HZ BOOKS
华章IT

云计算与虚拟化技术丛书

Understanding the
OpenStack Neutron

深入理解 OpenStack Neutron

李宗标 著

华为资深网络工程师撰写，云计算专家联袂推荐，行文风趣流畅、循序渐进

抽丝剥茧，深入分析Neutron网络、资源、架构与服务、插件、消息机制和经典API函数处理的核心技术原理与实现

机械工业出版社
China Machine Press

云计算与虚拟化技术丛书

Understanding the
OpenStack Neutron

深入理解 OpenStack Neutron

李宗标 著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

深入理解 OpenStack Neutron / 李宗标著 . —北京：机械工业出版社，2017.12（2018.5 重印）
（云计算与虚拟化技术丛书）

ISBN 978-7-111-58448-3

I. 深… II. 李… III. 计算机网络 IV. TP393

中国版本图书馆 CIP 数据核字（2017）第 278307 号

深入理解 OpenStack Neutron

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）
责任编辑：高婧雅 责任校对：张惠兰
印 刷：三河市宏图印务有限公司 版 次：2018 年 5 月第 1 版第 2 次印刷
开 本：186mm × 240mm 1/16 印 张：22.75
书 号：ISBN 978-7-111-58448-3 定 价：89.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88379426 88361066 投稿热线：（010）88379604
购书热线：（010）68326294 88379649 68995259 读者信箱：hzit@hzbook.com

Foreword 序

自从我在 OpenStack 香港峰会做了“深入探索 Neutron”的主题分享后，很少看见有从业者如此专心研究 Neutron 代码并且整理和分享出来。于是我一看到样章便欣然答应审稿，并索要了全文稿件阅读。正值国庆并中秋假期，本是出去游玩的计划也取消了，不但免了外面喧嚣、拥堵之苦，还饱尝了稿内流畅、风趣之美，值！

作者不是简单地罗列 Neutron 代码，而是从头到尾都有自己的总结和理解。细致的图文解说令人记忆深刻。Neutron Ocata 版代码近乎 30 万行，要想透彻掌握，除了扎实的 Python 语言知识技能、丰富的网络领域知识，还要有铁杵磨成针的信念和毅力、为公也为己的开源分享精神、踏实不轻浮的从业素质。从本书来看，作者在这些方面都有比较深的造诣，值得本人学习。

虽然此书只讲述了 Neutron 社区实现版本中基本的二层和三层部分，但是脉络清晰，行文循序渐进。阅读本书，建议读者先安装好 OpenStack 环境，有了基本的 Neutron 网络操作体验后，下载好源代码，准备好 UML 画图工具，从第 1 章开始一直读到最后。读完之后，如果读者能自己看着源代码把各种功能的 UML 相关图整理出来，本书的目的就达到一半了。“师傅领进门，修行在个人”，我想获取知识，自我成长的道理都是如此。

学习 Neutron 的另一个关键是不要有固定模式。Neutron 的核心是 API 以及背后的资源模型，社区实现版本可以作为参考，因为我们在给客户实施部署时，可能要换成其他厂商的实现版本。在深知 Neutron 的内涵之后，提供出灵活多变，适应客户需求的虚拟网络解决方案才是我们的目的。也只有深知内涵，才能有变化，我想这也是本书“深入理解”几个字的内在含义。所以读完此书，不要停止，继续挖掘 Neutron 虚拟网络的背后逻辑、问题和可变部分，这样才能达到“应用自如，万变不离其宗”的境界。

Neutron 定义了一组云计算中使用的网络模型，其后面实现可以是实在的网络硬件，也可以是虚拟的网络功能（网元）。虚虚实实，实中有虚，虚中有实，能根据客户的现实环境进

行虚实结合，然后对 Neutron 进行定制化的部署甚至实现，是我对我们公司 Neutron 从业人员的要求。我想这个要求和本书作者对 Neutron 源码进行深入分析的目的是一致的。

总之，这本《深入理解 OpenStack Neutron》既有对 Neutron 虚拟网络背后的网络原理方面的阐述，也有对 Neutron 的数据模型、启动过程、消息处理机制和经典 API 函数处理的源码分析。语言网络化，风趣而又流畅；知识通俗化，深刻又易懂。相信此书能帮助读者进一步掌握 Neutron 虚拟网络，为以后的实践打下扎实的基础。

九州云 CTO　龚永生

2017 年 10 月　中秋夜

Preface 前　言

为什么写作本书

2016 年 1 月 16 号，我在微信公众号（标哥说天下）发表了 Neutron 系列的第一篇文章，当时计划是半年写完，没想到写了一年半。也许是由于冲动吧，那天我决定写一系列有关 Neutron 的文章。

手里拎个锤子，认为满世界都是钉子，这是一种要命的思维逻辑。写 Neutron 系列，最初的原因，不是因为需要用它做什么，而是想要说明它不能做什么。这对于从事云计算的人来说，可能根本就不是问题，因为 Neutron 的适用范围也恰好在他们的工作范围之内，没有逾越半步。

对于那些从事非云计算行业，却又与 Neutron 有着千丝万缕的联系的人来说，这可能也不是问题。但是如果领导与“专家”太多，这可能就是问题了。

然而在写作的过程中，我逐步放弃了想说明“ Neutron 能做什么，不能做什么”的想法，**慢慢改变为现在的思路：努力描述 Neutron 原本是什么！**

本书首先从 Linux 虚拟网络知识讲起，步步推进，最后深入到 Neutron 的代码。当剥开 Neutron 代码的面纱以后，Linux 虚拟网络既是构建 Neutron 网络的基础，也是构建 Neutron 代码的基础。Neutron 的代码写到最后，不过是调用 Linux 的命令行而已（也包括调用 OVS 的命令行）。

这么说，当然对 Neutron 不公平，调用命令行只是 Neutron 代码的最后一步，在这之前，Neutron 还需知道调用什么命令行。这不是是否熟练掌握各种命令行的问题，而是能否正确分配逻辑资源的问题。举个最简单的例子，为一个端口配置一个 IP 地址，在调用命令行之前，Neutron 需要通过各种机制和算法为这个端口分配一个正确的 IP 地址。

Neutron 不仅需要具备分配逻辑资源的机制，还需要创建相应的虚拟网元（物理资源），并将这些虚拟网元正确地连接和配置，构建成正确的网络。Neutron 所能支持的二、三层网络特性，不仅体现在它的代码中，也体现在它的资源模型中，并且以 RESTful 接口的形式对外提供服务。

