Lab2-OpenFlow

（一）自学习交换机

（1）工作流程

①控制器为每个交换机维护一张 mac-port 映射表；

②控制器收到 packet_in 消息后，解析其中携带的数据包；

③控制器学习 src_mac-in_port 映射；

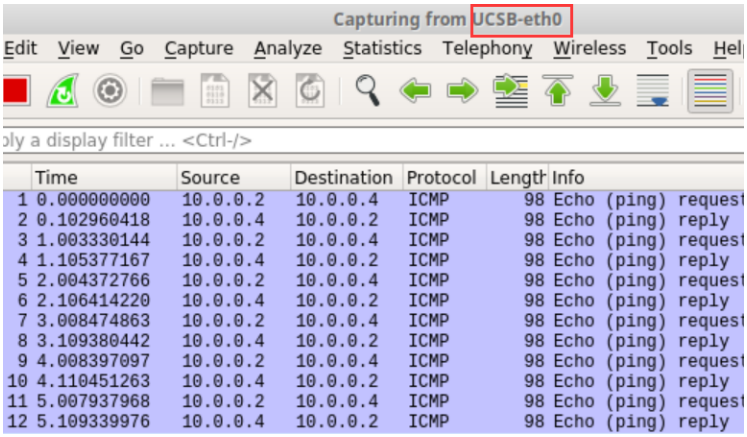
④控制器查询 dst_mac，如果未学习，则洪泛数据包；如果已学习，则向指定端口转发数据包（packet_out），并向交换机下发流表项（flow_mod），指导交换机转发同类型的数据包。

（2）UCLA ping UTAH

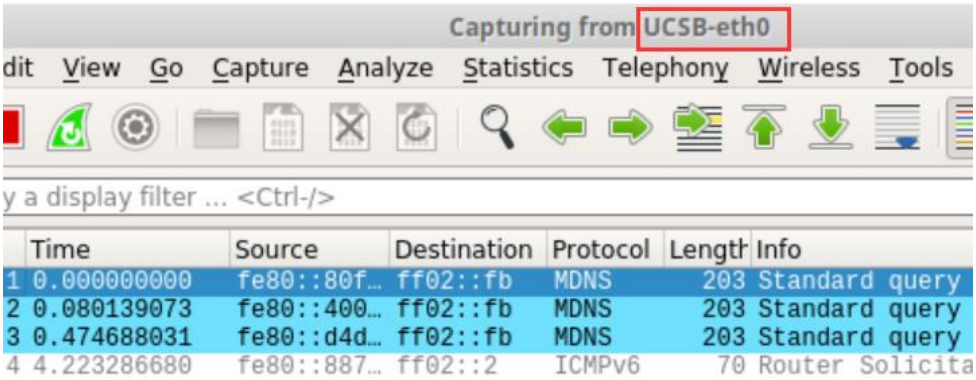
```
SRI SRI-eth0:s1-eth1
UCLA UCLA-eth0:s4-eth1
UCSB UCSB-eth0:s3-eth1
UTAH UTAH-eth0:s2-eth1
*** Starting CLI:
mininet> xterm UCSB
mininet> xterm SRI
mininet> UCLA ping UTAH
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=269 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=127 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=127 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=64 time=127 ms
64 bytes from 10.0.0.4: icmp_seq=5 ttl=64 time=126 ms
64 bytes from 10.0.0.4: icmp_seq=6 ttl=64 time=127 ms
64 bytes from 10.0.0.4: icmp_seq=7 ttl=64 time=129 ms
```

由于在构建拓扑的过程中，为交换机之间的链路指定了时延，因此这里的时延都在 100ms 以上。其中，第一次时延较大是控制器与交换机之间的转发数据包及流表项导致的，后续的数据包则通过匹配流表项后直接转发。

