“十二五”
国家重点图书出版规划项目

学术中国 · 院士系列
ACADEMIC CHINA

软件定义网络
核心原理与应用实践
（第二版）下册

黄韬 刘江 魏亮 张娇 杨帆 刘韵洁 著

SDN Core Principles and Application Practice
（2nd Edition）Volume 2

中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

“十二五”
国家重点图书出版规划项目

学术中国 · 院士系列

软件定义网络
核心原理与应用实践

（第二版）下册

■ 黄韬 刘江 魏亮 张娇 杨帆 刘韵洁 著

SDN Core Principles and Application Practice

（2nd Edition）Volume 2

人民邮电出版社
北京

图书在版编目（CIP）数据

软件定义网络核心原理与应用实践. 下册 / 黄韬等著. -- 2版. -- 北京 : 人民邮电出版社, 2016.9
（学术中国. 院士系列）
ISBN 978-7-115-43088-5

Ⅰ. ①软… Ⅱ. ①黄… Ⅲ. ①计算机网络－研究
Ⅳ. ①TP393

中国版本图书馆CIP数据核字(2016)第186090号

内 容 提 要

本系列丛书包含核心原理和应用实践上下两册，对软件定义网络（SDN）技术进行了全面剖析和深入解读。本书属于应用实践部分，首先介绍了虚拟交换机 Open vSwitch 和网络仿真工具 Mininet，其次介绍了开源控制器 NOX/POX、Ryu、Floodlight、OpenDaylight 以及 ONOS，接下来介绍了网络虚拟化工具 FlowVisor 和 OpenVirtex，最后介绍了实验测试工具的基本知识。本书结合模拟网络环境搭建、虚拟网络设备部署、实际业务开发等具体应用实践场景，深入讲解了利用 SDN 技术进行创新研发的过程。

本书突出实用性，深入浅出地讲解了 SDN 的核心软件和相关应用开发过程，对从事 SDN 技术研发的专业人士、网络运营管理人员、相关专业的高校学生以及对 SDN 技术感兴趣的读者，都具有一定的参考价值。

◆ 著　　黄 韬 刘 江 魏 亮 张 娇 杨 帆 刘韵洁
责任编辑　代晓丽
责任印制　彭志环

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路 11 号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
固安县铭成印刷有限公司印刷

◆ 开本：700×1000　1/16
印张：18.5　　2016 年 9 月第 2 版
字数：363 千字　　2016 年 9 月河北第 1 次印刷

定价：98.00 元

读者服务热线：(010)81055488　印装质量热线：(010)81055316
反盗版热线：(010)81055315

前 言

软件定义网络（Software Defined Networking，SDN）正在成为行业注目的焦点，越来越多的业界专家相信它将推动传统网络的发展与变革。但是，究竟什么是 SDN？为什么需要 SDN？它将对现有网络架构产生何种影响？这些问题的背后，是大家对网络技术发展的思考与期许。

当前正在运行的互联网体系架构已经有超过 40 年的历史，随着网络规模的急剧膨胀和业务类型的不断丰富，互联网的结构和功能也日趋复杂，网络管控难度日渐增加，网络新功能难以快速部署。这促使人们重新思考网络体系架构的设计，而此时 SDN 的提出与兴起为未来网络的发展提供了一个新的可行方向。

SDN 之所以是一种革新性的技术，是因为它打破了传统网络架构的设计理念，一方面实现了控制平面与数据平面相分离；另一方面开放了网络可编程能力，从而提高了网络的灵活性和可管控性。此外，SDN 网络运营建立在开放软件的基础上，可以显著降低业务部署和维护成本。

本系列丛书期望对当前 SDN 的概念定义、核心原理、关键技术和部署应用几方面进行较为全面的介绍，具体包含核心原理和应用实践上下两册。本册属于应用实践部分，重点介绍了 SDN 应用实践相关的软件，并分享 SDN 的实践经验，期望通过实践案例使读者对 SDN 有一个更加深入的认识。具体章节内容介绍如下。

