COMP550705 软件定义网络
实验报告

第 1 次

姓名

班级

学号

日期

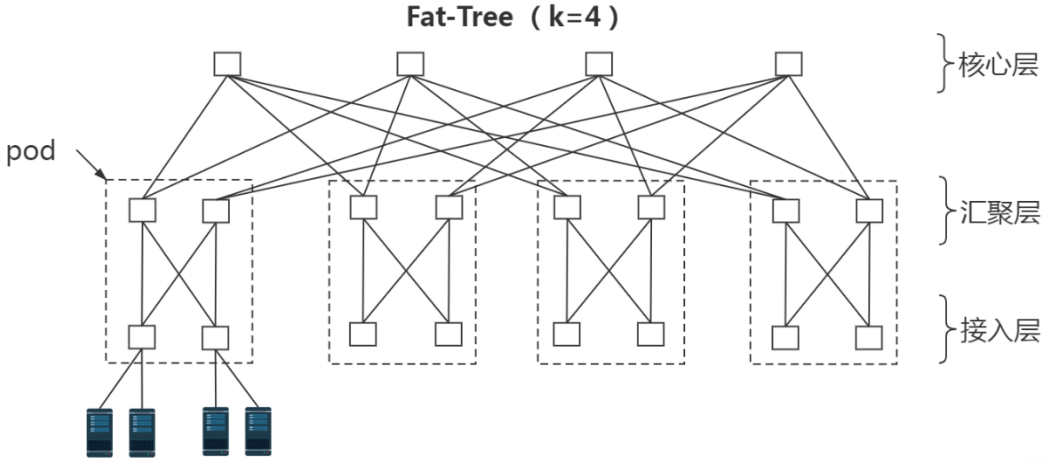
一、 Fat Tree 与实验工具简介

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。

Fat Tree 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。

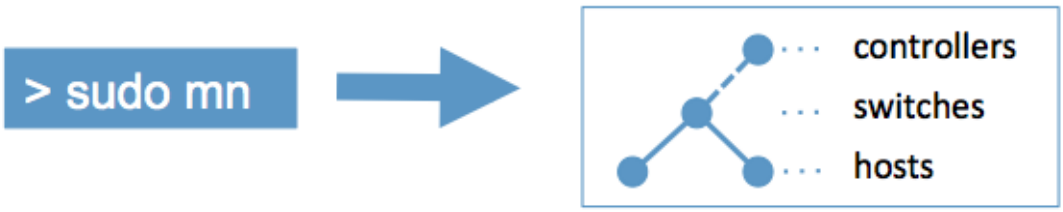
Fat-Tree 结构共分为三层：核心层、汇聚层、接入层。一个 k 元的 Fat-Tree 可以归纳为 5 个特征：

- 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；
- 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$k/2$台；
- 接入层每个交换机可以容纳$k/2$台服务器，所有 pod 共能容纳$k^3/4$台服务器；
- 任意两个 pod 之间存在k条路径。

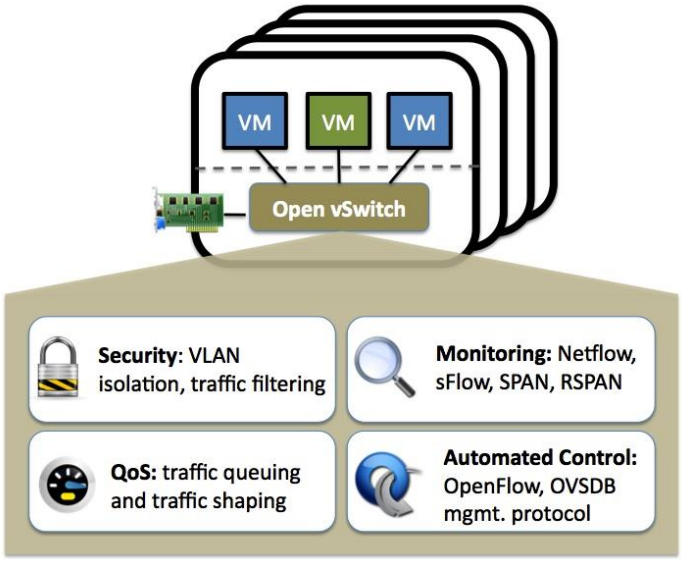


Mininet 是由斯坦福大学基于 Linux Container 架构开发的一个进程虚拟化网络仿真工具，可以创建一个包含主机，交换机，控制器和链路的虚拟网络，其交换机支持 OpenFlow，具备高度灵活的自定义软件定义网络。Mininet 提供了以下功能：

