ryu Documentation

Release 3.6

ryu development team

April 10, 2020

CONTENTS

Contents:

CHAPTER

ONE

GETTING STARTED

1.1 What's Ryu

Ryu is a component-based software defined networking framework.

Ryu provides software components with well defined API that make it easy for developers to create new network management and control applications. Ryu supports various protocols for managing network devices, such as OpenFlow, Netconf, OF-config, etc. About OpenFlow, Ryu supports fully 1.0, 1.2, 1.3, 1.4 and Nicira Extensions.

All of the code is freely available under the Apache 2.0 license. Ryu is fully written in Python.

1.2 Quick Start

Installing Ryu is quite easy:

```
% pip install ryu
```

If you prefer to install Ryu from the source code:

```
% git clone git://github.com/osrg/ryu.git
% cd ryu; python ./setup.py install
```

If you want to use Ryu with OpenStack, please refer detailed documents. You can create tens of thousands of isolated virtual networks without using VLAN. The Ryu application is included in OpenStack mainline as of Essex release.

If you want to write your Ryu application, have a look at Writing ryu application document. After writing your application, just type:

```
% ryu-manager yourapp.py
```

1.3 Support

Ryu Official site is http://osrg.github.io/ryu/.

If you have any questions, suggestions, and patches, the mailing list is available at ryu-devel ML. The ML archive at Gmane is also available.

CHAPTER

TWO

WRITING YOUR RYU APPLICATION

2.1 The First Application

2.1.1 Whetting Your Appetite

If you want to manage the network gears (switches, routers, etc) at your way, you need to write your Ryu application. Your application tells Ryu how you want to manage the gears. Then Ryu configures the gears by using OpenFlow protocol, etc.

Writing Ryu application is easy. It's just Python scripts.

2.1.2 Start Writing

We show a Ryu application that make OpenFlow switches work as a dumb layer 2 switch.

Open a text editor creating a new file with the following content:

```
from ryu.base import app_manager

class L2Switch(app_manager.RyuApp):
    def __init__(self, *args, **kwargs):
        super(L2Switch, self).__init__(*args, **kwargs)
```

Ryu application is just a Python script so you can save the file with any name, extensions, and any place you want. Let's name the file 'l2.py' at your home directory.

This application does nothing useful yet, however it's a complete Ryu application. In fact, you can run this Ryu application:

```
% ryu-manager ~/l2.py
loading app /Users/fujita/l2.py
instantiating app /Users/fujita/l2.py
```

All you have to do is defining needs a new subclass of RyuApp to run your Python script as a Ryu application.

Next let's add the functionality of sending a received packet to all the ports.

```
from ryu.base import app_manager
from ryu.controller import ofp_event
from ryu.controller.handler import MAIN_DISPATCHER
from ryu.controller.handler import set_ev_cls

class L2Switch(app_manager.RyuApp):
    def __init__(self, *args, **kwargs):
        super(L2Switch, self).__init__(*args, **kwargs)
    @set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
    def packet_in_handler(self, ev):
        msg = ev.msg
        dp = msg.datapath
        ofp = dp.ofproto
        ofp_parser = dp.ofproto_parser

        actions = [ofp_parser.OFPActionOutput(ofp.OFPP_FLOOD)]
        out = ofp_parser.OFPPacketOut(
            datapath=dp, buffer_id=msg.buffer_id, in_port=msg.in_port,
            actions=actions)
        dp.send_msg(out)
```

A new method 'packet_in_handler' is added to L2Switch class. This is called when Ryu receives an OpenFlow packet_in message. The trick is 'set_ev_cls' decorator. This decorator tells Ryu when the decorated function should be called.

The first argument of the decorator indicates an event that makes function called. As you expect easily, every time Ryu gets a packet_in message, this function is called.

The second argument indicates the state of the switch. Probably, you want to ignore packet_in messages before the negotiation between Ryu and the switch finishes. Using 'MAIN_DISPATCHER' as the second argument means this function is called only after the negotiation completes.

Next let's look at the first half of the 'packet_in_handler' function.

- ev.msg is an object that represents a packet_in data structure.
- msg.dp is an object that represents a datapath (switch).
- dp.ofproto and dp.ofproto_parser are objects that represent the OpenFlow protocol that Ryu and the switch negotiated.

Ready for the second half.

- OFPActionOutput class is used with a packet_out message to specify a switch port that you want to send the packet out of. This application need a switch to send out of all the ports so OFPP_FLOOD constant is used.
- OFPPacketOut class is used to build a packet_out message.
- If you call Datapath class's send_msg method with a OpenFlow message class object, Ryu builds and send the on-wire data format to the switch.

Here, you finished implementing your first Ryu application. You are ready to run this Ryu application that does something useful.

A dumb l2 switch is too dumb? You want to implement a learning l2 switch? Move to the next step. You can learn from the existing Ryu applications at ryu/app directory and integrated tests directory.

2.2 Ryu application API

2.2.1 Ryu application programming model

Threads, events, and event queues

Ryu applications are single-threaded entities which implement various functionalities in Ryu. Events are messages between them.

Ryu applications send asynchronous events each other. Besides that, there are some Ryu-internal event sources which are not Ryu applications. One of examples of such event sources is OpenFlow controller. While an event can currently contain arbitrary python objects, it's discouraged to pass complex objects (eg. unpickleable objects) between Ryu applications.

Each Ryu application has a receive queue for events. The queue is FIFO and preserves the order of events. Each Ryu application has a thread for event processing. The thread keep draining the receive queue by dequeueing an event and calling the appropriate event handler for the event type. Because the event handler is called in the context of the event processing thread, it should be careful for blocking. I.e. while an event handler is blocked, no further events for the Ryu application will be processed.

There are kinds of events which are used to implement synchronous inter-application calls between Ryu applications. While such requests uses the same machinary as ordinary events, their replies are put on a queue dedicated to the transaction to avoid deadlock.

While threads and queues is currently implemented with eventlet/greenlet, a direct use of them in a Ryu application is strongly discouraged.

Contexts

Contexts are ordinary python objects shared among Ryu applications. The use of contexts are discouraged for new code.

2.2.2 Create a Ryu application

A Ryu application is a python module which defines a subclass of ryu.base.app_manager.RyuApp. If two or more such classes are defined in a module, the first one (by name order) will be picked by app_manager. Ryu application is singleton: only single instance of a given Ryu application is supported.

2.2.3 Observe events

A Ryu application can register itself to listen for specific events using ryu.controller.handler.set_ev_cls decorator.

2.2.4 Generate events

A Ryu application can raise events by calling appropriate ryu.base.app_manager.RyuApp's methods like send_event or send_event_to_observers.

2.2.5 Event classes

An event class describes a Ryu event generated in the system. By convention, event class names are prefixed by "Event". Events are generated either by the core part of Ryu or Ryu applications. A Ryu application can register its interest for a specific type of event by providing a handler method using ryu.controller.handler.set_ev_cls decorator.

OpenFlow event classes

ryu.controller.ofp_event module exports event classes which describe receptions of OpenFlow messages from connected switches. By convention, they are named as ryu.controller.ofp_event.EventOFPxxxx where xxxx is the name of the corresponding OpenFlow message. For example, EventOFPPacketIn for packet-in message. The OpenFlow controller part of Ryu automatically decodes OpenFlow messages received from switches and send these events to Ryu applications which expressed an interest using ryu.controller.handler.set_ev_cls. OpenFlow event classes have at least the following attributes.

Attribute	Description
msg	An object which describes the corresponding OpenFlow message.
msg.datapath	A ryu.controller.controller.Datapath instance which describes an OpenFlow switch from which we received this OpenFlow message.

The msg object has some more additional members whose values are extracted from the original OpenFlow message. See *OpenFlow protocol API Reference* for more info about OpenFlow messages.

2.2.6 ryu.base.app_manager.RyuApp

See *Ryu API Reference*.

2.2.7 ryu.controller.handler.set_ev_cls(ev_cls, dispatchers=None)

A decorator for Ryu application to declare an event handler. Decorated method will become an event handler. ev_cls is an event class whose instances this RyuApp wants to receive. dispatchers argument specifies one of the following negotiation phases (or a list of them) for which events should be generated for this handler. Note that, in case an event changes the phase, the phase before the change is used to check the interest.

Negotiation phase	Description
ryu.controller.handler.HANDSHAKE_DISPATCHER	Sending and waiting for hello message
ryu.controller.handler.CONFIG_DISPATCHER	Version negotiated and sent features-request message
ryu.controller.handler.MAIN_DISPATCHER	Switch-features message received and sent set-config message
ryu.controller.handler.DEAD_DISPATCHER	Disconnect from the peer. Or disconnecting due to some unrecoverable errors.

2.2.8 ryu.controller.controller.Datapath

A class to describe an OpenFlow switch connected to this controller. An instance has the following attributes.

Attribute	Description
id	64-bit OpenFlow Datapath ID. Only available for ryu.controller.handler.MAIN_DISPATCHER phase.
ofproto	A module which exports OpenFlow definitions, mainly constants appeared in the specification, for the negotiated OpenFlow version. For example, ryu.ofproto.ofproto_v1_0 for OpenFlow 1.0.
ofproto_parser	A module which exports OpenFlow wire message encoder and decoder for the negotiated OpenFlow version. For example, ryu.ofproto.ofproto_v1_0_parser for OpenFlow 1.0.
of-proto_parser.OFPxxxx(datapath, ...)	A callable to prepare an OpenFlow message for the given switch. It can be sent with Datapath.send_msg later. xxxx is a name of the message. For example OFPFlowMod for flow-mod message. Arguemnts depend on the message.
set_xid(self, msg)	Generate an OpenFlow XID and put it in msg.xid.
send_msg(self, msg)	Queue an OpenFlow message to send to the corresponding switch. If msg.xid is None, set_xid is automatically called on the message before queueing.
send_packet_out	deprecated
send_flow_mod	deprecated
send_flow_del	deprecated
send_delete_all_flows	deprecated
send_barrier	Queue an OpenFlow barrier message to send to the switch.
send_nxt_set_flow_format	deprecated
is_reserved_port	deprecated

2.2.9 ryu.controller.event.EventBase

The base of all event classes. A Ryu application can define its own event type by creating a subclass.

2.2.10 ryu.controller.event.EventRequestBase

The base class for synchronous request for RyuApp.send_request.

2.2.11 ryu.controller.event.EventReplyBase

The base class for synchronous request reply for RyuApp.send_reply.

2.2.12 ryu.controller.ofp_event.EventOFPStateChange

An event class for negotiation phase change notification. An instance of this class is sent to observer after changing the negotiation phase. An instance has at least the following attributes.

Attribute	Description
datapath	ryu.controller.controller.Datapath instance of the switch

2.2.13 ryu.controller.dpset.EventDP

An event class to notify connect/disconnect of a switch. For OpenFlow switches, one can get the same notification by observing ryu.controller.ofp_event.EventOFPStateChange. An instance has at least the following attributes.

Attribute	Description
dp	A ryu.controller.controller.Datapath instance of the switch
enter	True when the switch connected to our controller. False for disconnect.

2.2.14 ryu.controller.dpset.EventPortAdd

An event class for switch port status notification. This event is generated when a new port is added to a switch. For OpenFlow switches, one can get the same notification by observing ryu.controller.ofp_event.EventOFPPortStatus. An instance has at least the following attributes.

Attribute	Description
dp	A ryu.controller.controller.Datapath instance of the switch
port	port number

2.2.15 ryu.controller.dpset.EventPortDelete

An event class for switch port status notification. This event is generated when a port is removed from a switch. For OpenFlow switches, one can get the same notification by observing ryu.controller.ofp_event.EventOFPPortStatus. An instance has at least the following attributes.

Attribute	Description
dp	A ryu.controller.controller.Datapath instance of the switch
port	port number

2.2.16 ryu.controller.dpset.EventPortModify

An event class for switch port status notification. This event is generated when some attribute of a port is changed. For OpenFlow switches, one can get the same notification by observing ryu.controller.ofp_event.EventOFPPortStatus. An instance has at least the following attributes.

Attribute	Description
dp	A ryu.controller.controller.Datapath instance of the switch
port	port number

2.2.17 ryu.controller.network.EventNetworkPort

An event class for notification of port arrival and deperture. This event is generated when a port is introduced to or removed from a network by the REST API. An instance has at least the following attributes.

Attribute	Description
network_id	Network ID
dpid	OpenFlow Datapath ID of the switch to which the port belongs.
port_no	OpenFlow port number of the port
add_del	True for adding a port. False for removing a port.

2.2.18 ryu.controller.network.EventNetworkDel

An event class for network deletion. This event is generated when a network is deleted by the REST API. An instance has at least the following attributes.

Attribute	Description
network_id	Network ID

2.2.19 ryu.controller.network.EventMacAddress

An event class for end-point MAC address registration. This event is generated when a end-point MAC address is updated by the REST API. An instance has at least the following attributes.

Attribute	Description
network_id	Network ID
dpid	OpenFlow Datapath ID of the switch to which the port belongs.
port_no	OpenFlow port number of the port
mac_address	The old MAC address of the port if add_del is False. Otherwise the new MAC address.
add_del	False if this event is a result of a port removal. Otherwise True.

2.2.20 ryu.controller.tunnels.EventTunnelKeyAdd

An event class for tunnel key registration. This event is generated when a tunnel key is registered or updated by the REST API. An instance has at least the following attributes.

Attribute	Description
network_id	Network ID
tunnel_key	Tunnel Key

2.2.21 ryu.controller.tunnels.EventTunnelKeyDel

An event class for tunnel key registration. This event is generated when a tunnel key is removed by the REST API. An instance has at least the following attributes.

Attribute	Description
network_id	Network ID
tunnel_key	Tunnel Key

2.2.22 ryu.controller.tunnels.EventTunnelPort

An event class for tunnel port registration. This event is generated when a tunnel port is added or removed by the REST API. An instance has at least the following attributes.

Attribute	Description
dpid	OpenFlow Datapath ID
port_no	OpenFlow port number
remote_dpid	OpenFlow port number of the tunnel peer
add_del	True for adding a tunnel. False for removal.

2.3 Library

Ryu provides some useful library for your network applications.

2.3.1 Packet library

Introduction

Ryu packet library helps you to parse and build various protocol packets. dpkt is the popular library for the same purpose, however it is not designed to handle protocols that are interleaved; vlan, mpls, gre, etc. So we implemented our own packet library.

Network Addresses

Unless otherwise specified, MAC/IPv4/IPv6 addresses are specified using human readable strings for this library. For example, ‘08:60:6e:7f:74:e7’, ‘192.0.2.1’, ‘fe80::a60:6eff:fe7f:74e7’.

Parsing Packet

First, let’s look at how we can use the library to parse the received packets in a handler for OFPPacketIn messages.

```
from ryu.lib.packet import packet

@handler.set_ev_cls(ofp_event.EventOFPPacketIn, handler.MAIN_DISPATCHER)
def packet_in_handler(self, ev):
    pkt = packet.Packet(array.array('B', ev.msg.data))
    for p in pkt.protocols:
        print p
```

You can create a Packet class instance with the received raw data. Then the packet library parses the data and creates protocol class instances included the data. The packet class ‘protocols’ has the protocol class instances.

If a TCP packet is received, something like the following is printed:

```
<ryu.lib.packet.ethernet.ethernet object at 0x107a5d790>
<ryu.lib.packet.vlan.vlan object at 0x107a5d7d0>
<ryu.lib.packet.ipv4.ipv4 object at 0x107a5d810>
<ryu.lib.packet.tcp.tcp object at 0x107a5d850>
```

If vlan is not used, you see something like:

```
<ryu.lib.packet.ethernet.ethernet object at 0x107a5d790>
<ryu.lib.packet.ipv4.ipv4 object at 0x107a5d810>
<ryu.lib.packet.tcp.tcp object at 0x107a5d850>
```

You can access to a specific protocol class instance by using the packet class iterator. Let’s try to check VLAN id if VLAN is used:

```
from ryu.lib.packet import packet

@handler.set_ev_cls(ofp_event.EventOFPPacketIn, handler.MAIN_DISPATCHER)
def packet_in_handler(self, ev):
    pkt = packet.Packet(array.array('B', ev.msg.data))
    for p in pkt:
        print p.protocol_name, p
        if p.protocol_name == 'vlan':
            print 'vid = ', p.vid
```

You see something like:

```
ethernet <ryu.lib.packet.ethernet.ethernet object at 0x107a5d790>
vlan <ryu.lib.packet.vlan.vlan object at 0x107a5d7d0>
vid = 10
ipv4 <ryu.lib.packet.ipv4.ipv4 object at 0x107a5d810>
tcp <ryu.lib.packet.tcp.tcp object at 0x107a5d850>
```

Building Packet

You need to create protocol class instances that you want to send, add them to a packet class instance via add_protocol method, and then call serialize method. You have the raw data to send. The following example is building an arp packet.

```
e = ethernet.ethernet(dst, src, ether.ETH_TYPE_8021Q)
a = arp.arp(hwtype=1, proto=0x0800, hlen=6, plen=4, opcode=2,
            src='08:60:6e:7f:74:e7', src_ip='192.0.2.1',
            dst='00:00:00:00:00:00', dst_ip='192.0.2.2')
p = packet.Packet()
p.add_protocol(e)
p.add_protocol(a)
p.serialize()
```

2.3.2 Packet library API Reference

Packet class

class `ryu.lib.packet.packet.`**`Packet`**(*data=None*, *protocols=None*, *parse_cls=<class 'ryu.lib.packet.ethernet.ethernet'>*)

A packet decoder/encoder class.

An instance is used to either decode or encode a single packet.

data is a bytearray to describe a raw datagram to decode. When decoding, a Packet object is iteratable. Iterated values are protocol (ethernet, ipv4, ...) headers and the payload. Protocol headers are instances of subclass of packet_base.PacketBase. The payload is a bytearray. They are iterated in on-wire order.

data should be omitted when encoding a packet.

`add_protocol`(*proto*)

Register a protocol *proto* for this packet.

This method is legal only when encoding a packet.

When encoding a packet, register a protocol (ethernet, ipv4, ...) header to add to this packet. Protocol headers should be registered in on-wire order before calling self.serialize.

`get_protocol`(*protocol*)

Returns the firstly found protocol that matches to the specified protocol.

`get_protocols`(*protocol*)

Returns a list of protocols that matches to the specified protocol.

`serialize`()

Encode a packet and store the resulted bytearray in self.data.

This method is legal only when encoding a packet.

Stream Parser class

class `ryu.lib.packet.stream_parser.`**`StreamParser`**

Streaming parser base class.

An instance of a subclass of this class is used to extract messages from a raw byte stream.

It's designed to be used for data read from a transport which doesn't preserve message boundaries. A typical example of such a transport is TCP.

`parse`(*data*)
    Tries to extract messages from a raw byte stream.

    The data argument would be python bytes newly read from the input stream.

    Returns an ordered list of extracted messages. It can be an empty list.

    The rest of data which doesn't produce a complete message is kept internally and will be used when more data is come. I.e. next time this method is called again.

`try_parse`(*q*)
    Try to extract a message from the given bytes.

    This is an override point for subclasses.

    This method tries to extract a message from bytes given by the argument.

    Raises TooSmallException if the given data is not enough to extract a complete message but there's still a chance to extract a message if more data is come later.

class `ryu.lib.packet.bgp.`**`StreamParser`**
Streaming parser for BGP-4 messages.

This is a subclass of ryu.lib.packet.stream_parser.StreamParser. Its parse method returns a list of BGPMessage subclass instances.

Protocol Header classes

class `ryu.lib.packet.packet_base.`**`PacketBase`**
A base class for a protocol (ethernet, ipv4, ...) header.

classmethod **`get_packet_type`**(*type_*)
    Per-protocol dict-like get method.

    Provided for convenience of protocol implementers. Internal use only.

classmethod **`parser`**(*buf*)
    Decode a protocol header.

    This method is used only when decoding a packet.

    Decode a protocol header at offset 0 in bytearray *buf*. Returns the following three objects.

    - An object to describe the decoded header.
    - A packet_base.PacketBase subclass appropriate for the rest of the packet. None when the rest of the packet should be considered as raw payload.
    - The rest of packet.

classmethod **`register_packet_type`**(*cls_*, *type_*)
    Per-protocol dict-like set method.

    Provided for convenience of protocol implementers. Internal use only.

`serialize`(*payload*, *prev*)
    Encode a protocol header.

    This method is used only when encoding a packet.

    Encode a protocol header. Returns a bytearray which contains the header.

payload is the rest of the packet which will immediately follow this header.

prev is a packet_base.PacketBase subclass for the outer protocol header. *prev* is None if the current header is the outer-most. For example, *prev* is ipv4 or ipv6 for tcp.serialize.

class `ryu.lib.packet.ethernet.`**`ethernet`**(*dst='ff:ff:ff:ff:ff:ff'*, *src='00:00:00:00:00:00'*, *ethertype=2048*)

Ethernet header encoder/decoder class.

An instance has the following attributes at least. MAC addresses are represented as a string like '08:60:6e:7f:74:e7'. __init__ takes the corresponding args in this order.

Attribute	Description	Example
dst	destination address	'ff:ff:ff:ff:ff:ff'
src	source address	'08:60:6e:7f:74:e7'
ethertype	ether type	0x0800

classmethod **`get_packet_type`**(*type_*)

Override method for the ethernet IEEE802.3 Length/Type field (self.ethertype).

If the value of Length/Type field is less than or equal to 1500 decimal(05DC hexadecimal), it means Length interpretation and be passed to the LLC sublayer.

class `ryu.lib.packet.vlan.`**`svlan`**(*pcp=0*, *cfi=0*, *vid=0*, *ethertype=33024*)

S-VLAN (IEEE 802.1ad) 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
pcp	Priority Code Point
cfi	Canonical Format Indicator. In a case to be used as B-TAG, this field means DEI(Drop Eligible Indication).
vid	VLAN Identifier
ethertype	EtherType

class `ryu.lib.packet.vlan.`**`vlan`**(*pcp=0*, *cfi=0*, *vid=0*, *ethertype=2048*)

VLAN (IEEE 802.1Q) 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
pcp	Priority Code Point
cfi	Canonical Format Indicator
vid	VLAN Identifier
ethertype	EtherType

classmethod **`get_packet_type`**(*type_*)

Override method for the Length/Type field (self.ethertype). The Length/Type field means Length or Type interpretation, same as ethernet IEEE802.3. If the value of Length/Type field is less than or equal to 1500 decimal(05DC hexadecimal), it means Length interpretation and be passed to the LLC sublayer.

class `ryu.lib.packet.pbb.`**`itag`**(*pcp=0*, *dei=0*, *uca=0*, *sid=0*)

I-TAG (IEEE 802.1ah-2008) 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
pcp	Priority Code Point
dei	Drop Eligible Indication
uca	Use Customer Address
sid	Service Instance ID

class `ryu.lib.packet.mpls.`**`mpls`**(*label=0*, *exp=0*, *bsb=1*, *ttl=255*)

MPLS (RFC 3032) header encoder/decoder class.

NOTE: When decoding, this implementation assumes that the inner protocol is IPv4.

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
label	Label Value
exp	Experimental Use
bsb	Bottom of Stack
ttl	Time To Live

class `ryu.lib.packet.arp.`**`arp`**(*hwtype=1*, *proto=2048*, *hlen=6*, *plen=4*, *opcode=1*, *src_mac='ff:ff:ff:ff:ff:ff'*, *src_ip='0.0.0.0'*, *dst_mac='ff:ff:ff:ff:ff:ff'*, *dst_ip='0.0.0.0'*)

ARP (RFC 826) 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. IPv4 addresses are represented as a string like '192.0.2.1'. MAC addresses are represented as a string like '08:60:6e:7f:74:e7'. __init__ takes the corresponding args in this order.

Attribute	Description	Example
hwtype	ar$hrd	
proto	ar$pro	
hlen	ar$hln	
plen	ar$pln	
opcode	ar$op	
src_mac	ar$sha	'08:60:6e:7f:74:e7'
src_ip	ar$spa	'192.0.2.1'
dst_mac	ar$tha	'00:00:00:00:00:00'
dst_ip	ar$tpa	'192.0.2.2'

`ryu.lib.packet.arp.`**`arp_ip`**(*opcode*, *src_mac*, *src_ip*, *dst_mac*, *dst_ip*)

A convenient wrapper for IPv4 ARP for Ethernet.

This is an equivalent of the following code.

> arp(ARP_HW_TYPE_ETHERNET, ether.ETH_TYPE_IP, 6, 4, opcode, src_mac, src_ip, dst_mac, dst_ip)

class `ryu.lib.packet.ipv4.`**`ipv4`**(*version=4*, *header_length=5*, *tos=0*, *total_length=0*, *identification=0*, *flags=0*, *offset=0*, *ttl=255*, *proto=0*, *csum=0*, *src='0.0.0.0'*, *dst='0.0.0.0'*, *option=None*)

IPv4 (RFC 791) header encoder/decoder class.

NOTE: When decoding, this implementation tries to decode the upper layer protocol even for a fragmented datagram. It isn't likely what a user would want.

An instance has the following attributes at least. Most of them are same to the on-wire counterparts but in host byte order. IPv4 addresses are represented as a string like '192.0.2.1'. __init__ takes the corresponding args in this order.

Attribute	Description	Example
version	Version	
header_length	IHL	
tos	Type of Service	
total_length	Total Length (0 means automatically-calculate when encoding)	
identification	Identification	
flags	Flags	
offset	Fragment Offset	
ttl	Time to Live	
proto	Protocol	
csum	Header Checksum (Ignored and automatically-calculated when encoding)	
src	Source Address	'192.0.2.1'
dst	Destination Address	'192.0.2.2'
option	A bytearray which contains the entire Options, or None for no Options	

class `ryu.lib.packet.icmp.`**`TimeExceeded`**(*data_len=0*, *data=None*)

ICMP sub encoder/decoder class for Time Exceeded Message.

This is used with ryu.lib.packet.icmp.icmp for ICMP Time Exceeded Message.

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.

[RFC4884] introduced 8-bit data length attribute.

Attribute	Description
data_len	data length
data	Internet Header + leading octets of original datagram

class `ryu.lib.packet.icmp.`**`dest_unreach`**(*data_len=0*, *mtu=0*, *data=None*)

ICMP sub encoder/decoder class for Destination Unreachable Message.

This is used with ryu.lib.packet.icmp.icmp for ICMP Destination Unreachable Message.

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.

[RFC1191] reserves bits for the "Next-Hop MTU" field. [RFC4884] introduced 8-bit data length attribute.

Attribute	Description
data_len	data length
mtu	Next-Hop MTU NOTE: This field is required when icmp code is 4 code 4 = fragmentation needed and DF set
data	Internet Header + leading octets of original datagram

class `ryu.lib.packet.icmp.`**`echo`**(*id_=0*, *seq=0*, *data=None*)

ICMP sub encoder/decoder class for Echo and Echo Reply messages.

This is used with ryu.lib.packet.icmp.icmp for ICMP Echo and Echo Reply messages.

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
id	Identifier
seq	Sequence Number
data	Internet Header + 64 bits of Original Data Datagram

class `ryu.lib.packet.icmp.`**`icmp`**(*type_=8*, *code=0*, *csum=0*, *data=None*)

ICMP (RFC 792) 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
type	Type
code	Code
csum	CheckSum (0 means automatically-calculate when encoding)
data	Payload. Either a bytearray, or ryu.lib.packet.icmp.echo or ryu.lib.packet.icmp.dest_unreach or ryu.lib.packet.icmp.TimeExceeded object NOTE for icmp.echo: This includes "unused" 16 bits and the following "Internet Header + 64 bits of Original Data Datagram" of the ICMP header. NOTE for icmp.dest_unreach and icmp.TimeExceeded: This includes "unused" 8 or 24 bits and the following "Internet Header + leading octets of original datagram" of the original packet.

class `ryu.lib.packet.ipv6.`**`auth`**(*nxt=6*, *size=3*, *spi=0*, *seq=0*, *data='x00x00x00x00'*)
IP Authentication header (RFC 2402) encoder/decoder class.

This is used with ryu.lib.packet.ipv6.ipv6.

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
nxt	Next Header
size	the length of the Authentication Header in 64-bit words, subtracting 1.
spi	security parameters index.
seq	sequence number.
data	authentication data.

class `ryu.lib.packet.ipv6.`**`dst_opts`**(*nxt=6*, *size=0*, *data=None*)
IPv6 (RFC 2460) destination header encoder/decoder class.

This is used with ryu.lib.packet.ipv6.ipv6.

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
nxt	Next Header
size	the length of the destination header, not include the first 8 octet.
data	IPv6 options.

class `ryu.lib.packet.ipv6.`**`fragment`**(*nxt=6*, *offset=0*, *more=0*, *id_=0*)
IPv6 (RFC 2460) fragment header encoder/decoder class.

This is used with ryu.lib.packet.ipv6.ipv6.

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
nxt	Next Header
offset	offset, in 8-octet units, relative to the start of the fragmentable part of the original packet.
more	1 means more fragments follow; 0 means last fragment.
id_	packet identification value.

class `ryu.lib.packet.ipv6.`**`header`**(*nxt*)
extension header abstract class.

class `ryu.lib.packet.ipv6.`**`hop_opts`**(*nxt=6*, *size=0*, *data=None*)
IPv6 (RFC 2460) Hop-by-Hop Options header encoder/decoder class.

This is used with ryu.lib.packet.ipv6.ipv6.

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
nxt	Next Header
size	the length of the Hop-by-Hop Options header, not include the first 8 octet.
data	IPv6 options.

class `ryu.lib.packet.ipv6.`**`ipv6`**(*version=6*, *traffic_class=0*, *flow_label=0*, *payload_length=0*, *nxt=6*, *hop_limit=255*, *src='::'*, *dst='::'*, *ext_hdrs=None*)

IPv6 (RFC 2460) 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. IPv6 addresses are represented as a string like 'ff02::1'. __init__ takes the corresponding args in this order.

Attribute	Description	Example
version	Version	
traffic_class	Traffic Class	
flow_label	When decoding, Flow Label. When encoding, the most significant 8 bits of Flow Label.	
payload_length	Payload Length	
nxt	Next Header	
hop_limit	Hop Limit	
src	Source Address	'ff02::1'
dst	Destination Address	'::'
ext_hdrs	Extension Headers	

class `ryu.lib.packet.ipv6.`**`opt_header`**(*nxt*, *size*, *data*)

an abstract class for Hop-by-Hop Options header and destination header.

class `ryu.lib.packet.ipv6.`**`option`**(*type_=0*, *len_=-1*, *data=None*)

IPv6 (RFC 2460) Options header encoder/decoder class.

This is used with ryu.lib.packet.ipv6.hop_opts or ryu.lib.packet.ipv6.dst_opts.

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
type_	option type.
len_	the length of data. -1 if type_ is 0.
data	an option value. None if len_ is 0 or -1.

class `ryu.lib.packet.icmpv6.`**`echo`**(*id_=0*, *seq=0*, *data=None*)

ICMPv6 sub encoder/decoder class for Echo Request and Echo Reply messages.

This is used with ryu.lib.packet.icmpv6.icmpv6 for ICMPv6 Echo Request and Echo Reply messages.

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
id	Identifier
seq	Sequence Number
data	Data

class `ryu.lib.packet.icmpv6.`**`icmpv6`**(*type_=0*, *code=0*, *csum=0*, *data=None*)

ICMPv6 (RFC 2463) 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
type_	Type
code	Code
csum	CheckSum (0 means automatically-calculate when encoding)
data	Payload. ryu.lib.packet.icmpv6.echo object, ryu.lib.packet.icmpv6.nd_neighbor object, ryu.lib.packet.icmpv6.nd_router_solicit object, ryu.lib.packet.icmpv6.nd_router_advert object, ryu.lib.packet.icmpv6.mld object, or a bytearray.

class `ryu.lib.packet.icmpv6.`**`mld`**(*maxresp=0*, *address='::'*)

ICMPv6 sub encoder/decoder class for MLD Lister Query, MLD Listener Report, and MLD Listener Done messages. (RFC 2710)

http://www.ietf.org/rfc/rfc2710.txt

This is used with ryu.lib.packet.icmpv6.icmpv6.

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
maxresp	max response time in millisecond. it is meaningful only in Query Message.
address	a group address value.

class `ryu.lib.packet.icmpv6.`**`mldv2_query`**(*maxresp=0*, *address='::'*, *s_flg=0*, *qrv=2*, *qqic=0*, *num=0*, *srcs=None*)

ICMPv6 sub encoder/decoder class for MLD v2 Lister Query messages. (RFC 3810)

http://www.ietf.org/rfc/rfc3810.txt

This is used with ryu.lib.packet.icmpv6.icmpv6.

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
maxresp	max response time in millisecond. it is meaningful only in Query Message.
address	a group address value.
s_flg	when set to 1, routers suppress the timer process.
qrv	robustness variable for a querier.
qqic	an interval time for a querier in unit of seconds.
num	a number of the multicast servers.
srcs	a list of IPv6 addresses of the multicast servers.

class `ryu.lib.packet.icmpv6.`**`mldv2_report`**(*record_num=0*, *records=None*)

ICMPv6 sub encoder/decoder class for MLD v2 Lister Report messages. (RFC 3810)

http://www.ietf.org/rfc/rfc3810.txt

This is used with ryu.lib.packet.icmpv6.icmpv6.

