2021 年软件定义网络复习笔记

考试时间：2021-5-27 2:00 PM-4:00 PM
考试地点：A3-202
考试题型：单选、多选、判断、简答
试卷语言：中文

目录

1. SDN 概述和基本原理

1.1 SDN 的产生与发展

1.1.1 什么是 SDN？

软件定义网络（Software Defined Networking, SDN）是一种新型的网络技术，其设计理念是将网络的控制平面与数据转发平面进行分离，并实现可编程化的集中控制。

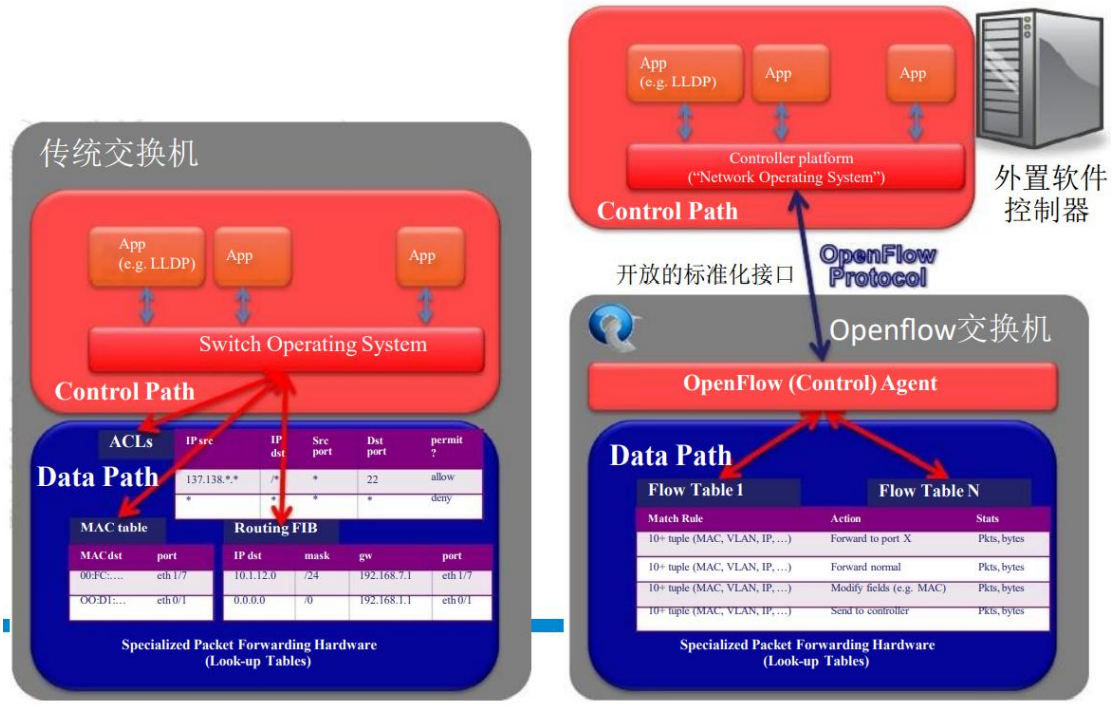
传统网络设备紧耦合的网络架构被分拆成应用、控制、转发三层分离的架构。控制功能被转移到了服务器，上层应用、底层转发设施被抽象成多个逻辑实体。

1.1.2 SDN 的特点（数控分离）

控制平面与数据平面分离

1.1.3 SDN 与传统网络的对比？

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，由其负责对不同业务特性的适配，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是可以通过编程实现。



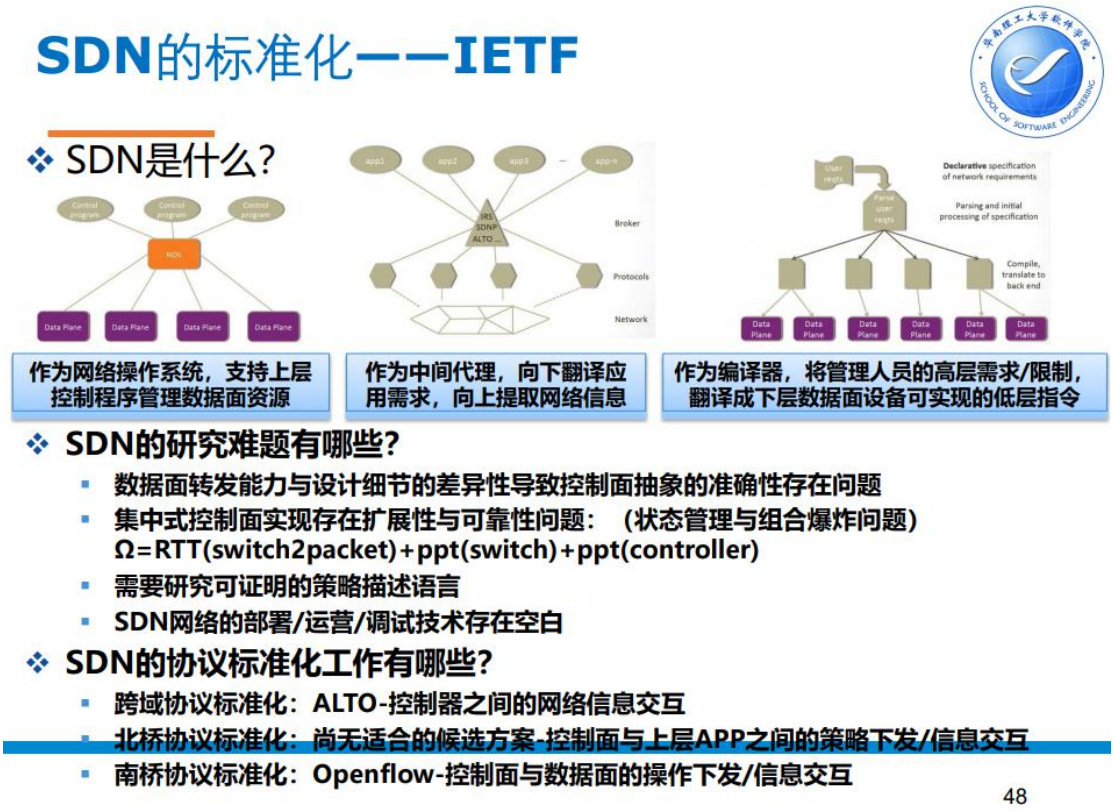
1.2 SDN 标准化

1.2.1 标准化组织：ONF

Open Networking Foundation（以下简称 ONF）是目前 SDN 领域最有影响力的国际标准化组织，同时，还是 SDN 关键标准 OpenFlow 的制定者。

ONF 的核心目标是通过产业界的共同努力，完善 SDN 架构、相关的功能、协议，并制定 国际标准，从而推进 SDN 的商用化，促进 SDN 产业链和生态系统的成熟，以期未来网络运营商可以更加灵活的、采用更低廉价格的交换/路由设备进行组网。

1.2.2 IETF



（原文如此）

IETF 的主要任务是负责互联网相关技术标准的研发和制定，是国际互联网业界具有一定权威的网络相关技术研究团体。IETF 大量的技术性工作均由其内部的各种工作组（Working Group，简称 WG）承担和完成。这些工作组依据各项不同类别的研究课题而组建。

参考：互联网工程任务组
https://baike.baidu.com/item/%E4%BA%92%E8%81%94%E7%BD%91%E5%B7%A5%E7%A8%8B%E4%BB%BB%E5%8A%A1%E7%BB%84

