



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

SSE316 : 云计算技术 Cloud Computing Technology

陈壮彬

软件工程学院

<https://zbchern.github.io/sse316.html>



软件定义网络

- ❖ 控制面、管理面和数据面
- ❖ 传统网络架构
- ❖ 软件定义网络
- ❖ OpenFlow 协议



软件定义网络

- ❖ 控制面、管理面和数据面
- ❖ 传统网络架构
- ❖ 软件定义网络
- ❖ OpenFlow 协议

网络的三个平面



管理平面 (Management Plane)

控制平面 (Control Plane)

数据平面 (Data Plane)



管理面

- 网络管理员与网络互动的方式，允许管理员配置和管理网络设备的接口
- 这是输入命令和进行配置更改的地方。例如，登录路由器以设置新的网络规则（谁能连接Wi-Fi）即在使用管理平面

管理面



- 它有点像你车上的仪表板——它不会让汽车移动，但它可以让你控制它的移动方式和位置



控制面

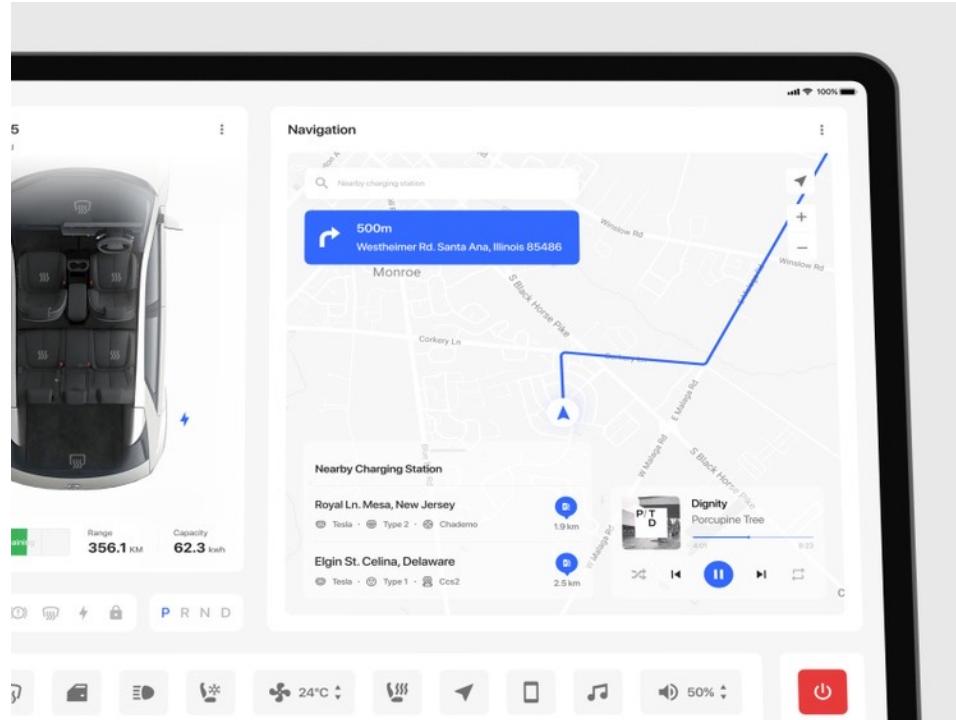


- 一旦网络被配置，控制平面就**负责**网络如何做出决策
- 它涉及到确定数据通过网络的路径的所有协议。例如，在路由器中，控制平面将负责了解网络拓扑，创建路由表，并帮助路由器决定将接收到的数据包发送到哪里

控制面



- 如果管理平面就像你汽车的仪表板（选择哪个地图软件），那么控制平面就像地图软件实际的运行，它可以帮助你的汽车决定到达目的地的最佳路线



数据/转发面



- 数据的**实际移动的地方**
- 一旦控制平面做出决定，数据平面就负责基于这些决定将数据包从其源实际移动到其目的地

数据/转发面



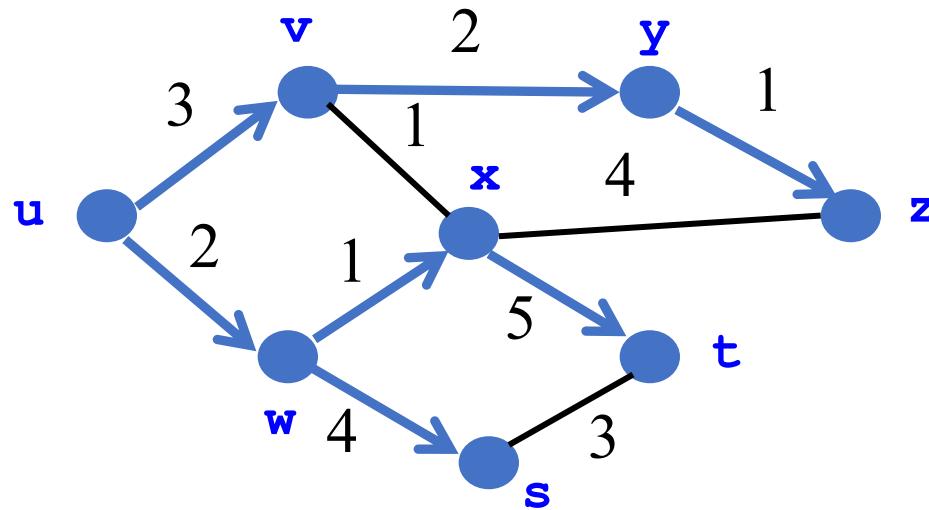
- 汽车的发动机和车轮实际上沿着地图软件（控制平面）确定的路径移动汽车



例子：最短距离路由



- 计算从源节点到所有节点的最短路径

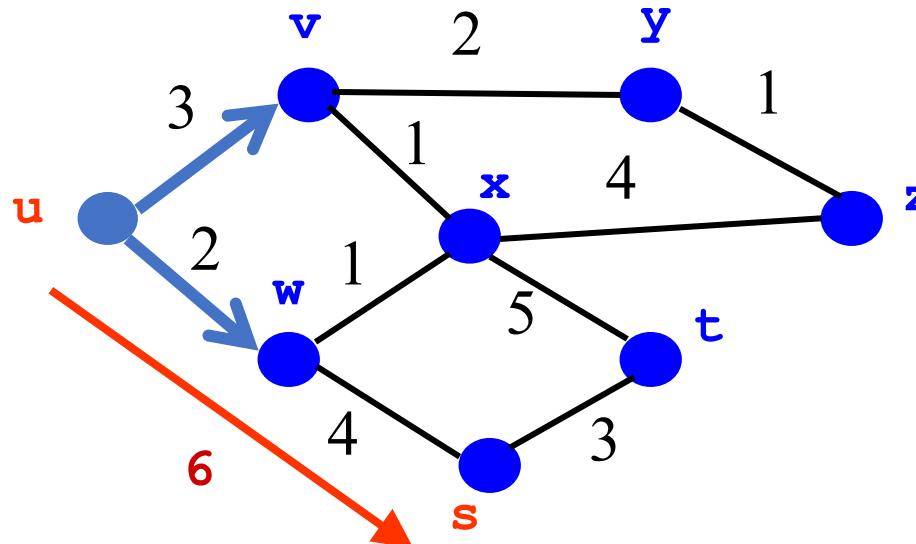


控制平面



- 链路状态路由

- ✓ 整个网络拓扑泛洪到所有节点：OSPF, IS-IS
- ✓ 每个节点计算最短路径
- ✓ Dijkstra 算法

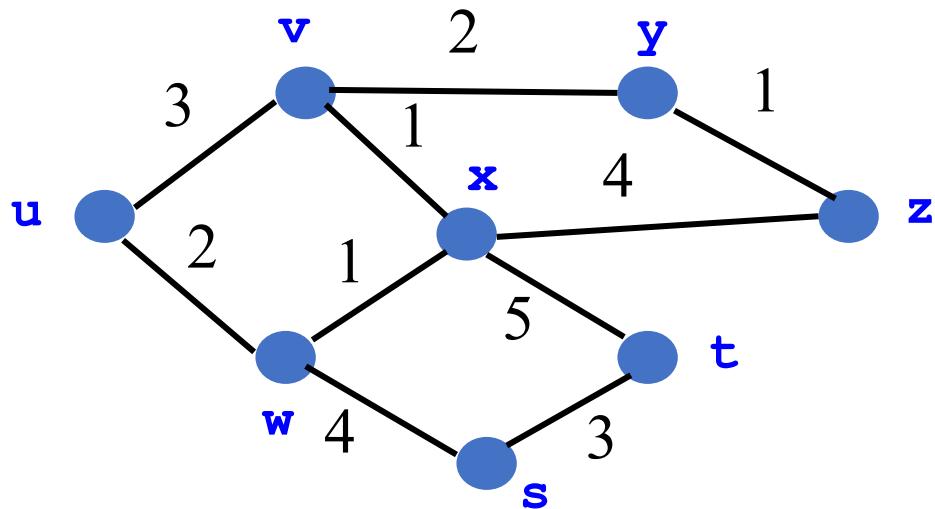


	link
v	(u,v)
w	(u,w)
x	(u,w)
y	(u,v)
z	(u,v)
s	(u,w)
t	(u,w)

控制平面



- 距离矢量路由
 - ✓ 每个节点计算路径成本
 - ✓ 基于每个邻居的路径成本做计算
 - ✓ Bellman-Ford 算法

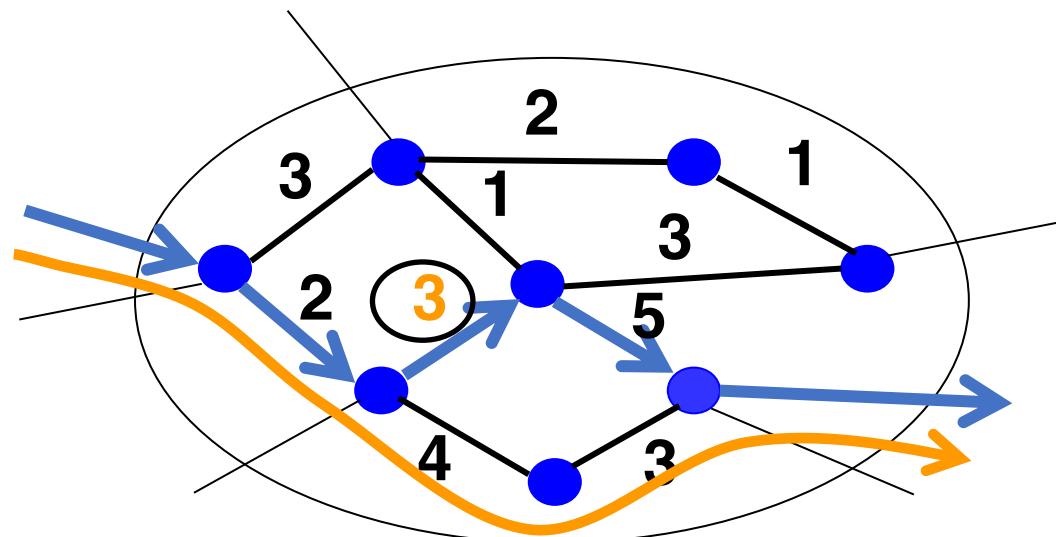


$$d_u(z) = \min\{c(u,v) + d_v(z), c(u,w) + d_w(z)\}$$

管理面



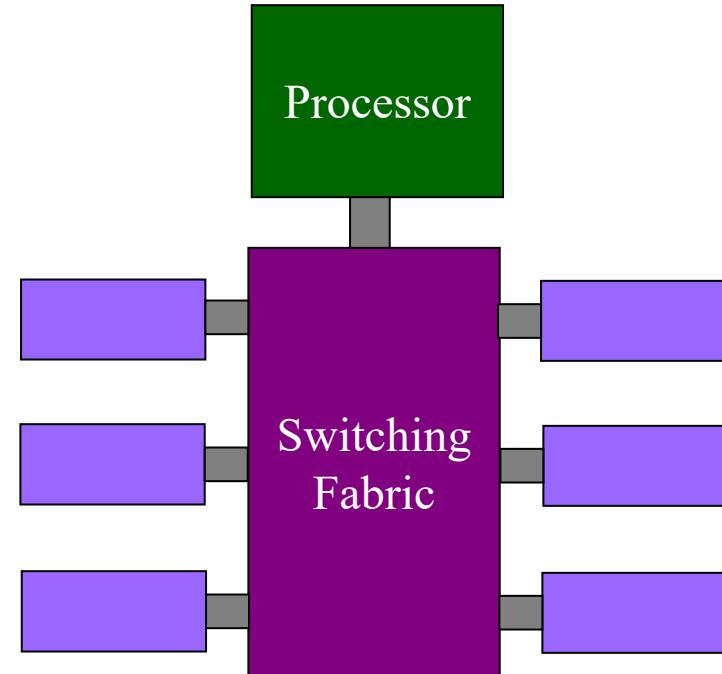
- 管理面选择路由算法
- 管理面设置路径成本
 - ✓ 与链路容量成反比 ?
 - ✓ 与传播延迟成比例 ?
 - ✓ 基于流量的全网优化 ?



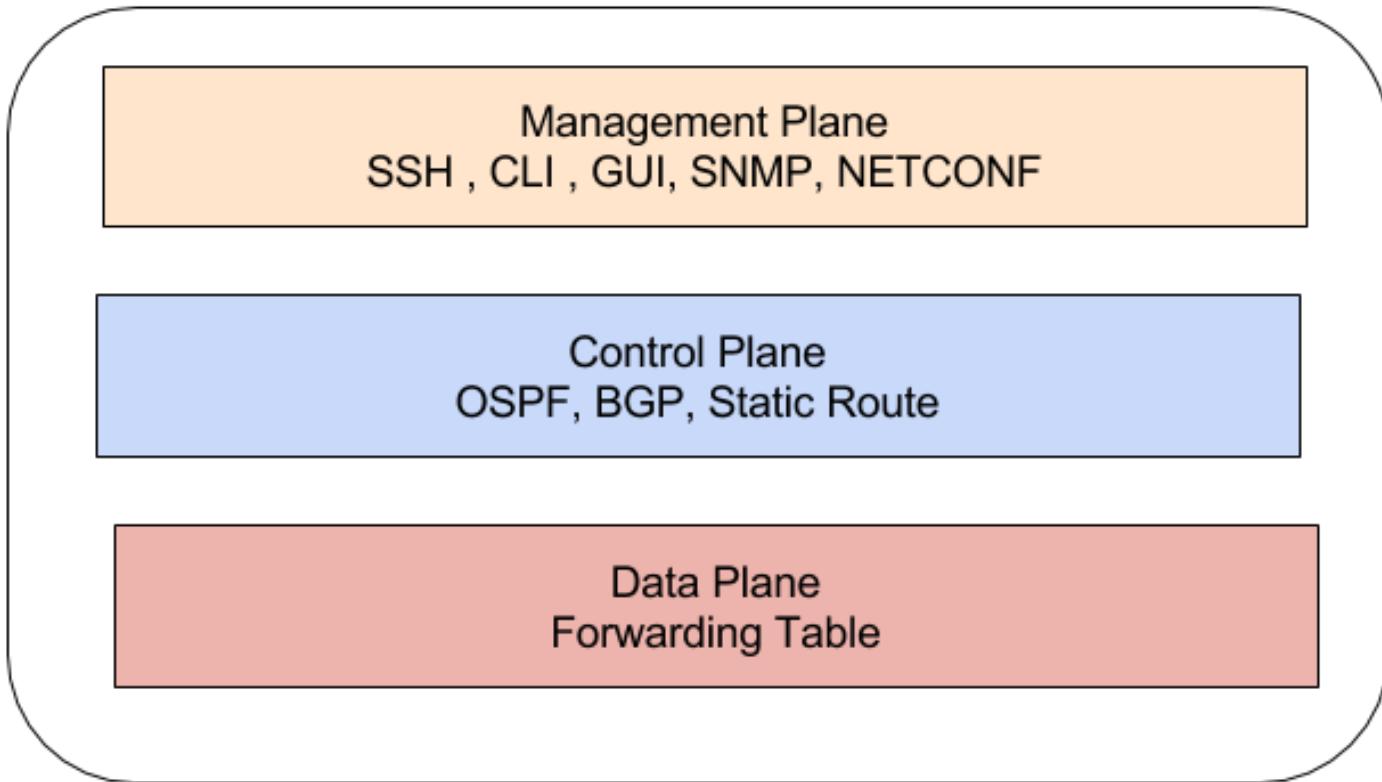
数据平面



- 对数据包进行操作
 - ✓ 规则匹配
 - ✓ 执行操作
- 操作举例
 - ✓ Forwarding
 - ✓ Access control
 - ✓ Mapping header fields
 - ✓ Traffic monitoring
 - ✓ Buffering and marking
 - ✓ Shaping and scheduling
 - ✓ Deep packet inspection



