Lecture 10：软件定义网络

SSE316：云计算技术
Cloud Computing Technologies

陈壮彬
软件工程学院
chenzhb36@mail.sysu.edu.cn

Today’s topics

- 网络的管理、控制、数据平面
- 传统网络的局限性
- 软件定义网络
- OpenFlow 协议
- 网络功能虚拟化

计算机时代的演进

大型机 (Mainframe)

APP APP … APP

—— Open Interface ——

Windows | Linux | Mac OS

—— Open Interface ——

Microprocessor

垂直继承，封闭接口
小规模行业应用

水平集成，开放接口
大规模跨产业应用

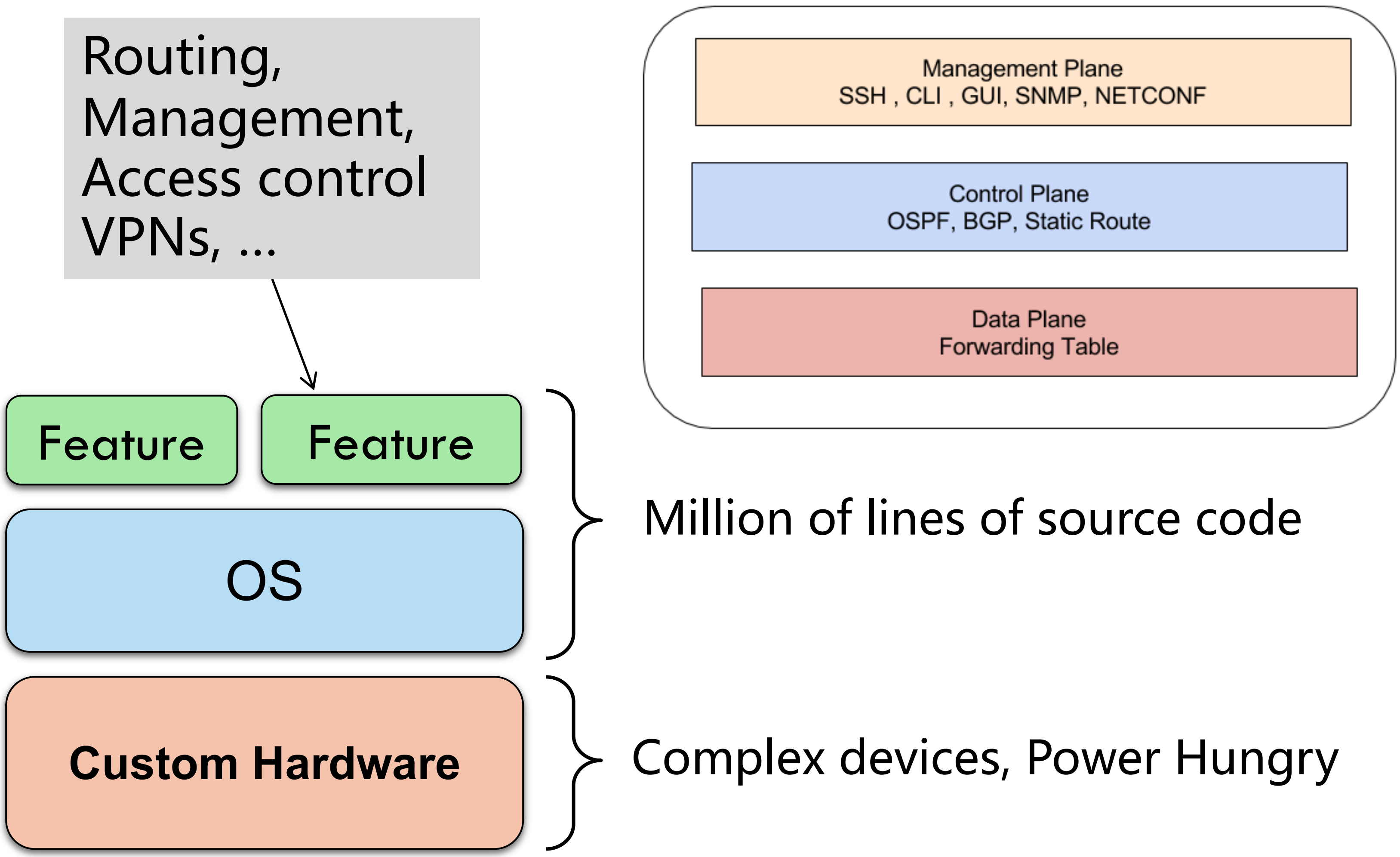
网络产业发展：来自 IT 行业的启示

- IT 产业的变革引发了网络产业的思考，业界开始提出 SDN
- 不断尝试将其商业化，希望网络变得更**开放**、**灵活**和**简单**

经典网络的局限性

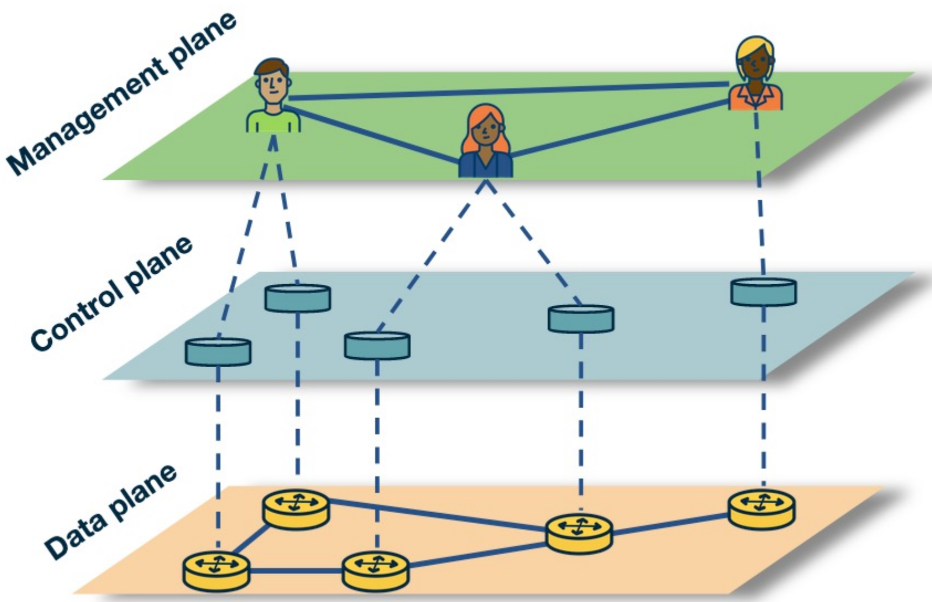
传统网络设备

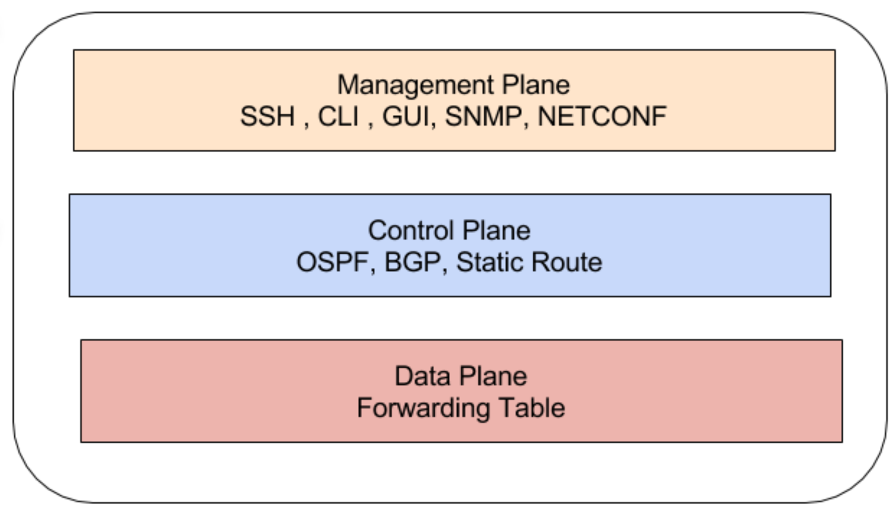
- 每台设备存在独立的**管理平面**、**控制平面**和**数据平面**



管理平面

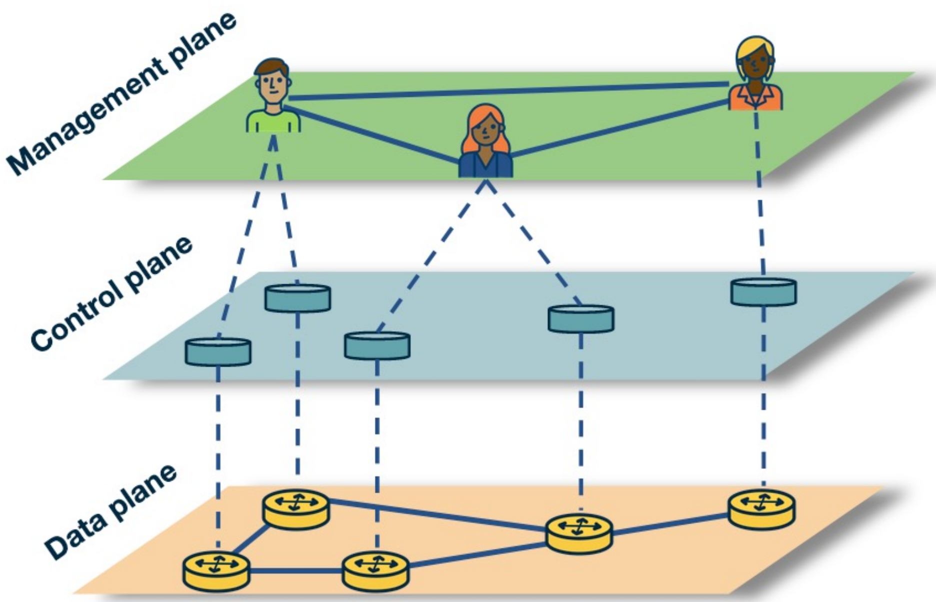
- □**网络管理员与网络设备交互的界面**
- □负责网络设备的参数配置、软件升级、故障排除、性能监控等
- □例如，网络管理员可以通过 SSH 或者网络管理协议 (如SNMP) 来访问和配置路由器、交换机等设备的管理接口

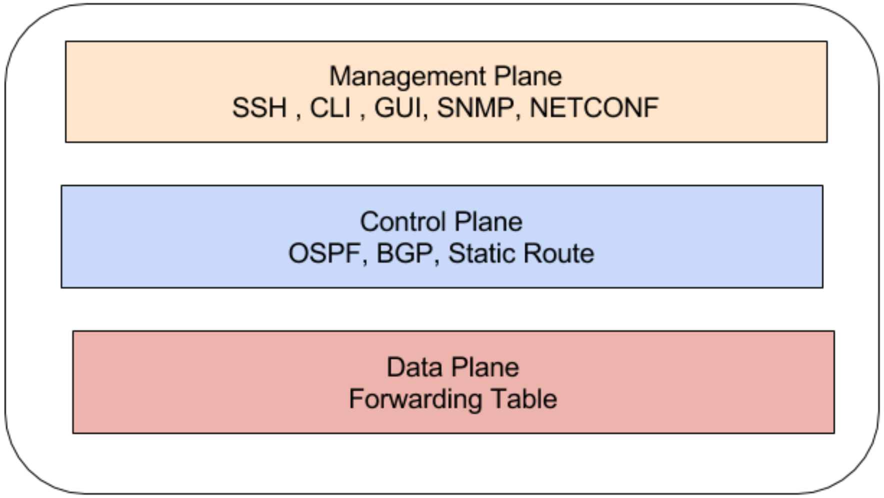




控制平面

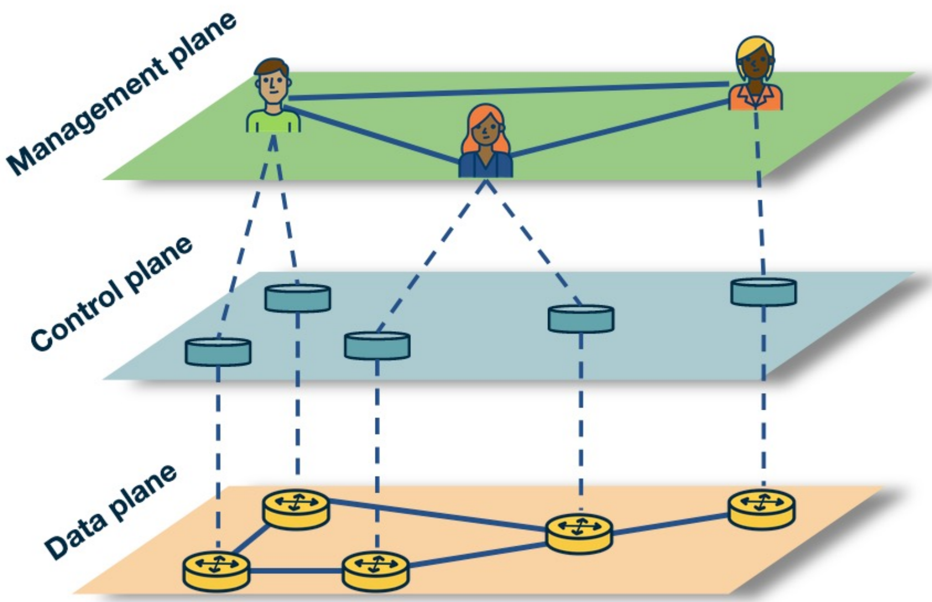
- 一旦网络被配置，控制平面就**负责网络如何做出决策**
- 涉及确定数据通过网络的路径的所有协议
- 例如，路由协议 (如OSPF、BGP等) 运行在控制平面，负责计算网络的最佳路径，并将这些决策传递给数据平面

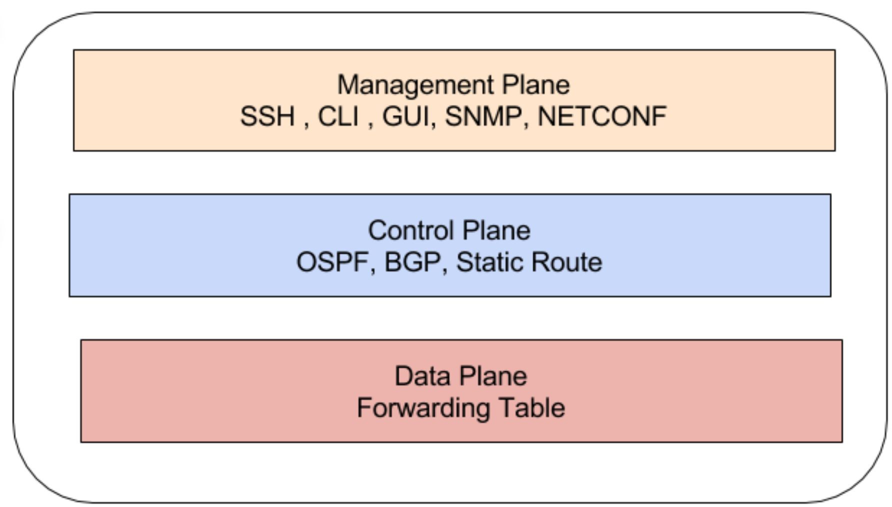




数据平面 (或转发平面)

- 负责**实际的数据包的接收、处理和转发**
- 例如，当一个数据包到达一个路由器时，数据平面的硬件或软件将根据控制平面的决策，将数据包转发到正确的接口

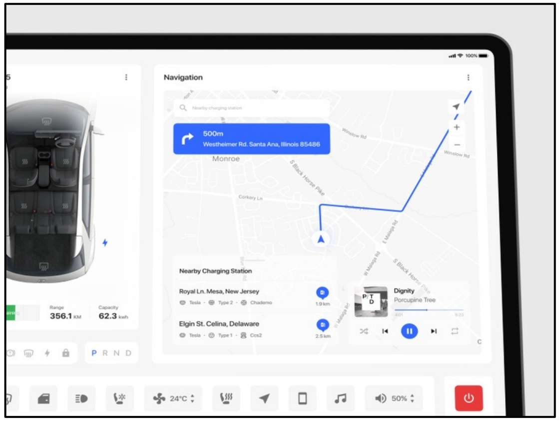




类比例子 – 智能汽车

管理平面：在仪表盘上选择导航软件 (高德还是百度)，并设置出发点和目的地

控制平面：导航软件接收这些设置，然后计算最优的驾驶路线



数据平面：发动机转动，依据导航软件确定的路线行驶

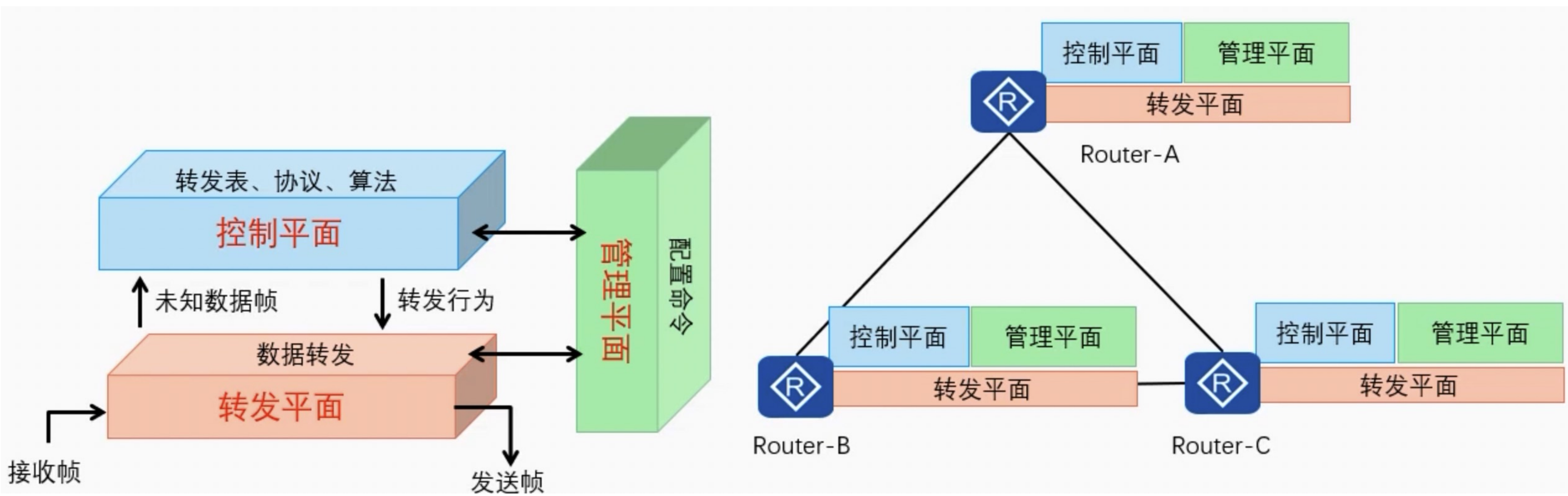
传统网络 (分布式网络)

