OpenFlow协议分析

2015.12.23

江苏省未来网络创新研究院

提 纲

1 OpenFlow协议概述

2 OpenFlow 1.0协议介绍

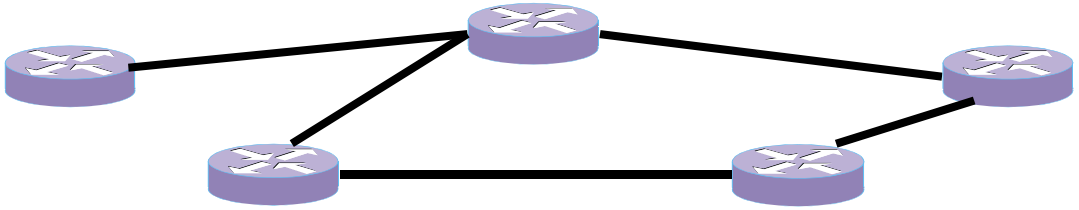
3 OpenFlow协议演进

OpenFlow概述

- **最早由Stanford大学的Nick McKeown教授等研究人员在2008年4月发表的论文*OpenFlow: Enabling Innovation in Campus Networks*中提出**
  - 提出控制和转发分离的架构，将控制逻辑从网络设备盒子中引出来，供研究者对其进行任意的编程从而实现新型的网络协议、拓扑架构而无需改动网络设备本身
- **当前，*OpenFlow Switch Specification*规范由ONF（Open Networking Foundation）主导，获得业界众多支持**
  - ONF成立于2011年，倡导“用户驱动”的网络架构和技术，致力于推动SDN的标准化，其愿景是使得基于OpenFlow的SDN成为网络新标准

基于OpenFlow协议的软件定义网络

控制器：控制器知道所有网络信息，负责指挥设备如何工作



交换机：交换机不知道任何网络信息，只会按照控制器的指挥工作

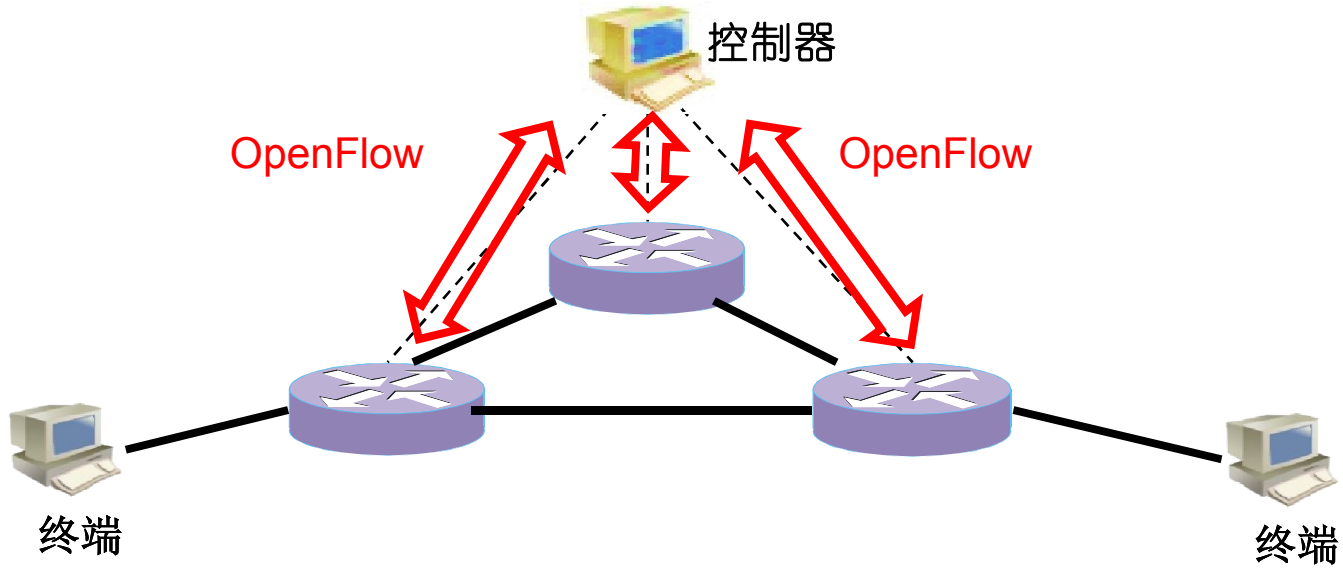
软件定义网络的网络设备之间不运行任何协议，网络设备的转发表由控制器配置生成。控制器与网络设备之间通过OpenFlow协议来互相通信