网路设备的三个方面





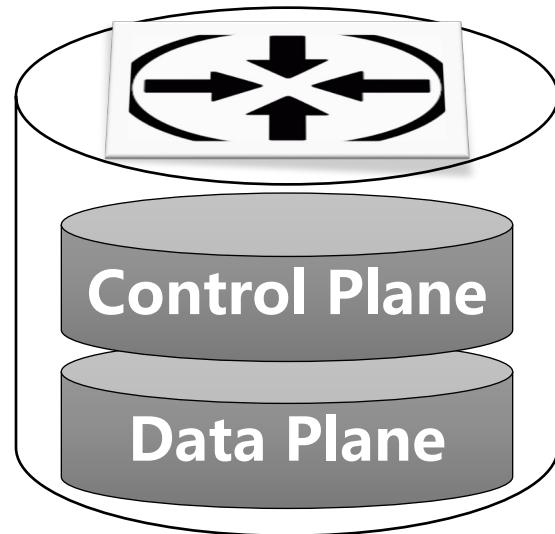
软件定义网络

- ❖ 控制面、管理面和数据面
- ❖ 传统网络架构
- ❖ 软件定义网络
- ❖ OpenFlow 协议

传统网络架构



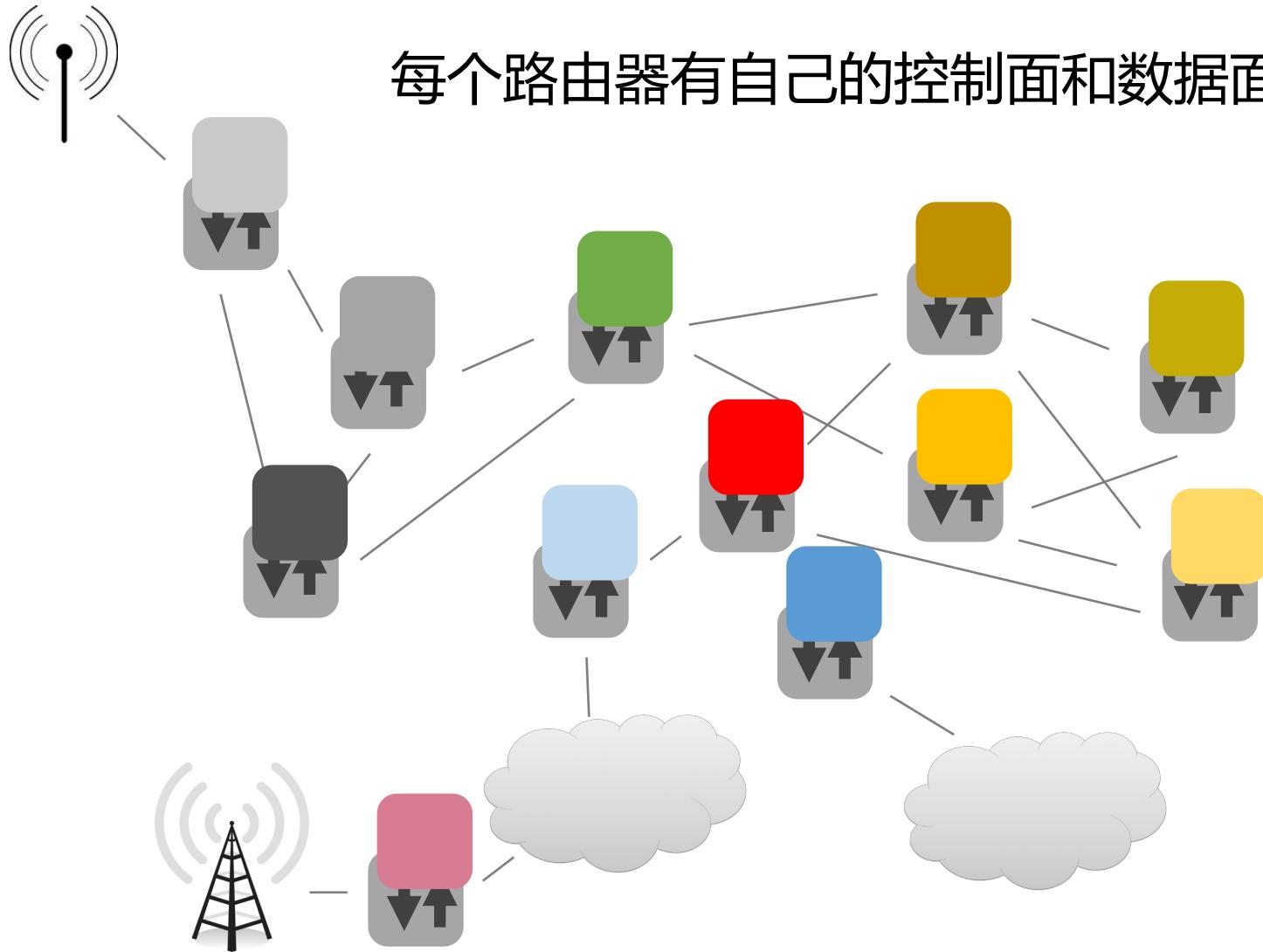
- 控制和数据平面位于物理设备内



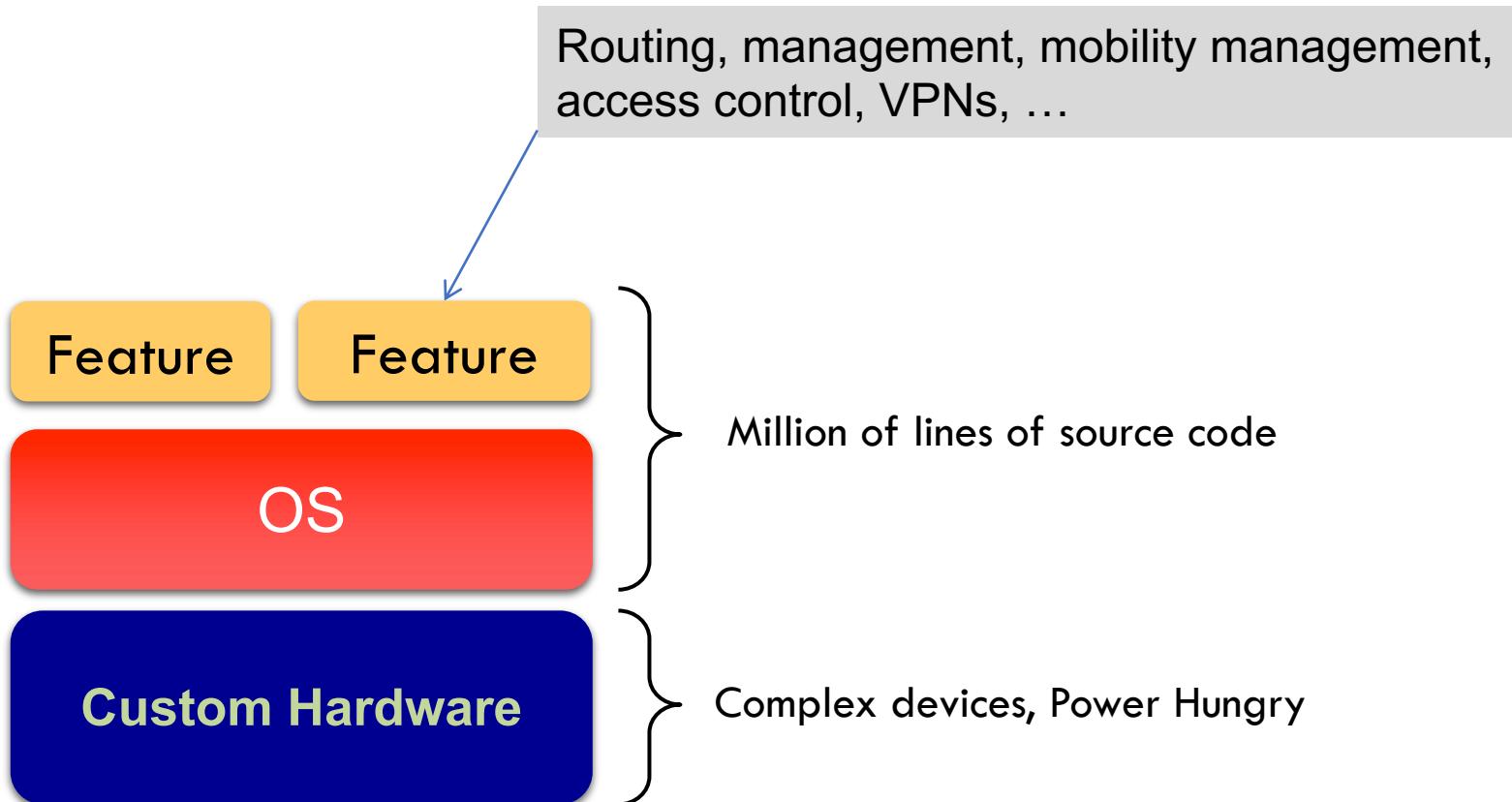
传统网络架构



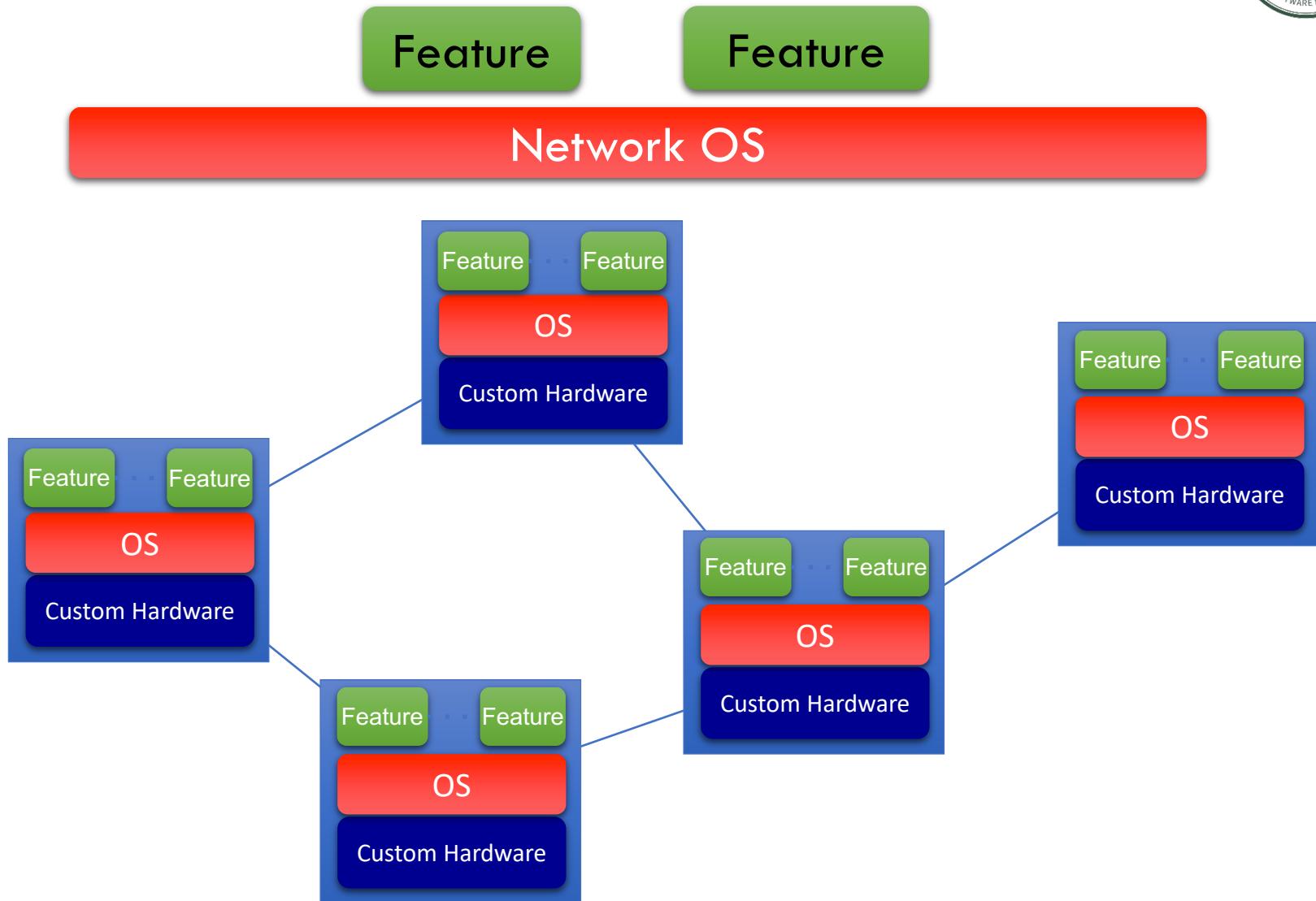
每个路由器有自己的控制面和数据面



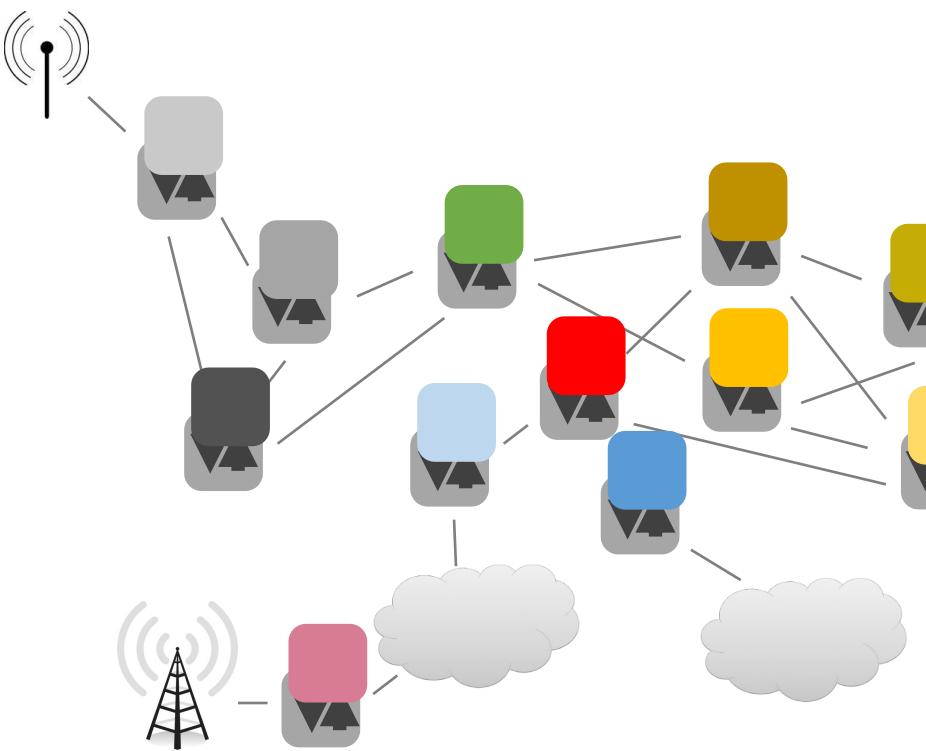
传统网络设备



传统网络架构



传统网络管理例子



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. ...