第 1 章和第 2 章分别讲解了开源 SDN 交换机 Open vSwitch 和 SDN 仿真工具 Mininet 的功能原理、代码解析以及安装配置等相关内容，并通过具体实践案例使读者可以参考这两章内容快速构建 SDN 仿真环境。

第 3～9 章介绍了目前广泛使用的 SDN 开源控制器 NOX/POX、Ryu、Floodlight、OpenDaylight 以及 ONOS。对各控制器进行了代码解读和分析，并通过模拟网络环境搭建、虚拟网络设备部署、实际业务研发等具体应用场景，介绍了 SDN 开源控制器的基本使用与开发方法。

第 10 章和第 11 章分别介绍了网络虚拟化工具 FlowVisor 和 OpenVirteX。从

代码、模块、结构及其安装使用方法对 FlowVisor 和 OpenVirteX 进行解析，并列举了相关实例应用的例子加深读者对这两个网络虚拟化工具的理解。

第 12 章介绍了 5 个常用的扩展实验工具，分别是功能测试工具 OFTest、性能测试工具 Cbench、拓扑生成工具 VND、报文分析工具 Wireshark 以及流量监控工具 sFlow，读者通过掌握这些工具能更好地研究 SDN 技术。

为便于读者检索，本书在附录中给出了 SDN 相关缩略语、名词索引以及有关 SDN 资源网址列表。

本书参与撰写和审校的人员包括：北京邮电大学的汪硕、张晨、李呈、丁健、王健、胡文博、俞淑妍、张健男、谢俊峰、肖海洋、杨潇、顾莹、于洁、张丽、刘娟、张歌、胡晓露、张然、辛远铭、吴理炫、耿直、李婕妤、吴畏虹、邹贵今、王彬、尹弼柏，以及江苏省未来网络研究院的方辉、冀烨、周洁、魏静波、邓晓涛、周洪利、张欣慰、陈俊霞、张小雅等。在此对大家表示衷心感谢。

感谢相关企业给予的大力支持与帮助，特别感谢中国联通研究院张云勇副院长、房秉毅博士，H3C 公司的孙晖部长、翟传璞部长，盛科公司孙剑勇总经理、张卫峰总监，Pica8 公司杨勇涛经理在本书编写过程中提出的诸多宝贵建议。

最后，感谢人民邮电出版社的大力支持和高效工作，使本书能尽早与读者见面。

本书内容是作者所在团队科研过程中一些实际经验的总结，希望能够对读者有所帮助。由于作者水平所限，同时 SDN 技术仍处于快速发展之中，因此书中难免会存在不少疏漏，真诚地期盼读者批评指正。

作　者

2016 年 5 月

目 录

第1章

Open vSwitch 应用实践

数据平面交换设备是 SDN 系统的重要组成部分之一，目前，SDN 的基础设施层可采用商用硬件交换设备，也可使用软件实现的交换设备。二者的主要差异体现在转发性能上，基于软件实现的交换设备能否大规模应用于商用系统成为很多用户普遍担心的问题。然而，随着通用处理器性能的提升，软件交换设备的转发性能也逐步增强，因此业界也开始积极尝试采用基于软件实现的可编程网络设备来构建软件定义网络。

Open vSwitch（OVS）[1]作为一款开源的 OpenFlow 软件交换机，基于与平台无关的 C 代码实现，容易移植到其他环境，既能够作为虚拟机管理平台的软件交换机[2,3]，也可以作为交换芯片的控制堆栈[4]，因此得到了广泛关注。本章将详细地对 OVS 的核心原理及使用方法进行介绍，并给出应用 OVS 进行 SDN 组网的实例，使读者对 OVS 有一个较为全面的认识，为日后使用打下基础。

1.1 OVS 系统架构

OVS 功能全面可以实现大规模网络的自动化配置、管理、维护，且支持大量现有标准管理接口和协议，利用其作为 SDN 的基础设施层转发设备，可大幅降低部署成本，还可以提高网络的灵活性以及可扩展性[5,6]。

