第5章 软件定义网络

在第4章中已经提到基于控制与转发分离思想的可编程网络范式——软件定义网络(SDN)是近几年来受到广泛关注的一个技术理念。本章首先对SDN技术从定义、发展背景、整体架构方面进行介绍,然后分别对数据平面和控制平面的SDN关键技术进行讨论分析。

5.1 SDN技术简介

5.1.1 基本概念

对于当前的网络架构而言,SDN是一种新的实现方法,但它仍然属于一种网络概念。这里首先对相关的网络概念进行简单介绍,其中包括一些基本的传统网络概念与SDN网络模型下的相关名词。

1. OSI模型

OSI模型,即开放式通信系统互联参考模型(open system interconnection reference model,OSI/RM),是国际标准化组织(ISO)提出的一个试图使各种计算机在世界范围内互连为网络的标准框架,简称OSI。OSI将计算机网络体系结构(architecture)划分为以下7层:

(1) 物理层——将数据转换为可通过物理介质传送的电子信号。

(2) 数据链路层——决定访问网络介质的方式,在此层将数据分帧,并处理流控制。本层指定拓扑结构并提供硬件寻址。

(3) 网络层——设计数据路由经过大型网络。

(4) 传输层——提供终端到终端的可靠连接。

(5) 会话层——允许用户使用简单易记的名称建立连接。

(6) 表示层——协商数据交换格式。

(7) 应用层——用户的应用程序和网络之间的接口。

因此,这个模型允许网络工程师和网络供应商将技术应用于特定的OSI层次。这种分割方式可以让工程师把某一网络应用的整个问

题分解到不同层次以方便管理。每一层都有特定的属性描述，每一层与相邻的层之间的交互方式都已经明确定义。

2. 交换机与路由器

交换机又称为交换式集线器，可以简单地理解为把一些计算机连接在一起组成一个局域网。交换机就是用来进行报文交换的机器。多为链路层设备（二层交换机），能够进行地址学习，采用存储转发的形式来交换报文。交换机拥有一条很高带宽的背部总线和内部交换矩阵。交换机的所有端口都挂接在这条总线上，控制电路收到数据包以后，处理端口会查找内存中的地址对照表以确定目的 MAC（网卡的硬件地址）的 NIC（网卡）挂接在哪个端口上，通过内部交换矩阵迅速将数据包传送到目的端口，目的 MAC 若不存在则广播到所有的端口，接收端口回应后交换机会“学习”新的地址，并把它添加入内部 MAC 地址表中。

路由器又称网关设备，是用于连接多个逻辑上地分开的网络，逻辑网络代表一个单独的网络或者一个子网。当数据从一个子网传输到另一个子网时，可通过路由器的路由功能来完成。因此，路由器具有判断网络地址和选择 IP 路径的功能，它能在多网络互联环境中建立灵活的连接，可用完全不同的数据分组和介质访问方法连接各种子网，路由器只接受源站或其他路由器的信息，属网络层的一种互联设备，它的作用在于连接不同的网段并且找到网络中数据传输最合适的路径。

因此，交换机与路由器最明显的区别是工作层次不同，最初的交换机工作在数据链路层，而路由器设计在网络层。在 SDN 的网络概念中，它们都属于数据转发设备，了解交换机与路由器的基本概念，通过比较可以更好地理解 SDN 数据交换设备的工作方式。

3. 以太网

多个终端通过一个共享的基础设备连接到一起，形成一个广播域，主机之间使用 OSI 中第二层的介质访问控制（MAC）地址互相通信。

4. IP 地址和子网划分

IP 地址是指互联网协议地址，又译为网际协议地址。IP 地址是 IP 协议提供的一种统一的地址格式，它为互联网上的每一个网络和每一台主机分配一个逻辑地址，以此来屏蔽物理地址的差异。IP 地址是一个 32 位的二进制数，通常被分隔为 4 个“8 位二进制数”（也就是 4 个字节）。IP 地址通常用“点分十进制”表示成 a.b.c.d 的形式，其中，a、b、c、d 都是 0～255 之间的十进制整数。例如，点分十进 IP 地址 100.4.5.6 实际上是 32 位二进制数 01100100.00000100.00000101.00000110。

