**《高级计算机网络》**

**课程报告**

**题目： 基于SDN网络理解OpenFlow**

**协议机制**

**校区： 主校区**

**专业： 电子信息（计算机技术）**

**班级： 3班**

**学号： 202222172013465**

**姓名： 韩梅**

**郑州大学计算机与人工智能学院**

# 《高级计算机网络》课程报告

**基于SDN网络理解OpenFlow 协议机制**

**目录**

[《高级计算机网络》课程报告 2](#_Toc117847903)

[一、实践目的 1](#_Toc117847904)

[二、实践平台与环境 1](#_Toc117847905)

[三、设计思想与网络拓扑结构 1](#_Toc117847906)

[四、实践步骤与结果记录 1](#_Toc117847907)

[五、数据包分析 2](#_Toc117847908)

[六、遇到的问题及解决方法 2](#_Toc117847909)

[七、收获与体会 2](#_Toc117847910)

**目录说明：为了报告的规范性，报告内容可按层次结构设定三级标题，自动生成索引目录。如果报告内容修改，则更新目录，保持章节内容页面与目录页码的对应！**

**三级标题示例：**

**一、XXX（一级标题）**

**1. （二级标题）**

**2.**

**…**

**(1) （三级标题）**

**(2)**

**…**

**二、XXX（一级标题）**

**…**

# 实践目的

1、 在VMware虚拟机上使用Ubuntu环境利用mininet及ryu设计网络场景，构建简单的SDN网络拓扑结构，用wireshark软件对模拟环境的网络进行抓包，主要对TCP、openflow、ARP等SDN网络结构的数据包进行分析，分析控制器与OF交换机的交互报文；

2、理解SDN网络结构以及SDN和传统网络结构的。掌握SDN结构的主要组成部分；

3、 熟练掌握南向接口协议OpenFlow，并且对OpenFlow的三种类型报文：controller-to-switch、同步asynchronous、对称symmetric，并对这三类报文进行分析；

4、 了解构建实验环境的软件；

# 实践平台与环境

1、虚拟机：VMware；

2、系统：ubuntu 16.04；

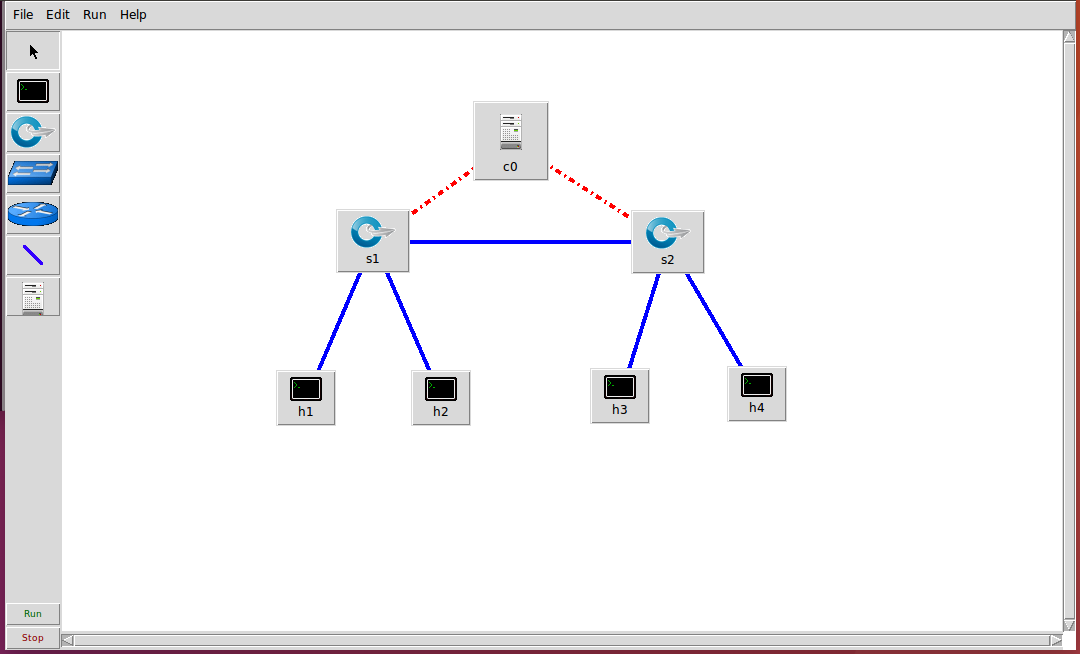
3、软件：Ryu v4.34和mininet 2.3.0；

4、使用miniedit创建SDN网络拓扑；

5、Wireshark进行抓包；

# 设计思想与网络拓扑结构

本文设计的网络拓扑中有一台控制器c0，地址为127.0.0.1，并且为远程控制器；两台交换机，分别为s1和s2，pathdata分别为0000000000000001和0000000000000002，其中s1连接h1和h2两台主机，地址分别为10.0.0.1和10.0.0.2，s2连接h3，h4两台主机，地址分别为10.0.0.3，10.0.0.4。本次实验主要是通过h1 ping h4来理解openflow协议，了解在网络连通过程中都有哪些报文出现，各个网络部件是如何进行通信的。