①控制器未开启 mac 自学习，只洪泛数据包时，UCSB 也能收到 UCLA ping UTAH 的 ICMP 报文：

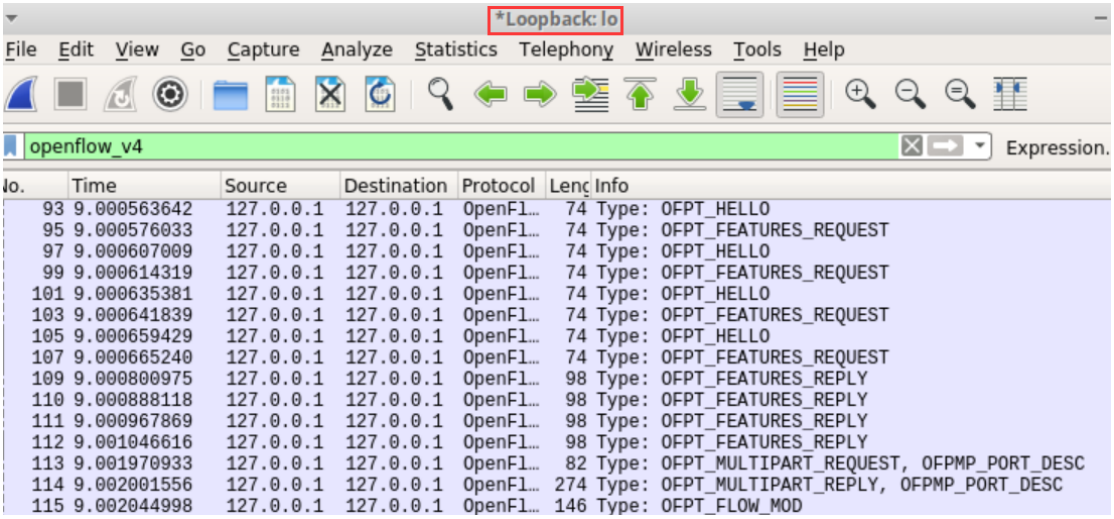


②控制器开启 mac 自学习后，UCSB 不再收到相关数据包：



③OpenFlow 协议分析：

Ryu 控制器与交换机建立连接的过程：



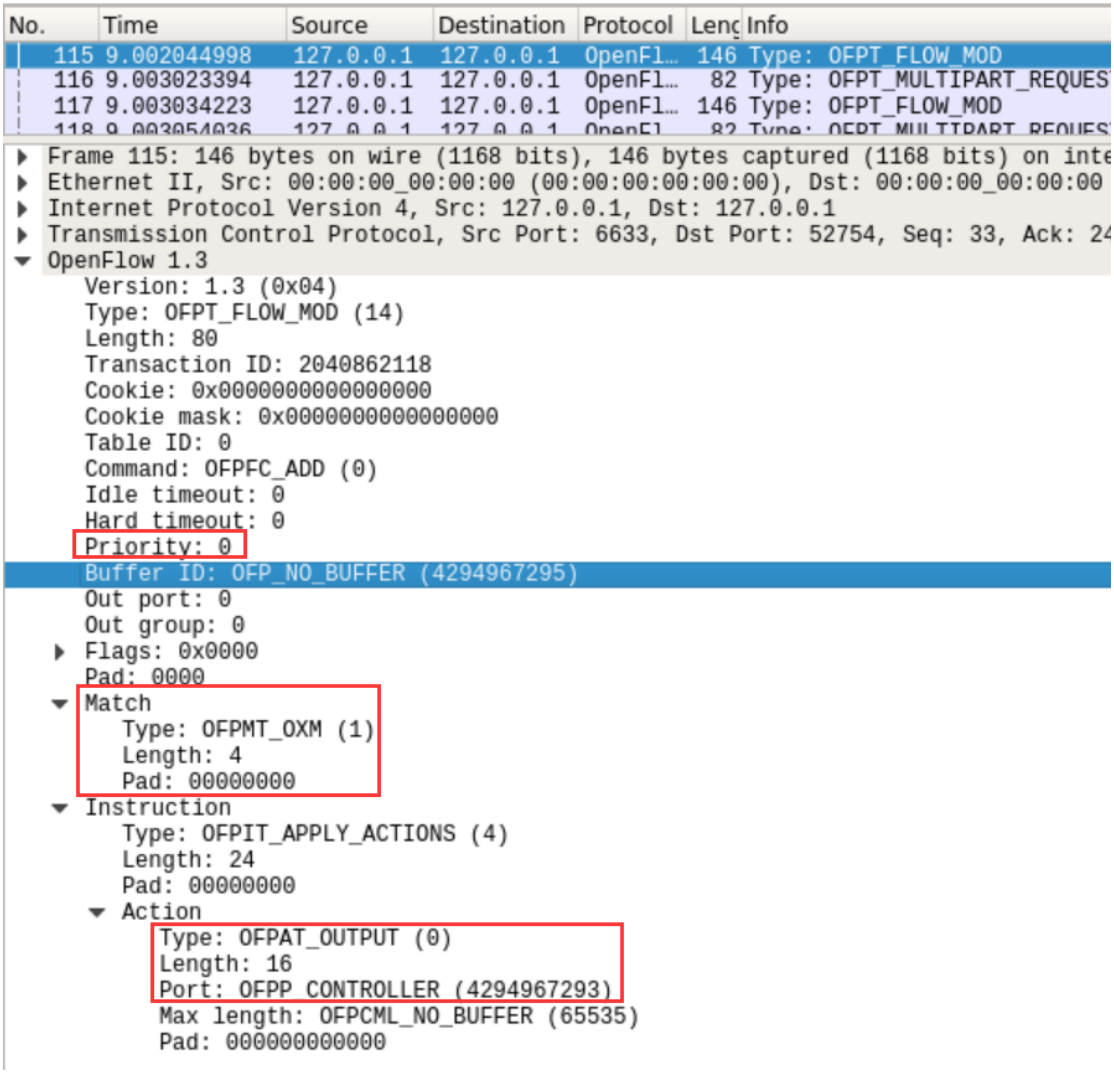
OFPT_HELLO 消息用于交换 OpenFlow 协议版本信息，以便交换机和控制器确定它们都支持的 OpenFlow 协议版本。该消息可以由交换机和控制器发送，在 TCP/TLS 连接建立时发送，是 OpenFlow 通信渠道建立过程的一部分。

