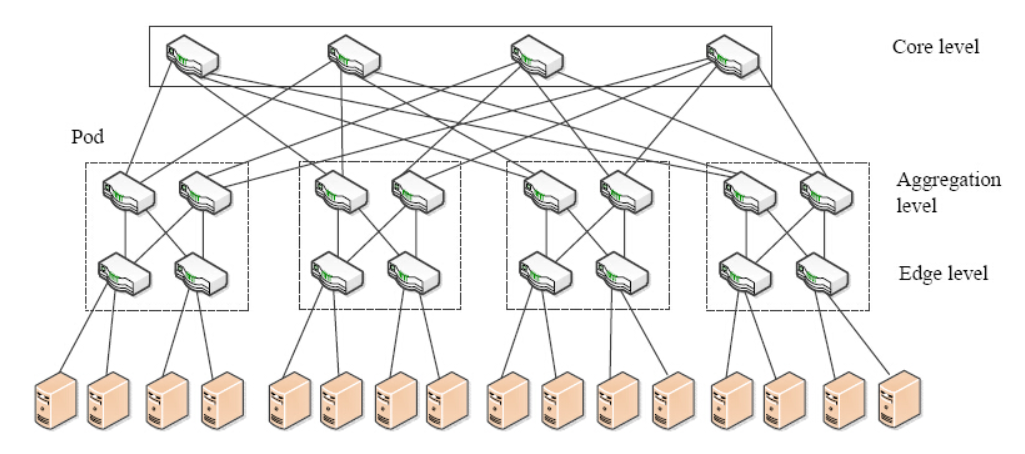
# 实验报告

1. **实验原理**
2. 数据中心拓扑结构

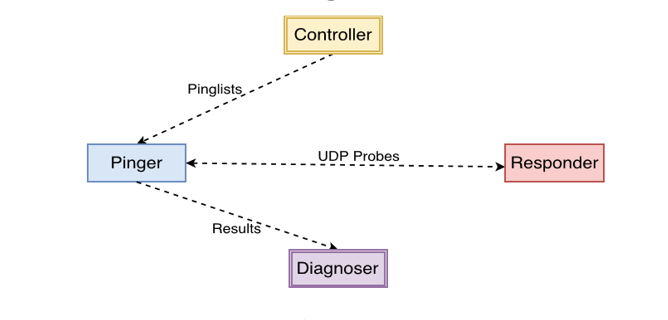
Fat-Tree拓扑结构是由MIT的Al-Fares等人在改进传统树形结构性能的基础上提出的，属于switch-only型拓扑。整个拓扑网络分为三个层次（如下图所示）：自下而上分别为边缘层（Edge）、汇聚层（Aggregate）及核心层（Core），其中汇聚层交换机与边缘层交换机构成一个Pod，交换设备均是采用商用交换设备。



Fat-Tree构建拓扑规则如下：Fat-Tree拓扑中包含的Pod数目为k，每一Pod连接的server数目为(k/2)2，每一Pod内的边缘交换机及聚合交换机数量均为k/2，核心交换机的数量为(k/2)2，网络中每一交换机的端口数目为k，网络所能支持的服务器总数为k3/4。Fat-Tree结构采用水平扩展（scale-up）的方式，当拓扑中包含的Pod数目增加，交换机的端口数目增加时，Fat-Tree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，Fat-Tree能够支持的服务器数目为27648；Fat-Tree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。

1. deTector原理

deTector包括四个松散耦合的组件：控制器，诊断器，探测器和响应器，如下图所示。



控制器。 逻辑控制器周期性地构建探测矩阵，指明了发送探测器的路径。 我们主要关注交换机间链路故障定位，因为连接服务器与ToR交换机的链路故障可以很容易地识别。 探测矩阵表示ToR之间的路径，因此我们不依赖具有ping功能的ToRs，探测由每个ToR下的2-4个选定的服务器发送。