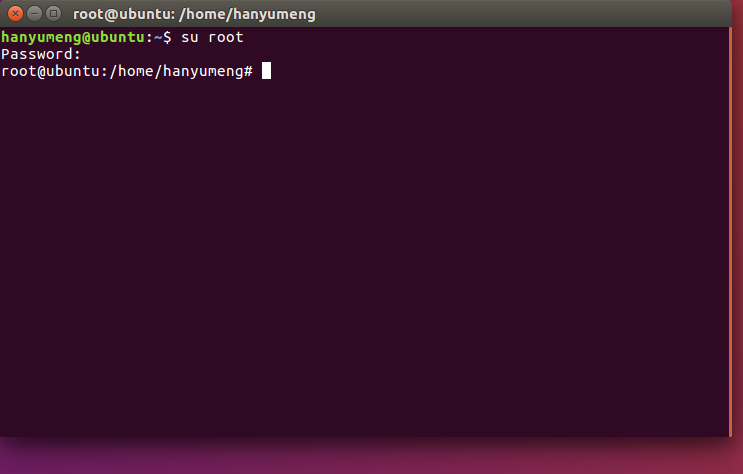
图 3-１网络拓扑结构

# 四、实践步骤与结果记录

**要求：根据实践内容和步骤，认真完成。采取截图、拍照等形式记录自己的实践操作步骤和结果。**

1. 使用mininet构建SDN网络拓扑结构。
2. 登陆进入root，如图4-1所示

图 4-１ 进入root

1. 打开miniedit进行可视化建立拓扑结构

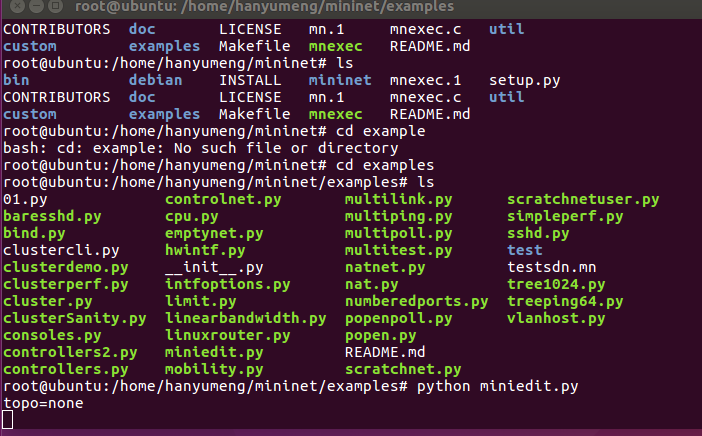


图 4-2 打开mininet可视化界面

1. 进入可视化界面，构建网络拓扑结构。

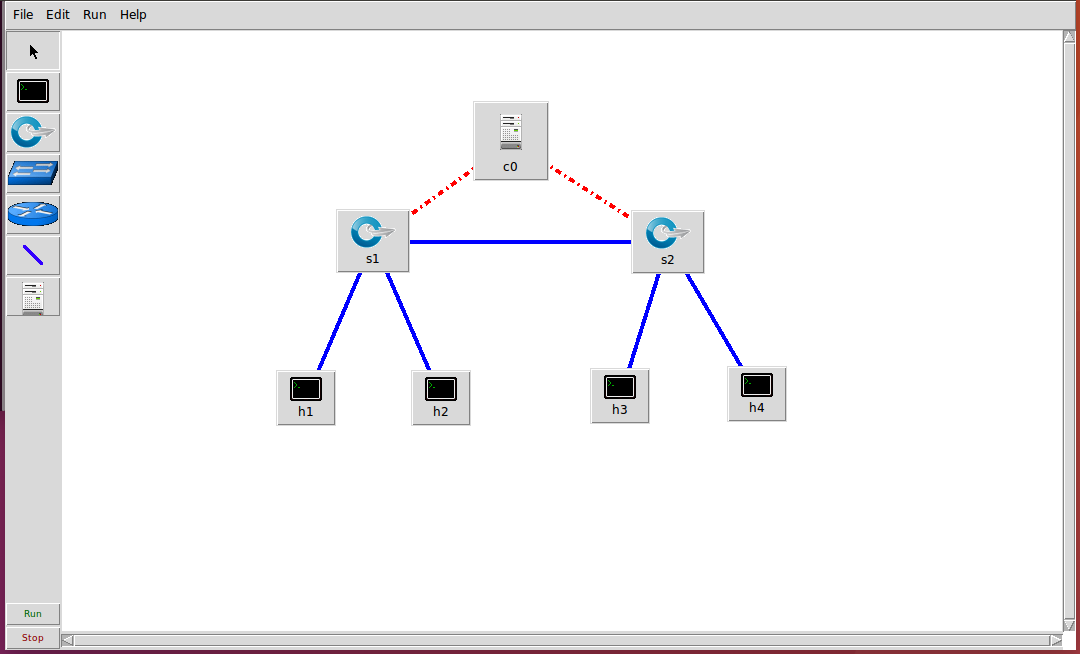


图 4-3 可视化界面

1. 基础设置

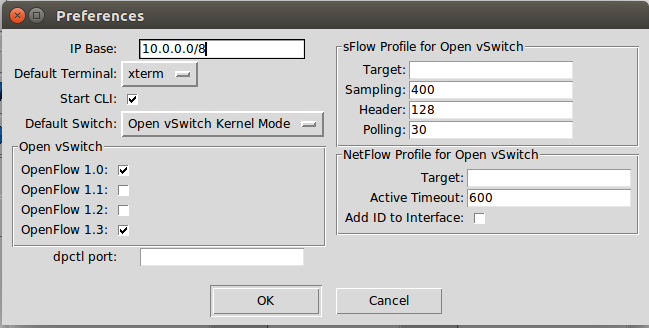


图 4-4 基础设置

1. 赋予控制器IP地址，控制器c0的IP地址为系统默认127.0.0.1。

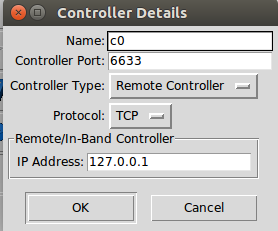


图 4-5 控制器设置

1. 交换机s1的DPID为0000000000000001

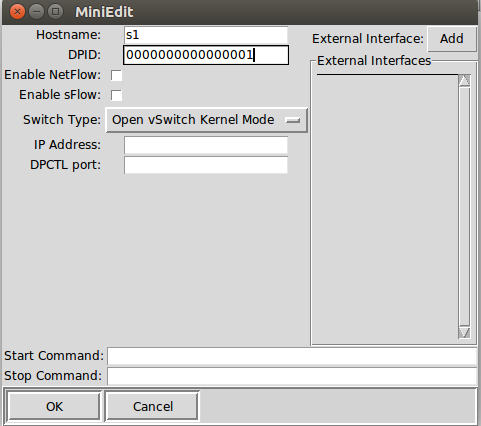


图 4-6交换机s1设置

1. 交换机s1的DPID为0000000000000002

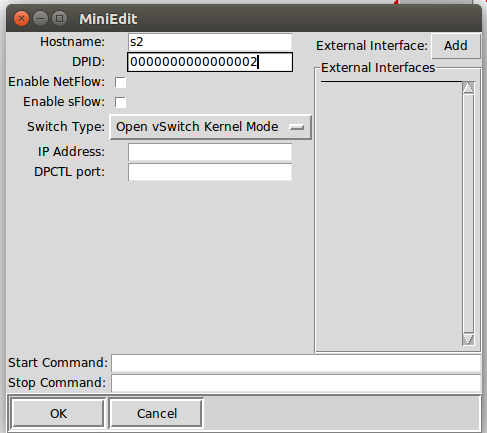


图 4-7交换机s2设置

1. 主机的IP地址分别为h1：10.0.0.1；h2：10.0.0.2；h3：10.0.0.3；h4：10.0.0.4.

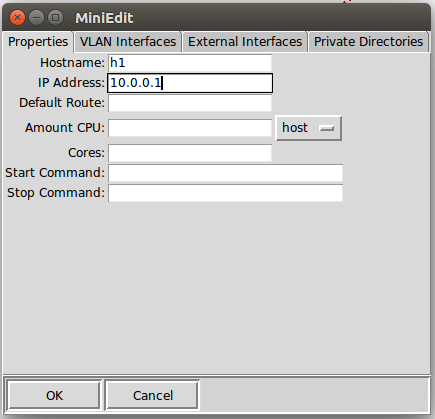


图 4-8 h1IP地址

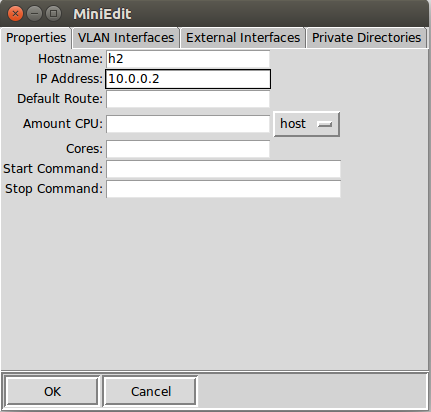


图 4-9 h2IP地址

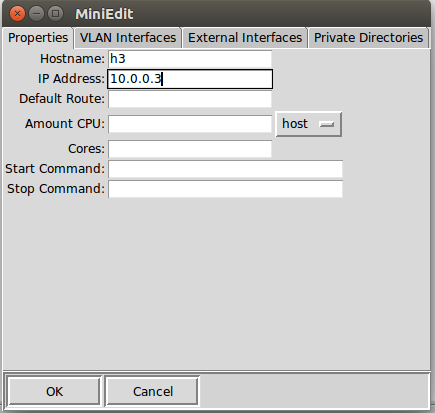


图 4-10 h3IP地址

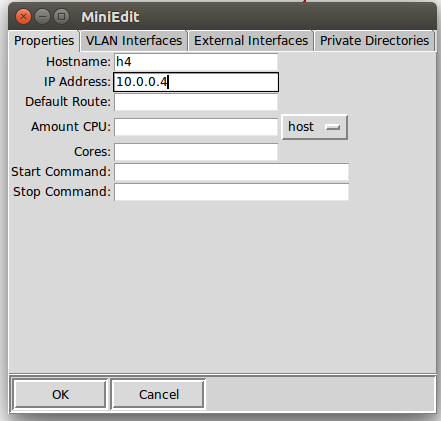


图 4-11 h4IP地址

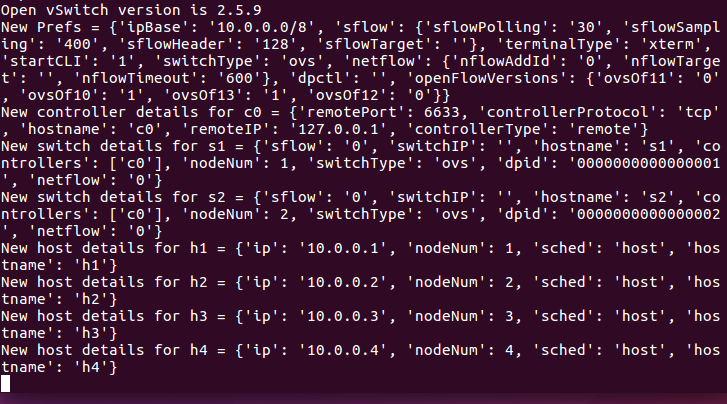
1. 配置信息自动录入到mininet的后台，此为拓扑结构的基本信息。

图 4-12 mininet后台

1. 将拓扑结构保存为python文件，并命名为003.py

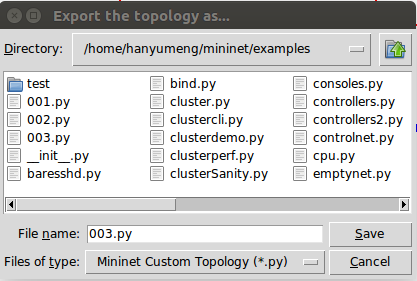
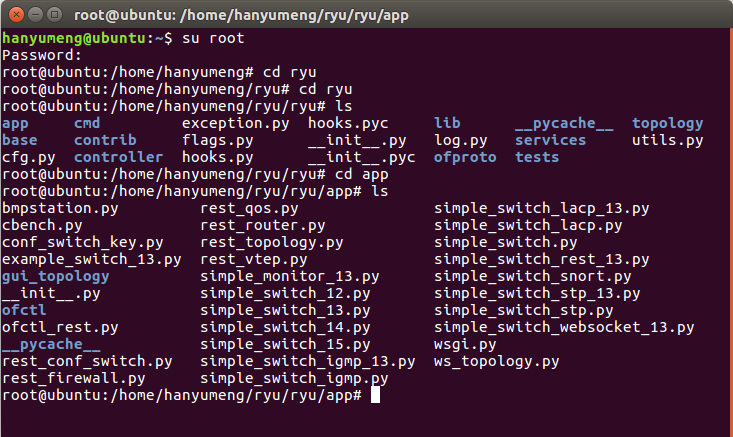


