中国科学技术大学
University of Science and Technology of China

OpenFlow: Enabling Innovation in Campus Networks

Nick McKeown, Tom Anderson, Hari Balakrishnan, Jonathan Turner

SIGCOMM 2008

授课教师：赵功名

中国科大计算机学院

2025年秋·高级计算机网络

Outline

I. Background

II. Software Defined Networking

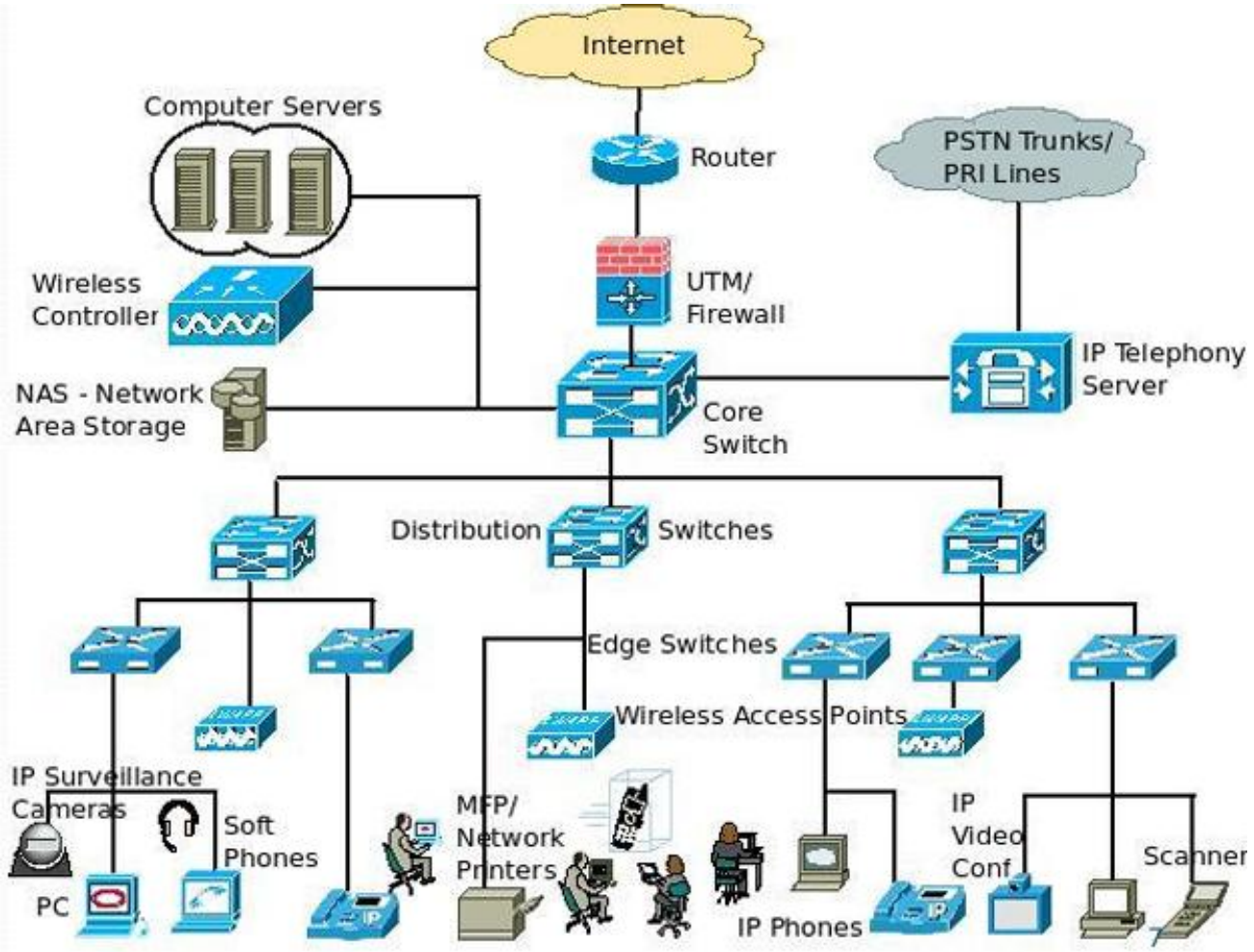
III. The OpenFlow Protocol

IV. Using OpenFlow

V. Review

The Conventional Network

- 分层的以太网交换机结构
- 如以Fat-Tree为例的网络架构：
  - 扩展性差：层级固定，增加规模需堆叠设备，复杂度和成本急剧上升
  - 灵活性不足：拓扑结构僵化，难以适应应用和流量的动态变化
  - 运维复杂：设备数量庞大，策略分散，配置与管理依赖人工



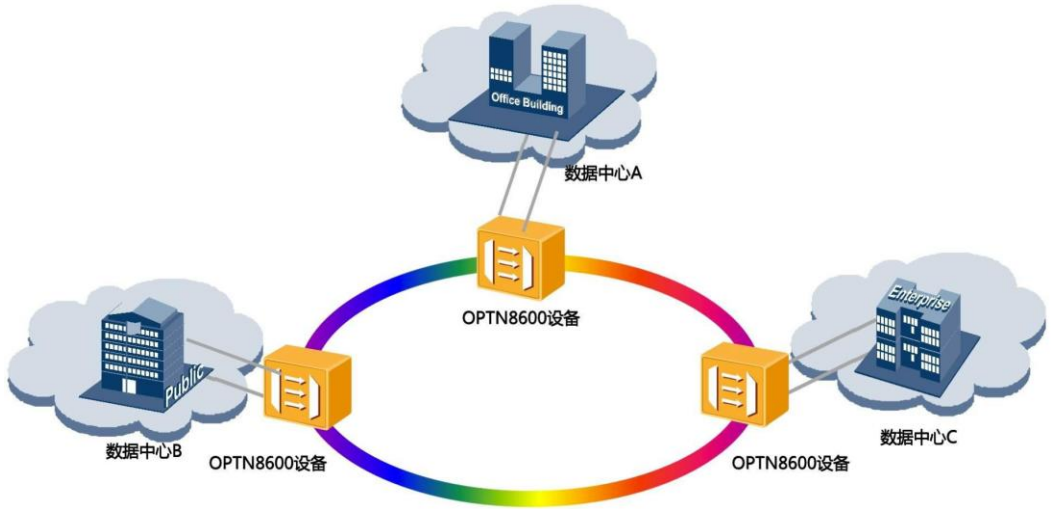
Limitations of Current Networking Technologies

- 停滞的根源：网络复杂性
  - 协议割裂：各各协议孤立设计，仅解决单一问题 → 叠加导致复杂性
  - 静态僵化：网络静态特性 vs. 服务器/应用的高度动态化（跨虚拟机分布）
  - 人工配置：QoS 等差异化服务虽可实现，但资源配置依赖繁琐手工操作

- 扩展性受限
  - 复杂配置：上百/上千设备，管理极其繁琐
  - 超大规模：Hyperscale 网络需高性能、低成本连接
  - 人工瓶颈：人工配置无法支撑规模化需求

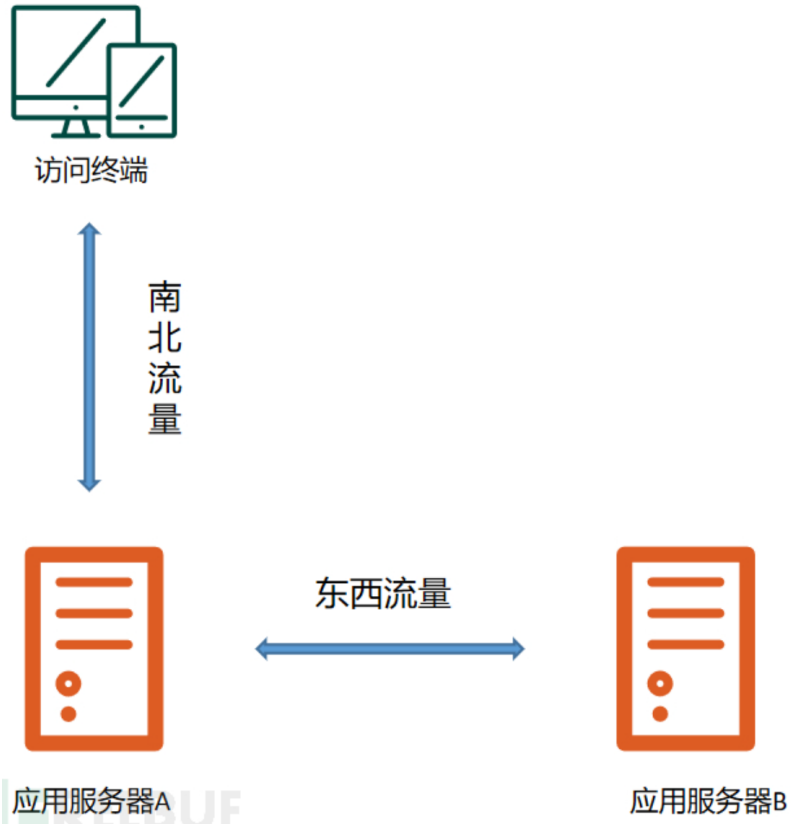
Limitations of Current Networking Technologies

- 依赖供应商
  - 部署受限：运营商/企业希望快速上线新功能，但受制于设备迭代周期（3年以上）
  - 接口封闭：缺乏标准化、开放接口 → 难以按需定制
  - 创新受阻：网络演进速度跟不上业务与用户需求变化
- 策略不一致
  - 配置繁琐：成千上万设备，耗时且复杂
  - 缺乏统一：难以推行一致的策略和规则
  - 风险增加：容易出现配置错误，影响安全与稳定



New Computing Trends

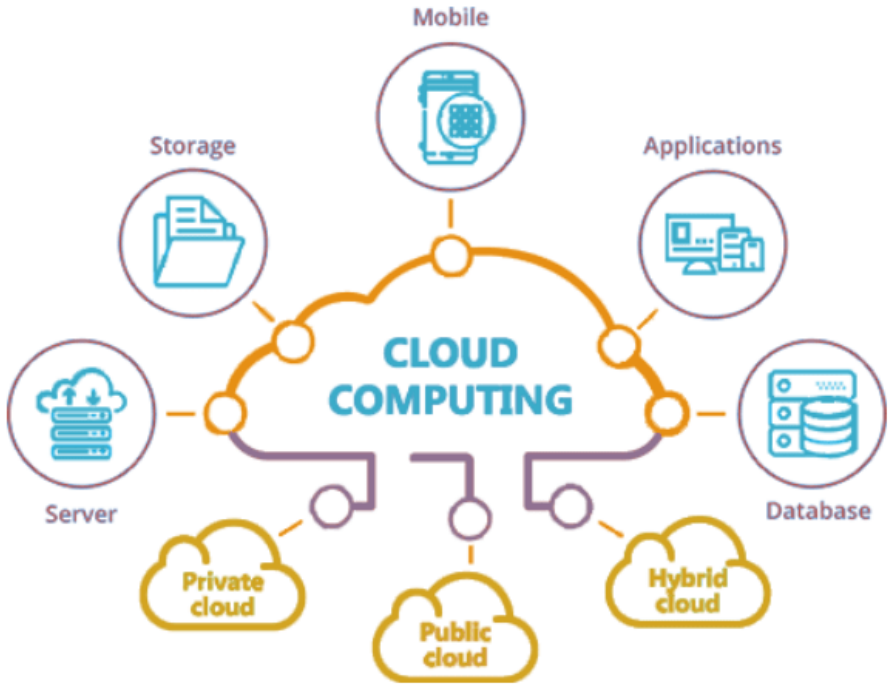
- 变化的流量模式
  - 南北向为主 → 东西向激增：应用间通信量大，先于返回用户终端产生
  - 云环境带来额外压力：私有/公有云增加跨广域网流量
  - 传统架构失配：原有设计难以应对新型流量模式



