

# 漫步云中网络，第 2 部分

本文作为《[漫步云中网络](#)》的姊妹篇，在它讲述 L2-L3 层网络技术原理的基础之上，继续向大家讲述最新的 Neutron 背后所依赖的 L2-L7 层网络技术内幕，特别是这些复杂的网络技术的来龙去脉，尽量深入浅出地用浅显易懂的语言从纯技术角度切中这些技术的本质。帮您在进入研究 Neutron 细节之前就清楚知道 Neutron 是什么、有什么、为什么有，这样您在研究这些复杂网络技术时始终清楚定位不致于迷失方向。本文适合希望迅速了解 Neutron 全部特性的架构师，适合希望研究 Neutron 代码的程序员以及希望运行 Neutron 的测试人员，也适合想了解基础网络内幕知识的读者。

张华，IBM 高级软件工程师热衷于技术钻研，拥有丰富的搜索引擎、应用服务器、互联网、云计算领域的行业经验，精通 Java、JavaEE、Linux、Network 等技术，目前正从事 OpenStack 相关的工作。

2013 年 11 月 14 日

本文不会讲解每一种网络技术的细节，也不会讲解 Neutron 网络的实现细节，而是高度概括这些基础网络技术的技术本质，试图帮您在这些网络技术和 Neutron 之间建立更高级别的联系，让大家举重若轻，全局系统把握 Neutron。所以阅读本文前，了解以下知识将有助于本文的理解：

- 了解 OSI 七层模型，了解基本的 L2 层帧转发、L3 层路由转发等网络基础知识。
- 了解 Neutron 网络或者其他任何云网络也将有助于本文的理解。

## Neutron 是什么？

一句话描述，Neutron 是 OpenStack 的一个子模块，它的实质是一个定义良好的框架用来驱动 L2-L7 层不同的底层网络技术来为第三方应用独立地提供租户隔离的虚拟网络服务。

上面的定义只是笔者对 Neutron 长期以来一个最直观的感受，仁者见仁，智者见智，相信您在读完本文后，对于“Neutron 是什么”这个问题会有自己的看法。

笔者之前在 developerworks 上曾发表过一篇文章，《[漫步云中网络](#)》，在那篇文章中，笔者也没有直接具体讲 Quantum HOWTO 的问题(目前 Quantum 因为与一家公司重名，所以已更名为 Neturon)，而是描述了 Quantum 网络背后的一般原理，读者至少可以从那篇文章获取如下知识：

- 知道 Linux 下实现虚拟网卡一般使用 TAP/TUN 技术。一个 TAP 设备就是 Linux 下的一个进程，两个虚拟机通过虚拟网卡的通信，实际上就是 Linux 中两个进程间的通



在 IBM Bluemix 云上开发并部署您的下一个应用。

开始您的试用

信。所以很多 Hypervisor 为了提升同一物理机中的两台虚机之间的网络 IO 性能采用 DMA（直接内存访问）技术也就毫不奇怪了。

- 知道在 L2 层，Linux 网桥是虚拟交换机的一种实现，知道无论是虚拟交换机还是物理交换机，原理都是一样的。知道 L2 层用于使用 VLAN 来做物理隔离。知道 FLAT 网络和 VLAN 网络的根本区别。
- 知道在 L3 层如何通过 ipv4 forward 功能进行静态路由转发，知道如何使用 iptables 的 SNAT 和 DNAT 规则实现内网中的虚机访问外网和外网访问内网上的虚机（也就是所谓的浮动 IP）。

在我写第一季的时候，Quantum 只实现了 L2，L3 两层，所以在《漫步云中网络》一文也就只涉及了 L2、L3 两层背后的网络原理知识。但是现在 Neutron 在 L2 和 L3 层上实现了更多的网络技术，同时在 L4-L7 层也有更多的动作，所以有必要出第二季对整个 L2-L7 层的网络进行一个全面的梳理。本季中也会概括 L2、L3 的理论知识，但不会像第一季中那么详细，大家也可以结合第一季进行学习。所以，本文的主要内容有：

- L2 层：交换机的原理；为什么会出现 VLAN；Neutron 中 FLAT 与 VLAN 的区别；
- L3 层：Linux 上实现静态路由的技术（namespace + ipv4 forward + iptables）；动态路由；Neutron 用 L3 层的 GRE 隧道技术克服 VLAN 大小的限制；
- 利用 L3 层扩展 L2 层的隧道技术：VXLAN；NVGRE；
- 利用 L2 层扩展 L3 层的标签技术：MPLS；
- 区别于传统路由转发的流转发技术：OpenFlow 以及 SDN 的实质；
- L4—L7 层：如 LBaaS；FWaaS；VPNaaS；NATaaS