探测器。 每个指针从控制器接收Pinglist，其中包含目标服务器，探测格式和ping配置。从ToR交换机到不同目的地的探测路径分布在ToR交换机下的探测器的Pinlist中，每个路径分配到至少2个探测器以进行容错。以这种方式，如果一个探测器停机，则同一机架中的其他探测器仍然可以探测路径，避免链路覆盖范围的大幅下降。为了检测连接服务器和相应ToR的链路的故障，探测器还负责探测同一个ToR下的其他服务器。即使对于一个大的DCN，每个探测器的探测路径数量不应超过一百个。探测数据包是通过UDP发送的。尽管TCP用于携带DCN中的大部分业务，但在绝大多数情况下，DCN并不区分TCP和UDP流量，因此UDP探测器也可以显示网络性能。当探测器检测到探测损失时，通过发送两个相同内容的探测数据包来确认丢失模式。

响应器。 响应者是在所有服务器上运行的轻量级模块。 响应者收到探测数据包后，回传。 响应者不保留任何状态，并且所有探测结果都由探测器记录。

诊断器。 每个探测器记录丢包信息，并将其发送给诊断器进行丢失定位。 这些日志将保存到数据库中进行实时分析和后续查询。 诊断器运行PLL算法以确定数据包丢失并估计可疑链路的丢失率。

1. **算法流程**

deTector以三个步骤循环工作：路径计算，网络探测和损失定位。

（1）路径计算。

在每个周期的开始，控制器从数据中心管理服务器读取数据中心拓扑和服务器运行状况，并选择最小数量的探测路径。 然后，控制器在每个ToR中选择探测器，构建并分派Pinglist给探测器。

（2）网络探测。

接下来，探测数据包沿DCN的指定路径发送。由于数据中心通常采用ECMP进行负载平衡，因此我们必须使用源路由来控制每个探测包所走路径，这可以通过各种方法来实现.一般而可行的解决方案是采用分组封装和解封装来创建端到端的管道，尽管它可能导致在VXLAN或NVGRE创建的虚拟化网络中两次封装数据包。 以Fat-Tree网络为例：固定核心交换机，两个pod内的服务器之间只有一条路径; 我们可以使用IP-in-IP封装服务器上的探测包；在数据包到达核心交换机之后，外部报头被去除，数据包被路由到真实的目的地。 这种源路由机制在服务器和核心交换机上几乎没有开销。