□传统网络是一个分布式、对等控制的网络

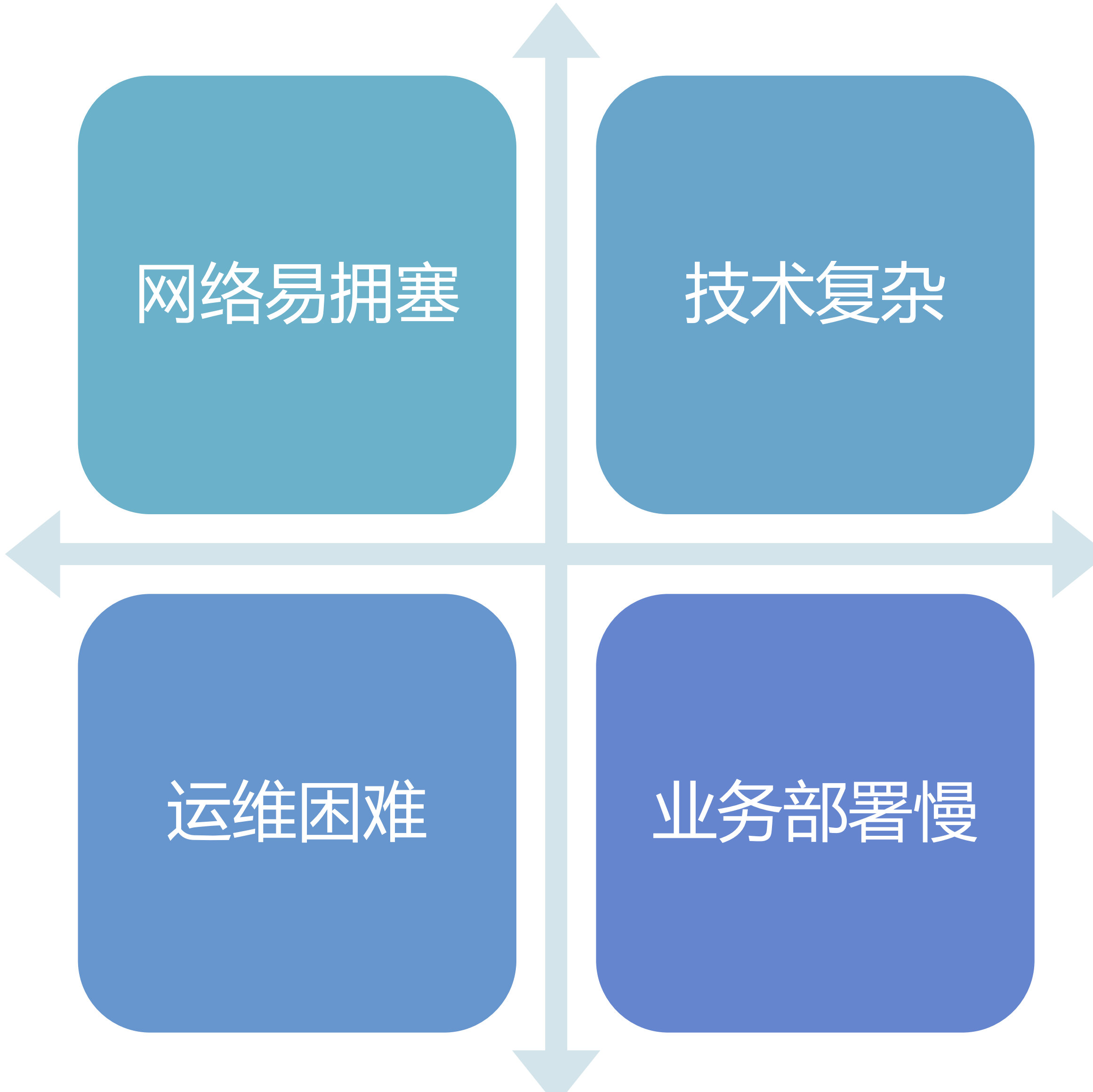
□设备的控制平面对等地交互路由协议，独立地在数据平面转发报文

□传统网络的优势

- 设备与协议解耦
- 厂家之间兼容性较好
- 故障场景下协议保证网络收敛

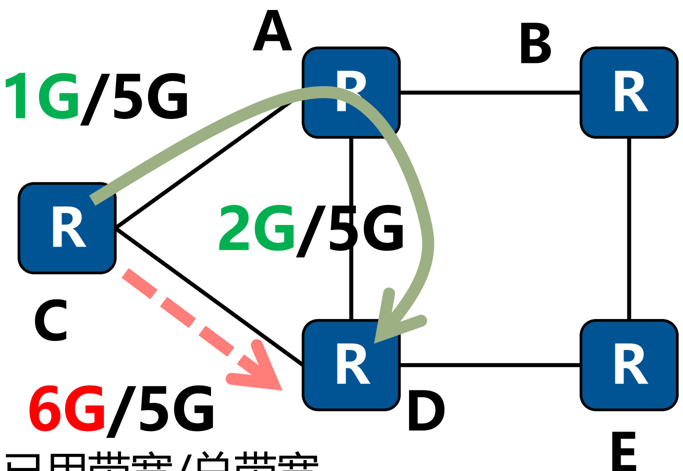


经典网络面临的问题



网络易拥塞

基于带宽固定选路的问题和解决思路



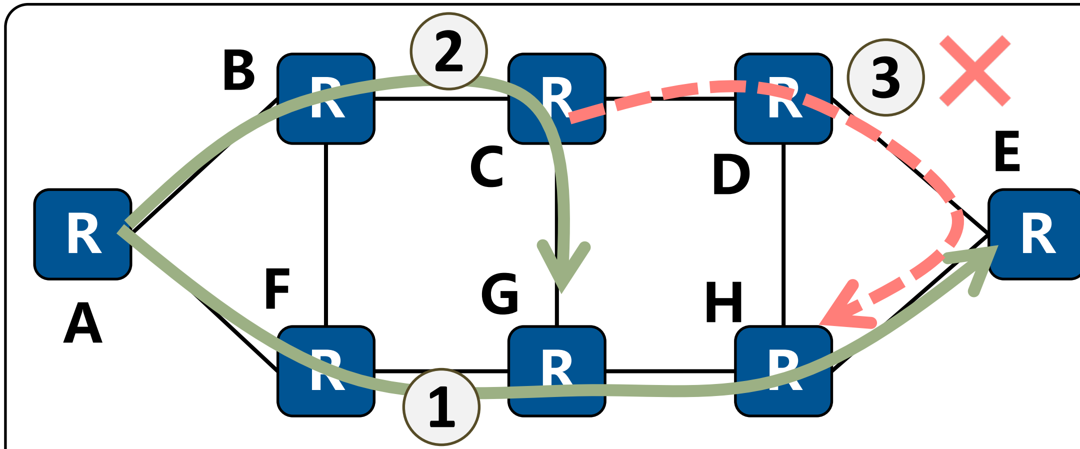
已用带宽/总带宽

网络基于带宽计算转发路径；

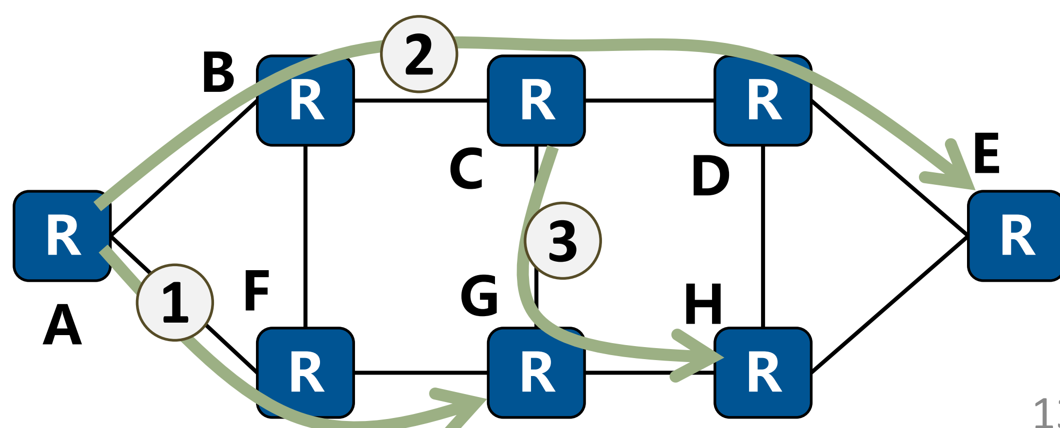
路由器C到D的链路为最短转发路径。C-D的业务流量超过带宽出现丢包现象。虽然其他链路空闲，但算法依选择最短路径转发；

全局考虑，此时最优的流量转发路径为C-A-D。

基于固定顺序建立隧道的问题和解决思路



全局计算，最优调整路径为：



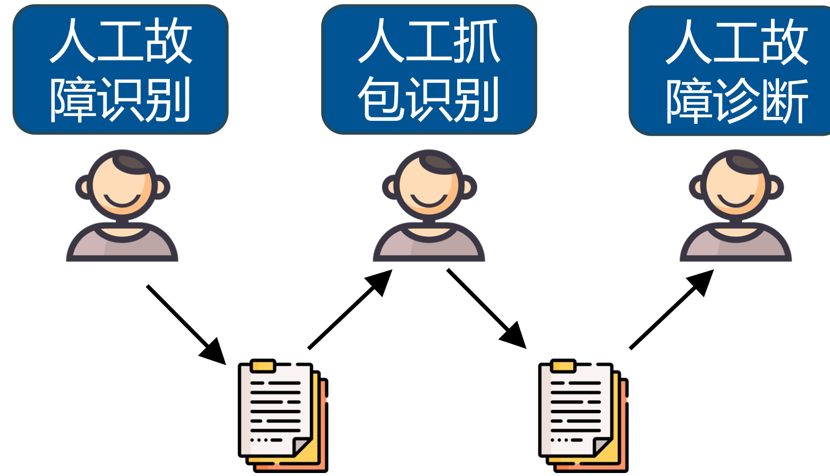
网络技术复杂

- 网络协议多
  - 成为一名网络技术专家，需要阅读网络设备相关 RFC 2500篇
  - 一天阅读一篇，需要6年时间，这只是整个 RFC 的 1/3，其数量还在持续增加
- 网络配置难
  - 成为某个设备商设备的百事通，需要掌握超过 10000 条命令

网络故障定位、诊断难

□故障发现难

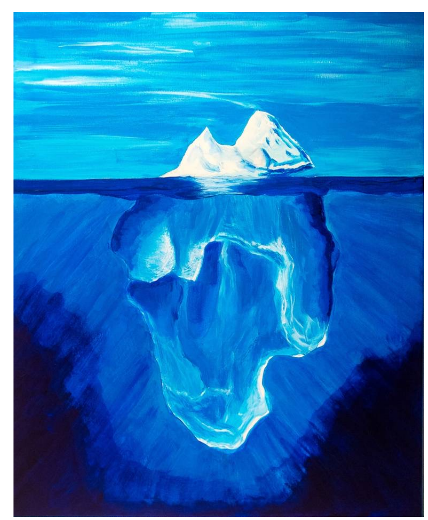
- 传统运维网络依靠人工识别、定位和诊断
- 超过 85% 的网络故障业务投诉后才发现，无法有效主动识别、分析问题



□故障定位难

- 传统运维仅监控设备指标，存在指标正常但用户体验差的情况，缺少用户和网络的关联分析
- 数据中心网络统计，一个故障定位**平均耗时 76 mins**

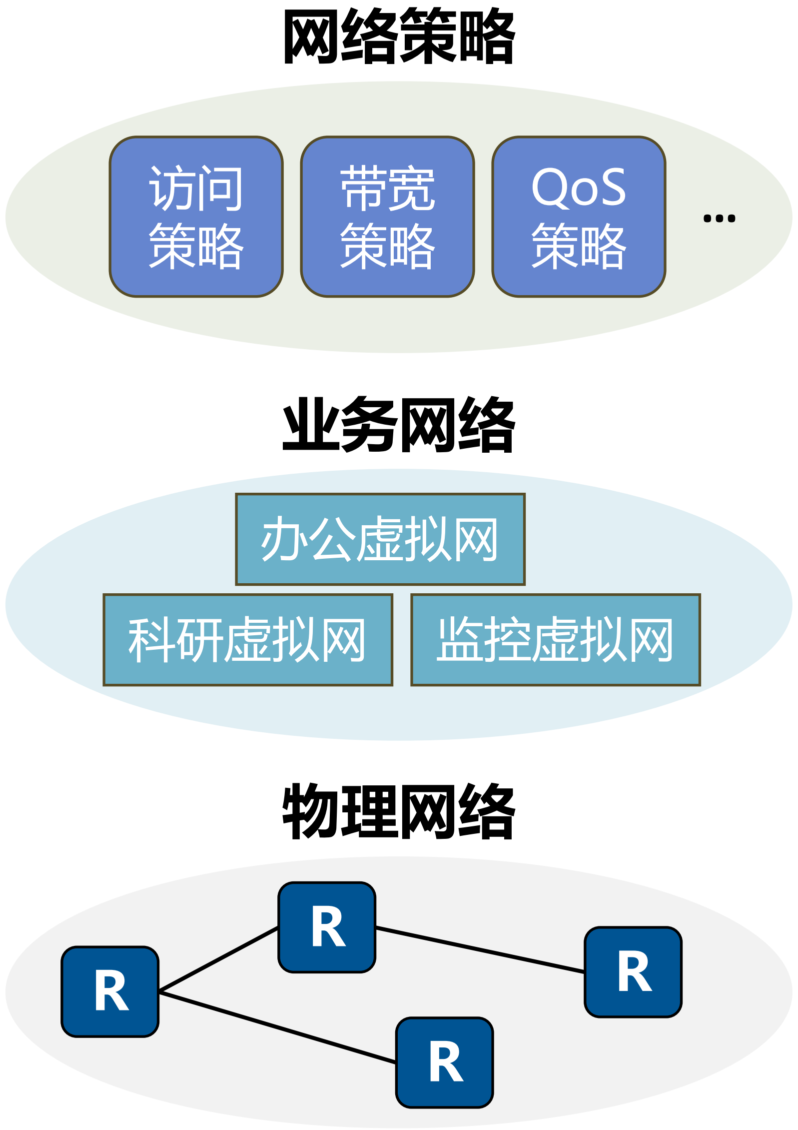
异常流占全网流 3.65%



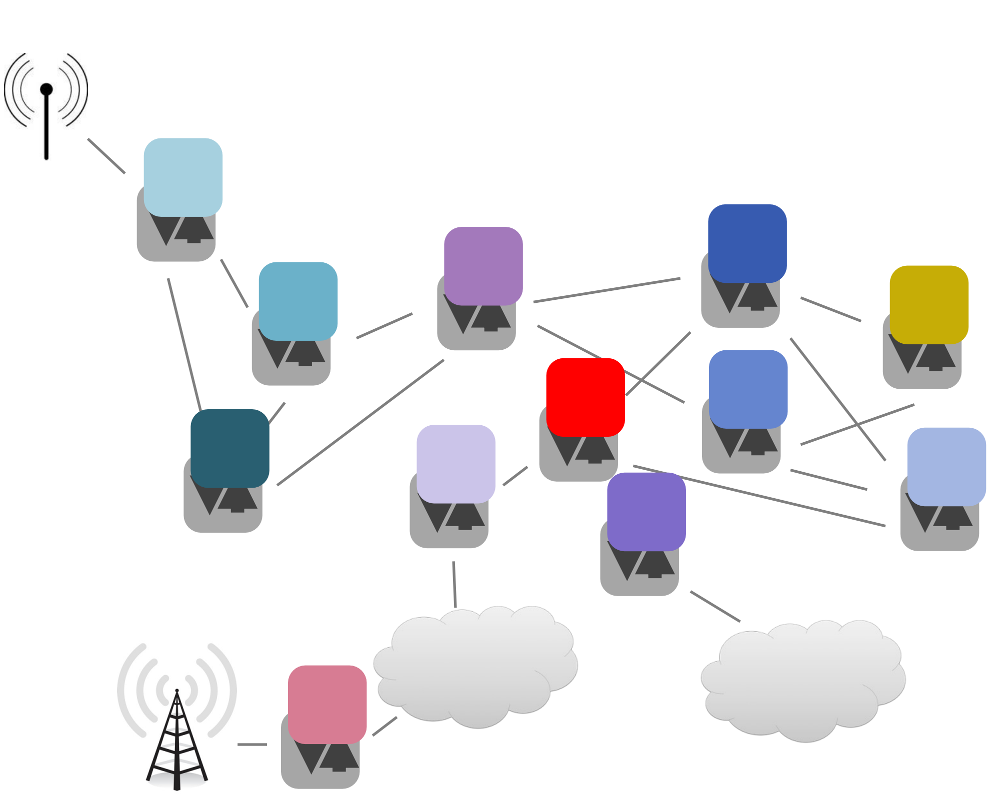
经用户投诉而定位的网络故障仅是冰山一角

网络业务部署慢

- 网络策略变更复杂、不灵活
  - 大数据带来的策略频繁变更需求
  - 网络策略无法细化到用户
- 新业务部署周期长
  - 新业务涉及端到端设备配置修改
  - 为了贯彻某种全网策略需兴师动众
  - 容易导致安全漏洞等问题
- 物理网络部署效率低
  - 物理网络无零配置部署能力
  - 存在一定程度的供应商锁定，缺乏标准的开放式接口



网络业务部署例子



一家大型企业的网络，拥有分布在多个地点的数百台交换机、路由器和其他网络设备

该公司决定实施一项新的安全策略，要求阻止或重新路由某些类型的流量。

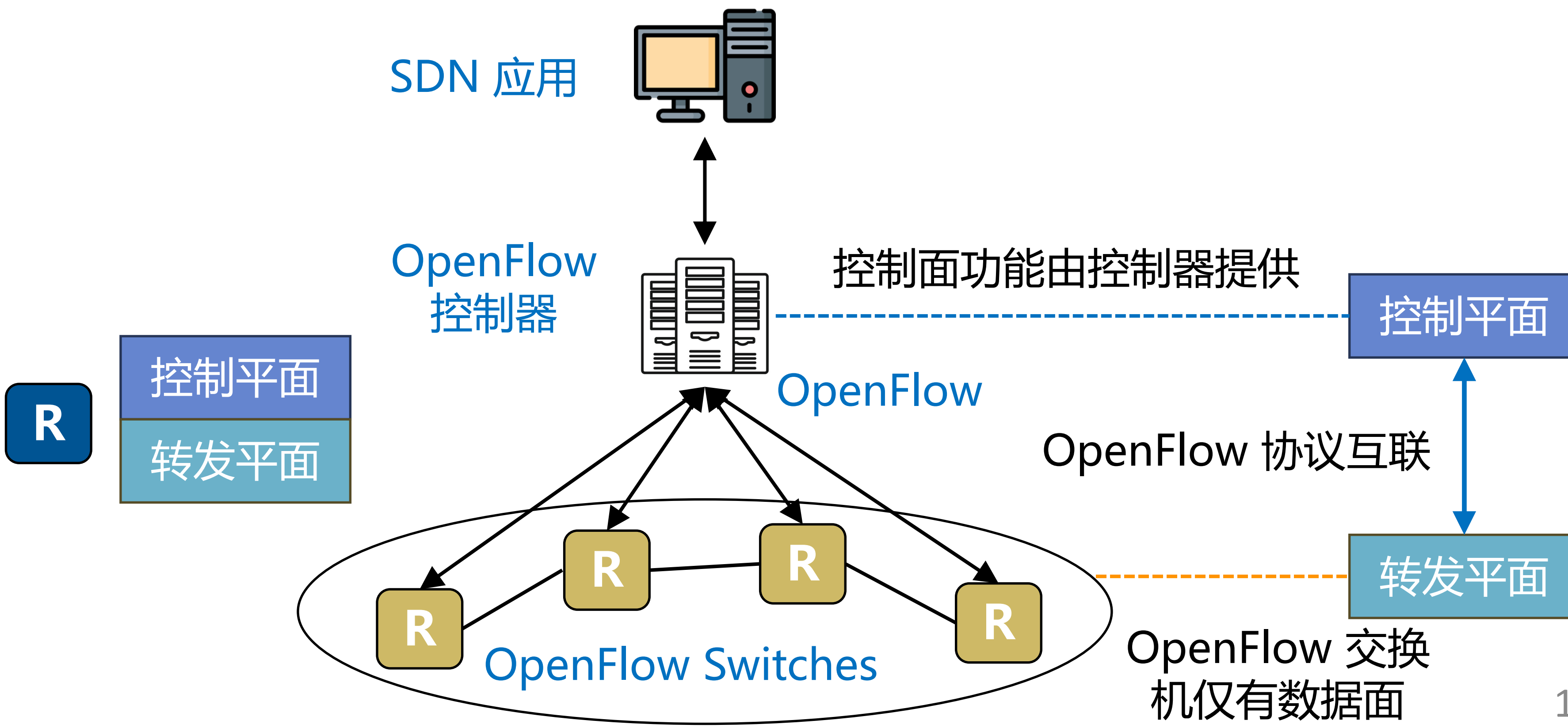
传统网络做法：

1. 登陆到每一台设备
2. 修改每一台设备的配置
3. 确保配置正确生效
4. 处理其中发送的任何问题（大概率会发生）
5. …

软件定义网络

软件定义网络（SDN）

- SDN (Software-defined Network) 软件定义网络
  - 由斯坦福大学 Clean Slate 研究组提出的一种新型网络架构
  - 通过将网络设备**控制平面与数据平面分离**，实现**网络的集中控制**
  - 三个特征：**转控分离**、**集中控制**和**开放可编程接口**



