第40讲 | 搭建一个网络实验环境：授人以鱼不如授人以渔

2018-08-17 刘超

第40讲 | 搭建一个网络实验环境：授人以鱼不如授人以渔
朗读人：刘超　09′43″ | 4.46M

因为这门课是基础课程，而且配合音频的形式发布，所以我多以理论为主来进行讲解。在专栏更新的过程中，不断有同学让我推荐一些网络方面的书籍，还有同学说能不能配合一些实验来说明理论。

的确，网络是一门实验性很强的学科，就像我在开篇词里面说的一样：**一看觉得懂，一问就打鼓，一用就糊涂。** 在写专栏的过程中，我自己也深深体会到了。这个时候，我常常会拿一个现实的环境，上手操作一下，抓个包看看，这样心里就会有定论。

《TCP/IP 详解》实验环境搭建

对于网络方面的书籍，我当然首推 Rechard Stevens 的《TCP/IP illustrated》（《TCP/IP 详解》）。这本书把理论讲得深入浅出，还配有大量的上手实践和抓包，看到这些抓包，原来不理解的很多理论，一下子就能懂了。

这本书里有个拓扑图，书上的很多实验都是基于这个图的，但是这个拓扑图还是挺复杂的。我这里先不说，一会儿详细讲。

Rechard Stevens，因为工作中有这么一个环境，很方便做实验，最终才写出了这样一本书，而我们一般人学习网络，没有这个环境应该怎么办呢？

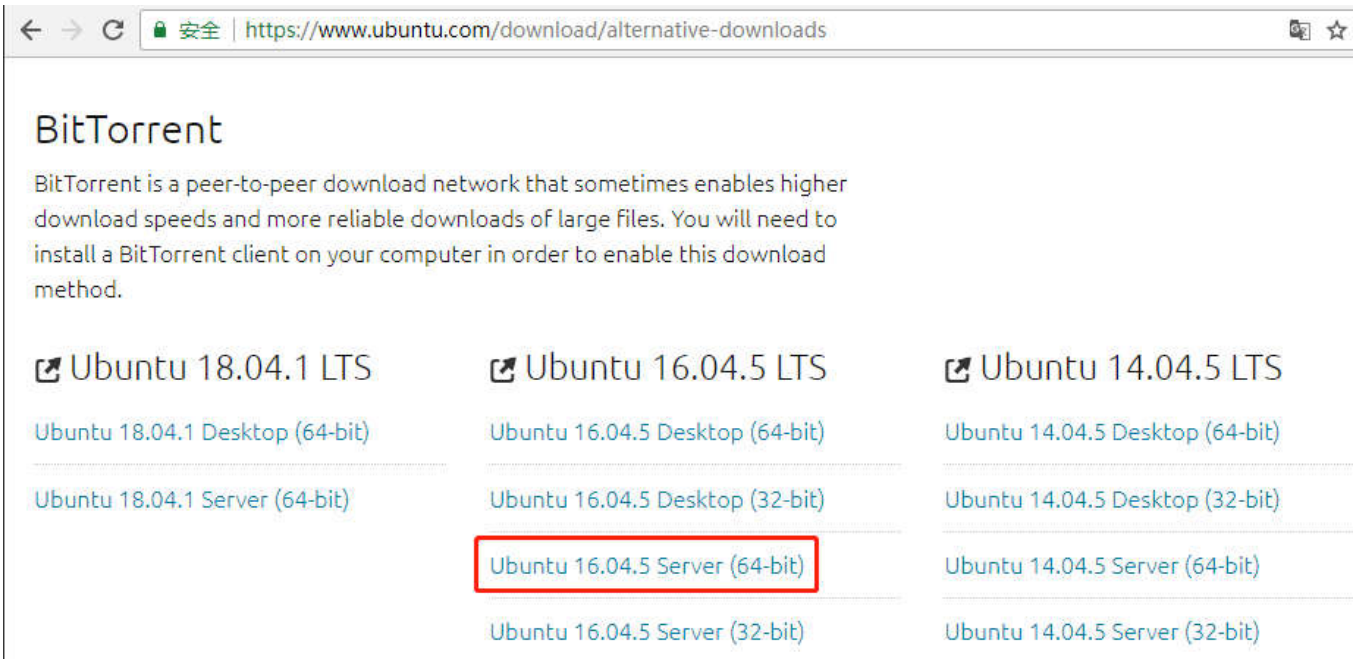
时代不同了，咱们现在有更加强大的工具了。例如，这里这么多的机器，我们可以用 Docker 来实现，多个网络可以用 Open vSwitch 来实现。你甚至不需要一台物理机，只要一台 1 核 2G 的虚拟机，就能将这个环境搭建起来。