1.3 SDN 的基本架构

1.3.1 SDN 主流架构

ONF 定义的基于 OpenFlow 的架构：

特点：（1）转发与控制分离；（2）标准化转发面。

优点：易于流量调度

IETF 提出的技术架构：

特点：（1）开放网络设备能力；（2）标准化 API。

优点：（1）充分利用现有设备；（2）快速实现。

NICIRA 提出的 Overlay 技术架构：

特点：（1）网络边缘软件化；（2）Overlay 技术。

优点：（1）与物理网络解耦；（2）部署灵活。

ETSI 提出的 NFV 技术架构：

NFV 与 SDN 技术互补；但有本质区别，严格来说不能算是。

1.3.2 ONF 定义的 SDN 基本架构（分层，架构图、内容）

分为 3 层：

应用层：不同的应用逻辑通过控制层开放的 API 管理能力控制设备的报文转发功能

控制层：由 SDN 控制软件组成，与下层可用 OpenFlow 协议通信

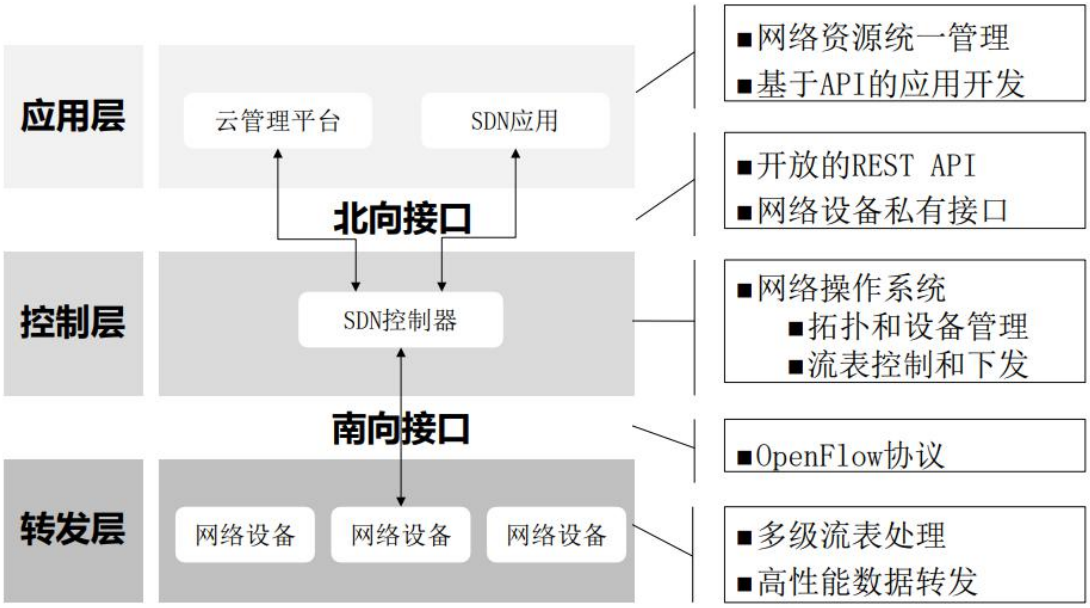
基础设施层（物理层）：由转发设备组成

意义：

（1）一种新型网络创新架构，实现了网络设备控制与转发的分离。

（2）网络虚拟化的一种实现方式，其核心技术 OpenFlow。

（3）实现了网络流量的灵活控制， 使网络变得更加智能。



1.4 SDN 的核心思想

1.4.1 解耦、抽象和可编程及其涵义

A. 数据平面与控制平面的解耦

通过解耦合，控制平面负责上层的控制决策，数据平面负责数据的交换转发，双方遵循一定的开放接口进行通信；

实现网络逻辑集中控制的前提；

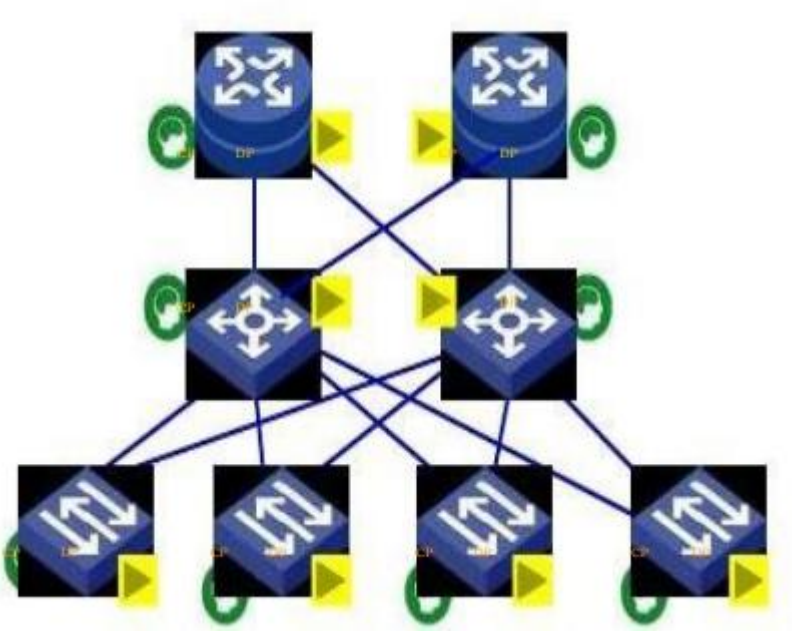
两个平面独立完成体系结构和技术的发展演进，有利于网络的技术创新与发展。

控制与转发分离：

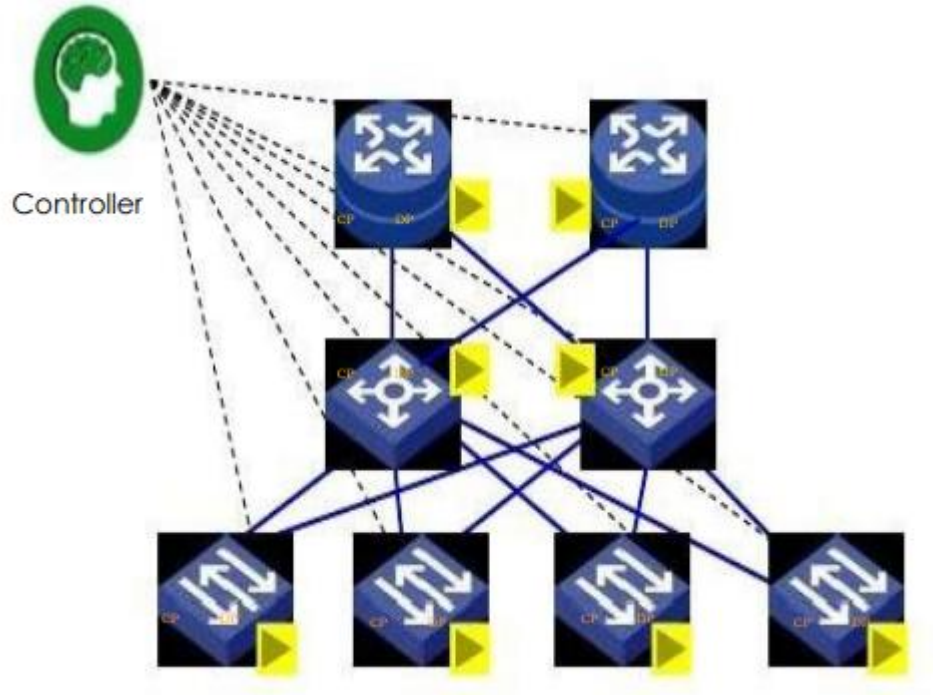
传统网络设备的控制平面（Control Plane, CP）与数据平面（Data Plane, DP）不分离；设备之间通过控制协议交互转发信息。

SDN 将网络设备的控制平面集中上收到控制器；网络设备上只保留转发平面（转发表项）；软件可以实现灵活的控制面功能满足用户多元化需求；硬件成为简单哑资源，专注转发。

传统网络设备：



SDN：



B. 网络功能的抽象

ONF 网络架构实现转发抽象、分布状态抽象和配置抽象：

转发抽象（forwarding abstration）：隐藏了底层的硬件实现，转发行为与硬件无关；

分布状态抽象（distribution abstration）：屏蔽分布式控制的实现细节，为上层应用提供全局网络视图；

配置抽象（specification abstration）：网络行为的表达通过网络编程语言实现，将抽象配置映射为物理配置。

Overlay 网络架构实现对基础网络设施的抽象。

C. 可编程网络

可编程网络（Programmable Network）主要利用在网络节点中提供标准的网络应用编程接口向用户和网络业务供应者提供一个“开放”的网络控制机制，与传统 Internet 的区别就在于 Internet 是无状态的，而可编程网络是有状态的并且可以由用户控制和改变的。

传统网络的管理接口：CLI、SNMP 等，是初级的网络编程方式；

网络管理者需要基于整个网络的，而不是基于某一设备的可编程；

网络可编程相关研究：主动网络（Active Networking）、4D 架构。

主动网络：

一是被称为 ANN（Active Node Network）的网络中间节点，不仅完成存储转发等网络功能，而且可以对包含数据和代码的所谓主动包和普通包进行计算；

二是用户根据网络应用和服务的要求可以对网络进行编程以完成这些计算。

4D 架构：

4D 架构（4D Architecture）研究就是这样一个采用“白板设计”方式的研究项目。该项目继承了 D. Clark 等关于“知识平面”的设想，提出了一种新型的互联网控制管理架构。

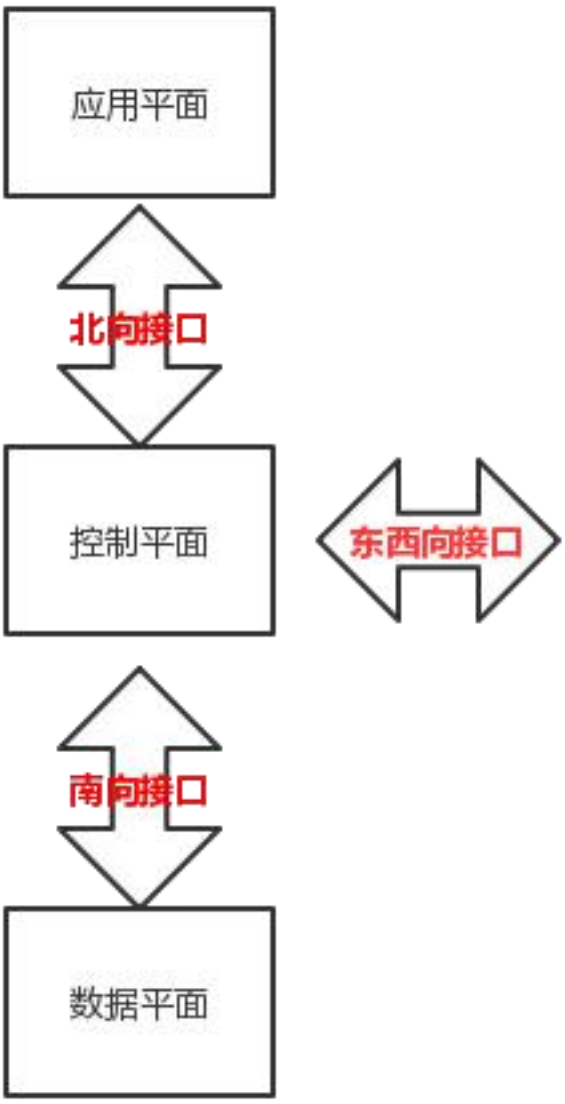
4D 架构将可编程的决策平面（即控制层）从数据平面分离，使控制平面逻辑中心化与自动化，其设计思想产生 SDN 控制器的雏形。

SDN 网络可编程接口：

北向接口：REST（Representational State Transfer） API、RESTCONF 协议；

南向接口：OpenFlow、OF-Config、NETCONF、OVSDB、XMPP、PCEP、I2RS、OPFlex 等协议；

东西向接口：负责控制器之间的通信，未形成统一标准。



2. 数据平面

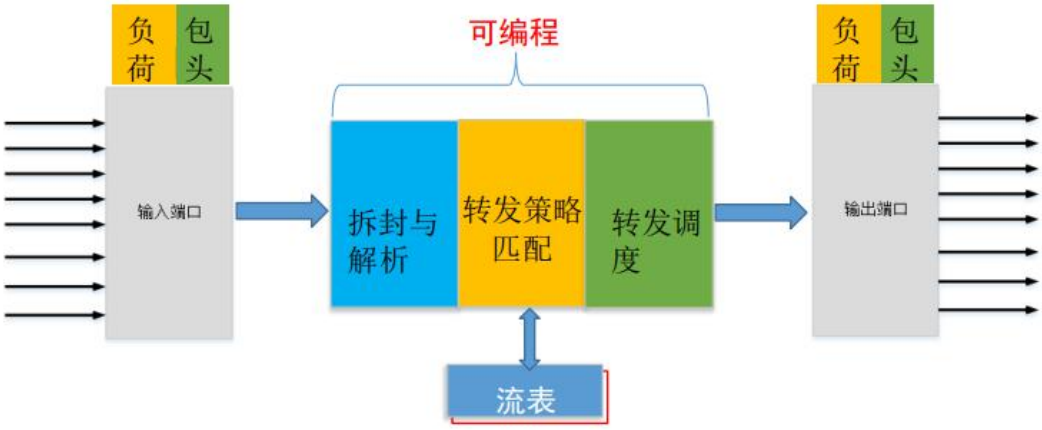
2.1 SDN 数据平面

2.1.1 SDN 数据平面的定义和特点

网络设备的基本任务是处理和转发不同端口上各种类型的数据，对于数据处理过程中各种具体的处理转发过程，例如 L2/L3/ACL/QOS/组播/安全防护等各功能的具体执行过程，都属于数据转发平面的任务范畴。

