OpenFlow1.3 协议总结

1 介绍

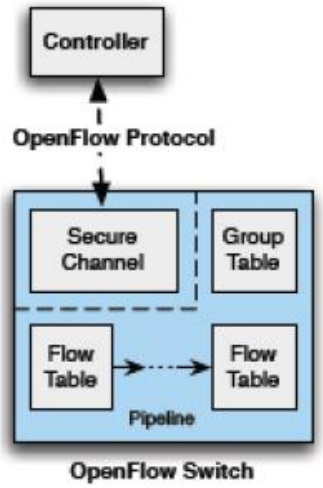


Figure 1: Main components of an OpenFlow switch.

2 交换机组成

OpenFlow 的交换机包括一个或多个流表和一组表，执行分组查找和转发，和到一个外部控制器 OpenFlow 的信道。
控制器使用 OpenFlow 的协议，它可以添加、更新和删除流流表中的表项，既主动或者被动响应数据包。 在交换机中的每个流表中包含的一组流 表项;每个流表项包含匹配字段，计数器和一组指令，用来匹配数据包。

3 名称解释

Pipeline（流水线）：在一个 openflow 交换机中提供匹配、转发和数据包修改功能的流表连接集合。
Metadata（元数据）：一个可屏蔽寄存器的值，用于携带信息从一个表到下一个。
Group（组）：一系列的行动存储段和一些选择一个或者多个存储段应用到数据包单元的手段。
Meter（计量）：一个交换机元件，可以测量和控制数据包的速度。当数据包速率或通过计量的字节速率超过预定义的阈值时，计量触发计量带。如果计量带丢弃该数据包，它则被称为一个速率限制器。

4 端口

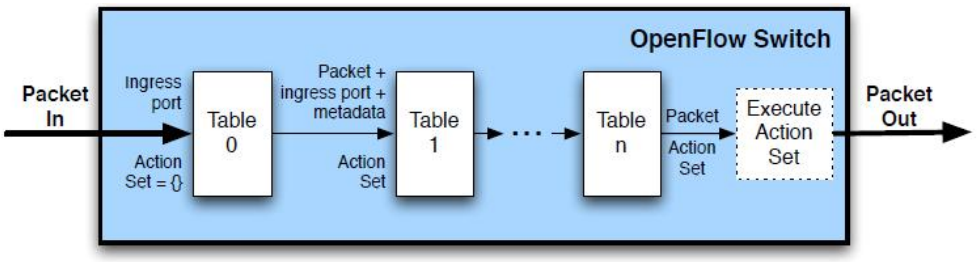
物理端口：交换机定义的端口，对应交换机的硬件接口
逻辑端口：交换机定义的端口，包括报文封装，可以映射到不同的物理端口。
保留端口：本协议定义的端口，指定通用的转发动作，如发送到控制器、泛洪或使用非

OpenFlow 的方法转发。

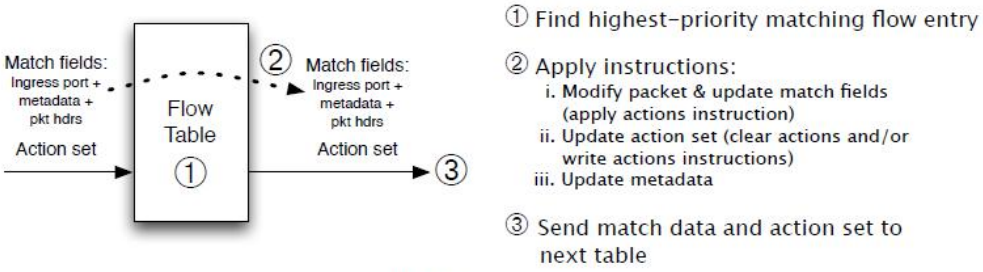
5 OpenFlow 表

5.1 Pipeline 处理

1、OpenFlow-only 交换机只支持 OpenFlow 操作，所有数据包都由 OpenFlow 流水线处理。OpenFlow-hybrid 交换机支持 OpenFlow 的操作和普通的以太网交换操作，即传统的 L2 以太网交换、VLAN 隔离、L3 路由（ IPv4 的路由， IPv6 路由）、ACL 和 QoS 处理。这种交换机必须提供一个 OpenFlow 外的分类机制，使流量路由到 OpenFlow 流水线或普通流水线。

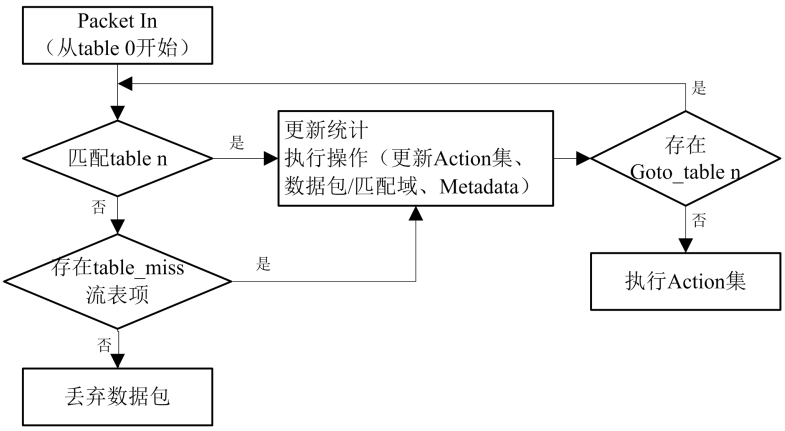


(a) Packets are matched against multiple tables in the pipeline



(b) Per-table packet processing

Figure 2: Packet flow through the processing pipeline.



OpenFlow交换机的流表按顺序编号的，从0开始。流水线处理总是从第一流表开始：数据包第一个与流表0的流表项匹配。其他流表根据第一个表的匹配结果来调用。

根据某个流表进行处理时，将数据包与流表中的流表项进行匹配，从而选择流表项（见5.3）。如果匹配到了流表项，那么包括在该流表项的指令集被执行时，这些指令可能明确指导数据

包传递到另一个流表（使用Goto指令，见5.9），在那里同样的处理被重复执行。表项只能指导数据包到大于自己表号的流表，换句话说流水线处理，只能前进，而不能后退。显然，流水线的最后一个表项可以不包括GOTO指令。如果匹配的流表项并没有指导数据包到另一个流表，流水线处理将停止在该表中。当流水线处理停止，数据包被与之相关的行动集处理并通常被转发（见5.10）。
如果数据包在流表中没有匹配到流表项，这是一个 table-miss 的行为。table-miss 行为依赖于表的配置（见 5.4）。一个 table-miss 的流表中的表项可以指定如何处理无法匹配的数据包：包括丢弃，传递到另一个表中，或凭借数据包中的信息通过控制通道发送到控制器（见 6.1.2）。

5.2 Flow Table

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie

匹配字段：数据包匹配，包括入端口和数据包头（如以太网源地址或 IPv4 目的地址），以及由前一个表指定的可选的元数据（用来在一个交换机的不同表里面传递信息）。
优先级：流表项的匹配次序。
计数器：数据包匹配时更新计数。
指令：修改行动集或流水线处理。
超时：最大时间计数值或流在交换机中失效之前的剩余时间。
cookie ：由控制器选择的不透明数据值。控制器用来过滤流统计数据、流修改和流删除。但处理数据包时不能使用。

5.3 Match

5.4 Table-miss

table-miss 表项指定在流表中如何处理与其他流表项未匹配的数据包（见 5.1）。比如数据包发送到控制器，丢弃数据包或直接将包扔到后续的表。table-miss 的流表项也有它的匹配字段和优先级，它通配所有匹配字段（所有领域省略），并具有最低的优先级（0）。table-miss 流表项指令至少支持利用将数据包发送到控制器保留端口，和使用 Clear-Actions 指令丢弃数据包

5.5 流表项删除

1）交换机超时：每个流的表项具有一个和它相关的 idle_timeout 和 hard_timeout 值。如果两个值中有一个不为零，交换机必须注意的流表项的老化时间，因为交换机可能删除该项。如果给定非零 hard_timeout 的值，那么一段时间后，可以导致流表项被删除，无论有多少数据包与之匹配。如果给定非零 idle_timeout 的值，那么如果在一段时间没有报文与之匹配，可以导致流表项被删除。交换机必须实现流表项超时和删除功能。
2）控制器请求：发送流表修改信息（OFPFC_DELETE，或 OFPFC_DELETE_STRICT）删除流表项。流表项被删除时，无论是控制器控制或流表项超时机制，交换机必须检查流表项的 OFPFF_SEND_FLOW_REM 标志。如果该标志被设置，该交换机必须将流删除消息发送到控制器。每个流清除消息中包含的流表项的完整的描述、清除的原因（超时或删除），在

清除时的流表项的持续时间，在清除时的流的统计数据。

5.6 组表

组表是一组泛洪的指令集，以及更复杂的转发（如多路径，快速重路由，链路聚合）。组表包含若干组表项，每个组表项包含一系列依赖于组类型的特定含义行动存储段。一个或多个行动存储段里的行动会作用到发送到该组的数据包。

Group Identifier	Group Type	Counters	Action Buckets

Group_ID: 一个 32 位的无符号整数，唯一标识该组

Group Type：All(Required):执行组的全部存储段，用于多播或广播的转发

Select(Optional):执行组的一个存储段

Indirect(Required):执行组内的一个定义存储段，只支持单一的存储段，但允许多个流表项或组指向一个共同的 Group_ID

Fast failover(Optional):执行第一个有效的存储段

*交换机只支持那些标记为“Required”的组类型，控制器可以查询交换机支持哪些“Optional”组类型

Counters：当数据包被组处理时更新

Action Buckets：有序的行动存储段，其中的每个行动存储段包含了一组要执行的行动和相关参数。

5.7 Meter Table

计量器可以测试数据包分配的速率，并可以控制数据包的速率。计量器直接连接到流表项（而不是被连接到端口的队列）。任意的流表项可以在它的指令集中定义一个计量器（见 5.9），计量器测量和控制和它有关的所有流表项的总速率。

Meter Identifier	Meter Bands	Counters

Meter_Identifier：一个 32 位的无符号整数唯一识别

Meter_band:

Band Type	Rate	Counters	Type specific arguments

Band Type：定义数据包如何被处理

Rate：选择 meter band，定义 band 应用的最低速率

Counters：meter_band 处理数据包时更新

type specic arguments：带类型的可选参数

drop(Optional)：丢弃数据包，可以用来定义速率限制带。

dscp remark(Optional)：增加数据包的 IP 头部 DSCP 字段丢弃的优先级。可用于定义一个简单的 DiffServ 策略。

5.8 Counters

交换机不要求支持所有的计数器，只有那些标记为“ Required ”是必须支持的。

Counter	Bits	
Per Flow Table		
Reference Count (active entries)	32	*Required*
Packet Lookups	64	*Optional*
Packet Matches	64	*Optional*
Per Flow Entry		
Received Packets	64	*Optional*
Received Bytes	64	*Optional*
Duration (seconds)	32	*Required*
Duration (nanoseconds)	32	*Optional*
Per Port		
Received Packets	64	*Required*
Transmitted Packets	64	*Required*
Received Bytes	64	*Optional*
Transmitted Bytes	64	*Optional*
Receive Drops	64	*Optional*
Transmit Drops	64	*Optional*
Receive Errors	64	*Optional*
Transmit Errors	64	*Optional*
Receive Frame Alignment Errors	64	*Optional*
Receive Overrun Errors	64	*Optional*
Receive CRC Errors	64	*Optional*
Collisions	64	*Optional*
Duration (seconds)	32	*Required*
Duration (nanoseconds)	32	*Optional*
Per Queue		
Transmit Packets	64	*Required*
Transmit Bytes	64	*Optional*
Transmit Overrun Errors	64	*Optional*
Duration (seconds)	32	*Required*
Duration (nanoseconds)	32	*Optional*
Per Group		
Reference Count (flow entries)	32	*Optional*
Packet Count	64	*Optional*
Byte Count	64	*Optional*
Duration (seconds)	32	*Required*
Duration (nanoseconds)	32	*Optional*
Per Group Bucket		
Packet Count	64	*Optional*
Byte Count	64	*Optional*