搭建这个环境的时候，需要一些脚本。我把脚本都放在了Github里面，你可以自己取用。

1. 创建一个 Ubuntu 虚拟机

在你的笔记本电脑上，用 VirtualBox 创建就行。1 核 2G，随便一台电脑都能搭建起来。

首先，我们先下载一个 Ubuntu 的镜像。我是从Ubuntu 官方网站下载的。



然后，在 VirtualBox 里面安装 Ubuntu。安装过程网上一大堆教程，你可以自己去看，我这里就不详细说了。

这里我需要说明的是网络的配置。

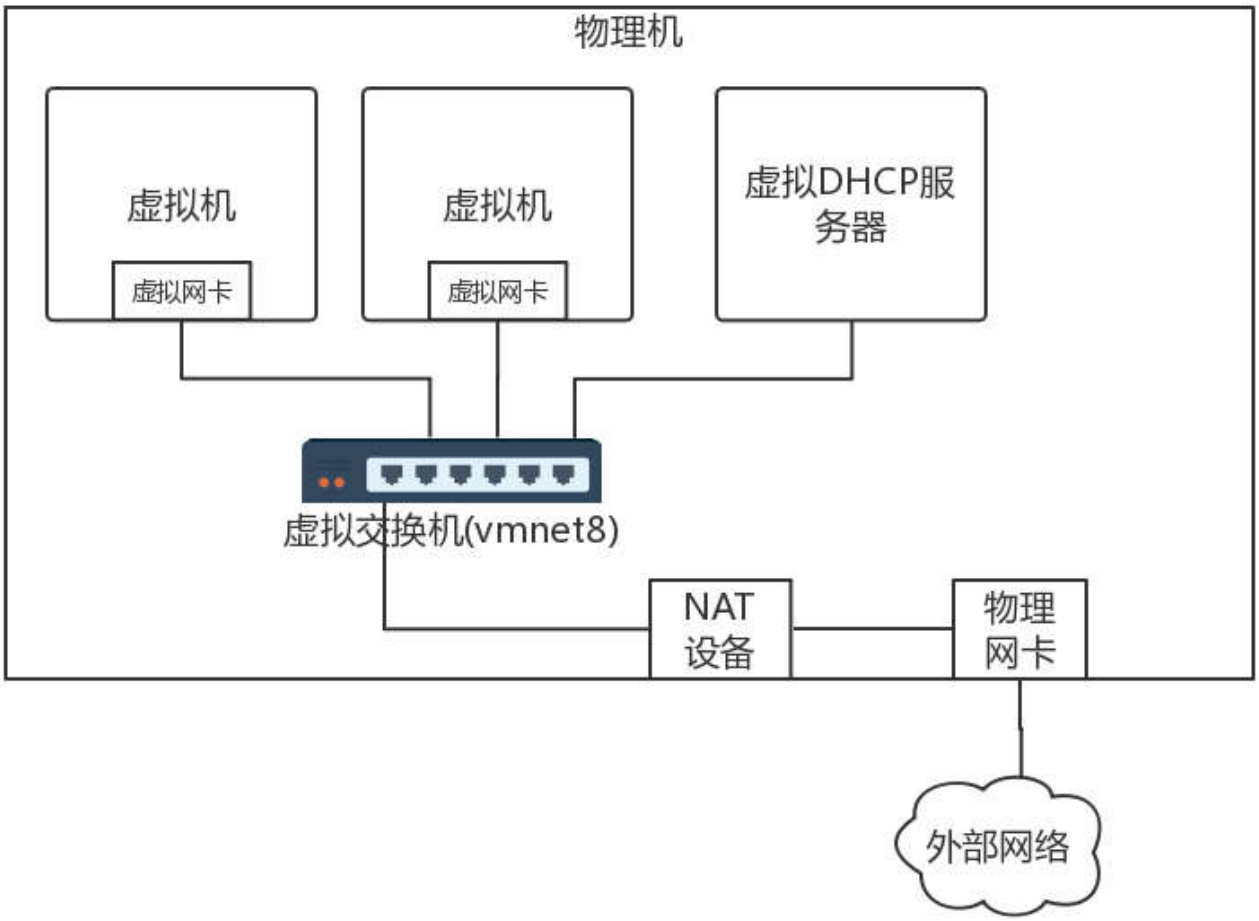
对于这个虚拟机，我们创建两个网卡，一个是 Host-only，只有你的笔记本电脑上能够登录进去。这个网卡上的 IP 地址也只有在你的笔记本电脑上管用。这个网卡的配置比较稳定，用于在 SSH 上做操作。这样你的笔记本电脑就可以搬来搬去，在公司里安装一半，回家接着安装另一半都没问题。