大型数据中心



百万级设备！

传统网络架构的缺点



- 缺乏抽象
- 无法表达意图（如提供高质量服务，重定向流量）
- 复杂分布式算法结果不可预测
- 除非设备配置正确，否则无法管理设备
 - ✓ 控制和管理平面取决于正确的数据平面
 - ✓ 脆弱性、变化风险
- 创新步伐缓慢（依赖硬件），所有厂家必须使用相同的协议
- 成本高、可拓展性弱...



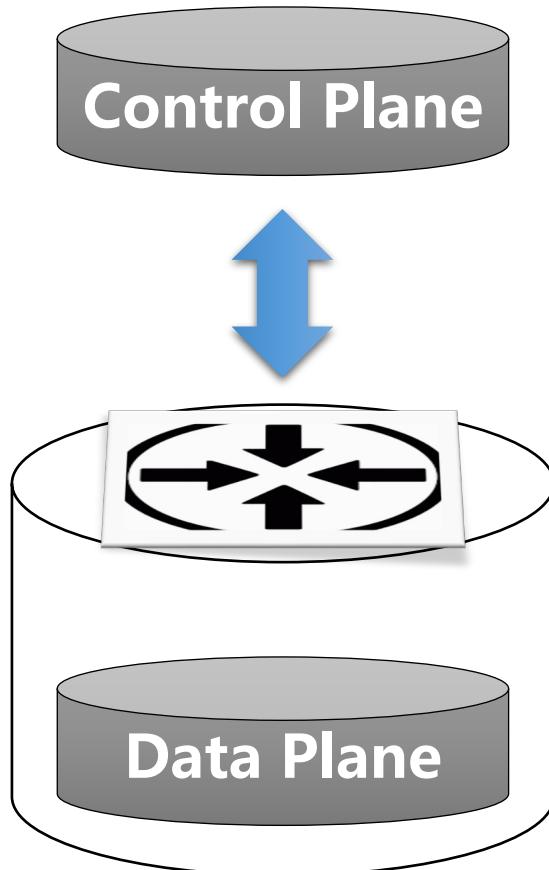
软件定义网络

- ❖ 控制面、管理面和数据面
- ❖ 传统网络架构
- ❖ 软件定义网络
- ❖ OpenFlow 协议

软件定义网络(SDN)



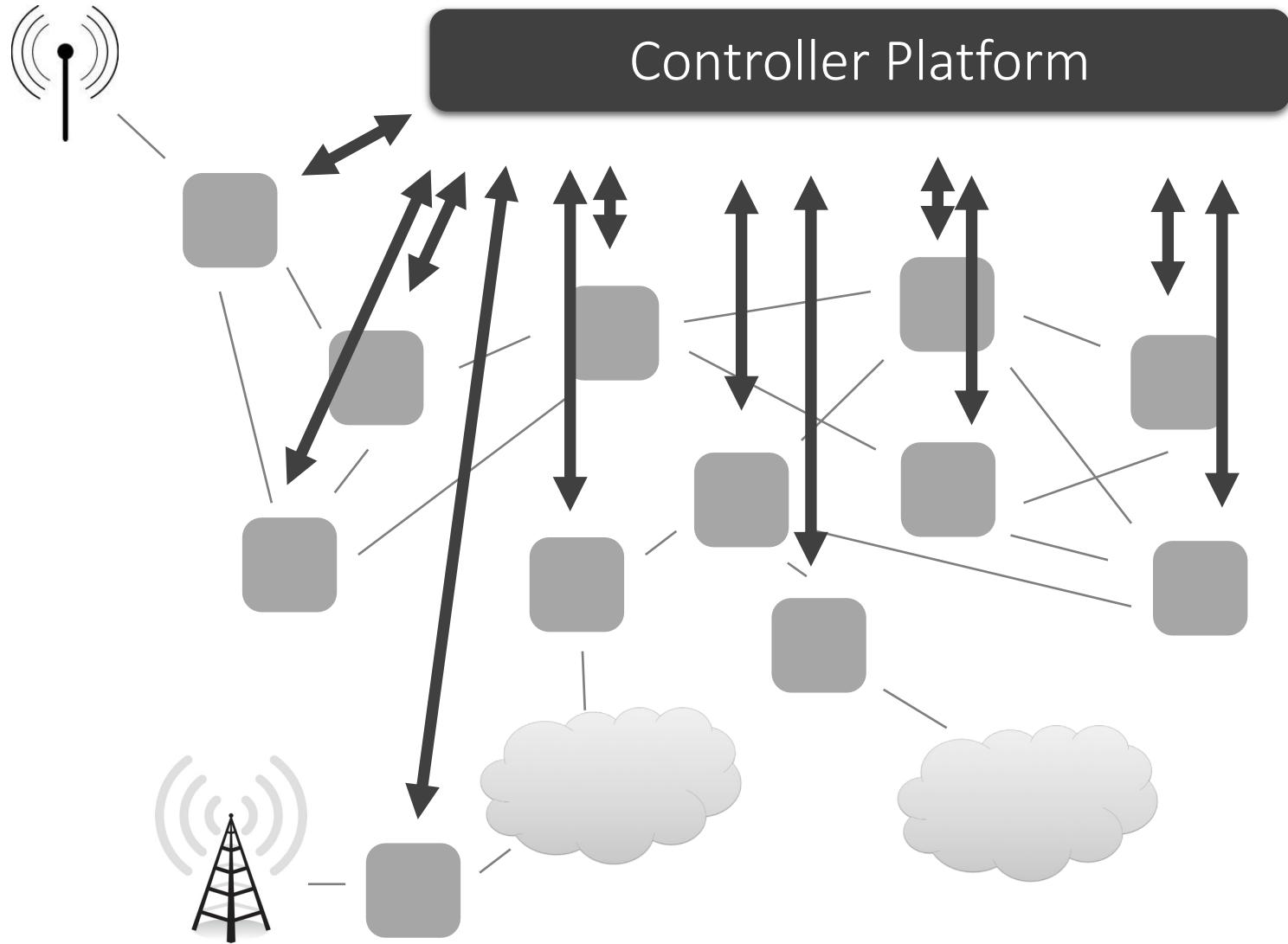
- Software-defined Network



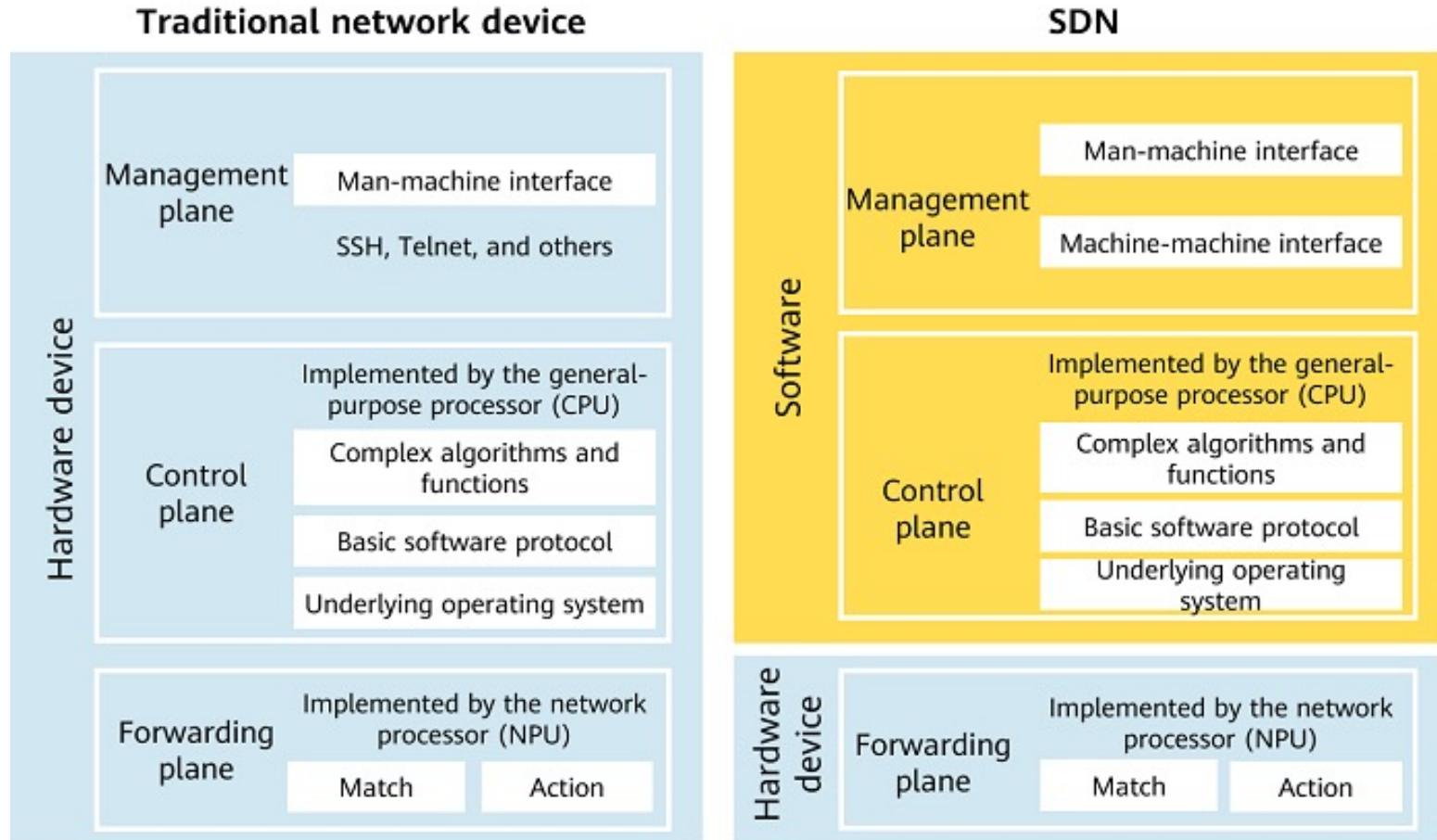
SDN 的概念是

- (1) 将网络设备的**控制面**和**数据面解耦**
- (2) 可以直接对网络设备的**控制面**进行编程
- (3) 从底层硬件设备中**抽象出****网络服务**

统一的控制中心



传统网络设备 vs. SDN

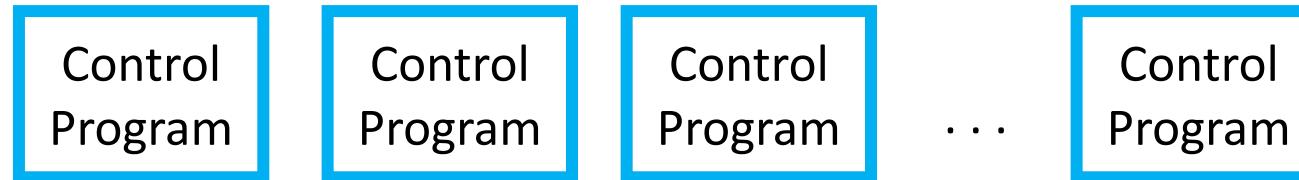


SDN技术路线图强调了控制平面的分离，期望网络设备可以改为白盒设备，以实现用户定义的网络功能。

SDN 架构



Control Plane



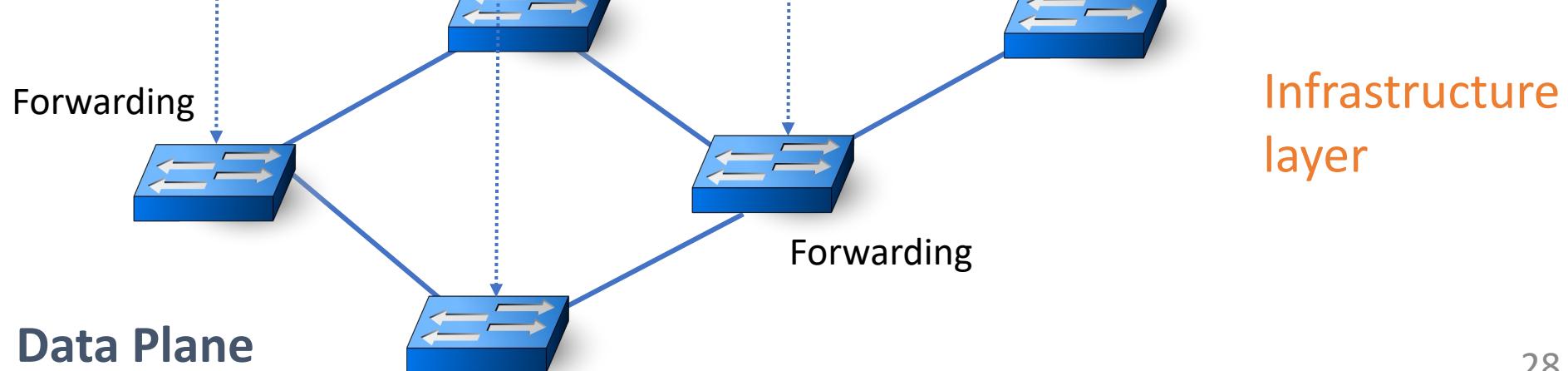
Application layer

...



Control layer

Forwarding



Infrastructure layer

Data Plane

控制程序 (Control Program)



- 控制程序通常存在于SDN架构的应用层，通过定义网络的**高级行为和策略**来决定网络应该如何运行
- 例如，一个控制程序可能定义了一种安全策略，规定了哪些类型的网络流量应该被阻止
- 控制程序使用网络操作系统提供的API来与控制层进行交互，指定网络的期望状态和行为



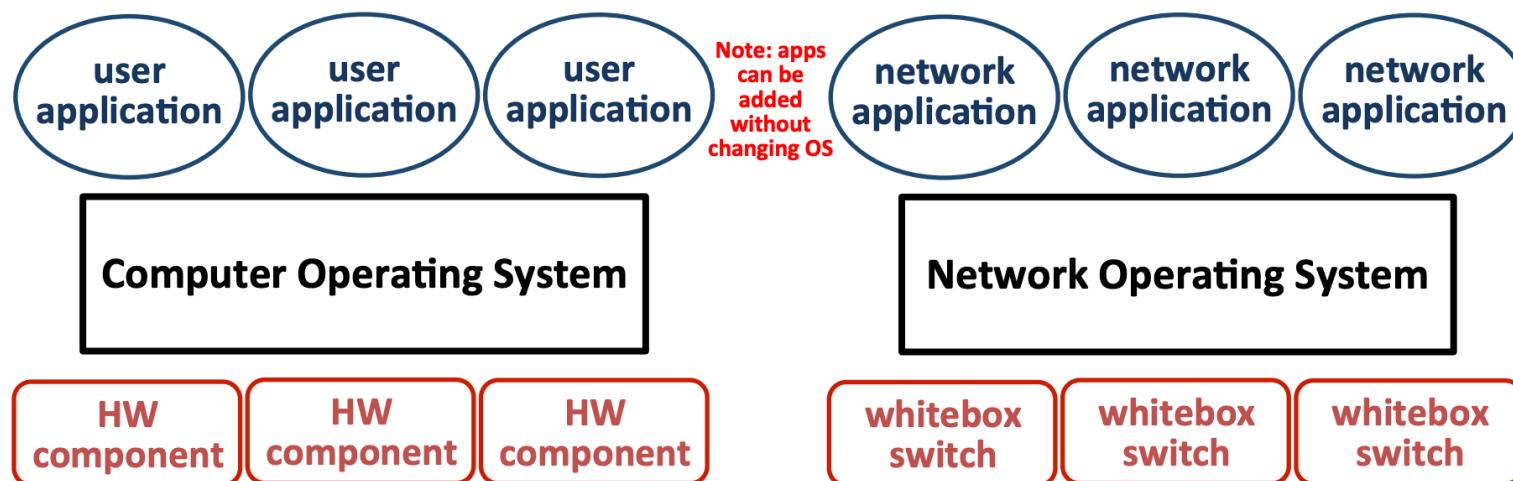
网络操作系统 (Network OS)