(3)损失定位

在诊断器上，通过我们的损失定位算法探测损失进行聚合和分析，精确定位故障链路，估计损失率。

1. **源码清单**

https://github.com/yhpeng-git/deTector

1. **实验内容**

1.实验环境

1、python 2.7.6

2、使用mininet建立虚拟拓扑

2.论文需要完成的实验内容

1、构建一个4-ary的Fattree实现1-identifiability，3-coverage

2、探测矩阵的构造

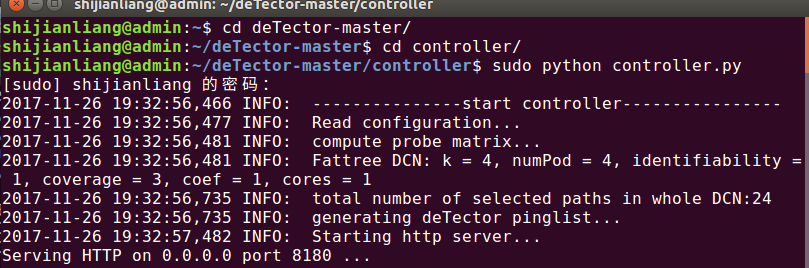
3、数据中心网络的网络探测列表

4、对故障链路的损失定位

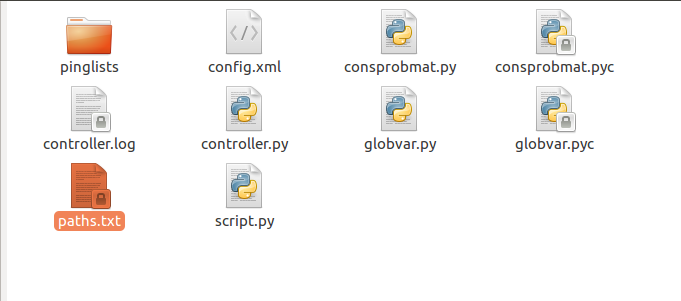
3.已完成的实验内容

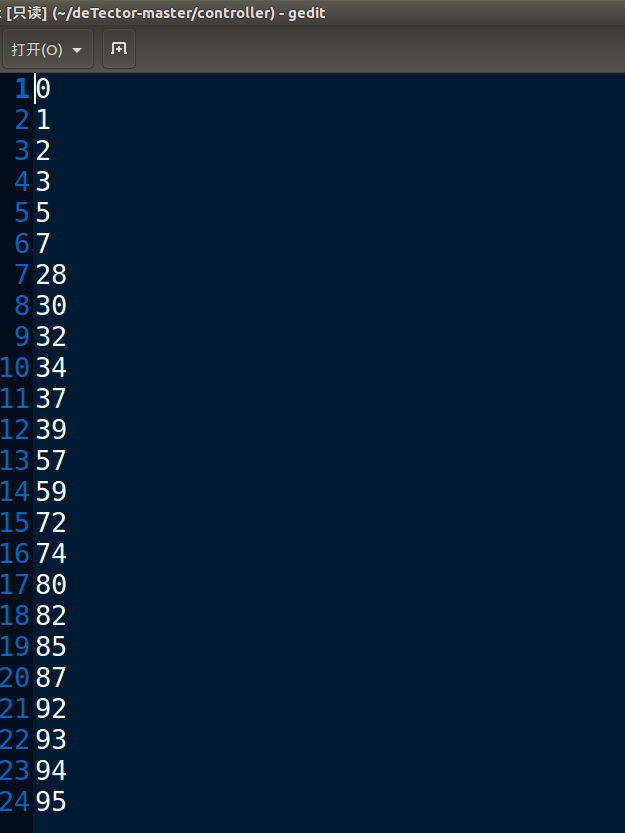
①我们小组创建了一个4-ary的Fattree拓扑，并实现了1-识别 、3-覆盖探测矩阵的构造，具体细节如下：