软件定义网络的平面划分

SDN网络分为两张网

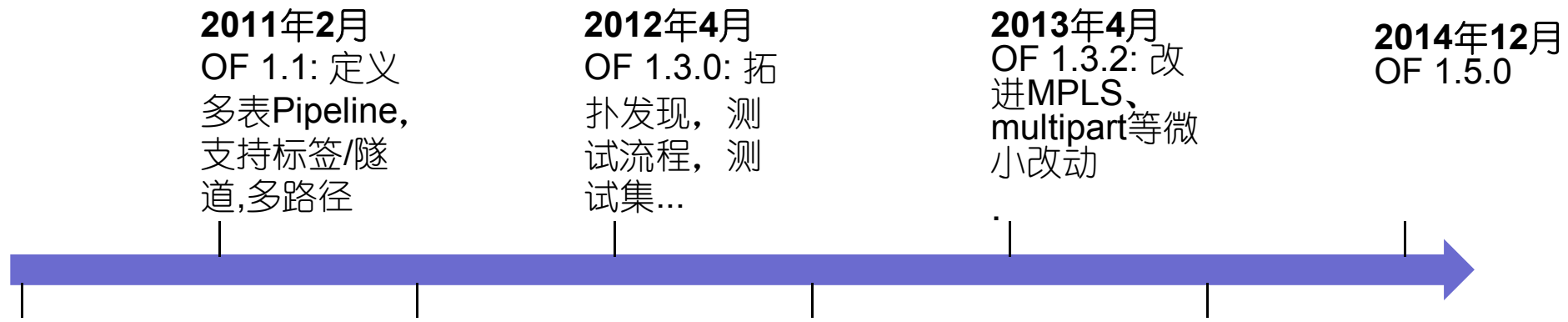
1、数据网(数据平面)：Traffic

2、信令网(控制平面)：OpenFlow Protocol

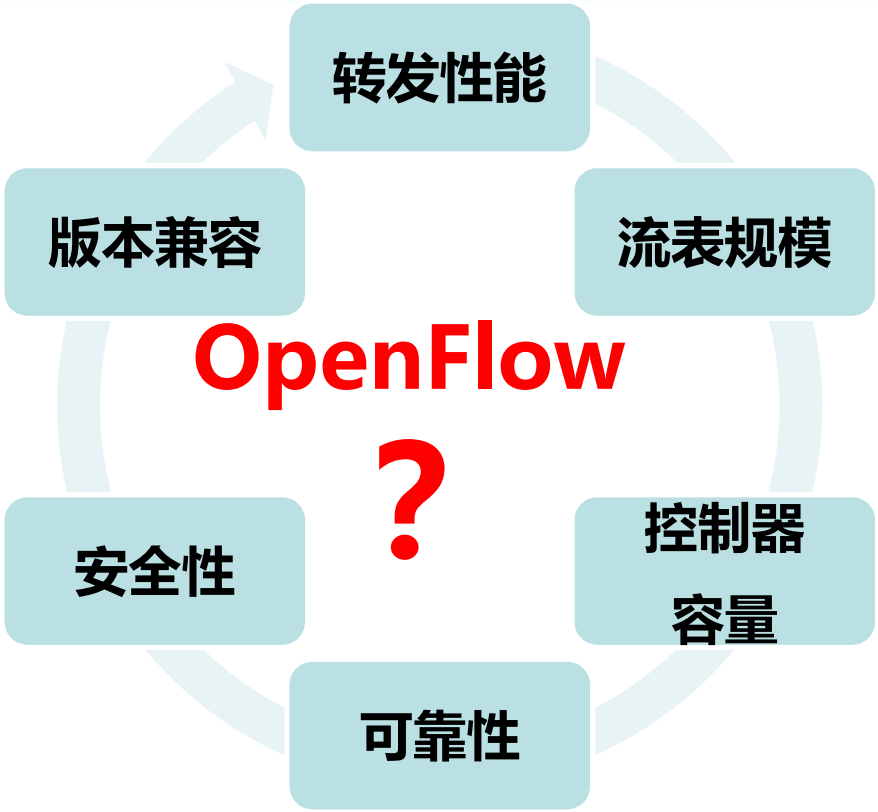


OpenFlow发展历史

- 自**2009年12月**发布第一个版本（**v1.0**）以来，已经有多个版本的**OpenFlow**规范（**OF**）被发布



OpenFlow的问题一览



OpenFlow的问题

```
/* OXM Flow match field types for OpenFlow basic class. */
enum oxm_ofb_match_fields {
    OFPXMT_OFB_IN_PORT        = 0,  /* Switch input port. */
    OFPXMT_OFB_IN_PHY_PORT    = 1,  /* Switch physical input port. */
    OFPXMT_OFB_METADATA       = 2,  /* Metadata passed between tables. */
    OFPXMT_OFB_ETH_DST        = 3,  /* Ethernet destination address. */
    OFPXMT_OFB_ETH_SRC        = 4,  /* Ethernet source address. */
    OFPXMT_OFB_ETH_TYPE       = 5,  /* Ethernet frame type. */
    OFPXMT_OFB_VLAN_VID       = 6,  /* VLAN id. */
    OFPXMT_OFB_VLAN_PCP       = 7,  /* VLAN priority. */
    OFPXMT_OFB_IP_DSCP        = 8,  /* IP DSCP (6 bits in ToS field). */
    OFPXMT_OFB_IP_ECN         = 9,  /* IP ECN (2 bits in ToS field). */
    OFPXMT_OFB_IP_PROTO       = 10, /* IP protocol. */
    OFPXMT_OFB_IPV4_SRC       = 11, /* IPv4 source address. */
    OFPXMT_OFB_IPV4_DST       = 12, /* IPv4 destination address. */
    OFPXMT_OFB_TCP_SRC        = 13, /* TCP source port. */
    OFPXMT_OFB_TCP_DST        = 14, /* TCP destination port. */
    OFPXMT_OFB_UDP_SRC        = 15, /* UDP source port. */
    OFPXMT_OFB_UDP_DST        = 16, /* UDP destination port. */
    OFPXMT_OFB_SCTP_SRC       = 17, /* SCTP source port. */
    OFPXMT_OFB_SCTP_DST       = 18, /* SCTP destination port. */
    OFPXMT_OFB_ICMPV4_TYPE    = 19, /* ICMP type. */
    OFPXMT_OFB_ICMPV4_CODE    = 20, /* ICMP code. */
    OFPXMT_OFB_ARP_OP         = 21, /* ARP opcode. */
    OFPXMT_OFB_ARP_SPA        = 22, /* ARP source IPv4 address. */
    OFPXMT_OFB_ARP_TPA        = 23, /* ARP target IPv4 address. */
    OFPXMT_OFB_ARP_SHA        = 24, /* ARP source hardware address. */
    OFPXMT_OFB_ARP_THA        = 25, /* ARP target hardware address. */
    OFPXMT_OFB_IPV6_SRC       = 26, /* IPv6 source address. */
    OFPXMT_OFB_IPV6_DST       = 27, /* IPv6 destination address. */
    OFPXMT_OFB_IPV6_FLABEL    = 28, /* IPv6 Flow Label */
    OFPXMT_OFB_ICMPV6_TYPE    = 29, /* ICMPv6 type. */
    OFPXMT_OFB_ICMPV6_CODE    = 30, /* ICMPv6 code. */
    OFPXMT_OFB_IPV6_ND_TARGET = 31, /* Target address for ND. */
    OFPXMT_OFB_IPV6_ND_SLL    = 32, /* Source link-layer for ND. */
    OFPXMT_OFB_IPV6_ND_TLL    = 33, /* Target link-layer for ND. */
    OFPXMT_OFB_MPLS_LABEL     = 34, /* MPLS label. */
    OFPXMT_OFB_MPLS_TC        = 35, /* MPLS TC. */
    OFPXMT_OFP_MPLS_BOS       = 36, /* MPLS BoS bit. */
    OFPXMT_OFB_PBB_ISID       = 37, /* PBB I-SID. */
    OFPXMT_OFB_TUNNEL_ID      = 38, /* Logical Port Metadata. */
    OFPXMT_OFB_IPV6_EXTHDR    = 39, /* IPv6 Extension Header pseudo-field */
    OFPXMT_OFB_PBB_UCA        = 41, /* PBB UCA header field. */
    OFPXMT_OFB_TCP_FLAGS      = 42, /* TCP flags. */
    OFPXMT_OFB_ACTSET_OUTPUT  = 43, /* Output port from action set metadata. */
    OFPXMT_OFB_PACKET_TYPE    = 44, /* Packet type value. */
```

新协议的设计严重依赖于**OF**协议中的匹配字段，而现有**OpenFlow**仅能是基于现有协议字段，且越来越臃肿

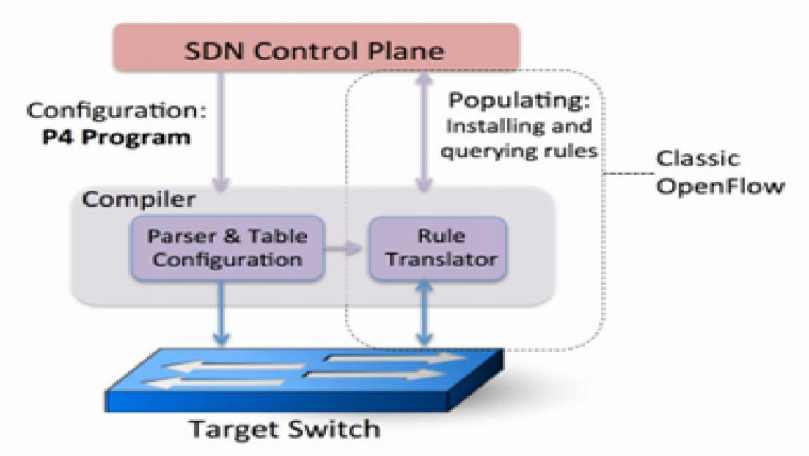
OpenFlow的问题

- 随着软件定义网络应用场景的扩展和网络技术的自身发展，OpenFlow需要支持越来越多的协议和报文处理方式
  - 数据中心的VXLAN、NVGRE 和STT 等网络虚拟化技术
  - NFV网络功能虚拟化技术
  - 网络安全以及监控和诊断
- 即便是针对常用标准协议，如TCP协议，OpenFlow 也不能对其头部的任意域进行匹配和处理
  - 防火墙中，很多场景需要对TCP某些状态位进行特定的报文处理动作
- 上述这些，目前的OpenFlow协议是都无法支持
- 如何能实现SDN倡导的网络灵活可编程理念？
  - 演进：P4、POF

新型SDN数据平面技术——P4之一

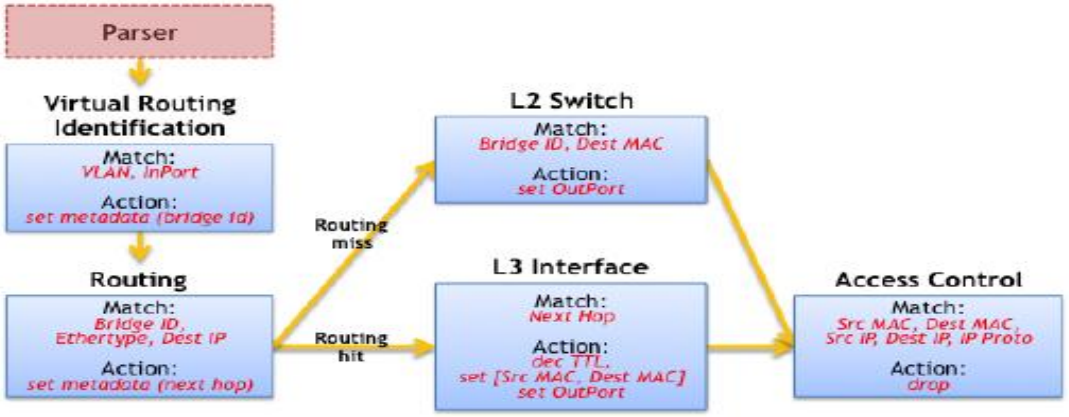
P4: Programming Protocol-Independent Packet Processors

- P4项目由众多SDN缔造者创建
- 致力于协议无关的网络编译语言和网络操作系统
  - Protocol independent
    - 自定义匹配字段
  - Packet processor
    - 定义表的能力与包处理流程
  - Programming
    - 一切硬件能力开放给用户



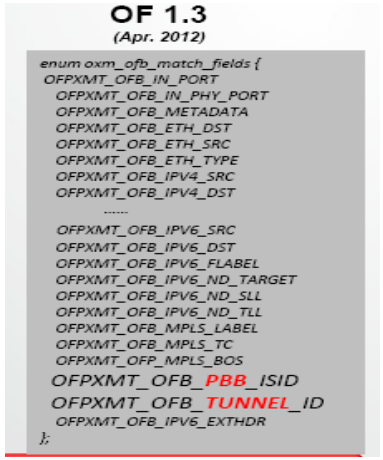
新型SDN数据平面技术——P4之二

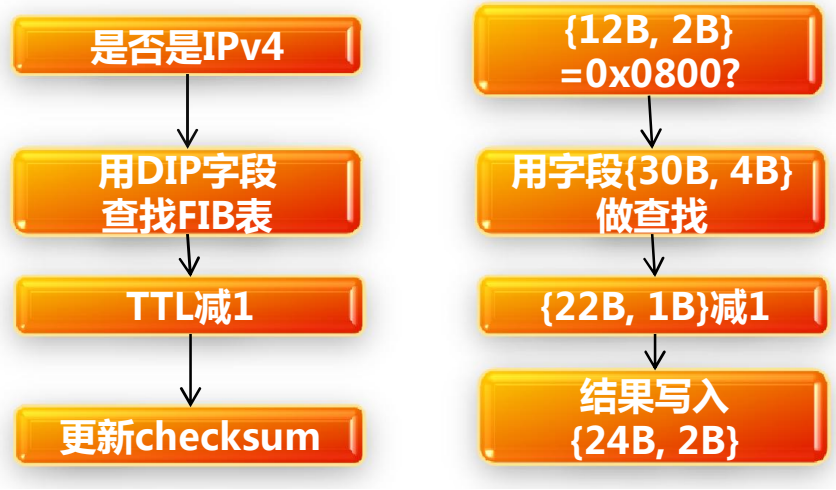
- P4采用两级结构
  - P4语言：抽象封装语句，任意组合描述流信息和处理动作
  - TDG（Table Dependency Graphs）：将P4语言翻译映射到硬件switch上
  - 向上支持不同的APP软件，向下支持不同的硬件
- 使网络开发者不需要懂交换设备和芯片，就可以像使用高级编程语言一样，通过书写语句和编译来开发业务APP、配置网络、创建新协议