---

## OSI 七层模型

提到网络不得不提到 OSI 七层模型，从上到下，OSI 分为七层：

- L7，应用层
- L6，表示层
- L5，会话层
- L4，运输层
- L3，网络层
- L2，数据链路层
- L1，物理层

对于 OSI 七层模型至少得知道下列常识：

- L2 层主要通过 MAC 地址进行帧转发

- L3 层主要通过 IP 地址进行包转发
- L4 层再结合端口 PORT 来唯一标志一个应用程序
- 协议是通信双方对数据的一个理解，例如在 L7 层有我们常用的协议 HTTP 协议，在 HTTP 协议上传输的是通信双方都理解的 HTML 数据；在 L4 层有两大重要协议，无连接的 UDP 和面向连接的 TCP。可靠传输可以通过 TCP 协议来实现，对于下面的 L2, L3 层并不需要实现可靠传输机制，像 L2 层，传输数据帧的过程中出了错误简单丢弃就行，上层的 TCP 自然会控制它重传。socket 不是协议，只是 L4 层传输数据的一个接口定义。
- 当网卡接收到数据之后，硬件网卡会给 CPU 发中断，CPU 在指令周期内指示操作系统软件从网卡缓冲区取走数据，然后操作系统将数据交给 TCP/IP 栈来处理，到了 L2 层，解析 L2 层数据帧头中的 MAC 地址来决定 L2 中的转发，如需 3 层转发就交给上面的 L3 层解析出数据包头中的 IP 地址来决定 L3 中的转发，依此类推。

---

## L1

L1 是物理层，主要是涉及硬件的一些电气特性，与偏软件的 Neutron 虚拟网络从知识脉络上关系甚少，不展开。

---

## L2

### FLAT

L2 数据链路层通过交换机设备进行帧转发。交换机在接收到帧之后(L2 层叫帧，L3 层叫包)先解析出帧头中的 MAC 地址，再在转发表中查找是否有对应 MAC 地址的端口，有的话就从相应端口转发出去。没有，就洪泛（专业术语，即将帧转发到交换机的所有端口），每个端口上的计算机都检查帧头中的 MAC 地址是否与本网卡的 MAC 地址一致，一致的话就接收数据帧，不一致就直接丢弃。而转发表是通过自学习自动建立的。

这里引出一个重要概念，混杂模式。默认情况下计算机只接收和本机 MAC 地址一致的数据帧，不一致就丢弃，如果要求计算机接受所有帧的话，就要设置网卡为混杂模式（`ifconfig eth0 0.0.0.0 promisc up`）。所以在虚拟网桥中，如果希望虚机和外部通讯，必须打开桥接到虚拟网桥中物理网卡的混杂模式特性。

### VLAN

FLAT 中的洪泛，经常会在一个局域网内产生大量的广播，这也就是所谓的“广播风暴”。为了隔离广播风暴，引入了 VLAN 的概念。即为交换机的每一个端口设置一个 1-4094 的数字，交换机根据 MAC 地址进行转发的同时也要结合 VLAN 号这个数字，不同的话也要丢弃。这样就实现了 L2 层数据帧的物理隔离，避免了广播风暴。

在 Neutron 中，我们知道，截止到笔者写这篇文章之时已经实现了 FLAT、VLAN、GRE、VXLAN 四种网络拓扑。那么如何区分 FLAT 和 VLAN 呢？很简单，结合

VLAN 号和 MAC 地址进行转发的是 VLAN 模式，只根据 MAC 地址进行转发的是 FLAT 模式。

## VLAN 的缺点与大 L2 层技术

实际上，隧道技术并不能完全归类于 L2 层。因为有基于 L2 层的隧道协议，例如 PPTP 和 L2TP 等；也有基于 L3 层的隧道，如 GRE、VXLAN、NVGRE 等；但是这些隧道从技术原理上讲差不多，所以笔者将这些技术作为“大 L2 层”放在一块来描述，但希望读者不要误解。

