OpenFlow与Floodlight实验文档

一、实验目的

- 以mininet为实验平台，了解SDN网络基本架构
- 了解FloodLight为代表的网络控制器
- 基于mininet和FloodLight搭建SDN网络

二、实验简介

软件定义网络简介

软件定义网络（Software Defined Network, SDN ），是一种新型网络创新架构，其核心思想是网络开放可编程，并在架构层面提出以下两个创新思路：

1）针对网络设备复杂、封闭的问题，SDN将网络数据平面和控制平面解耦。网络的数据平面是通用的网络设备，其不具有网络控制功能，仅负责网络流量的转发，具有可编程的数据包处理能力。网络设备通过统一的控制接口开放给控制平面，使得网络设备变得结构简单、功能开放。

2）针对现有互联网僵化、低效的控制方式，SDN对控制平面进行抽象。利用逻辑集中的控制器对网络进行全局控制，控制器作为网络操作系统为用户开放应用程序编程接口（Application Programming Interface，API）。网络用户基于全局网络视图定义自己的网络应用，使得网络控制变得灵活、高效。

基于以上思路，SDN 提出了三层网络架构，如图 1 所示。SDN 将数据平面和控制平面分离，数据平面为转发设备层，控制平面被划分为网络控制层和网络应用层。在转发设备层中，具有通用转发功能的设备互连形成网络数据通路，转发设备通过标准的转发控制接口与控制平面进行通信；网络控制层中的控制器实现拓扑发现、路由计算、服务质量控制等网络控制功能；网络应用层通过 API 与网络控制层进行通信，使得网络用户能够灵活地部署网络应用。

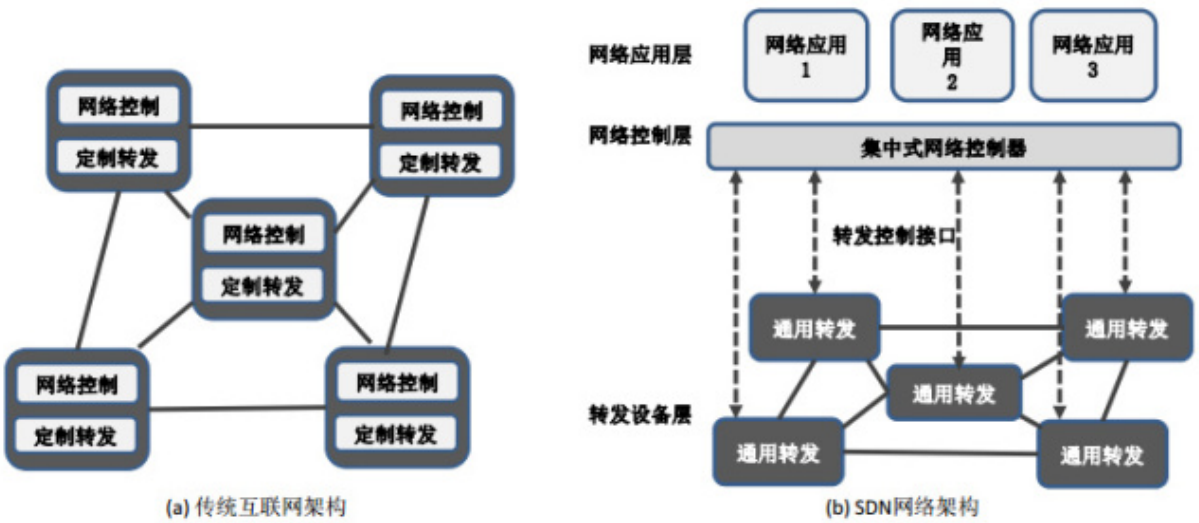


图 1　传统互联网与 SDN 网络架构

Nick McKeown等在SDN白皮书中提出设计数据平面的四个基本原则：实现网络的有效隔离（虚拟化）、高性能低成本的设备（资源高效利用）、支持新出现的协议和更广泛的网络研究（可编程）和适应厂商封闭内部实现的需求。其中虚拟化、资源高效利用、可编程这三个原则是SDN对数据平面的核心诉求。下面分别进行介绍：