Per Meter		
Flow Count	32	*Optional*
Input Packet Count	64	*Optional*
Input Byte Count	64	*Optional*
Duration (seconds)	32	*Required*
Duration (nanoseconds)	32	*Optional*
Per Meter Band		
In Band Packet Count	64	*Optional*
In Band Byte Count	64	*Optional*

5.9 指令 Instructions

Optional Instruction: Meter meter id：将包转给指定的计量器。计量的结果可能会丢弃这个数据包（依赖于计量器的配置和状态）

Optional Instruction: Apply-Actions action(s)：立即执行指定的行动，而不改变行动集。在两个表之间传递或者执行同类型的多个行动的时候， 这个指令可用来修改数据包。这些行动被指定为一个行动列表（见 5.11）。

Optional Instruction: Clear-Actions：立即清除行动集中的所有行动。

Required Instruction: Write-Actions action(s)：将指定的行动添加到当前的行动集中。如果行动存在于当前集合中，则进行覆盖，否则进行追加。

Optional Instruction: Write-Metadata metadata / mask：在元数据字段写入掩码的元数据数值。掩码指的是元数据寄存器应进行修改的比特。 new_metadata=old_metadata&~mask

|value&mask）。

Required Instruction: Goto-Table next-table-id：指示流水线处理的下一张表。表 ID 必须大于当前表 ID。流水线最后一张表的流表项不能含有这个指令（见 5.1 ）。Openflow 交换机若只有一个流表则不需要实现这个指令。

5.10 Action

- Action Set：跟每一个数据包绑定的，默认为空，一起走过流水线的每一步，受 Instruction(如 Write-Action、Clear-Action)的修改，直到 Goto_table 指令最后统一执行到数据包上的。Action Set 中，每种类型只能有一个，可以多个 set-field，但每种 set-field 只能有一个。
  1. copy TTL inwards: 向数据包内复制 TTL 的行动
  2. pop: 从数据包弹出所有标记的行动
  3. push-MPLS: 向数据包压入 MPLS 标记的行动
  4. push-PBB: 向数据包压入 PBB 标记的行动
  5. push-VLAN: 向数据包压入 VLAN 标记的行动
  6. copy TTL outwards: 向数据包外复制 TTL 的行动
  7. decrement TTL: 将数据包的 TTL 字段减 1
  8. set: 数据包使用所有的 set_field 行动
  9. qos: 使用所有的 QOS 行动，如对数据包排队
  10. group: 如果指定了组行动，那么按顺序执行组行动存储段里的行动。
  11. output: 如果没有指定组行动，数据包就会按照 output 行动中指定的端口转发。
- Action List：Apply-Actions 指令和 Packet-out 消息包含一个行动列表。行动列表的含义与 Openflow1.0 规范的相同。行动列表中的行动按照列表中的次序执行，并立即作用到数据包。一个 Apply_Actions 指令执行完一个行动列表后，流水线继续处理已修改的数据包。数据包的行动集本身在行动列表执行的时候没有改变。

1. Required Action: Output. 数据包输出到指定 Openflow 端口。

2. Optional Action: Set-Queue. 设置数据包的队列 ID。当数据包使用输出行动转发到一个端口，队列 ID 决定数据包安排到端口所属的哪个队列并转发。转发行为受队列配置控制，并用来提供 Qos 支持（见 7.2.2 ）。

3. Required Action: Drop. 没有明确的行动来表现丢弃。相反，那些行动集中没有输出行动的数据包应该被丢弃。当流水线处理时或执行 Clear_Actions 指令后，空指令集或空指令行动存储段会导致丢弃这个结果。

4. Required Action: Group. 通过指定的组处理数据包，准确的解释依靠组类型。

5. Optional Action: Push-Tag/Pop-Tag. 交换机可具有压入/ 弹出表 6 所示标记的能力。为了和已有网络更好结合，建议支持压入/ 弹出 VLAN 标记的能力。最新的压入标记应插入到最外侧有效位置作为最外侧的标记。当压入一个新 VLAN 标记，应作为最外侧标记来插入，位于以太头部后面，其它标记前面。同样的，当压入一个新 MPLS 标记，也应作为最外侧标记来插入，位于以太头部后面，其它标记前面。当多个压入行动添加到数据包行动集，按照行动集定义的规则依次作用到数据包，开始时 MPLS，接着是 PBB，后面是 VLAN（见 5.10 ）。当一个行动列表中有多个压入行动，按照列表次序（见 5.11 ）作用到数据包。

注意： 5.12 节所涉及的信息都是默认字段值。

6 OpenFlow1.3 版本协议新增消息

1) **Controller-to-Switch**
   Modify-State：添加、删除和修改的流表项、组项，设置交换机端口优先级；
   Read-State：收集来自交换机的各种信息，如当前配置、统计和容量；
   Role-Request:设置 role;
   Asynchronous-Conguration:设置 Asynchronous 消息的额外过滤器
2) **Asynchronous**
   Error
3) **Symmetric**
   Experimenter:在 OpenFlow 消息的 type space 中提供额外功能的标准方式

```
enum ofp_flow_mod_command {
    OFPFC_ADD          = 0, /* New flow. */
    OFPFC_MODIFY       = 1, /* Modify all matching flows. */
    OFPFC_MODIFY_STRICT = 2, /* Modify entry strictly matching wildcards and
                                priority. */
    OFPFC_DELETE       = 3, /* Delete all matching flows. */
    OFPFC_DELETE_STRICT = 4, /* Delete entry strictly matching wildcards and
                                priority. */
};
```

```
/* Meter commands */
enum ofp_meter_mod_command {
    OFPMC_ADD,              /* New meter. */
    OFPMC_MODIFY,           /* Modify specified meter. */
    OFPMC_DELETE,           /* Delete specified meter. */
};
```

7 OpenFlow Protocol

7.1 OpenFlow Header

ofp_header

version：OpenFlow 协议版本，0x04
length：消息总长度
type：

```
enum ofp_type {
    /* Immutable messages. */
    OFPT_HELLO              = 0,  /* Symmetric message */
    OFPT_ERROR              = 1,  /* Symmetric message */
    OFPT_ECHO_REQUEST       = 2,  /* Symmetric message */
    OFPT_ECHO_REPLY         = 3,  /* Symmetric message */
    OFPT_EXPERIMENTER       = 4,  /* Symmetric message */

    /* Switch configuration messages. */
    OFPT_FEATURES_REQUEST   = 5,  /* Controller/switch message */
    OFPT_FEATURES_REPLY     = 6,  /* Controller/switch message */
    OFPT_GET_CONFIG_REQUEST = 7,  /* Controller/switch message */
    OFPT_GET_CONFIG_REPLY   = 8,  /* Controller/switch message */
    OFPT_SET_CONFIG         = 9,  /* Controller/switch message */

    /* Asynchronous messages. */
    OFPT_PACKET_IN          = 10, /* Async message */
    OFPT_FLOW_REMOVED       = 11, /* Async message */


    OFPT_PORT_STATUS        = 12, /* Async message */

    /* Controller command messages. */
    OFPT_PACKET_OUT         = 13, /* Controller/switch message */
    OFPT_FLOW_MOD           = 14, /* Controller/switch message */
    OFPT_GROUP_MOD          = 15, /* Controller/switch message */
    OFPT_PORT_MOD           = 16, /* Controller/switch message */
    OFPT_TABLE_MOD          = 17, /* Controller/switch message */

    /* Multipart messages. */
    OFPT_MULTIPART_REQUEST      = 18, /* Controller/switch message */
    OFPT_MULTIPART_REPLY        = 19, /* Controller/switch message */

    /* Barrier messages. */
    OFPT_BARRIER_REQUEST    = 20, /* Controller/switch message */
    OFPT_BARRIER_REPLY      = 21, /* Controller/switch message */

    /* Queue Configuration messages. */
    OFPT_QUEUE_GET_CONFIG_REQUEST = 22,  /* Controller/switch message */
    OFPT_QUEUE_GET_CONFIG_REPLY   = 23,  /* Controller/switch message */

    /* Controller role change request messages. */
    OFPT_ROLE_REQUEST       = 24, /* Controller/switch message */
    OFPT_ROLE_REPLY         = 25, /* Controller/switch message */

    /* Asynchronous message configuration. */
    OFPT_GET_ASYNC_REQUEST  = 26, /* Controller/switch message */
    OFPT_GET_ASYNC_REPLY    = 27, /* Controller/switch message */
    OFPT_SET_ASYNC          = 28, /* Controller/switch message */

    /* Meters and rate limiters configuration messages. */
    OFPT_METER_MOD          = 29, /* Controller/switch message */
};
```

7.2 Common Structures

7.2.1 端口

物理端口：交换机定义的端口，对应于一个交换机的硬件接口

逻辑端口：交换机定义的端口，并不直接对应一个交换机的硬件接口。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有一个叫做隧道 ID 的额外的元数据字段与它相关联（见 7.2.3.7 ）；而当一个逻辑端口上接收到的分组被发送到控制器时，其逻辑端口和底层的物理端口都要报告给控制器（见 7.4.1 ）

保留端口：由本规范定义。它们指定通用的转发动作，如发送到控制器，泛洪，或使用非 OpenFlow 的方法转发，如“正常”交换机处理。

只能作为输出端口的保留端口：ALL、IN_PORT、NORMAL、FLOOD

既能作为入端口又能作为输出端口：CONTROLLER、LOCAL

既不能作为入端口又不能作为输出端口：ANY

Required: ALL: 表示交换机可转发指定数据包到所有端口，它仅可用作输出端口。在这种情况下，数据包被复制后发送到所有的标准端口，不包括数据包的入端口和端口被配置为OFPPC_NO_F。WD

Required: CONTROLLER:表示 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，数据包封装进输入包消息，并使用 OpenFlow 协议发送。当用作一个入口端口，确认数据包来自控制器。

Required: TABLE: 表示 openflow 流水线的开始（见 5.1 ）。这个端口仅在输出包消息的行动列表里的输出行为时候有效（见 7.3.7 ），此时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。

Required: IN PORT:代表数据包的进入端口。当数据包通过它的入端口发送出去的话，只能用作输出端口。

Required: ANY: 特别值，用在未指定端口的 OpenFlow 命令（端口通配符） 。既不能作为入口端口，也不能作为一个输出端口。

Optional: LOCAL: 表示交换机的本地网络堆栈和管理堆栈。可以用作一个入口端口或作为一个输出端口。远程实体通过本地端口与交换机和网络服务互通，而不是通过一个独立的控制网络。利用一组合适的默认流表项，本地端口被用来实现一个带内控制器连接。

Optional: NORMAL:代表传统的非 OpenFlow 流水线（见 5.1 ）。仅可用于为一个输出端口，使用普通的流水线处理数据包。如果交换机不能转发数据包从 OpenFlow 流水线到普通流水线，它必须表明它不支持这一行动。

Optional: FLOOD: 表示使用普通流水线处理进行泛洪（见 5.1 ）。只作为一个输出端口，一般可以将数据包发往所有标准端口， 但不能发往入端口或 OFPPS_BLOCKE 状 D 态的端口。交换机也可以通过数据包的 VLAN ID 选择哪些端口泛洪。

