目录

[软件定义网络mininet实验 1](#_Toc130411820)

[环境配置 1](#_Toc130411821)

[搭建fat tree拓扑 1](#_Toc130411822)

[原理说明 1](#_Toc130411823)

[实验代码 2](#_Toc130411824)

[交换机配置 3](#_Toc130411825)

[查看各主机ping通情况 4](#_Toc130411826)

[分析数据包路径 5](#_Toc130411827)

# 软件定义网络mininet实验

## 环境配置

本次实验在VMware虚拟机软件上完成，操作系统为Ubuntu22.04.安装过程与指导书上一致，故此处不给出安装命令。

另：本人电脑python命令无法执行，用python3命令替换之。

## 搭建fat tree拓扑

### 原理说明

fat tree原理：Fat-Tree是以交换机为中心的拓扑。支持在横向拓展的同时拓展路径数目；且所有交换机均为相同端口数量的普通设备，降低了网络建设成本。

Fat-Tree结构共分为三层：核心层、汇聚层、接入层。一个k元的Fat-Tree可以归纳为5个特征：每台交换机都有k个端口；核心层为顶层，一共有(k/2)^2个交换机；一共有k个pod，每个pod有k台交换机组成。其中汇聚层和接入层各占k/2台交换机；接入层每个交换机可以容纳k/2台服务器，因此，k元Fat-Tree一共有k个pod，每个pod容纳k\*k/4个服务器，所有pod共能容纳k\*k\*k/4台服务器，任意两个pod之间存在k条路径。常见的有2元、4元、6元等结构。本次要搭建的4层结构如图1所示。

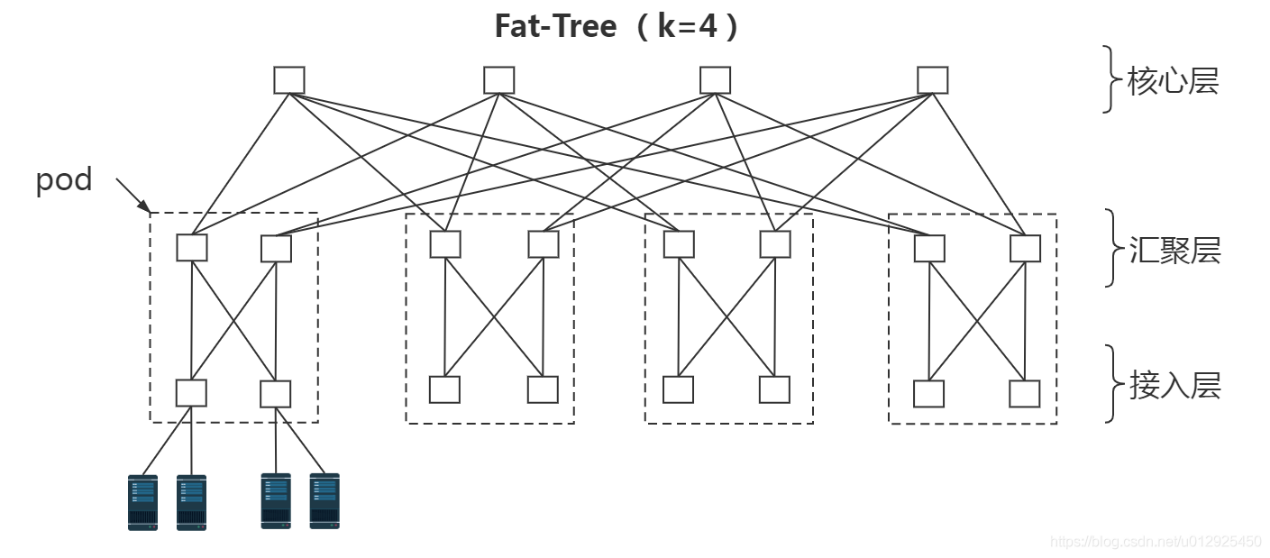
Fat-tree使用了多个小规模、低成本的交换机替代了原来旧式设计的核心层、汇聚层昂贵的、高密度端口的交换机。相同点是，也都是三层架构，核心层、汇聚层、接入层。

图1 fat-tree拓扑结构（k=4）

这种设计的好处：

实惠且任一组件都容易替换；

非阻塞，任一主机可获得100%完整带宽；

Fat-tree架构的拓扑连接特点：

汇聚层k/2的端口连接k/2的核心交换机，剩下的端口连接接入层的交换机；

接入层同上，一半的端口连接上一层，一半的端口连接下面的主机。

以k=48的拓扑图为例，计算各个节点的数量，如下：

有48个pod；

每个pod有24个接入层switch；

每个接入层switch连接24个host；

### 实验代码

本次实验代码如图2所示。其中为了停用控制器在Mininet函数参数中添加了controller=None，其余代码较为基础，且大部分在参考书上有介绍，故此处不予详解。

实验代码为python3编制。为方便起见，用列表来保存了所有交换机与主机。



图2 本次实验所用代码

### 交换机配置

按照指令书上命令，分别对每台交换机执行以下命令：

sudo ovs-vsctl set bridge s1 stp\_enable=true #开启STP，s1为设备名

sudo ovs-vsctl get bridge s1 stp\_enable

sudo ovs-vsctl del-fail-mode xx

由于交换机数量较多，故采用循环处理命令，见图3：

## 查看各主机ping通情况

图3 交换机所用命令

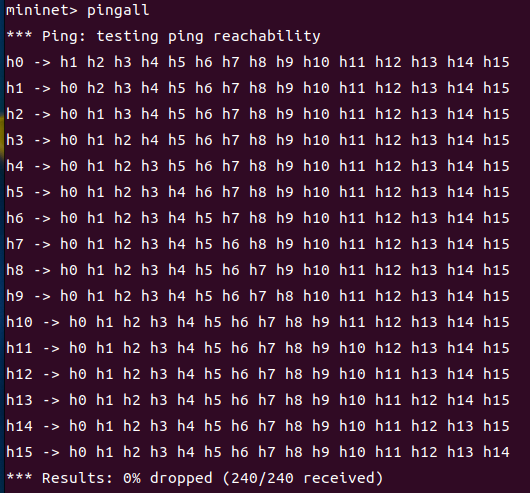
所有主机均能互相ping通，见图4.