OFPT_FEATURES_REQUEST 是一种 OpenFlow 消息，用于向交换机请求相关信息，比如交换机的功能、可用端口和流表等信息。该请求消息可以由控制器向交换机发送，交换机在接收到请求后，会相应地返回 OFPT_FEATURES_REPLY 消息，其中包含了与交换机相关的详细信息，比如交换机支持的 OpenFlow 版本、交换机所支持的协议特性以及交换机端口的数量、类型和状态等。

OFPT_MULTIPART_REQUEST 消息用于向交换机发送一组相关联的请求，以查询交换机上的状态和属性信息，例如流表、队列、端口统计数据等。每个请求都会被分配一个唯一的 ID 号，方便控制器和交换机对相应的请求和响应进行匹配。OFPT_MULTIPART_REPLY 消息则用

于向控制器返回对应的多部分信息请求结果，包含了请求中所描述的一系列交换机状态或配置信息。响应消息包含了与请求消息相对应的请求 ID，以便控制器能够识别并解析响应消息。当 type 字段取值为 OFPMP_PORT_DESC 时，表示该消息携带的是与端口描述相关的信息。通过 OFPT_MULTIPART_REQUEST 和 OFPT_MULTIPART_REPLY 消息交互，控制器可以获取关于交换机的详细状态信息，并根据这些信息来配置流表、调整网络拓扑结构等，从而实现更高效、安全、可靠的网络通信。

控制器下发 table-miss 流表项：



当交换机与控制器建立连接后，控制器下发一条默认流表项，优先级为 0（最低），匹配域为空（表示可以匹配任何数据包），执行动作为将数据包转发给控制器

测试 Ryu 控制器与交换机的连通性

```
190 13.999841090  127.0.0.1  127.0.0.1  OpenFl…  74 Type: OFPT_ECHO_REQUEST
191 14.000205783  127.0.0.1  127.0.0.1  OpenFl…  74 Type: OFPT_ECHO_REPLY
193 14.999513549  127.0.0.1  127.0.0.1  OpenFl…  74 Type: OFPT_ECHO_REQUEST
194 14.999573088  127.0.0.1  127.0.0.1  OpenFl…  74 Type: OFPT_ECHO_REQUEST
195 14.999603783  127.0.0.1  127.0.0.1  OpenFl…  74 Type: OFPT_ECHO_REQUEST
196 15.000929522  127.0.0.1  127.0.0.1  OpenFl…  74 Type: OFPT_ECHO_REPLY
198 15.001312895  127.0.0.1  127.0.0.1  OpenFl…  74 Type: OFPT_ECHO_REPLY
200 15.001618438  127.0.0.1  127.0.0.1  OpenFl…  74 Type: OFPT_ECHO_REPLY
```

