# 四、SDN虚拟环境搭建

## 4.1实验的网络技术

### 4.1.1 Mininet仿真平台

使用源码编译安装的方式安装mininet,安装步骤如下：

1. 更新软件

# apt-get update

# apt-get upgrade

如果ubuntu没有安装git,需要安装

# apt install git

1. 从github上获取Mininet源码

# git clone git://github.com/mininet/mininet

获取源码后可以查看当前获取的Mininet版本，在~/mininet目录下，可以通过git tag 命令列出所有可用的Mininet版本

# cd mininet

# git tag

或者你想安装的任意版本

# git checkout -b 2.2.1 2.2.1

1. 获取源码树并安装Mininet

#cat INSTALL

切换到mininet文件下：

# cd mininet 注意：如果前面进行了cd mininet此时则不需要再cd mininet

这里有多个安装选项：mininet/util/install.sh[options]   
“-a”：完整安装包括Mininet VM，还包括如Open vSwitch等依赖关系软件，以及像的OpenFlow Wireshark和POX。默认情况下，这些工具将被安装在你的home目录中。完整安装命令：

# ./util/install.sh -a 我选择的是完整安装，装完这步可以跳到（4）了

“-nfv”：安装Mininet、基于OpenFlow的交换机和Open vSwitch。命令：

# ./util/install.sh –nfv

“-s mydir” ：使用此选项可将源代码建立在一个指定的目录中，而不是在home目录中。

#./util/install.sh -s mydir

另外，你只想安装OpenFlow1.3和Open vSwitch2.3.0，可以使用安装命令：

# mininet/util/install.sh –n3V 2.3.0

1. 安装完成后可通过简单的命令测试Mininet的基本功能

# sudo mn --test pingall

可以查看安装好的Mininet版本：

# mn --version

### 4.1.2 Ryu控制器

安装RYU，具体步骤如下：

（1）安装前先更新一下系统：

sudo apt-get upgrade

（2）安装RYU,需要安装一些python的套件：

$ sudo apt-get install python-pip python-dev build-essential

$ sudo pip install --upgrade pip

$ sudo apt-get install python-eventlet

$ sudo apt-get install python-routes

$ sudo apt-get install python-webob

$ sudo apt-get install python-paramiko

（3）安装RYU，采用获取源码的方式，也可以采用pip安装方式（ pip install ryu 即可）

$ git clone git://github.com/osrg/ryu.git

$ cd ryu

$ sudo pip install -r tools/pip-requires

$ sudo python setup.py install

1. 测试

$ cd

$ ryu-manager

## 4.2实验的设计过程

### 4.2.1 拓扑结构设计

数据中心是对SDN应用较快较广泛的地方，其中最迫切地需求是为服务器群组提供高效的双向带宽互联，通常采用层次性多根网络拓扑，其中胖树（fat-tree）结构因其简单易用而得到广泛应用，本实验基于胖树结构的SDN网络进行仿真，胖树网络的好处是具有一个多层次的树状拓扑结构固有的容错能力。

Fat-tree拓扑结构是由MIT的Al-Fares等人在改进传统树形结构性能的基础上提出的，属于switch-only型拓扑。整个拓扑网络分为三个层次，自下而上分别为边缘层（Edge）、汇聚层（Aggregate）以及核心层（Core），其中汇聚层交换机与边缘层交换机构成一个pod。Fat-tree构建拓扑规则如下：FatTree拓扑中包含的Pod数目为 k，每一个pod连接的sever数目为，每一个pod内的边缘交换机及聚合交换机数量均为，核心交换机数量为，网络中每一个交换机的端口数量为k，网络所能支持的服务器总数为

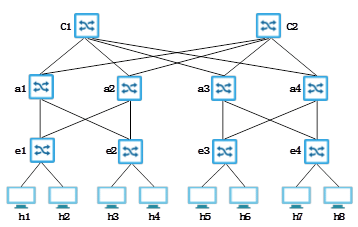
FatTree结构采用水平扩展的方式，当拓扑中所包含的pod数目增加，交换机的端口数目增加时，FatTree拓扑能够支持更多的服务器，满足数据中心的扩展需求，如k=48时，FatTree能够支持的服务器数目为27648。