如何阅读本书

总体来说，Neutron 并不神秘，也不深奥，却比较庞大。如果你对 Neutron 的代码细节比

较感兴趣，本书会有大量的篇幅进行代码剖析。如果你仅仅想了解 Neutron 的基本原理，本书的前几章也正是这个目的，希望能对你有些帮助。

Neutron 是一个关于网络的系统，本书第 1 章介绍了一些背景知识。在阅读本书之前，读者首先得具备一定的网络知识，也正是出于这样的目的，第 2 章介绍了 Linux 的虚拟网络知识。限于主题和篇幅的原因，本书没法再过多介绍其他内容。本书假设你对基本的 TCP/IP 协议、VXLAN、OVS 等有一定的了解。当然，如果要阅读代码剖析那些章节，那么还需要对 Python 有一定的了解，因为 Neutron 的编程语言就是 Python。第 3 章讲述了 Neutron 的实现模型。第 4 章讲述的是 Neutron 的资源模型。第 5 章讲述了 Neutron 的基本架构，以及架构中所涉及的 Web 机制、通信机制、并发机制等，这些都是 Neutron 的基本原理。第 6 章主要讲述 Neutron 如何启动 Web Server，并通过 WSGI Pipeline 机制调用合适的 WSGI Application，以及 WSGI Application 如何巧妙地寻址到正确的 Plugin。第 7 章主要讲述 Plugin 如何如何处理 Neutron 的 RESTful 请求，如何进行逻辑资源分配，如何调用 Agent。第 8 章主要讲述 Agent 如何配置（虚拟）网元，以构建 Neutron 网络。

但是无论多么细节的代码剖析，也没法做到将 Neutron 的每一行代码都讲述到。所以本

目　录 *Contents*

序

前　言

第 1 章　Neutron 概述 ····· 1

1.1　Neutron 的由来 ····· 1

1.2　Neutron 的特性与应用 ····· 3

1.2.1　基于 OpenStack 的应用 ····· 4

1.2.2　基于 SDN 的应用 ····· 6

1.3　Neutron 的扩展能力 ····· 8

1.4　本章小结 ····· 9

第 2 章　Linux 虚拟网络基础 ····· 11

2.1　tap ····· 11

2.2　namespace ····· 13

2.3　veth pair ····· 16

2.4　Bridge ····· 17

2.5　Router ····· 19

2.6　tun ····· 21

2.7　iptables ····· 24

2.7.1　NAT ····· 27

2.7.2　Firewall ····· 30

2.7.3　mangle ····· 32

2.8　本章小结 ····· 32

第 3 章　Neutron 的网络实现模型 ····· 34

3.1　Neutron 的三类节点 ····· 34

3.2　计算节点的实现模型 ····· 35

3.2.1　VLAN 实现模型 ····· 37

3.2.2　VXLAN 实现模型 ····· 41

3.2.3　GRE 实现模型 ····· 44

3.2.4　计算节点的实现模型小结 ····· 45

3.3　网络节点的实现模型 ····· 46

3.4　控制节点的实现模型 ····· 49

3.5　本章小结 ····· 49

第 4 章　Neutron 的资源模型 ····· 51

4.1　Neutron 资源的租户隔离 ····· 51

4.1.1　Neutron 语境下租户隔离的含义 ····· 52

4.1.2　Neutron 在租户隔离中的无限责任和有限责任 ····· 53

4.1.3　Neutron 的租户隔离实现方案 ····· 54

4.1.4　租户隔离小结 ····· 56

4.2　Network ····· 57

4.2.1 运营商网络和租户网络 …… 58
4.2.2 物理网络 …… 61
4.2.3 Network 小结 …… 64
4.3 Trunk Networking …… 65
4.3.1 Bridge 的 VLAN 接口模式 …… 65
4.3.2 VLAN aware VM 与 Trunk Networking …… 69
4.3.3 Trunk Networking 小结 …… 78
4.4 Subnet …… 79
4.4.1 IP 核心网络服务 …… 80
4.4.2 Subnet 资源池 …… 81
4.5 Port …… 83
4.6 Router …… 86
4.6.1 Router 的外部网关 …… 88
4.6.2 增加 Router 接口 …… 89
4.6.3 Router 的路由表 …… 91
4.6.4 Floating IP …… 92
4.6.5 Router 小结 …… 94
4.7 Multi-Segments …… 95
4.7.1 Multi-Segments 的困惑 …… 96
4.7.2 Multi-Segments 的几个应用场景 …… 98
4.8 BGP VPN …… 102
4.8.1 BGP VPN 的使用场景 …… 103
4.8.2 BGP VPN 的实现模型 …… 104
4.8.3 BGP VPN 的资源模型 …… 105
4.9 本章小结 …… 109

第 5 章 Neutron 架构分析 …… 112

5.1 Neutron 的 Web 框架与规范 …… 115
5.2 Neutron 的消息通信机制 …… 117
5.2.1 AMQP 基本概念 …… 118
5.2.2 AMQP 的消息转发 …… 118
5.3 Neutron 的并发机制 …… 122
5.3.1 协程概述 …… 122
5.3.2 Neutron 中的协程 …… 124
5.4 通用库 Oslo …… 131
5.5 本章小结 …… 131

第 6 章 Neutron 的服务 …… 132

6.1 Neutron 启动一个 Web Server …… 133
6.1.1 Web Server 的启动过程 …… 133
6.1.2 Web Server 启动过程中的关键参数 …… 135
6.1.3 Web Server 的进程与协程 …… 138
6.1.4 小结 …… 142
6.2 加载 WSGI Application …… 142
6.2.1 api-paste.ini 对应的 WSGI Application …… 144
6.2.2 neutronapi_v2_0 section …… 146
6.3 Core Service API（RESTful）的处理流程 …… 148
6.3.1 Core Service 的 WSGI Application …… 149
6.3.2 Core Service 处理 HTTP Request 的基本流程 …… 149
6.3.3 Core Service 处理 HTTP Request 的函数映射 …… 153
6.3.4 小结 …… 162
6.4 Extension Service API（RESTful）的处理流程 …… 164
6.4.1 Extension Service 的类图与加载 …… 164

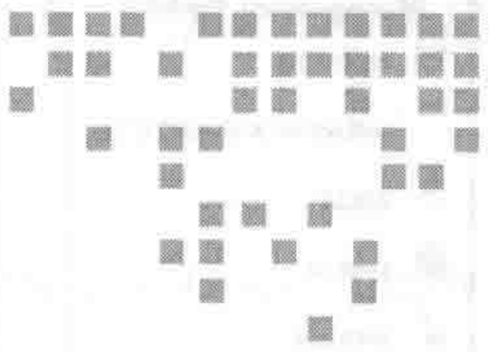
6.4.2 Extension Service 的 WSGI Application ····· 167
6.4.3 Extension Service 处理 HTTP Request 的基本流程 ····· 169
6.4.4 Extension Service 处理 HTTP Request 的函数映射 ····· 171
6.4.5 小结 ····· 176
6.5 Plugin 的加载 ····· 178
6.5.1 Core Service Plugin 的加载 ····· 179
6.5.2 Extension Services Plugin 的加载 ····· 180
6.6 RPC Consumer 的创建 ····· 181
6.6.1 Neutron Plugin 创建 RPC Consumer 的接口 ····· 182
6.6.2 Neutron Server 启动 RPC Consumer ····· 183
6.7 本章小结 ····· 187