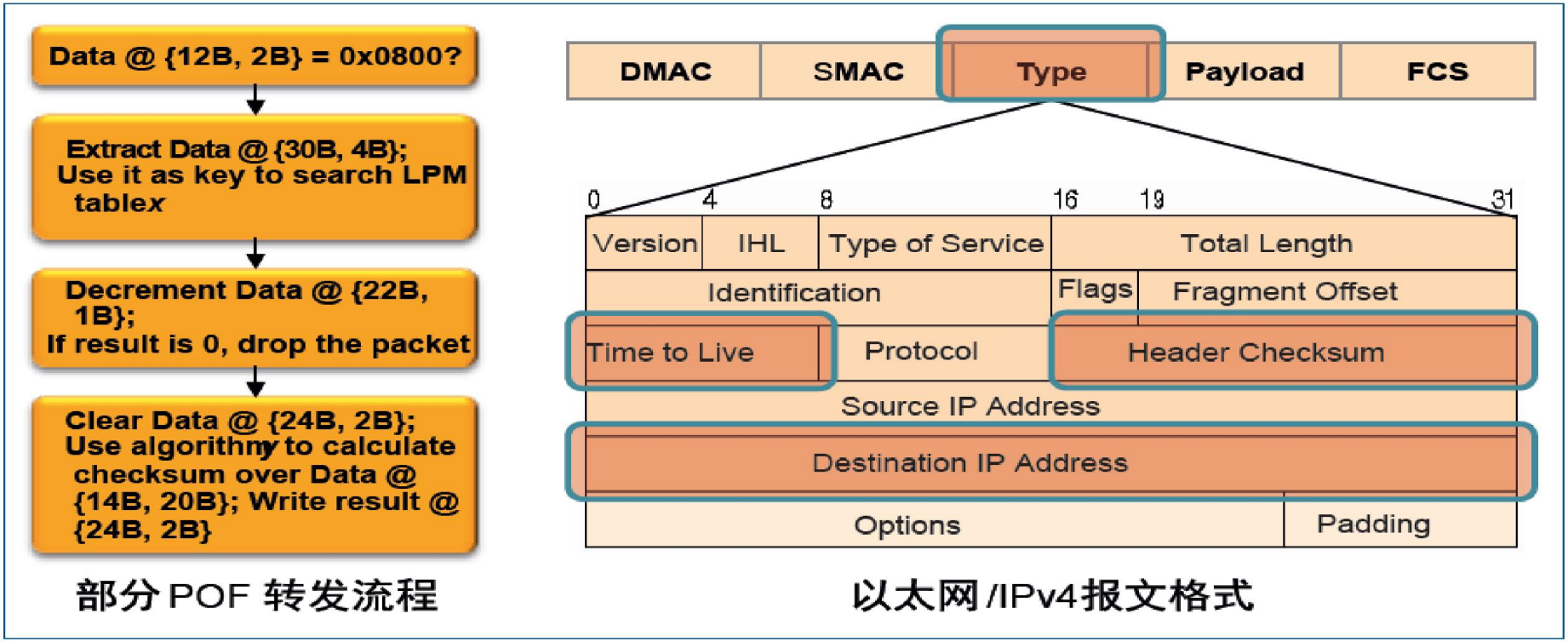
新型SDN数据平面技术——POF之一

- POF（ Protocol Oblivious Forwarding）项目由华为公司创建
- 面向协议无关的SDN底层平台和网络处理器芯片
  - 表项查找匹配key使用{offset，length，value}，而不用具体协议字段
  - 命令/动作，也采用{offset，length ，value} ，而不用具体协议字段
  - 灵活自由添加新协议、定义报文新动作，无需经过国际标准组织和厂商支持
  - 固定功能的NP芯片，通过微码编写{offset，length，value}通配符适配驱动，成为通用芯片

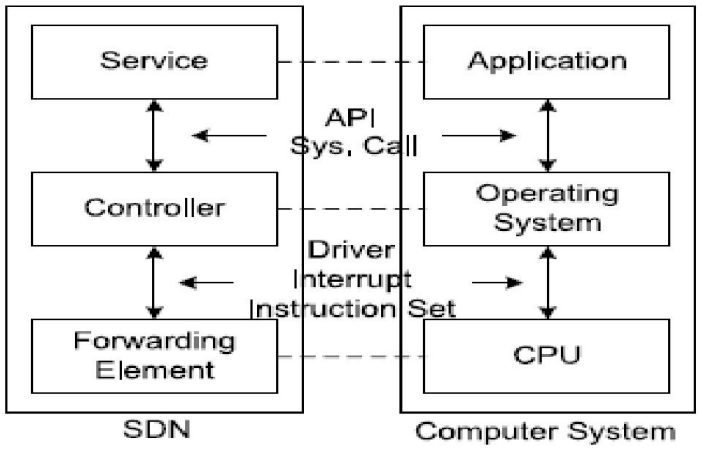




新型SDN数据平面技术——POF之二

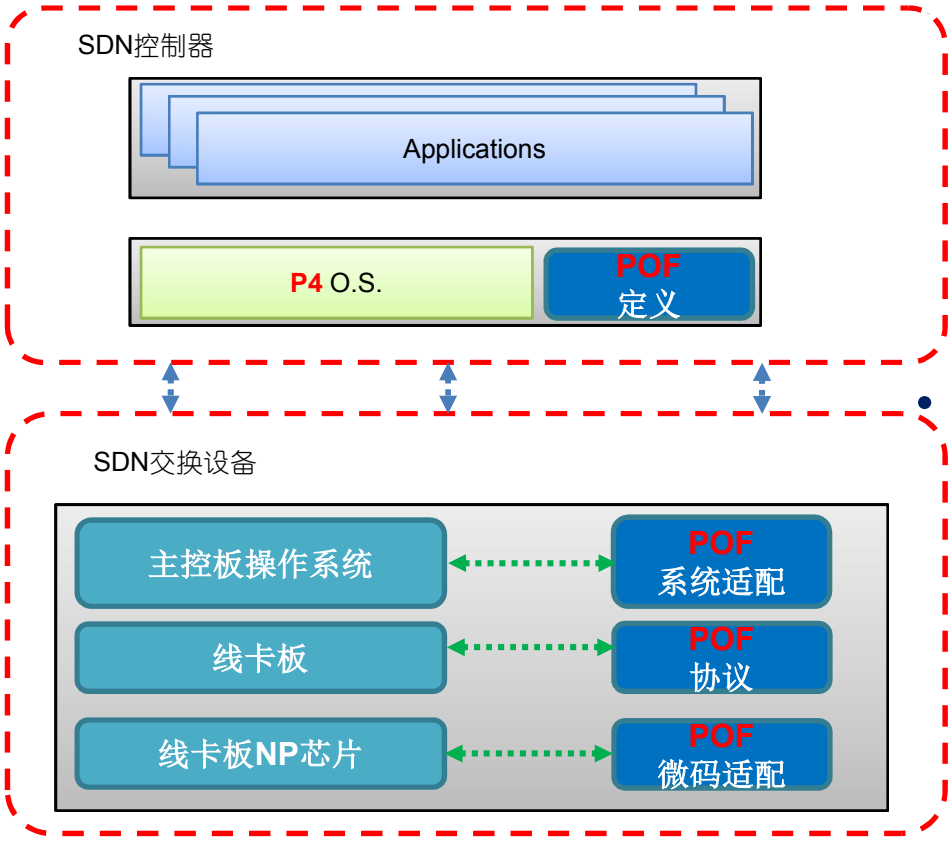


新型SDN数据平面技术——POF之三



- POF从真正意义上实现了网络设备的“电脑化”
  - 控制平面和数据平面所有协议的{offset，length，value}统一抽象化，打通了OpenFlow控制器和交换机间厚重的协议壁垒
  - 微码{offset，length，value}通配符驱动层，使专用网络处理器芯片成为CPU式的通用芯片
  - 保持了传统网络设备的高性能
  - 实现了OVS式的灵活可编程
- POF已成为OpenFlow2.0候选标准

新型SDN数据平面技术——P4与POF的区别与联系



P4

- 高级语言
- 可以编写出协议无感知的SDN控制器平台
- P4需要POF这样的转发元素抽象定义

POF

- 设备功能抽象和南向接口
- 可以编写出协议无感知的SDN交换设备
- POF需要P4这样的控制器高级编程语言

P4团队与POF团队也的确在密切合作，建立协议无感知转发生态系统

提 纲

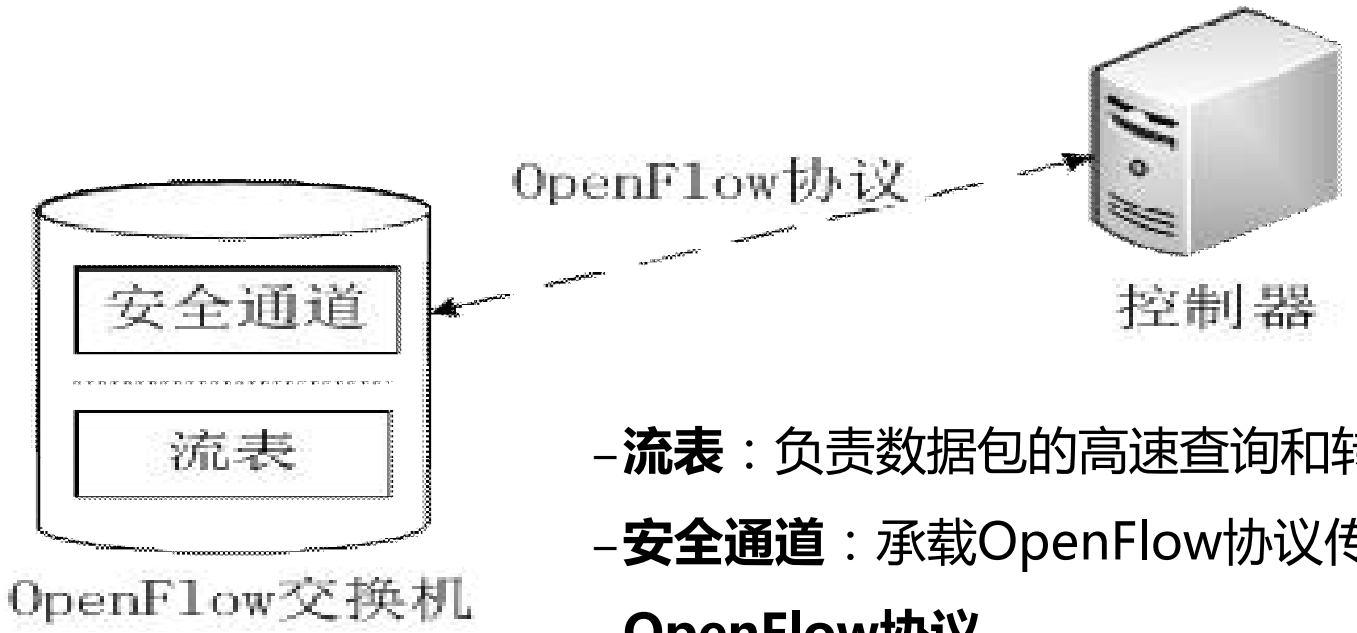
1 OpenFlow协议概述

2 OpenFlow 1.0协议介绍

3 OpenFlow协议演进

OpenFlowv1.0之协议架构

■**OpenFlow交换机利用基于安全连接的OpenFlow协议与远程控制器相通信**



- **流表**：负责数据包的高速查询和转发
- **安全通道**：承载OpenFlow协议传输
- **OpenFlow协议**

OpenFlowv1.0之流表结构

■**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

包头域	计数器	动作

–**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

OpenFlow1.0之流表包头域

- **用于匹配交换机接收到的数据包的包头内容，OpenFlow 1.0包头域包含12个元组（tuple）**
  - 涵盖ISO网络模型中第二至第四层的网络配置信息
  - 每一个元组中的数值可以是一个确定的值或者是“ANY
- **OpenFlow 1.1及后续版本将“包头域”更名为“匹配域”**

入端口	源MAC地址	目的MAC地址	以太网类型	VLAN ID	VLAN优先级	源IP地址	目的IP地址	IP协议	IP TOS位	TCP/UDP源端口	TCP/UDP目的端口
Ingress Port	Ether Source	Ether Des	Ether Type	VLAN ID	VLAN Priority	IP Src	IP Dst	IP Proto	IP ToS bits	TCP/UDP Src Port	TCP/UDP Dst Port