图 4-13 保存python文件

1. 运行网络环境

（1）打开ryu控制器。

图 4-14 打开ryu控制器

1. 在mininet测试网络连通性。

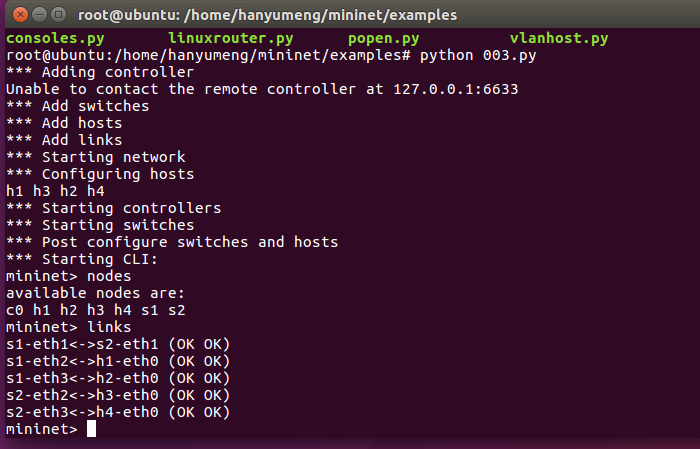


图 4-15 mininet测试连通性

1. 打开wireshark。

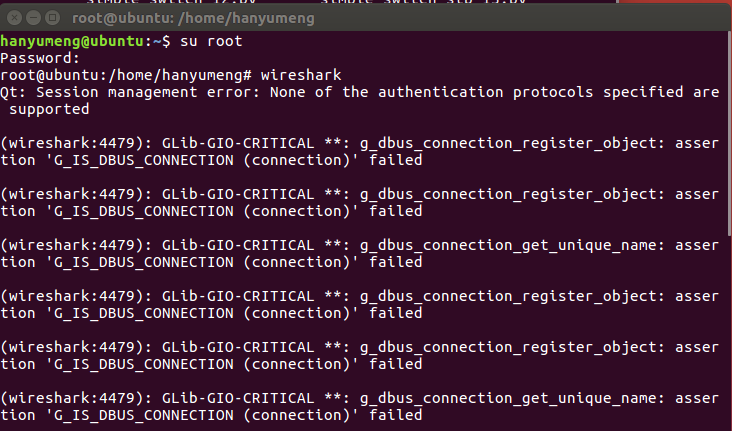


图 4-16 wireshark

1. h1 ping h4

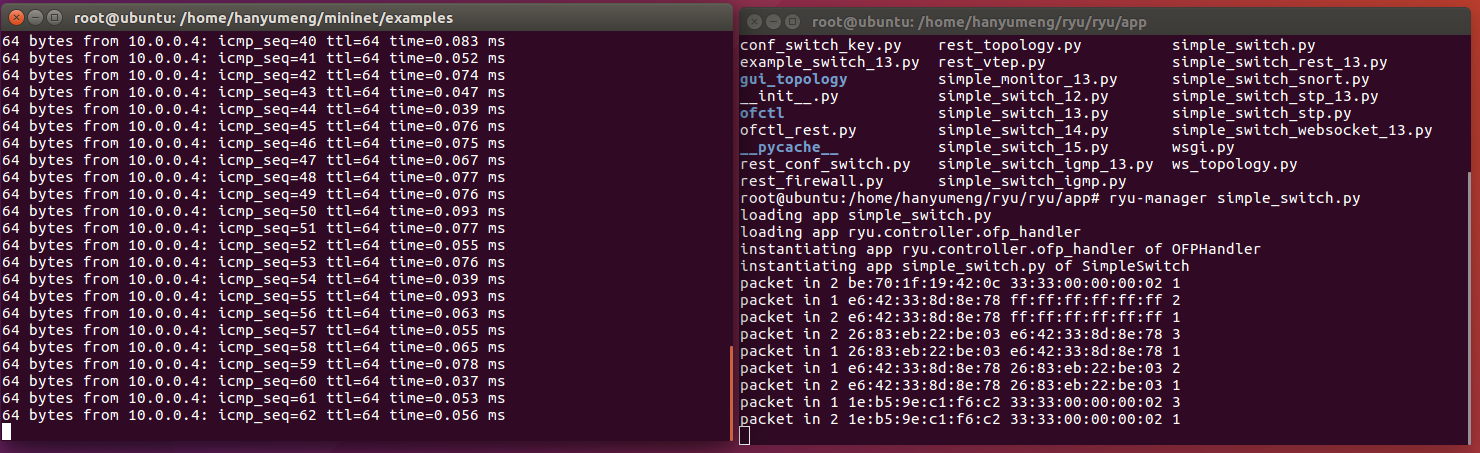


图 4-17 网络连通性

1. 使用wireshark抓包

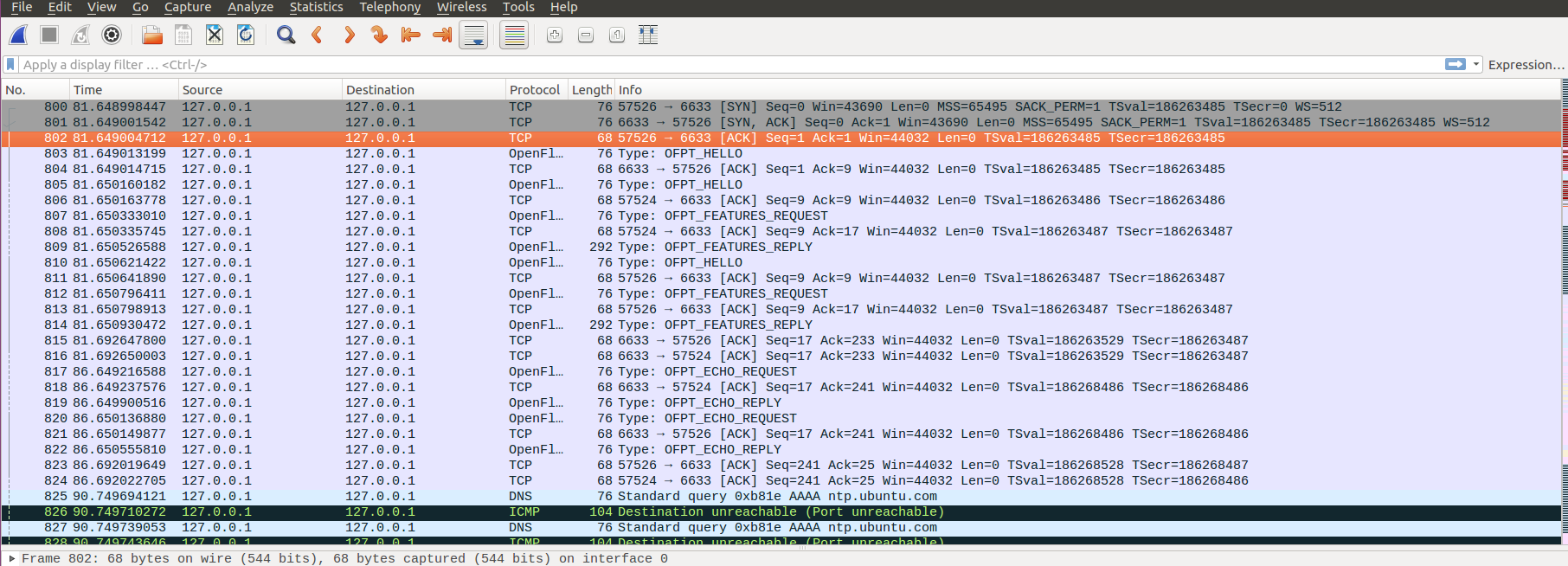


图 4-18 wireshark抓包

# 五、数据包分析

1、建立TCP连接，控制器和交换机之间要进行“三次握手”才能进行openflow协议进行信息交换。

（1）交换机port57524与控制器port6633“三次握手”。

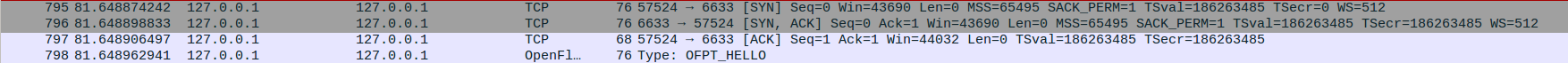


图 5-1交换机s1与控制器“三次握手”