OpenFlow-only 交换机不支持 NORMAL 端口和 FLOOD 端口，而 OpenFlow-hybrid 交换机均支持上述端口（见 5.1 ）。转发数据包到 FLOOD 端口依赖交换机的实现和配置， 若使用一组 all 类型进行转发，则可以使控制器能更灵活地实现泛洪（见 5.6.1 ）

ofp_port

port_no：交换机端口号

```
/* Port numbering. Ports are numbered starting from 1. */
enum ofp_port_no {
    /* Maximum number of physical and logical switch ports. */
    OFPP_MAX        = 0xffffff00,

    /* Reserved OpenFlow Port (fake output "ports"). */
    OFPP_IN_PORT    = 0xfffffff8,  /* Send the packet out the input port.  This
                                      reserved port must be explicitly used
                                      in order to send back out of the input
                                      port. */
    OFPP_TABLE      = 0xfffffff9,  /* Submit the packet to the first flow table
                                      NB: This destination port can only be
                                      used in packet-out messages. */
    OFPP_NORMAL     = 0xfffffffa,  /* Process with normal L2/L3 switching. */
    OFPP_FLOOD      = 0xfffffffb,  /* All physical ports in VLAN, except input
                                      port and those blocked or link down. */
    OFPP_ALL        = 0xfffffffc,  /* All physical ports except input port. */
    OFPP_CONTROLLER = 0xfffffffd,  /* Send to controller. */
    OFPP_LOCAL      = 0xfffffffe,  /* Local openflow "port". */
    OFPP_ANY        = 0xffffffff   /* Wildcard port used only for flow mod
                                      (delete) and flow stats requests. Selects
```

hw_addr：端口 MAC 地址

其中，OFP_ETH_ALEN=6；OFP_MAX_PORT_NAME_LEN=16

config：端口管理设置；

```
/* Flags to indicate behavior of the physical port.  These flags are
 * used in ofp_port to describe the current configuration.  They are
 * used in the ofp_port_mod message to configure the port's behavior.
 */
enum ofp_port_config {
    OFPPC_PORT_DOWN    = 1 << 0,  /* Port is administratively down. */

    OFPPC_NO_RECV      = 1 << 2,  /* Drop all packets received by port. */
    OFPPC_NO_FWD       = 1 << 5,  /* Drop packets forwarded to port. */
    OFPPC_NO_PACKET_IN = 1 << 6   /* Do not send packet-in msgs for port. */
};
```

state：端口内部状态。所有端口状态位都是只读的，不能被控制器改变。当端口标志改变时，交换机发送 OFPT_PORT_STATUS 消息通知控制器。

```
/* Current state of the physical port.  These are not configurable from
 * the controller.
 */
enum ofp_port_state {
    OFPPS_LINK_DOWN    = 1 << 0,  /* No physical link present. */
    OFPPS_BLOCKED      = 1 << 1,  /* Port is blocked */
    OFPPS_LIVE         = 1 << 2,  /* Live for Fast Failover Group. */
};
```

curr：当前连接方式（speed and duplexity；10M 到 10G，全双工、半双工）、类型（铜/光纤）、特征（自动协商和暂停）。

advertised：广播给对方的连接方式、类型、特征

supported：支持的链路方式、类型、特征

peer：对方的连接方式、类型、特征

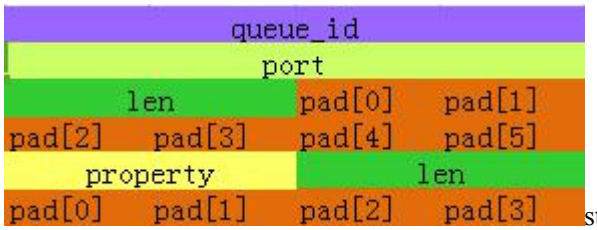
```
/* Features of ports available in a datapath. */
enum ofp_port_features {
    OFPPF_10MB_HD    = 1 << 0,  /* 10 Mb half-duplex rate support. */
    OFPPF_10MB_FD    = 1 << 1,  /* 10 Mb full-duplex rate support. */
    OFPPF_100MB_HD   = 1 << 2,  /* 100 Mb half-duplex rate support. */
    OFPPF_100MB_FD   = 1 << 3,  /* 100 Mb full-duplex rate support. */
    OFPPF_1GB_HD     = 1 << 4,  /* 1 Gb half-duplex rate support. */
    OFPPF_1GB_FD     = 1 << 5,  /* 1 Gb full-duplex rate support. */
    OFPPF_10GB_FD    = 1 << 6,  /* 10 Gb full-duplex rate support. */
    OFPPF_40GB_FD    = 1 << 7,  /* 40 Gb full-duplex rate support. */
    OFPPF_100GB_FD   = 1 << 8,  /* 100 Gb full-duplex rate support. */
    OFPPF_1TB_FD     = 1 << 9,  /* 1 Tb full-duplex rate support. */
    OFPPF_OTHER      = 1 << 10, /* Other rate, not in the list. */

    OFPPF_COPPER     = 1 << 11, /* Copper medium. */
    OFPPF_FIBER      = 1 << 12, /* Fiber medium. */
    OFPPF_AUTONEG    = 1 << 13, /* Auto-negotiation. */
    OFPPF_PAUSE      = 1 << 14, /* Pause. */
    OFPPF_PAUSE_ASYM = 1 << 15  /* Asymmetric pause. */
};
```

curr_speed：当前速率，kbps
Max_speed：最大速率，kbps

7.2.2 队列

一个 openflow 交换机通过简单的排队机制提供有限的 QoS 服务。一个（或多个）队列可以连接到端口，用来与流表项映射。流表项映射的某个队列，就根据这个队列的配置处理。

ofp_packet_queue



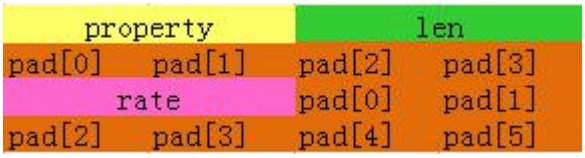
struct ofp_queue_prop_header properties[0];

port：队列所属端口

properties：

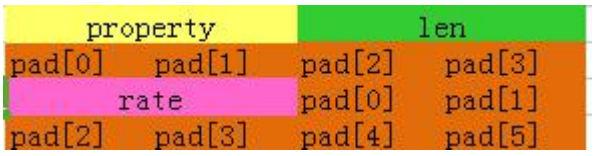
```
enum ofp_queue_properties {
    OFPQT_MIN_RATE      = 1,      /* Minimum datarate guaranteed. */
    OFPQT_MAX_RATE      = 2,      /* Maximum datarate. */
    OFPQT_EXPERIMENTER  = 0xffff  /* Experimenter defined property. */
};
```

- ofp_queue_prop_min_rate



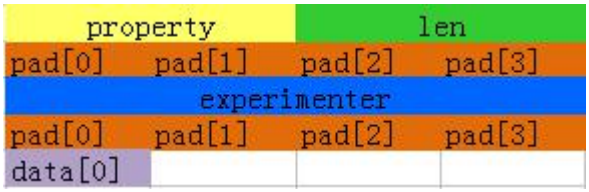
Prop：OFPQT_MIN,
len：16
rate：以 0.1%为单位; 禁止大于 1000。默认 0xfff

- ofp_queue_prop_max_rate



Prop：OFPQT_MIN,
len：16
rate：以 0.1%为单位; 禁止大于 1000。默认 0fff

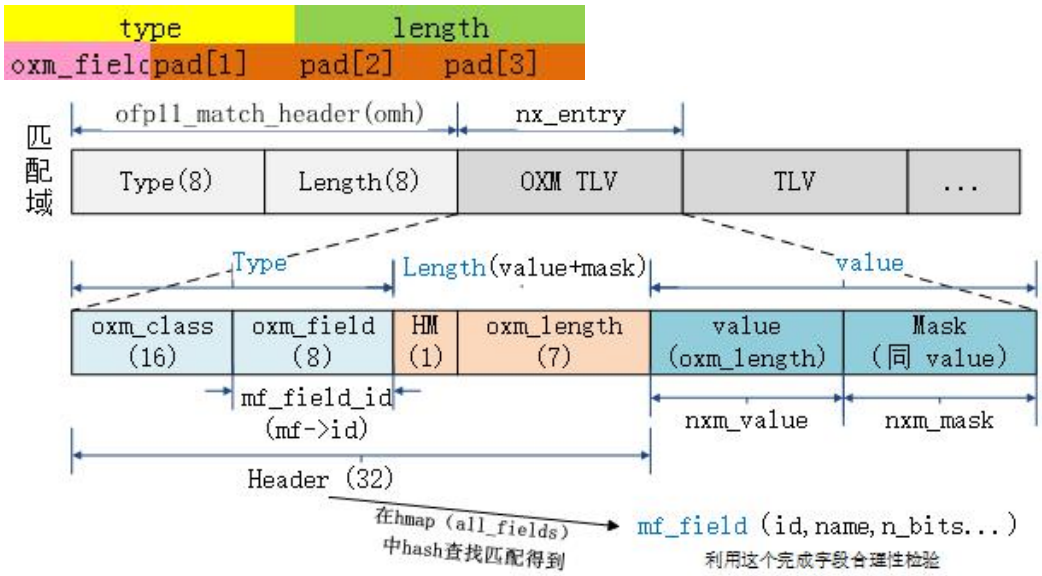
- ofp_queue_prop_experimenter



Experimenter ID 与 ofp_experimenter_header 结构体形式相同

7.2.3 匹配域

struct ofp_match

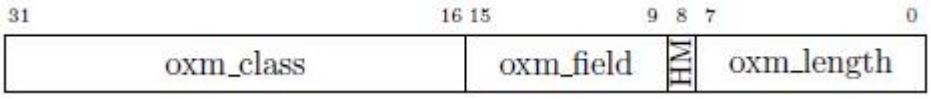


type：OFPMT_OXM

```
OFPMT_STANDARD = 0,        /* Deprecated. */
OFPMT_OXM      = 1,        /* OpenFlow Extensible Match */
```

OXMTLV 没有相同的长度， 也没有采取填充来对齐

OXM：



oxm_class：

区别 match 类型。匹配类型包括 ONF member classes and ONF reserved classes 两种。

高位比特为 1 是 ONF 保留类，供 openflow 规范本身使用。

高位比特为 0 的是 ONF 成员类，必要时由 ONF 分配，标识一个 ONF 成员，并且可任意使用

```
/* OXM Class IDs.
 * The high order bit differentiate reserved classes from member classes.
 * Classes 0x0000 to 0x7FFF are member classes, allocated by ONF.
 * Classes 0x8000 to 0xFFFE are reserved classes, reserved for standardisation.
 */
enum ofp_oxm_class {
    OFPXMC_NXM_0          = 0x0000,    /* Backward compatibility with NXM */
    OFPXMC_NXM_1          = 0x0001,    /* Backward compatibility with NXM */
    OFPXMC_OPENFLOW_BASIC = 0x8000,    /* Basic class for OpenFlow */
    OFPXMC_EXPERIMENTER   = 0xFFFF,    /* Experimenter class */
};
```

*NXM：Nicira Extensible Match

oxm_field：

```
/* 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 */
};
```

```
enum ofp_vlan_id {
OFPVID_PRESENT = 0x1000,
OFPVID_NONE = 0x0000,
};
```

其中

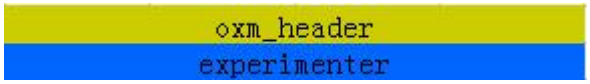
```
/* Bit definitions for IPv6 Extension Header pseudo-field. */
enum ofp_ipv6exthdr_flags {
    OFPIEH_NONEXT = 1 << 0,     /* "No next header" encountered. */
    OFPIEH_ESP    = 1 << 1,     /* Encrypted Sec Payload header present. */
    OFPIEH_AUTH   = 1 << 2,     /* Authentication header present. */
    OFPIEH_DEST   = 1 << 3,     /* 1 or 2 dest headers present. */
    OFPIEH_FRAG   = 1 << 4,     /* Fragment header present. */
    OFPIEH_ROUTER = 1 << 5,     /* Router header present. */
    OFPIEH_HOP    = 1 << 6,     /* Hop-by-hop header present. */
    OFPIEH_UNREP  = 1 << 7,     /* Unexpected repeats encountered. */
    OFPIEH_UNSEQ  = 1 << 8,     /* Unexpected sequencing encountered. */
};
```