本文只将着重讲 Neutron 中用到的 GRE 和 VXLAN 技术。

### L3 层的 GRE 隧道

VLAN 技术能有效隔离广播域，但同时有很多缺点：

- 要求穿越的所有物理交换机都配置允许带有某个 VLAN 号的数据帧通过，因为物理交换机通常只提供 CLI 命令，不提供远程接口可编程调用，所以都需要手工配置它，容易出错且工作量巨大，纯粹的体力劳动，影响了大规模部署。
- VLAN 号只能是 1—4094 中的一个数字，对于小规模私有云可能不是个问题，但对于租户众多的公有云，可选择的 VLAN 号的范围是不是少了点呢？

为了克服上面的缺点，Neutron 开发了对 GRE 模式的支持。GRE 是 L3 层的隧道技术，本质是在隧道的两端的 L4 层建立 UDP 连接传输重新包装的 L3 层包头，在目的地再取出包装后的包头进行解析。因为直接在隧道两端建立 UDP 连接，所以不需要在隧道两端路径的物理交换机上配置 TRUNK 的操作。

说说 GRE 的缺点吧：

- GRE 将 L2 层的操作上移到 L3 层来做，性能肯定是会下降的。同时，由于隧道只能是点对点的，所以可以想象，如果有 N 个节点，就会有  $N*(N-1)$  条 GRE 隧道，对于宝贵的 L4 层的端口资源也是一个浪费哦。那也是为什么 GRE 隧道使用 UDP 而不使用 TCP 的原因了，因为 UDP 用完后就可以释放，不用老占着端口又不用。
- 在 Neutron 的 GRE 实现中，即使两台物理机上的两台虚机也不是直接建立 GRE 隧道的。而是通过 L3-agent 网络节点进行中转，这样做是基于方便使用 iptables 隔离网络流量的考虑。

利用 L3 层扩展 L2 层的隧道技术 VXLAN 与 SDN 的本质

VXLAN 是 VMware 的技术，可以克服 VLAN 号不足的问题；同时也可以克服 GRE 实质上作为点对点隧道扩展性太差的问题。

如果说 GRE 的本质是将 L3 层的数据包头重新定义后再通过 L4 层的 UDP 进行传输，那么 VXLAN 的本质就是对 L2 层的数据帧头重新定义再通过 L4 层的 UDP 进行传输。也就是笔者说的所谓的利用 L3 层扩展 L2 层。



既然 L2 层的数据帧头要重新定义，那就随便定义啦，是否满足以下三点是笔者从技术角度将一件产是否视为 SDN 的关键因素：

- 可将 L7 层租户 **tenant** 的概念做到 L2 层。在笔者的前一篇《漫步云中网络》姊妹篇中已经介绍了 **IaaS** 云的本质就是卖虚机，即将虚机租给多租户使用后收费。所以位于 L7 应用层的 **tenant** 的概念是云用来对计算、存储、网络资源进行隔离的重要概念。那么将 **tenant** 的概念从 L7 层下移到 L2 层中意义重大。
- 也可将类似 **VLAN** 的概念做到 L2 层，并且由于是软件定义的（不像物理交换机那样受硬件限制），那么很容易突破 **VLAN** 号在 1—4094 之间的限制。
- 是否提供集中式的控制器对外提供北向 **API** 供第三方调用来动态改变网络拓扑。

清楚了 SDN 的本质，那么 **VXLAN** 的本质又是什么呢？

- **VXLAN** 是一种 **L2** 层帧头的重新封装的数据格式。
- **VXLAN** 仍然使用 **GRE** 隧道通过 **UDP** 传输重新封装帧头后的数据帧。

**VXLAN** 重新定义的 L2 层帧头格式如下图 1 所示，其中 **VNI** **VXLAN ID** 与 **VLAN** 的概念上等同，但是因为它用软件实现的，范围可用 **24 bits** 来表示，这就比 **VLAN** 的 1—4094 大多了。

图 1. **VXLAN** 帧头格式

Outer MAC DA	Outer MAC SA	Outer 802.1Q Tag	Outer IP DA	Outer IP SA	Outer UDP	VNI/VXLAN ID (24 Bits)	Inner MAC DA	Inner MAC SA	Optional Inner 802.1Q Tag	Original Ethernet L2 Frame
-----------------	-----------------	------------------------	----------------	----------------	--------------	---------------------------	-----------------	-----------------	---------------------------------	----------------------------------