OFPT_ECHO_REQUEST 是 OpenFlow 协议中的一种消息类型，用于测试控制器和交换机之间的连接状态。当控制器接收到 OFPT_ECHO_REQUEST 消息时，应该立即返回一个相同数据的 OFPT_ECHO_REPLY 消息给交换机。OFPT_ECHO_REQUEST 消息包含一个 OpenFlow 头部和一个任意长度的未定义数据字段。数据字段可以填充时间戳、测量带宽等信息，也可以为空以检查连接的活性。OFPT_ECHO_REQUEST 消息的作用类似于网络诊断工具中的“ping”命令，可用来快速检查网络的连通性和延迟情况。

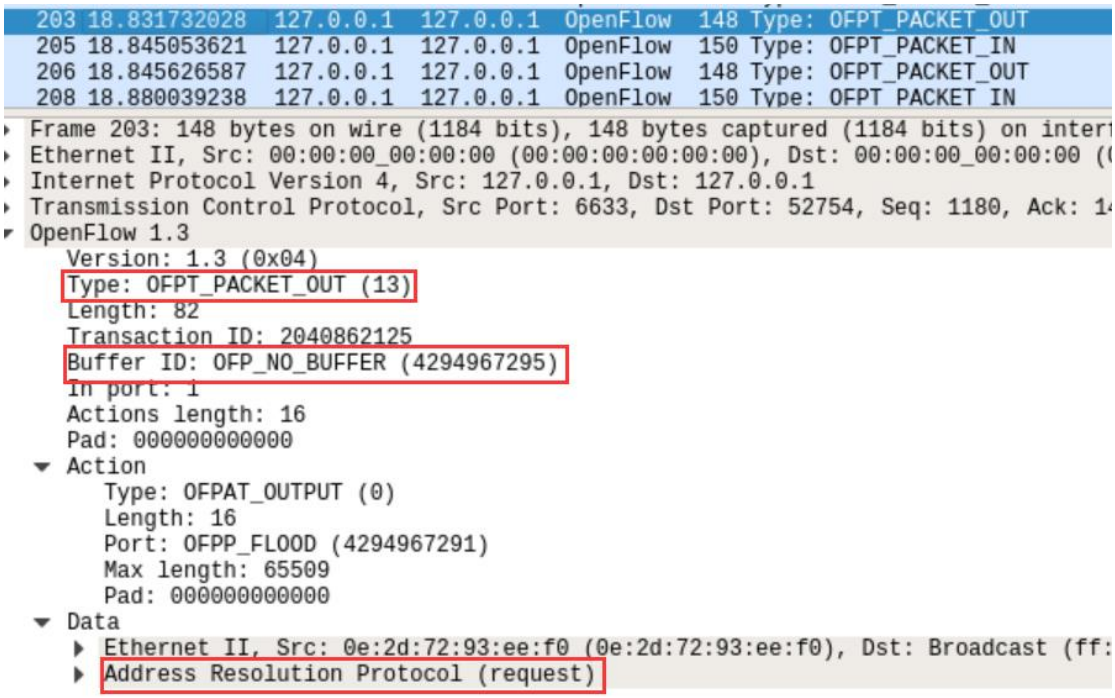
ARP 请求报文：

```
202 18.831127932  127.0.0.1  127.0.0.1  OpenFlow  150 Type: OFPT_PACKET_IN
203 18.831732028  127.0.0.1  127.0.0.1  OpenFlow  148 Type: OFPT_PACKET_OUT
205 18.845053621  127.0.0.1  127.0.0.1  OpenFlow  150 Type: OFPT_PACKET_IN
206 18.845626587  127.0.0.1  127.0.0.1  OpenFlow  148 Type: OFPT_PACKET_OUT
208 18.880039238  127.0.0.1  127.0.0.1  OpenFlow  150 Type: OFPT_PACKET_IN

▸ Frame 202: 150 bytes on wire (1200 bits), 150 bytes captured (1200 bits) on inte
▸ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▸ Transmission Control Protocol, Src Port: 52754, Dst Port: 6633, Seq: 1328, Ack:
▾ OpenFlow 1.3
    Version: 1.3 (0x04)
    Type: OFPT_PACKET_IN (10)
    Length: 84
    Transaction ID: 0
    Buffer ID: OFP_NO_BUFFER (4294967295)
    Total length: 42
    Reason: OFPR_NO_MATCH (0)
    Table ID: 0
    Cookie: 0x0000000000000000
  ▾ Match
      Type: OFPMT_OXM (1)
      Length: 12
    ▸ OXM field
      Pad: 00000000
    Pad: 0000
  ▾ Data
    ▸ Ethernet II, Src: 0e:2d:72:93:ee:f0 (0e:2d:72:93:ee:f0), Dst: Broadcast (f
    ▸ Address Resolution Protocol (request)
```