（2）交换机port57526与控制器port6633“三次握手”。

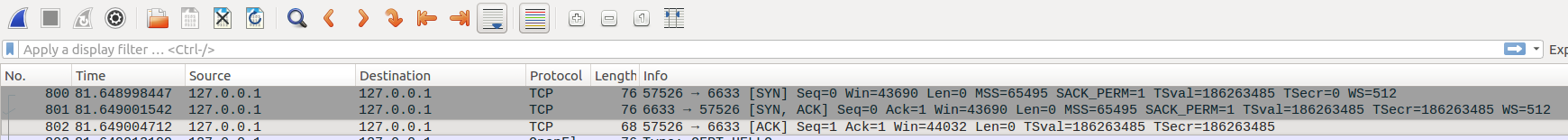


图 5-2交换机s2与控制器“三次握手”

2、OFPT\_HELLO报文：交换机和控制器均建立了TCP连接之后，开始互通hello报文：控制器与交换机建立连接时由其中某一方发起Hello消息，该消息携带支持的最高协议版本号，接受方将采用双方都支持的最低协议版本进行通信。一旦发现共同支持的协议版本，则连接建立，否则发送OFPT\_ERROR消息，描述失败原因，并中断连接。由于有两个交换机s1和s2，因此两个交换机都要和控制器c01进行hello报文的交换。于是会有四个hello报文。

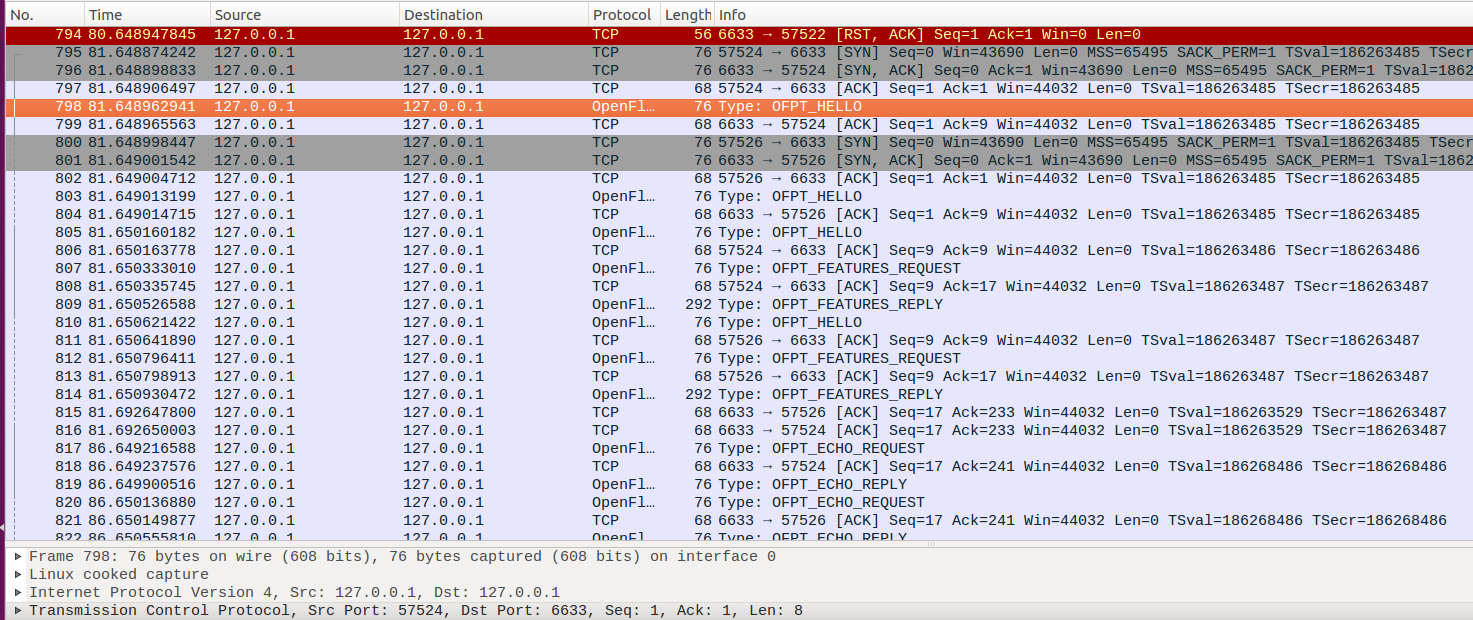
1. 第1个hello报文为端口号为57524的交换机向端口号为6633的控制器发送hello报文协商openflow版本号。

