ryu Documentation

Release 4.19

ryu development team

Nov 05, 2017

Contents

Contents:

CHAPTER 1

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, 1.5 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; pip install .
```

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 Optional Requirements

Some functionalities of ryu requires extra packages:

- OF-Config requires lxml and ncclient
- NETCONF requires paramiko
- BGP speaker (SSH console) requires paramiko
- Zebra protocol service (database) requires SQLAlchemy

If you want to use the functionalities, please install requirements:

```
% pip install -r tools/optional-requires
```

Please refer to tools/optional-requires for details.

1.4 Prerequisites

If you got some error messages at installation step, please confirm dependencies for building required Python packages.

On Ubuntu(16.04 LTS or later):

```
% apt install gcc python-dev libffi-dev libssl-dev libxml2-dev libxslt1-dev zlib1g-dev
```

1.5 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 2

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
from ryu.ofproto import ofproto_v1_0

class L2Switch(app_manager.RyuApp):
    OFP_VERSIONS = [ofproto_v1_0.OFP_VERSION]

    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 Components of Ryu

2.2.1 Executables

bin/ryu-manager

The main executable.

2.2.2 Base components

ryu.base.app_manager

The central management of Ryu applications.

- Load Ryu applications
- Provide *contexts* to Ryu applications
- Route messages among Ryu applications

2.2.3 OpenFlow controller

ryu.controller.controller

The main component of OpenFlow controller.

- Handle connections from switches
- Generate and route events to appropriate entities like Ryu applications

ryu.controller.dpset

Manage switches.

Planned to be replaced by ryu/topology.

ryu.controller.ofp_event

OpenFlow event definitions.

ryu.controller.ofp_handler

Basic OpenFlow handling including negotiation.

2.2.4 OpenFlow wire protocol encoder and decoder

ryu.ofproto.ofproto_v1_0

OpenFlow 1.0 definitions.

ryu.ofproto.ofproto_v1_0_parser

Decoder/Encoder implementations of OpenFlow 1.0.

ryu.ofproto.ofproto_v1_2

OpenFlow 1.2 definitions.

ryu.ofproto.ofproto_v1_2_parser

Decoder/Encoder implementations of OpenFlow 1.2.

ryu.ofproto.ofproto_v1_3

OpenFlow 1.3 definitions.

ryu.ofproto.ofproto_v1_3_parser

This module implements OpenFlow 1.3.x.

This module also implements some of extensions shown in “OpenFlow Extensions for 1.3.X Pack 1”. Namely, the following extensions are implemented.

- EXT-236 Bad flow entry priority error Extension
- EXT-237 Set async config error Extension
- EXT-256 PBB UCA header field Extension
- EXT-260 Duplicate instruction error Extension
- EXT-264 Multipart timeout errors Extension

The following extensions are partially implemented.

- EXT-187 Flow entry notifications Extension (ONFMP_FLOW_MONITOR only)
- EXT-230 Bundle Extension (Error codes only)
- EXT-232 Table synchronisation Extension (Error codes only)

The following extensions are not implemented yet.

- EXT-191 Role Status Extension
- EXT-192-e Flow entry eviction Extension
- EXT-192-v Vacancy events Extension

ryu.ofproto.ofproto_v1_4

OpenFlow 1.4 definitions.

ryu.ofproto.ofproto_v1_4_parser

Decoder/Encoder implementations of OpenFlow 1.4.

ryu.ofproto.ofproto_v1_5

OpenFlow 1.5 definitions.

ryu.ofproto.ofproto_v1_5_parser

Decoder/Encoder implementations of OpenFlow 1.5.

2.2.5 Ryu applications

ryu.app.cbench

A dumb OpenFlow 1.0 responder for benchmarking the controller framework. Intended to be used with oflops cbench.

ryu.app.simple_switch

An OpenFlow 1.0 L2 learning switch implementation.

ryu.topology

Switch and link discovery module. Planned to replace ryu/controller/dpset.

2.2.6 Libraries

ryu.lib.packet

Ryu packet library. Decoder/Encoder implementations of popular protocols like TCP/IP.

ryu.lib.ovs

ovsdb interaction library.

ryu.lib.of_config

OF-Config implementation.

ryu.lib.netconf

NETCONF definitions used by ryu/lib/of_config.

ryu.lib.xflow

An implementation of sFlow and NetFlow.

2.2.7 Third party libraries

ryu.contrib.ovs

Open vSwitch python binding. Used by ryu.lib.ovs.

ryu.contrib.oslo.config

Oslo configuration library. Used for ryu-manager's command-line options and configuration files.

ryu.contrib.ncclient

Python library for NETCONF client. Used by ryu.lib.of_config.

2.3 Ryu application API

2.3.1 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 to 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 keeps draining the receive queue by dequeueing an event and calling the appropritate event handler for the event type. Because the event handler is called in the context of the event processing thread, it should be careful when blocking. 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.

2.3.2 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.

2.3.3 Observe events

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

2.3.4 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.

2.3.5 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 are subclasses of the following class.

class `ryu.controller.ofp_event.`**`EventOFPMsgBase`**(*msg*)

The base class of OpenFlow event class.

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.
timestamp	Timestamp when Datapath instance generated this event.

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.

2.3.6 ryu.base.app_manager.RyuApp

See *Ryu API Reference*.

2.3.7 ryu.controller.handler.set_ev_cls

`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.

2.3.8 ryu.controller.controller.Datapath

class `ryu.controller.controller.`**`Datapath`**(*socket*, *address*)

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.
ofproto_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

2.3.9 ryu.controller.event.EventBase

class `ryu.controller.event.`**`EventBase`**
The base of all event classes.

A Ryu application can define its own event type by creating a subclass.

2.3.10 ryu.controller.event.EventRequestBase

class `ryu.controller.event.`**`EventRequestBase`**
The base class for synchronous request for RyuApp.send_request.

2.3.11 ryu.controller.event.EventReplyBase

class `ryu.controller.event.`**`EventReplyBase`**(*dst*)
The base class for synchronous request reply for RyuApp.send_reply.

2.3.12 ryu.controller.ofp_event.EventOFPStateChange

class `ryu.controller.ofp_event.`**`EventOFPStateChange`**(*dp*)
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

2.3.13 ryu.controller.ofp_event.EventOFPPortStateChange

class `ryu.controller.ofp_event.`**`EventOFPPortStateChange`**(*dp, reason, port_no*)
An event class to notify the port state changes of Dtatapath instance.

This event performs like EventOFPPortStatus, but Ryu will send this event after updating `ports` dict of Datapath instances. An instance has at least the following attributes.

Attribute	Description
datapath	ryu.controller.controller.Datapath instance of the switch
reason	one of OFPPR_*
port_no	Port number which state was changed

2.3.14 ryu.controller.dpset.EventDP

class `ryu.controller.dpset.`**`EventDP`**(*dp, enter_leave*)
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.
ports	A list of port instances.

2.3.15 ryu.controller.dpset.EventPortAdd

class `ryu.controller.dpset.`**`EventPortAdd`**(*dp*, *port*)

An event class for switch port status "ADD" 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

2.3.16 ryu.controller.dpset.EventPortDelete

class `ryu.controller.dpset.`**`EventPortDelete`**(*dp*, *port*)

An event class for switch port status "DELETE" 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

2.3.17 ryu.controller.dpset.EventPortModify

class `ryu.controller.dpset.`**`EventPortModify`**(*dp*, *new_port*)

An event class for switch port status "MODIFY" 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

2.3.18 ryu.controller.network.EventNetworkPort

class `ryu.controller.network.`**`EventNetworkPort`**(*network_id*, *dpid*, *port_no*, *add_del*)

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.

2.3.19 ryu.controller.network.EventNetworkDel

class `ryu.controller.network.`**`EventNetworkDel`**(*network_id*)

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

2.3.20 ryu.controller.network.EventMacAddress

class `ryu.controller.network.`**`EventMacAddress`**(*dpid*, *port_no*, *network_id*, *mac_address*, *add_del*)

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.

2.3.21 ryu.controller.tunnels.EventTunnelKeyAdd

class `ryu.controller.tunnels.`**`EventTunnelKeyAdd`**(*network_id*, *tunnel_key*)

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

2.3.22 ryu.controller.tunnels.EventTunnelKeyDel

class `ryu.controller.tunnels.`**`EventTunnelKeyDel`**(*network_id*, *tunnel_key*)

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

2.3.23 ryu.controller.tunnels.EventTunnelPort

class `ryu.controller.tunnels.`**`EventTunnelPort`**(*dpid*, *port_no*, *remote_dpid*, *add_del*)

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.

2.4 Library

Ryu provides some useful library for your network applications.

2.4.1 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.

```
from ryu.ofproto import ether
from ryu.lib.packet import ethernet, arp, packet