三类常用的 IP：

（1）A 类 IP 段——1.0.0.0～126.255.255.255（0 段和 127 段不使用）。

（2）B 类 IP 段——128.0.0.0～191.255.255.255。

（3）C 类 IP 段——192.0.0.0～223.255.255.255。

XP 默认分配的子网掩码每段只有 255 或 0：

（1）A 类的默认子网掩码——255.0.0.0（一个子网最多可以容纳 1677 万多台计

算机)。

(2) B 类的默认子网掩码——255.255.0.0(一个子网最多可以容纳 6 万台计算机)。

(3) C 类的默认子网掩码——255.255.255.0(一个子网最多可以容纳 254 台计算机)。

必须有一个子网掩码,因为当配置 IP 时,所有计算机都必须填写子网掩码;必须在网络中设置一些逻辑边界;必须至少输入所使用 IP 类的默认子网掩码。子网的划分,实际上就是设计子网掩码的过程。子网掩码主要是用来区分 IP 地址中的网络 ID 和主机 ID,它用来屏蔽 IP 地址的一部分,从 IP 地址中分离出网络 ID 和主机 ID。子网掩码是由 4 个十进制数组成的数值,中间用“.”分隔。

5. 传输控制协议和用户数据报协议

OSI 模型第 4 层中的协议定义了网络上主机之间通信的方法。传输控制协议(TCP)使用面向连接的方式通信,而用户数据报协议(UDP)使用无连接模式通信。TCP 还具有流量控制、动态调整窗口/缓存大小以及显式确认等优点。

6. 网际控制报文协议

ICMP 是 Internet 控制报文协议。它是 TCP/IP 协议族的一个子协议,用于在 IP 主机、路由器之间传递控制消息。控制消息是指网络通不通、主机是否可达、路由是否可用等网络本身的消息。这些控制消息虽然并不传输用户数据,但是对于用户数据的传递起着重要的作用。

它可以帮助网络工程师排除网络故障和调整网络运作,某些平台上,ICMP 是 Ping 通信协议和路由跟踪的核心,并且可以用于 IP 网络主机之间通告传输错误和其他信息。

7. 北向接口

北向接口(northbound interface)是提供给其他厂家或运营商进行接入和管理的接口,即向上提供的接口。网络管理者必须使用北向接口定义和开发应用层中的网络管理应用程序,这样才能通过这些应用程序接入和管理网络,通常这种应用都是以简单、易用且直观的界面形式呈现给操作者,操作者通过界面单击或配置发送命令,而系统内部则使用北向接口将这些命令发送给数据处理层,数据处理层中驻留有随时运行着的北向接口处理进程,此进程接收到应用层发送来的命令后,将控制命令转发给下一层——数据管理层,以便继续执行。而接收到的如果是请求报文,则将来自数据管理层的数据按北向接口规定的格式封装后返回给应用层。JSON 和 Thrift 就是典型的北向接口。

8. 南向接口

南向接口(southbound interface)是与北向接口对应的一个概念,指管理其他厂家网管或设备的接口,即向下提供的接口。提供对其他厂家网元的管理功能,支持多种形式的接口协议,包括 SNMP、TR069、SYSLOG、SOAP、SSH 等接口,以及 I2RS、NETCONF 或者某些命令行操作界面。

在 SDN 中,OpenFlow 为典型的南向接口。

9. 东西向接口

东西向接口指在SDN架构中，用于多个控制器之间沟通和联系的接口。SDN控制能力集中化，使得控制器的安全性和性能成为网络的最大限制，一旦控制器在性能或安全性上得不到保障，随之而来的是全网的服务能力的降级甚至是瘫痪。另外，单一的控制器也无法应对跨多个地域的SDN网络问题，需要多个SDN控制器组成的分布式集群，以避免单一的控制器节点在可靠性、扩展性、性能方面的问题。

10. 网络拓扑

网络拓扑是计算机网络各种网络组件（如链接、节点、接口、主机等）的分配布置方式。网络拓扑结构可以是物理上的，也可以是逻辑上的。物理拓扑结构是指网络的各种组件的位置，包括设备位置和线路的连接；而逻辑拓扑结构表示数据流在网络中的走向，与网络的实际物理结构无关。节点之间的物理连接方式和传输速率可能不同，两个网络之间的信号类型也可能不同，但是它们的拓扑结构却可能是相同的。