处理流程中的：解析（Parser）、转发（Forwarding）和调度（Scheduling）都是可编程、协议无关的；

传统网络设备的二层或三层转发表抽象成流表；



参考：

SDN 数据平面简介 https://blog.csdn.net/jlwuqi/article/details/90141033

控制平面与数据平面定义 https://blog.csdn.net/jiayanhui2877/article/details/8684555

2.2 OpenFlow

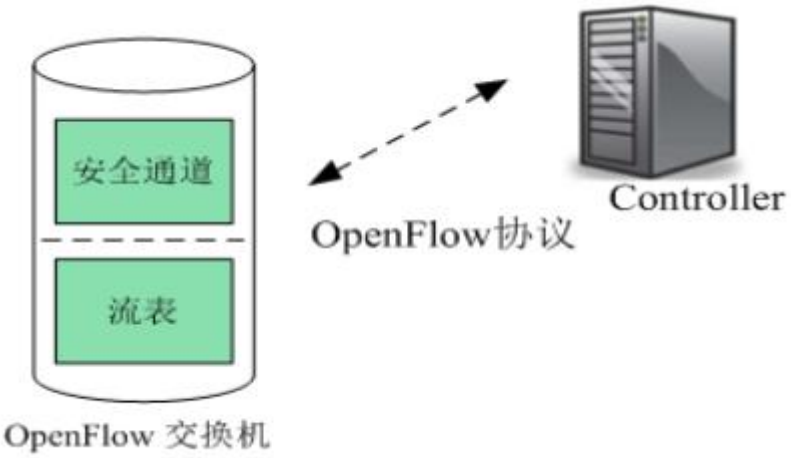
2.2.1 OpenFlow 架构

OpenFlow 是一整套软件应用程序接口，OpenFlow 控制器可以通过规范与支持 OpenFlow 交换机沟通配置信息，决定数据转发平面的转发表，控制器与交换机间通过 SSL 加密传输。

流表（Flow Table），每个动作（Action）关联一个流表项（Flow Entry），指示交换机如何进行流（Flow）的处理；

安全通道（Sccure Channel），OpenFlow 交换机通过安全通道与远端控制器连接，负责控制器与交换机之间的交互；

OpenFlow 协议（Protocol），定义了一种南向接口标准，为控制器与交换机之间的通信提供了一种开放标准的方式。



2.2.2 OpenFlow 主要版本和特点（重点是 OpenFlow v1.0）

版本号	发布时间	主要特性
1.0	2009-12	单流表

1.1	2011-2	流水线、组表
1.2	2011-12	多控制器、IPv6 基本头
1.3	2012-6	计量表、IPv6 扩展头
1.4	2013-10	流表同步机制
1.5	2014-12	出向流表

2009 年 10 月，ONF 发布了具有里程碑意义的可用于商业化的 OpenFlow1.0 版本。

ONF 将 1.0 和 1.3 版本作为长期支持的稳定版本，此后一段时间内后续版本的发展要维持版本的兼容性。

重点技术规范文件：

openflow-spec-v1.0.0

openflow-spec-v1.3.0

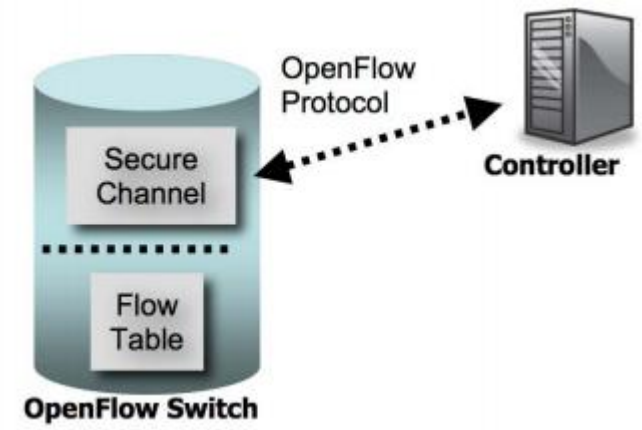
参考：

中文版:openflow-spec-v1.0.0 https://www.sdnlab.com/resource/11948.html

中文版：openflow-spec-v1.3 https://www.sdnlab.com/resource/11949.html

2.2.3 OpenFlow v1.0 的组成结构

每个 OpenFlow 交换机（switch）都有一张流表，进行包查找和转发。交换机可以通过 OpenFlow 协议经一个安全通道连接到外部控制器（controller），对流表进行查询和管理。



2.3 OpenFlow 流表

2.3.1 流表的定义

流的概念：

同一时间，经过同一网络中具有某种共同特征（属性）的数据，抽象为一个流。比如，可以将访问同一目的地址的数据视为一个流；

流一般由网络管理员定义，根据不同的流执行不同的策略；

OpenFlow 体系中，数据以“流”为单位进行处理。

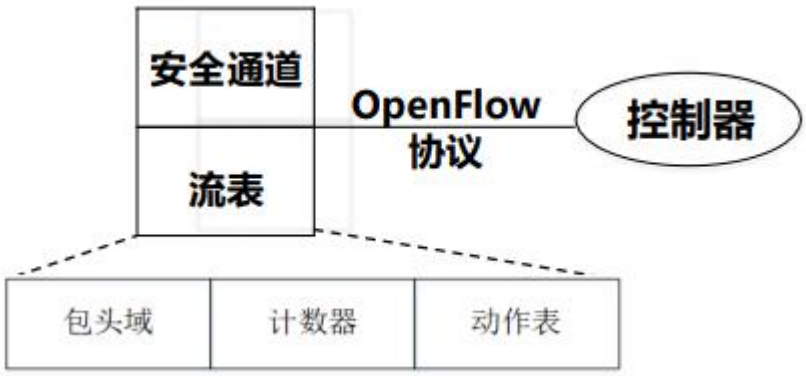
流表：

针对特定流的策略表项的集合，负责数据包的查找与转发。一张流表包含了一系列的流表项（flow entries）。

2.3.2 V1.0 的流表结构及其内容（包头域、计数器和动作）

流表是 OpenFlow 对网络设备的数据转发功能的抽象，表项包括了网络中各个层次的网

络配置信息，下图为 Openflow V1.0 版本的流表结构。



包头域：用于对交换机接收到的数据包的包头内容进行匹配

计数器：用于统计数据流量相关信息，可以针对交换机中的每张流表、每个数据流、每个设备端口、每个转发队列进行维护

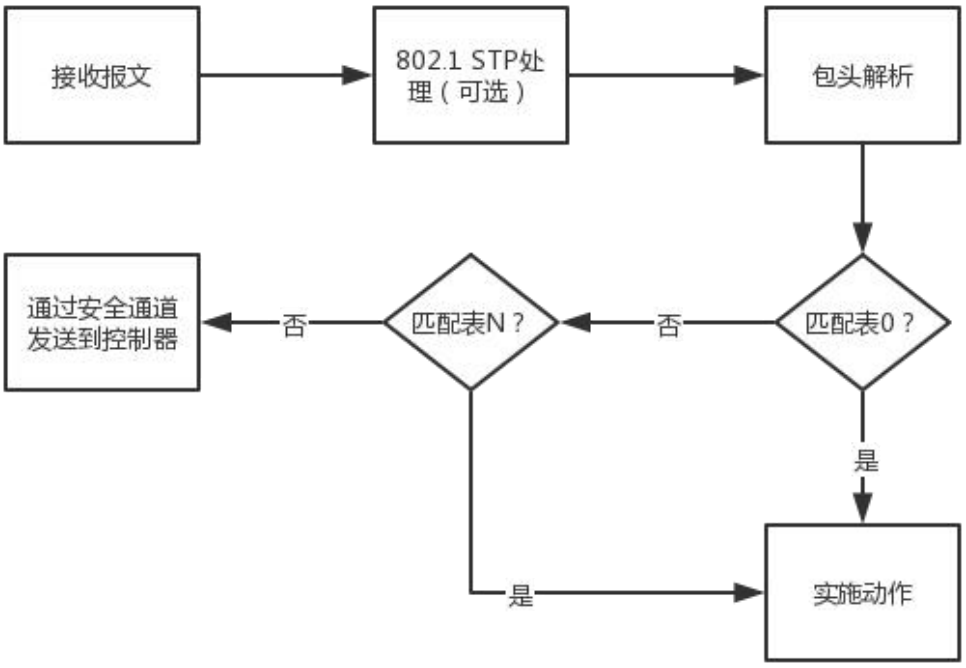
动作（action）：用于指示交换机在收到匹配数据包后如何对其进行处理。决定了 OpenFlow 对转发面行为的抽象能力

2.3.3 流表动作列表（常用必备动作、可选动作）

类型	名称	说明
必备动作	转发（Forward）	交换机必须支持将数据包转发给设备的物理端口及如下的一个或多个虚拟端口，实现组播、多路径转发、负载均衡等功能： ALL：转发给所有出端口，但不包括入端口 CONTROLLER：封装数据包并转发给控制器 LOCAL：转发给本地的网络栈 TABLE：对 packet_out 消息执行流表的动作 IN_PORT：从入端口发出
	丢弃（Drop）	交换机对没有明确指明处理动作的流表项，将会对与其所匹配的所有数据包进行默认的丢弃处理
可选动作	转发（Forward）	交换机可选支持将数据包转发给如下的虚拟端口： NORMAL：利用交换机所能支持的传统转发机制（例如二层的 MAC、VLAN 信息或者三层的 IP 信息）处理数据包 FLOOD：遵照最小生成树从设备出端口洪泛发出，但不包括入端口
	排队（Enqueue）	交换机将数据包转发到某个出端口对应的转发队列中，便于提供 QOS 支持
	修改域（Modify-Field）	交换机修改数据包的包头内容，各个字段值、封装/去封装，具体可以包括： 设置 VLAN ID、VLAN 优先级，剥离 VLAN 头 修改源 MAC 地址、目的 MAC 地址 修改源 IPv4 地址、目的 IPv4 地址、ToS 位 修改源 TCP/IP 端口、目的 TCP/IP 端口

2.3.4 OpenFlow 数据包处理流程

每个包按照优先级依次去匹配流表中表项，匹配包的优先级最高的表项即为匹配结果。匹配成功，对应的计数器将更新，同时实施转发动作；如果没能找到匹配的表项，则转发给控制器。



2.3.5 OpenFlow 的保留端口及内容

OpenFlow 保留端口用于特定的转发动作，如发送到控制器、洪泛，或使用非 OpenFlow 的方法转发，如使用传统交换机的处理过程。

类型	名称	说明
必备	ALL	转发给所有出端口，但不包括入端口
	CONTROLLER	封装数据包并转发给控制器
	TABLE	对 packet_out 数据包执行流表操作
	IN PORT	从入端口发出
	ANY	没有指定端口时，不能用于入端口和出端口
可选	LOCAL	转发给本地的网络栈
	NORMAL	利用交换机的传统转发机制处理数据包
	FLOOD	按照最小生成树从设备出端口洪泛发出

2.4 SDN 交换机

2.4.1 交换机类型

根据交换机的应用场景及其所能够支持的流表动作类型，OpenFlow 交换机可以被分为：

OpenFlow 专用交换机（OpenFlow-only）

只支持 OpenFlow 协议

OpenFlow 使能交换机（OpenFlow-enabled）

OpenFlow 1.1 及后续版本将其更名为“OpenFlow-hybrid”

考虑了 OpenFlow 交换机与传统交换机混合组网时可能遇到的协议栈不兼容问题，能同时运行 OpenFlow 协议和传统的二层/三层协议栈

支持 OpenFlow 可选转发动作中的 NORMAL 动作

2.4.2 交换芯片类型

芯片类型	优点	缺点	应用场景
通用 CPU	易扩展、通用性强	处理性能较低	应用于网络设备的控制和管理
专用集成电路芯片（Application-Specific Integrated Circuit，ASIC）	性能高、处理能力很强	不易扩展、研发周期长	应用于实现各种成熟的协议
现场可编程门阵列（Field Programmable Gate Array，FPGA）芯片	支持反复擦写、可编程	处理能力有限	应用于科研和验证
网络处理器（Network Processor，NP）	可编程、可进行复杂的多业务扩展	性能不及 ASIC 芯片	应用于路由器、防火墙等协议更为复杂的网络设备