图 5-3第一个hello报文

1. 第2个hello报文为端口号为57526的交换机向端口号为6633的控制器发送hello报文协商openflow版本号。

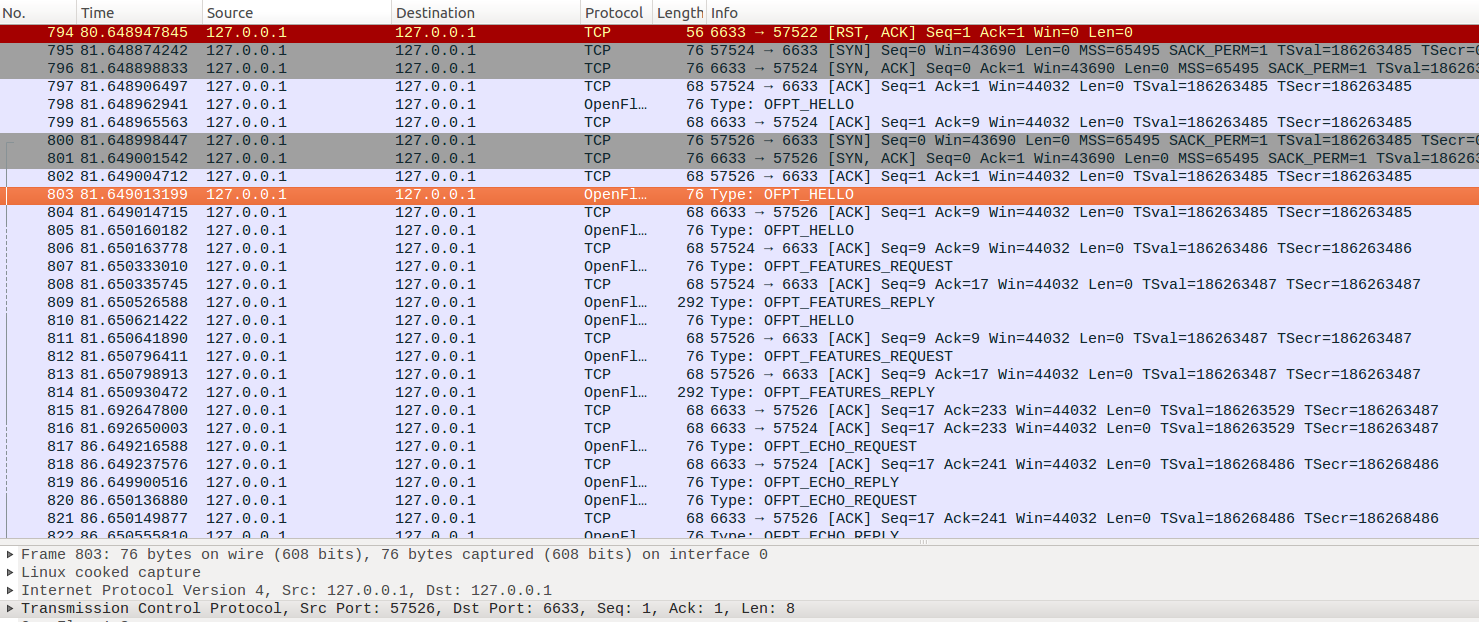


图 5-4第2个hello报文

（3）第3个hello报文为端口号为6633的控制器向端口号为57524的控制器发送的hello报文。

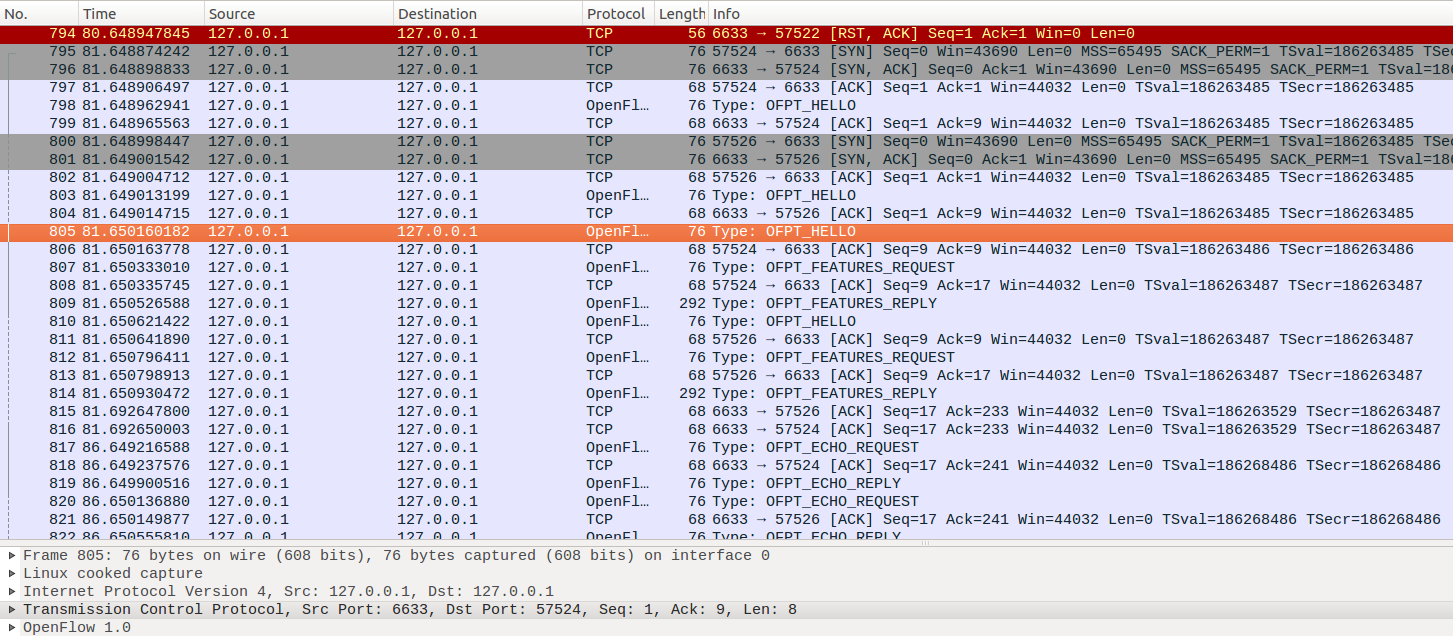


图 5-5第3个hello报文

（4）第4个hello报文为端口号为6633的控制器向端口号为57526的控制器发送的hello报文。

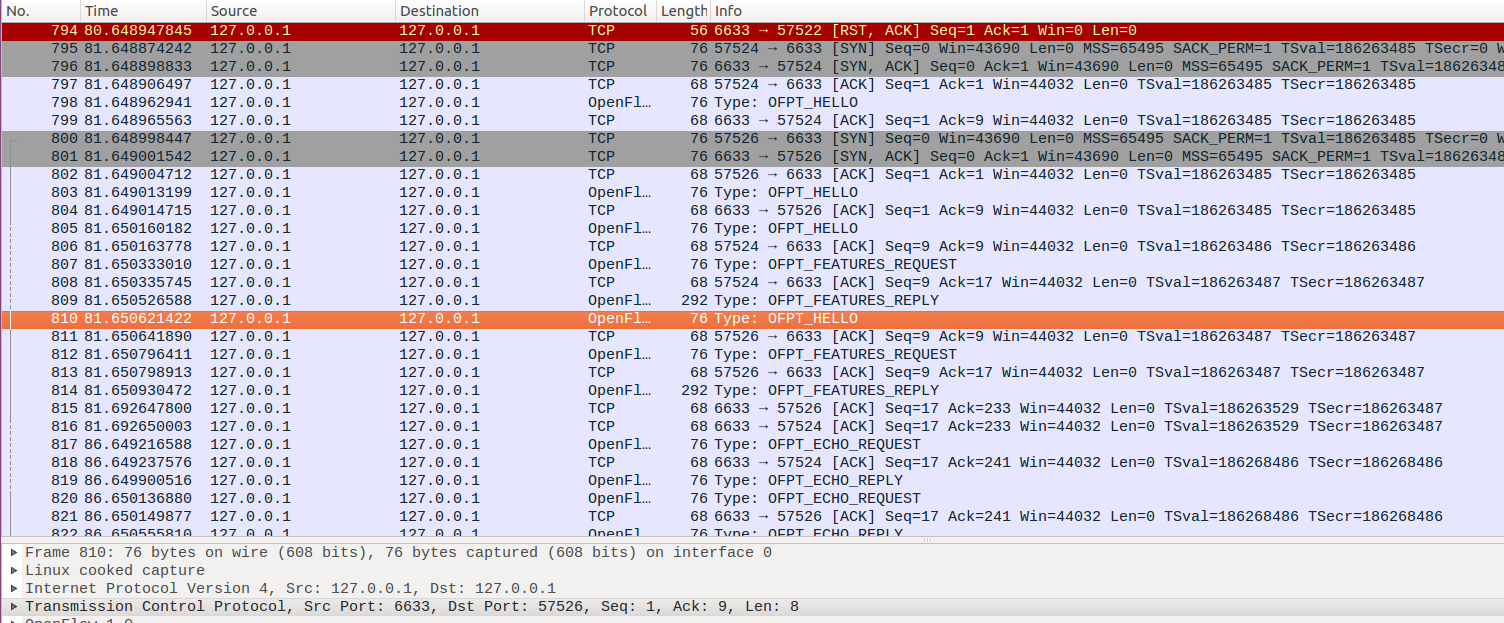


图 5-6第4个hello报文

3、OFPT\_FEATURES\_REQUEST和OFPT\_FEATURES\_REPLY报文

TLS会话一建立，sw跟controller完成连接之后，控制器会向交换机下发OFPT\_FEATYRES\_REQUEST的数据包，目的是请求交换机的信息。控制器向交换机发送 Features Request 消息查询交换机特性，Features Request 消息只包含 OpenFlow Header，其中 type 字段为 OFPT\_FEATURES\_REQUEST。

交换机和端口的配置信息在整一个通信过程起着至关的作用，因为所有关于的操作都需要从features里面提取相关的信息，如dpid,port\_no，等在整个通信过程中多次被用到的重要数据。每一次交换机连到控制器，都会收到控制器的features\_request,当sw将自己的features回复给控制器之后，控制器就对交换机有了一个全面的了解，从而为后面的控制提供的控制信息。request报文是ryu向OF交换机请求获取获取交换机特征的报文，reply报文是交换机向控制器说明交换机的特征信息的，datapath\_id、缓存包的最大数量、流表的最大数量等都是reply报文内部所包含的。