SDN 关键能力

□集中管理

- 通过控制器直接检测整个网络的状态，便于网络全局管理

□网络可编程性

- 网络设备提供API，管理人员使用编程语言向网络设备发送指令

□网络抽象

- 控制器作为中间层隔离服务和硬件，二者不再紧密耦合

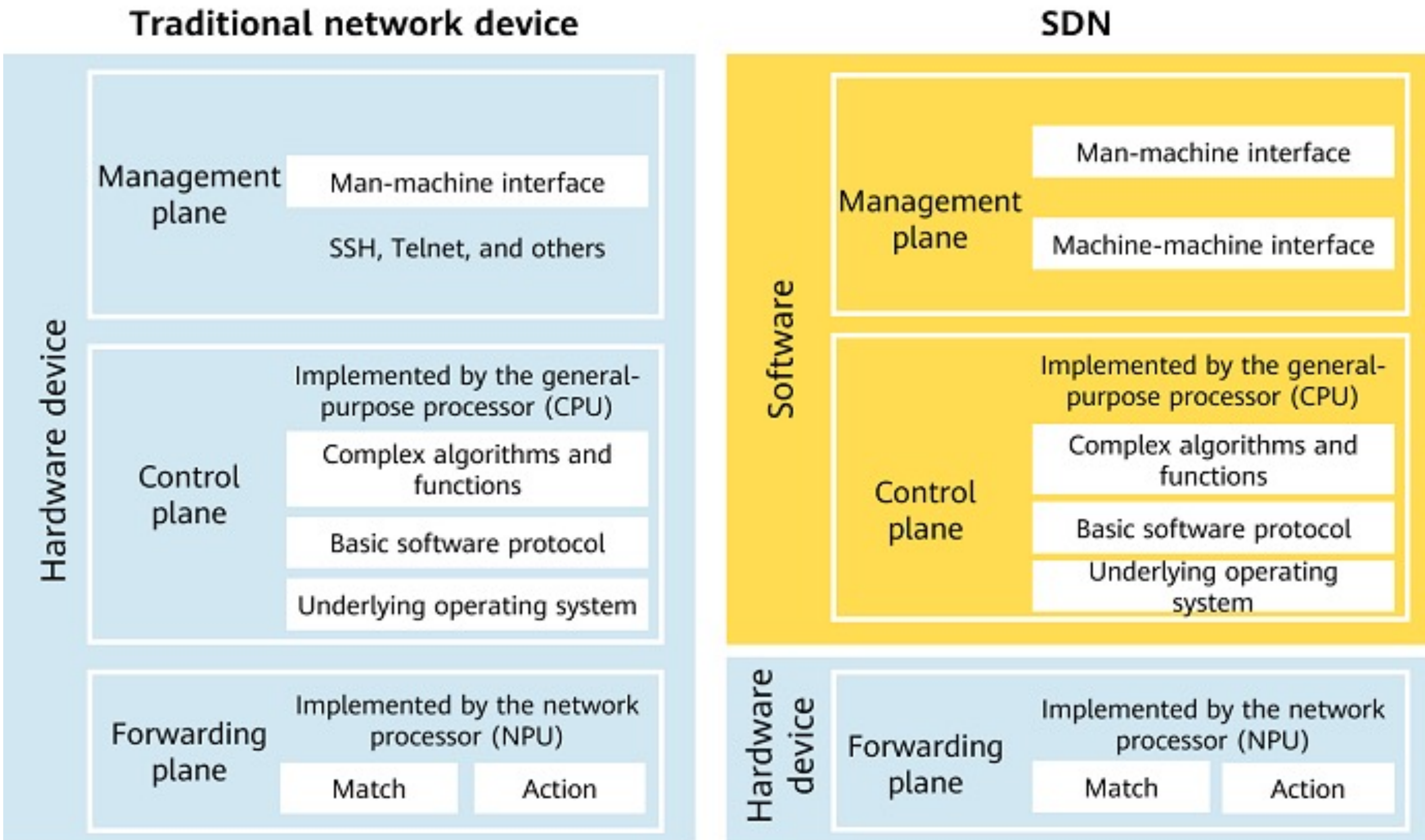
□降低成本

- 将手动配置转为基于机器的配置，降低了网络更新和运维成本

□开放性

- SDN 架构允许供应商通过开放的API开发自己的生态系统

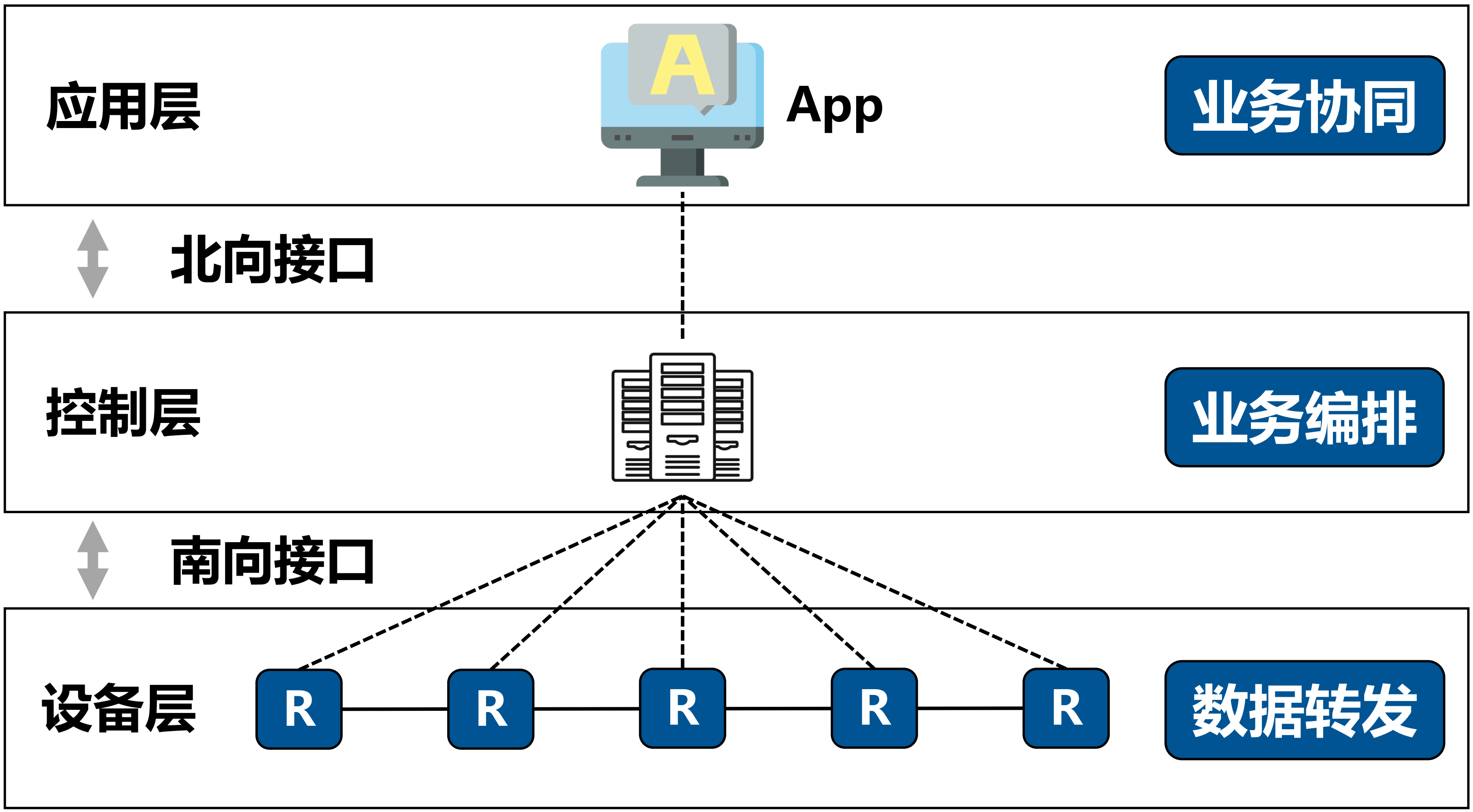
传统网络设备 vs. 软件定义网络设备



SDN技术将网络设备改为**可编程的白盒设备**，以实现用户定义的网络功能

SDN 网络架构

- 分为**应用层**、**控制层**和**基础设施层**，层次之间**开放接口连接**
- 以控制层为视角，区分面向基础设施层和应用层的南北接口



SDN 网络架构

□应用层

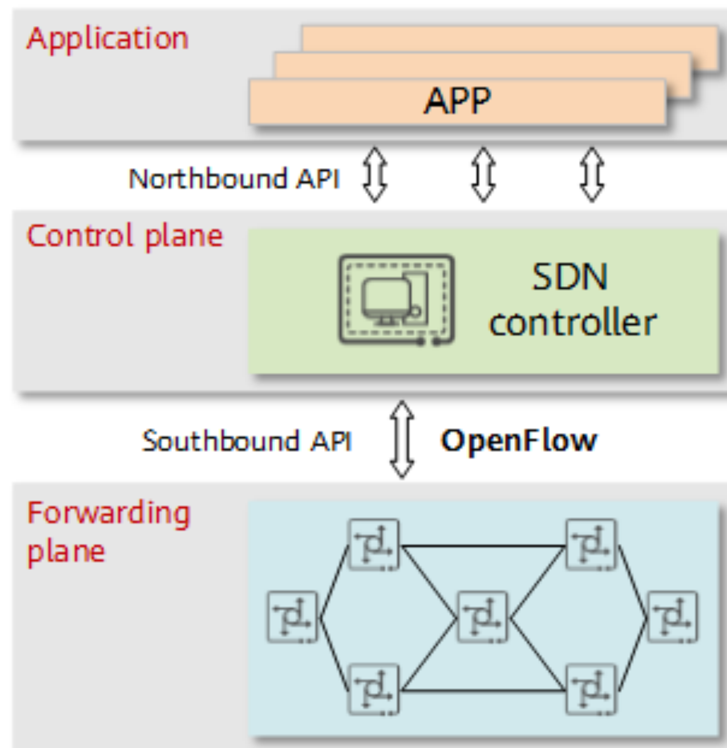
- 利用 SDN 网络服务的各种应用
- 通过北向接口与控制层交互
- 例如创建/修改网络流、实现特定负载均衡策略

□控制层

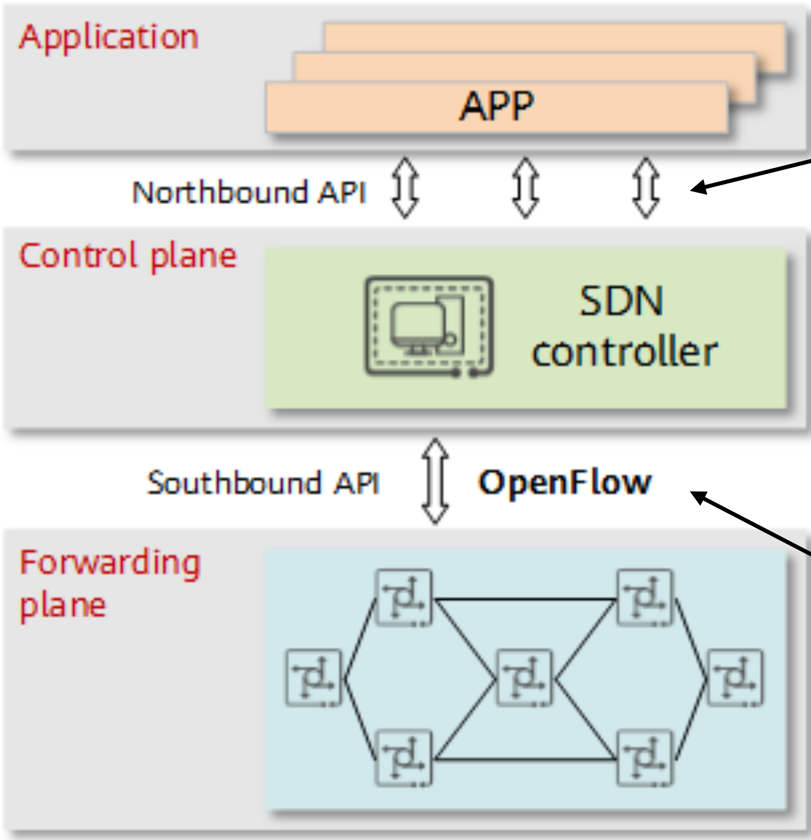
- SDN 的核心
- 将来应用请求转化为具体的网络设备配置，并发给设备

□设备层

- 负责处理和转发数据包
- 依据控制层的指示进行操作



SDN 网络架构



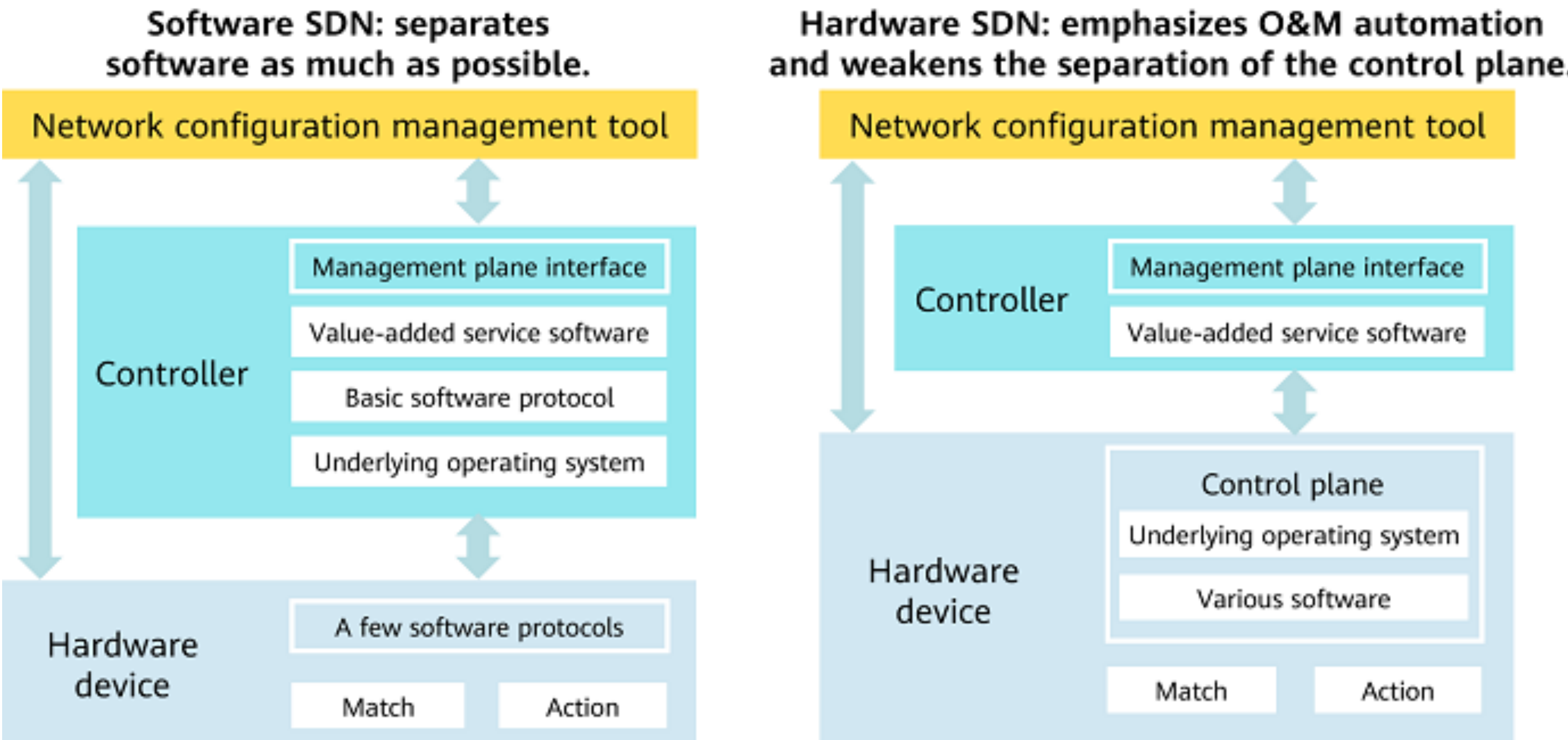
北向 API：控制器与在其上运行的应用程序之间的接口。北向API允许应用程序发送请求以改变网络的行为。

南向 API：控制器与网络设备之间的接口，允许控制器读取网络设备的状态和统计信息。

软件 SDN vs. 硬件 SDN

- 随着 SDN 的发展，由于复杂的底层协议和巨大的软件开发投资等，供应商逐渐将重点从控制平面的分离转移到运维自动化
- 控制器用于与硬件设备和网络配置管理工具互连，以实现管理平面上硬件设备的统一管理和服务编排

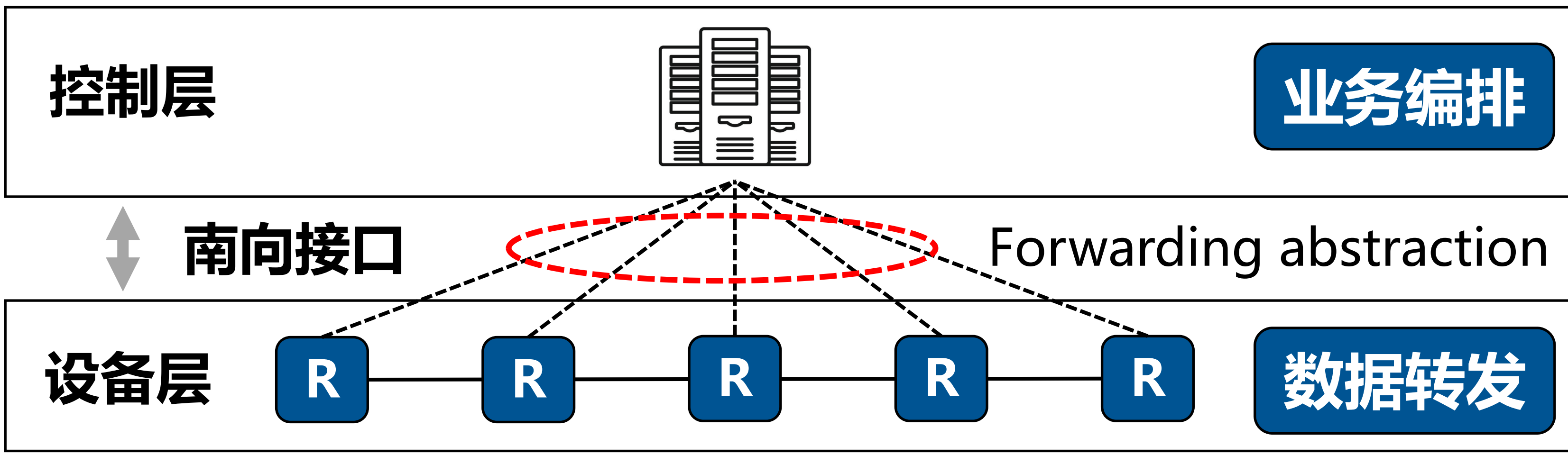
软件 SDN vs. 硬件 SDN



OpenFlow 协议

转发设备的抽象

- 控制层希望对底层转发设备有一层抽象

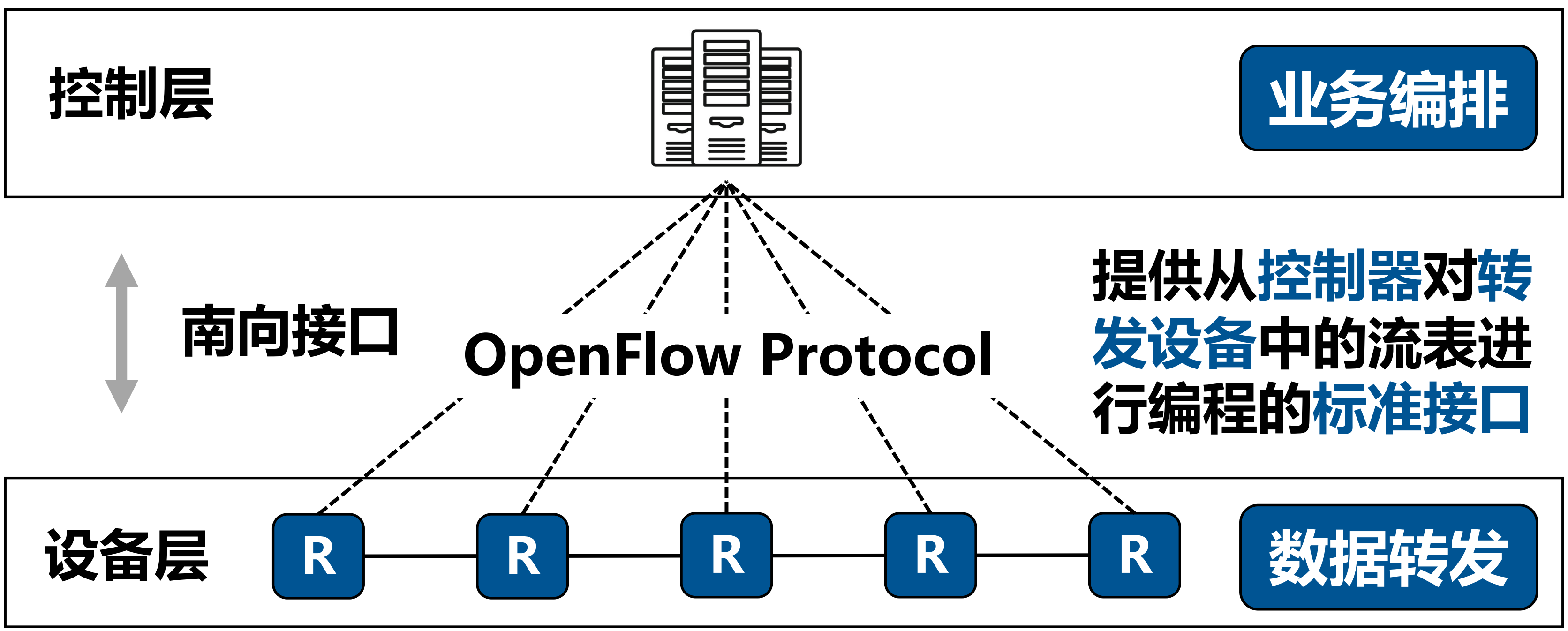


- **灵活性 Flexible**
  - 控制平面指定转发行为
  - 由基本的转发原语构建
- **最小化 Minimal**
  - 精简控制以实现速度和低功耗
  - 控制程序不特定于供应商