- 为 OpenFlow 应用程序提供一个简单，便宜的网络测试平台；
- 启用复杂的拓扑测试，无需连接物理网络；
- 具备拓扑感知和 OpenFlow 感知的 CLI，用于调试或运行网络范围的测试；
- 支持任意自定义拓扑，主机数可达 4096，并包括一组基本的参数化拓扑；
- 提供用户网络创建和实验的可拓展 Python API。



OpenvSwitch 简称 OVS，是一个高质量、多层的虚拟交换软件。它的目的是通过编程扩展支持大规模网络自动化，同时它还支持标准的管理接口和协议与多个物理机的分布式环境。虽然是虚拟交换机，但是其工作原理与物理交换机类似。在虚拟交换机的实现中，其两端分别连接着物理网卡和多块虚拟网卡，同时虚拟交换机内部会维护一张映射表，根据 MAC 地址寻找对应的虚拟机链路进而完成数据转发。OpenvSwitch 可以实现访问控制功能，通过转发规则，可以实现简单的安全行为，包括通过、禁止等。



二、实验任务

使用 Mininet 的 Python API 搭建 k=4 的 fat tree 拓扑；
使用 pingall 查看各主机之间的连通情况;
若主机之间未连通，分析原因并解决（使用 wireshark 抓包分析）
若主机连通，分析数据包的路径（提示：ovs-appctl fdb/show 查看 MAC 表）
完成实验报告并提交到思源学堂
要求不能使用控制器

三、实验内容与分析

3.1 拓扑结构创建

使用 Mininet Python API 搭建 k=4 的 fat tree 拓扑，见附件 fattree.py。如图 3.1

所示，网络拓扑成功创建。

```
sdn@ubuntu:~/Desktop/mininet$ sudo python fattree.py
*** Creating network
*** Adding hosts:
h0_0 h0_1 h0_2 h0_3 h1_0 h1_1 h1_2 h1_3 h2_0 h2_1 h2_2 h2_3 h3_0 h3_1 h3_2 h
3_3
*** Adding switches:
Arpod0no0 Arpod0no1 Arpod1no0 Arpod1no1 Arpod2no0 Arpod2no1 Arpod3no0 Arpod3
no1 Edpod0no0 Edpod0no1 Edpod1no0 Edpod1no1 Edpod2no0 Edpod2no1 Edpod3no0 Ed
pod3no1 core0 core1 core2 core3
*** Adding links:
(Arpod0no0, Edpod0no0) (Arpod0no0, Edpod0no1) (Arpod0no1, Edpod0no0) (Arpod0
no1, Edpod0no1) (Arpod1no0, Edpod1no0) (Arpod1no0, Edpod1no1) (Arpod1no1, Ed
pod1no0) (Arpod1no1, Edpod1no1) (Arpod2no0, Edpod2no0) (Arpod2no0, Edpod2no1
) (Arpod2no1, Edpod2no0) (Arpod2no1, Edpod2no1) (Arpod3no0, Edpod3no0) (Arpo
d3no0, Edpod3no1) (Arpod3no1, Edpod3no0) (Arpod3no1, Edpod3no1) (Edpod0no0,
h0_0) (Edpod0no0, h0_1) (Edpod0no1, h0_2) (Edpod0no1, h0_3) (Edpod1no0, h1_0
) (Edpod1no0, h1_1) (Edpod1no1, h1_2) (Edpod1no1, h1_3) (Edpod2no0, h2_0) (E
dpod2no0, h2_1) (Edpod2no1, h2_2) (Edpod2no1, h2_3) (Edpod3no0, h3_0) (Edpod
3no0, h3_1) (Edpod3no1, h3_2) (Edpod3no1, h3_3) (core0, Arpod0no0) (core0, A
rpod1no0) (core0, Arpod2no0) (core0, Arpod3no0) (core1, Arpod0no0) (core1, A
rpod1no0) (core1, Arpod2no0) (core1, Arpod3no0) (core2, Arpod0no1) (core2, A
rpod1no1) (core2, Arpod2no1) (core2, Arpod3no1) (core3, Arpod0no1) (core3, A
rpod1no1) (core3, Arpod2no1) (core3, Arpod3no1)
*** Configuring hosts
h0_0 h0_1 h0_2 h0_3 h1_0 h1_1 h1_2 h1_3 h2_0 h2_1 h2_2 h2_3 h3_0 h3_1 h3_2 h
3_3
*** Starting controller

*** Starting 20 switches
Arpod0no0 Arpod0no1 Arpod1no0 Arpod1no1 Arpod2no0 Arpod2no1 Arpod3no0 Arpod3
no1 Edpod0no0 Edpod0no1 Edpod1no0 Edpod1no1 Edpod2no0 Edpod2no1 Edpod3no0 Ed
pod3no1 core0 core1 core2 core3 ...
*** Starting CLI:
```