（1）控制器6633端口（我需要你的特征信息） ---> 交换机57524端口；

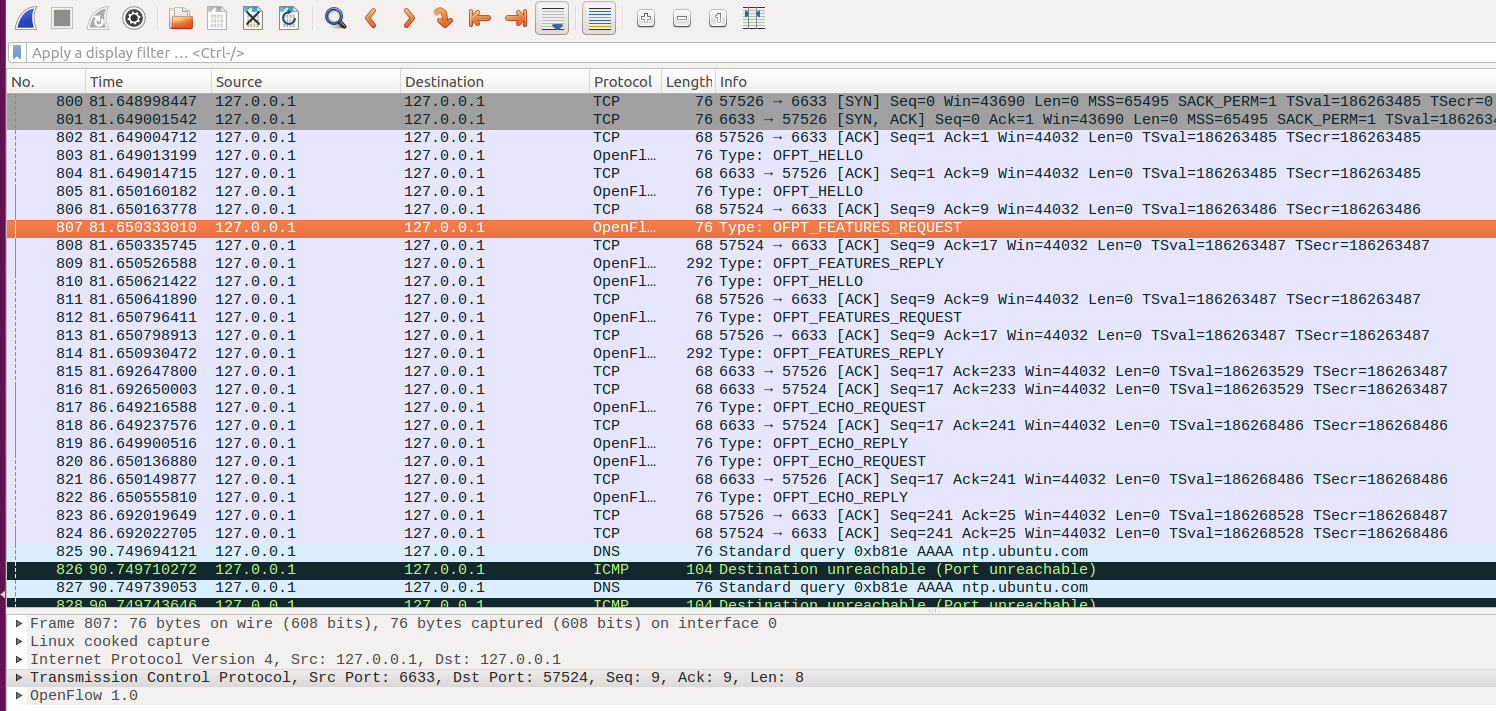


图 5-7 OFPT\_FEATURES\_REQUEST报文

（2）交换机60375端口（这是我的特征信息，请查收）--- 控制器6633端口

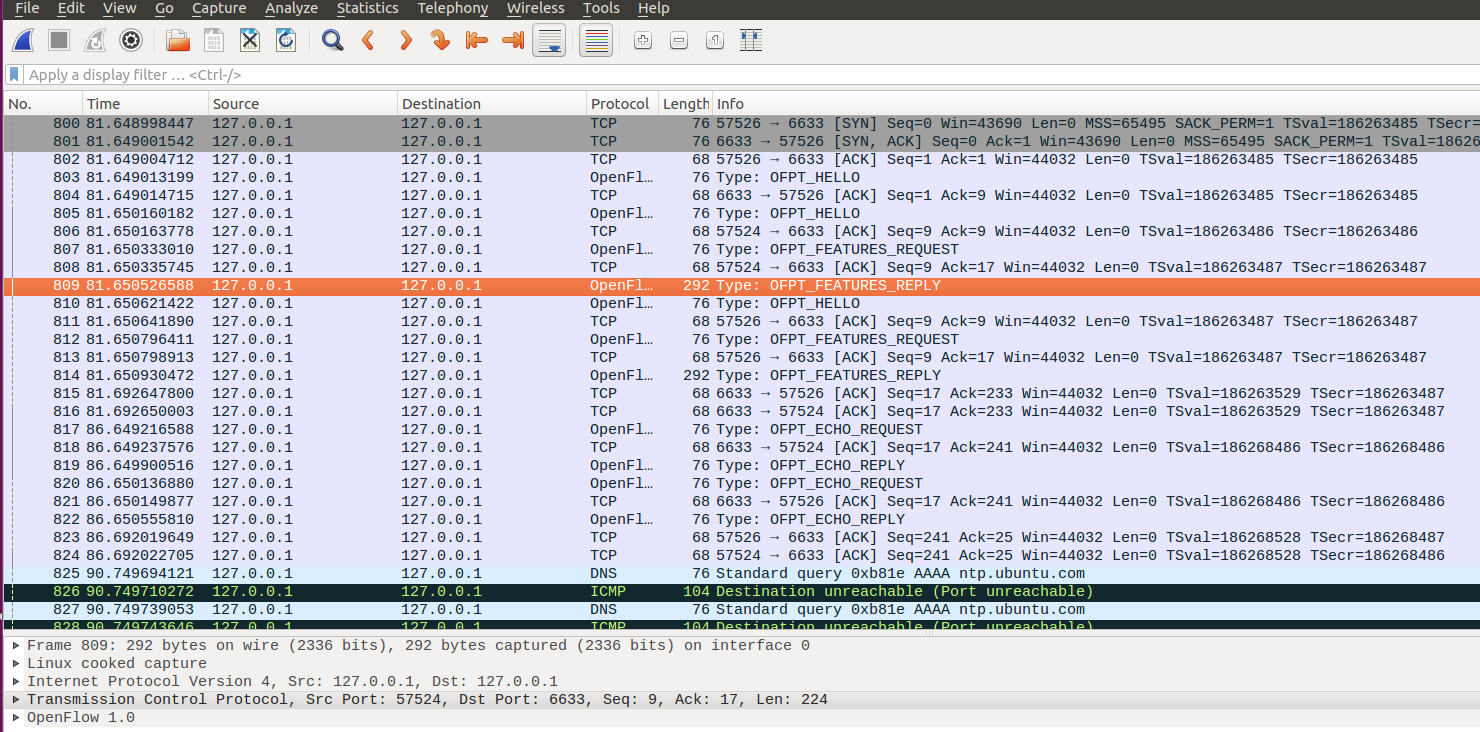


图 5-8 OFPT\_FEATURES\_REPLY报文

4、OFPT\_ECHO：

对称信息：OFPT\_ECHO\_REQUEST, OFPT\_ECHO\_REPLY

作用：查询连接状态，确保通信通畅。当没有其他的数据包进行交换时，controller会定期循环给sw发送OFPT\_ECHO\_REQUEST。

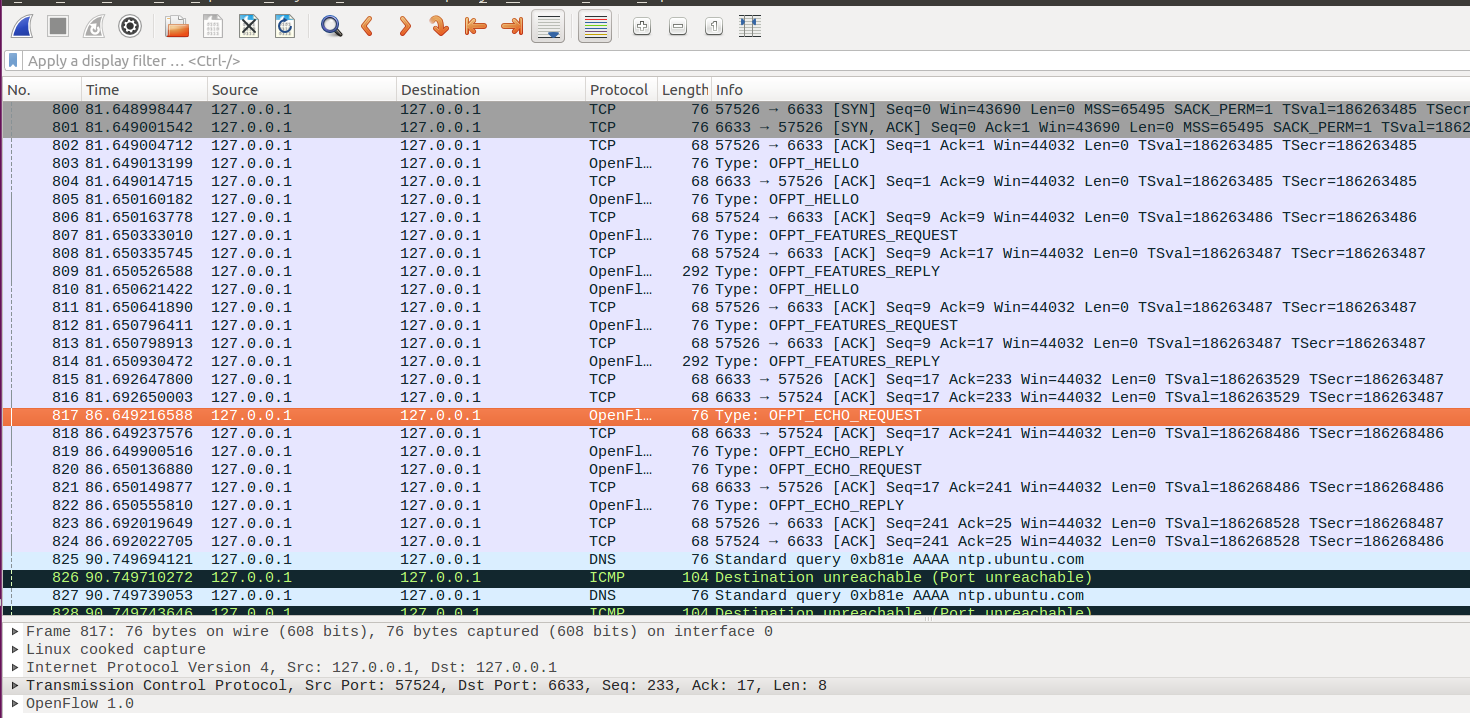


图 5-8 OFPT\_ECHO\_REQUEST报文

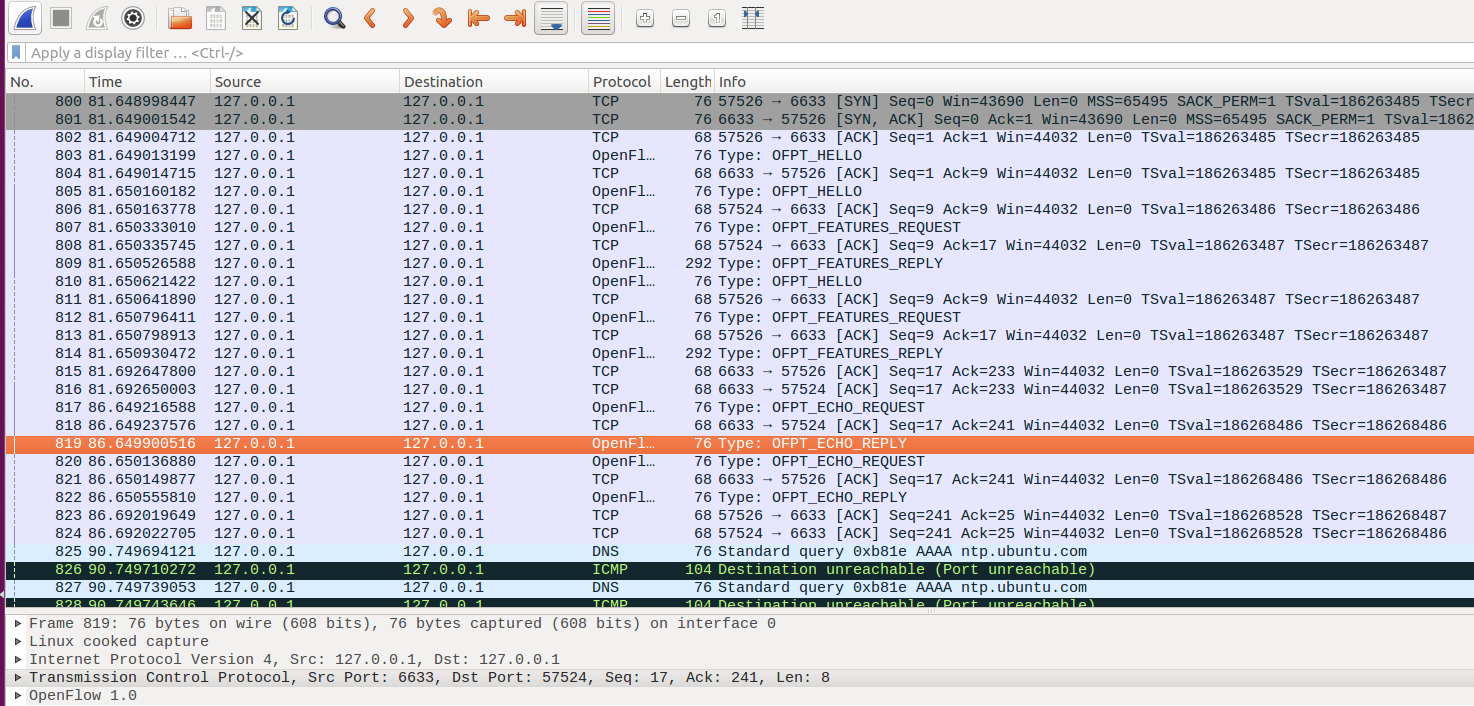


图 5-8 OFPT\_ECHO\_REPLY报文

5、控制器与交换机建立连接后，控制器会自发给交换机发送一组flow\_mod消息，