第 7 章 Neutron 的插件 ····· 190

7.1 核心插件 ····· 191
7.1.1 ML2 插件简介 ····· 193
7.1.2 类型驱动 ····· 193
7.1.3 机制驱动 ····· 202
7.1.4 ML2 插件 create_network 函数剖析 ····· 224
7.1.5 ML2 插件 create_subnet 函数剖析 ····· 229
7.1.6 ML2 插件 create_port 函数剖析 ····· 240
7.2 业务插件 ····· 249
7.2.1 Router Plugin 的 create_router 函数分析 ····· 250
7.2.2 Router Plugin 的 add_router_interface 代码分析 ····· 257
7.3 Neutron Plugin 的消息发布和订阅 ····· 260
7.3.1 Neutron Plugin 中的 Callbacks Module 机制 ····· 261
7.3.2 Neutron Plugin 中的 RPC 机制 ····· 265
7.4 本章小结 ····· 266

第 8 章 Neutron 的代理 ····· 268

8.1 OVS Agent ····· 270
8.1.1 三类关键的 Bridge ····· 270
8.1.2 内外 VID 的转换 ····· 288
8.1.3 OVS Agent 代码分析 ····· 295
8.1.4 OVS Agent 小结 ····· 309
8.2 L3 Agent ····· 311
8.2.1 class OVSInterfaceDriver 分析 ····· 312
8.2.2 class RouterInfo 分析 ····· 317
8.2.3 L3 Agent 代码分析 ····· 326
8.2.4 L3 Agent 小结 ····· 351
8.3 本章小结 ····· 352



第 1 章 Chapter 1

Neutron 概述

2010 年 7 月，OpenStack 开源社区（遵循 Apache License 2.0）成立，NASA（美国航天局）和 Rackspace 分别贡献出各自已有的虚拟化管理项目 Nova 和对象存储项目 Swift 作为 OpenStack 初始项目。2010 年 10 月，OpenStack 第一个正式版本 Austin 发布。

2012 年 Rackspace 为了社区的健康发展，捐出所有 OpenStack 相关的版权成立 OpenStack 基金会。基金会的设立使得 OpenStack 社区参与者更加多样化，操作系统、服务器、网络、运营商等纷纷加入。由此开始，OpenStack 社区开启了快速发展的繁荣局面。