New Computing Trends

- 云服务的兴起
  - 资源弹性：计算、存储和网络资源的弹性扩展，理想情况下应通过统一的视角和一套通用的工具来实现
  - 精简运维：极少数的员工管理着非常多的机器

- “大数据”意味着需要更多带宽
  - 网络容量需求激增：海量数据集的出现正在推动数据中心对额外网络容量的持续需求



THE NEED FOR PROGRAMMABLE NETWORKS

- 网络已成为我们生活中的关键基础设施
- 但是现有的真实网络环境存在“僵化”问题
  - 网络设备已经大规模部署，并且网络管理员非常不愿意在承载着重要生产流量的设备上进行实验
  - 研究人员几乎没有可行的方法在足够真实的环境中（例如，大规模网络、真实流量）去测试新的网络协议

- 最终结果是，网络研究界的许多新想法都未经尝试和检验，导致了人们普遍认为网络基础设施已经“僵化” (ossified)

THE NEED FOR PROGRAMMABLE NETWORKS

- 解决方案的方向：可编程网络 (Programmable Networks)
- GENI项目：旨在实验新型网络架构的全国性研究设施
  - 研究人员可以被分配到整个网络资源的“切片”，包括一部分网络链路、包处理器（如路由器）和终端主机。研究人员可以随心所欲地对自己的“切片”进行编程，从而在一个隔离的环境中进行实验

- 这种虚拟化的可编程网络可以有效降低创新门槛，加速网络基础设施的革新速度

Software Defined Networking

- 通过软件定义网络（SDN）可以实现Programmable Network
  - **传统网络：**控制平面和数据平面都集成在每一台网络设备（如交换机、路由器）中。每台设备根据自己本地的信息和运行的分布式协议（如OSPF）来决定如何转发数据包。这种方式是分布式的，且**设备功能由硬件和供应商软件固化，难以更改**
  - **SDN：**将决定数据包路径的“大脑”（控制平面）从硬件设备中抽离出来，集中到一个被称为SDN控制器的中央软件中。而网络硬件设备（数据平面）则只负责根据控制器的指令执行数据包的实际转发
  - 因此，通过**编程来自动化网络配置**、快速部署新服务和自定义流量处理逻辑，可以根据业务需求的变化，通过**软件动态地、实时地调整流量路径、分配带宽和部署安全策略**，使网络能够快速适应应用的需求

Outline

I. Background

II. Software Defined Networking

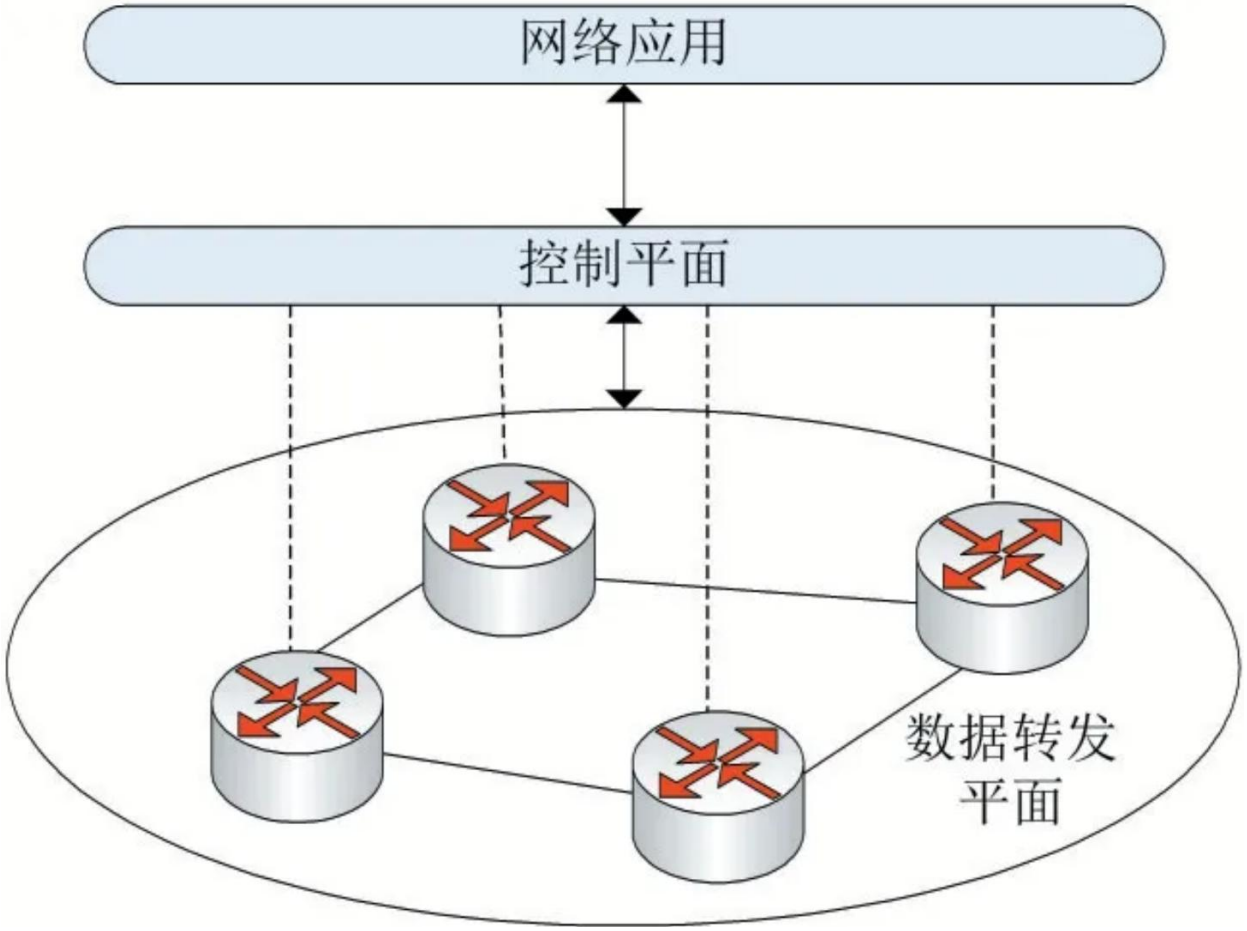
III. The OpenFlow Protocol

IV. Using OpenFlow

V. Review

Software Defined Networking

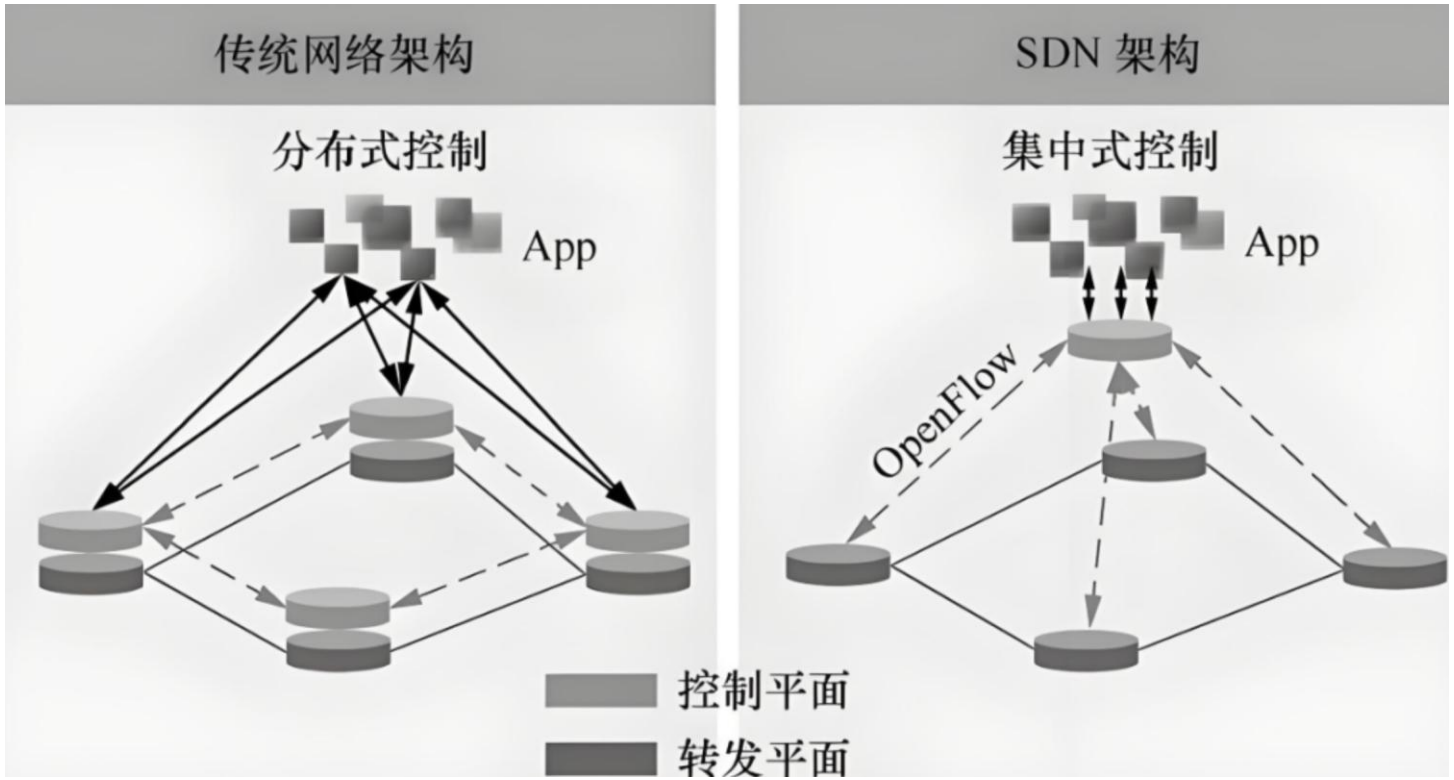
- 转控分离：将网络的控制平面与数据平面分离，使硬件设备专注于高速转发，而将智能决策交给集中控制器
  - 控制平面：拥有全局网络拓扑、实时状态和流量信息。如路径计算、访问控制策略、负载均衡策略等
  - 数据平面：根据控制器下发的“流表” 规则，对数据包进行极速匹配和转发



12

Software Defined Networking

- 集中控制：通过集中控制器掌握全网拓扑，SDN能够进行全局优化和统一策略部署，避免了传统网络中分布式控制带来的次优路径和策略不一致问题，保证了策略的即时性、准确性和一致性



13

Software Defined Networking

- 网络可编程：SDN通过开放的北向接口（API），允许上层应用对网络进行编程和自动化管理
  - 灵活性：网络的行为和策略可以根据业务需求，通过软件编程实时动态调整
  - 创新能力：任何开发者均可利用这套API编写应用程序、调用网络资源