OpenFlow1.0之流表计数器

- **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，用于统计数据流量的相关信息**
  - 针对每张流表，统计当前活动的表项数、数据包查询次数、数据包匹配次数等
  - 针对每个数据流，统计接收到的数据包数、字节数、数据流持续时间等
  - 针对每个设备端口，除统计接收到的数据包数、发送数据包数、接收字节数、发送字节数等指标之外，还可以对各种错误发生的次数进行统计
  - 针对每个队列，统计发送的数据包数和字节数，还有发送时的溢出（Overrun）错误次数等

OpenFlow1.0之流表动作

■**用于指示交换机在收到匹配的数据包后应该如何对其进行处理**

- OpenFlow交换机缺少控制平面的能力，因此需要用动作来详细说明交换机将要对数据包所做的处理
- 每个流表项可以对应有零至多个动作，如果没有定义转发动作，那么与流表项包头域匹配的数据包将被默认丢弃
- 同一流表项中的多个动作的执行可以具有优先级，但是在数据包的发送上并不保证其顺序
- 如果流表项中出现有OpenFlow交换机不支持的参数值，交换机将向控制器返回相应的出错信息

■**必备动作(Required Actions) vs. 可选动作(Optional Actions)**

- 必备动作：需要所有OpenFlow交换机默认支持
- 可选动作：需要交换机告知控制器它所能支持的动作种类

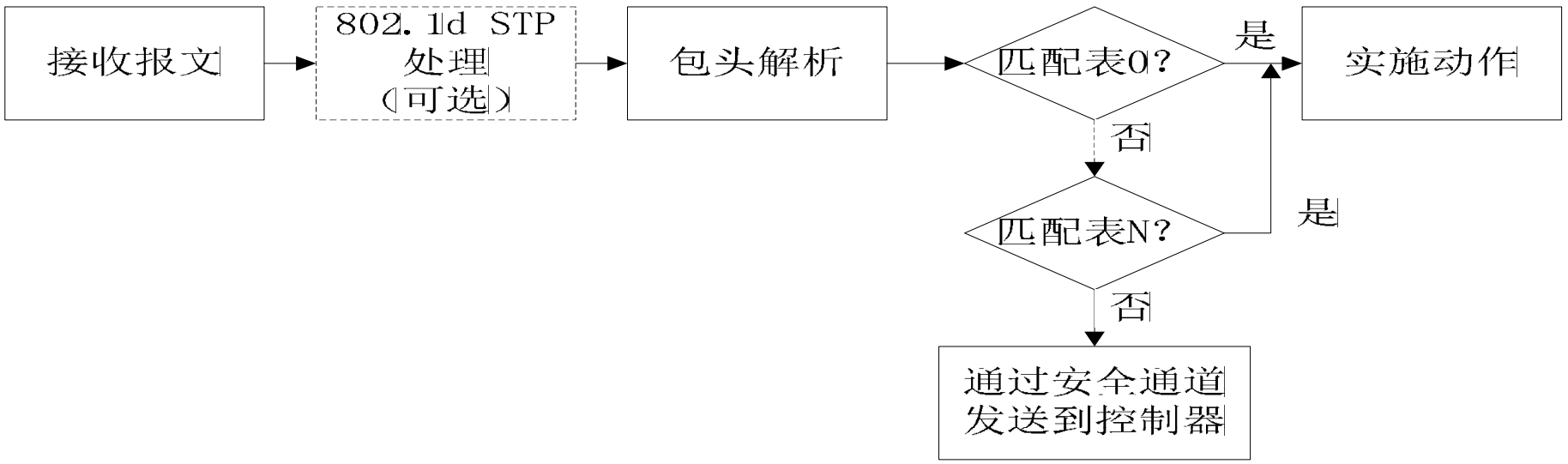
OpenFlow1.0——流表动作列表

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： ➢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，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

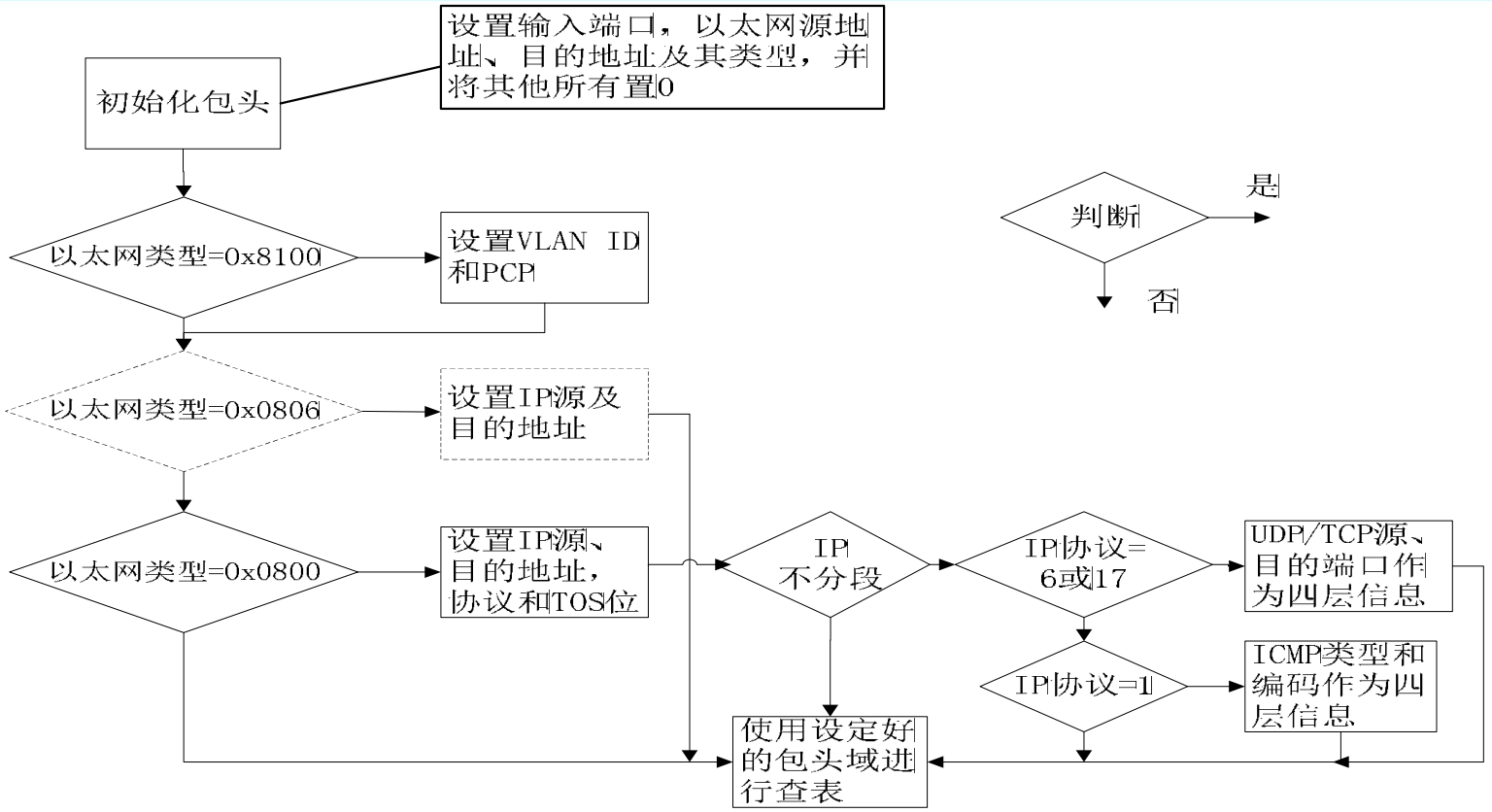
OpenFlow1.0——OpenFlow交换机分类

- **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。

OpenFlow1.0——数据包处理流程



OpenFlow1.0 ——包头解析匹配流程



OpenFlow1.0——协议消息类型

- **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（每一类消息又可以拥有多个子消息类型）**
  - **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

OpenFlow1.0——controller-to-switch消息列表

名称	说明	备注
Features	在建立TIS会话时，控制器发送features请求消息给交换机，交换机需要应答自身支持的功能	➢由控制器发起，对OpenFlow交换机进行状态查询和修改配置等操作 ➢OpenFlow交换机接收并处理可能发送或不需要发送的应答消息
Configuration	控制器设置或查询交换机上的配置参数,交换机仅需要应答查询消息	
Modify-state	控制器管理交换机流表项和端口状态等	
Read-state	控制器向交换机请求诸如流表、端口、各个流表项等方面的统计信息	
Send-packet	控制器通过交换机指定端口发出数据包	
Barrier	控制器通过barrier请求及相应报文，确认相关消息已经被满足或收到完成操作的通知	

OpenFlow1.0——asynchronous消息列表

名称	说明	备注
Packet-in	交换机收到一个数据包，在流表中没有匹配项，或者在流表中规定的行为是“发送到控制器”，则发送Packet-in消息给控制器；如果交换机缓存足够多，数据包被临时放在缓存中，数据包的部分内容（默认128字节）和在交换机缓存中的序号也一同发给控制器；如果交换机缓存不足以存储数据包，则将整个数据包作为消息的附带内容发给控制器	➢由OpenFlow交换机主动发起，用来通知交换机上发生的某些异步事件，消息是单向的，不需要控制器应答 ➢主要用于交换机向控制器通知收到报文、状态变化及出席错误等事件信息
Flow-removed	OpenFlow交换机中的流表项因为超时或收到修改/删除命令等原因被删除掉，会触发Flow-removed消息	
Port-status	OpenFlow交换机端口状态发生变化时，触发Port-status消息	
Error	OpenFlow交换机通过Error消息通知控制器发生的问题	

OpenFlow1.0——symmetric消息列表

名称	说明	备注
Hello	用于在OpenFlow交换机和控制器之间发起连接建立	➢不必通过请求建立，控制器和交换机都可以主动发起，并需要接受方应答 ➢消息为双向对称，主要用于建立连接、检测对方是否在线等
Echo	交换机和控制器均可以向对方发出Echo消息，接收者则需要回复Echo reply。该消息用来协商延迟、带宽、是否连接保持等控制器到OpenFlow交换机之间隧道的连接参数	
Vendor	用于OpenFlow交换机协商厂家自定义的附加功能，为未来版本预留	

