基于 ryu 控制器的流量处理

一、数据包格式

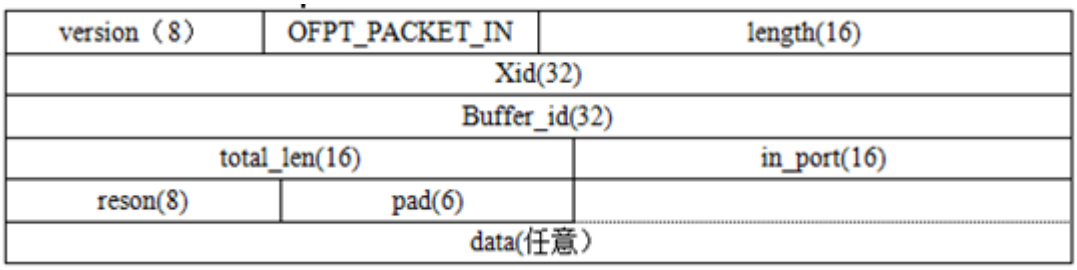
交换机和控制器之间传输的数据格式是 **packet-in/out** 数据包，当交换机收到数据包，首先会匹配交换机内的流表，如果匹配到就直接转发，没有匹配到，说明交换机不知道怎么处理当前这个数据包，就需要进行学习或者说，通知控制器下发新的相应的规则，这个时候，交换机就会把数据包上送到 ryu 控制器，交换机和控制器之间通过 openflow 协议交互，数据包格式是 packet-in/packet-out 消息。那么我们想要在 ryu 控制器进行数据处理，首先就要了解数据格式。

Packet-in 消息的功能是：将到达 OpenFlow 交换机的数据包发送到控制器。触发 Packet-in 消息的原因有以下两种：

1）不存在与流表项一致的项目（OFPR_NO_MATCH）

2）匹配的流表项中的行动为“发往控制器”（OFPR_ACTION）

Packet-in 消息格式如下图所示：

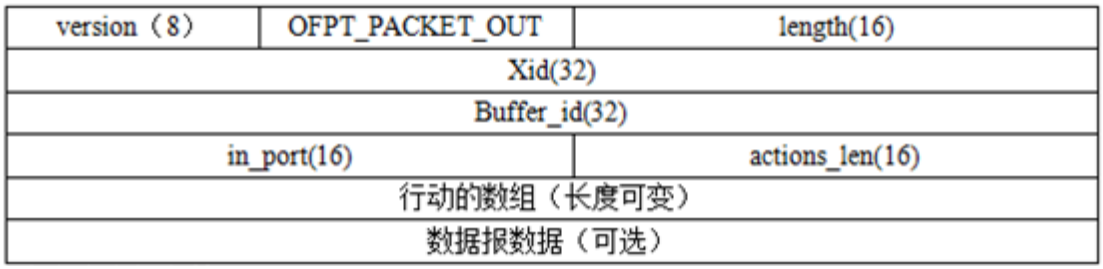


各字段具体含义如下表所示：

字段	比特数	内容
buffer_id	32	表示 OpenFlow 交换机中保存的数据包的缓存 id
Total_len	16	帧的长度
in_port	16	接受帧的端口
reason	8	发送 Packet-in 消息的原因
pad	8	用于调整对齐的填充

data	任意	包含以太网帧的数据时使用的字段。

Packet-out 消息的功能是：将控制器的相关数据发送到 OpenFlow 交换机，是包含数据包发送命令的消息。Packet-out 消息格式如下图所示。



各字段具体含义如下表所示。

字段	比特数	内容
buffer_id	32	表示 OpenFlow 交换机中保存的数据包的缓存 id
in_port	16	数据包的输入端口
actions_len	16	行动信息的长度

二、ryu 控制器给交换机下发流表

后面再描述 ryu 控制器解析处理数据包的流程，先来看 ryu 控制器给交换机下发流表的接口。在交换机（数据面），流表是 match-action 的形式，在控制器给交换机传递消息的时候同样也是用这种形式去构造 packet-out 数据包。这些接口函数主要包含在 ofproto 库中。