这里有一个虚拟的网桥，这个网络可以在管理 > 主机网络管理里面进行配置。

在这里可以虚拟网桥的的 IP 地址，同时启用一个 DHCP 服务器，为新创建的虚拟机配置 IP 地址。

另一个网卡配置为 NAT 网络，用于访问互联网。配置了 NAT 网络之后，只要你的笔记本电脑能上网，虚拟机就能上网。由于咱们在 Ubuntu 里面要安装一些东西，因而需要联网。

你可能会问了，这个配置复杂吗？一点儿都不复杂。咱们讲虚拟机网络的时候，讲过这个。



安装完了 Ubuntu 之后，需要对 Ubuntu 里面的网卡进行配置。对于 Ubuntu 来讲，网卡的配置在 /etc/network/interfaces 这个文件里面。在我的环境里，NAT 的网卡名称为 enp0s3，Host-only 的网卡的名称为 enp0s8，都可以配置为自动配置。

```
auto lo
iface lo inet loopback

auto enp0s3
iface enp0s3 inet dhcp
```

```
auto enp0s8

iface enp0s8 inet dhcp
```

这样，重启之后，IP 就配置好了。

2. 安装 Docker 和 Open vSwitch

接下来，在 Ubuntu 里面，以 root 用户，安装 Docker 和 Open vSwitch。

你可以按照 Docker 的官方安装文档来做。我这里也贴一下我的安装过程。

```
apt-get remove docker docker-engine docker.io

apt-get -y update

apt-get -y install apt-transport-https ca-certificates curl software-properties-common

curl -fsSL https://download.docker.com/linux/ubuntu/gpg > gpg

apt-key add gpg

apt-key fingerprint 0EBFCD88

add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -

apt-get -y update

apt-cache madison docker-ce

apt-get -y install docker-ce=18.06.0~ce~3-0~ubuntu
```