2.4.3 SDN 交换机

SDN 物理（硬件）交换机：

多数为支持 OpenFlow 的混合模式交换机，支持传统二、三层处理模式和 SDN 处理模式；

SDN 虚拟（软件）交换机：

成本低、配置灵活，性能满足中小规模网络要求。

2.4.4 SDN 交换机选型的参数考虑（例如背板带宽、频率等）

背板带宽：

交换机接口处理器和数据总线间的最大吞吐数据量，背板带宽越高，所能处理数据的能力就越强。

背板带宽从几个 Gbps 到几百 Gbps 不等，不是越高越好，只要满足端口线速转发就可以。

对于线速转发的交换机，背板的带宽计算方法（在全双工模式下）：$B = 2pb$，

其中 p 为端口数，b 相应端口的带宽。

端口密度：

代表交换机转发能力。密度越大，设备的转发能力越强；

端口速率：

速率越高，设备的处理性能越强；

支持模块类型：

类型越多，实用性越强，可应用于不同的网络环境，比如：LAN 接口、WAN 接口、ATM 接口。

端口带宽类型：

越丰富越好，既支持 40G、100G 高速端口，又支持百兆、千兆低速端口，还支持

XFP 又支持 SFP、SFP+、CFP 等等多种光接口类型。

其它参数：

时延（延迟）、功耗、支持 OpenFlow 版本、机架单元、网管功能

3. 南向接口协议

3.1 南向接口协议概述

3.1.1 南向接口的定义（SDN 架构中的位置）

管理其他厂家网管或设备的接口，即向下提供的接口。

参考：南向接口

https://baike.baidu.com/item/%E5%8D%97%E5%90%91%E6%8E%A5%E5%8F%A3

3.1.2 什么是南向接口协议？

南向接口协议：为控制平面的控制器和数据平面的交换机之间的信息交互而设计的协议。

其设计目标：

实现数据平面与控制平面的信息交互

向上收集数据平面信息；

向下下发控制策略，指导转发行为。

实现网络的配置与管理；

实现路径计算，包括链路属性（带宽与开销）、链路状态、和拓扑信息等。

3.1.3 常见南向接口协议及其设计目标（OpenFlow、OF-config、NETCONF、OVSDB）

南向接口协议	设计目标
OpenFlow	用于 OpenFlow 交换机与控制器的信息交互
OF-Config	用于 OpenFlow 交换机的配置与管理
NETCONF	用于网络设备的配置与管理
OVSDB	用于 Open vSwitch 的配置与管理，主要管理对象是 OVSDB 数据库
XMPP	用于即时通信、游戏平台、语音与视频会议系统，OpenContrail 控制器利用 XMPP 与 vRouter 进行信息交互
PCEP	PCEP 为 PCE 和 PCC 之间的通信协议，实现路径计算
I2RS	I2RS 体系架构中的南向接口协议
OpFlex	思科 ACI 体系中的策略控制协议

3.2 OpenFlow 协议

3.2.1 OpenFlow 协议的消息类型

OpenFlow 协议是用来描述控制器和 OpenFlow 交换机之间数据交互所用信息的接口标准，其核心是 OpenFlow 协议信息的集合。

controller-to-switch：由控制器发起，用来管理或获取 OpenFlow 交换机的状态。

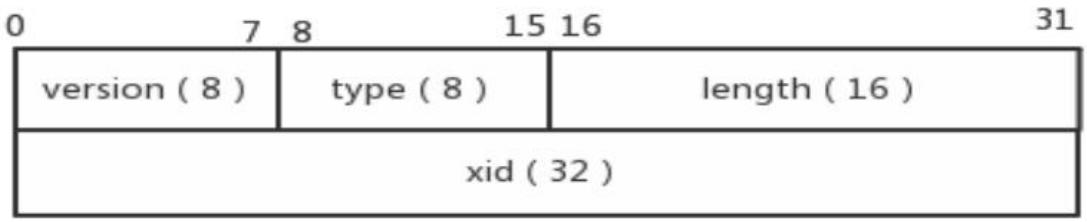
asynchronous（异步）：由 OpenFlow 交换机发起，用来将网络事件或交换机状态变化更新到控制器。

symmetric（对称）：由交换机或控制器发起。

3.2.2 消息格式

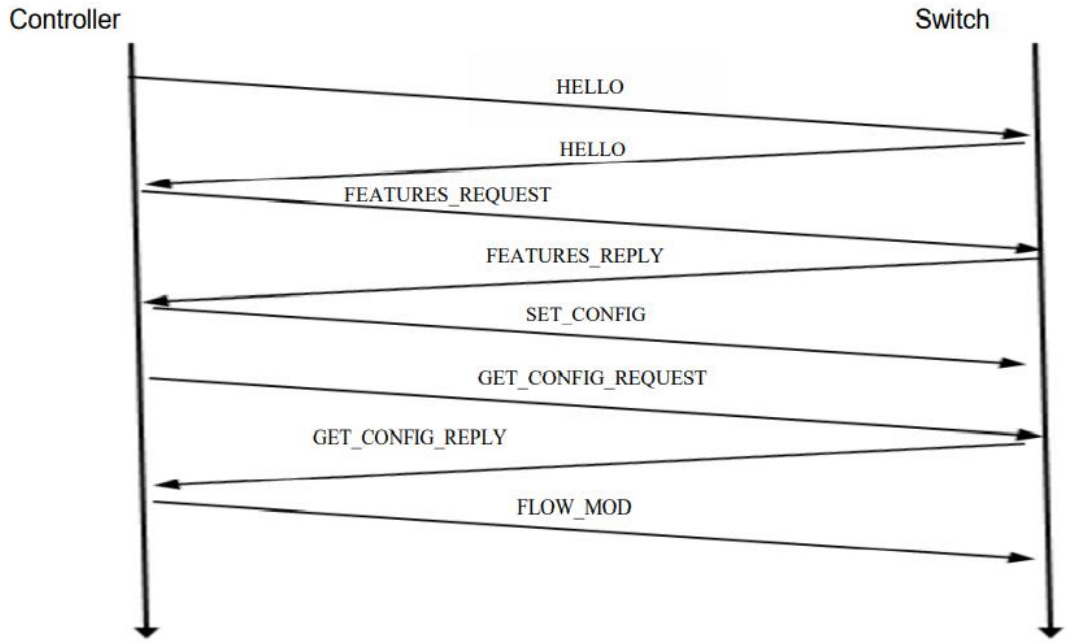
Openflow 协议数据包：Openflow Header 和 Openflow Message

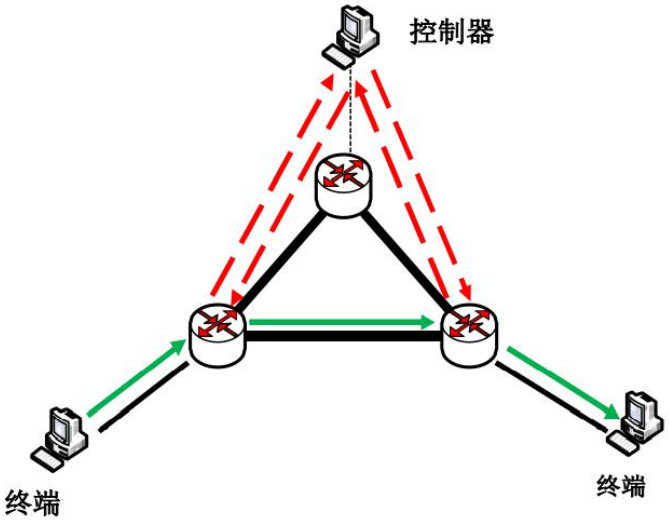
Openflow Header 格式：




```
/* Header on all OpenFlow packets. */
struct ofp_header {
    uint8_t version; /* OpenFlow 的协议版本号*/
    uint8_t type; /*消息类型，是个常数*/
    uint16_t length; /*数据包字节数*/
    uint32_t xid; /*数据包的标识 id*/
};
```

3.2.3 OpenFlow 的 SDN 通信流程





- ①主机向网络发送数据包
- ②OF交换机流表无匹配项，通过 PacketIn事件将数据包上报给控制器
- ③控制器下发流表（或PacketOut）
- ④数据包转发
- ⑤同②
- ⑥同③
- ⑦数据包转发

3.3 OF-Config

3.3.1 协议概述、设计目标和设计思想

A. 协议概述

OF-Config 协议就是一种 OpenFlow 交换机管理配置协议（OpenFlow Management and Configuration Protocol）；

OpenFlow 的伴侣协议（Considered a complementary protocol）；

OpenFlow 提出了由控制器向 OpenFlow 交换机发送流表以控制数据流通过网络所经过的路径的方式，但是并没有规定如何管理和配置这些网络设备；

OF-config（OpenFlow Configuration and Management Protocol）提供了开放的接口用于远程配置和控制 OpenFlow 交换机，但是它并不会影响到流表的内容和数据转发行为。

OpenFlow 协议的同伴协议，目前采用 NETCONF 协议进行传输。

ONF 的 Configuration & Management 工作组负责协议维护。

OF-config 发展历史及其与 OpenFlow 的版本对应关系：

OF-config 规范版本	规范发布时间	对应 OpenFlow 版本
OF-Config 1.0	2012-1-6	OpenFlow 1.2
OF-Config 1.1	2012-6-25	OpenFlow 1.3
OF-Config 1.1.1	2013-3-23	OpenFlow 1.3.1
OF-Config 1.2 + Yang Model	2014	OpenFlow 1.3 & 1.0, 1.1, 1.2

B. 设计目标

协议设计需求（Requirements）：

（1）配置需求（Specification Requirements）

需要在支持 OpenFlow 1.2 的网络设备上实现以下基本功能配置：

配置一至多个控制器的 IP 地址；

配置设备的队列、端口等资源；

支持远程修改设备的端口状态。

针对 OpenFlow 1.2 功能配置的需求定义：

项目	说明

控制器连接	支持在交换机上配置控制器参数
多控制器	支持多个控制器的参数配置
逻辑交换机	支持逻辑交换机（即 OpenFlow 交换机的实例）资源设置，且支持带外设置
连接中断	支持故障安全、故障脱机等两种应对模式的设置
加密传输	支持控制器与交换机之间 TSL 隧道参数的设置
队列	支持队列参数的配置，包括：最小速率、最大速率、自定义参数等
端口	支持交换机端口参数/特征配置，即使 OpenFlow 1.2 中并无额外方式要求
数据通路标识	支持长度为 64 位的数据通路标识的配置

