第5章 软件定义网络集成技术

本章学习目标

- 了解软件定义网络的概念、历史、分类及产业生态系统；
- 熟悉 OpenFlow 通信协议和控制器 OpenDaylight；
- 掌握软件定义网络的工作原理和整体架构。

构成网络的核心是交换机、路由器以及诸多的网络中间盒。而这些设备的制造规范大多为 Cisco、Broadcom 等通信厂商所垄断，并不具有开放性与扩展性。因此，长期以来，网络设备的硬件规范和软件规范都十分闭塞。尤其是对于路由协议等标准的支持，用户并没有主导权。对于新的网络控制协议的支持，需要通过用户与厂商沟通之后，经过长期的生产线流程，才能形成最终可用的产品。尽管对于网络的自动化管理，已有 SNMP 等规范化的协议来定义，但这些网络管理协议并不能直接对网络设备的行为，尤其是路由转发策略等进行控制。

为了能够更快速地改变网络的性能，软件定义网络的理念便应运而生。

5.1 SDN 概述

软件定义网络(Software Defined Network，SDN)，是由美国斯坦福大学 Clean Slate 研究组提出的新型网络创新架构。SDN 将网络分为控制层(Control Plane)和数据层(Data Plane)。控制层的控制器软件，通过特定传输通道，统一下达命令给数据层设备，数据层设备仅依靠控制层的命令转发数据包。

5.1.1 SDN 介绍

SDN 的核心理念是希望应用软件可以参与对网络的控制管理，满足上层的业务需求，并通过自动化业务部署简化网络的运维。

传统网络设备紧耦合的网络架构被拆分成应用、控制、转发三层分离的架构。控制功能被转移到了服务器，上层应用、底层转发设施被抽象成多个逻辑实体。

1. SDN 的工作原理

SDN 的核心思想就是控制和转发分离。

众所周知，网络的作用就是连接。通过无数的结点(例如路由器、交换机)，将数据从起点传送到终点，这就是网络的基本功能。数据传输过程中，各结点不断接收和转发数据包。

控制负责下命令,转发负责传输。然而,考虑到安全冗余等因素,现实中的网络绝对不会是一条直线那么简单,它会是一个复杂的拓扑结构。于是,命令该怎么下,直接决定了网络的效率。传统网络中,各个转发结点都是独立工作的,内部管理命令和接口也是厂商私有的,不对外开放。每个结点,都在说各自的"方言",我们可以把它理解为"各自为战"的模式。虽然"战略层面"的规划和设计可能是统一的,但"战术层面"的执行却是复杂且低效的。

而SDN,就是在网络之上建立了一个SDN控制器结点,统一管理和控制下层设备的数据转发。所有的下级结点,管理功能被剥离(交给了SDN控制器),只剩下转发功能。

SDN控制下的网络,变得更加简单。管理者只需要像配置软件一样,进行简单部署,就可以让网络实现新的路由转发策略,如果是传统网络,每个网络设备都需要单独配置。

除了简化部署之外,SDN更深层次的意义,是赋予了网络的"可编程性"。也就是说,控制和转发分离之后,借助规范化的API,用户可以通过编写软件的方式,对网络进行管理。整个网络,就像个完整的机器人一样可供驱使。

2. SDN的整体架构

SDN的整体架构分为三层,从上到下分别是应用平面、控制平面和转发平面,如图5-1所示。其中,转发平面由交换机等网络通用硬件组成,各个网络设备之间通过不同规则形成的SDN数据通路连接;控制平面包含逻辑上为中心的SDN控制器,它掌握着全局网络信息,负责各种转发规则的控制;应用平面包含着各种基于SDN的网络应用,用户无须关心底层细节就可以编程、部署新应用。

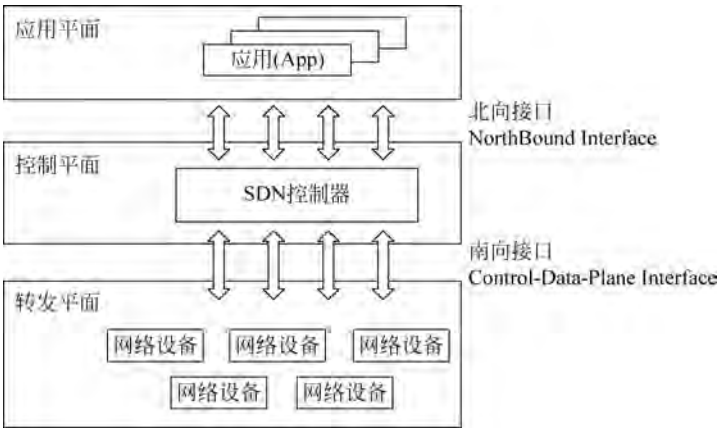

图5-1 SDN网络的整体架构

整个架构的核心,就是SDN控制器。SDN控制器向上与应用平面进行通信的接口,叫作北向接口,也叫NBI接口(NorthBound Interface),而NBI并无统一标准,它允许用户根据自身需求定制开发各种网络管理应用。SDN控制器向下与转发平面(也即数据平面)进行通信的接口,叫作南向接口,也叫CDPI(Control-Data-Plane Interface,控制数据平面接口),它具有统一的通信标准,主要负责将控制器中的转发规则下发至转发设备,最主要应用的是OpenFlow协议。

SDN中的接口具有开放性,以控制器为逻辑中心,南向接口负责与数据平面进行通信,北向接口负责与应用平面进行通信,东西向接口负责多控制器之间的通信。最主流的南向

接口 CDPI 采用的是 OpenFlow 协议。针对北向接口，应用程序通过北向接口编程来调用所需的各种网络资源，实现对网络的快速配置和部署。东西向接口使控制器具有可扩展性，为负载均衡和性能提升提供了技术保障。

上述 SDN 体系结构实现物理设施与功能的分离，带来如下几方面的技术优势。

(1) 开放网络创新能力：平面间标准化的数据面配置协议(即 OpenFlow 协议)实现控制面与数据面分离，实现廉价、水平可伸缩和开放的网络体系结构，替代传统昂贵、垂直集成和封闭的路由器体系结构，避免底层网络的复杂性，使得新的网络业务能够快速构建和测试。

(2) 网络控制可伸缩性、灵活性和可编程能力：SDN 域内数据面结点和域间结点通过集中全局控制避免传统动态路由控制系统所带来的局部性问题，如域内路由震荡、路由环路、路由黑洞、流量控制局部最优以及路由收敛过程路由结点上控制状态不一致等问题。

(3) 网络安全性与可管性：现有 BGP、OSPF、IS-IS 和 RIP 等协议的控制的稳定过程难以确保安全性，而 SDN 有助于避免现有网络中出现的 Sybil、RIB Poisoning 和 DoS 攻击等问题，并通过控制策略的实施实现端到端的资源管理。

(4) 数据面可扩展性：SDN 允许控制面通过对流表动态编程实现交换机转发行为的动态控制，现有相关研究表明，在 SDN 上扩展 IP 协议或新的非 IP 协议更容易。

(5) 网络测量感觉能力：OpenFlow 交换机流表项能够快速响应数据面流的高度动态性(通过在交换机中优化其中 Heavy-hitters 项)，通过准确测量，快速响应网络的异常问题和动态变化，提高网络控制系统实时性。

3. SDN 的分类

网络作为企业和运营商客户的基础设施，本质是上层业务的支撑系统，任何网络的技术或者架构变革都是为上层业务服务的。SDN 作为网络领域的技术热点也是一次技术革新，同样最终目的是服务于上层业务的需要。总结起来，当前业界对于 SDN 的认识，主要分为以下两种。

转发型 SDN：从架构上主要围绕转控分离展开，从业务角度主要围绕满足客户对于网络带宽的集中调控和提升带宽使用效率进行，因此将其归类为 Software Defined Network (forwarding)，即转发型 SDN。SDN 是一种集中控制的架构，与是否使用 OpenFlow 无必然关系。SDN 只是一种结构，主要特征为控制转发分离与集中式的控制，转控分离架构的具体实现多种多样，OpenFlow 只是其中一种。控制与转发分离不等价于网络的设备转发面与控制面彻底分离，并不意味着一定要使用 OpenFlow 这类新标准去改变传统的网络架构，相反，在保留设备的控制平面的同时，仍然可以在控制器上部署统一集中的控制面，通过传统的控制接口(如 BGP、静态路由等，而非 OpenFlow)，实现对于传统设备的集中控制，因此 OpenFlow 不是必要选择。例如，华为的 Agile TE 和 PCE+都属于此类产品。

业务型 SDN：本质是为了实现面向业务的网络快速自动化发放以及提升网络运营效率而设计，将其归类为 Software Defined Network(Provision/Policy)，简称为业务型 SDN。网络软件 Overlay 就是 SDN 业界软件 Overlay 技术的典型代表，是 Nicira 的软件网络虚拟化，Nicira 是一家专注于 SDN 与网络虚拟化的公司，由 OpenFlow 发明者 Martin Casado 创建，Nicira 创建了自己私有的 OpenFlow、Open vSwitch 等软件产品和技术。其 SDN 的核心是基于 OpenFlow 的控制转发分离+基于软件的网络虚拟化。原理是在传统的物理网络