e = ethernet.ethernet(dst='ff:ff:ff:ff:ff:ff',
                      src='08:60:6e:7f:74:e7',
                      ethertype=ether.ETH_TYPE_ARP)
a = arp.arp(hwtype=1, proto=0x0800, hlen=6, plen=4, opcode=2,
            src_mac='08:60:6e:7f:74:e7', src_ip='192.0.2.1',
            dst_mac='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()
print repr(p.data)  # the on-wire packet
```

2.4.2 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.

List of the sub-classes:

- *`ryu.lib.packet.bgp.StreamParser`*

Protocol Header classes

Packet Base Class

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.

ARP

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	Hardware address.	
proto	Protocol address.	
hlen	byte length of each hardware address.	
plen	byte length of each protocol address.	
opcode	operation codes.	
src_mac	Hardware address of sender.	'08:60:6e:7f:74:e7'
src_ip	Protocol address of sender.	'192.0.2.1'
dst_mac	Hardware address of target.	'00:00:00:00:00:00'
dst_ip	Protocol address of target.	'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)

BFD

BFD Control packet parser/serializer

[RFC 5880] BFD Control packet format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Vers |  Diag   |Sta|P|F|C|A|D|M|  Detect Mult  |    Length     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       My Discriminator                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Your Discriminator                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Desired Min TX Interval                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Required Min RX Interval                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Required Min Echo RX Interval                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

An optional Authentication Section MAY be present in the following format of types:

1. Format of Simple Password Authentication Section:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Auth Type   |   Auth Len    |  Auth Key ID  |  Password...  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

2. Format of Keyed MD5 and Meticulous Keyed MD5 Authentication Section:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Auth Type   |   Auth Len    |  Auth Key ID  |   Reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Auth Key/Digest...                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3. Format of Keyed SHA1 and Meticulous Keyed SHA1 Authentication Section:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Auth Type   |   Auth Len    |  Auth Key ID  |   Reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Auth Key/Hash...                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

class `ryu.lib.packet.bfd.`**`BFDAuth`**(*auth_len=None*)

Base class of BFD (RFC 5880) Authentication Section

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
auth_type	The authentication type in use.
auth_len	The length, in bytes, of the authentication section, including the `auth_type` and `auth_len` fields.

classmethod **`parser_hdr`**(*buf*)

Parser for common part of authentication section.

`serialize_hdr`()

Serialization function for common part of authentication section.

class `ryu.lib.packet.bfd.`**`KeyedMD5`**(*auth_key_id*, *seq*, *auth_key=None*, *digest=None*, *auth_len=None*)

BFD (RFC 5880) Keyed MD5 Authentication Section class

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

Attribute	Description
auth_type	(Fixed) The authentication type in use.
auth_key_id	The authentication Key ID in use.
seq	The sequence number for this packet. This value is incremented occasionally.
auth_key	The shared MD5 key for this packet.
digest	(Optional) The 16-byte MD5 digest for the packet.
auth_len	(Fixed) The length of the authentication section is 24 bytes.

`authenticate`(*prev*, *auth_keys=None*)

Authenticate the MD5 digest for this packet.

This method can be invoked only when `self.digest` is defined.

Returns a boolean indicates whether the digest can be authenticated by the correspondent Auth Key or not.

`prev` is a `bfd` instance for the BFD Control header which this authentication section belongs to. It's necessary to be assigned because an MD5 digest must be calculated over the entire BFD Control packet.

`auth_keys` is a dictionary of authentication key chain which key is an integer of *Auth Key ID* and value is a string of *Auth Key*.

`serialize`(*payload*, *prev*)
Encode a Keyed MD5 Authentication Section.

This method is used only when encoding an BFD Control packet.

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

`prev` is a `bfd` instance for the BFD Control header which this authentication section belongs to. It's necessary to be assigned because an MD5 digest must be calculated over the entire BFD Control packet.

class `ryu.lib.packet.bfd.`**`KeyedSHA1`**(*auth_key_id*, *seq*, *auth_key=None*, *auth_hash=None*, *auth_len=None*)
BFD (RFC 5880) Keyed SHA1 Authentication Section class

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

Attribute	Description
auth_type	(Fixed) The authentication type in use.
auth_key_id	The authentication Key ID in use.
seq	The sequence number for this packet. This value is incremented occasionally.
auth_key	The shared SHA1 key for this packet.
auth_hash	(Optional) The 20-byte SHA1 hash for the packet.
auth_len	(Fixed) The length of the authentication section is 28 bytes.

`authenticate`(*prev*, *auth_keys=None*)
Authenticate the SHA1 hash for this packet.

This method can be invoked only when `self.auth_hash` is defined.

Returns a boolean indicates whether the hash can be authenticated by the correspondent Auth Key or not.

`prev` is a `bfd` instance for the BFD Control header which this authentication section belongs to. It's necessary to be assigned because an SHA1 hash must be calculated over the entire BFD Control packet.

`auth_keys` is a dictionary of authentication key chain which key is an integer of *Auth Key ID* and value is a string of *Auth Key*.

`serialize`(*payload*, *prev*)
Encode a Keyed SHA1 Authentication Section.

This method is used only when encoding an BFD Control packet.

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

`prev` is a `bfd` instance for the BFD Control header which this authentication section belongs to. It's necessary to be assigned because an SHA1 hash must be calculated over the entire BFD Control packet.

class `ryu.lib.packet.bfd.`**`MeticulousKeyedMD5`**(*auth_key_id*, *seq*, *auth_key=None*, *digest=None*, *auth_len=None*)
BFD (RFC 5880) Meticulous Keyed MD5 Authentication Section class

All methods of this class are inherited from `KeyedMD5`.

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

Attribute	Description
auth_type	(Fixed) The authentication type in use.
auth_key_id	The authentication Key ID in use.
seq	The sequence number for this packet. This value is incremented for each successive packet transmitted for a session.
auth_key	The shared MD5 key for this packet.
digest	(Optional) The 16-byte MD5 digest for the packet.
auth_len	(Fixed) The length of the authentication section is 24 bytes.

class `ryu.lib.packet.bfd.`**`MeticulousKeyedSHA1`**(*auth_key_id*, *seq*, *auth_key=None*, *auth_hash=None*, *auth_len=None*)

BFD (RFC 5880) Meticulous Keyed SHA1 Authentication Section class

All methods of this class are inherited from `KeyedSHA1`.

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

Attribute	Description
auth_type	(Fixed) The authentication type in use.
auth_key_id	The authentication Key ID in use.
seq	The sequence number for this packet. This value is incremented for each successive packet transmitted for a session.
auth_key	The shared SHA1 key for this packet.
auth_hash	(Optional) The 20-byte SHA1 hash for the packet.
auth_len	(Fixed) The length of the authentication section is 28 bytes.

class `ryu.lib.packet.bfd.`**`SimplePassword`**(*auth_key_id*, *password*, *auth_len=None*)

BFD (RFC 5880) Simple Password Authentication Section class

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

Attribute	Description
auth_type	(Fixed) The authentication type in use.
auth_key_id	The authentication Key ID in use.
password	The simple password in use on this session. The password is a binary string, and MUST be from 1 to 16 bytes in length.
auth_len	The length, in bytes, of the authentication section, including the `auth_type` and `auth_len` fields.

`authenticate`(*prev=None*, *auth_keys=None*)

Authenticate the password for this packet.

This method can be invoked only when `self.password` is defined.

Returns a boolean indicates whether the password can be authenticated or not.

`prev` is a `bfd` instance for the BFD Control header. It's not necessary for authenticating the Simple Password.

`auth_keys` is a dictionary of authentication key chain which key is an integer of *Auth Key ID* and value is a string of *Password*.

`serialize`(*payload*, *prev*)

Encode a Simple Password Authentication Section.

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

`prev` is a `bfd` instance for the BFD Control header. It's not necessary for encoding only the Simple Password section.

class `ryu.lib.packet.bfd.`**`bfd`**(*ver=1, diag=0, state=0, flags=0, detect_mult=0, my_discr=0, your_discr=0, desired_min_tx_interval=0, required_min_rx_interval=0, required_min_echo_rx_interval=0, auth_cls=None, length=None*)

BFD (RFC 5880) Control packet encoder/decoder class.

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

- RFC 5880 Generic BFD Control 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
ver	The version number of the protocol. This class implements protocol version 1.
diag	A diagnostic code specifying the local system's reason for the last change in session state.
state	The current BFD session state as seen by the transmitting system.
flags	Bitmap of the following flags. BFD_FLAG_POLL BFD_FLAG_FINAL BFD_FLAG_CTRL_PLANE_INDEP BFD_FLAG_AUTH_PRESENT BFD_FLAG_DEMAND BFD_FLAG_MULTIPOINT
detect_mult	Detection time multiplier.
my_discr	My Discriminator.
your_discr	Your Discriminator.
desired_min_tx_interval	Desired Min TX Interval. (in microseconds)
required_min_rx_interval	Required Min RX Interval. (in microseconds)
required_min_echo_rx_interval	Required Min Echo RX Interval. (in microseconds)
auth_cls	(Optional) Authentication Section instance. It's defined only when the Authentication Present (A) bit is set in flags. Assign an instance of the following classes: `SimplePassword`, `KeyedMD5`, `MeticulousKeyedMD5`, `KeyedSHA1`, and `MeticulousKeyedSHA1`.
length	(Optional) Length of the BFD Control packet, in bytes.

`authenticate`(**args, **kwargs*)

Authenticate this packet.

Returns a boolean indicates whether the packet can be authenticated or not.

Returns `False` if the Authentication Present (A) is not set in the flag of this packet.

Returns `False` if the Authentication Section for this packet is not present.

For the description of the arguemnts of this method, refer to the authentication method of the Authentication Section classes.

`pack`()

Encode a BFD Control packet without authentication section.

BGP

RFC 4271 BGP-4

exception `ryu.lib.packet.bgp.`**`AdminShutdown`**`(`*data=''*`)`
Error to indicate Administrative shutdown.

RFC says: If a BGP speaker decides to administratively shut down its peering with a neighbor, then the speaker SHOULD send a NOTIFICATION message with the Error Code Cease and the Error Subcode 'Administrative Shutdown'.

exception `ryu.lib.packet.bgp.`**`AttrFlagError`**`(`*data=''*`)`
Error to indicate recognized path attributes have incorrect flags.

RFC says: If any recognized attribute has Attribute Flags that conflict with the Attribute Type Code, then the Error Subcode MUST be set to Attribute Flags Error. The Data field MUST contain the erroneous attribute (type, length, and value).

class `ryu.lib.packet.bgp.`**`BGPEvpnEsImportRTExtendedCommunity`**`(`***kwargs*`)`
ES-Import Route Target Extended Community

class `ryu.lib.packet.bgp.`**`BGPEvpnEsiLabelExtendedCommunity`**`(`*label=None, mpls_label=None, vni=None, **kwargs*`)`
ESI Label Extended Community

class `ryu.lib.packet.bgp.`**`BGPEvpnMacMobilityExtendedCommunity`**`(`***kwargs*`)`
MAC Mobility Extended Community

class `ryu.lib.packet.bgp.`**`BGPFlowSpecRedirectCommunity`**`(`***kwargs*`)`
Flow Specification Traffic Filtering Actions for Redirect.

Attribute	Description
as_number	Autonomous System number.
local_administrator	Local Administrator.

class `ryu.lib.packet.bgp.`**`BGPFlowSpecTPIDActionCommunity`**`(`***kwargs*`)`
Flow Specification TPID Actions.

Attribute	Description
actions	Bit representation of actions. Supported actions are `TI(inner TPID action)` and `TO(outer TPID action)`.
tpid_1	TPID used by `TI`.
tpid_2	TPID used by `TO`.

class `ryu.lib.packet.bgp.`**`BGPFlowSpecTrafficActionCommunity`**`(`***kwargs*`)`
Flow Specification Traffic Filtering Actions for Traffic Action.

Attribute	Description
action	Apply action. The supported action are `SAMPLE` and `TERMINAL`.

class `ryu.lib.packet.bgp.`**`BGPFlowSpecTrafficMarkingCommunity`**`(`***kwargs*`)`
Flow Specification Traffic Filtering Actions for Traffic Marking.

Attribute	Description
dscp	Differentiated Services Code Point.

class `ryu.lib.packet.bgp.`**`BGPFlowSpecTrafficRateCommunity`**`(`***kwargs*`)`
Flow Specification Traffic Filtering Actions for Traffic Rate.

Attribute	Description
as_number	Autonomous System number.
rate_info	rate information.

class `ryu.lib.packet.bgp.`**`BGPFlowSpecVlanActionCommunity`**(***kwargs*)
Flow Specification Vlan Actions.

Attribute	Description
actions_1	Bit representation of actions. Supported actions are `POP`, `PUSH`, `SWAP`, `REWRITE_INNER`, `REWRITE_OUTER`.
actions_2	Same as `actions_1`.
vlan_1	VLAN ID used by `actions_1`.
cos_1	Class of Service used by `actions_1`.
vlan_2	VLAN ID used by `actions_2`.
cos_2	Class of Service used by `actions_2`.

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.

class `ryu.lib.packet.bgp.`**`BGPMessage`**(*marker=None*, *len_=None*, *type_=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.`**`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.
error_code	Error code field.
error_subcode	Error subcode field.
data	Data field.

class `ryu.lib.packet.bgp.`**`BGPOpen`**(*my_as*, *bgp_identifier*, *type_=1*, *opt_param_len=0*, *opt_param=None*, *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.
version	Version field.
my_as	My Autonomous System field. 2 octet unsigned integer.
hold_time	Hold Time field. 2 octet unsigned integer.
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.`**`BGPPathAttributePmsiTunnel`**(*pmsi_flags, tunnel_type, mpls_label=None, label=None, vni=None, tunnel_id=None, flags=0, type_=None, length=None*)

P-Multicast Service Interface Tunnel (PMSI Tunnel) attribute

class `ryu.lib.packet.bgp.`**`BGPRouteRefresh`**(*afi, safi, demarcation=0, type_=5, len_=None, marker=None*)

BGP-4 ROUTE REFRESH Message (RFC 2918) 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.
afi	Address Family Identifier
safi	Subsequent Address Family Identifier

class `ryu.lib.packet.bgp.`**`BGPUpdate`**(*type_=2, withdrawn_routes_len=None, withdrawn_routes=None, total_path_attribute_len=None, path_attributes=None, nlri=None, 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.
withdrawn_routes_len	Withdrawn Routes Length field. Ignored when encoding.
withdrawn_routes	Withdrawn Routes field. A list of BGPWithdrawnRoute instances. The default is [].
total_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 [].

exception `ryu.lib.packet.bgp.`**`BadBgpId`**(*data=''*)

Error to indicate incorrect BGP Identifier.

RFC says: If the BGP Identifier field of the OPEN message is syntactically incorrect, then the Error Subcode MUST be set to Bad BGP Identifier. Syntactic correctness means that the BGP Identifier field represents a valid unicast IP host address.

exception `ryu.lib.packet.bgp.`**`BadMsg`**(*msg_type*)
Error to indicate un-recognized message type.

RFC says: If the Type field of the message header is not recognized, then the Error Subcode MUST be set to Bad Message Type. The Data field MUST contain the erroneous Type field.

exception `ryu.lib.packet.bgp.`**`BadPeerAs`**(*data=''*)
Error to indicate open message has incorrect AS number.

RFC says: If the Autonomous System field of the OPEN message is unacceptable, then the Error Subcode MUST be set to Bad Peer AS. The determination of acceptable Autonomous System numbers is configure peer AS.

exception `ryu.lib.packet.bgp.`**`BgpExc`**(*data=''*)
Base bgp exception.

`CODE` **= 0**
BGP error code.

`SEND_ERROR` **= True**
Flag if set indicates Notification message should be sent to peer.

`SUB_CODE` **= 0**
BGP error sub-code.

exception `ryu.lib.packet.bgp.`**`CollisionResolution`**(*data=''*)
Error to indicate Connection Collision Resolution.

RFC says: If a BGP speaker decides to send a NOTIFICATION message with the Error Code Cease as a result of the collision resolution procedure (as described in [BGP-4]), then the subcode SHOULD be set to "Connection Collision Resolution".

exception `ryu.lib.packet.bgp.`**`ConnRejected`**(*data=''*)
Error to indicate Connection Rejected.

RFC says: If a BGP speaker decides to disallow a BGP connection (e.g., the peer is not configured locally) after the speaker accepts a transport protocol connection, then the BGP speaker SHOULD send a NOTIFICATION message with the Error Code Cease and the Error Subcode "Connection Rejected".

class `ryu.lib.packet.bgp.`**`EvpnASBasedEsi`**(*as_number*, *local_disc*, *type_=None*)
AS based ESI value

This type indicates an Autonomous System(AS)-based ESI Value that can be auto-generated or configured by the operator.

class `ryu.lib.packet.bgp.`**`EvpnArbitraryEsi`**(*value*, *type_=None*)
Arbitrary 9-octet ESI value

This type indicates an arbitrary 9-octet ESI value, which is managed and configured by the operator.

class `ryu.lib.packet.bgp.`**`EvpnEsi`**(*type_=None*)
Ethernet Segment Identifier

The supported ESI Types:

- `EvpnEsi.ARBITRARY` indicates EvpnArbitraryEsi.
- `EvpnEsi.LACP` indicates EvpnLACPEsi.
- `EvpnEsi.L2_BRIDGE` indicates EvpnL2BridgeEsi.

- `EvpnEsi.MAC_BASED` indicates EvpnMacBasedEsi.
- `EvpnEsi.ROUTER_ID` indicates EvpnRouterIDEsi.
- `EvpnEsi.AS_BASED` indicates EvpnASBasedEsi.

class `ryu.lib.packet.bgp.`**`EvpnEthernetAutoDiscoveryNLRI`**(*route_dist*, *esi*, *ethernet_tag_id*, *mpls_label=None*, *vni=None*, *label=None*, *type_=None*, *length=None*)

Ethernet A-D route type specific EVPN NLRI

class `ryu.lib.packet.bgp.`**`EvpnEthernetSegmentNLRI`**(*route_dist*, *esi*, *ip_addr*, *ip_addr_len=None*, *type_=None*, *length=None*)

Ethernet Segment route type specific EVPN NLRI

class `ryu.lib.packet.bgp.`**`EvpnInclusiveMulticastEthernetTagNLRI`**(*route_dist*, *ethernet_tag_id*, *ip_addr*, *ip_addr_len=None*, *type_=None*, *length=None*)

Inclusive Multicast Ethernet Tag route type specific EVPN NLRI

class `ryu.lib.packet.bgp.`**`EvpnIpPrefixNLRI`**(*route_dist*, *ethernet_tag_id*, *ip_prefix*, *esi=None*, *gw_ip_addr=None*, *mpls_label=None*, *vni=None*, *label=None*, *type_=None*, *length=None*)

IP Prefix advertisement route NLRI

class `ryu.lib.packet.bgp.`**`EvpnL2BridgeEsi`**(*mac_addr*, *priority*, *type_=None*)

ESI value for Layer 2 Bridge

This type is used in the case of indirectly connected hosts via a bridged LAN between the CEs and the PEs. The ESI Value is auto-generated and determined based on the Layer 2 bridge protocol.

class `ryu.lib.packet.bgp.`**`EvpnLACPEsi`**(*mac_addr*, *port_key*, *type_=None*)

ESI value for LACP

When IEEE 802.1AX LACP is used between the PEs and CEs, this ESI type indicates an auto-generated ESI value determined from LACP.

class `ryu.lib.packet.bgp.`**`EvpnMacBasedEsi`**(*mac_addr*, *local_disc*, *type_=None*)

MAC-based ESI Value

This type indicates a MAC-based ESI Value that can be auto-generated or configured by the operator.

class `ryu.lib.packet.bgp.`**`EvpnMacIPAdvertisementNLRI`**(*route_dist*, *ethernet_tag_id*, *mac_addr*, *ip_addr*, *esi=None*, *mpls_labels=None*, *vni=None*, *labels=None*, *mac_addr_len=None*, *ip_addr_len=None*, *type_=None*, *length=None*)

MAC/IP Advertisement route type specific EVPN NLRI

class `ryu.lib.packet.bgp.`**`EvpnNLRI`**(*type_=None*, *length=None*)

BGP Network Layer Reachability Information (NLRI) for EVPN

class `ryu.lib.packet.bgp.`**`EvpnRouterIDEsi`**(*router_id*, *local_disc*, *type_=None*)

Router-ID ESI Value

This type indicates a router-ID ESI Value that can be auto-generated or configured by the operator.

class `ryu.lib.packet.bgp.`**`EvpnUnknownEsi`**(*value*, *type_=None*)
ESI value for unknown type

class `ryu.lib.packet.bgp.`**`EvpnUnknownNLRI`**(*value*, *type_*, *length=None*)
Unknown route type specific EVPN NLRI

exception `ryu.lib.packet.bgp.`**`FiniteStateMachineError`**(*data=''*)
Error to indicate any Finite State Machine Error.

RFC says: Any error detected by the BGP Finite State Machine (e.g., receipt of an unexpected event) is indicated by sending the NOTIFICATION message with the Error Code Finite State Machine Error.

class `ryu.lib.packet.bgp.`**`FlowSpecComponentUnknown`**(*buf*, *type_=None*)
Unknown component type for Flow Specification NLRI component

class `ryu.lib.packet.bgp.`**`FlowSpecDSCP`**(*operator*, *value*, *type_=None*)
Diffserv Code Point for Flow Specification NLRI component

Set the 6-bit DSCP field at value. [RFC2474]

class `ryu.lib.packet.bgp.`**`FlowSpecDestPort`**(*operator*, *value*, *type_=None*)
Destination port number for Flow Specification NLRI component

Set the destination port of a TCP or UDP packet at value.

class `ryu.lib.packet.bgp.`**`FlowSpecDestPrefix`**(*length*, *addr*, *type_=None*)
Destination Prefix for Flow Specification NLRI component

class `ryu.lib.packet.bgp.`**`FlowSpecDestinationMac`**(*length*, *addr*, *type_=None*)
Destination Mac Address.

Set the Mac Address at value.

class `ryu.lib.packet.bgp.`**`FlowSpecEtherType`**(*operator*, *value*, *type_=None*)
Ethernet Type field in an Ethernet frame.

Set the 2 byte value of an Ethernet Type field at value.

class `ryu.lib.packet.bgp.`**`FlowSpecFragment`**(*operator*, *value*, *type_=None*)
Fragment for Flow Specification NLRI component

Set the bitmask for operand format at value. The following values are supported.

Attribute	Description
LF	Last fragment
FF	First fragment
ISF	Is a fragment
DF	Don't fragment

class `ryu.lib.packet.bgp.`**`FlowSpecIPProtocol`**(*operator*, *value*, *type_=None*)
IP Protocol for Flow Specification NLRI component

Set the IP protocol number at value.

class `ryu.lib.packet.bgp.`**`FlowSpecIPv4NLRI`**(*length=0*, *rules=None*)
Flow Specification NLRI class for IPv4 [RFC 5575]

classmethod **`from_user`**(***kwargs*)
Utility method for creating a NLRI instance.

This function returns a NLRI instance from human readable format value.

Parameters **`kwargs`** – The following arguments are available.

Argument	Value	Operator	Description
dst_prefix	IPv4 Prefix	Nothing	Destination Prefix.
src_prefix	IPv4 Prefix	Nothing	Source Prefix.
ip_proto	Integer	Numeric	IP Protocol.
port	Integer	Numeric	Port number.
dst_port	Integer	Numeric	Destination port number.
src_port	Integer	Numeric	Source port number.
icmp_type	Integer	Numeric	ICMP type.
icmp_code	Integer	Numeric	ICMP code.
tcp_flags	Fixed string	Bitmask	TCP flags. Supported values are `CWR`, `ECN`, `URGENT`, `ACK`, `PUSH`, `RST`, `SYN` and `FIN`.
packet_len	Integer	Numeric	Packet length.
dscp	Integer	Numeric	Differentiated Services Code Point.
fragment	Fixed string	Bitmask	Fragment. Supported values are `DF` (Don't fragment), `ISF` (Is a fragment), `FF` (First fragment) and `LF` (Last fragment)

Example:

```
>>> msg = bgp.FlowSpecIPv4NLRI.from_user(
...     dst_prefix='10.0.0.0/24',
...     src_prefix='20.0.0.1/24',
...     ip_proto=6,
...     port='80 | 8000',
...     dst_port='>9000 & <9050',
...     src_port='>=8500 & <=9000',
...     icmp_type=0,
...     icmp_code=6,
...     tcp_flags='SYN+ACK & !=URGENT',
...     packet_len=1000,
...     dscp='22 | 24',
...     fragment='LF | ==FF')
>>>
```

You can specify conditions with the following keywords.

The following keywords can be used when the operator type is Numeric.

Keyword	Description
<	Less than comparison between data and value.
<=	Less than or equal to comparison between data and value.
>	Greater than comparison between data and value.
>=	Greater than or equal to comparison between data and value.
==	Equality between data and value. This operator can be omitted.

The following keywords can be used when the operator type is Bitmask.

Keyword	Description
!=	Not equal operation.
==	Exact match operation if specified. Otherwise partial match operation.
+	Used for the summation of bitmask values. (e.g., SYN+ACK)

You can combine the multiple conditions with the following operators.

Keyword	Description
|	Logical OR operation
&	Logical AND operation

Returns A instance of FlowSpecVPNv4NLRI.

class `ryu.lib.packet.bgp.`**`FlowSpecIPv6DestPrefix`**(*length*, *addr*, *offset=0*, *type_=None*)
IPv6 destination Prefix for Flow Specification NLRI component

class `ryu.lib.packet.bgp.`**`FlowSpecIPv6Fragment`**(*operator*, *value*, *type_=None*)
Fragment for Flow Specification for IPv6 NLRI component

Attribute	Description
LF	Last fragment
FF	First fragment
ISF	Is a fragment

class `ryu.lib.packet.bgp.`**`FlowSpecIPv6NLRI`**(*length=0*, *rules=None*)
Flow Specification NLRI class for IPv6 [RFC draft-ietf-idr-flow-spec-v6-08]

classmethod **`from_user`**(***kwargs*)
Utility method for creating a NLRI instance.

This function returns a NLRI instance from human readable format value.

Parameters **`kwargs`** – The following arguments are available.

Argument	Value	Operator	Description
dst_prefix	IPv6 Prefix	Nothing	Destination Prefix.
src_prefix	IPv6 Prefix	Nothing	Source Prefix.
next_header	Integer	Numeric	Next Header.
port	Integer	Numeric	Port number.
dst_port	Integer	Numeric	Destination port number.
src_port	Integer	Numeric	Source port number.
icmp_type	Integer	Numeric	ICMP type.
icmp_code	Integer	Numeric	ICMP code.
tcp_flags	Fixed string	Bitmask	TCP flags. Supported values are `CWR`, `ECN`, `URGENT`, `ACK`, `PUSH`, `RST`, `SYN` and `FIN`.
packet_len	Integer	Numeric	Packet length.
dscp	Integer	Numeric	Differentiated Services Code Point.
fragment	Fixed string	Bitmask	Fragment. Supported values are `ISF` (Is a fragment), `FF` (First fragment) and `LF` (Last fragment)
flow_label	Intefer	Numeric	Flow Label.

Note: For `dst_prefix` and `src_prefix`, you can give "offset" value like this: `2001::2/128/32`. At this case, `offset` is 32. `offset` can be omitted, then `offset` is treated as 0.

class `ryu.lib.packet.bgp.`**`FlowSpecIPv6SrcPrefix`**(*length*, *addr*, *offset=0*, *type_=None*)
IPv6 source Prefix for Flow Specification NLRI component

class `ryu.lib.packet.bgp.`**`FlowSpecIcmpCode`**(*operator*, *value*, *type_=None*)
ICMP code Flow Specification NLRI component

Set the code field of an ICMP packet at value.

class `ryu.lib.packet.bgp.`**`FlowSpecIcmpType`**(*operator*, *value*, *type_=None*)
ICMP type for Flow Specification NLRI component

Set the type field of an ICMP packet at value.

class `ryu.lib.packet.bgp.`**`FlowSpecInnerVLANCoS`**(*operator*, *value*, *type_=None*)
VLAN CoS Fields in an Inner Ethernet frame.

Set the 3 bit CoS field at value..

class `ryu.lib.packet.bgp.`**`FlowSpecInnerVLANID`**(*operator*, *value*, *type_=None*)
Inner VLAN ID.

Set VLAN ID at value.

class `ryu.lib.packet.bgp.`**`FlowSpecL2VPNNLRI`**(*length=0*, *route_dist=None*, *rules=None*)
Flow Specification NLRI class for L2VPN [draft-ietf-idr-flowspec-l2vpn-05]

classmethod **`from_user`**(*route_dist*, ***kwargs*)

Utility method for creating a L2VPN NLRI instance.

This function returns a L2VPN NLRI instance from human readable format value.

Parameters **`kwargs`** – The following arguments are available.

Argument	Value	Operator	Description
ether_type	Integer	Numeric	Ethernet Type.
src_mac	Mac Address	Nothing	Source Mac address.
dst_mac	Mac Address	Nothing	Destination Mac address.
llc_ssap	Integer	Numeric	Source Service Access Point in LLC.
llc_dsap	Integer	Numeric	Destination Service Access Point in LLC.
llc_control	Integer	Numeric	Control field in LLC.
snap	Integer	Numeric	Sub-Network Access Protocol field.
vlan_id	Integer	Numeric	VLAN ID.
vlan_cos	Integer	Numeric	VLAN COS field.
inner_vlan_id	Integer	Numeric	Inner VLAN ID.
inner_vlan_cos	Integer	Numeric	Inner VLAN COS field.

class `ryu.lib.packet.bgp.`**`FlowSpecLLCControl`**(*operator*, *value*, *type_=None*)

Control field in LLC header in an Ethernet frame.

Set the Contorol field at value.

class `ryu.lib.packet.bgp.`**`FlowSpecLLCDSAP`**(*operator*, *value*, *type_=None*)

Destination SAP field in LLC header in an Ethernet frame.

Set the 2 byte value of an Destination SAP at value.

class `ryu.lib.packet.bgp.`**`FlowSpecLLCSSAP`**(*operator*, *value*, *type_=None*)

Source SAP field in LLC header in an Ethernet frame.

Set the 2 byte value of an Source SAP at value.

class `ryu.lib.packet.bgp.`**`FlowSpecNextHeader`**(*operator*, *value*, *type_=None*)

Next Header value in IPv6 packets

Set the IP protocol number at value

class `ryu.lib.packet.bgp.`**`FlowSpecPacketLen`**(*operator*, *value*, *type_=None*)

Packet length for Flow Specification NLRI component

Set the total IP packet length at value.

class `ryu.lib.packet.bgp.`**`FlowSpecPort`**(*operator*, *value*, *type_=None*)

Port number for Flow Specification NLRI component

Set the source or destination TCP/UDP ports at value.

class `ryu.lib.packet.bgp.`**`FlowSpecSNAP`**(*operator*, *value*, *type_=None*)

Sub-Network Access Protocol field in an Ethernet frame.

Set the 5 byte SNAP field at value.

class `ryu.lib.packet.bgp.`**`FlowSpecSourceMac`**(*length*, *addr*, *type_=None*)

Source Mac Address.

Set the Mac Address at value.

class `ryu.lib.packet.bgp.`**`FlowSpecSrcPort`**(*operator*, *value*, *type_=None*)

Source port number for Flow Specification NLRI component

Set the source port of a TCP or UDP packet at value.

class `ryu.lib.packet.bgp.`**`FlowSpecSrcPrefix`**(*length*, *addr*, *type_=None*)
Source Prefix for Flow Specification NLRI component

class `ryu.lib.packet.bgp.`**`FlowSpecTCPFlags`**(*operator*, *value*, *type_=None*)
TCP flags for Flow Specification NLRI component

Supported TCP flags are CWR, ECN, URGENT, ACK, PUSH, RST, SYN and FIN.

class `ryu.lib.packet.bgp.`**`FlowSpecVLANCoS`**(*operator*, *value*, *type_=None*)
VLAN CoS Fields in an Ethernet frame.

Set the 3 bit CoS field at value.

class `ryu.lib.packet.bgp.`**`FlowSpecVLANID`**(*operator*, *value*, *type_=None*)
VLAN ID.

Set VLAN ID at value.

class `ryu.lib.packet.bgp.`**`FlowSpecVPNv4NLRI`**(*length=0*, *route_dist=None*, *rules=None*)
Flow Specification NLRI class for VPNv4 [RFC 5575]

classmethod **`from_user`**(*route_dist*, ***kwargs*)
Utility method for creating a NLRI instance.

This function returns a NLRI instance from human readable format value.

Parameters

- **`route_dist`** – Route Distinguisher.
- **`kwargs`** – See *`ryu.lib.packet.bgp.FlowSpecIPv4NLRI`*

Example:

```
>>> msg = bgp.FlowSpecIPv4NLRI.from_user(
...     route_dist='65000:1000',
...     dst_prefix='10.0.0.0/24',
...     src_prefix='20.0.0.1/24',
...     ip_proto=6,
...     port='80 | 8000',
...     dst_port='>9000 & <9050',
...     src_port='>=8500 & <=9000',
...     icmp_type=0,
...     icmp_code=6,
...     tcp_flags='SYN+ACK & !=URGENT',
...     packet_len=1000,
...     dscp='22 | 24',
...     fragment='LF | ==FF')
>>>
```

class `ryu.lib.packet.bgp.`**`FlowSpecVPNv6NLRI`**(*length=0*, *route_dist=None*, *rules=None*)
Flow Specification NLRI class for VPNv6 [draft-ietf-idr-flow-spec-v6-08]

classmethod **`from_user`**(*route_dist*, ***kwargs*)
Utility method for creating a NLRI instance.

This function returns a NLRI instance from human readable format value.

Parameters

- **`route_dist`** – Route Distinguisher.
- **`kwargs`** – See *`ryu.lib.packet.bgp.FlowSpecIPv6NLRI`*

exception `ryu.lib.packet.bgp.`**`HoldTimerExpired`**(*data=''*)
Error to indicate Hold Timer expired.

RFC says: If a system does not receive successive KEEPALIVE, UPDATE, and/or NOTIFICATION messages within the period specified in the Hold Time field of the OPEN message, then the NOTIFICATION message with the Hold Timer Expired Error Code is sent and the BGP connection is closed.

exception `ryu.lib.packet.bgp.`**`InvalidOriginError`**(*data=''*)
Error indicates undefined Origin attribute value.

RFC says: If the ORIGIN attribute has an undefined value, then the Error Sub- code MUST be set to Invalid Origin Attribute. The Data field MUST contain the unrecognized attribute (type, length, and value).

exception `ryu.lib.packet.bgp.`**`MalformedAsPath`**(*data=''*)
Error to indicate if AP_PATH attribute is syntactically incorrect.

RFC says: The AS_PATH attribute is checked for syntactic correctness. If the path is syntactically incorrect, then the Error Subcode MUST be set to Malformed AS_PATH.

exception `ryu.lib.packet.bgp.`**`MalformedAttrList`**(*data=''*)
Error to indicate UPDATE message is malformed.

RFC says: Error checking of an UPDATE message begins by examining the path attributes. If the Withdrawn Routes Length or Total Attribute Length is too large (i.e., if Withdrawn Routes Length + Total Attribute Length + 23 exceeds the message Length), then the Error Subcode MUST be set to Malformed Attribute List.

exception `ryu.lib.packet.bgp.`**`MalformedOptionalParam`**(*data=''*)
If recognized optional parameters are malformed.

RFC says: If one of the Optional Parameters in the OPEN message is recognized, but is malformed, then the Error Subcode MUST be set to 0 (Unspecific).

exception `ryu.lib.packet.bgp.`**`MissingWellKnown`**(*pattr_type_code*)
Error to indicate missing well-known attribute.

RFC says: If any of the well-known mandatory attributes are not present, then the Error Subcode MUST be set to Missing Well-known Attribute. The Data field MUST contain the Attribute Type Code of the missing, well-known attribute.

exception `ryu.lib.packet.bgp.`**`OptAttrError`**(*data=''*)
Error indicates Optional Attribute is malformed.

RFC says: If an optional attribute is recognized, then the value of this attribute MUST be checked. If an error is detected, the attribute MUST be discarded, and the Error Subcode MUST be set to Optional Attribute Error. The Data field MUST contain the attribute (type, length, and value).

class `ryu.lib.packet.bgp.`**`PmsiTunnelIdUnknown`**(*value*)
Unknown route type specific _PmsiTunnelId

class `ryu.lib.packet.bgp.`**`RouteTargetMembershipNLRI`**(*origin_as*, *route_target*)
Route Target Membership NLRI.

Route Target membership NLRI is advertised in BGP UPDATE messages using the MP_REACH_NLRI and MP_UNREACH_NLRI attributes.

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.

exception `ryu.lib.packet.bgp.`**`UnacceptableHoldTime`**(*data=''*)
Error to indicate Unacceptable Hold Time in open message.

RFC says: If the Hold Time field of the OPEN message is unacceptable, then the Error Subcode MUST be set to Unacceptable Hold Time.

exception `ryu.lib.packet.bgp.`**`UnsupportedOptParam`**(*data=''*)
Error to indicate unsupported optional parameters.

RFC says: If one of the Optional Parameters in the OPEN message is not recognized, then the Error Subcode MUST be set to Unsupported Optional Parameters.

exception `ryu.lib.packet.bgp.`**`UnsupportedVersion`**(*locally_support_version*)
Error to indicate unsupport bgp version number.

RFC says: If the version number in the Version field of the received OPEN message is not supported, then the Error Subcode MUST be set to Unsupported Version Number. The Data field is a 2-octet unsigned integer, which indicates the largest, locally-supported version number less than the version the remote BGP peer bid (as indicated in the received OPEN message), or if the smallest, locally-supported version number is greater than the version the remote BGP peer bid, then the smallest, locally- supported version number.

BMP

BGP Monitoring Protocol draft-ietf-grow-bmp-07

class `ryu.lib.packet.bmp.`**`BMPInitiation`**(*info*, *type_=4*, *len_=None*, *version=3*)
BMP Initiation Message

Attribute	Description
version	Version. this packet lib defines BMP ver. 3
len	Length field. Ignored when encoding.
type	Type field. one of BMP_MSG_ constants.
info	One or more piece of information encoded as a TLV

class `ryu.lib.packet.bmp.`**`BMPMessage`**(*type_*, *len_=None*, *version=3*)
Base class for BGP Monitoring Protocol 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
version	Version. this packet lib defines BMP ver. 3
len	Length field. Ignored when encoding.
type	Type field. one of BMP_MSG_ constants.

class `ryu.lib.packet.bmp.`**`BMPPeerDownNotification`**(*reason*, *data*, *peer_type*, *is_post_policy*, *peer_distinguisher*, *peer_address*, *peer_as*, *peer_bgp_id*, *timestamp*, *version=3*, *type_=2*, *len_=None*)
BMP Peer Down Notification Message

Attribute	Description
version	Version. this packet lib defines BMP ver. 3
len	Length field. Ignored when encoding.
type	Type field. one of BMP_MSG_ constants.
reason	Reason indicates why the session was closed.
data	vary by the reason.

class `ryu.lib.packet.bmp.`**`BMPPeerMessage`**(*peer_type*, *is_post_policy*, *peer_distinguisher*, *peer_address*, *peer_as*, *peer_bgp_id*, *timestamp*, *version=3*, *type_=None*, *len_=None*)
BMP Message with Per Peer Header

Following BMP Messages contain Per Peer Header after Common BMP Header.

- BMP_MSG_TYPE_ROUTE_MONITRING
- BMP_MSG_TYPE_STATISTICS_REPORT
- BMP_MSG_PEER_UP_NOTIFICATION

Attribute	Description
version	Version. this packet lib defines BMP ver. 3
len	Length field. Ignored when encoding.
type	Type field. one of BMP_MSG_ constants.
peer_type	The type of the peer.
is_post_policy	Indicate the message reflects the post-policy Adj-RIB-In
peer_distinguisher	Use for L3VPN router which can have multiple instance.
peer_address	The remote IP address associated with the TCP session.
peer_as	The Autonomous System number of the peer.
peer_bgp_id	The BGP Identifier of the peer
timestamp	The time when the encapsulated routes were received.

class `ryu.lib.packet.bmp.`**`BMPPeerUpNotification`**(*local_address*, *local_port*, *remote_port*, *sent_open_message*, *received_open_message*, *peer_type*, *is_post_policy*, *peer_distinguisher*, *peer_address*, *peer_as*, *peer_bgp_id*, *timestamp*, *version=3*, *type_=3*, *len_=None*)

BMP Peer Up Notification Message

Attribute	Description
version	Version. this packet lib defines BMP ver. 3
len	Length field. Ignored when encoding.
type	Type field. one of BMP_MSG_ constants.
peer_type	The type of the peer.
peer_flags	Provide more information about the peer.
peer_distinguisher	Use for L3VPN router which can have multiple instance.
peer_address	The remote IP address associated with the TCP session.
peer_as	The Autonomous System number of the peer.
peer_bgp_id	The BGP Identifier of the peer
timestamp	The time when the encapsulated routes were received.
local_address	The local IP address associated with the peering TCP session.
local_port	The local port number associated with the peering TCP session.
remote_port	The remote port number associated with the peering TCP session.
sent_open_message	The full OPEN message transmitted by the monitored router to its peer.
received_open_message	The full OPEN message received by the monitored router from its peer.

class `ryu.lib.packet.bmp.`**`BMPRouteMonitoring`**(*bgp_update*, *peer_type*, *is_post_policy*, *peer_distinguisher*, *peer_address*, *peer_as*, *peer_bgp_id*, *timestamp*, *version=3*, *type_=0*, *len_=None*)

BMP Route Monitoring Message

Attribute	Description
version	Version. this packet lib defines BMP ver. 3
len	Length field. Ignored when encoding.
type	Type field. one of BMP_MSG_ constants.
peer_type	The type of the peer.
peer_flags	Provide more information about the peer.
peer_distinguisher	Use for L3VPN router which can have multiple instance.
peer_address	The remote IP address associated with the TCP session.
peer_as	The Autonomous System number of the peer.
peer_bgp_id	The BGP Identifier of the peer
timestamp	The time when the encapsulated routes were received.
bgp_update	BGP Update PDU

class `ryu.lib.packet.bmp.`**`BMPStatisticsReport`**(*stats*, *peer_type*, *is_post_policy*, *peer_distinguisher*, *peer_address*, *peer_as*, *peer_bgp_id*, *timestamp*, *version=3*, *type_=1*, *len_=None*)

BMP Statistics Report Message

Attribute	Description
version	Version. this packet lib defines BMP ver. 3
len	Length field. Ignored when encoding.
type	Type field. one of BMP_MSG_ constants.
peer_type	The type of the peer.
peer_flags	Provide more information about the peer.
peer_distinguisher	Use for L3VPN router which can have multiple instance.
peer_address	The remote IP address associated with the TCP session.
peer_as	The Autonomous System number of the peer.
peer_bgp_id	The BGP Identifier of the peer
timestamp	The time when the encapsulated routes were received.
stats	Statistics (one or more stats encoded as a TLV)

class `ryu.lib.packet.bmp.`**`BMPTermination`**(*info*, *type_=5*, *len_=None*, *version=3*)

BMP Termination Message

Attribute	Description
version	Version. this packet lib defines BMP ver. 3
len	Length field. Ignored when encoding.
type	Type field. one of BMP_MSG_ constants.
info	One or more piece of information encoded as a TLV

BPDU

Bridge Protocol Data Unit(BPDU, IEEE 802.1D) parser/serializer http://standards.ieee.org/getieee802/download/802.1D-2004.pdf

Configuration BPDUs format

Structure	Octet
Protocol Identifier = 0000 0000 0000 0000	1 - 2
Protocol Version Identifier = 0000 0000	3
BPDU Type = 0000 0000	4
Flags	5
Root Identifier **include - priority** system ID extension MAC address	6 - 13
Root Path Cost	14 - 17
Bridge Identifier **include - priority** system ID extension MAC address	18 - 25
Port Identifier **include - priority** port number	26 - 27
Message Age	28 - 29
Max Age	30 - 31
Hello Time	32 - 33
Forward Delay	34 - 35

Topology Change NotificationBPDUs format

Structure	Octet
Protocol Identifier = 0000 0000 0000 0000	1 - 2
Protocol Version Identifier = 0000 0000	3
BPDU Type = 1000 0000	4

Rapid Spanning Tree BPDUs(RST BPDUs) format

Structure	Octet
Protocol Identifier = 0000 0000 0000 0000	1 - 2
Protocol Version Identifier = 0000 0010	3
BPDU Type = 0000 0010	4
Flags	5
Root Identifier **include - priority** system ID extension MAC address	6 - 13
Root Path Cost	14 - 17
Bridge Identifier **include - priority** system ID extension MAC address	18 - 25
Port Identifier **include - priority** port number	26 - 27
Message Age	28 - 29
Max Age	30 - 31
Hello Time	32 - 33
Forward Delay	34 - 35
Version 1 Length = 0000 0000	36

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.`**`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.bpdu.`**`TopologyChangeNotificationBPDUs`**

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

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

Bridge Protocol Data Unit(BPDU) header encoder/decoder base class.

CFM

class `ryu.lib.packet.cfm.`**`cc_message`**(*md_lv=0, version=0, rdi=0, interval=4, seq_num=0, mep_id=1, md_name_format=4, md_name_length=0, md_name='0', short_ma_name_format=2, short_ma_name_length=0, short_ma_name='1', tlvs=None*)

CFM (IEEE Std 802.1ag-2007) Continuity Check Message (CCM) encoder/decoder class.

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

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
md_lv	Maintenance Domain Level.
version	The protocol version number.
rdi	RDI bit.
interval	CCM Interval.The default is 4 (1 frame/s)
seq_num	Sequence Number.
mep_id	Maintenance association End Point Identifier.
md_name_format	Maintenance Domain Name Format. The default is 4 (Character string)
md_name_length	Maintenance Domain Name Length. (0 means automatically-calculate when encoding.)
md_name	Maintenance Domain Name.
short_ma_name_format	Short MA Name Format. The default is 2 (Character string)
short_ma_name_length	Short MA Name Format Length. (0 means automatically-calculate when encoding.)
short_ma_name	Short MA Name.
tlvs	TLVs.

class `ryu.lib.packet.cfm.`**`cfm`**(*op=None*)

CFM (Connectivity Fault Management) Protocol header class.

http://standards.ieee.org/getieee802/download/802.1ag-2007.pdf

OpCode Field range assignments

OpCode range	CFM PDU or organization
0	Reserved for IEEE 802.1
1	Continuity Check Message (CCM)
2	Loopback Reply (LBR)
3	Loopback Message (LBM)
4	Linktrace Reply (LTR)
5	Linktrace Message (LTM)
06 - 31	Reserved for IEEE 802.1
32 - 63	Defined by ITU-T Y.1731
64 - 255	Reserved for IEEE 802.1.

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	CFM PDU

class `ryu.lib.packet.cfm.`**`data_tlv`**(*length=0*, *data_value=''*)

CFM (IEEE Std 802.1ag-2007) Data TLV encoder/decoder class.

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

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 Value field. (0 means automatically-calculate when encoding)
data_value	Bit pattern of any of n octets.(n = length)

class `ryu.lib.packet.cfm.`**`interface_status_tlv`**(*length=0*, *interface_status=1*)

CFM (IEEE Std 802.1ag-2007) Interface Status TLV encoder/decoder class.

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

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 Value field. (0 means automatically-calculate when encoding.)
interface_status	Interface Status.The default is 1 (isUp)

class `ryu.lib.packet.cfm.`**`link_trace_message`**(*md_lv=0*, *version=0*, *use_fdb_only=1*, *transaction_id=0*, *ttl=64*, *ltm_orig_addr='00:00:00:00:00:00'*, *ltm_targ_addr='00:00:00:00:00:00'*, *tlvs=None*)

CFM (IEEE Std 802.1ag-2007) Linktrace Message (LTM) encoder/decoder class.

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

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
md_lv	Maintenance Domain Level.
version	The protocol version number.
use_fdb_only	UseFDBonly bit.
transaction_id	LTM Transaction Identifier.
ttl	LTM TTL.
ltm_orig_addr	Original MAC Address.
ltm_targ_addr	Target MAC Address.
tlvs	TLVs.

class `ryu.lib.packet.cfm.`**`link_trace_reply`**(*md_lv=0*, *version=0*, *use_fdb_only=1*, *fwd_yes=0*, *terminal_mep=1*, *transaction_id=0*, *ttl=64*, *relay_action=1*, *tlvs=None*)

CFM (IEEE Std 802.1ag-2007) Linktrace Reply (LTR) encoder/decoder class.

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

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	The protocol version number.
use_fdb_only	UseFDBonly bit.
fwd_yes	FwdYes bit.
terminal_mep	TerminalMep bit.
transaction_id	LTR Transaction Identifier.
ttl	Reply TTL.
relay_action	Relay Action.The default is 1 (RlyHit)
tlvs	TLVs.

class `ryu.lib.packet.cfm.`**`loopback_message`**(*md_lv=0*, *version=0*, *transaction_id=0*, *tlvs=None*)

CFM (IEEE Std 802.1ag-2007) Loopback Message (LBM) encoder/decoder class.

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

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
md_lv	Maintenance Domain Level.
version	The protocol version number.
transaction_id	Loopback Transaction Identifier.
tlvs	TLVs.

class `ryu.lib.packet.cfm.`**`loopback_reply`**(*md_lv=0*, *version=0*, *transaction_id=0*, *tlvs=None*)

CFM (IEEE Std 802.1ag-2007) Loopback Reply (LBR) encoder/decoder class.

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

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
md_lv	Maintenance Domain Level.
version	The protocol version number.
transaction_id	Loopback Transaction Identifier.
tlvs	TLVs.

class `ryu.lib.packet.cfm.`**`ltm_egress_identifier_tlv`**(*length=0*, *egress_id_ui=0*, *egress_id_mac='00:00:00:00:00:00'*)

CFM (IEEE Std 802.1ag-2007) LTM EGRESS TLV encoder/decoder class.

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

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 Value field. (0 means automatically-calculate when encoding.)
egress_id_ui	Egress Identifier of Unique ID.
egress_id_mac	Egress Identifier of MAC address.

class `ryu.lib.packet.cfm.`**`ltr_egress_identifier_tlv`**(*length=0*, *last_egress_id_ui=0*, *last_egress_id_mac='00:00:00:00:00:00'*, *next_egress_id_ui=0*, *next_egress_id_mac='00:00:00:00:00:00'*)

CFM (IEEE Std 802.1ag-2007) LTR EGRESS TLV encoder/decoder class.

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

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 Value field. (0 means automatically-calculate when encoding.)
last_egress_id_ui	Last Egress Identifier of Unique ID.
last_egress_id_mac	Last Egress Identifier of MAC address.
next_egress_id_ui	Next Egress Identifier of Unique ID.
next_egress_id_mac	Next Egress Identifier of MAC address.

class `ryu.lib.packet.cfm.`**`organization_specific_tlv`**(*length=0*, *oui='x00x00x00'*, *subtype=0*, *value=''*)

CFM (IEEE Std 802.1ag-2007) Organization Specific TLV encoder/decoder class.

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

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 Value field. (0 means automatically-calculate when encoding.)
oui	Organizationally Unique Identifier.
subtype	Subtype.
value	Value.(optional)

class `ryu.lib.packet.cfm.`**`port_status_tlv`**`(`*length=0*, *port_status=2*`)`

CFM (IEEE Std 802.1ag-2007) Port Status TLV encoder/decoder class.

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

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 Value field. (0 means automatically-calculate when encoding.)
port_status	Port Status.The default is 1 (psUp)

class `ryu.lib.packet.cfm.`**`reply_egress_tlv`**`(`*length=0*, *action=1*, *mac_address='00:00:00:00:00:00'*, *port_id_length=0*, *port_id_subtype=0*, *port_id=''*`)`

CFM (IEEE Std 802.1ag-2007) Reply Egress TLV encoder/decoder class.

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

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 Value field. (0 means automatically-calculate when encoding.)
action	Egress Action.The default is 1 (EgrOK)
mac_address	Egress MAC Address.
port_id_length	Egress PortID Length. (0 means automatically-calculate when encoding.)
port_id_subtype	Egress PortID Subtype.
port_id	Egress PortID.

class `ryu.lib.packet.cfm.`**`reply_ingress_tlv`**`(`*length=0*, *action=1*, *mac_address='00:00:00:00:00:00'*, *port_id_length=0*, *port_id_subtype=0*, *port_id=''*`)`

CFM (IEEE Std 802.1ag-2007) Reply Ingress TLV encoder/decoder class.

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

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 Value field. (0 means automatically-calculate when encoding.)
action	Ingress Action.The default is 1 (IngOK)
mac_address	Ingress MAC Address.
port_id_length	Ingress PortID Length. (0 means automatically-calculate when encoding.)
port_id_subtype	Ingress PortID Subtype.
port_id	Ingress PortID.

class `ryu.lib.packet.cfm.`**`sender_id_tlv`**`(`*length=0*, *chassis_id_length=0*, *chassis_id_subtype=4*, *chassis_id=''*, *ma_domain_length=0*, *ma_domain=''*, *ma_length=0*, *ma=''*`)`

CFM (IEEE Std 802.1ag-2007) Sender ID TLV encoder/decoder class.

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

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 Value field. (0 means automatically-calculate when encoding.)
chassis_id_length	Chassis ID Length. (0 means automatically-calculate when encoding.)
chassis_id_subtype	Chassis ID Subtype. The default is 4 (Mac Address)
chassis_id	Chassis ID.
ma_domain_length	Management Address Domain Length. (0 means automatically-calculate when encoding.)
ma_domain	Management Address Domain.
ma_length	Management Address Length. (0 means automatically-calculate when encoding.)
ma	Management Address.

DHCP

DHCP packet parser/serializer

class `ryu.lib.packet.dhcp.`**`dhcp`**(*op*, *chaddr*, *options=None*, *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.`**`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.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.

DHCP6

DHCPv6 packet parser/serializer

[RFC 3315] DHCPv6 packet format:

The following diagram illustrates the format of DHCP messages sent between clients and servers:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    msg_type   |               transaction_id                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
.                            options                            .
.                           (variable)                          .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

There are two relay agent messages, which share the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    msg_type   |   hop_count   |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
|                                                               |
|                         link_address                          |
|                                                               |
|                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-|
|                               |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
|                                                               |
|                         peer_address                          |
|                                                               |
|                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-|
|                               |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
.                                                               .
.            options (variable number and length)   ....        .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

class `ryu.lib.packet.dhcp6.`**`dhcp6`**(*msg_type*, *options*, *transaction_id=None*, *hop_count=0*, *link_address='::'*, *peer_address='::'*)

DHCPv6 (RFC 3315) header encoder/decoder class.

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

- RFC 3315 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
msg_type	Identifies the DHCP message type
transaction_id	For unrelayed messages only: the transaction ID for this message exchange.
hop_count	For relayed messages only: number of relay agents that have relayed this message.
link_address	For relayed messages only: a global or site-local address that will be used by the server to identify the link on which the client is located.
peer_address	For relayed messages only: the address of the client or relay agent from which the message to be relayed was received.
options	Options carried in this message

class `ryu.lib.packet.dhcp6.`**`option`**(*code*, *data*, *length=0*)

DHCP (RFC 3315) options encoder/decoder class.

This is used with ryu.lib.packet.dhcp6.dhcp6.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.

The format of DHCP options is:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          option-code          |           option-len          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          option-data                          |
|                      (option-len octets)                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Attribute	Description
option-code	An unsigned integer identifying the specific option type carried in this option.
option-len	An unsigned integer giving the length of the option-data field in this option in octets.
option-data	The data for the option; the format of this data depends on the definition of the option.

class `ryu.lib.packet.dhcp6.`**`options`**(*option_list=None*, *options_len=0*)

DHCP (RFC 3315) options encoder/decoder class.

This is used with ryu.lib.packet.dhcp6.dhcp6.

Ehternet

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.

Geneve

Geneve packet parser/serializer

class `ryu.lib.packet.geneve.`**`Option`**(*option_class=None*, *type_=None*, *length=0*)

Tunnel Options

class `ryu.lib.packet.geneve.`**`OptionDataUnknown`**(*buf*, *option_class=None*, *type_=None*, *length=0*)

Unknown Option Class and Type specific Option

class `ryu.lib.packet.geneve.`**`geneve`**(*version=0*, *opt_len=0*, *flags=0*, *protocol=25944*, *vni=None*, *options=None*)

Geneve (RFC draft-ietf-nvo3-geneve-03) 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
version	Version.
opt_len	The length of the options fields.
flags	Flag field for OAM packet and Critical options present.
protocol	Protocol Type field. The Protocol Type is defined as "ETHER TYPES".
vni	Identifier for unique element of virtual network.
options	List of `Option*` instance.

GRE

class `ryu.lib.packet.gre.`**`gre`**(*version=0*, *protocol=2048*, *checksum=None*, *key=None*, *vsid=None*, *flow_id=None*, *seq_number=None*)

GRE (RFC2784,RFC2890) 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.

At-tribute	Description
version	Version.
protocol	Protocol Type field. The Protocol Type is defined as "ETHER TYPES".
check-sum	Checksum field(optional). When you set a value other than None, this field will be automatically calculated.
key	Key field(optional) This field is intended to be used for identifying an individual traffic flow within a tunnel.
vsid	Virtual Subnet ID field(optional) This field is a 24-bit value that is used to identify the NVGRE-based Virtual Layer 2 Network.
flow_id	FlowID field(optional) This field is an 8-bit value that is used to provide per-flow entropy for flows in the same VSID.
seq_number	Sequence Number field(optional)

`ryu.lib.packet.gre.`**`nvgre`**(*version=0*, *vsid=0*, *flow_id=0*)

Generate instance of GRE class with information for NVGRE (RFC7637).

Parameters

- **`version`** – Version.
- **`vsid`** – Virtual Subnet ID.
- **`flow_id`** – FlowID.

Returns Instance of GRE class with information for NVGRE.

ICMP

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.

ICMPv6

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.

IGMP

Internet Group Management Protocol(IGMP) packet parser/serializer

[RFC 1112] IGMP v1 format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| Type  |    Unused     |           Checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Group Address                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[RFC 2236] IGMP v2 format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Type     | Max Resp Time |           Checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Group Address                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[RFC 3376] IGMP v3 Membership Query format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Type = 0x11  | Max Resp Code |           Checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Group Address                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Resv  |S| QRV |     QQIC      |     Number of Sources (N)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address [1]                      |
+-                                                             -+
|                       Source Address [2]                      |
+-                              .                              -+
.                               .                               .
.                               .                               .
+-                                                             -+
|                       Source Address [N]                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

IGMP v3 Membership Report format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Type = 0x22  |    Reserved   |           Checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Reserved            |  Number of Group Records (M)  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
.                                                               .
.                        Group Record [1]                       .
.                                                               .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
.                                                               .
.                        Group Record [2]                       .
.                                                               .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               .                               |
.                               .                               .
|                               .                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
.                                                               .
.                         Group Record [M]                      .
.                                                               .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Where each Group Record has the following internal format:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Record Type  |  Aux Data Len |     Number of Sources (N)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Multicast Address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address [1]                      |
+-                                                             -+
|                       Source Address [2]                      |
+-                                                             -+
.                               .                               .
.                               .                               .
.                               .                               .
+-                                                             -+
|                       Source Address [N]                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
.                                                               .
.                         Auxiliary Data                        .
.                                                               .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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.

IPv4

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='10.0.0.1'*, *dst='10.0.0.2'*, *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	

IPv6

class `ryu.lib.packet.ipv6.`**`auth`**(*nxt=6*, *size=2*, *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='10::10'*, *dst='20::20'*, *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.ipv6.`**`routing`**(*nxt*)
An IPv6 Routing Header decoder class. This class has only the parser method.

IPv6 Routing Header types.

http://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml

Value	Description	Reference
0	Source Route (DEPRECATED)	[[IPV6]][RFC5095]
1	Nimrod (DEPRECATED 2009-05-06)	
2	Type 2 Routing Header	[RFC6275]
3	RPL Source Route Header	[RFC6554]
4 - 252	Unassigned	
253	RFC3692-style Experiment 1 [2]	[RFC4727]
254	RFC3692-style Experiment 2 [2]	[RFC4727]
255	Reserved	

class `ryu.lib.packet.ipv6.`**`routing_type3`**(*nxt=6*, *size=0*, *type_=3*, *seg=0*, *cmpi=0*, *cmpe=0*, *adrs=None*)

An IPv6 Routing Header for Source Routes with the RPL (RFC 6554) 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 Routing header, not include the first 8 octet. (0 means automatically-calculate when encoding)
type	Identifies the particular Routing header variant.
seg	Number of route segments remaining.
cmpi	Number of prefix octets from segments 1 through n-1.
cmpe	Number of prefix octets from segment n.
pad	Number of octets that are used for padding after Address[n] at the end of the SRH.
adrs	Vector of addresses, numbered 1 to n.

LLC

Logical Link Control(LLC, IEEE 802.2) parser/serializer http://standards.ieee.org/getieee802/download/802.2-1998.pdf

LLC format:

```
+-----------------+--------------+
| DSAP address    | 8 bits       |
+-----------------+--------------+
| SSAP address    | 8 bits       |
+-----------------+--------------+
| Control         | 8 or 16 bits |
+-----------------+--------------+
```

DSAP address field:

```
  LSB
+-----+---+---+---+---+---+---+---+
| I/G | D | D | D | D | D | D | D |
+-----+---+---+---+---+---+---+---+
 I/G bit = 0 : Individual DSAP
 I/G bit = 1 : Group DSA
 D : DSAP address
```

SSAP address field:

```
  LSB
+-----+---+---+---+---+---+---+---+
| C/R | S | S | S | S | S | S | S |
+-----+---+---+---+---+---+---+---+
 C/R bit = 0 : Command
 C/R bit = 1 : Response
 S : SSAP address
```

Control field:

Information transfer command/response (I-format PDU):

```
  1   2   3   4   5   6   7   8    9   10-16
+---+---+---+---+---+---+---+---+-----+------+
| 0 |           N(S)            | P/F | N(R) |
+---+---+---+---+---+---+---+---+-----+------+
```

Supervisory commands/responses (S-format PDUs):

```
  1   2   3   4   5   6   7   8    9   10-16
+---+---+---+---+---+---+---+---+-----+------+
| 1   0 | S   S | 0   0   0   0 | P/F | N(R) |
+---+---+---+---+---+---+---+---+-----+------+
```

Unnumbered commands/responses (U-format PDUs):

```
  1   2   3    4    5    6   7    8
+---+---+----+---+-----+---+----+---+
| 1   1 | M1   M1 | P/F | M2   M2   M2 |
+---+---+----+---+-----+---+----+---+

N(S) : sender send sequence number (Bit 2=lower-order-bit)
N(R) : sender receive sequence number (Bit 10=lower-order-bit)
S    : supervisory function bit
M1/M2: modifier function bit
P/F  : poll bit - command LLC PDUs
       final bit - response LLC PDUs
```

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.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.

LLDP

Link Layer Discovery Protocol(LLDP, IEEE 802.1AB) http://standards.ieee.org/getieee802/download/802.1AB-2009.pdf

basic TLV format:

```
octets | 1          | 2             | 3 ...             n + 2 |
       --------------------------------------------------------
       | TLV type | TLV information | TLV information string  |
       | (7bits)  | string length   | (0-507 octets)          |
       |          | (9bits)         |                         |
       --------------------------------------------------------
bits   |8        2|1|8             1|
```

Organizationally specific TLV format:

```
octets | 1          | 2         | 3 ...  5 | 6          | 7 ...    n + 6 |
       ---------------------------------------------------------------
       | TLV type | Length      | OUI       | Subtype  | Infomation      |
       | (7bits)  | (9bits)     | (24bits) | (8bits)  | (0-507 octets) |
       ---------------------------------------------------------------
bits   |8        2|1|8         1|
```

LLDPDU format:

```
--------------------------------------------------------------------
| Chassis ID | Port ID | TTL | optional TLV | ... | optional TLV | End |
--------------------------------------------------------------------
```

Chasis ID, Port ID, TTL, End are mandatory optional TLV may be inserted in any order

class `ryu.lib.packet.lldp.`**`ChassisID`**(*buf=None*, **args*, ***kwargs*)

Chassis ID TLV encoder/decoder class

Attribute	Description
buf	Binary data to parse.
subtype	Subtype.
chassis_id	Chassis id corresponding to subtype.

class `ryu.lib.packet.lldp.`**`End`**(*buf=None*, **args*, ***kwargs*)

End TLV encoder/decoder class

Attribute	Description
buf	Binary data to parse.

class `ryu.lib.packet.lldp.`**`ManagementAddress`**(*buf=None*, **args*, ***kwargs*)

Management Address TLV encoder/decoder class

Attribute	Description
buf	Binary data to parse.
addr_subtype	Address type.
addr	Device address.
intf_subtype	Interface type.
intf_num	Interface number.
oid	Object ID.

class `ryu.lib.packet.lldp.`**`OrganizationallySpecific`**(*buf=None*, **args*, ***kwargs*)

Organizationally Specific TLV encoder/decoder class

Attribute	Description
buf	Binary data to parse.
oui	Organizationally unique ID.
subtype	Organizationally defined subtype.
info	Organizationally defined information string.

class `ryu.lib.packet.lldp.`**`PortDescription`**(*buf=None*, **args*, ***kwargs*)

Port description TLV encoder/decoder class

Attribute	Description
buf	Binary data to parse.
port_description	Port description.

class `ryu.lib.packet.lldp.`**`PortID`**(*buf=None*, **args*, ***kwargs*)

Port ID TLV encoder/decoder class

Attribute	Description
buf	Binary data to parse.
subtype	Subtype.
port_id	Port ID corresponding to subtype.

class `ryu.lib.packet.lldp.`**`SystemCapabilities`**(*buf=None*, **args*, ***kwargs*)

System Capabilities TLV encoder/decoder class

Attribute	Description
buf	Binary data to parse.
subtype	Subtype.
system_cap	System Capabilities.
enabled_cap	Enabled Capabilities.

class `ryu.lib.packet.lldp.`**`SystemDescription`**(*buf=None*, **args*, ***kwargs*)
System description TLV encoder/decoder class

Attribute	Description
buf	Binary data to parse.
system_description	System description.

class `ryu.lib.packet.lldp.`**`SystemName`**(*buf=None*, **args*, ***kwargs*)
System name TLV encoder/decoder class

Attribute	Description
buf	Binary data to parse.
system_name	System name.

class `ryu.lib.packet.lldp.`**`TTL`**(*buf=None*, **args*, ***kwargs*)
Time To Live TLV encoder/decoder class

Attribute	Description
buf	Binary data to parse.
ttl	Time To Live.

class `ryu.lib.packet.lldp.`**`lldp`**(*tlvs*)
LLDPDU encoder/decoder class.

An instance has the following attributes at least.

Attribute	Description
tlvs	List of TLV instance.

MPLS

`ryu.lib.packet.mpls.`**`label_from_bin`**(*buf*)
Converts binary representation label to integer.

Parameters **`buf`** – Binary representation of label.

Returns MPLS Label and BoS bit.

`ryu.lib.packet.mpls.`**`label_to_bin`**(*mpls_label*, *is_bos=True*)
Converts integer label to binary representation.

Parameters

- **`mpls_label`** – MPLS Label.
- **`is_bos`** – BoS bit.

Returns Binary representation of label.

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

OpenFlow

class `ryu.lib.packet.openflow.`**`OFPUnparseableMsg`**(*datapath*, *version*, *msg_type*, *msg_len*, *xid*, *body*)

Unparseable OpenFlow message encoder class.

An instance has the following attributes at least.

Attribute	Description
datapath	A ryu.ofproto.ofproto_protocol.ProtocolDesc instance for this message or None if OpenFlow protocol version is unsupported version.
version	OpenFlow protocol version
msg_type	Type of OpenFlow message
msg_len	Length of the message
xid	Transaction id
body	OpenFlow body data

Note: "datapath" attribute is different from ryu.controller.controller.Datapath. So you can not use "datapath" attribute to send OpenFlow messages. For example, "datapath" attribute does not have send_msg method.

class `ryu.lib.packet.openflow.`**`openflow`**(*msg*)

OpenFlow message encoder/decoder class.

An instance has the following attributes at least.

Attribute	Description
msg	An instance of OpenFlow message (see *OpenFlow protocol API Reference*) or an instance of OFPUnparseableMsg if failed to parse packet as OpenFlow message.

OSPF

RFC 2328 OSPF version 2

class `ryu.lib.packet.ospf.`**`OSPFMessage`**(*type_*, *length=None*, *router_id='0.0.0.0'*, *area_id='0.0.0.0'*, *au_type=1*, *authentication=0*, *checksum=None*, *version=2*)

Base class for OSPF version 2 messages.

`ryu.lib.packet.ospf.`**`ospf`**

alias of *`OSPFMessage`*

PBB

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

SCTP

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 class is used with the following.

- ryu.lib.packet.sctp.chunk_abort
- 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 class is used with the following.

- ryu.lib.packet.sctp.chunk_abort
- 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 class is used with the following.

- ryu.lib.packet.sctp.chunk_abort
- 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 class is used with the following.

- ryu.lib.packet.sctp.chunk_abort
- 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 class is used with the following.

- ryu.lib.packet.sctp.chunk_abort
- 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 class is used with the following.

- ryu.lib.packet.sctp.chunk_abort
- 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 class is used with the following.

- ryu.lib.packet.sctp.chunk_abort
- 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 class is used with the following.

- ryu.lib.packet.sctp.chunk_abort
- 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 class is used with the following.

- ryu.lib.packet.sctp.chunk_abort
- 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 class is used with the following.

- ryu.lib.packet.sctp.chunk_abort
- 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 class is used with the following.

- ryu.lib.packet.sctp.chunk_abort
- 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 class is used with the following.

- ryu.lib.packet.sctp.chunk_abort
- 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 class is used with the following.

- ryu.lib.packet.sctp.chunk_abort
- 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 class is used with the following.

- 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 class is used with the following.

- 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 class is used with the following.

- 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 class is used with the following.

- 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 class is used with the following.

- 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 class is used with the following.

- 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 class is used with the following.

- 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 class is used with the following.

- 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 class is used with the following.

- 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 class is used with the following.

- 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 class is used with the following.

- 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 class is used with the following.

- 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 class is used with the following.

- 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 class is used with the following.

- 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 class is used with the following.

- 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 class is used with the following.

- 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 class is used with the following.

- ryu.lib.packet.sctp.chunk_init
- 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 class is used with the following.

- ryu.lib.packet.sctp.chunk_heartbeat
- 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 class is used with the following.

- ryu.lib.packet.sctp.chunk_init
- 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 class is used with the following.

- ryu.lib.packet.sctp.chunk_init
- 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 class is used with the following.

- ryu.lib.packet.sctp.chunk_init
- 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 class is used with the following.

- 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 class is used with the following.

- 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 class is used with the following.

- 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=1*, *dst_port=1*, *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.

Slow

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.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

TCP

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

TCP (RFC 793) header encoder/decoder class.

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

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

`has_flags`(**flags*)

Check if flags are set on this packet.

returns boolean if all passed flags is set

Example:

```
>>> pkt = tcp.tcp(bits=(tcp.TCP_SYN | tcp.TCP_ACK))
>>> pkt.has_flags(tcp.TCP_SYN, tcp.TCP_ACK)
True
```

UDP

class `ryu.lib.packet.udp.`**`udp`**(*src_port=1, dst_port=1, 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)

VLAN

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.

VRRP

VRRP packet parser/serializer

[RFC 3768] VRRP v2 packet format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| Type  | Virtual Rtr ID|   Priority    | Count IP Addrs|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Auth Type   |   Adver Int   |          Checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         IP Address (1)                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            .                                  |
|                            .                                  |
|                            .                                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         IP Address (n)                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Authentication Data (1)                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Authentication Data (2)                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[RFC 5798] VRRP v3 packet format:

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                    IPv4 Fields or IPv6 Fields                 |
...                                                             ...
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |Version| Type  | Virtual Rtr ID|   Priority    |Count IPvX Addr|
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |(rsvd) |     Max Adver Int     |          Checksum             |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                                                               |
 +                                                               +
 |                       IPvX Address(es)                        |
 +                                                               +
 +                                                               +
 +                                                               +
 +                                                               +
 |                                                               |
 +                                                               +
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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.)

VXLAN

`ryu.lib.packet.vxlan.`**`vni_from_bin`**(*buf*)

Converts binary representation VNI to integer.

Parameters **`buf`** – binary representation of VNI.

Returns VNI integer.

`ryu.lib.packet.vxlan.`**`vni_to_bin`**(*vni*)

Converts integer VNI to binary representation.

Parameters **`vni`** – integer of VNI

Returns binary representation of VNI.

class `ryu.lib.packet.vxlan.`**`vxlan`**(*vni*)

VXLAN (RFC 7348) 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
vni	VXLAN Network Identifier

Zebra

Zebra protocol parser/serializer

Zebra Protocol is used to communicate with the zebra daemon.

class `ryu.lib.packet.zebra.`**`InterfaceLinkParams`**(*lp_status*, *te_metric*, *max_bw*, *max_reserved_bw*, *unreserved_bw*, *admin_group*, *remote_as*, *remote_ip*, *average_delay*, *min_delay*, *max_delay*, *delay_var*, *pkt_loss*, *residual_bw*, *average_bw*, *utilized_bw*)
Interface Link Parameters class for if_link_params structure.

class `ryu.lib.packet.zebra.`**`NextHopBlackhole`**(*ifindex=None*, *ifname=None*, *addr=None*, *type_=None*)
Nexthop class for ZEBRA_NEXTHOP_BLACKHOLE type.

class `ryu.lib.packet.zebra.`**`NextHopIFIndex`**(*ifindex=None*, *ifname=None*, *addr=None*, *type_=None*)
Nexthop class for ZEBRA_NEXTHOP_IFINDEX type.

class `ryu.lib.packet.zebra.`**`NextHopIFName`**(*ifindex=None*, *ifname=None*, *addr=None*, *type_=None*)
Nexthop class for ZEBRA_NEXTHOP_IFNAME type.

class `ryu.lib.packet.zebra.`**`NextHopIPv4`**(*ifindex=None*, *ifname=None*, *addr=None*, *type_=None*)
Nexthop class for ZEBRA_NEXTHOP_IPV4 type.

class `ryu.lib.packet.zebra.`**`NextHopIPv4IFIndex`**(*ifindex=None*, *ifname=None*, *addr=None*, *type_=None*)
Nexthop class for ZEBRA_NEXTHOP_IPV4_IFINDEX type.

class `ryu.lib.packet.zebra.`**`NextHopIPv4IFName`**(*ifindex=None*, *ifname=None*, *addr=None*, *type_=None*)
Nexthop class for ZEBRA_NEXTHOP_IPV4_IFNAME type.

class `ryu.lib.packet.zebra.`**`NextHopIPv6`**(*ifindex=None*, *ifname=None*, *addr=None*, *type_=None*)
Nexthop class for ZEBRA_NEXTHOP_IPV6 type.

class `ryu.lib.packet.zebra.`**`NextHopIPv6IFIndex`**(*ifindex=None*, *ifname=None*, *addr=None*, *type_=None*)
Nexthop class for ZEBRA_NEXTHOP_IPV6_IFINDEX type.

class `ryu.lib.packet.zebra.`**`NextHopIPv6IFName`**(*ifindex=None*, *ifname=None*, *addr=None*, *type_=None*)
Nexthop class for ZEBRA_NEXTHOP_IPV6_IFNAME type.

class `ryu.lib.packet.zebra.`**`RegisteredNexthop`**(*connected*, *family*, *prefix*)
Unit of ZEBRA_NEXTHOP_REGISTER message body.

class `ryu.lib.packet.zebra.`**`ZebraBfdClientRegister`**(*pid*)
Message body class for FRR_ZEBRA_BFD_CLIENT_REGISTER.

class `ryu.lib.packet.zebra.`**`ZebraBfdDestinationDeregister`**(*pid*, *dst_family*, *dst_prefix*, *multi_hop*, *src_family*, *src_prefix*, *multi_hop_count=None*, *ifname=None*)
Message body class for FRR_ZEBRA_BFD_DEST_DEREGISTER.

class `ryu.lib.packet.zebra.`**`ZebraBfdDestinationRegister`**(*pid*, *dst_family*, *dst_prefix*, *min_rx_timer*, *min_tx_timer*, *detect_mult*, *multi_hop*, *src_family*, *src_prefix*, *multi_hop_count=None*, *ifname=None*)
Message body class for FRR_ZEBRA_BFD_DEST_REGISTER.

class `ryu.lib.packet.zebra.`**`ZebraBfdDestinationReply`**
Message body class for FRR_ZEBRA_BFD_DEST_REPLAY.

class `ryu.lib.packet.zebra.`**`ZebraBfdDestinationUpdate`**(*pid*, *dst_family*, *dst_prefix*, *min_rx_timer*, *min_tx_timer*, *detect_mult*, *multi_hop*, *src_family*, *src_prefix*, *multi_hop_count=None*, *ifname=None*)
Message body class for FRR_ZEBRA_BFD_DEST_UPDATE.

class `ryu.lib.packet.zebra.`**`ZebraHello`**(*route_type*, *instance=None*)
Message body class for ZEBRA_HELLO.

class `ryu.lib.packet.zebra.`**`ZebraIPv4ImportLookup`**(*prefix*, *metric=None*, *nexthops=None*, *from_zebra=False*)
Message body class for ZEBRA_IPV4_IMPORT_LOOKUP.

class `ryu.lib.packet.zebra.`**`ZebraIPv4NexthopAdd`**(*route_type*, *flags*, *message*, *safi=None*, *prefix=None*, *src_prefix=None*, *nexthops=None*, *ifindexes=None*, *distance=None*, *metric=None*, *mtu=None*, *tag=None*, *instance=None*, *from_zebra=False*)
Message body class for FRR_ZEBRA_IPV4_NEXTHOP_ADD.

class `ryu.lib.packet.zebra.`**`ZebraIPv4NexthopDelete`**(*route_type*, *flags*, *message*, *safi=None*, *prefix=None*, *src_prefix=None*, *nexthops=None*, *ifindexes=None*, *distance=None*, *metric=None*, *mtu=None*, *tag=None*, *instance=None*, *from_zebra=False*)
Message body class for FRR_ZEBRA_IPV4_NEXTHOP_DELETE.

class `ryu.lib.packet.zebra.`**`ZebraIPv4NexthopLookup`**(*addr*, *metric=None*, *nexthops=None*)
Message body class for ZEBRA_IPV4_NEXTHOP_LOOKUP.

class `ryu.lib.packet.zebra.`**`ZebraIPv4NexthopLookupMRib`**(*addr*, *distance=None*, *metric=None*, *nexthops=None*)
Message body class for ZEBRA_IPV4_NEXTHOP_LOOKUP_MRIB.

class `ryu.lib.packet.zebra.`**`ZebraIPv4RouteAdd`**(*route_type*, *flags*, *message*, *safi=None*, *prefix=None*, *src_prefix=None*, *nexthops=None*, *ifindexes=None*, *distance=None*, *metric=None*, *mtu=None*, *tag=None*, *instance=None*, *from_zebra=False*)

Message body class for ZEBRA_IPV4_ROUTE_ADD.

class `ryu.lib.packet.zebra.`**`ZebraIPv4RouteDelete`**(*route_type*, *flags*, *message*, *safi=None*, *prefix=None*, *src_prefix=None*, *nexthops=None*, *ifindexes=None*, *distance=None*, *metric=None*, *mtu=None*, *tag=None*, *instance=None*, *from_zebra=False*)
Message body class for ZEBRA_IPV4_ROUTE_DELETE.

class `ryu.lib.packet.zebra.`**`ZebraIPv4RouteIPv6NexthopAdd`**(*route_type*, *flags*, *message*, *safi=None*, *prefix=None*, *src_prefix=None*, *nexthops=None*, *ifindexes=None*, *distance=None*, *metric=None*, *mtu=None*, *tag=None*, *instance=None*, *from_zebra=False*)
Message body class for FRR_ZEBRA_IPV4_ROUTE_IPV6_NEXTHOP_ADD.

class `ryu.lib.packet.zebra.`**`ZebraIPv6ImportLookup`**(*prefix*, *metric=None*, *nexthops=None*, *from_zebra=False*)
Message body class for ZEBRA_IPV6_IMPORT_LOOKUP.

class `ryu.lib.packet.zebra.`**`ZebraIPv6NexthopAdd`**(*route_type*, *flags*, *message*, *safi=None*, *prefix=None*, *src_prefix=None*, *nexthops=None*, *ifindexes=None*, *distance=None*, *metric=None*, *mtu=None*, *tag=None*, *instance=None*, *from_zebra=False*)
Message body class for FRR_ZEBRA_IPV6_NEXTHOP_ADD.

class `ryu.lib.packet.zebra.`**`ZebraIPv6NexthopDelete`**(*route_type*, *flags*, *message*, *safi=None*, *prefix=None*, *src_prefix=None*, *nexthops=None*, *ifindexes=None*, *distance=None*, *metric=None*, *mtu=None*, *tag=None*, *instance=None*, *from_zebra=False*)
Message body class for FRR_ZEBRA_IPV6_NEXTHOP_DELETE.

class `ryu.lib.packet.zebra.`**`ZebraIPv6NexthopLookup`**(*addr*, *metric=None*, *nexthops=None*)
Message body class for ZEBRA_IPV6_NEXTHOP_LOOKUP.

class `ryu.lib.packet.zebra.`**`ZebraIPv6RouteAdd`**(*route_type*, *flags*, *message*, *safi=None*, *prefix=None*, *src_prefix=None*, *nexthops=None*, *ifindexes=None*, *distance=None*, *metric=None*, *mtu=None*, *tag=None*, *instance=None*, *from_zebra=False*)
Message body class for ZEBRA_IPV6_ROUTE_ADD.

class `ryu.lib.packet.zebra.`**`ZebraIPv6RouteDelete`**(*route_type*, *flags*, *message*, *safi=None*, *prefix=None*, *src_prefix=None*, *nexthops=None*, *ifindexes=None*, *distance=None*, *metric=None*, *mtu=None*, *tag=None*, *instance=None*, *from_zebra=False*)
Message body class for ZEBRA_IPV6_ROUTE_DELETE.

class `ryu.lib.packet.zebra.`**`ZebraImportCheckUpdate`**(*family*, *prefix*, *distance=None*, *metric=None*, *nexthops=None*)

Message body class for FRR_ZEBRA_IMPORT_CHECK_UPDATE.

class `ryu.lib.packet.zebra.`**`ZebraImportRouteRegister`**(*nexthops*)
Message body class for FRR_ZEBRA_IMPORT_ROUTE_REGISTER.

class `ryu.lib.packet.zebra.`**`ZebraImportRouteUnregister`**(*nexthops*)
Message body class for FRR_ZEBRA_IMPORT_ROUTE_UNREGISTER.

class `ryu.lib.packet.zebra.`**`ZebraInterfaceAdd`**(*ifname=None*, *ifindex=None*, *status=None*, *if_flags=None*, *ptm_enable=None*, *ptm_status=None*, *metric=None*, *speed=None*, *ifmtu=None*, *ifmtu6=None*, *bandwidth=None*, *ll_type=None*, *hw_addr_len=0*, *hw_addr=None*, *link_params=None*)
Message body class for ZEBRA_INTERFACE_ADD.

class `ryu.lib.packet.zebra.`**`ZebraInterfaceAddressAdd`**(*ifindex*, *ifc_flags*, *family*, *prefix*, *dest*)
Message body class for ZEBRA_INTERFACE_ADDRESS_ADD.

class `ryu.lib.packet.zebra.`**`ZebraInterfaceAddressDelete`**(*ifindex*, *ifc_flags*, *family*, *prefix*, *dest*)
Message body class for ZEBRA_INTERFACE_ADDRESS_DELETE.

class `ryu.lib.packet.zebra.`**`ZebraInterfaceBfdDestinationUpdate`**(*ifindex*, *dst_family*, *dst_prefix*, *status*, *src_family*, *src_prefix*)
Message body class for FRR_ZEBRA_INTERFACE_BFD_DEST_UPDATE.

class `ryu.lib.packet.zebra.`**`ZebraInterfaceDelete`**(*ifname=None*, *ifindex=None*, *status=None*, *if_flags=None*, *ptm_enable=None*, *ptm_status=None*, *metric=None*, *speed=None*, *ifmtu=None*, *ifmtu6=None*, *bandwidth=None*, *ll_type=None*, *hw_addr_len=0*, *hw_addr=None*, *link_params=None*)
Message body class for ZEBRA_INTERFACE_DELETE.

class `ryu.lib.packet.zebra.`**`ZebraInterfaceDisableRadv`**(*ifindex*, *interval*)
Message body class for FRR_ZEBRA_INTERFACE_DISABLE_RADV.

class `ryu.lib.packet.zebra.`**`ZebraInterfaceDown`**(*ifname=None*, *ifindex=None*, *status=None*, *if_flags=None*, *ptm_enable=None*, *ptm_status=None*, *metric=None*, *speed=None*, *ifmtu=None*, *ifmtu6=None*, *bandwidth=None*, *ll_type=None*, *hw_addr_len=0*, *hw_addr=None*, *link_params=None*)
Message body class for ZEBRA_INTERFACE_DOWN.

class `ryu.lib.packet.zebra.`**`ZebraInterfaceEnableRadv`**(*ifindex*, *interval*)
Message body class for FRR_ZEBRA_INTERFACE_ENABLE_RADV.

class `ryu.lib.packet.zebra.`**`ZebraInterfaceLinkParams`**(*ifindex*, *link_params*)
Message body class for ZEBRA_INTERFACE_LINK_PARAMS.

class `ryu.lib.packet.zebra.`**`ZebraInterfaceNbrAddressAdd`**(*ifindex*, *family*, *prefix*)
Message body class for FRR_ZEBRA_INTERFACE_NBR_ADDRESS_ADD.

class `ryu.lib.packet.zebra.`**`ZebraInterfaceNbrAddressDelete`**(*ifindex*, *family*, *prefix*)
Message body class for FRR_ZEBRA_INTERFACE_NBR_ADDRESS_DELETE.

class `ryu.lib.packet.zebra.`**`ZebraInterfaceUp`**(*ifname=None*, *ifindex=None*, *status=None*, *if_flags=None*, *ptm_enable=None*, *ptm_status=None*, *metric=None*, *speed=None*, *ifmtu=None*, *ifmtu6=None*, *bandwidth=None*, *ll_type=None*, *hw_addr_len=0*, *hw_addr=None*, *link_params=None*)

Message body class for ZEBRA_INTERFACE_UP.

class `ryu.lib.packet.zebra.`**`ZebraInterfaceVrfUpdate`**(*ifindex*, *vrf_id*)

Message body class for FRR_ZEBRA_INTERFACE_VRF_UPDATE.

class `ryu.lib.packet.zebra.`**`ZebraMessage`**(*length=None*, *version=3*, *vrf_id=0*, *command=None*, *body=None*)

Zebra protocol parser/serializer 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
length	Total packet length including this header. The minimum length is 3 bytes for version 0 messages, 6 bytes for version 1/2 messages and 8 bytes for version 3 messages.
version	Version number of the Zebra protocol message. To instantiate messages with other than the default version, `version` must be specified.
vrf_id	VRF ID for the route contained in message. Not present in version 0/1/2 messages in the on-wire structure, and always 0 for theses version.
command	Zebra Protocol command, which denotes message type.
body	Messages body. An instance of subclass of `_ZebraMessageBody` named like "Zebra + <message name>" (e.g., `ZebraHello`). Or `None` if message does not contain any body.

Note: To instantiate Zebra messages, `command` can be omitted when the valid `body` is specified.

```
>>> from ryu.lib.packet import zebra
>>> zebra.ZebraMessage(body=zebra.ZebraHello())
ZebraMessage(body=ZebraHello(route_type=14),command=23,
length=None,version=3,vrf_id=0)
```

On the other hand, if `body` is omitted, `command` must be specified.

```
>>> zebra.ZebraMessage(command=zebra.ZEBRA_INTERFACE_ADD)
ZebraMessage(body=None,command=1,length=None,version=3,vrf_id=0)
```

class `ryu.lib.packet.zebra.`**`ZebraMplsLabelsAdd`**(*route_type*, *family*, *prefix*, *gate_addr*, *ifindex=None*, *distance=None*, *in_label=None*, *out_label=None*)

Message body class for FRR_ZEBRA_MPLS_LABELS_ADD.

class `ryu.lib.packet.zebra.`**`ZebraMplsLabelsDelete`**(*route_type*, *family*, *prefix*, *gate_addr*, *ifindex=None*, *distance=None*, *in_label=None*, *out_label=None*)

Message body class for FRR_ZEBRA_MPLS_LABELS_DELETE.

class `ryu.lib.packet.zebra.`**`ZebraNexthopRegister`**(*nexthops*)

Message body class for ZEBRA_NEXTHOP_REGISTER.

class `ryu.lib.packet.zebra.`**`ZebraNexthopUnregister`**(*nexthops*)

Message body class for ZEBRA_NEXTHOP_UNREGISTER.

class `ryu.lib.packet.zebra.`**`ZebraNexthopUpdate`**(*family*, *prefix*, *distance=None*, *metric=None*, *nexthops=None*)
Message body class for ZEBRA_NEXTHOP_UPDATE.

class `ryu.lib.packet.zebra.`**`ZebraRedistributeAdd`**(*route_type*, *afi=None*, *instance=None*)
Message body class for ZEBRA_REDISTRIBUTE_ADD.

class `ryu.lib.packet.zebra.`**`ZebraRedistributeDefaultAdd`**(*route_type*, *afi=None*, *instance=None*)
Message body class for ZEBRA_REDISTRIBUTE_DEFAULT_ADD.

class `ryu.lib.packet.zebra.`**`ZebraRedistributeDefaultDelete`**(*route_type*, *afi=None*, *instance=None*)
Message body class for ZEBRA_REDISTRIBUTE_DEFAULT_DELETE.

class `ryu.lib.packet.zebra.`**`ZebraRedistributeDelete`**(*route_type*, *afi=None*, *instance=None*)
Message body class for ZEBRA_REDISTRIBUTE_DELETE.

class `ryu.lib.packet.zebra.`**`ZebraRedistributeIPv4Add`**(*route_type*, *flags*, *message*, *safi=None*, *prefix=None*, *src_prefix=None*, *nexthops=None*, *ifindexes=None*, *distance=None*, *metric=None*, *mtu=None*, *tag=None*, *instance=None*, *from_zebra=False*)
Message body class for FRR_ZEBRA_IPV4_ROUTE_ADD.

class `ryu.lib.packet.zebra.`**`ZebraRedistributeIPv4Delete`**(*route_type*, *flags*, *message*, *safi=None*, *prefix=None*, *src_prefix=None*, *nexthops=None*, *ifindexes=None*, *distance=None*, *metric=None*, *mtu=None*, *tag=None*, *instance=None*, *from_zebra=False*)
Message body class for FRR_ZEBRA_IPV4_ROUTE_DELETE.

class `ryu.lib.packet.zebra.`**`ZebraRedistributeIPv6Add`**(*route_type*, *flags*, *message*, *safi=None*, *prefix=None*, *src_prefix=None*, *nexthops=None*, *ifindexes=None*, *distance=None*, *metric=None*, *mtu=None*, *tag=None*, *instance=None*, *from_zebra=False*)
Message body class for FRR_ZEBRA_REDISTRIBUTE_IPV6_ADD.

class `ryu.lib.packet.zebra.`**`ZebraRedistributeIPv6Delete`**(*route_type*, *flags*, *message*, *safi=None*, *prefix=None*, *src_prefix=None*, *nexthops=None*, *ifindexes=None*, *distance=None*, *metric=None*, *mtu=None*, *tag=None*, *instance=None*, *from_zebra=False*)
Message body class for FRR_ZEBRA_REDISTRIBUTE_IPV6_DEL.

class `ryu.lib.packet.zebra.`**`ZebraRouterIDAdd`**
Message body class for ZEBRA_ROUTER_ID_ADD.

class `ryu.lib.packet.zebra.`**`ZebraRouterIDDelete`**
Message body class for ZEBRA_ROUTER_ID_DELETE.

class `ryu.lib.packet.zebra.`**`ZebraRouterIDUpdate`**(*family*, *prefix*)
Message body class for ZEBRA_ROUTER_ID_UPDATE.

class `ryu.lib.packet.zebra.`**`ZebraUnknownMessage`**(*buf*)
Message body class for Unknown command.

class `ryu.lib.packet.zebra.`**`ZebraVrfAdd`**(*vrf_name*)
Message body class for FRR_ZEBRA_VRF_ADD.

class `ryu.lib.packet.zebra.`**`ZebraVrfDelete`**(*vrf_name*)
Message body class for FRR_ZEBRA_VRF_DELETE.

class `ryu.lib.packet.zebra.`**`ZebraVrfUnregister`**
Message body class for ZEBRA_VRF_UNREGISTER.

`ryu.lib.packet.zebra.`**`zebra`**
alias of *`ZebraMessage`*

2.4.3 PCAP file library

Introduction

Ryu PCAP file library helps you to read/write PCAP file which file format are described in The Wireshark Wiki.

Reading PCAP file

For loading the packet data containing in PCAP files, you can use pcaplib.Reader.

class `ryu.lib.pcaplib.`**`Reader`**(*file_obj*)
PCAP file reader

Argument	Description
file_obj	File object which reading PCAP file in binary mode

Example of usage:

```python
from ryu.lib import pcaplib
from ryu.lib.packet import packet

frame_count = 0
# iterate pcaplib.Reader that yields (timestamp, packet_data)
# in the PCAP file
for ts, buf in pcaplib.Reader(open('test.pcap', 'rb')):
    frame_count += 1
    pkt = packet.Packet(buf)
    print("%d, %f, %s" % (frame_count, ts, pkt))
```

Writing PCAP file

For dumping the packet data which your RyuApp received, you can use pcaplib.Writer.

class `ryu.lib.pcaplib.`**`Writer`**(*file_obj*, *snaplen=65535*, *network=1*)
PCAP file writer

Argument	Description
file_obj	File object which writing PCAP file in binary mode
snaplen	Max length of captured packets (in octets)
network	Data link type. (e.g. 1 for Ethernet, see tcpdump.org for details)

Example of usage:

```
...
from ryu.lib import pcaplib


class SimpleSwitch13(app_manager.RyuApp):
    OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]

    def __init__(self, *args, **kwargs):
        super(SimpleSwitch13, self).__init__(*args, **kwargs)
        self.mac_to_port = {}

        # Create pcaplib.Writer instance with a file object
        # for the PCAP file
        self.pcap_writer = pcaplib.Writer(open('mypcap.pcap', 'wb'))

    ...