（2）操作运维需求（Operational Requirements）

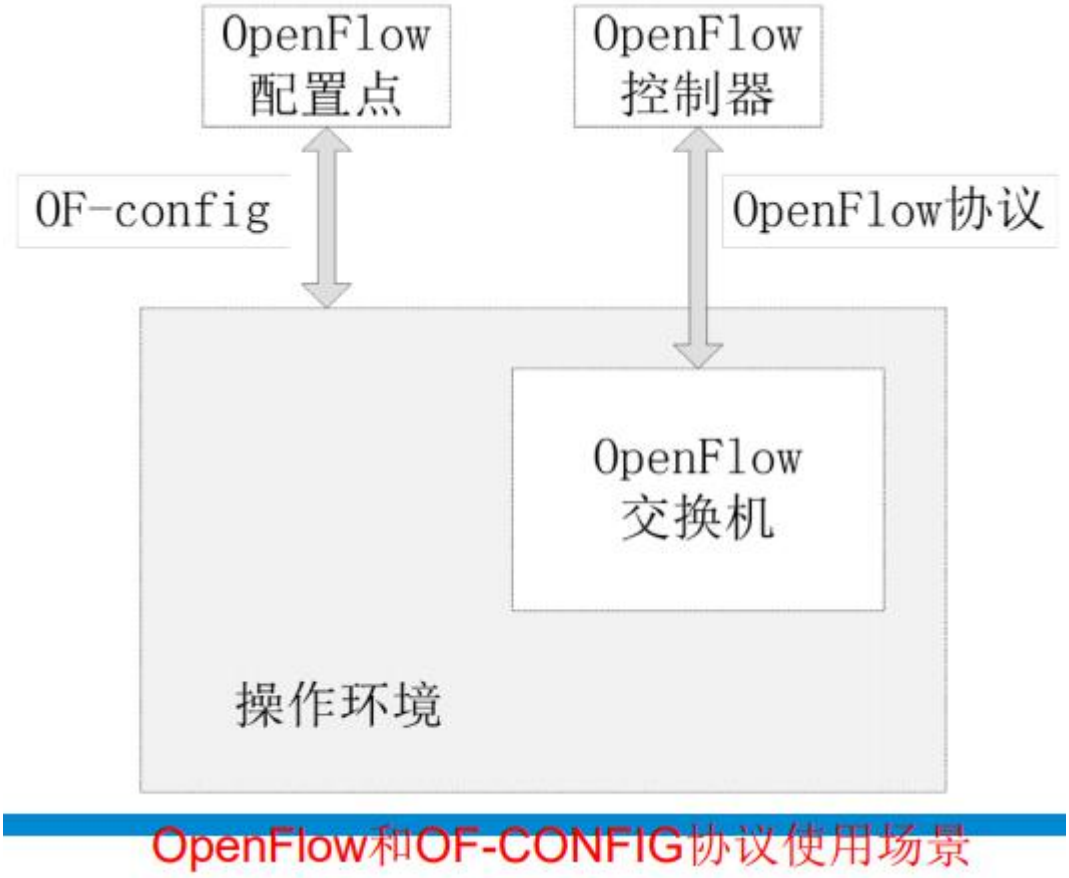
支持 OF 交换机被多个 OpenFlow 配置点配置；
支持一个 OpenFlow 配置点管理多个 OF 交换机；
支持一个 OpenFlow 逻辑交换机被多个控制器控制；
支持持对已分配给逻辑交换机的端口和队列的配置。

（3）管理协议需求（Management Protocol Requirements）

必须安全，能够确保完整性和私密性，并提供双向身份认证；
支持由交换机或者配置点发起连接，支持对部分交换机的配置；
必须具有良好的扩展性，能够提供协议能力报告等等。

C. 设计思想

在 OpenFlow 架构上增加一个被称作 OpenFlow Configuration Point 的配置节点，该节点既可以是控制器上一个软件进程，也可以是传统的网管设备，它通过 OF-config 协议对 OpenFlow 交换机进行管理。



OpenFlow和OF-CONFIG协议使用场景

OpenFlow 交换机上所有参与数据转发的软硬件（例如端口、队列等）都可被视作网络资源。

3.3.2 OF-Config 数据模型（XML）和传输协议（NETCONF）

数据模型：为应对 OpenFlow 演进，采用基于 XML 定义的数据模型描述系统的各个组成部分，具有良好的扩展性。

传输协议：选取利用 NETCONF 进行 OF-config 协议的传输。

3.4 NETCONF

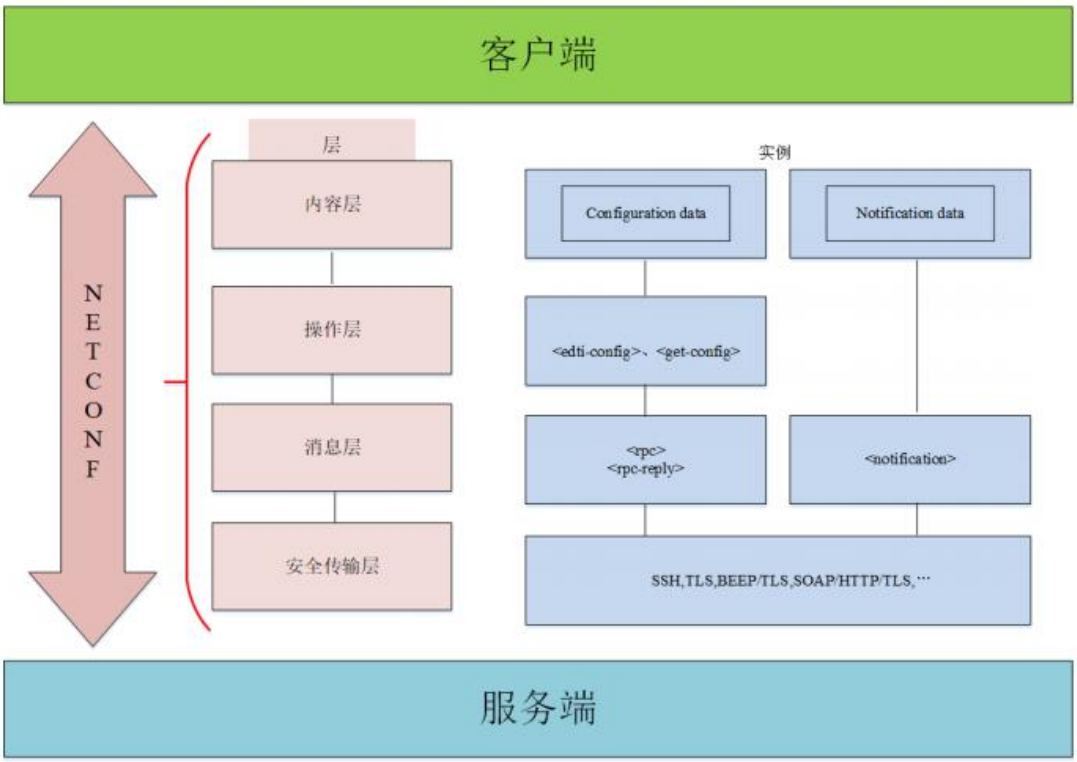
3.4.1 什么是 NETCONF 协议？设计目标

新一代网管协议；

网络配置协议 NETCONF（Network Configuration Protocol）提供一套管理网络设备的机制；

2003 年成立了 NETCONF 工作组，2011 年更新版的 RFC 6241 发布。

3.4.2 NETCONF 协议框架（操作层和内容层的内容）



操作层：

对数据库信息的获取、配置、复制和删除等功能

基本操作	说明
<get-config>	从<running/>、<candidate/>和<startup/>配置数据库中获取配置数据。
<get>	从<running/>配置数据库中获取配置数据和设备的状态数据。
<edit-config>	修改、创建、删除配置数据。
<copy-config>	源配置数据库替换目标配置数据库。如果目标配置数据库没有创建，则直接创建配置数据库，否则用源配置数据库直接覆盖目标配置数据库。
<delete-config>	删除一个配置数据库，但不能删除<running/>配置数据库。
<lock>	锁定设备的<running/>数据库，独占配置数据库的修改权。这种锁定防止产生冲突。
<unlock>	取消用户自己之前执行的<lock>操作。
<close-session>	正常关闭当前 NETCONF 会话。
<kill-session>	强制关闭另一个 NETCONF 会话，只有管理员用户才有权限。

内容层：

描述了网络管理所涉及的配置数据：

<running/>

<candidate/>

<startup/>

使用 YANG 语言进行建模，YANG 具有以下特点：

层级树形结构

可以直接映射到 XML

可读性好，易学习

可复用、可扩展

4. SDN 控制平面与北向接口协议

4.1 SDN 控制平面概述

4.1.1 什么是 SDN 控制平面（SDN 架构中的位置）？

一个或多个 SDN 控制器组成，是网络的大脑。

对底层网络交换设备进行集中管理，状态监测、转发决策以及处理和调度数据平面的流量；

向上层应用开放多个层次的可编程能力。

4.1.2 南向控制协议的任务和功能（链路发现、拓扑管理、策略制定、表项下发及其涵义）

通过南向接口协议进行链路发现、拓扑管理、策略制定、表项下发等：

链路发现：

获得 SDN 全网信息，实现网络地址学习、VLAN、路由转发

拓扑管理：

监控和采集 SDN 交换机的信息，反馈工作状态和链路连接状态

策略制定：

流表生成算法是影响控制器智能化水平的关键因素

针对不同层次的传输需求，制定相应的转发策略并生成对应的流表项

表项下发：

通过流表下发机制控制交换机的数据包转发

主动（proactive）下发：数据包到达交换机之前进行流表设置。

被动（reactive）下发：交换机接收到一个数据包并且没有发现匹配的流表项，将其送给控制器处理。

4.1.3 北向业务支撑方式

通过北向接口为上层业务应用及资源管理系统提供灵活的网络资源抽象；

北向接口定义是 SDN 领域关注和争论的焦点；

REST API 是用户比较容易接受的方式。

4.2 主流开源控制器

名称	编程语言	特征简介
NOX	C++	由 Nicira 开发，业界第一款 OpenFlow 控制器，是众多 SDN 研发项目的基础
POX	Python	由 Nicira 开发，是 NOX 的纯 Python 实现版本，支持控制器原型功能的快速开发
Ryu	Python	由 NTT 开发，能够与 OpenStack 平台整合，具有丰富的控制器 API，支持网络管控应用的创建
Floodlight	Java	由 Big Switch Networks 开发，是企业级的 OpenFlow 控制器，基于 Beacon
OpenDayLight	Java	ODL 主要由设备厂商驱动，如 Cisco、IBM、HP、NEC 等