11. SDN应用程序

SDN应用程序（SDN application）更像是熟悉的网络上层功能，如QoS、路由功能、Overlay功能等。相比于传统网络，原本孤立的管理/配置如今被SDN统一化了，一个APP代表了整个SDN管理域内的所有此APP功能。例如网络出口要防DDOS攻击，调用了一个APP就能自动做黑洞引流操作，或者领导要开视频会议，调用一个QoS的APP就能全局做带宽质量保障，又如，通过SDN负载均衡APP，可以实现根据不同业务/参数进行负载轮询[1]。

12. 应用程序接口

应用程序接口（API）是一组软件组件之间交互的编程规范。实际上，API通常是一个包含变量规范、例程、对象类和数据结构的库。API规范有国际标准（如POSIX）、厂商技术文档（如JunOS SDK）或编程语言的库文件等多种形式。

5.1.2 SDN定义

随着SDN从学术界走到商业化，越来越多的开源项目出现，不同的标准化组织从各自的角度给出了相应的参考架构，同时各个运营商、网络公司、设备厂商均提出自己的SDN方案，因此至今业界对SDN的确切定义依然众说纷纭。

狭义上可以将SDN的定义等同于OpenFlow，OpenFlow起源于斯坦福大学的Clean Slate项目，该项目的最终目的是重新定义因特网架构，改变目前已经面临瓶颈且进化困难的网络基础架构，而OpenFlow协议将以太网的设计更一般化，将传统网络设备的数据转发（data plane）和路由控制（control plane）两个功能模块相分离，通过集中式的控制器（controller）以标准化的接口对各种网络设备进行管理和配置，为网络资源的设计、管理和使用提供更多的可能性，从而开创了SDN网络的革新与发展。从直观上来看，

OpenFlow 就是 SDN 概念在具体协议和技术上的体现形式。

从广义上，即网络架构层面定义 SDN，SDN 是一种数据控制分离、软件定义可编程的新型网络体系架构，开放网络基金会（open networking foundation，ONF）作为当前业界最活跃、规模最大的 SDN 标准组织，提出的 SDN 基本架构如图 5-1 所示。它具有集中式的控制平面和分布式的数据转发平面，两个平面互相分离，控制平面利用北向接口对转发平面上的网络设备进行集中式控制，并提供灵活的可编程能力，因此可以将广义 SDN 的特性概括为控制与转发分离、开放的编程接口、集中式的控制。

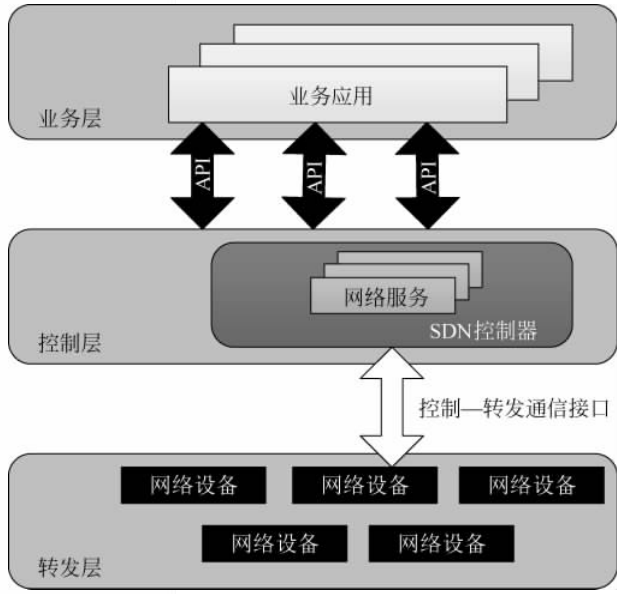

图 5-1　ONF 提出的 SDN 基本架构（引自参考文献[2]）

在 SDN 架构中，控制平面通过控制—转发通信接口对网络设备进行集中控制，这部分控制信令的流量发生在控制器与网络设备之间，独立于终端间通信产生的数据流量，网络设备通过接收控制信令生成转发表，并据此决定数据流量的处理，不再需要使用复杂的分布式网络协议来进行数据转发，如图 5-2 所示。