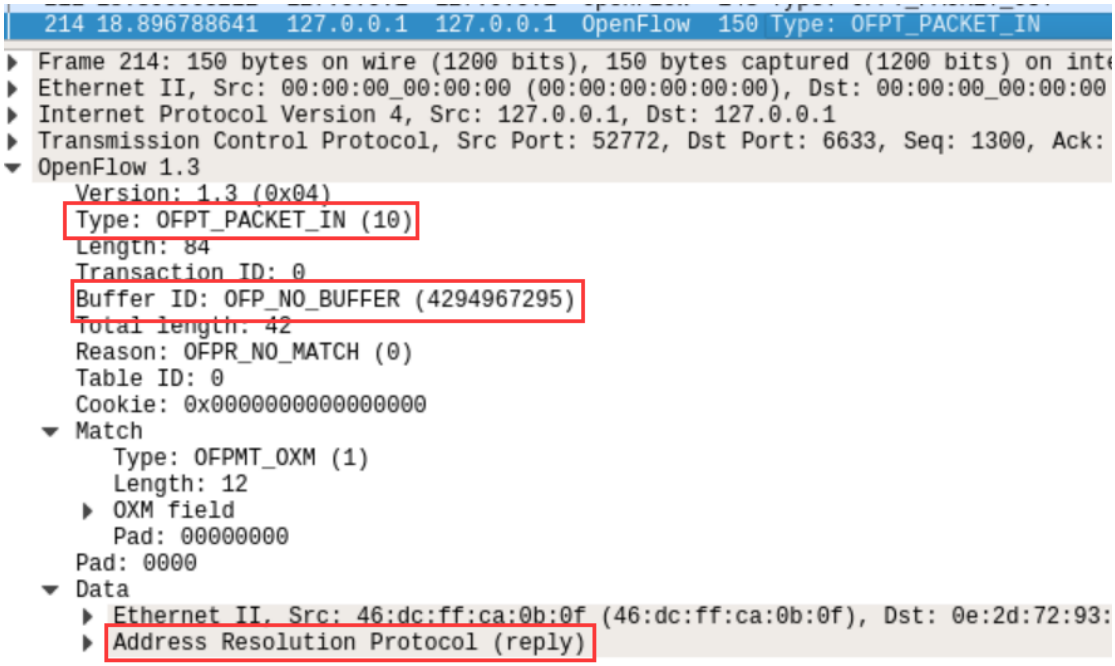
当 UCLA ping UTAH 时，UCLA 首先广播 ARP 请求报文以获得 UTAH 的 mac 地址。交换机在收到该报文后，匹配 table-miss 流表项，转发给控制器，因此该报文类型为 OFPT_PACKET_IN。此外，由于交换机不缓存数据包，因此控制器收到的 PACKET_IN 报文中含有 ARP 请求的数据包。