FatTree结构通过在核心层多条链路实现负载的及时处理，避免网络热点；通过在pod内合理分流，避免过载问题。

FatTree对分带宽随着网络规模的扩展而增大，因此能够为数据中心提供高吞吐传输服务；不同pod之间的服务器间通信，源、目的节点之间具有多条并行路径，因此网络的容错性能良好，一般不会出现单点故障；采用商用设备取代高性能交换设备，大幅度降低网络设备开销；网络直径小，能够保证视频、在线会与等服务对网络实时性的要求；拓扑结构规则、对称，利于网络布线及自动化配置、优化升级等。

Fat-Tree结构也存在一定的缺陷：Fat-Tree结构的扩展规模在理论上受限于核心交换机的端口数目，不利于数据中心的长期发展要求；对于Pod内部，Fat-Tree容错性能差，对底层交换设备故障非常敏感，当底层交换设备故障时，难以保证服务质量；拓扑结构的特点决定了网络不能很好的支持one-to-all及all-to-all网络通信模式，不利于部署MapReduce、Dryad等现代高性能应用；网络中交换机与服务器的比值较大，在一定程度上使得网络设备成本依然很高，不利于企业的经济发展。

一个简单的基于两个数据中心的网络拓扑如下图：



图中c1和c2为核心交换机，a1-a4为聚合交换机，e1-e4为边缘交换机，h1-h8为主机。

Mininet提供了一个内置的可视化工具miniedit，因此可以直接使用可视化工具通过拖拽图标与修改属性的方式搭建出理想的拓扑，也可以通过编写python文件来自定义拓扑结构。

使用Mininet设计模拟不同网络数据中心拓扑，可以用来分析网络的总流量，而且除此之外，还可以通过负载均衡策略来保证数据中心的可用性。不同数据中心网络拓扑管理设计，主要是基于胖树（Fat-Tree）拓扑创建网络，胖树网络的好处是具有一个多层次的树状拓扑结构固有的容错能力。

构造自定义拓扑主要使用以下三个函数：

1. 增加主机

addHost('hostname')

1. 增加交换机

addSwitch('switchname')

1. 增加链路

addLink(node1,node2,node1\_port,node2\_port)

### 4.2.3 开发Ryu应用

Ryu是控制器，通过下发流表给交换机来进行数据包转发决策。首先，我们应该知道：

控制器和交换机握手完成之后，一条默认流表项应该被添加进流表。它的作用是为了让交换机发送Packet-In消息。这条默认流表项被称为Table-Miss，不过Table-Miss不是默认存在的，他也是由控制器下发的。我们在控制器代码里面定义的MAC地址表是只有控制器知道的，对于交换机来说，只有流表才能帮助它转发，所以控制器要给交换机下发流表，如果你不在交换机添加详细的流表项，那么你应该让交换机收到每一个数据包都上报控制器，由控制器下发Packet-Out报文进行处理，否则数据包会被丢弃，控制器下发的流表项包含了应该对数据包进行的动作。