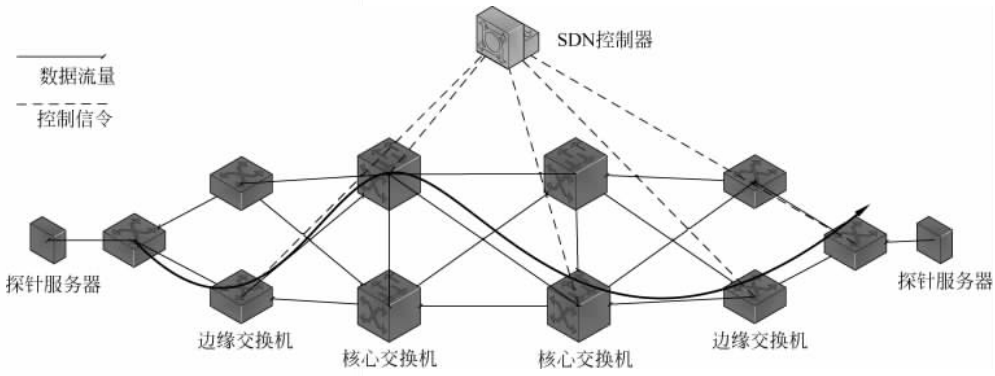

图 5-2　SDN 工作场景

(1) 关于解耦合控制层面与数据转发层面。现有网络中,对流量的控制和转发都依赖于网络设备实现,且设备中集成了与业务特性紧耦合的操作系统和专用硬件,这些操作系统和专用硬件都是各个厂家自己开发和设计的。而在 SDN 网络中,网络设备只负责单纯的数据转发,可以采用通用的硬件;而原来负责控制的操作系统将提炼为独立的网络操作系统,负责对不同业务特性进行适配,而且网络操作系统和业务特性以及硬件设备之间的通信都可以通过编程实现。

(2) 关于开放性的编程接口。SDN 体系架构通过对整个网络进行抽象,为用户提供完备的编程接口,使用户可以根据上层的业务与应用个性化地定制网络资源来满足其特有的需求。由于其开放可编程的特性,SDN 有可能打破某些厂商对设备、协议以及软件等方面的垄断,从而使更多人可以参与到网络技术的研发工作中来。

(3) 关于集中控制的概念。对于传统的设备,因为不同的硬件、供应商私有的软件,使得网络本身相对封闭,只能通过标准的互通协议与计算设备配合运行。网络中所有设备的自身系统都是相对孤立和分散的,网络控制分布在所有设备中,网络变更复杂、工作量大,并且因为设备异构,管理上兼容性很差,不同设备的功能与配置差异极大;同时网络功能的修改或演进,会涉及全网的升级与更新。而在 SDN 的开放架构下,一定范围内的网络(或称 SDN 域),由集中统一的控制逻辑单元来实施管理,由此解决了网络中大量设备分散独立运行管理的问题,使得网络的设计、部署、运维、管理在一个控制点完成,而底层网络差异性也因为解耦合的架构得到了消除。集中控制在网络中引入了 SDN 区别于传统网络架构的角色——SDN 控制器,也就是运行 SDN 网络操作系统并控制所有网络节点的控制单元。SDN 能够提供网络应用的接口,在此基础上按照业务需求进行软件设计与编程,并且是在 SDN 控制器上加载,从而使得全网迅速升级新的网络功能,而不必再对每个网元节点进行独立操作。

综上所述,目前业界对 SDN 概念较为普遍的定义主要基于控制与转发分离、开放的编程接口与逻辑上的集中控制。随着 SDN 应用场景的复杂化,不同组织对 SDN 研究侧重点的多样化,SDN 的定义在基于 ONF 的核心框架的基础上,也在不断完善补充的过程中。

5.1.3 SDN 的发展背景

通过 SDN 的来源可以知道,SDN 的初衷是改善目前网络遇到的瓶颈问题,然而,SDN 具体是怎么改善网络的? 与之前的网络改善途径有什么不同点? 下面通过分析 SDN 的发展背景,并结合目前几种典型的商业 SDN 解决方案分析 SDN 的发展背景与应用前景。