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
record_num	a number of the group records.
records	a list of ryu.lib.packet.icmpv6.mldv2_report_group. None if no records.

class `ryu.lib.packet.icmpv6.`**`mldv2_report_group`**(*type_=0*, *aux_len=0*, *num=0*, *address='::'*, *srcs=None*, *aux=None*)

ICMPv6 sub encoder/decoder class for MLD v2 Lister Report Group Record messages. (RFC 3810)

This is used with ryu.lib.packet.icmpv6.mldv2_report.

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
type_	a group record type for v3.
aux_len	the length of the auxiliary data in 32-bit words.
num	a number of the multicast servers.
address	a group address value.
srcs	a list of IPv6 addresses of the multicast servers.
aux	the auxiliary data.

class `ryu.lib.packet.icmpv6.`**`nd_neighbor`**(*res=0*, *dst='::'*, *option=None*)

ICMPv6 sub encoder/decoder class for Neighbor Solicitation and Neighbor Advertisement messages. (RFC 4861)

This is used with ryu.lib.packet.icmpv6.icmpv6.

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
res	R,S,O Flags for Neighbor Advertisement. The 3 MSBs of "Reserved" field for Neighbor Solicitation.
dst	Target Address
option	a derived object of ryu.lib.packet.icmpv6.nd_option or a bytearray. None if no options.

class `ryu.lib.packet.icmpv6.`**`nd_option_pi`**(*length=0*, *pl=0*, *res1=0*, *val_l=0*, *pre_l=0*, *res2=0*, *prefix='::'*)

ICMPv6 sub encoder/decoder class for Neighbor discovery Prefix Information Option. (RFC 4861)

This is used with ryu.lib.packet.icmpv6.nd_router_advert.

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
length	length of the option. (0 means automatically-calculate when encoding)
pl	Prefix Length.
res1	L,A,R* Flags for Prefix Information.
val_l	Valid Lifetime.
pre_l	Preferred Lifetime.
res2	This field is unused. It MUST be initialized to zero.
prefix	An IP address or a prefix of an IP address.

*R flag is defined in (RFC 3775)

class `ryu.lib.packet.icmpv6.`**`nd_option_sla`**(*length=0*, *hw_src='00:00:00:00:00:00'*, *data=None*)

ICMPv6 sub encoder/decoder class for Neighbor discovery Source Link-Layer Address Option. (RFC 4861)

This is used with ryu.lib.packet.icmpv6.nd_neighbor, ryu.lib.packet.icmpv6.nd_router_solicit or ryu.lib.packet.icmpv6.nd_router_advert.

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
length	length of the option. (0 means automatically-calculate when encoding)
hw_src	Link-Layer Address. NOTE: If the address is longer than 6 octets this contains the first 6 octets in the address. This implementation assumes the address has at least 6 octets.
data	A bytearray which contains the rest of Link-Layer Address and padding. When encoding a packet, it's user's responsibility to provide necessary padding for 8-octets alignment required by the protocol.

class `ryu.lib.packet.icmpv6.`**`nd_option_tla`**(*length=0*, *hw_src='00:00:00:00:00:00'*, *data=None*)

ICMPv6 sub encoder/decoder class for Neighbor discovery Target Link-Layer Address Option. (RFC 4861)

This is used with ryu.lib.packet.icmpv6.nd_neighbor.

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
length	length of the option. (0 means automatically-calculate when encoding)
hw_src	Link-Layer Address. NOTE: If the address is longer than 6 octets this contains the first 6 octets in the address. This implementation assumes the address has at least 6 octets.
data	A bytearray which contains the rest of Link-Layer Address and padding. When encoding a packet, it's user's responsibility to provide necessary padding for 8-octets alignment required by the protocol.

class `ryu.lib.packet.icmpv6.`**`nd_router_advert`**(*ch_l=0*, *res=0*, *rou_l=0*, *rea_t=0*, *ret_t=0*, *options=None*)

ICMPv6 sub encoder/decoder class for Router Advertisement messages. (RFC 4861)

This is used with ryu.lib.packet.icmpv6.icmpv6.

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
ch_l	Cur Hop Limit.
res	M,O Flags for Router Advertisement.
rou_l	Router Lifetime.
rea_t	Reachable Time.
ret_t	Retrans Timer.
options	List of a derived object of ryu.lib.packet.icmpv6.nd_option or a bytearray. None if no options.

class `ryu.lib.packet.icmpv6.`**`nd_router_solicit`**(*res=0*, *option=None*)

ICMPv6 sub encoder/decoder class for Router Solicitation messages. (RFC 4861)

This is used with ryu.lib.packet.icmpv6.icmpv6.

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
res	This field is unused. It MUST be initialized to zero.
option	a derived object of ryu.lib.packet.icmpv6.nd_option or a bytearray. None if no options.

class `ryu.lib.packet.tcp.`**`tcp`**(*src_port=0*, *dst_port=0*, *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	An bytearray containing Options and following Padding. None if no options.

class `ryu.lib.packet.udp.`**`udp`**(*src_port=0*, *dst_port=0*, *total_length=0*, *csum=0*)

UDP (RFC 768) 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
total_length	Length (0 means automatically-calculate when encoding)
csum	Checksum (0 means automatically-calculate when encoding)

class `ryu.lib.packet.dhcp.`**`dhcp`**(*op*, *chaddr*, *options*, *htype=1*, *hlen=0*, *hops=0*, *xid=None*, *secs=0*, *flags=0*, *ciaddr='0.0.0.0'*, *yiaddr='0.0.0.0'*, *siaddr='0.0.0.0'*, *giaddr='0.0.0.0'*, *sname=''*, *boot_file=''*)

DHCP (RFC 2131) header encoder/decoder class.

The serialized packet would looks like the ones described in the following sections.

- RFC 2131 DHCP packet format

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
op	Message op code / message type. 1 = BOOTREQUEST, 2 = BOOTREPLY
htype	Hardware address type (e.g. '1' = 10mb ethernet).
hlen	Hardware address length (e.g. '6' = 10mb ethernet).
hops	Client sets to zero, optionally used by relay agent when booting via a relay agent.
xid	Transaction ID, a random number chosen by the client, used by the client and serverto associate messages and responses between a client and a server.
secs	Filled in by client, seconds elapsed since client began address acquisition or renewal process.
flags	Flags.
ciaddr	Client IP address; only filled in if client is in BOUND, RENEW or REBINDING state and can respond to ARP requests.
yiaddr	'your' (client) IP address.
siaddr	IP address of next server to use in bootstrap; returned in DHCPOFFER, DHCPACK by server.
giaddr	Relay agent IP address, used in booting via a relay agent.
chaddr	Client hardware address.
sname	Optional server host name, null terminated string.
boot_file	Boot file name, null terminated string; "generic" name or null in DHCPDISCOVER, fully qualified directory-path name in DHCPOFFER.
options	Optional parameters field ('DHCP message type' option must be included in every DHCP message).

class `ryu.lib.packet.dhcp.`**`options`**(*option_list=None*, *options_len=0*, *magic_cookie='99.130.83.99'*)

DHCP (RFC 2132) options encoder/decoder class.

This is used with ryu.lib.packet.dhcp.dhcp.

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
option_list	'end option' and 'pad option' are added automatically after the option class is stored in array.
options_len	Option's byte length. ('magic cookie', 'end option' and 'pad option' length including.)
magic_cookie	The first four octets contain the decimal values 99, 130, 83 and 99.

class `ryu.lib.packet.dhcp.`**`option`**(*tag*, *value*, *length=0*)

DHCP (RFC 2132) options encoder/decoder class.

This is used with ryu.lib.packet.dhcp.dhcp.options.

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
tag	Option type. (except for the 'magic cookie', 'pad option' and 'end option'.)
value	Option's value. (set the value that has been converted to hexadecimal.)
length	Option's value length. (calculated automatically from the length of value.)

class `ryu.lib.packet.vrrp.`**`vrrp`**(*version*, *type_*, *vrid*, *priority*, *count_ip*, *max_adver_int*, *checksum*, *ip_addresses*, *auth_type=None*, *auth_data=None*)

The base class for VRRPv2 (RFC 3768) and VRRPv3 (RFC 5798) header encoder/decoder classes.

Unlike other ryu.lib.packet.packet_base.PacketBase derived classes, This class should not be directly instantiated by user.

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

Attribute	Description
version	Version
type	Type
vrid	Virtual Rtr ID (VRID)
priority	Priority
count_ip	Count IPvX Addr. Calculated automatically when encoding.
max_adver_int	Maximum Advertisement Interval (Max Adver Int)
checksum	Checksum. Calculated automatically when encoding.
ip_addresses	IPvX Address(es). A python list of IP addresses.
auth_type	Authentication Type (only for VRRPv2)
auth_data	Authentication Data (only for VRRPv2)

`create_packet`(*primary_ip_address*, *vlan_id=None*)

Prepare a VRRP packet.

Returns a newly created ryu.lib.packet.packet.Packet object with appropriate protocol header objects added by add_protocol(). It's caller's responsibility to serialize(). The serialized packet would looks like the ones described in the following sections.

- RFC 3768 5.1. VRRP Packet Format
- RFC 5798 5.1. VRRP Packet Format

Argument	Description
primary_ip_address	Source IP address
vlan_id	VLAN ID. None for no VLAN.

class `ryu.lib.packet.vrrp.`**`vrrpv2`**(*version*, *type_*, *vrid*, *priority*, *count_ip*, *max_adver_int*, *checksum*, *ip_addresses*, *auth_type=None*, *auth_data=None*)

VRRPv2 (RFC 3768) header encoder/decoder class.

Unlike other ryu.lib.packet.packet_base.PacketBase derived classes, *create* method should be used to instantiate an object of this class.

static **`create`**(*type_*, *vrid*, *priority*, *max_adver_int*, *ip_addresses*)

Unlike other ryu.lib.packet.packet_base.PacketBase derived classes, this method should be used to instantiate an object of this class.

This method's arguments are same as ryu.lib.packet.vrrp.vrrp object's attributes of the same name. (except that *type_* corresponds to *type* attribute.)

class `ryu.lib.packet.vrrp.`**`vrrpv3`**(*version*, *type_*, *vrid*, *priority*, *count_ip*, *max_adver_int*, *checksum*, *ip_addresses*, *auth_type=None*, *auth_data=None*)

VRRPv3 (RFC 5798) header encoder/decoder class.

Unlike other ryu.lib.packet.packet_base.PacketBase derived classes, *create* method should be used to instantiate an object of this class.

static **`create`**(*type_*, *vrid*, *priority*, *max_adver_int*, *ip_addresses*)

Unlike other ryu.lib.packet.packet_base.PacketBase derived classes, this method should be used to instantiate an object of this class.

This method's arguments are same as ryu.lib.packet.vrrp.vrrp object's attributes of the same name. (except that *type_* corresponds to *type* attribute.)

class `ryu.lib.packet.slow.`**`slow`**

Slow Protocol header decoder class. This class has only the parser method.

http://standards.ieee.org/getieee802/download/802.3-2012_section5.pdf

Slow Protocols Subtypes

Subtype Value	Protocol Name
0	Unused - Illegal Value
1	Link Aggregation Control Protocol(LACP)
2	Link Aggregation - Marker Protocol
3	Operations, Administration, and Maintenance(OAM)
4 - 9	Reserved for future use
10	Organization Specific Slow Protocol(OSSP)
11 - 255	Unused - Illegal values

class `ryu.lib.packet.slow.`**`lacp`**(*version=1*, *actor_system_priority=0*, *actor_system='00:00:00:00:00:00'*, *actor_key=0*, *actor_port_priority=0*, *actor_port=0*, *actor_state_activity=0*, *actor_state_timeout=0*, *actor_state_aggregation=0*, *actor_state_synchronization=0*, *actor_state_collecting=0*, *actor_state_distributing=0*, *actor_state_defaulted=0*, *actor_state_expired=0*, *partner_system_priority=0*, *partner_system='00:00:00:00:00:00'*, *partner_key=0*, *partner_port_priority=0*, *partner_port=0*, *partner_state_activity=0*, *partner_state_timeout=0*, *partner_state_aggregation=0*, *partner_state_synchronization=0*, *partner_state_collecting=0*, *partner_state_distributing=0*, *partner_state_defaulted=0*, *partner_state_expired=0*, *collector_max_delay=0*)

Link Aggregation Control Protocol(LACP, IEEE 802.1AX) header encoder/decoder class.

http://standards.ieee.org/getieee802/download/802.1AX-2008.pdf

LACPDU format

LACPDU structure		**Octets**
Subtype = LACP		1
Version Number		1
TLV Actor	TLV_type = Actor Information	1
	Actor_Information_Length = 20	1
	Actor_System_Priority	2
	Actor_System	6
	Actor_Key	2
	Actor_Port_Priority	2
	Actor_Port	2
	Actor_State	1
	Reserved	3
TLV Partner	TLV_type = Partner Information	1
	Partner_Information_Length = 20	1
	Partner_System_Priority	2
	Partner_System	6
	Partner_Key	2
	Partner_Port_Priority	2
	Partner_Port	2
	Partner_State	1
	Reserved	3
TLV Collector	TLV_type = Collector Information	1
	Collector_Information_Length = 16	1
	Collector_Max_Delay	2
	Reserved	12
TLV Terminator	TLV_type = Terminator	1
	Terminator_Length = 0	1
	Reserved	50

Terminator information uses a length value of 0 (0x00).

NOTE–The use of a Terminator_Length of 0 is intentional. In TLV encoding schemes it is common practice for the terminator encoding to be 0 both for the type and the length.

Actor_State and Partner_State encoded as individual bits within a single octet as follows:

7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EXPR	DFLT	DIST	CLCT	SYNC	AGGR	TMO	ACT

ACT bit 0. about the activity control value with regard to this link.

TMO bit 1. about the timeout control value with regard to this link.

AGGR bit 2. about how the system regards this link from the point of view of the aggregation.

SYNC bit 3. about how the system regards this link from the point of view of the synchronization.

CLCT bit 4. about collecting of incoming frames.

DIST bit 5. about distributing of outgoing frames.

DFLT bit 6. about the opposite system information which the system use.

EXPR bit 7. about the expire state of the system.

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
version	LACP version. This parameter must be set to LACP_VERSION_NUMBER(i.e. 1).
actor_system_priority	The priority assigned to this System.
actor_system	The Actor's System ID, encoded as a MAC address.
actor_key	The operational Key value assigned to the port by the Actor.
actor_port_priority	The priority assigned to this port.
actor_port	The port number assigned to the port by the Actor.

actor_state_activity	about the activity control value with regard to this link.
	LACP_STATE_ACTIVE(1)
	LACP_STATE_PASSIVE(0)

actor_state_timeout	about the timeout control value with regard to this link.
	LACP_STATE_SHORT_TIMEOUT(1)
	LACP_STATE_LONG_TIMEOUT(0)

actor_state_aggregation	about how the system regards this link from the point of view of the aggregation.
	LACP_STATE_AGGREGATEABLE(1)
	LACP_STATE_INDIVIDUAL(0)

actor_state_synchronization	about how the system regards this link from the point of view of the synchronization.
	LACP_STATE_IN_SYNC(1)
	LACP_STATE_OUT_OF_SYNC(0)

actor_state_collecting	about collecting of incoming frames.
	LACP_STATE_COLLECTING_ENABLED(1)
	LACP_STATE_COLLECTING_DISABLED(0)

actor_state_distributing	about distributing of outgoing frames.
	LACP_STATE_DISTRIBUTING_ENABLED(1)
	LACP_STATE_DISTRIBUTING_DISABLED(0)

actor_state_defaulted	about the Partner information which the the Actor use.
	LACP_STATE_DEFAULTED_PARTNER(1)
	LACP_STATE_OPERATIONAL_PARTNER(0)

actor_state_expired	about the state of the Actor.
	LACP_STATE_EXPIRED(1)
	LACP_STATE_NOT_EXPIRED(0)

partner_system_priority	The priority assigned to the Partner System.
partner_system	The Partner's System ID, encoded as a MAC address.
partner_key	The operational Key value assigned to the port by the Partner.
partner_port_priority	The priority assigned to this port by the Partner.
partner_port	The port number assigned to the port by the Partner.
partner_state_activity	See *actor_state_activity*.
partner_state_timeout	See *actor_state_timeout*.
partner_state_aggregation	See *actor_state_aggregation*.
partner_state_synchronization	See *actor_state_synchronization*.
partner_state_collecting	See *actor_state_collecting*.
partner_state_distributing	See *actor_state_distributing*.
partner_state_defaulted	See *actor_state_defaulted*.
partner_state_expired	See *actor_state_expired*.
collector_max_delay	the maximum time that the Frame Collector may delay.

class `ryu.lib.packet.llc.`**`llc`**(*dsap_addr*, *ssap_addr*, *control*)
LLC(IEEE 802.2) 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
dsap_addr	Destination service access point address field includes I/G bit at least significant bit.
ssap_addr	Source service access point address field includes C/R bit at least significant bit.
control	Control field [16 bits for formats that include sequence numbering, and 8 bits for formats that do not]. Either ryu.lib.packet.llc.ControlFormatI or ryu.lib.packet.llc.ControlFormatS or ryu.lib.packet.llc.ControlFormatU object.

class `ryu.lib.packet.llc.`**`ControlFormatI`**(*send_sequence_number=0*, *pf_bit=0*, *receive_sequence_number=0*)
LLC sub encoder/decoder class for control I-format field.

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
send_sequence_number	sender send sequence number
pf_bit	poll/final bit
receive_sequence_number	sender receive sequence number

class `ryu.lib.packet.llc.`**`ControlFormatS`**(*supervisory_function=0*, *pf_bit=0*, *receive_sequence_number=0*)
LLC sub encoder/decoder class for control S-format field.

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
supervisory_function	supervisory function bit
pf_bit	poll/final bit
receive_sequence_number	sender receive sequence number

class `ryu.lib.packet.llc.`**`ControlFormatU`**(*modifier_function1=0*, *pf_bit=0*, *modifier_function2=0*)
LLC sub encoder/decoder class for control U-format field.

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
modifier_function1	modifier function bit
pf_bit	poll/final bit
modifier_function2	modifier function bit

class `ryu.lib.packet.bpdu.`**`bpdu`**
Bridge Protocol Data Unit(BPDU) header encoder/decoder base class.

class `ryu.lib.packet.bpdu.`**`ConfigurationBPDUs`**(*flags=0*, *root_priority=32768*, *root_system_id_extension=0*, *root_mac_address='00:00:00:00:00:00'*, *root_path_cost=0*, *bridge_priority=32768*, *bridge_system_id_extension=0*, *bridge_mac_address='00:00:00:00:00:00'*, *port_priority=128*, *port_number=0*, *message_age=0*, *max_age=20*, *hello_time=2*, *forward_delay=15*)

Configuration BPDUs(IEEE 802.1D) 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
flags	Bit 1: Topology Change flag Bits 2 through 7: unused and take the value 0 Bit 8: Topology Change Acknowledgment flag
root_priority	Root Identifier priority set 0-61440 in steps of 4096
root_system_id_extension	Root Identifier system ID extension
root_mac_address	Root Identifier MAC address
root_path_cost	Root Path Cost
bridge_priority	Bridge Identifier priority set 0-61440 in steps of 4096
bridge_system_id_extension	Bridge Identifier system ID extension
bridge_mac_address	Bridge Identifier MAC address
port_priority	Port Identifier priority set 0-240 in steps of 16
port_number	Port Identifier number
message_age	Message Age timer value
max_age	Max Age timer value
hello_time	Hello Time timer value
forward_delay	Forward Delay timer value

class `ryu.lib.packet.bpdu.`**`TopologyChangeNotificationBPDUs`**

Topology Change Notification BPDUs(IEEE 802.1D) header encoder/decoder class.

class `ryu.lib.packet.bpdu.`**`RstBPDUs`** (*flags=0*, *root_priority=32768*, *root_system_id_extension=0*, *root_mac_address='00:00:00:00:00:00'*, *root_path_cost=0*, *bridge_priority=32768*, *bridge_system_id_extension=0*, *bridge_mac_address='00:00:00:00:00:00'*, *port_priority=128*, *port_number=0*, *message_age=0*, *max_age=20*, *hello_time=2*, *forward_delay=15*)

Rapid Spanning Tree BPDUs(RST BPDUs, IEEE 802.1D) 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
flags	Bit 1: Topology Change flag Bit 2: Proposal flag Bits 3 and 4: Port Role Bit 5: Learning flag Bit 6: Forwarding flag Bit 7: Agreement flag Bit 8: Topology Change Acknowledgment flag
root_priority	Root Identifier priority set 0-61440 in steps of 4096
root_system_id_extension	Root Identifier system ID extension
root_mac_address	Root Identifier MAC address
root_path_cost	Root Path Cost
bridge_priority	Bridge Identifier priority set 0-61440 in steps of 4096
bridge_system_id_extension	Bridge Identifier system ID extension
bridge_mac_address	Bridge Identifier MAC address
port_priority	Port Identifier priority set 0-240 in steps of 16
port_number	Port Identifier number
message_age	Message Age timer value
max_age	Max Age timer value
hello_time	Hello Time timer value
forward_delay	Forward Delay timer value

class `ryu.lib.packet.igmp.`**`igmp`**(*msgtype=17*, *maxresp=0*, *csum=0*, *address='0.0.0.0'*)

Internet Group Management Protocol(IGMP, RFC 1112, RFC 2236) header encoder/decoder class.

http://www.ietf.org/rfc/rfc1112.txt

http://www.ietf.org/rfc/rfc2236.txt

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
msgtype	a message type for v2, or a combination of version and a message type for v1.
maxresp	max response time in unit of 1/10 second. it is meaningful only in Query Message.
csum	a check sum value. 0 means automatically-calculate when encoding.
address	a group address value.

class `ryu.lib.packet.igmp.`**`igmpv3_query`**(*msgtype=17*, *maxresp=100*, *csum=0*, *address='0.0.0.0'*, *s_flg=0*, *qrv=2*, *qqic=0*, *num=0*, *srcs=None*)

Internet Group Management Protocol(IGMP, RFC 3376) Membership Query message encoder/decoder class.

http://www.ietf.org/rfc/rfc3376.txt

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
msgtype	a message type for v3.
maxresp	max response time in unit of 1/10 second.
csum	a check sum value. 0 means automatically-calculate when encoding.
address	a group address value.
s_flg	when set to 1, routers suppress the timer process.
qrv	robustness variable for a querier.
qqic	an interval time for a querier in unit of seconds.
num	a number of the multicast servers.
srcs	a list of IPv4 addresses of the multicast servers.

class `ryu.lib.packet.igmp.`**`igmpv3_report`**(*msgtype=34*, *csum=0*, *record_num=0*, *records=None*)

Internet Group Management Protocol(IGMP, RFC 3376) Membership Report message encoder/decoder class.

http://www.ietf.org/rfc/rfc3376.txt

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
msgtype	a message type for v3.
csum	a check sum value. 0 means automatically-calculate when encoding.
record_num	a number of the group records.
records	a list of ryu.lib.packet.igmp.igmpv3_report_group. None if no records.

class `ryu.lib.packet.igmp.`**`igmpv3_report_group`**(*type_=0*, *aux_len=0*, *num=0*, *address='0.0.0.0'*, *srcs=None*, *aux=None*)

Internet Group Management Protocol(IGMP, RFC 3376) Membership Report Group Record message encoder/decoder class.

http://www.ietf.org/rfc/rfc3376.txt

This is used with ryu.lib.packet.igmp.igmpv3_report.

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
type_	a group record type for v3.
aux_len	the length of the auxiliary data.
num	a number of the multicast servers.
address	a group address value.
srcs	a list of IPv4 addresses of the multicast servers.
aux	the auxiliary data.

class `ryu.lib.packet.bgp.`**`BGPMessage`**(*type_*, *len_=None*, *marker=None*)

Base class for BGP-4 messages.

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
marker	Marker field. Ignored when encoding.
len	Length field. Ignored when encoding.
type	Type field. one of BGP_MSG_ constants.

class `ryu.lib.packet.bgp.`**`BGPOpen`**(*my_as*, *bgp_identifier*, *type_=1*, *opt_param_len=0*, *opt_param=[ ]*, *version=4*, *hold_time=0*, *len_=None*, *marker=None*)

BGP-4 OPEN Message 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
marker	Marker field. Ignored when encoding.
len	Length field. Ignored when encoding.
type	Type field. The default is BGP_MSG_OPEN.
version	Version field. The default is 4.
my_as	My Autonomous System field. 2 octet unsigned integer.
hold_time	Hold Time field. 2 octet unsigned integer. The default is 0.
bgp_identifier	BGP Identifier field. An IPv4 address. For example, '192.0.2.1'
opt_param_len	Optional Parameters Length field. Ignored when encoding.
opt_param	Optional Parameters field. A list of BGPOptParam instances. The default is [].

class `ryu.lib.packet.bgp.`**`BGPUpdate`**(*type_=2*, *withdrawn_routes_len=None*, *withdrawn_routes=[ ]*, *total_path_attribute_len=None*, *path_attributes=[ ]*, *nlri=[ ]*, *len_=None*, *marker=None*)

BGP-4 UPDATE Message 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
marker	Marker field. Ignored when encoding.
len	Length field. Ignored when encoding.
type	Type field. The default is BGP_MSG_UPDATE.
with-drawn_routes_len	Withdrawn Routes Length field. Ignored when encoding.
withdrawn_routes	Withdrawn Routes field. A list of BGPWithdrawnRoute instances. The default is [].
to-tal_path_attribute_len	Total Path Attribute Length field. Ignored when encoding.
path_attributes	Path Attributes field. A list of BGPPathAttribute instances. The default is [].
nlri	Network Layer Reachability Information field. A list of BGPNLRI instances. The default is [].

class `ryu.lib.packet.bgp.`**`BGPKeepAlive`**(*type_=4*, *len_=None*, *marker=None*)

BGP-4 KEEPALIVE Message 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
marker	Marker field. Ignored when encoding.
len	Length field. Ignored when encoding.
type	Type field. The default is BGP_MSG_KEEPALIVE.

class `ryu.lib.packet.bgp.`**`BGPNotification`**(*error_code*, *error_subcode*, *data=''*, *type_=3*, *len_=None*, *marker=None*)

BGP-4 NOTIFICATION Message 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
marker	Marker field. Ignored when encoding.
len	Length field. Ignored when encoding.
type	Type field. The default is BGP_MSG_NOTIFICATION.
error_code	Error code field.
error_subcode	Error subcode field.
data	Data field. The default is ‘’.

class `ryu.lib.packet.sctp.`**`cause_cookie_while_shutdown`**(*length=0*)

Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for Cookie Received While Shutting Down (RFC 4960).

This is used with ryu.lib.packet.sctp.chunk_abort and ryu.lib.packet.sctp.chunk_error.

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
length	length of this cause containing this header. (0 means automatically-calculate when encoding)

class `ryu.lib.packet.sctp.`**`cause_invalid_param`**(*length=0*)

Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for Invalid Mandatory Parameter (RFC 4960).

This is used with ryu.lib.packet.sctp.chunk_abort and ryu.lib.packet.sctp.chunk_error.

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
length	length of this cause containing this header. (0 means automatically-calculate when encoding)

class `ryu.lib.packet.sctp.`**`cause_invalid_stream_id`**(*value=0*, *length=0*)

Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for Invalid Stream Identifier (RFC 4960).

This is used with ryu.lib.packet.sctp.chunk_abort and ryu.lib.packet.sctp.chunk_error.

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
value	stream id.
length	length of this cause containing this header. (0 means automatically-calculate when encoding)

class `ryu.lib.packet.sctp.`**`cause_missing_param`**(*types=None*, *num=0*, *length=0*)

Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for Missing Mandatory Parameter (RFC 4960).

This is used with ryu.lib.packet.sctp.chunk_abort and ryu.lib.packet.sctp.chunk_error.

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
types	a list of missing params.
num	Number of missing params. (0 means automatically-calculate when encoding)
length	length of this cause containing this header. (0 means automatically-calculate when encoding)

class `ryu.lib.packet.sctp.`**`cause_no_userdata`**(*value=None*, *length=0*)

Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for No User Data (RFC 4960).

This is used with ryu.lib.packet.sctp.chunk_abort and ryu.lib.packet.sctp.chunk_error.

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
value	the TSN of the DATA chunk received with no user data field.
length	length of this cause containing this header. (0 means automatically-calculate when encoding)

class `ryu.lib.packet.sctp.`**`cause_out_of_resource`**(*length=0*)
Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for Out of Resource (RFC 4960).

This is used with ryu.lib.packet.sctp.chunk_abort and ryu.lib.packet.sctp.chunk_error.

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
length	length of this cause containing this header. (0 means automatically-calculate when encoding)

class `ryu.lib.packet.sctp.`**`cause_protocol_violation`**(*value=None*, *length=0*)
Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for Protocol Violation (RFC 4960).

This is used with ryu.lib.packet.sctp.chunk_abort and ryu.lib.packet.sctp.chunk_error.

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
value	Additional Information.
length	length of this cause containing this header. (0 means automatically-calculate when encoding)

class `ryu.lib.packet.sctp.`**`cause_restart_with_new_addr`**(*value=None*, *length=0*)
Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for Restart of an Association with New Addresses (RFC 4960).

This is used with ryu.lib.packet.sctp.chunk_abort and ryu.lib.packet.sctp.chunk_error.

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
value	New Address TLVs.
length	length of this cause containing this header. (0 means automatically-calculate when encoding)

class `ryu.lib.packet.sctp.`**`cause_stale_cookie`**(*value=None*, *length=0*)
Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for Stale Cookie Error (RFC 4960).

This is used with ryu.lib.packet.sctp.chunk_abort and ryu.lib.packet.sctp.chunk_error.

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
value	Measure of Staleness.
length	length of this cause containing this header. (0 means automatically-calculate when encoding)

class `ryu.lib.packet.sctp.`**`cause_unrecognized_chunk`**(*value=None*, *length=0*)
Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for Unrecognized Chunk Type (RFC 4960).

This is used with ryu.lib.packet.sctp.chunk_abort and ryu.lib.packet.sctp.chunk_error.

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
value	Unrecognized Chunk.
length	length of this cause containing this header. (0 means automatically-calculate when encoding)

class `ryu.lib.packet.sctp.`**`cause_unrecognized_param`**(*value=None*, *length=0*)

Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for Unrecognized Parameters (RFC 4960).

This is used with ryu.lib.packet.sctp.chunk_abort and ryu.lib.packet.sctp.chunk_error.

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
value	Unrecognized Parameter.
length	length of this cause containing this header. (0 means automatically-calculate when encoding)

class `ryu.lib.packet.sctp.`**`cause_unresolvable_addr`**(*value=None*, *length=0*)

Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for Unresolvable Address (RFC 4960).

This is used with ryu.lib.packet.sctp.chunk_abort and ryu.lib.packet.sctp.chunk_error.

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
value	Unresolvable Address. one of follows: ryu.lib.packet.sctp.param_host_addr, ryu.lib.packet.sctp.param_ipv4, or ryu.lib.packet.sctp.param_ipv6.
length	length of this cause containing this header. (0 means automatically-calculate when encoding)

class `ryu.lib.packet.sctp.`**`cause_user_initiated_abort`**(*value=None*, *length=0*)

Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for User-Initiated Abort (RFC 4960).

This is used with ryu.lib.packet.sctp.chunk_abort and ryu.lib.packet.sctp.chunk_error.

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
value	Upper Layer Abort Reason.
length	length of this cause containing this header. (0 means automatically-calculate when encoding)

class `ryu.lib.packet.sctp.`**`chunk_abort`**(*tflag=0*, *length=0*, *causes=None*)

Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for Abort Association (ABORT) chunk (RFC 4960).

This is used with ryu.lib.packet.sctp.sctp.

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
tflag	'0' means the Verification tag is normal. '1' means the Verification tag is copy of the sender.
length	length of this chunk containing this header. (0 means automatically-calculate when encoding)
causes	a list of derived classes of ryu.lib.packet.sctp.causes.

class `ryu.lib.packet.sctp.`**`chunk_cookie_ack`**(*flags=0*, *length=0*)

Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for Cookie Acknowledgement (COOKIE ACK) chunk (RFC 4960).

This is used with ryu.lib.packet.sctp.sctp.

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
flags	set to '0'. this field will be ignored.
length	length of this chunk containing this header. (0 means automatically-calculate when encoding)

class `ryu.lib.packet.sctp.`**`chunk_cookie_echo`** (*flags=0*, *length=0*, *cookie=None*)

Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for Cookie Echo (COOKIE ECHO) chunk (RFC 4960).

This is used with ryu.lib.packet.sctp.sctp.

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
flags	set to '0'. this field will be ignored.
length	length of this chunk containing this header. (0 means automatically-calculate when encoding)
cookie	cookie data.

class `ryu.lib.packet.sctp.`**`chunk_cwr`** (*flags=0*, *length=0*, *low_tsn=0*)

Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for CWR chunk (RFC 4960 Appendix A.).

This is used with ryu.lib.packet.sctp.sctp.

