

Lab 1 Tutorial

Aim

1. 使用 Mininet 的Python API搭建 k=4 的 fat tree 拓扑;
2. 使用 pingall 查看各主机之间的连通情况;
3. 若主机之间未连通, 分析原因并解决 (使用 wireshark 抓包分析)
4. 若主机连通, 分析数据包的路径 (ovs-appctl fdb/show 查看MAC表)
5. 不可以使用controller

Reference

- [Mininet](#)
- [OpenVSwitch](#)
- [WireShark](#)

Part 1 实验环境搭建

方式1:

- 使用 virtual box 镜像搭建 虚拟机软件 virtual box , 可在[官网](#)查询
- 本实验提供 virtual box 虚拟机的镜像文件 (sdn_exp_2023.ova) 已配置 Mininet 和 Ryu
- 环境搭建步骤如下: 安装 virtual box 导入镜像文件 sdn_exp_2023.ova (root 账户密码并未设置, 需要的同学可以参考 sudo passwd root 指令)

方式2:

- 使用 VMWare 镜像搭建 虚拟机软件 VMWare , 可在官网自行下载
- 本实验也提供 VMWare 虚拟机镜像文件 (sdn_exp_2023_vmware) , 已配置 Mininet 和 Ryu
- 环境搭建步骤如下:
 - 安装 VMWare
 - 导入镜像文件 sdn_exp_2023_vmware (密码 sdn)

方式3:

源码安装

```
bash # 参考视频
2 # `Workstaion`和`Ubuntu`的安装: https://www.bilibili.com/video/BV1ng4y1z77g
3 # SDN环境搭建 (`Mininet`) : https://www.bilibili.com/video/BV1nC4y1x7Z8
4
5 # 安装mininet
6 git clone https://github.com/mininet/mininet.git
7 cd mininet/util
8 sudo ./install.sh -n3v
9
10 # 安装wireshark
11 sudo add-apt-repository ppa:wireshark-dev/stable
12 sudo apt update
13 sudo apt install wireshark
```

说明:

针对 Apple Silicon

- 本质上安装任一Linux虚拟机都可以使用
 - VMware
 - VirtualBox
 - Parallels Desktop
 - ...
- 但是Orbstack不行!
 - 它轻量化, 既可以开docker也可以开vm
 - 它的linux虚拟机内核没有支持openvswitch
 - 除非你手动编译内核:)
- 具体问题详见笔者在mininet-github中提出的[issue](#)

Part 2 实验工具介绍

2.1 三板斧

- mininet: 用来在单台计算机上创建一个包含多台网络设备的虚拟网络
- Open vSwitch: Mininet 中使用的虚拟交换机
- WireShark: 抓包工具

2.2 Mininet

- 启动

shell prompt

`mn -h` # 查看mininet命令中的各个选项

`sudo mn -c` # 不正确退出时清理mininet

`sudo mn` # 创建默认拓扑，两个主机h1、h2连接到同一交换机s1

```
sdn@ubuntu:~/Desktop$ sudo mn
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

- 常用命令

inputs in Mininet CLI

`nodes`

`links`

`net` # show the whole network topo now

`dump` # show the detailed information of current net-topo

`xterm h1` # open a Terminal Simulator for node-h1

`sh [CMD]` #

`h1 ping -c3 h2` # h1 send PING to h2 for 3 times

`ping all` # PING in every node-pair

`h1 ifconfig` # lookup the Interface and configuration of h1

`h1 arp` # lookup the ARP map of h1

`link s1 h1 down/up` # disconnect/connect the link between s1 and h1

`exit` # exit the mininet CLI

```
mininet> nodes
available nodes are:
c0 h1 h2 s1
mininet> links
h1-eth0<->s1-eth1 (OK OK)
h2-eth0<->s1-eth2 (OK OK)
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo: s1-eth1:h1-eth0 s1-eth2:h2-eth0
c0
mininet> █
```

2.3 创建拓扑

CLI 创建

原始版，详见Lecture 4内容

```
sudo mn --mac --topo=tree,m,m
```

SHELL

- --mac 指定mac地址从1开始递增，而不是无序的mac，方便观察
- --topo 指定拓扑参数，可选用single和linear等参数

自建拓扑 - Method1

- 用 Mininet Python API 创建自定义拓扑
- 通过命令行运行 `pwd = 'mininet/custom/topo-2sw-2host.py'`

```
from mininet.topo
import Topo
class MyTopo( Topo ):
    "Simple topology example."
    def build( self ):
        "Create custom topo."
        # Add hosts and switches
        leftHost = self.addHost( 'h1' )
        rightHost = self.addHost( 'h2' )
        leftSwitch = self.addSwitch( 's3' )
        rightSwitch = self.addSwitch( 's4' )
        # Add links
        self.addLink( leftHost, leftSwitch )
        self.addLink( leftSwitch, rightSwitch )
        self.addLink( rightSwitch, rightHost )

topos = { 'mytopo': ( lambda: MyTopo() ) }
```

运行

```
mn> cd ~/sdn/mininet/custom
mn> sudo mn --custom topo-2sw-2host.py --topo mytopo --controller=none
```

自建拓扑 - Method2

PYTHON