第一，网络虚拟化是指在同一物理网络上并行运行多个逻辑上相互隔离的网络。通过网络虚拟化，不同的业务可以相互隔离地运行在多个虚拟网络中，使得网络管理者可以为不同业务定义不同的服务质量，还使得用户能够在不影响现有网络运行的条件下在网络中快速部署创新。为了实现虚拟化，数据平面需要提供灵活的虚拟化结构和机制，使得单个物理设备能够同时运行多个虚拟转发实例，虚拟转发实例在功能、控制和性能方面互不干扰。

第二，随着SDN的发展，其应用场景已经从实验网络走向商用网络，并将在更大规模更高性能的网络上应用。这就需要SDN数据平面设备能够高效地利用计算和存储等物理资源，在有限的硬件条件下承载更多的网络节点、处理更多类型的业务，并且能够在转发性能方面与现有的商用路由器相匹配。

第三，SDN的核心目的是实现网络可编程。数据平面可编程是实现网络可编程的基础，是指用户可以根据不同应用和协议的需求，对数据包的处理和转发过程进行灵活的定义和修改。现有路由器和交换机通过固定的逻辑实现了数据包的处理流程，对支持的协议进行专门功能优化，难以支持用户自定义的协议。SDN认为网络的数据平面应该是通用的，新出现协议的部署不能受限于数据平面的硬件结构。

OpenFlow作为SDN数据平面的事实标准，在研究者的广泛参与和ONF的大力推动下，发展十分迅速，其协议标准保持了以半年为周期的更新速度，已经从 1.0 更新到了 1.4版本。

OpenFlow网络由OpenFlow交换机和控制器(Controller)两部分组成。OpenFlow交换机进行数据层的转发；Controller对网络进行集中控制，实现控制层的功能。OpenFlow交换机是整个OpenFlow网络的核心部件，它由流表、安全通道和OpenFlow协议三部分组成主要管理数据层的转发。OpenFlow交换机接收到数据包后，首先在本地的流表上查找转发目标端口，如果没有匹配，则把数据包转发给Controller，由控制层决定转发端口。安全通道是连接OpenFlow交换机到控制器的接口。控制器通过这个接口控制和管理交换机，同时控制器接收来自交换机的事件并向交换机发送数据包。交换机和控制器通过安全通道进行通信，而且所有的信息必须按照OpenFlow协议规定的格式来执行。OpenFlow协议用来描述控制器和交换机之间交互所用信息的标准，以及控制器和交换机的接口标准。协议的核心部分是用于OpenFlow协议信息结构的集合。

FloodLight 简介

Floodlight 是由开源社区（主要是 BigSwitch）开发的一款开源控制器。开发语言为Java，目前支持 OpenFlow1.1。 Floodlight 以 Apache 开源协议发行，任何使用者都可以基于 Floodlight 进一步开发自己需要的功能。Floodlight 内部提供了拓扑管理（Topology Manager）、基于最短路径的路由（Routing）、终端设备的跟踪管理（Device Manager）等模块。并在此基础上提供了基于MAC learning 的数据包转发、基于源 IP 和目的 IP 的数据包转发以及写入静态流表的应用。这些应用以 RestAPI 的形式对外开放， 第三方用户可以基于提供的 RestAPI 独立开发新的网络应用。

Floodlight 经编译后生成一个可执行的 jar 文件，类似于 C 或 C++中的.exe 文件。假设 Floodlight 编译生成的可执行 jar 文件为 floodlight.jar, 其调用格式为：

```
java –jar floodlight.jar –cf floodlight.properties
```

其中 floodlight.properties 文件用来指定运行 floodlight 时需要加载的模块和配置floodlight 运行时的参数，例如 RestAPI Server 监听的端口，Floodlight Web 界面的端口，流表默认的 Timeout 等。未加指定时 Floodlight 会调用默认配置文件，在以下目录中：

```
src/main/resources/floodlightdefault.properties
```