传统互联网的主要问题在于配置复杂度高、研发周期长、架构封闭等问题,这些问题说明网络架构需要革新,可编程网络的相关研究为 SDN 的产生提供了可参考的理论依据。主动网络允许数据包携带用户程序,并能够由网络设备自动执行。用户可以通过编程方式动态地配置网络,达到了方便管理网络的目的。然而由于需求低、协议兼容性差等问题,并未在工业界实际部署。

针对当前网络的逻辑决策平面和分布式的硬件设备结合过紧的问题，Greenberg 等重新设计了互联网控制和管理结构，提出了如图 5-3 所示的 4D(decision，dissemination，discovery，data，决策，传播，发现，数据)体系架构。在 4D 架构下，决策平面通过全局网络视图做出网络控制决策，并直接下发到数据平面；传播平面在决策平面和路由器之间建立可靠的通信通道；发现平面负责发现网络中的物理组件，并为决策平面提供构建网络视图的基本信息；数据平面则实现数据转发功能。这种架构有助于实现健壮、安全的网络，便于对异构网络进行有效管理[3]。这一架构将可编程的决策平面(即控制层)从数据平面分离，使控制平面逻辑中心化与自动化，其设计思想产生 SDN 控制器的雏形[4]。

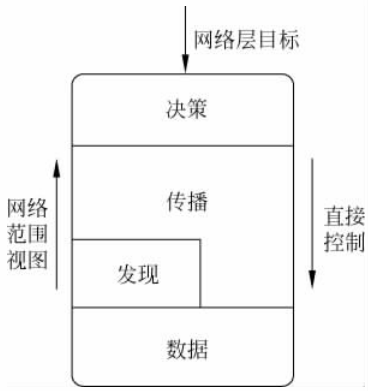

图 5-3　4D 架构

借鉴计算机系统的抽象结构，未来的网络结构将存在转发抽象、分布状态抽象和配置抽象这三类虚拟化概念[5]。转发抽象剥离了传统交换机的控制功能，将控制功能交由控制层来完成，并在数据层和控制层之间提供了标准接口，确保交换机完成识别转发数据的任务。控制层需要将设备的分布状态抽象成全网视图，以便众多应用能够通过全网信息进行网络的统一配置。配置抽象进一步简化了网络模型，用户仅需通过控制层提供的应用接口对网络进行简单配置，就可自动完成沿路径转发设备的统一部署。因此，网络抽象思想解耦了路径依赖，成为数据控制分离且接口统一架构(SDN)产生的决定因素。

SDN 接口的主流标准目前为 OpenFlow 协议，许多运营商和生产厂商根据该标准进行研发。互联网工程任务组(IETF)的 ForCES 工作组、互联网研究专门工作组(IRTF)的 SDNRG 研究组以及国际电信联盟远程通信标准化组织(ITU-T)的多个工作组同样针对 SDN 的新方法和新应用等展开研究。标准化组织的跟进是 SDN 市场快速发展的先决条件。据悉，SDN 市场已于 2013 年达到约 2 亿美元的产值，预计到 2016 年将达到 12 亿美元[6]，市场需求确保 SDN 有足够的发展空间。由此可见，SDN 具有广阔的发展前景和巨大的研究价值。下面从商业化方面，根据几个典型的 SDN 解决方案分析 SDN 带来的网络变革。

首先从性能上，可以以 SDN 的先期“试水者”谷歌为例，谷歌曾经部署过一个 SDN WAN 骨干，现在这个网络正在为其带来巨大的性能飞跃。事实证明，采用 SDN 可以获得更高的网络利用率。据了解，目前业界最佳的线路利用率也就在 30%～40%，而谷歌所运营的 SDN WAN 线路，只是通过细致的流量工程和优先策略，便可将利用率提升到接近 100%。换句话说，在出现故障时，可以牺牲没有任何严格交付期限的弹性流量，来保护高优先度的流量。

在网络简化方面，First Tracks 的首席分析师 Mark Leary 称，SDN 的初期收益就是对网络的简化。“围绕集中控制结构的整合可获得更高程度的自动化。这是一个可以立即看到影响的地方。渐进式的采用是成功的关键，”Mark Leary 说。“有些 SDN 解决方案的好处就在于它可以分成各自独立的部分。你可以把这些部分置入企业网络的有些