图 3.1

使用 pingall 命令，发现主机未连通，如图 3.2 所示。在 mininet CLI 中执行 xterm hopod0no0 打开的终端使用 wireshark 抓包。抓包截图如图 3.3 所示，发现主机一直发送 ARP 查询包询问目标主机 MAC 地址却始终得不到回复。这是因为 fat-tree 结构存在环路，引起广播包指数递增，整个网络流量被广播包占据，其他的转发业务不能进行。之所以会产生这样的广播风暴，是因为以太交换机不管从哪个端口收到广播包，都完整地复制一份转发到其他端口（除接收到的端口外）。

```
mininet> pingall
*** Ping: testing ping reachability
h0_0 -> X X X X X X X X X X X X X X X
h0_1 -> X X X X X X X
```

图 3.2

4	6.486616675	1e:07:4e:6d:30:df	Broadcast	ARP	42	Who has 10.0.0.2? Tell 10.0.0.1
5	7.487134241	1e:07:4e:6d:30:df	Broadcast	ARP	42	Who has 10.0.0.2? Tell 10.0.0.1
6	8.511345743	1e:07:4e:6d:30:df	Broadcast	ARP	42	Who has 10.0.0.2? Tell 10.0.0.1
7	9.537547014	1e:07:4e:6d:30:df	Broadcast	ARP	42	Who has 10.0.0.3? Tell 10.0.0.1
8	10.558918638	1e:07:4e:6d:30:df	Broadcast	ARP	42	Who has 10.0.0.3? Tell 10.0.0.1
9	11.582922743	1e:07:4e:6d:30:df	Broadcast	ARP	42	Who has 10.0.0.3? Tell 10.0.0.1
10	12.609346443	1e:07:4e:6d:30:df	Broadcast	ARP	42	Who has 10.0.0.4? Tell 10.0.0.1
11	13.631615108	1e:07:4e:6d:30:df	Broadcast	ARP	42	Who has 10.0.0.4? Tell 10.0.0.1
12	14.655762213	1e:07:4e:6d:30:df	Broadcast	ARP	42	Who has 10.0.0.4? Tell 10.0.0.1
13	15.681082988	1e:07:4e:6d:30:df	Broadcast	ARP	42	Who has 10.0.0.5? Tell 10.0.0.1
14	16.703002264	1e:07:4e:6d:30:df	Broadcast	ARP	42	Who has 10.0.0.5? Tell 10.0.0.1

图 3.3

3.2 配置 STP 解决环路问题

为所有交换机配置 STP（生成树协议）可以解决主机间无法连通的问题。STP 通过阻塞端口来消除环路，可以使用以下命令为路由器开启此协议：

```
sudo ovs-vsctl set bridge s1 stp_enable=true #开启 STP，s1 为设备名
```