配置文件的格式如下：

```
floodlight.modules=
net.floodlightcontroller.storage.memory.MemoryStorageSource,\
net.floodlightcontroller.core.FloodlightProvider,\
net.floodlightcontroller.threadpool.ThreadPool,\
net.floodlightcontroller.devicemanager.internal.DeviceManagerImpl,\
net.floodlightcontroller.staticflowentry.StaticFlowEntryPusher,\
net.floodlightcontroller.firewall.Firewall,\
net.floodlightcontroller.forwarding.Forwarding,\
net.floodlightcontroller.jython.JythonDebugInterface,\
net.floodlightcontroller.counter.CounterStore,\
net.floodlightcontroller.perfmon.PktInProcessingTime,\
net.floodlightcontroller.ui.web.StaticWebRoutable
net.floodlightcontroller.restserver.RestApiServer.port = 8080
net.floodlightcontroller.core.FloodlightProvider.openflowport = 6633
net.floodlightcontroller.jython.JythonDebugInterface.port = 6655
net.floodlightcontroller.forwarding.Forwarding.idletimeout = 5
net.floodlightcontroller.forwarding.Forwarding.hardtimeout = 0
```

其中前面11行表示Floodlight运行时需要加载的模块，后面几行为其他参数的配置。由于各模块之间可能存在依赖关系，在写需要加载的模块时需要注意。

Floodlight 提供了 Web 界面供管理员监控当前的网络状态，在启动 Floodlight 之后，在浏览器中输入如下 URL 可以进入相应的 Web 界面：

Mininet 简介

Mininet 是一个轻量级软件定义网络和测试平台；它采用轻量级的虚拟化技术使一个单一的系统看起来像一个完整的网络运行想过的内核系统和用户代码，也可简单理解为 SDN 网络系统中的一种基于进程虚拟化平台，它支持 OpenFlow、OpenvSwith 等各种协议，Mininet 也可以模拟一个完整的网络主机、链接和交换机在同一台计算机上且有助于互动开发、测试和演示，尤其是那些使用 OpenFlow 和 SDN 技术；同时也可将此进程虚拟化的平台下代码迁移到真实的环境中。

Mininet 当前支持的特性

- 支持 OpenFlow、OpenvSwitch 等软定义网路部件
- 支持系统级的还原测试，支持复杂拓扑，自定义拓扑等
- 提供 Python API, 方便多人协作开发
- 很好的硬件移植性与高扩展性
- 支持数千台主机的网络结构