5. SDN 北向接口协议

5.1 SDN 北向接口概述

5.1.1 什么是北向接口？

应用平面与控制平面之间的接口（NBI），通过控制器向上层业务应用开放的接口，为

上层业务应用和资源管理系统提供灵活的网络资源抽象；

需要满足多样性、合理性和开放性，未形成业界公认标准。

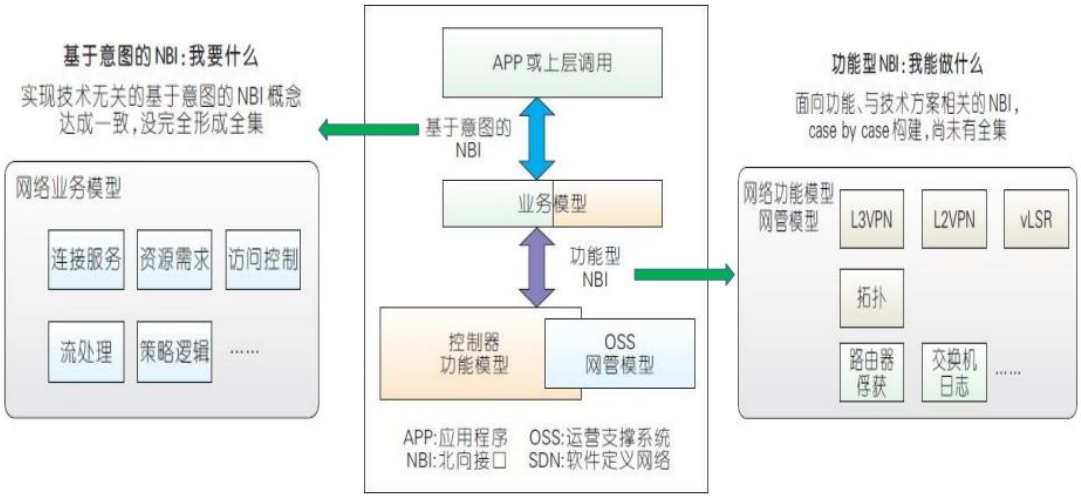
5.1.2 北向接口的设计（功能型、基于意图）与网络模型

北向接口设计：

功能型北向接口（Functional NBI）：自下而上看网络，重点在网络资源抽象及控制能力的开放，包括 Topology、L2VPN、L3VPN、Tunnel 等接口。

基于意图的北向接口（Intent-based Interface）：自上而下看网络，关注应用或者服务的需求，同具体的网络技术无关。

北向接口与网络模型：



5.1.3 北向接口的实现（主流实现——REST API）

Rest API：SDN 北向接口的主流实现方式。实现 Rest API 的控制器 有 RYU、Floodlight、Opendaylight 等

其他方案：RPC、JAVAAPI、CORBA、SOAP 等

5.2 REST API 及其设计

5.2.1 REST API 的设计

HTTP 动词+URI，其中，HTTP 动词描述操作，URI 是标识资源。

5.2.2 HTTP 动词

HTTP 动词	描述
HEAD	获取某个资源的头部信息
GET	获取资源
POST	创建资源
PATCH	更新资源的部分属性
PUT	更新资源
DELETE	删除资源

5.2.3 URI 规范

A. URL 命名规范

资源命名规范：

文档（Document）类型的资源用名词单数命名

集合（Collection）类型的资源用名词复数命名

仓库（Store）类型的资源用名词复数命名

控制器（Controller）类型的资源用**动词**命名

URI 中有些字段可以是变量，在实际使用中可以按需替换，例如：

http://api.soccer.restapi.org/leagues/{leagueId}/teams/{teamId}/players/{playerId}

其中：leagueId、teamId、playerId 是变量（数字，字符串等类型都可以）。

B. URI 格式规范

URI 中分隔符“/”一般用来对资源层级的划分，”/“不应该出现在 URI 的末尾；

URI 中尽量使用连字符"-"代替下划线"_"的使用；

URI 中统一使用小写字母；

URI 中不要包含文件（或脚本）的扩展名；

CRUD（增删改查）的操作不要体现在 URI 中；

URI 可以使用 query 字段，作为查询的参数补充，以表示一个唯一的资源。query 可以作为过滤条件使用，例如 GET /users?role=admin。也作为资源列表分页标示使用，例如：GET /users?pageSize=25&pageStartIndex=50。

5.2.4 HTTP 响应状态码

REST API 相关的响应状态码：

2xx：操作成功

3xx：重定向

4xx：客户端错误

5xx：服务器错误

常用状态码：

200 (“OK”) ：一般性的成功返回，不可用于请求错误返回；

201 (“Created”) ：资源被创建；

202 (“Accepted”) ：Controller 控制类资源异步处理的返回，仅表示请求已经收到；

204 (“No Content”) ：可能会出现在 PUT、POST、DELETE 的请求中；

303 (“See Other”) ：返回一个资源地址 URI 的引用，但不强制要求客户端获取该地址的状态；

400 (“Bad Request”) ：客户端一般性错误返回，其它 4xx 的错误，也可以使用 400，具体错误信息可以放在 body 中；

401 (“Unauthorized”) ：认证错误；

404 (“Not Found”) ：找不到 URI 对应的资源；

500 Internal Server Error：服务器处理请求时发生了意外；

503 Service Unavailable：服务器无法处理请求，一般用于网站维护状态。

6. 网络虚拟化&NFV

6.1 虚拟化技术

6.1.1 什么是虚拟化？

虚拟化是资源的逻辑表示，它不受物理限制的约束

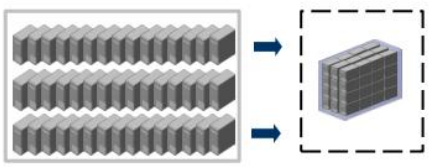
“虚拟化”的定义包含了三层含义：

虚拟化的对象是各种各样的资源；

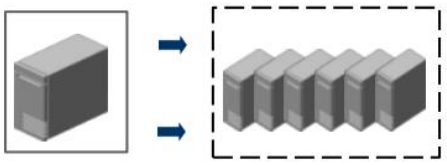
经过虚拟化后的逻辑资源对用户隐藏了不必要的细节；

用户可以在虚拟环境中实现其真实环境的部分或全部功能；

“虚拟化”的两种形式：



N:1 虚拟化



1:N 虚拟化

6.1.2 虚拟化的特点与优点

传统 1 个应用要至少 1 台服务器的弊端：虚拟化整合的优势

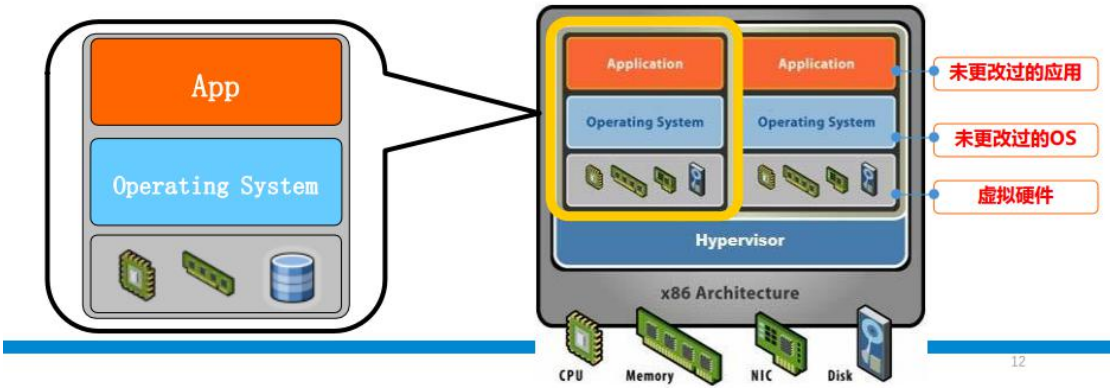
业务上线周期长：创建虚拟机和业务软件部署快捷、扩容方便

IT 资源利用率低： 通过虚拟化实现硬件资源共享

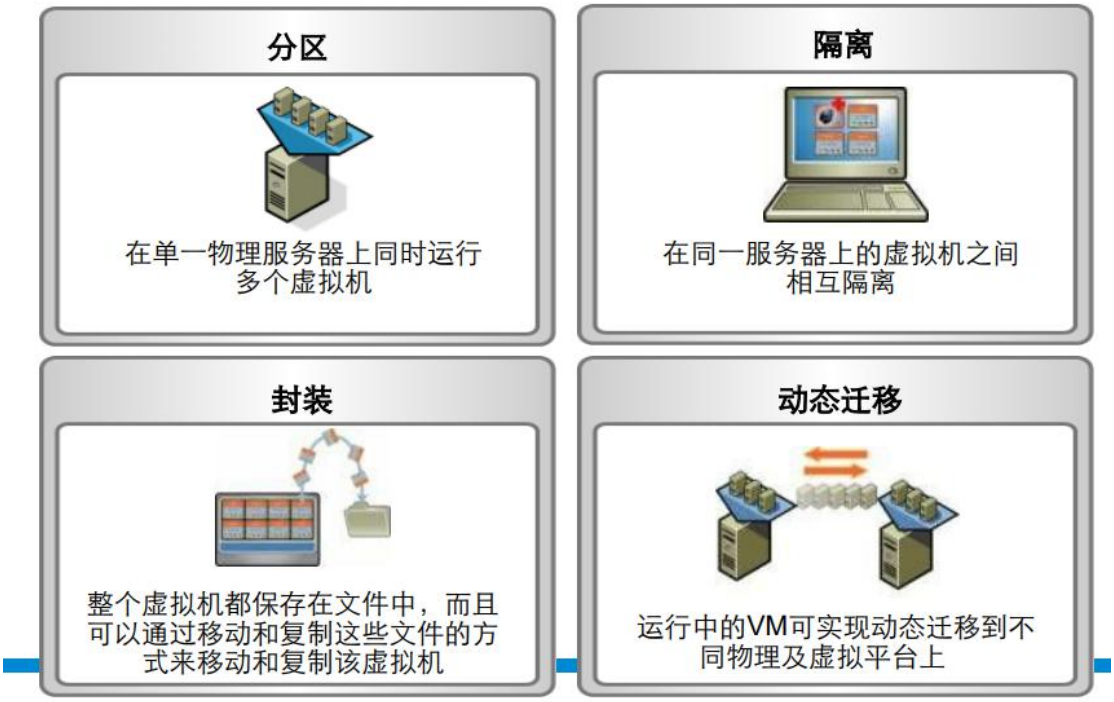
能耗高：根据负载自动迁移虚拟机，自动下电空闲服务器

管理复杂、故障恢复时间长：硬件、软件、虚拟机统一管理；虚拟机 HA 高可靠

服务器虚拟化后



服务器虚拟化的关键特性



6.2 网络虚拟化

6.2.1 网络虚拟化，vNetwork 及其组件

vNetwork 的组件主要包括虚拟网络接口卡 vNIC、vNetwork 标准交换机 vSwitch 和 vNetwork 分布式交换机 dvSwitch。

（1）虚拟网络接口卡

每个虚拟机都可以配置一个或者多个虚拟网络接口卡 vNIC，vNIC 拥有独立的 MAC 地址以及一个或多个 IP 地址，且遵守标准的以太网协议。

（2）虚拟交换机 vSwitch

虚拟交换机用来满足不同的虚拟机和管理界面进行互联。虚拟交换机的工作原理与以太网中的第 2 层物理交换机一样。

（3）分布式交换机

dvSwitch 将原来分布在一台 ESX 主机上的交换机进行集成，成为一个单一的管理界面，在所有关联主机之间作为单个虚拟交换机使用。这使得虚拟机可在跨多个主机进行迁移时确保其网络配置保持一致。