Neutron 作为 OpenStack 的核心项目之一，在 OpenStack 的发展中，自然也功不可没。

1.1 Neutron 的由来

OpenStack 发展至今，已经有 46 个正式项目，Neutron 属于其中一个核心项目，如图 1-1 所示。

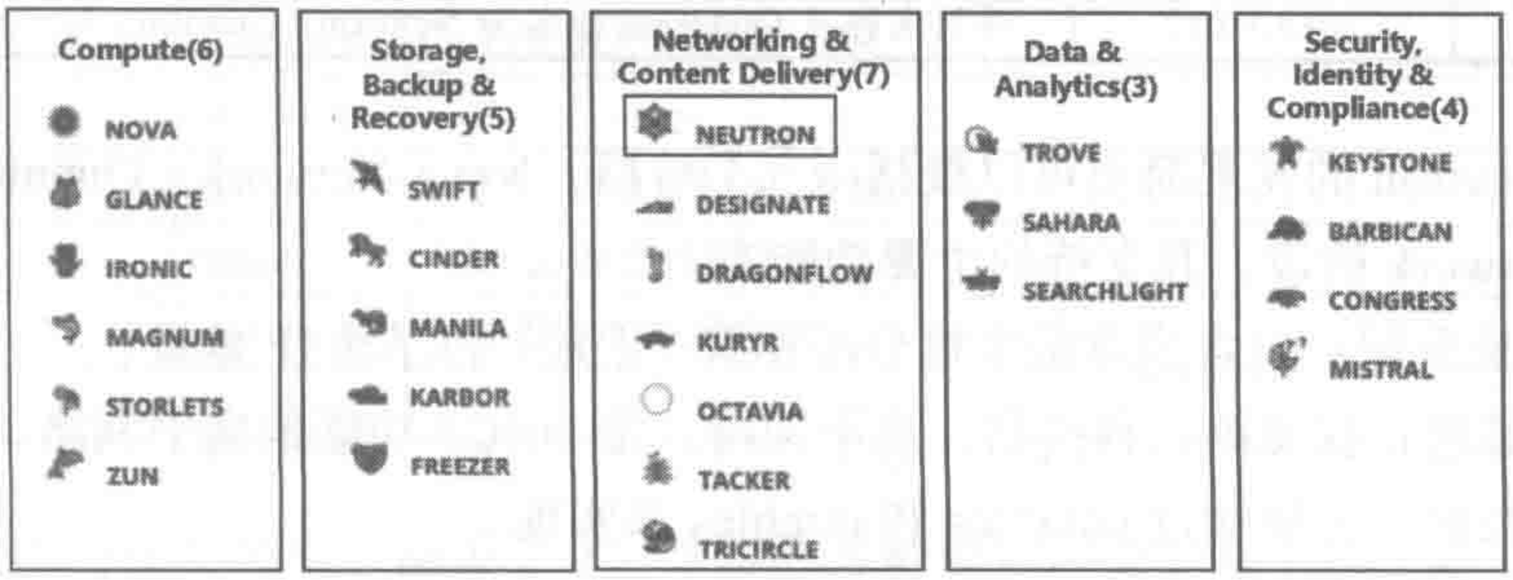


图 1-1　OpenStack 的项目（截至 2017.07）

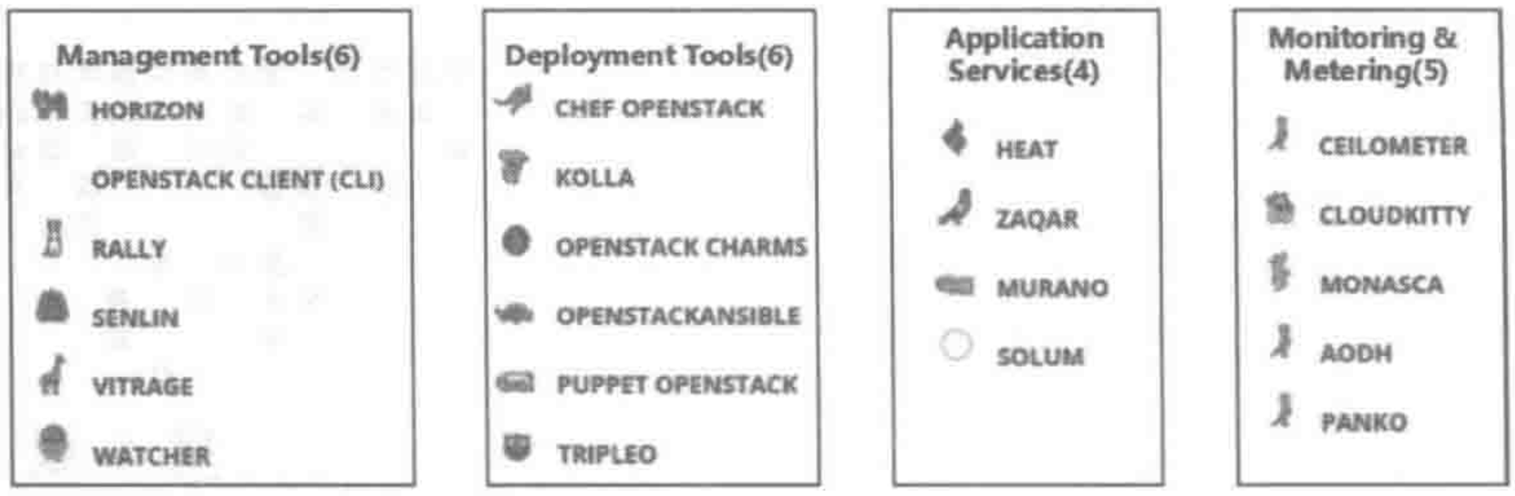


图 1-1 （续）

Neutron 在 OpenStack 的主要服务中所处的上下文，如图 1-2 所示。

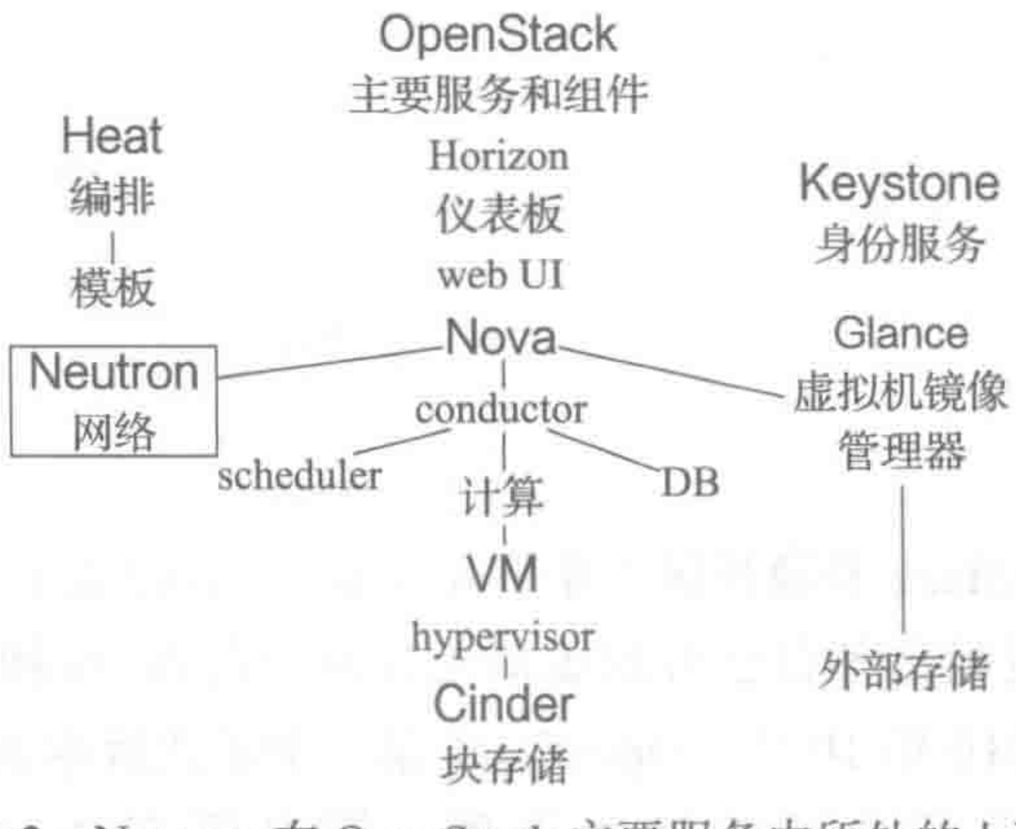


图 1-2 Neutron 在 OpenStack 主要服务中所处的上下文

当前，Neutron 已经成为 OpenStack 三大核心（存储、计算、网络）之一，对外提供 NaaS（Network as a Service）服务。但是当初 Neutron 只是 Nova 项目中的一个模块而已，到 Folsom 版本才正式从中剥离出来，成为一个正式并且核心的项目，如表 1-1 所示[⊖]。

表 1-1 Neutron 的由来

版本号	发行日期	Neutron 历史
Austin	2010.10.21	作为 Nova 中的一个模块 nova-network 存在
Essex	2012.04.05	网络功能数据模型开始从 Nova 中剥离，为独立项目做准备
Folsom	2012.09.27	正式从 Nova 中剥离，成立新的独立项目 Quantum，并且是核心项目
Havana	2013.10.17	项目名称从 Quantum 改名为 Neutron（Quantum 与一家公司名称冲突）

由此，Neutron 的发展简史可以概括为三个阶段：Nova-Network、Quantum、Neutron。

Nova-Network 阶段，其支持的主要功能有：

1）IP 地址分配：包含为虚拟主机分配私有（固定）和浮动 IP 地址；

2）网络管理：仅支持三种网络，扁平网络、带 DHCP 功能的扁平网络、VLAN 网络；

3）安全控制：主要通过 ebtables 和 iptables 来实现。

⊖ 参考维基百科 OpenStack 条目。

可以看到，Nova-Network 所支持的功能还比较简单。到了 Quantum 阶段，其支持的主要功能有：

1）支持多租户隔离，并提供面向租户的 API；

2）插件式结构支持多种网络后端技术，包括 Open vSwitch、Cisco、Linux Bridge、Nicira NVP、Ryu、NEC 等；

3）支持位于不同的 2 层网络的 IP 地址重叠；

4）支持基本的 3 层转发和多路由器；

5）支持隧道技术（Tunneling）；

6）支持三层代理和 DHCP 代理的多节点部署，增强了扩展性和可靠性；

7）提供负载均衡 API（试用版本）。

Quantum 阶段所支持的功能已经初具规模。有了 Quantum 打下的良好基础，进入第三阶段以后，Neutron 所支持的功能和应用场景，得到了更大的发展。

1.2　Neutron 的特性与应用

从 Nova-Network 起步，经过 Quantum，再跨入新时代以后，经过几年的积累，Neutron 在网络的各个方面都取得了长足的发展。当前 Neutron 支持的特性，如表 1-2 所示。

