**任务目的**

1、了解OpenFlow交换机与OpenFlow控制器建立TCP连接的过程。  
2、掌握配置安全通道中的OpenFlow版本的方法。  
3、掌握OpenFlow交换机和OpenFlow控制器的消息交互流程。

**任务环境**

|  |  |  |
| --- | --- | --- |
| **设备名称** | **软件环境（镜像）** | **硬件环境** |
| 控制器 | Ubuntu 14.04桌面版 Floodlight 1.0 | CPU：1核 内存：2G 磁盘：20G |
| 主机 | Ubuntu 14.04桌面版 Mininet 2.2.0 | CPU：1核 内存：2G 磁盘：20G |

注：系统默认的账户为root/root@openlab，openlab/user@openlab。

**任务内容**

1、配置交换机与控制器，使其支持OpenFlow1.3。  
2、分析消息包，掌握交换机与控制器的消息交互流程。

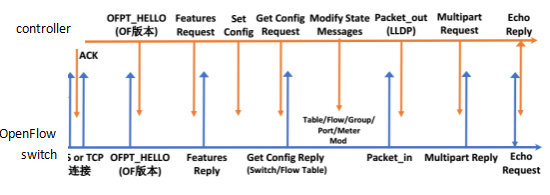
**实验原理**

**一、OpenFlow协议简介**

2006年，斯坦福大学Clean Slate计划资助的Ethane项目开始部署，致力于企业网架构的创新，OpenFlow协议的雏形就诞生于这个项目。2008年，Nick McKeown教授的一篇重要论文“OpenFlow：Enabling Innovation in Campus Networks”使得OpenFlow正式进入人们的视野，继而成为了标准化组织ONF（Open Network Foundation，开放网络基金会）主推的南向接口协议。经过多年的发展，OpenFlow现已成为SDN的主流南向接口协议之一。目前，OpenFlow协议还在不断地演进中，本实验采用OpenFlow v1.3协议，并对控制器与OpenFlow交换机之间的交互过程进行深入分析。  
OpenFlow主要有3种类型的消息，分别是Controller-to-Switch、Asynchronous和Symmetric，其中每个类型又包含多个子类型。Controller-to-Switch消息由控制器发起，用于管理、查看交换机的状态。Asynchronous消息由交换机发起，向控制器汇报交换机的事件和改变。Symmetric消息由控制器或交换机任一方发起，无需请求直接发起消息。详细信息如下表所示：

|  |  |  |
| --- | --- | --- |
| **消息类型** | **消息例子** | **描述** |
| Controller-to-Switch | Packet\_out Barrier Switch Configuration Switch Features Multipart | 控制器使用这些消息可以添加、修改或删除流表项 查询交换机的功能和统计 配置交换机 配置交换机端口属性 将数据包发送出指定交换机端口 |
| Asynchronous（异步） | Error Packet\_in Port Status Table Status Controller Role Status | 由交换机发起，发送消息给控制器 这些消息可以是没有匹配交换机中任意流表项的数据包或数据包头，因此需要发送给控制器进行处理 在流状态、端口状态改变，或者产生错误消息时，进行通知 |
| Symmetric（对称） | Hello Echo Experimenter | Hello消息在控制器与交换机建立连接过程中使用 Echo消息用来确定Controller-to-Switch连接的延时，验证连接是否处于活跃状态 Experimenter消息用于未来消息的扩展 |