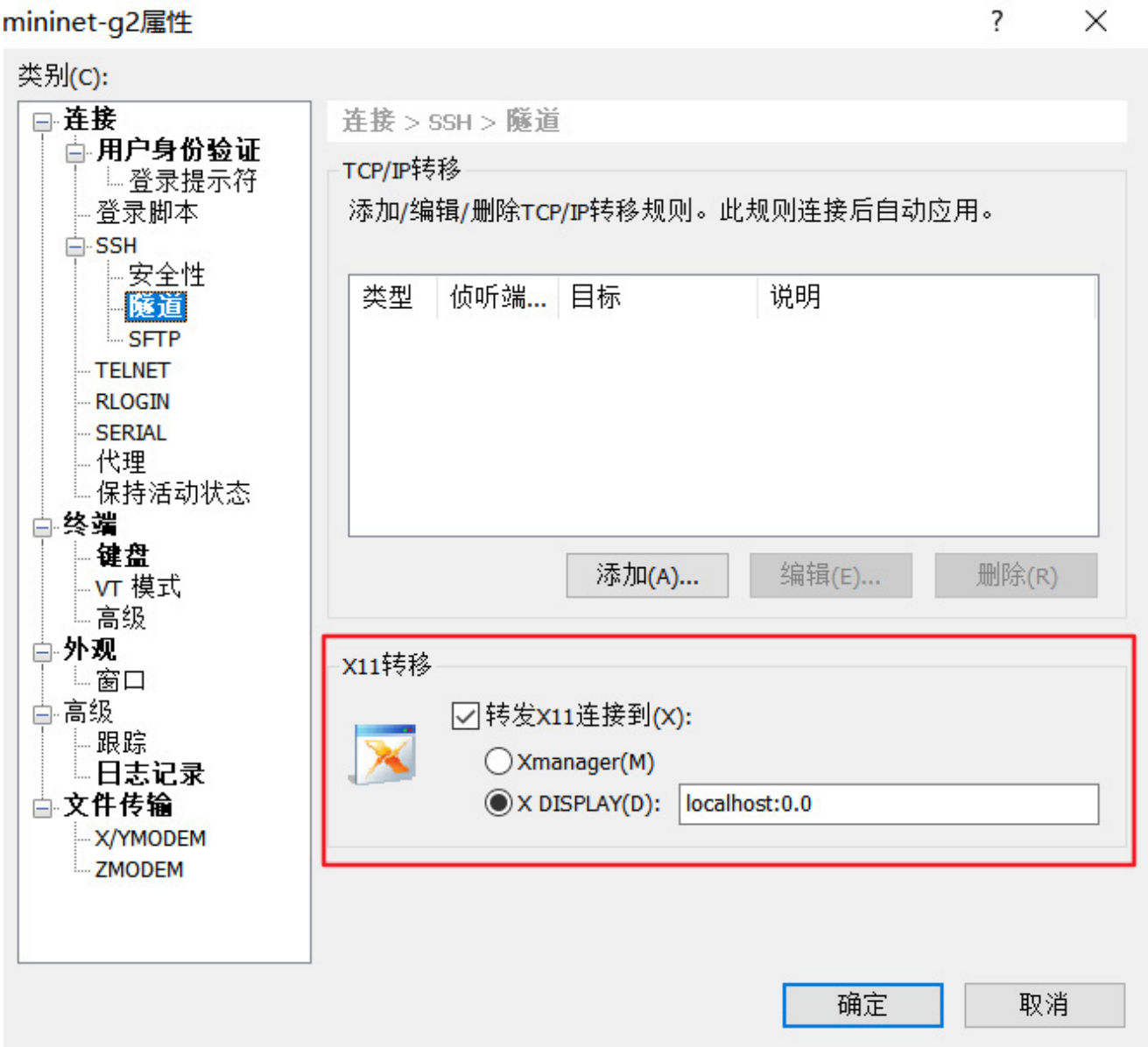
本次实验，我们将基于Mininet搭建基本的实验网络仿真平台，关于Mininet的基本使用可以参考其官方主页提供的相关内容。

三、实验准备

我们已经提前为各组同学创建了Linux虚拟机，预装了Mininet和Floodlight，方便大家直接使用。下面的配置均针对Windows。

SSH登录：

推荐使用 Xshell 作为客户端，可以个人免费使用。虚拟机地址：mininet-gX.lab.fib-lab.com（X为组号），用户名及密码均为mininet。另外需如下图配置X11转发即可。