1）启动controller，实现探测矩阵的的计算



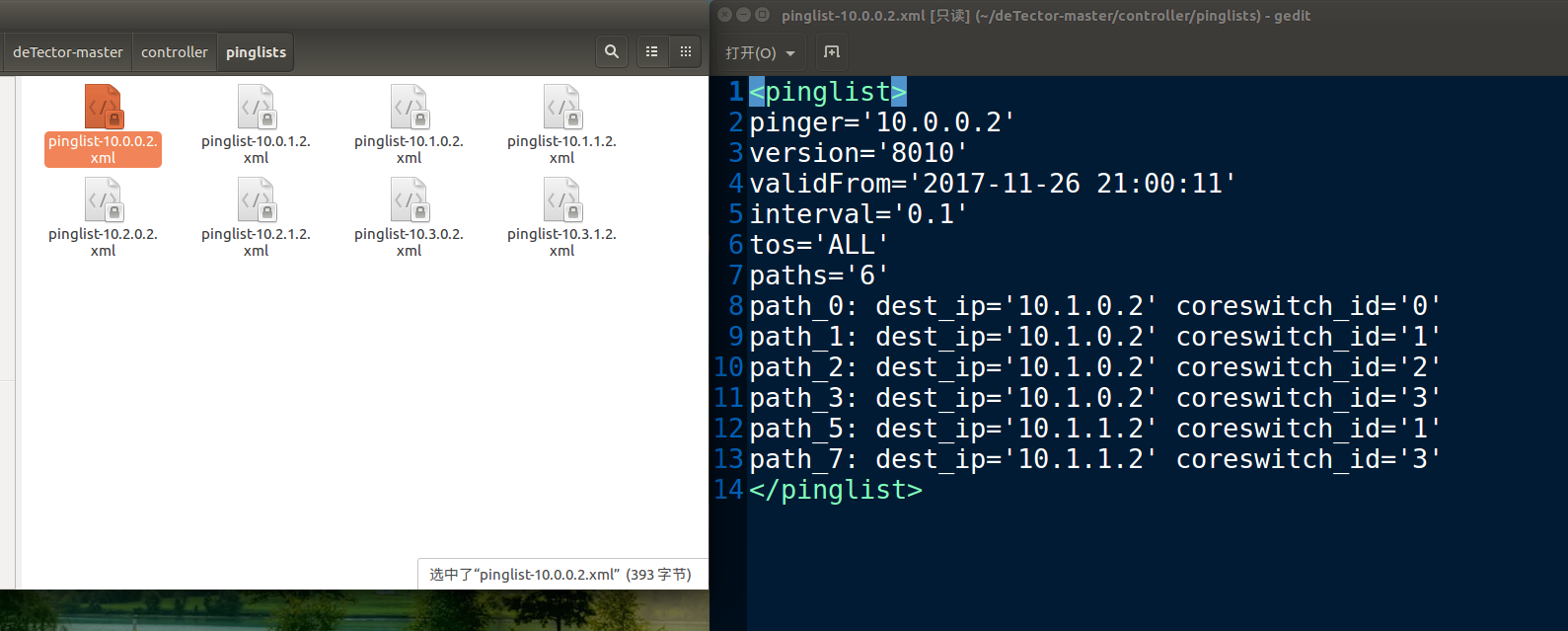
2） 生成探测路径并周期性地产生pinglists

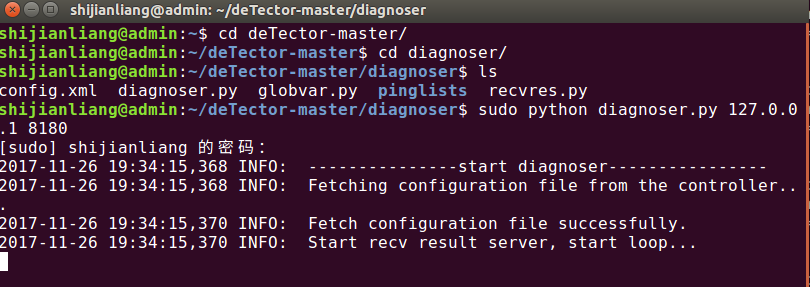
****

****

最小化探测路径：

周期性生成的pinglist:

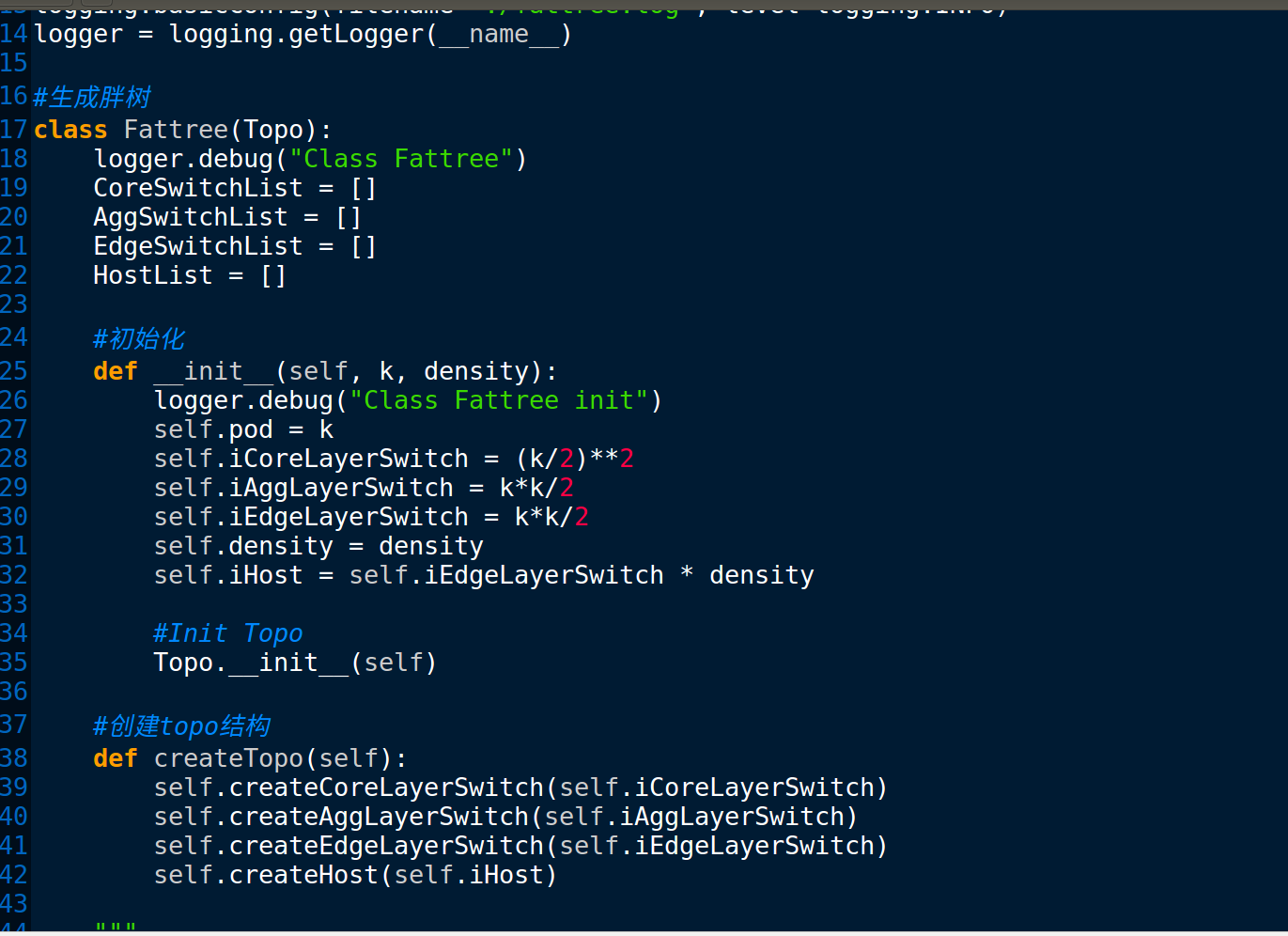


3）diagnoser 启动