泛洪 ARP 请求报文：



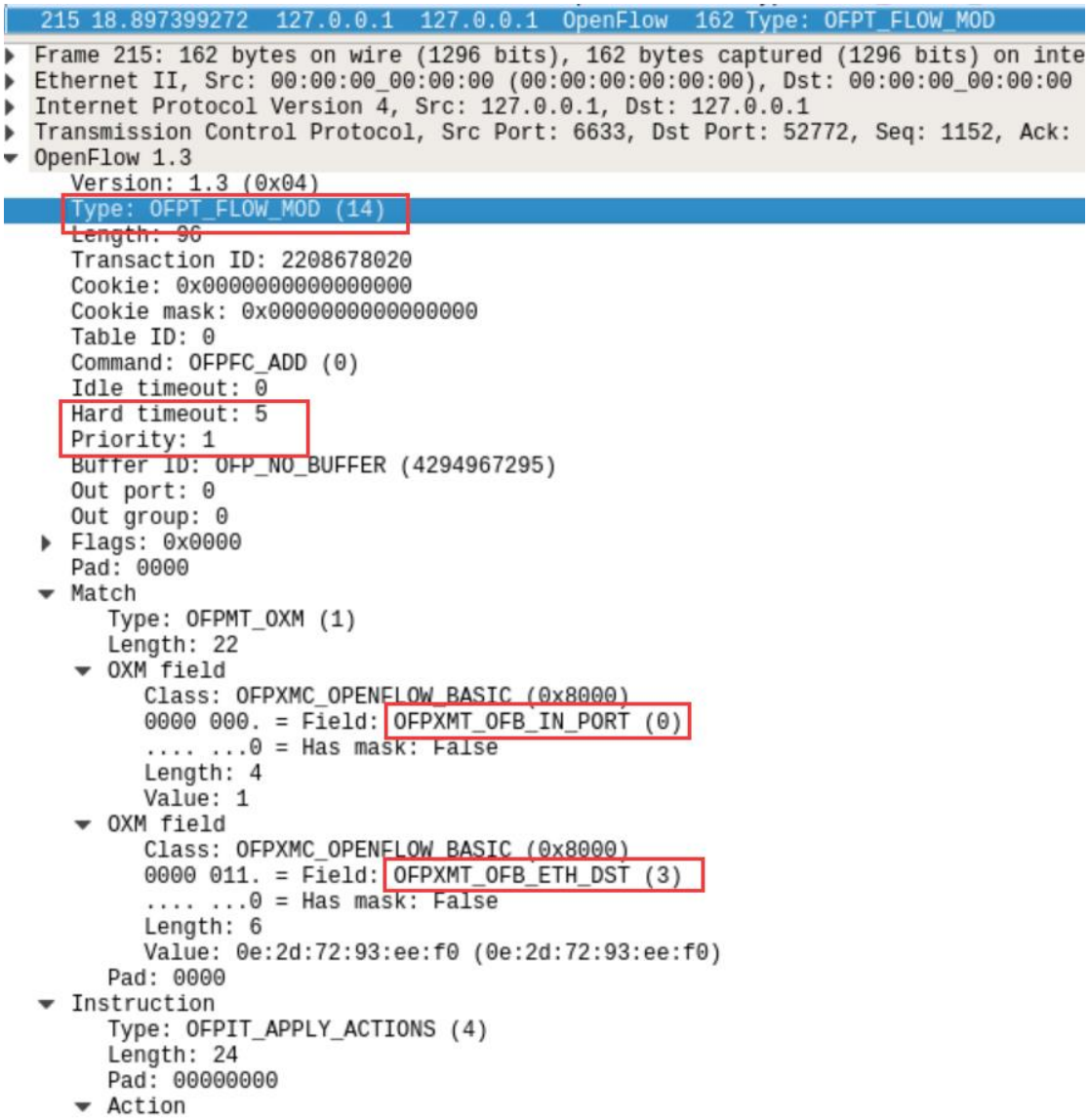
由于 ARP 报文的目的 mac 地址为广播地址，而控制器的 mac-port 表中未学习到该地址，因此，控制器在下发 ARP 请求报文时让交换机执行泛洪。此外，控制器通过该报文能够学习到 UCLA 的 mac 地址以及对应交换机的端口号。