1. match 字段构造：OFPMatch

OFPMatch()的参数可以是数据包解析出来的任何字段，比如 in_port=1，eth_dst=aa:aa:aa:aa:aa:aa 等等，如果不带参数就相当于默认流表，任何数据包都可以匹配到这一个条目。例如：

match = ofp_parser.**OFPMatch**()

match = parser.**OFPMatch**(ipv4_src=192.168.0.1)

2. actions 字段构造：OFPActionOutput

首先流表有动作字段，那么，actions 列表格式是什么？怎么构造？for example:

actions = [ev.msg.datapath.ofproto_parser. **OFPActionOutput** (out_port)]

ryu.ofproto.ofproto_v1_3_parser**.OFPActionOutput**(port, max_len=65509, type_=None, len_=None)

将数据报文转发到端口 port，port 可以是出口端口的编号，比如 1、2、3 号端口：out_port = 3，actions = [parser.OFPActionOutput(out_port)]。

也有一些定义好的物理端口和逻辑端口，如下图。比如我们现在不知道一个数据包应该往哪一个端口转发，就准备进行一个洪泛，向所有端口转发，这时：

out_port = ofproto.OFPP_FLOOD

actions = [parser.OFPActionOutput(out_port)]

openfaucet.ofproto. **OFPP_MAX**
The maximum number of physical ports in a datapath.

In addition to physical ports, several logical ports are defined:

openfaucet.ofproto. **OFPP_IN_PORT**
The input port, i.e. the packet was received on. This virtual port must be explicitly used in order to send back out of the input port.

openfaucet.ofproto. **OFPP_TABLE**
Perform actions in the flow table on the packet. This can be the destination port only for `OFPT_PACKET_OUT` messages, as sent using `send_packet_out()`.

openfaucet.ofproto. **OFPP_NORMAL**
Process with normal L2/L3 switching performed by the datapath.

openfaucet.ofproto. **OFPP_FLOOD**
All physical ports, except the input port and those disabled by Spanning Tree Protocol.

openfaucet.ofproto. **OFPP_ALL**
All physical ports except the input port.

openfaucet.ofproto. **OFPP_CONTROLLER**
Send to the controller, in an `OFPT_PACKET_IN`, as received by a controller in the `handle_packet_in()` callback.

openfaucet.ofproto. **OFPP_LOCAL**
The local OpenFlow virtual port used for in-band control traffic.

openfaucet.ofproto. **OFPP_NONE**
Not associated with any port.

The following actions output packets to ports directly or to queues associated with specific ports.

class openfaucet.ofaction. **ActionOutput**
An `OFPAT_OUTPUT` action, which sends packets out a given port.

设置好动作 action 之后，我们要将这个动作用符合规定的方式构造为下发动作的一个指令。

3. **指令构造：OFPInstructionActions**

ryu.ofproto.ofproto_v1_3_parser.**OFPInstructionActions**(type_, actions=None, len_=None)

action instruction 动作说明函数

type 参数有三种：

1）OFPIT_APPLY_ACTIONS，这个指令是真正包含动作的指令，它的行为是立即对报文应用这些指令，不要改变报文的 action set；

2）OFPIT_CLEAR_ACTION，这个指令并不包含任何的动作，它的行为是立即清除报文的 action set 中所有的动作；

3）OFPIT_WRITE_ACTIONS，这个指令真正的包含动作，它的行为是将自己包含的动作合并到报文的 action set 中；

看到这里，我们会疑惑，action set 是什么？一个报文走完所有流表时，其 action set 里面有什么动作，就接着执行什么动作，这就是 action set 的作用了。

actions 参数：一个动作列表，表示这个指令包含的动作

那么，actions 列表格式是什么？怎么构造？for example:

actions = [ev.msg.datapath.ofproto_parser. **OFPActionOutput** (out_port)]