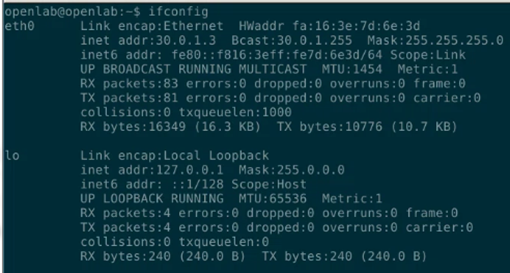
**二、OpenFlow连接建立交互流程**

  
在OpenFlow1.3协议的情况下，控制器与OpenFlow交换机的消息完整交互流程如下：  
1、 控制器与OpenFlow交换机通过TCP“三次握手”，建立有效的连接。其中，控制器一端的端口号为6633。  
2、 控制器与OpenFlow交换机之间相互发送Hello消息，用于协商双方的OpenFlow版本号。在双方支持的最高版本号不一致的情况下，协商的结果将以较低的OpenFlow版本为准。如果双方协商不一致，还会产生Error消息。  
3、 控制器向OpenFlow交换机发送Features Request消息，请求OpenFlow交换机上传自己的详细 参数。OpenFlow交换机收到请求后，向控制器发送Features Reply消息，详细汇报自身参数，包括支持的buffer数目、流表数以及Actions等。  
4、 控制器通过Set Config消息下发配置参数，然后通过Get config Request消息请求OpenFlow交换机上传修改后的配置信息。OpenFlow交换机通过Get config Reply消息向控制器发送当前的配置信息。  
5、 控制器与OpenFlow交换机之间发送Packet\_out、Packet\_in消息，通过Packet\_out中内置的LLDP包，进行网络拓扑的探测。  
6、 控制器与OpenFlow交换机之间通过发送Multipart Request、Mutipart Reply消息，控制器能获取OpenFlow交换机的状态信息，包括流的信息、端口信息等。  
7、 控制器与OpenFlow交换机之间通过发送Echo Request、Echo Reply消息，保证二者之间存在有效连接，避免失联。  
**说明：以上为控制器与OpenFlow交换机之间的标准交互流程，在具体实验过程中某些阶段可能会缺失。**

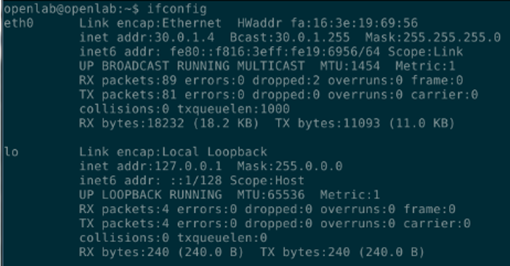
**实验步骤**

**一、实验环境检查**

**步骤1 登录控制器，查看控制器所在主机的IP地址。桌面版镜像可以通过双击桌面上的“Terminal”图标打开命令终端，也可以使用Ctrl+Alt+t快捷方式打开命令终端，如下图所示。**



**步骤2 登录主机1，查看Mininet所在主机的IP地址，如下图所示。**



**二、捕获数据包**

**步骤1 登录Floodlight控制器，启动抓包工具Wireshark，捕获控制器与交换机建立连接过程中的数据包，通过分析这些数据包了解控制器与交换机基于OpenFlow协议进行交互的流程。执行以下命令：**

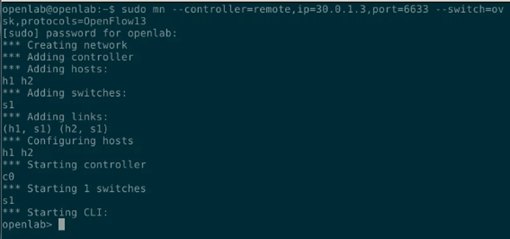
$ sudo wireshark

**步骤2 双击eth0网卡，查看eth0网卡上数据包收发情况，如下图所示。**

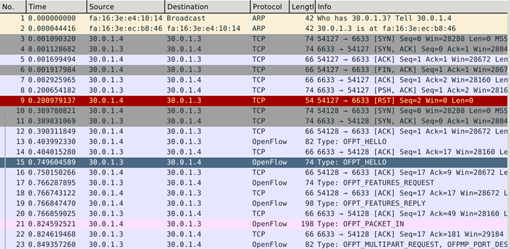


**步骤3 登录Mininet虚拟机，启动Mininet。通过“--controller”参数设置Mininet连接远程控制器，并指定控制器的IP和端口号。**

$ sudo mn --controller=remote,ip=30.0.1.3,port=6633 --switch=ovsk,protocols=OpenFlow13