Flow\_mod这个消息是OpenFlow中最重要的消息之一，用来添加、删除、修改OpenFlow交换机的流表信息。当交换机收到一个无法处理的数据包，封装到Packet\_in消息转发给控制器后，控制器可以发送一个Flow\_mod消息下发一个流表到交换机，并且指定该数据包按照此流表项的action处理。

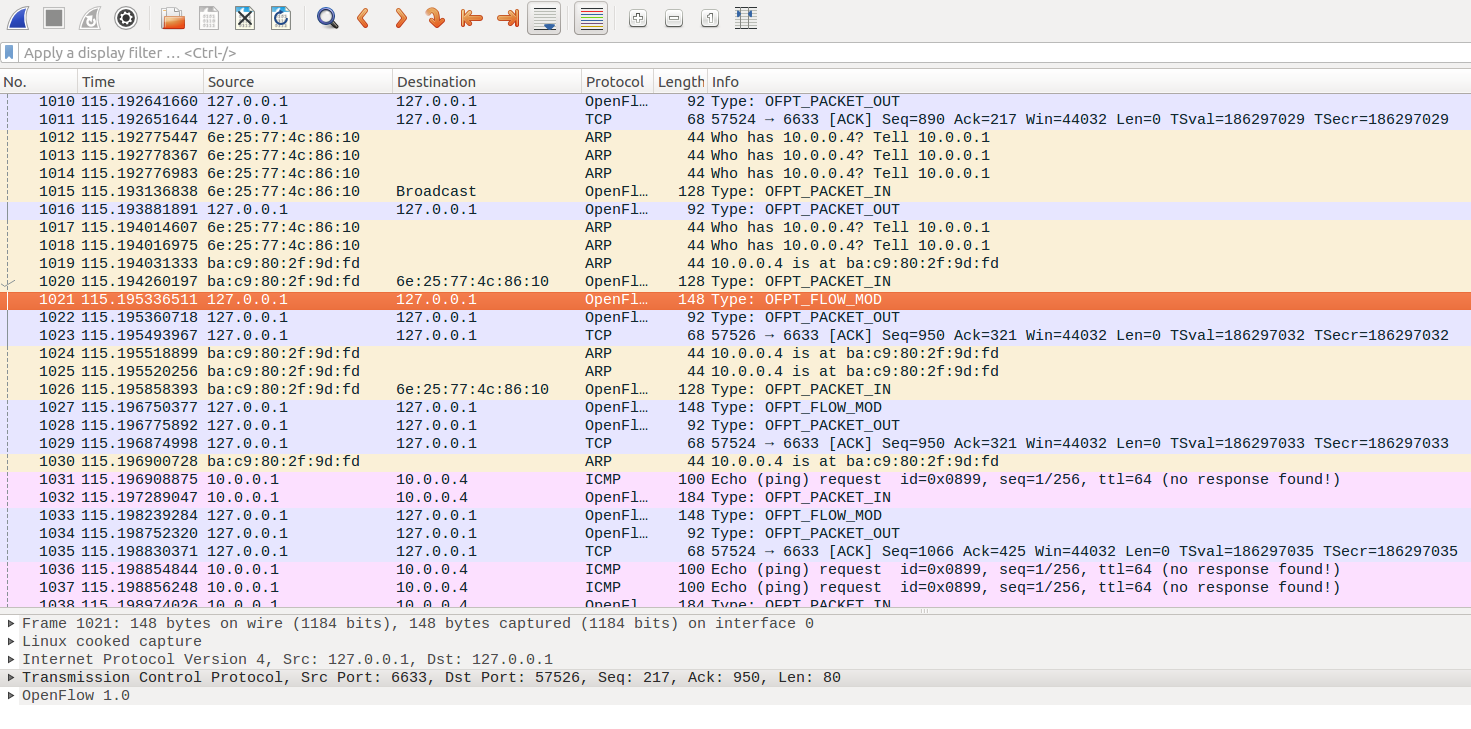


图 5-8 OFPT\_FLOW\_MOD报文

6、在控制器获取完交换机的特性之后，交换机开始处理数据。当交换机碰到新数据包不知道如何处理，或者action要求发送给控制器，那么交换机就会用packet\_in消息发送给控制器。一般将数据包缓存在交换机中，将有效的数据包信息和缓存id发送给控制器，不过，如果交换机不支持缓存或者内存用光了，那么就把整个数据包放在数据部分发给控制器。包含在packet\_in中的数据可能是很多种类型，arp和icmp是最常见的类型。OFPT\_PACKET\_IN交换机发送Packet\_in消息给控制器有以下两种情况：报文没有匹配到流表项；流表项的action要求发给控制器。无法匹配的数据包会产生packet\_in,action也可以指定将数据包发给packet\_in,也就是说我们可以利用这一点，将需要的数据包发给控制器。