OpenFlow is an example of such an abstraction

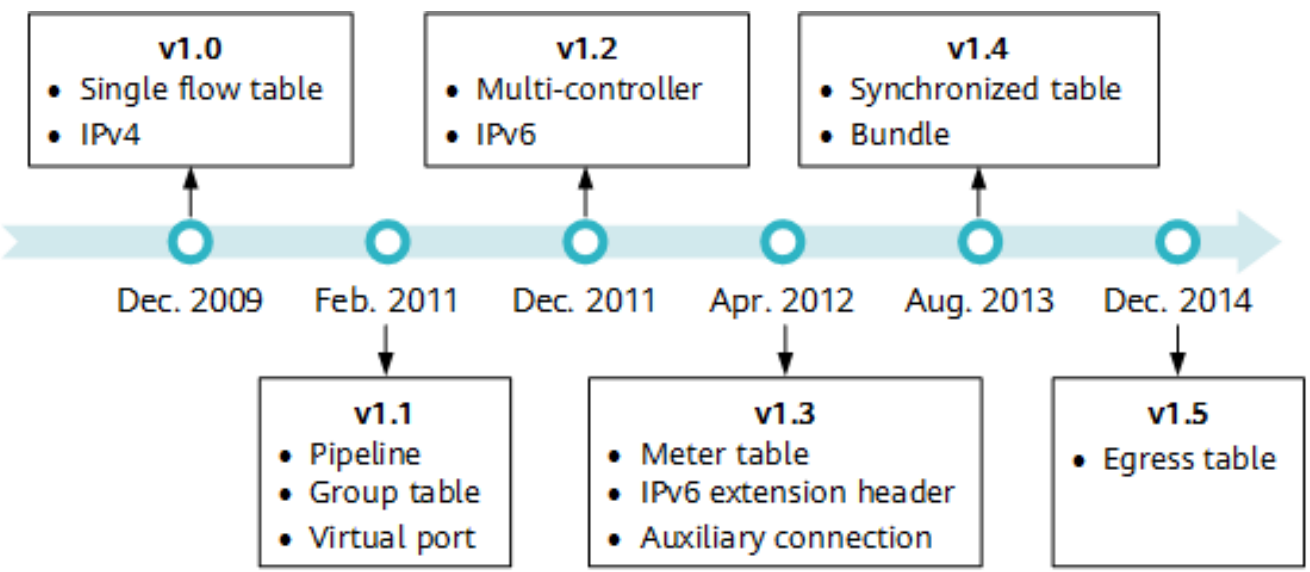
OpenFlow 协议

- OpenFlow 是一种在 SDN 架构中的**控制器和转发器之间使用的网络通信协议**
- 为了实现转发与控制平面分离，必须在**控制器和转发器之间建立通信标准**，以允许控制器直接访问和控制转发器的转发平面



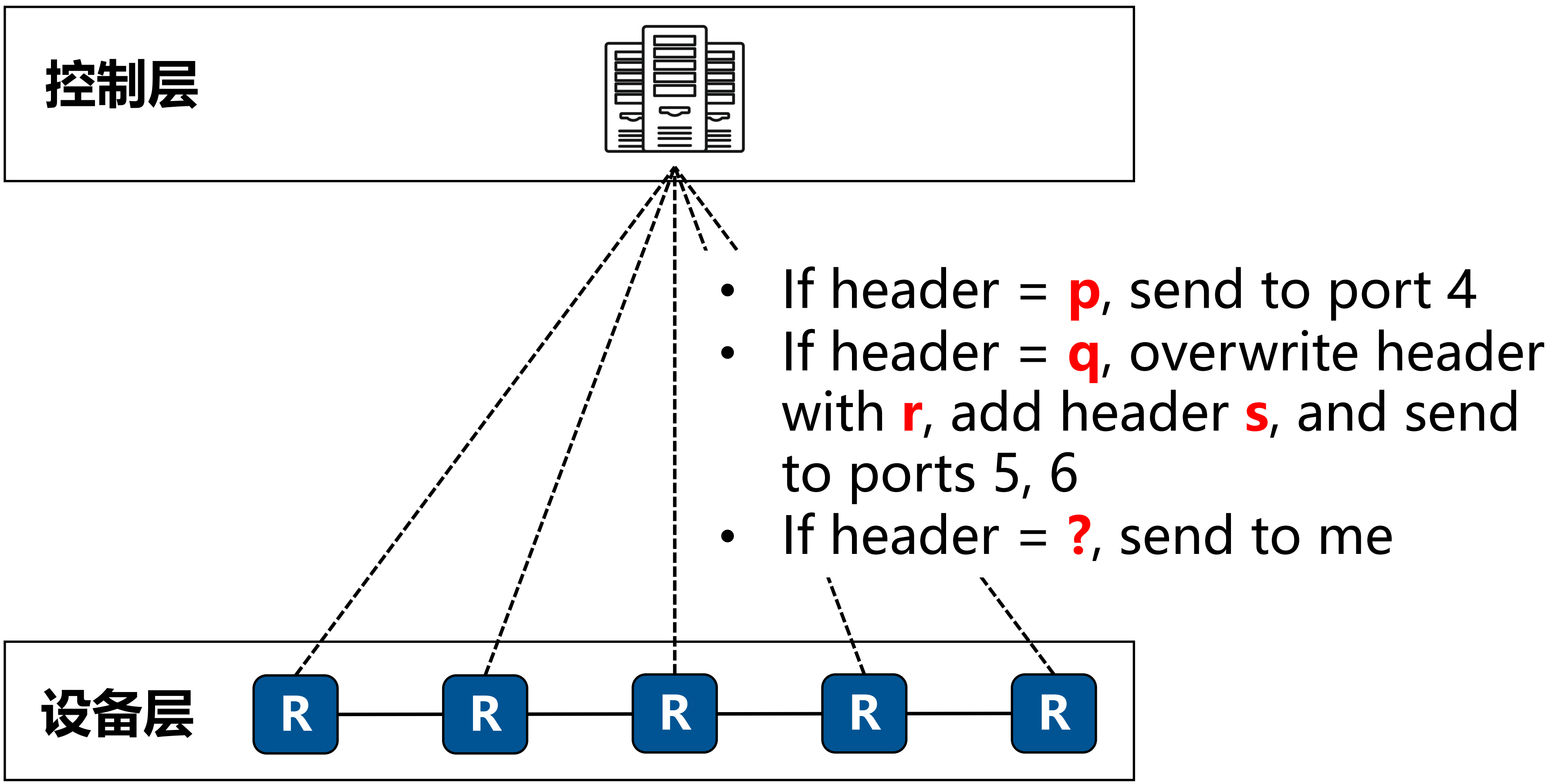
OpenFlow 的起源和发展

- OpenFlow 起源于斯坦福大学的 Clean Slate Program，这个项目探讨了如何从零开始重新设计互联网，以解决当时网络基础设施的一些缺陷
- 2006年，斯坦福大学的学生 Martin Casado 带领了一个关注网络安全和管理的项目。这个项目试图使用一个集中的控制器，使网络管理员能够基于网络流量轻松定义和实施安全策略
- 受到这个项目的启发，Clean Slate Program 的主管，Nick McKeown 教授，提出了将传统网络设备的数据转发和路由控制模块分离的想法
- 2008年，McKeown 教授在一篇名为 "OpenFlow: Enabling Innovation in Campus Networks" 的论文中首次详细公开介绍了 OpenFlow 的概念



OpenFlow 版本的演变和主要变化

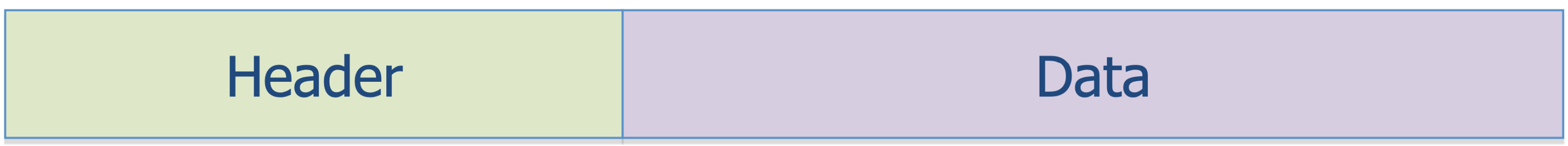
OpenFlow 基础


控制层
设备层
R
R
R
R
R
• If header = **p**, send to port 4
• If header = **q**, overwrite header with **r**, add header **s**, and send to ports 5, 6
• If header = **?**, send to me

OpenFlow Primitives 原语

<Match, Action>

- Match arbitrary bits in headers
  - 能匹配任何头部
  - 允许任何流粒度 (Flow granularity)

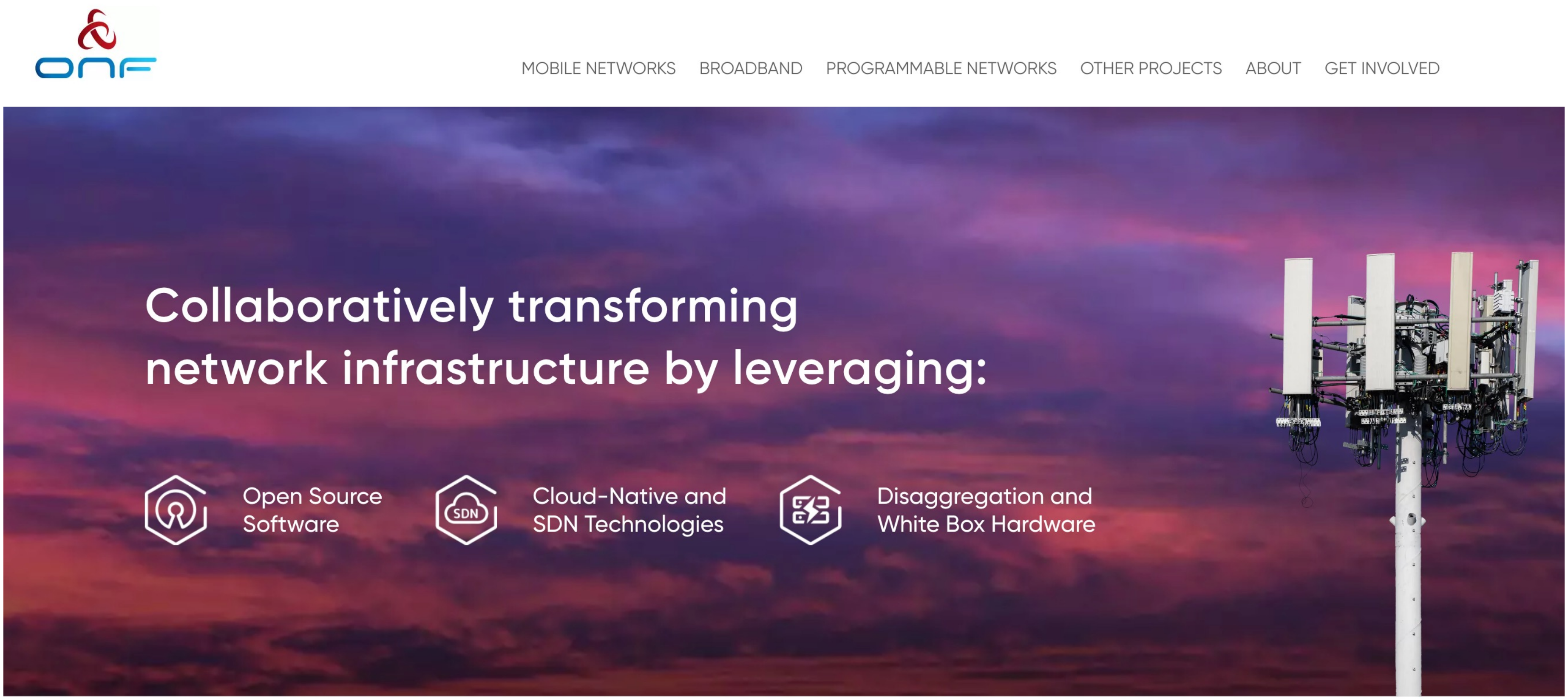


- Action
  - 转发到端口，丢弃，或发送给控制器
  - 使用掩码覆写头部，推送或弹出
  - 以特定比特率转发

理解 OpenFlow

- OpenFlow类似于网络的x86指令集
- 为"黑箱"网络节点（例如路由器，L2/L3交换机）提供开放接口，以实现网络的可见性和开放性
- 控制平面和数据平面的分离
  - OpenFlow 交换机的**数据路径**包括一个流表，以及与每个流条目相关联的动作
  - **控制路径**包括一个控制器，用于在流表中编程条目
- OpenFlow 基于以太网交换机，具有内部流表，并有一个标准化的接口来添加和删除流条目

Open Networking Founda7on (ONF)



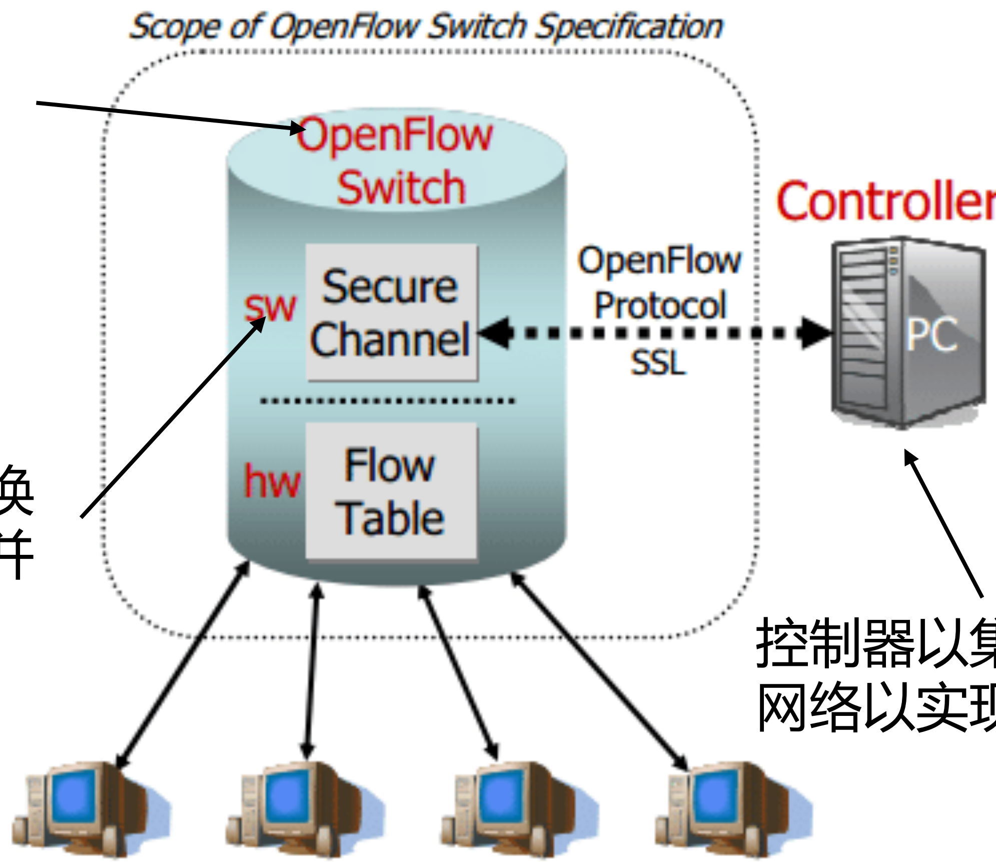
https://opennetworking.org/

OpenFlow 架构

□OpenFlow主要由**控制器**、**OpenFlow交换机**和**安全通道**组成

OpenFlow 交换机负责在数据层进行转发

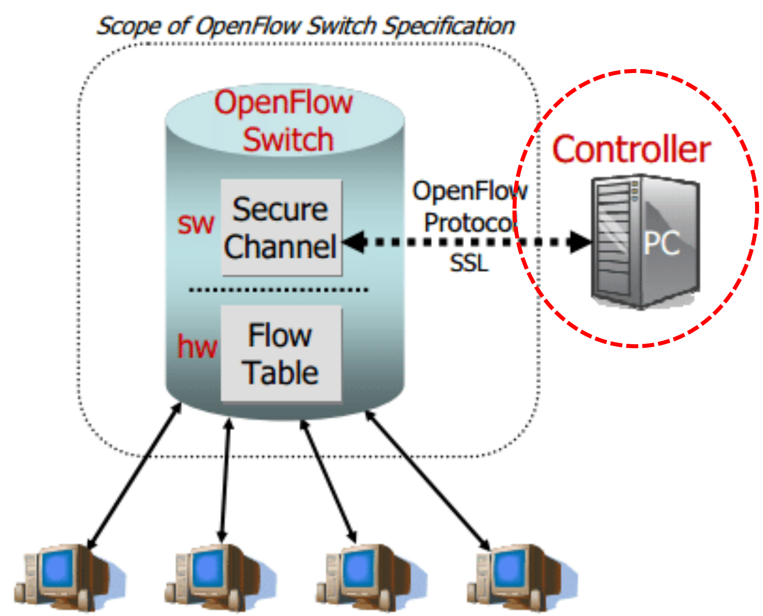
安全通道与控制器交换消息以接收转发条目并报告其状态



控制器以集中的方式控制网络以实现控制层的功能

OpenFlow 控制器

- OpenFlow 控制器是 SDN 架构的大脑
  - ✓**管理网络设备**：通过 OpenFlow 协议管理和控制网络中的所有 OpenFlow 兼容设备（例如交换机）
  - ✓**路由决策**：OpenFlow 控制器负责决定网络流量的路由
  - ✓**网络监控**：OpenFlow 控制器可以收集网络设备的状态信息和统计数据



- 主流的 OpenFlow 控制器分为两类
  - 开源控制器和供应商的商业控制器
  - 广泛使用的开源控制器包括 NOX、POX 和 OpenDaylight

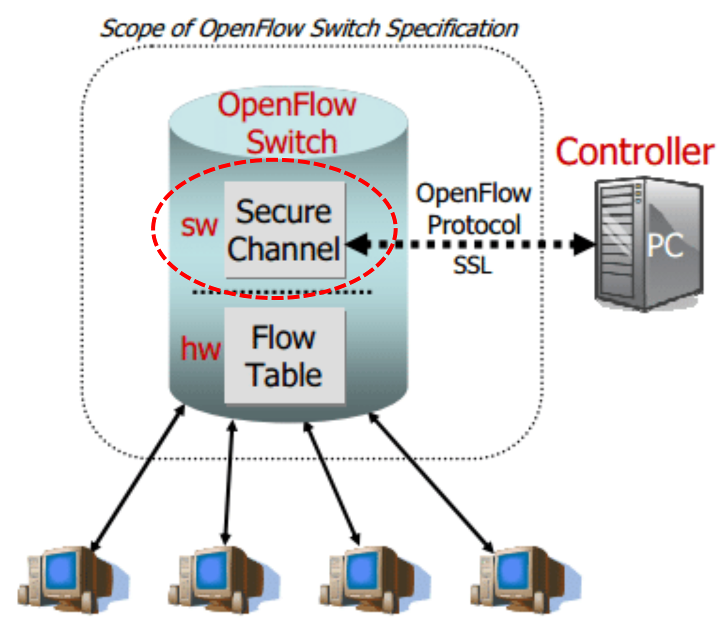
OpenFlow 控制器

Name	Lang	Platform(s)	License	Original Author	Notes
OpenFlow Reference	C	Linux	OpenFlow License	Stanford/ Nicira	not designed for extensibility
NOX	Python, C++	Linux	GPL	Nicira	actively developed
Beacon	Java	Win, Mac, Linux, Android	GPL (core), FOSS Licenses for your code	David Erickson (Stanford)	runtime modular, web UI framework, regression test framework
Maestro	Java	Win, Mac, Linux	LGPL	Zheng Cai (Rice)	
Trema	Ruby, C	Linux	GPL	NEC	includes emulator, regression test framework
RouteFlow	?	Linux	Apache	CPqD (Brazil)	virtual IP routing as a service

OpenFlow 安全通道

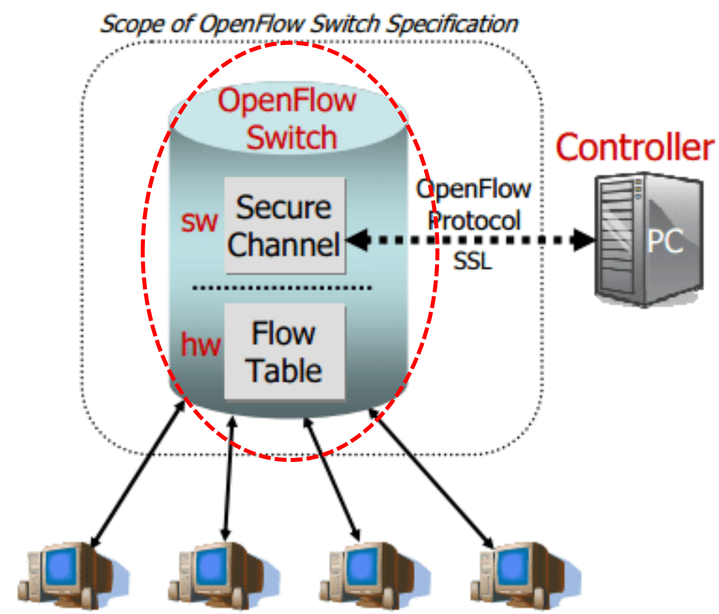
- 安全通道在控制器和 OpenFlow 交换机之间建立；通过该通道，控制器控制和管理交换机，并接收来自交换机的反馈

- 以下 OpenFlow 消息由安全通道传输：
  - **控制器到交换机消息**：由控制器发送到 OpenFlow 交换机，以管理或查询 OpenFlow 交换机状态
  - **异步消息**：由 OpenFlow 交换机发送到控制器，以更新控制器的网络事件或状态更改
  - **对称消息**：由 OpenFlow 交换机或控制器在没有请求的情况下发送。它主要用于建立连接和检测对等端是否在线