```
# sudo python topo_recommend.py
from mininet.topo import Topo
from mininet.net import Mininet
from mininet.cli import CLI
from mininet.log import setLogLevel

class S1H2(Topo):
    def build(self):
        s1 = self.addSwitch('s1')
        h1 = self.addHost('h1')
        h2 = self.addHost('h2')

        self.addLink(s1, h1)
        self.addLink(s1, h2)

    def run():
        topo = S1H2()
        net = Mininet(topo)
        net.start()
        CLI(net)
        net.stop()

if __name__ == '__main__':
    setLogLevel('info') # output, info, debug
    run()
```

运行

SHELL

```
sudo python topo_recommend.py
```

2.4 OpenVSwitch (OVS)

查看交换机基本信息

SHELL

```
su
mn
ovs-vsctl show
```

- **ovs-vsctl** 命令允许查看、修改和管理 OVS 的配置，包括网桥、端口、控制器、流表等
- **show** 参数指示该命令显示当前 OVS 的配置信息
- vsctl 的全称是 "Virtual Switch Control"

```
sdn@ubuntu:~/Desktop$ sudo ovs-vsctl show
02e3be6e-16d8-44c3-9f04-998d777c591f
    Bridge s1
        Controller "ptcp:6654"
        Controller "tcp:127.0.0.1:6653"
        fail_mode: secure
        Port s1-eth2
            Interface s1-eth2
        Port s1-eth1
            Interface s1-eth1
        Port s1
            Interface s1
                type: internal
    ovs version: "2.13.8"
```

生成树协议

SHELL

```
ovs-vsctl set bridge s1 stp_enable=true # open "STP" for Sw.s1
ovs-vsctl get bridge s1 stp_enable # check if STP is open for s1
ovs-vsctl list bridge # 列出所有 OVS 网桥及其相关信息
```

查看mac表

- 启动mininet，记得要禁用控制器，否则MAC表可能学习不到内容
- 对每个Switch执行 **ovs-vsctl del-fail-mode Node_Name**，否则MAC表仍可能学不到内容
- **pingall** 令所有主机发送数据包，防止“沉默主机”现象
- **ovs-appctl fdb/show Node_Name** 查看每个节点的MAC表

```
sdn@ubuntu:~/Desktop$ sudo mn --mac --topo=tree,2,2 --controller=none
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3
*** Adding links:
(s1, s2) (s1, s3) (s2, h1) (s2, h2) (s3, h3) (s3, h4)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
mininet> nodes
available nodes are:
h1 h2 h3 h4 s1 s2 s3

sdn@ubuntu:~/Desktop$ sudo ovs-vsctl del-fail-mode s1
sdn@ubuntu:~/Desktop$ sudo ovs-vsctl del-fail-mode s2
sdn@ubuntu:~/Desktop$ sudo ovs-vsctl del-fail-mode s3
```

del-fail-mode XXX:

该选项表示“删除先前设置的故障模式”，以允许交换机重新恢复到默认行为或者进行新的配置

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
```

```
sdn@ubuntu:~/Desktop$ sudo ovs-appctl fdb/show s1
port  VLAN  MAC  Age
1      0      5a:14:cc:b4:16:a9  27
1      0      00:00:00:00:00:02  26
1      0      00:00:00:00:00:01  26
2      0      00:00:00:00:00:03  25
2      0      56:45:65:64:3b:83  24
2      0      00:00:00:00:00:04  24
sdn@ubuntu:~/Desktop$ sudo ovs-appctl fdb/show s2
port  VLAN  MAC  Age
3      0      46:82:18:f7:23:1c  29
1      0      00:00:00:00:00:01  28
2      0      00:00:00:00:00:02  27
3      0      00:00:00:00:00:03  27
3      0      56:45:65:64:3b:83  26
3      0      00:00:00:00:00:04  26
sdn@ubuntu:~/Desktop$ sudo ovs-appctl fdb/show s3
port  VLAN  MAC  Age
3      0      46:02:11:b0:73:f1  30
3      0      5a:14:cc:b4:16:a9  30
3      0      00:00:00:00:00:02  29
3      0      00:00:00:00:00:01  29
1      0      00:00:00:00:00:03  28
2      0      00:00:00:00:00:04  27
```

ovs-appctl fdb/show XXX:

显示指定网桥的转发数据库（FDB）内容

- XXX: 网桥的名称or标识符
- FDB: 是一个表, 存放的是 **MAC地址和与之关联的端口** 信息

2.5 WireShark

- 抓交换机的packet
 - **sudo wireshark**, 并选择相应的端口
- 抓主机的packet
 - 在mininet CLI中执行 **xterm h1**, 打开host1的终端
- 在h1终端里运行wireshark

Part 3 实验解析