OpenFlow1.0——主要协议交互过程之连接建立

- **连接建立**
  - 控制器与OpenFlow交换机建立TLS隧道后，隧道中传送的都是控制协议消息，因此隧道中的所有流量转发都无需查询交换机中的流表
  - 当OpenFlow安全隧道建立起来后，双方必须首先发送HELLO消息给对方，该消息携带本方支持的最高协议版本号，接收方将采用双方都支持的最低协议版本进行通信
  - 一旦发现两者拥有共同支持的协议版本，则连接建立，否则发送ERROR消息，描述失败原因，并终止连接

OpenFlow1.0——主要协议交互过程之连接中断

- **连接中断**
  - 当交换机与控制器之间的连接发生异常时，OpenFlow交换机应尝试连接备份控制器
  - 当多次尝试均失败后，OpenFlow交换机将进入紧急模式，并重置所有的TCP连接。此时所有包将匹配指定的紧急模式表项，而其它所有正常表项将被从流表中删除
  - 当交换机刚启动时，默认进入紧急模式

OpenFlow1.0——主要协议交互过程之通道加密

- **通道加密**
  - 控制器与OpenFlow交换机之间的安全通道采用TLS（Transport Layer Security）连接加密
  - 交换机启动时，尝试连接到控制器的6633 TCP 端口，进而双方通过交换证书进行认证
  - 每个交换机至少需配置两个证书，一个用来认证控制器，一个用来向控制器发出认证

OpenFlow1.0——主要协议交互过程之生成树支持

- **生成树支持**
  - OpenFlow交换机可以选择支持802.1d生成树协议。如果支持，所有相关包在查找流表之前应该先在本地进行传统生成树处理
  - 支持生成树协议的交换机在应答控制器的FEATURES消息的相应应答域中设置STP（Spanning Tree Protocol，生成树协议）支持位，并且需要所有物理端口均支持生成树协议，但无需在虚拟端口支持
  - 生成树协议会设置端口状态，从而限制发往FLOOD的数据包仅被转发到生成树指定端口。需要注意的是，已经指定了出端口的转发或发往ALL的数据包会忽略生成树所指定的端口，而按照规则的设置进行端口转发
  - 如果交换机不支持802.1d 生成树协议，则必须允许控制器指定洪泛（FLOOD）时的端口状态

OpenFlow1.0——主要协议交互过程之流表项修改

■**流表项修改**

■核心交互过程，通过控制器下发的流表项修改指令完成

■每条指令可能触发一系列OpenFlow协议消息

名称	说明
ADD	增加一个新的流表项
MODIFY	修改所有匹配的流表项
MODIFY_STRICT	修改严格匹配的流表项
DELETE	删除所有匹配流表项
DELETE_STRICT	删除严格匹配的流表项

OpenFlow1.0——主要协议交互过程之流表项移除

- **流表项移除**
  - 定时器计时结束：每个表项均有一个idle_timeout定时器和一个hard_timeout定时器（两者的计量单位都是秒），前者计算的是没有Flow匹配发生的时间，而后者则计算的是表项在流表中的总时间。一旦到达时间期限，交换机将自动删除该表项，同时发出一个流删除的消息
  - 控制器主动删除流表项：控制器通过下发DELETE_STRICT、 DELETE等指令相关的协议消息主动删除流表项

OpenFlow1.0——数据包处理方法

OpenFlow1.0提供两种数据包的处理方法：

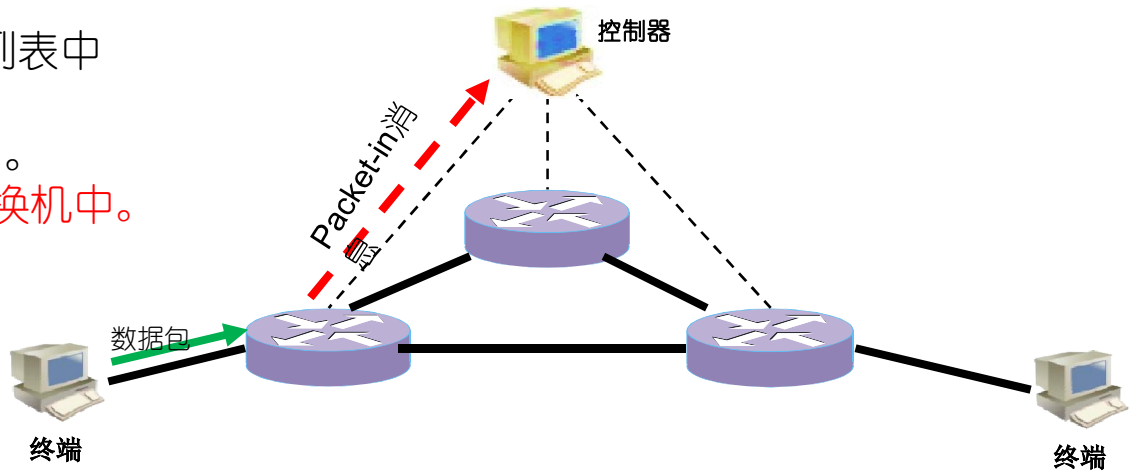
- 转发（Forward）
- 修改包头（Modify field）
  - SET_VLAN_VID 修改VLAN标签
  - SET_VLAN_PCP 修改VLAN优先级
  - STRIP_VLAN 弹出VLAN标签
  - SET_DL_SRC 修改源MAC地址
  - SET_DL_DST 修改目的MAC地址
  - SET_NW_SRC 修改源IP地址
  - SET_NW_DST 修改目的IP地址
  - SET_NW_TOS 修改IP服务类型字段
  - SET_TP_SRC 修改源端口号
  - SET_TP_DST 修改目的端口号

以上每一种操作称为一个动作（Action），流表中的数据包处理方法是一个动作列表（Action List），动作列表由以上各种动作组合合成。

Packet-in事件（交换机接收数据包）

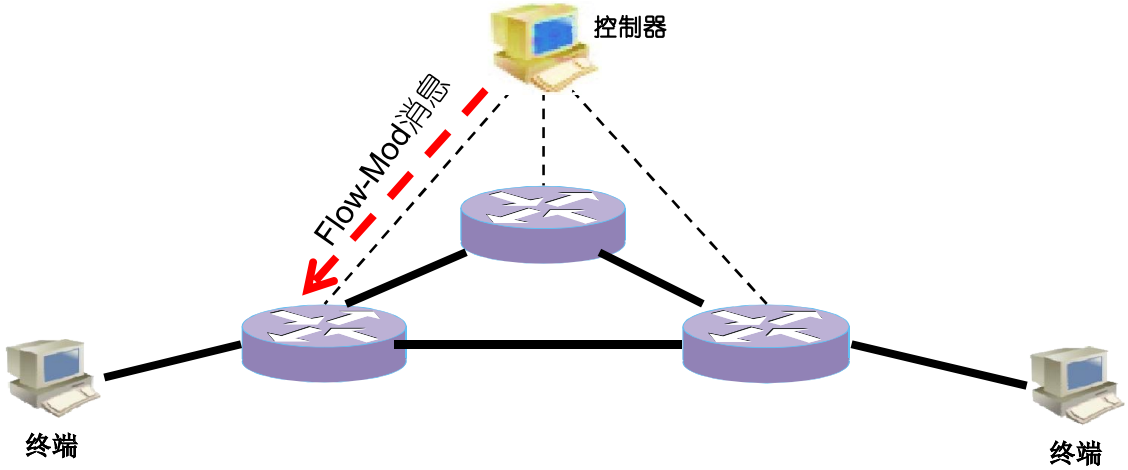
Packet-in消息触发情况1：
当交换机收到一个数据包后，会查找流表，找出与数据包包头相匹配的条目。
如果流表中有匹配条目，则交换机按照流表所指示的action列表处理数据包。
如果流表中没有匹配条目，则交换机会将数据包封装在Packet-in消息中发送给控制器处理。此时数据包会被缓存在交换机中等待处理。

Packet-in消息触发情况2：
交换机流表所指示的action列表中包含转发给控制器的动作（Output=CONTROLLER）。
此时数据包不会被缓存在交换机中。



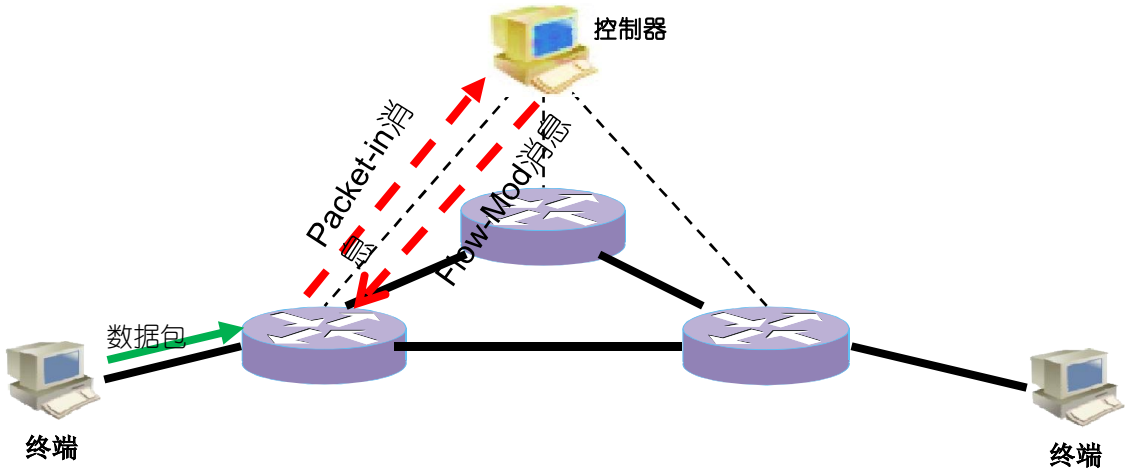
控制器配置流表（Flow-Mod消息）

Flow-Mod消息用来添加、删除、修改OpenFlow交换机的流表信息
Flow-Mod消息共有五种类型：
ADD、DELETE、DELETE-STRICT、MODIFY、MODIFY-STRICT
ADD类型的Flow-Mod消息用来添加一条新的流表项
DELETE类型的Flow-Mod消息用来删除所有符合一定条件的流表项
DELETE-STRICT类型的Flow-Mod消息用来删除某一条指定的流表项
MODIFY类型的Flow-Mod消息用来修改所有符合一定条件的流表项
MODIFY-STRICT类型的Flow-Mod消息用来修改某一条指定的流表项