OpenFlow 交换机

- OpenFlow 网络的核心组件，主要负责数据层的转发，可以是物理交换机或虚拟交换机/路由器
- OpenFlow 交换机根据流表转发进入交换机的数据包，流表包含一组指示交换机如何处理流量的策略条目
- 流量条目由控制器生成、维护和传递



OpenFlow 流表

- 交换机和路由器等传统网络设备基二层MAC 地址表、三层 IP 地址路由表和传输层端口号来转发数据
- OpenFlow 基于包含网络上所有层的网络配置信息的流表而不是五元组信息来转发数据
- 流表中的条目是某些关键字和操作的灵活组合

	规则（Rule）	动作（Action）	统计（Stats）
Flow 1.	Rule (exact & wildcard)	Action	Statistics
Flow 2.	Rule (exact & wildcard)	Action	Statistics
Flow 3.	Rule (exact & wildcard)	Action	Statistics
	....		
Flow N.	Rule (exact & wildcard)	Default Action	Statistics

OpenFlow 流表

□**规则 (Rule)**：对流量进行分类的标准

✓匹配字段包括源 IP 地址、目标 IP 地址、协议类型、端口号等

✓当数据包到达交换机时，进行规则匹配并进行相应的动作

□**动作 (Action)**：定义一旦匹配规则，交换机处理数据包的指令

✓转发数据包到特定的端口

✓修改数据包的某些字段（如改变优先级或VLAN标签）

✓丢弃数据包

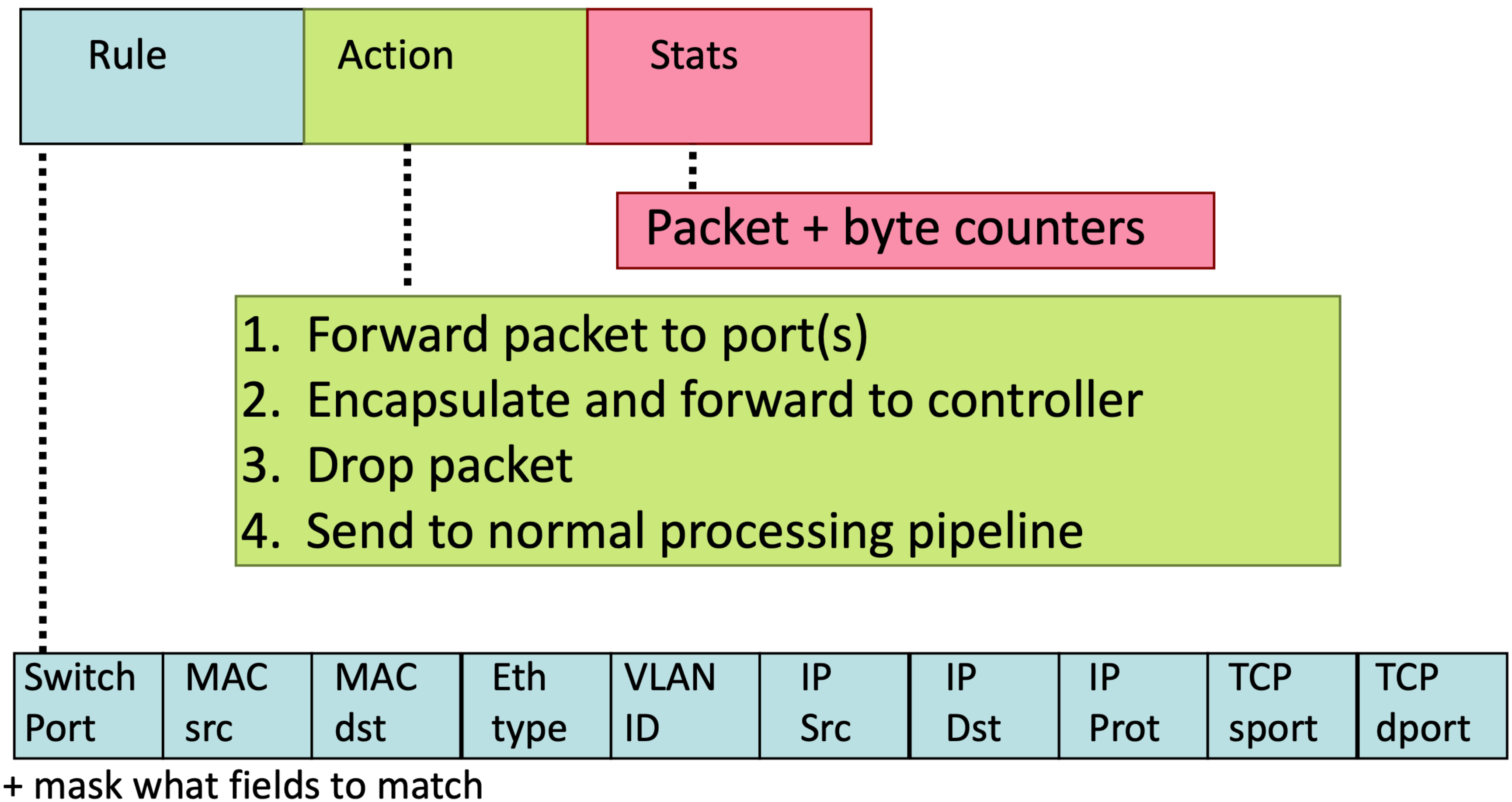
✓或者将其发送到控制器以进行进一步的处理

□**统计 (Stats)**：流量的统计数据，由交换机收集并报告给控制器

✓特定规则匹配的数据包数量、通过特定端口转发的数据包数量、丢弃的数据包数量等

✓这些统计信息对于网络管理和故障排除非常有用，因为它们可以提供关于网络流量模式和设备性能的详细信息

OpenFlow 协议（Version 1.0)



OpenFlow 流表例子

Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f:..	*	*	*	*	*	*	*	port6

Flow Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
port3	00:2e..	00:1f..	0800	vlan1	1.2.3.4	5.6.7.8	4	17264	80	port6

Firewall

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Forward
*	*	*	*	*	*	*	*	*	22	drop

OpenFlow 流表例子

Routing

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	5.6.7.8	*	*	*	port6

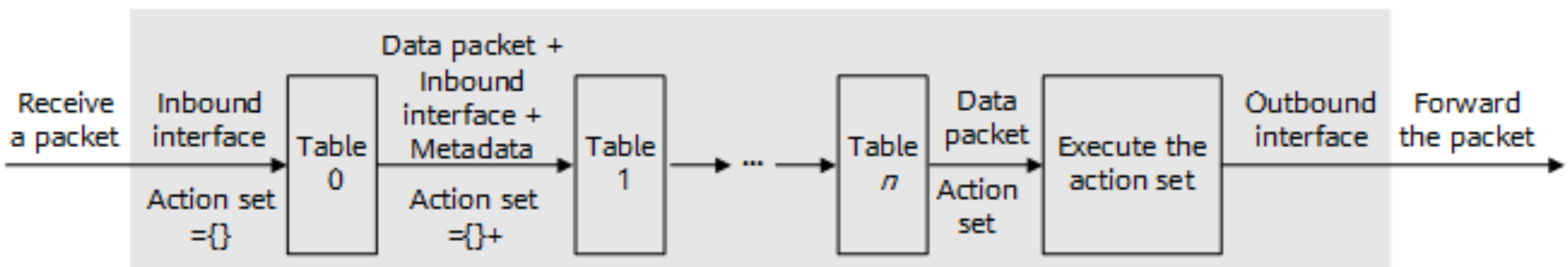
VLAN

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	vlan1	*	*	*	*	*	port6, port7, port9

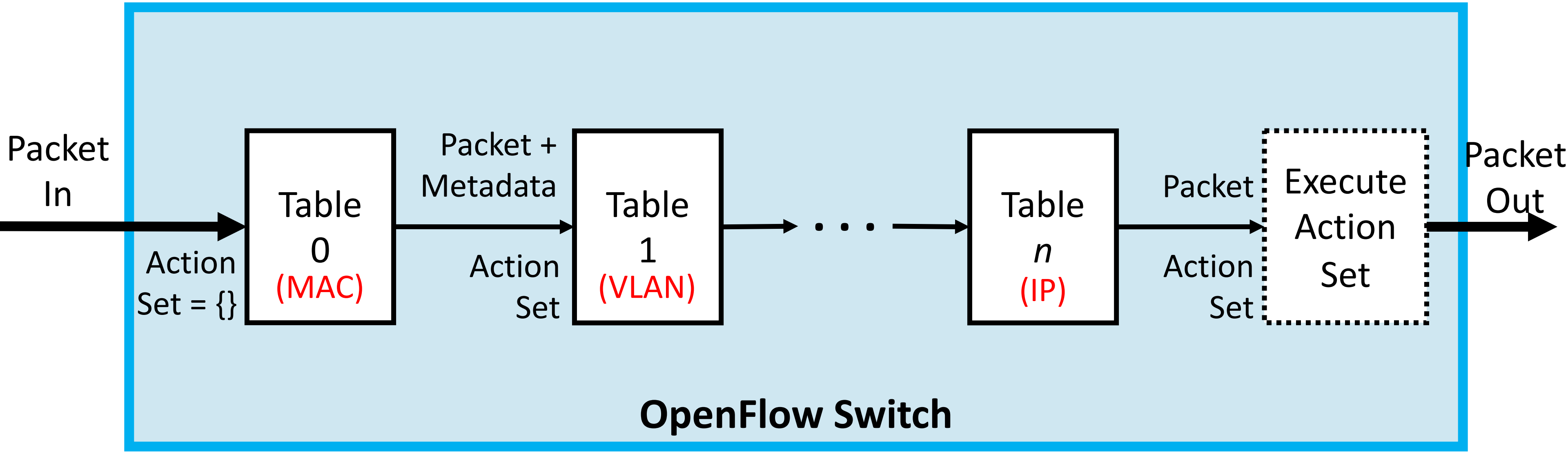
多级流表和流水线处理

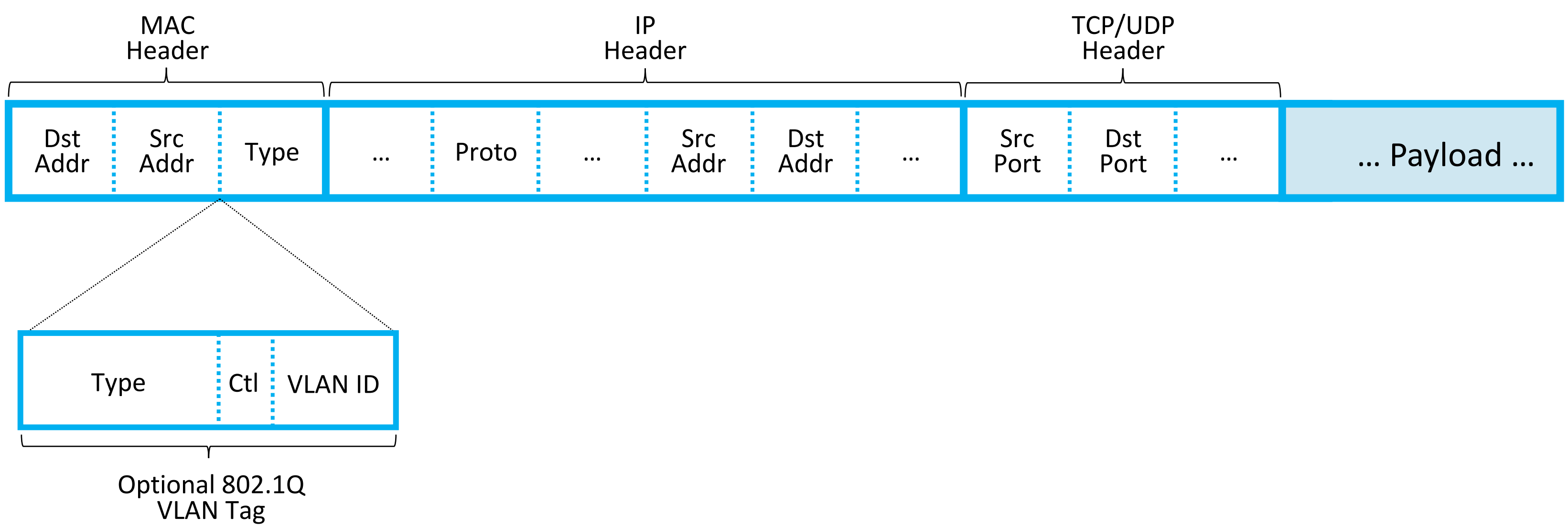
- ☐OpenFlow v1.0 使用单个流表来匹配数据包
  - ✓实现很简单，但随着网络需求越来越复杂，流表变得相当大
  - ✓控制平面的管理更加困难，并且对硬件提出了更高的要求

- ☐OpenFlow v1.1 及更高版本支持多级流表和管道处理
  - ▪ 多级流表可以对数据包进行复杂的处理
  - ▪ 可以减少单个流表的长度，从而提高条目查找效率



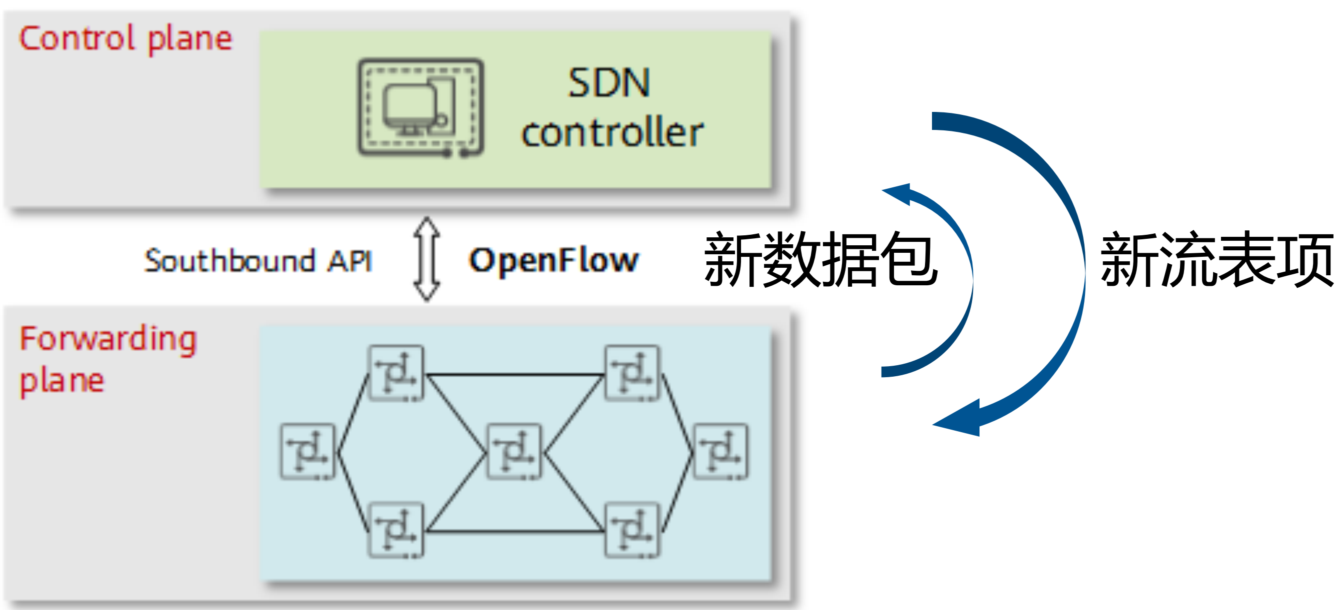
多级流表和流水线处理


Packet
In
Action
Set = {}
Table
0
(MAC)
Packet +
Metadata
Action
Set
Table
1
(VLAN)
Table
n
(IP)
Packet
Action
Set
Execute
Action
Set
Packet
Out
OpenFlow Switch

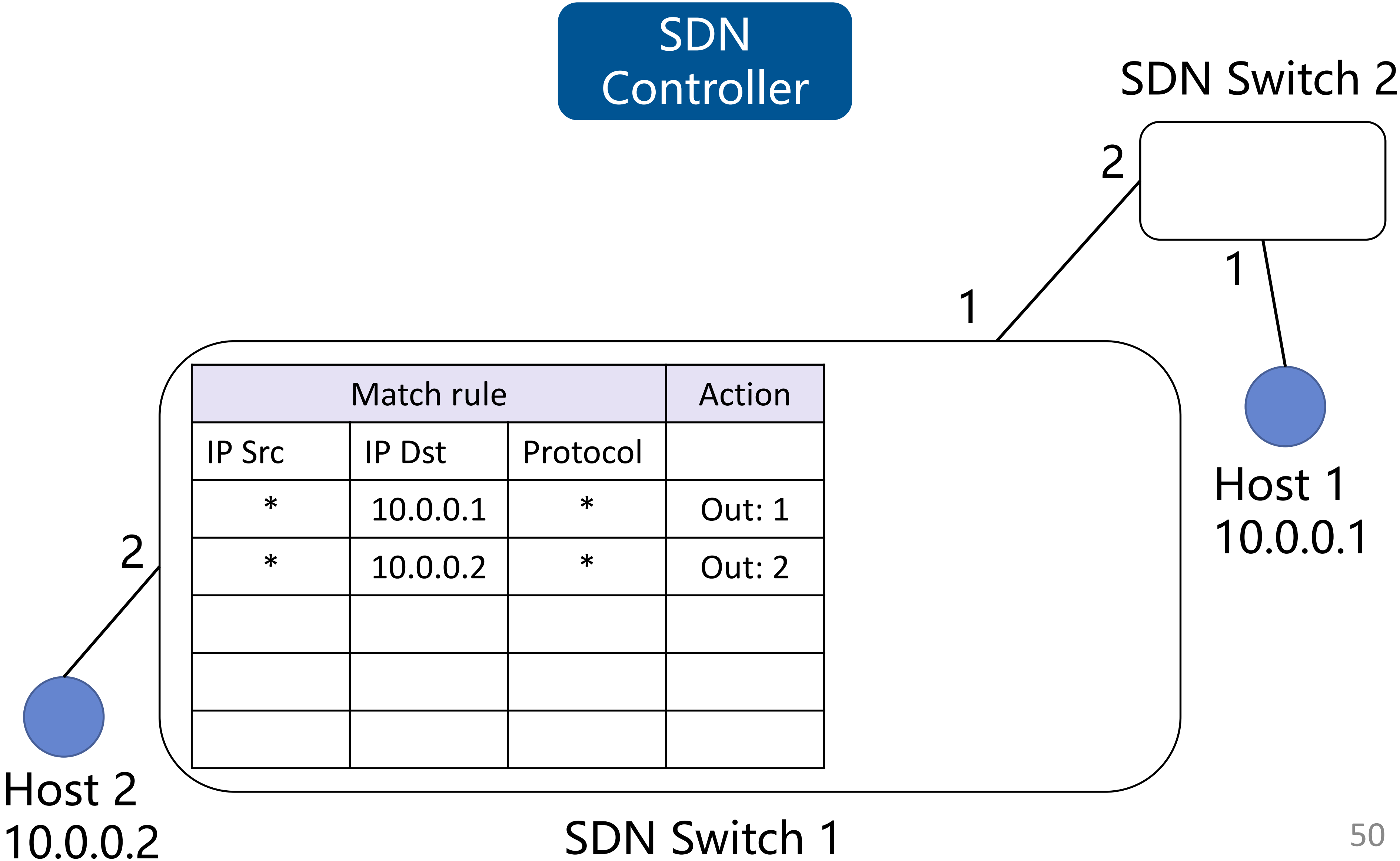

MAC
Header
IP
Header
TCP/UDP
Header
Dst
Addr
Src
Addr
Type
...
Proto
...
Src
Addr
Dst
Addr
...
Src
Port
Dst
Port
...
... Payload ...
Type
Ctl
VLAN ID
Optional 802.1Q
VLAN Tag

流表传递（Flow Table Delivery）

- 控制器创建一个或多个流表项，然后发送这些条目到交换机的流表中，分为主动和被动模式
  - ✓在主动模式中，控制器主动向交换机发送流表信息
  - ✓在被动模式中，当交换机没有找到任何与接收到的数据包匹配的流表项时，它向控制器发送消息，控制器计算相应的表项并将其传送到交换机