ARP 应答报文：

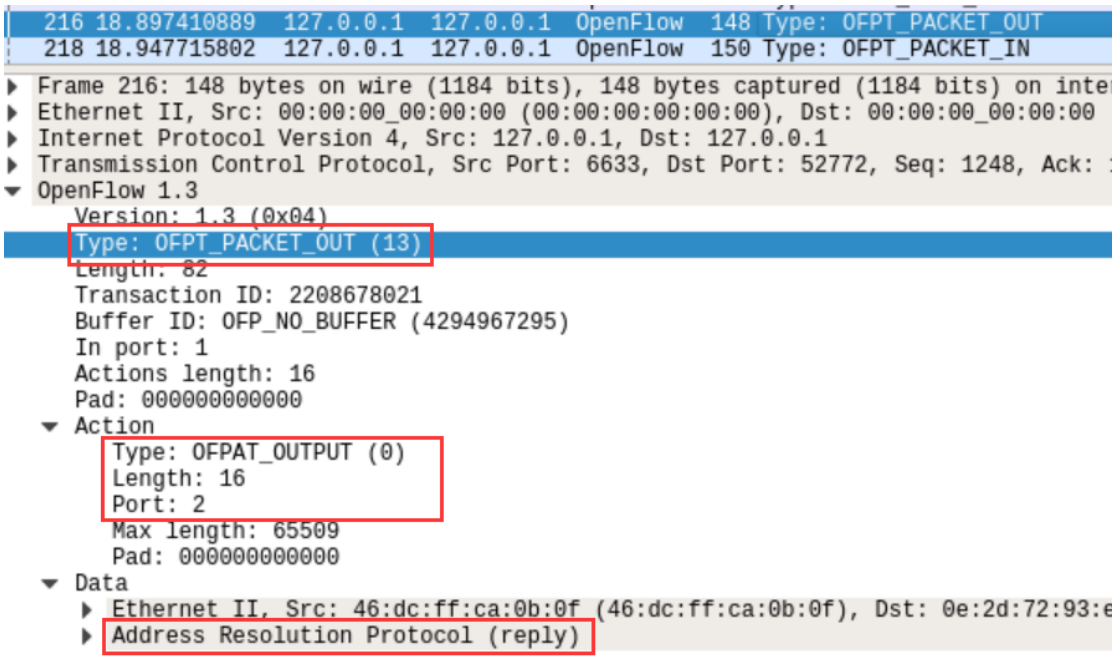


同样的，交换机在收到 UTAH 的 ARP 应答报文后，匹配 table-miss 流表项，转发给控制器，并在 PACKET_IN 中携带了 ARP 请求报文

控制器下发流表项：



由于控制器已经学习了 ARP 应答报文的目的 mac 地址（在之前收到 ARP 请求报文时），因此，控制器下发流表项，指定匹配域为入端口与目的 mac 地址，执行动作为转发到指定的端口，硬超时为 5 秒。注意，这里的优先级为 1（大于 table-miss，避免下次继续匹配 table-miss）。同时，控制器还将收到的 ARP 应答报文转发给交换机并执行刚才指定的动作，如下图所示。



交换机在收到控制器发来的 PACKET_OUT 数据包后，将其中的 ARP 应答报文转发到指定的端口，从而 UCLA 便获得了 UTAH 的 mac 地址。

之后，UCLA 与 UTAH 之间的 ICMP 报文传输过程与此类似，这里不再赘述。

（二）处理环路广播

（1）工作流程

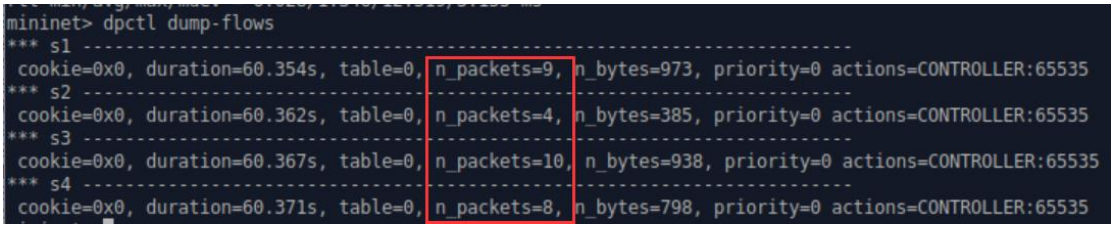
当序号为 dpid 的交换机从 in_port 第一次收到某个 src_mac 主机发出的、询问 dst_ip 的广播 ARP 请求数据包时，控制器将会记录一个映射(dpid,src_mac,dst_ip)->in_port。下一次该交换机收到同一(src_mac,dst_ip)但 in_port 不同的 ARP 请求数据包时直接丢弃，否则泛洪。

（2）UCLA ping UTAH

```
mininet> UCLA ping UCSB
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=8.33 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.131 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.048 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=0.085 ms
64 bytes from 10.0.0.3: icmp_seq=5 ttl=64 time=0.035 ms
64 bytes from 10.0.0.3: icmp_seq=6 ttl=64 time=12.5 ms
```

与（一）类似，由于在构建拓扑的过程中，交换机之间的链路没有指定时延，因此这里的时延与（一）相比都较小。同样，第一次时延较大是控制器与交换机之间的转发数据包及流表项导致的，后续的数据包则通过匹配流表项后直接转发。