14

SDN标准三层模型

应用层

- 应用层运行各类网络应用，如负载均衡、安全策略和流量工程等，通过北向接口与控制层交互，实现网络功能的灵活定制

控制层

- 控制层的核心是SDN控制器，它负责维护全局网络视图，计算最优路径，并通过南向接口将策略下发到数据平面

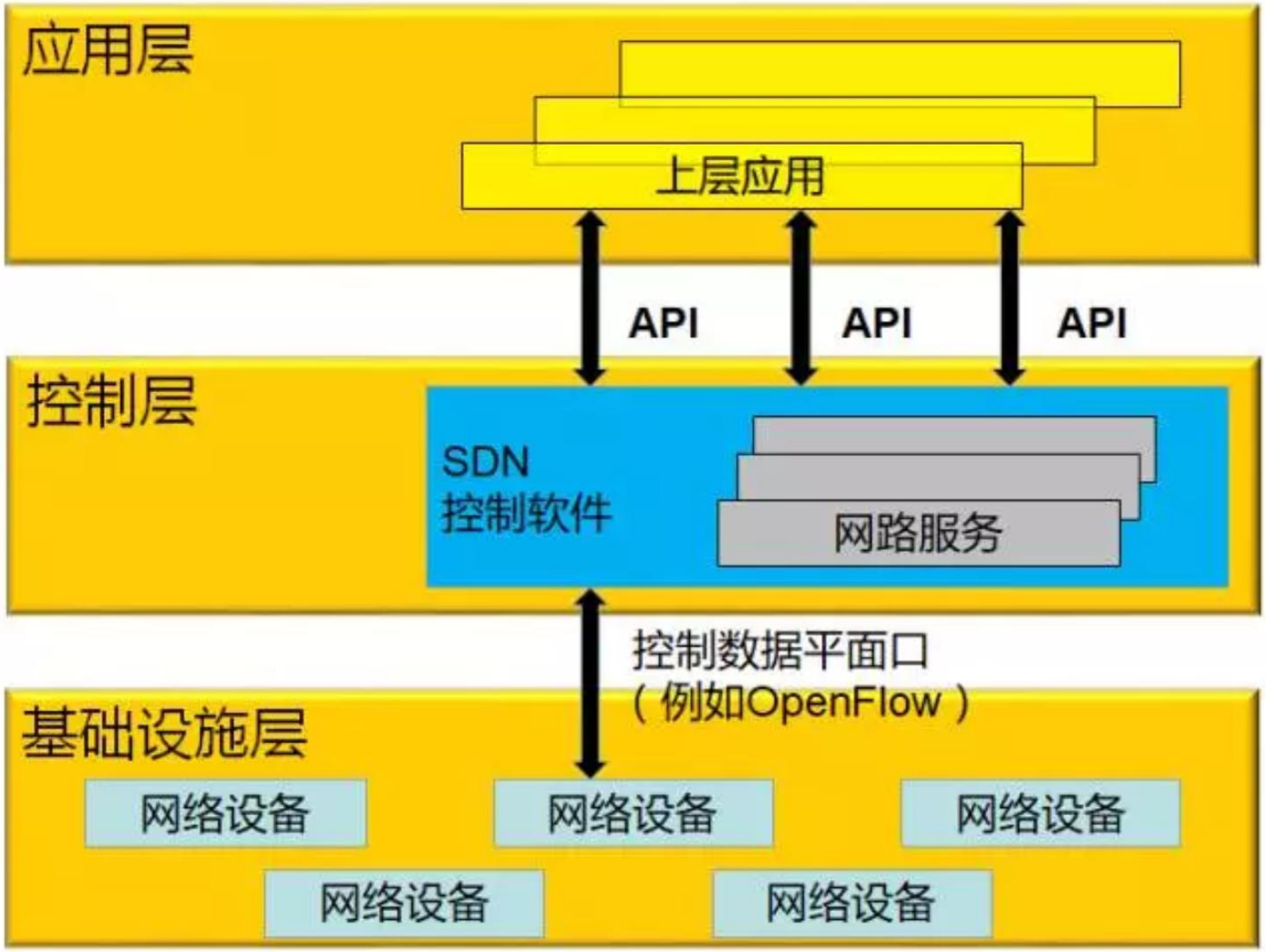
基础设施层

- 基础设施层由简单的SDN交换机组成，仅根据流表进行数据包转发，专注于高效的数据传输，无需运行复杂的路由协议

Software Defined Networking

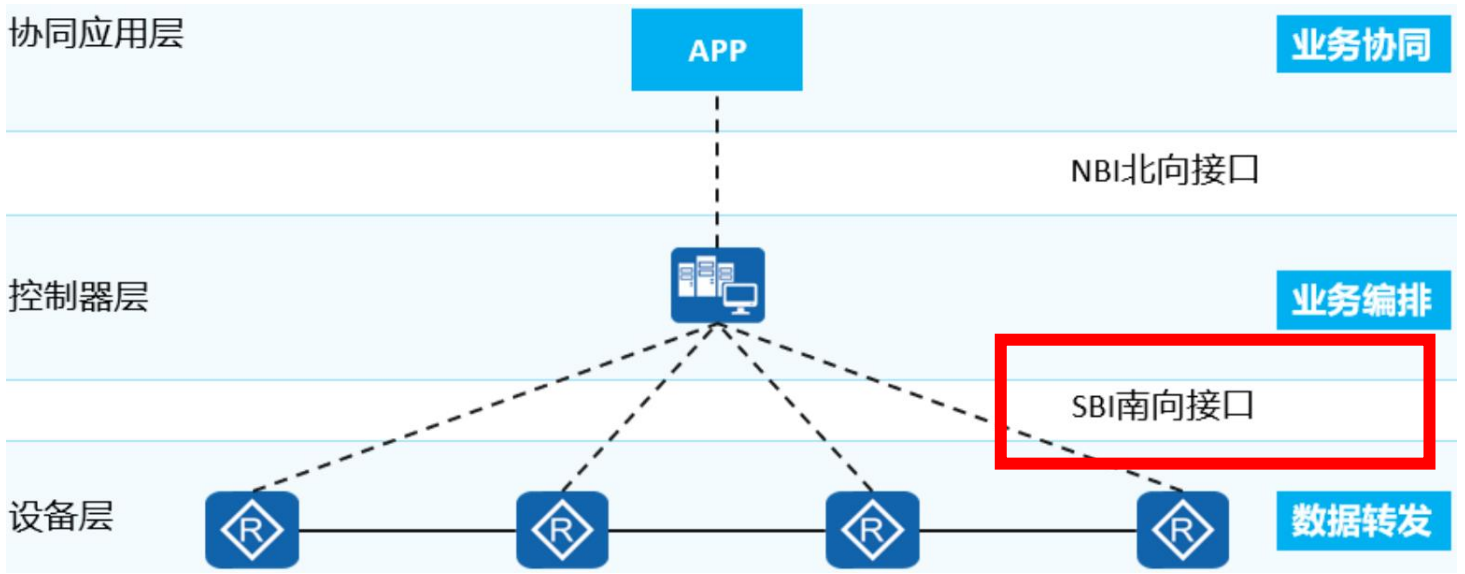
SDN标准三层模型

- 这个分层解耦的架构，使得网络的每一个层面都可以独立演进和创新，最终实现了整个网络的高度自动化、智能化和灵活性
  - **基础设施层**负责执行
  - **控制层**负责决策
  - **应用层**负责意图



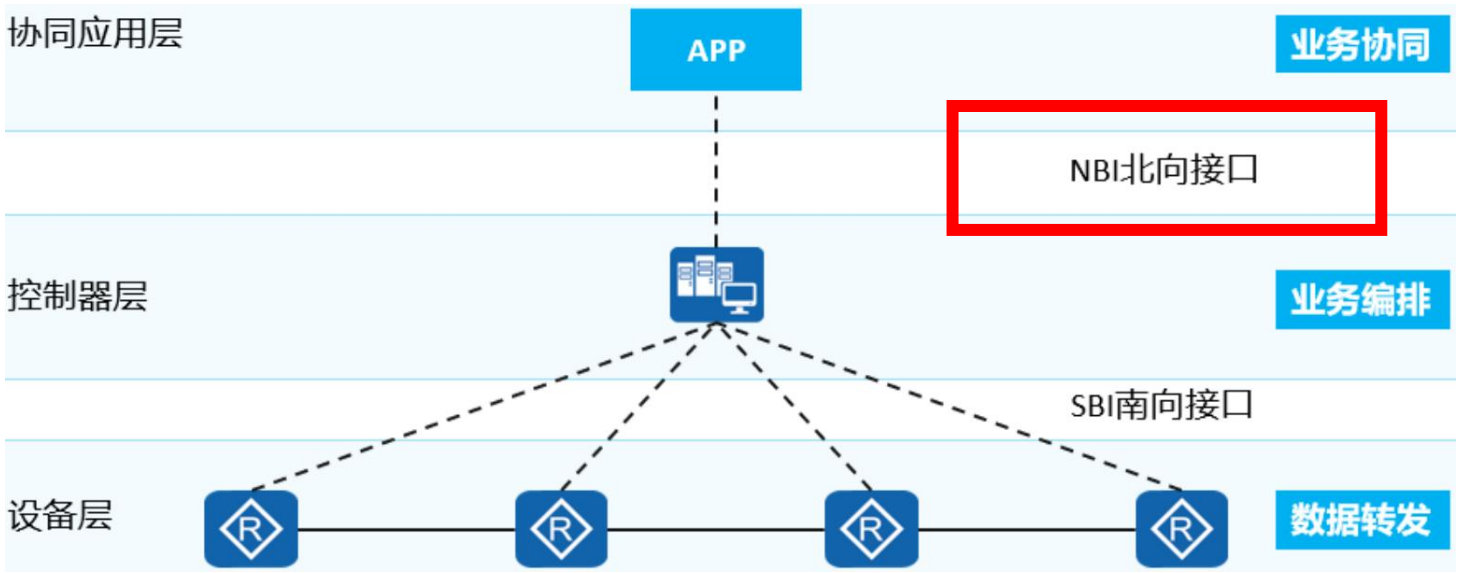
南向接口(Southbound Interface, SBI)

- 南向接口是控制指令的通道：它的核心作用是将控制器抽象的决策，翻译成网络硬件可以理解和执行的低级语言。控制器通过它来发现网络拓扑、下发流表规则、获取设备状态和统计信息
  - 趋向标准化：为了实现不同厂商设备之间的互操作性，南向接口的标准化至关重要
  - 关注底层细节： 协议内容涉及硬件能力、端口状态、流表结构等物理和逻辑细节



北向接口 (Northbound Interface, NBI)