需要对每个交换机执行以下命令，否则 mac 表将仍然学习不到东西：

```
sudo ovs-vsctl del-fail-mode xx
```

由于有 20 个交换机，逐个开启非常麻烦，因此在另一终端运行 stp_en.py 脚本，该脚本通过系统调用 os.system() 执行命令。交换机开启 STP 协议后再次使用 pingall 命令，发现可以 ping 通，不过在交换机启用 STP 协议后的一小段时间内主机间仍然未连通，这是因为交换机通过 STP 学习网络拓扑需要时间。

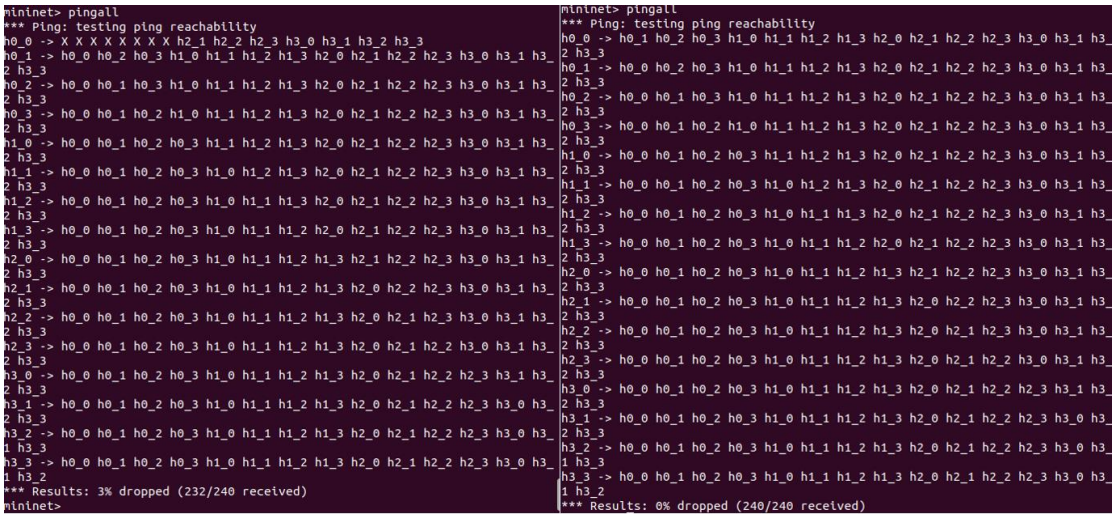
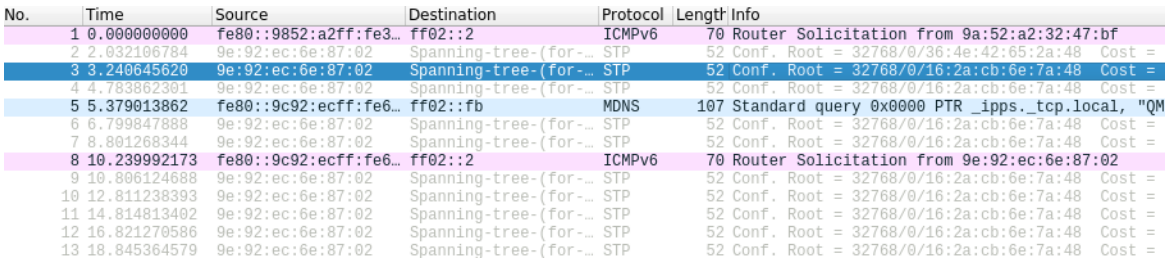