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
flags	set to '0'. this field will be ignored.
length	length of this chunk containing this header. (0 means automatically-calculate when encoding)
low_tsn	the lowest TSN.

class `ryu.lib.packet.sctp.`**`chunk_data`** (*unordered=0*, *begin=0*, *end=0*, *length=0*, *tsn=0*, *sid=0*, *seq=0*, *payload_id=0*, *payload_data=None*)

Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for Payload Data (DATA) chunk (RFC 4960).

This is used with ryu.lib.packet.sctp.sctp.

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
unordered	if set to '1', the receiver ignores the sequence number.
begin	if set to '1', this chunk is the first fragment.
end	if set to '1', this chunk is the last fragment.
length	length of this chunk containing this header. (0 means automatically-calculate when encoding)
tsn	Transmission Sequence Number.
sid	stream id.
seq	the sequence number.
payload_id	application specified protocol id. '0' means that no application id is identified.
payload_data	user data.

class `ryu.lib.packet.sctp.`**`chunk_ecn_echo`** (*flags=0*, *length=0*, *low_tsn=0*)

Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for ECN-Echo chunk (RFC 4960 Appendix A.).

This is used with ryu.lib.packet.sctp.sctp.

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
flags	set to '0'. this field will be ignored.
length	length of this chunk containing this header. (0 means automatically-calculate when encoding)
low_tsn	the lowest TSN.

class `ryu.lib.packet.sctp.`**`chunk_error`** (*flags=0*, *length=0*, *causes=None*)

Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for Operation Error (ERROR) chunk (RFC 4960).

This is used with ryu.lib.packet.sctp.sctp.

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
flags	set to '0'. this field will be ignored.
length	length of this chunk containing this header. (0 means automatically-calculate when encoding)
causes	a list of derived classes of ryu.lib.packet.sctp.causes.

class `ryu.lib.packet.sctp.`**`chunk_heartbeat`** (*flags=0*, *length=0*, *info=None*)

Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for Heartbeat Request (HEARTBEAT) chunk (RFC 4960).

This is used with ryu.lib.packet.sctp.sctp.

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
flags	set to '0'. this field will be ignored.
length	length of this chunk containing this header. (0 means automatically-calculate when encoding)
info	ryu.lib.packet.sctp.param_heartbeat.

class `ryu.lib.packet.sctp.`**`chunk_heartbeat_ack`** (*flags=0*, *length=0*, *info=None*)

Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for Heartbeat Acknowledgement (HEARTBEAT ACK) chunk (RFC 4960).

This is used with ryu.lib.packet.sctp.sctp.

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
flags	set to '0'. this field will be ignored.
length	length of this chunk containing this header. (0 means automatically-calculate when encoding)
info	ryu.lib.packet.sctp.param_heartbeat.

class `ryu.lib.packet.sctp.`**`chunk_init`** (*flags=0*, *length=0*, *init_tag=0*, *a_rwnd=0*, *os=0*, *mis=0*, *i_tsn=0*, *params=None*)

Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for Initiation (INIT) chunk (RFC 4960).

This is used with ryu.lib.packet.sctp.sctp.

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
flags	set to ‘0’. this field will be ignored.
length	length of this chunk containing this header. (0 means automatically-calculate when encoding)
init_tag	the tag that be used as Verification Tag.
a_rwnd	Advertised Receiver Window Credit.
os	number of outbound streams.
mis	number of inbound streams.
i_tsn	Transmission Sequence Number that the sender will use.
params	Optional/Variable-Length Parameters. a list of derived classes of ryu.lib.packet.sctp.param.

class `ryu.lib.packet.sctp.`**`chunk_init_ack`** (*flags=0*, *length=0*, *init_tag=0*, *a_rwnd=0*, *os=0*, *mis=0*, *i_tsn=0*, *params=None*)

Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for Initiation Acknowledgement (INIT ACK) chunk (RFC 4960).

This is used with ryu.lib.packet.sctp.sctp.

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
flags	set to ‘0’. this field will be ignored.
length	length of this chunk containing this header. (0 means automatically-calculate when encoding)
init_tag	the tag that be used as Verification Tag.
a_rwnd	Advertised Receiver Window Credit.
os	number of outbound streams.
mis	number of inbound streams.
i_tsn	Transmission Sequence Number that the sender will use.
params	Optional/Variable-Length Parameters. a list of derived classes of ryu.lib.packet.sctp.param.

class `ryu.lib.packet.sctp.`**`chunk_sack`** (*flags=0*, *length=0*, *tsn_ack=0*, *a_rwnd=0*, *gapack_num=0*, *duptsn_num=0*, *gapacks=None*, *duptsns=None*)

Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for Selective Acknowledgement (SACK) chunk (RFC 4960).

This is used with ryu.lib.packet.sctp.sctp.

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
flags	set to ‘0’. this field will be ignored.
length	length of this chunk containing this header. (0 means automatically-calculate when encoding)
tsn_ack	TSN of the last DATA chunk received in sequence before a gap.
a_rwnd	Advertised Receiver Window Credit.
gapack_num	number of Gap Ack blocks.
duptsn_num	number of duplicate TSNs.
gapacks	a list of Gap Ack blocks. one block is made of a list with the start offset and the end offset from tsn_ack. e.g.) gapacks = [[2, 3], [10, 12], [19, 21]]
duptsns	a list of duplicate TSN.

class `ryu.lib.packet.sctp.`**`chunk_shutdown`** (*flags=0*, *length=0*, *tsn_ack=0*)

Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for Shutdown Association (SHUTDOWN) chunk (RFC 4960).

This is used with ryu.lib.packet.sctp.sctp.

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
flags	set to '0'. this field will be ignored.
length	length of this chunk containing this header. (0 means automatically-calculate when encoding)
tsn_ack	TSN of the last DATA chunk received in sequence before a gap.

class `ryu.lib.packet.sctp.`**`chunk_shutdown_ack`**(*flags=0*, *length=0*)

Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for Shutdown Acknowledgement (SHUTDOWN ACK) chunk (RFC 4960).

This is used with ryu.lib.packet.sctp.sctp.

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
flags	set to '0'. this field will be ignored.
length	length of this chunk containing this header. (0 means automatically-calculate when encoding)

class `ryu.lib.packet.sctp.`**`chunk_shutdown_complete`**(*tflag=0*, *length=0*)

Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for Shutdown Complete (SHUTDOWN COMPLETE) chunk (RFC 4960).

This is used with ryu.lib.packet.sctp.sctp.

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
tflag	'0' means the Verification tag is normal. '1' means the Verification tag is copy of the sender.
length	length of this chunk containing this header. (0 means automatically-calculate when encoding)

class `ryu.lib.packet.sctp.`**`param_cookie_preserve`**(*value=0*, *length=0*)

Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for Cookie Preservative Parameter (RFC 4960).

This is used with ryu.lib.packet.sctp.chunk_init.

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
value	Suggested Cookie Life-Span Increment (msec).
length	length of this param containing this header. (0 means automatically-calculate when encoding)

class `ryu.lib.packet.sctp.`**`param_ecn`**(*value=None*, *length=0*)

Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for ECN Parameter (RFC 4960 Appendix A.).

This is used with ryu.lib.packet.sctp.chunk_init and ryu.lib.packet.sctp.chunk_init_ack.

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
value	set to None.
length	length of this param containing this header. (0 means automatically-calculate when encoding)

class `ryu.lib.packet.sctp.`**`param_heartbeat`**(*value=None*, *length=0*)

Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for Heartbeat Info Parameter (RFC 4960).

This is used with ryu.lib.packet.sctp.chunk_heartbeat and ryu.lib.packet.sctp.chunk_heartbeat_ack.

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
value	the sender-specific heartbeat information.
length	length of this param containing this header. (0 means automatically-calculate when encoding)

class `ryu.lib.packet.sctp.`**`param_host_addr`**(*value=None*, *length=0*)
Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for Host Name Address Parameter (RFC 4960).

This is used with ryu.lib.packet.sctp.chunk_init and ryu.lib.packet.sctp.chunk_init_ack.

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
value	a host name that ends with null terminator.
length	length of this param containing this header. (0 means automatically-calculate when encoding)

class `ryu.lib.packet.sctp.`**`param_ipv4`**(*value='127.0.0.1'*, *length=0*)
Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for IPv4 Address Parameter (RFC 4960).

This is used with ryu.lib.packet.sctp.chunk_init and ryu.lib.packet.sctp.chunk_init_ack.

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
value	IPv4 address of the sending endpoint.
length	length of this param containing this header. (0 means automatically-calculate when encoding)

class `ryu.lib.packet.sctp.`**`param_ipv6`**(*value='::1'*, *length=0*)
Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for IPv6 Address Parameter (RFC 4960).

This is used with ryu.lib.packet.sctp.chunk_init and ryu.lib.packet.sctp.chunk_init_ack.

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
value	IPv6 address of the sending endpoint.
length	length of this param containing this header. (0 means automatically-calculate when encoding)

class `ryu.lib.packet.sctp.`**`param_state_cookie`**(*value=None*, *length=0*)
Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for State Cookie Parameter (RFC 4960).

This is used with ryu.lib.packet.sctp.chunk_init_ack.

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
value	the state cookie. see Section 5.1.3 in RFC 4960.
length	length of this param containing this header. (0 means automatically-calculate when encoding)

class `ryu.lib.packet.sctp.`**`param_supported_addr`**(*value=None*, *length=0*)

Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for Supported Address Types Parameter (RFC 4960).

This is used with ryu.lib.packet.sctp.chunk_init.

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
value	a list of parameter types. odd cases pad with 0x0000.
length	length of this param containing this header. (0 means automatically-calculate when encoding)

class `ryu.lib.packet.sctp.`**`param_unrecognized_param`**(*value=None*, *length=0*)

Stream Control Transmission Protocol (SCTP) sub encoder/decoder class for Unrecognized Parameter (RFC 4960).

This is used with ryu.lib.packet.sctp.chunk_init_ack.

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
value	the unrecognized parameter in the INIT chunk.
length	length of this param containing this header. (0 means automatically-calculate when encoding)

class `ryu.lib.packet.sctp.`**`sctp`**(*src_port=0*, *dst_port=0*, *vtag=0*, *csum=0*, *chunks=None*)

Stream Control Transmission Protocol (SCTP) header encoder/decoder class (RFC 4960).

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
vtag	Verification Tag
csum	Checksum (0 means automatically-calculate when encoding)
chunks	a list of derived classes of ryu.lib.packet.sctp.chunk.

2.3.3 OF-Config support

Ryu has a library for OF-Config support.

XML schema files for NETCONFIG and OFConfig

XML schema files for NETCONF and OFConfig are stolen from LINC whose licence is Apache 2.0. It supports only part of OFConfig so that its schema files are (intentionally) limited such that operation attributes are allowed only in several limited places. Once our library is tested with other OFConfig switches, the schema files should be updated to allow operation attribute in more places.

References

- NETCONF ietf,
- NETCONF ietf wiki,
- OF-Config spec,
- ncclient,
- ncclient repo,

- LINC git repo

2.4 OpenFlow protocol API Reference

2.4.1 OpenFlow version independent classes and functions

Base class for OpenFlow messages

class `ryu.ofproto.ofproto_parser.`**`MsgBase`**`(`**args, **kwargs*`)`

This is a base class for OpenFlow message classes.

An instance of this class has at least the following attributes.

Attribute	Description
datapath	A ryu.controller.controller.Datapath instance for this message
version	OpenFlow protocol version
msg_type	Type of OpenFlow message
msg_len	Length of the message
xid	Transaction id
buf	Raw data

`_TYPE`

_TYPE class attribute is used to annotate types of attributes.

This type information is used to find an appropriate conversion for a JSON style dictionary.

Currently the following types are implemented.

Type	Descrption
ascii	US-ASCII
utf-8	UTF-8

Example:

```
_TYPE = {
    'ascii': [
        'hw_addr',
    ],
    'utf-8': [
        'name',
    ]
}
```

`to_jsondict``(`*encode_string=<function b64encode at 0x152dd70>*`)`

This method returns a JSON style dict to describe this object.

The returned dict is compatible with json.dumps() and json.loads().

Suppose ClassName object inherits StringifyMixin. For an object like the following:

```
ClassName(Param1=100, Param2=200)
```

this method would produce:

```
{ "ClassName": {"Param1": 100, "Param2": 200} }
```

This method takes the following arguments.

Argument	Description
encode_string	(Optional) specify how to encode attributes which has python 'str' type. The default is base64. This argument is used only for attributes which don't have explicit type annotations in _TYPE class attribute.

classmethod **`from_jsondict`**(*dict_*, *decode_string=<function b64decode at 0x152dde8>*, ***additional_args*)

Create an instance from a JSON style dict.

Instantiate this class with parameters specified by the dict.

This method takes the following arguments.

Argument	Descrpition
dict_	A dictionary which describes the parameters. For example, {"Param1": 100, "Param2": 200}
decode_string	(Optional) specify how to decode strings. The default is base64. This argument is used only for attributes which don't have explicit type annotations in _TYPE class attribute.
additional_args	(Optional) Additional kwargs for constructor.

Functions

`ryu.ofproto.ofproto_parser.`**`ofp_msg_from_jsondict`**(*dp*, *jsondict*)

This function instanticates an appropriate OpenFlow message class from the given JSON style dictionary. The objects created by following two code fragments are equivalent.

Code A:

```
jsonstr = '{ "OFPSetConfig": { "flags": 0, "miss_send_len": 128 } }'
jsondict = json.loads(jsonstr)
o = ofp_msg_from_jsondict(dp, jsondict)
```

Code B:

```
o = dp.ofproto_parser.OFPSetConfig(flags=0, miss_send_len=128)
```

This function takes the following arguments.

Argument	Description
dp	An instance of ryu.controller.Datapath.
jsondict	A JSON style dict.

2.4.2 OpenFlow v1.2 Messages and Structures

Controller-to-Switch Messages

Handshake

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPFeaturesRequest`**(*datapath*)

Features request message

The controller sends a feature request to the switch upon session establishment.

This message is handled by the Ryu framework, so the Ryu application do not need to process this typically.

Example:

```
def send_features_request(self, datapath):
    ofp_parser = datapath.ofproto_parser

    req = ofp_parser.OFPFeaturesRequest(datapath)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPFeaturesRequest": {}
}
```

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPSwitchFeatures`**(*datapath*, *datapath_id=None*, *n_buffers=None*, *n_tables=None*, *capabilities=None*, *ports=None*)

Features reply message

The switch responds with a features reply message to a features request.

This message is handled by the Ryu framework, so the Ryu application do not need to process this typically.

Example:

```
@set_ev_cls(ofp_event.EventOFPSwitchFeatures, CONFIG_DISPATCHER)
def switch_features_handler(self, ev):
    msg = ev.msg

    self.logger.debug('OFPSwitchFeatures received: '
                      'datapath_id=0x%016x n_buffers=%d '
                      'n_tables=%d capabilities=0x%08x ports=%s',
                      msg.datapath_id, msg.n_buffers, msg.n_tables,
                      msg.capabilities, msg.ports)
```

JSON Example:

```
{
   "OFPSwitchFeatures": {
      "capabilities": 79,
      "datapath_id": 9210263729383,
      "n_buffers": 0,
      "n_tables": 255,
      "ports": {
         "6": {
            "OFPPort": {
               "advertised": 10240,
               "config": 0,
               "curr": 10248,
               "curr_speed": 5000,
               "hw_addr": "f2:0b:a4:7d:f8:ea",
               "max_speed": 5000,
               "name": "Port6",
               "peer": 10248,
               "port_no": 6,
               "state": 4,
               "supported": 10248
            }
         },
         "7": {
            "OFPPort": {
```

```
                        "advertised": 10240,
                        "config": 0,
                        "curr": 10248,
                        "curr_speed": 5000,
                        "hw_addr": "f2:0b:a4:d0:3f:70",
                        "max_speed": 5000,
                        "name": "Port7",
                        "peer": 10248,
                        "port_no": 7,
                        "state": 4,
                        "supported": 10248
                    }
                }
            }
        }
    }
```

Switch Configuration

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPSetConfig`**(*datapath*, *flags=0*, *miss_send_len=0*)

Set config request message

The controller sends a set config request message to set configuraion parameters.

Attribute	Description
flags	One of the following configuration flags. OFPC_FRAG_NORMAL OFPC_FRAG_DROP OFPC_FRAG_REASM OFPC_FRAG_MASK OFPC_INVALID_TTL_TO_CONTROLLER
miss_send_len	Max bytes of new flow that datapath should send to the controller

Example:

```
def send_set_config(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    req = ofp_parser.OFPSetConfig(datapath, ofp.OFPC_FRAG_NORMAL, 256)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPSetConfig": {
      "flags": 0,
      "miss_send_len": 128
   }
}
```

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPGetConfigRequest`**(*datapath*)

Get config request message

The controller sends a get config request to query configuration parameters in the switch.

Example:

```
def send_get_config_request(self, datapath):
    ofp_parser = datapath.ofproto_parser

    req = ofp_parser.OFPGetConfigRequest(datapath)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPGetConfigRequest": {}
}
```

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPGetConfigReply`**(*datapath*, *flags=None*, *miss_send_len=None*)

Get config reply message

The switch responds to a configuration request with a get config reply message.

Attribute	Description
flags	One of the following configuration flags. OFPC_FRAG_NORMAL OFPC_FRAG_DROP OFPC_FRAG_REASM OFPC_FRAG_MASK OFPC_INVALID_TTL_TO_CONTROLLER
miss_send_len	Max bytes of new flow that datapath should send to the controller

Example:

```
@set_ev_cls(ofp_event.EventOFPGetConfigReply, MAIN_DISPATCHER)
def get_config_reply_handler(self, ev):
    msg = ev.msg
    dp = msg.datapath
    ofp = dp.ofproto

    if msg.flags == ofp.OFPC_FRAG_NORMAL:
        flags = 'NORMAL'
    elif msg.flags == ofp.OFPC_FRAG_DROP:
        flags = 'DROP'
    elif msg.flags == ofp.OFPC_FRAG_REASM:
        flags = 'REASM'
    elif msg.flags == ofp.OFPC_FRAG_MASK:
        flags = 'MASK'
    elif msg.flags == ofp.OFPC_INVALID_TTL_TO_CONTROLLER:
        flags = 'INVALID TTL TO CONTROLLER'
    else:
        flags = 'unknown'
    self.logger.debug('OFPGetConfigReply received: '
                      'flags=%s miss_send_len=%d',
                      flags, msg.miss_send_len)
```

JSON Example:

```
{
   "OFPGetConfigReply": {
      "flags": 0,
      "miss_send_len": 128
   }
}
```

Flow Table Configuration

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPTableMod`**(*datapath*, *table_id*, *config*)

Flow table configuration message

The controller sends this message to configure table state.

Attribute	Description
table_id	ID of the table (OFPTT_ALL indicates all tables)
config	Bitmap of the following flags. OFPTC_TABLE_MISS_CONTROLLER OFPTC_TABLE_MISS_CONTINUE OFPTC_TABLE_MISS_DROP OFPTC_TABLE_MISS_MASK

Example:

```
def send_table_mod(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    req = ofp_parser.OFPTableMod(datapath, ofp.OFPTT_ALL,
                                 ofp.OFPTC_TABLE_MISS_DROP)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPTableMod": {
      "config": 0,
      "table_id": 255
   }
}
```

Modify State Messages

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPFlowMod`**(*datapath*, *cookie=0*, *cookie_mask=0*, *table_id=0*, *command=0*, *idle_timeout=0*, *hard_timeout=0*, *priority=0*, *buffer_id=4294967295*, *out_port=0*, *out_group=0*, *flags=0*, *match=None*, *instructions=[ ]*)

Modify Flow entry message

The controller sends this message to modify the flow table.

Attribute	Description
cookie	Opaque controller-issued identifier
cookie_mask	Mask used to restrict the cookie bits that must match when the command is `OPFFC_MODIFY*` or `OFPFC_DELETE*`
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 port
out_group	For `OFPFC_DELETE*` commands, require matching entries to include this as an output group
flags	One of the following values. OFPFF_SEND_FLOW_REM OFPFF_CHECK_OVERLAP OFPFF_RESET_COUNTS
match	Instance of `OFPMatch`
instructions	list of `OFPInstruction*` instance

Example:

```python
def send_flow_mod(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    cookie = cookie_mask = 0
    table_id = 0
    idle_timeout = hard_timeout = 0
    priority = 32768
    buffer_id = ofp.OFP_NO_BUFFER
    match = ofp_parser.OFPMatch(in_port=1, eth_dst='ff:ff:ff:ff:ff:ff')
    actions = [ofp_parser.OFPActionOutput(ofp.OFPP_NORMAL, 0)]
    inst = [ofp_parser.OFPInstructionActions(ofp.OFPIT_APPLY_ACTIONS,
                                             actions)]
    req = ofp_parser.OFPFlowMod(datapath, cookie, cookie_mask,
                                table_id, ofp.OFPFC_ADD,
                                idle_timeout, hard_timeout,
                                priority, buffer_id,
                                ofp.OFPP_ANY, ofp.OFPG_ANY,
                                ofp.OFPFF_SEND_FLOW_REM,
                                match, inst)
    datapath.send_msg(req)
```

JSON Example:

```json
{
   "OFPFlowMod": {
      "buffer_id": 65535,
      "command": 0,
      "cookie": 0,
      "cookie_mask": 0,
      "flags": 0,
      "hard_timeout": 0,
      "idle_timeout": 0,
      "instructions": [
         {
            "OFPInstructionActions": {
                    "actions": [
                        {
                            "OFPActionSetField": {
                                "field": {
                                    "OXMTlv": {
                                        "field": "vlan_vid",
                                        "mask": null,
                                        "value": 258
                                    }
                                }
                            }
                        },
                        {
                            "OFPActionOutput": {
                                "len": 16,
                                "max_len": 65535,
                                "port": 6,
                                "type": 0
                            }
                        }
                    ],
                    "len": 40,
                    "type": 3
                }
            },
            {
                "OFPInstructionActions": {
                    "actions": [
                        {
                            "OFPActionSetField": {
                                "field": {
                                    "OXMTlv": {
                                        "field": "eth_src",
                                        "mask": null,
                                        "value": "01:02:03:04:05:06"
                                    }
                                }
                            }
                        }
                    ],
                    "len": 24,
                    "type": 4
                }
            }
        ],
        "match": {
            "OFPMatch": {
                "length": 14,
                "oxm_fields": [
                    {
                        "OXMTlv": {
                            "field": "eth_dst",
                            "mask": null,
                            "value": "f2:0b:a4:7d:f8:ea"
                        }
                    }
                ],
                "type": 1
                }
            },
            "out_group": 4294967295,
            "out_port": 4294967295,
            "priority": 123,
            "table_id": 1
        }
    }

    {
        "OFPFlowMod": {
            "buffer_id": 65535,
            "command": 0,
            "cookie": 0,
            "cookie_mask": 0,
            "flags": 0,
            "hard_timeout": 0,
            "idle_timeout": 0,
            "instructions": [
                {
                    "OFPInstructionGotoTable": {
                        "len": 8,
                        "table_id": 1,
                        "type": 1
                    }
                }
            ],
            "match": {
                "OFPMatch": {
                    "length": 22,
                    "oxm_fields": [
                        {
                            "OXMTlv": {
                                "field": "in_port",
                                "mask": null,
                                "value": 6
                            }
                        },
                        {
                            "OXMTlv": {
                                "field": "eth_src",
                                "mask": null,
                                "value": "f2:0b:a4:7d:f8:ea"
                            }
                        }
                    ],
                    "type": 1
                }
            },
            "out_group": 4294967295,
            "out_port": 4294967295,
            "priority": 123,
            "table_id": 0
        }
    }
```

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPGroupMod`**(*datapath*, *command*, *type_*, *group_id*, *buckets*)

Modify group entry message

The controller sends this message to modify the group table.

Attribute	Description
command	One of the following values. OFPGC_ADD OFPGC_MODIFY OFPGC_DELETE
type	One of the following values. OFPGT_ALL OFPGT_SELECT OFPGT_INDIRECT OFPGT_FF
group_id	Group identifier
buckets	list of `OFPBucket`

`type` attribute corresponds to `type_` parameter of __init__.

Example:

```python
def send_group_mod(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    port = 1
    max_len = 2000
    actions = [ofp_parser.OFPActionOutput(port, max_len)]

    weight = 100
    watch_port = 0
    watch_group = 0
    buckets = [ofp_parser.OFPBucket(weight, watch_port, watch_group,
                                    actions)]

    group_id = 1
    req = ofp_parser.OFPGroupMod(datapath, ofp.OFPGC_ADD,
                                 ofp.OFPGT_SELECT, group_id, buckets)
    datapath.send_msg(req)
```

JSON Example:

```json
{
   "OFPGroupMod": {
      "buckets": [
         {
            "OFPBucket": {
               "actions": [
                  {
                     "OFPActionOutput": {
                        "len": 16,
                        "max_len": 65535,
                        "port": 2,
                        "type": 0
                     }
                  }
               ],
               "len": 32,
               "watch_group": 1,
               "watch_port": 1,
               "weight": 1
            }
         }
      ],
      "command": 0,
      "group_id": 1,
      "type": 0
   }
}
```

class ryu.ofproto.ofproto_v1_2_parser.**OFPPortMod**(*datapath*, *port_no*, *hw_addr*, *config*, *mask*, *advertise*)

Port modification message

The controller sneds this message to modify the behavior of the port.

Attribute	Description
port_no	Port number to modify
hw_addr	The hardware address that must be the same as hw_addr of `OFPPort` of `OFPSwitchFeatures`
config	Bitmap of configuration flags. OFPPC_PORT_DOWN OFPPC_NO_RECV OFPPC_NO_FWD OFPPC_NO_PACKET_IN
mask	Bitmap of configuration flags above to be changed
advertise	Bitmap of the following flags. OFPPF_10MB_HD OFPPF_10MB_FD OFPPF_100MB_HD OFPPF_100MB_FD OFPPF_1GB_HD OFPPF_1GB_FD OFPPF_10GB_FD OFPPF_40GB_FD OFPPF_100GB_FD OFPPF_1TB_FD OFPPF_OTHER OFPPF_COPPER OFPPF_FIBER OFPPF_AUTONEG OFPPF_PAUSE OFPPF_PAUSE_ASYM

Example:

```
def send_port_mod(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    port_no = 3
    hw_addr = 'fa:c8:e8:76:1d:7e'
    config = 0
    mask = (ofp.OFPPC_PORT_DOWN | ofp.OFPPC_NO_RECV |
            ofp.OFPPC_NO_FWD | ofp.OFPPC_NO_PACKET_IN)
    advertise = (ofp.OFPPF_10MB_HD | ofp.OFPPF_100MB_FD |
                 ofp.OFPPF_1GB_FD | ofp.OFPPF_COPPER |
                 ofp.OFPPF_AUTONEG | ofp.OFPPF_PAUSE |
                 ofp.OFPPF_PAUSE_ASYM)
    req = ofp_parser.OFPPortMod(datapath, port_no, hw_addr, config,
                                mask, advertise)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPPortMod": {
      "advertise": 4096,
      "config": 0,
      "hw_addr": "00-11-00-00-11-11",
      "mask": 0,
      "port_no": 1
   }
}
```

Read State Messages

class ryu.ofproto.ofproto_v1_2_parser.**OFPDescStatsRequest**(*datapath*, *flags=0*)

Description statistics request message

The controller uses this message to query description of the switch.

Attribute	Description
flags	Zero (none yet defined in the spec)

Example:

```
def send_desc_stats_request(self, datapath):
    ofp_parser = datapath.ofproto_parser

    req = ofp_parser.OFPDescStatsRequest(datapath)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPDescStatsRequest": {
      "flags": 0
   }
}
```

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPDescStats`**

Description statistics reply message

The switch responds with a stats reply that include this message to a description statistics request.

Attribute	Description
mfr_desc	Manufacturer description
hw_desc	Hardware description
sw_desc	Software description
serial_num	Serial number
dp_desc	Human readable description of datapath

Example:

```
@set_ev_cls(ofp_event.EventOFPStatsReply, MAIN_DISPATCHER)
def stats_reply_handler(self, ev):
    msg = ev.msg
    ofp = msg.datapath.ofproto
    body = ev.msg.body

    if msg.type == ofp.OFPST_DESC:
        self.desc_stats_reply_handler(body)

def desc_stats_reply_handler(self, body):
    self.logger.debug('DescStats: mfr_desc=%s hw_desc=%s sw_desc=%s '
                      'serial_num=%s dp_desc=%s',
                      body.mfr_desc, body.hw_desc, body.sw_desc,
                      body.serial_num, body.dp_desc)
```

JSON Example:

```
{
   "OFPStatsReply": {
      "body": {
         "OFPDescStats": {
            "dp_desc": "dp",
            "hw_desc": "hw",
            "mfr_desc": "mfr",
            "serial_num": "serial",
            "sw_desc": "sw"
         }
      },
      "flags": 0,
      "type": 0
   }
}
```

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPFlowStatsRequest`**(*datapath*, *table_id=255*, *out_port=4294967295*, *out_group=4294967295*, *cookie=0*, *cookie_mask=0*, *match=None*, *flags=0*)

Individual flow statistics request message

The controller uses this message to query individual flow statistics.

Attribute	Description
table_id	ID of table to read
out_port	Require matching entries to include this as an output port
out_group	Require matching entries to include this as an output group
cookie	Require matching entries to contain this cookie value
cookie_mask	Mask used to restrict the cookie bits that must match
match	Instance of `OFPMatch`
flags	Zero (none yet defined in the spec)

Example:

```
def send_flow_stats_request(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    cookie = cookie_mask = 0
    match = ofp_parser.OFPMatch(in_port=1)
    req = ofp_parser.OFPFlowStatsRequest(datapath,
                                         ofp.OFPTT_ALL,
                                         ofp.OFPP_ANY, ofp.OFPG_ANY,
                                         cookie, cookie_mask, match)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPFlowStatsRequest": {
      "cookie": 0,
      "cookie_mask": 0,
      "flags": 0,
      "match": {
         "OFPMatch": {
            "length": 4,
            "oxm_fields": [],
            "type": 1
         }
      },
      "out_group": 4294967295,
      "out_port": 4294967295,
      "table_id": 0
   }
}
```

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPFlowStats`**(*table_id*, *duration_sec*, *duration_nsec*, *priority*, *idle_timeout*, *hard_timeout*, *cookie*, *packet_count*, *byte_count*, *match*, *instructions=None*, *length=None*)

Individual flow statistics reply message

The switch responds with a stats reply that include this message to an individual flow statistics request.

Attribute	Description
table_id	ID of table flow came from
duration_sec	Time flow has been alive in seconds
duration_nsec	Time flow has been alive in nanoseconds beyond duration_sec
priority	Priority of the entry
idle_timeout	Number of seconds idle before expiration
hard_timeout	Number of seconds before expiration
cookie	Opaque controller-issued identifier
packet_count	Number of packets in flow
byte_count	Number of bytes in flow
match	Instance of `OFPMatch`
instructions	list of `OFPInstruction*` instance

Example:

```
@set_ev_cls(ofp_event.EventOFPStatsReply, MAIN_DISPATCHER)
def stats_reply_handler(self, ev):
    msg = ev.msg
    ofp = msg.datapath.ofproto
    body = ev.msg.body

    if msg.type == ofp.OFPST_FLOW:
         self.flow_stats_reply_handler(body)

def flow_stats_reply_handler(self, body):
    flows = []
    for stat in body:
        flows.append('table_id=%s '
                     'duration_sec=%d duration_nsec=%d '
                     'priority=%d '
                     'idle_timeout=%d hard_timeout=%d '
                     'cookie=%d packet_count=%d byte_count=%d '
                     'match=%s instructions=%s' %
                     (stat.table_id,
                      stat.duration_sec, stat.duration_nsec,
                      stat.priority,
                      stat.idle_timeout, stat.hard_timeout,
                      stat.cookie, stat.packet_count, stat.byte_count,
                      stat.match, stat.instructions))
    self.logger.debug('FlowStats: %s', flows)
```

JSON Example:

```
{
   "OFPStatsReply": {
      "body": [
         {
            "OFPFlowStats": {
               "byte_count": 0,
               "cookie": 0,
               "duration_nsec": 115277000,
               "duration_sec": 358,
               "hard_timeout": 0,
               "idle_timeout": 0,
               "instructions": [],
               "length": 56,
               "match": {
                  "OFPMatch": {
                     "length": 4,
                     "oxm_fields": [],
                     "type": 1
                  }
               },
               "packet_count": 0,
               "priority": 65535,
               "table_id": 0
            }
         },
         {
            "OFPFlowStats": {
               "byte_count": 0,
               "cookie": 0,
               "duration_nsec": 115055000,
               "duration_sec": 358,
               "hard_timeout": 0,
               "idle_timeout": 0,
               "instructions": [
                  {
                     "OFPInstructionActions": {
                        "actions": [
                           {
                              "OFPActionOutput": {
                                 "len": 16,
                                 "max_len": 0,
                                 "port": 4294967290,
                                 "type": 0
                              }
                           }
                        ],
                        "len": 24,
                        "type": 4
                     }
                  }
               ],
               "length": 88,
               "match": {
                  "OFPMatch": {
                     "length": 10,
                     "oxm_fields": [
                        {
                           "OXMTlv": {
                              "field": "eth_type",
                              "mask": null,
                              "value": 2054
                           }
                        }
                     ],
                     "type": 1
                  }
               },
               "packet_count": 0,
               "priority": 65534,
               "table_id": 0
            }
         },
         {
```

```
    "OFPFlowStats": {
       "byte_count": 238,
       "cookie": 0,
       "duration_nsec": 511582000,
       "duration_sec": 316220,
       "hard_timeout": 0,
       "idle_timeout": 0,
       "instructions": [
          {
             "OFPInstructionGotoTable": {
                "len": 8,
                "table_id": 1,
                "type": 1
             }
          }
       ],
       "length": 80,
       "match": {
          "OFPMatch": {
             "length": 22,
             "oxm_fields": [
                {
                   "OXMTlv": {
                      "field": "in_port",
                      "mask": null,
                      "value": 6
                   }
                },
                {
                   "OXMTlv": {
                      "field": "eth_src",
                      "mask": null,
                      "value": "f2:0b:a4:7d:f8:ea"
                   }
                }
             ],
             "type": 1
          }
       },
       "packet_count": 3,
       "priority": 123,
       "table_id": 0
    }
 },
 {
    "OFPFlowStats": {
       "byte_count": 98,
       "cookie": 0,
       "duration_nsec": 980901000,
       "duration_sec": 313499,
       "hard_timeout": 0,
       "idle_timeout": 0,
       "instructions": [
          {
             "OFPInstructionActions": {
                "actions": [
                   {
                      "OFPActionOutput": {
```

```
                                                "len": 16,
                                                "max_len": 65535,
                                                "port": 4294967293,
                                                "type": 0
                                            }
                                        }
                                    ],
                                    "len": 24,
                                    "type": 3
                                }
                            }
                        ],
                        "length": 80,
                        "match": {
                            "OFPMatch": {
                                "length": 4,
                                "oxm_fields": [],
                                "type": 1
                            }
                        },
                        "packet_count": 1,
                        "priority": 0,
                        "table_id": 0
                    }
                }
            ],
            "flags": 0,
            "type": 1
        }
    }
```

class ryu.ofproto.ofproto_v1_2_parser.**OFPAggregateStatsRequest**(*datapath, table_id=255, out_port=4294967295, out_group=4294967295, cookie=0, cookie_mask=0, match=None, flags=0*)

Aggregate flow statistics request message

The controller uses this message to query aggregate flow statictics.