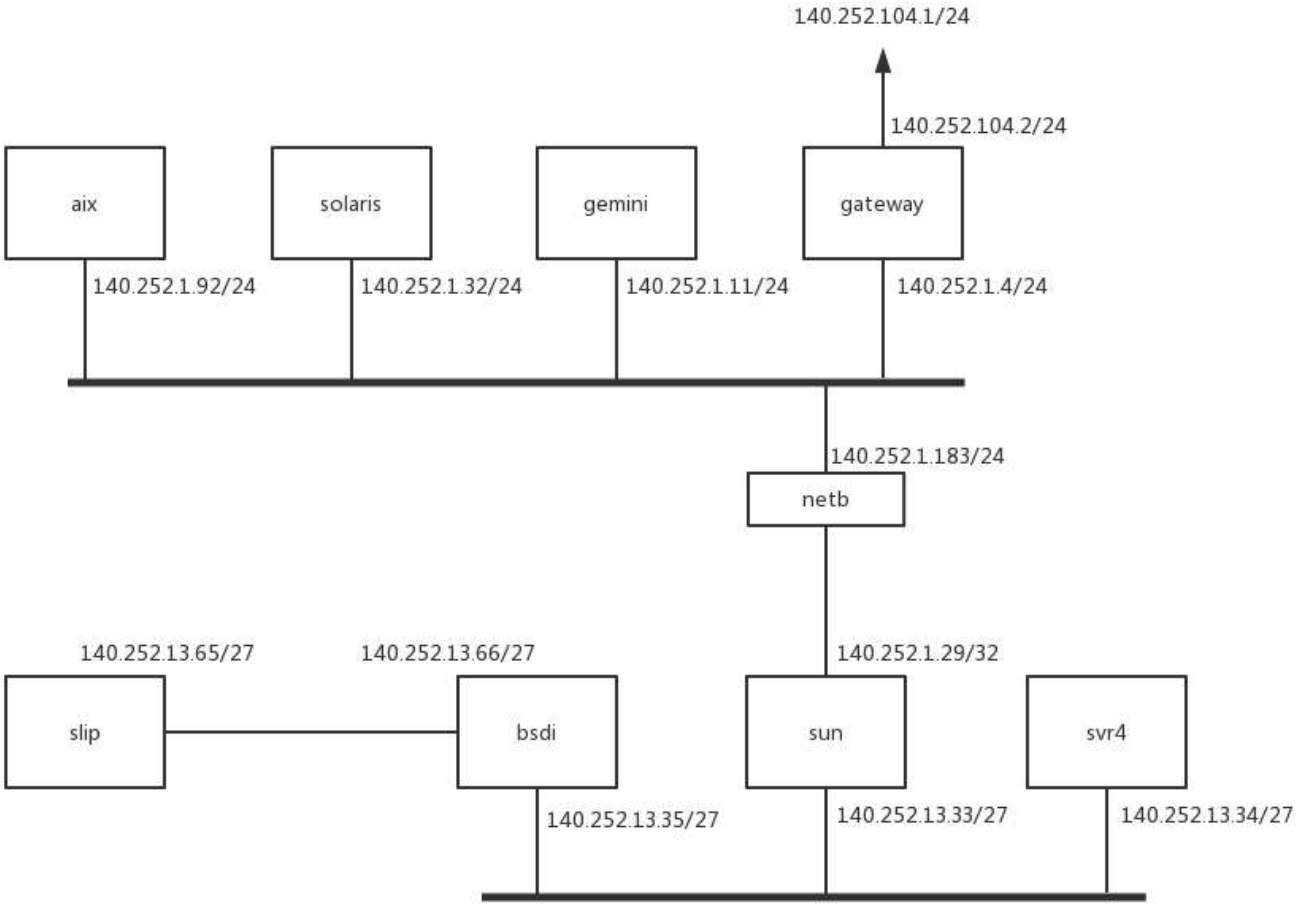
之后，还需要安装 Open vSwitch 和 Bridge。

```
apt-get -y install openvswitch-common openvswitch-dbg openvswitch-switch python-openvswitch o


apt-get -y install bridge-utils


apt-get -y install arping
```

3. 准备一个 Docker 的镜像



每个节点都是一个 Docker，对应要有一个 Docker 镜像。这个镜像我已经打好了，你可以直接使用。

```
docker pull hub.c.163.com/liuchao110119163/ubuntu:tcpip
```

当然你也可以自己打这个镜像。Dockerfile 就像这样：

```
FROM hub.c.163.com/public/ubuntu:14.04
RUN apt-get -y update && apt-get install -y iproute2 iputils-arping net-tools tcpdump curl te
RUN mv /usr/sbin/tcpdump /usr/bin/tcpdump
ENTRYPOINT /usr/sbin/sshd -D
```

4. 启动整个环境

启动这个环境还是比较复杂的，我写成了一个脚本。在 Git 仓库里面，有一个文件 setupenv.sh，可以执行这个脚本，里面有两个参数，一个参数是 NAT 网卡的名字，一个是镜像的名称。

```
git clone https://github.com/popsuper1982/tcpipillustrated.git
cd tcpipillustrated
```

```
docker pull hub.c.163.com/liuchao110119163/ubuntu:tcpip

chmod +x setupenv.sh

./setupenv.sh enp0s3 hub.c.163.com/liuchao110119163/ubuntu:tcpip
```

这样，整个环境就搭建起来了，所有的容器之间都可以 ping 通，而且都可以上网。

不过，我写的这个脚本对一些人来说可能会有点儿复杂，我这里也解释一下。

首先每一个节点，都启动一个容器。使用–privileged=true 方式，网络先不配置–net none。有两个二层网络，使用 ovs-vsctl 的 add-br 命令，创建两个网桥。

pipework 是一个很好的命令行工具，可以将容器连接到两个二层网络上。

但是我们上面那个图里有两个比较特殊的网络，一个是从 slip 到 bsdi 的 P2P 网络，需要创建一个 peer 的两个网卡，然后两个 Docker 的网络 namespace 里面各塞进去一个。