XForwarding设置：

下载 Xming 并安装，启动安装路径下的XLaunch.exe，一路下一步即可正常启动XForwarding。

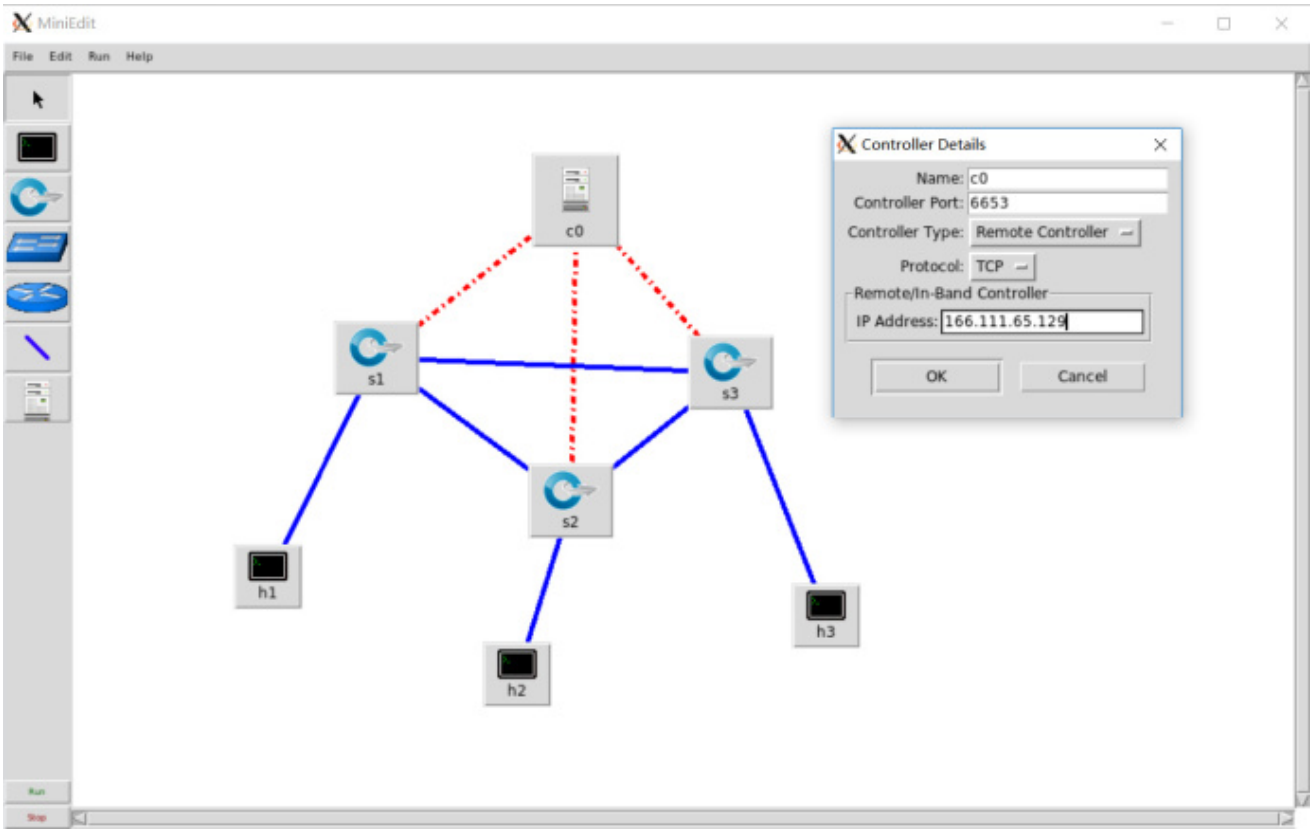
四、实验内容

1. 建立网络拓扑

我们可以选择直接编写python脚本来定义拓扑，也可以直接使用mininet提供的可视化界面生成需要拓扑。自定义脚本请感兴趣的同学自行学习，这里我们将介绍可视化的操作方法，这里首先根据前面提到的方式设置XForwarding。

```
python mininet/examples/miniedit.py  # 启动mininet可视化编辑程序，该界面建议保留观察，可以再起一个连接用于其他操作，拓扑确认后即可关闭
```

通过编辑界面选择、放置、连线，构建如下拓扑图，其中c0表示控制器，s0-s2表示交换机，h0-h2表示主机，并在File中选择export l2 script选项保存构建拓扑的脚本至合适位置。本次实验要求建立环形拓扑，除此之外大家可以自由发挥。




```
sudo python topology.py # 按照保存的拓扑结构，启动mininet平台
```

如果mininet没有正常退出，需要执行sudo mn -c清理

2. 启动控制器

控制器可以放置于虚拟机或者自己的PC，如果要使用远程控制器，搭建拓扑时需要按照上图做出适当配置，包括修改Port为6653(Floodlight默认端口)、修改类型为 remote controller，并填写远程IP(即你自己本机PC的IP地址)。