Attribute	Description
table_id	ID of table to read
out_port	Require matching entries to include this as an output port
out_group	Require matching entries to include this as an output group
cookie	Require matching entries to contain this cookie value
cookie_mask	Mask used to restrict the cookie bits that must match
match	Instance of `OFPMatch`
flags	Zero (none yet defined in the spec)

Example:

```
def send_aggregate_stats_request(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser
    cookie = cookie_mask = 0
    match = ofp_parser.OFPMatch(in_port=1)
    req = ofp_parser.OFPAggregateStatsRequest(datapath, 0,
                                              ofp.OFPTT_ALL,
                                              ofp.OFPP_ANY,
                                              ofp.OFPG_ANY,
                                              cookie, cookie_mask,
                                              match)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPAggregateStatsRequest": {
      "cookie": 0,
      "cookie_mask": 0,
      "flags": 0,
      "match": {
         "OFPMatch": {
            "length": 4,
            "oxm_fields": [],
            "type": 1
         }
      },
      "out_group": 4294967295,
      "out_port": 4294967295,
      "table_id": 255
   }
}
```

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPAggregateStatsReply`**

Aggregate flow statistics reply message

The switch responds with a stats reply that include this message to an aggregate flow statistics request.

Attribute	Description
packet_count	Number of packets in flows
byte_count	Number of bytes in flows
flow_count	Number of flows

Example:

```
@set_ev_cls(ofp_event.EventOFPStatsReply, MAIN_DISPATCHER)
def stats_reply_handler(self, ev):
    msg = ev.msg
    ofp = msg.datapath.ofproto
    body = ev.msg.body

    if msg.type == ofp.OFPST_AGGREGATE:
        self.aggregate_stats_reply_handler(body)

def aggregate_stats_reply_handler(self, body):
    self.logger.debug('AggregateStats: packet_count=%d byte_count=%d '
                      'flow_count=%d',
                      body.packet_count, body.byte_count,
                      body.flow_count)
```

JSON Example:

```
{
   "OFPStatsReply": {
      "body": {
         "OFPAggregateStatsReply": {
            "byte_count": 574,
            "flow_count": 6,
            "packet_count": 7
         }
      },
      "flags": 0,
      "type": 2
   }
}
```

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPTableStatsRequest`**(*datapath*, *flags=0*)

Table statistics request message

The controller uses this message to query flow table statictics.

Attribute	Description
flags	Zero (none yet defined in the spec)

Example:

```
def send_table_stats_request(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    req = ofp_parser.OFPTableStatsRequest(datapath)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPTableStatsRequest": {
      "flags": 0
   }
}
```

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPTableStats`**

Table statistics reply message

The switch responds with a stats reply that include this message to a table statistics request.

Attribute	Description
table_id	ID of table
name	table name
match	Bitmap of (1 << OFPXMT_*) that indicate the fields the table can match on
wildcards	Bitmap of (1 << OFPXMT_*) wildcards that are supported by the table
write_actions	Bitmap of OFPAT_* that are supported by the table with OFPIT_WRITE_ACTIONS
apply_actions	Bitmap of OFPAT_* that are supported by the table with OFPIT_APPLY_ACTIONS
write_setfields	Bitmap of (1 << OFPXMT_*) header fields that can be set with OFPIT_WRITE_ACTIONS
apply_setfields	Bitmap of (1 << OFPXMT_*) header fields that can be set with OFPIT_APPLY_ACTIONS
metadata_match	Bits of metadata table can match
metadata_write	Bits of metadata table can write
instructions	Bitmap of OFPIT_* values supported
config	Bitmap of OFPTC_* values
max_entries	Max number of entries supported
active_count	Number of active entries
lookup_count	Number of packets looked up in table
matched_count	Number of packets that hit table

Example:

```
@set_ev_cls(ofp_event.EventOFPStatsReply, MAIN_DISPATCHER)
def stats_reply_handler(self, ev):
    msg = ev.msg
    ofp = msg.datapath.ofproto
    body = ev.msg.body

    if msg.type == ofp.OFPST_TABLE:
        self.table_stats_reply_handler(body)

def table_stats_reply_handler(self, body):
    tables = []
    for stat in body:
        tables.append('table_id=%d active_count=%d lookup_count=%d '
                      ' matched_count=%d' %
                      (stat.table_id, stat.active_count,
                       stat.lookup_count, stat.matched_count))
    self.logger.debug('TableStats: %s', tables)
```

class ryu.ofproto.ofproto_v1_2_parser.**OFPPortStatsRequest**(*datapath*, *port_no=4294967295*, *flags=0*)

Port statistics request message

The controller uses this message to query information about ports statistics.

Attribute	Description
port_no	Port number to read (OFPP_ANY to all ports)
flags	Zero (none yet defined in the spec)

Example:

```
def send_port_stats_request(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    req = ofp_parser.OFPPortStatsRequest(datapath, ofp.OFPP_ANY)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPPortStatsRequest": {
      "flags": 0,
      "port_no": 4294967295
   }
}
```

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPPortStats`**

Port statistics reply message

The switch responds with a stats reply that include this message to a port statistics request.

Attribute	Description
port_no	Port number
rx_packets	Number of received packets
tx_packets	Number of transmitted packets
rx_bytes	Number of received bytes
tx_bytes	Number of transmitted bytes
rx_dropped	Number of packets dropped by RX
tx_dropped	Number of packets dropped by TX
rx_errors	Number of receive errors
tx_errors	Number of transmit errors
rx_frame_err	Number of frame alignment errors
rx_over_err	Number of packet with RX overrun
rx_crc_err	Number of CRC errors
collisions	Number of collisions

Example:

```
@set_ev_cls(ofp_event.EventOFPStatsReply, MAIN_DISPATCHER)
def stats_reply_handler(self, ev):
    msg = ev.msg
    ofp = msg.datapath.ofproto
    body = ev.msg.body

    if msg.type == ofp.OFPST_PORT:
        self.port_stats_reply_handler(body)

def port_stats_reply_handler(self, body):
    ports = []
    for stat in body:
        ports.append('port_no=%d '
                     'rx_packets=%d tx_packets=%d '
                     'rx_bytes=%d tx_bytes=%d '
                     'rx_dropped=%d tx_dropped=%d '
                     'rx_errors=%d tx_errors=%d '
                     'rx_frame_err=%d rx_over_err=%d rx_crc_err=%d '
                     'collisions=%d' %
                     (stat.port_no,
                      stat.rx_packets, stat.tx_packets,
                      stat.rx_bytes, stat.tx_bytes,
                      stat.rx_dropped, stat.tx_dropped,
                      stat.rx_errors, stat.tx_errors,
                      stat.rx_frame_err, stat.rx_over_err,
                      stat.rx_crc_err, stat.collisions))
    self.logger.debug('PortStats: %s', ports)
```

JSON Example:

```
{
   "OFPStatsReply": {
      "body": [
         {
            "OFPPortStats": {
               "collisions": 0,
               "port_no": 7,
               "rx_bytes": 0,
               "rx_crc_err": 0,
               "rx_dropped": 0,
               "rx_errors": 0,
               "rx_frame_err": 0,
               "rx_over_err": 0,
               "rx_packets": 0,
               "tx_bytes": 336,
               "tx_dropped": 0,
               "tx_errors": 0,
               "tx_packets": 4
            }
         },
         {
            "OFPPortStats": {
               "collisions": 0,
               "port_no": 6,
               "rx_bytes": 336,
               "rx_crc_err": 0,
               "rx_dropped": 0,
               "rx_errors": 0,
               "rx_frame_err": 0,
               "rx_over_err": 0,
               "rx_packets": 4,
               "tx_bytes": 336,
               "tx_dropped": 0,
               "tx_errors": 0,
               "tx_packets": 4
            }
         }
      ],
      "flags": 0,
      "type": 4
   }
}
```

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPQueueStatsRequest`**(*datapath*, *port_no=4294967295*, *queue_id=4294967295*, *flags=0*)

Queue statistics request message

The controller uses this message to query queue statictics.

Attribute	Description
port_no	Port number to read
queue_id	ID of queue to read
flags	Zero (none yet defined in the spec)

Example:

```
def send_queue_stats_request(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    req = ofp_parser.OFPQueueStatsRequest(datapath, ofp.OFPP_ANY,
                                          ofp.OFPQ_ALL)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPQueueStatsRequest": {
      "flags": 0,
      "port_no": 4294967295,
      "queue_id": 4294967295
   }
}
```

class ryu.ofproto.ofproto_v1_2_parser.**OFPQueueStats**

Queue statistics reply message

The switch responds with a stats reply that include this message to an aggregate flow statistics request.

Attribute	Description
port_no	Port number
queue_id	ID of queue
tx_bytes	Number of transmitted bytes
tx_packets	Number of transmitted packets
tx_errors	Number of packets dropped due to overrun

Example:

```
@set_ev_cls(ofp_event.EventOFPStatsReply, MAIN_DISPATCHER)
def stats_reply_handler(self, ev):
    msg = ev.msg
    ofp = msg.datapath.ofproto
    body = ev.msg.body

    if msg.type == ofp.OFPST_QUEUE:
        self.queue_stats_reply_handler(body)

def queue_stats_reply_handler(self, body):
    queues = []
    for stat in body:
        queues.append('port_no=%d queue_id=%d '
                      'tx_bytes=%d tx_packets=%d tx_errors=%d ' %
                      (stat.port_no, stat.queue_id,
                       stat.tx_bytes, stat.tx_packets, stat.tx_errors))
    self.logger.debug('QueueStats: %s', queues)
```

JSON Example:

```
{
   "OFPStatsReply": {
      "body": [
         {
            "OFPQueueStats": {
               "port_no": 7,
               "queue_id": 1,
               "tx_bytes": 0,
                    "tx_errors": 0,
                    "tx_packets": 0
                 }
              },
              {
                 "OFPQueueStats": {
                    "port_no": 6,
                    "queue_id": 1,
                    "tx_bytes": 0,
                    "tx_errors": 0,
                    "tx_packets": 0
                 }
              },
              {
                 "OFPQueueStats": {
                    "port_no": 7,
                    "queue_id": 2,
                    "tx_bytes": 0,
                    "tx_errors": 0,
                    "tx_packets": 0
                 }
              }
           ],
           "flags": 0,
           "type": 5
        }
     }
```

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPGroupStatsRequest`**(*datapath*, *group_id=4294967292*, *flags=0*)

Group statistics request message

The controller uses this message to query statistics of one or more groups.

Attribute	Description
group_id	ID of group to read (OFPG_ALL to all groups)
flags	Zero (none yet defined in the spec)

Example:

```
def send_group_stats_request(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    req = ofp_parser.OFPGroupStatsRequest(datapath, ofp.OFPG_ALL)
    datapath.send_msg(req)
```

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPGroupStats`**(*group_id*, *ref_count*, *packet_count*, *byte_count*, *bucket_counters*)

Group statistics reply message

The switch responds with a stats reply that include this message to a group statistics request.

Attribute	Description
group_id	Group identifier
ref_count	Number of flows or groups that directly forward to this group
packet_count	Number of packets processed by group
byte_count	Number of bytes processed by group
bucket_counters	List of `OFPBucketCounter` instance

Example:

```
@set_ev_cls(ofp_event.EventOFPStatsReply, MAIN_DISPATCHER)
def stats_reply_handler(self, ev):
    msg = ev.msg
    ofp = msg.datapath.ofproto
    body = ev.msg.body

    if msg.type == ofp.OFPST_GROUP:
        self.group_stats_reply_handler(body)

def group_stats_reply_handler(self, body):
    groups = []
    for stat in body:
        groups.append('group_id=%d ref_count=%d packet_count=%d '
                      'byte_count=%d bucket_counters=%s' %
                      (stat.group_id,
                       stat.ref_count, stat.packet_count,
                       stat.byte_count, stat.bucket_counters))
    self.logger.debug('GroupStats: %s', groups)
```

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPGroupDescStatsRequest`**(*datapath*, *flags=0*)

Group description request message

The controller uses this message to list the set of groups on a switch.

Attribute	Description
flags	Zero (none yet defined in the spec)

Example:

```
def send_group_desc_stats_request(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    req = ofp_parser.OFPGroupDescStatsRequest(datapath)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPGroupDescStatsRequest": {
      "flags": 0
   }
}
```

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPGroupDescStats`**(*type_*, *group_id*, *buckets*, *length=None*)

Group description reply message

The switch responds with a stats reply that include this message to a group description request.

Attribute	Description
type	One of OFPGT_*
group_id	Group identifier
buckets	List of `OFPBucket` instance

`type` attribute corresponds to `type_` parameter of __init__.

Example:

```
@set_ev_cls(ofp_event.EventOFPStatsReply, MAIN_DISPATCHER)
def stats_reply_handler(self, ev):
    msg = ev.msg
    ofp = msg.datapath.ofproto
    body = ev.msg.body

    if msg.type == ofp.OFPST_GROUP_DESC:
        self.group_desc_stats_reply_handler(body)

def group_desc_stats_reply_handler(self, body):
    descs = []
    for stat in body:
        descs.append('type=%d group_id=%d buckets=%s' %
                     (stat.type, stat.group_id, stat.buckets))
    self.logger.debug('GroupDescStats: %s', descs)
```

JSON Example:

```
{
   "OFPStatsReply": {
      "body": [
         {
            "OFPGroupDescStats": {
               "buckets": [
                  {
                     "OFPBucket": {
                        "actions": [
                           {
                              "OFPActionOutput": {
                                 "len": 16,
                                 "max_len": 65535,
                                 "port": 2,
                                 "type": 0
                              }
                           }
                        ],
                        "len": 32,
                        "watch_group": 1,
                        "watch_port": 1,
                        "weight": 1
                     }
                  }
               ],
               "group_id": 1,
               "length": 40,
               "type": 0
            }
         }
      ],
      "flags": 0,
      "type": 7
   }
}
```

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPGroupFeaturesStatsRequest`**(*datapath*, *flags=0*)

Group features request message

The controller uses this message to list the capabilities of groups on a switch.

Attribute	Description
flags	Zero (none yet defined in the spec)

Example:

```
def send_group_features_stats_request(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    req = ofp_parser.OFPGroupFeaturesStatsRequest(datapath)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPGroupFeaturesStatsRequest": {
      "flags": 0
   }
}
```

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPGroupFeaturesStats`**(*types*, *capabilities*, *max_groups*, *actions*, *length=None*)

Group features reply message

The switch responds with a stats reply that include this message to a group features request.

Attribute	Description
types	Bitmap of OFPGT_* values supported
capabilities	Bitmap of OFPGFC_* capability supported
max_groups	Maximum number of groups for each type
actions	Bitmaps of OFPAT_* that are supported

Example:

```
@set_ev_cls(ofp_event.EventOFPStatsReply, MAIN_DISPATCHER)
def stats_reply_handler(self, ev):
    msg = ev.msg
    ofp = msg.datapath.ofproto
    body = ev.msg.body

    if msg.type == ofp.OFPST_GROUP_FEATURES:
        self.group_features_stats_reply_handler(body)

def group_features_stats_reply_handler(self, body):
    self.logger.debug('GroupFeaturesStats: types=%d '
                      'capabilities=0x%08x max_groups=%s '
                      'actions=%s',
                      body.types, body.capabilities, body.max_groups,
                      body.actions)
```

JSON Example:

```
{
   "OFPStatsReply": {
      "body": {
         "OFPGroupFeaturesStats": {
            "actions": [
               67082241,
               67082241,
               67082241,
                    67082241
                ],
                "capabilities": 5,
                "length": 40,
                "max_groups": [
                    16777216,
                    16777216,
                    16777216,
                    16777216
                ],
                "types": 15
            }
        },
        "flags": 0,
        "type": 8
    }
}
```

Queue Configuration Messages

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPQueueGetConfigRequest`**(*datapath*, *port*)

Queue configuration request message

Attribute	Description
port	Port to be queried (OFPP_ANY to all configured queues)

Example:

```
def send_queue_get_config_request(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    req = ofp_parser.OFPQueueGetConfigRequest(datapath, ofp.OFPP_ANY)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPQueueGetConfigRequest": {
      "port": 4294967295
   }
}
```

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPQueueGetConfigReply`**(*datapath*, *port=None*, *queues=None*)

Queue configuration reply message

The switch responds with this message to a queue configuration request.

Attribute	Description
port	Port which was queried
queues	list of `OFPPacketQueue` instance

Example:

```
@set_ev_cls(ofp_event.EventOFPQueueGetConfigReply, MAIN_DISPATCHER)
def queue_get_config_reply_handler(self, ev):
    msg = ev.msg

    self.logger.debug('OFPQueueGetConfigReply received: '
                      'port=%s queues=%s',
                      msg.port, msg.queues)
```

JSON Example:

```
{
   "OFPQueueGetConfigReply": {
      "port": 4294967295,
      "queues": [
         {
            "OFPPacketQueue": {
               "len": 48,
               "port": 77,
               "properties": [
                  {
                     "OFPQueuePropMinRate": {
                        "len": 16,
                        "property": 1,
                        "rate": 10
                     }
                  },
                  {
                     "OFPQueuePropMaxRate": {
                        "len": 16,
                        "property": 2,
                        "rate": 900
                     }
                  }
               ],
               "queue_id": 99
            }
         },
         {
            "OFPPacketQueue": {
               "len": 48,
               "port": 77,
               "properties": [
                  {
                     "OFPQueuePropMinRate": {
                        "len": 16,
                        "property": 1,
                        "rate": 100
                     }
                  },
                  {
                     "OFPQueuePropMaxRate": {
                        "len": 16,
                        "property": 2,
                        "rate": 200
                     }
                  }
               ],
               "queue_id": 88
            }
         }
      ]
   }
}
```

Packet-Out Message

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPPacketOut`**(*datapath*, *buffer_id=None*, *in_port=None*, *actions=None*, *data=None*, *actions_len=None*)

Packet-Out message

The controller uses this message to send a packet out throught the switch.

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

Example:

```
def send_packet_out(self, datapath, buffer_id, in_port):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    actions = [ofp_parser.OFPActionOutput(ofp.OFPP_FLOOD, 0)]
    req = ofp_parser.OFPPacketOut(datapath, buffer_id,
                                  in_port, actions)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPPacketOut": {
      "actions": [
         {
            "OFPActionOutput": {
               "len": 16,
               "max_len": 65535,
               "port": 4294967292,
               "type": 0
            }
         }
      ],
      "actions_len": 16,
      "buffer_id": 4294967295,
      "data": "8guk0D9w8gukffjqCABFAABU+BoAAP8Br4sKAAABCgAAAggAAgj3YAAAMdYCAAAAAACrjS0xAAAAABARE
      "in_port": 4294967293
   }
}
```

Barrier Message

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPBarrierRequest`**(*datapath*)

Barrier request message

The controller sends this message to ensure message dependencies have been met or receive notifications for completed operations.

Example:

```
def send_barrier_request(self, datapath):
    ofp_parser = datapath.ofproto_parser
    req = ofp_parser.OFPBarrierRequest(datapath)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPBarrierRequest": {}
}
```

class ryu.ofproto.ofproto_v1_2_parser.**OFPBarrierReply**(*datapath*)

Barrier reply message

The switch responds with this message to a barrier request.

Example:

```
@set_ev_cls(ofp_event.EventOFPBarrierReply, MAIN_DISPATCHER)
def barrier_reply_handler(self, ev):
    self.logger.debug('OFPBarrierReply received')
```

JSON Example:

```
{
   "OFPBarrierReply": {}
}
```

Role Request Message

class ryu.ofproto.ofproto_v1_2_parser.**OFPRoleRequest**(*datapath*, *role*, *generation_id*)

Role request message

The controller uses this message to change its role.

Attribute	Description
role	One of the following values. OFPCR_ROLE_NOCHANGE OFPCR_ROLE_EQUAL OFPCR_ROLE_MASTER OFPCR_ROLE_SLAVE
generation_id	Master Election Generation ID

Example:

```
def send_role_request(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    req = ofp_parser.OFPRoleRequest(datapath, ofp.OFPCR_ROLE_EQUAL, 0)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPRoleRequest": {
      "generation_id": 17294086455919964160,
      "role": 2
   }
}
```

class ryu.ofproto.ofproto_v1_2_parser.**OFPRoleReply**(*datapath*, *role=None*, *generation_id=None*)

Role reply message

The switch responds with this message to a role request.

Attribute	Description
role	One of the following values. OFPCR_ROLE_NOCHANGE OFPCR_ROLE_EQUAL OFPCR_ROLE_MASTER OFPCR_ROLE_SLAVE
generation_id	Master Election Generation ID

Example:

```
@set_ev_cls(ofp_event.EventOFPRoleReply, MAIN_DISPATCHER)
def role_reply_handler(self, ev):
    msg = ev.msg
    ofp = dp.ofproto

    if msg.role == ofp.OFPCR_ROLE_NOCHANGE:
        role = 'NOCHANGE'
    elif msg.role == ofp.OFPCR_ROLE_EQUAL:
        role = 'EQUAL'
    elif msg.role == ofp.OFPCR_ROLE_MASTER:
        role = 'MASTER'
    elif msg.role == ofp.OFPCR_ROLE_SLAVE:
        role = 'SLAVE'
    else:
        role = 'unknown'

    self.logger.debug('OFPRoleReply received: '
                      'role=%s generation_id=%d',
                      role, msg.generation_id)
```

JSON Example:

```
{
   "OFPRoleReply": {
      "generation_id": 17294086455919964160,
      "role": 3
   }
}
```

Asynchronous Messages

Packet-In Message

class ryu.ofproto.ofproto_v1_2_parser.**OFPPacketIn**(*datapath*, *buffer_id=None*, *total_len=None*, *reason=None*, *table_id=None*, *match=None*, *data=None*)

Packet-In message

The switch sends the packet that received to the controller by this message.

Attribute	Description
buffer_id	ID assigned by datapath
total_len	Full length of frame
reason	Reason packet is being sent. OFPR_NO_MATCH OFPR_ACTION OFPR_INVALID_TTL
table_id	ID of the table that was looked up
match	Instance of `OFPMatch`
data	Ethernet frame

Example:

```
@set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
def packet_in_handler(self, ev):
    msg = ev.msg
    ofp = dp.ofproto

    if msg.reason == ofp.OFPR_NO_MATCH:
        reason = 'NO MATCH'
    elif msg.reason == ofp.OFPR_ACTION:
        reason = 'ACTION'
    elif msg.reason == ofp.OFPR_INVALID_TTL:
        reason = 'INVALID TTL'
    else:
        reason = 'unknown'

    self.logger.debug('OFPPacketIn received: '
                      'buffer_id=%x total_len=%d reason=%s '
                      'table_id=%d match=%s data=%s',
                      msg.buffer_id, msg.total_len, reason,
                      msg.table_id, msg.match,
                      utils.hex_array(msg.data))
```

JSON Example:

```
{
   "OFPPacketIn": {
      "buffer_id": 2,
      "data": "////////8gukffjqCAYAAQgABgQAAfILpH346goAAAEAAAAAAAAKAAAD",
      "match": {
         "OFPMatch": {
            "length": 80,
            "oxm_fields": [
               {
                  "OXMTlv": {
                     "field": "in_port",
                     "mask": null,
                     "value": 6
                  }
               },
               {
                  "OXMTlv": {
                     "field": "eth_type",
                     "mask": null,
                     "value": 2054
                  }
               },
               {
                  "OXMTlv": {
                     "field": "eth_dst",
                     "mask": null,
                     "value": "ff:ff:ff:ff:ff:ff"
                  }
               },
               {
                  "OXMTlv": {
                     "field": "eth_src",
                     "mask": null,
                     "value": "f2:0b:a4:7d:f8:ea"
```

```
                            }
                        },
                        {
                            "OXMTlv": {
                                "field": "arp_op",
                                "mask": null,
                                "value": 1
                            }
                        },
                        {
                            "OXMTlv": {
                                "field": "arp_spa",
                                "mask": null,
                                "value": "10.0.0.1"
                            }
                        },
                        {
                            "OXMTlv": {
                                "field": "arp_tpa",
                                "mask": null,
                                "value": "10.0.0.3"
                            }
                        },
                        {
                            "OXMTlv": {
                                "field": "arp_sha",
                                "mask": null,
                                "value": "f2:0b:a4:7d:f8:ea"
                            }
                        },
                        {
                            "OXMTlv": {
                                "field": "arp_tha",
                                "mask": null,
                                "value": "00:00:00:00:00:00"
                            }
                        }
                    ],
                    "type": 1
                }
            },
            "reason": 1,
            "table_id": 1,
            "total_len": 42
        }
    }
```

Flow Removed Message

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPFlowRemoved`**(*datapath, cookie=None, priority=None, reason=None, table_id=None, duration_sec=None, duration_nsec=None, idle_timeout=None, hard_timeout=None, packet_count=None, byte_count=None, match=None*)

Flow removed message

When flow entries time out or are deleted, the switch notifies controller with this message.

Attribute	Description
cookie	Opaque controller-issued identifier
priority	Priority level of flow entry
reason	One of the following values. OFPRR_IDLE_TIMEOUT OFPRR_HARD_TIMEOUT OFPRR_DELETE OFPRR_GROUP_DELETE
table_id	ID of the table
duration_sec	Time flow was alive in seconds
duration_nsec	Time flow was alive in nanoseconds beyond duration_sec
idle_timeout	Idle timeout from original flow mod
hard_timeout	Hard timeout from original flow mod
packet_count	Number of packets that was associated with the flow
byte_count	Number of bytes that was associated with the flow
match	Instance of `OFPMatch`

Example:

```
@set_ev_cls(ofp_event.EventOFPFlowRemoved, MAIN_DISPATCHER)
def flow_removed_handler(self, ev):
    msg = ev.msg
    dp = msg.datapath
    ofp = dp.ofproto

    if msg.reason == ofp.OFPRR_IDLE_TIMEOUT:
        reason = 'IDLE TIMEOUT'
    elif msg.reason == ofp.OFPRR_HARD_TIMEOUT:
        reason = 'HARD TIMEOUT'
    elif msg.reason == ofp.OFPRR_DELETE:
        reason = 'DELETE'
    elif msg.reason == ofp.OFPRR_GROUP_DELETE:
        reason = 'GROUP DELETE'
    else:
        reason = 'unknown'

    self.logger.debug('OFPFlowRemoved received: '
                      'cookie=%d priority=%d reason=%s table_id=%d '
                      'duration_sec=%d duration_nsec=%d '
                      'idle_timeout=%d hard_timeout=%d '
                      'packet_count=%d byte_count=%d match.fields=%s',
                      msg.cookie, msg.priority, reason, msg.table_id,
                      msg.duration_sec, msg.duration_nsec,
                      msg.idle_timeout, msg.hard_timeout,
                              msg.packet_count, msg.byte_count, msg.match)
```

JSON Example:

```
{
   "OFPFlowRemoved": {
      "byte_count": 86,
      "cookie": 0,
      "duration_nsec": 48825000,
      "duration_sec": 3,
      "hard_timeout": 0,
      "idle_timeout": 3,
      "match": {
         "OFPMatch": {
            "length": 14,
            "oxm_fields": [
               {
                  "OXMTlv": {
                     "field": "eth_dst",
                     "mask": null,
                     "value": "f2:0b:a4:7d:f8:ea"
                  }
               }
            ],
            "type": 1
         }
      },
      "packet_count": 1,
      "priority": 65535,
      "reason": 0,
      "table_id": 0
   }
}
```

Port Status Message

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPPortStatus`**(*datapath*, *reason=None*, *desc=None*)

Port status message

The switch notifies controller of change of ports.

Attribute	Description
reason	One of the following values. OFPPR_ADD OFPPR_DELETE OFPPR_MODIFY
desc	instance of `OFPPort`

Example:

```
@set_ev_cls(ofp_event.EventOFPPortStatus, MAIN_DISPATCHER)
def port_status_handler(self, ev):
    msg = ev.msg
    dp = msg.datapath
    ofp = dp.ofproto

    if msg.reason == ofp.OFPPR_ADD:
        reason = 'ADD'
    elif msg.reason == ofp.OFPPR_DELETE:
        reason = 'DELETE'
    elif msg.reason == ofp.OFPPR_MODIFY:
        reason = 'MODIFY'
    else:
        reason = 'unknown'

    self.logger.debug('OFPPortStatus received: reason=%s desc=%s',
                      reason, msg.desc)
```

JSON Example:

```
{
   "OFPPortStatus": {
      "desc": {
         "OFPPort": {
            "advertised": 10240,
            "config": 0,
            "curr": 10248,
            "curr_speed": 5000,
            "hw_addr": "f2:0b:a4:d0:3f:70",
            "max_speed": 5000,
            "name": "\u79c1\u306e\u30dd\u30fc\u30c8",
            "peer": 10248,
            "port_no": 7,
            "state": 4,
            "supported": 10248
         }
      },
      "reason": 0
   }
}
```

Error Message

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPErrorMsg`**(*datapath*, *type_=None*, *code=None*, *data=None*)

Error message

The switch notifies controller of problems by this message.

Attribute	Description
type	High level type of error
code	Details depending on the type
data	Variable length data depending on the type and code

`type` attribute corresponds to `type_` parameter of __init__.

Types and codes are defined in `ryu.ofproto.ofproto`.

Type	Code
OFPET_HELLO_FAILED	OFPHFC_*
OFPET_BAD_REQUEST	OFPBRC_*
OFPET_BAD_ACTION	OFPBAC_*
OFPET_BAD_INSTRUCTION	OFPBIC_*
OFPET_BAD_MATCH	OFPBMC_*
OFPET_FLOW_MOD_FAILED	OFPFMFC_*
OFPET_GROUP_MOD_FAILED	OFPGMFC_*
OFPET_PORT_MOD_FAILED	OFPPMFC_*
OFPET_TABLE_MOD_FAILED	OFPTMFC_*
OFPET_QUEUE_OP_FAILED	OFPQOFC_*
OFPET_SWITCH_CONFIG_FAILED	OFPSCFC_*
OFPET_ROLE_REQUEST_FAILED	OFPRRFC_*
OFPET_EXPERIMENTER	N/A

Example:

```
@set_ev_cls(ofp_event.EventOFPErrorMsg,
            [HANDSHAKE_DISPATCHER, CONFIG_DISPATCHER, MAIN_DISPATCHER])
def error_msg_handler(self, ev):
    msg = ev.msg

    self.logger.debug('OFPErrorMsg received: type=0x%02x code=0x%02x '
                      'message=%s',
                      msg.type, msg.code, utils.hex_array(msg.data))
```

JSON Example:

```
{
   "OFPErrorMsg": {
      "code": 11,
      "data": "ZnVnYWZ1Z2E=",
      "type": 2
   }
}

{
   "OFPErrorExperimenterMsg": {
      "data": "amlra2VuIGRhdGE=",
      "exp_type": 60000,
      "experimenter": 999999,
      "type": 65535
   }
}
```

Symmetric Messages

Hello

class ryu.ofproto.ofproto_v1_2_parser.**OFPHello**(*datapath*)

Hello message

When connection is started, the hello message is exchanged between a switch and a controller.