OVS 架构分为内核空间、用户空间、配置管理层 3 个部分[5]，如图 1-1 所示。其中，内核空间包含了流表（Flow Table）和一个或多个 Datapath 模块，其中 Datapath 模块类似于网桥，主要负责对数据分组进行操作[7]，另外，内核空间中维护的流表规定了针对数据分组应该进行的操作，Datapath 通过关联流表与其协同完成分组处理过程。用户空间中运行着 OVS 的守护进程（Open vSwitch Daemon，

vswitchd）和数据库（Open vSwitch Database，ovsdb），它们是 OVS 的核心功能模块，vswitchd 类似于 OVS 的心脏，它维持着 OVS 的生命周期，而 ovsdb 就像 OVS 的大脑，它存储着 OVS 的配置信息和数据流信息，vswitchd 和 ovsdb 协调工作确保 OVS 健康的运行状态。配置管理层包括 ovs-dpctl、ovs-ofctl、ovs-appctl、ovs-vsctl 和 ovsdb-tool 等，主要用于和 vswitchd、ovsdb 之间进行交互操作以及 OVS 的安装配置和部署。

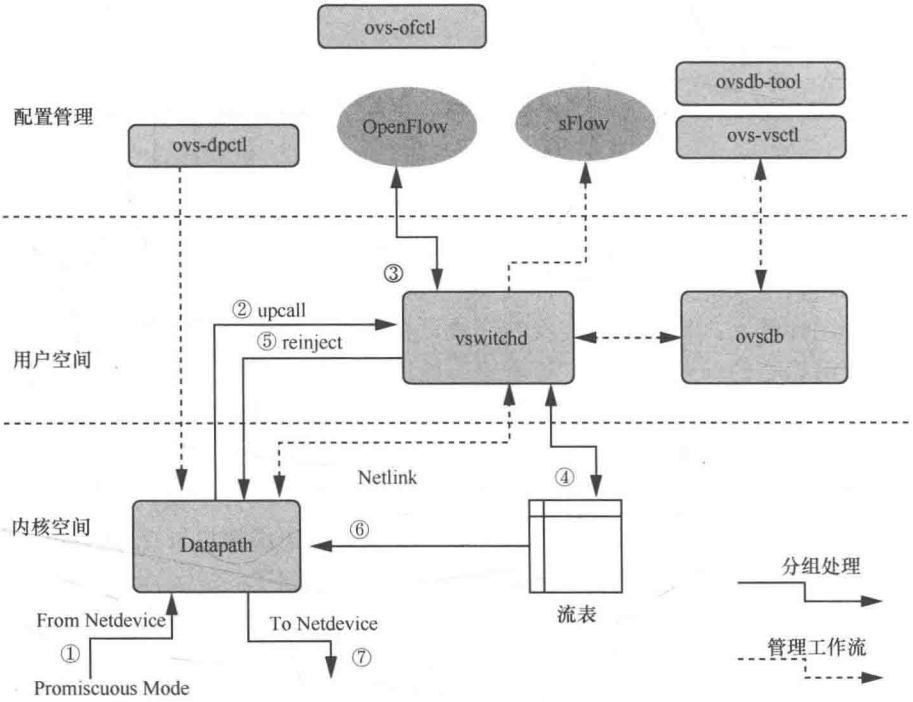


图 1-1　OVS 系统架构

下面对 OVS 架构中内核空间、用户空间以及配置管理工具 3 个部分的各个模块进行简要介绍。

（1）OVS 内核空间（包含 Datapath 和流表）

- Datapath：主要负责实际的数据分组处理，它同时与 vswitchd 和流表保持关联，使 OVS 上层可以对数据分组处理进行控制。
- 流表：流表中存储着分组处理的依据——流表项，它指导 Datapath 做出正确的分组处理判断，同时还与 vswitchd 上下关联，是 OVS 上层对底层分组处理过程进行管理的接口。