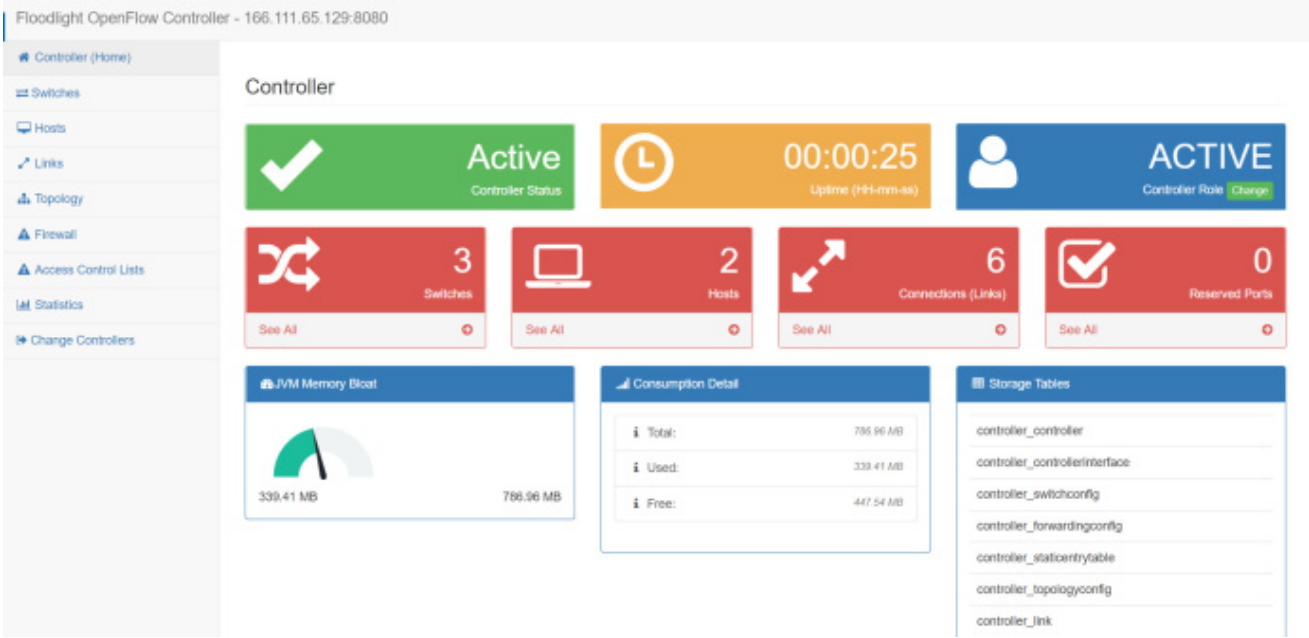
本机需要操作包括将虚拟机目录下的Floodlight文件夹拷贝到自己的PC（需要安装JAVA环境，相信大家已经搞定了，另外远程控制器需要临时关闭自己PC的防火墙，否则无法连接服务），具体操作如下即可启动控制器：

```
java -jar target/floodlight.jar -cf src/main/resources/learningswitch.properties // 进入floodligth文件夹后执行
```

其后的配置文件需要根据具体任务而更改，比如替换为floodlightdefault.properties等，这里需要注意为允许远程访问，learningswithch配置文件需要加一行内容如下：

```
net.floodlightcontroller.restserver.RestApiServer.accessControlAllowAllOrigins=TRUE
```

配置成功后，可以在浏览器里输入 http://floodlight-ui.lab.fib-lab.com:8080/pages/index.html 登录控制器Web界面，并选择Change Controller选项根据提示登录后，可以看到如下画面，通过该界面观察与控制器相连的交换机、终端 Host、拓扑以及交换机中的流表信息，并作记录。【Point0】



3.控制器功能实验

1）基于 MAC learning 的转发实验

组内的每位同学在不同的主机上h0-h5，ping其他主机h0-h5，在此过程中：

- 观察并记录 ping 的延时的变化【Point1】；
- 观察并记录交换机中流表的变化【Point2】；
- 将拓扑连成环状，观察 ping 的结果是否有变化，如有变化，并向助教解释分析可能的原因【Point3】。

2）基于 forwarding 模块的转发实验

停止原来运行的控制器程序（在 Floodlight 的进程终端中 ctrl+c），将配置文件替换为forwarding.properties 后重新运行。重复 1）中的过程，观察两者结果的差别，并向助教解释分析看到的现象【Point4】。

3） Floodlight RestAPI 调用（选作）

Floodlight 提供了向某个 OpenFlow 交换机中写入静态流表的 API。请通过该API 向自己的交换机中写入流表，过滤掉发送到相邻同学的所有 IP 数据包，通过ping测试结果；然后删除该流表，再次进行ping测试。 提醒： 在写流表时需要正确的设置与协议相关的匹配域，例如 IP 协议对应的 ether-type为 0x0800。 与静态流表相关 RestAPI 的介绍及调用方法请参考如下网址： https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343518/Static+Entry+Pusher+API

五、实验验收

如实验过程所示，记录实验现象，并向助教解释相应原因，本次实验不需要提交实验报告。