This message is handled by the Ryu framework, so the Ryu application do not need to process this typically.

JSON Example:

```
{
   "OFPHello": {}
}
```

Echo Request

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPEchoRequest`**(*datapath*, *data=None*)

Echo request message

This message is handled by the Ryu framework, so the Ryu application do not need to process this typically.

Attribute	Description
data	An arbitrary length data

Example:

```
def send_echo_request(self, datapath, data):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    req = ofp_parser.OFPEchoRequest(datapath, data)
    datapath.send_msg(req)

@set_ev_cls(ofp_event.EventOFPEchoRequest,
            [HANDSHAKE_DISPATCHER, CONFIG_DISPATCHER, MAIN_DISPATCHER])
def echo_request_handler(self, ev):
    self.logger.debug('OFPEchoRequest received: data=%s',
                      utils.hex_array(ev.msg.data))
```

JSON Example:

```
{
   "OFPEchoRequest": {
      "data": "aG9nZQ=="
   }
}
```

Echo Reply

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPEchoReply`**(*datapath*, *data=None*)

Echo reply message

This message is handled by the Ryu framework, so the Ryu application do not need to process this typically.

Attribute	Description
data	An arbitrary length data

Example:

```
def send_echo_reply(self, datapath, data):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    reply = ofp_parser.OFPEchoReply(datapath, data)
    datapath.send_msg(reply)

@set_ev_cls(ofp_event.EventOFPEchoReply,
            [HANDSHAKE_DISPATCHER, CONFIG_DISPATCHER, MAIN_DISPATCHER])
def echo_reply_handler(self, ev):
    self.logger.debug('OFPEchoReply received: data=%s',
                      utils.hex_array(ev.msg.data))
```

JSON Example:

```
{
   "OFPEchoReply": {
      "data": "aG9nZQ=="
   }
}
```

Experimenter

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPExperimenter`**(*datapath*, *experimenter=None*, *exp_type=None*, *data=None*)

Experimenter extension message

Attribute	Description
experimenter	Experimenter ID
exp_type	Experimenter defined
data	Experimenter defined arbitrary additional data

JSON Example:

```
{
   "OFPExperimenter": {
      "data": "bmF6bw==",
      "exp_type": 123456789,
      "experimenter": 98765432
   }
}
```

Flow Match Structure

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPMatch`**(*type_=None*, *length=None*, *_ordered_fields=None*, ***kwargs*)

Flow Match Structure

This class is implementation of the flow match structure having compose/query API. There are new API and old API for compatibility. the old API is supposed to be removed later.

You can define the flow match by the keyword arguments. The following arguments are available.

Argument	Value	Description
in_port	Integer 32bit	Switch input port
in_phy_port	Integer 32bit	Switch physical input port
metadata	Integer 64bit	Metadata passed between tables
eth_dst	MAC address	Ethernet destination address
eth_src	MAC address	Ethernet source address
eth_type	Integer 16bit	Ethernet frame type
vlan_vid	Integer 16bit	VLAN id
vlan_pcp	Integer 8bit	VLAN priority
ip_dscp	Integer 8bit	IP DSCP (6 bits in ToS field)
ip_ecn	Integer 8bit	IP ECN (2 bits in ToS field)
ip_proto	Integer 8bit	IP protocol
ipv4_src	IPv4 address	IPv4 source address

Continued on next page

Table 2.1 – continued from previous page

Argument	Value	Description
ipv4_dst	IPv4 address	IPv4 destination address
tcp_src	Integer 16bit	TCP source port
tcp_dst	Integer 16bit	TCP destination port
udp_src	Integer 16bit	UDP source port
udp_dst	Integer 16bit	UDP destination port
sctp_src	Integer 16bit	SCTP source port
sctp_dst	Integer 16bit	SCTP destination port
icmpv4_type	Integer 8bit	ICMP type
icmpv4_code	Integer 8bit	ICMP code
arp_op	Integer 16bit	ARP opcode
arp_spa	IPv4 address	ARP source IPv4 address
arp_tpa	IPv4 address	ARP target IPv4 address
arp_sha	MAC address	ARP source hardware address
arp_tha	MAC address	ARP target hardware address
ipv6_src	IPv6 address	IPv6 source address
ipv6_dst	IPv6 address	IPv6 destination address
ipv6_flabel	Integer 32bit	IPv6 Flow Label
icmpv6_type	Integer 8bit	ICMPv6 type
icmpv6_code	Integer 8bit	ICMPv6 code
ipv6_nd_target	IPv6 address	Target address for ND
ipv6_nd_sll	MAC address	Source link-layer for ND
ipv6_nd_tll	MAC address	Target link-layer for ND
mpls_label	Integer 32bit	MPLS label
mpls_tc	Integer 8bit	MPLS TC

Example:

```
>>> # compose
>>> match = parser.OFPMatch(
...     in_port=1,
...     eth_type=0x86dd,
...     ipv6_src=('2001:db8:bd05:1d2:288a:1fc0:1:10ee',
...               'ffff:ffff:ffff:ffff::'),
...     ipv6_dst='2001:db8:bd05:1d2:288a:1fc0:1:10ee')
>>> # query
>>> if 'ipv6_src' in match:
...     print match['ipv6_src']
...
('2001:db8:bd05:1d2:288a:1fc0:1:10ee', 'ffff:ffff:ffff:ffff::')
```

Flow Instruction Structures

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPInstructionGotoTable`**(*table_id*, *type_=None*, *len_=None*)

Goto table instruction

This instruction indicates the next table in the processing pipeline.

Attribute	Description
table_id	Next table

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPInstructionWriteMetadata`**(*metadata*, *metadata_mask*, *type_=None*, *len_=None*)

Write metadata instruction

This instruction writes the masked metadata value into the metadata field.

Attribute	Description
metadata	Metadata value to write
metadata_mask	Metadata write bitmask

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPInstructionActions`**(*type_*, *actions=None*, *len_=None*)

Actions instruction

This instruction writes/applies/clears the actions.

Attribute	Description
type	One of following values. OFPIT_WRITE_ACTIONS OFPIT_APPLY_ACTIONS OFPIT_CLEAR_ACTIONS
actions	list of OpenFlow action class

`type` attribute corresponds to `type_` parameter of __init__.

Action Structures

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPActionOutput`**(*port*, *max_len=65509*, *type_=None*, *len_=None*)

Output action

This action indicates output a packet to the switch port.

Attribute	Description
port	Output port
max_len	Max length to send to controller

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPActionGroup`**(*group_id*, *type_=None*, *len_=None*)

Group action

This action indicates the group used to process the packet.

Attribute	Description
group_id	Group identifier

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPActionSetQueue`**(*queue_id*, *type_=None*, *len_=None*)

Set queue action

This action sets the queue id that will be used to map a flow to an already-configured queue on a port.

Attribute	Description
queue_id	Queue ID for the packets

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPActionSetMplsTtl`**(*mpls_ttl*, *type_=None*, *len_=None*)

Set MPLS TTL action

This action sets the MPLS TTL.

Attribute	Description
mpls_ttl	MPLS TTL

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPActionDecMplsTtl`**(*type_=None*, *len_=None*)
Decrement MPLS TTL action

This action decrements the MPLS TTL.

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPActionSetNwTtl`**(*nw_ttl*, *type_=None*, *len_=None*)
Set IP TTL action

This action sets the IP TTL.

Attribute	Description
nw_ttl	IP TTL

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPActionDecNwTtl`**(*type_=None*, *len_=None*)
Decrement IP TTL action

This action decrements the IP TTL.

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPActionCopyTtlOut`**(*type_=None*, *len_=None*)
Copy TTL Out action

This action copies the TTL from the next-to-outermost header with TTL to the outermost header with TTL.

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPActionCopyTtlIn`**(*type_=None*, *len_=None*)
Copy TTL In action

This action copies the TTL from the outermost header with TTL to the next-to-outermost header with TTL.

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPActionPushVlan`**(*ethertype=33024*, *type_=None*, *len_=None*)
Push VLAN action

This action pushes a new VLAN tag to the packet.

Attribute	Description
ethertype	Ether type. The default is 802.1Q. (0x8100)

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPActionPushMpls`**(*ethertype=34887*, *type_=None*, *len_=None*)
Push MPLS action

This action pushes a new MPLS header to the packet.

Attribute	Description
ethertype	Ether type

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPActionPopVlan`**(*type_=None*, *len_=None*)
Pop VLAN action

This action pops the outermost VLAN tag from the packet.

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPActionPopMpls`**(*ethertype=2048*, *type_=None*, *len_=None*)
Pop MPLS action

This action pops the MPLS header from the packet.

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPActionSetField`**(*field=None*, ***kwargs*)
Set field action

This action modifies a header field in the packet.

Attribute	Description
field	Instance of `OFPMatchField`

class `ryu.ofproto.ofproto_v1_2_parser.`**`OFPActionExperimenter`**(*experimenter*, *type_=None*, *len_=None*)

Experimenter action

This action is an extensible action for the experimenter.

Attribute	Description
experimenter	Experimenter ID

2.4.3 OpenFlow v1.3 Messages and Structures

Controller-to-Switch Messages

Handshake

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPFeaturesRequest`**(*datapath*)

Features request message

The controller sends a feature request to the switch upon session establishment.

This message is handled by the Ryu framework, so the Ryu application do not need to process this typically.

Example:

```
def send_features_request(self, datapath):
    ofp_parser = datapath.ofproto_parser

    req = ofp_parser.OFPFeaturesRequest(datapath)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPFeaturesRequest": {}
}
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPSwitchFeatures`**(*datapath*, *datapath_id=None*, *n_buffers=None*, *n_tables=None*, *auxiliary_id=None*, *capabilities=None*)

Features reply message

The switch responds with a features reply message to a features request.

This message is handled by the Ryu framework, so the Ryu application do not need to process this typically.

Example:

```
@set_ev_cls(ofp_event.EventOFPSwitchFeatures, CONFIG_DISPATCHER)
def switch_features_handler(self, ev):
    msg = ev.msg

    self.logger.debug('OFPSwitchFeatures received: '
                      'datapath_id=0x%016x n_buffers=%d '
                      'n_tables=%d auxiliary_id=%d '
                      'capabilities=0x%08x',
                      msg.datapath_id, msg.n_buffers, msg.n_tables,
                      msg.auxiliary_id, msg.capabilities)
```

JSON Example:

```
{
   "OFPSwitchFeatures": {
      "auxiliary_id": 99,
      "capabilities": 79,
      "datapath_id": 9210263729383,
      "n_buffers": 0,
      "n_tables": 255
   }
}
```

Switch Configuration

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPSetConfig`**(*datapath*, *flags=0*, *miss_send_len=0*)

Set config request message

The controller sends a set config request message to set configuraion parameters.

Attribute	Description
flags	One of the following configuration flags. OFPC_FRAG_NORMAL OFPC_FRAG_DROP OFPC_FRAG_REASM OFPC_FRAG_MASK
miss_send_len	Max bytes of new flow that datapath should send to the controller

Example:

```
def send_set_config(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    req = ofp_parser.OFPSetConfig(datapath, ofp.OFPC_FRAG_NORMAL, 256)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPSetConfig": {
      "flags": 0,
      "miss_send_len": 128
   }
}
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPGetConfigRequest`**(*datapath*)

Get config request message

The controller sends a get config request to query configuration parameters in the switch.

Example:

```
def send_get_config_request(self, datapath):
    ofp_parser = datapath.ofproto_parser

    req = ofp_parser.OFPGetConfigRequest(datapath)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPGetConfigRequest": {}
}
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPGetConfigReply`**(*datapath*, *flags=None*, *miss_send_len=None*)

Get config reply message

The switch responds to a configuration request with a get config reply message.

Attribute	Description
flags	One of the following configuration flags. OFPC_FRAG_NORMAL OFPC_FRAG_DROP OFPC_FRAG_REASM OFPC_FRAG_MASK
miss_send_len	Max bytes of new flow that datapath should send to the controller

Example:

```
@set_ev_cls(ofp_event.EventOFPGetConfigReply, MAIN_DISPATCHER)
def get_config_reply_handler(self, ev):
    msg = ev.msg
    dp = msg.datapath
    ofp = dp.ofproto

    if msg.flags == ofp.OFPC_FRAG_NORMAL:
        flags = 'NORMAL'
    elif msg.flags == ofp.OFPC_FRAG_DROP:
        flags = 'DROP'
    elif msg.flags == ofp.OFPC_FRAG_REASM:
        flags = 'REASM'
    elif msg.flags == ofp.OFPC_FRAG_MASK:
        flags = 'MASK'
    else:
        flags = 'unknown'
    self.logger.debug('OFPGetConfigReply received: '
                      'flags=%s miss_send_len=%d',
                      flags, msg.miss_send_len)
```

JSON Example:

```
{
   "OFPGetConfigReply": {
      "flags": 0,
      "miss_send_len": 128
   }
}
```

Flow Table Configuration

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPTableMod`**(*datapath*, *table_id*, *config*)

Flow table configuration message

The controller sends this message to configure table state.

Attribute	Description
table_id	ID of the table (OFPTT_ALL indicates all tables)
config	Bitmap of the following flags. OFPTC_DEPRECATED_MASK (3)

Example:

```
def send_table_mod(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    req = ofp_parser.OFPTableMod(datapath, 1, 3)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPTableMod": {
      "config": 0,
      "table_id": 255
   }
}
```

Modify State Messages

class `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=[ ]*)

Modify Flow entry message

The controller sends this message to modify the flow table.

Attribute	Description
cookie	Opaque controller-issued identifier
cookie_mask	Mask used to restrict the cookie bits that must match when the command is `OPFFC_MODIFY*` or `OFPFC_DELETE*`
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 port
out_group	For `OFPFC_DELETE*` commands, require matching entries to include this as an output group
flags	One of the following values. 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

Example:

```
def send_flow_mod(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    cookie = cookie_mask = 0
    table_id = 0
    idle_timeout = hard_timeout = 0
    priority = 32768
    buffer_id = ofp.OFP_NO_BUFFER
    match = ofp_parser.OFPMatch(in_port=1, eth_dst='ff:ff:ff:ff:ff:ff')
    actions = [ofp_parser.OFPActionOutput(ofp.OFPP_NORMAL, 0)]
    inst = [ofp_parser.OFPInstructionActions(ofp.OFPIT_APPLY_ACTIONS,
                                             actions)]
    req = ofp_parser.OFPFlowMod(datapath, cookie, cookie_mask,
                                table_id, ofp.OFPFC_ADD,
                                idle_timeout, hard_timeout,
                                priority, buffer_id,
                                ofp.OFPP_ANY, ofp.OFPG_ANY,
                                ofp.OFPFF_SEND_FLOW_REM,
                                match, inst)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPFlowMod": {
      "buffer_id": 65535,
      "command": 0,
      "cookie": 0,
      "cookie_mask": 0,
      "flags": 0,
      "hard_timeout": 0,
      "idle_timeout": 0,
      "instructions": [
         {
            "OFPInstructionActions": {
               "actions": [
                  {
                     "OFPActionSetField": {
                        "field": {
                           "OXMTlv": {
                              "field": "vlan_vid",
                              "mask": null,
                              "value": 258
                           }
                        }
                     }
                  },
                  {
                     "OFPActionCopyTtlOut": {
                        "len": 8,
                        "type": 11
                     }
                  },
                  {
                     "OFPActionCopyTtlIn": {
                        "len": 8,
                        "type": 12
                     }
                  },
                  {
                     "OFPActionCopyTtlIn": {
                        "len": 8,
                        "type": 12
            }
        },
        {
            "OFPActionPopPbb": {
                "len": 8,
                "type": 27
            }
        },
        {
            "OFPActionPushPbb": {
                "ethertype": 4660,
                "len": 8,
                "type": 26
            }
        },
        {
            "OFPActionPopMpls": {
                "ethertype": 39030,
                "len": 8,
                "type": 20
            }
        },
        {
            "OFPActionPushMpls": {
                "ethertype": 34887,
                "len": 8,
                "type": 19
            }
        },
        {
            "OFPActionPopVlan": {
                "len": 8,
                "type": 18
            }
        },
        {
            "OFPActionPushVlan": {
                "ethertype": 33024,
                "len": 8,
                "type": 17
            }
        },
        {
            "OFPActionDecMplsTtl": {
                "len": 8,
                "type": 16
            }
        },
        {
            "OFPActionSetMplsTtl": {
                "len": 8,
                "mpls_ttl": 10,
                "type": 15
            }
        },
        {
            "OFPActionDecNwTtl": {
                "len": 8,
                           "type": 24
                        }
                     },
                     {
                        "OFPActionSetNwTtl": {
                           "len": 8,
                           "nw_ttl": 10,
                           "type": 23
                        }
                     },
                     {
                        "OFPActionSetQueue": {
                           "len": 8,
                           "queue_id": 3,
                           "type": 21
                        }
                     },
                     {
                        "OFPActionGroup": {
                           "group_id": 99,
                           "len": 8,
                           "type": 22
                        }
                     },
                     {
                        "OFPActionOutput": {
                           "len": 16,
                           "max_len": 65535,
                           "port": 6,
                           "type": 0
                        }
                     }
                  ],
                  "len": 160,
                  "type": 3
               }
            },
            {
               "OFPInstructionActions": {
                  "actions": [
                     {
                        "OFPActionSetField": {
                           "field": {
                              "OXMTlv": {
                                 "field": "eth_src",
                                 "mask": null,
                                 "value": "01:02:03:04:05:06"
                              }
                           }
                        }
                     },
                     {
                        "OFPActionSetField": {
                           "field": {
                              "OXMTlv": {
                                 "field": "pbb_uca",
                                 "mask": null,
                                 "value": 1
                           }
                        }
                     }
                  }
               ],
               "len": 40,
               "type": 4
            }
         }
      ],
      "match": {
         "OFPMatch": {
            "length": 14,
            "oxm_fields": [
               {
                  "OXMTlv": {
                     "field": "eth_dst",
                     "mask": null,
                     "value": "f2:0b:a4:7d:f8:ea"
                  }
               }
            ],
            "type": 1
         }
      },
      "out_group": 4294967295,
      "out_port": 4294967295,
      "priority": 123,
      "table_id": 1
   }
}

{
   "OFPFlowMod": {
      "buffer_id": 65535,
      "command": 0,
      "cookie": 0,
      "cookie_mask": 0,
      "flags": 0,
      "hard_timeout": 0,
      "idle_timeout": 0,
      "instructions": [
         {
            "OFPInstructionGotoTable": {
               "len": 8,
               "table_id": 1,
               "type": 1
            }
         }
      ],
      "match": {
         "OFPMatch": {
            "length": 22,
            "oxm_fields": [
               {
                  "OXMTlv": {
                     "field": "in_port",
                     "mask": null,
                     "value": 6
                  }
               },
               {
                  "OXMTlv": {
                     "field": "eth_src",
                     "mask": null,
                     "value": "f2:0b:a4:7d:f8:ea"
                  }
               }
            ],
            "type": 1
         }
      },
      "out_group": 4294967295,
      "out_port": 4294967295,
      "priority": 123,
      "table_id": 0
   }
}

{
   "OFPFlowMod": {
      "buffer_id": 65535,
      "command": 0,
      "cookie": 0,
      "cookie_mask": 0,
      "flags": 0,
      "hard_timeout": 0,
      "idle_timeout": 0,
      "instructions": [
         {
            "OFPInstructionMeter": {
               "len": 8,
               "type": 6,
               "meter_id": 1
            }
         },
         {
            "OFPInstructionActions": {
               "actions": [
                  {
                     "OFPActionOutput": {
                        "len": 16,
                        "max_len": 65535,
                        "port": 6,
                        "type": 0
                     }
                  }
               ],
               "len": 24,
               "type": 3
            }
         }
      ],
      "match": {
         "OFPMatch": {
            "length": 14,
            "oxm_fields": [
               {
                  "OXMTlv": {
                     "field": "eth_dst",
                     "mask": null,
                     "value": "f2:0b:a4:7d:f8:ea"
                  }
               }
            ],
            "type": 1
         }
      },
      "out_group": 4294967295,
      "out_port": 4294967295,
      "priority": 123,
      "table_id": 1
   }
}
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPGroupMod`**(*datapath*, *command*, *type_*, *group_id*, *buckets*)

Modify group entry message

The controller sends this message to modify the group table.

Attribute	Description
command	One of the following values. OFPGC_ADD OFPGC_MODIFY OFPGC_DELETE
type	One of the following values. OFPGT_ALL OFPGT_SELECT OFPGT_INDIRECT OFPGT_FF
group_id	Group identifier
buckets	list of `OFPBucket`

`type` attribute corresponds to `type_` parameter of __init__.

Example:

```
def send_group_mod(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    port = 1
    max_len = 2000
    actions = [ofp_parser.OFPActionOutput(port, max_len)]

    weight = 100
    watch_port = 0
    watch_group = 0
    buckets = [ofp_parser.OFPBucket(weight, watch_port, watch_group,
                                    actions)]

    group_id = 1
    req = ofp_parser.OFPGroupMod(datapath, ofp.OFPGC_ADD,
                                 ofp.OFPGT_SELECT, group_id, buckets)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPGroupMod": {
      "buckets": [
         {
            "OFPBucket": {
               "actions": [
                  {
                     "OFPActionOutput": {
                        "len": 16,
                        "max_len": 65535,
                        "port": 2,
                        "type": 0
                     }
                  }
               ],
               "len": 32,
               "watch_group": 1,
               "watch_port": 1,
               "weight": 1
            }
         }
      ],
      "command": 0,
      "group_id": 1,
      "type": 0
   }
}
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPPortMod`**(*datapath*, *port_no*, *hw_addr*, *config*, *mask*, *advertise*)

Port modification message

The controller sneds this message to modify the behavior of the port.

Attribute	Description
port_no	Port number to modify
hw_addr	The hardware address that must be the same as hw_addr of `OFPPort` of `OFPSwitchFeatures`
config	Bitmap of configuration flags. OFPPC_PORT_DOWN OFPPC_NO_RECV OFPPC_NO_FWD OFPPC_NO_PACKET_IN
mask	Bitmap of configuration flags above to be changed
advertise	Bitmap of the following flags. OFPPF_10MB_HD OFPPF_10MB_FD OFPPF_100MB_HD OFPPF_100MB_FD OFPPF_1GB_HD OFPPF_1GB_FD OFPPF_10GB_FD OFPPF_40GB_FD OFPPF_100GB_FD OFPPF_1TB_FD OFPPF_OTHER OFPPF_COPPER OFPPF_FIBER OFPPF_AUTONEG OFPPF_PAUSE OFPPF_PAUSE_ASYM

Example:

```
def send_port_mod(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    port_no = 3
    hw_addr = 'fa:c8:e8:76:1d:7e'
    config = 0
    mask = (ofp.OFPPC_PORT_DOWN | ofp.OFPPC_NO_RECV |
            ofp.OFPPC_NO_FWD | ofp.OFPPC_NO_PACKET_IN)
    advertise = (ofp.OFPPF_10MB_HD | ofp.OFPPF_100MB_FD |
                 ofp.OFPPF_1GB_FD | ofp.OFPPF_COPPER |
                 ofp.OFPPF_AUTONEG | ofp.OFPPF_PAUSE |
                 ofp.OFPPF_PAUSE_ASYM)
    req = ofp_parser.OFPPortMod(datapath, port_no, hw_addr, config,
                                mask, advertise)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPPortMod": {
      "advertise": 4096,
      "config": 0,
      "hw_addr": "00:11:00:00:11:11",
      "mask": 0,
      "port_no": 1
   }
}
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPMeterMod`**(*datapath*, *command=0*, *flags=1*, *meter_id=1*, *bands=[]*)

Meter modification message

The controller sends this message to modify the meter.

Attribute	Description
command	One of the following values. OFPMC_ADD OFPMC_MODIFY OFPMC_DELETE
flags	One of the following flags. OFPMF_KBPS OFPMF_PKTPS OFPMF_BURST OFPMF_STATS
meter_id	Meter instance
bands	list of the following class instance. OFPMeterBandDrop OFPMeterBandDscpRemark OFPMeterBandExperimenter

JSON Example:

```
{
   "OFPMeterMod": {
      "bands": [
         {
            "OFPMeterBandDrop": {
               "burst_size": 10,
               "len": 16,
               "rate": 1000,
               "type": 1
            }
         },
         {
            "OFPMeterBandDscpRemark": {
               "burst_size": 10,
               "len": 16,
               "prec_level": 1,
               "rate": 1000,
               "type": 2
            }
         },
         {
            "OFPMeterBandExperimenter": {
               "burst_size": 10,
               "experimenter": 999,
               "len": 16,
               "rate": 1000,
               "type": 65535
```

```
                }
            }
        ],
        "command": 0,
        "flags": 14,
        "meter_id": 100
    }
}
```

Multipart Messages

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPDescStatsRequest`**(*datapath*, *flags=0*, *type_=None*)

Description statistics request message

The controller uses this message to query description of the switch.

Attribute	Description
flags	Zero or `OFPMPF_REQ_MORE`

Example:

```
def send_desc_stats_request(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    req = ofp_parser.OFPDescStatsRequest(datapath, 0)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPDescStatsRequest": {
      "flags": 0,
      "type": 0
   }
}
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPDescStatsReply`**(*datapath*, *type_=None*, ***kwargs*)

Description statistics reply message

The switch responds with this message to a description statistics request.

Attribute	Description
body	Instance of `OFPDescStats`

Example:

```
@set_ev_cls(ofp_event.EventOFPDescStatsReply, MAIN_DISPATCHER)
def desc_stats_reply_handler(self, ev):
    body = ev.msg.body

    self.logger.debug('DescStats: mfr_desc=%s hw_desc=%s sw_desc=%s '
                      'serial_num=%s dp_desc=%s',
                      body.mfr_desc, body.hw_desc, body.sw_desc,
                      body.serial_num, body.dp_desc)
```

JSON Example:

```
{
   "OFPDescStatsReply": {
      "body": {
         "OFPDescStats": {
            "dp_desc": "dp",
            "hw_desc": "hw",
            "mfr_desc": "mfr",
            "serial_num": "serial",
            "sw_desc": "sw"
         }
      },
      "flags": 0,
      "type": 0
   }
}
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPFlowStatsRequest`**(*datapath*, *flags=0*, *table_id=255*, *out_port=4294967295*, *out_group=4294967295*, *cookie=0*, *cookie_mask=0*, *match=None*, *type_=None*)

Individual flow statistics request message

The controller uses this message to query individual flow statistics.

Attribute	Description
flags	Zero or `OFPMPF_REQ_MORE`
table_id	ID of table to read
out_port	Require matching entries to include this as an output port
out_group	Require matching entries to include this as an output group
cookie	Require matching entries to contain this cookie value
cookie_mask	Mask used to restrict the cookie bits that must match
match	Instance of `OFPMatch`

Example:

```
def send_flow_stats_request(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    cookie = cookie_mask = 0
    match = ofp_parser.OFPMatch(in_port=1)
    req = ofp_parser.OFPFlowStatsRequest(datapath, 0,
                                         ofp.OFPTT_ALL,
                                         ofp.OFPP_ANY, ofp.OFPG_ANY,
                                         cookie, cookie_mask,
                                         match)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPFlowStatsRequest": {
      "cookie": 0,
      "cookie_mask": 0,
      "flags": 0,
      "match": {
         "OFPMatch": {
            "length": 4,
            "oxm_fields": [],
            "type": 1
         }
      },
      "out_group": 4294967295,
      "out_port": 4294967295,
      "table_id": 0,
      "type": 1
   }
}
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPFlowStatsReply`**(*datapath*, *type_=None*, ***kwargs*)

Individual flow statistics reply message

The switch responds with this message to an individual flow statistics request.

Attribute	Description
body	List of `OFPFlowStats` instance

Example:

```
@set_ev_cls(ofp_event.EventOFPFlowStatsReply, MAIN_DISPATCHER)
def flow_stats_reply_handler(self, ev):
    flows = []
    for stat in ev.msg.body:
        flows.append('table_id=%s '
                     'duration_sec=%d duration_nsec=%d '
                     'priority=%d '
                     'idle_timeout=%d hard_timeout=%d flags=0x%04x '
                     'cookie=%d packet_count=%d byte_count=%d '
                     'match=%s instructions=%s' %
                     (stat.table_id,
                      stat.duration_sec, stat.duration_nsec,
                      stat.priority,
                      stat.idle_timeout, stat.hard_timeout, stat.flags,
                      stat.cookie, stat.packet_count, stat.byte_count,
                      stat.match, stat.instructions))
    self.logger.debug('FlowStats: %s', flows)
```

JSON Example:

```
{
   "OFPFlowStatsReply": {
      "body": [
         {
            "OFPFlowStats": {
               "byte_count": 0,
               "cookie": 0,
               "duration_nsec": 115277000,
               "duration_sec": 358,
               "flags": 0,
               "hard_timeout": 0,
               "idle_timeout": 0,
               "instructions": [],
               "length": 56,
               "match": {
                  "OFPMatch": {
```

```
            "length": 4,
            "oxm_fields": [],
            "type": 1
         }
      },
      "packet_count": 0,
      "priority": 65535,
      "table_id": 0
   }
},
{
   "OFPFlowStats": {
      "byte_count": 0,
      "cookie": 0,
      "duration_nsec": 115055000,
      "duration_sec": 358,
      "flags": 0,
      "hard_timeout": 0,
      "idle_timeout": 0,
      "instructions": [
         {
            "OFPInstructionActions": {
               "actions": [
                  {
                     "OFPActionOutput": {
                        "len": 16,
                        "max_len": 0,
                        "port": 4294967290,
                        "type": 0
                     }
                  }
               ],
               "len": 24,
               "type": 4
            }
         }
      ],
      "length": 88,
      "match": {
         "OFPMatch": {
            "length": 10,
            "oxm_fields": [
               {
                  "OXMTlv": {
                     "field": "eth_type",
                     "mask": null,
                     "value": 2054
                  }
               }
            ],
            "type": 1
         }
      },
      "packet_count": 0,
      "priority": 65534,
      "table_id": 0
   }
},
{
   "OFPFlowStats": {
      "byte_count": 238,
      "cookie": 0,
      "duration_nsec": 511582000,
      "duration_sec": 316220,
      "flags": 0,
      "hard_timeout": 0,
      "idle_timeout": 0,
      "instructions": [
         {
            "OFPInstructionGotoTable": {
               "len": 8,
               "table_id": 1,
               "type": 1
            }
         }
      ],
      "length": 80,
      "match": {
         "OFPMatch": {
            "length": 22,
            "oxm_fields": [
               {
                  "OXMTlv": {
                     "field": "in_port",
                     "mask": null,
                     "value": 6
                  }
               },
               {
                  "OXMTlv": {
                     "field": "eth_src",
                     "mask": null,
                     "value": "f2:0b:a4:7d:f8:ea"
                  }
               }
            ],
            "type": 1
         }
      },
      "packet_count": 3,
      "priority": 123,
      "table_id": 0
   }
},
{
   "OFPFlowStats": {
      "byte_count": 98,
      "cookie": 0,
      "duration_nsec": 980901000,
      "duration_sec": 313499,
      "flags": 0,
      "hard_timeout": 0,
      "idle_timeout": 0,
      "instructions": [
         {
            "OFPInstructionActions": {
```

```
"actions": [
   {
      "OFPActionSetField": {
         "field": {
            "OXMTlv": {
               "field": "vlan_vid",
               "mask": null,
               "value": 258
            }
         }
      }
   },
   {
      "OFPActionCopyTtlOut": {
         "len": 8,
         "type": 11
      }
   },
   {
      "OFPActionCopyTtlIn": {
         "len": 8,
         "type": 12
      }
   },
   {
      "OFPActionCopyTtlIn": {
         "len": 8,
         "type": 12
      }
   },
   {
      "OFPActionPopPbb": {
         "len": 8,
         "type": 27
      }
   },
   {
      "OFPActionPushPbb": {
         "ethertype": 4660,
         "len": 8,
         "type": 26
      }
   },
   {
      "OFPActionPopMpls": {
         "ethertype": 39030,
         "len": 8,
         "type": 20
      }
   },
   {
      "OFPActionPushMpls": {
         "ethertype": 34887,
         "len": 8,
         "type": 19
      }
   },
   {
```

```
            "OFPActionPopVlan": {
               "len": 8,
               "type": 18
            }
         },
         {
            "OFPActionPushVlan": {
               "ethertype": 33024,
               "len": 8,
               "type": 17
            }
         },
         {
            "OFPActionDecMplsTtl": {
               "len": 8,
               "type": 16
            }
         },
         {
            "OFPActionSetMplsTtl": {
               "len": 8,
               "mpls_ttl": 10,
               "type": 15
            }
         },
         {
            "OFPActionDecNwTtl": {
               "len": 8,
               "type": 24
            }
         },
         {
            "OFPActionSetNwTtl": {
               "len": 8,
               "nw_ttl": 10,
               "type": 23
            }
         },
         {
            "OFPActionSetQueue": {
               "len": 8,
               "queue_id": 3,
               "type": 21
            }
         },
         {
            "OFPActionGroup": {
               "group_id": 99,
               "len": 8,
               "type": 22
            }
         },
         {
            "OFPActionOutput": {
               "len": 16,
               "max_len": 65535,
               "port": 6,
               "type": 0
               }
            }
         ],
         "len": 160,
         "type": 3
      }
   },
   {
      "OFPInstructionActions": {
         "actions": [
            {
               "OFPActionSetField": {
                  "field": {
                     "OXMTlv": {
                        "field": "eth_src",
                        "mask": null,
                        "value": "01:02:03:04:05:06"
                     }
                  }
               }
            },
            {
               "OFPActionSetField": {
                  "field": {
                     "OXMTlv": {
                        "field": "pbb_uca",
                        "mask": null,
                        "value": 1
                     }
                  }
               }
            }
         ],
         "len": 40,
         "type": 4
      }
   },
   {
      "OFPInstructionActions": {
         "actions": [
            {
               "OFPActionOutput": {
                  "len": 16,
                  "max_len": 65535,
                  "port": 4294967293,
                  "type": 0
               }
            }
         ],
         "len": 24,
         "type": 3
      }
   }
],
"length": 280,
"match": {
   "OFPMatch": {
      "length": 4,
                        "oxm_fields": [],
                        "type": 1
                     }
                  },
                  "packet_count": 1,
                  "priority": 0,
                  "table_id": 0
               }
            }
         ],
         "flags": 0,
         "type": 1
      }
   }
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPAggregateStatsRequest`**(*datapath*, *flags*, *table_id*, *out_port*, *out_group*, *cookie*, *cookie_mask*, *match*, *type_=None*)

Aggregate flow statistics request message

The controller uses this message to query aggregate flow statictics.