图4

## 分析数据包路径

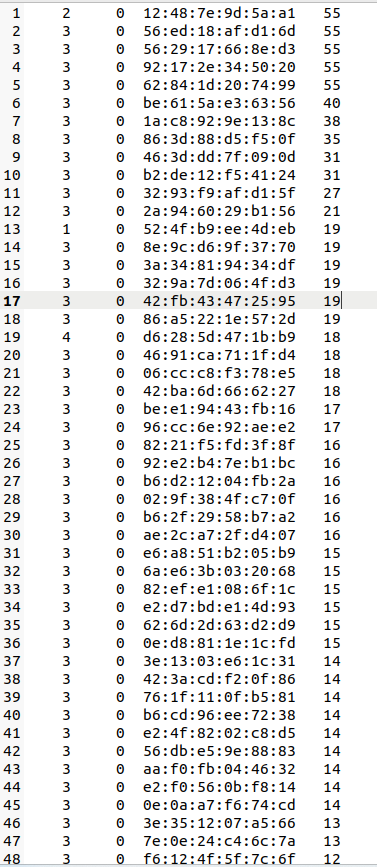
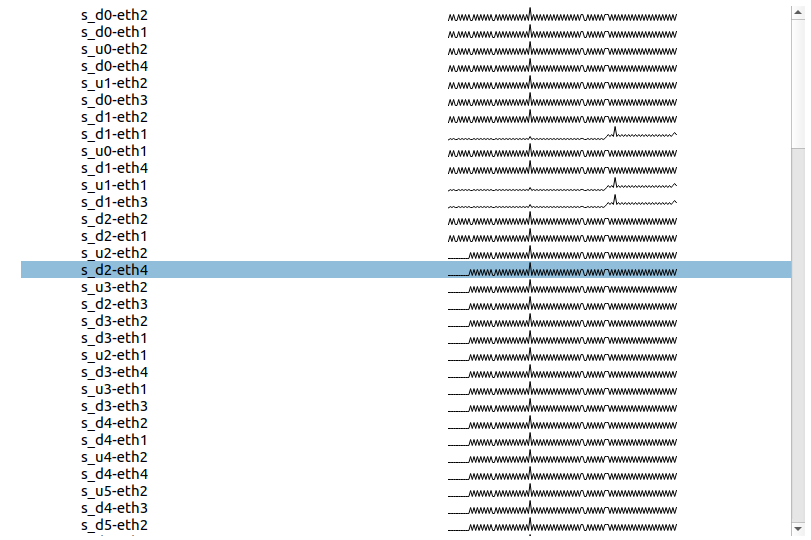
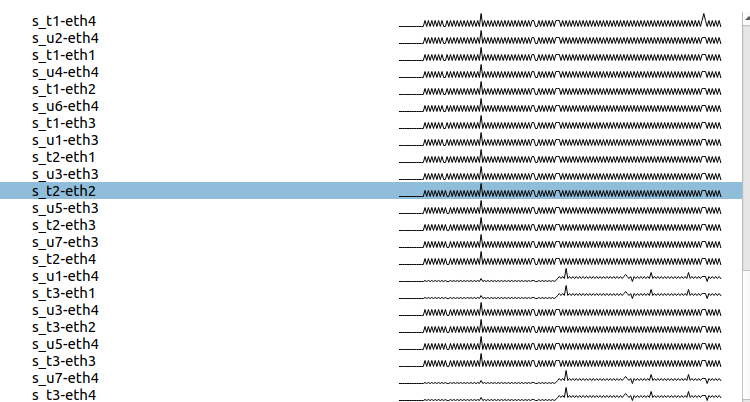
图5为s\_d1处在pingall之后的交换机MAC表。

Figure 图5 MAC表

由图中可得，除了3个MAC地址分别指向端口1、2与4、其余都发向端口3，由数据与端口号不难得出，3个只被分配了一个MAC地址的端口分别直接连通两台主机与另一台交换机，且其对应的MAC地址就是这三台设备的MAC地址，而其余设备的MAC地址全部通过端口3到达。故s\_d1在收到向除了直接连接的4台设备发送信息均通过3号端口。

其余交换机类似。

另：命名方法：核心层为s\_t(top),汇聚层为s\_u(up),接入层为s\_d(down)，数字为从左到右编号，从0开始。

此处我们用h3向h15发送数据包（持续ping直到按ctrl+c），用wireshark抓包分析可得以下波形：

可以很明显的看出d7的1接口，3接口，u7的1接口，4接口，t3的4接口，1接口、u1的4接口，1接口，d1的1接口，3接口均有信息，而其他接口除生成树协议外无其他信息，故可推断出从h1到h15的信息路径为h1->s\_d1.eth1->s\_d1.eth3->s\_u1.eth1->s\_u1.eth4->s\_t3.eth1->s\_t3.eth4->s\_u7.eth4->s\_u7.eth1->s\_d7.eth3->s\_d7.eth1->h15,与推断一致，根本原因是选择路径时从较小数字到较大数字建立连接且信道速度相同，故交换机在相同情况下更倾向于选择较小编号的交换机。而当部分信道拥塞时，生成树协议与路由选择方法又会使得路径改变。