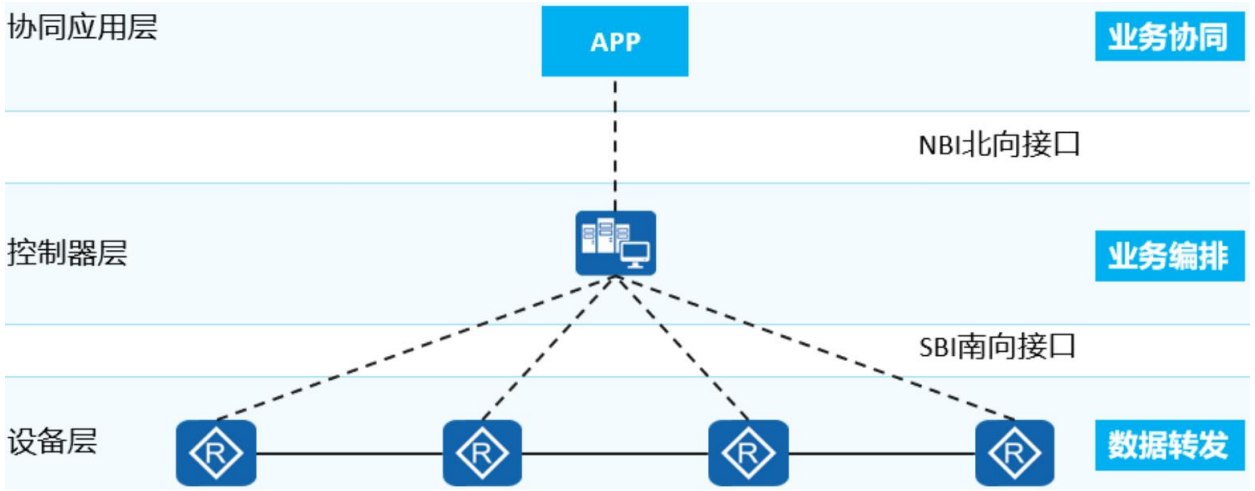
- 北向接口是网络能力的开放窗口：它将复杂的底层网络拓扑和资源抽象成简单、易用的服务，供上层应用以编程方式调用。这是实现“网络可编程”和自动化的关键
  - REST API：绝大多数SDN控制器都提供基于HTTP的REST API，使用特定请求查询网络状态
  - 关注业务逻辑： 接口的设计和服务直接面向业务需求，如租户管理、安全策略、负载均衡等



Software Defined Networking

SDN将网络从一个封闭的、功能固化的“黑盒”，转变成了一个开放的、可编程的“平台”

- 从“网络管理员”到“网络开发者”
  - 网络运营商和管理员可以通过编程方式配置这个简化的网络抽象
  - 可以自己编写程序，而无需等待新功能被嵌入到供应商专有的封闭软件中
- SDN 架构支持一组 API，使得实现常见的网络服务成为可能
  - 路由、多播、安全、访问控制、带宽管理、流量工程、服务质量等，都可以根据业务目标进行定制



19

Software Defined Networking

SDN将网络从一个封闭的、功能固化的“黑盒”，转变成了一个开放的、可编程的“平台”

- 从“功能堆砌”到“服务软件化”
  - **路由：**可以实现一种比OSPF更适合数据中心“东西向流量”的路由算法
  - **安全：**当防火墙、入侵检测系统（IDS）检测到攻击时，它不再只是简单告警，而是可以直接调用API实时地、精准地拦截恶意流量
  - **流量工程：**可以实时监控所有链路的带宽使用情况，当发现某条链路即将拥塞时，主动将新建立的“大象流” 引导到其他空闲链路上
  - **服务质量 (QoS)：**可以开发一个针对特定业务的QoS应用。当识别到对应业务的流量时，自动调用API提升其在网络中的转发优先级，保证流畅体验

20

Outline

I. Background

II. Software Defined Networking

III. The OpenFlow Protocol

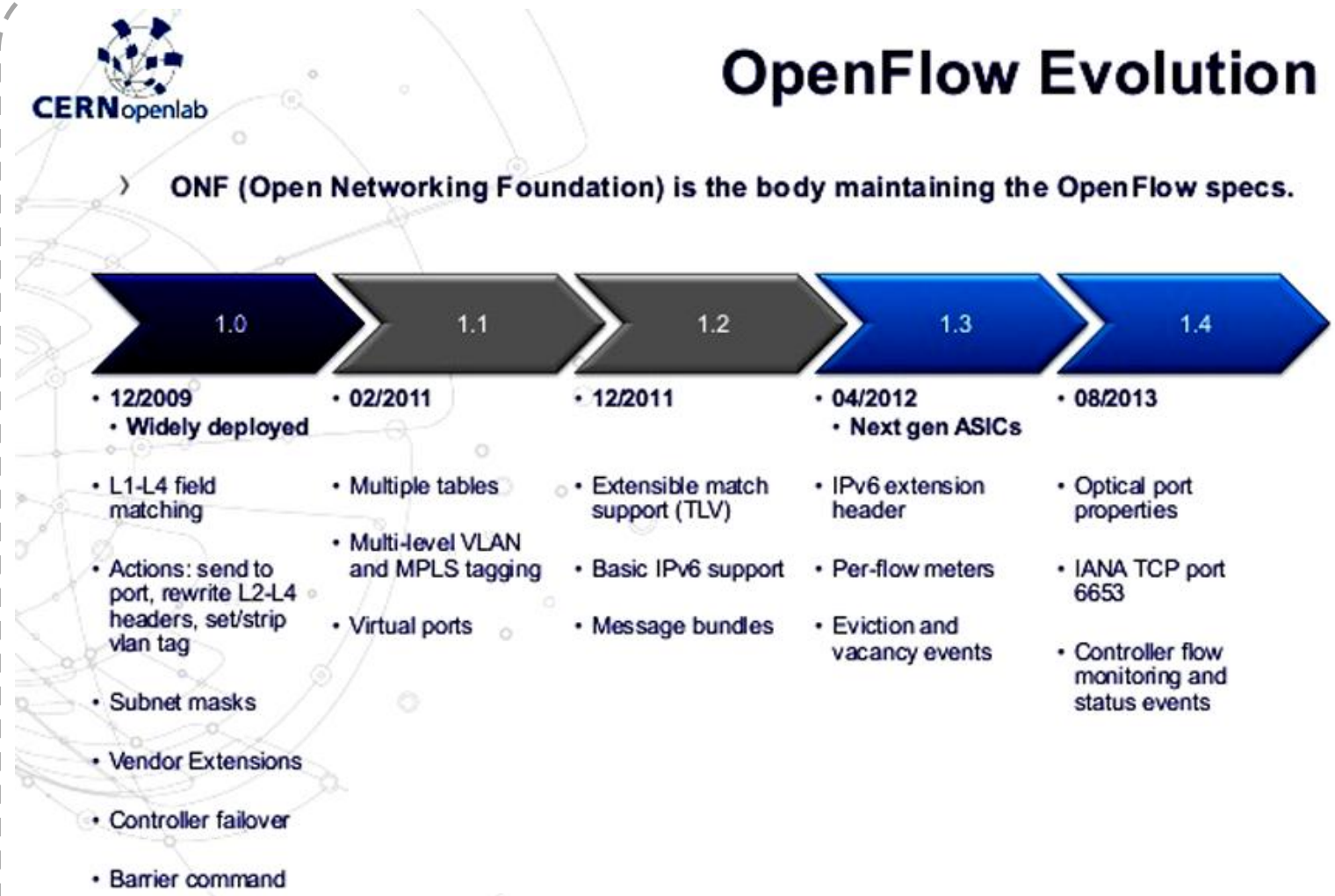
IV. Using OpenFlow

V. Review

About OpenFlow

- OpenFlow
  - 它在软件定义网络（SDN）架构下，规范控制平面和数据平面之间通信方式的一种协议
  - 一种使交换机能够作为 OpenFlow 交换机运行的规范
- 由开放网络基金会提出（ONF）
  - 白皮书：《软件定义网络：网络领域的全新标准》

About OpenFlow



- OpenFlow v1.5（2014 年 8 月发布）
- OpenFlow v1.6（2016 年 4 月发布）
- OpenFlow v1.7（2017 年 3 月发布）
- OpenFlow v1.8（2017 年 11 月发布）
  - 支持匹配数据包的 PBB 标签
  - 支持数据平面交换机间的多路径协调，以便实现负载平衡和高可用性
  - 支持控制平面的网络拓扑发现和状态感知功能
  - 支持对控制器的负载平衡和故障转移，以便实现高可用性
  - 支持对数据平面的安全性和可靠性进行评估和控制

OpenFlow Switches

- OpenFlow 提供了一个开放协议，用于对不同交换机和路由器中的流表进行编程
- 一个 OpenFlow 交换机至少包含三个部分：
  - **流表 (Flow Table)**：每个流表项都关联一个动作，用于告知交换机如何处理该数据流
  - **安全通道 (Secure Channel)**：用于将交换机连接到远程控制器，允许命令和数据包在控制器和交换机之间传输
  - **OpenFlow 协议 (The OpenFlow Protocol)**：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

24

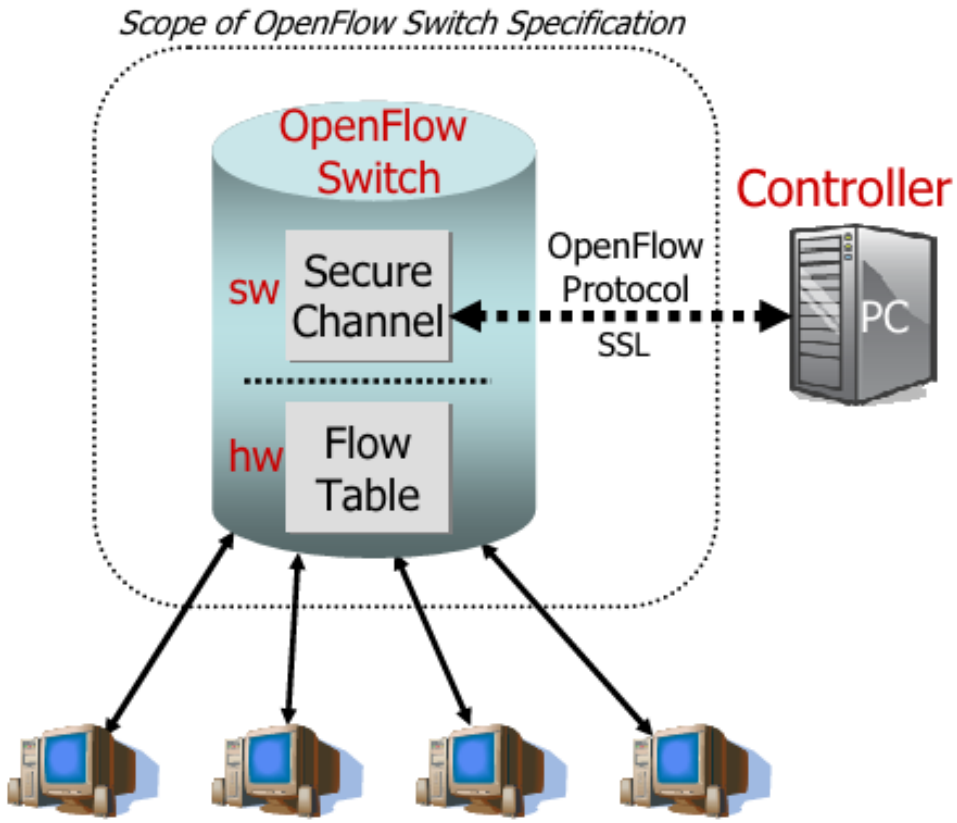
OpenFlow Switches

- 有必要将交换机分为两类：
  - 一类是专用的OpenFlow交换机（Dedicated OpenFlow switches），是一种不具备传统二层和三层处理功能的、简单的数据路径元件
  - 另一类是支持OpenFlow协议的通用商用以太网交换机（OpenFlow-enabled switches）和路由器，它们在原有功能的基础上增加了OpenFlow协议和相关接口