Oxm_hasmask：

1) 当 oxm_hasmask 为 0 时，OXM_TLV 只匹配相关 field 等于 oxm_value 的数据包
2) 当 oxm_hasmask 为 1 时，只对比 oxm_mask 值为 1 的比特位与 oxm_value 的比特位进行比较

Oxm_length：

ofp_oxm_experimenter_header



7.2.4 Flow Instruction Structures

ofp_instruction

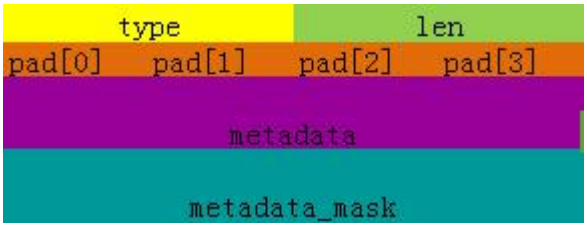

```
enum ofp_instruction_type {
    OFPIT_GOTO_TABLE = 1,       /* Setup the next table in the lookup
                                   pipeline */
    OFPIT_WRITE_METADATA = 2,   /* Setup the metadata field for use later in
                                   pipeline */
    OFPIT_WRITE_ACTIONS = 3,    /* Write the action(s) onto the datapath action
                                   set */
    OFPIT_APPLY_ACTIONS = 4,    /* Applies the action(s) immediately */
    OFPIT_CLEAR_ACTIONS = 5,    /* Clears all actions from the datapath
                                   action set */
    OFPIT_METER = 6,            /* Apply meter (rate limiter) */

   OFPIT_EXPERIMENTER = 0xFFFF  /* Experimenter instruction */
};
```

ofp_instruction_goto_table

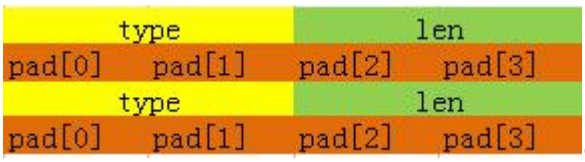
type：OFPIT_GOTO_TABLE

table_id：设置数据包进程中下一流表 ID

ofp_instruction_write_metadata



ofp_instruction_actions



ofp_action_header actions[0];

ofp_instruction_meter

ofp_instruction_experimenter

experimenter:experimenter ID

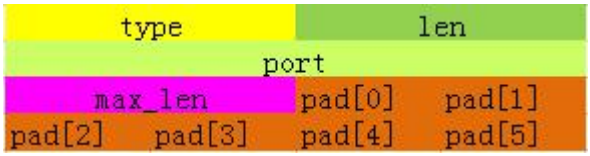
7.2.5 Action Structures

ofp_action_header

Type:

```
enum ofp_action_type {
        OFPAT_OUTPUT       = 0,  /* Output to switch port. */
        OFPAT_COPY_TTL_OUT = 11, /* Copy TTL "outwards" -- from next-to-outermost
                                    to outermost */
        OFPAT_COPY_TTL_IN  = 12, /* Copy TTL "inwards" -- from outermost to
                                    next-to-outermost */
        OFPAT_SET_MPLS_TTL = 15, /* MPLS TTL */
        OFPAT_DEC_MPLS_TTL = 16, /* Decrement MPLS TTL */

        OFPAT_PUSH_VLAN    = 17, /* Push a new VLAN tag */
        OFPAT_POP_VLAN     = 18, /* Pop the outer VLAN tag */
        OFPAT_PUSH_MPLS    = 19, /* Push a new MPLS tag */
        OFPAT_POP_MPLS     = 20, /* Pop the outer MPLS tag */
        OFPAT_SET_QUEUE    = 21, /* Set queue id when outputting to a port */
        OFPAT_GROUP        = 22, /* Apply group. */
        OFPAT_SET_NW_TTL   = 23, /* IP TTL. */
        OFPAT_DEC_NW_TTL   = 24, /* Decrement IP TTL. */
        OFPAT_SET_FIELD    = 25, /* Set a header field using OXM TLV format. */
        OFPAT_PUSH_PBB     = 26, /* Push a new PBB service tag (I-TAG) */
        OFPAT_POP_PBB      = 27, /* Pop the outer PBB service tag (I-TAG) */
        OFPAT_EXPERIMENTER = 0xffff
    };
```

ofp_action_output



type：OFPAT_OUTPUT

port：数据包发送到的目的端口

max_len：发送到控制器的最大长度

```
enum ofp_controller_max_len {
OFPCML_MAX       = 0xffe5, /* maximum max_len value which can be used
                              to request a specific byte length. */
OFPCML_NO_BUFFER = 0xffff  /* indicates that no buffering should be
                              applied and the whole packet is to be
                              sent to the controller. */
};
```

ofp_action_group

Type:OFPAT_GROUP.
group_id：处理数据包使用的组

ofp_action_set_queue

type：OFPAT_SET_QUEUE

ofp_action_mpls_ttl

ofp_action_push

ofp_action_pop_mpls

MPLS 负载的以太网

ofp_action_set_field

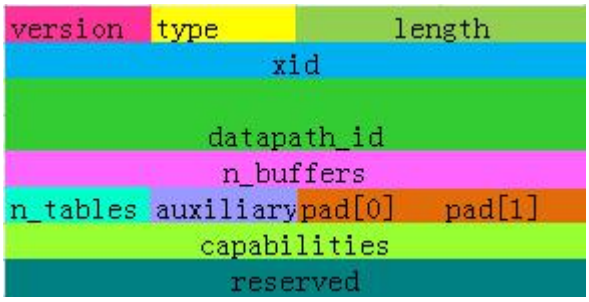
ofp_action_experimenter_header

7.3 Controller-to-Switch Messages

7.3.1 Handshake

控制器应该发送一个 OFPT_FEATURES_REQUEST 消息。这个消息只包含 OpenFlow 头部。
交换机必须发送一个 OFPT_FEATURES_REPLY 响应消息

ofp_switch_features



datapath_id：数据路的唯一标识。

自定义(48~63)　　交换机 MAC 地址(0~47)

n_buffers：一次缓存最多数据包的数量

n_table：数据链路支持的流表数，每一种表都可以对支持的匹配字段，行动和表项数量有不同的设置。

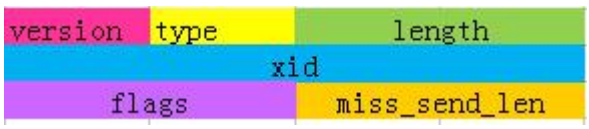
auxiliary_id：识别交换机与控制器的连接类型。主连接这个字段的设置为零，辅助连接这个字段设置为非零值。

capabilities：

```
/* Capabilities supported by the datapath. */
enum ofp_capabilities {
    OFPC_FLOW_STATS     = 1 << 0,  /* Flow statistics. */
    OFPC_TABLE_STATS    = 1 << 1,  /* Table statistics. */
    OFPC_PORT_STATS     = 1 << 2,  /* Port statistics. */
    OFPC_GROUP_STATS    = 1 << 3,  /* Group statistics. */
    OFPC_IP_REASM       = 1 << 5,  /* Can reassemble IP fragments. */
    OFPC_QUEUE_STATS    = 1 << 6,  /* Queue statistics. */
    OFPC_PORT_BLOCKED   = 1 << 8   /* Switch will block looping ports. */
};
```

7.3.2 交换机配置

ofp_switch_config



flags：

```
enum ofp_config_flags {
    /* Handling of IP fragments. */
    OFPC_FRAG_NORMAL   = 0,       /* No special handling for fragments. */
    OFPC_FRAG_DROP     = 1 << 0,  /* Drop fragments. */
    OFPC_FRAG_REASM    = 1 << 1,  /* Reassemble (only if OFPC_IP_REASM set). */
    OFPC_FRAG_MASK     = 3,
};
```

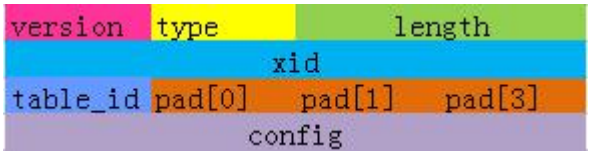
miss_send_len：数据通路发送交控制器的数据包的最大字节数，ofp_controller_max_len 指明有效值

miss_send_len=0 时，交换机发送零字节的 packet_in 消息；

miss_send_len= OFPCML_NO_BUFFER 时，消息中完全是数据包，没有缓存。

7.3.3 流表配置

ofp_table_mod



table_id：流表 ID

```
/* Table numbering. Tables can use any number up to OFPT_MAX. */
enum ofp_table {
    /* Last usable table number. */
    OFPTT_MAX        = 0xfe,

    /* Fake tables. */
    OFPTT_ALL        = 0xff   /* Wildcard table used for table config,
                                 flow stats and flow deletes. */
};
```

OFPTT_ALL 代表所有流表

config：

```
/* Flags to configure the table. Reserved for future use. */
enum ofp_table_config {
    OFPTC_DEPRECATED_MASK       = 3,  /* Deprecated bits */
};
```

7.3.4 Modify State Messages

ofp_flow_mod

Cookie：用于流删除消息、流表统计、以及流表的过滤、修改和删除

OFPFC_ADD 消息中，*cookie* 设为规定值

OFPFC_MODIFY 和 OFPFC_MODIFY_STRICT 消息中，cookie 值不变

如果 cookie_mask 字段不为零，当修改或删除流表项时，和 cookie 字段一起限制流匹配。OFPFC_ADD 消息忽略这个字段。

table_id：识别流表。0 代表 pineline 中的第一个流表，OFPTT_ALL 是唯一用于删除请求的声明符。

command：负责流表的添加、修改、删除

```
enum ofp_flow_mod_command {
    OFPFC_ADD           = 0, /* New flow. */
    OFPFC_MODIFY        = 1, /* Modify all matching flows. */
    OFPFC_MODIFY_STRICT = 2, /* Modify entry strictly matching wildcards and
                                priority. */
    OFPFC_DELETE        = 3, /* Delete all matching flows. */
    OFPFC_DELETE_STRICT = 4, /* Delete entry strictly matching wildcards and
                                priority. */
};
```

修改和删除流 flow-mod 命令有非严格的版本(OFPFC_MODIFYand OFPFC_DELETE)和严格的版本(OFPFC_MODIFY_STRICTor OFPFC_DELETE_STRICT)。在严格的版本中，匹配字段集,所有匹配字段,包括他们的掩码、优先级，是和表项严格匹配的,只有一个相同的流表项被修改或删除。比如，发送一个移除表项的信息，如果没有字段能够匹配，则 OFPFC_DELETE 命令就要从表中删除所有的流表项，而 OFPFC_DELETE_STRICT 命令只会删除一个应用在指定优先级数据包上的流表项。