有关操作 Docker 的网络 namespace 的方式，咱们在容器网络那一节讲过 ip netns 命令。

这里需要注意的是，P2P 网络和下面的二层网络不是同一个网络。P2P 网络的 CIDR 是 140.252.13.64/27，而下面的二层网络的 CIDR 是 140.252.13.32/27。如果按照 /24，看起来是一个网络，但是 /27 就不是了。至于CIDR 的计算方法，你可以回去复习一下。

另外需要配置从 sun 到 netb 的点对点网络，方法还是通过 peer 网卡和 ip netns 的方式。

这里有个特殊的地方，对于 netb 来讲，不是一个普通的路由器，因为 netb 两边是同一个二层网络，所以需要配置 arp proxy。

为了所有的节点之间互通，要配置一下路由策略，这里需要通过 ip route 命令。

- 对于 slip 来讲，bsdi 左面 13.66 这个网口是网关。
- 对于 bsdi 和 svr4 来讲，如果去外网，sun 下面的网口 13.33 是网关。
- 对于 sun 来讲，上面的网口 1.29 属于上面的二层网络了，它如果去外网，gateway 下面的网口 1.4 就是外网网关。
- 对于 aix，solaris，gemini 来讲，如果去外网，网关也是 gateway 下面的网口 1.4。如果去下面的二层网口，网关是 sun 上面的网口 1.29。

配置完了这些，图中的所有的节点都能相互访问了，最后还要解决如何访问外网的问题。

我们还是需要创建一个 peer 网卡对。一个放在 gateway 里面，一个放在 gateway 外面。外面的网卡去外网的网关。

在虚拟机上面，还需要配置一个 iptables 的地址伪装规则 MASQUERADE，其实就是一个 SNAT。因为容器里面要访问外网，因为外网是不认的，所以源地址不能用容器的地址，需要 SNAT 成为虚拟机的地址出去，回来的时候再 NAT 回来。

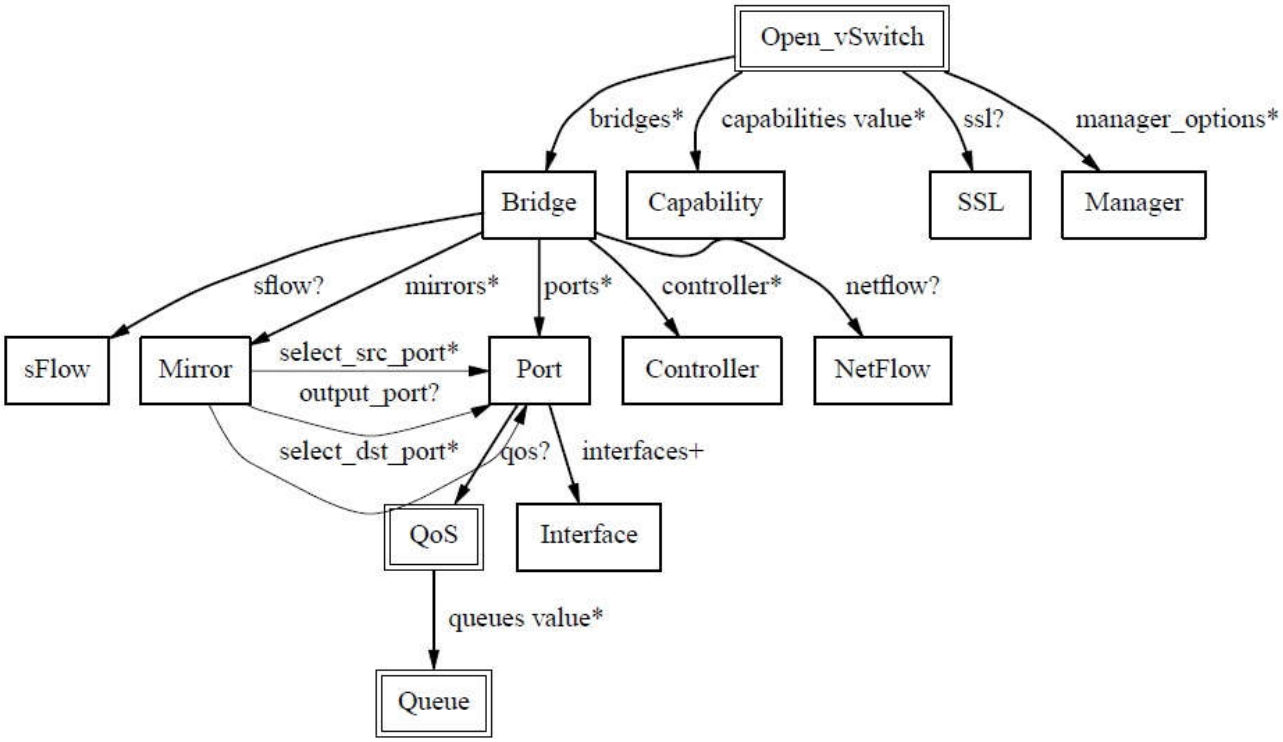
配置这个环境还是挺复杂的，要用到咱们学到的很多知识。如果没有学习前面那些知识，直接就做这个实验，你肯定会很晕。但是只学理论也不行，要把理论都学过一遍，再做一遍实验，这才是一个不断迭代、更新知识库的过程。