25

Dedicated OpenFlow switches

- Dedicated OpenFlow switches是一种简单的网络数据转发设备，它根据远程控制进程的指令在端口之间转发数据包：
  - 它不进行自主的二层（MAC学习）或三层（IP路由）处理 。所有的转发决策都由外部的控制器来定义
  - 它的唯一功能是根据远程控制器的指令在端口之间转发数据包

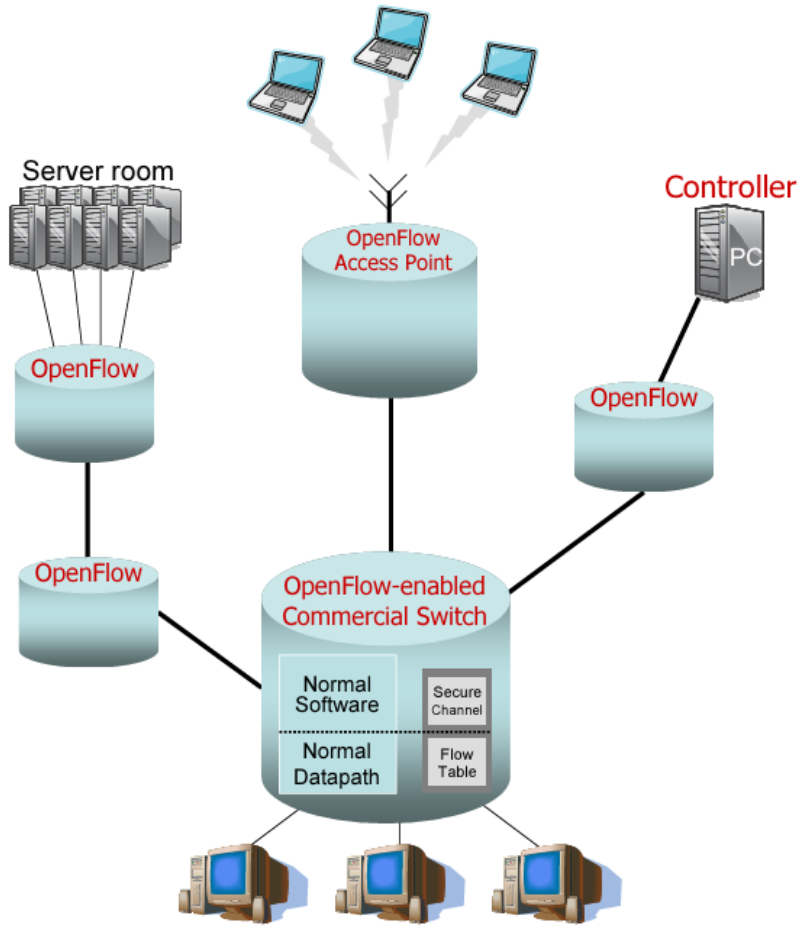


- Dedicated OpenFlow switches：流量表由远程控制器通过安全通道进行控制

26

OpenFlow-enabled switches

- 支持OpenFlow的交换机能够同时处理正常的生产流量和实验性的OpenFlow流量：
  - OpenFlow协议允许交换机同时由两个或多个控制器控制，以提高性能和可靠性
  - 支持OpenFlow的交换机必须将实验流量（由流表处理）与正常的二层和三层业务流量（由交换机的标准处理流程处理）隔离



- 一个由支持OpenFlow协议的商用交换机和路由器组成的网络示例

27

OpenFlow Switches

- 什么是流 (flow)?
  - 一个流可以是一个 TCP 连接，或者来自特定 MAC 或 IP 地址的所有数据包，或者带有相同 VLAN 标签的所有数据包，或者来自同一交换机端口的所有数据包
  - 简单来说，就是一系列具有相同特征、可被识别的数据包
- 每个流表项都有一个与之关联的简单动作 (action)
  - **转发 (Forward)**：将此流的数据包发送到一个或多个指定的端口
  - **送交控制器 (Packet-In)**：将此流的数据包上报给控制器
  - **丢弃 (Drop)**：丢弃此流的数据包
  - **修改 (Modify)**：修改数据包的某些字段

28

OpenFlow Switches

- 流表中的一个条目包含三个字段：
  - 用于定义流的包头 (header)
  - 定义应如何处理数据包的动作 (action)
  - 记录每个流的数据包和字节数，以及自上次数据包匹配以来时间的统计信息 (Statistics)
- 一个 10 元组的包头：

In Port	VLAN ID	Ethernet			IP			TCP	
		SA	DA	Type	SA	DA	Proto	Src	Dst

29

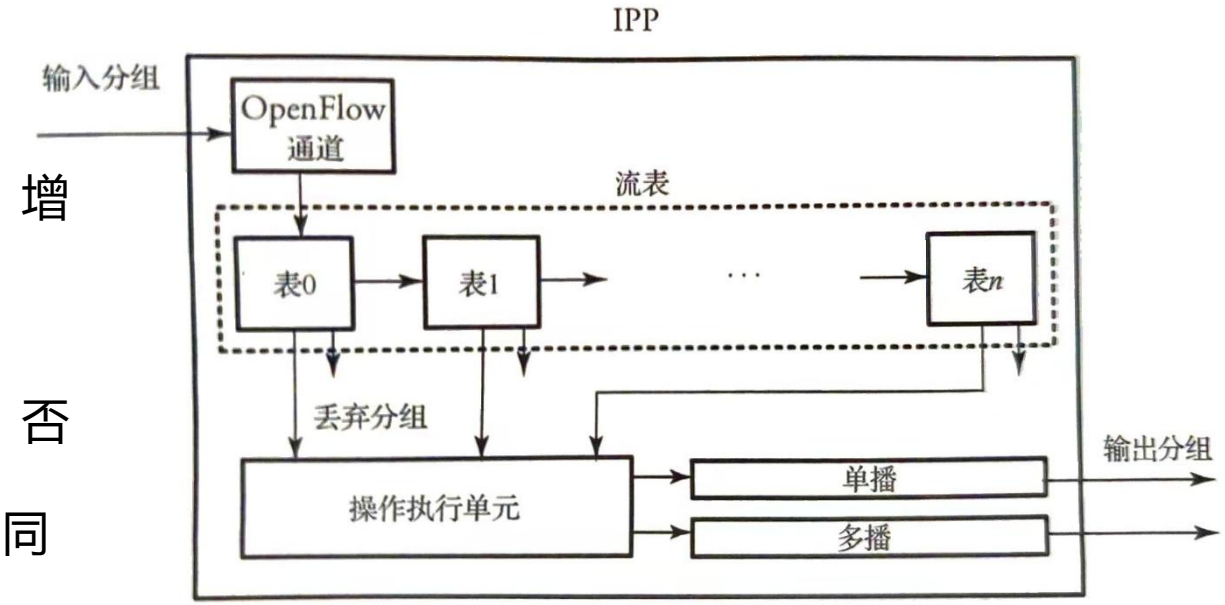
OpenFlow Switches

- OpenFlow交换机的输入端口处理器（IPP）
- OpenFlow交换机的数据平面并非一个简单的、巨大的查找表，而是被设计成一个由多个流表串联而成的**流水线**
  - 数据包进入
  - 解析与提取
  - 进入流水线起点，根据OpenFlow规范，数据包的处理必须从第0号流表 (Table 0) 开始
  - 在当前流表中查找，寻找匹配的流表项
  - 如果在当前表中找不到任何可以匹配的规则，交换机将执行该表的 **(Table-miss)** 规则，一旦找到优先级最高的匹配规则，交换机就会执行该规则中定义的指令集

30

OpenFlow Switches

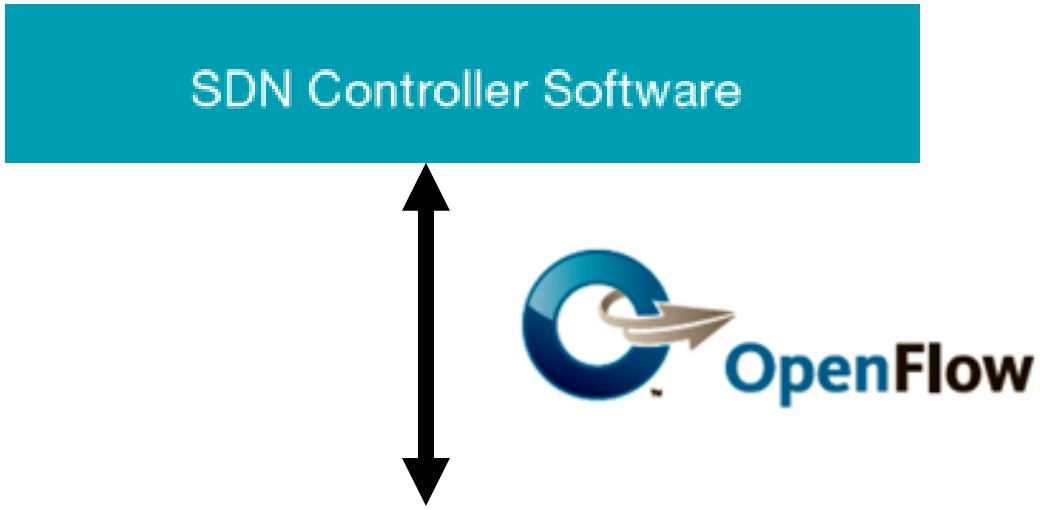
- OpenFlow交换机的输入端口处理器（IPP）
  - 每个输入的分组首先通过一个 OpenFlow 通道到达流表，然后用表 0 评估分组状态
  - 该表检查是否与到达的分组匹配。找到匹配项后，增加流计数器并执行相应的指令集
  - 如果找到完全匹配，则将分组送到操作执行单元。否则就将分组依顺序送到下一个表，执行与表0中相同的处理。最后将在操作执行单元中处理分组所携带的累积匹配项
  - 分组最终是通过单播还是多播发送取决于分组要求的模式



31

OpenFlow Controller

- SDN控制器软件通过OpenFlow协议与支持OpenFlow的网络设备进行通信。设备内的流表类似于一个指令集，根据匹配到的规则执行相应动作



OpenFlow-enabled Network Device

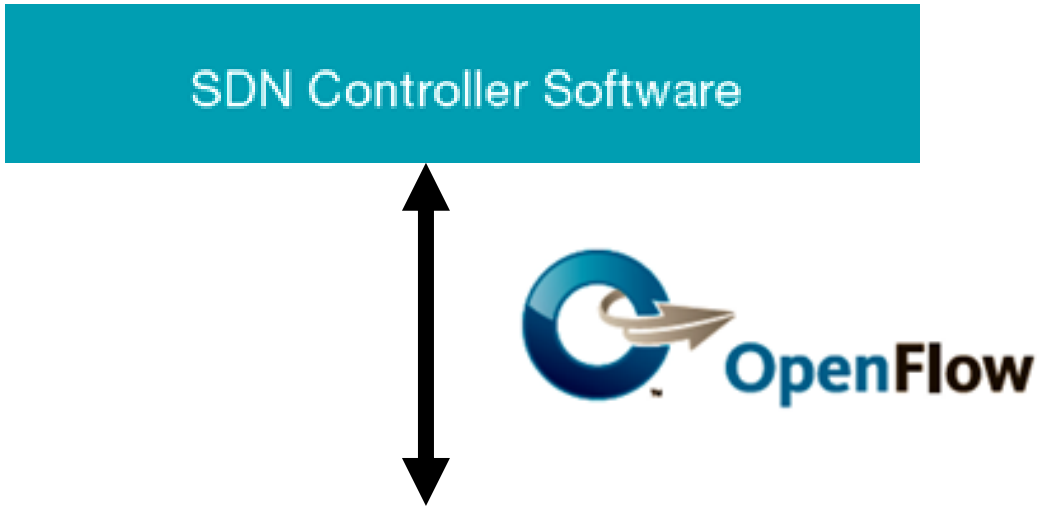
Flow Table comparable to an instruction set