用Flow-Mod消息响应Packet-in消息

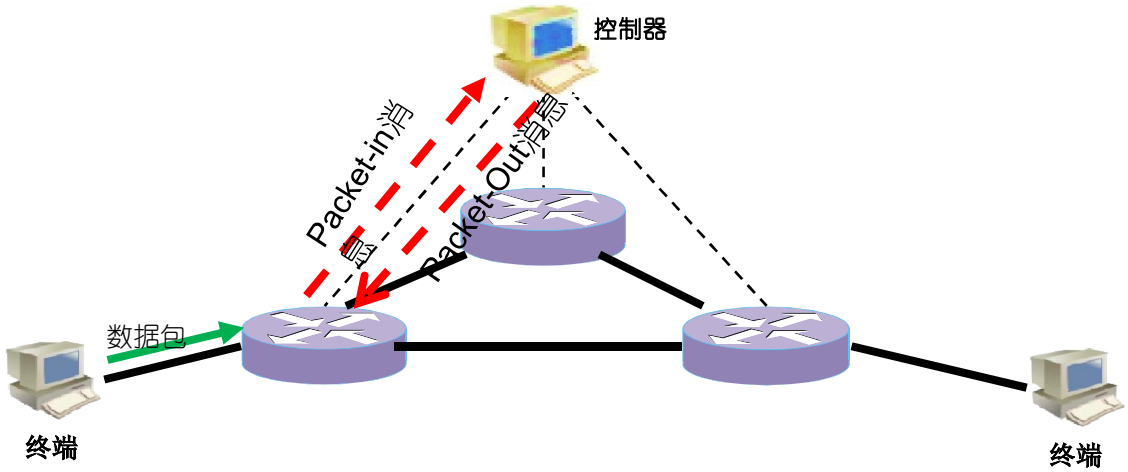
当交换机收到一个数据包并且交换机中没有与该数据包匹配的流表项时，交换机将此数据包封装到Packet-in消息中发送给控制器，并且交换机会将该数据包缓存。
控制器收到Packet-in消息后，可以发送Flow-Mod消向交换机写一个流表项。并且将**Flow-Mod消息中的buffer_id字段设置为Packet-in消息中的buffer_id值**。从而控制器向交换机写入了一条与数据包相关的流表项，并且指定该数据包按照此流表项的aciton列表处理。



交换机转发数据包（Packet-Out）

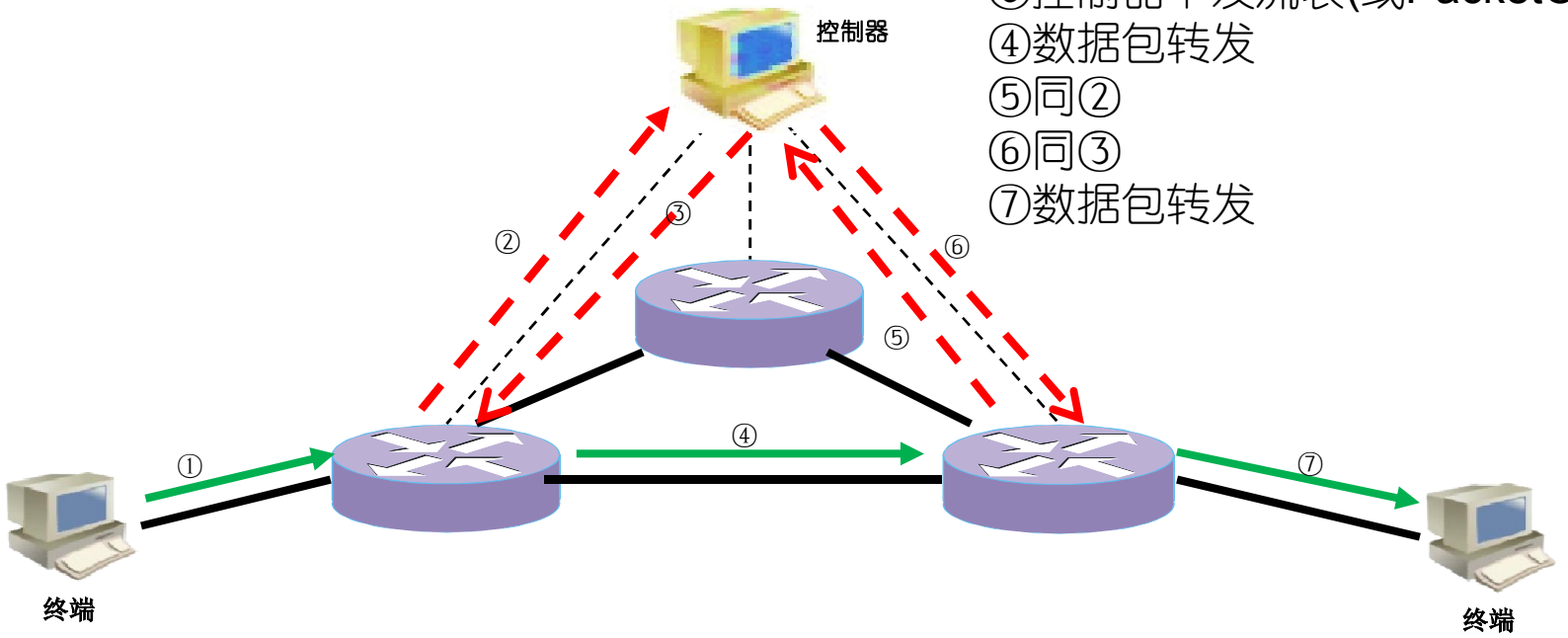
并不是所有的数据包都需要向交换机中添加一条流表项来匹配处理，网络中还存在多种数据包，它出现的数量很少（如ARP、IGMP等），以至于没有必要通过流表项来指定这一类数据包的处理方法。

此时，控制器可以使用PacketOut消息，告诉交换机某一个数据包如何处理。



基于OpenFlow的SDN工作流程

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



提 纲

1 OpenFlow协议概述

2 OpenFlow 1.0协议介绍

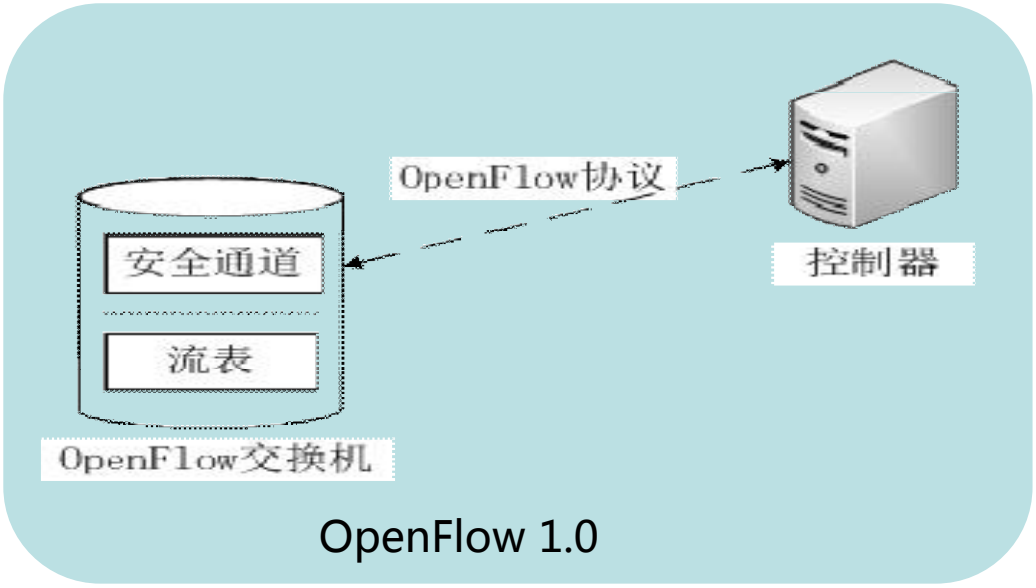
3 OpenFlow协议演进

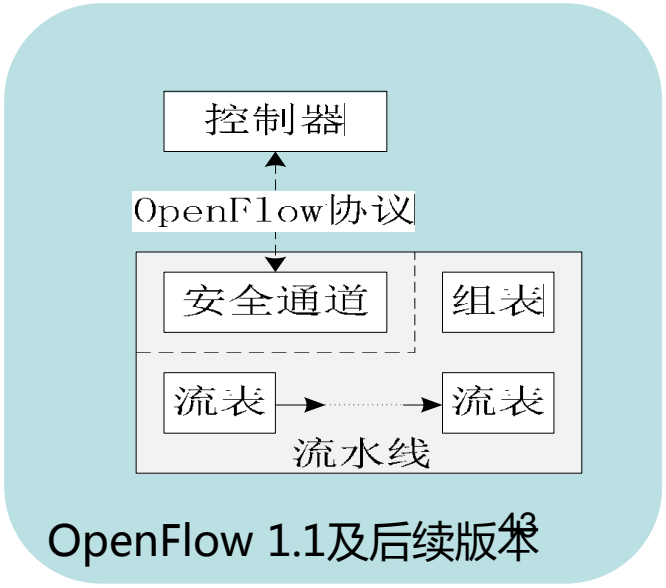
OpenFlow协议演进——交换机架构

■**自OpenFlow 1.1开始，交换机架构发生了变化以实现性能提升**

–流表由单一流表演变为由流水线串联而成的多流表

–增加了组表（Group Table）

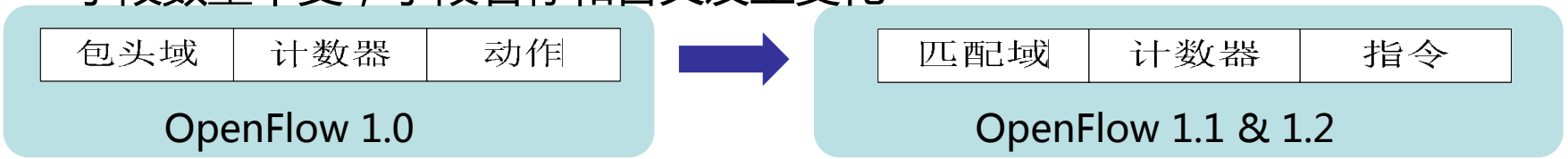




OpenFlow协议演进——流表结构

■OpenFlow 1.1 & 1.2

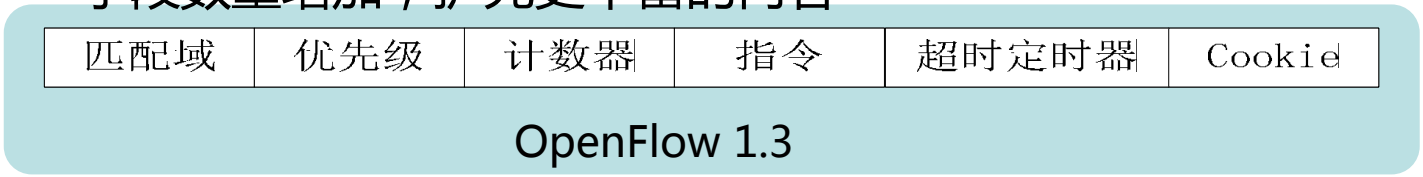
–字段数量不变，字段名称和含义发生变化



- Header Fields → Match Fields：表项中的入端口等信息不属于数据包头内容
- Actions → Instructions：多流表引入，使得数据包处理更为复杂