（2）OVS 用户空间（包含 vswitchd 和 ovsdb）

- vswitchd：OVS 的守护进程，属于核心模块。负责检索和更新数据库信息，

并根据数据库中的配置信息维护和管理 OVS。vswitchd 可以配置一系列特性：基于 MAC 地址学习的二层交换、支持 IEEE 802.1Q VLAN、端口镜像、sFlow 监测、连接 OpenFlow 控制器等。vswitchd 也可以通过 Netlink 协议与内核模块 Datapath 直接通信。

- ovsdb：一个轻量级的 OVS 数据库，用于管理 OVS 的配置信息，主要负责保存 OVS 配置信息和数据流信息等。轻量级数据库服务器 ovsdb-server 直接管理 ovsdb，与 ovsdb 通信进行数据库的增、删、改、查操作。同时负责向 vswitchd 提供操作 ovsdb 的能力。vswitchd 可以通过 UNIX Socket 机制与 ovsdb-server 进行通信，用于查询和更新数据库信息，或者在检索数据库信息后做出首个分组（First Packet）的转发策略。

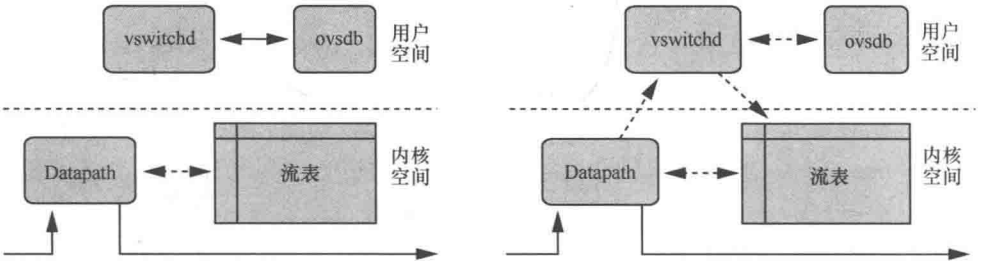
（3）OVS 管理工具（包含 ovs-dpctl、ovs-ofctl、ovs-appctl、ovs-vsctl）

- ovs-dpctl：管理 OVS Datapath 的实用工具，用来配置交换机内核模块，控制数据分组的转发规则。用户使用该工具可以创建、修改和删除 Datapath。
- ovs-ofctl：OF 交换机的命令行管理工具，用于管理和配置 OVS 作为 OF 交换机时的各种参数。用户使用该工具可以查询或修改 OF 交换机的状态、配置和流表项等信息。
- ovs-appctl：用于向运行时的 OVS 守护进程发送命令的工具。
- ovs-vsctl：查询和配置 OVS 数据库的实用工具，用于查询或者变更 vswitchd 的配置信息，该工具会直接更新 ovsdb 数据库。

作为数据转发设备，OVS 的最主要功能是数据分组处理功能，它对数据分组的处理过程视情况可分为以下两个步骤：第一步是由内核空间的 Datapath 尝试直接对数据分组进行转发操作；第二步是由用户空间和内核空间协同工作进行分组处理。以一个最简单的过程为例，当数据分组到达 OVS 后，首先将数据分组头信息与 OVS 内核空间中流表的表项进行匹配，查找与之对应的表项。如果查找到匹配的表项则根据表项中的规定进行分组处理，如果流表中没有匹配表项则需要由用户空间完成整个分组处理过程，也就是说，到达 OVS 的数据分组仅在第一步处理不了的情况下才会进行到第二步处理流程。

OVS 对数据分组处理的第一步流程如图 1-2（a）所示，当数据分组到达内核空间时，内核模块提取数据分组关键信息查找 Flow Table 存储的流表项并转发数据分组。OVS 的内核模块支持多个 Datapath，每个 Datapath 可以有多个 vport（为便于理解可将 Datapath 类比为普通的网桥，将 vport 类比为网桥下的端口），当数据分组到达 Datapath 时，Datapath 通过匹配流表的表项，按照匹配成功的表项中指定的规则进行数据分组转发，将数据分组转发到另一个 vport。当数据分组到达 vport 时，内核模块再一次提取数据分组的关键信息并查询流表，若存在匹配的流表项，则执行流表项中的对应操作。