**一、**编写Ryu组件addtag.py,功能是实现打标签。

首先是继承Ryu的app\_manager.RyuApp基类，然后定义以下函数：

1. switch\_features\_handler函数的作用就是下发一条Table miss流表，用于将不知如何转发的数据包上报控制器。鉴于流表项的特殊性，我们在交换机向控制器发送完自身信息的时候让控制器下发这条流表项，这个时候修饰器会调用这个函数。这条流表项需要具有以下特点：优先级最低，保证最后匹配；必须要能匹配所有数据包。所以我们指定了一个空的match，并且把这条流表项的优先级设置为0，将OFPCML\_NO\_BUFFER指定为max\_len。调用的add\_flow函数是我们自己定义的。
2. add\_flow函数的参数表较多，因为这些在下发流表的时候调用的FlowMod类里面都要用到。所以简要的介绍一下用到的参数（括号内是默认值）:datapath：是传来数据的交换机，内有ID编号;priority(0)：指定流表项的优先级，影响匹配顺序，优先级越大越先匹配;match(None)：流表项要匹配的内容，比如进入接口，目的地址等;buffer\_id（ofproto\_v1\_3.OFP\_NO\_BUFFER）:指定在OpenFlow交换机上缓冲的数据包的缓冲区ID。当未指定缓冲区ID时，请设置OFP\_NO\_BUFFER;instructions ([]):交换机要执行的指令列表。指定好所有的值最后就调用send\_msg来发送流表项。
3. delete\_flow函数即负责删除流的函数
4. push\_tag\_action函数实际定义了一个操作数据包的动作，该动作负责添加vlan或者修改vlan，本实验以链路号作为标签，把标签保存在vlan中，因此vlan的值即为链路号的有序组合，在该函数中，需要编写代码来实现打标签的逻辑，比如需要判断链路号是否需要作为标签存入vlan中（当该链路是唯一可达路径上的一部分时，该链路号不需要作为标签保存），需要判断前面是否已经存入了标签来决定保存该标签的方式等。
5. \_packet\_in\_handler函数要完成的功能比较多，首先它需要获取进入的接口，源和目的MAC地址，然后填充MAC地址表，注意一个细节，我们需要根据交换机编号和源地址来指定接口，因为同一个接口可能会对应不同的MAC地址。然后做一下判断，如果MAC地址表里面没有目的MAC地址，就设置为广播；有地址就设置为出接口。然后是添加流表项操作，我们只能在找到了目的地的情况下才添加流表项，对于本次实验来说，定义流表项的动作集时，填充的最关键的动作就是打标签的动作，调用push\_tag\_action函数定义该动作。这里有个概念buffer\_id需要理解，buffer\_id: 指定openflow交换机上封包对应的缓冲区，若不需要，则指定为OFP\_NO\_BUFFER。openflow中buffer\_id分别在三类消息中定义，并且起到的作用均是不同的，第一类、Packetin消息：用于标记缓存在交换机中的数据报文id，如报文被action上送到控制器中maxlen字段或者table\_miss消息限制长度，而通过bufferid将报文缓存在交换机中，以便被另外两种消息来调用；第二类、Packetout消息：用于控制器将原先buffer在交换机中的报文，通过Packetout个形式从交换机的某个物理口送出去；第三类、Flowmod消息：如果flowmod中带有bufferid，那么说明这个flowmod需要做两件事情，第一是正常下发一条flow，其次是把交换机中先前buffer的那个数据报文，Packetout到table来匹配一次下的这条flow；以上两个指令都是通过这个带有bufferid的消息执行的，不需要控制器另外下packet\_out消息，这种设计思路是非常巧妙的。

二、编写Ryu组件getpath.py，功能是提取数据包的标签，回溯路径并记录当前时间。

该组件的目的不是定义数据包转发规则，而是读取数据包内容，因此\_packet\_in\_handler函数中主要需要实现的是，按照保存标签的逻辑提取出标签并组成队列，对队列中相邻的元素进行树的广度优先遍历来找出两个元素之间的唯一路径，最后将这些路径连接起来组成一条完整的路径。

### 4.2.4 Mininet与Ryu连接

本实验定义了胖树网络拓扑，topo.py，在该文件中，使用net.addController函数（该函数有两个参数，第一个参数为控制器的名称，第二个参数为控制器的监听端口）添加了控制该拓扑的两个控制器con0（监听的端口号为6633）和con1（监听的端口号为6634），让con0与拓扑中的所有交换机相连，con1则只与边缘交换机相连。

打开终端1，运行：sudo ryu-manager addtag.py --ofp-tcp-listen-port=6633

打开终端2，运行：sudo ryu-manager getpath.py --ofp-tcp-listen-port=6653

打开终端3，运行：sudo python build\_topo.py进入mininet终端，在mininet的终端可进行数据包的ping操作，如果在实现组件的时候有将结果展示到终端，那么可以在终端1,2看到相应的结果。

### 4.2.6 联合导航系统

不知道？？？