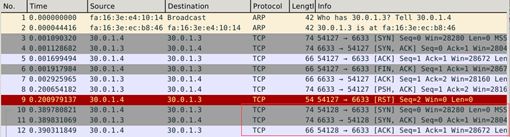


**步骤4 登录Floodlight控制器，停止Wireshark，观察数据包列表，可以看出控制器与交换机的基本交互流程。**

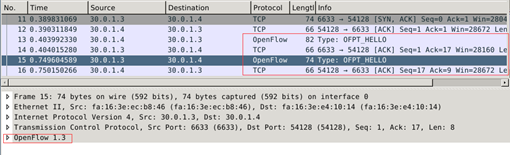


**三、OpenFlow1.3交互流程分析**

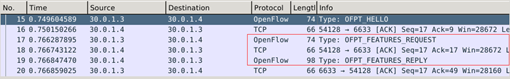
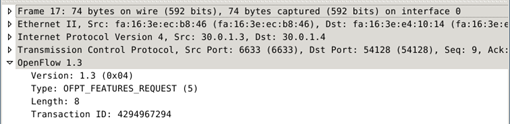
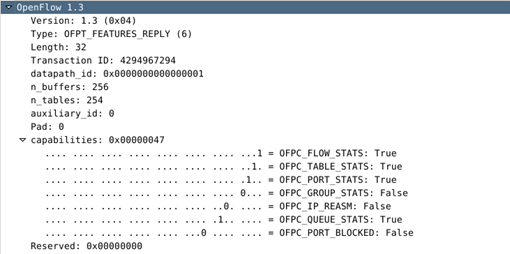
**步骤1 交换机连接控制器的6633端口，经过3次握手后双方建立TCP连接。查看捕获到的数据包，分析交换机与控制器建立TCP连接的流程。分析TCP连接建立过程，需要先了解TCP的状态位，主要包括SYN、FIN、ACK、PSH、RST和URG。SYN表示建立连接，FIN表示关闭连接，ACK表示响应，PSH表示有DATA数据传输，RST表示连接重置。可以看出交换机与控制器经历一次连接重置后，成功完成三次握手，建立TCP连接，如下图所示。**



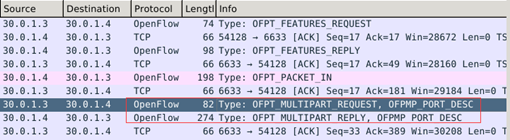
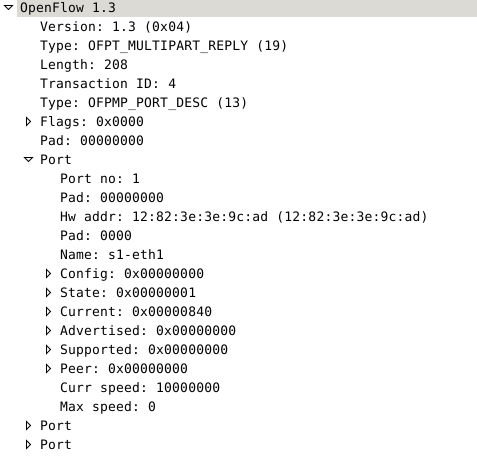
**步骤2 当控制器与交换机建立TCP连接后，由其中某一方发起Hello消息，双方协调协OpenFlow议版本号。控制器和交换机都会向对方发送一条Hello消息，消息中附上自己支持的OpenFlow的最高版本。接收到对方Hello消息后，判断自己能否支持对方发送的版本，能支持则版本协商成功，不能支持则回复一条OFPT\_ERROR消息。查看Hello消息详情，本实验中由于交换机和控制器都能支持OpenFlow1.3版本，所以版本协商为1.3，如下图所示。**



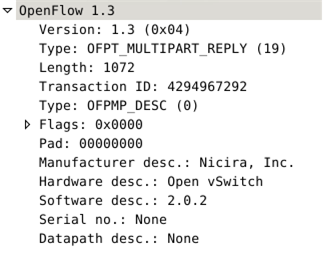
**步骤3 OpenFlow版本协商完成后，控制器发送一条features\_request消息获取交换机的特性信息，包括交换机的ID（DPID）、缓冲区数量、端口及端口属性等等。相应的，交换机回复features\_reply消息，如下图所示。**

  
查看数据包详情，ofpt\_feature\_request消息只有包头，如下图所示。  
  
ofpt\_feature\_reply数据包详情如下，交换机的DPID是数据通道独一无二的标识符，低48位是一个MAC地址，高16位是自定义的。本实验中交换机缓冲区数量（n\_buffers）为256，交换机支持的流表数量（n\_tables）为254，交换机所支持的功能，如下图所示。  


**步骤4 OpenFlow1.0协议中feature\_reply消息还包含交换机端口信息，OpenFlow 1.3协议将‘stats’框架更名为‘multipart’框架，并且将端口描述移植到multipart消息中。其中OPPT\_PORT\_DESC类型的multipart消息就是用于获取交换机端口信息的。**

  
查看OPPT\_PORT\_DESC类型multipart\_reply消息，消息中列出了交换机的端口以及每个端口的详细信息，包括端口名称和mac地址等，如下图所示。  


**步骤5 OFPMP\_DESC类型的multipart\_reply消息包含了交换机的其他信息，包括交换机厂商名称、交换机名称以及交换机版本等。本实验中使用的是Mininet仿真软件中自带的开源交换机Open vSwitch(2.0.2)，而Open vSwitch是由Nicira Networks主导开发的，如下图所示。**



**步骤6 在连接过程中，控制器不断的发送echo\_request消息给交换机，确认交换机与控制器之间的连接状态。相应的，交换机会回复echo\_reply消息，如下图所示。**