上叠加一层虚拟网络,虚拟网络和底层物理网络解耦,虚拟网络集中控制。具体虚拟网络平台由虚拟交换机 Open vSwitch 和控制器组成。这个虚拟交换机是开源的,可以对任何人开放使用,虚拟交换机之间的连接由控制器进行管理。Nicara 将网络的控制从网络硬件中脱离出来,交给虚拟的网络层处理,实现了对物理网络的解耦,基于软件的虚拟化,物理网络被泛化为网络能力池。

无论是控制与转发分离,还是管理与控制分离,还是强调开放可编程,都不是 SDN 的本质定义,都是实现 SDN 的手段。为了实现 SDN 的核心诉求,转发型 SDN 和业务型 SDN 都有用武之地,但是总体来说,未来业务型 SDN 的发展空间将更大,更多地解决客户的问题,从而得到客户的青睐。SDN 起源于转发型 SDN,将大发展于业务型 SDN。

4. SDN 的实现方式及局限性

SDN 就是靠转控分离、集中控制、开放可编程这三个途径来颠覆传统网络。具体的实现有以下三个方案。

1) 基于开放协议的方案

此类协议方案根据 SDN 理念创建理想网络架构,能够真正意义、全方位地将控制层和转发层剥离,是最具有革命性意义的方案,能为用户摆脱厂商锁定而推出的方案,实现的方案包括 ONF SDN 和 ETSI NFV。当然,这种方案要求也更高,目前能够推出的厂商屈指可数,代表企业有华为、博科、戴尔和 NICIRA(已被 VMware 收购)。

2) 基于叠加网络的方案

这种方案通过在原有网络基础上创建虚拟网络隔离底层设备之间的不同和复杂性,从而实现网络资源池化。对已有的网络资源进行逻辑分离,并运用多租户的模式来管理网络,更好地满足大数据、云计算等新兴业务的需求。目前主要实现的方案包括 VXLAN、NVGRE、NVP 等,代表企业有 VMvare 和微软。

3) 基于专用接口的方案

这种方案的实现思路和以上两种不太一样,它不会改变传统网络的实现机制和工作方式,而是通过改动网络设备和操作系统,在网络设备上开发出专用的 API。管理人员可以通过 API 实现网络设备的统一配置管理和下发,替换了原先需要一台台设备登录配置的手工操作方式。同时,这些 API 也可供用户自主开发网络应用,将网络设备可编程化,这类方案由目前主流的网络设备厂商主导,应用最广。

尽管 SDN 在网络技术上的改变巨大,但不可否认的是,它自身仍存在着不小的缺陷。

(1) 标准化

虽然 SDN 自提出已经时隔多年,但仍旧没有一个统一的标准,各大厂家在细节上都有差异,很难对接。所以客户只能选择同一个厂家的控制器和硬件设备,造成的后果就是数据中心网络就必须一家网络厂商绑定,需要承担不小的风险。数据是非常重要的,一旦出现意外后果极其严重。在这种大环境下,SDN 部署的意愿不够强烈,很多人不愿意去试。这也是 SDN 始终不能做大的最主要原因。

(2) 安全性

这是个老生常谈的问题了,SDN 简化了操作层面,但像 Underlay(物理网络)、Overlay(控制转发)在部署和运行中同样会发生故障,排查起来也有不小的难度,更关键的是,SDN 不出问题还好,一旦出现故障影响的就是网络全局,造成的后果远比传统网络严重得多,只

能用灾难性来形容，尤其是核心网域。所以出于安全考虑，不少企业对 SDN 都心怀忌惮。

(3) 网络设备

SDN 是一种比较新颖的技术，它需要新式的网络设备支持。而现在网络设备五花八门，各种品牌，什么年代的都有。想要全部更换是一个很长的过程，保守估计也要十年时间，这样的网络环境不具备部署 SDN 的条件。毕竟现在网络规模已经基本成型，转型绝非易事，在这种情况下选择支持 SDN 的设备自然也要比传统网络设备价格要高得多，投入成本反而更高，这是厂家和客户都不能接收的。

综合以上种种原因，SDN 想要大规模普及并投入使用，还有很长的一段路要走。

5.1.2 SDN 历史

2006 年，美国斯坦福大学启动了一个名叫 Clean Slate 的研究课题。该课题由美国 GENI 项目资助，目的是“重塑互联网”。当时的互联网，已经历经了 30 多年的高速发展，从最初的小型专用局域网络，变成了空前庞大和复杂的世界级网络。网络规模的持续扩张，网络设备的不断增加，超过了早期设计的承受能力，也使得网络维护变得举步维艰。于是，专家们开始探讨未来网络的可能性架构，希望在互联网崩溃之前，将它拉回正轨。而 GENI 项目和 Clean Slate 课题，就是这些尝试之一。

2007 年，斯坦福大学博士生 Martin Casado 等提出了关于网络安全与管理的项目——Ethane。该项目试图通过一个集中式的控制器，将网络管理人员制定的安全控制策略，下发到各个网络设备中，从而实现对整个网络的安全控制。

2008 年，Clean Slate 课题的项目负责人，斯坦福大学教授 Nick McKeown 及其团队，受到 Ethane 项目的启发，提出了 OpenFlow 的概念，并发布了名为 *OpenFlow: Enabling Innovation in Campus Networks*(OpenFlow：校园网的创新使能)的论文。OpenFlow，字面意思就是“开放的流”。

2009 年，基于 OpenFlow，Nick McKeown 教授正式提出了 SDN(Software Defined Network，软件定义网络)。同年，SDN 概念成功入围 Technology Review 年度十大前沿技术，获得了行业的广泛关注和重视。12 月，OpenFlow 规范的 1.0 版本正式发布。这是首个可用于商业化产品的版本，具有里程碑意义。

在 SDN 被提出之后，第一个控制器平台是 NOX。它是一种单一集中式结构的控制器，南向接口采用的是 OpenFlow 协议。

2011 年，由 Google、Facebook、微软、雅虎、德国电信等公司共同发起成立了一个对 SDN 影响深远的组织，那就是 ONF(Open Networking Foundation，开放网络基金会)。

早在 SDN 被提出之前，Google 就在寻找提升自身网络效率的方法。当看到 SDN 之后，Google 确认，这就是他们想要的。于是，它们果断决定将 SDN 应用于自己的数据网络。

2010 年，Google 开始将数据中心与数据中心之间的网络连线(G-scale)，转换成 SDN 架构。整个改造分为三个阶段。到了 2012 年，整个 Google B4 网络完全切换到了 OpenFlow 网络。改造之后，Google B4 网络的链路带宽利用率提高了 3 倍以上，接近 100%。这样的结果毫无疑问是令人震撼的，也坚定了行业对 SDN 的信心。

2013 年，Google 在 SIGCOMM 上发表了论文 *B4: Experience with a Globally-Deployed Software Defined WAN*，详细介绍了 Google 的 WAN 加速 SDN 方案。论文中

提及,Google使用的控制器名叫ONIX。

2013年4月8日,在Linux基金会的支持下,作为网络设备商中的领导者,Cisco与IBM、微软等公司一起,发起成立了开源组织OpenDaylight,共同开发SDN控制器。ODL的发起公司有IBM、微软、Big Switch、博科、思杰、戴尔、爱立信、富士通、英特尔、Juniper、NEC、惠普、红帽和VMware等,基本都是设备厂商。

OpenDaylight提出,SDN不等于OpenFlow,人们需要对SDN进行"重新定义"。也就是说,OpenDaylight强调SDN控制器不仅局限于OpenFlow,而是应该支持多种南向协议。同时,OpenDaylight还强调,应该用分布式的控制平台,取代单实例的控制器。这样可以管理更大的网络,提供更强劲的性能,还能增强系统的安全性和可靠性。

OpenDaylight成立之后,成员数量增长迅速。Cisco作为OpenDaylight项目的牵头人,主导了其中大部分项目的开发,推出了OpFlex。Big Switch推出Big Network Controller以及对应的开源版本Floodlight。Juniper推出的是Contrial以及对应的开源版本OpenContrial。总而言之,这一时期各种各样的SDN控制器处于百家争鸣的状态,发展势头一片大好。OpenDaylight也成了行业里最具影响力的技术组织之一。

2014年12月5日,ON. Lab推出了一款创新性的网络操作系统——ONOS(Open Network Operating System),对OpenDaylight发起了强有力的挑战。ONOS直接将自身定位提升到网络操作系统层面。ON. Lab全名是Open Networking Lab(开放网络实验室),最初是由Parulkar和Nick McKeown共同成立的。On. Lab的某些职能和ONF很类似。2016年10月19日,两个组织宣布正式合并,组成了新的ONF。

就这样,围绕SDN控制器和协议,各大流派及厂商进行了十多年的明争暗斗,并最终形成了现在的局面。

从趋势来看,网络操作系统的概念深入人心,是大势所趋。SDN控制器作为网络操作系统的核心,重要性不言而喻。

未来,随着网络规模的扩大,SDN控制器肯定会继续往分布式的方向发展。控制器之间的分工协作会更加深入,甚至可能出现集群。控制器也会引入NFV虚拟化技术,与OpenStack等云平台进行整合。