因为 **VXLAN** 通过重新定义 **L2** 层帧头（相当于通过 **MAC** 地址进行 **NAT** 的方案）而让两个跨 **L3** 层的甚至广域网内的两个子网从 **L2** 层上互通。所以 **VXLAN** 的优点很多，能让遗留子网不改变 **IP** 地址的情况下无缝的迁移到云上来；也可以让虚机跨数据中心进行迁移（以前顶多只能在同一个 **VLAN** 里迁移）。

当然，**VXLAN** 也不是没有缺点的，通过上面的学习，大家已经知道 **VXLAN** 也是通过 **GRE** 走 **UDP** 来传输重定义的标准化的 **VXLAN** 封装格式的帧头的。由于在隧道的两端之间直接建立隧道，那么它是无法与途经的一些物理设备（如 **DHCP** 服务器）通信的，那么就需要 **VXLAN** 网关这种物理设备在隧道的中途截获 **VXLAN** 数据包（网关网关嘛，就是进行数据截获再转换的），解析里面的数据，然后做一些事情（像流量统计，**DHCP** 信息等等），最后再将数据重新打成 **VXLAN** 数据包交给隧道继续传输。可以想象，在所有需要与物理设备打交道的地方都是需要 **VXLAN** 网关的。觉得麻烦了吗？

利用 **L2** 层扩展 **L3** 层的标签技术 **MPLS**

**VLAN** 是一种标签技术，**VLAN** 一般用在局域网的交换机上，标签很容易在 **L2** 层通过硬件来实现转发。

MPLS 也是一种标签技术，原理类似于 VLAN，但一般用在 WAN 上的路由器上，下面我们说道说道。

对于 L3 层的传统路由转发来说一般是在路由表里查找相应的路由来实现。因为路由嘛，不同的 CIDR 之间可长可短，如 10.0.0.0/8 与 10.0.100.0/24 这两个子网首先长度不一样，其次 CIDR 号一个是 8，一个是 24，在路由器中，是通过字符串的按最长匹配算法来匹配路由的，那么应该选择 10.0.100.0/24 这个路由。字符串的按最长匹配算法很难通过硬件来实现，通过软件速度慢啊。那么广域网的路由器能不能也引入标签通过硬件转发的优点，在路由器作转发时，不再在 L3 层的通过软件去查找路由来实现转发，而是直接在 L2 层就通过标签通过硬件转发了呢？答案就是 MPLS。

例如 A 路由器通过 B 路由器给 C 路由器发包时，传统的路由器是根据目的地址进行转发，A 在开始转发时，并不知道去 C 的完整路由，它只知道转给 B，B 再决定转给 C，这种走一步看一步的叫基于目的地址的路由转发。但是 MPLS 的原理完全不同，A 在给 C 发包时，先发个 HELLO 包从 A 到 C 走一遍，在走的过程中就已经知道了 A 到 C 的路由路径并且根据 LDP（标签分发协议）将 A 到 C 的标签序列就事先确定好了。那样 A 再给 C 发数据时，A 就提前知道了在 L2 层怎么根据标签来转发，所以它不用再到 L3 层查找路由信息了，另外它在 L2 层通过标签转发可以很容易通过硬件来实现，这样速度更快了，最后标签的确定是通过 LDP 协议动态分配的这也避免了像 VLAN 的那种标签需要手工去设置的麻烦。

---

## SDN

在 VXLAN 一节中，笔者已经说过了笔者判断一个产品是否是 SDN 产品的核心判断因素，即是否将 tenant 租户的概念做到了 L2 层并且提供了北向 API 供第三方应用调用动态调整网络拓扑。做到了，就是 SDN 产品。目前，SDN 产品主要有三类：

- 基于 Overlay 技术的，如 VMware 的 VxLAN，如 Microsoft 的 NVGRE，如 IBM 的 Dove
- 基于 OpenFlow 技术的
- 基于专有技术的，如 Cisco 的 onePK

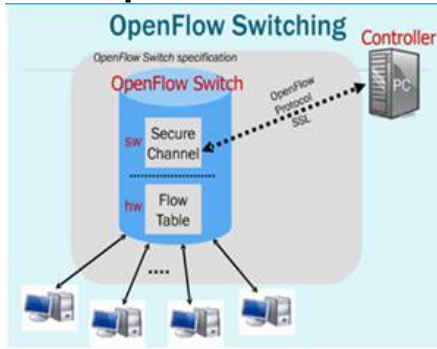
基于 Overlay 技术的 VxLAN 已经在上面介绍过了，这里着重介绍一下 OpenFlow 和 Dove。