Attribute	Description
flags	Zero or `OFPMPF_REQ_MORE`
table_id	ID of table to read
out_port	Require matching entries to include this as an output port
out_group	Require matching entries to include this as an output group
cookie	Require matching entries to contain this cookie value
cookie_mask	Mask used to restrict the cookie bits that must match
match	Instance of `OFPMatch`

Example:

```
def send_aggregate_stats_request(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    cookie = cookie_mask = 0
    match = ofp_parser.OFPMatch(in_port=1)
    req = ofp_parser.OFPAggregateStatsRequest(datapath, 0,
                                              ofp.OFPTT_ALL,
                                              ofp.OFPP_ANY,
                                              ofp.OFPG_ANY,
                                              cookie, cookie_mask,
                                              match)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPAggregateStatsRequest": {
      "cookie": 0,
      "cookie_mask": 0,
      "flags": 0,
      "match": {
         "OFPMatch": {
                "length": 4,
                "oxm_fields": [],
                "type": 1
            }
        },
        "out_group": 4294967295,
        "out_port": 4294967295,
        "table_id": 255,
        "type": 2
    }
}
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPAggregateStatsReply`**(*datapath*, *type_=None*, ***kwargs*)

Aggregate flow statistics reply message

The switch responds with this message to an aggregate flow statistics request.

Attribute	Description
body	Instance of `OFPAggregateStats`

Example:

```
@set_ev_cls(ofp_event.EventOFPAggregateStatsReply, MAIN_DISPATCHER)
def aggregate_stats_reply_handler(self, ev):
    body = ev.msg.body

    self.logger.debug('AggregateStats: packet_count=%d byte_count=%d '
                      'flow_count=%d',
                      body.packet_count, body.byte_count,
                      body.flow_count)
```

JSON Example:

```
{
   "OFPAggregateStatsReply": {
      "body": {
         "OFPAggregateStats": {
            "byte_count": 574,
            "flow_count": 6,
            "packet_count": 7
         }
      },
      "flags": 0,
      "type": 2
   }
}
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPTableStatsRequest`**(*datapath*, *flags=0*, *type_=None*)

Table statistics request message

The controller uses this message to query flow table statictics.

Attribute	Description
flags	Zero or `OFPMPF_REQ_MORE`

Example:

```
def send_table_stats_request(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    req = ofp_parser.OFPTableStatsRequest(datapath, 0)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPTableStatsRequest": {
      "flags": 0,
      "type": 3
   }
}
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPTableStatsReply`**(*datapath*, *type_=None*, ***kwargs*)

Table statistics reply message

The switch responds with this message to a table statistics request.

Attribute	Description
body	List of `OFPTableStats` instance

Example:

```
@set_ev_cls(ofp_event.EventOFPTableStatsReply, MAIN_DISPATCHER)
def table_stats_reply_handler(self, ev):
    tables = []
    for stat in ev.msg.body:
        tables.append('table_id=%d active_count=%d lookup_count=%d '
                      ' matched_count=%d' %
                      (stat.table_id, stat.active_count,
                       stat.lookup_count, stat.matched_count))
     self.logger.debug('TableStats: %s', tables)
```

JSON Example:

```
{
   "OFPTableStatsReply": {
      "body": [
         {
            "OFPTableStats": {
               "active_count": 4,
               "lookup_count": 4,
               "matched_count": 4,
               "table_id": 0
            }
         },
         {
            "OFPTableStats": {
               "active_count": 4,
               "lookup_count": 4,
               "matched_count": 4,
               "table_id": 1
            }
         }
      ],
      "flags": 0,
      "type": 3
```

```
    }
}
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPPortStatsRequest`**(*datapath*, *flags=0*, *port_no=4294967295*, *type_=None*)

Port statistics request message

The controller uses this message to query information about ports statistics.

Attribute	Description
flags	Zero or `OFPMPF_REQ_MORE`
port_no	Port number to read (OFPP_ANY to all ports)

Example:

```
def send_port_stats_request(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    req = ofp_parser.OFPPortStatsRequest(datapath, 0, ofp.OFPP_ANY)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPPortStatsRequest": {
      "flags": 0,
      "port_no": 4294967295,
      "type": 4
   }
}
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPPortStatsReply`**(*datapath*, *type_=None*, ***kwargs*)

Port statistics reply message

The switch responds with this message to a port statistics request.

Attribute	Description
body	List of `OFPPortStats` instance

Example:

```
@set_ev_cls(ofp_event.EventOFPPortStatsReply, MAIN_DISPATCHER)
def port_stats_reply_handler(self, ev):
    ports = []
    for stat in ev.msg.body:
        ports.append('port_no=%d '
                     'rx_packets=%d tx_packets=%d '
                     'rx_bytes=%d tx_bytes=%d '
                     'rx_dropped=%d tx_dropped=%d '
                     'rx_errors=%d tx_errors=%d '
                     'rx_frame_err=%d rx_over_err=%d rx_crc_err=%d '
                     'collisions=%d duration_sec=%d duration_nsec=%d' %
                     (stat.port_no,
                      stat.rx_packets, stat.tx_packets,
                      stat.rx_bytes, stat.tx_bytes,
                      stat.rx_dropped, stat.tx_dropped,
                      stat.rx_errors, stat.tx_errors,
                      stat.rx_frame_err, stat.rx_over_err,
                      stat.rx_crc_err, stat.collisions,
                      stat.duration_sec, stat.duration_nsec))
self.logger.debug('PortStats: %s', ports)
```

JSON Example:

```
{
   "OFPPortStatsReply": {
      "body": [
         {
            "OFPPortStats": {
               "collisions": 0,
               "duration_nsec": 0,
               "duration_sec": 0,
               "port_no": 7,
               "rx_bytes": 0,
               "rx_crc_err": 0,
               "rx_dropped": 0,
               "rx_errors": 0,
               "rx_frame_err": 0,
               "rx_over_err": 0,
               "rx_packets": 0,
               "tx_bytes": 336,
               "tx_dropped": 0,
               "tx_errors": 0,
               "tx_packets": 4
            }
         },
         {
            "OFPPortStats": {
               "collisions": 0,
               "duration_nsec": 0,
               "duration_sec": 0,
               "port_no": 6,
               "rx_bytes": 336,
               "rx_crc_err": 0,
               "rx_dropped": 0,
               "rx_errors": 0,
               "rx_frame_err": 0,
               "rx_over_err": 0,
               "rx_packets": 4,
               "tx_bytes": 336,
               "tx_dropped": 0,
               "tx_errors": 0,
               "tx_packets": 4
            }
         }
      ],
      "flags": 0,
      "type": 4
   }
}
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPPortDescStatsRequest`**(*datapath*, *flags=0*, *type_=None*)

Port description request message

The controller uses this message to query description of all the ports.

Attribute	Description
flags	Zero or `OFPMPF_REQ_MORE`

Example:

```
def send_port_desc_stats_request(self, datapath):
    ofp_parser = datapath.ofproto_parser

    req = ofp_parser.OFPPortDescStatsRequest(datapath, 0)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPPortDescStatsRequest": {
      "flags": 0,
      "type": 13
   }
}
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPPortDescStatsReply`**(*datapath*, *type_=None*, ***kwargs*)

Port description reply message

The switch responds with this message to a port description request.

Attribute	Description
body	List of `OFPPortDescStats` instance

Example:

```
@set_ev_cls(ofp_event.EventOFPPortDescStatsReply, MAIN_DISPATCHER)
def port_desc_stats_reply_handler(self, ev):
    ports = []
    for p in ev.msg.body:
        ports.append('port_no=%d hw_addr=%s name=%s config=0x%08x '
                     'state=0x%08x curr=0x%08x advertised=0x%08x '
                     'supported=0x%08x peer=0x%08x curr_speed=%d '
                     'max_speed=%d' %
                     (p.port_no, p.hw_addr,
                      p.name, p.config,
                      p.state, p.curr, p.advertised,
                      p.supported, p.peer, p.curr_speed,
                      p.max_speed))
    self.logger.debug('OFPPortDescStatsReply received: %s', ports)
```

JSON Example:

```
{
   "OFPPortDescStatsReply": {
      "body": [
         {
            "OFPPort": {
               "advertised": 10240,
               "config": 0,
               "curr": 10248,
               "curr_speed": 5000,
               "hw_addr": "f2:0b:a4:d0:3f:70",
               "max_speed": 5000,
               "name": "Port7",
               "peer": 10248,
               "port_no": 7,
               "state": 4,
               "supported": 10248
            }
         },
         {
            "OFPPort": {
               "advertised": 10240,
               "config": 0,
               "curr": 10248,
               "curr_speed": 5000,
               "hw_addr": "f2:0b:a4:7d:f8:ea",
               "max_speed": 5000,
               "name": "Port6",
               "peer": 10248,
               "port_no": 6,
               "state": 4,
               "supported": 10248
            }
         }
      ],
      "flags": 0,
      "type": 13
   }
}
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPQueueStatsRequest`**(*datapath*, *flags=0*, *port_no=4294967295*, *queue_id=4294967295*, *type_=None*)

Queue statistics request message

The controller uses this message to query queue statictics.

Attribute	Description
flags	Zero or `OFPMPF_REQ_MORE`
port_no	Port number to read
queue_id	ID of queue to read

Example:

```
def send_queue_stats_request(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    req = ofp_parser.OFPQueueStatsRequest(datapath, 0, ofp.OFPP_ANY,
                                          ofp.OFPQ_ALL)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPQueueStatsRequest": {
      "flags": 0,
      "port_no": 4294967295,
      "queue_id": 4294967295,
      "type": 5
   }
}
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPQueueStatsReply`**(*datapath*, *type_=None*, ***kwargs*)

Queue statistics reply message

The switch responds with this message to an aggregate flow statistics request.

Attribute	Description
body	List of `OFPQueueStats` instance

Example:

```
@set_ev_cls(ofp_event.EventOFPQueueStatsReply, MAIN_DISPATCHER)
def queue_stats_reply_handler(self, ev):
    queues = []
    for stat in ev.msg.body:
        queues.append('port_no=%d queue_id=%d '
                      'tx_bytes=%d tx_packets=%d tx_errors=%d '
                      'duration_sec=%d duration_nsec=%d' %
                      (stat.port_no, stat.queue_id,
                       stat.tx_bytes, stat.tx_packets, stat.tx_errors,
                       stat.duration_sec, stat.duration_nsec))
    self.logger.debug('QueueStats: %s', queues)
```

JSON Example:

```
{
   "OFPQueueStatsReply": {
      "body": [
         {
            "OFPQueueStats": {
               "duration_nsec": 0,
               "duration_sec": 0,
               "port_no": 7,
               "queue_id": 1,
               "tx_bytes": 0,
               "tx_errors": 0,
               "tx_packets": 0
            }
         },
         {
            "OFPQueueStats": {
               "duration_nsec": 0,
               "duration_sec": 0,
               "port_no": 6,
               "queue_id": 1,
               "tx_bytes": 0,
               "tx_errors": 0,
               "tx_packets": 0
            }
         },
         {
            "OFPQueueStats": {
               "duration_nsec": 0,
               "duration_sec": 0,
               "port_no": 7,
               "queue_id": 2,
               "tx_bytes": 0,
               "tx_errors": 0,
               "tx_packets": 0
            }
            }
        ],
        "flags": 0,
        "type": 5
    }
}
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPGroupStatsRequest`**(*datapath*, *flags=0*, *group_id=4294967292*, *type_=None*)

Group statistics request message

The controller uses this message to query statistics of one or more groups.

Attribute	Description
flags	Zero or `OFPMPF_REQ_MORE`
group_id	ID of group to read (OFPG_ALL to all groups)

Example:

```
def send_group_stats_request(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    req = ofp_parser.OFPGroupStatsRequest(datapath, 0, ofp.OFPG_ALL)
    datapath.send_msg(req)
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPGroupStatsReply`**(*datapath*, *type_=None*, ***kwargs*)

Group statistics reply message

The switch responds with this message to a group statistics request.

Attribute	Description
body	List of `OFPGroupStats` instance

Example:

```
@set_ev_cls(ofp_event.EventOFPGroupStatsReply, MAIN_DISPATCHER)
def group_stats_reply_handler(self, ev):
    groups = []
    for stat in ev.msg.body:
        groups.append('length=%d group_id=%d '
                      'ref_count=%d packet_count=%d byte_count=%d '
                      'duration_sec=%d duration_nsec=%d' %
                      (stat.length, stat.group_id,
                       stat.ref_count, stat.packet_count,
                       stat.byte_count, stat.duration_sec,
                       stat.duration_nsec))
    self.logger.debug('GroupStats: %s', groups)
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPGroupDescStatsRequest`**(*datapath*, *flags=0*, *type_=None*)

Group description request message

The controller uses this message to list the set of groups on a switch.

Attribute	Description
flags	Zero or `OFPMPF_REQ_MORE`

Example:

```
def send_group_desc_stats_request(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    req = ofp_parser.OFPGroupDescStatsRequest(datapath, 0)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPGroupDescStatsRequest": {
      "flags": 0,
      "type": 7
   }
}
```

class ryu.ofproto.ofproto_v1_3_parser.**OFPGroupDescStatsReply**(*datapath*, *type_=None*, ***kwargs*)

Group description reply message

The switch responds with this message to a group description request.

Attribute	Description
body	List of `OFPGroupDescStats` instance

Example:

```
@set_ev_cls(ofp_event.EventOFPGroupDescStatsReply, MAIN_DISPATCHER)
def group_desc_stats_reply_handler(self, ev):
    descs = []
    for stat in ev.msg.body:
        descs.append('length=%d type=%d group_id=%d '
                     'buckets=%s' %
                     (stat.length, stat.type, stat.group_id,
                      stat.bucket))
    self.logger.debug('GroupDescStats: %s', groups)
```

JSON Example:

```
{
   "OFPGroupDescStatsReply": {
      "body": [
         {
            "OFPGroupDescStats": {
               "buckets": [
                  {
                     "OFPBucket": {
                        "actions": [
                           {
                              "OFPActionOutput": {
                                 "len": 16,
                                 "max_len": 65535,
                                 "port": 2,
                                 "type": 0
                              }
                           }
                        ],
                        "len": 32,
                        "watch_group": 1,
                        "watch_port": 1,
                        "weight": 1
                    }
                }
            ],
            "group_id": 1,
            "length": 40,
            "type": 0
        }
    }
],
"flags": 0,
"type": 7
    }
}
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPGroupFeaturesStatsRequest`**(*datapath*, *flags=0*, *type_=None*)

Group features request message

The controller uses this message to list the capabilities of groups on a switch.

Attribute	Description
flags	Zero or `OFPMPF_REQ_MORE`

Example:

```
def send_group_features_stats_request(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    req = ofp_parser.OFPGroupFeaturesStatsRequest(datapath, 0)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPGroupFeaturesStatsRequest": {
      "flags": 0,
      "type": 8
   }
}
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPGroupFeaturesStatsReply`**(*datapath*, *type_=None*, ***kwargs*)

Group features reply message

The switch responds with this message to a group features request.

Attribute	Description
body	Instance of `OFPGroupFeaturesStats`

Example:

```
@set_ev_cls(ofp_event.EventOFPGroupFeaturesStatsReply, MAIN_DISPATCHER)
def group_features_stats_reply_handler(self, ev):
    body = ev.msg.body

    self.logger.debug('GroupFeaturesStats: types=%d '
                      'capabilities=0x%08x max_groups=%s '
```

```
                              'actions=%s',
                              body.types, body.capabilities,
                              body.max_groups, body.actions)
```

JSON Example:

```
{
   "OFPGroupFeaturesStatsReply": {
      "body": {
         "OFPGroupFeaturesStats": {
            "actions": [
               67082241,
               67082241,
               67082241,
               67082241
            ],
            "capabilities": 5,
            "max_groups": [
               16777216,
               16777216,
               16777216,
               16777216
            ],
            "types": 15
         }
      },
      "flags": 0,
      "type": 8
   }
}
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPMeterStatsRequest`**(*datapath*, *flags=0*, *meter_id=4294967295*, *type_=None*)

Meter statistics request message

The controller uses this message to query statistics for one or more meters.

Attribute	Description
flags	Zero or `OFPMPF_REQ_MORE`
meter_id	ID of meter to read (OFPM_ALL to all meters)

Example:

```python
def send_meter_stats_request(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    req = ofp_parser.OFPMeterStatsRequest(datapath, 0, ofp.OFPM_ALL)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPMeterStatsRequest": {
      "flags": 0,
      "meter_id": 4294967295,
      "type": 9
   }
}
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPMeterStatsReply`**(*datapath*, *type_=None*, ***kwargs*)

Meter statistics reply message

The switch responds with this message to a meter statistics request.

Attribute	Description
body	List of `OFPMeterStats` instance

Example:

```
@set_ev_cls(ofp_event.EventOFPMeterStatsReply, MAIN_DISPATCHER)
def meter_stats_reply_handler(self, ev):
    meters = []
    for stat in ev.msg.body:
        meters.append('meter_id=0x%08x len=%d flow_count=%d '
                      'packet_in_count=%d byte_in_count=%d '
                      'duration_sec=%d duration_nsec=%d '
                      'band_stats=%s' %
                      (stat.meter_id, stat.len, stat.flow_count,
                       stat.packet_in_count, stat.byte_in_count,
                       stat.duration_sec, stat.duration_nsec,
                       stat.band_stats))
    self.logger.debug('MeterStats: %s', meters)
```

JSON Example:

```
{
   "OFPMeterStatsReply": {
      "body": [
         {
            "OFPMeterStats": {
               "band_stats": [
                  {
                     "OFPMeterBandStats": {
                        "byte_band_count": 0,
                        "packet_band_count": 0
                     }
                  }
               ],
               "byte_in_count": 0,
               "duration_nsec": 480000,
               "duration_sec": 0,
               "flow_count": 0,
               "len": 56,
               "meter_id": 100,
               "packet_in_count": 0
            }
         }
      ],
      "flags": 0,
      "type": 9
   }
}
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPMeterConfigStatsRequest`**(*datapath*, *flags=0*, *meter_id=4294967295*, *type_=None*)

Meter configuration statistics request message

The controller uses this message to query configuration for one or more meters.

Attribute	Description
flags	Zero or `OFPMPF_REQ_MORE`
meter_id	ID of meter to read (OFPM_ALL to all meters)

Example:

```
def send_meter_config_stats_request(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    req = ofp_parser.OFPMeterConfigStatsRequest(datapath, 0,
                                                ofp.OFPM_ALL)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPMeterConfigStatsRequest": {
      "flags": 0,
      "meter_id": 4294967295,
      "type": 10
   }
}
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPMeterConfigStatsReply`**(*datapath*, *type_=None*, ***kwargs*)

Meter configuration statistics reply message

The switch responds with this message to a meter configuration statistics request.

Attribute	Description
body	List of `OFPMeterConfigStats` instance

Example:

```
@set_ev_cls(ofp_event.EventOFPMeterConfigStatsReply, MAIN_DISPATCHER)
def meter_config_stats_reply_handler(self, ev):
    configs = []
    for stat in ev.msg.body:
        configs.append('length=%d flags=0x%04x meter_id=0x%08x '
                       'bands=%s' %
                       (stat.length, stat.flags, stat.meter_id,
                        stat.bands))
    self.logger.debug('MeterConfigStats: %s', configs)
```

JSON Example:

```
{
   "OFPMeterConfigStatsReply": {
      "body": [
         {
            "OFPMeterConfigStats": {
               "bands": [
                  {
                     "OFPMeterBandDrop": {
                        "burst_size": 10,
                        "len": 16,
                        "rate": 1000,
                        "type": 1
                    }
                }
            ],
            "flags": 14,
            "length": 24,
            "meter_id": 100
        }
    }
  ],
  "flags": 0,
  "type": 10
 }
}
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPMeterFeaturesStatsRequest`**(*datapath*, *flags=0*, *type_=None*)

Meter features statistics request message

The controller uses this message to query the set of features of the metering subsystem.

Attribute	Description
flags	Zero or `OFPMPF_REQ_MORE`

Example:

```
def send_meter_features_stats_request(self, datapath):
    ofp_parser = datapath.ofproto_parser

    req = ofp_parser.OFPMeterFeaturesStatsRequest(datapath, 0)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPMeterFeaturesStatsRequest": {
      "flags": 0,
      "type": 11
   }
}
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPMeterFeaturesStatsReply`**(*datapath*, *type_=None*, ***kwargs*)

Meter features statistics reply message

The switch responds with this message to a meter features statistics request.

Attribute	Description
body	List of `OFPMeterFeaturesStats` instance

Example:

```
@set_ev_cls(ofp_event.EventOFPMeterFeaturesStatsReply, MAIN_DISPATCHER)
def meter_features_stats_reply_handler(self, ev):
    features = []
    for stat in ev.msg.body:
        features.append('max_meter=%d band_types=0x%08x '
                        'capabilities=0x%08x max_band=%d '
                        'max_color=%d' %
```

```
                              (stat.max_meter, stat.band_types,
                               stat.capabilities, stat.max_band,
                               stat.max_color))
    self.logger.debug('MeterFeaturesStats: %s', configs)
```

JSON Example:

```
{
   "OFPMeterFeaturesStatsReply": {
      "body": [
         {
            "OFPMeterFeaturesStats": {
               "band_types": 2147483654,
               "capabilities": 15,
               "max_band": 255,
               "max_color": 0,
               "max_meter": 16777216
            }
         }
      ],
      "flags": 0,
      "type": 11
   }
}
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPTableFeaturesStatsRequest`**(*datapath*, *flags=0*, *body=*[ ], *type_=None*)

Table features statistics request message

The controller uses this message to query table features.

Attribute	Description
body	List of `OFPTableFeaturesStats` instances. The default is [].

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPTableFeaturesStatsReply`**(*datapath*, *type_=None*, ***kwargs*)

Table features statistics reply message

The switch responds with this message to a table features statistics request.

This implmentation is still incomplete. Namely, this implementation does not parse `properties` list and always reports it empty.

Attribute	Description
body	List of `OFPTableFeaturesStats` instance

JSON Example:

```
{
   "OFPTableFeaturesStatsReply": {
      "body": [
         {
            "OFPTableFeaturesStats": {
               "config": 0,
               "length": 1112,
               "max_entries": 16777216,
               "metadata_match": 18446744073709551615,
               "metadata_write": 18446744073709551615,
"name": "\u79c1\u306e\u30c6\u30fc\u30d6\u30eb",
"properties": [
   {
      "OFPTableFeaturePropInstructions": {
         "instruction_ids": [
            {
               "OFPInstructionId": {
                  "len": 4,
                  "type": 1
               }
            },
            {
               "OFPInstructionId": {
                  "len": 4,
                  "type": 2
               }
            },
            {
               "OFPInstructionId": {
                  "len": 4,
                  "type": 3
               }
            },
            {
               "OFPInstructionId": {
                  "len": 4,
                  "type": 4
               }
            },
            {
               "OFPInstructionId": {
                  "len": 4,
                  "type": 5
               }
            },
            {
               "OFPInstructionId": {
                  "len": 4,
                  "type": 6
               }
            }
         ],
         "length": 28,
         "type": 0
      }
   },
   {
      "OFPTableFeaturePropNextTables": {
         "length": 258,
         "table_ids": [
            1,
            2,
            3,
            4,
            5,
            6,
            7,
            8,
                                9,
                                10,
                                11,
                                12,
                                13,
                                14,
                                15,
                                16,
                                17,
                                18,
                                19,
                                20,
                                21,
                                22,
                                23,
                                24,
                                25,
                                26,
                                27,
                                28,
                                29,
                                30,
                                31,
                                32,
                                33,
                                34,
                                35,
                                36,
                                37,
                                38,
                                39,
                                40,
                                41,
                                42,
                                43,
                                44,
                                45,
                                46,
                                47,
                                48,
                                49,
                                50,
                                51,
                                52,
                                53,
                                54,
                                55,
                                56,
                                57,
                                58,
                                59,
                                60,
                                61,
                                62,
                                63,
                                64,
                                65,
                                66,
                                 67,
                                 68,
                                 69,
                                 70,
                                 71,
                                 72,
                                 73,
                                 74,
                                 75,
                                 76,
                                 77,
                                 78,
                                 79,
                                 80,
                                 81,
                                 82,
                                 83,
                                 84,
                                 85,
                                 86,
                                 87,
                                 88,
                                 89,
                                 90,
                                 91,
                                 92,
                                 93,
                                 94,
                                 95,
                                 96,
                                 97,
                                 98,
                                 99,
                                 100,
                                 101,
                                 102,
                                 103,
                                 104,
                                 105,
                                 106,
                                 107,
                                 108,
                                 109,
                                 110,
                                 111,
                                 112,
                                 113,
                                 114,
                                 115,
                                 116,
                                 117,
                                 118,
                                 119,
                                 120,
                                 121,
                                 122,
                                 123,
                                 124,
125,
126,
127,
128,
129,
130,
131,
132,
133,
134,
135,
136,
137,
138,
139,
140,
141,
142,
143,
144,
145,
146,
147,
148,
149,
150,
151,
152,
153,
154,
155,
156,
157,
158,
159,
160,
161,
162,
163,
164,
165,
166,
167,
168,
169,
170,
171,
172,
173,
174,
175,
176,
177,
178,
179,
180,
181,
182,
                                    183,
                                    184,
                                    185,
                                    186,
                                    187,
                                    188,
                                    189,
                                    190,
                                    191,
                                    192,
                                    193,
                                    194,
                                    195,
                                    196,
                                    197,
                                    198,
                                    199,
                                    200,
                                    201,
                                    202,
                                    203,
                                    204,
                                    205,
                                    206,
                                    207,
                                    208,
                                    209,
                                    210,
                                    211,
                                    212,
                                    213,
                                    214,
                                    215,
                                    216,
                                    217,
                                    218,
                                    219,
                                    220,
                                    221,
                                    222,
                                    223,
                                    224,
                                    225,
                                    226,
                                    227,
                                    228,
                                    229,
                                    230,
                                    231,
                                    232,
                                    233,
                                    234,
                                    235,
                                    236,
                                    237,
                                    238,
                                    239,
                                    240,
         241,
         242,
         243,
         244,
         245,
         246,
         247,
         248,
         249,
         250,
         251,
         252,
         253,
         254
      ],
      "type": 2
   }
},
{
   "OFPTableFeaturePropActions": {
      "action_ids": [
         {
            "OFPActionId": {
               "len": 4,
               "type": 0
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 22
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 21
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 15
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 16
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 23
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 24
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 11
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 12
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 17
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 18
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 19
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 20
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 26
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 27
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 25
            }
         }
      ],
      "length": 68,
      "type": 4
   }
},
{
   "OFPTableFeaturePropActions": {
      "action_ids": [
         {
            "OFPActionId": {
               "len": 4,
               "type": 0
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 22
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 21
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 15
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 16
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 23
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 24
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 11
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 12
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 17
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 18
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 19
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 20
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 26
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 27
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 25
            }
         }
      ],
      "length": 68,
      "type": 6
   }
},
{
   "OFPTableFeaturePropOxm": {
      "length": 152,
      "oxm_ids": [
         {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "in_port"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "metadata"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "eth_dst"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "eth_src"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "eth_type"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "vlan_vid"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "vlan_pcp"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "ip_dscp"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
      "type": "ip_ecn"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "ip_proto"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "ipv4_src"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "ipv4_dst"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "tcp_src"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "tcp_dst"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "udp_src"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "udp_dst"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "sctp_src"
   }
```

```
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "sctp_dst"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "icmpv4_type"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "icmpv4_code"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "arp_op"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "arp_spa"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "arp_tpa"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "arp_sha"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "arp_tha"
   }
},
{
```

```
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "ipv6_src"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "ipv6_dst"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "ipv6_flabel"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "icmpv6_type"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "icmpv6_code"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "ipv6_nd_target"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "ipv6_nd_sll"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "ipv6_nd_tll"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "mpls_label"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "mpls_tc"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "mpls_bos"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "pbb_isid"
            }
         }
      ],
      "type": 8
   }
},
{
   "OFPTableFeaturePropOxm": {
      "length": 152,
      "oxm_ids": [
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "in_port"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "metadata"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "eth_dst"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
            "type": "eth_src"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "eth_type"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "vlan_vid"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "vlan_pcp"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "ip_dscp"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "ip_ecn"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "ip_proto"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "ipv4_src"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "ipv4_dst"
        }
```

```
            },
            {
               "OFPOxmId": {
                  "hasmask": 0,
                  "length": 0,
                  "type": "tcp_src"
               }
            },
            {
               "OFPOxmId": {
                  "hasmask": 0,
                  "length": 0,
                  "type": "tcp_dst"
               }
            },
            {
               "OFPOxmId": {
                  "hasmask": 0,
                  "length": 0,
                  "type": "udp_src"
               }
            },
            {
               "OFPOxmId": {
                  "hasmask": 0,
                  "length": 0,
                  "type": "udp_dst"
               }
            },
            {
               "OFPOxmId": {
                  "hasmask": 0,
                  "length": 0,
                  "type": "sctp_src"
               }
            },
            {
               "OFPOxmId": {
                  "hasmask": 0,
                  "length": 0,
                  "type": "sctp_dst"
               }
            },
            {
               "OFPOxmId": {
                  "hasmask": 0,
                  "length": 0,
                  "type": "icmpv4_type"
               }
            },
            {
               "OFPOxmId": {
                  "hasmask": 0,
                  "length": 0,
                  "type": "icmpv4_code"
               }
            },
            {
```

```
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "arp_op"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "arp_spa"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "arp_tpa"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "arp_sha"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "arp_tha"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "ipv6_src"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "ipv6_dst"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "ipv6_flabel"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
                "length": 0,
                "type": "icmpv6_type"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "icmpv6_code"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "ipv6_nd_target"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "ipv6_nd_sll"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "ipv6_nd_tll"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "mpls_label"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "mpls_tc"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "mpls_bos"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "pbb_isid"
            }
         }
      ],
      "type": 10
   }
},
{
   "OFPTableFeaturePropOxm": {
      "length": 152,
      "oxm_ids": [
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "in_port"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "metadata"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "eth_dst"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "eth_src"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "eth_type"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "vlan_vid"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "vlan_pcp"
            }
```

```
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "ip_dscp"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "ip_ecn"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "ip_proto"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "ipv4_src"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "ipv4_dst"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "tcp_src"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "tcp_dst"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "udp_src"
   }
},
{
```

```
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "udp_dst"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "sctp_src"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "sctp_dst"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "icmpv4_type"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "icmpv4_code"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "arp_op"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "arp_spa"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "arp_tpa"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
                    "length": 0,
                    "type": "arp_sha"
                }
            },
            {
                "OFPOxmId": {
                    "hasmask": 0,
                    "length": 0,
                    "type": "arp_tha"
                }
            },
            {
                "OFPOxmId": {
                    "hasmask": 0,
                    "length": 0,
                    "type": "ipv6_src"
                }
            },
            {
                "OFPOxmId": {
                    "hasmask": 0,
                    "length": 0,
                    "type": "ipv6_dst"
                }
            },
            {
                "OFPOxmId": {
                    "hasmask": 0,
                    "length": 0,
                    "type": "ipv6_flabel"
                }
            },
            {
                "OFPOxmId": {
                    "hasmask": 0,
                    "length": 0,
                    "type": "icmpv6_type"
                }
            },
            {
                "OFPOxmId": {
                    "hasmask": 0,
                    "length": 0,
                    "type": "icmpv6_code"
                }
            },
            {
                "OFPOxmId": {
                    "hasmask": 0,
                    "length": 0,
                    "type": "ipv6_nd_target"
                }
            },
            {
                "OFPOxmId": {
                    "hasmask": 0,
                    "length": 0,
                    "type": "ipv6_nd_sll"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "ipv6_nd_tll"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "mpls_label"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "mpls_tc"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "mpls_bos"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "pbb_isid"
            }
         }
      ],
      "type": 12
   }
},
{
   "OFPTableFeaturePropOxm": {
      "length": 152,
      "oxm_ids": [
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "in_port"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "metadata"
            }
```