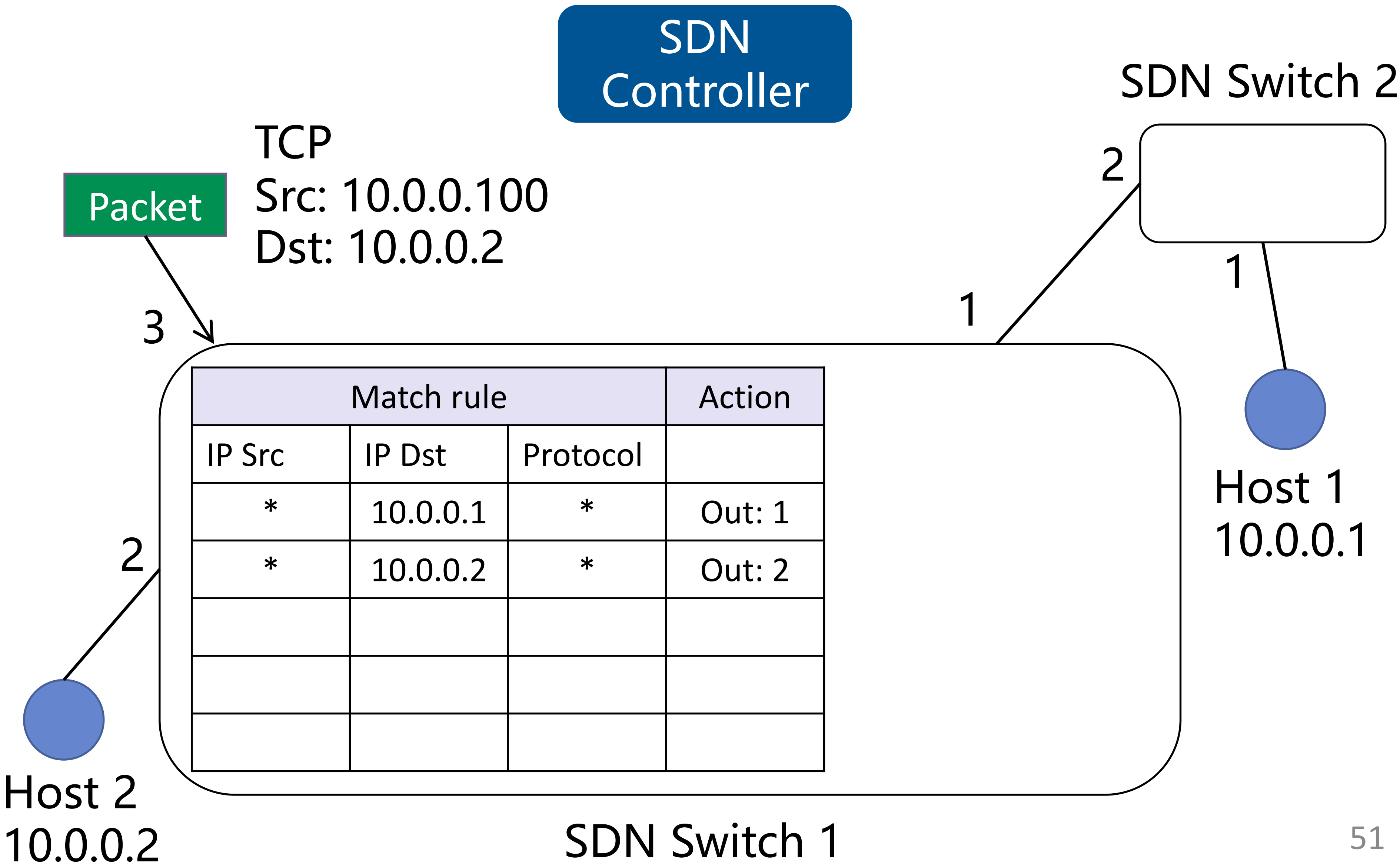


OpenFlow 例子



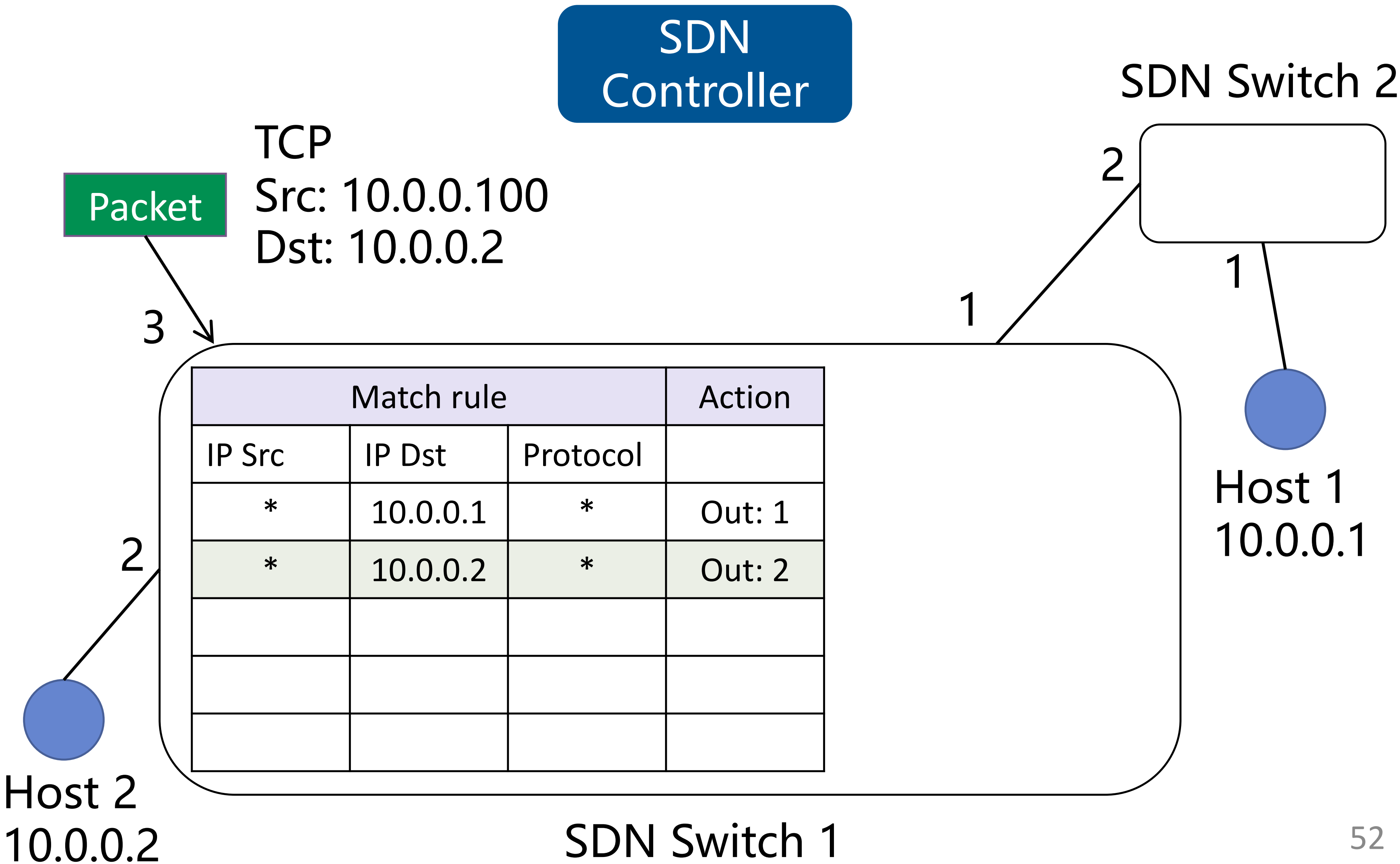
Match rule			Action
IP Src	IP Dst	Protocol	
*	10.0.0.1	*	Out: 1
*	10.0.0.2	*	Out: 2

OpenFlow 例子



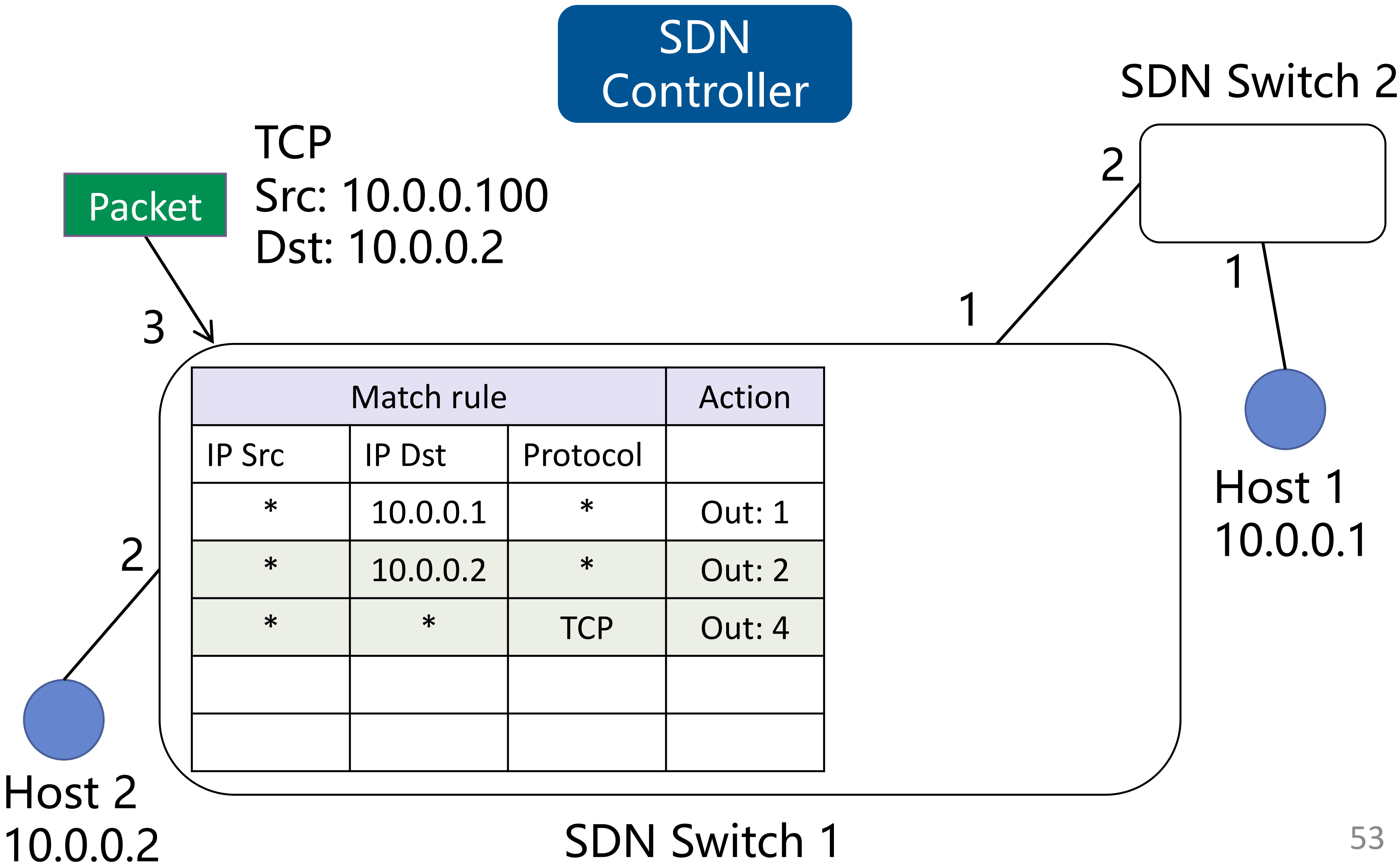
Match rule			Action
IP Src	IP Dst	Protocol	
*	10.0.0.1	*	Out: 1
*	10.0.0.2	*	Out: 2

OpenFlow 例子



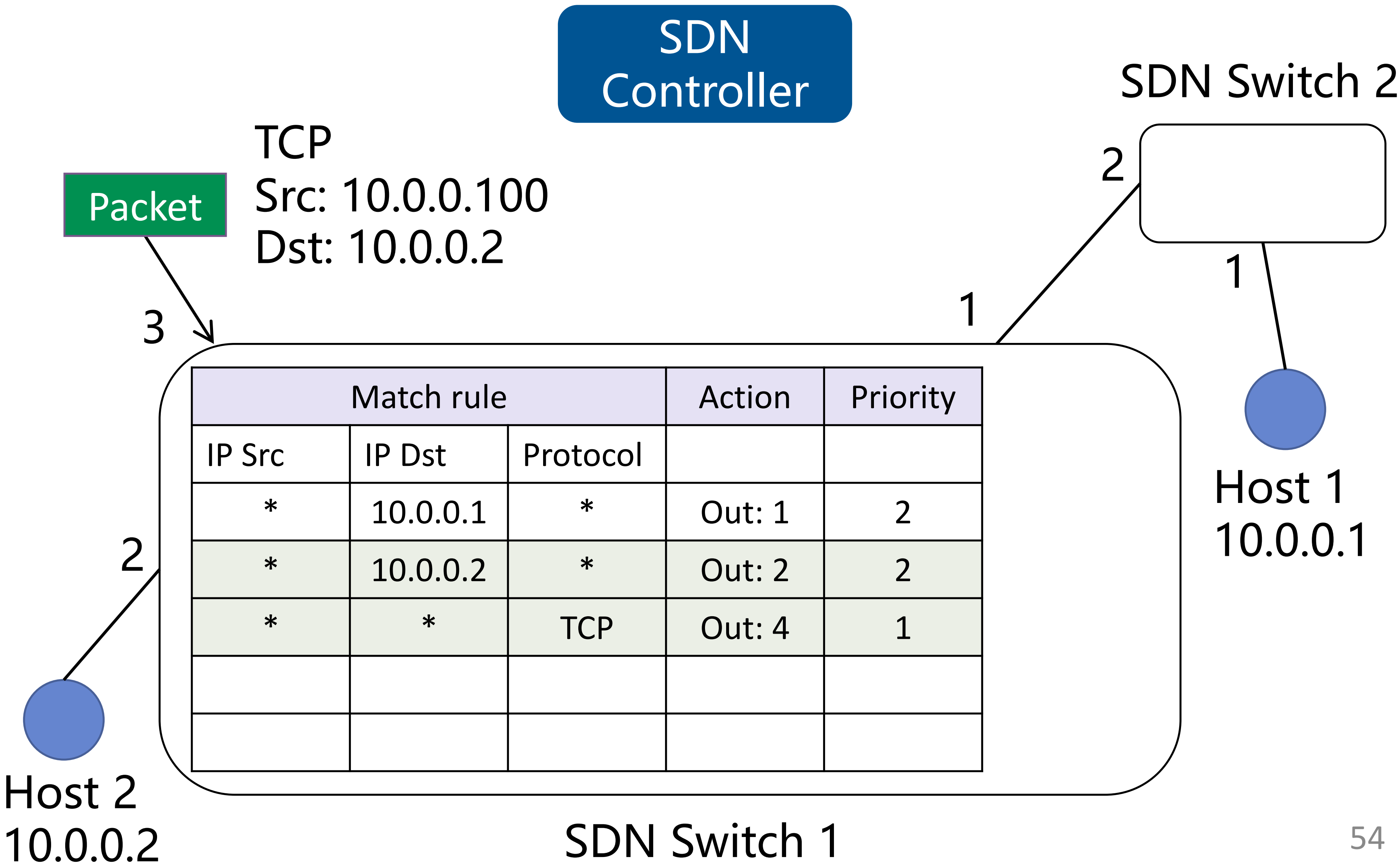
Match rule			Action
IP Src	IP Dst	Protocol	
*	10.0.0.1	*	Out: 1
*	10.0.0.2	*	Out: 2

OpenFlow 例子



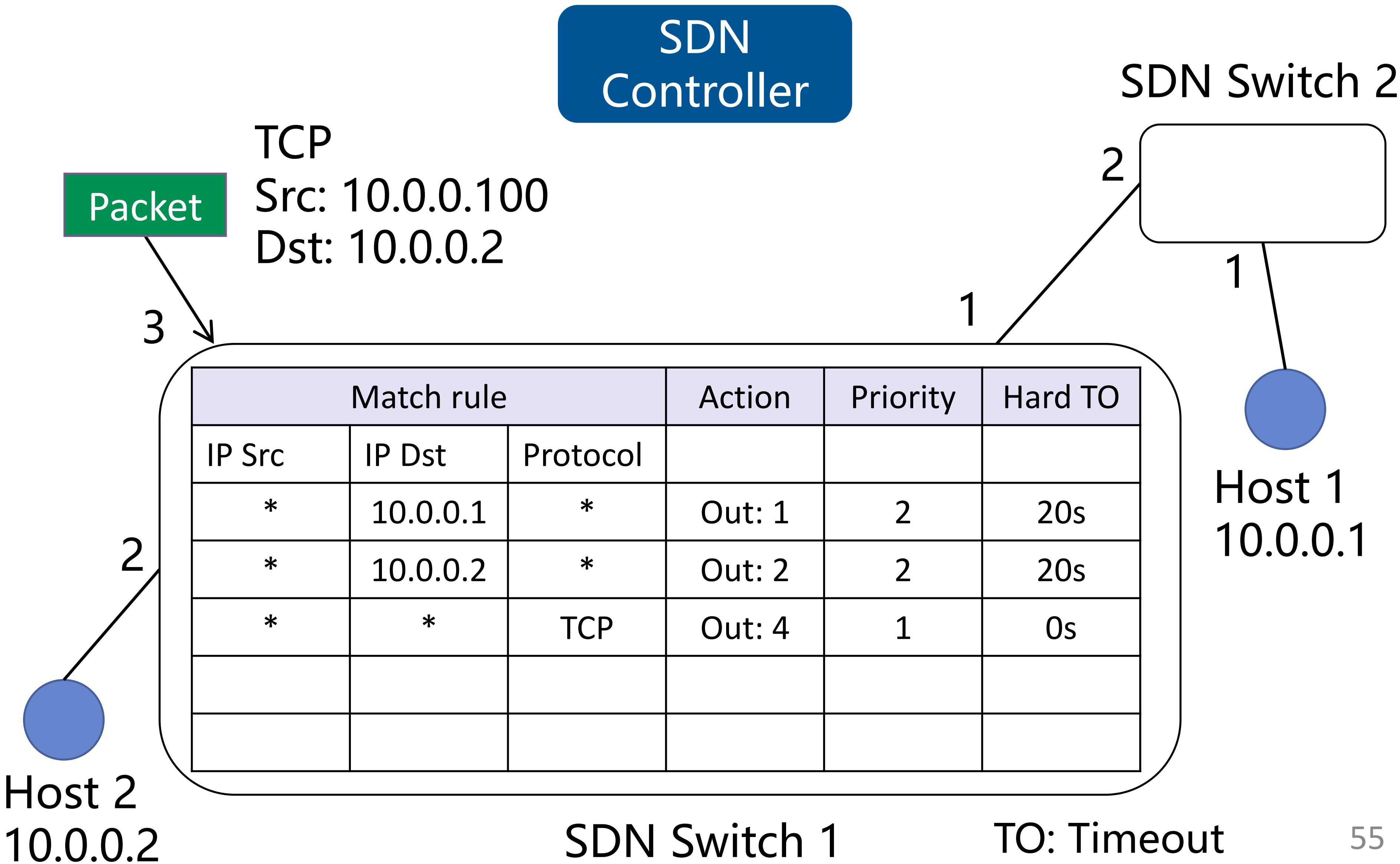
Match rule			Action
IP Src	IP Dst	Protocol	
*	10.0.0.1	*	Out: 1
*	10.0.0.2	*	Out: 2
*	*	TCP	Out: 4

OpenFlow 例子



Match rule			Action	Priority
IP Src	IP Dst	Protocol		
*	10.0.0.1	*	Out: 1	2
*	10.0.0.2	*	Out: 2	2
*	*	TCP	Out: 4	1

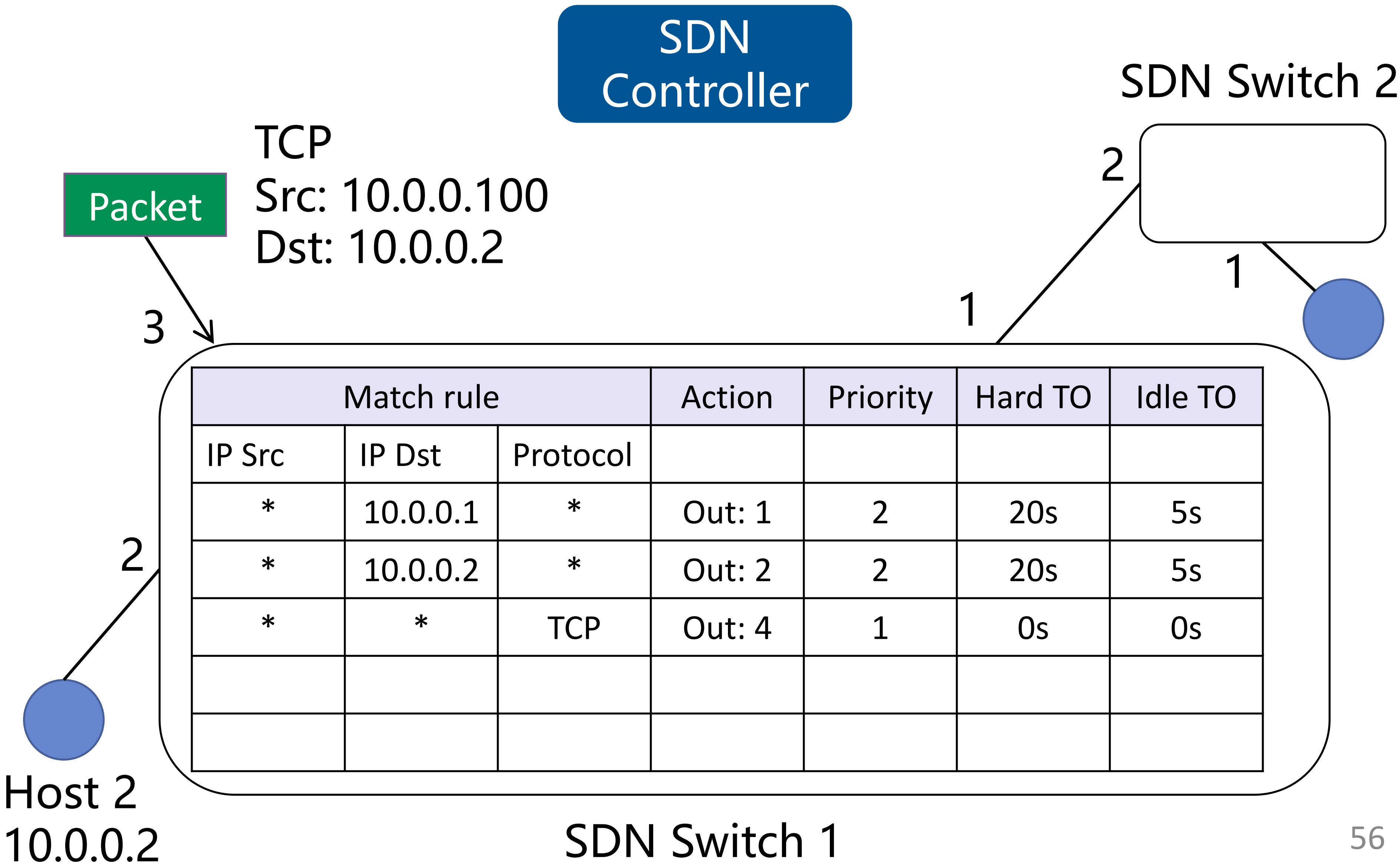
OpenFlow 例子



Match rule			Action	Priority	Hard TO
IP Src	IP Dst	Protocol			
*	10.0.0.1	*	Out: 1	2	20s
*	10.0.0.2	*	Out: 2	2	20s
*	*	TCP	Out: 4	1	0s