有了 match 匹配字段和 action 动作字段，一条转发条目的基本信息具备了，接着我们需要把这样一条转发条目构造成消息，从控制器下发给交换机。那么构造成消息的 API 是什么？

4. 用于修改流表的消息构造 OFPFlowMod

ryu.ofproto.ofproto_v1_3_parser.**OFPFlowMod**(datapath, cookie=0, cookie_mask=0, table_id=0, command=0, idle_timeout=0, hard_timeout=0, priority=32768, buffer_id=4294967295, out_port=0, out_group=0, flags=0, match=None, instructions=None)

参数具体含义见下图，比如 instructions 就是我们前面用 OFPInstructionActions 函数构造的动作说明，举个例子：

mod =

ofp_parser.OFPFlowMod(datapath=datapath,priority=priority,hard_timeout=hard_timeout,match=match,instructions=inst)

Attribute	Description
cookie	Opaque controller-issued identifier
cookie_mask	Mask used to restrict the cookie bits that must match when the command is `OPFFC_M`
table_id	ID of the table to put the flow in
command	One of the following values. OFPFC_ADD OFPFC_MODIFY OFPFC_MODIFY_STRICT OFPFC_DELETE OFPFC_DELETE_STRICT
idle_timeout	Idle time before discarding (seconds)
hard_timeout	Max time before discarding (seconds)
priority	Priority level of flow entry
buffer_id	Buffered packet to apply to (or OFP_NO_BUFFER)
out_port	For `OFPFC_DELETE*` commands, require matching entries to include this as an output
out_group	For `OFPFC_DELETE*` commands, require matching entries to include this as an output
flags	Bitmap of the following flags. OFPFF_SEND_FLOW_REM OFPFF_CHECK_OVERLAP OFPFF_RESET_COUNTS OFPFF_NO_PKT_COUNTS OFPFF_NO_BYT_COUNTS
match	Instance of `OFPMatch`
instructions	list of `OFPInstruction*` instance

5.控制器发送消息到交换机：send_msg()

最后 datapath.send_msg()函数发送的是一个 OpenFlow 的数据结构（就是上面构造的消息），RYU 将把这个数据发送到对应的 datapath。

datapath.**send_msg**(mod)

到这一步，对当前这个报文的新的处理条目以及下发到交换的流表中了，但是交换机并没有将当前这个新的流表应用于当前这个报文，因为理论上，这个报

文是已经匹配了一遍流表，然后才上送到控制器的，所以我们需要用一个命令明确指示交换机把 buffer 中的那个数据报文来匹配这次下发的流表项，就要用到 OFPPacketOut()。

6. 控制器通知交换机处理当前数据包：OFPPacketOut()

ryu.ofproto.ofproto_v1_3_parser.**OFPPacketOut**(datapath, buffer_id=None, in_port=None, actions=None, data=None, actions_len=None)

这个函数的作用是控制器发送消息使 buffer 中的一个数据包输出交换机。参数详解如下图。

Attribute	Description
buffer_id	ID assigned by datapath (OFP_NO_BUFFER if none)
in_port	Packet's input port or `OFPP_CONTROLLER`
actions	list of OpenFlow action class
data	Packet data of a binary type value or an instances of packet.Packet.

举个例子：

if msg.buffer_id == ofproto.OFP_NO_BUFFER: //如果是没有缓存数据包的，就需要把控制器这边的数据包数据赋给 data，作为参数传入，如果有缓存报文，buffer_id=msg.buffer_id 会直接从 buffer 中去读取当前的报文。

```
        data = msg.data
    out = parser.OFPPacketOut(datapath=datapath, buffer_id=msg.buffer_id,
                              in_port=in_port, actions=actions, data=data)
    datapath.send_msg(out)
```