5.1.3 SDN产业生态系统

目前,SDN的产业生态系统已初现雏形,基本形成了芯片提供商、设备和解决方案提供商、互联网企业和运营商三大产业角色。

1. 芯片提供商

伴随着网络形态的变化,交换设备厂商一方面对芯片尺寸、系统成本、功耗等方面的要求不断提升,另一方面,它们也从定制化、大型模块化交换芯片转向了采用通用的商用交换芯片,通过外形尺寸固定的交换机实现数据中心的部署。目前可以提供支持SDN设备的芯片厂商包括盛科、博通、英特尔等。

(1) 盛科。盛科在最新发布的第3代GreatBelt系列以太网芯片中,推出N-FlowTM技术,以支持SDN的更多高级应用,为OpenFlow技术的大规模商用创造条件。N-FlowTM技术引入了基于TCAM的模糊匹配和基于哈希的精确匹配相结合的流表模式,从而在保持灵活性、低功耗、低成本的前提下大大增加流表的数目。

(2) 博通。博通也宣布推出实现 SDN 的芯片解决方案,其 StrataXGS 芯片是业界首款支持 VXLAN 虚拟化的芯片。基于智能虚拟化技术,可以线速转发支持云网络基础设施的虚拟化,支持 NVGRE、VXLAN 等网络虚拟化技术;基于智能缓冲器技术,可以实现基于负载均衡的动态数据分组,提升对突发流量的处理性能,提升缓存利用效率;基于智能哈希技术,在业务量繁重及业务量分布多种多样的树形网络中,消除极化和负载失衡问题,提高网络性能和可视性。芯片支持 VXLAN 的交换和网关功能,可提供多用户网络的扩展能力,使基于软件的逻辑网络可以按需创建,通过在物理服务器上增加虚拟机规模来保证用户所需的服务质量。

2. 设备和解决方案提供商——传统设备制造商

传统的网络设备制造商,以思科、瞻博、华为、阿朗等为代表。思科目前有 3 个 SDN 控制器的应用程序,分别是 Network Slicing、Network Tapping 和 Custom Forwarding。Network Slicing 可依据控制器所示的拓扑图,逻辑性地部署网络资源;Network Tapping 提供网络流量监控、分析及除错功能,这和 SDN 控制器厂商 Big Switch 推出的 Big Tap 应用程序提供的功能类似;Custom Forwarding 则可让网络管理者根据不同的参数来制定特别的转发规则。思科也公布了将会支持 onePK API 或 OpenFlow 代理程序的硬体平台。

3. 设备和解决方案提供商——创新公司

比较典型的创新公司包括 Nicira、Big Switch、Peca8 等,它们有可能打破传统网络设备巨头的垄断地位。

1) Nicira

Nicira 推出的产品是 NVP,其思路是在现有的 IP 网络上建立交叠的二层虚拟网络。在具体实现中,NVP 利用 Open vSwitch 虚拟交换机为各个数据流建立虚拟的网络连接。只要满足 IP 网络可达的条件,它就可以在现有的 IP 网络的基础上实现完全隔离的多租户网络环境,而无须升级或更换任何网络设备。基于 NVP 网络虚拟化方案,二层逻辑网络将不再受到传统的物理限制,属于同一个二层逻辑网络的结点之间可以跨越三层网络进行通信,有力地支撑了“大二层网络”的概念,因此在云数据中心领域拥有非常广阔的应用前景。

2) Big Switch

Big Switch 提出了 Open SDN 的架构。该架构主要包括三个层次:基于标准的南向协议、开放的核心控制器及北向开放的 API。其中,架构的核心是名为 Floodlight 的控制器,它已经获得了广泛的认可,拥有业界最大的 SDN 控制器开发社区支持。Floodlight 具有的模块化结构使得其功能易于扩展和增强,它既能够支持以 Open vSwitch 为代表的虚拟交换机,又能够支持众多 OpenFlow 物理交换机,并且可以对由 OpenFlow 交换机和非 OpenFlow 交换机组成的混合网络提供支持。

Big Switch 除了控制器之外,还提供网络应用平台 Big Network Controller 及运行在其上的网络虚拟化应用 Big Virtual Switch,统一网络监控应用 Big Tap 等产品。

4. 互联网企业和运营商

(1) 互联网企业。由于 SDN 在数据中心网络虚拟化、流量优化和管理方面具有天然的优势,全球与数据中心相关的 ISP、ICP 企业都对 SDN 充满期待,部分巨头已经开始在自身的内部网络中部署 SDN。

(2) 运营商。SDN 在数据中心、融合接入、网络管理等应用领域有望给网络运营商带

来诸多好处,因此 SDN 自诞生以来一直受到来自网络运营企业的高度关注。

5.2 OpenFlow 规范

OpenFlow 是 SDN 最重要的实现方案,分别由 OpenFlow 控制器和交换机构成,之间通过标准化 OpenFlow(OF)协议通信;其中,控制器以集中方式控制本管理域内的多个交换机,通过提供北向接口来实现业务逻辑,支撑多种网络业务创新研发,诸如流量控制、路由控制、安全控制等功能。现有 SDN/OF 多纳入虚拟化技术,如以 OpenStack 为首的云计算平台也已广泛采用了 SDN 技术为业务提供网络支撑环境,Nicira 以虚拟化技术为基础构建第一个 SDN 操作系统。

5.2.1 OpenFlow 概述

OpenFlow 是在 2008 年 3 月由 Nick Mckeown 等提出并在斯坦福大学成立了 OpenFlow 论坛,它是 SDN 的一个实例,是第一个遵循 SDN 架构的协议。

1. OpenFlow 组件

OpenFlow 网络由 OpenFlow 网络设备(OpenFlow 交换机)、控制器(OpenFlow 控制器)、用于连接设备和控制器的安全通道(Secure Channel)以及 OpenFlow 表项组成。其中,OpenFlow 交换机设备和 OpenFlow 控制器是组成 OpenFlow 网络的实体,要求能够支持安全信道和 OpenFlow 表项。OpenFlow 结构如图 5-2 所示。

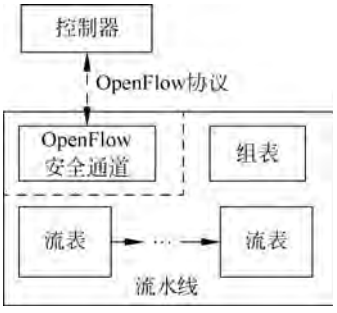

图 5-2 OpenFlow 结构

1) OpenFlow 控制器

OpenFlow 控制器位于 SDN 架构中的控制层,通过 OpenFlow 协议南向指导设备的转发。目前主流的 OpenFlow 控制器分为两大类:开源控制器和厂商开发的商用控制器。较为知名的开源控制器有:NOX/POX、ONOS、OpenDaylight 等。

2) OpenFlow 交换机

OpenFlow 交换机由硬件平面上的 OpenFlow 表项和软件平面上的安全通道构成,OpenFlow 表项为 OpenFlow 的关键组成部分,由 Controller 下发来实现控制平面对转发平面的控制。

OpenFlow 交换机主要有下面两种。

(1) OpenFlow-Only Switch:仅支持 OpenFlow 转发。

(2) OpenFlow-Hybrid Switch:既支持 OpenFlow 转发,也支持普通二三层转发。

一个 OpenFlow 交换机可以有若干个 OpenFlow 实例,每个 OpenFlow 实例可以单独连接控制器,相当于一台独立的交换机,根据控制器下发的流表项指导流量转发。OpenFlow 实例使得一个 OpenFlow 交换机同时被多组控制器控制成为可能。

OpenFlow 交换机实际在转发过程中,依赖于 OpenFlow 表项,转发动作则是由交换机的 OpenFlow 接口完成。OpenFlow 接口有下面三类。

物理接口:比如交换机的以太网口等。可以作为匹配的入接口和出接口。

逻辑接口：如聚合接口、Tunnel 接口等。可以作为匹配的入接口和出接口。

保留接口：由转发动作定义的接口，实现 OpenFlow 转发功能。

2. OpenFlow 表项

OpenFlow 的表项在 V1.0 阶段，只有普通的单播表项，也即人们通常所说的 OpenFlow 流表。随着 OpenFlow 协议的发展，更多的 OpenFlow 表项被添加进来，如组表(Group Table)、计量表(Meter Table)等，以实现更多的转发特性以及 QoS 功能。

1) OpenFlow 流表

OpenFlow 最基本的特点是基于流(Flow)的概念来匹配转发规则，每个交换机都维护一个流表(Flow Table)，依据流表中的转发规则进行转发，而流表的建立、维护和下发都是由控制器完成的。

OpenFlow 使用了广义流的概念：符合某些可识别特征的所有数据包的集合。这些特征主要是数据包的头部字段，如 MAC 层源地址和目的地址、IP 层源地址与目的地址、VLAN 标签以及用户自动字段 TLV(Type Length Vaule)等。流是对网络数据包特征的高度抽象，流表代替路由表并提供更多功能，以流为单位来指定数据包的处理比以 IP 目的地址指定数据包的处理更具灵活性。

流表是交换机支持 OpenFlow 的最关键技术，它由多个流表项组成，如图 5-3 所示。每个流表项代表一个流，定义了匹配和处理数据包的规则，这些流表项都由控制器写入。每个流表项由三部分组成：特征域、指令集和计数器。特征域包含用来识别流的所有域，指令集指定了与该流表项匹配的数据包将被如何处理，计数器记录每个流表项的一些统计信息。