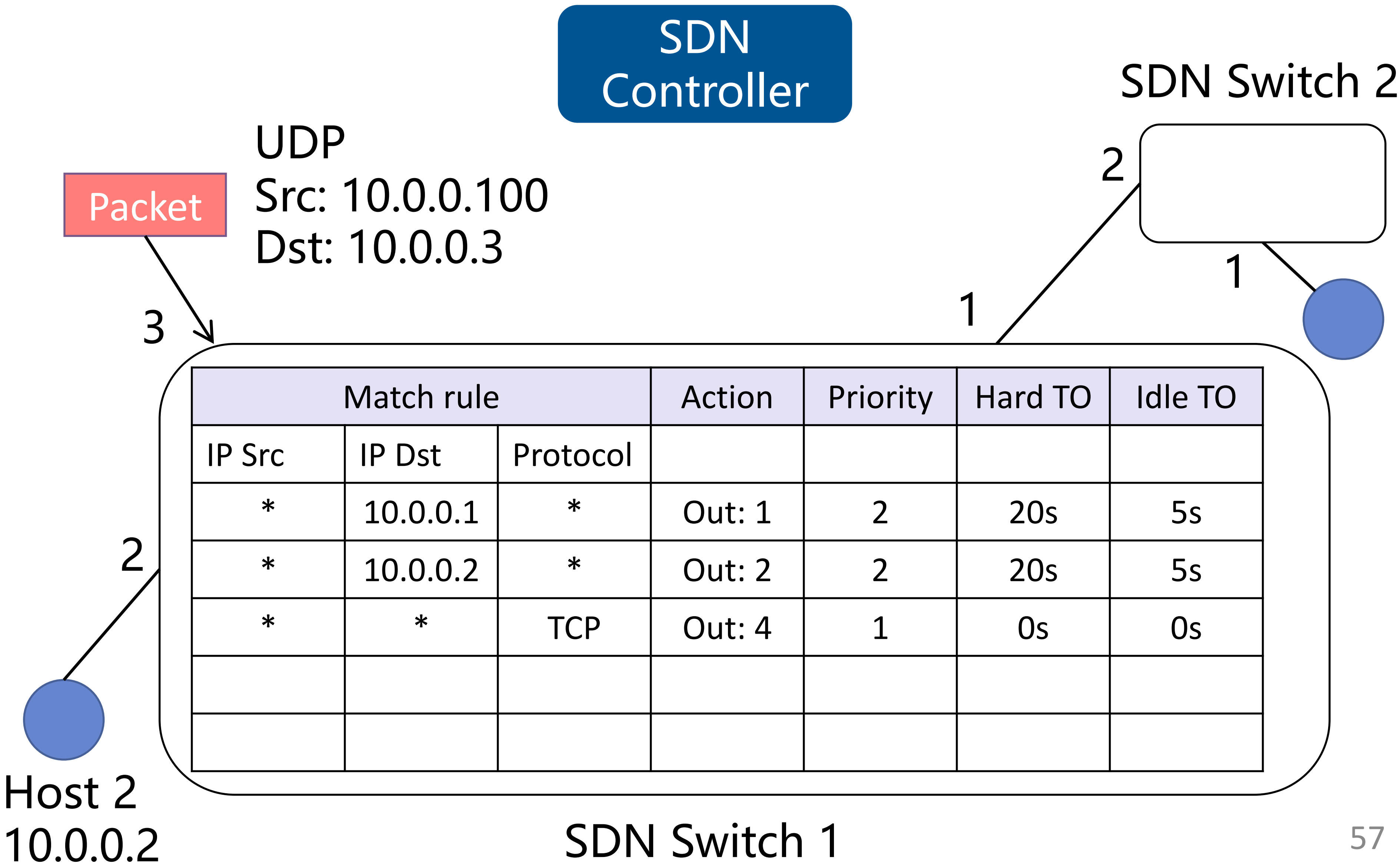
TO: Timeout

OpenFlow 例子



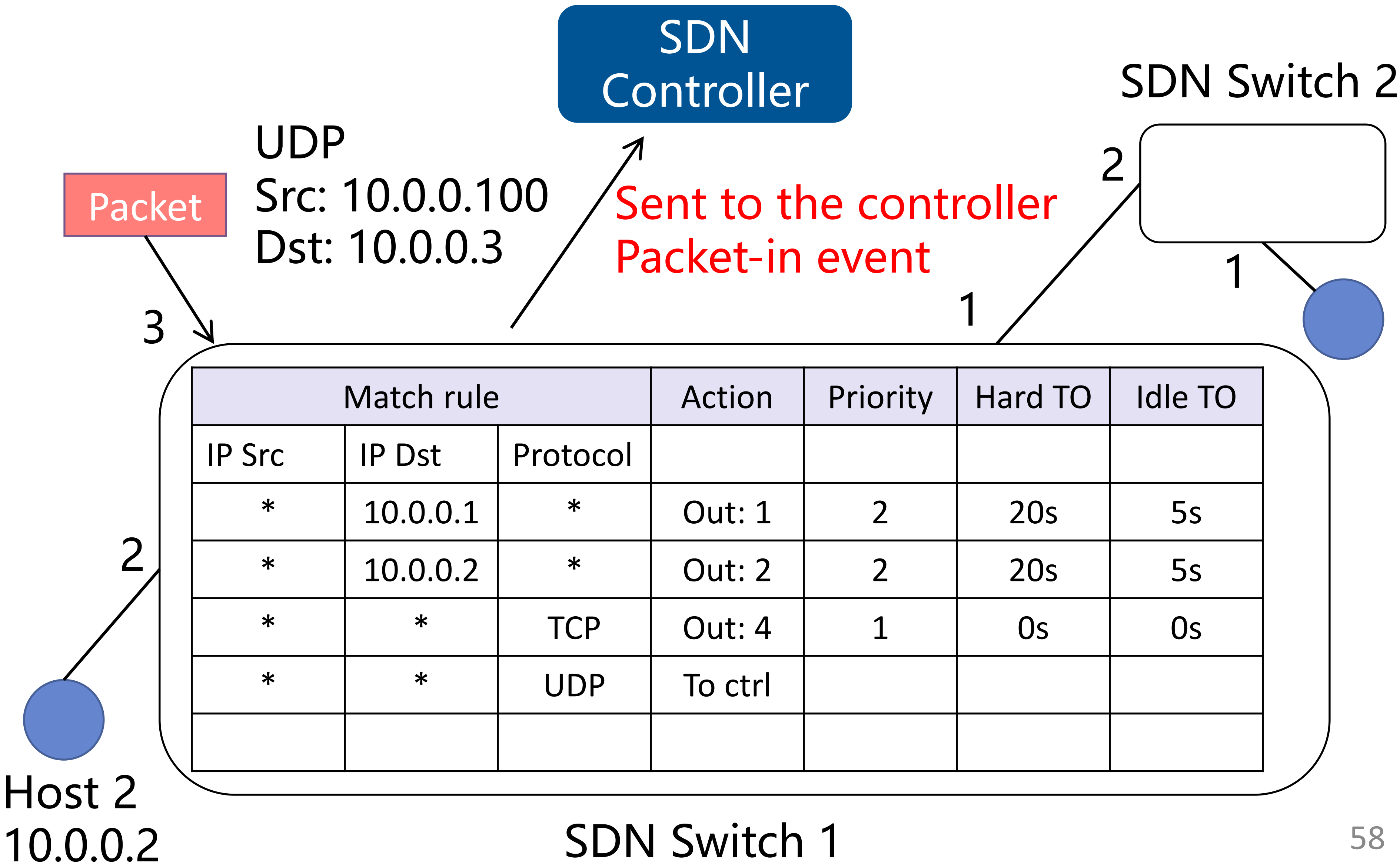
Match rule			Action	Priority	Hard TO	Idle TO
IP Src	IP Dst	Protocol				
*	10.0.0.1	*	Out: 1	2	20s	5s
*	10.0.0.2	*	Out: 2	2	20s	5s
*	*	TCP	Out: 4	1	0s	0s

OpenFlow 例子



Match rule			Action	Priority	Hard TO	Idle TO
IP Src	IP Dst	Protocol				
*	10.0.0.1	*	Out: 1	2	20s	5s
*	10.0.0.2	*	Out: 2	2	20s	5s
*	*	TCP	Out: 4	1	0s	0s

OpenFlow 例子



Match rule			Action	Priority	Hard TO	Idle TO
IP Src	IP Dst	Protocol				
*	10.0.0.1	*	Out: 1	2	20s	5s
*	10.0.0.2	*	Out: 2	2	20s	5s
*	*	TCP	Out: 4	1	0s	0s
*	*	UDP	To ctrl			

OpenFlow 例子

SDN Controller

SDN Switch 2

UDP
Src: 10.0.0.100
Dst: 10.0.0.3

Packet

Reply with a new rule

3

2

1

2

1

Match rule			Action	Priority	Hard TO	Idle TO
IP Src	IP Dst	Protocol				
*	10.0.0.1	*	Out: 1	2	20s	5s
*	10.0.0.2	*	Out: 2	2	20s	5s
*	*	TCP	Out: 4	1	0s	0s
*	*	UDP	To ctrl			

2

Host 2
10.0.0.2

SDN Switch 1

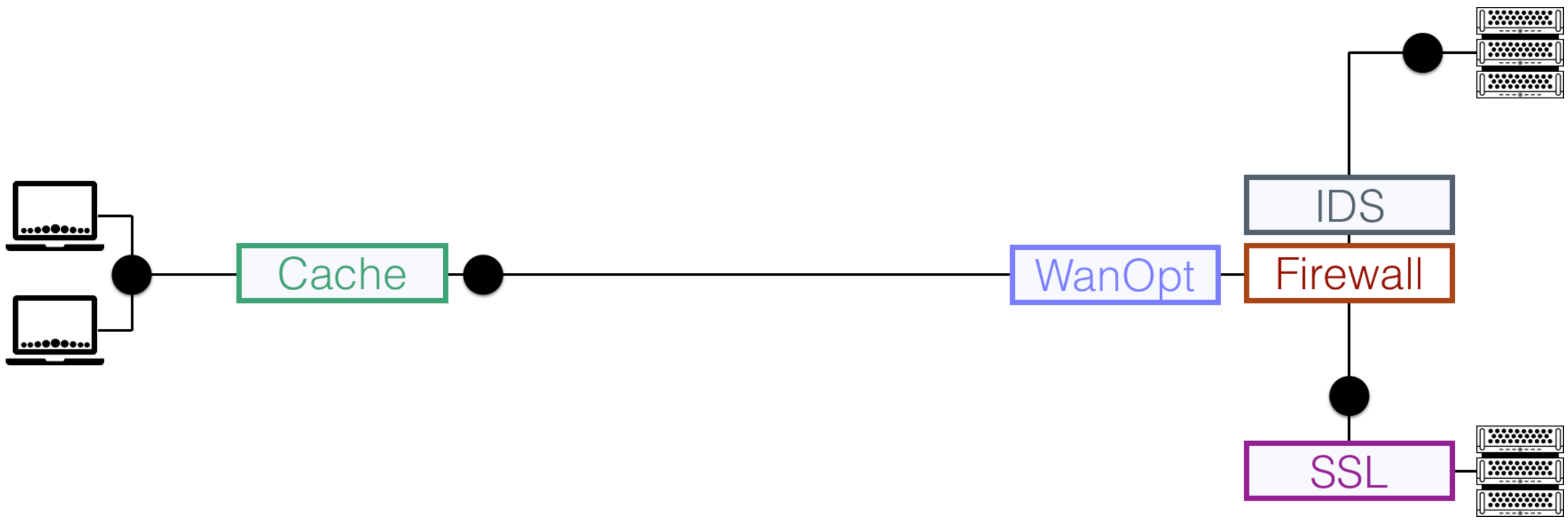
网络功能虚拟化

网络的传统视角

□这样的网络仅提供数据传输 (Data delivery) 功能

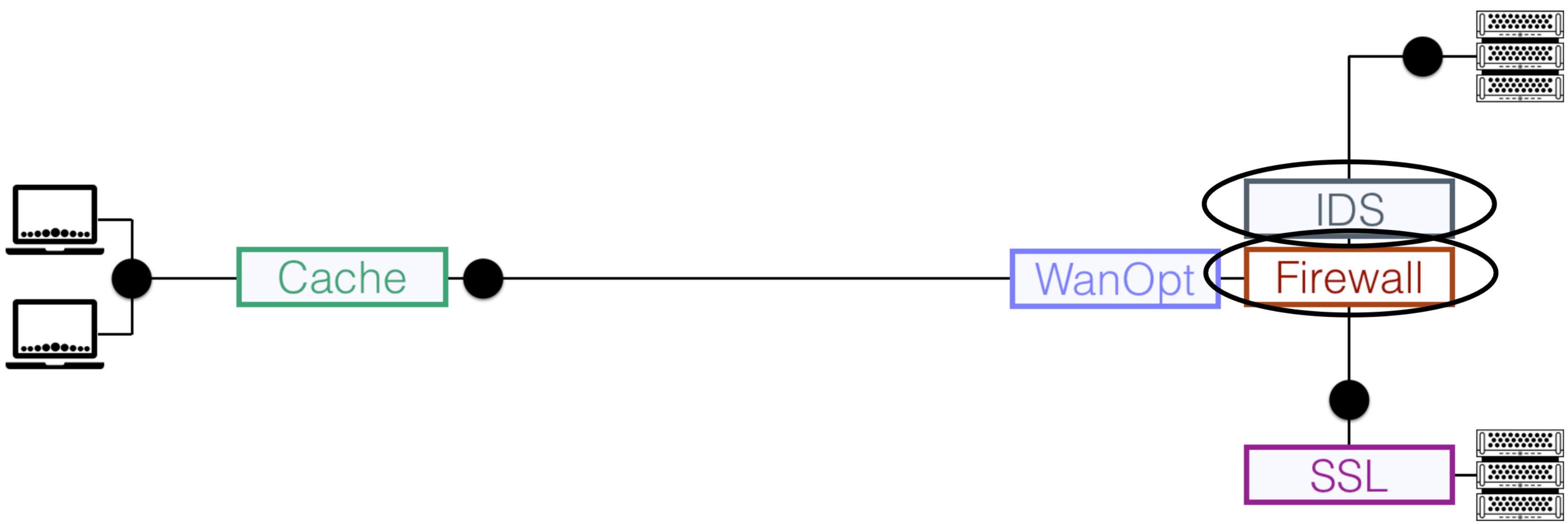
网络中间设备

- 在实际网络中，数据传输不是唯一需要的功能



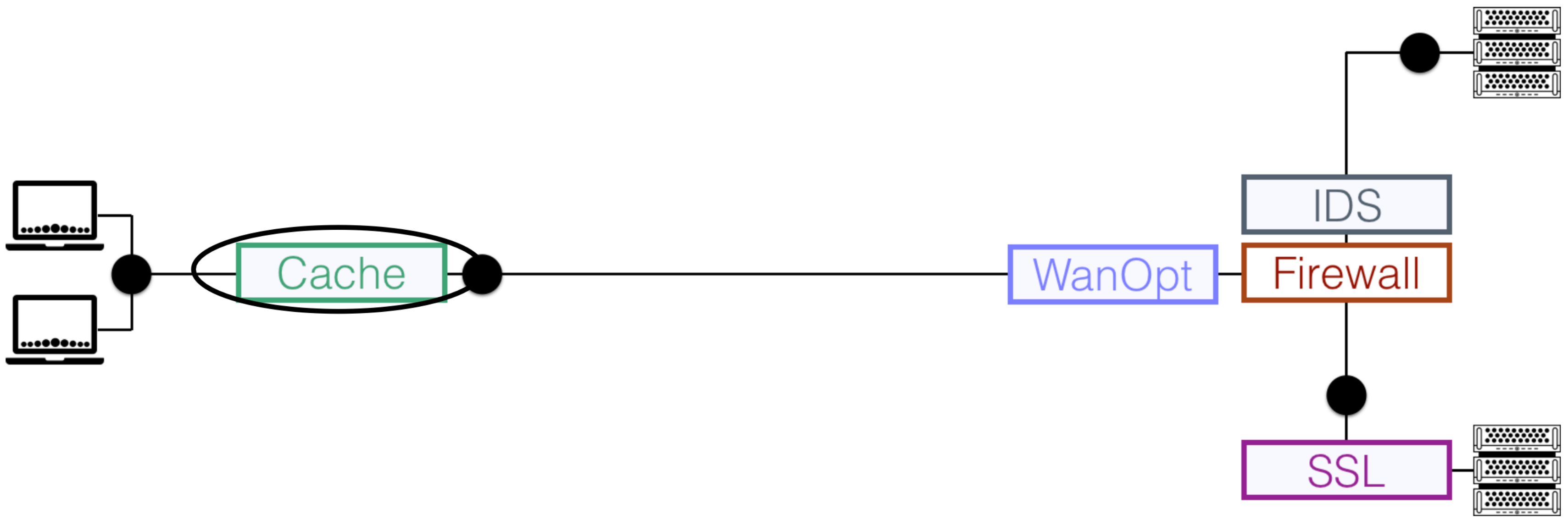
网络中间设备

- **安全 Security**：识别并拦截不想要的流量
  - IDS (入侵检测系统，Intrusion Detection System)
  - Firewall (防火墙)



网络中间设备

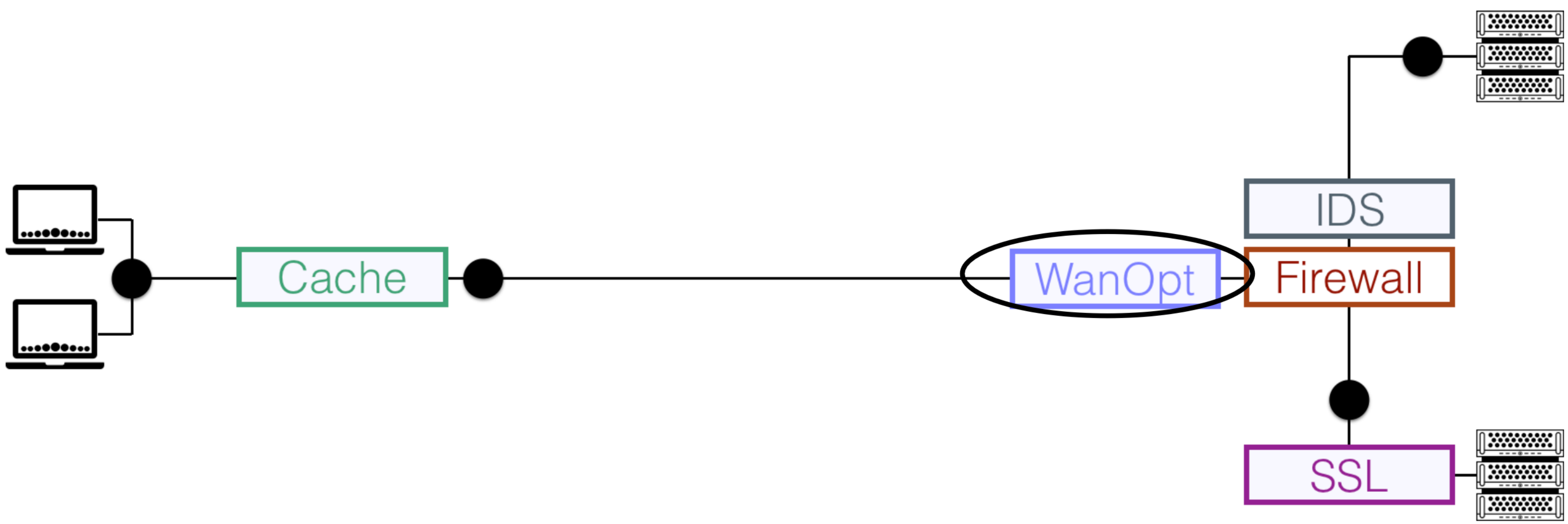
- **性能 Performance**：更快地加载内容
  - Cache (缓存)