关于非严格修改和删除命令，与流模式描述相匹配的所有流表项都会被修改和删除。在非严格版本，当流表项正好匹配或者比流模式命令描述的更多，一个匹配就会产生。在流模式中失配字段变为通配的，字段掩码是有效的，比如优先级等其他的流模式字段则被忽略。比如，如果一个 OFPFC_DELETE 命令去删除目标端口为 80 的所有流表项，那么通配所有匹配字段的流表项将不会被删除。然而，通配所有字段的一个 OFPFC_DELETE 命令将会删除一个匹配端口 80 的表项。同样的解释混合通配符和精确匹配字段也适用于单个和汇聚流数据请求。目标组或输出端口可以有选择地过滤删除命令。如果输出端口字段包含一个值除了 OFPP_ANY。匹配时它引入一个约束， 这个约束是 , 每个匹配流表项必须包含一个针对指定端口的输出行动。这个约束只对直接与流表项关联的行动进行限制。换句话说, 交换机不能通过点到组的行动集递归 , 这可能已经匹配输出操作。 out_group, 如果不同于 OFPG_ANY, 对组引入了类似的限制行动。这些字段被 OFPFC_ADD,OFPFC_MODI 和 FYOFPFC_MODIFY_STRICT 消息忽略

Idle_timeout 和 hard_timeout 字段控制流表项过期的速度。当一个流表项修改(OFPFC_ MODIFY 或 OFPFC_MODIFY_STRICT 消息),idle_timeout 和 hard_timeout 字段被忽略。

Priority 表示指定的流表中表的优先级。仅用于 OFPFC_ADD 消息匹配和添加流表项时,和当 OFPFC_MODIFY_STRICT 或 OFPFC_DELETE_STRICT 消息匹配流条目时。

buffer_id 指向交换机缓冲的数据包并用 packet-in 消息发送给控制器。

flags：

```
enum ofp_flow_mod_flags {
    OFPFF_SEND_FLOW_REM = 1 << 0,  /* Send flow removed message when flow
                                    * expires or is deleted. */
    OFPFF_CHECK_OVERLAP = 1 << 1,  /* Check for overlapping entries first. */
    OFPFF_RESET_COUNTS  = 1 << 2,  /* Reset flow packet and byte counts. */
    OFPFF_NO_PKT_COUNTS = 1 << 3,  /* Don't keep track of packet count. */
    OFPFF_NO_BYT_COUNTS = 1 << 4,  /* Don't keep track of byte count. */
};
```

OFPFF_SEND_FLOW_REM：当流表项过期或删除时交换机必须发送一个流删除消息。

OFPFF_CHECK_OVERLAP：交换机在向流表中插入前，必须检查没有相同优先级的表项冲突。如果有的话,flow mod 失败并返回一个错误信息。

OFPFF_NO_PKT_COUNTS：交换机不需要监测流的数据包计数 。

OFPFF_NO_BYT_COUNTS：交换机不需要监测流的字节计数。

ofp_group_mod

conmmand：

```
/* Group commands */
enum ofp_group_mod_command {
    OFPGC_ADD    = 0,       /* New group. */
    OFPGC_MODIFY = 1,       /* Modify all matching groups. */
    OFPGC_DELETE = 2,       /* Delete all matching groups. */
};
```

type：

```
/* Group types.  Values in the range [128, 255] are reserved for experimental
 * use. */
enum ofp_group_type {
    OFPGT_ALL      = 0, /* All (multicast/broadcast) group.  */
    OFPGT_SELECT   = 1, /* Select group. */
    OFPGT_INDIRECT = 2, /* Indirect group. */
    OFPGT_FF       = 3, /* Fast failover group. */
};
```

group_id：

```
/* Group numbering. Groups can use any number up to OFPG_MAX. */
enum ofp_group {
    /* Last usable group number. */
    OFPG_MAX        = 0xffffff00,

    /* Fake groups. */
    OFPG_ALL        = 0xfffffffc,  /* Represents all groups for group delete
                                      commands. */
    OFPG_ANY        = 0xffffffff   /* Wildcard group used only for flow stats
                                      requests. Selects all flows regardless of
                                      group (including flows with no group).
                                      */
};
```

bucket：一个存储段的数组。对间接组，数组必须包含一个存储段,其它组类型在数组里可能有多个存储段。对快速故障转移组，存储段次序就定义了其优先级,通过修改组可以改变存储段的排序(例如使用带 OFPGC_MODIFY 命令的 OFPT_GROUP_MOD 消息)。

ofp_bucket：

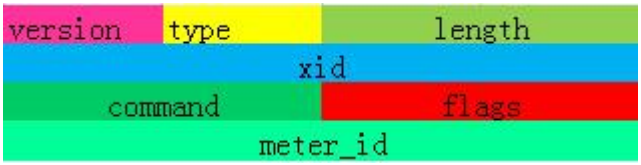
weight：存储段的有关重量，为选择的组定义

ofp_port_mod

mask：config 域的掩码，用于选择比特位

advertise：没有掩码

ofp_meter_mod



Command：One of OFPMC_*.

```
/* Meter commands */
enum ofp_meter_mod_command {
    OFPMC_ADD,              /* New meter. */
    OFPMC_MODIFY,           /* Modify specified meter. */
    OFPMC_DELETE,           /* Delete specified meter. */
};
```

Flags: Bitmap of OFPMF_* flags.

```
/* Meter configuration flags */
enum ofp_meter_flags {
    OFPMF_KBPS    = 1 << 0,     /* Rate value in kb/s (kilo-bit per second). */
    OFPMF_PKTPS   = 1 << 1,     /* Rate value in packet/sec. */
    OFPMF_BURST   = 1 << 2,     /* Do burst size. */
    OFPMF_STATS   = 1 << 3,     /* Collect statistics. */
};
```

BURST:处理突发大小

STATS:收集状态

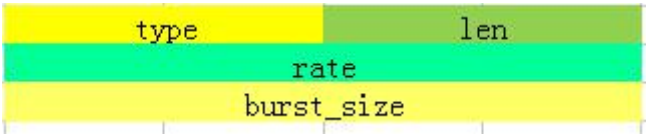
Meter_id:带交换机的应用表

```
/* Meter numbering. Flow meters can use any number up to OFPM_MAX. */
enum ofp_meter {
    /* Last usable meter. */
    OFPM_MAX        = 0xffff0000,

    /* Virtual meters. */
    OFPM_SLOWPATH   = 0xfffffffd,  /* Meter for slow datapath. */
    OFPM_CONTROLLER = 0xfffffffe,  /* Meter for controller connection. */
    OFPM_ALL        = 0xffffffff,  /* Represents all meters for stat requests
                                      commands. */
};
```

CONTROLLER:通过 Packet_in 控制发送到控制器的数据包，保留端口，限制发送到控制器的流量。

SLOWPATH: 交换机的缓慢数据通路

ofp_mater_band_header



rate:带宽速率，单位 kilobit per seconds。当 flag 域值为 OFPMF_PKTPS 时，rate 为每秒发送的数据包

burst_size 用于 flag 域为 OFPMF_BURST，数据包和字节的突发长度值，单位为 kilobits

type:

```
/* Meter band types */
enum ofp_meter_band_type {
    OFPMBT_DROP           = 1,      /* Drop packet. */
    OFPMBT_DSCP_REMARK    = 2,      /* Remark DSCP in the IP header. */
    OFPMBT_EXPERIMENTER   = 0xFFFF  /* Experimenter meter band. */
};
```

DROP:速率限制器，当超过带宽时进行丢包

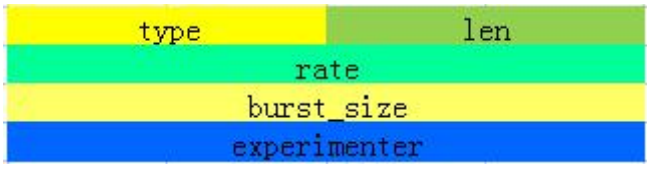
ofp_meter_band_drop

丢弃超过带宽速率的数据包

ofp_meter_band_dscp_remark

prec_level:drop precedence of the packet

ofp_meter_band_experimenter

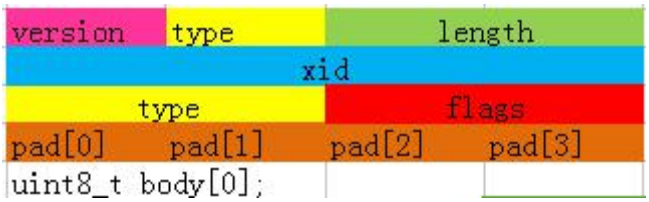


7.3.5 Multipart Messages

request statistics or state information from the switch

多条 OpenFlow 消息进行一系列转发，接收机进行组装

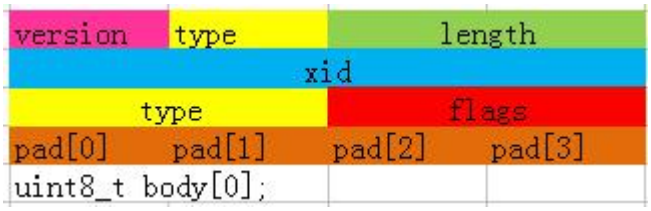
ofp_multipart_request



flags：

```
enum ofp_multipart_request_flags {
    OFPMPF_REQ_MORE  = 1 << 0  /* More requests to follow. */
};
```

ofp_multipart_reply




```
enum ofp_multipart_reply_flags {
    OFPMPF_REPLY_MORE  = 1 << 0  /* More replies to follow. */
};
```

type:

名称	值	含义	Request body	Reply body
OFPMP_DESC	0	OpenFlow 交换机描述	empty	struct ofp_desc.
OFPMP_FLOW	1	独立流表状态	struct ofp_flow_stats_request.	struct ofp_flow_stats.
OFPMP_AGGREGATE	2	邻接流表状态	struct ofp_aggregate_stats_request.	struct ofp_aggregate_stats_reply.
OFPMP_TABLE	3	流表状态	empty	struct ofp_table_stats.
OFPMP_PORT_STATS	4	端口状态	struct ofp_port_stats_request	struct ofp_port_stats
OFPMP_QUEUE	5	端口队列状态	struct ofp_queue_stats_request.	struct ofp_queue_stats
OFPMP_GROUP	6	Group counter statistics	struct ofp_group_stats_request	struct ofp_group_stats
OFPMP_GROUP_DESC	7	Group description.	empty	struct ofp_group_desc.
OFPMP_GROUP_FEATURES	8	Group features.	empty	struct ofp_group_features.
OFPMP_METER	9	Meter statistics.	struct ofp_meter_multipart_requests.	struct ofp_meter_stats.