表 1-2　Neutron 支持的特性

特性	状态	备　注
Networks	required	支持的网络类型有：Local、Flat、VLAN、VXLAN、GRE、Geneve 6 种
Subnets	required	—
Ports	required	—
Routers	required	支持路由转发、SNAT、DNAT、外部网关等功能
Security Groups	mature	—
External Networks	mature	支持 Floating IP 和安全组
DVR	immature	分布式虚拟路由（Distributed Virtual Routers）
L3 High Availability	immature	支持 VRRP（Virtual Router Redundancy Protocol，虚拟路由冗余协议）
Quality of Service	mature	QoS
Border Gateway Protocol	immature	支持 BGP/MPLS VPN
DNS	mature	—
Trunk Ports	mature	支持 VLAN aware VM
Metering	mature	L3 路由器级别流量度量
Routed Provider	immature	将多段 3 层网络封装为一个网络实体
Networks		—

Neutron 支持的这些特性，涵盖了 2 ～ 7 层的各种服务。除了基本的、必须支持的二层、三层服务，Neutron 在 4 ～ 7 层支持的服务有：LBaaS（负载均衡即服务）、FWaaS（防火墙

即服务）、VPNaaS（VPN 即服务）、Metering（网络计量服务）、DNSaaS（DNS 即服务）等。Neutron 在大规模高性能层面，还支持 L2pop、DVR、VRRP 等特性。

Neutron 的应用分为两大类：基于 OpenStack 的应用、基于 SDN 的应用。前者是在云的场景下，与 OpenStack 其他部件一起配合，为用户提供云服务。后者是在 SDN 场景下，与 SDN Controller 一起配合，为用户提供网络服务。

1.2.1 基于 OpenStack 的应用

基于 OpenStack 的应用，就是原生的云应用，Neutron 作为 OpenStack 中的网络部件，为用户提供网络服务。

早期的 Nova-Network 时代，Neutron 所支持的网络模型还比较简单，仅仅是单一的网络平面（或者是 Flat，或者是 VLAN），而且没有租户隔离，如图 1-3 所示[⊖]。

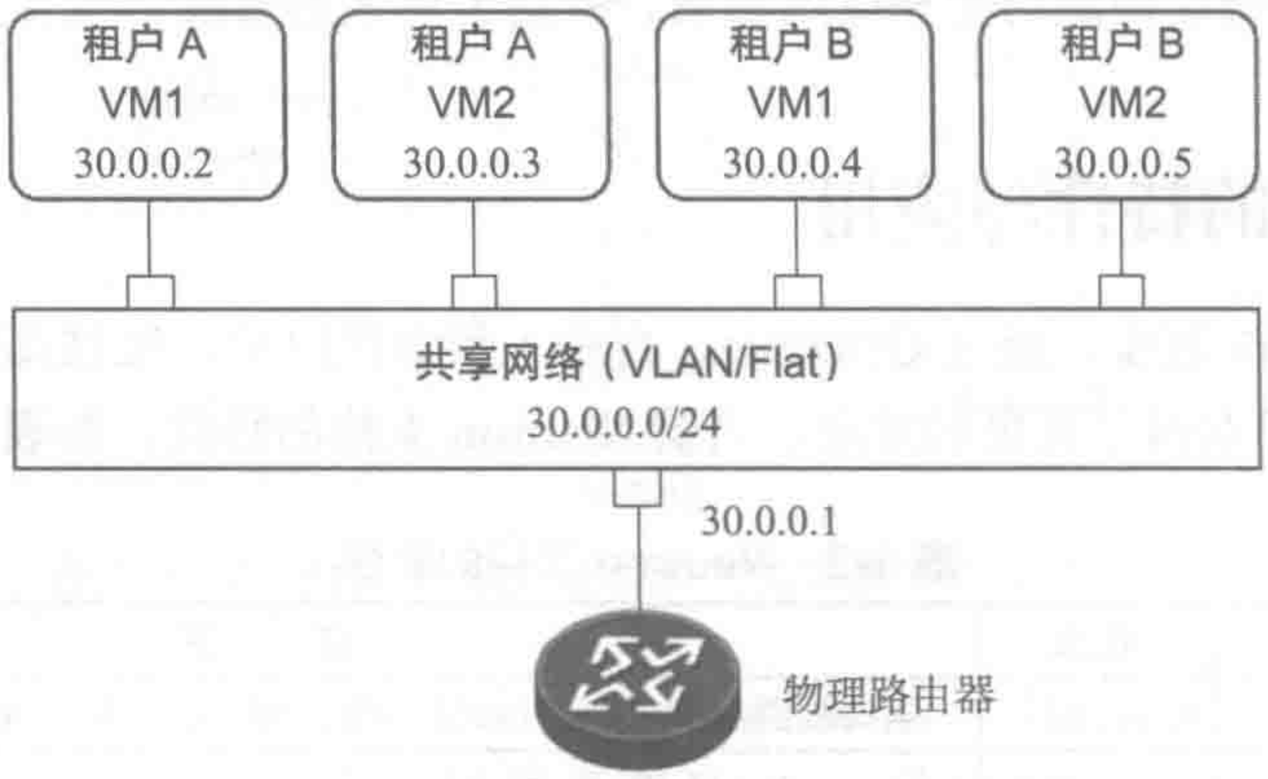


图 1-3 单平面租户共享网络

图 1-3 中，共享网络那个方框是 Neutron 的管理范围，它所表达的意思是：租户可以创建许多虚拟机，但是这些虚拟机只能连接到一个网络上。不同的租户共享一个网络，而且这个网络要么是 VLAN 网络，要么是 Flat 网络，只能是单一的网络类型。虽然 Neutron 在当时能够支持两种网络类型，但是在同一时刻，只能选择一种（虽然也没什么好选的）。另外，Neutron 还不支持路由器，它必须借助外部路由器（图 1-3 中的 Physical Router）才能有三层路由功能，这也意味着，不同租户的 VM 也必须在同一个网段，而且 IP 地址也不能重叠。

笔者当初第一眼看到这个图的时候，内心是崩溃的。不是因为它简单，而是因为它太简单。如果我们以物理世界的交换机、路由器来组网的话，单一平面网络就相当于图 1-4 所示的组网图：