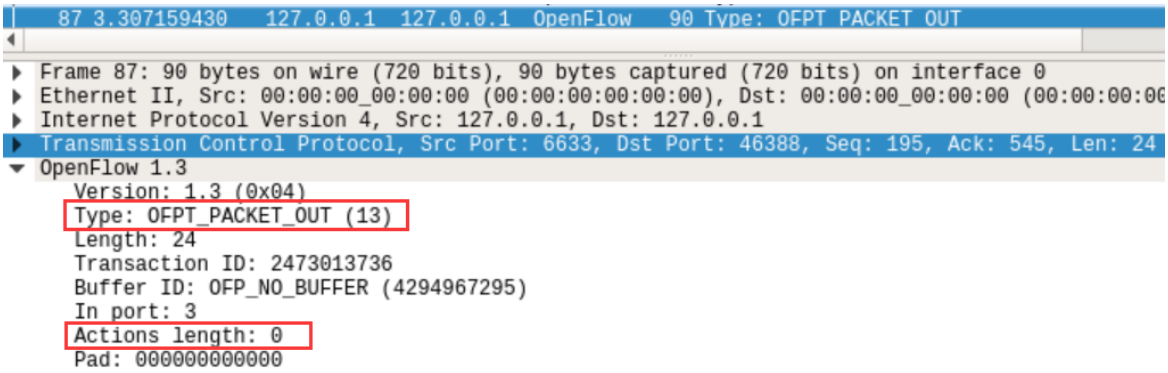
流表项：



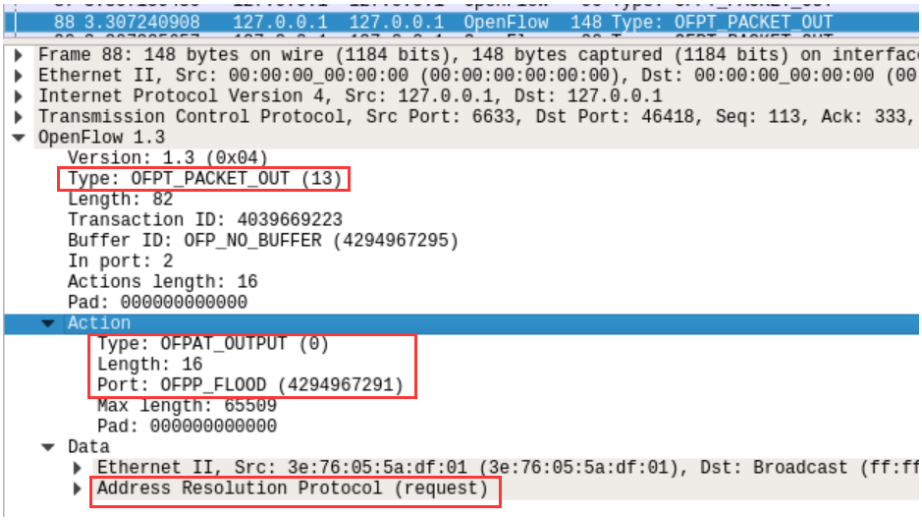
解决 ARP 数据包在环状拓扑的洪泛问题后，UCLA 与 UTAH 之间可以 ping 通，且流表项的匹配次数明显减少。此外，多次重复实验发现，同一交换机中某一流表项的匹配次数总比另一条流表项多 1。这里多出来的一次匹配过程应该是 ARP 应答报文，其余的则是 ICMP 报文的匹配。因此，可以推测匹配次数多的流表项 output 的端口是 ping 过程中 UCLA 的接收端口，而匹配次数少的流表项 output 的端口则是发送端口。

OpenFlow 协议分析：

当控制器收到同一(dpid,src_mac,dst_ip)但 in_port 不同的 ARP 请求数据包时，表明出现了环路。因此，控制器发出的 PACKET_OUT 数据包中，未指明 Actions（表示丢弃报文），且也没有携带 ARP 请求报文：



当控制器收到同一(dpid,src_mac,dst_ip)且 in_port 相同的 ARP 请求数据包、或者是第一次收到(dpid,src_mac,dst_ip) 的 ARP 请求数据包时，控制器将数据包转发给交换机并让交换机执行洪范：



（4）解决环路广播问题的其它方案

由控制器完成广播 ARP 请求报文的转发：控制器记录每个主机所连接的交换机及其端口号，让所有的 ARP 请求报文都触发 PACKET_IN，再由控制器直接将 ARP 请求报文转发到交换机与主机相连的端口。

存在的困难：控制器如何获取每个主机所连接的交换机及其端口号。尝试使用 get_host 与 get_link 函数，但未取得预期的结果