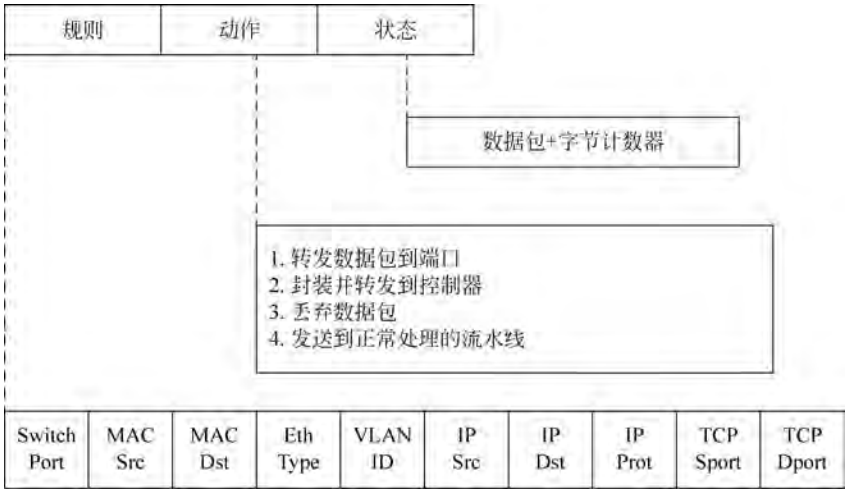

图 5-3　OpenFlow 流表示例

OpenFlow 交换机通过流水线来处理数据包，如图 5-4 所示。当数据包进入 OpenFlow 交换机后，随即进入数据包处理流水线，OpenFlow 交换机为每个数据包关联一个操作集(操作集中的操作等到数据包要退出流水线时执行)。数据包从第一个流表开始匹配，把被匹配流表中某条能满足该数据包所有域的最靠前的流表项称为命中表项。一般会把流表中最后一个表项配置为能匹配所有的数据包，其优先级最低。若没有配置此表项，未与所有流表项匹配成功的数据包将被丢弃。匹配完成后，命中表项中的指令会被执行：修改数据包，

把操作加入到操作集,通过把数据包发到后面的流表或组表来改变流水线处理。当命中表项中的指令没有指定下一个流表或组表,流水线处理就结束,此时,与数据包关联的集中操作会被执行。

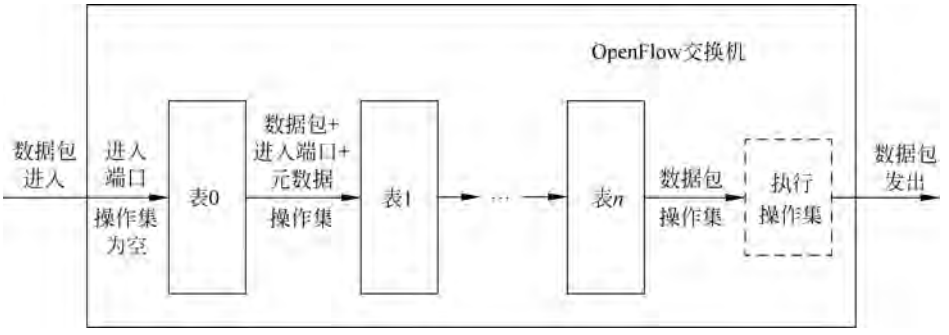

图 5-4　数据包的流水线处理过程

2) OpenFlow 组表

OpenFlow 组表的表项被流表项(Flow Entry)所引用,提供组播报文转发功能。一系列的 Group 表项组成了 Group Table。组表定义了比流表更加复杂的转发语法,组表由多个组表项组成,每个组表项由一系列操作桶组成,可以根据每个组表项中定义的组类型选择一个或多个操作桶执行,从而有效地实现多播或广播、高效的聚合、快速故障恢复等。

3) OpenFlow 计量表

Meter 计量表项被流表项(Flow Entry)所引用,为所有引用 Meter 表项的流表项提供报文限速的功能。一系列的 Meter 表项组成了 Meter Table,一个 Meter 表项可以包含一个或者多个 Meter Bands,每个 Meter Band 定义了速率以及动作,报文的速率超过了某些 MeterBand,根据这些 MeterBand 中速率最大的那个定义的动作进行处理。

3. OpenFlow 安全通道

OpenFlow 安全通道是连接 OpenFlow 交换机与控制器的接口,通常由 TLS 加密,实现控制器对交换机的配置和管理。

OpenFlow 设备与 Controller 通过建立 OpenFlow 信道,进行 OpenFlow 消息交互,实现表项下发、查询以及状态上报等功能。通过 OpenFlow 信道的报文都是根据 OpenFlow 协议定义的,通常采用 TLS(Transport Layer Security)加密,但也支持简单的 TCP 直接传输。如果安全通道采用 TLS 连接加密,当交换机启动时,会尝试连接到控制器的 6633 TCP 端口(Openflow 端口通常默认建议设置为 6633)。双方通过交换证书进行认证。因此,在加密时,每个交换机至少需配置两个证书。

5.2.2　OpenFlow 通信协议

1. OpenFlow 信道建立

1) OpenFlow 消息类型

要了解 OpenFlow 信道的建立过程,首先需要了解 OpenFlow 协议目前支持的三种报文类型:控制器-交换机消息、异步消息、同步消息。

(1) Controller to Switch 消息。

由 Controller 发起、Switch 接收并处理的消息。这些消息主要用于 Controller 对 Switch 进行状态查询和修改配置等管理操作，可能不需要交换机响应，如图 5-5 所示。

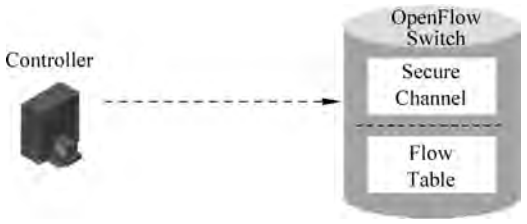

图 5-5　Controller to Switch

Controller to Switch 消息主要包含以下几种类型。

Features：用于控制器发送请求来了解交换机的性能，交换机必须回应该报文。

Modify-State：用于管理交换机的状态，如流表项和端口状态。该命令主要用于增加、删除、修改 OpenFlow 交换机内的流表表项、组表表项以及交换机端口的属性。

Read-State：用于控制器收集交换机各方面的信息，例如当前配置、统计信息等。

Flow-Mod：Flow-Mod 消息用来添加、删除、修改 OpenFlow 交换机的流表信息。Flow-Mod 消息共有五种类型：ADD、DELETE、DELETE-STRICT、MODIFY、MODIFY-STRICT。

Packet-out：用于通过交换机特定端口发送报文，这些报文是通过 Packet-in 消息接收到的。通常 Packet-out 消息包含整个之前接收到的 Packet-in 消息所携带的报文或者 buffer ID(用于指示存储在交换机内的特定报文)。这个消息需要包含一个动作列表，当 OpenFlow 交换机收到该动作列表后会对 Packet-out 消息所携带的报文执行该动作列表。如果动作列表为空，Packet-out 消息所携带的报文将被 OpenFlow 交换机丢弃。

Asynchronous-Configuration：控制器使用该报文设定异步消息过滤器来接收其只希望接收到的异步消息报文，或者向 OpenFlow 交换机查询该过滤器。该消息通常用于 OpenFlow 交换机和多个控制器相连的情况。

(2) 异步(Asynchronous)消息。

由 Switch 发送给 Controller，用来通知 Switch 上发生的某些异步事件的消息，主要包括 Packet-in、Flow-Removed、Port-Status 和 Error 等。例如，当某一条规则因为超时而被删除时，Switch 将自动发送一条 Flow-Removed 消息通知 Controller，以方便 Controller 做出相应的操作，如重新设置相关规则等，如图 5-6 所示。

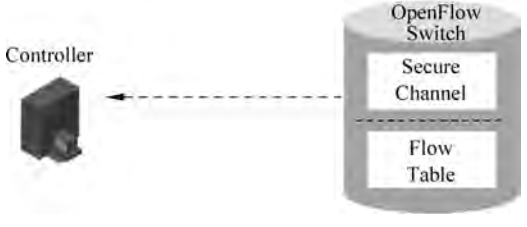

图 5-6　Switch to Controller

异步消息具体包含以下几种类型。

Packet-in：转移报文的控制权到控制器。对于所有通过匹配流表项或者 Table Miss 后转发到 Controller 端口的报文均要通过 Packet-in 消息送到 Controller。也有部分其他流程(如 TTL 检查等)也需要通过该消息和 Controller 交互。Packet-in 既可以携带整个需要转移控制权的报文，也可以通过在交换机内部设置报文的 Buffer 来仅携带报文头以及其 Buffer ID 传输给 Controller。Controller 在接收到 Packet-in 消息后会对其接收到的报文或者报文头和 Buffer ID 进行处理，并发回 Packet-out 消息通知 OpenFlow 交换机如何处理该报文。

Flow-Removed：通知控制器将某个流表项从流表的移除。通常该消息在控制器发送删除流表项的消息或者流表项的定时器超时后产生。

Port-Status：通知控制器端口状态或设置的改变。

(3) 同步(Symmetric)消息。

顾名思义，同步(Symmetric)消息是双向对称的消息，主要用来建立连接、检测对方是否在线等，是控制器和 OpenFlow 交换机都会在无请求情况下发送的消息，如图 5-7 所示，包括 Hello、Echo 和 Experimenter 三种消息，这里介绍应用中最常见的前两种。