有了这个环境，《TCP/IP 详解》里面的所有实验都能做了，而且我打的这个 Docker 镜像里面，tcpdump 等网络工具都安装了，你可以“为所欲为”了。

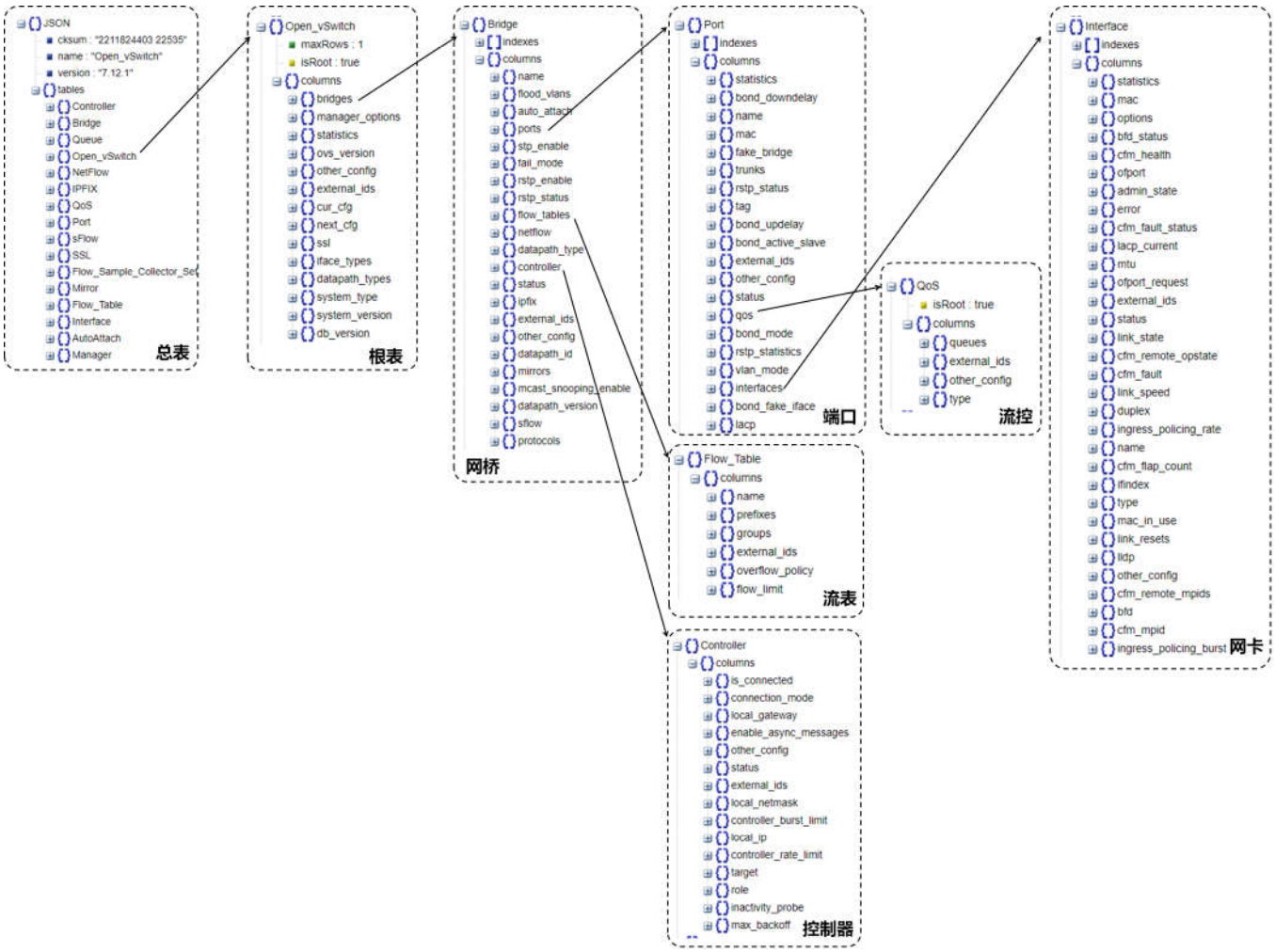
Open vSwitch 的实验

做了 TCP/IP 详解的实验之后，网络程序设计这部分，你就有了坚实的基础。但是涉及到数据中心内部的一些网络技术，什么 VLAN、VXLAN、STP 等偏运维方向的，学习还是会比较困难。好在我们有 Open vSwitch，也可以做大量的实验。

Open vSwitch 门槛比较高，里面的概念也非常多，可谓千头万绪。不过，通过我这么多年研究的经验，可以告诉你，这里面有一个很好的线索，那就是 Open vSwitch 会将自己对于网络的配置保存在一个本地库里面。这个库的表结构之间的关系就像这样：



这个库其实是一个 JSON，如果把这个 JSON 打印出来，能够看到更加详细的特性。按照这些特性——实验，可以逐渐把 Open vSwitch 各个特性都掌握。



这里面最重要的概念就是网桥。一个网桥会有流表控制网络包的处理过程，会有控制器下发流表，一个网桥上会有多个端口，可以对端口进行流控，一个端口可以设置 VLAN，一个端口可以包含多个网卡，可以做绑定，网卡可以设置成为 GRE 和 VXLAN。

我写过一个 Open vSwitch 的实验教程，也放在了 Github 里面。这里面有这么几个比较重要的实验，你可以看一看。

- 实验一：查看 Open vSwitch 的架构。我们在讲 Open vSwitch 的时候，提过 Open vSwitch 的架构，在这个实验中，我们可以查看 Open vSwitch 的各个模块以及启动的参数。