## OpenFlow

OpenFlow 是完全不同于传统网络技术的新兴网络技术。

传统交换机能通过自主学习建立转发表，然后根据转发表转发。但对于 OpenFlow 交换机，它是很“笨”的，数据帧来了它并不知道往哪个端口转发，不知道怎么办呢？不懂就问呗，找 OpenFlow 控制器问。控制器通过一定算法告诉它往哪个端口转发，然后 OpenFlow 交换机照着做就行了。下图 2 显示了 OpenFlow 的结构：

图 2. OpenFlow 的结构



我并没有将 OpenFlow 归于 L2 层，因为从理论上讲，它并不严格属于 L2 层，因为它设计了 L2—L4 层的转发项进行转发。现在 OpenFlow 的 FIB（转发信息表，Forward Info Base）中至少有下列条目：

- L2 层的 MAC、VLAN 等
- L3 层的 source ip, dest ip
- L4 层的 source port, dest port
- 甚至有基于 WAN 的动态路由的转发项，如 BGP、MPLS 等

OpenFlow 进行转发的时候和传统的网络技术不一样，传统的是存储包转发（数据到了 L2 层就解析出 MAC 地址进行转发，到了 L3 层后就解析出 IP 地址进行转发）。而 OpenFlow 则根据所谓的流进行转发。它将上面的 MAC、VLAN, source ip, dest ip, source port, dest port 等视作平等的平面，直接通过某个高效的字符串匹配算法匹配到了后再做某些动作，如 accept 或者 drop 掉。

再聊聊笔者所理解的 OpenFlow 的缺点。理论上，如果 OpenFlow 通过 L3 层的 IP 进行转发的话，那它就成了一个路由器了。但实际上，目前，OpenFlow 主要还是用在 L2 层当做一个交换机来使用的。如果真要在 L3 层和路由结合的话，那么它将以什么样的方式和现存的分布式的动态路由协议（如 RIP、OSPF、BGP、MPLS）协作呢？并且传统的 WAN 本来为了可靠就是以分布式来设计的，现在搞成 OpenFlow 式的集中式控制，监控什么的是方便了，但一旦控制器挂了那么整个网络也全挂了，并且 OpenFlow 控制器一旦被某方势力所掌握，那么整个 OpenFlow 网络也就全被掌握了，毕竟 WAN 是要考虑这些因素的。所以笔者不认为 OpenFlow 这种集中式控制的路由器能在 WAN 上将取代传统分布式的路由协议，顶多在 LAN 的 L2 层会有一些作为吧。

## Dove/OpenDaylight

上面已经说到 SDN 需要对外提供北向 API 供第三方调用。

对于控制器与交换机之间的南向 API 的通讯标准就是由 google, facebook 这些用户主导定制的 OpenFlow 协议。

北向 API 没有标准，南向 API 也五花八门，将 tenant 和 VLAN 的概念做到 SDN 里也没有标准（虽然有 VMware 主导了一个 VXLAN 的数据封装格式），所以业界存在

很多 SDN 的产品，Dove 便是由 IBM 实现的其中一种，OpenDaylight 的插件结构可以同时和众多的 SDN 控制器打交道，降低第三方应用同时和众多 SDN 控制器打交道的复杂性。

---

## L3

L3 层的路由分静态路由和动态路由两种：

- 在 Linux 中，通过打开 `ipv4 forward` 特性可以让数据包从一块网卡路由到另外一块网卡上。
- 动态路由，如内部网关协议 RIP，OSPF；如外部网关协议 BGP。能够自动地学习建立路由表。

目前 Neutron 只支持静态路由，要点如下：

- 对于不同 tenant 子网通过 namespace 功能进行隔离，在 Linux 中，一个命名空间 namespace 您可以简单理解成 linux 又启动了一个新的 TCP/IP 栈的进程。多个 tenant 意味着多个 namespace，也意味着多个 TCP/IP 栈。
  - 对于同一 tenant 中的不同子网的隔离通过 iptables 来做，也意味着，同一 tenant 中的相同子网的两个虚机不用走 L3 层，直接走 L2 层能通，没问题；但如果同一 tenant 中的不同子网的两个虚机要通讯的话，必须得绕道 L3-agent 网络节点，这是影响性能的。
- 