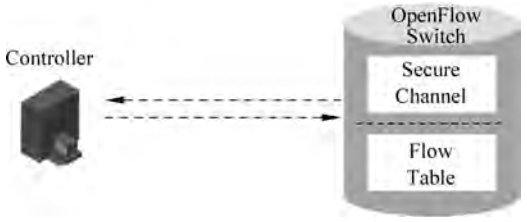

图 5-7　同步消息

Hello：当连接启动时交换机和控制器会发送 Hello 交互。

Echo：用于验证控制器与交换机之间连接的存活，控制器和 OpenFlow 交换机都会发送 Echo Request/Reply 消息。对于接收到的 Echo Request 消息必须能返回 Echo Reply 消息。Echo 消息也可用于测量控制器与交换机之间链路的延迟和带宽。

2) 信道建立过程解析

OpenFlow 控制器和 OpenFlow 交换机之间建立信道连接的基本过程如下。

(1) OpenFlow 交换机与 OpenFlow 控制器之间通过 TCP 三次握手过程建立连接，使用的 TCP 端口号为 6633。

(2) TCP 连接建立后，交换机和控制器就会互相发送 Hello 报文。Hello 报文负责在交换机和控制器之间进行版本协商，该报文中 OpenFlow 数据头的类型值为 0。

(3) 功能请求(Feature Request)：控制器发向交换机的一条 OpenFlow 消息，目的是为了获取交换机性能、功能以及一些系统参数。该报文中 OpenFlow 数据头的类型值为 5。

(4) 功能响应(Feature Reply)：由交换机向控制器发送的功能响应(Feature Reply)报文，描述了 OpenFlow 交换机的详细细节。控制器获得交换机功能信息后，OpenFlow 协议相关的特定操作就可以开始了。

(5) Echo 请求(Echo Request)和 Echo 响应(Echo Reply)属于 OpenFlow 中的对称型

报文，通常用于 OpenFlow 交换机和 OpenFlow 控制器之间的保活。通常 Echo 请求报文中 OpenFlow 数据头的类型值为 2，Echo 响应的类型值为 3。不同厂商提供的不同实现中，Echo 请求和响应报文中携带的信息也会有所不同。

3）信道连接断开模式

当 OpenFlow 设备与所有 Controller 断开连接后，设备进入 Fail Open 模式。OpenFlow 设备存在以下两种 Fail Open 模式。

Fail Secure mode 交换机：在该模式下的 OpenFlow 交换机，流表项继续生效，直到流表项超时删除。OpenFlow 交换机内的流表表项会正常老化。

Fail Standalone mode 交换机：所有报文都会通过保留端口 Normal 处理。即此时的 OpenFlow 交换机变成传统的以太网交换机。Fail Standalone mode 只适用于 OpenFlow-Hybrid 交换机。

安全通道也有两种模式，不同模式下安全通道重连的机制不同。

并行模式：并行模式下，Switch 允许同时与多个 Controller 建立连接，Switch 与每个 Controller 单独进行保活和重连，互相之间不影响。当且仅当 Switch 与所有 Controller 的连接断开后，Switch 才进入 Fail Open 状态。

串行模式：串行模式下，Switch 在同一时刻仅允许与一个 Controller 建立连接。一旦与该 Controller 连接断开后，Switch 并不会进入 Fail Open 状态，而是立即根据 Controller 的 ID 顺序依次尝试与 Controller 连接。如果与所有 Controller 都无法建立连接，则等待重连时间后，继续遍历 Controller 尝试建立连接。在三次尝试后，仍然没有成功建立连接，则 Switch 进入 Fail Open 状态。

2. OpenFlow 消息处理

1）OpenFlow 流表下发与初始流表

OpenFlow 流表下发分为主动和被动两种机制。

主动模式下，Controller 将自己收集的流表信息主动下发给网络设备，随后网络设备可以直接根据流表进行转发。

被动模式下，网络设备收到一个报文没有匹配的 FlowTable 记录时，会将该报文转发给 Controller，由后者进行决策该如何转发，并下发相应的流表。被动模式的好处是网络设备无须维护全部的流表，只有当实际的流量产生时才向 Controller 获取流表记录并存储，当老化定时器超时后可以删除相应的流表，因此可以大大节省交换机芯片空间。

在实际应用中，通常是主动模式与被动模式结合使用。

当 OpenFlow 交换机和 Controller 建立连接后，Controller 需要主动给 OpenFlow 交换机下发初始流表，否则进入 OpenFlow 交换机的报文查找不到流表项，就会做丢弃处理。这里的初始流表保证了 OpenFlow 的未知报文能够上送控制器。而后续正常业务报文的转发流表，则在实际流量产生时，由主动下发的初始流表将业务报文的首包上送给控制器后，触发控制器以被动模式下发。

这里以 H3C VCFC 控制器给交换机下发的一个初始流表举例。

前面了解到，OpenFlow 流表是分级匹配的，通常按 0 表、1 表、2 表这样依次匹配过去，每个级别的表中则由优先级高的表项先进行匹配。

如图 5-8 所示，0 表优先级最高为 65535 的两条流表匹配到的是端口号为 67、68 的

UDP报文,也就是DHCP报文,匹配动作为goto_table 1,剩下的其他所有报文也命中优先级最低为0的表项后goto_table 1。而在表1中,优先级最低的表项对应的动作为output controller,这保证了虚拟机的DHCP请求可以发送给控制器,由控制器作为网络中的DHCP Server,避免DHCP请求泛洪,同时还保证了交换机上所有未知的无流表匹配的报文都可以上送控制器,触发控制器被动下发流表给交换机指导转发。这里,把表1里优先级最低为0,匹配所有未知报文的表项叫作table-miss表项。

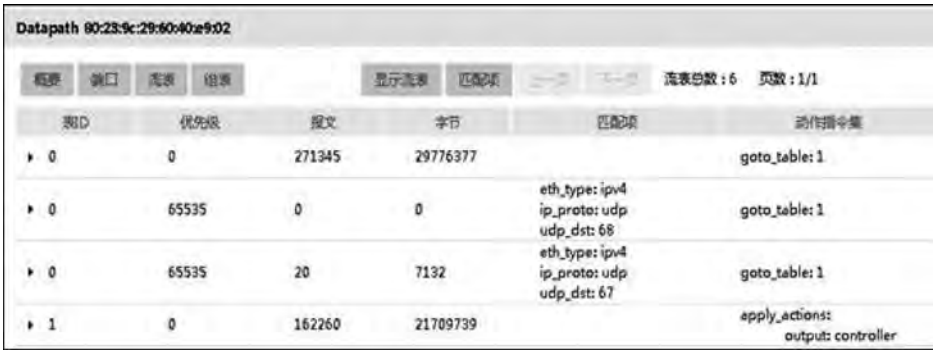

图5-8　OpenFlow流表举例

我们在OpenFlow交换机上同样可以观察到初始流表,这里以H3C S6800交换机上的一个初始流表举例。如图5-9中的这条表项匹配报文类型为以太网报文,UDP端口67、68说明匹配DHCP请求报文,动作为上送控制器。

```
<H3C>display openflow instance 1 flow-table
......
Flow entry 5 information:
 cookie: 0x2c324757415057, priority: 61000, hard time: 0, idle time: 0, flags:
 flow_send_rem, byte count: 0, packet count: 0
Match information:
 Ethernet type: 0x0800
 IP protocol: 17
 UDP source port: 67, mask: 0xffff
 UDP destination port: 68, mask: 0xffff
Instruction information:
 Write actions:
  Output interface: Controller, send length: 65509 bytes
......
```

图5-9　OpenFlow交换机上流表举例

2) OpenFlow报文上送控制器

OpenFlow报文上送控制器详细过程如图5-10所示。

(1) 控制器和交换机建立连接事件是Packet-in事件发生的前提。

(2) 当OpenFlow交换机收到数据包后,如果明细流表中与数据包没有任何匹配条目,就会命中table-miss表项,触发Packet-in事件,交换机会将这个数据包封装在OpenFlow协议报文中发送至控制器。

(3) 一旦交换机触发了Packet-in事件,Packet-in报文就将发送至控制器。

Packet-in数据头包括:缓冲ID、数据包长度和输入端口。

Packet-in的原因分为以下两种。

0:无匹配。

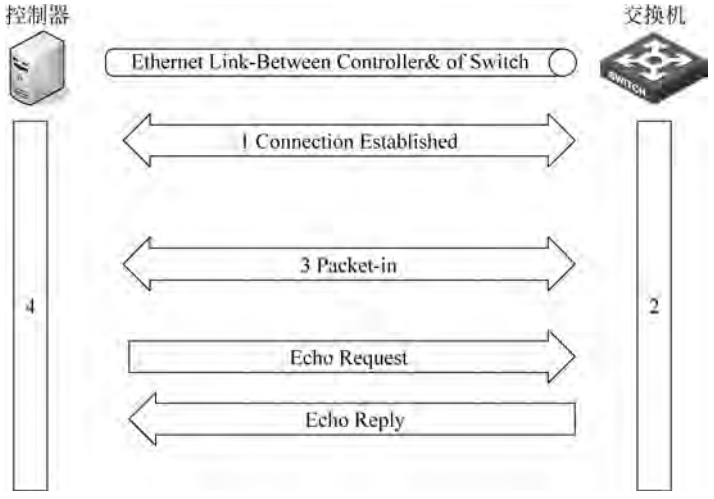

图 5-10　OpenFlow 报文上送控制器过程

1：流表中明确提到将数据包发送至控制器。

3）控制器回应 OpenFlow 报文

控制器收到 Packet-in 消息后，可以发送 Flow-Mod 消息向交换机写一个流表项。并且将 Flow-Mod 消息中的 buffer_id 字段设置为 Packet-in 消息中的 buffer_id 值，从而控制器向交换机写入了一条与数据包相关的流表项，并且指定该数据包按照此流表项的 action 列表处理。