MAC src	MAC dst	IP Src	IP Dst	TCP dport	...	Action	Count
*	10:20:.	*	*	*	*	port 1	250
*	*	*	5.6.7.8	*	*	port 2	300
*	*	*	*	25	*	drop	892
*	*	*	192.*	*	*	local	120
*	*	*	*	*	*	controller	11

32

OpenFlow Controller

- 流表可类比为一个指令集
- 表格展示了流表的示例，包含匹配字段（MAC地址、IP地址、TCP端口等）、动作（转发到端口1、端口2、丢弃、本地处理、送至控制器）和计数



OpenFlow-enabled Network Device

Flow Table comparable to an instruction set

MAC src	MAC dst	IP Src	IP Dst	TCP dport	...	Action	Count
*	10:20:.	*	*	*	*	port 1	250
*	*	*	5.6.7.8	*	*	port 2	300
*	*	*	*	25	*	drop	892
*	*	*	192.*	*	*	local	120
*	*	*	*	*	*	controller	11

33

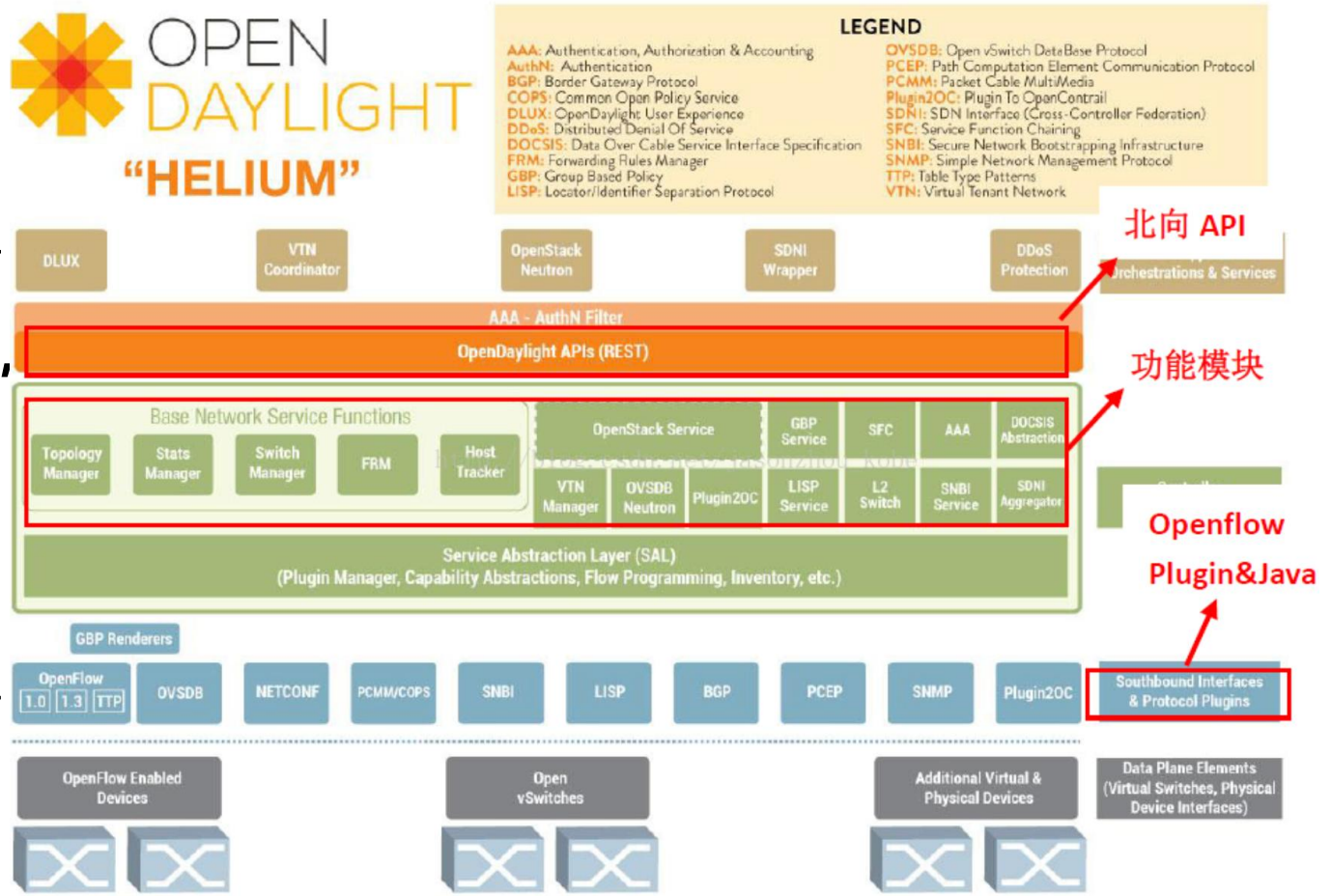
OpenFlow Controller

- SDN 控制器必须负责如下事项：
  - 为网络中所有设备计算最小代价路径
  - 根据网络策略更新底层网络设备的流表项
  - 将网络策略转换成分组转发规则
  - 建立到各个网络设备的连接
  - 更新网络设备中的分组转发规则
  - 应对到达的分组入（packet - in ）事件和设备加入（device – join）事件
  - 对 packet - in 事件的设备设置相关转发规则
  - 与 device - join 事件的设备建立新连接
  - 把流量引导到其他路径后,关闭链路或交换机以节省能源
  - 执行负载均衡策略，特别是在靠近数据中心的位置

34

OpenFlow Controller

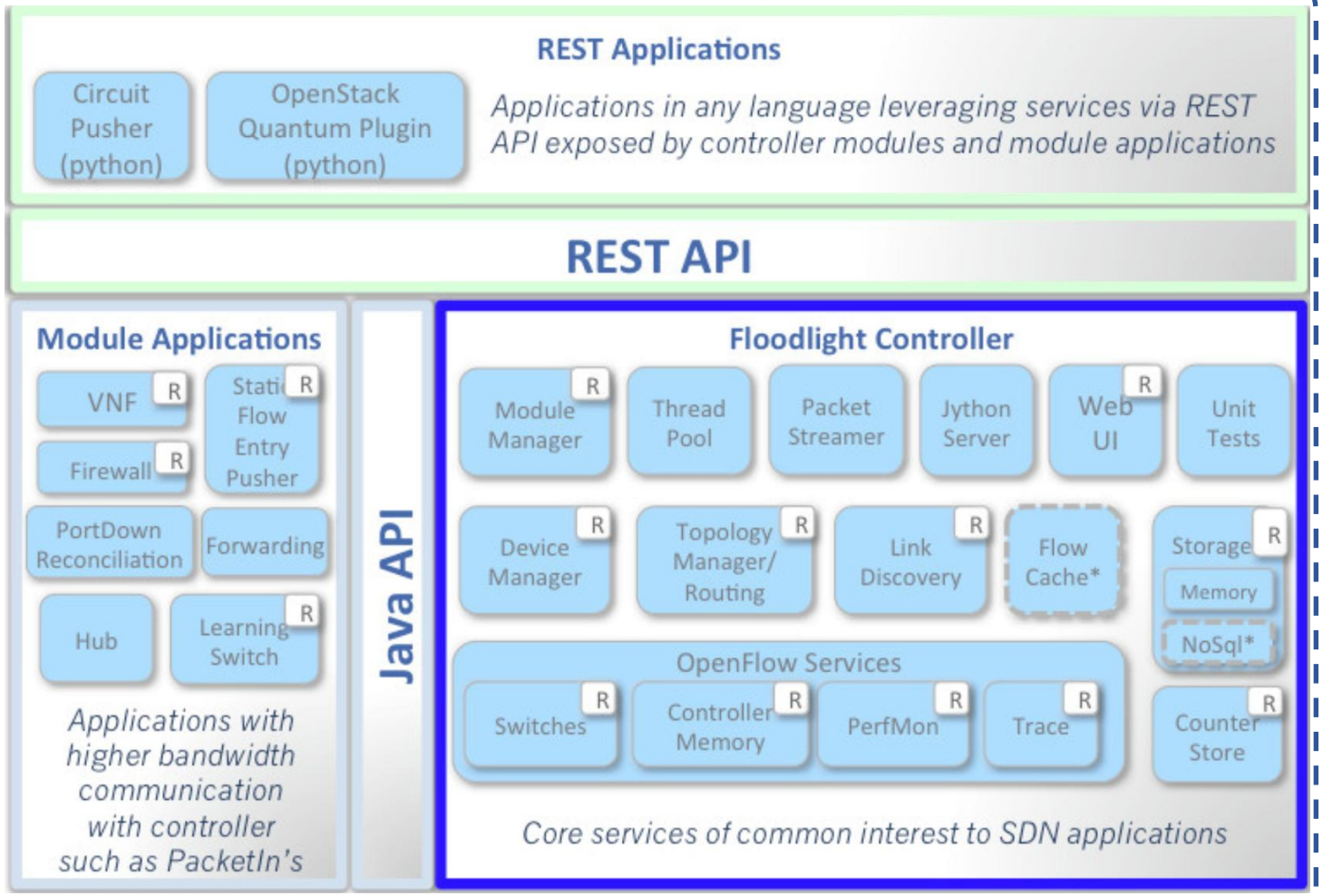
- OpenDaylight 控制器
  - OpenDaylight 架构分为南向接口层、控制平面层、北向接口层和网络应用层
  - 该控制器的核心是**服务抽象层（SAL）**，它将内部和外部的服务查询映射到适当的南向插件上
  - 该控制器还能引入用于构建和跟踪覆盖网络的虚拟化管理



35

OpenFlow Controller

- Floodlight 控制器
  - 该控制器的定位是一个企业级的、基于Apache开源协议的、纯粹的OpenFlow控制器
  - 所有核心模块（如设备管理、拓扑管理、路由模块）在启动时加载，并通过Java内部接口直接调用，效率很高
  - 由于其轻量级和紧密集成的架构，其处理性能非常出色