## L4-L7

### LBaaS

OpenStack 最初的目标是做 x86 平台下的开源 IaaS 云，但它目前也有了类似于 LBaaS (Load Balance as a Service) 这样本属于 SaaS 的特性。

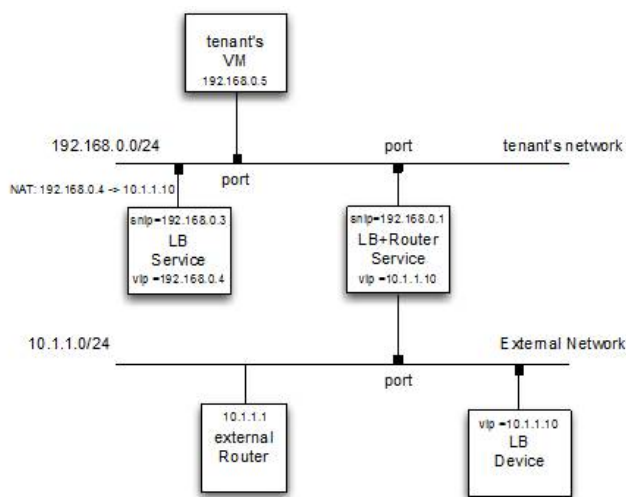
LbaaS 服务可以为一个 tenant 下的多台虚机提供负载均衡服务，要点如下：

- 提供负载均衡的虚机之间组成一个服务组
- 虚机之间提供心跳检查
- 外部通过 VIP 来访问 LB 服务组提供的服务，动态地将 VIP 根据负载均衡策略绑定到具体提供服务虚机的 fixed ip 上

下图 3 显示了 LBaaS 应用的几种情形：

图 3. LBaaS 的应用场景





说明如下：

- 有两个虚机 192.168.0.3 和 192.168.0.1 组成负载均衡池，它们之间做心跳
- 如果这个 LB 服务只用在内网，可以只为虚机提供一个内部的 vip，如 192.168.0.4
- 如果这个 LB 服务也需要从外部访问，那么这个 vip 可以做成 floating ip，如 10.1.1.10

所以实现上述 LB 服务的 neutron 命令如下：

#### 清单 1. 实现例子 LB 服务的 neutron 命令

```
neutron lb-pool-create --lb-method ROUND_ROBIN --name mypool --protocol HTTP --subnet-id <subnet-id>

neutron lb-member-create --address 192.168.0.3 --protocol-port 80 mypool
neutron lb-member-create --address 192.168.0.1 --protocol-port 80 mypool
neutron lb-healthmonitor-create --delay 3 --type HTTP --max-retries 3 --timeout 3
neutron lb-healthmonitor-associate <lb-healthmonitor-id> mypool

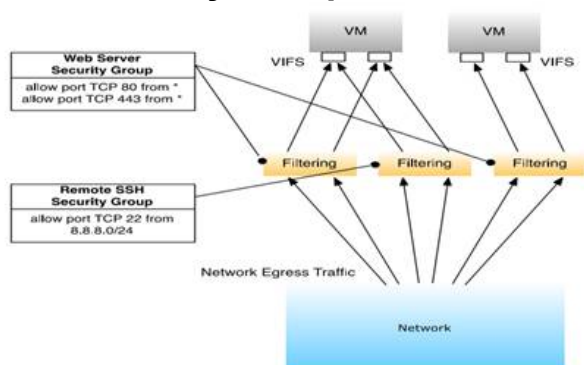
neutron lb-vip-create --name myvip --protocol-port 80 --protocol HTTP --subnet-id <subnet-id> mypool
neutron lb-vip-create --name myvip --protocol-port 80 --protocol HTTP --subnet-id <subnet-id> mypool

neutron floatingip-create public
neutron floatingip-associate <floating-ip-id> <lb-vip-port-id>
```

## FwaaS

目前，Neutron 已经实现了一个 Security Group 服务，FWaaS 服务正在开发中。FWaaS 和 Security Group 的原理差不多，只不过代码重构将以前 Security Group 的代码挪到了现有的 L4/L7 层框架来了；这样，以前 Security Group 作为 nova-compute 的一个组件只能运行在计算节点上，单纯拆出来做为一个独立进程之后也可以部署在 I3-agent 的那个物理机上提供边缘防火墙特性。下图 4 显示了 Security Group 服务的原理图：