OFPMP_METER_CONFIG	10	Meter configuration.	struct ofp_meter_multipart_requests.	struct ofp_meter_config.
OFPMP_METER_FEATURES	11	Meter features.	empty	struct ofp_meter_features.
OFPMP_TABLE_FEATURES	12	Table features.	struct ofp_table_features	struct ofp_table_features.
OFPMP_PORT_DESC	13	Port description.	empty	struct ofp_port.
OFPMP_EXPERIMENTER	14	Experimenter extension.	struct ofp_experimenter_multipart_header	struct ofp_experimenter_multipart_header

desc

```
/* Body of reply to OFPMP_DESC request.  Each entry is a NULL-terminated
 * ASCII string. */
struct ofp_desc {
    char mfr_desc[DESC_STR_LEN];       /* Manufacturer description. */
    char hw_desc[DESC_STR_LEN];        /* Hardware description. */
    char sw_desc[DESC_STR_LEN];        /* Software description. */
    char serial_num[SERIAL_NUM_LEN];   /* Serial number. */
    char dp_desc[DESC_STR_LEN];        /* Human readable description of datapath. */
};
OFP_ASSERT(sizeof(struct ofp_desc) == 1056);
```

ofp_flow_stats_request

struct ofp_match match

table_id：单个流表索引号，或 OFPTT_ALL

out_port 和 out_group

cookie：

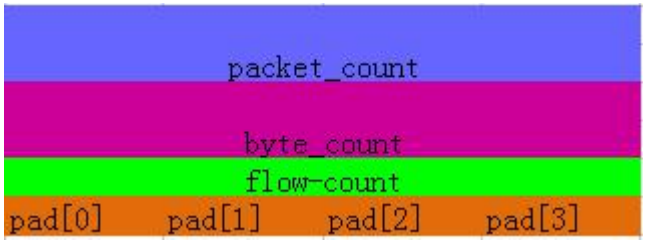
ofp_flow_stats

struct ofp_match match

table_id：流表 ID
duration_sec 流表存活时间(second)
duration_nsec 流表超过 duration_sec 的存活时间(nanosecond)
priority：流表项优先级
cookie：控制器发出的不透明标识符
packet_count：流表中数据包统计
byte_count：

ofp_aggregate_stats_request

ofp_aggregate_stats_reply



ofp_table_stats

Active_count：活跃流表项统计

lookup_count：

ofp_table_feature

Max_entries：支持的最大流表数量

struct ofp_table_feature_prop_header properties[0]：List of properties

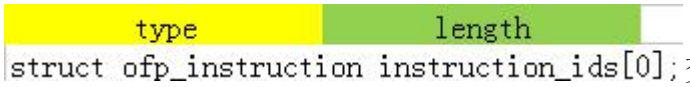
Table Feature properties

ofp_table_feature_prop_type

```
/* Table Feature property types.
 * Low order bit cleared indicates a property for a regular Flow Entry.
 * Low order bit set indicates a property for the Table-Miss Flow Entry.
 */
enum ofp_table_feature_prop_type {
    OFPTFPT_INSTRUCTIONS           = 0,  /* Instructions property. */
    OFPTFPT_INSTRUCTIONS_MISS      = 1,  /* Instructions for table-miss. */
    OFPTFPT_NEXT_TABLES            = 2,  /* Next Table property. */
    OFPTFPT_NEXT_TABLES_MISS       = 3,  /* Next Table for table-miss. */
    OFPTFPT_WRITE_ACTIONS          = 4,  /* Write Actions property. */
    OFPTFPT_WRITE_ACTIONS_MISS     = 5,  /* Write Actions for table-miss. */
    OFPTFPT_APPLY_ACTIONS          = 6,  /* Apply Actions property. */
    OFPTFPT_APPLY_ACTIONS_MISS     = 7,  /* Apply Actions for table-miss. */
    OFPTFPT_MATCH                  = 8,  /* Match property. */
    OFPTFPT_WILDCARDS              = 10, /* Wildcards property. */
    OFPTFPT_WRITE_SETFIELD         = 12, /* Write Set-Field property. */
    OFPTFPT_WRITE_SETFIELD_MISS    = 13, /* Write Set-Field for table-miss. */
    OFPTFPT_APPLY_SETFIELD         = 14, /* Apply Set-Field property. */
    OFPTFPT_APPLY_SETFIELD_MISS    = 15, /* Apply Set-Field for table-miss. */
    OFPTFPT_EXPERIMENTER           = 0xFFFE, /* Experimenter property. */
    OFPTFPT_EXPERIMENTER_MISS      = 0xFFFF, /* Experimenter for table-miss. */
};
```

ofp_table_feature_prop_header

ofp_table_feature_prop_instructions




```
struct ofp_instruction instruction_ids[0];
```
交换机支持的指令列表

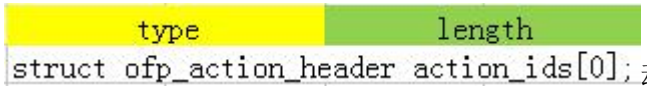
type:OFPTFPT_INSTRUCTIONS OFPTFPT_INSTRUCTIONS_MISS

ofp_table_feature_prop_next_tables


```
uint8_t next_table_ids[0];
```

type:OFPTFPT_NEXT_TABLES　OFPTFPT_NEXT_TABLES_MISS

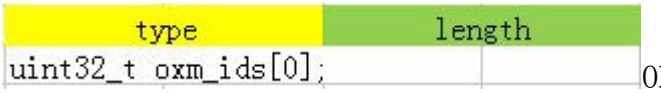
ofp_table_feature_prop_actions




```
struct ofp_action_header action_ids[0];
```
动作列表

type:
OFPTFPT_WRITE_ACTIONS, OFPTFPT_WRITE_ACTIONS_MISS:
OFPTFPT_APPLY_ACTIONS and
OFPTFPT_APPLY_ACTIONS_MISS

ofp_table_feature_prop_oxm



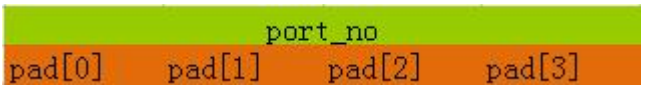
OXM 类型列表

type:OFPTFPT_MATCH, OFPTFPT_WILDCARDS, OFPTFPT_WRITE_SETFIELD, OFPTFPT_WRITE_SETFIELD_MISS, OFPTFPT_APPLY_SETFIELD and OFPTFPT_APPLY_SETFIELD_MISS

ofp_table_feature_prop_instructions

Experimenter ID

ofp_port_stats_request

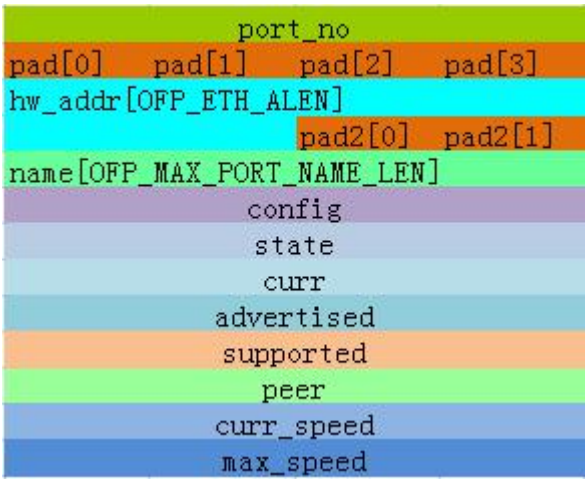


port_no：过滤到指定端口的统计请求

ofp_port_stats

rx_packets：接收数据包数量
tx_packets：传输数据包数量
rx_bytes：接收比特值
tx_bytes：发送比特值
rx_droped：接收机丢包数
tx_droped：发送机丢包数
rx_error：接收端错误数
tx_error：发送端错误数
rx_over_err
rx_crc_err
Collisions：冲突值
duration_sec
duration_nsec

ofp_port



curr：当前特性

advertised：端口传输特性

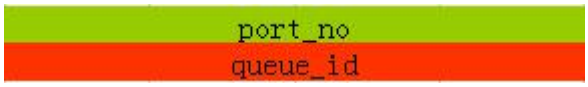
supported:端口支持特性

peer：

curr_speed：当前速率

max_speed：最大速率

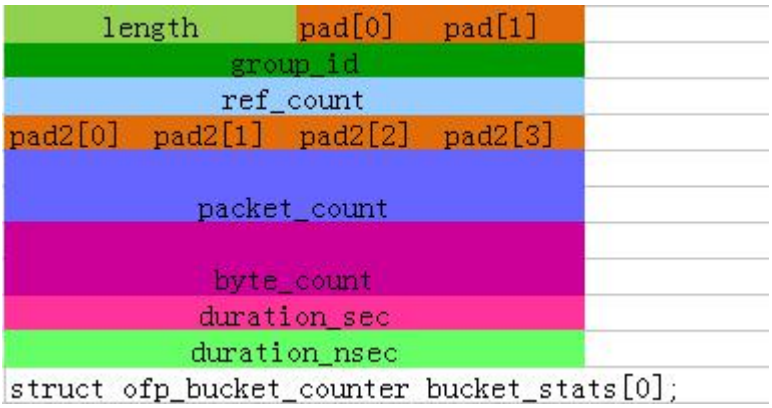
ofp_queue_stats_request



ofp_queue_stats

ofp_group_stats_request

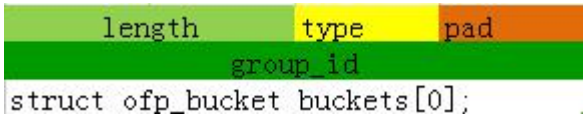
ofp_group_stats



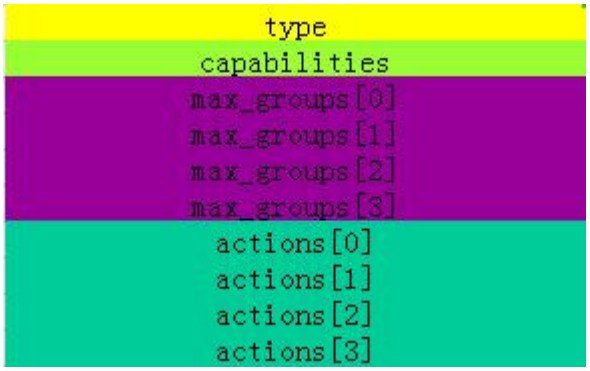
ref_count:直接向组转发的流表或组

ofp_bucket_counter

ofp_group_desc



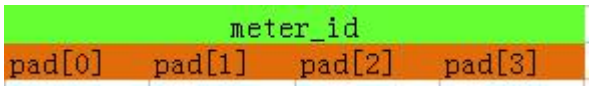
ofp_group_features



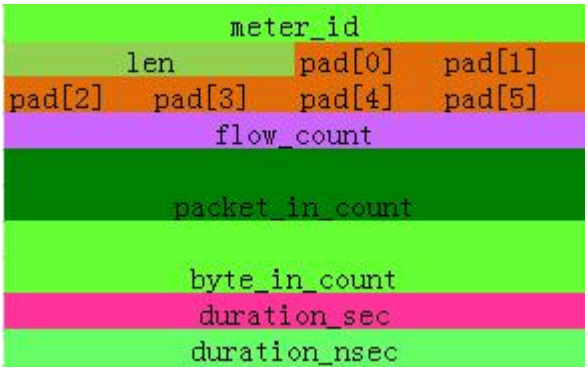
capabilities:

```
/* Group configuration flags */
enum ofp_group_capabilities {
    OFPGFC_SELECT_WEIGHT   = 1 << 0,  /* Support weight for select groups */
    OFPGFC_SELECT_LIVENESS = 1 << 1,  /* Support liveness for select groups */
    OFPGFC_CHAINING        = 1 << 2,  /* Support chaining groups */
    OFPGFC_CHAINING_CHECKS = 1 << 3,  /* Check chaining for loops and delete */
};
```

ofp_group_feature



ofp_meter_multipart_stats

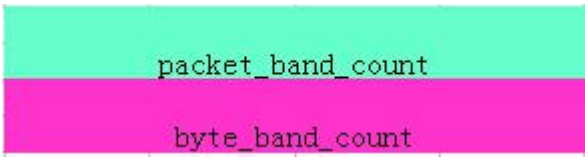


flow_count：Number of flows bound to meter.

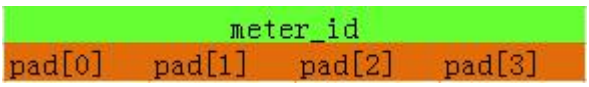
packet_in_count：Number of packets in input.

byte_in_count：Number of bytes in input.