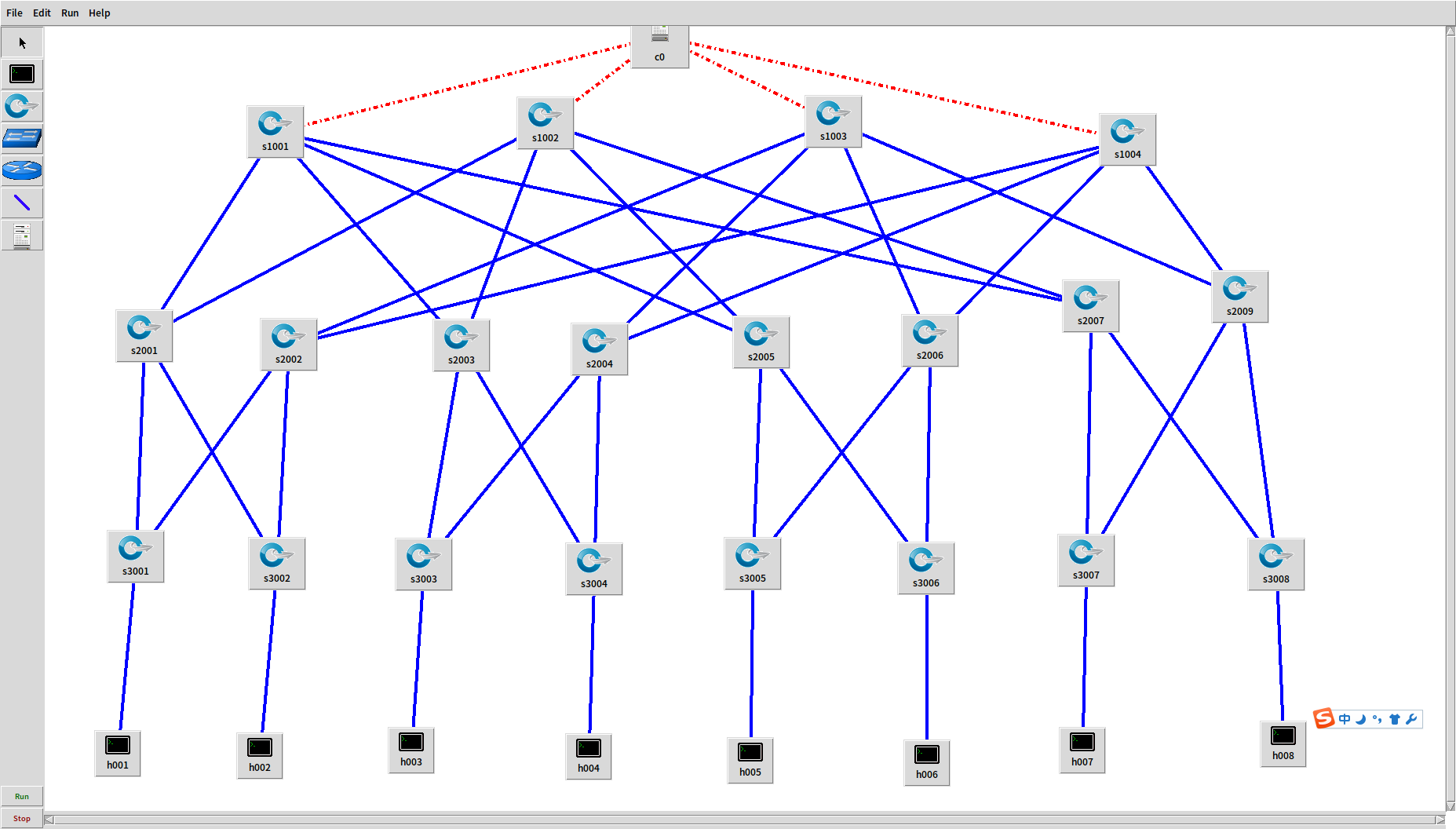
4）responder启动 , 需要在相应的服务器上运行，受条件限制，暂无法运行