- 一般操作系统

- ✓ 位于用户程序和物理计算机硬件之间
- ✓ 揭示高级功能（例如，分配内存块或写入磁盘）
- ✓ 隐藏特定于硬件的细节（例如，内存芯片和磁盘驱动器）

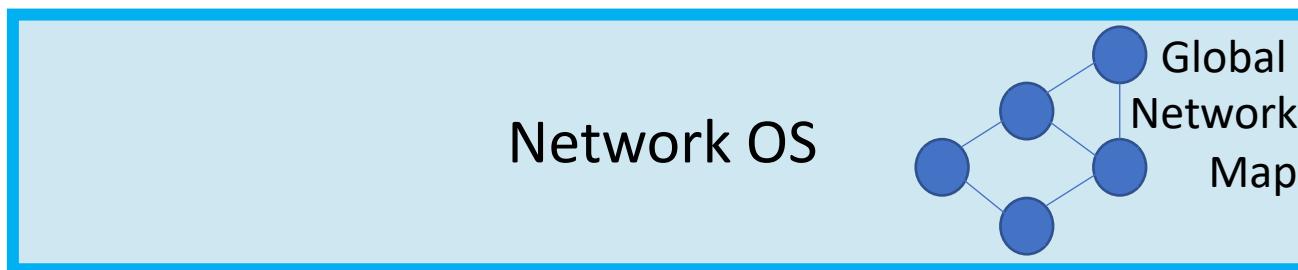
我们可以把 SDN 当成是网络的操作系统



网络操作系统 (Network OS)



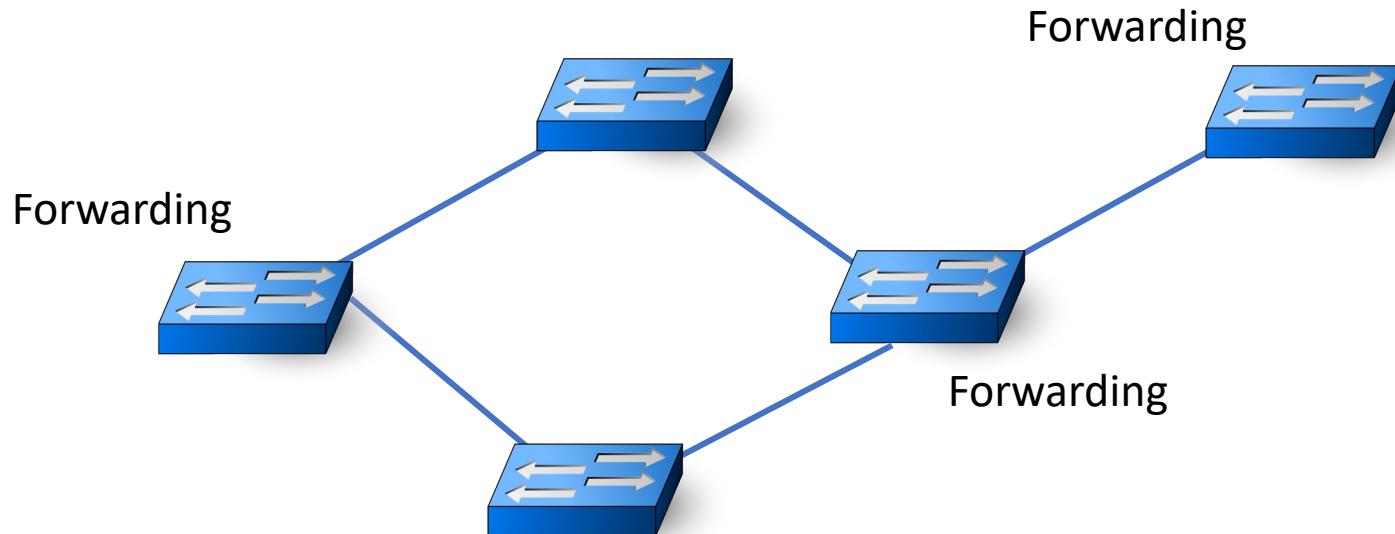
- 网络操作系统位于 SDN 架构的控制层，通常与 SDN 控制器一起工作
- 网络操作系统为控制程序和数据层设备之间提供了一个抽象层，负责将控制程序的高级指令转化为可以被数据层设备理解和执行的具体规则和配置



转发 (Forwarding)



- 转发涉及到 SDN 架构的数据层
- 在SDN架构中，数据层设备（如交换机和路由器）负责处理（如接收、转发和丢弃）实际的网络流量。这些设备根据控制层下发的规则和配置来执行转发决策



SDN 关键能力



- 集中控制平面可以实现更强大的抽象
 - ✓ 例如，X和Y应该能够进行通信
 - ✓ 全网络范围的意图意味着更好的安全性
- 分布式系统技术，使中央控制具有可扩展性和容错性
- 中央控制意味着网络共用一套 API，而不是每个设备用自己独有的 API
- 通过软件控制网络，而非人

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



Software SDN: separates software as much as possible.

Network configuration management tool



Controller

Management plane interface
Value-added service software
Basic software protocol
Underlying operating system



Hardware device

A few software protocols
Match Action

Hardware SDN: emphasizes O&M automation and weakens the separation of the control plane.

Network configuration management tool

Controller

Management plane interface
Value-added service software

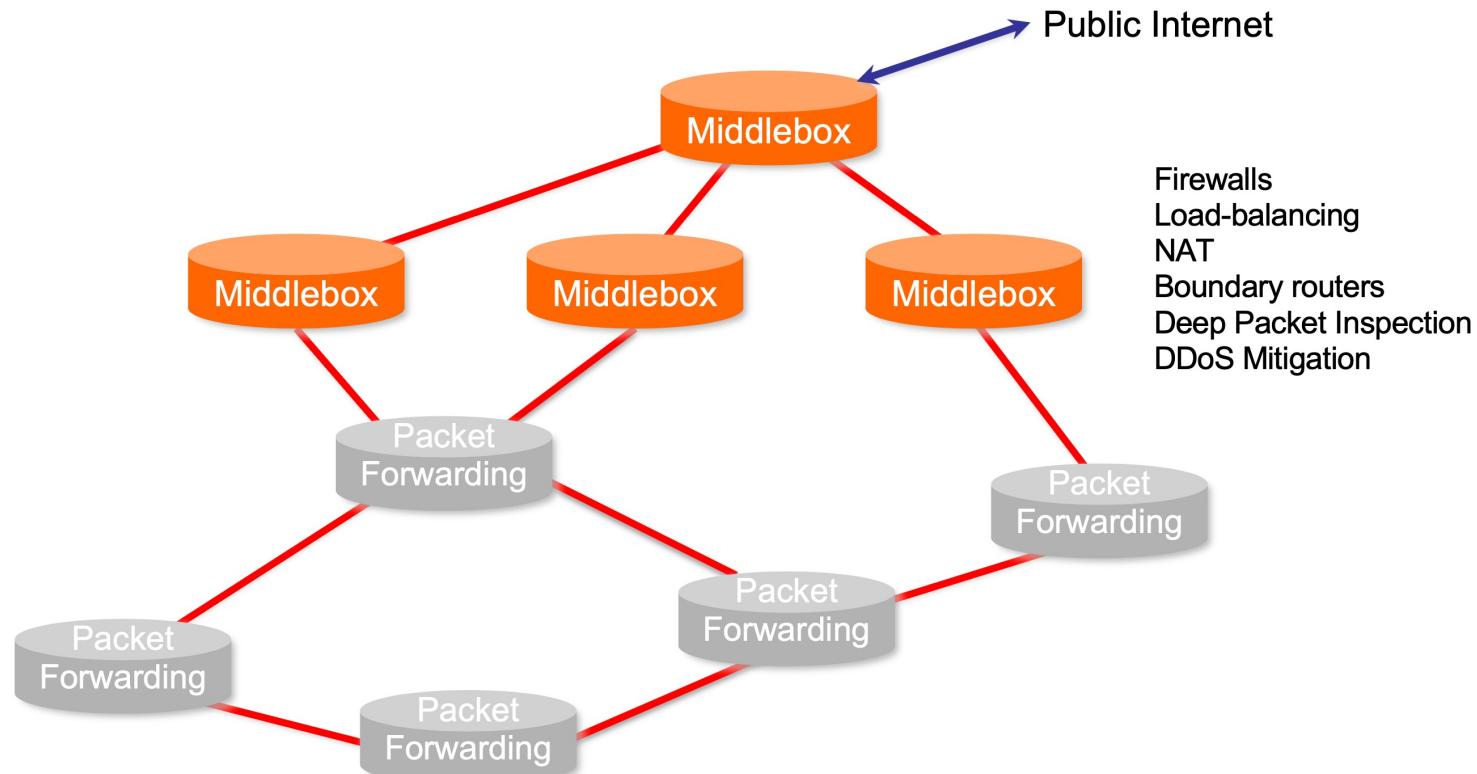


Hardware device

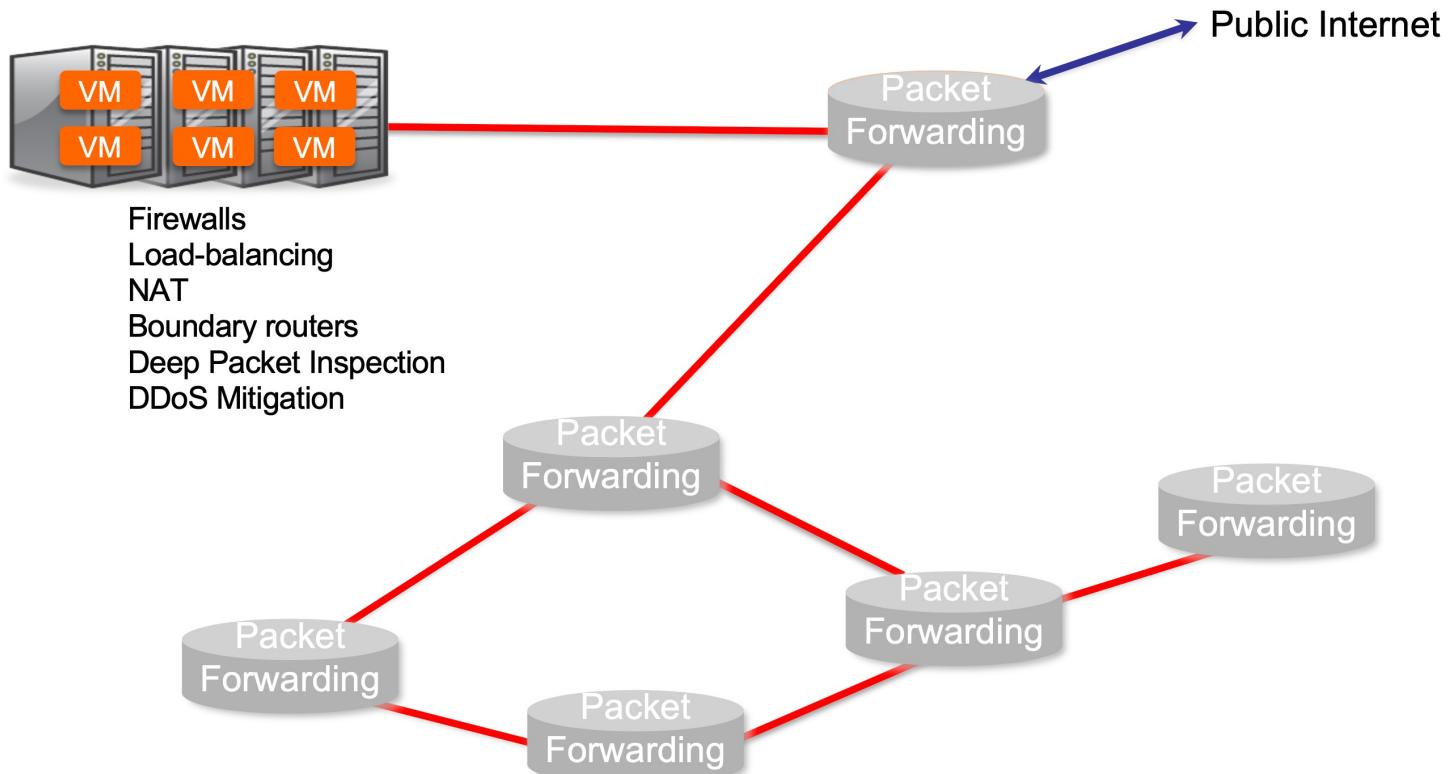
Control plane
Underlying operating system
Various software

Match Action

网络功能虚拟化



网络功能虚拟化





软件定义网络

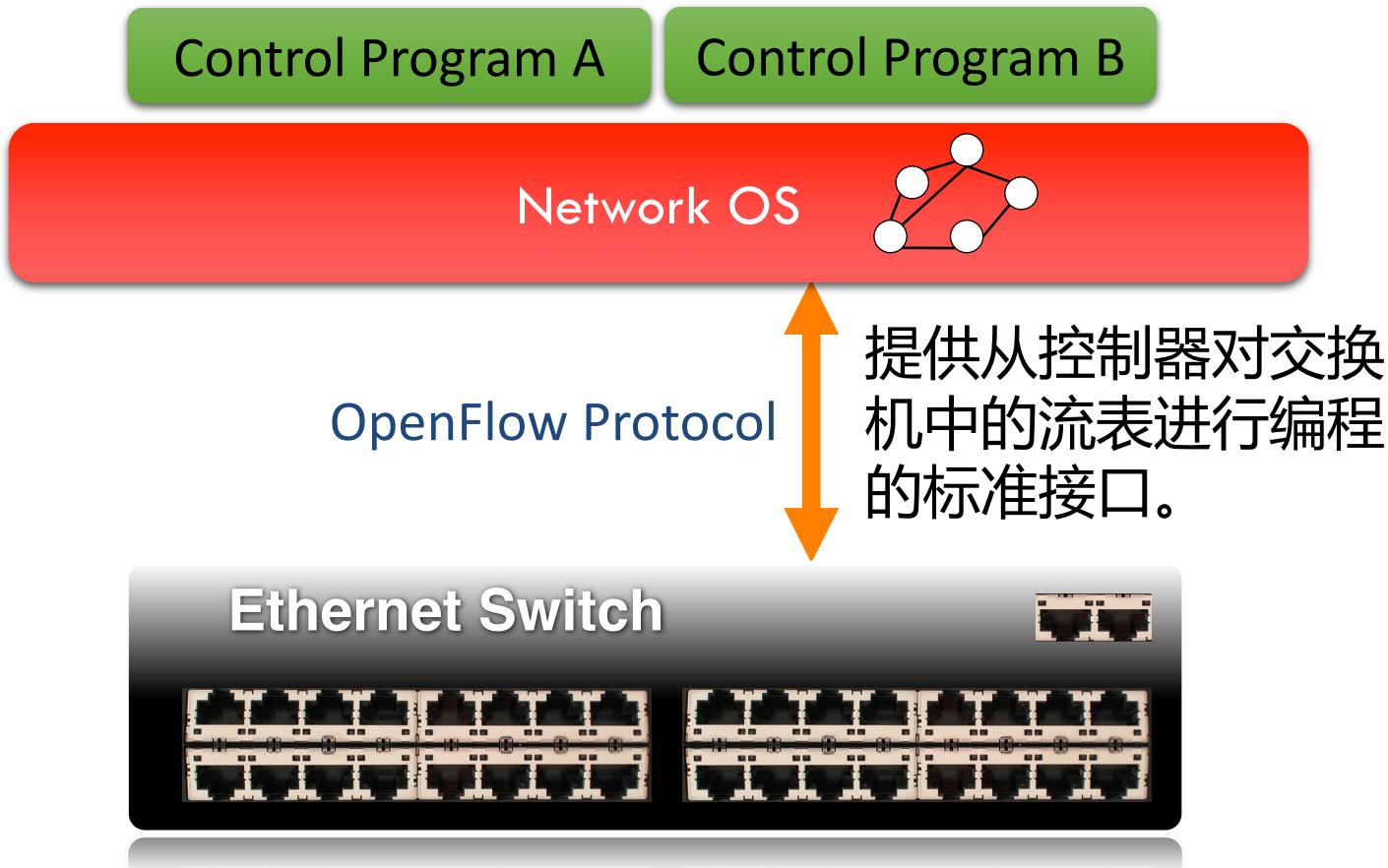
- ❖ 控制面、管理面和数据面
- ❖ 传统网络架构
- ❖ 软件定义网络
- ❖ OpenFlow 协议