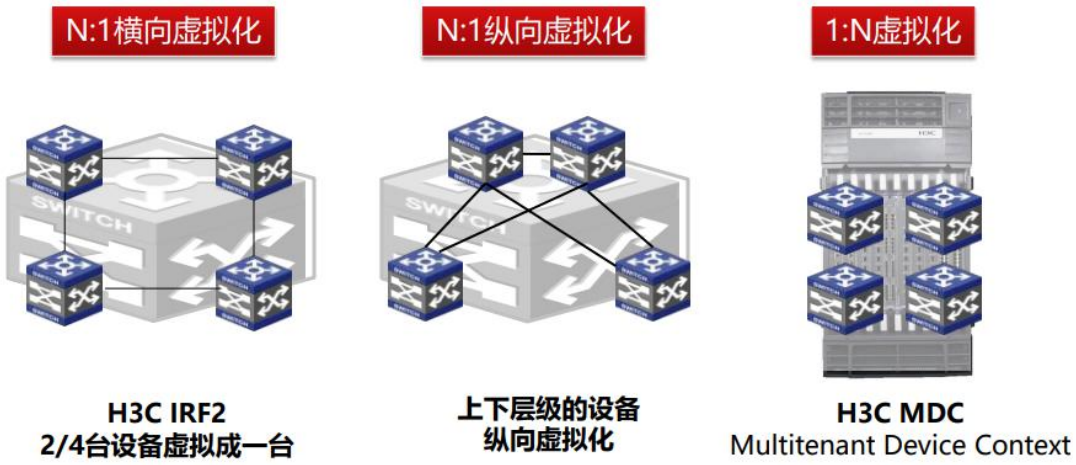
（4）端口组

端口组是一种策略设置机制，这些策略用于管理与端口组相连的网络。

（5）VLAN

VLAN 支持将虚拟网络与物理网络 VLAN 集成。

6.2.2 虚拟设备



网络资源利用率提升 100%以上，故障收敛时间减少 90%以上

6.3 网络功能虚拟化 NFV

6.3.1 NFV 的产生背景

NFV 让电信、移动提供商和运营商有能力提供更好的数字业务，加快新服务投放市场的速度，可以借此与大的软件公司竞争。NFV 提供一定程度的架构、资本、vendor-sourcing 等方面的敏捷性，与传统的基于专用运营商级网络设备的实现方式不同。

传统的通过物理设备提供服务的方式有如下缺点，大大降低了敏捷性：

部署周期和成本高，物理设备必须购买且部署这些物理设备需要大量的人力，可能需要数月的时间才能完成部署。

高运营成本，物理设备需要特殊地理环境，能源和冷却消费很高。

效率低，专用设备必须部署在能够提供最小服务覆盖的地方，不管实际的使用和需求是多少，这导致了不能充分利用和成本浪费。

厂商锁定，单一厂商的电气设备和采购其他产品的负担非常重。

6.3.2 NFV 的定义

NFV（Network Function Virtulization），即网络功能虚拟化，背景是电信网络,旨在采用虚拟化的方法，将原本运行在专用中间设备（middlebox）的网络功能（如网关、防火墙）用软件的方式实现， 通过在标准的通用设备（服务器、存储器、交换机）中运行的虚拟网络功能（VNF）得以实现。

多个 VNF 可通过动态的逻辑链接串联成业务链（Service Function Chain）。

6.3.3 NFV 框架

NFV 从纵向和横向上进行了解构，从纵向看分为三层：

基础设施层：NFVI 是 NFV Infrastructure 的简称，从云计算的角度看，就是一个资源池。NFVI 映射到物理基础设施就是多个地理上分散的数据中心，通过高速通信网连接起来。NFVI 需要将物理计算/存储/交换资源通过虚拟化转换为虚拟的计算/存储/交换资源池。

虚拟网络层：虚拟网络层对应的就是目前各个电信业务网络，每个物理网元映射为一个虚拟网元 VNF，VNF 所需资源需要分解为虚拟的计算/存储/交换资源，由 NFVI 来承载，VNF 之间的接口依然采用传统网络定义的信令接口（3GPP+ITU-T），VNF 的业务网管依然采用 NE-EMS-NMS 体制。

运营支撑层：运营支撑层就是目前的 OSS/BSS 系统，需要为虚拟化进行必要的修改和调整。

从横向看分为两个域：

业务网络域：就是目前的各电信业务网络。

管理编排域：同传统网络最大区别就是，NFV 增加了一个管理编排域，简称 MANO，MANO 负责对整个 NFVI 资源的管理和编排，负责业务网络和 NFVI 资源的映射和关联，负责 OSS 业务资源流程的实施等，MANO 内部包括 VIM，VNFM 和 Orchestrator（也称为 NFVO）三个实体，分别完成对 NFVI，VNF 和 NS（Network Service：即业务网络提供的网络服务）三个层次的管理。

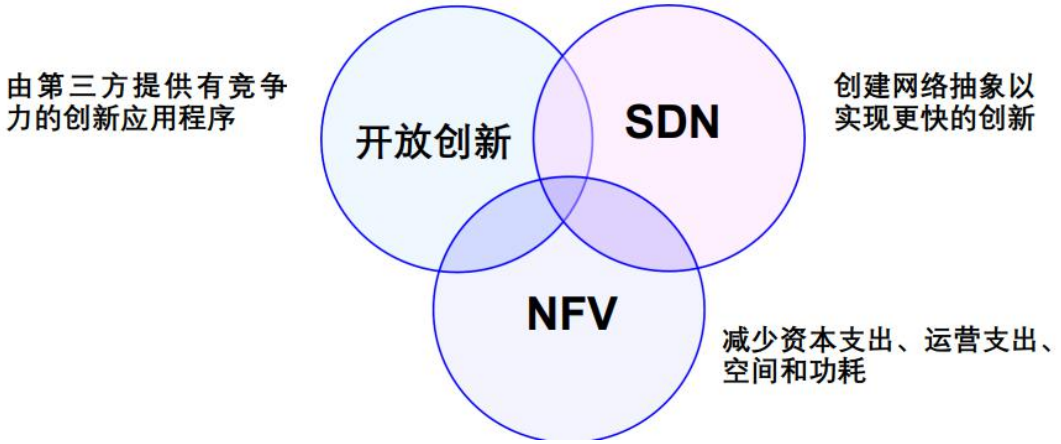
参考：NFV 基本概念 https://blog.csdn.net/xili2532/article/details/97624315

6.4 NFV 与 SDN

6.4.1 有哪些区别与联系？

NFV 和 SDN 是高度互补的

这两个主题是互利的，但不是相互依赖的



NFV：网络设备体系结构的再定义

NFV 的诞生是为了满足服务提供商（SP）的需求：

通过减少/消除专有硬件降低资本支出

将多种网络功能整合到行业标准平台上

SDN ：网络体系结构的重新定义

SDN 来自 IT 世界：

分离数据层和控制层，同时集中控制

提供使用定义良好的接口编程网络行为的能力

7. 云计算网络与 Overlay

7.1 云计算网络

7.1.1 虚拟化对传统数据中心提出的挑战

传统的三层数据中心架构结构的设计是为了应付服务客户端-服务器应用程序的纵贯式大流量，同时使网络管理员能够对流量流进行管理。工程师在这些架构中采用生成树协议（STP）来优化客户端到服务器的路径和支持连接冗余。

虚拟化从根本上改变了数据中心网络架构的需求。最重要的一点就是，虚拟化引入了虚拟机动态迁移技术。从而要求网络支持大范围的二层域。从根本上改变了传统三层网络统治

数据中心网络的局面。

7.2 Overlay 网络

7.2.1 Overlay 技术的由来

Overlay 的网络架构是物理网络向云和虚拟化的深度延伸。

随着企业业务的快速扩展需求，云计算可以提供可用的、便捷的、按需的资源提供，成为当前企业 IT 建设的常规形态。

在云计算中大量采用和部署的虚拟化几乎成为一个基本技术模式。部署虚拟机需要在网络中无限制地迁移到目的物理位置。

虚拟机增长的快速性以及虚拟机迁移成为一个常态性业务，传统的网络已经不能很好满足企业的这种需求。

7.2.2 Overlay 技术的定义及其特征，组成部分

A. Overlay 概述

Overlay 指一种网络架构上叠加的虚拟化技术模式，对基础网络不进行大规模修改下，以基于 IP 的网络技术为主承载应用，并与其它网络业务分离。

Overlay 主要技术路线颠覆了数据中心网络的建设模式，原有的接入层、汇聚层、核心层的三层架构逐渐演变为二层汇聚与三层网关的叶脊架构。

早期的标准 RFC3378（Ethernet in IP）就是在 IP 上的二层 Overlay 技术。基于 Ethernet over GRE 的技术，H3C 与 Cisco 都在物理网络基础上发展了各自的私有二层 Overlay

H3C：EVI（Ethernet Virtual Interconnection）

Cisco：OTV（Overlay Transport Virtualization）

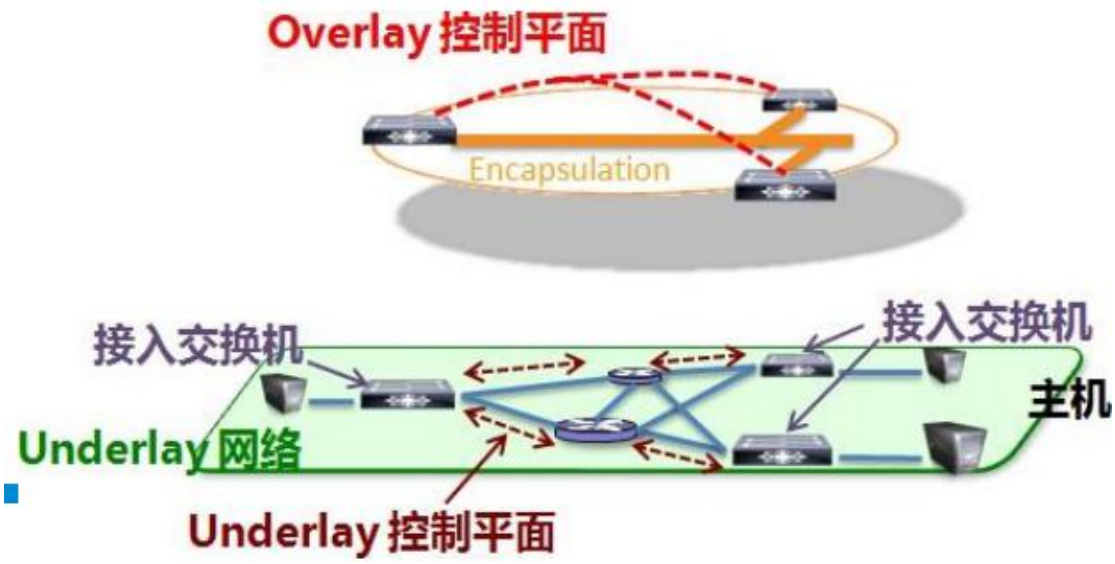
EVI 与 OTV 都主要用于解决数据中心之间的二层互联与业务扩展问题，并且对于承载网络的基本要求是 IP 可达，部署上简单且扩展方便。