部分以便减少网络的复杂性，并可立刻看到收获。然后再进行扩展，最终简化企业的整个架构。SDN 的改进是动态的，例如可允许网络逐渐地适应负载的变化。”

从成本效益上看，另外一个典型的 SDN 成功案例，是印第安纳州立大学。该大学原本需要一台负载均衡设备放到 10Gb/s 互联网主干上去，把流量解析到多个入侵检测系统上去做分析，但市场反馈说这将花费 10～20 万美元去购买设备。这时他们才采用了 SDN，购买了一台价值仅 4 万美元、可支持 OpenFlow 的交换机去处理负载均衡任务，其成本效益是显而易见的。

“未来交换机将是白牌机的天下”，提到 SDN 的成本效益，许多人认为网络硬件的“白菜”成本将可能实现，低廉的白牌机将统治网络市场。按照“理想化”的 SDN 理念来说，这是一个不争的事实。但是，传统的网络市场被众多的巨头厂商所把持，这些厂商不会坐以待毙的，而事实也证明了这一点。目前，很多厂商都会在可用性、可管理性、性能、容量等方面极力实现差异化，因此 SDN 不太可能只是在构建网络时采用一些廉价的白牌机这么简单。

另外，SDN 使得产品升级周期更短，运营商们认为，SDN 能够高效使用通用服务器，从而帮助他们同时削减资本支出和运营成本，并为创造新服务提供一个更好的环境。此外，SDN 的关键益处还在于，它将使产品升级和更换周期变得更短、更容易，可以在通用的 X86 硬件上部署 SDN。然而，业内普遍也都承认，一些功能尚不能在这类服务器上进行部署，这在业界仍是一个富有争议的领域。

5.1.4 SDN 架构

目前各大厂商对 SDN 的系统架构都有自己的方案，针对不同的需求，各个组织也从不同的侧重点提出了许多相应的 SDN 参考架构。除了上述 ONF 提出的 SDN 基本模型，还有欧洲电信标准化组织（ETSI）针对运营商需求提出的 NFV 架构，得到了业界的广泛支持，以及各大设备厂商和软件公司提出的 OpenDaylight，作为一种 SDN 架构的具体实现形式已经部署在实际场景中。本节首先详细分析 ONF 提出的 SDN 体系结构中的组成部分，再与 NFV 和 OpenDaylight 的架构对比，以便读者对软件定义网络的架构以及相关项目有更明确的了解。

ONF 定义的 SDN 系统架构如图 5-4 所示，由下到上可以分为数据平面、控制平面和应用平面。该架构的最终目的是实现一套完整的编程接口开放给应用软件，这套接口称为 SDN 控制数据平面接口（control-data-plane in interface，CDPI）。控制平面上层的软件应用（SDN 应用程序）可以通过 CDPI 控制网络中的资源（数据层面设备资源）以及经过这些网络资源中的流量，并能根据应用需求灵活地调度这些流量。各个平面直接通过不同的协议进行交互，下面分别介绍一下该框架中的组成部分。

（1）数据平面：主要由网络元素构成，每个网元可以包含一个或者多个 SDN 数据路径，每个 SDN 数据路径是一个逻辑上的网络设备，数据平面没有控制功能，仅仅用于转发和处理数据。在逻辑上，数据平面代表全部或者部分物理资源。如图 5-4 所示，一个完整的网络元素包括 CDPI 代理、转发引擎或处理函数。