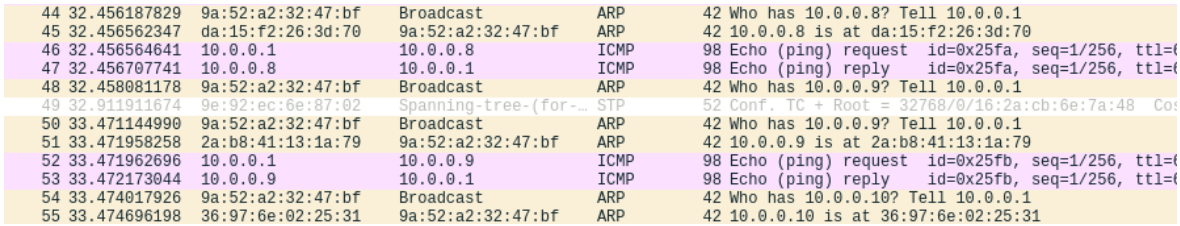
图 3.4-连续两次 pingall 结果

在此过程中抓包分析，见图 3.5、3.6。可以发现启用 STP 协议后交换机会发送 STP 包确定生成树。在网络连接稳定后主机向目标机发送报文时，一般会发送 1 个 ARP 查询包并收到 1 个 ARP 回复包，主机确定目标机 MAC 地址后即发送 ICMP 包。中间间或收到 STP 包，这是由于网络发生了问题，需要通过 STP 更新生成树状态。



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	fe80::9852:a2ff:fe3…	ff02::2	ICMPv6	70	Router Solicitation from 9a:52:a2:32:47:bf
2	2.032106784	9e:92:ec:6e:87:02	Spanning-tree-(for-…	STP	52	Conf. Root = 32768/0/36:4e:42:65:2a:48 Cost =
3	3.240645620	9e:92:ec:6e:87:02	Spanning-tree-(for-…	STP	52	Conf. Root = 32768/0/16:2a:cb:6e:7a:48 Cost =
4	4.783862301	9e:92:ec:6e:87:02	Spanning-tree-(for-…	STP	52	Conf. Root = 32768/0/16:2a:cb:6e:7a:48 Cost =
5	5.379013862	fe80::9c92:ecff:fe6…	ff02::fb	MDNS	107	Standard query 0x0000 PTR _ipps._tcp.local, "QM
6	6.799847888	9e:92:ec:6e:87:02	Spanning-tree-(for-…	STP	52	Conf. Root = 32768/0/16:2a:cb:6e:7a:48 Cost =
7	8.801268344	9e:92:ec:6e:87:02	Spanning-tree-(for-…	STP	52	Conf. Root = 32768/0/16:2a:cb:6e:7a:48 Cost =
8	10.239992173	fe80::9c92:ecff:fe6…	ff02::2	ICMPv6	70	Router Solicitation from 9e:92:ec:6e:87:02
9	10.806124688	9e:92:ec:6e:87:02	Spanning-tree-(for-…	STP	52	Conf. Root = 32768/0/16:2a:cb:6e:7a:48 Cost =
10	12.811238393	9e:92:ec:6e:87:02	Spanning-tree-(for-…	STP	52	Conf. Root = 32768/0/16:2a:cb:6e:7a:48 Cost =
11	14.814813402	9e:92:ec:6e:87:02	Spanning-tree-(for-…	STP	52	Conf. Root = 32768/0/16:2a:cb:6e:7a:48 Cost =
12	16.821270586	9e:92:ec:6e:87:02	Spanning-tree-(for-…	STP	52	Conf. Root = 32768/0/16:2a:cb:6e:7a:48 Cost =
13	18.845364579	9e:92:ec:6e:87:02	Spanning-tree-(for-…	STP	52	Conf. Root = 32768/0/16:2a:cb:6e:7a:48 Cost =

图 3.5



44	32.456187829	9a:52:a2:32:47:bf	Broadcast	ARP	42	Who has 10.0.0.8? Tell 10.0.0.1
45	32.456562347	da:15:f2:26:3d:70	9a:52:a2:32:47:bf	ARP	42	10.0.0.8 is at da:15:f2:26:3d:70
46	32.456564641	10.0.0.1	10.0.0.8	ICMP	98	Echo (ping) request id=0x25fa, seq=1/256, ttl=
47	32.456707741	10.0.0.8	10.0.0.1	ICMP	98	Echo (ping) reply id=0x25fa, seq=1/256, ttl=
48	32.458081178	9a:52:a2:32:47:bf	Broadcast	ARP	42	Who has 10.0.0.9? Tell 10.0.0.1
49	32.911911674	9e:92:ec:6e:87:02	Spanning-tree-(for-…	STP	52	Conf. TC + Root = 32768/0/16:2a:cb:6e:7a:48 Co
50	33.471144990	9a:52:a2:32:47:bf	Broadcast	ARP	42	Who has 10.0.0.9? Tell 10.0.0.1
51	33.471958258	2a:b8:41:13:1a:79	9a:52:a2:32:47:bf	ARP	42	10.0.0.9 is at 2a:b8:41:13:1a:79
52	33.471962696	10.0.0.1	10.0.0.9	ICMP	98	Echo (ping) request id=0x25fb, seq=1/256, ttl=
53	33.472173044	10.0.0.9	10.0.0.1	ICMP	98	Echo (ping) reply id=0x25fb, seq=1/256, ttl=
54	33.474017926	9a:52:a2:32:47:bf	Broadcast	ARP	42	Who has 10.0.0.10? Tell 10.0.0.1
55	33.474696198	36:97:6e:02:25:31	9a:52:a2:32:47:bf	ARP	42	10.0.0.10 is at 36:97:6e:02:25:31