图 4. Security Group 原理图



## Security Group 由一系列的 iptables rule 组成

- 这些 iptables rule 运行在计算节点上
- 可以控制从虚拟机出去的流量
- 可以控制进来虚拟机的流量
- 可以控制虚拟机之间的流量

实施 Security Group 的步骤如下，例如，四个虚拟机，可以从 80 端口访问 web 虚拟机，从 web 虚拟机上可以访问 database 虚拟机，ssh 跳转虚拟机上可以同时访问 database 和 web 虚拟机，可以通过端口 22 访问 ssh 虚拟机。

### 清单2. 实施Security Group 的步骤

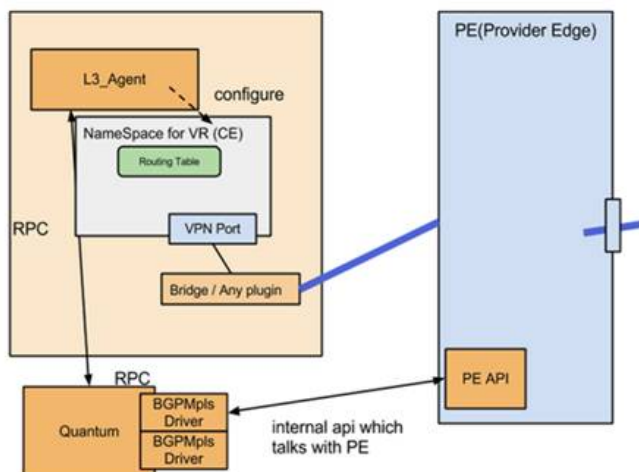
```
neutron security-group-create web
neutron security-group-create database
neutron security-group-create ssh
neutron security-group-rule-create --direction ingress --protocol TCP \
--port-range-min 80 --port-range-max 80 web
neutron security-group-rule-create --direction ingress --protocol TCP \
--port-range-min 3306 --port-range-max 3306 --remote-group-id web database
neutron security-group-rule-create --direction ingress --protocol TCP \
--port-range-min 22 --port-range-max 22 --remote-group-id ssh database
neutron security-group-rule-create --direction ingress --protocol TCP \
--port-range-min 22 --port-range-max 22 --remote-group-id ssh web
neutron security-group-rule-create --direction ingress --protocol tcp \
--port-range-min 22 --port-range-max 22 ssh
```

## VPNaaS

我们仍然从技术的本质角度出发来解释什么是 VPN。VPN 类似于 GRE，也是一种隧道。但是不同的是 VPN 也要求不同的用户可以使用相同的子网，所以在每个建立 VPN 的两个端点所对应的边缘路由器上都会为每个用户建立它自己的 VPN 路由实例 VRF，白话一点的话，VRF 就是 Linux 里上面提到过的 namespace 的概念。

建立 GRE 隧道容易，但是 VPN 强调私密性，一个组织的 VPN 总不希望不经授权的访问吧。所以一般使用 SSL 或者 IPSec 结合 GRE 来建立 VPN。也可以通过 MPLS 来建立 VPN。Neutron 中这三类 VPN 目前均在开发之中。从下面 MPLS VPN 的 blueprint 中的图 5 可以看到将来要实现的样子：

图 5. MPLS VPN 实现原理图



目前，l3-agent 没有使用动态路由，使用的是 namespace + ipv4 forward + iptables 的静态路由技术。l3-agent 现在充当着 CE 的角色（Custom Edge）。我们设想一下，

笔者认为 BGP Mpls Driver 将来至少要实现下列几个方面的内容：

- 调用 PE API 在边缘路由器 PE 上通过 namespace 特性定义并配置 tenant 之间隔离的自己的 VPN 路由实例 VRF
- tenant 定义的 VRF 路由并不需要在每个 WAN 核心路由上都存在，所以可以通过配置输入和输出策略实现，即使用 router-target import/export 命令导入或导出相应的 VRF 条目
- 配置 PE 到 CE 的链路
- 将 CE 的接口关联到预先定义的 VRF

---

## Neutron 有什么？

Neutron 就是一个定义良好的插件框架，用来调用上述 L2-L7 层不同实现，且对外提供北向 API 来提供虚拟网络服务。

它自己提供了 tenant 的概念，它只是一个 SDN 框架，和底层的 SDN 软件如 Dove 集成时可以提供 SDN 的功能，但需要将它的 tenant 和 SDN 自身的 tenant 概念之间做一个映射。

