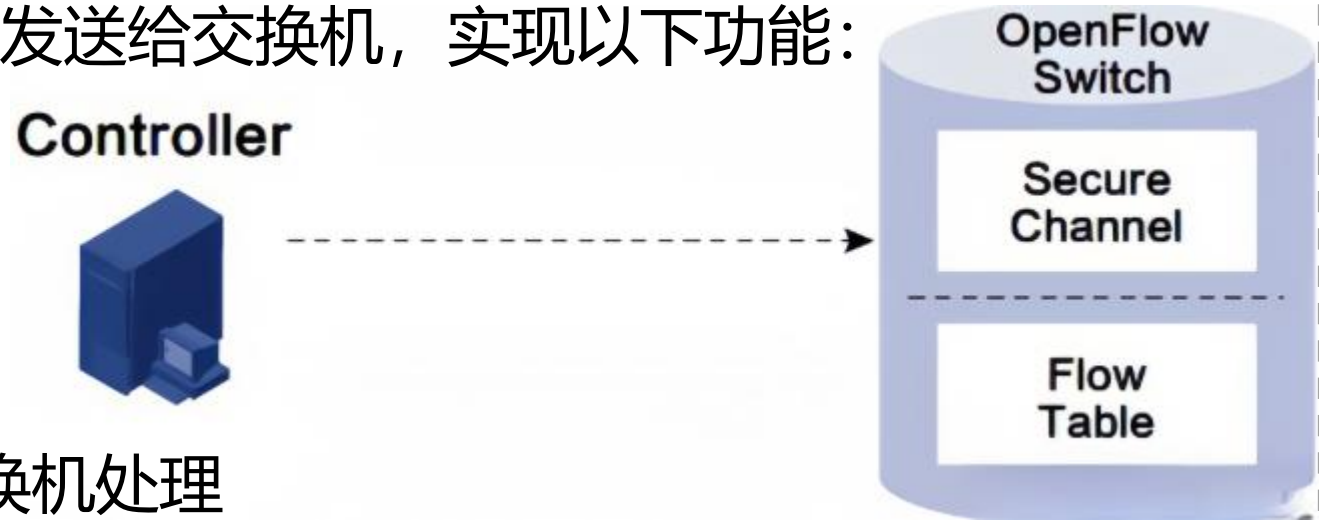


36

OpenFlow 消息

- OpenFlow协议可以生成 3 种类型的消息：
  - 控制器到交换机消息
  - 交换机到控制器消息
  - 混合消息

- 控制器到交换机消息由控制器产生并发送给交换机，实现以下功能：
  - 指定或修改流表
  - 请求交换机能力的特定信息
  - 读取交换机中某个流的信息
  - 定义新的流后，将分组返回给交换机处理



37

OpenFlow 消息

- 交换机到控制器消息由控制器产生并发送给交换机，实现以下功能：
  - 将与不匹配流的分组发送给控制器
  - 通知控制器因计时器过期而丢弃了一个流
  - 通知控制器交换机错误或端口变化

- 混合消息由交换机或控制器发出，有 3 种类型：
  - Hello消息：是控制器与交换机之间交换的启动消息
  - Echo消息：用于检测控制器到交换机连接的存活性
  - Experimenter消息：为OpenFlow技术提供扩展

38

Benefit of Openflow-based SDN

- 对多厂商环境的集中控制
- 通过自动化降低复杂性
- 更快的创新速度
- 提高网络可靠性和安全性
  - 可确保访问控制、流量工程、服务质量、安全和其他策略在有线和无线网络基础设施中得到一致的执行
- 更精细的网络控制
  - 从按地址块控制 -> 到按流控制
- 更好的用户体验
  - 例如，自动带宽检测以适配视频分辨率

39

Outline

I. Background

II. Software Defined Networking

III. The OpenFlow Protocol

IV. Using OpenFlow

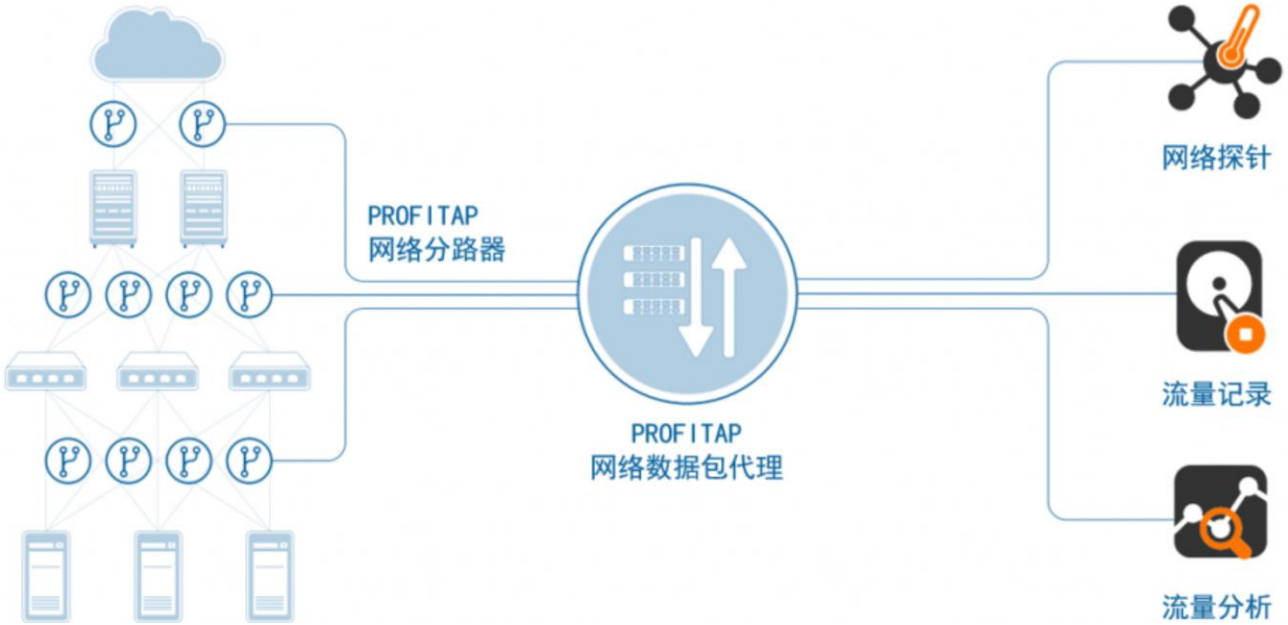
V. Review

Using OpenFlow

- Amy-OSPF 示例
  - **定义初始流：**研究员Amy首先定义一个初始规则，将来自她自己电脑的所有流量（即从她连接的交换机端口进入的流量）捕获
  - **发送至控制器：**该规则的动作是“封装并转发所有数据包到控制器”
  - **控制器决策：**当Amy电脑发出第一个数据包后，它会被发送到控制器。控制器上运行着Amy的新路由协议“Amy-OSPF”。该协议会为这个新的应用流选择一条通过OpenFlow交换机网络的路径
  - **下发流表：**控制器沿着选定的路径，在每一台交换机的流表（Flow Table）中添加一个新的流条目
  - **线速转发：**完成设置后，该应用流的后续所有数据包在到达交换机时，都能被流表快速匹配并以线速处理，无需再经过控制器

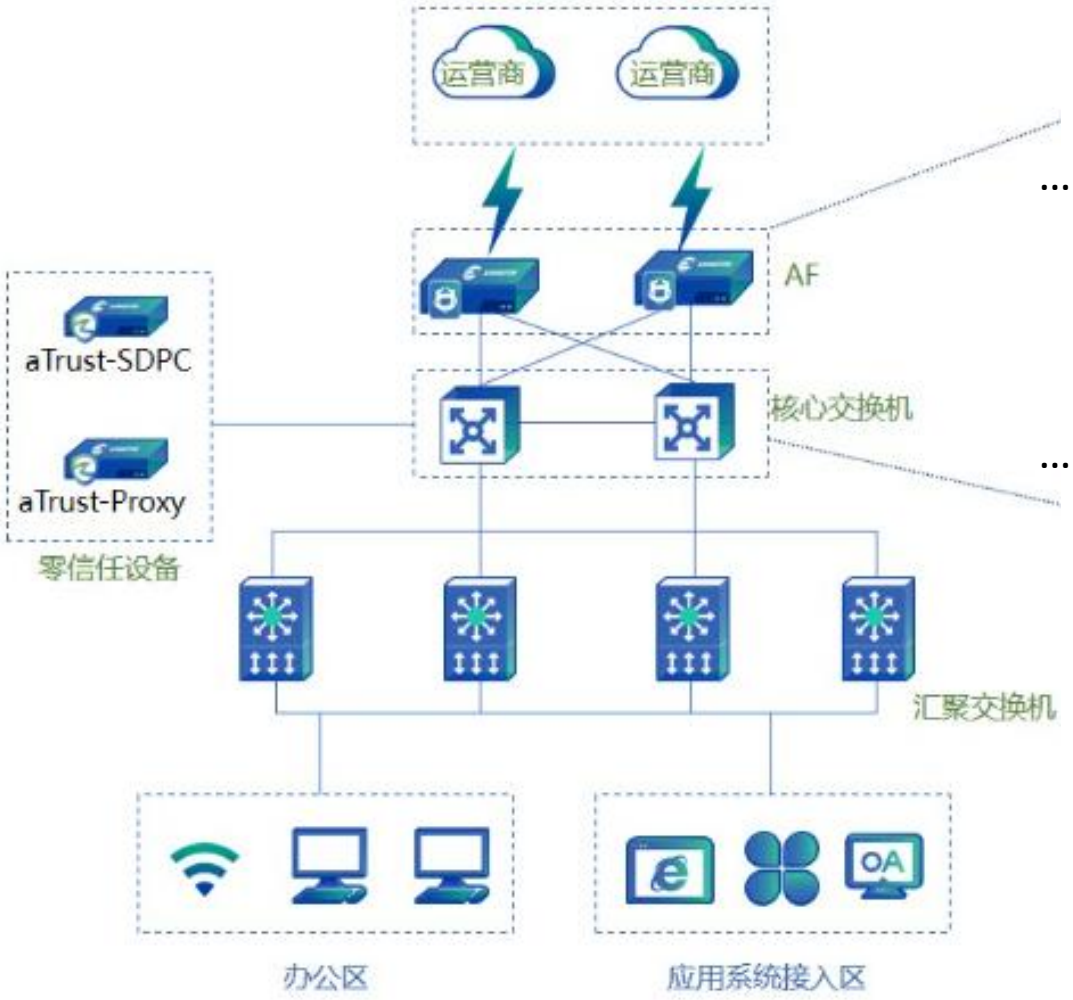
Using OpenFlow

- Amy-OSPF 示例
- 为了让实验不影响网络中的其他用户，必须保证两个关键属性：
  - 隔离生产流量：不属于Amy实验的其他用户的流量，必须由交换机内置的标准、经过测试的路由协议（即通过交换机正常的处理流水线）来处理
  - 权限控制：Amy应该只能为她自己的流量，或者网络管理员授权她控制的流量添加流表条目。这个属性的实现依赖于控制器的权限管理机制



