存在流表项来处理 ICMP 数据包，会执行 packet-in 发送到控制器

**748 行：**控制器中的 mac-to-port 表中存在了 H3 的相应信息，所以通过 packet-out 直接发送到端口。

**747 行：**控制器下发关于入端口为 1，目的地址 H3，输出端口 3 的 ICMP 流表项，增加交换机流表中的流表项。

**756 行：**主机 h3 向接收到 h1 发送的 ICMP 包，并且进行回应，发送一个 ICMP 响应包。

**752 行：**交换机接收到这个包之后没有匹配到流表项，就向上通过 Packet-in 方式发送给控制器。

**753 行：**控制器接收到包之后，直到主机 1 在端口 1 所在网络中，所以下发关于入端口 3，目的地址 H1，输出端口 1 的 ICMP 流表项。

**754 行：**控制器通过 packet-out 让交换机把包直接发送到端口 1，h1 收到 ICMP 响应包完成一次 ping 的过程。

解释为什么 ping 的第二次比第一次快很多？

```
*** Starting CLI:  
mininet> h1 ping -c 2 h3  
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.  
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=12.0 ms  
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.320 ms
```

如实验截图所示，第二次 ping 时间是 0.32ms，第一次 ping 的时间是 12ms，远远大于第二次的时间。

主要因为两点原因：

首先在第一次发送 ICMP 请求包的时候 h1 不知道 h3 主机的 MAC 地址，所以需要进行 ARP 的请求，ARP 包在交换机处也需要进行 packet in 和 packet out 操作，这些操作是比较耗费时间的。

其次经过第一次 ping,交换机的流表中已经存在了相应的流表项，所以第二次 ping 时交换机不需要向控制器发送 packet in 包，直接匹配流表项，然后按照相应的 action 进行转发即可，这样也极大减少了传送时间。