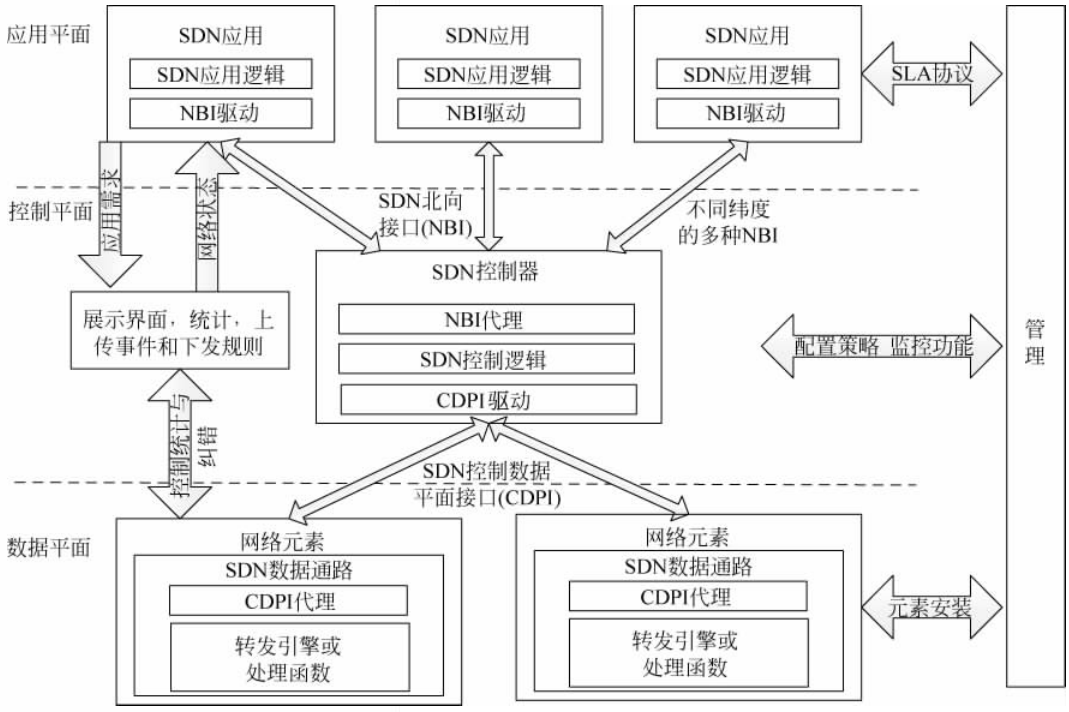

图 5-4　ONF 定义的 SDN 系统架构

(2) 控制平面：控制平面将全网视图抽象成网络服务，通过访问 CDPI 代理来调用相应的网络数据通路，并为运营商、科研人员及第三方等提供易用的北向接口(NBI)，方便这些人员订制私有化应用，实现对网络的逻辑管理。一般来说，SDN 控制器应该包括 CDPI 驱动、SDN 控制逻辑与 NBI 代理。

(3) 应用平面：包含着各类基于 SDN 的网络应用，用户无须关心底层设备的技术细节，仅通过简单的编程就能实现新应用的快速部署。CDPI 负责将转发规则从网络操作系统发送到网络设备，它要求能够匹配不同厂商和型号的设备，而并不影响控制层及以上的逻辑。NBI 允许第三方开发个人网络管理软件和应用，为管理人员提供更多的选择。网络抽象特性允许用户可以根据需求选择不同的网络操作系统，而并不影响物理设备的正常运行。

再来看一下 NFV。NFV 是针对运营商网络出现的问题而提出的 SDN 解决方案。网络运营商的网络由专用设备来部署，随着各种新型网络服务的产生，这些专属设备功能变得繁杂，而管理这些繁杂的硬件设备造成运营成本及能耗的增加，从而导致运营商网络的发展遇到瓶颈。针对上述问题，NFV 将传统网络设备的软件与硬件相分离，使网络功能更新独立于硬件设备。为此，NFV 采用了资源虚拟化的方式，在硬件设备中建立一个网络虚拟层，负责将硬件资源虚拟化，形成虚拟计算资源、虚拟存储资源和虚拟网络资源等，运营商通过软件来管理这些虚拟资源。由于采用的是通用硬件设备，NFV 降低了设备成本，减少了能耗，缩短了新网络服务的部署周期，从而适应网络运营商的发展需求。NFV 架构在设计时参考了 ONF 的 SDN 架构设计，也体现了转发和控制层面的分离，控制层上提出了类似 SDN 中应用层的虚拟化架构的管理编排层，在接口设计方面，

NFV 既可以基于非 OpenFlow 协议，又能与 OpenFlow 协同工作，同时还支持多种传统接口标准化协议，以便适应网络运营商对设备的不同需求，并与 ONF 的 SDN 的发展保持相对独立。NFV 白皮书中提出的 NFV 架构如图 5-5 所示。