三、报文解析

通过 **ev.msg** 获取报文信息，每一个事件类 ev 中都有 msg 成员，用于携带触发事件的数据包。利用 packet.Packet 函数可以按格式获取到数据包内容，然后用 get_protocol()函数依次按协议解析数据包各协议层的内容。

```
pkt = packet.Packet(msg.data)
```

eth_pkt = pkt.get_protocol(ethernet.ethernet)

ip_pkt = pkt.get_protocol(ipv4.ipv4)

ipv6_pkt = pkt.get_protocol(ipv6.ipv6)

icmp_pkt = pkt.get_protocol(icmp.icmp)

arp_pkt = pkt.get_protocol(arp.arp)

icmpv6_pkt = pkt.get_protocol(icmpv6.icmpv6)

tcp_pkt = pkt.get_protocol(tcp.tcp)

udp_pkt = pkt.get_protocol(udp.udp)

以太头部的源/目的 mac：eth_pkt.src/eth_pkt.dst

ip 头部的源/目的 ip：ip_pkt.src/ ip_pkt.dst

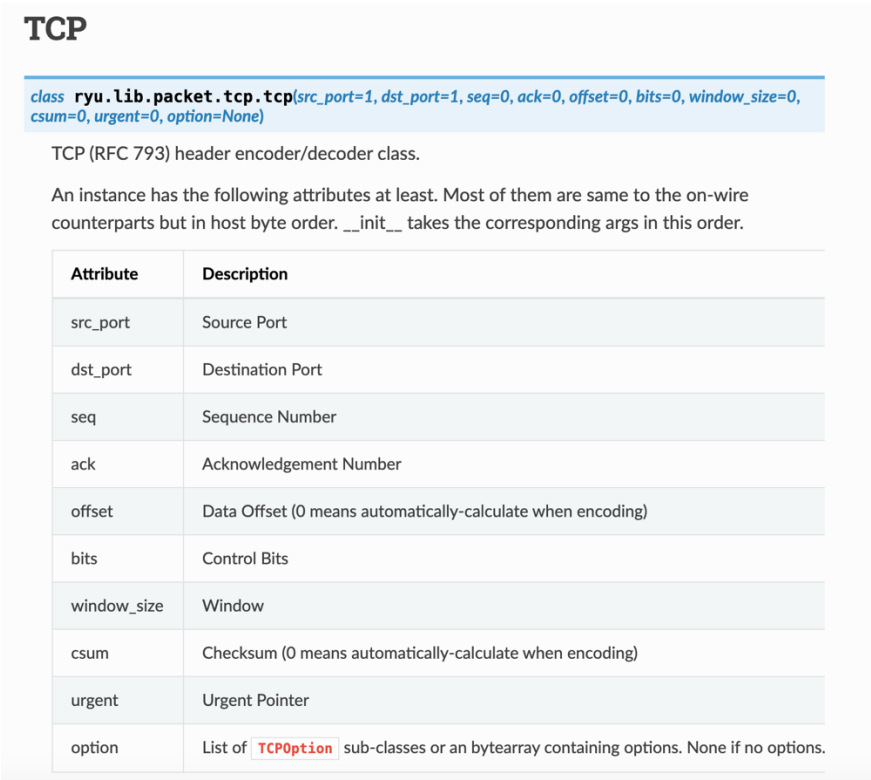
tcp 端口：tcp_pkt.src_port/ tcp_pkt.dst_port

… …

每个字段都有对应的接口，上网搜即可:

https://ryu.readthedocs.io/en/latest/library_packet_ref.html

例如 tcp 的各字段如下图。



TCP

class **ryu.lib.packet.tcp.tcp**(*src_port=1, dst_port=1, seq=0, ack=0, offset=0, bits=0, window_size=0, csum=0, urgent=0, option=None*)

TCP (RFC 793) header encoder/decoder class.

An instance has the following attributes at least. Most of them are same to the on-wire counterparts but in host byte order. __init__ takes the corresponding args in this order.

Attribute	Description
src_port	Source Port
dst_port	Destination Port
seq	Sequence Number
ack	Acknowledgement Number
offset	Data Offset (0 means automatically-calculate when encoding)
bits	Control Bits
window_size	Window
csum	Checksum (0 means automatically-calculate when encoding)
urgent	Urgent Pointer
option	List of `TCPOption` sub-classes or an bytearray containing options. None if no options.