交换机57526端口（有数据包进来，请指示）--- 控制器6633端口

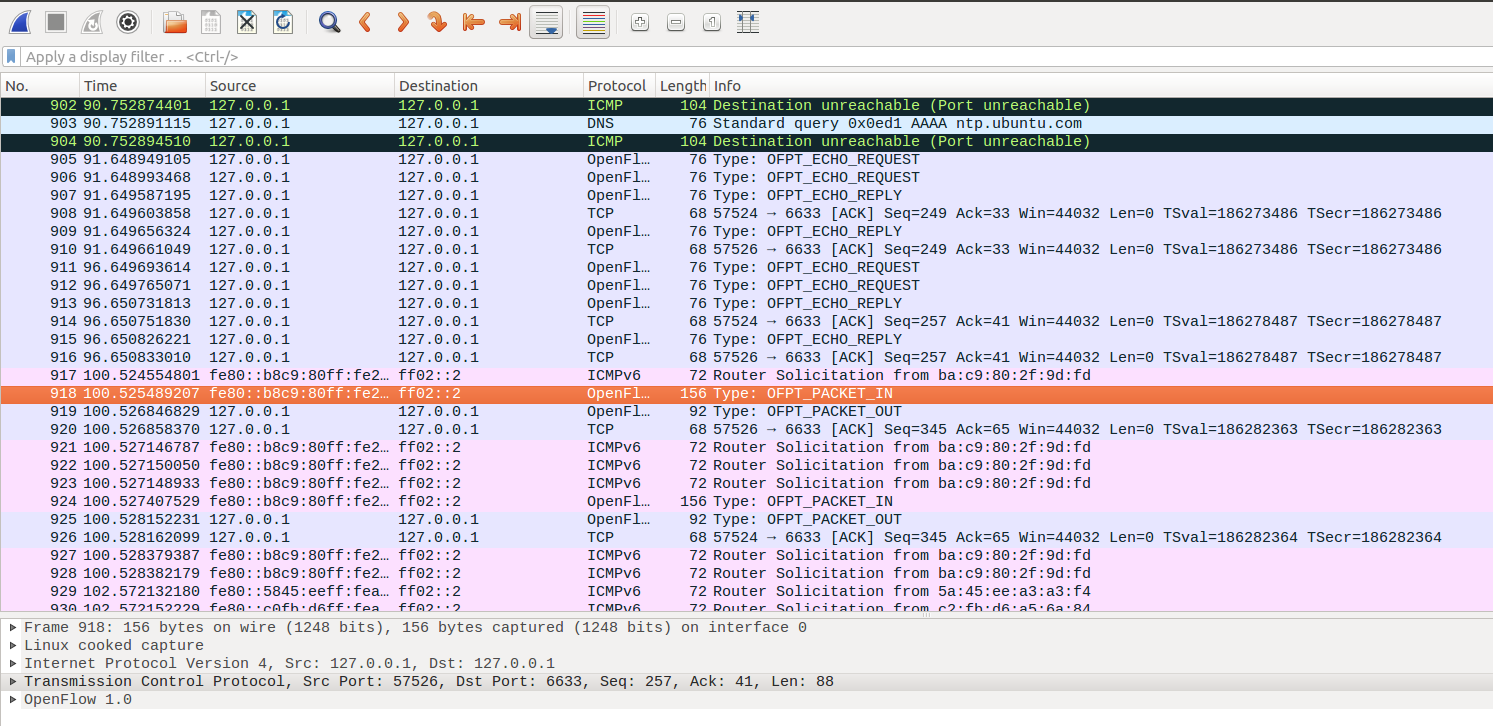


图 5-8 OFPT\_PACKET\_IN报文

7、当控制器希望交换机发送某个数据包，就使用packet\_out消息。packet\_out的作用在与指导交换机做flood操作，并把ARP包广播出去，packet\_out本身并不携带arp数据包。

控制器6633端口（请按照我给你的action进行处理） ---> 交换机57526端口

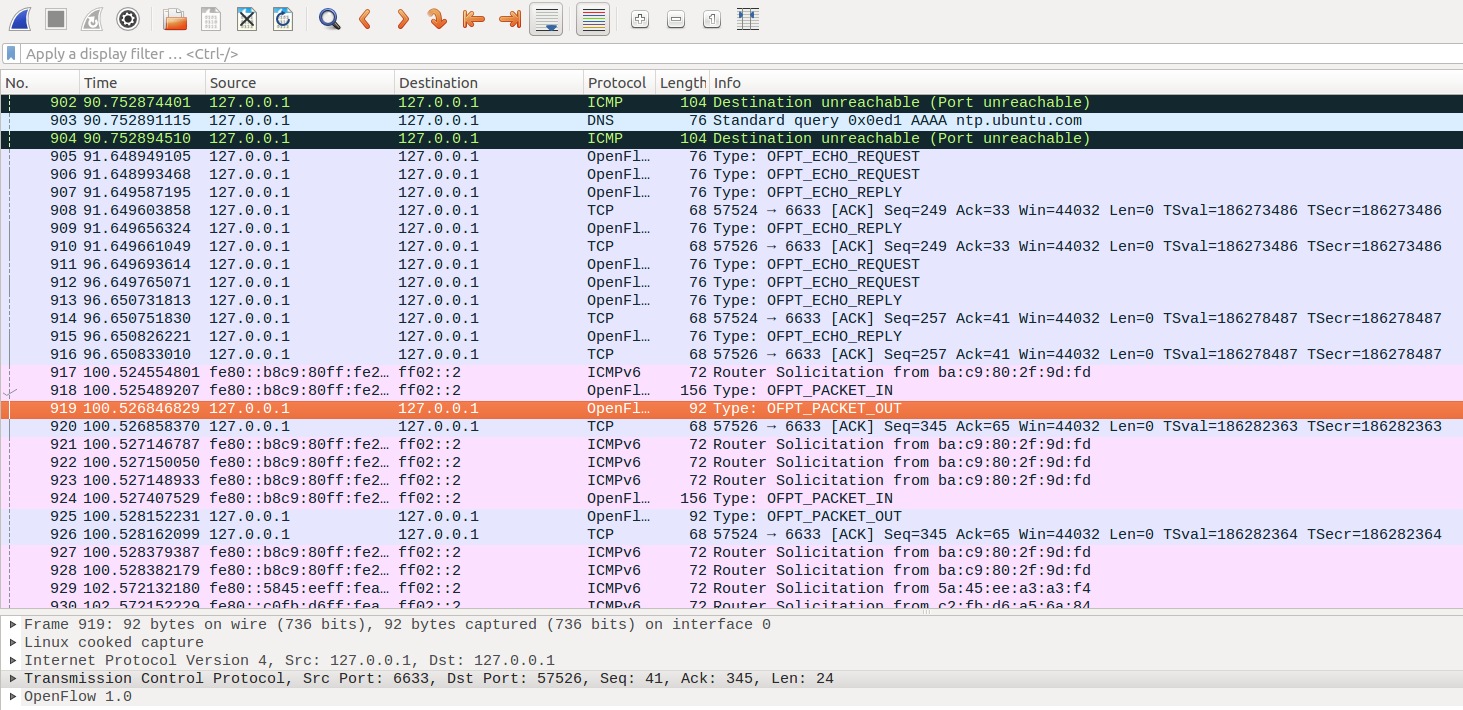


图 5-8 OFPT\_PACKET\_OUT报文

# 六、遇到的问题及解决方法

1、打开wireshark后创建拓扑，过滤Openflow数据包，并没有发现Flow\_Mod数据包。解决：查阅资料之后，发现Flow\_Mod数据包是控制器想交换机下发流表项，指导数据的转发处理，所以在启动wireshark之后，再尝试执行pingall，最后发现了Flow\_mod数据包。

2、在进行抓包时必须一次性完成所有包的查找分析，否则再次启动虚拟机时，会发生两次抓包同一个包而端口不一致的情况。

3、第一次抓包没有找到OFPT\_HELLO，后查资料发现是因为我先运行了拓扑文件后才打开了wireshark，应该先打开wireshark再运行拓扑。

# 七、收获与体会

1、主要是考验如何使用wireshark进行数据抓包的操作以及需要将openflow源码与抓包结果对应起来。在更加熟悉使用wireshark进行抓包操作的同时，让我更进一步理解交换机和控制器之间消息交互过程，在进阶部分找到的源码可以作进一步的分析，这样可以对OpenFlow协议有更深刻的认识

1. 利用VMware虚拟机构建SDN的网络拓扑，了解了虚拟机的运行机理，掌握了SDN网络拓扑结构；对照目的端口和源端口以及查到的各种消息的数据结构定义，对交换机与控制器的消息交互过程更加熟悉。将课程所讲内容进行实际运用，加深理解和记忆。

3、进行IP配置时，既可以在./miniedit.py中选中主机进行配置，也可在生成的拓扑文件中进行设置。在wireshark的Apply a display filte中输入openflow\_v1或其他进行筛选能够提高查询效率。