■OpenFlow 1.3

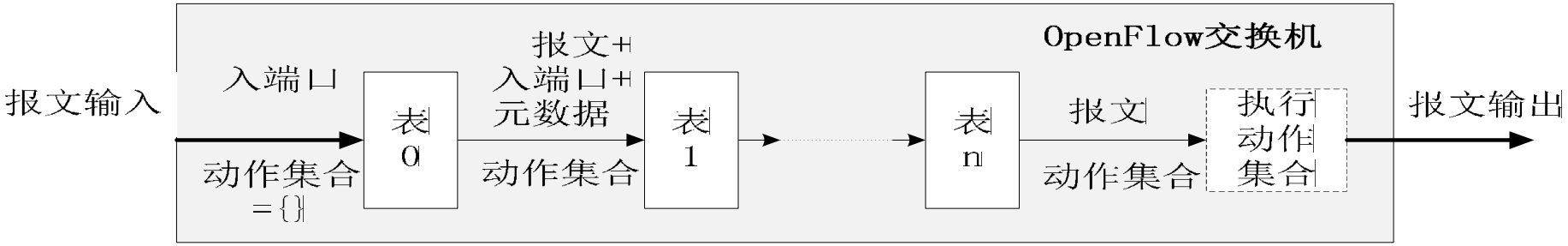
–字段数量增加，扩充更丰富的内容



- 优先级：设定表项的匹配顺序
- 超时定时器：限定流的最长有效时间或最大的空闲时间
- Cookie：过滤流统计数据、流改变和流删除等，不用于数据包处理

OpenFlow协议演进——多级流表

- **OpenFlow交换机对匹配内容长度不作区分，均以具有最大长度的表项为准（OpenFlow 1.0中，匹配字段为252位），造成TCAM成本的增加**
- **为了减小流表规模，OpenFlow 1.1及后续版本引入多级流表，基于流表特征提取将匹配过程分解为多个步骤，形成流水线处理模式，降低流表的记录条数**



–OpenFlow 1.3提出在多流表每个表的最都增加Table-miss项

OpenFlow协议演进——组表

■**OpenFlow 1.1及后续版本引入组表（Group Table）**

–适合于实现广播或组播，或者规定只执行某些特定的操作集

组标识符	组类型	计数器	动作桶

–其中，组类型规定了是否所有的动作桶中的指令都会被执行

–所有（all）：执行所有动作桶中的动作，可用于组播或广播

–选择（select）：执行该组中的一个动作桶中的动作，可用于多路径

–间接（indirect）：执行该组的一个确定的动作桶中的动作

–快速故障恢复（fast failover）：执行第一个具有有效活动端口的动作桶中的指令

OpenFlow协议演进——匹配域

■**随着OpenFlow的演进，匹配域（OpenFlow v1.0称作包头域）的覆盖范围越来越广，以满足更灵活的转发决策**

规范版本	匹配域数量	匹配域主要变化	备　注
1.0	12		
1.1	15	• 在OpenFlow 1.0基础上增加了元数据（Metadata）、MPLS标签、MPLS业务类别等3个匹配域 • 支持将OpenFlow 1.0中定义的IP协议、TCP/UDP源端口、目的端口进行复用	元数据不是数据包自带的，而是多流表处理时所需的附加信息，用于在同一交换机的不同流表间传递信息
1.2	36	• 将OpenFlow 1.1定义为可复用的匹配域全部拆分 • 增加对IPv6信息的匹配	规定OpenFlow交换机必须支持13个匹配域
1.3	39	• 增加了PBB、tunnel-ID及IPv6扩展头的匹配	OpenFlow交换机必须支持的匹配域与OpenFlow 1.2相同

OpenFlow协议演进——计数器

- **OpenFlow 1.1增加了组表的概念，因此计数器相应增加了针对每组、每个动作桶的相关统计**
- **OpenFlow 1.3增加了针对数据流的计量，因此计数器相应增加了针对各个数据流计量表（meter）的统计**

OpenFlow协议演进——指令

- **OpenFlow 1.0中将流表项与数据包匹配后对其进行的操作称为动作，而在OpenFlow 1.1及其后续版本中都将其改称为指令**
- **针对OpenFlow功能的增加（例如多流表、组表、数据流计量），OpenFlow 1.1和1.3先后增加了5条指令和1条指令**

规范版本	指令类型	指令名称	说明
v1.1	可选指令	Apply-Actions	立即进行指定动作，而不改变动作集合。经常在修改数据包，以及在两个表之间执行同类型的多个动作时使用
	可选指令	Clear-Actions	在动作集合中立即清除所有的动作
	必备指令	Write-Actions	将指定的动作添加到正在运行的动作集合中
	可选指令	Write-Metadata metadata/mask	在元数据区域记录元数据
	必备指令	Goto-Table next-table-id	转到流水线处理进程中的下一张表的ID
v1.3	可选指令	Meter meter id	直接将包计量后丢弃

OpenFlow协议演进——动作

- **OpenFlow指令的执行可能会导致数据包在多流表之间的转移，也可能会指示交换机对数据包采取真正的动作**
- **OpenFlow 1.0定义了必备的转发、丢弃，以及可选的转发、排队、修改域等动作，后续版本对其进行了完善**
  - 必备动作**Output**用于将数据包输出到指定端口。OpenFlow 1.1新增了可选动作Output及相应的LOCAL端口，在OpenFlow 1.2及后续版本中定义了必备动作Output及相应的OpenFlow端口。因此，必备动作与可选动作被合并为统一的Output执行，并由端口属性决定其是否为可选
  - 必备动作**Drop**用于丢弃数据包，其定义没有变化，与OpenFlow 1.0相同
  - 可选动作**Set-Queue**用于设置数据包的队列ID，以完成QoS功能。在OpenFlow 1.0中该动作被称作Enqueue，OpenFlow 1.1将其更名
  - 必备动作**Group**用于利用组表处理数据包，在OpenFlow 1.1中增加
  - 可选动作**Push-Tag/Pop-Tag**用于VLAN、MPLA Tag的入栈和出栈，在OpenFlow 1.1中增加
  - 可选动作**Set-Field**用于设置数据包头的类型和修改数据包头的值，该动作在OpenFlow 1.0中被称作Modify-field，OpenFlow1.1及后续版本将其更名为Set-Field
  - 可选动作**Change-TTL**用于修改TTL值，在OpenFlow 1.2中增加

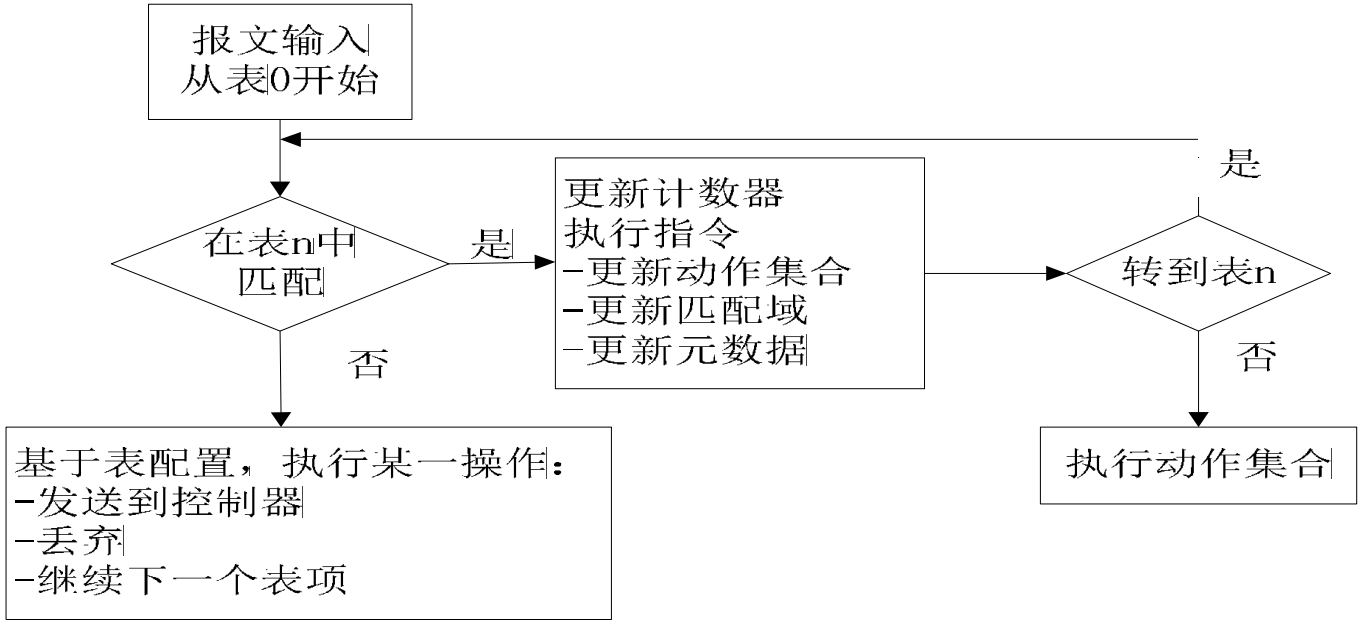
OpenFlow协议演进——动作集合

■**OpenFlow 1.1引入多级流表，提出动作集合（action set）**

- 动作集合与数据包处理相关，最初为一个空集合，每匹配一次流表项，匹配项对应的指令都可能增加数据包的动作集合，从而多个流表项的指令中的动作会在动作集合中累加
- 当一个表项的指令集没有包含Goto-Table指令时，流水线处理就停止了，那么数据包对应动作集合中的动作指令将被执行
- 动作集合中包含所有的动作/指令，无论它们以什么样的顺序加入到集合中，其执行都是按照一定顺序的
- 如果动作集合中包含了组（Group）动作桶中的操作，那么动作桶中的指令也按照相应的顺序执行
- OpenFlow交换机可以利用Apply-Actions指令修改具体的动作/指令的执行顺序