OpenFlow 基础



- 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
- SDN 的核心思想是将转发平面与控制平面分离。为了实现这一点，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 交换机



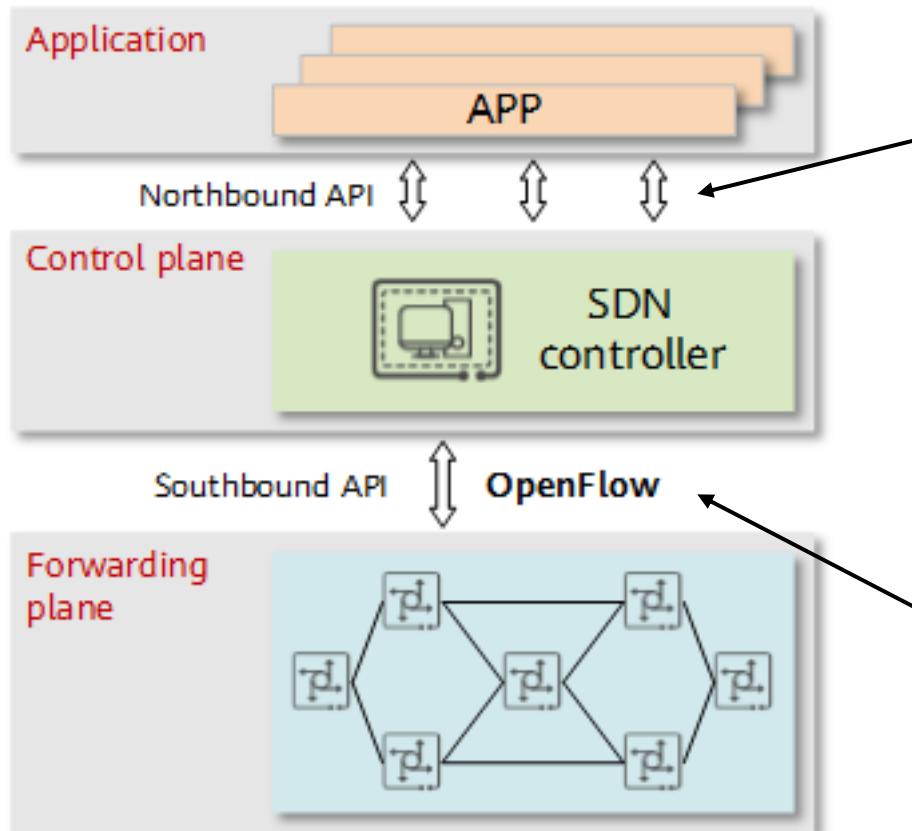
OpenFlow 的起源和发展



- 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，这些策略能够应用到各种网络设备上。
- 受到这个项目的启发，Clean Slate Program 的主管，Nick McKeown 教授，和他的团队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。
- 这种分离允许集中控制器通过标准接口管理和配置各种网络设备，可能会带来网络设计、管理和利用方面的更大创新和灵活性。
- 2008年，McKeown 教授和他的团队在一篇名为 "OpenFlow: Enabling Innovation in Campus Networks" 的论文中首次详细公开介绍了 OpenFlow 的概念。这篇论文阐述了 OpenFlow 的原理，并描述了其可能的应用，标志着 OpenFlow 概念的第一次详细公开描述。

OpenFlow 和 SDN

- 在OpenFlow的基础上，该团队于2009年进一步提出了SDN的概念，引起了业界的广泛关注

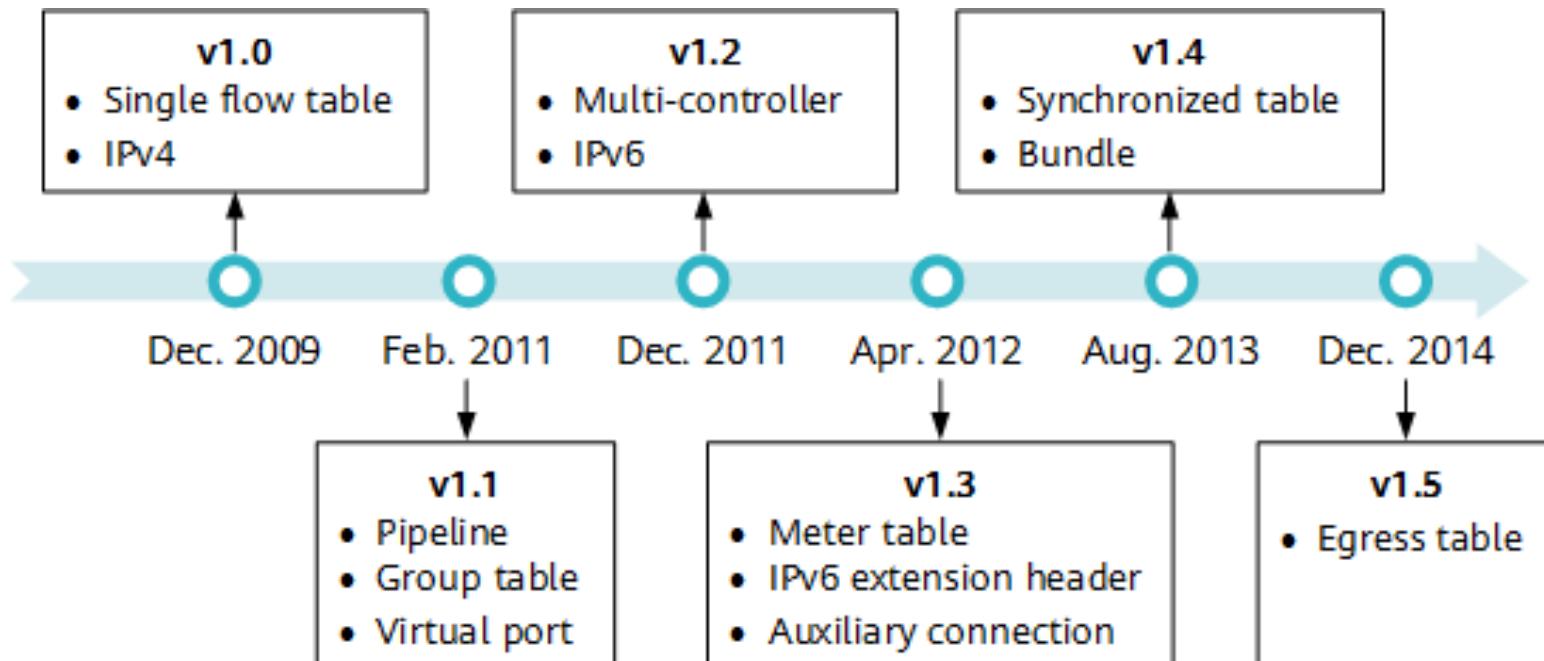


北向 API：控制器与其上运行的应用程序之间的接口。北向API允许应用程序发送请求以改变网络的行为。

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。



OpenFlow版本的演变和主要变化



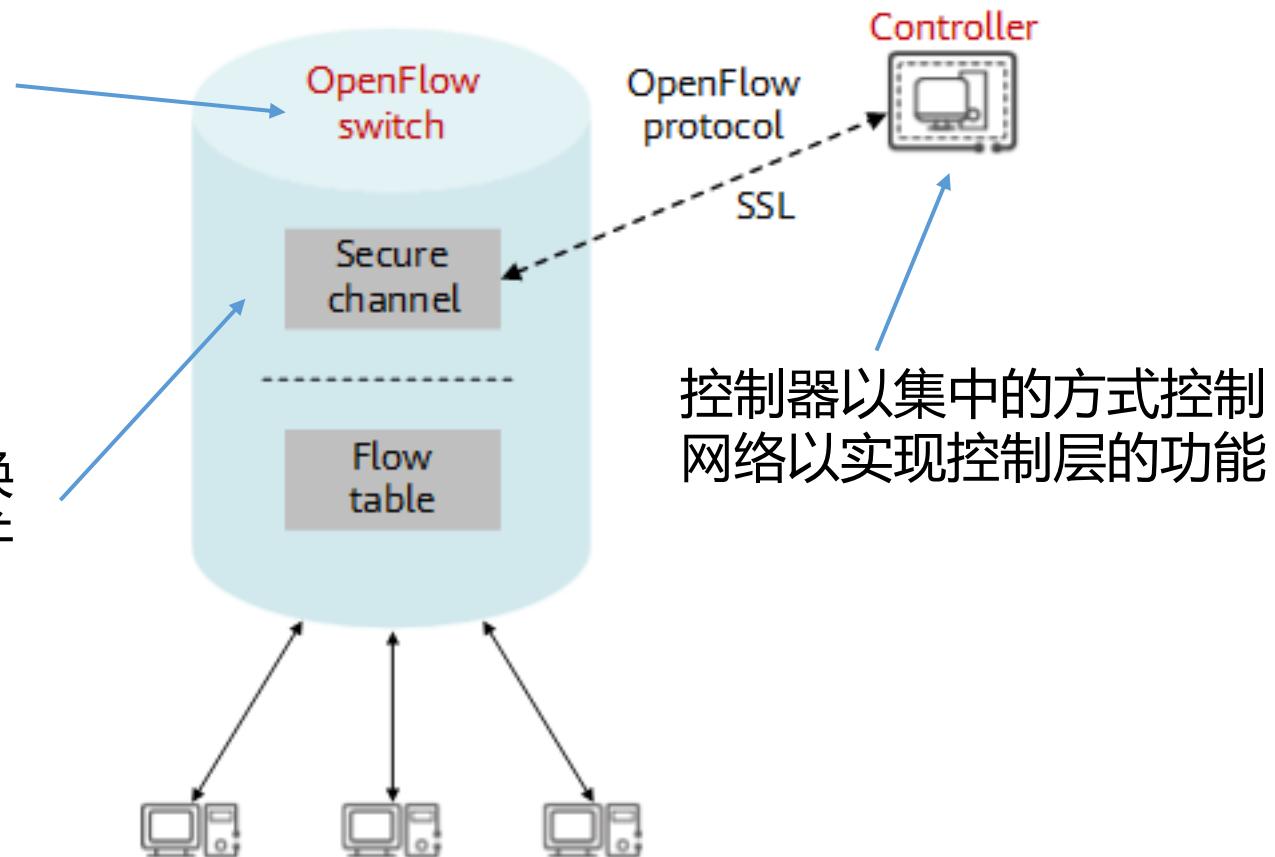
OpenFlow 架构



- 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



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（一种南向 API）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开发的商业控制器。广泛使用的开源控制器包括 NOX、POX 和 OpenDaylight

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，以管理或获取 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 交换机是 OpenFlow 网络的核心组件，主要负责**数据层的转发**。它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

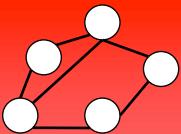
OpenFlow 流表项



Control Program A

Control Program B

Network OS



Packet
Forwarding

Packet
Forwarding

Flow
Table(s)

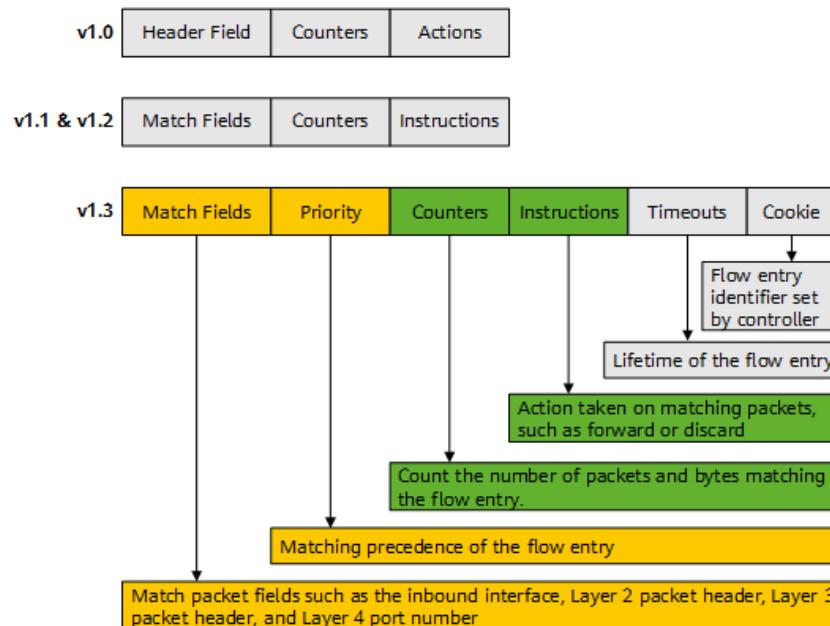
Packet
Forwarding

“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 流表项



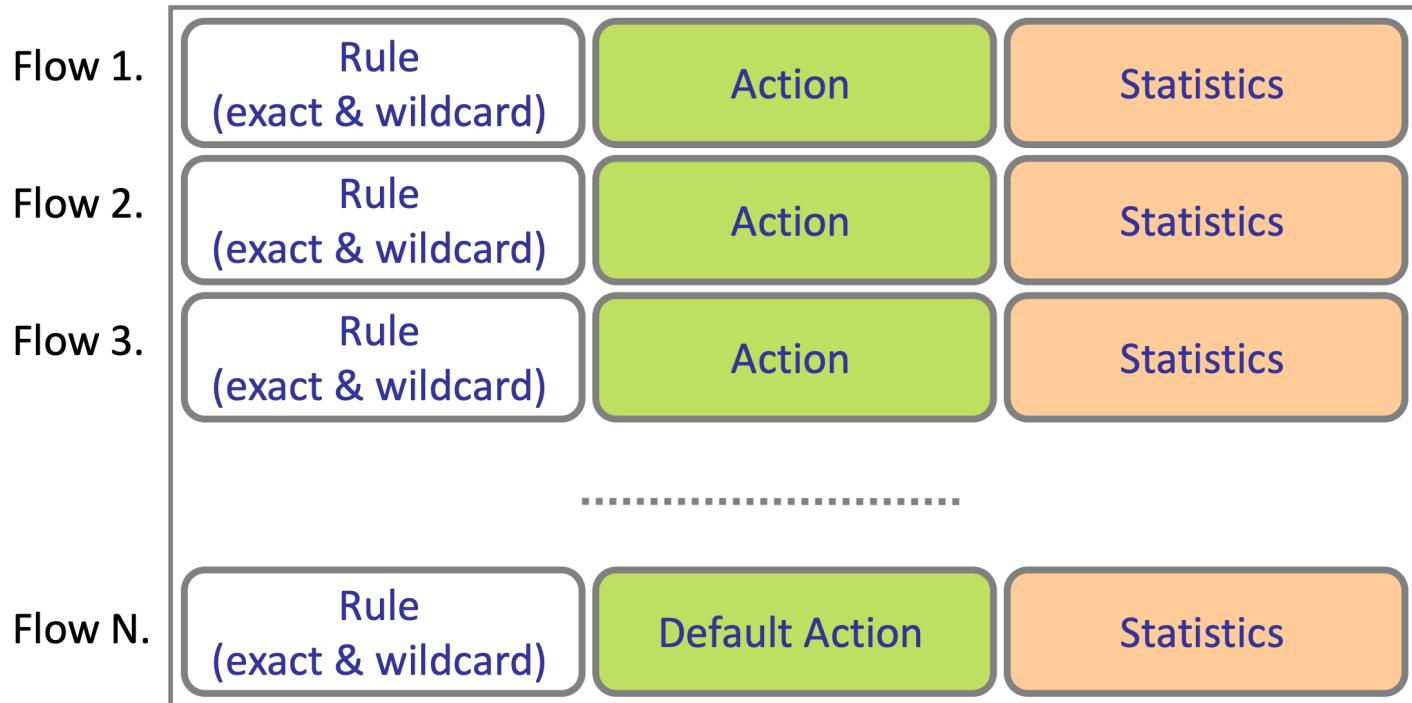
- 交换机和路由器等传统网络设备基于本地保存的第 2 层 MAC 地址转发表、第 3 层 IP 地址路由表和传输层端口号来转发数据
- OpenFlow 基于包含网络上**所有层的网络配置信息**的流表而不是 5 元组信息来切换数据。流表中的条目是某些关键字和操作的灵活组合