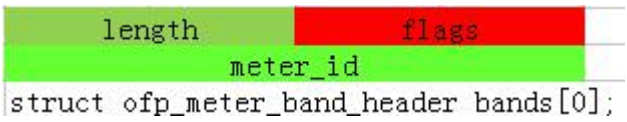
ofp_meter_band_stats



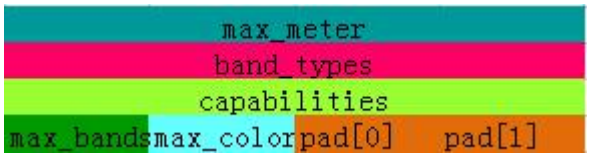
ofp_meter_multipart_request



ofp_meter_config



ofp_meter_features



max_meter：应用表最大数量

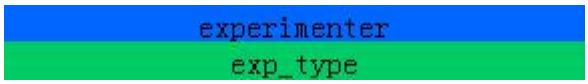
band_types:Bitmaps of OFPMBT_* values supported.

capabilities:Bitmaps of "ofp_meter_flags"

max_band:

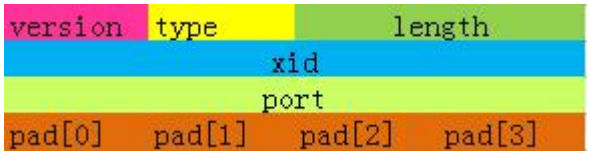
max_color:

ofp_experimenter_multipart_header



7.3.6 队列配置信息

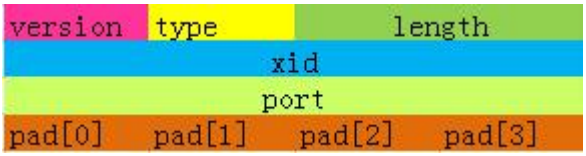
ofp_queue_get_config_request



struct ofp_header header

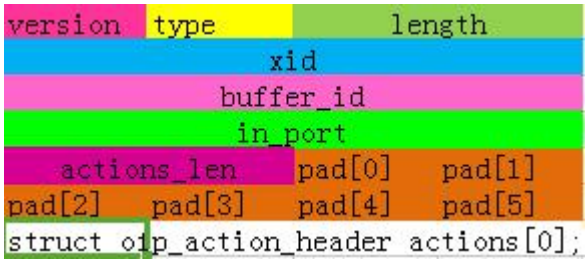
port：待查询端口

ofp_queue_get_config_reply



7.3.7 Packet_Out 消息

ofp_packet_out



buffer_id：与 Packet_In 相同

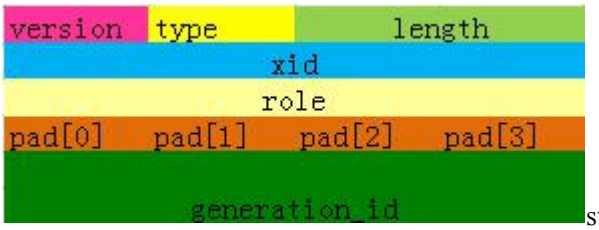
in_port：进入端口

actions_len：动作列表，定义数据包如何处理

7.3.8 Barrier Message

7.3.9 Role Request Message

ofp_role_request



struct ofp_header header

role：

```
/* Controller roles. */
enum ofp_controller_role {
    OFPCR_ROLE_NOCHANGE = 0,    /* Don't change current role. */
    OFPCR_ROLE_EQUAL    = 1,    /* Default role, full access. */
    OFPCR_ROLE_MASTER   = 2,    /* Full access, at most one master. */
    OFPCR_ROLE_SLAVE    = 3,    /* Read-only access. */
};
```

ROLE_NOCHANGE：当前的 role 不改变

ROLE_EQUAL：Equal 表明这个控制器并没有什么特殊之处，他和其他同样为 equal 的控制器是同等级的。equal 类型控制器相当于一个独立的，具有完全权限的控制器。

ROLE_MASTER：Master 角色具有和 Equal 一样的完全权限。其他 role 为 ROLE_NOCHANGE 控制器变为 ROLE_SLAVE

ROLE_SLAVE：作为 Slave 角色的控制器对交换机仅有可读权限,不能接受异步消息（除去 port_status 以外的其他异步等消息）不能向交换机发送写消息（ofp_flow_mod 等），若交换机收到 slave 控制器发送的写消息，将产生 ERROR。

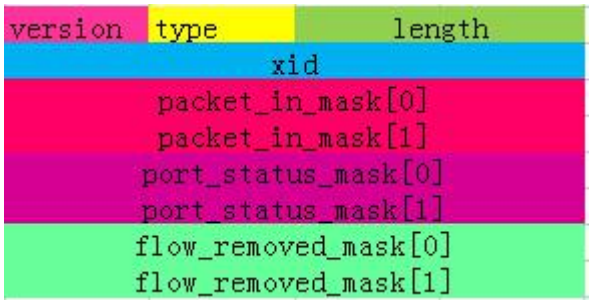
generation_id：当 role 的值为 OFPCR_ROLE_MASTER 或 OFPCR_ROLE_SLAVE 时，，
验证 generation_id 检查过期消息

role_reply

控制器发送 role_request，如果没有错误，交换机需要回复 role_reply

7.3.10 Set Asynchronous Conguration Message

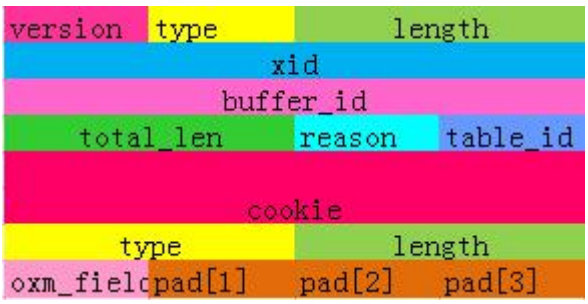
ofp_async_config



7.4 Asynchronous 消息

7.4.1 Packet-In Message

ofp_packet_in



ofp_header header;

ofp_match match;

buffer_id：数据通路用于识别缓冲数据包的不透明值。当包被缓冲时，消息中的一些字节数将包含在消息的数据部分中。如果数据包的发送是因为一个“发送到控制器”的动作，然后 max_len 字节的流量设置要求 ofp_action_output 发送。如果数据包发送的其他原因，如无效的 TTL，然后从 ofpt_set_config 至少 miss_send_len 字节发送消息。默认 miss_send_len 是 128 字节。如果数据没有被缓存-要么是因为没有可用的缓冲区，或由于明确要求通过 ofpcml_no_buffer -整个包中包含的数据部分，和 buffer_id 是 ofp_no_buffer。

若 reason 为 PFPR_ACTION，

reason:

```
/* Why is this packet being sent to the controller? */
enum ofp_packet_in_reason {
    OFPR_NO_MATCH    = 0,   /* No matching flow (table-miss flow entry). */
    OFPR_ACTION      = 1,   /* Action explicitly output to controller. */
    OFPR_INVALID_TTL = 2,   /* Packet has invalid TTL */
};
```

OFPAT_DEC_MPLS_TTL 或 OFPAT_DEC_NW_TTL 应用于数据包，检验无效的 TTL。

Cookie：the cookie of the flow entry that caused the packet to be sent to the controller

当 cookie 不能和特定的流表相关联时，值为-1

7.4.2 Flow Removed Message

ofp_flow_removed

ofp_header header;

ofp_match match;

match、cookie 和 priority 域和 flow mod request 相同

reason:

```
/* Why was this flow removed? */
enum ofp_flow_removed_reason {
    OFPRR_IDLE_TIMEOUT = 0,     /* Flow idle time exceeded idle_timeout.
    OFPRR_HARD_TIMEOUT = 1,     /* Time exceeded hard_timeout. */
    OFPRR_DELETE       = 2,     /* Evicted by a DELETE flow mod. */
    OFPRR_GROUP_DELETE = 3,     /* Group was removed. */
};
```

duration_sec 和 duration_nsec 分别在 7.3.5.2 进行描述。

packet_count 和 byte_count 分别代表关于流表项的数据包数和字节数.

7.4.3 Port Status Message

ofp_port_status

struct ofp_header header

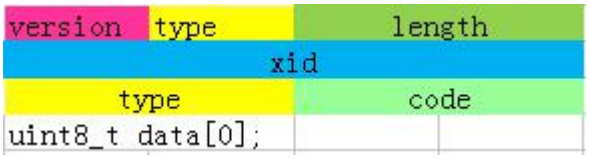
struct ofp_port

reason:

```
/* What changed about the physical port */
enum ofp_port_reason {
    OFPPR_ADD    = 0,           /* The port was added. */
    OFPPR_DELETE = 1,           /* The port was removed. */
    OFPPR_MODIFY = 2,           /* Some attribute of the port has changed. */
};
```

7.4.4 Error Message

ofp_error_msg



type：error 的等级类型

```
/* Values for 'type' in ofp_error_message.  These values are immutable: they
 * will not change in future versions of the protocol (although new values may
 * be added). */
enum ofp_error_type {
    OFPET_HELLO_FAILED         = 0,  /* Hello protocol failed. */
    OFPET_BAD_REQUEST          = 1,  /* Request was not understood. */
    OFPET_BAD_ACTION           = 2,  /* Error in action description. */
    OFPET_BAD_INSTRUCTION      = 3,  /* Error in instruction list. */
    OFPET_BAD_MATCH            = 4,  /* Error in match. */
    OFPET_FLOW_MOD_FAILED      = 5,  /* Problem modifying flow entry. */
    OFPET_GROUP_MOD_FAILED     = 6,  /* Problem modifying group entry. */
    OFPET_PORT_MOD_FAILED      = 7,  /* Port mod request failed. */
    OFPET_TABLE_MOD_FAILED     = 8,  /* Table mod request failed. */
    OFPET_QUEUE_OP_FAILED      = 9,  /* Queue operation failed. */
    OFPET_SWITCH_CONFIG_FAILED = 10, /* Switch config request failed. */
    OFPET_ROLE_REQUEST_FAILED  = 11, /* Controller Role request failed. */
    OFPET_METER_MOD_FAILED     = 12, /* Error in meter. */
    OFPET_TABLE_FEATURES_FAILED = 13, /* Setting table features failed. */
    OFPET_EXPERIMENTER = 0xffff      /* Experimenter error messages. */
};
```

code:type 解释

- OFPET_HELLO_FAILED:

```
enum ofp_hello_failed_code      {
OFPHFC_INCOMPATIBLE 0,  /* 不兼容的版本     */
OFPHFC_EPERM 1,         /*    允许失败      */
};
```

- OFPET_BAD_REQUEST

```
/* ofp_error_msg 'code' values for OFPET_BAD_REQUEST.  'data' contains at least
 * the first 64 bytes of the failed request. */
enum ofp_bad_request_code {
    OFPBRC_BAD_VERSION      = 0,  /* ofp_header.version not supported. */
    OFPBRC_BAD_TYPE         = 1,  /* ofp_header.type not supported. */
    OFPBRC_BAD_MULTIPART    = 2,  /* ofp_multipart_request.type not supported. */
    OFPBRC_BAD_EXPERIMENTER = 3,  /* Experimenter id not supported
                                   * (in ofp_experimenter_header or
                                   * ofp_multipart_request or
                                   * ofp_multipart_reply). */
    OFPBRC_BAD_EXP_TYPE     = 4,  /* Experimenter type not supported. */
    OFPBRC_EPERM            = 5,  /* Permissions error. */
    OFPBRC_BAD_LEN          = 6,  /* Wrong request length for type. */
    OFPBRC_BUFFER_EMPTY     = 7,  /* Specified buffer has already been used. */
    OFPBRC_BUFFER_UNKNOWN   = 8,  /* Specified buffer does not exist. */
    OFPBRC_BAD_TABLE_ID     = 9,  /* Specified table-id invalid or does not
                                   * exist. */
    OFPBRC_IS_SLAVE         = 10, /* Denied because controller is slave. */
    OFPBRC_BAD_PORT         = 11, /* Invalid port. */
    OFPBRC_BAD_PACKET       = 12, /* Invalid packet in packet-out. */
    OFPBRC_MULTIPART_BUFFER_OVERFLOW    = 13, /* ofp_multipart_request
```