OpenFlow协议演进——流表匹配流程之一

- **流表的匹配在OpenFlow 1.1引入多流表后变得更加复杂，需要依次对多张流表中的流表项进行比对后，按照匹配成功与否执行相应的操作，直到不再有GOTO的流转后，才最终执行对应动作集合**

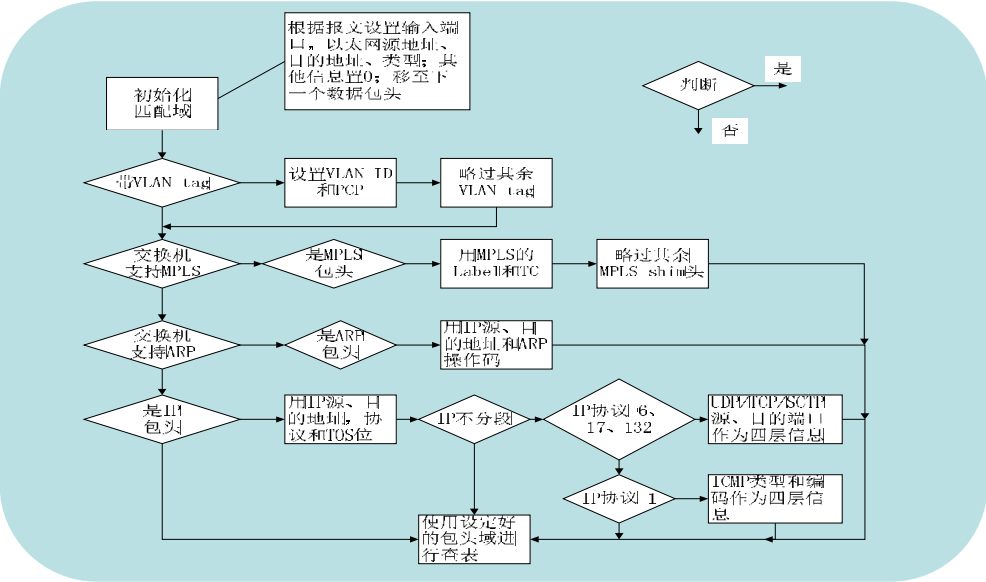


OpenFlow协议演进——流表匹配流程之二

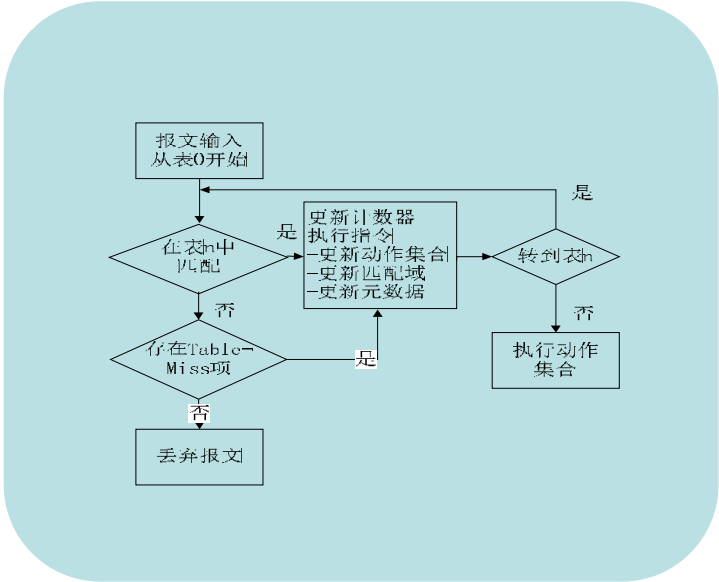
■**不同OpenFlow版本具有不同的匹配域定义**

– OpenFlow 1.1增加针对MPLS匹配域复用的匹配过程

– OpenFlow 1.3增加table-miss流表项处理未发生匹配的数据包



OpenFlow 1.1单个数据包匹配



OpenFlow 1.3多流表匹配

OpenFlow协议演进——OpenFlow协议消息

- **OpenFlow 1.1保持消息数量不变，但是更改了部分消息名称**
  - controller-to-switch消息类型中的Send-packet消息改名为Packet-out消息
  - symmetric消息类型中的Vendor改名为Experimenter
- **OpenFlow 1.3增加了两个controller-to-switch消息**
  - Role-Request：用于控制器向其OpenFlow通道进行角色的设置或查询
  - Asynchronous-Configuration：用于控制器设置或查询异步消息的附加过滤器，一般用于多控制器的连接建立

OpenFlow协议演进——安全通道

- **安全通道是控制器与交换机之间的接口**
  - OpenFlow 1.0规定该安全通道需要使用TLS安全隧道
  - OpenFlow 1.1开始，OpenFlow不再强制要求使用TLS隧道，而是可以使用普通的TCP连接
  - 各个OpenFlow版本均建议在缺省情况下将TCP 6633端口用于安全通道

OpenFlow协议演进——OpenFlow端口之一

■**OpenFlow 1.2及后续版本提出OpenFlow端口的概念，它是OpenFlow进程和网络之间传递数据包的网络接口，OpenFlow交换机之间通过OpenFlow端口在逻辑上相互连接**

- **物理端口**：OpenFlow物理端口为交换机定义的端口，与OpenFlow交换机上的硬件接口一一对应。在某些部署中，OpenFlow交换机可以实现交换机的硬件虚拟化。在此情况下，一个OpenFlow物理端口可以对应交换机硬件接口的一个虚拟接口

- **逻辑端口**：OpenFlow逻辑端口为交换机定义的端口，但并不直接对应一个交换机的硬件接口。逻辑端口是更高层次的抽象概念，可以是交换机中定义的其它一些端口（例如链路聚合组、隧道、环回接口等）。逻辑端口可能支持报文封装并被映射到不同的物理端口上，但其处理动作必须是透明的，即OpenFlow在处理上并不刻意区分逻辑端口和物理端口的差异。物理端口和逻辑端口之间的唯一区别是：一个逻辑端口的数据包可能增加了一个额外的元数据字段即隧道ID，而当一个逻辑端口上接收到的报文被发送到控制器时，其逻辑端口和底层的物理端口都要报告给控制器

- **保留端口**：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

OpenFlow协议演进——OpenFlow端口之二

■OpenFlow定义了8种端口

类型	名称	说明
必备	ALL	表示所有端口均可用于转发指定数据包。当其被用作输出端口时，数据包被复制后发送到所有的标准端口，但不包括数据包的入端口及被配置为OFPPC_NO_FWD的端口
	CONTROLLER	表示到OpenFlow控制器的控制通道，它可以用作一个入端口或作为一个出端口。当用作一个出端口时，封装数据包中为数据包消息，并使用OpenFlow协议发送。当用作一个入端口时，确认来自控制器的数据包
	TABLE	表示OpenFlow流水线的开始。这个端口仅在输出行为的时候有效，此时OpenFlow交换机提交报文给第一流表使数据包可以通过OpenFlow流水线处理
	IN PORT	代表数据包进入端口。只有一种情况用于输出端口，即从入端口发送数据包
	ANY	某些OpenFlow命令中的特定值，用在没有指定端口的（端口通配符）情形，不能用于入端口和出端口
可选	LOCAL	表示交换机的本地网络堆栈和管理堆栈。可以用作一个入端口或者一个出端口。该端口使得远程实体可以与交换机通过OpenFlow网络交互，而不再需要单独的控制网络。通过配置一组合适的默认流表项，该端口可用于实现一个带内控制器的连接
	NORMAL	代表传统的非OpenFlow流水线处理。仅可用作普通流水线的输出端口。如果交换机不能从OpenFlow流水线转发数据包到普通流水线，它必须表明它不支持这一动作
	FLOOD	表示使用普通流水线处理洪泛过程。可用于作为一个输出端口，除入端口或OFPPS_BLOCKED状态的端口外，可以将数据包发往其他所有标准端口。交换机也可以通过数据包的VLAN ID选择哪些端口洪泛

OpenFlow协议演进——IPv6支持

- **OpenFlow 1.2规定了对IPv6基本协议的支持**
  - 增加IPv6源地址和目的地址、IP协议号（与IPv4相同，不区分IPv4还是IPv6）、业务类型（同IPv4 TOS）、ICMPv6类型和编码，以及对IPv6邻居发现头域及IPv6流标签等匹配域的支持
- **OpenFlow 1.3增加了对IPv6扩展头的支持**

OpenFlow协议演进——多控制器

- **OpenFlow采用集中化的控制方式，一旦控制器出现故障或者其与交换机之间的连接中断，将会对整个网络造成影响**
- **OpenFlow 1.2引入多控制器的理念，希望通过多个控制器的协同工作提高全网的可靠性**
  - OpenFlow交换机在其初始化时，即与一至多个配置好的控制器建立连接。多个控制器之间可以提供负载均衡能力和快速故障倒换，同时增加角色类消息用于控制器之间协商主备关系
  - 控制器的角色在缺省情况下为EQUAL，在此状态下的控制器可以响应来自OpenFlow交换机发来的请求；控制器角色也可以设为SLAVE，在此状态下控制器只负责监听，不响应交换机发送的消息；另外，控制器还可以是MASTER角色，这种状态下的控制器行为与EQUAL类似，唯一的差异在于系统中只能有一台MASTER

OpenFlow协议演进——计量表之一

- **OpenFlow 1.3增加了对单个数据流的计量功能，使得OpenFlow能够实现简单的QoS服务（例如端口限速），并且可以结合每个端口队列来实现更复杂的QoS框架（例如DiffServ）**
- **计量表（Meter Table）：包含若干针对各个数据流的计量表项**

计量标识符	计量带	计数器

  - 计量标识符：32位的无符号整数，用来唯一识别该计量表项。
  - 计量带：由计量带组成的无序列表，其中每个计量带都指明了其速率及处理数据包的方式。
  - 计数器：用于在报文被计量表项处理时更新相关计数

OpenFlow协议演进——计量表之二

■**每个计量表项可能具有有一个或多个计量带，每个计量带都指定了其所适用的速率和数据被处理的方式**

带类型	计量速率	计数器	类型参数

- （计量）带类型：定义了数据包怎样被处理。
- 计量速率：用于选择计量带，定义了带可以运行的最低速率。
- 计数器：当数据报文被计量带处理时，更新计数。
- 类型的具体参数：带类型的可选参数

■**计量带未作做必备要求，当前只有两种可选的计量带类型**

- drop：通过丢弃数据包，定义带宽速率限制
- dscp remark：降低数据包的IP头中的DSCP字段丢弃的优先级，可用于定义简单的DiffServ策略

Q & A

谢谢大家！