OpenFlow 流表



规则 (Rule) 动作 (Action) 统计 (Stats)

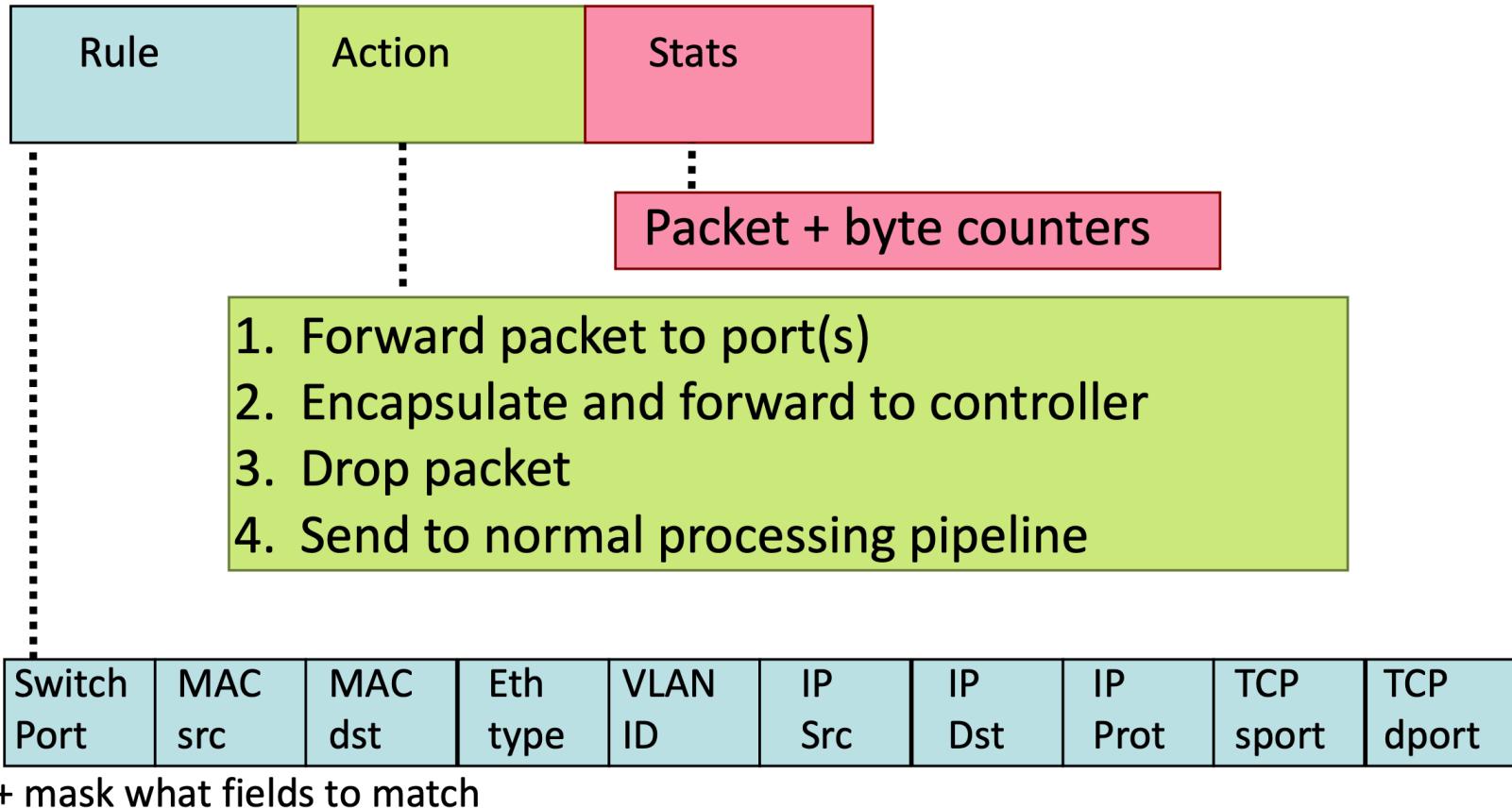




OpenFlow 流表

- **规则 (Rule)** : 也称流表项 (Flow Entry) , 是一组定义了
特定网络流量模式的字段
 - ✓ 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 , 它会匹配的规则并进行相应的操作
- **动作 (Action)** : 定义了一旦找到匹配的规则 , 交换机应该**如何处理数
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更深入的处理
- **统计 (Stats)** : **流量的统计数据** , 被交换机收集并可以报告给控制器
 - ✓ 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)



例子 (1)



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

例子 (2)



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

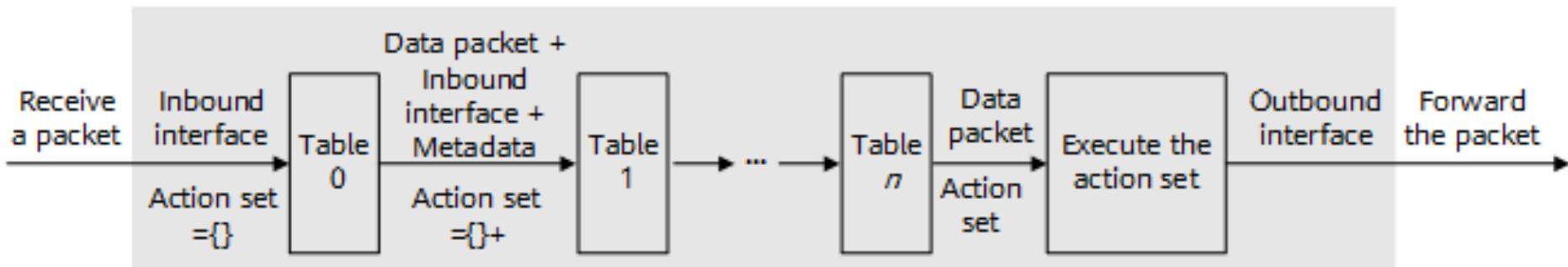
VLAN

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

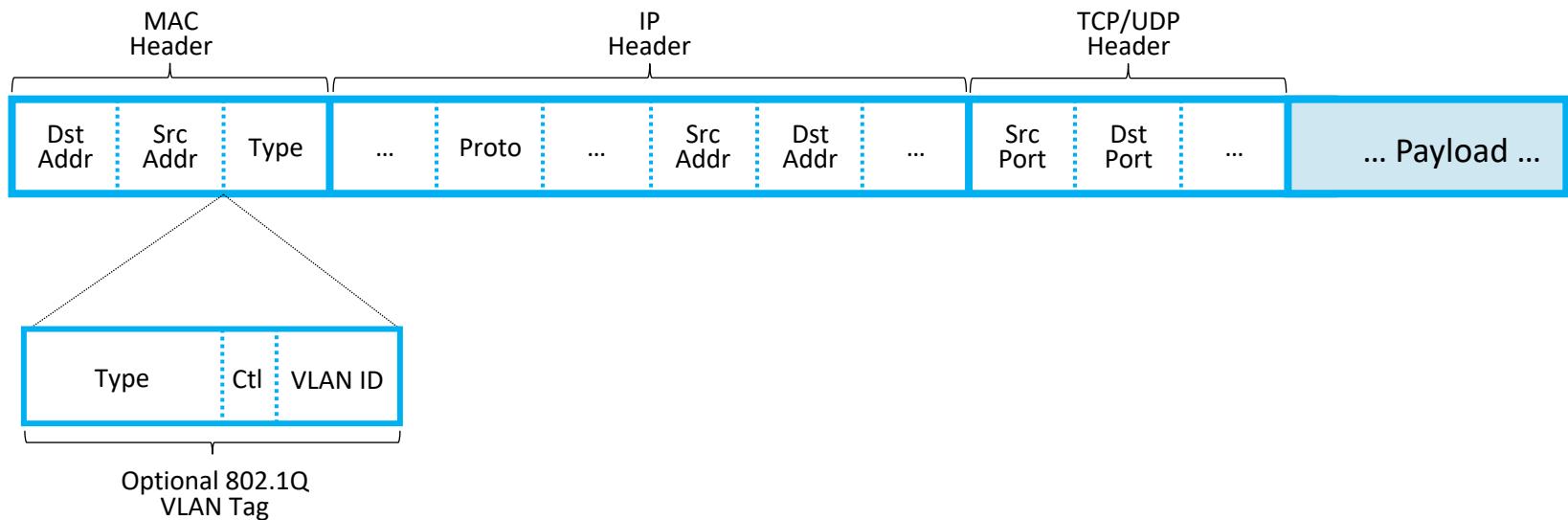
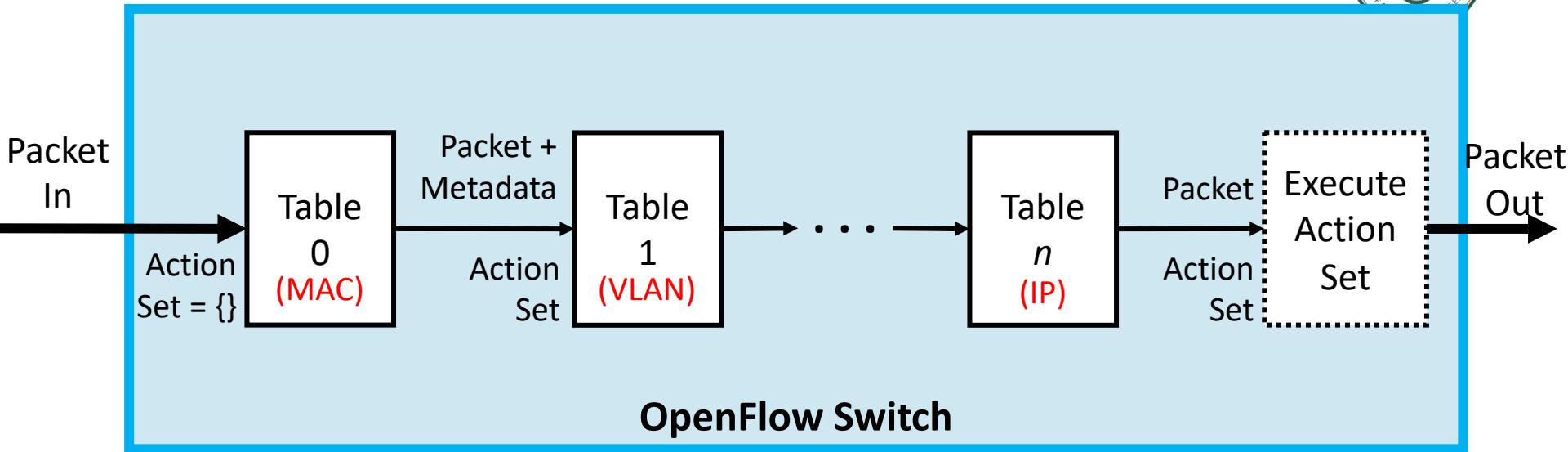
多级流表和流水线处理



- 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



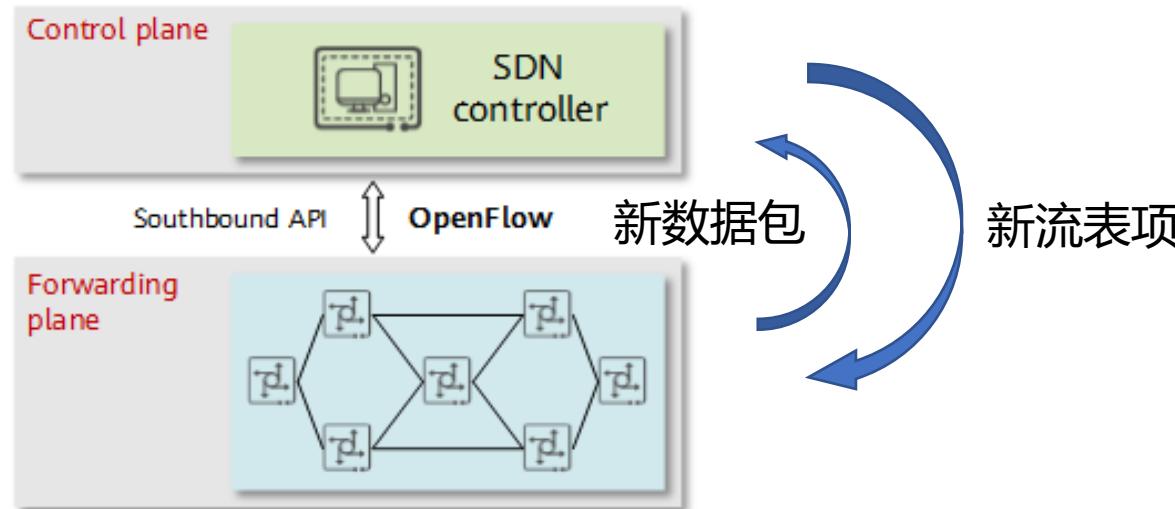
多级流表和流水线处理



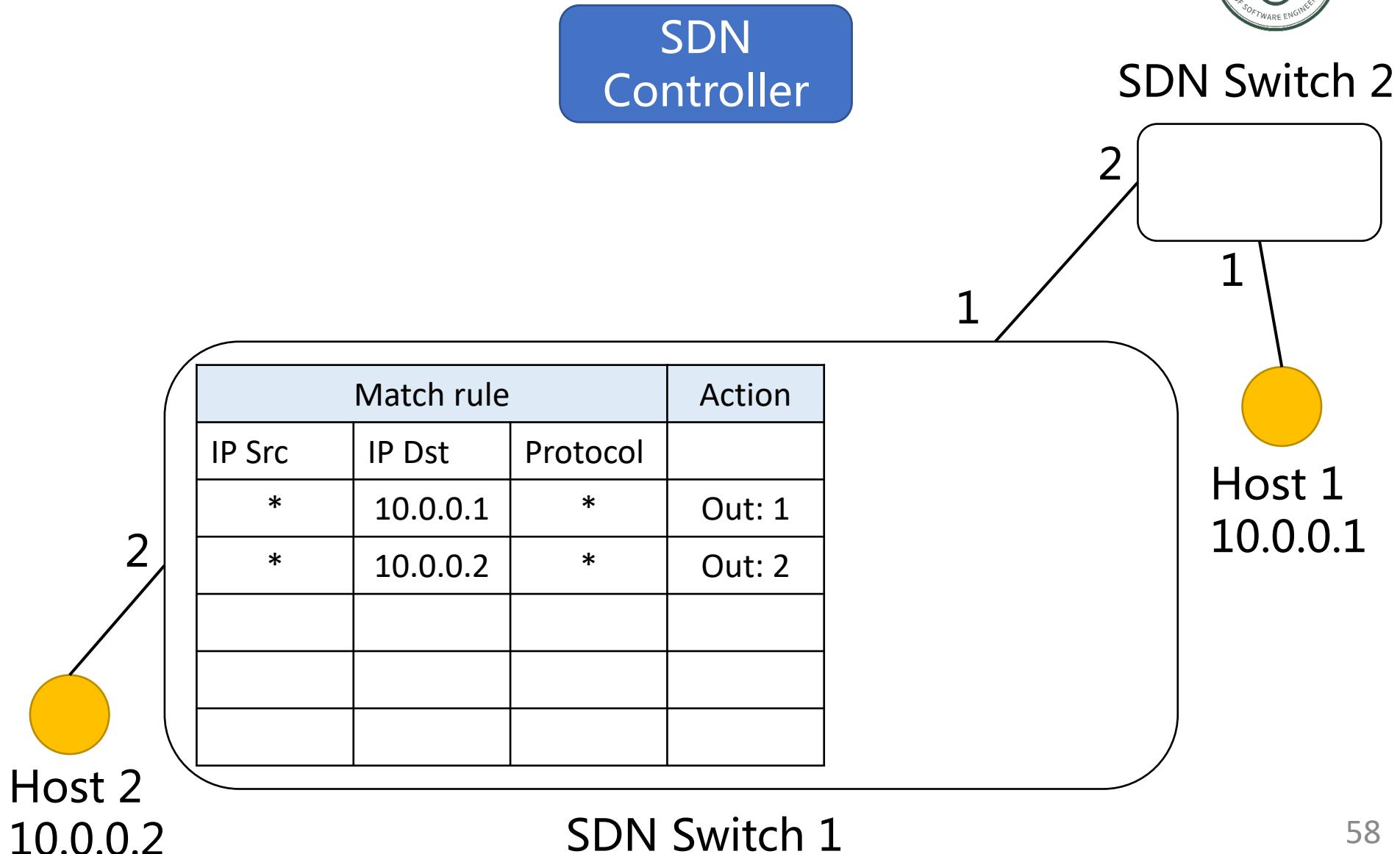
流表传递 (Flow Table Delivery)