单一平面网络，名字取得这么清新脱俗，不过是实现了一个低端交换机而已。人生若只如初见，何事秋风悲画扇。说句心里话，如果网络技术真的这么简单倒挺好，大家也不用费那么大力气去学习了。

⊖ 参考 https://www.ibm.com/developerworks/cn/cloud/library/1402_chenhy_openstacknetwork/。

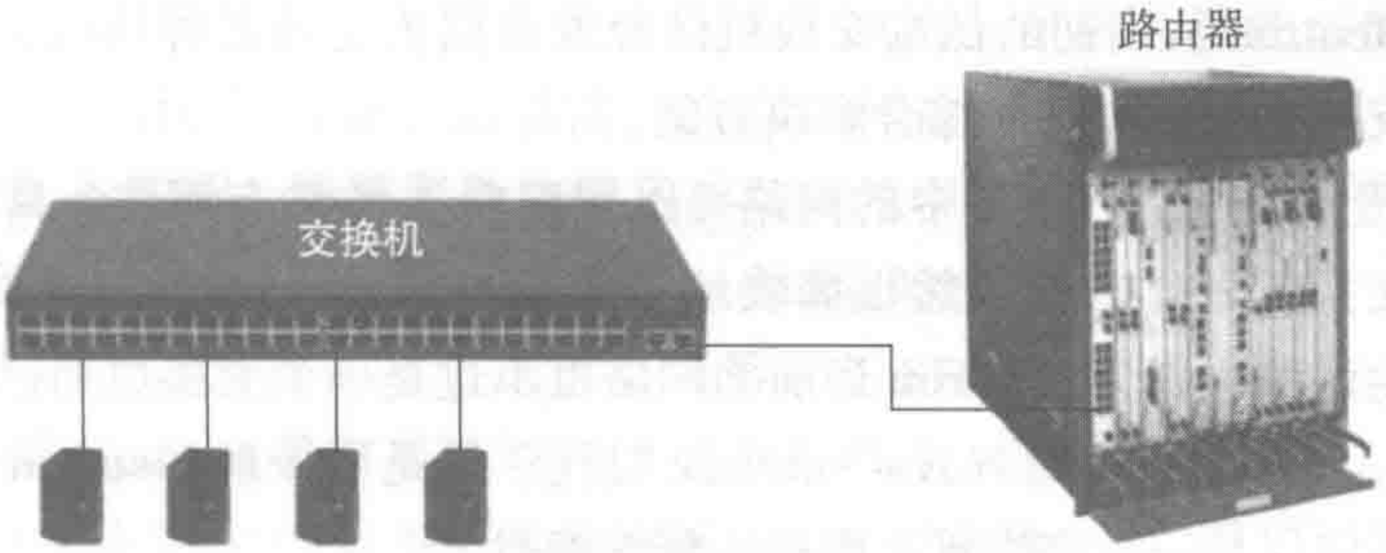


图 1-4　单平面网络在物理世界的映射

世界是复杂的，网络也是复杂的，Neutron 也在慢慢成长，变成了今天的模样——支持多平面租户私有网络，如图 1-5 所示。

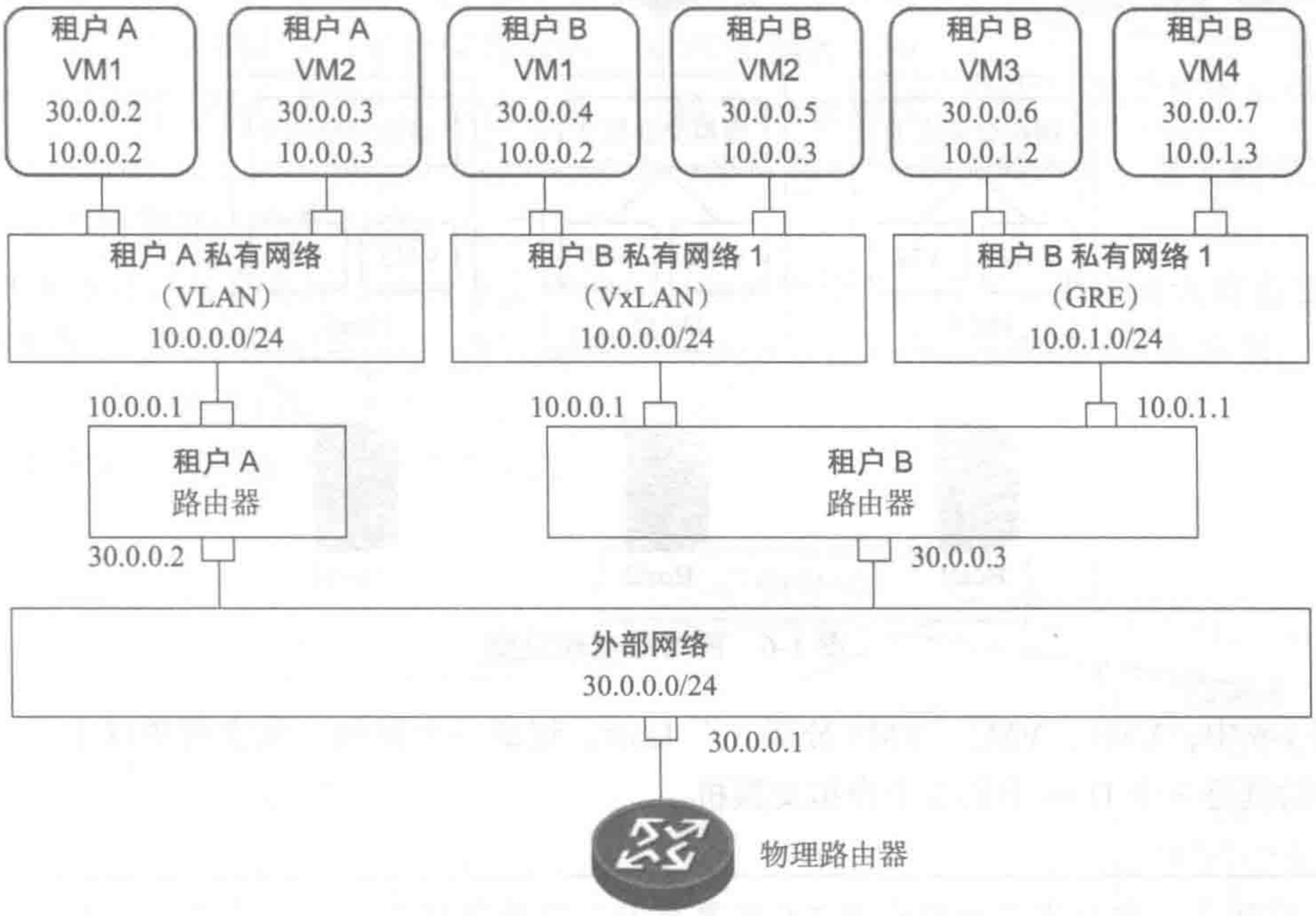


图 1-5　多平面租户私有网络

图 1-5 所表达的网络特征，总结如表 1-3 所示。

表 1-3　多平面租户私有网络的特征

特征	说　明
多平面	图 1-5 中同时有 VLAN、VXLAN、GRE 三种类型的网络
租户私有	租户 A 独占一个网络，租户 B 独占两个网络；租户 A 独占一个路由器，租户 B 独占一个路由器
地址重叠	租户 A 的 VM1、VM2 与租户 B 的 VM1、VM2，两者地址重叠
SNAT/DNAT	租户 A、B 的虚拟机访问外部网络时，支持 SNAT/DNAT
其他	表 1-2 所列的服务都可以支持，只是图 1-5 中没有表现出来而已