Controller 根据报文的特征信息（如 IP、MAC 等）下发一条新的流表项到 OpenFlow 交换机或者做其他处理之后，下发 Packet-out 消息动作为 output 到 table，具体过程如图 5-11 所示。

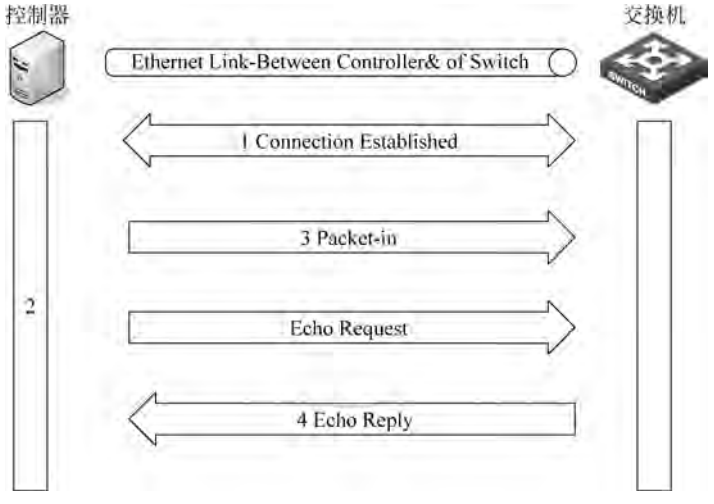

图 5-11　控制器回应 OpenFlow 报文过程

（1）控制器和交换机之间建立连接事件是 Packet-out 事件发生的前提。

（2）控制器要发送数据包至交换机时，就会触发 Packet-out 事件将数据包发送至交换机。这一事件的触发可以看作控制器主动通知交换机发送一些数据报文的操作。通常，当

控制器想对交换机的某一端口进行操作时,就会使用 Packet-out 报文。

(3) 该数据包由控制器发往交换机,内部信息使用 Packet-out,并由 OpenFlow 数据头封装。OpenFlow Packet-out 信息包括:缓冲 ID、入口端口编号、动作明细(添加为动作描述符)、输出动作描述符、VLAN VID 动作描述符、VLAN PCP 动作描述符、提取 VLAN 标签动作描述符、以太网地址动作描述符、IPv4 地址动作描述符、IPv4 DSCP 动作描述符、TCP/UDP 端口动作描述、队列动作描述符和各厂商动作描述符。

3. OpenFlow 交换机转发

1) 单播报文转发流程

当 OpenFlow 交换机接收到 Flow-Mod 消息,生成流表后,就可以按照流表转发接收到的 Packet-out 报文了,过程举例如图 5-12 所示。

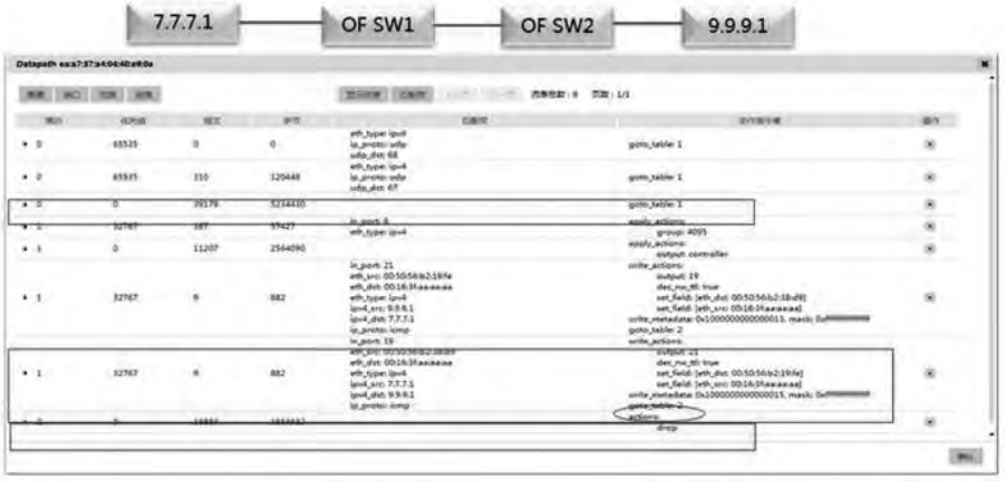

图 5-12　单播报文转发流表

在本例中,OpenFlow 交换机需要转发一个从 7.7.7.1 到 9.9.9.1 的流量。当流量上送到 OpenFlow 交换机后,流量的第一个包会先进行 Packet-in、Flow-Mod、Packet-out 的过程,之后同流量的报文就能匹配控制器已经下发的流表进行转发了。

2) 组播报文转发

当终端发出的组播报文到达 OpenFlow 交换机后,OpenFlow 交换机 Packet-in 给控制器,控制器会为网络下发指导查询组表的流表,并进行流表与组表关联。交换机参考流表,引用组表进行转发。举例如图 5-13 所示。

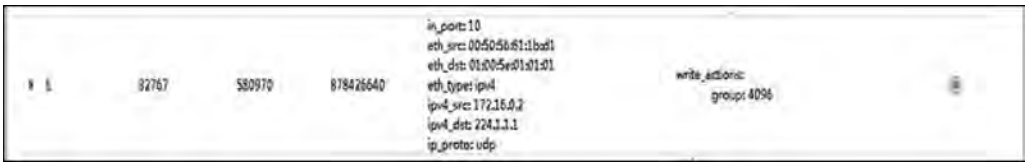

图 5-13　组播报文转发流表

上条流表的动作为引用组表 4096,组表 4096 详细内容如图 5-14 所示。

组表ID	类型	属性				操作
		权重	监控组	监控端口	动作	
4095	select	1	4294967295	2	output:2	⊗
		权重	监控组	监控端口	动作	
4096	all	0	4294967295	4294967295	output:9	⊗
		0	4294967295	4294967295	output:10	

图 5-14　组表详细内容

5.3　控制器与 OpenDaylight

传统的网络控制功能是由分布式设备实现的。SDN 实现了控制平面与数据平面的分离，将控制平面迁移到一个可控的计算设备之中，使得上层的网络服务和应用程序可以抽象和控制底层的网络设备，并最终通过开放可编程的软件模式实现网络的自动化控制。控制层主要负责集中维护网络拓扑和状态信息，针对用户需求定制数据传输模式并对数据平面的资源编排进行处理等。控制层通过控制平面和数据平面之间的南向接口获取底层网络设施信息，同时为应用层提供可扩展的北向接口。应用层调用控制层的北向接口以实现不同网络功能的应用程序。通过这种调用模式，网络管理者可以动态地配置、管理和优化底层的网络资源，实现灵活、可控的网络功能。

5.3.1　控制器概述

从第一个控制器平台 NOX 出现至今，已有了一系列基于 OpenFlow 的网络控制器平台。这些控制器平台在向下封装与交换机通信的 OpenFlow 协议的同时，也向上层网络控制应用提供相对更高层的开放编程接口。SDN 控制器的基本架构如图 5-15 所示。

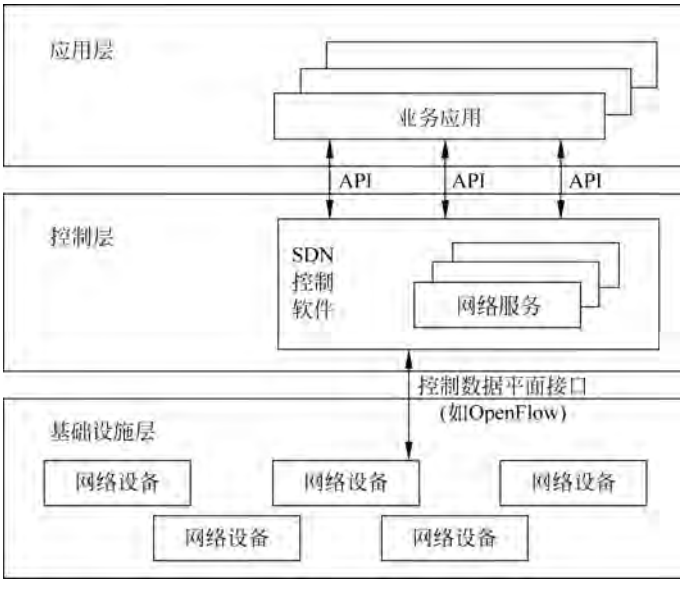

图 5-15　SDN 控制器基本架构

当前,SDN控制器已经比较成熟,种类也相当繁多,而且活跃的一些控制器项目还在不断发展之中,如OpenDaylight项目不到一年就发布一个新的版本。

SDN控制器分为开源控制器和商业控制器。有些商业控制器是在某个开源控制器的基础上优化和修改而来的,其中一些公司本身也是这个开源控制器的贡献成员之一。

1. OpenDaylight控制器

目前最具影响力、活跃度最高的控制器项目是OpenDaylight,有许多商业控制器是基于ODL改造生成的。OpenDaylight项目中的很多子项目已经在商用领域得到了部署,成效不断。