图 3.6

3.3 数据包的路径分析

这里分析主机 hopod0no0 到 hopod3no2 的路径。通过 ifconfig 命令可以查询两机的 MAC 地址。其中 hopod0no0 为 9a:52:a2:32:47:bf; hopod3no2 为 f2:d3:cf:fc:43:ba。使用以下命令可以查询各交换机的路由表，并以此确定转发路径：

```
sudo ovs-appctl fdb/show switchname
```

图 3.7 和图 3.8 分别是交换机 Edpod0no0 和 Arpod0no1 的 MAC 表，其中有 hopod3no2 的表项（已标出），对应的 port 分别为 2 和 1。其它交换机的 MAC 表就不一一列出了。

```
sdn@ubuntu:~/Desktop/mininet$ sudo ovs-appctl fdb/show Edpod0no0
 port  VLAN  MAC                Age
    2     0  86:b0:47:e4:9c:4e  290
    2     0  f2:d3:cf:fc:43:ba  290
    2     0  ae:85:9e:e6:56:78  290
    2     0  da:15:f2:26:3d:70  290
    3     0  9a:52:a2:32:47:bf  273
    2     0  0a:32:6a:fa:1a:f7  273
    2     0  ee:2f:5f:c5:81:5d  273
    2     0  62:87:97:63:d8:07  273
```

图 3.7

```
sdn@ubuntu:~/Desktop/mininet$ sudo ovs-appctl fdb/show Arpod0no1
 port  VLAN  MAC                Age
    4     0  86:b0:47:e4:9c:4e   28
    3     0  56:ae:89:02:c1:31   28
    4     0  56:4e:5f:c7:7b:a3   28
    3     0  9a:52:a2:32:47:bf   28
    1     0  f2:d3:cf:fc:43:ba   28
    1     0  da:15:f2:26:3d:70   28
```

图 3.8

通过各交换机的 MAC 表可以找出 hopod0no0 到 hopod3no2 的路径为：

$$hopod0no0 - Edpod0no0 - Arpod0no1 - core2 - Arpod3no1 - Edpod3no1 - hopod3no2$$

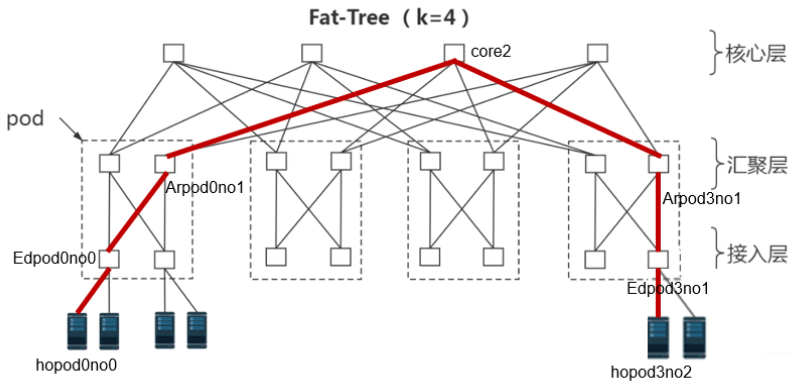


图 3.9-hopod0no0 到 hopod3no2 的路径