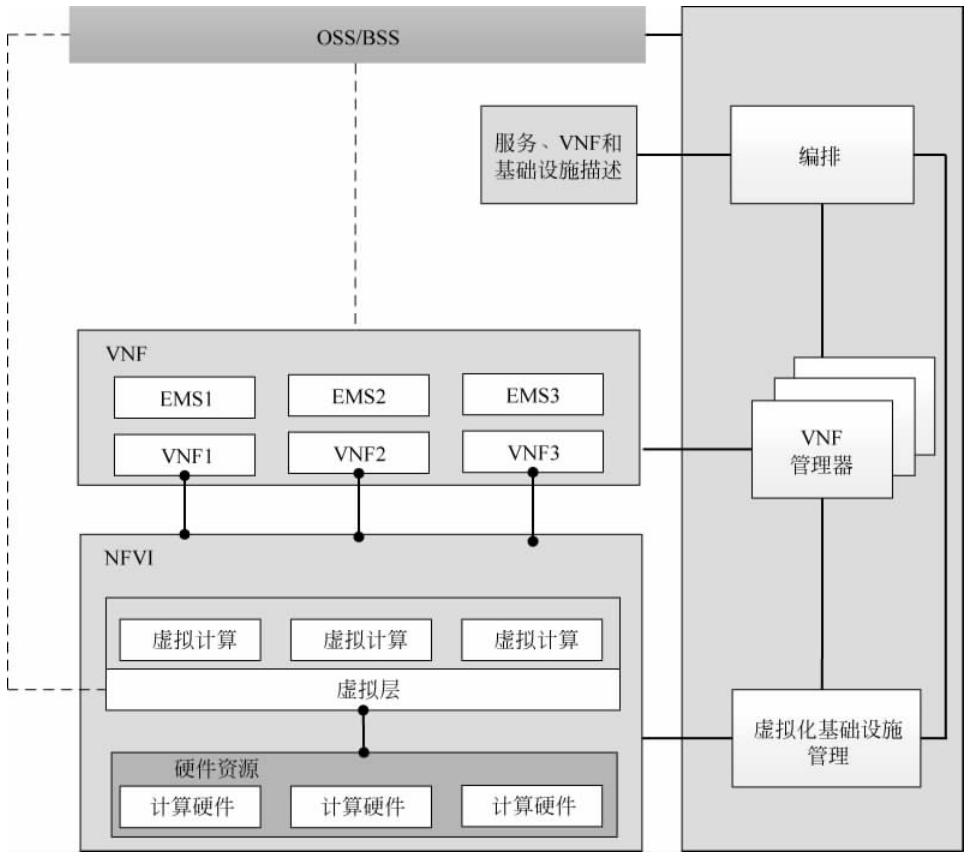

图 5-5　NFV 架构

OpenDaylight[7]开源项目提出于 2013 年，主要参与者包括思科、Juniper 这类传统网络设备生产商、IBM、微软等传统 IT 软硬件设备厂商，以及 Arista、Big Switch 等新兴 IT 软件厂商。因此，OpenDaylight 考虑了兼容性问题，继承 SDN 架构形式的同时结合了 NFV 的特点。架构整体分为三个层次，分别是网络应用与业务流程（即应用层）、控制平台（即控制层）和物理与虚拟网络设备（即数据层）、OpenDaylight 的控制平台直接由自带的 Java 虚拟机实现。针对不同的网络任务，控制器自身携带了一系列可插入模块，并兼容第三方模块以增强 SDN 的功能。与 ONF 的 SDN 架构最大的不同在于：OpenDaylight 控制器的南向接口除了支持 OpenFlow 协议之外，还支持 NETCONF 等配置协议与 BGP 等路由协议，并支持生产厂商的专有协议（如思科的 OnePK 协议）。为了能够处理不同的标准协议，OpenDaylight 增加了服务抽象层 SAL，它负责将不同的底层协议标准转换成 OpenDaylight 控制层所理解的请求服务，保持了底层协议的透明性，并提高了整体架构的可扩展性。OpenDaylight 架构如图 5-6 所示。