2. ONOS控制器

ONOS(Open Network Operating System,开放网络操作系统)是一款为服务提供商打造的基于集群的分布式SDN操作系统,具有可扩展性、高可用性、高性能以及南北向的抽象化,使得服务提供商能轻松地采用模块化结构来开发应用提供服务。

3. Floodlight控制器

Floodlight控制器是较早出现的知名度较广的开源SDN控制器之一,它实现了控制和查询一个OpenFlow网络的通用功能集,而在此控制器上的应用集则满足了不同用户对于网络所需的各种功能。

4. Ryu控制器

Ryu是一个基于组件的SDN网络框架,它是由日本NTT公司使用Python语言研发完成的开源软件,采用Apache License标准。Ryu提供了包含良好定义的API的网络组件,开发者使用这些API能轻松地创建新的网络管理和控制应用。Ryu支持管理网络设置的多种协议。

5. 思科公司的APIC控制器和OpenSDN控制器

思科公司的SDN控制器有两个:APIC控制器和OpenSDN控制器。思科的APIC控制器在商业上有着很大的影响力,在商业上到了很好的部署。OpenSDN控制器是一个OpenDaylight的商业级版本,通过基于网络基础设施标准的自动化来提供业务的灵活性。

6. OpenContrail控制器(Tungsten Fabric)

Juniper网络(瞻博网络)发布的OpenContrail项目包括OpenContrail控制器和OpenContrail虚拟路由。OpenContrail控制器是一个逻辑上集中,但是物理上分布的SDN控制器,为虚拟网络提供管理、控制和分析功能。OpenContrail虚拟路由是一个分布式的路由服务。

Tungsten Fabric曾用名OpenContrail,最初是由Juniper开源的一个可扩展的多云网络平台,拥有一个充满活力的开发者和最终用户社区。2018年3月完成向Linux基金会的迁移,并且正式更名为Tungsten Fabric。

7. NOX控制器

NOX控制器是由斯坦福大学在2008年提出的第一款Open Flow控制器,NOX控制器是第一个实现的SDN控制器,它的早期版本(NOX-Classic)是由C++和Python语言实现的,其中,NOX核心架构及其关键部分都是使用C++实现的。

8. POX控制器

POX控制器是由NOX控制器分割演变出来的一款基于Open Flow控制器,是使用

Python 语言开发的。POX 控制器具有将交换机送来的协议包交给制定软件模块的功能。

9. Beacon 控制器

Beacon 项目是基于 Java 语言开发实现的开源控制器，依赖于 OpenFlow 项目，以高效性和稳定性应用在多个科研项目实验环境中。除此之外，具有很好的跨平合性，并支持多线程，可以通过相对友好的 UI 界面进行访问控制、使用和部署。

10. Big Network 控制器

Big Network 项目是一款 SDN 商用控制器，由 Big Switch 网络公司推出。Big Switch 网络公司将此控制器放入 Open SDN 套件中，供数据中心运营商使用。

11. Brocade SDN 控制器

2015 年，博科推出基于 OpenDaylight 代码研发的 Brocade SDN 控制器（原名称为博科 Vyatta 控制器），新版本控制器基于 OpenDaylight 项目进行了优化，添加了两个管理应用，以加强提供对 SDN 操作的支持。Brocade SDN 控制器实际上就是 OpenDaylight 控制器的商用版。

12. Maestro 控制器

Maestro 是莱斯大学于 2011 年的一篇学位论文中提出的用 Java 语言实现的一款基于 LGPI V2.1 开源协议标准的 OpenFlow 多线程控制器。Maestro 主要应用于科研领域，具有很好的平台适应性，可以有效地在多种操作系统和体系结构上运行。

13. IRIS 控制器

IRIS 是由 ETRI 研究团队创建的递归式 SDN OpenFlow 控制器，OpenIRIS 是 IRS 的一个开源版本。IRIS 旨在解决 SDN 中可扩展性和可用性的问题。IRIS 是在 Beacon 控制器和 Floodlight 控制器的基础上构建的。

14. OneController 控制器

OneController 控制器是 Extreme 公司基于开源控制器 OpenDaylight 的 Helium SR 1.1 版本开发的。One Controller 控制器旨在提供一个开放、功能灵活加载或卸载、可拓展的平台，使得 SDN 和 NFV 的规则能达到任意规模大小。

5.3.2 OpenDaylight 控制器

2013 年，Linux 基金会联合思科、Juniper 和 Broadcom 等多家网络设备商创立了开源项目 OpenDaylight，它的发起者和赞助商多为设备厂商而非运营商等网络设备消费者。OpenDaylight 项目的发展目标在于推出一个通用的 SDN 控制平台和网络操作系统，从而管理不同的网络设备，正如 Linux 和 Windows 等操作系统可以在不同的底层设备上运行一样。OpenDaylight 支持多种南向协议，如 OpenFlow 和 BEG-LS 等，底层支持混合模式交换机和经典 OpenFlow 交换机。OpenDaylight 是一个广义的 SDN 控制平台，而不是仅支持 OpenFlow 的狭义 SDN 控制器。

OpenDaylight 的架构如图 5-16 所示，可分为南向接口层、控制平面层、北向接口层和网络应用层。

南向接口层中包含如 OpenFlow、NET-CONF 和 SNMP 等多种南向协议的实现。控制平面层是 OpenDaylight 的核心，包括 MD-SALI、基础的网络功能模块、网络服务和网络抽象等模块，其中，MD-SAL 是 OpenDaylight 最具特色的设计，也是 OpenDaylight 架构中

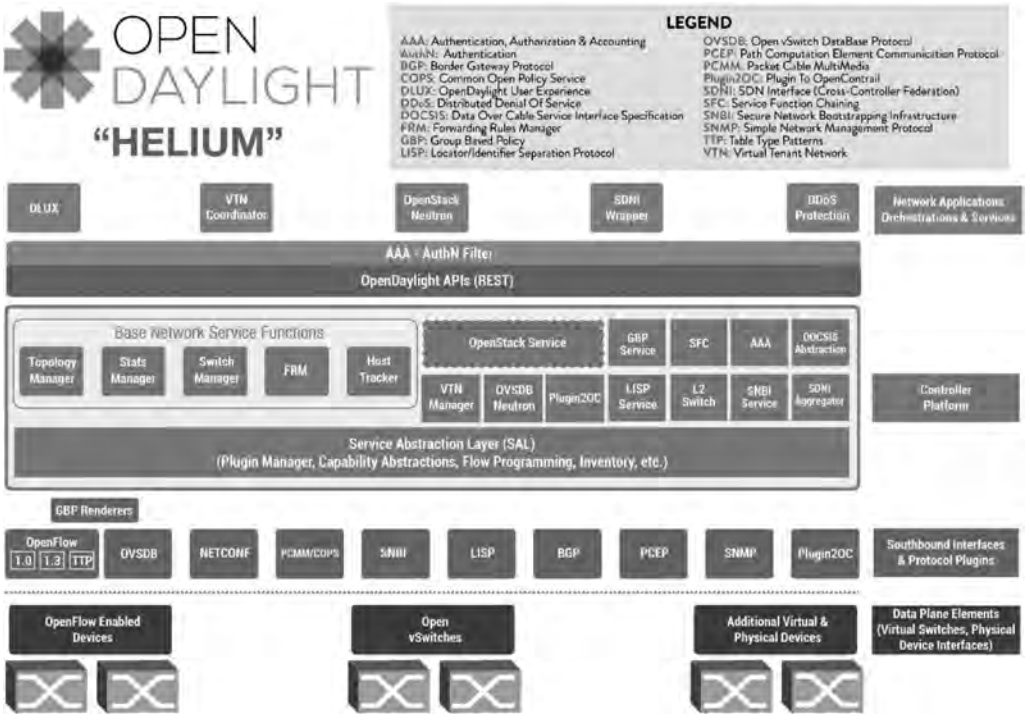

图 5-16　OpenDaylight 的架构

最重要的核心模块。无论是南向模块还是北向模块,或者其他模块,都需要在 MD-SAL 中注册才能正常工作。MD-SAL 也是逻辑上的信息容器,是 OpenDaylight 控制器的管理中心,负责数据存储、请求路由、消息的订阅和发布等内容。北向接口层包含开放的 REST API 及 AAA 认证部分。应用层是基于 OpenDaylight 北向接口层的接口所开发出的应用集合。