如果流表中查找不到与数据分组对应的表项，OVS 就无法在第一步中完成分组处理，OVS 将执行第二步操作，第二步的分组处理流程如图 1-2（b）所示。当数据分组到达 OVS Datapath 时，Flow Table 中没有可匹配的流表项，导致无法处理该数据分组，从而将数据分组送到用户空间的处理队列进行决策。在用户空间中，vswitchd 检索 ovsdb 获取数据分组的相关信息，制定数据分组的转发策略并通知内核模块。同时，vswitchd 会设置流表项用于后续相同类型数据分组的处理。当第一个到达的数据分组成功转发后，内核模块更新流表项，这样后续接收的该类数据分组都执行同样的转发动作，避免每个数据分组都交给用户空间处理而降低效率。



(a) 第一步OVS内核空间尝试直接处理数据分组

(b) 第二步OVS用户空间和内核空间共同处理数据分组（在第一步处理不了的情况下才进行此步骤）

图 1-2　OVS 中数据分组的转发机制

以上两步分组处理步骤的显著差别在于 OVS 的用户空间是否参与分组处理过程，其本质在于若当前流表过期了或者流表未匹配，则需要由第二步来更新流表中的表项，使流表内容始终处于最新版本。相较之下，同时由内核空间和用户空间参与的第二步分组处理速率明显慢于仅有内核空间参与的第一步分组处理速率，但是需要指出的是这种经由用户空间带来的额外分组处理时延代价仅存在于数据流的第一个分组的处理过程中，因为由第二步完成第一个分组的处理后，流表得到了更新，该数据流此后到达的数据分组便可以在第一步中完成分组处理。

1.2　OVS 代码解读

1.2.1　代码结构

OVS 系统由多个组件构成，核心代码结构如图 1-3 所示。

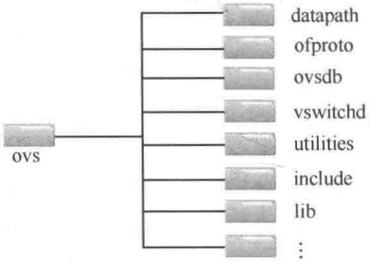


图 1-3　OVS 核心代码结构

其中，datapath 目录是 OVS 的内核模块，包括 VLAN、Tunnel、Datapath 等相关的 C 文件以及 vport 管理文件，可实现普通以太网交换机，具有处理 VLAN、网络访问控制、基于流的控制等功能；ofproto 目录是存储 OpenFlow 协议相关文件；ovsdb 目录是数据库相关代码；vswitchd 目录保存 OVS 守护进程的源代码；utilities 目录包括一些工具代码，如 ofctl、vsctl、appctl 等；include 目录包含常用的头文件；lib 目录则保存所有公共模块的实现，如 OpenFlow 协议解析、JSON 数据解析、ovsdb 数据解析以及日志模块等。