- 实验五：配置使用 OpenFlow Controller，体验一把作为小区物业在监控室里面管控整个小区道路的样子。
- 实验八：测试 Port 的 VLAN 功能。看一下 VLAN 隔离究竟是什么样的。
- 实验十：QoS 功能。体验一把如果使用 HTB 进行网卡限流。
- 实验十一：GRE 和 VXLAN 隧道功能，看虚拟网络如何进行租户隔离。
- 实验十五：对 Flow Table 的操作，体验流表对网络包随心所欲的处理。

好了，关于整个环境的搭建我就讲到这里了。

其实到这里，对于网络世界的探索才刚刚开始，只有经过你自己动手和思考产生的内容，才是真正属于你的知识！打开你的电脑，上手去实验吧！

欢迎你留言和我讨论。趣谈网络协议，我们下期见！

精选留言

qpzm7903 4

不说了，回去装机开始为所欲为！

2018-08-17

upstream 2

用gns3 模拟器可以吗?

2018-08-17

logic 2

很棒，回去实践一下

2018-08-17

youth 1

很棒，就喜欢这种操作性比较强的文章。

2018-08-17

favorlm 0

还是挺复杂的，以为自己懂了。

2018-08-19

咸鱼

显示pipework 68有问题

2018-08-19

0

咸鱼

为什么一直显示pipeeork 68有问题呢

2018-08-19

0

Eric

马上实践

2018-08-17

0

志翔

老师讲的太好了 既系统又有实操，棒棒哒！

2018-08-17

0