参考：讲堂：Overlay 网络与物理网络的关系
http://www.h3c.com/cn/d_201501/852551_30008_0.htm

B. Overlay 无状态网络技术

未来数据中心网络发展的一个重要组成部分就是 Overlay（叠加），通过其定义的逻辑网络实现业务需要，解决数据中心云化的网络问题，将（业务的）二层网络构架在（传统网络的）三层/四层报文中进行传递的网络技术。

Overlay 是一种隧道封装技术，最关键的业务模型就是要实现一种无状态的网络模型，即使跨越运营商资源，实现多个数据中心互访，甚至虚拟机迁移都可以无感知地在这张逻辑网络上运行，对上层应用提供无感知的网络服务。



参考：Overlay 网络 https://blog.csdn.net/zhaihaifei/article/details/74340428

7.2.3 什么是 VXLAN？

VXLAN 是由 Cisco、VMware、Broadcom 等厂家向 IETF 提出一项云计算环境下，大二层网络解决方案的一项草案，全称 Virtual eXtensible Local Area Network，即虚拟扩展本地网络。

实现 Overlay 网络的三大方案之一

参考：vxlan https://baike.baidu.com/item/vxlan

7.2.4 VXLAN 的报文格式（封装与解封装，UDP header、VXLAN header）

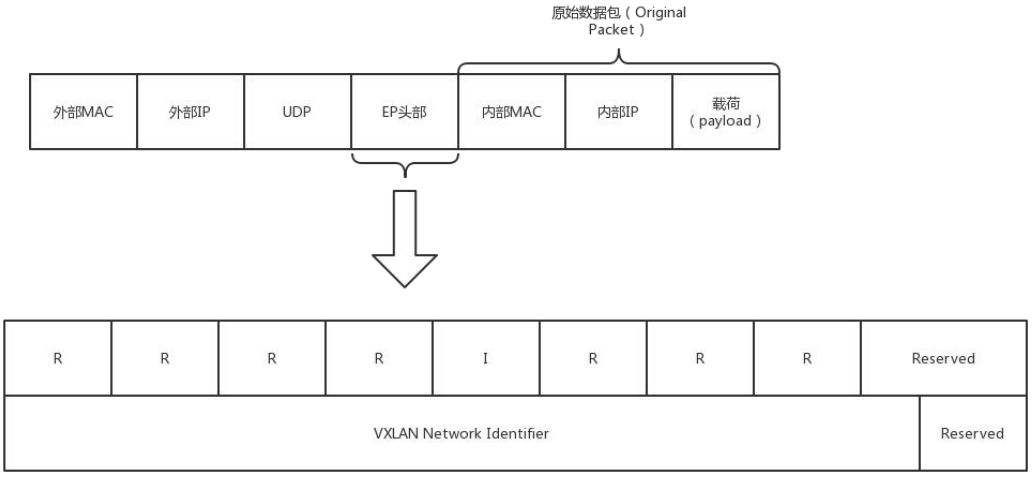
A. VXLAN 封装/解封装

VXLAN 报文是在原始二层报文前面再封装一个新的报文，新的报文中和传统的以太网报文类似，拥有源目 MAC、源目 IP 等元组。

当原始的二层报文来到 vtep 节点后会被封装上 VXLAN 包头（在 VXLAN 网络中把可以封装和解封装 VXLAN 报文的设备称为 vtep，vtep 可以是虚拟 switch 也可以是物理 switch），打上 VXLAN 包头的报文到了目标的 vtep 后会将 VXLAN 包头解封装，并获取原始的二层报文。

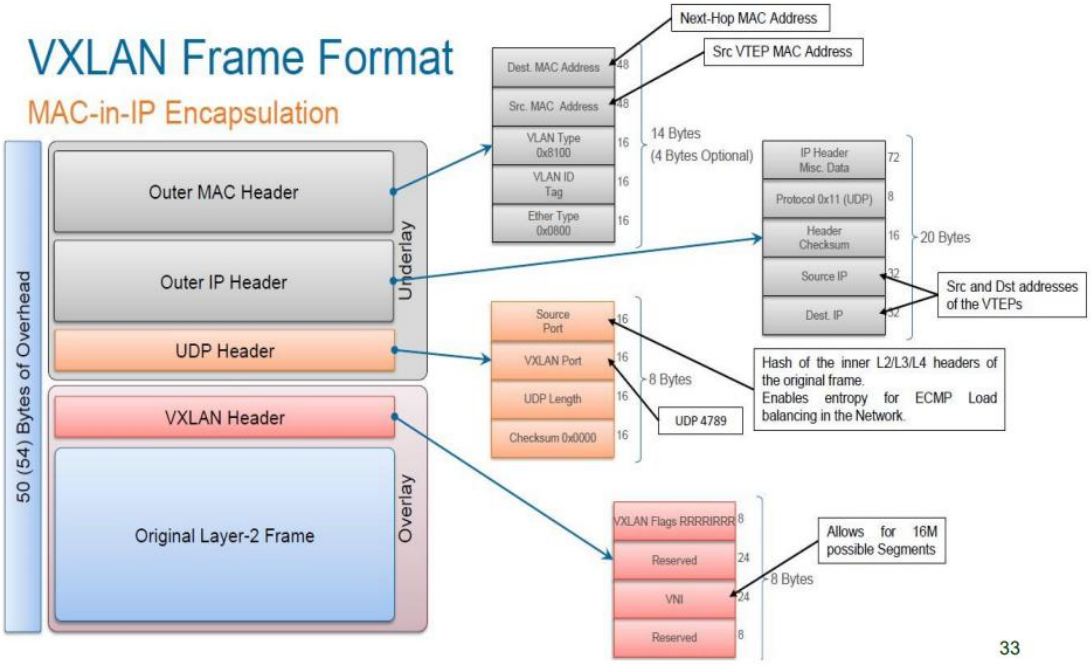
外部 MAC 头（outer MAC header）以及外部 IP 头（outer IP header）里面的相关元组信息都是 vtep 的信息，和原始的二层报文没有任何关系。

数据包在源目 vtep 节点之间的传输和原始的二层报文是毫无关系的，依靠的是外层的包头完成。



封装协议头【Encapsulation Protocol(EP) Header】，基于 VXLAN 的例子

B. VXLAN 报文结构



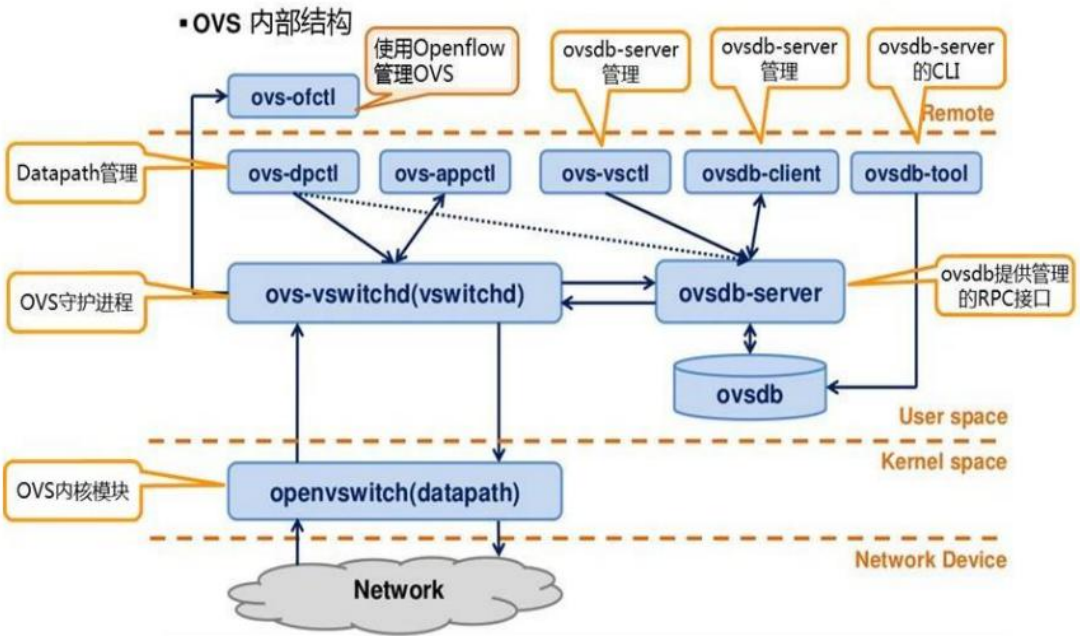
也可以说报文是 MAC in UDP

8. SDN 开源项目

8.1 OVS

8.1.1 OVS 的组成结构

Open vSwitch（OVS）组成结构

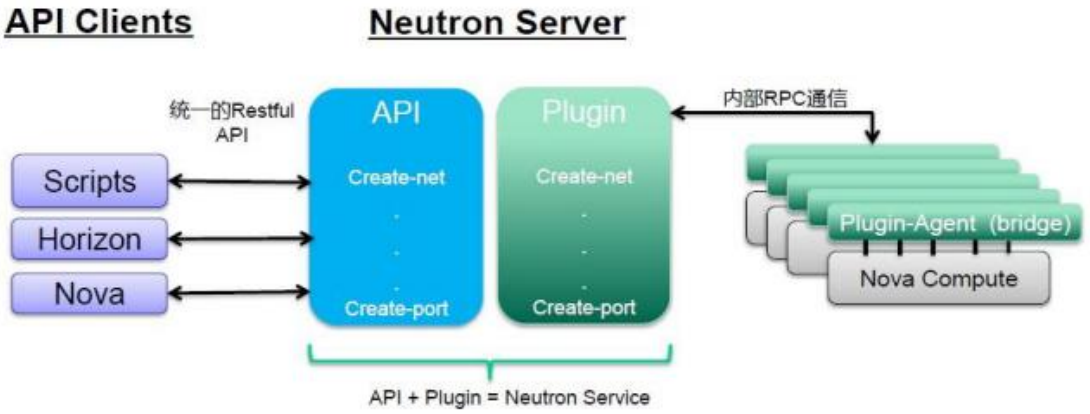


8.2 OpenStack

8.2.1 Neutron 概述

Neutron：软件定义网络框架，用于交付 NaaS 服务

对网络的虚拟化，它可以解释为一个网络管理服务，为创建和管理虚拟网络公开了一组可扩展的 API（通过创建虚拟网络为 OpenStack Compute 节点上的虚拟机提供网络服务）。



neutron-server 接受 API 请求，然后转发到对应的 plugin 进行处理

plugin 和 agent 执行实际的网络操作，例如添加端口，创建子网，绑定端口

neutron-server 和 agent 之间，需要通过消息队列来传递各种路由信息

其他参考资料

[1] 黄敏老师的课件

[2] 软件定义网络-软件学院-2021-复习提纲.docx

[3] 黄辉，施晓秋，彭达卫．软件定义网络技术[M]．北京：高等教育出版社, 2020.