- OFPET_BAD_ACTION：

```
/* ofp_error_msg 'code' values for OFPET_BAD_ACTION.  'data' contains at least
 * the first 64 bytes of the failed request. */
enum ofp_bad_action_code {
    OFPBAC_BAD_TYPE           = 0,  /* Unknown action type. */
    OFPBAC_BAD_LEN            = 1,  /* Length problem in actions. */
    OFPBAC_BAD_EXPERIMENTER   = 2,  /* Unknown experimenter id specified. */
    OFPBAC_BAD_EXP_TYPE       = 3,  /* Unknown action for experimenter id. */
    OFPBAC_BAD_OUT_PORT       = 4,  /* Problem validating output port. */
    OFPBAC_BAD_ARGUMENT       = 5,  /* Bad action argument. */
    OFPBAC_EPERM              = 6,  /* Permissions error. */
    OFPBAC_TOO_MANY           = 7,  /* Can't handle this many actions. */
    OFPBAC_BAD_QUEUE          = 8,  /* Problem validating output queue. */
    OFPBAC_BAD_OUT_GROUP      = 9,  /* Invalid group id in forward action. */
    OFPBAC_MATCH_INCONSISTENT = 10, /* Action can't apply for this match,
                                       or Set-Field missing prerequisite. */
    OFPBAC_UNSUPPORTED_ORDER  = 11, /* Action order is unsupported for the
                                       action list in an Apply-Actions instruction */
    OFPBAC_BAD_TAG            = 12, /* Actions uses an unsupported
                                       tag/encap. */
    OFPBAC_BAD_SET_TYPE       = 13, /* Unsupported type in SET_FIELD action. */
    OFPBAC_BAD_SET_LEN        = 14, /* Length problem in SET_FIELD action. */
    OFPBAC_BAD_SET_ARGUMENT   = 15, /* Bad argument in SET_FIELD action. */
};
```

- OFPET_BAD_INSTRUCTION：

```
/* ofp_error_msg 'code' values for OFPET_BAD_INSTRUCTION.  'data' contains at least
 * the first 64 bytes of the failed request. */
enum ofp_bad_instruction_code {
    OFPBIC_UNKNOWN_INST     = 0, /* Unknown instruction. */
    OFPBIC_UNSUP_INST       = 1, /* Switch or table does not support the
                                    instruction. */
    OFPBIC_BAD_TABLE_ID     = 2, /* Invalid Table-ID specified. */
    OFPBIC_UNSUP_METADATA   = 3, /* Metadata value unsupported by datapath. */
    OFPBIC_UNSUP_METADATA_MASK = 4, /* Metadata mask value unsupported by
                                       datapath. */
    OFPBIC_BAD_EXPERIMENTER = 5, /* Unknown experimenter id specified. */
    OFPBIC_BAD_EXP_TYPE     = 6, /* Unknown instruction for experimenter id. */
    OFPBIC_BAD_LEN          = 7, /* Length problem in instructions. */
    OFPBIC_EPERM            = 8, /* Permissions error. */
};
```

- OFPET_BAD_MATCH

```
/* ofp_error_msg 'code' values for OFPET_BAD_MATCH.  'data' contains at least
 * the first 64 bytes of the failed request. */
enum ofp_bad_match_code {
    OFPBMC_BAD_TYPE         = 0,  /* Unsupported match type specified by the
                                     match */
    OFPBMC_BAD_LEN          = 1,  /* Length problem in match. */
    OFPBMC_BAD_TAG          = 2,  /* Match uses an unsupported tag/encap. */
    OFPBMC_BAD_DL_ADDR_MASK = 3,  /* Unsupported datalink addr mask - switch
                                     does not support arbitrary datalink
                                     address mask. */
    OFPBMC_BAD_NW_ADDR_MASK = 4,  /* Unsupported network addr mask - switch
                                     does not support arbitrary network
                                     address mask. */
    OFPBMC_BAD_WILDCARDS    = 5,  /* Unsupported combination of fields masked
                                     or omitted in the match. */
    OFPBMC_BAD_FIELD        = 6,  /* Unsupported field type in the match. */
    OFPBMC_BAD_VALUE        = 7,  /* Unsupported value in a match field. */
    OFPBMC_BAD_MASK         = 8,  /* Unsupported mask specified in the match,
                                     field is not dl-address or nw-address. */
    OFPBMC_BAD_PREREQ       = 9,  /* A prerequisite was not met. */
    OFPBMC_DUP_FIELD        = 10, /* A field type was duplicated. */
    OFPBMC_EPERM            = 11, /* Permissions error. */
};
```

- OFPET_TABLE_MOD_FAILED

```
enum ofp_flow_mod_failed_code {
OFPFMFC_UNKNOWN = 0,      /* 未指定的错误 */
OFPFMFC_TABLE_FULL = 1,   /* 因为表已满，流不能添加。 */
OFPFMFC_BAD_TABLE_ID = 2, /* 流表不存在 */
OFPFMFC_OVERLAP = 3,      /* 使用 check_overlap 标志尝试添加重叠的流 */
OFPFMFC_EPERM = 4,        /* 允许的错误。 */
OFPFMFC_BAD_TIMEOUT = 5,  /* 因为不支持空闲 / 硬超时，流表不被添加。 */
OFPFMFC_BAD_COMMAND = 6,  /* 不支持的或未知的命令。 */
OFPFMFC_BAD_FLAGS = 7,    /* 不支持的或未知的标志。 */
};
```

- OFPET_GROUP_MOD_FAIL=ED

```
enum ofp_group_mod_failed_code {
OFPGMFC_GROUP_EXISTS = 0,  /* 组不添加因为组添加试图取代一个已经存在的组 */
OFPGMFC_INVALID_GROUP = 1, /* 组不能添加因为组指定的组是无效的 */
OFPGMFC_WEIGHT_UNSUPPORTED = 2, /* 交换机在所选组中不支持不均等的负荷分担 */
OFPGMFC_OUT_OF_GROUPS = 3, /* 组表已满。 */
OFPGMFC_OUT_OF_BUCKETS = 4, /* 组的行动存储段已超过最大值。 */
OFPGMFC_CHAINING_UNSUPPORTED = 5, /* 交换机不支持需要转发去的组 */
OFPGMFC_WATCH_UNSUPPORTED = 6, /* 这个组不能监视 w指定的 atch_port 或watch_group 。 */
OFPGMFC_LOOP = 7,          /* 组表项会引起循环 */
OFPGMFC_UNKNOWN_GROUP = 8, /* 组不能修改，因为修改组试图修改一个不存在的组 */
OFPGMFC_CHAINED_GROUP = 9, /* 组不能删除因为另一组是向其转发。 */
OFPGMFC_BAD_TYPE = 10,     /* 不支持的或未知的组类型 */
OFPGMFC_BAD_COMMAND = 11,  /* 不支持的或未知的命令。 */
OFPGMFC_BAD_BUCKET = 12,   /* 存储段里的错误 */
OFPGMFC_BAD_WATCH = 13,    /* 在观察端口 / 组的错误。 */
OFPGMFC_EPERM = 14,        /* 允许的错误。 */
};

enum ofp_port_mod_failed_code {
OFPPMFC_BAD_PORT = 0,     /* 指定的端口号不存在 */
OFPPMFC_BAD_HW_ADDR = 1,  /* 指定的硬件地址不匹配端口号
OFPPMFC_BAD_CONFIG = 2,   /* 指定的配置无效 */
OFPPMFC_BAD_ADVERTISE = 3, /* 指定的标识符是无效的。 */
OFPPMFC_EPERM = 4,        /* 允许的错误. */
};

enum ofp_table_mod_failed_code {
OFPTMFC_BAD_TABLE = 0, /* 指定的流表不存在 */
OFPTMFC_BAD_CONFIG = 1, /* 指定的配置无效 */
OFPTMFC_EPERM = 2, /* 允许的错误 */
};

enum ofp_queue_op_failed_code {
OFPQOFC_BAD_PORT = 0, /* 无效的端口（或端口不存在） */
OFPQOFC_BAD_QUEUE = 1, /* 队列不存在 */
OFPQOFC_EPERM = 2, /* 允许的错误 */
};
```

```
enum ofp_switch_config_failed_code {
OFPSCFC_BAD_FLAGS=0, /* 指定的标志无效 */
OFPSCFC_BAD_LEN=1, /* 指定的长度无效 */
OFPSCFC_EPERM=2, /* 允许的错误 */
};

enum ofp_role_request_failed_code {
OFPRRFC_STALE=0, /* 过时消息： 旧的 generation_id. */
OFPRRFC_UNSUP=1, /* 控制器不支持角色的变化 */
OFPRRFC_BAD_ROLE=2, /* 无效的角色 */

enum ofp_meter_mod_failed_code {
OFPMMFC_UNKNOWN=0, /* 未指定的错误. */
OFPMMFC_METER_EXISTS=1, /* 计量器不加因为试图取代现有的计量器 */
OFPMMFC_INVALID_METER=2, /* 计量器不加因为指定的计量器是无效的 */
OFPMMFC_UNKNOWN_METER /* 计量器不修改因为试图修改一个不存在的计量器
OFPMMFC_BAD_COMMAND=3, /* 不支持的或未知的命令 */
OFPMMFC_BAD_FLAGS=5, /* 配置的标志不支持 */
OFPMMFC_BAD_RATE=6, /* 速率不支持 */
OFPMMFC_BAD_BURST=7, /* 不支持的突发大小. */
OFPMMFC_BAD_BAND=8, /* 频带不支持 */
OFPMMFC_BAD_BAND_VALUE=9, /* 频带值不被支持 */
OFPMMFC_OUT_OF_METERS=10, /* 没有更多的计量器有效 */
OFPMMFC_OUT_OF_BANDS=11, /* 对于计量器的属性数量已超过最大值. */
};

enum ofp_table_features_failed_code {
OFPTFFC_BAD_TABLE=0, /* 指定的流表不存在 */
OFPTFFC_BAD_METADATA=1, /* 无效的元数据的掩码 */
OFPTFFC_BAD_TYPE=2, /* 未知的属性类型. */
OFPTFFC_BAD_LEN=3, /* 属性长度问题 */
OFPTFFC_BAD_ARGUMENT=4, /* 不支持的属性值 */
OFPTFFC_EPERM=5, /* 允许的错误 */
};

struct ofp_error_experimenter_msg {
struct ofp_header header;
uint16_t type; /* ofpet_experimenter*/
uint16_t exp_type; /* 实验者定义 */
uint32_t experimenter; / 实验者 ID与ofp_experimenter_header 中相同 */
uint8_t data[0]; /* 可变长度的数据。基于类型和代码解析。无填充
};
OFP_ASSERT(sizeof(struct ofp_error_experimenter_msg) == 16);
```

7.5 Symmetric 消息

7.5.1 Hello

ofp_hello

struct ofp_header header

ofp_hello_elem_header

hello 元素，包含连接进行初次握手时传递的可选数据

elements:

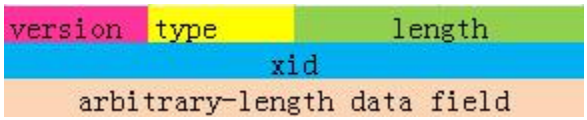
```
/* Hello elements types.
 */
enum ofp_hello_elem_type {
    OFPHET_VERSIONBITMAP          = 1,  /* Bitmap of version supported. */
};
```

OFPHET_VERSIONBITMAP：

struct ofp_hello_elem_versionbitmap:

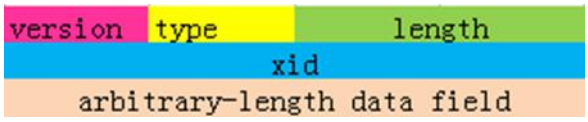
Bitmap：OpenFlow 协议支持的版本集合

7.5.2 Echo Request



arbitrary-length data field：可能是消息的时间戳来检查时延、不同长度来测量带宽，或零大小来验证交换机和控制器之间是否活跃。

7.5.3 Echo Reply



接收一个格式正确的回应则表示端到端功能比在用户空间的进程中实现的回声请求/ 应答更可靠，同时也提供了更精确的端到端延迟时间

7.5.4 Experimenter

ofp_experimenter_header