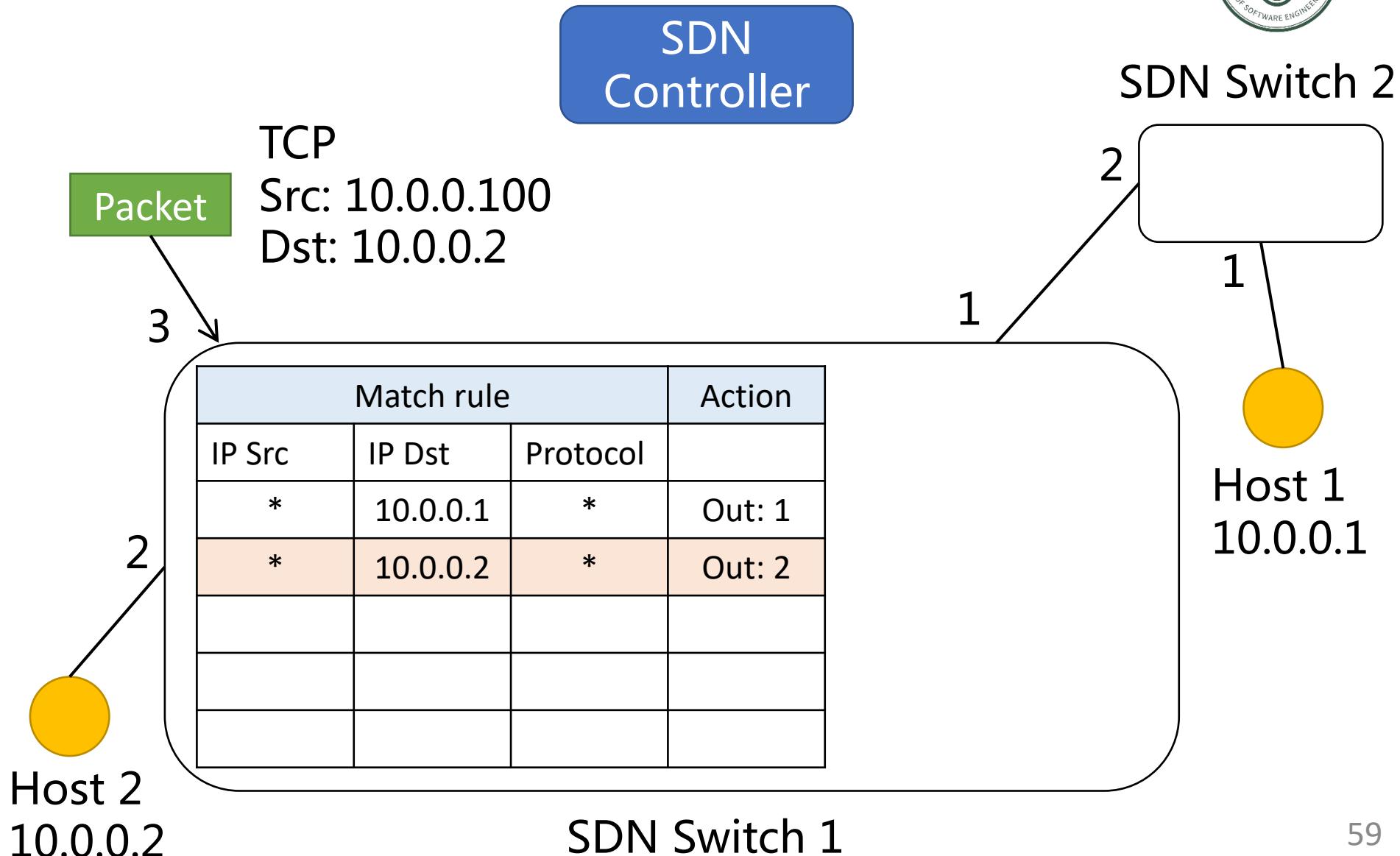
- 控制器创建一个或多个流表项，然后发送这些条目到交换机的流表中，分为主动和被动模式
 - ✓ 在主动模式中，控制器主动向交换机发送流表信息
 - ✓ 在被动模式中，当交换机没有找到任何与接收到的数据包匹配的流表项时，它向控制器发送消息，控制器计算相应的表项并将其传送到交换机