### L2

在 L2 层，由虚拟交换机来实现。虚拟交换机有下列这些种：

- Linux 网桥，基于 Linux 内核的网桥。需要说明的是，网桥就是交换机，是交换机的一种常用的书面叫法。
- OpenvSwitch (OVS)：OVS 有两种模式，一种是当普通的虚拟交换机来使用，另一个是和 OpenFlow 控制器协作当作 OpenFlow 控制器来使用。
- 一些基于 Overlay 技术的 SDN 实现，如 Dove 等。
- 一些非开源的商业交换机。

目前，Neutron 已经实现的 L2 层插件如下图 6 所示，linuxbridge 实现了 Linux 网桥，openvswitch 插件实现了 openvswitch 网桥，bigswitch 插件实现了一种 SDN 控制器，ml2 是一种通用的插件（这些 L2 层的插件主要分写数据库的 plugin 部分和运行在计算节点的 agent 部分，plugin 写数据库的字段有差异但不多，所以代码重复，ml2 可以理解为一个公共的 plugin）。且每种插件基本上实现了 FLAT, VLAN, VXLAN, GRE 等四种拓扑。

图 6. Neutron 中 L2 层插件

- ▼  plugins
  - ▷  bigswitch
  - ▷  brocade
  - ▶  cisco
  - ▷  common
  - ▷  hyperv
  - ▷  linuxbridge
  - ▷  metaplugin
  - ▷  midonet
  - ▷  ml2
  - ▷  mlnx
  - ▷  nec
  - ▷  nicira
  - ▷  openvswitch
  - ▷  plumgrid
  - ▷  ryu

## L3

Neutron 的 L3 层由基于 namespace ipv4 forward + iptables 实现。

需要启动 l3-agent 进程，如果需要 DHCP 服务的话也需要启动 dhcp-agent 进程。

## L4-L7

LBaaS 基于 Haproxy 实现；FWaaS 基于 iptables 实现；VPNaaS 有基于 IPsec 的 VPN 实现，也有基于 MPLS 的 VPN 实现，也有基于 SSL 的 VPN 实现。

截止到笔者写此文为止，目前只有 LBaaS 能用，其他的 FWaaS, VPNaaS 和 NATaaS 仍在开发中。但对于 FWaaS 有 Security Group 功能可用，区别在于前者由于作为独立服务也能部署在网络节点上提供边缘防火墙特性，后者依然能够在计算节点上使用 iptables 规则控制从虚机出去的，进来虚机的，虚机之间的流量。

## 结论

本文在《漫步云中网络》姊妹篇讲述 L2-L3 网络原理的基础上，继续讲述了最新的 Neutron 代码背后 L2-L7 层所依赖的网络技术内幕，让读者能够在心中建立起一个 High Level 的完整的 Neutron 架构图。

## 参考资料

学习

- [漫步云中网络](#)：在生产环境中，云中的网络通常被划分为公共网络、管理网络和服务网络。本文首先通过三个小试验向您介绍了如何通过 TAP/TUN、NAT、Linux Bridge、VLAN 等技术实现云中网络的一般原理。
- 参考 [Devstack 官网](#)，您可以从 devstack 脚本快速上手 Openstack。



**IBM Bluemix** 资源中心  
文章、教程、演示，帮助您构建、部署和管理云应用。



**developerWorks** 中文社区  
立即加入来自 IBM 的专业 IT 社交网络。



**IBM 软件资源中心**  
免费下载、试用软件产品，构建应用并提升技能。



- 参考 [Openstack 官网](#)，您可以获得更多关于 Openstack 的知识。
- 参考 [Neutron Wiki](#)，您可以获得关于 Neutron 相关的知识。
- [developerWorks 云计算站点](#) 提供了有关云计算的更新资源，包括
  - 云计算 [简介](#)。
  - 更新的 [技术文章和教程，以及网络广播](#)，让您的开发变得轻松，[专家研讨会和录制会议](#) 帮助您成为高效的云开发人员。
  - 连接转为云计算设计的 [IBM 产品下载和信息](#)。
  - 关于 [社区最新话题](#) 的活动聚合。

## 讨论

- 加入 [developerWorks 中文社区](#)。查看开发人员推动的博客、论坛、组和维基，并与其他 developerWorks 用户交流。
-