```
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "eth_dst"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "eth_src"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "eth_type"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "vlan_vid"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "vlan_pcp"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "ip_dscp"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "ip_ecn"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "ip_proto"
   }
},
{
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "ipv4_src"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "ipv4_dst"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "tcp_src"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "tcp_dst"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "udp_src"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "udp_dst"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "sctp_src"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "sctp_dst"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
                "length": 0,
                "type": "icmpv4_type"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "icmpv4_code"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "arp_op"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "arp_spa"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "arp_tpa"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "arp_sha"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "arp_tha"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "ipv6_src"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "ipv6_dst"
```

```
      }
   },
   {
      "OFPOxmId": {
         "hasmask": 0,
         "length": 0,
         "type": "ipv6_flabel"
      }
   },
   {
      "OFPOxmId": {
         "hasmask": 0,
         "length": 0,
         "type": "icmpv6_type"
      }
   },
   {
      "OFPOxmId": {
         "hasmask": 0,
         "length": 0,
         "type": "icmpv6_code"
      }
   },
   {
      "OFPOxmId": {
         "hasmask": 0,
         "length": 0,
         "type": "ipv6_nd_target"
      }
   },
   {
      "OFPOxmId": {
         "hasmask": 0,
         "length": 0,
         "type": "ipv6_nd_sll"
      }
   },
   {
      "OFPOxmId": {
         "hasmask": 0,
         "length": 0,
         "type": "ipv6_nd_tll"
      }
   },
   {
      "OFPOxmId": {
         "hasmask": 0,
         "length": 0,
         "type": "mpls_label"
      }
   },
   {
      "OFPOxmId": {
         "hasmask": 0,
         "length": 0,
         "type": "mpls_tc"
      }
   },
                              {
                                 "OFPOxmId": {
                                    "hasmask": 0,
                                    "length": 0,
                                    "type": "mpls_bos"
                                 }
                              },
                              {
                                 "OFPOxmId": {
                                    "hasmask": 0,
                                    "length": 0,
                                    "type": "pbb_isid"
                                 }
                              }
                           ],
                           "type": 14
                        }
                     }
                  ],
                  "table_id": 0
               }
            },
            {
               "OFPTableFeaturesStats": {
                  "config": 0,
                  "length": 1112,
                  "max_entries": 16777216,
                  "metadata_match": 18446744073709551615,
                  "metadata_write": 18446744073709551615,
                  "name": "Flow Table 0x01",
                  "properties": [
                     {
                        "OFPTableFeaturePropInstructions": {
                           "instruction_ids": [
                              {
                                 "OFPInstructionId": {
                                    "len": 4,
                                    "type": 1
                                 }
                              },
                              {
                                 "OFPInstructionId": {
                                    "len": 4,
                                    "type": 2
                                 }
                              },
                              {
                                 "OFPInstructionId": {
                                    "len": 4,
                                    "type": 3
                                 }
                              },
                              {
                                 "OFPInstructionId": {
                                    "len": 4,
                                    "type": 4
                                 }
                              },
                        {
                           "OFPInstructionId": {
                              "len": 4,
                              "type": 5
                           }
                        },
                        {
                           "OFPInstructionId": {
                              "len": 4,
                              "type": 6
                           }
                        }
                     ],
                     "length": 28,
                     "type": 0
                  }
               },
               {
                  "OFPTableFeaturePropNextTables": {
                     "length": 257,
                     "table_ids": [
                        2,
                        3,
                        4,
                        5,
                        6,
                        7,
                        8,
                        9,
                        10,
                        11,
                        12,
                        13,
                        14,
                        15,
                        16,
                        17,
                        18,
                        19,
                        20,
                        21,
                        22,
                        23,
                        24,
                        25,
                        26,
                        27,
                        28,
                        29,
                        30,
                        31,
                        32,
                        33,
                        34,
                        35,
                        36,
                        37,
                        38,
                                         39,
                                         40,
                                         41,
                                         42,
                                         43,
                                         44,
                                         45,
                                         46,
                                         47,
                                         48,
                                         49,
                                         50,
                                         51,
                                         52,
                                         53,
                                         54,
                                         55,
                                         56,
                                         57,
                                         58,
                                         59,
                                         60,
                                         61,
                                         62,
                                         63,
                                         64,
                                         65,
                                         66,
                                         67,
                                         68,
                                         69,
                                         70,
                                         71,
                                         72,
                                         73,
                                         74,
                                         75,
                                         76,
                                         77,
                                         78,
                                         79,
                                         80,
                                         81,
                                         82,
                                         83,
                                         84,
                                         85,
                                         86,
                                         87,
                                         88,
                                         89,
                                         90,
                                         91,
                                         92,
                                         93,
                                         94,
                                         95,
                                         96,
                                    97,
                                    98,
                                    99,
                                    100,
                                    101,
                                    102,
                                    103,
                                    104,
                                    105,
                                    106,
                                    107,
                                    108,
                                    109,
                                    110,
                                    111,
                                    112,
                                    113,
                                    114,
                                    115,
                                    116,
                                    117,
                                    118,
                                    119,
                                    120,
                                    121,
                                    122,
                                    123,
                                    124,
                                    125,
                                    126,
                                    127,
                                    128,
                                    129,
                                    130,
                                    131,
                                    132,
                                    133,
                                    134,
                                    135,
                                    136,
                                    137,
                                    138,
                                    139,
                                    140,
                                    141,
                                    142,
                                    143,
                                    144,
                                    145,
                                    146,
                                    147,
                                    148,
                                    149,
                                    150,
                                    151,
                                    152,
                                    153,
                                    154,
                                        155,
                                        156,
                                        157,
                                        158,
                                        159,
                                        160,
                                        161,
                                        162,
                                        163,
                                        164,
                                        165,
                                        166,
                                        167,
                                        168,
                                        169,
                                        170,
                                        171,
                                        172,
                                        173,
                                        174,
                                        175,
                                        176,
                                        177,
                                        178,
                                        179,
                                        180,
                                        181,
                                        182,
                                        183,
                                        184,
                                        185,
                                        186,
                                        187,
                                        188,
                                        189,
                                        190,
                                        191,
                                        192,
                                        193,
                                        194,
                                        195,
                                        196,
                                        197,
                                        198,
                                        199,
                                        200,
                                        201,
                                        202,
                                        203,
                                        204,
                                        205,
                                        206,
                                        207,
                                        208,
                                        209,
                                        210,
                                        211,
                                        212,
                                 213,
                                 214,
                                 215,
                                 216,
                                 217,
                                 218,
                                 219,
                                 220,
                                 221,
                                 222,
                                 223,
                                 224,
                                 225,
                                 226,
                                 227,
                                 228,
                                 229,
                                 230,
                                 231,
                                 232,
                                 233,
                                 234,
                                 235,
                                 236,
                                 237,
                                 238,
                                 239,
                                 240,
                                 241,
                                 242,
                                 243,
                                 244,
                                 245,
                                 246,
                                 247,
                                 248,
                                 249,
                                 250,
                                 251,
                                 252,
                                 253,
                                 254
                              ],
                              "type": 2
                           }
                        },
                        {
                           "OFPTableFeaturePropActions": {
                              "action_ids": [
                                 {
                                    "OFPActionId": {
                                       "len": 4,
                                       "type": 0
                                    }
                                 },
                                 {
                                    "OFPActionId": {
                                       "len": 4,
                "type": 22
            }
        },
        {
            "OFPActionId": {
                "len": 4,
                "type": 21
            }
        },
        {
            "OFPActionId": {
                "len": 4,
                "type": 15
            }
        },
        {
            "OFPActionId": {
                "len": 4,
                "type": 16
            }
        },
        {
            "OFPActionId": {
                "len": 4,
                "type": 23
            }
        },
        {
            "OFPActionId": {
                "len": 4,
                "type": 24
            }
        },
        {
            "OFPActionId": {
                "len": 4,
                "type": 11
            }
        },
        {
            "OFPActionId": {
                "len": 4,
                "type": 12
            }
        },
        {
            "OFPActionId": {
                "len": 4,
                "type": 17
            }
        },
        {
            "OFPActionId": {
                "len": 4,
                "type": 18
            }
        },
        {
            "OFPActionId": {
               "len": 4,
               "type": 19
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 20
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 26
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 27
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 25
            }
         }
      ],
      "length": 68,
      "type": 4
   }
},
{
   "OFPTableFeaturePropActions": {
      "action_ids": [
         {
            "OFPActionId": {
               "len": 4,
               "type": 0
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 22
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 21
            }
         },
         {
            "OFPActionId": {
               "len": 4,
            "type": 15
        }
    },
    {
        "OFPActionId": {
            "len": 4,
            "type": 16
        }
    },
    {
        "OFPActionId": {
            "len": 4,
            "type": 23
        }
    },
    {
        "OFPActionId": {
            "len": 4,
            "type": 24
        }
    },
    {
        "OFPActionId": {
            "len": 4,
            "type": 11
        }
    },
    {
        "OFPActionId": {
            "len": 4,
            "type": 12
        }
    },
    {
        "OFPActionId": {
            "len": 4,
            "type": 17
        }
    },
    {
        "OFPActionId": {
            "len": 4,
            "type": 18
        }
    },
    {
        "OFPActionId": {
            "len": 4,
            "type": 19
        }
    },
    {
        "OFPActionId": {
            "len": 4,
            "type": 20
        }
    },
    {
```

```
                        "OFPActionId": {
                           "len": 4,
                           "type": 26
                        }
                     },
                     {
                        "OFPActionId": {
                           "len": 4,
                           "type": 27
                        }
                     },
                     {
                        "OFPActionId": {
                           "len": 4,
                           "type": 25
                        }
                     }
                  ],
                  "length": 68,
                  "type": 6
               }
            },
            {
               "OFPTableFeaturePropOxm": {
                  "length": 152,
                  "oxm_ids": [
                     {
                        "OFPOxmId": {
                           "hasmask": 0,
                           "length": 0,
                           "type": "in_port"
                        }
                     },
                     {
                        "OFPOxmId": {
                           "hasmask": 0,
                           "length": 0,
                           "type": "metadata"
                        }
                     },
                     {
                        "OFPOxmId": {
                           "hasmask": 0,
                           "length": 0,
                           "type": "eth_dst"
                        }
                     },
                     {
                        "OFPOxmId": {
                           "hasmask": 0,
                           "length": 0,
                           "type": "eth_src"
                        }
                     },
                     {
                        "OFPOxmId": {
                           "hasmask": 0,
                           "length": 0,
                "type": "eth_type"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "vlan_vid"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "vlan_pcp"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "ip_dscp"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "ip_ecn"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "ip_proto"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "ipv4_src"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "ipv4_dst"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "tcp_src"
            }
```

```
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "tcp_dst"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "udp_src"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "udp_dst"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "sctp_src"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "sctp_dst"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "icmpv4_type"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "icmpv4_code"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "arp_op"
   }
},
{
```

```
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "arp_spa"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "arp_tpa"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "arp_sha"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "arp_tha"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "ipv6_src"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "ipv6_dst"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "ipv6_flabel"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "icmpv6_type"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
                    "length": 0,
                    "type": "icmpv6_code"
                }
            },
            {
                "OFPOxmId": {
                    "hasmask": 0,
                    "length": 0,
                    "type": "ipv6_nd_target"
                }
            },
            {
                "OFPOxmId": {
                    "hasmask": 0,
                    "length": 0,
                    "type": "ipv6_nd_sll"
                }
            },
            {
                "OFPOxmId": {
                    "hasmask": 0,
                    "length": 0,
                    "type": "ipv6_nd_tll"
                }
            },
            {
                "OFPOxmId": {
                    "hasmask": 0,
                    "length": 0,
                    "type": "mpls_label"
                }
            },
            {
                "OFPOxmId": {
                    "hasmask": 0,
                    "length": 0,
                    "type": "mpls_tc"
                }
            },
            {
                "OFPOxmId": {
                    "hasmask": 0,
                    "length": 0,
                    "type": "mpls_bos"
                }
            },
            {
                "OFPOxmId": {
                    "hasmask": 0,
                    "length": 0,
                    "type": "pbb_isid"
                }
            }
        ],
        "type": 8
    }
},
{
```

```
                    "OFPTableFeaturePropOxm": {
                       "length": 152,
                       "oxm_ids": [
                          {
                             "OFPOxmId": {
                                "hasmask": 0,
                                "length": 0,
                                "type": "in_port"
                             }
                          },
                          {
                             "OFPOxmId": {
                                "hasmask": 0,
                                "length": 0,
                                "type": "metadata"
                             }
                          },
                          {
                             "OFPOxmId": {
                                "hasmask": 0,
                                "length": 0,
                                "type": "eth_dst"
                             }
                          },
                          {
                             "OFPOxmId": {
                                "hasmask": 0,
                                "length": 0,
                                "type": "eth_src"
                             }
                          },
                          {
                             "OFPOxmId": {
                                "hasmask": 0,
                                "length": 0,
                                "type": "eth_type"
                             }
                          },
                          {
                             "OFPOxmId": {
                                "hasmask": 0,
                                "length": 0,
                                "type": "vlan_vid"
                             }
                          },
                          {
                             "OFPOxmId": {
                                "hasmask": 0,
                                "length": 0,
                                "type": "vlan_pcp"
                             }
                          },
                          {
                             "OFPOxmId": {
                                "hasmask": 0,
                                "length": 0,
                                "type": "ip_dscp"
                             }
```

```
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "ip_ecn"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "ip_proto"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "ipv4_src"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "ipv4_dst"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "tcp_src"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "tcp_dst"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "udp_src"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "udp_dst"
   }
},
{
```

```
                "OFPOxmId": {
                   "hasmask": 0,
                   "length": 0,
                   "type": "sctp_src"
                }
             },
             {
                "OFPOxmId": {
                   "hasmask": 0,
                   "length": 0,
                   "type": "sctp_dst"
                }
             },
             {
                "OFPOxmId": {
                   "hasmask": 0,
                   "length": 0,
                   "type": "icmpv4_type"
                }
             },
             {
                "OFPOxmId": {
                   "hasmask": 0,
                   "length": 0,
                   "type": "icmpv4_code"
                }
             },
             {
                "OFPOxmId": {
                   "hasmask": 0,
                   "length": 0,
                   "type": "arp_op"
                }
             },
             {
                "OFPOxmId": {
                   "hasmask": 0,
                   "length": 0,
                   "type": "arp_spa"
                }
             },
             {
                "OFPOxmId": {
                   "hasmask": 0,
                   "length": 0,
                   "type": "arp_tpa"
                }
             },
             {
                "OFPOxmId": {
                   "hasmask": 0,
                   "length": 0,
                   "type": "arp_sha"
                }
             },
             {
                "OFPOxmId": {
                   "hasmask": 0,
            "length": 0,
            "type": "arp_tha"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "ipv6_src"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "ipv6_dst"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "ipv6_flabel"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "icmpv6_type"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "icmpv6_code"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "ipv6_nd_target"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "ipv6_nd_sll"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "ipv6_nd_tll"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "mpls_label"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "mpls_tc"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "mpls_bos"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "pbb_isid"
            }
         }
      ],
      "type": 10
   }
},
{
   "OFPTableFeaturePropOxm": {
      "length": 152,
      "oxm_ids": [
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "in_port"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "metadata"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "eth_dst"
            }
```

```
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "eth_src"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "eth_type"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "vlan_vid"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "vlan_pcp"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "ip_dscp"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "ip_ecn"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "ip_proto"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "ipv4_src"
            }
        },
        {
```

```
                                    "OFPOxmId": {
                                        "hasmask": 0,
                                        "length": 0,
                                        "type": "ipv4_dst"
                                    }
                                },
                                {
                                    "OFPOxmId": {
                                        "hasmask": 0,
                                        "length": 0,
                                        "type": "tcp_src"
                                    }
                                },
                                {
                                    "OFPOxmId": {
                                        "hasmask": 0,
                                        "length": 0,
                                        "type": "tcp_dst"
                                    }
                                },
                                {
                                    "OFPOxmId": {
                                        "hasmask": 0,
                                        "length": 0,
                                        "type": "udp_src"
                                    }
                                },
                                {
                                    "OFPOxmId": {
                                        "hasmask": 0,
                                        "length": 0,
                                        "type": "udp_dst"
                                    }
                                },
                                {
                                    "OFPOxmId": {
                                        "hasmask": 0,
                                        "length": 0,
                                        "type": "sctp_src"
                                    }
                                },
                                {
                                    "OFPOxmId": {
                                        "hasmask": 0,
                                        "length": 0,
                                        "type": "sctp_dst"
                                    }
                                },
                                {
                                    "OFPOxmId": {
                                        "hasmask": 0,
                                        "length": 0,
                                        "type": "icmpv4_type"
                                    }
                                },
                                {
                                    "OFPOxmId": {
                                        "hasmask": 0,
                    "length": 0,
                    "type": "icmpv4_code"
                }
            },
            {
                "OFPOxmId": {
                    "hasmask": 0,
                    "length": 0,
                    "type": "arp_op"
                }
            },
            {
                "OFPOxmId": {
                    "hasmask": 0,
                    "length": 0,
                    "type": "arp_spa"
                }
            },
            {
                "OFPOxmId": {
                    "hasmask": 0,
                    "length": 0,
                    "type": "arp_tpa"
                }
            },
            {
                "OFPOxmId": {
                    "hasmask": 0,
                    "length": 0,
                    "type": "arp_sha"
                }
            },
            {
                "OFPOxmId": {
                    "hasmask": 0,
                    "length": 0,
                    "type": "arp_tha"
                }
            },
            {
                "OFPOxmId": {
                    "hasmask": 0,
                    "length": 0,
                    "type": "ipv6_src"
                }
            },
            {
                "OFPOxmId": {
                    "hasmask": 0,
                    "length": 0,
                    "type": "ipv6_dst"
                }
            },
            {
                "OFPOxmId": {
                    "hasmask": 0,
                    "length": 0,
                    "type": "ipv6_flabel"
```

```
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "icmpv6_type"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "icmpv6_code"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "ipv6_nd_target"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "ipv6_nd_sll"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "ipv6_nd_tll"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "mpls_label"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "mpls_tc"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "mpls_bos"
        }
    },
                    {
                        "OFPOxmId": {
                            "hasmask": 0,
                            "length": 0,
                            "type": "pbb_isid"
                        }
                    }
                ],
                "type": 12
            }
        },
        {
            "OFPTableFeaturePropOxm": {
                "length": 152,
                "oxm_ids": [
                    {
                        "OFPOxmId": {
                            "hasmask": 0,
                            "length": 0,
                            "type": "in_port"
                        }
                    },
                    {
                        "OFPOxmId": {
                            "hasmask": 0,
                            "length": 0,
                            "type": "metadata"
                        }
                    },
                    {
                        "OFPOxmId": {
                            "hasmask": 0,
                            "length": 0,
                            "type": "eth_dst"
                        }
                    },
                    {
                        "OFPOxmId": {
                            "hasmask": 0,
                            "length": 0,
                            "type": "eth_src"
                        }
                    },
                    {
                        "OFPOxmId": {
                            "hasmask": 0,
                            "length": 0,
                            "type": "eth_type"
                        }
                    },
                    {
                        "OFPOxmId": {
                            "hasmask": 0,
                            "length": 0,
                            "type": "vlan_vid"
                        }
                    },
                    {
```

```
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "vlan_pcp"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "ip_dscp"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "ip_ecn"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "ip_proto"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "ipv4_src"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "ipv4_dst"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "tcp_src"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "tcp_dst"
            }
         },
         {
            "OFPOxmId": {
               "hasmask": 0,
                "length": 0,
                "type": "udp_src"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "udp_dst"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "sctp_src"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "sctp_dst"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "icmpv4_type"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "icmpv4_code"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "arp_op"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "arp_spa"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "arp_tpa"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "arp_sha"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "arp_tha"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "ipv6_src"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "ipv6_dst"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "ipv6_flabel"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "icmpv6_type"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "icmpv6_code"
        }
    },
    {
        "OFPOxmId": {
            "hasmask": 0,
            "length": 0,
            "type": "ipv6_nd_target"
        }
    },
                  {
                     "OFPOxmId": {
                        "hasmask": 0,
                        "length": 0,
                        "type": "ipv6_nd_sll"
                     }
                  },
                  {
                     "OFPOxmId": {
                        "hasmask": 0,
                        "length": 0,
                        "type": "ipv6_nd_tll"
                     }
                  },
                  {
                     "OFPOxmId": {
                        "hasmask": 0,
                        "length": 0,
                        "type": "mpls_label"
                     }
                  },
                  {
                     "OFPOxmId": {
                        "hasmask": 0,
                        "length": 0,
                        "type": "mpls_tc"
                     }
                  },
                  {
                     "OFPOxmId": {
                        "hasmask": 0,
                        "length": 0,
                        "type": "mpls_bos"
                     }
                  },
                  {
                     "OFPOxmId": {
                        "hasmask": 0,
                        "length": 0,
                        "type": "pbb_isid"
                     }
                  }
               ],
               "type": 14
            }
         }
      ],
      "table_id": 1
   }
},
{
   "OFPTableFeaturesStats": {
      "config": 0,
      "length": 1104,
      "max_entries": 16777216,
      "metadata_match": 18446744073709551615,
      "metadata_write": 18446744073709551615,
      "name": "Flow Table 0x02",
                    "properties": [
                       {
                          "OFPTableFeaturePropInstructions": {
                             "instruction_ids": [
                                {
                                   "OFPInstructionId": {
                                      "len": 4,
                                      "type": 1
                                   }
                                },
                                {
                                   "OFPInstructionId": {
                                      "len": 4,
                                      "type": 2
                                   }
                                },
                                {
                                   "OFPInstructionId": {
                                      "len": 4,
                                      "type": 3
                                   }
                                },
                                {
                                   "OFPInstructionId": {
                                      "len": 4,
                                      "type": 4
                                   }
                                },
                                {
                                   "OFPInstructionId": {
                                      "len": 4,
                                      "type": 5
                                   }
                                },
                                {
                                   "OFPInstructionId": {
                                      "len": 4,
                                      "type": 6
                                   }
                                }
                             ],
                             "length": 28,
                             "type": 0
                          }
                       },
                       {
                          "OFPTableFeaturePropNextTables": {
                             "length": 256,
                             "table_ids": [
                                3,
                                4,
                                5,
                                6,
                                7,
                                8,
                                9,
                                10,
                                11,
                                12,
                                13,
                                14,
                                15,
                                16,
                                17,
                                18,
                                19,
                                20,
                                21,
                                22,
                                23,
                                24,
                                25,
                                26,
                                27,
                                28,
                                29,
                                30,
                                31,
                                32,
                                33,
                                34,
                                35,
                                36,
                                37,
                                38,
                                39,
                                40,
                                41,
                                42,
                                43,
                                44,
                                45,
                                46,
                                47,
                                48,
                                49,
                                50,
                                51,
                                52,
                                53,
                                54,
                                55,
                                56,
                                57,
                                58,
                                59,
                                60,
                                61,
                                62,
                                63,
                                64,
                                65,
                                66,
                                67,
                                68,
                                69,
                                  70,
                                  71,
                                  72,
                                  73,
                                  74,
                                  75,
                                  76,
                                  77,
                                  78,
                                  79,
                                  80,
                                  81,
                                  82,
                                  83,
                                  84,
                                  85,
                                  86,
                                  87,
                                  88,
                                  89,
                                  90,
                                  91,
                                  92,
                                  93,
                                  94,
                                  95,
                                  96,
                                  97,
                                  98,
                                  99,
                                  100,
                                  101,
                                  102,
                                  103,
                                  104,
                                  105,
                                  106,
                                  107,
                                  108,
                                  109,
                                  110,
                                  111,
                                  112,
                                  113,
                                  114,
                                  115,
                                  116,
                                  117,
                                  118,
                                  119,
                                  120,
                                  121,
                                  122,
                                  123,
                                  124,
                                  125,
                                  126,
                                  127,
                                            128,
                                            129,
                                            130,
                                            131,
                                            132,
                                            133,
                                            134,
                                            135,
                                            136,
                                            137,
                                            138,
                                            139,
                                            140,
                                            141,
                                            142,
                                            143,
                                            144,
                                            145,
                                            146,
                                            147,
                                            148,
                                            149,
                                            150,
                                            151,
                                            152,
                                            153,
                                            154,
                                            155,
                                            156,
                                            157,
                                            158,
                                            159,
                                            160,
                                            161,
                                            162,
                                            163,
                                            164,
                                            165,
                                            166,
                                            167,
                                            168,
                                            169,
                                            170,
                                            171,
                                            172,
                                            173,
                                            174,
                                            175,
                                            176,
                                            177,
                                            178,
                                            179,
                                            180,
                                            181,
                                            182,
                                            183,
                                            184,
                                            185,
                                186,
                                187,
                                188,
                                189,
                                190,
                                191,
                                192,
                                193,
                                194,
                                195,
                                196,
                                197,
                                198,
                                199,
                                200,
                                201,
                                202,
                                203,
                                204,
                                205,
                                206,
                                207,
                                208,
                                209,
                                210,
                                211,
                                212,
                                213,
                                214,
                                215,
                                216,
                                217,
                                218,
                                219,
                                220,
                                221,
                                222,
                                223,
                                224,
                                225,
                                226,
                                227,
                                228,
                                229,
                                230,
                                231,
                                232,
                                233,
                                234,
                                235,
                                236,
                                237,
                                238,
                                239,
                                240,
                                241,
                                242,
                                243,
                        244,
                        245,
                        246,
                        247,
                        248,
                        249,
                        250,
                        251,
                        252,
                        253,
                        254
                    ],
                    "type": 2
                }
            },
            {
                "OFPTableFeaturePropActions": {
                    "action_ids": [
                        {
                            "OFPActionId": {
                                "len": 4,
                                "type": 0
                            }
                        },
                        {
                            "OFPActionId": {
                                "len": 4,
                                "type": 22
                            }
                        },
                        {
                            "OFPActionId": {
                                "len": 4,
                                "type": 21
                            }
                        },
                        {
                            "OFPActionId": {
                                "len": 4,
                                "type": 15
                            }
                        },
                        {
                            "OFPActionId": {
                                "len": 4,
                                "type": 16
                            }
                        },
                        {
                            "OFPActionId": {
                                "len": 4,
                                "type": 23
                            }
                        },
                        {
                            "OFPActionId": {
                                "len": 4,
                                "type": 24
            }
        },
        {
            "OFPActionId": {
                "len": 4,
                "type": 11
            }
        },
        {
            "OFPActionId": {
                "len": 4,
                "type": 12
            }
        },
        {
            "OFPActionId": {
                "len": 4,
                "type": 17
            }
        },
        {
            "OFPActionId": {
                "len": 4,
                "type": 18
            }
        },
        {
            "OFPActionId": {
                "len": 4,
                "type": 19
            }
        },
        {
            "OFPActionId": {
                "len": 4,
                "type": 20
            }
        },
        {
            "OFPActionId": {
                "len": 4,
                "type": 26
            }
        },
        {
            "OFPActionId": {
                "len": 4,
                "type": 27
            }
        },
        {
            "OFPActionId": {
                "len": 4,
                "type": 25
            }
        }
    ],
    "length": 68,
                "type": 4
            }
        },
        {
            "OFPTableFeaturePropActions": {
                "action_ids": [
                    {
                        "OFPActionId": {
                            "len": 4,
                            "type": 0
                        }
                    },
                    {
                        "OFPActionId": {
                            "len": 4,
                            "type": 22
                        }
                    },
                    {
                        "OFPActionId": {
                            "len": 4,
                            "type": 21
                        }
                    },
                    {
                        "OFPActionId": {
                            "len": 4,
                            "type": 15
                        }
                    },
                    {
                        "OFPActionId": {
                            "len": 4,
                            "type": 16
                        }
                    },
                    {
                        "OFPActionId": {
                            "len": 4,
                            "type": 23
                        }
                    },
                    {
                        "OFPActionId": {
                            "len": 4,
                            "type": 24
                        }
                    },
                    {
                        "OFPActionId": {
                            "len": 4,
                            "type": 11
                        }
                    },
                    {
                        "OFPActionId": {
                            "len": 4,
                            "type": 12
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 17
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 18
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 19
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 20
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 26
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 27
            }
         },
         {
            "OFPActionId": {
               "len": 4,
               "type": 25
            }
         }
      ],
      "length": 68,
      "type": 6
   }
},
{
   "OFPTableFeaturePropOxm": {
      "length": 152,
      "oxm_ids": [
         {
            "OFPOxmId": {
               "hasmask": 0,
               "length": 0,
               "type": "in_port"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "metadata"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "eth_dst"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "eth_src"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "eth_type"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "vlan_vid"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "vlan_pcp"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "ip_dscp"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "ip_ecn"
            }
        },
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "ip_proto"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "ipv4_src"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "ipv4_dst"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "tcp_src"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "tcp_dst"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "udp_src"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "udp_dst"
   }
},
{
   "OFPOxmId": {
      "hasmask": 0,
      "length": 0,
      "type": "sctp_src"
   }
},
{
   "OFPOxmId": {
```

```
                "hasmask": 0,
                "length": 0,
                "type": "sctp_dst"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "icmpv4_type"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "icmpv4_code"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "arp_op"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "arp_spa"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "arp_tpa"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "arp_sha"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "arp_tha"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "ipv6_src"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "ipv6_dst"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "ipv6_flabel"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "icmpv6_type"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "icmpv6_code"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "ipv6_nd_target"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "ipv6_nd_sll"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "ipv6_nd_tll"
            }
        },
        {
            "OFPOxmId": {
                "hasmask": 0,
                "length": 0,
                "type": "mpls_label"
            }
            },
            {
                "OFPOxmId": {
                    "hasmask": 0,
                    "length": 0,
                    "type": "mpls_tc"
                }
            },
            {
                "OFPOxmId": {
                    "hasmask": 0,
                    "length": 0,
                    "type": "mpls_bos"
                }
            },
            {
                "OFPOxmId": {
                    "hasmask": 0,
                    "length": 0,
                    "type": "pbb_isid"
                }
            }
        ],
        "type": 8
    }
},
{
    "OFPTableFeaturePropOxm": {
        "length": 152,
        "oxm_ids": [
            {
                "OFPOxmId": {
                    "hasmask": 0,
                    "length": 0,
                    "type": "in_port"
                }
            },
            {
                "OFPOxmId": {
                    "hasmask": 0,
                    "length": 0,
                    "type": "metadata"
                }
            },
            {
                "OFPOxmId": {
                    "hasmask": 0,
                    "type": "eth_dst"
                }
            },
            {
                "OFPOxmId": {
                    "hasmask": 0,

            {
                "OFPOxmId": {
                    "hasmask": 0,
                    "length": 0,
                    "type": "eth_type"
                }
            },
```

Queue Configuration Messages

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPQueueGetConfigRequest`**(*datapath*, *port*)

Queue configuration request message

Attribute	Description
port	Port to be queried (OFPP_ANY to all configured queues)

Example:

```
def send_queue_get_config_request(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser
    req = ofp_parser.OFPQueueGetConfigRequest(datapath, ofp.OFPP_ANY)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPQueueGetConfigRequest": {
      "port": 4294967295
   }
}
```

class ryu.ofproto.ofproto_v1_3_parser.**OFPQueueGetConfigReply**(*datapath*, *queues=None*, *port=None*)

Queue configuration reply message

The switch responds with this message to a queue configuration request.

Attribute	Description
queues	list of `OFPPacketQueue` instance
port	Port which was queried

Example:

```
@set_ev_cls(ofp_event.EventOFPQueueGetConfigReply, MAIN_DISPATCHER)
def queue_get_config_reply_handler(self, ev):
    msg = ev.msg

    self.logger.debug('OFPQueueGetConfigReply received: '
                      'port=%s queues=%s',
                      msg.port, msg.queues)
```

JSON Example:

```
{
   "OFPQueueGetConfigReply": {
      "port": 4294967295,
      "queues": [
         {
            "OFPPacketQueue": {
               "len": 48,
               "port": 77,
               "properties": [
                  {
                     "OFPQueuePropMinRate": {
                        "len": 16,
                        "property": 1,
                        "rate": 10
                     }
                  },
                  {
                     "OFPQueuePropMaxRate": {
                        "len": 16,
                        "property": 2,
                        "rate": 900
                     }
                  }
               ],
               "queue_id": 99
            }
         },
         {
            "OFPPacketQueue": {
               "len": 48,
               "port": 77,
               "properties": [
                  {
                     "OFPQueuePropMinRate": {
                        "len": 16,
                        "property": 1,
                        "rate": 100
                     }
                  },
                  {
                     "OFPQueuePropMaxRate": {
                        "len": 16,
                        "property": 2,
                        "rate": 200
                     }
                  }
               ],
               "queue_id": 88
            }
         }
      ]
   }
}
```

Packet-Out Message

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

Packet-Out message

The controller uses this message to send a packet out throught the switch.

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

Example:

```
def send_packet_out(self, datapath, buffer_id, in_port):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    actions = [ofp_parser.OFPActionOutput(ofp.OFPP_FLOOD, 0)]
    req = ofp_parser.OFPPacketOut(datapath, buffer_id,
                                  in_port, actions)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPPacketOut": {
      "actions": [
         {
            "OFPActionOutput": {
               "len": 16,
               "max_len": 65535,
               "port": 4294967292,
               "type": 0
            }
         }
      ],
      "actions_len": 16,
      "buffer_id": 4294967295,
      "data": "8guk0D9w8gukffjqCABFAABU+BoAAP8Br4sKAAABCgAAAggAAgj3YAAAMdYCAAAAAACrjS0xAAAAABARE
      "in_port": 4294967293
   }
}
```

Barrier Message

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPBarrierRequest`**(*datapath*)

Barrier request message

The controller sends this message to ensure message dependencies have been met or receive notifications for completed operations.

Example:

```
def send_barrier_request(self, datapath):
    ofp_parser = datapath.ofproto_parser

    req = ofp_parser.OFPBarrierRequest(datapath)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPBarrierRequest": {}
}
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPBarrierReply`**(*datapath*)

Barrier reply message

The switch responds with this message to a barrier request.