网络中间设备

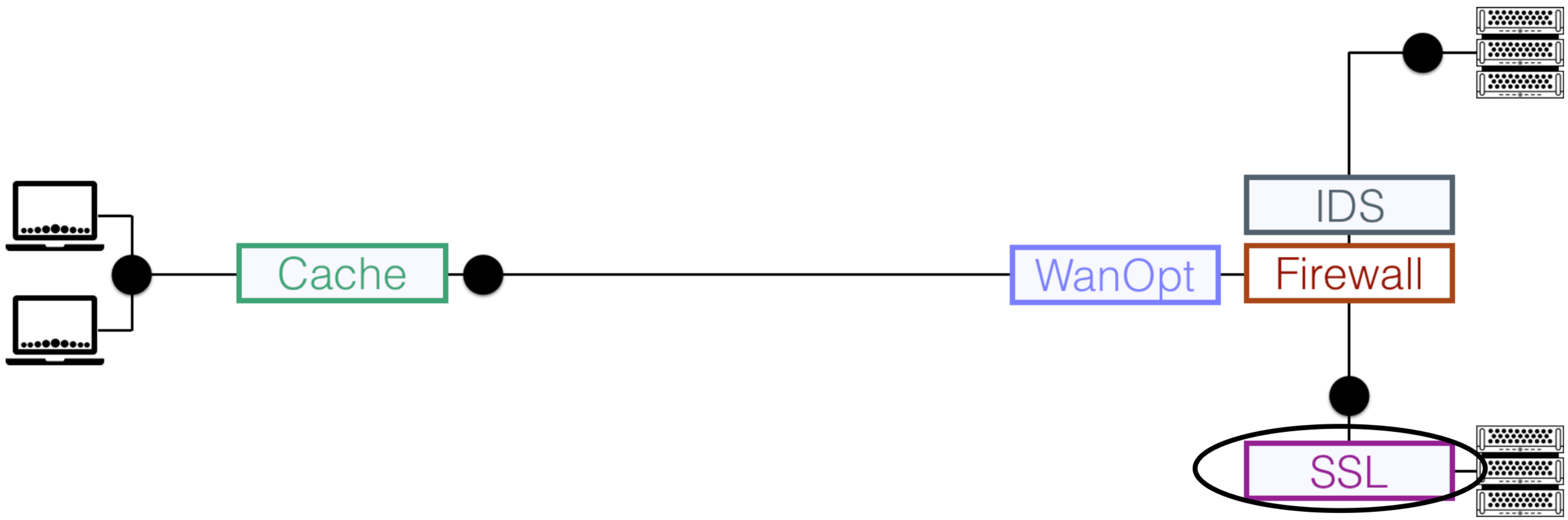
□性能 Performance：减少带宽使用

- WANOpt (广域网优化，Wide Area Network Optimization)



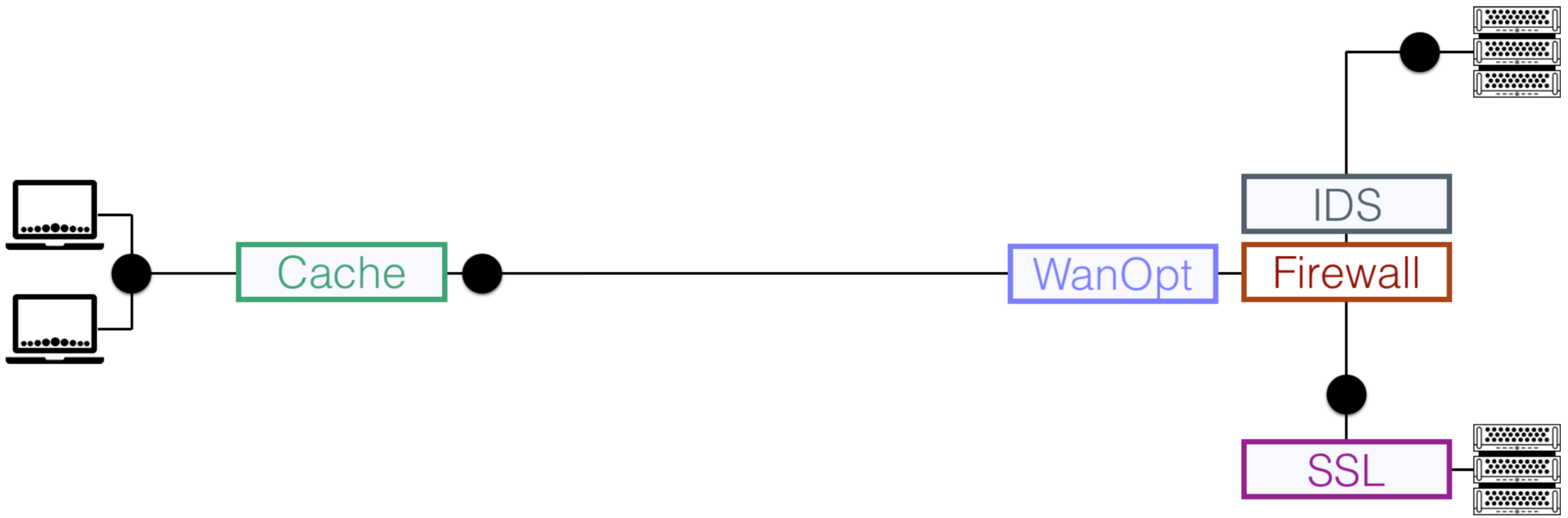
网络中间设备

- 应用支持 Application support：支持过时应用的协议
  - SSL (安全套接层，Secure Socket Layer)



网络中间设备

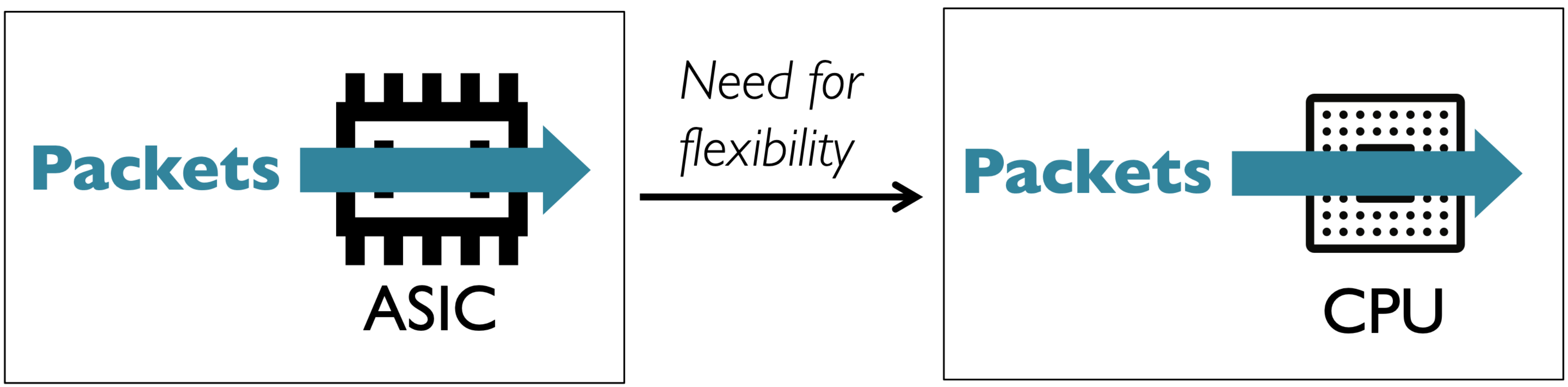
- 企业中所有网络设备的三分之一都是中间设备!
  - Sherry et. al., SIGCOMM’12



网络中间设备的变革

Dedicated hardware

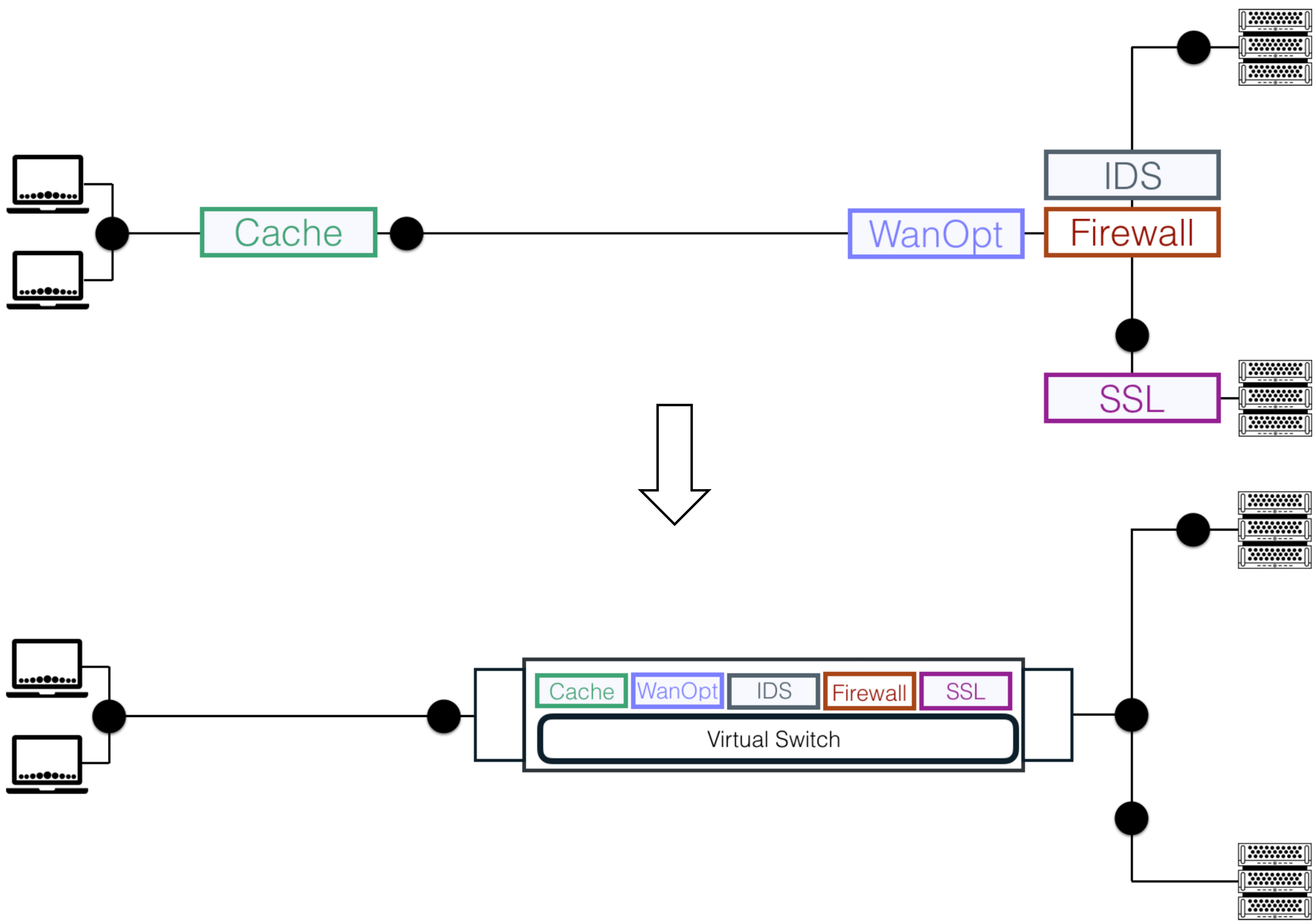
Software



Middleboxes

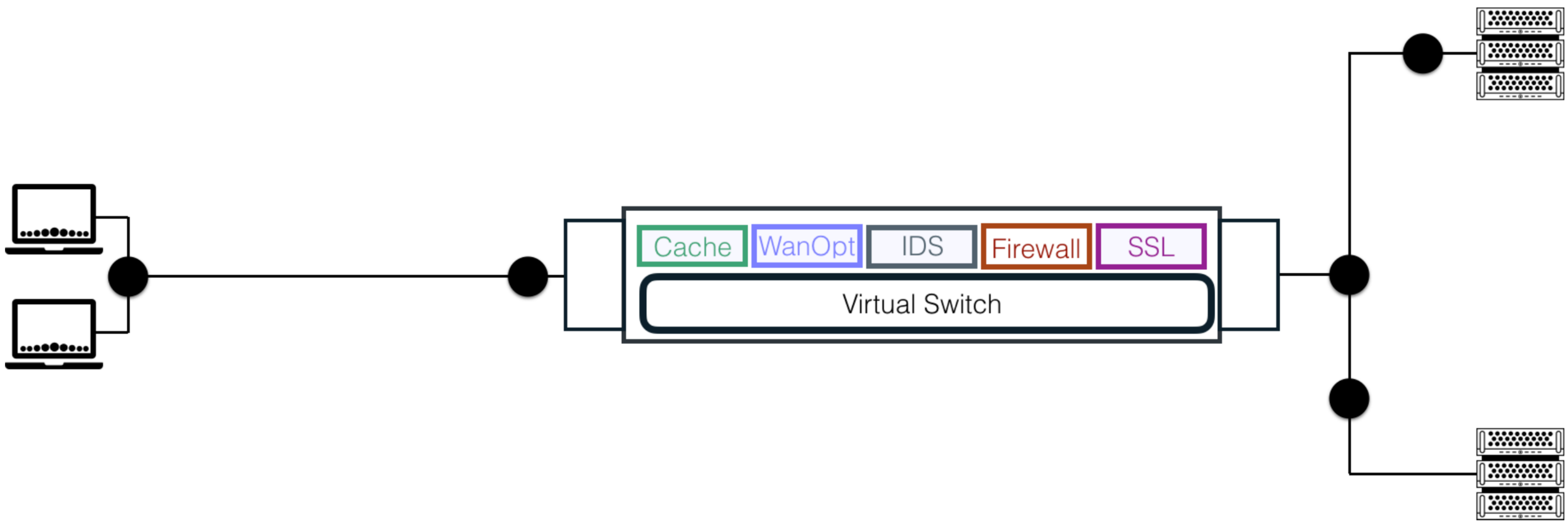
Network functions

网络中间设备 -> 网络功能


Cache
WanOpt
IDS
Firewall
SSL
Cache
WanOpt
IDS
Firewall
SSL
Virtual Switch

网络功能虚拟化

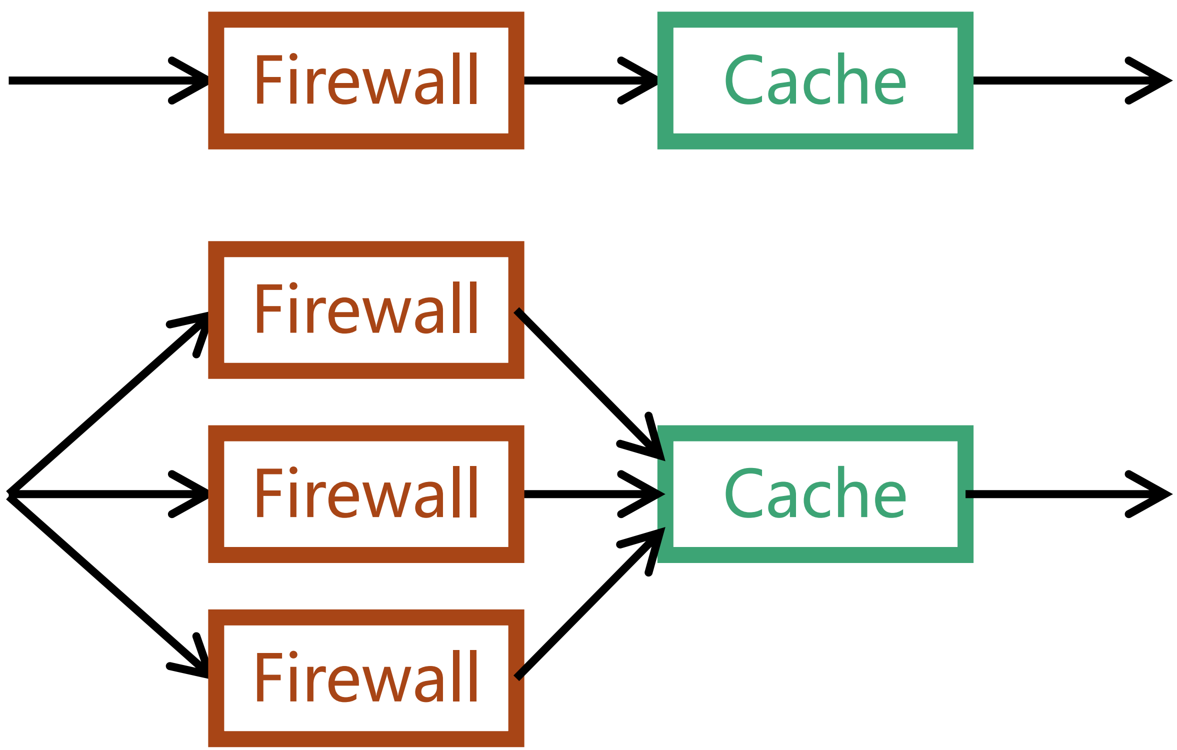
- NFV (Network Function Virtualization)
  - 主要部署在虚拟机中



网络功能虚拟化的优势

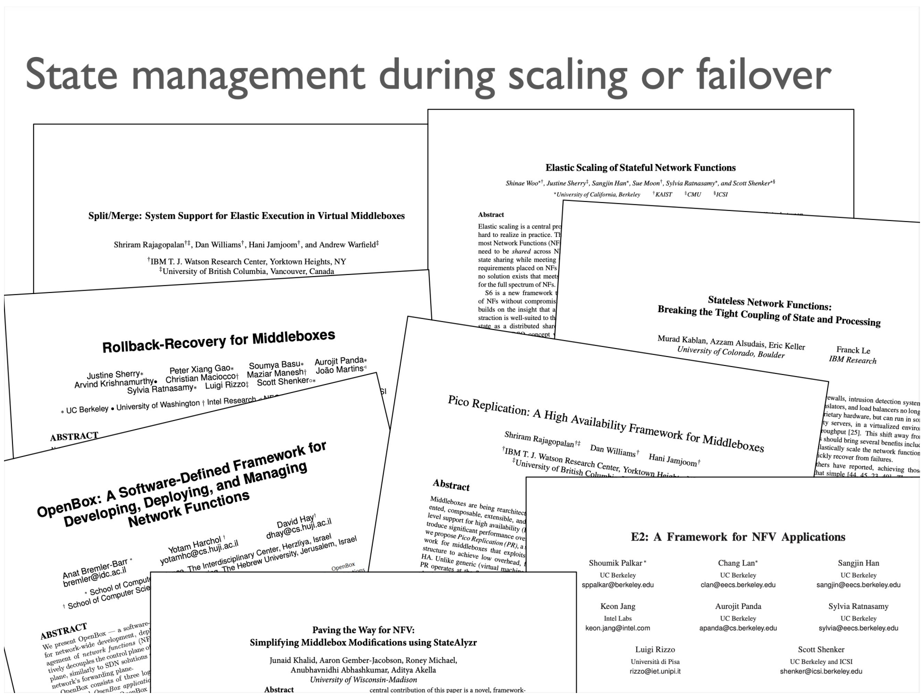
- 可编程性 – 更新和创造新网络功能的能力

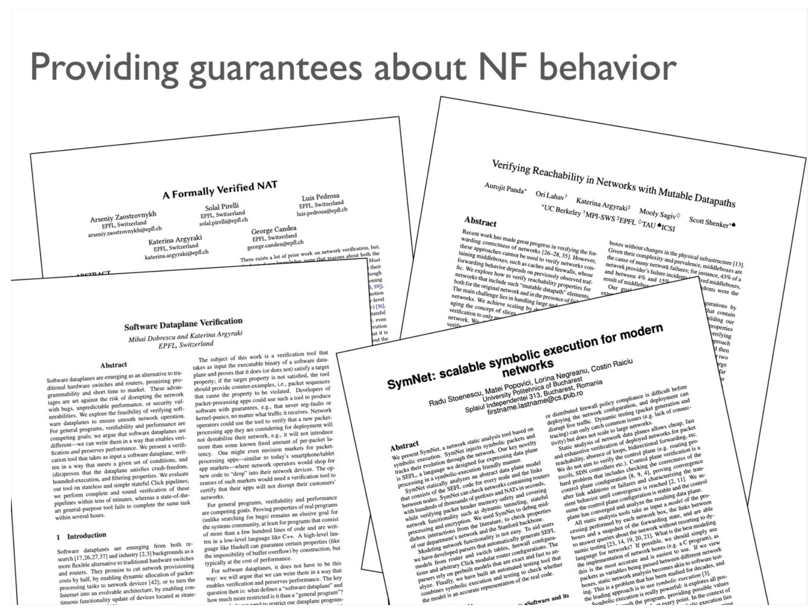
- 易于部署、配置和管理



网络功能虚拟化的代价

- 复杂的状态管理
- 无法预测的性能
- 性能下降





中山大學 软件工程学院
SUN YAT-SEN UNIVERSITY SCHOOL OF SOFTWARE ENGINEERING

谢谢

陈壮彬
软件工程学院
chenzhb36@mail.sysu.edu.cn