OpenDaylight 基于 Java 语言编写,采用 Maven 来构建模块项目代码。Maven 构建工程有许多好处,可以允许 OpenDaylight 对某些模块进行单独编译,使得在只修改某些模块代码时快速完成编译。为了实现 OpenDaylight 良好的拓展性,OpenDaylight 基于 OSGi (Open Service Gateway Initiative)框架运行,所有的模块均作为 OSGi 框架的 bundle 运行。OSGi 是一个 Java 框架,其中定义了应用程序即 bundle 的生命周期模式和服务注册等规范。OSGi 的优点是支持模块动态加载、卸载、启动和停止等行为,尤其适合需要热插拔的模块化大型项目。OpenDaylight 作为一个网络操作系统平台,基于 OSGi 框架开发可以实现灵活的模块加载和卸载等操作,而无须在对模块进行操作时重启整个控制器,在新版本中,其使用了 Kaaf 容器来运行项目。Kaaf 是 Apache 旗下的一个开源项目,是一个基于 oSGi 的运行环境,提供了一个轻量级的 oSGi 容器。基于 OpenDaylight 控制器开发模块时,还需要使用 YANG 语言来建模,然后使用 YANG Tools 生成对应的 Java API,并与其他 Maven 构建的插件代码共同完成服务实现。

特性方面,OpenDaylight 支持丰富的特性,而且在目前版本迭代中依然不断增加特性。南向协议支持方面,OpenDaylight 支持 OpenFlow、NET-CONF、SNMP 和 PCEP 等多种南向协议,所以 OpenDaylight 可以管理使用不同南向协议的网络。核心功能部分,

OpenDaylight 除了支持如拓扑发现等基础的控制器的功能以外，还支持许多新的服务，San VTN(Virtual Tenant Network)、ALTO(Application Layer Traffic Optimization)、DDoS 防御及 SDNi Wrapper 等服务和应用。值得一提的是，SDNi 是华为开发并提交给 IETF 的 SDN 域间通信的协议草案，目的是实现 SDN 控制器实例之间的信息交互。

此外，OpenDaylight 还正在大力开展 NFV 的研发。正如之前提到的，OpenDaylight 不仅是一个 SDN 控制器，OpenDaylight 是一个网络操作系统。除了 SDN 控制器的基础功能以外，还包括 NFV 等其他应用服务，可见其旨在打造一个通用的 SDN 操作系统。

5.4 SDN 应用案例

招商银行 SDN 实践之路介绍如下。

从 2013 年至今，招商银行(以下简称招行)陆续完成了数据中心服务能力成熟标准的制定，主机系统的异地容灾建设，能够实现业务在两分钟内“一键切换”，实现了所有应用系统的异地双活，以及部分数据库的异地双活，完成了 IT 基础架构的高可用性设计，实现重要业务系统的“双区接入”，全行所有业务系统、所有信息平台实现了统一管理，开展了招行金融云的建设，IaaS、PaaS 平台都正在逐步完善中，并且专门成立专有云。

由于招行的数据中心拥有庞大规模的 IT 基础设施，同时需要支撑种类繁多的开发测试项目，而在 Iaas 和 Saas 平台的建设过程中，网络的建设遇到了前所未有的挑战，具体来讲可以分为以下几点。

(1) 随着计算和存储的虚拟化进程不断加快，网络与业务尚处于紧耦合状态，这成为制约业务快速部署和上线的主要因素。

(2) 面对数据中心不同厂商、不同类型的网元，传统网络对于进行统一的自动化管理显得非常困难，例如，招行的数据中心有着不同厂商的不同设备，想要统一管理谈何容易？

(3) 银行对于数据流量和质量要求越来越高，而传统网络无法保障，例如，当某个业务变得缓慢，排查过程显得异常复杂，需要做大量的配置且容易出错。

因此金融云急需一套灵活、弹性、可扩展的网络架构来解决上述困难，提高业务部署的速度，对数据流量做一个精细化的控制，并且能够对接到云平台实现网络资源的统一管理。

SDN 的三大特性如集中化的管理、控制转发分离、开放的 API 正好可以弥补传统网络在云化过程中暴露出来的如上缺陷。

在抓住这些痛点需求之后，招行开始着手构建 SDN，选择实践部署的是招行位于深圳的科兴园区，该园区是一个拥有一千多人办公的场地，既有开发人员，又有测试人员，还有一部分是业务的数据中心。由于科兴园区主要承载的招行手机银行、网上银行产品的开发测试环境，它的特点是对于资源的分配和回收频率很快，网络的变化比生产环境更加频繁，这也是选择在此实践 SDN 的主要原因之一。

科兴园区网络的 SDN 化采取的是基于 Overlay 的方式实现，由于科兴园区的现有网络中存在虚拟化和非虚拟化的服务器，若采取彻底转发控制相分离的狭义 SDN 方案显然不现实。为了保证兼容已有的网络和其他 IT 设备，招行最终采取的是新华三提供的云平台、Overlay、SDN 的整体解决方案，该方案的特点如下。

(1) 采用业界通用的 Spine-Leaf 模型进行组网，其中，Spine 结点为 L3 网关，蓝色的

Leaf结点用于做VXLAN的VTEP的起点,而红色的Leaf只是用于接入非虚拟化的服务器。

(2) 通过SDN控制器,对Overlay进行集中部署管理,实现了IT资源的自动化分配和弹性扩展,提高了网络快速部署和灵活扩展能力,并且通过控制器集群来保证控制层面的高可靠性。

(3) 在物理网络之上构建了一层虚拟Overlay网络,各类IT资源统一接入Overlay网络。Overlay网络的部署,实现了服务器的接入位置、IP地址与物理网络的解耦合,实现了网络资源与物理网络设备的解耦合。

科兴园区的物理拓扑中使用静态路由跟传统网络对接,避免了传统网络和SDN之间的相互影响。其中,Spine结点使用的是H3C 125X,两台之间做IRF2,接入层用的H3C的SDN 68-1,这样可以保证物理网络这块没有环路,Spine和Leaf之间通过OSPF连接。但是对于SDN来说,如何保证其安全性对于金融行业来说是至关重要的。

在虚拟化的网络中如何实现安全防护?由于虚拟机漂移等原因,传统网络的那套安全策略实施起来会非常困难且低效,那么通过灵活自定义服务链的方式可以解决这一问题,具体来说就是将原来的从源到目的之间通过防火墙的形式,转变为通过定义服务链时将流指定到某个虚拟的防火墙,这里虚拟防火墙是通过NFV来提供的安全的服务结点。总结起来如下。

(1) NFV提供虚拟安全服务结点。

(2) VSwitch支持嵌入式安全。

(3) SDN控制器提供服务链的自定义和安全资源统一编排和控制。

虽然整个科兴园区用SDN的仅仅是开发测试网络,但是这仍旧给网络的运维管理成本带来了极大的降低,目前管理整个园区网络的只有三个人,这要是在以前是不可想象的。

小　　结

SDN的本质是一种新的网络管理和运营模式,面对网络规模扩张、网络管理动态化的挑战,传统以设备为中心、基于人工静态配置的模式已经不能适应发展的需要。SDN基于网络控制器将网络资源、网络管理和网络控制集中,形成网络服务能力,并且通过RESTful(一种互联网软件架构)化的机机接口将这些网络服务开放给部署在网络之上的IT业务系统。也就是希望代表业务的应用软件可以参与对网络的控制管理,满足上层业务需求,通过网络自动化部署,提升业务的敏捷性,提升网络运维效率,这是SDN的核心诉求。

本章首先介绍了SDN历史、SDN的工作原理、SDN的整体架构、SDN的分类、SDN产业生态系统及SDN应用案例。然后介绍了OpenFlow及其通信协议。最后概述了SDN控制器的种类并重点介绍了OpenDaylight控制器。

习题与实践

1. 填空题

(1) SDN是_____的缩写。

(2) SDN 的核心思想是：_____和_____分离。

(3) SDN 控制器的基本架构包括_____层、_____层和_____层。

(4) OpenFlow 是 SDN 最重要的实现方案，分别由_____和_____构成，之间通过标准化 OpenFlow(OF)协议通信。

(5) OpenDaylight 的架构可分为南向接口层、_____、_____和网络应用层。

2. 判断题

(1) SDN 的整体架构分为三层，从下到上分别是应用平面、控制平面和转发平面。(　　)

(2) OpenFlow 是 SDN 最重要的实现方案，分别由 OpenFlow 控制器和交换机构成，之间通过标准化 OpenFlow(OF)协议通信。(　　)

(3) OpenFlow 网络由 OpenFlow 网络设备(OpenFlow 交换机)、控制器(OpenFlow 控制器)、用于连接设备和控制器的安全通道(Secure Channel)以及 OpenFlow 表项组成。(　　)

(4) OpenFlow 是在 2008 年 3 月由 Nick McKeown 等提出并在斯坦福大学成立了 OpenFlow 论坛，它是 SDN 的一个实例，是第一个遵循 SDN 架构的协议。(　　)

(5) OpenFlow 安全通道是连接 OpenFlow 交换机与控制器的接口，通常由 TLS 加密，实现控制器对交换机的配置和管理。

(6) 为了实现 OpenDaylight 良好的拓展性，OpenDaylight 基于 OSGi(Open Service Gateway Initiative)框架运行，所有的模块均作为 OSGi 框架的 bundle 运行。(　　)

3. 简答题

(1) 什么是 DHCP？引入 DHCP 的好处有哪些？

(2) 如何实现 IP 地址与 MAC 地址的绑定？

(3) 什么是 DNS 域名系统？详细描述域名解析的过程。