Example:

```
@set_ev_cls(ofp_event.EventOFPBarrierReply, MAIN_DISPATCHER)
def barrier_reply_handler(self, ev):
    self.logger.debug('OFPBarrierReply received')
```

JSON Example:

```
{
   "OFPBarrierReply": {}
}
```

Role Request Message

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPRoleRequest`**(*datapath, role=None, generation_id=None*)

Role request message

The controller uses this message to change its role.

Attribute	Description
role	One of the following values. OFPCR_ROLE_NOCHANGE OFPCR_ROLE_EQUAL OFPCR_ROLE_MASTER OFPCR_ROLE_SLAVE
generation_id	Master Election Generation ID

Example:

```
def send_role_request(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    req = ofp_parser.OFPRoleRequest(datapath, ofp.OFPCR_ROLE_EQUAL, 0)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPRoleRequest": {
      "generation_id": 17294086455919964160,
      "role": 2
   }
}
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPRoleReply`**(*datapath*, *role=None*, *generation_id=None*)

Role reply message

The switch responds with this message to a role request.

Attribute	Description
role	One of the following values. OFPCR_ROLE_NOCHANGE OFPCR_ROLE_EQUAL OFPCR_ROLE_MASTER OFPCR_ROLE_SLAVE
generation_id	Master Election Generation ID

Example:

```
@set_ev_cls(ofp_event.EventOFPRoleReply, MAIN_DISPATCHER)
def role_reply_handler(self, ev):
    msg = ev.msg
    ofp = dp.ofproto

    if msg.role == ofp.OFPCR_ROLE_NOCHANGE:
        role = 'NOCHANGE'
    elif msg.role == ofp.OFPCR_ROLE_EQUAL:
        role = 'EQUAL'
    elif msg.role == ofp.OFPCR_ROLE_MASTER:
        role = 'MASTER'
    elif msg.role == ofp.OFPCR_ROLE_SLAVE:
        role = 'SLAVE'
    else:
        role = 'unknown'

    self.logger.debug('OFPRoleReply received: '
                      'role=%s generation_id=%d',
                      role, msg.generation_id)
```

JSON Example:

```
{
   "OFPRoleReply": {
      "generation_id": 17294086455919964160,
      "role": 3
   }
}
```

Set Asynchronous Configuration Message

class ryu.ofproto.ofproto_v1_3_parser.**OFPSetAsync**(*datapath*, *packet_in_mask*, *port_status_mask*, *flow_removed_mask*)

Set asynchronous configuration message

The controller sends this message to set the asynchronous messages that it wants to receive on a given OpneFlow channel.

Attribute	Description
packet_in_mask	2-element array: element 0, when the controller has a OFPCR_ROLE_EQUAL or OFPCR_ROLE_MASTER role. element 1, OFPCR_ROLE_SLAVE role controller. Bitmasks of following values. OFPR_NO_MATCH OFPR_ACTION OFPR_INVALID_TTL
port_status_mask	2-element array. Bitmasks of following values. OFPPR_ADD OFPPR_DELETE OFPPR_MODIFY
flow_removed_mask	2-element array. Bitmasks of following values. OFPRR_IDLE_TIMEOUT OFPRR_HARD_TIMEOUT OFPRR_DELETE OFPRR_GROUP_DELETE

Example:

```
def send_set_async(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    packet_in_mask = ofp.OFPR_ACTION | ofp.OFPR_INVALID_TTL
    port_status_mask = (ofp.OFPPR_ADD | ofp.OFPPR_DELETE |
                        ofp.OFPPR_MODIFY)
    flow_removed_mask = (ofp.OFPRR_IDLE_TIMEOUT |
                         ofp.OFPRR_HARD_TIMEOUT |
                         ofp.OFPRR_DELETE)
    req = ofp_parser.OFPSetAsync(datapath,
                                 [packet_in_mask, 0],
                                 [port_status_mask, 0],
                                 [flow_removed_mask, 0])
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPSetAsync": {
      "flow_removed_mask": [
         15,
         3
      ],
      "packet_in_mask": [
         5,
            1
        ],
        "port_status_mask": [
            7,
            3
        ]
    }
}
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPGetAsyncRequest`**`(`*datapath*`)`

Get asynchronous configuration request message

The controller uses this message to query the asynchronous message.

Example:

```
def send_get_async_request(self, datapath):
    ofp_parser = datapath.ofproto_parser

    req = ofp_parser.OFPGetAsyncRequest(datapath)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPGetAsyncRequest": {}
}
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPGetAsyncReply`**`(`*datapath, packet_in_mask=None, port_status_mask=None, flow_removed_mask=None*`)`

Get asynchronous configuration reply message

The switch responds with this message to a get asynchronous configuration request.

Attribute	Description
packet_in_mask	2-element array: element 0, when the controller has a OFPCR_ROLE_EQUAL or OFPCR_ROLE_MASTER role. element 1, OFPCR_ROLE_SLAVE role controller. Bitmasks of following values. OFPR_NO_MATCH OFPR_ACTION OFPR_INVALID_TTL
port_status_mask	2-element array. Bitmasks of following values. OFPPR_ADD OFPPR_DELETE OFPPR_MODIFY
flow_removed_mask	2-element array. Bitmasks of following values. OFPRR_IDLE_TIMEOUT OFPRR_HARD_TIMEOUT OFPRR_DELETE OFPRR_GROUP_DELETE

Example:

```
@set_ev_cls(ofp_event.EventOFPGetAsyncReply, MAIN_DISPATCHER)
def get_async_reply_handler(self, ev):
    msg = ev.msg

    self.logger.debug('OFPGetAsyncReply received: '
                      'packet_in_mask=0x%08x:0x%08x '
                      'port_status_mask=0x%08x:0x%08x '
                      'flow_removed_mask=0x%08x:0x%08x',
                      msg.packet_in_mask[0],
                      msg.packet_in_mask[1],
                      msg.port_status_mask[0],
                      msg.port_status_mask[1],
                      msg.flow_removed_mask[0],
                      msg.flow_removed_mask[1])
```

JSON Example:

```
{
   "OFPGetAsyncReply": {
      "flow_removed_mask": [
         15,
         3
      ],
      "packet_in_mask": [
         5,
         1
      ],
      "port_status_mask": [
         7,
         3
      ]
   }
}
```

Asynchronous Messages

Packet-In Message

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPPacketIn`**(*datapath*, *buffer_id=None*, *total_len=None*, *reason=None*, *table_id=None*, *cookie=None*, *match=None*, *data=None*)

Packet-In message

The switch sends the packet that received to the controller by this message.

Attribute	Description
buffer_id	ID assigned by datapath
total_len	Full length of frame
reason	Reason packet is being sent. OFPR_NO_MATCH OFPR_ACTION OFPR_INVALID_TTL
table_id	ID of the table that was looked up
cookie	Cookie of the flow entry that was looked up
match	Instance of `OFPMatch`
data	Ethernet frame

Example:

```
@set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
def packet_in_handler(self, ev):
    msg = ev.msg
    ofp = dp.ofproto

    if msg.reason == ofp.OFPR_NO_MATCH:
        reason = 'NO MATCH'
    elif msg.reason == ofp.OFPR_ACTION:
        reason = 'ACTION'
    elif msg.reason == ofp.OFPR_INVALID_TTL:
        reason = 'INVALID TTL'
    else:
        reason = 'unknown'
    self.logger.debug('OFPPacketIn received: '
                      'buffer_id=%x total_len=%d reason=%s '
                      'table_id=%d cookie=%d match=%s data=%s',
                      msg.buffer_id, msg.total_len, reason,
                      msg.table_id, msg.cookie, msg.match,
                      utils.hex_array(msg.data))
```

JSON Example:

```
{
   "OFPPacketIn": {
      "buffer_id": 2,
      "cookie": 283686884868096,
      "data": "////////8gukffjqCAYAAQgABgQAAfILpH346goAAAEAAAAAAAAKAAAD",
      "match": {
         "OFPMatch": {
            "length": 80,
            "oxm_fields": [
               {
                  "OXMTlv": {
                     "field": "in_port",
                     "mask": null,
                     "value": 6
                  }
               },
               {
                  "OXMTlv": {
                     "field": "eth_type",
                     "mask": null,
                     "value": 2054
                  }
               },
               {
                  "OXMTlv": {
                     "field": "eth_dst",
                     "mask": null,
                     "value": "ff:ff:ff:ff:ff:ff"
                  }
               },
               {
                  "OXMTlv": {
                     "field": "eth_src",
                     "mask": null,
                     "value": "f2:0b:a4:7d:f8:ea"
                  }
               },
               {
                  "OXMTlv": {
                     "field": "arp_op",
                     "mask": null,
                     "value": 1
                  }
               },
               {
                  "OXMTlv": {
                     "field": "arp_spa",
                     "mask": null,
                     "value": "10.0.0.1"
```

```
                        }
                    },
                    {
                        "OXMTlv": {
                            "field": "arp_tpa",
                            "mask": null,
                            "value": "10.0.0.3"
                        }
                    },
                    {
                        "OXMTlv": {
                            "field": "arp_sha",
                            "mask": null,
                            "value": "f2:0b:a4:7d:f8:ea"
                        }
                    },
                    {
                        "OXMTlv": {
                            "field": "arp_tha",
                            "mask": null,
                            "value": "00:00:00:00:00:00"
                        }
                    }
                ],
                "type": 1
            }
        },
        "reason": 1,
        "table_id": 1,
        "total_len": 42
    }
}
```

Flow Removed Message

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPFlowRemoved`**(*datapath, cookie=None, priority=None, reason=None, table_id=None, duration_sec=None, duration_nsec=None, idle_timeout=None, hard_timeout=None, packet_count=None, byte_count=None, match=None*)

Flow removed message

When flow entries time out or are deleted, the switch notifies controller with this message.

Attribute	Description
cookie	Opaque controller-issued identifier
priority	Priority level of flow entry
reason	One of the following values. OFPRR_IDLE_TIMEOUT OFPRR_HARD_TIMEOUT OFPRR_DELETE OFPRR_GROUP_DELETE
table_id	ID of the table
duration_sec	Time flow was alive in seconds
duration_nsec	Time flow was alive in nanoseconds beyond duration_sec
idle_timeout	Idle timeout from original flow mod
hard_timeout	Hard timeout from original flow mod
packet_count	Number of packets that was associated with the flow
byte_count	Number of bytes that was associated with the flow
match	Instance of `OFPMatch`

Example:

```
@set_ev_cls(ofp_event.EventOFPFlowRemoved, MAIN_DISPATCHER)
def flow_removed_handler(self, ev):
    msg = ev.msg
    dp = msg.datapath
    ofp = dp.ofproto

    if msg.reason == ofp.OFPRR_IDLE_TIMEOUT:
        reason = 'IDLE TIMEOUT'
    elif msg.reason == ofp.OFPRR_HARD_TIMEOUT:
        reason = 'HARD TIMEOUT'
    elif msg.reason == ofp.OFPRR_DELETE:
        reason = 'DELETE'
    elif msg.reason == ofp.OFPRR_GROUP_DELETE:
        reason = 'GROUP DELETE'
    else:
        reason = 'unknown'

    self.logger.debug('OFPFlowRemoved received: '
                      'cookie=%d priority=%d reason=%s table_id=%d '
                      'duration_sec=%d duration_nsec=%d '
                      'idle_timeout=%d hard_timeout=%d '
                      'packet_count=%d byte_count=%d match.fields=%s',
                      msg.cookie, msg.priority, reason, msg.table_id,
                      msg.duration_sec, msg.duration_nsec,
                      msg.idle_timeout, msg.hard_timeout,
                      msg.packet_count, msg.byte_count, msg.match)
```

JSON Example:

```
{
   "OFPFlowRemoved": {
      "byte_count": 86,
      "cookie": 0,
      "duration_nsec": 48825000,
      "duration_sec": 3,
      "hard_timeout": 0,
      "idle_timeout": 3,
      "match": {
         "OFPMatch": {
            "length": 14,
            "oxm_fields": [
               {
                  "OXMTlv": {
                     "field": "eth_dst",
                     "mask": null,
                     "value": "f2:0b:a4:7d:f8:ea"
                  }
               }
            ],
            "type": 1
         }
      },
      "packet_count": 1,
      "priority": 65535,
      "reason": 0,
      "table_id": 0
   }
}
```

Port Status Message

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPPortStatus`**(*datapath*, *reason=None*, *desc=None*)

Port status message

The switch notifies controller of change of ports.

Attribute	Description
reason	One of the following values. OFPPR_ADD OFPPR_DELETE OFPPR_MODIFY
desc	instance of `OFPPort`

Example:

```
@set_ev_cls(ofp_event.EventOFPPortStatus, MAIN_DISPATCHER)
def port_status_handler(self, ev):
    msg = ev.msg
    dp = msg.datapath
    ofp = dp.ofproto

    if msg.reason == ofp.OFPPR_ADD:
        reason = 'ADD'
    elif msg.reason == ofp.OFPPR_DELETE:
        reason = 'DELETE'
    elif msg.reason == ofp.OFPPR_MODIFY:
        reason = 'MODIFY'
    else:
        reason = 'unknown'

    self.logger.debug('OFPPortStatus received: reason=%s desc=%s',
                      reason, msg.desc)
```

JSON Example:

```
{
   "OFPPortStatus": {
      "desc": {
         "OFPPort": {
            "advertised": 10240,
            "config": 0,
            "curr": 10248,
            "curr_speed": 5000,
            "hw_addr": "f2:0b:a4:d0:3f:70",
            "max_speed": 5000,
            "name": "\u79c1\u306e\u30dd\u30fc\u30c8",
            "peer": 10248,
            "port_no": 7,
            "state": 4,
            "supported": 10248
         }
      },
      "reason": 0
   }
}
```

Error Message

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPErrorMsg`**(*datapath*, *type_=None*, *code=None*, *data=None*)

Error message

The switch notifies controller of problems by this message.

Attribute	Description
type	High level type of error
code	Details depending on the type
data	Variable length data depending on the type and code

`type` attribute corresponds to `type_` parameter of __init__.

Types and codes are defined in `ryu.ofproto.ofproto`.

Type	Code
OFPET_HELLO_FAILED	OFPHFC_*
OFPET_BAD_REQUEST	OFPBRC_*
OFPET_BAD_ACTION	OFPBAC_*
OFPET_BAD_INSTRUCTION	OFPBIC_*
OFPET_BAD_MATCH	OFPBMC_*
OFPET_FLOW_MOD_FAILED	OFPFMFC_*
OFPET_GROUP_MOD_FAILED	OFPGMFC_*
OFPET_PORT_MOD_FAILED	OFPPMFC_*
OFPET_TABLE_MOD_FAILED	OFPTMFC_*
OFPET_QUEUE_OP_FAILED	OFPQOFC_*
OFPET_SWITCH_CONFIG_FAILED	OFPSCFC_*
OFPET_ROLE_REQUEST_FAILED	OFPRRFC_*
OFPET_METER_MOD_FAILED	OFPMMFC_*
OFPET_TABLE_FEATURES_FAILED	OFPTFFC_*
OFPET_EXPERIMENTER	N/A

Example:

```
@set_ev_cls(ofp_event.EventOFPErrorMsg,
            [HANDSHAKE_DISPATCHER, CONFIG_DISPATCHER, MAIN_DISPATCHER])
def error_msg_handler(self, ev):
    msg = ev.msg

    self.logger.debug('OFPErrorMsg received: type=0x%02x code=0x%02x '
                      'message=%s',
                      msg.type, msg.code, utils.hex_array(msg.data))
```

JSON Example:

```
{
   "OFPErrorMsg": {
      "code": 11,
      "data": "ZnVnYWZ1Z2E=",
      "type": 2
   }
}
```

Symmetric Messages

Hello

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPHello`**(*datapath*, *elements=[]*)

Hello message

When connection is started, the hello message is exchanged between a switch and a controller.

This message is handled by the Ryu framework, so the Ryu application do not need to process this typically.

Attribute	Description
elements	list of `OFPHelloElemVersionBitmap` instance

JSON Example:

```
{
   "OFPHello": {
      "elements": [
         {
            "OFPHelloElemVersionBitmap": {
               "length": 8,
               "type": 1,
               "versions": [
                  1,
                  2,
                  3,
                  9,
                  10,
                  30
               ]
            }
         }
      ]
   }
}
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPHelloElemVersionBitmap`**(*versions*, *type_=None*, *length=None*)

Version bitmap Hello Element

Attribute	Description
versions	list of versions of OpenFlow protocol a device supports

Echo Request

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPEchoRequest`**(*datapath*, *data=None*)

Echo request message

This message is handled by the Ryu framework, so the Ryu application do not need to process this typically.

Attribute	Description
data	An arbitrary length data

Example:

```
def send_echo_request(self, datapath, data):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    req = ofp_parser.OFPEchoRequest(datapath, data)
    datapath.send_msg(req)

@set_ev_cls(ofp_event.EventOFPEchoRequest,
            [HANDSHAKE_DISPATCHER, CONFIG_DISPATCHER, MAIN_DISPATCHER])
def echo_request_handler(self, ev):
    self.logger.debug('OFPEchoRequest received: data=%s',
                      utils.hex_array(ev.msg.data))
```

JSON Example:

```
{
   "OFPEchoRequest": {
      "data": "aG9nZQ=="
   }
}
```

Echo Reply

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPEchoReply`**(*datapath*, *data=None*)

Echo reply message

This message is handled by the Ryu framework, so the Ryu application do not need to process this typically.

Attribute	Description
data	An arbitrary length data

Example:

```
def send_echo_reply(self, datapath, data):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    reply = ofp_parser.OFPEchoReply(datapath, data)
    datapath.send_msg(reply)

@set_ev_cls(ofp_event.EventOFPEchoReply,
            [HANDSHAKE_DISPATCHER, CONFIG_DISPATCHER, MAIN_DISPATCHER])
def echo_reply_handler(self, ev):
    self.logger.debug('OFPEchoReply received: data=%s',
                      utils.hex_array(ev.msg.data))
```

JSON Example:

```
{
   "OFPEchoReply": {
      "data": "aG9nZQ=="
   }
}
```

Experimenter

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPExperimenter`**(*datapath*, *experimenter=None*, *exp_type=None*, *data=None*)

Experimenter extension message

Attribute	Description
experimenter	Experimenter ID
exp_type	Experimenter defined
data	Experimenter defined arbitrary additional data

JSON Example:

```
{
   "OFPExperimenter": {
      "data": "bmF6bw==",
      "exp_type": 123456789,
      "experimenter": 98765432
   }
}
```

Flow Match Structure

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPMatch`**(*type_=None*, *length=None*, *_ordered_fields=None*, ***kwargs*)

Flow Match Structure

This class is implementation of the flow match structure having compose/query API. There are new API and old API for compatibility. the old API is supposed to be removed later.

You can define the flow match by the keyword arguments. The following arguments are available.

Argument	Value	Description
in_port	Integer 32bit	Switch input port
in_phy_port	Integer 32bit	Switch physical input port
metadata	Integer 64bit	Metadata passed between tables
eth_dst	MAC address	Ethernet destination address
eth_src	MAC address	Ethernet source address
eth_type	Integer 16bit	Ethernet frame type
vlan_vid	Integer 16bit	VLAN id
vlan_pcp	Integer 8bit	VLAN priority
ip_dscp	Integer 8bit	IP DSCP (6 bits in ToS field)
ip_ecn	Integer 8bit	IP ECN (2 bits in ToS field)
ip_proto	Integer 8bit	IP protocol
ipv4_src	IPv4 address	IPv4 source address
ipv4_dst	IPv4 address	IPv4 destination address
tcp_src	Integer 16bit	TCP source port
tcp_dst	Integer 16bit	TCP destination port
udp_src	Integer 16bit	UDP source port

Continued on next page

Table 2.2 – continued from previous page

Argument	Value	Description
udp_dst	Integer 16bit	UDP destination port
sctp_src	Integer 16bit	SCTP source port
sctp_dst	Integer 16bit	SCTP destination port
icmpv4_type	Integer 8bit	ICMP type
icmpv4_code	Integer 8bit	ICMP code
arp_op	Integer 16bit	ARP opcode
arp_spa	IPv4 address	ARP source IPv4 address
arp_tpa	IPv4 address	ARP target IPv4 address
arp_sha	MAC address	ARP source hardware address
arp_tha	MAC address	ARP target hardware address
ipv6_src	IPv6 address	IPv6 source address
ipv6_dst	IPv6 address	IPv6 destination address
ipv6_flabel	Integer 32bit	IPv6 Flow Label
icmpv6_type	Integer 8bit	ICMPv6 type
icmpv6_code	Integer 8bit	ICMPv6 code
ipv6_nd_target	IPv6 address	Target address for ND
ipv6_nd_sll	MAC address	Source link-layer for ND
ipv6_nd_tll	MAC address	Target link-layer for ND
mpls_label	Integer 32bit	MPLS label
mpls_tc	Integer 8bit	MPLS TC
mpls_bos	Integer 8bit	MPLS BoS bit
pbb_isid	Integer 24bit	PBB I-SID
tunnel_id	Integer 64bit	Logical Port Metadata
ipv6_exthdr	Integer 16bit	IPv6 Extension Header pseudo-field

Example:

```
>>> # compose
>>> match = parser.OFPMatch(
...     in_port=1,
...     eth_type=0x86dd,
...     ipv6_src=('2001:db8:bd05:1d2:288a:1fc0:1:10ee',
...               'ffff:ffff:ffff:ffff::'),
...     ipv6_dst='2001:db8:bd05:1d2:288a:1fc0:1:10ee')
>>> # query
>>> if 'ipv6_src' in match:
...     print match['ipv6_src']
...
('2001:db8:bd05:1d2:288a:1fc0:1:10ee', 'ffff:ffff:ffff:ffff::')
```

Flow Instruction Structures

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPInstructionGotoTable`**(*table_id*, *type_=None*, *len_=None*)

Goto table instruction

This instruction indicates the next table in the processing pipeline.

Attribute	Description
table_id	Next table

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPInstructionWriteMetadata`**(*metadata*, *metadata_mask*, *type_=None*, *len_=None*)

Write metadata instruction

This instruction writes the masked metadata value into the metadata field.

Attribute	Description
metadata	Metadata value to write
metadata_mask	Metadata write bitmask

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

Actions instruction

This instruction writes/applies/clears the actions.

Attribute	Description
type	One of following values. OFPIT_WRITE_ACTIONS OFPIT_APPLY_ACTIONS OFPIT_CLEAR_ACTIONS
actions	list of OpenFlow action class

`type` attribute corresponds to `type_` parameter of __init__.

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPInstructionMeter`**(*meter_id*, *type_=None*, *len_=None*)

Meter instruction

This instruction applies the meter.

Attribute	Description
meter_id	Meter instance

Action Structures

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

Output action

This action indicates output a packet to the switch port.

Attribute	Description
port	Output port
max_len	Max length to send to controller

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPActionGroup`**(*group_id*, *type_=None*, *len_=None*)

Group action

This action indicates the group used to process the packet.

Attribute	Description
group_id	Group identifier

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPActionSetQueue`**(*queue_id*, *type_=None*, *len_=None*)

Set queue action

This action sets the queue id that will be used to map a flow to an already-configured queue on a port.

Attribute	Description
queue_id	Queue ID for the packets

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPActionSetMplsTtl`**(*mpls_ttl*, *type_=None*, *len_=None*)

Set MPLS TTL action

This action sets the MPLS TTL.

Attribute	Description
mpls_ttl	MPLS TTL

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPActionDecMplsTtl`**(*type_=None*, *len_=None*)

Decrement MPLS TTL action

This action decrements the MPLS TTL.

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPActionSetNwTtl`**(*nw_ttl*, *type_=None*, *len_=None*)

Set IP TTL action

This action sets the IP TTL.

Attribute	Description
nw_ttl	IP TTL

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPActionDecNwTtl`**(*type_=None*, *len_=None*)

Decrement IP TTL action

This action decrements the IP TTL.

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPActionCopyTtlOut`**(*type_=None*, *len_=None*)

Copy TTL Out action

This action copies the TTL from the next-to-outermost header with TTL to the outermost header with TTL.

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPActionCopyTtlIn`**(*type_=None*, *len_=None*)

Copy TTL In action

This action copies the TTL from the outermost header with TTL to the next-to-outermost header with TTL.

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPActionPushVlan`**(*ethertype=33024*, *type_=None*, *len_=None*)

Push VLAN action

This action pushes a new VLAN tag to the packet.

Attribute	Description
ethertype	Ether type. The default is 802.1Q. (0x8100)

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPActionPushMpls`**(*ethertype=34887*, *type_=None*, *len_=None*)

Push MPLS action

This action pushes a new MPLS header to the packet.

Attribute	Description
ethertype	Ether type

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPActionPopVlan`**(*type_=None*, *len_=None*)

Pop VLAN action

This action pops the outermost VLAN tag from the packet.

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPActionPopMpls`**(*ethertype=2048*, *type_=None*, *len_=None*)

Pop MPLS action

This action pops the MPLS header from the packet.

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPActionSetField`**(*field=None*, ***kwargs*)
Set field action

This action modifies a header field in the packet.

Attribute	Description
field	Instance of `OFPMatchField`

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPActionExperimenter`**(*experimenter*, *type_=None*, *len_=None*)
Experimenter action

This action is an extensible action for the experimenter.

Attribute	Description
experimenter	Experimenter ID

2.5 Ryu API Reference

class `ryu.base.app_manager.`**`RyuApp`**(**_args*, ***_kwargs*)
The base class for Ryu applications.

RyuApp subclasses are instantiated after ryu-manager loaded all requested Ryu application modules. __init__ should call RyuApp.__init__ with the same arguments. It's illegal to send any events in __init__.

The instance attribute 'name' is the name of the class used for message routing among Ryu applications. (Cf. send_event) It's set to __class__.__name__ by RyuApp.__init__. It's discouraged for subclasses to override this.

`OFP_VERSIONS` **= None**
A list of supported OpenFlow versions for this RyuApp. The default is all versions supported by the framework.

Examples:

```
OFP_VERSIONS = [ofproto_v1_0.OFP_VERSION,
                ofproto_v1_2.OFP_VERSION]
```

If multiple Ryu applications are loaded in the system, the intersection of their OFP_VERSIONS is used.

`_CONTEXTS` **= {}**
A dictionary to specify contexts which this Ryu application wants to use. Its key is a name of context and its value is an ordinary class which implements the context. The class is instantiated by app_manager and the instance is shared among RyuApp subclasses which has _CONTEXTS member with the same key. A RyuApp subclass can obtain a reference to the instance via its __init__'s kwargs as the following.

Example:

```
_CONTEXTS = {
    'network': network.Network
}

def __init__(self, *args, *kwargs):
    self.network = kwargs['network']
```

`_EVENTS` **= []**
A list of event classes which this RyuApp subclass would generate. This should be specified if and only if event classes are defined in a different python module from the RyuApp subclass is.

close()
teardown method. The method name, close, is chosen for python context manager

classmethod **context_iteritems**()
Return iterator over the (key, contxt class) of application context

reply_to_request(*req*, *rep*)
Send a reply for a synchronous request sent by send_request. The first argument should be an instance of EventRequestBase. The second argument should be an instance of EventReplyBase.

send_event(*name*, *ev*, *state=None*)
Send the specified event to the RyuApp instance specified by name.

send_event_to_observers(*ev*, *state=None*)
Send the specified event to all observers of this RyuApp.

send_request(*req*)
Make a synchronous request. Set req.sync to True, send it to a Ryu application specified by req.dst, and block until receiving a reply. Returns the received reply. The argument should be an instance of EventRequestBase.

start()
Hook that is called after startup initialization is done.

class ryu.controller.dpset.**DPSet**
DPSet application manages a set of switches (datapaths) connected to this controller.

get(*dp_id*)
This method returns the ryu.controller.controller.Datapath instance for the given Datapath ID. Raises KeyError if no such a datapath connected to this controller.

get_all()
This method returns a list of tuples which represents instances for switches connected to this controller. The tuple consists of a Datapath Id and an instance of ryu.controller.controller.Datapath. A return value looks like the following:

[(dpid_A, Datapath_A), (dpid_B, Datapath_B), ...]

get_port(*dpid*, *port_no*)
This method returns the ryu.controller.dpset.PortState instance for the given Datapath ID and the port number. Raises ryu_exc.PortNotFound if no such a datapath connected to this controller or no such a port exists.

get_ports(*dpid*)
This method returns a list of ryu.controller.dpset.PortState instances for the given Datapath ID. Raises KeyError if no such a datapath connected to this controller.

CHAPTER

THREE

CONFIGURATION

3.1 Setup TLS Connection

If you want to use secure channel to connect OpenFlow switches, you need to use TLS connection. This document describes how to setup Ryu to connect to the Open vSwitch over TLS.

3.1.1 Configuring a Public Key Infrastructure

If you don't have a PKI, the ovs-pki script included with Open vSwitch can help you. This section is based on the INSTALL.SSL in the Open vSwitch source code.

NOTE: How to install Open vSwitch isn't described in this document. Please refer to the Open vSwitch documents.

Create a PKI by using ovs-pki script:

```
% ovs-pki init
(Default directory is /usr/local/var/lib/openvswitch/pki)
```

The pki directory consists of controllerca and switchca subdirectories. Each directory contains CA files.

Create a controller private key and certificate:

```
% ovs-pki req+sign ctl controller
```

ctl-privkey.pem and ctl-cert.pem are generated in the current directory.

Create a switch private key and certificate:

```
% ovs-pki req+sign sc switch
```

sc-privkey.pem and sc-cert.pem are generated in the current directory.

3.1.2 Testing TLS Connection

Configuring ovs-vswitchd to use CA files using the ovs-vsctl "set-ssl" command, e.g.:

```
% ovs-vsctl set-ssl /etc/openvswitch/sc-privkey.pem \
  /etc/openvswitch/sc-cert.pem \
  /usr/local/var/lib/openvswitch/pki/controllerca/cacert.pem
% ovs-vsctl add-br br0
% ovs-vsctl set-controller br0 ssl:127.0.0.1:6633
```

Substitute the correct file names, if they differ from the ones used above. You should use absolute file names.

Run Ryu with CA files:

```
% ryu-manager --ctl-privkey ctl-privkey.pem \
              --ctl-cert ctl-cert.pem \
              --ca-certs /usr/local/var/lib/openvswitch/pki/switchca/cacert.pem \
              --verbose
```

You can see something like:

```
loading app ryu.controller.ofp_handler
instantiating app ryu.controller.ofp_handler
BRICK ofp_event
  CONSUMES EventOFPSwitchFeatures
  CONSUMES EventOFPErrorMsg
  CONSUMES EventOFPHello
  CONSUMES EventOFPEchoRequest
connected socket:<SSLSocket fileno=4 sock=127.0.0.1:6633 peer=127.0.0.1:61302> a
ddress:('127.0.0.1', 61302)
hello ev <ryu.controller.ofp_event.EventOFPHello object at 0x1047806d0>
move onto config mode
switch features ev version: 0x1 msg_type 0x6 xid 0xb0bb34e5 port OFPPhyPort(port
_no=65534, hw_addr='\x16\xdc\xa2\xe2}K', name='br0\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00', config=0, state=0, curr=0, advertised=0, supported=0, p
eer=0)
move onto main mode
```

CHAPTER

FOUR

USING RYU NETWORK OPERATING SYSTEM WITH OPENSTACK AS OPENFLOW CONTROLLER

Ryu cooperates with OpenStack using Quantum Ryu plugin. The plugin is available in the official Quantum releases.

For more information, please visit http://github.com/osrg/ryu/wiki/OpenStack . We described instructions of the installation / configuration of OpenStack with Ryu, and we provide pre-configured VM image to be able to easily try OpenStack with Ryu.

- OpenStack: http://www.openstack.org/
- Quantum: https://github.com/openstack/quantum/

CHAPTER

FIVE

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

PYTHON MODULE INDEX