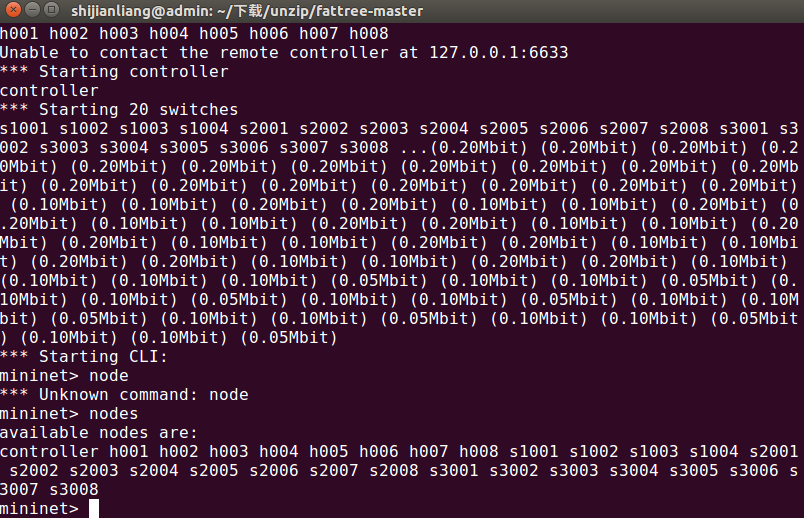
5）pinger启动，也需要在相应的服务器上运行，受条件限制，暂无法运行

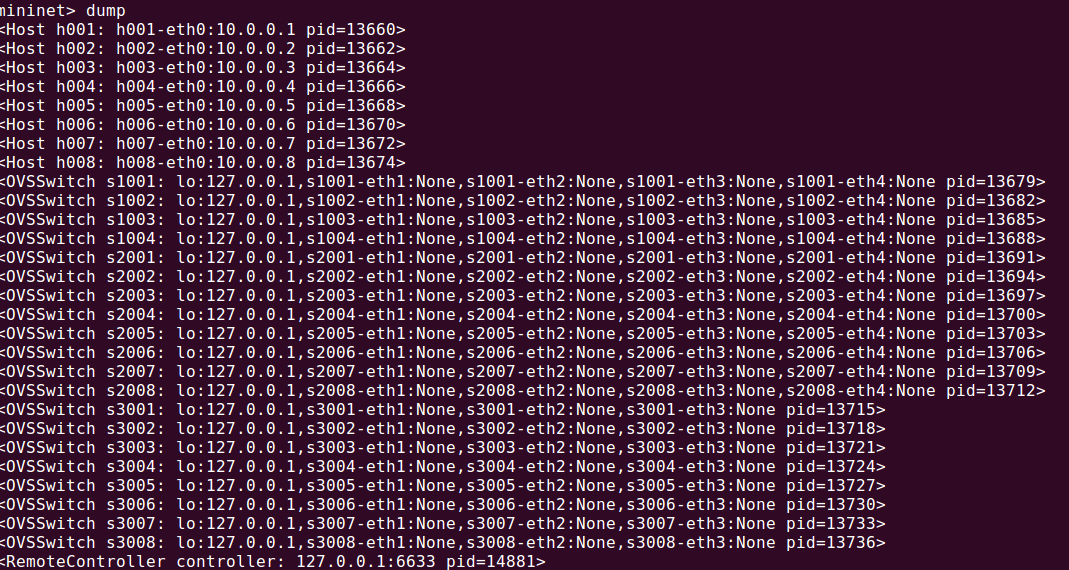
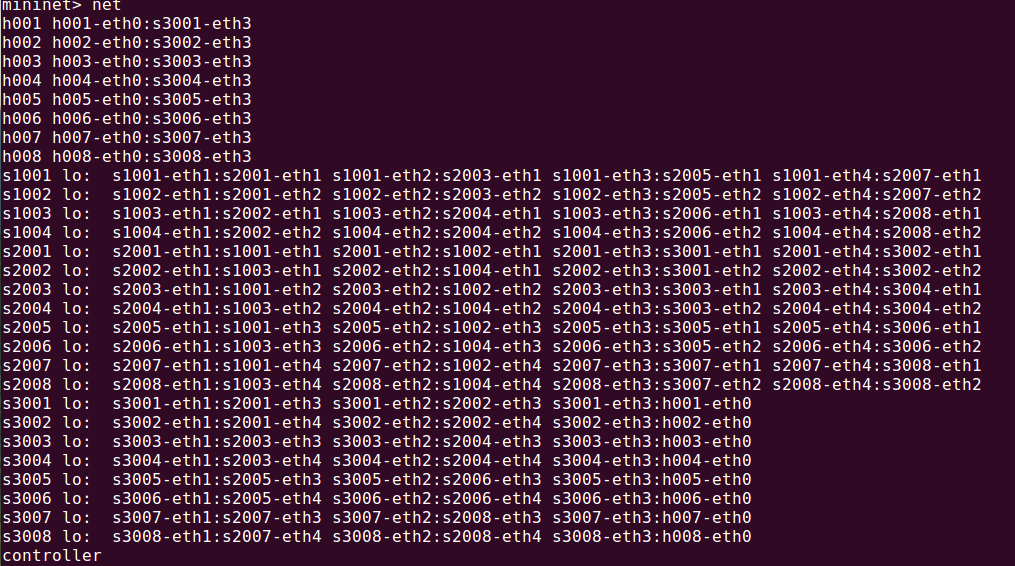
②我们小组在mininet下构建了一个Fattree拓扑

1) Fattree的部分代码截图

2) Fattree的可视化图形界面

****

3）启动并运行Fattree

****4）Fattree 的具体信息

1. **实验结果**

我们小组由于时间和设备等条件有限，没有完全重现论文中的实验。我们在虚拟机上搭建了Fattree的拓扑结构，并进行了网络的探测路径最小化和controller周期性的构建探测矩阵，实现了deTector的路径计算工作，实现1-identifiability和3-coverage；对于网络探测和损失定位工作，由于实验设备有限，pinger和responder无法在其他服务器上运行