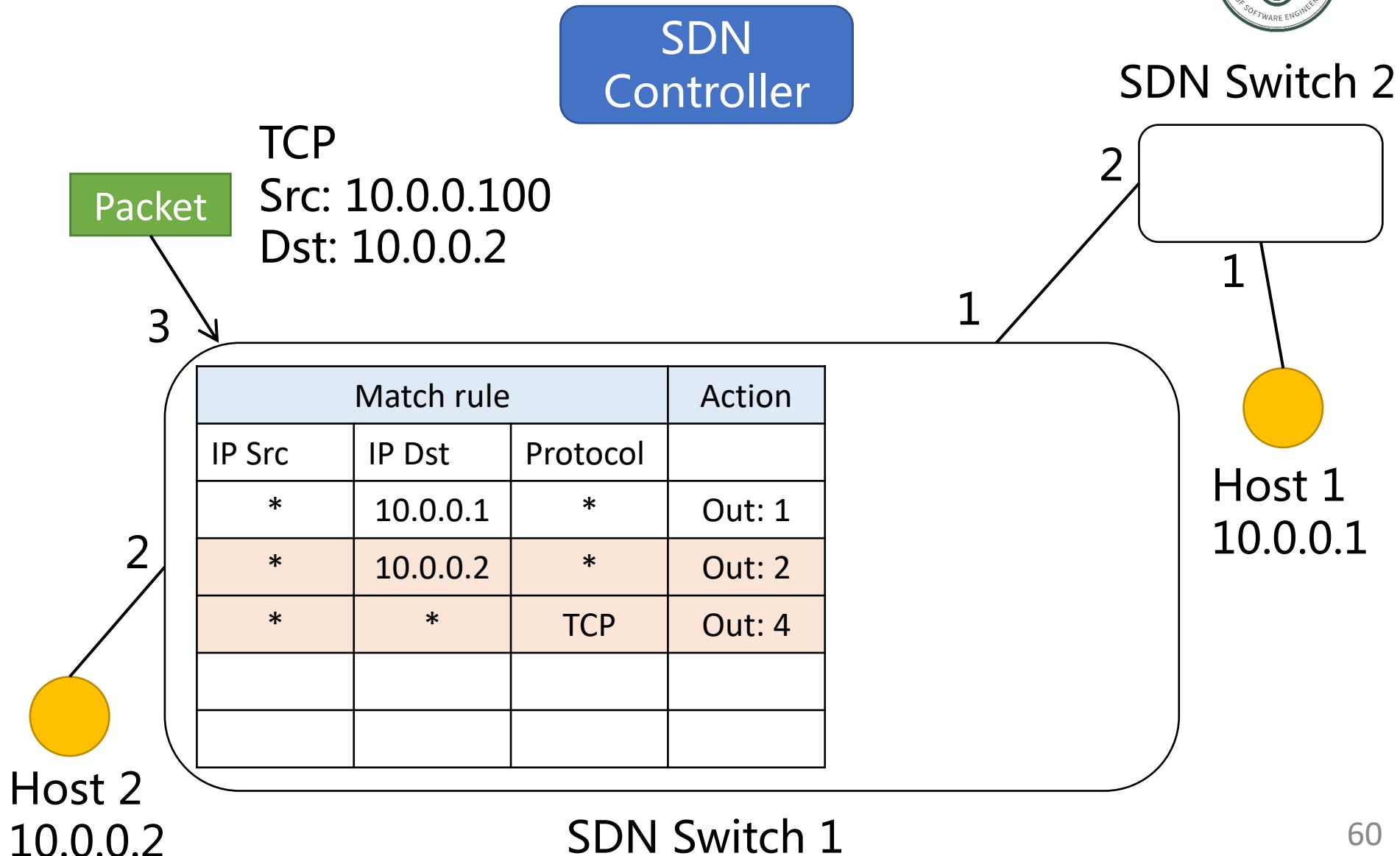
OpenFlow 例子



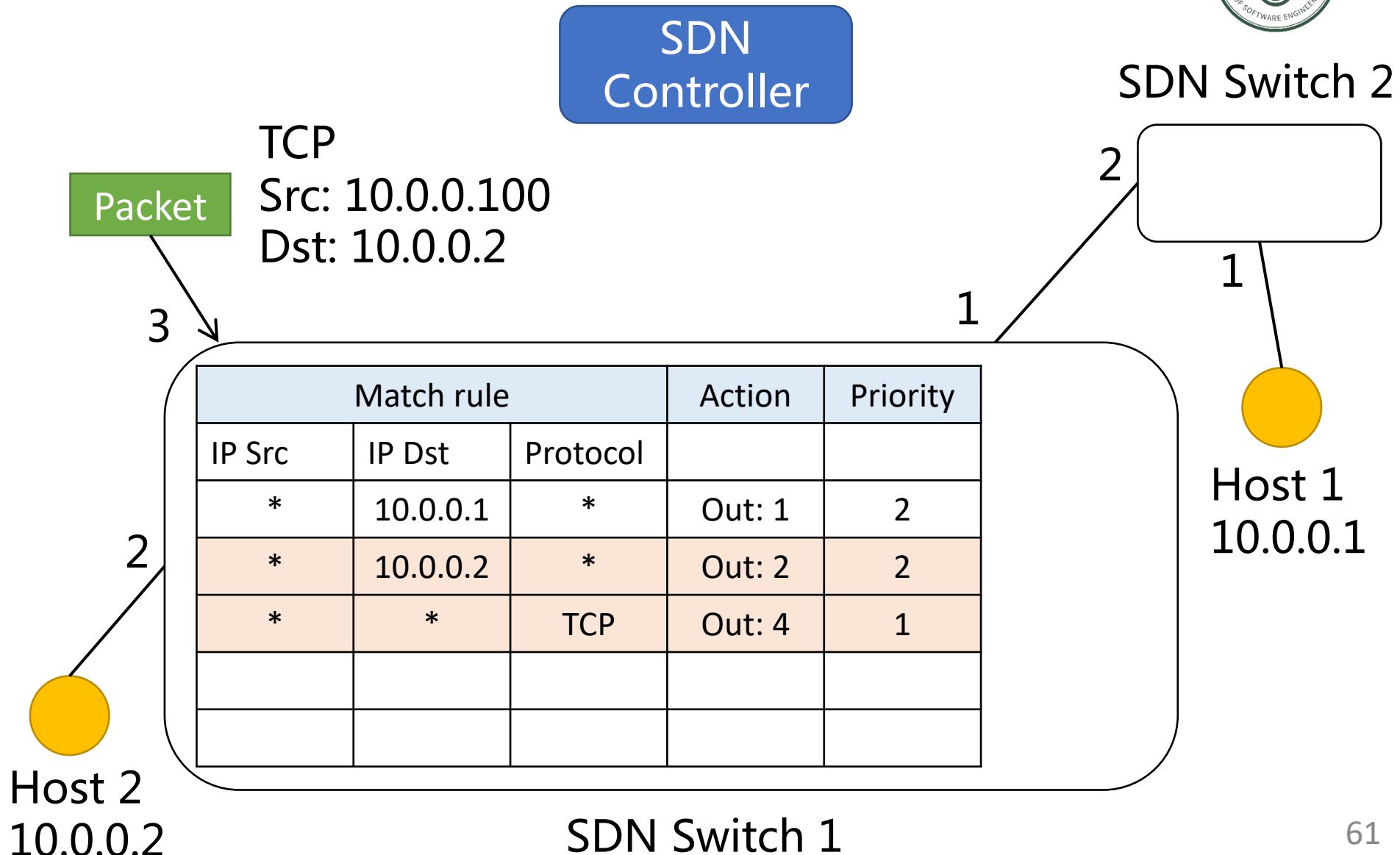
OpenFlow 例子



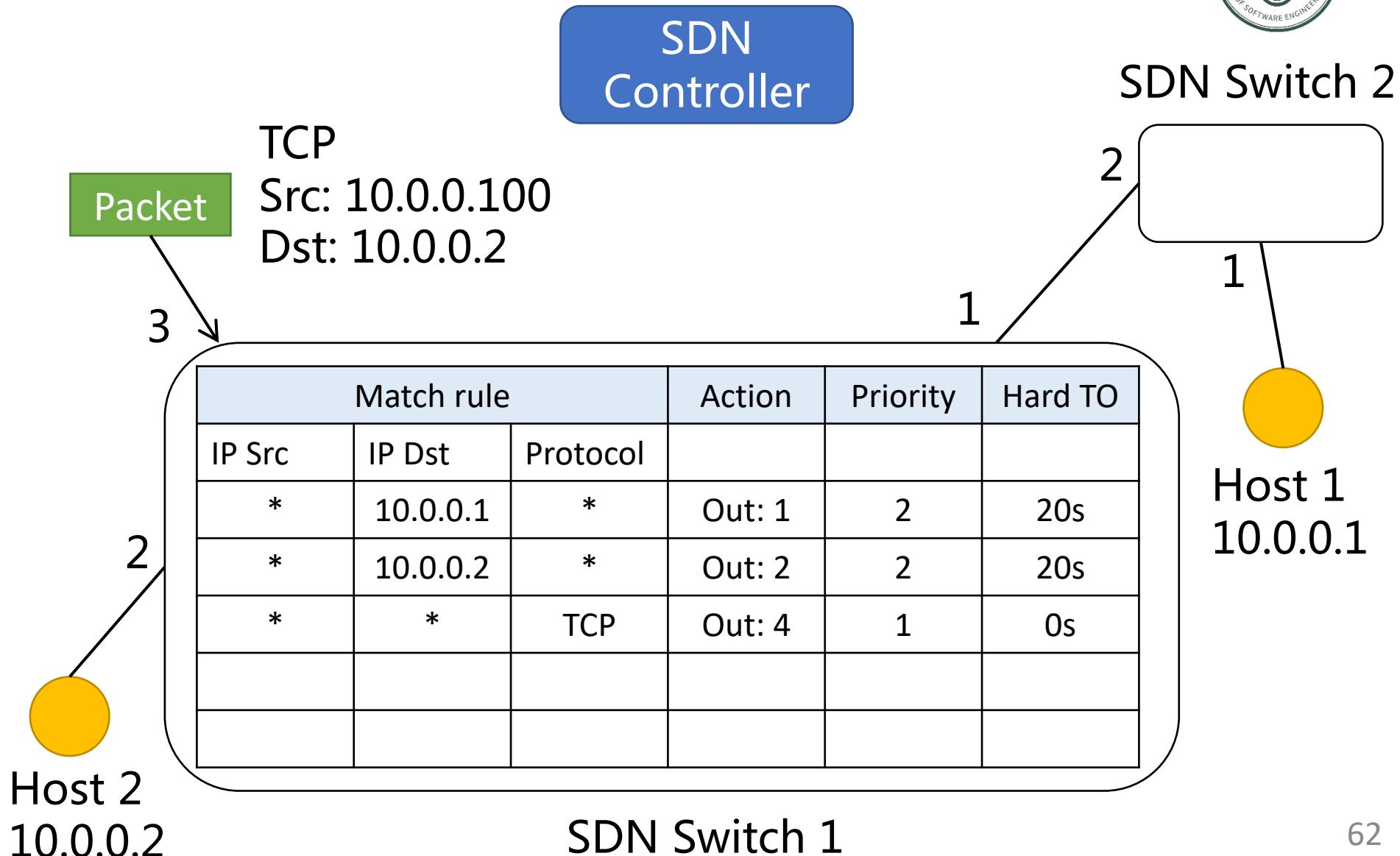
OpenFlow 例子



OpenFlow 例子

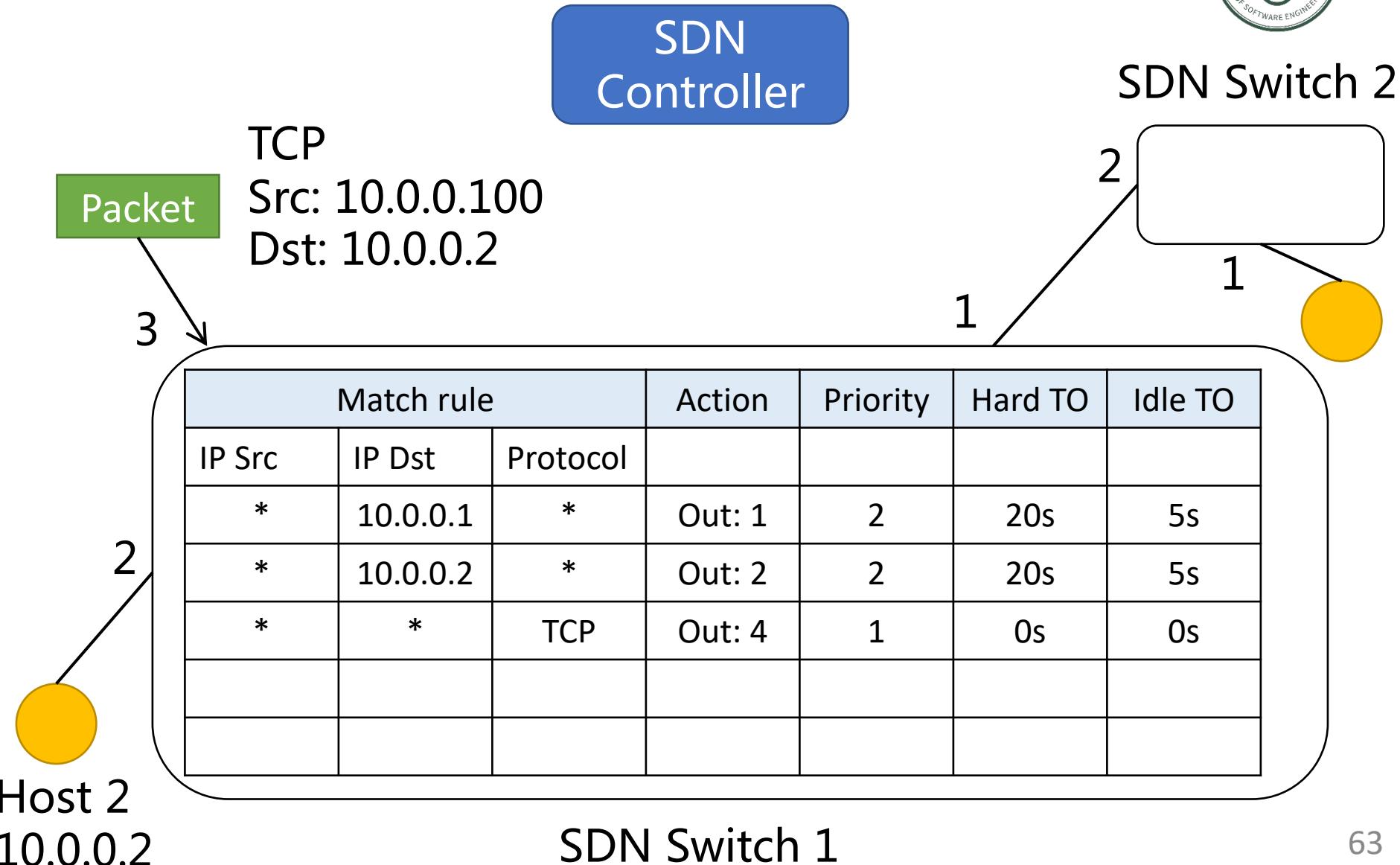


OpenFlow 例子

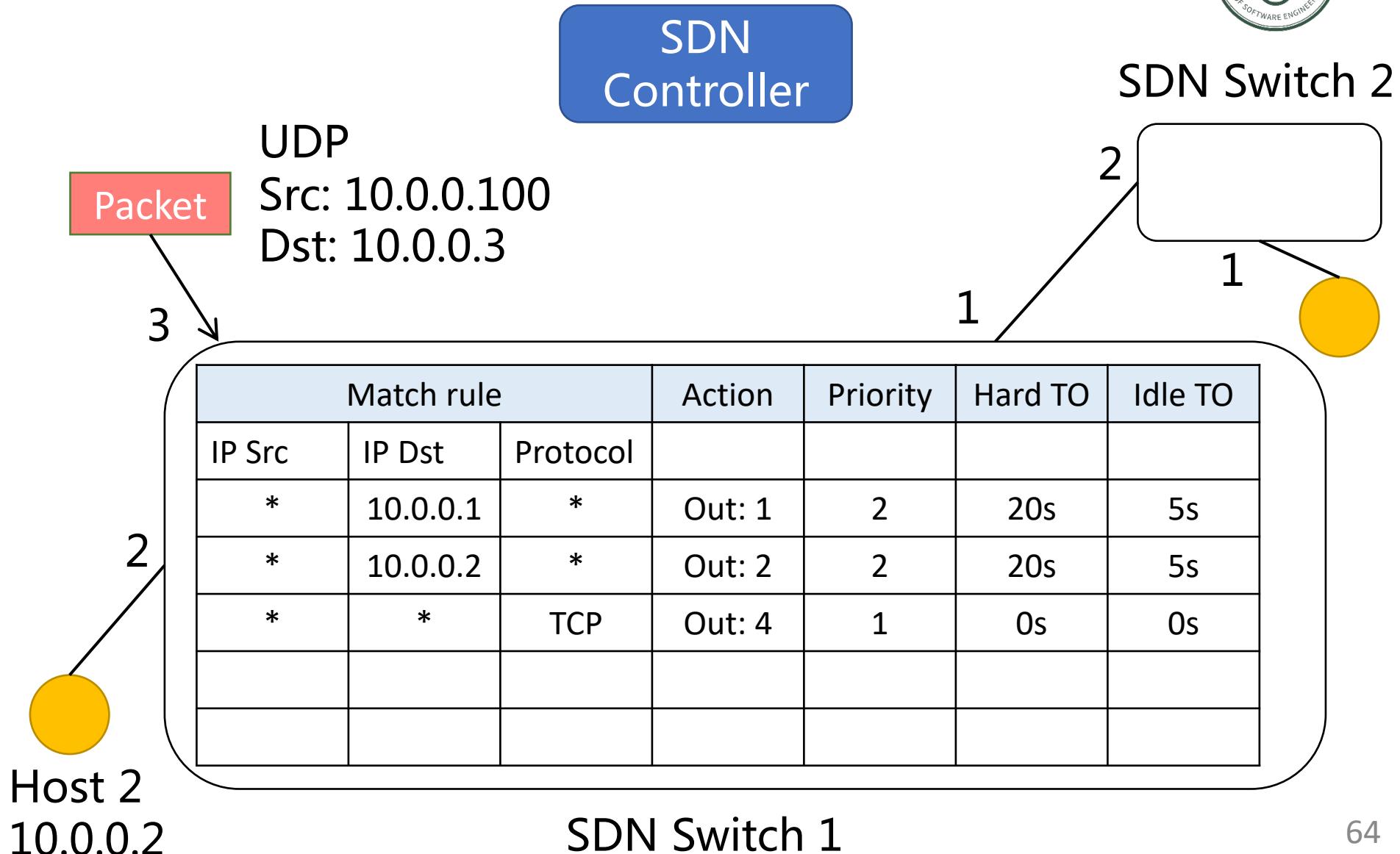




OpenFlow 例子

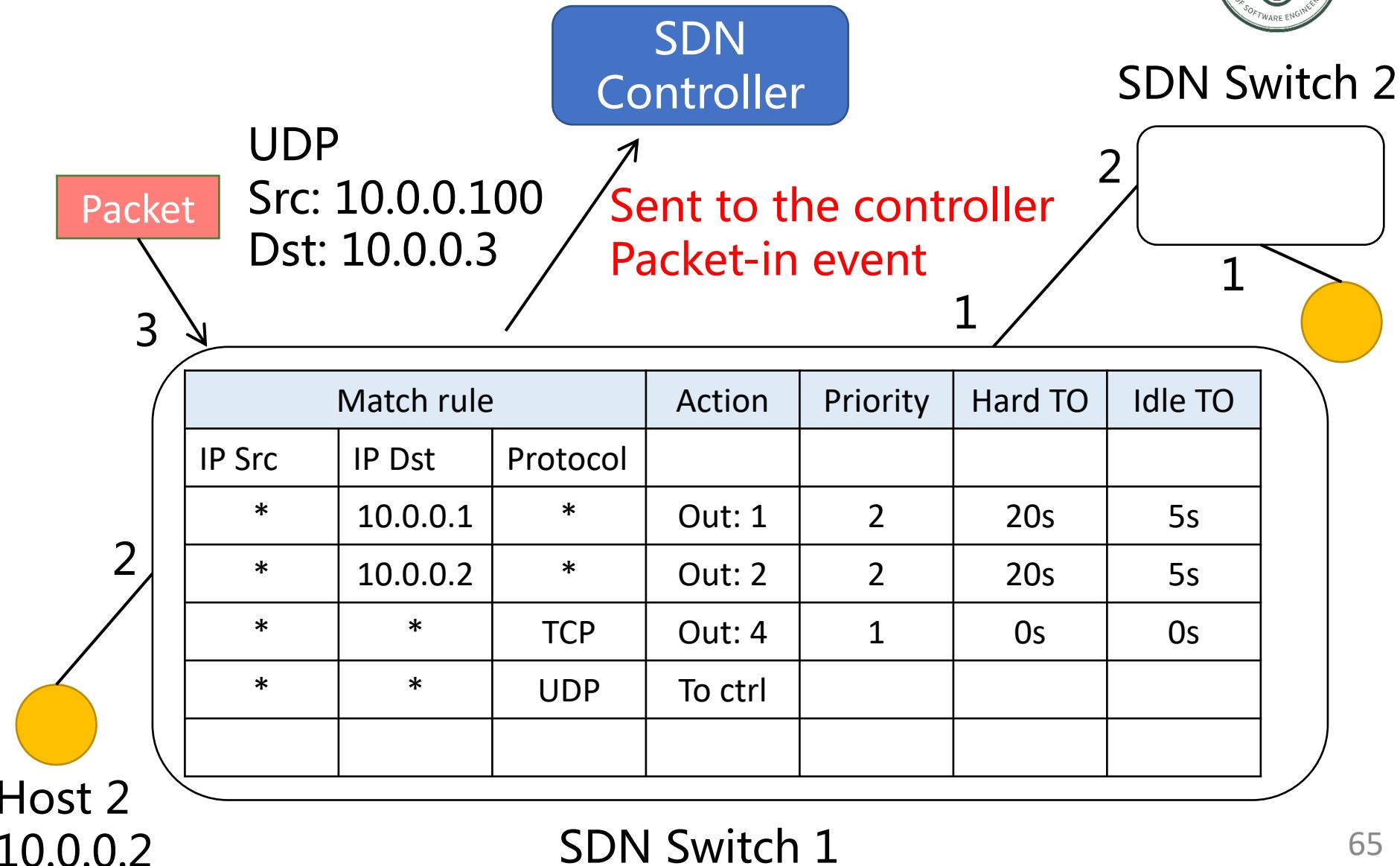


OpenFlow 例子

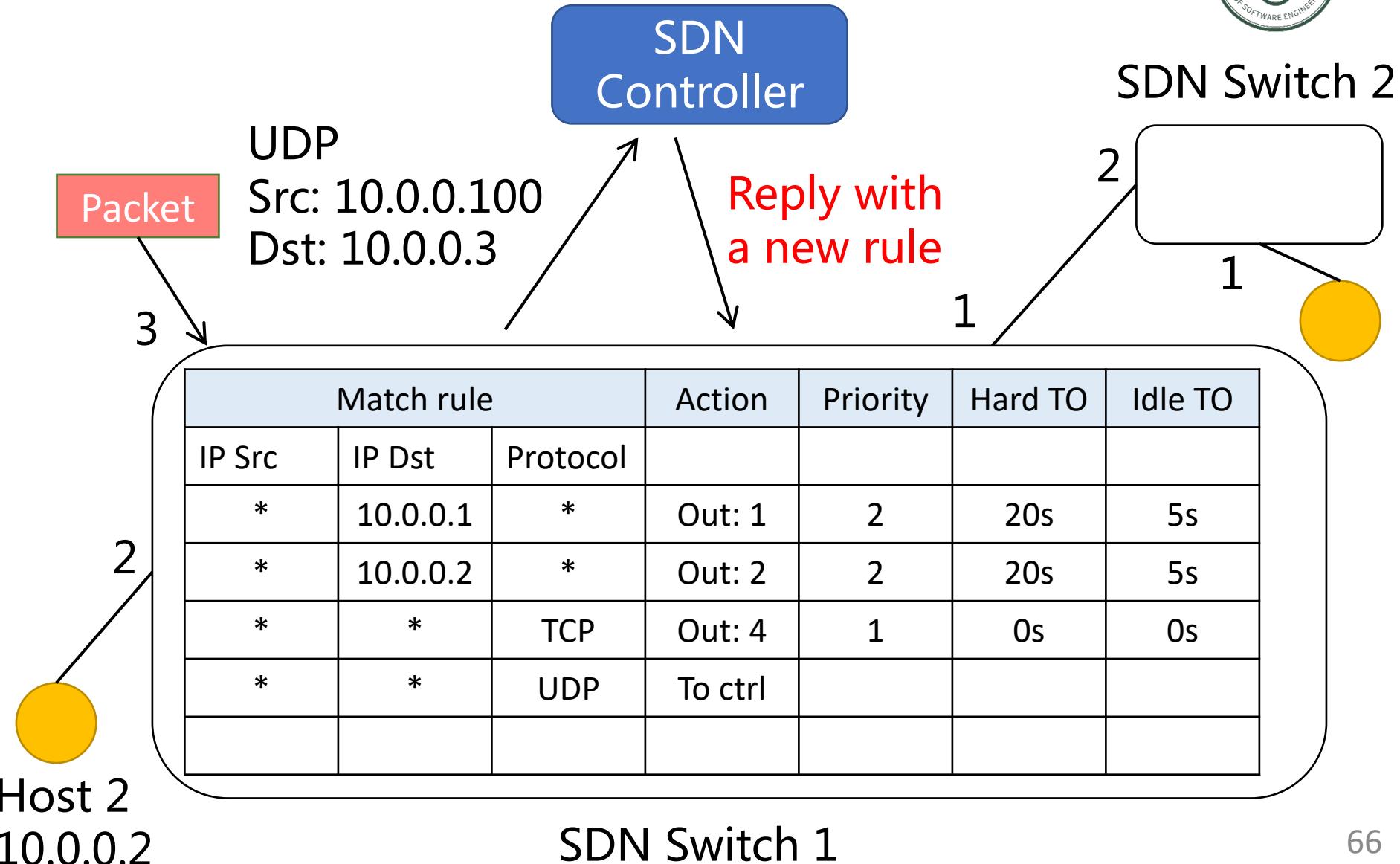




OpenFlow 例子



OpenFlow 例子





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

<https://zbchern.github.io/sse316.html>