可以看到，Neutron 从当初的低端交换机已经发展成为支持各种协议，融合交换机、路由器为一体的，支持多租户隔离的综合解决方案。

有一点需要澄清一下，图 1-5 中的网络（比如租户 A 的私有网络）具体指的是什么？图 1-4 用了一个交换机做比方，虽然很痛快地说明了当初 Nova-Network 时代功能的不足，但是容易让人产生误解，以为 Neutron 当前的网络也不过是一个交换机而已，只是高端一点罢了。其实不然，无论是当初的 Nova-Network 时代，还是当今的 Neutron 时代，网络从具体实现来说，都不仅仅是一个交换机，而是一群交换机。

从模型角度来说，网络指的是 Neutron 的资源模型，是一个逻辑概念。从实现来说，网络指的是一群交换机，如图 1-6 所示。

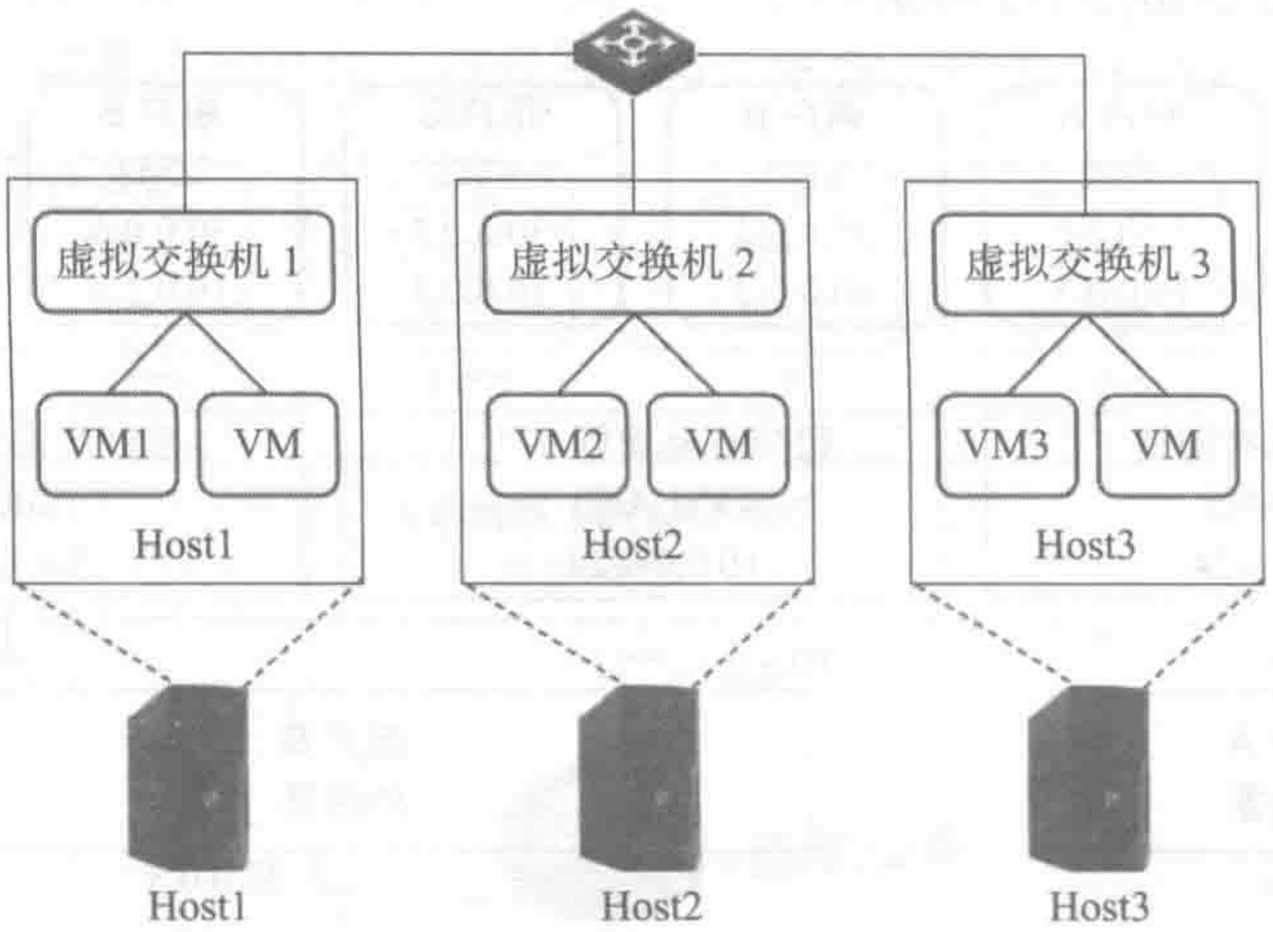


图 1-6　网络的实现模型

图 1-6 中，VM1、VM2、VM3 分属 3 个 Host，组成一个网络。从实现角度来说，这个网络指的就是 3 个 Host 中的 3 个虚拟交换机。

说明　实际上，虚拟机交换机比图 1-6 还要复杂，这里仅仅是一个简单示意。具体请参见第 3 章的描述。

Neutron 基于 OpenStack 的应用，网络的实现一般都是 Host 内的虚拟交换机。在 SDN 场景下，网络的实现很多选用厂商的硬件交换机（和路由器）。