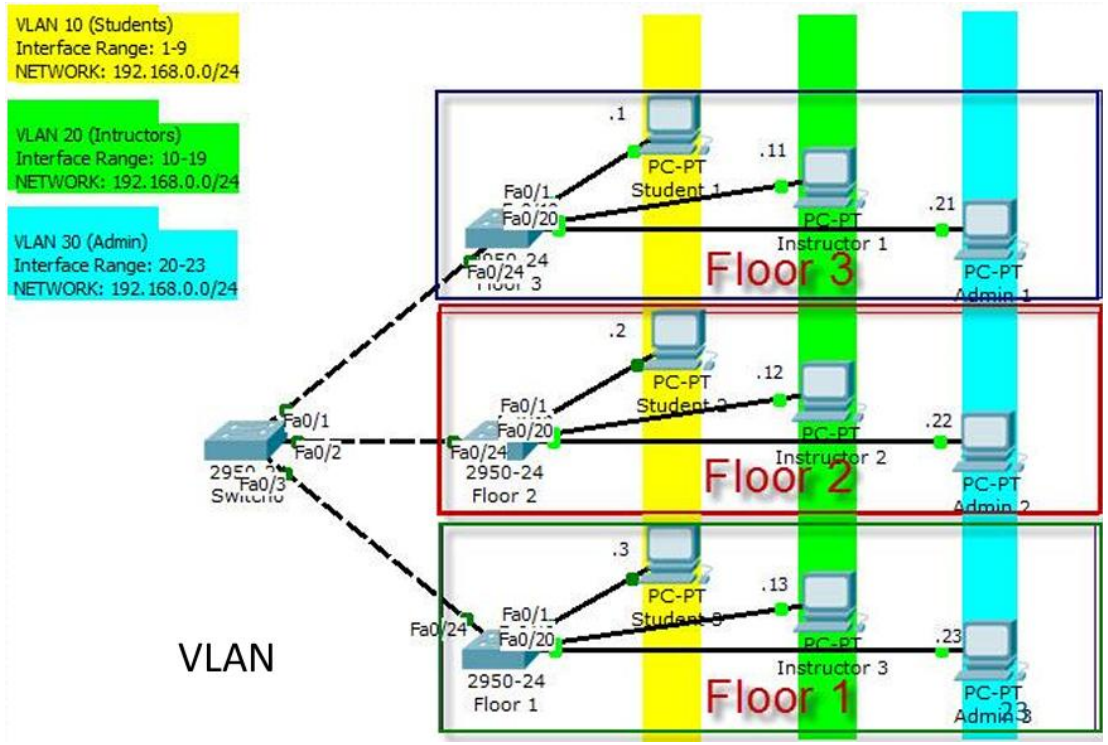
More Example

- 示例 1：网络管理和访问控制
  - Ethane 项目：其基本思想是允许网络管理员在中央控制器中定义全网范围的策略，该策略通过对每个新产生的流进行准入控制决策来直接执行
  - 控制器通过管理所有名称和地址之间的绑定关系，将数据包与其发送者关联起来——它基本上接管了 DNS、DHCP 的功能，并在所有用户加入时对其进行身份验证，同时跟踪他们连接到哪个交换机端口（或接入点）



More Example

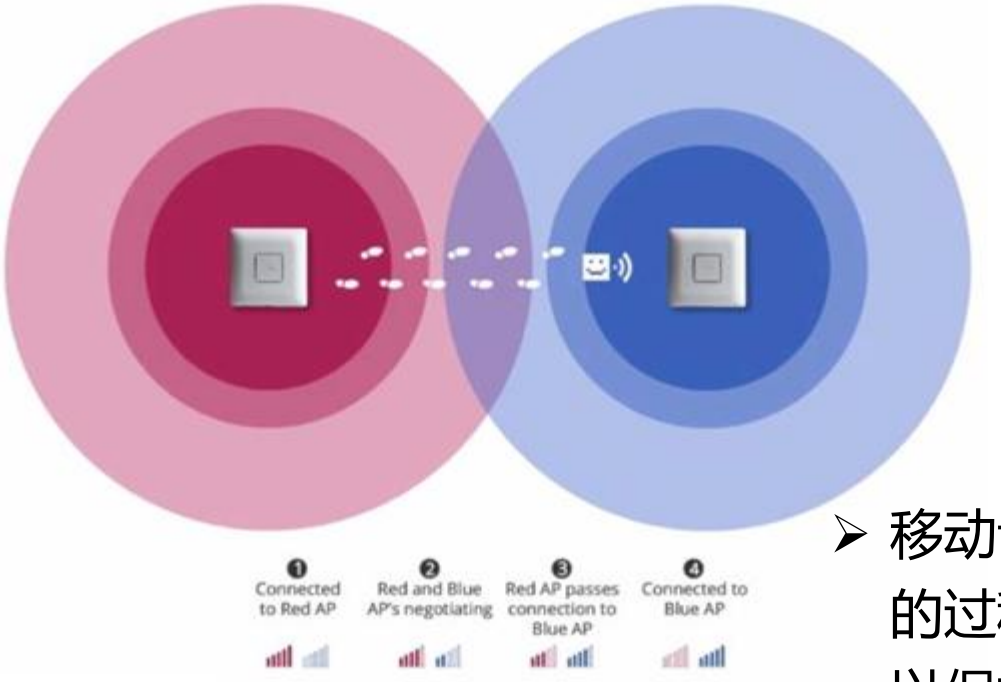
- 示例 2： 虚拟局域网（VLANs）
  - 最简单的方法是静态声明一组流，用于指定特定 VLAN ID 的流量可以访问哪些端口
  - 一种更动态的方法是使用控制器来管理用户身份验证，并利用用户的位置信息在运行时为流量打上标签

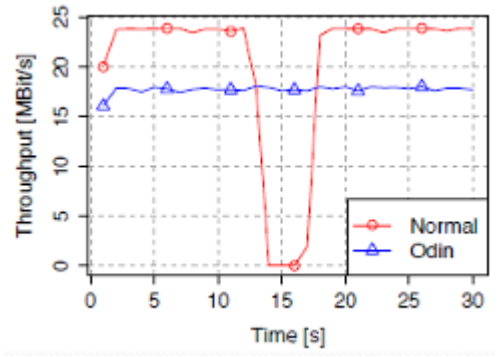


- 如图所示可以在多层楼宇中部署VLAN的场景，不同楼层的学生、教师和管理员设备被划分到不同的VLAN中

More Example

- 示例 3：移动无线 VoIP 客户端
  - 为支持 Wi-Fi 的电话提供通话切换机制
  - 通过实现一个控制器来跟踪客户端的位置，当用户在网络中移动时，通过重新编程流表来重新路由连接，从而实现从一个无线接入点到另一个的无缝切换

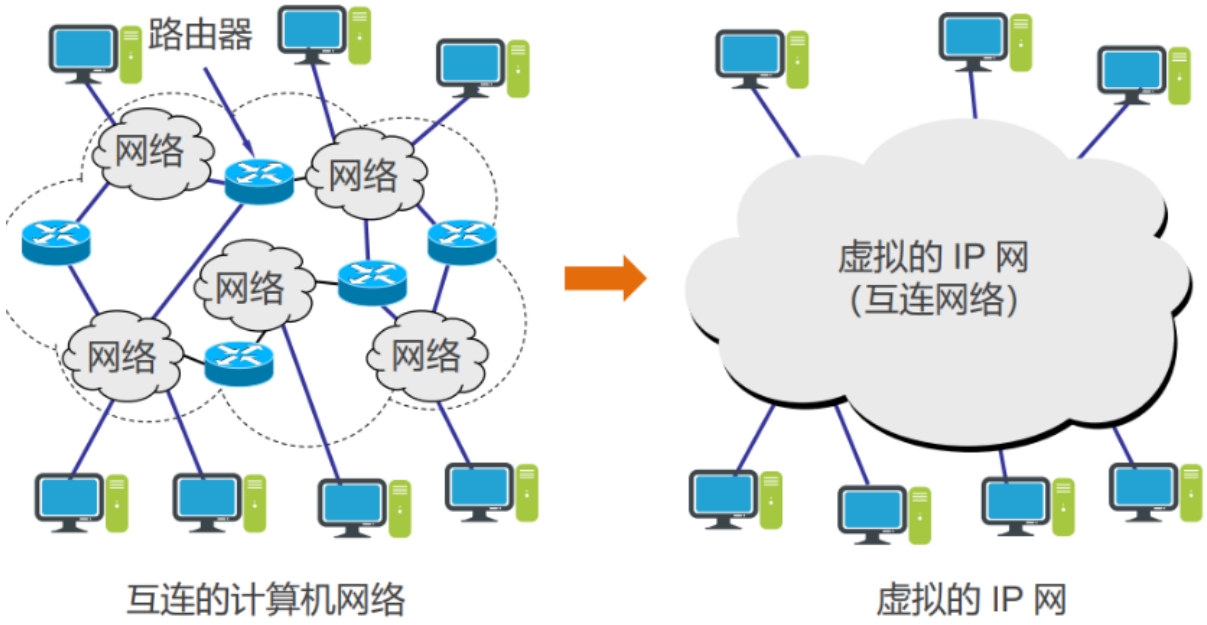




- 移动设备在两个无线接入点（红色AP和蓝色AP）之间无缝切换的过程，图表显示了在切换过程中，使用一种名为Odin的方案可以保持稳定的网络吞吐量，而普通方案则会出现短暂中断

More Example

- 示例 4：一个非IP网络 (A non-IP network)
  - **OpenFlow 的通用性，它不局限于处理 IP 协议，可以用来实验全新的网络架构**
  - 协议无关性：只要交换机中的流表能够匹配数据包的头部字段，OpenFlow 并不关心数据包的具体格式
  - 实现方式：支持非IP流量有多种方法
    - 可以通过匹配以太网头部（源/目的 MAC 地址）来识别流
    - 可以使用一个新的 EtherType 值来区分实验流量
    - 在IP层面，也可以通过一个新的 IP 版本号来识别



More Example

- 示例 5：处理数据包而非数据流
  - **通过控制器处理：**最简单的方法是强制一个流的所有数据包都经过控制器 。控制器不向交换机中添加新的流表条目，而是让交换机使用默认规则将每个数据包都转发给控制器处理
    - 这种方法的优点是灵活性高，但性能较差，更适合功能测试而非大规模部署
  - **通过外部设备处理：**更高效的方法是将需要处理的数据包路由到一个可编程的线速处理设备上，例如基于NetFPGA的可编程路由器
    - 在这种模式下，OpenFlow交换机的作用相当于一个可编程的“接线板”(patch-panel)，负责将特定流量精确地引导至NetFPGA设备进行处理，处理完毕后再导回网络 。这样既利用了OpenFlow的灵活性，又保证了处理性能

Outline

I. Background

II. Software Defined Networking

III. The OpenFlow Protocol

IV. Using OpenFlow

V. Review

Review

- SDN 是如何工作的?
- SDN 的三个层次
  - OpenFlow 位于哪个位置?
- OpenFlow 交换机
  - Header, action, statistics
- 如何使用 SDN?

Review

- SDN 优势总结：
- **敏捷性与灵活性：**通过自动化的方式，网络资源的调配和服务的上线时间可以从过去的数周、数天缩短到几分钟
- **动态资源调整：**网络不再是静态的，SDN控制器可以根据实时的业务流量和应用需求，动态地、智能化地调整网络路径和带宽分配（即流量工程）
- **简化网络运维：**通过控制器这一个中心点，即可实现全网的配置下发、策略部署和监控

Review

- SDN 优势总结：
- **策略高度一致性：**安全策略、访问控制（ACL）、服务质量（QoS）等策略只需在控制器上定义一次，便可精确、一致地应用到网络中的所有相关设备上
- **降低运营成本：**自动化运维显著减少了在网络配置、变更和故障排查上所需的人力投入和时间成本；更少的配置错误也意味着更少的业务中断损失，还能避免硬件被单一厂商的专有技术和高昂价格所绑定

Research topics in OpenFlow

- OpenFlow 交换机中的 TCAM 限制
- 使用 OpenFlow 交换机进行流量工程
- 分布式中央控制器
  - 博弈论
- 基于软件的流量分析
- 动态更新转发规则
  - zUpdate，瞬时拥塞
  - 最小化更新