1.2.2　代码解析

OVS 由多个模块构成，若详解每个模块则篇幅过长。下面以 vswitchd 和 ovsdb 两个核心模块为例进行详细介绍，其余模块读者可以按此方法逐个解析。

1.2.2.1　守护进程（vswitchd）

前面已经介绍 OVS 守护进程（vswitchd）的主要功能，在此将主要介绍 OVS 守护进程的内部模块组成、核心流程以及 OpenFlow 协议的数据结构。

（1）内部模块组成

守护进程 vswitchd 进程的模块组成如图 1-4 所示。

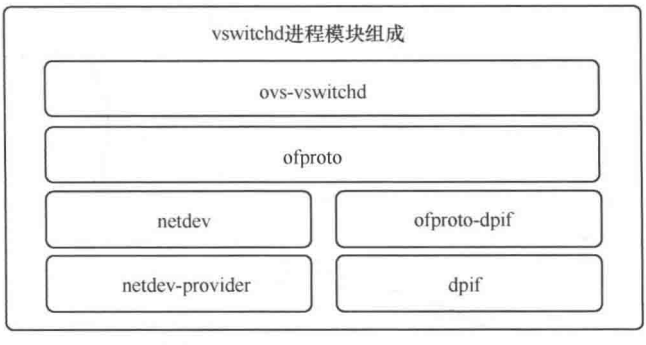


图 1-4　vswitchd 进程模块架构

其中，ovs-vswitchd 模块负责和 ovsdb-server 交互，启动时从数据库获取配置信息，并将配置信息下发给 ofproto 模块。当配置数据库发生变化时，vswitchd 也会实时感知并同步更新。ofproto 模块是一个处理 OpenFlow 协议的库，负责与控制器交互，并通过调用 dpif 等下层组件和交换机的硬件及软件交互。而 ofproto-dpif、dpif 是实现 ofproto 具体功能的下层支撑库，用来和底层的硬件和软件转发模块对接。netdev 是一个网络设备适配层，对网络设备的硬件接口和逻辑接口进行各种操作，如获取接口信息或者设置接口状态等，netdev 对网络设备的具体操作由 netdev-provider 提供。

由于 vswitchd 运行于用户态（用户空间），而 Datapath 运行于内核态（内核空间）。当 Datapath 无法匹配到流表项的报文时，会送给 vswitchd 的 ofproto-dpif 模块进行模糊匹配转发；如果还是无法匹配到，则由 ofproto 产生 Packet-in 事件到控制器，由控制器生成合适的流表项并下发给内核中的流表，然后进行相应的转发操作。

（2）vswitchd 的核心流程

vswitchd 是用户空间的守护进程，其核心是执行 ofproto 的逻辑，即创建网桥，并处理 OF 消息和 Netlink 消息。OVS 是遵从 OpenFlow 交换机规范实现的，以二层分组转发为例，传统交换机（包括 Linux 网桥的实现）通过查找 CAM（内容可寻址存储器）表，找到目的 MAC 地址对应的端口进行数据分组转发；而 OVS 的实现则是根据数据分组的 SKB（Socket Buffer，存储报文内容的缓存）查找是否有对应的流表项。如果有，说明这个 SKB 不是流的第一个分组，那么可以在流表项条目的动作表（Action）里找到转发的端口。这里要说明的是，SDN 的思想就是所有分组都需要对应一个流表项，基于流表项给出分组的 Action，传统的 Action 无非就是转发、接收或者丢弃，而在 SDN 中，Action 包含更多的定义，诸如修改 SKB 的内容、改变分组的路径、克隆多份并发到不同路径等。

如果 SKB 没有对应的流表项，说明这是流表项的第一个分组，需要为这个分组创建一个新的流表项，vswitchd 会在一个循环里反复检查是否有 ofproto 的请求，有可能是 ovs-ofctl 传过来的，也可能是 Datapath 通过 Netlink 发送的 upcall 请求，大部分情况下，都是 Flow Miss（指内核匹配流表项失败）导致的创建流表项的请求，这时 vswitchd 会基于 OpenFlow 规范创建流表项和 Action。vswitchd 守护进程启动流程如图 1-5 所示。

vswitchd 启动时，首先会从 ovsdb 中获取网桥信息，并将网桥在内核中运行起来。接下来会初始化 ofproto 库，并开始运行 ofproto，接收并处理来自网桥的信息。最后运行 ofproto 连接管理，接收并处理 OF 消息。OF 消息主要来自控制器和 ofctl 工具。来自控制器的 OF 消息，主要是指导流转发的流变更消息，而来自 ofctl 工具的消息主要是用于测试、监控以及查看等，如添加自定义调试的流表项、查看 OVS 中的流表信息等。