1.2.2　基于 SDN 的应用

一千个人的心中，就有一千个 SDN。本文所说的 SDN，指的是传统老牌设备厂商推出的 SDN 方案，如图 1-7 所示：

图 1-7 所描述的应用场景，可以说是 SDN 浪潮大背景下的运营商与传统老牌设备商心照不宣各取所需的“创新”方案。运营商一直以来都被各个设备厂商提供的形态各异、纷繁复杂的管理接口所深深“伤害”，一直期望能有一个统一的接口。随着 OpenStack 的发展，Neutron 接口成为不少运营商的一个选择。当然，在 SDN 浪潮之下，仅仅选择 Neutron 接口是不够的，必须得沾上 SDN 的仙气。于是图 1-7 中的 SDN 控制器的存在就成为一种必然。这正好也中了设备商的下怀。1.2.1 节提到，基于 OpenStack 的 Neutron 应用，其选择的基本是 Host 内的虚拟交换机、路由器，这是传统老牌设备商所不能接受的——如果都选择了虚拟设备，那自家生产的交换机、路由器卖给谁去？于是设备商就与运营商一拍即合，推出了自家的“ SDN 控制器 + 硬件设备（交换机、路由器）”综合解决方案，并将 SDN 控制器挂接在 Neutron 的下面，对外以 Neutron 的统一接口和 SDN 的面貌出现，其实醉翁之意不在酒，在乎销售自己的硬件设备也。

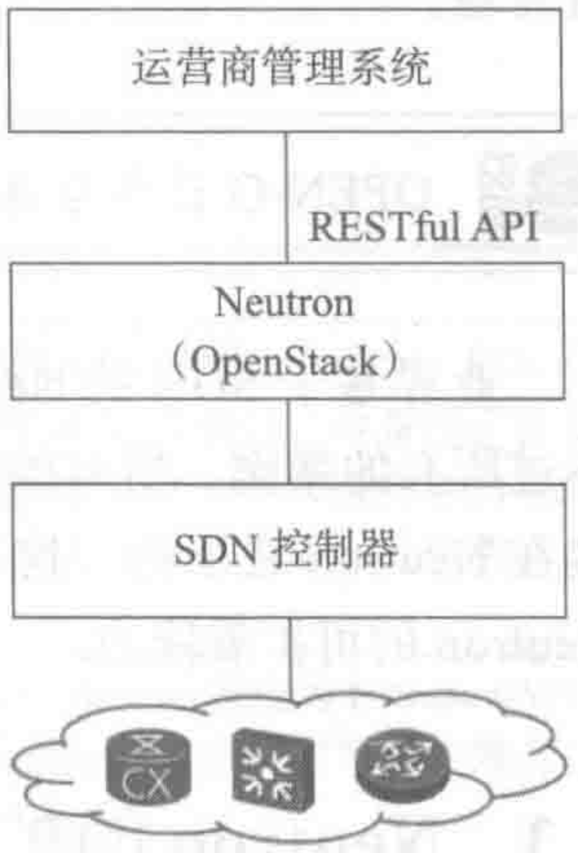


图 1-7　基于 SDN 的应用场景

客观地说，设备商以这样的方案推销自己的硬件设备，除了在商言商无可指责以外，从技术角度而言，其实也是一个比较正确的选择，毕竟当前的虚拟交换机、路由器，其性能还是不能跟硬件设备相比，在很多场景下还是有点力不从心。

下面我们就以开源组织 OPEN-O 的一个用例为例来直观感受一下，如图 1-8 所示。

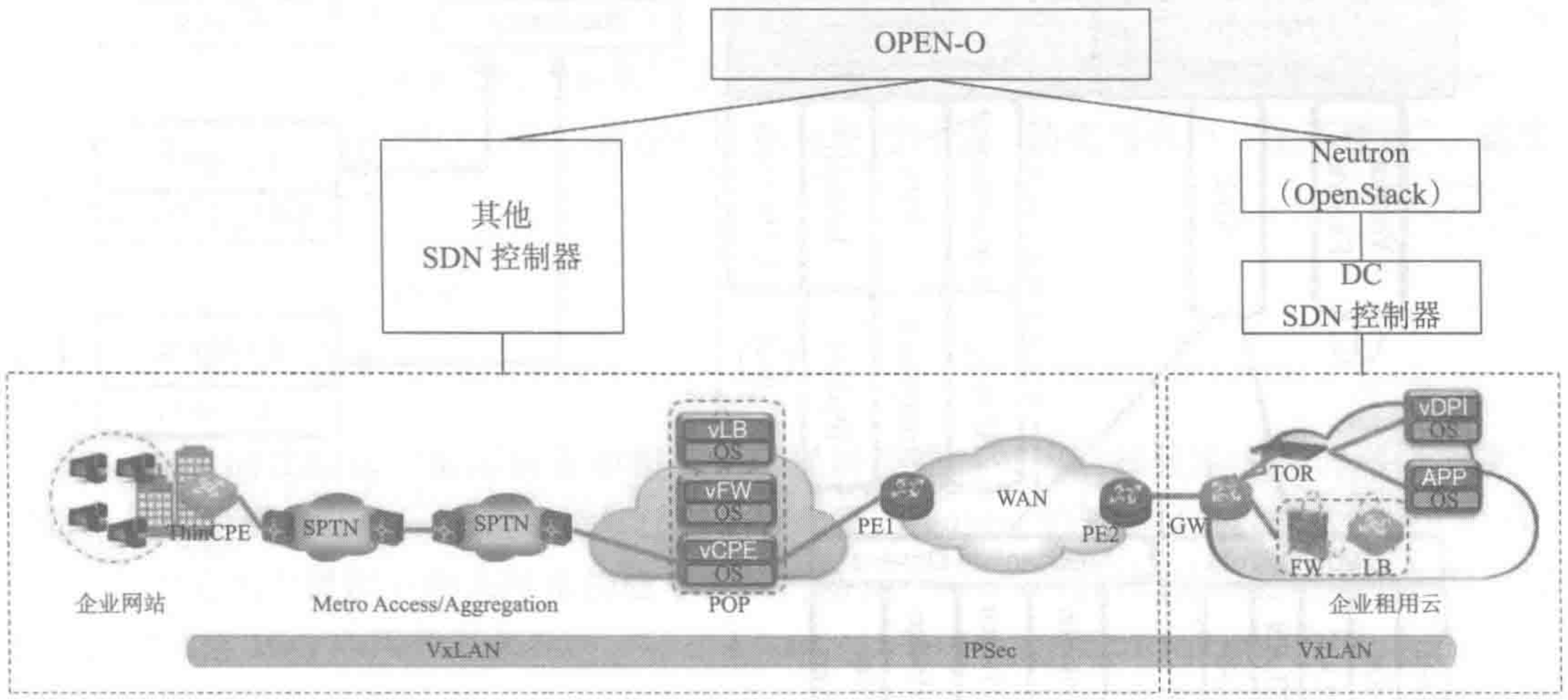


图 1-8　OPEN-O Enterprise 2 DC 场景解决方案

图 1-8 描述的是一个 Enterprise 2 DC 的场景，图左边的组网及“其他 SDN 控制器”与本文主题无关，我们先忽略。图右边是一个 DC（数据中心），OPEN-O 向下看到的是 Neutron