    @set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
    def _packet_in_handler(self, ev):
        # Dump the packet data into PCAP file
        self.pcap_writer.write_pkt(ev.msg.data)

        ...
```

2.4.4 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

2.4.5 BGP speaker library

Introduction

Ryu BGP speaker library helps you to enable your code to speak BGP protocol. The library supports IPv4, IPv4 MPLS-labeled VPN, IPv6 MPLS-labeled VPN and L2VPN EVPN address families.

Example

The following simple code creates a BGP instance with AS number 64512 and Router ID 10.0.0.1. It tries to establish a bgp session with a peer (its IP is 192.168.177.32 and the AS number is 64513). The instance advertizes some prefixes.

```
import eventlet

# BGPSpeaker needs sockets patched
eventlet.monkey_patch()

# initialize a log handler
# this is not strictly necessary but useful if you get messages like:
#    No handlers could be found for logger "ryu.lib.hub"
import logging
import sys
log = logging.getLogger()
log.addHandler(logging.StreamHandler(sys.stderr))

from ryu.services.protocols.bgp.bgpspeaker import BGPSpeaker

def dump_remote_best_path_change(event):
    print 'the best path changed:', event.remote_as, event.prefix,\
        event.nexthop, event.is_withdraw

def detect_peer_down(remote_ip, remote_as):
    print 'Peer down:', remote_ip, remote_as

if __name__ == "__main__":
    speaker = BGPSpeaker(as_number=64512, router_id='10.0.0.1',
                         best_path_change_handler=dump_remote_best_path_change,
                         peer_down_handler=detect_peer_down)

    speaker.neighbor_add('192.168.177.32', 64513)
    # uncomment the below line if the speaker needs to talk with a bmp server.
    # speaker.bmp_server_add('192.168.177.2', 11019)
    count = 1
    while True:
        eventlet.sleep(30)
        prefix = '10.20.' + str(count) + '.0/24'
        print "add a new prefix", prefix
        speaker.prefix_add(prefix)
        count += 1
        if count == 4:
            speaker.shutdown()
            break
```

2.4.6 BGP speaker library API Reference

BGPSpeaker class

class `ryu.services.protocols.bgp.bgpspeaker.`**`BGPSpeaker`**(*as_number, router_id, bgp_server_hosts=('0.0.0.0', '::'), bgp_server_port=179, refresh_stalepath_time=0, refresh_max_eor_time=0, best_path_change_handler=None, peer_down_handler=None, peer_up_handler=None, ssh_console=False, ssh_port=None, ssh_host=None, ssh_host_key=None, label_range=(100, 100000), allow_local_as_in_count=0, cluster_id=None, local_pref=100*)

`attribute_map_get`(*address, route_dist=None, route_family='ipv4'*)

This method gets in-bound filters of the specified neighbor.

`address` specifies the IP address of the neighbor.

`route_dist` specifies route distinguisher that has attribute_maps.

`route_family` specifies route family of the VRF. This parameter must be one of the following.

- RF_VPN_V4 (default) = 'ipv4'
- RF_VPN_V6 = 'ipv6'

Returns a list object containing an instance of AttributeMap

`attribute_map_set`(*address, attribute_maps, route_dist=None, route_family='ipv4'*)

This method sets attribute mapping to a neighbor. attribute mapping can be used when you want to apply attribute to BGPUpdate under specific conditions.

`address` specifies the IP address of the neighbor

`attribute_maps` specifies attribute_map list that are used before paths are advertised. All the items in the list must be an instance of AttributeMap class

`route_dist` specifies route dist in which attribute_maps are added.

`route_family` specifies route family of the VRF. This parameter must be one of the following.

- RF_VPN_V4 (default) = 'ipv4'
- RF_VPN_V6 = 'ipv6'

We can set AttributeMap to a neighbor as follows:

```
pref_filter = PrefixFilter('192.168.103.0/30',
                           PrefixFilter.POLICY_PERMIT)

attribute_map = AttributeMap([pref_filter],
                             AttributeMap.ATTR_LOCAL_PREF, 250)

speaker.attribute_map_set('192.168.50.102', [attribute_map])
```

bmp_server_add(*address*, *port*)

This method registers a new BMP (BGP monitoring Protocol) server. The BGP speaker starts to send BMP messages to the server. Currently, only one BMP server can be registered.

`address` specifies the IP address of a BMP server.

`port` specifies the listen port number of a BMP server.

bmp_server_del(*address*, *port*)

This method unregister the registered BMP server.

`address` specifies the IP address of a BMP server.

`port` specifies the listen port number of a BMP server.

evpn_prefix_add(*route_type*, *route_dist*, *esi=0*, *ethernet_tag_id=None*, *mac_addr=None*, *ip_addr=None*, *ip_prefix=None*, *gw_ip_addr=None*, *vni=None*, *next_hop=None*, *tunnel_type=None*, *pmsi_tunnel_type=None*, *redundancy_mode=None*)

This method adds a new EVPN route to be advertised.

`route_type` specifies one of the EVPN route type name. This parameter must be one of the following.

- EVPN_ETH_AUTO_DISCOVERY = 'eth_ad'
- EVPN_MAC_IP_ADV_ROUTE = 'mac_ip_adv'
- EVPN_MULTICAST_ETAG_ROUTE = 'multicast_etag'
- EVPN_ETH_SEGMENT = 'eth_seg'
- EVPN_IP_PREFIX_ROUTE = 'ip_prefix'

`route_dist` specifies a route distinguisher value.

`esi` is an value to specify the Ethernet Segment Identifier. 0 is the default and denotes a single-homed site. If you want to advertise esi other than 0, it must be set as dictionary type. If esi is dictionary type, 'type' key must be set and specifies ESI type. For the supported ESI type, see *`ryu.lib.packet.bgp.EvpnEsi`*. The remaining arguments are the same as that for the corresponding class.

`ethernet_tag_id` specifies the Ethernet Tag ID.

`mac_addr` specifies a MAC address to advertise.

`ip_addr` specifies an IPv4 or IPv6 address to advertise.

`ip_prefix` specifies an IPv4 or IPv6 prefix to advertise.

`gw_ip_addr` specifies an IPv4 or IPv6 address of gateway to advertise.

`vni` specifies an Virtual Network Identifier for VXLAN or Virtual Subnet Identifier for NVGRE. If tunnel_type is not TUNNEL_TYPE_VXLAN or TUNNEL_TYPE_NVGRE, this field is ignored.

`next_hop` specifies the next hop address for this prefix.

`tunnel_type` specifies the data plane encapsulation type to advertise. By the default, this attribute is not advertised. The supported encapsulation types are following.

- TUNNEL_TYPE_VXLAN = 'vxlan'
- TUNNEL_TYPE_NVGRE = 'nvgre

`pmsi_tunnel_type` specifies the type of the PMSI tunnel attribute used to encode the multicast tunnel identifier. This attribute is advertised only if route_type is EVPN_MULTICAST_ETAG_ROUTE and not advertised by the default. This attribute can also carry vni if tunnel_type is specified. The supported PMSI tunnel types are following.

- PMSI_TYPE_NO_TUNNEL_INFO = 0
- PMSI_TYPE_INGRESS_REP = 6

`redundancy_mode` specifies a redundancy mode type. This attribute is advertised only if route_type is EVPN_ETH_AUTO_DISCOVERY and not advertised by the default. The supported redundancy mode types are following.

- REDUNDANCY_MODE_ALL_ACTIVE = 'all_active'
- REDUNDANCY_MODE_SINGLE_ACTIVE = 'single_active'

evpn_prefix_del(*route_type*, *route_dist*, *esi=0*, *ethernet_tag_id=None*, *mac_addr=None*, *ip_addr=None*, *ip_prefix=None*)

This method deletes an advertised EVPN route.

`route_type` specifies one of the EVPN route type name.

`route_dist` specifies a route distinguisher value.

`esi` is an value to specify the Ethernet Segment Identifier.

`ethernet_tag_id` specifies the Ethernet Tag ID.

`mac_addr` specifies a MAC address to advertise.

`ip_addr` specifies an IPv4 or IPv6 address to advertise.

`ip_prefix` specifies an IPv4 or IPv6 prefix to advertise.

flowspec_prefix_add(*flowspec_family*, *rules*, *route_dist=None*, *actions=None*)

This method adds a new Flow Specification prefix to be advertised.

`flowspec_family` specifies one of the flowspec family name. This parameter must be one of the following.

- FLOWSPEC_FAMILY_IPV4 = 'ipv4fs'
- FLOWSPEC_FAMILY_IPV6 = 'ipv6fs'
- FLOWSPEC_FAMILY_VPNV4 = 'vpnv4fs'
- FLOWSPEC_FAMILY_VPNV6 = 'vpnv6fs'
- FLOWSPEC_FAMILY_L2VPN = 'l2vpnfs'

`rules` specifies NLRIs of Flow Specification as a dictionary type value. For the supported NLRI types and arguments, see *from_user()* method of the following classes.

- *ryu.lib.packet.bgp.FlowSpecIPv4NLRI*
- *ryu.lib.packet.bgp.FlowSpecIPv6NLRI*
- *ryu.lib.packet.bgp.FlowSpecVPNv4NLRI*
- *ryu.lib.packet.bgp.FlowSpecVPNv6NLRI*
- *ryu.lib.packet.bgp.FlowSpecL2VPNNLRI*

`route_dist` specifies a route distinguisher value. This parameter is required only if flowspec_family is one of the following address family.

- FLOWSPEC_FAMILY_VPNV4 = 'vpnv4fs'
- FLOWSPEC_FAMILY_VPNV6 = 'vpnv6fs'
- FLOWSPEC_FAMILY_L2VPN = 'l2vpnfs'

`actions` specifies Traffic Filtering Actions of Flow Specification as a dictionary type value. The keys are "ACTION_NAME" for each action class and values are used for the arguments to that class. For the supported "ACTION_NAME" and arguments, see the following table.

ACTION_NAME	Action Class
traffic_rate	*ryu.lib.packet.bgp.BGPFlowSpecTrafficRateCommunity*
traffic_action	*ryu.lib.packet.bgp.BGPFlowSpecTrafficActionCommunity*
redirect	*ryu.lib.packet.bgp.BGPFlowSpecRedirectCommunity*
traffic_marking	*ryu.lib.packet.bgp.BGPFlowSpecTrafficMarkingCommunity*
vlan_action	*ryu.lib.packet.bgp.BGPFlowSpecVlanActionCommunity*
tpid_action	*ryu.lib.packet.bgp.BGPFlowSpecTPIDActionCommunity*

Example(IPv4):

```
>>> speaker = BGPSpeaker(as_number=65001, router_id='172.17.0.1')
>>> speaker.neighbor_add(address='172.17.0.2',
...                      remote_as=65002,
...                      enable_ipv4fs=True)
>>> speaker.flowspec_prefix_add(
...     flowspec_family=FLOWSPEC_FAMILY_IPV4,
...     rules={
...         'dst_prefix': '10.60.1.0/24'
...     },
...     actions={
...         'traffic_marking': {
...             'dscp': 24
...         }
...     }
... )
```

Example(VPNv4):

```
>>> speaker = BGPSpeaker(as_number=65001, router_id='172.17.0.1')
>>> speaker.neighbor_add(address='172.17.0.2',
...                      remote_as=65002,
...                      enable_vpnv4fs=True)
>>> speaker.vrf_add(route_dist='65001:100',
...                 import_rts=['65001:100'],
...                 export_rts=['65001:100'],
...                 route_family=RF_VPNV4_FLOWSPEC)
>>> speaker.flowspec_prefix_add(
...     flowspec_family=FLOWSPEC_FAMILY_VPNV4,
...     route_dist='65000:100',
...     rules={
...         'dst_prefix': '10.60.1.0/24'
...     },
...     actions={
...         'traffic_marking': {
...             'dscp': 24
...         }
...     }
... )
```

`flowspec_prefix_del` (*flowspec_family*, *rules*, *route_dist=None*)

This method deletes an advertised Flow Specification route.

`flowspec_family` specifies one of the flowspec family name.

`rules` specifies NLRIs of Flow Specification as a dictionary type value.

`route_dist` specifies a route distinguisher value.

`in_filter_get`(*address*)
This method gets in-bound filters of the specified neighbor.

`address` specifies the IP address of the neighbor.

Returns a list object containing an instance of Filter sub-class

`in_filter_set`(*address*, *filters*)
This method sets in-bound filters to a neighbor.

`address` specifies the IP address of the neighbor

`filters` specifies filter list applied before advertised paths are imported to the global rib. All the items in the list must be an instance of Filter sub-class.

`neighbor_add`(*address*, *remote_as*, *enable_ipv4=True*, *enable_ipv6=False*, *enable_vpnv4=False*, *enable_vpnv6=False*, *enable_evpn=False*, *enable_ipv4fs=False*, *enable_ipv6fs=False*, *enable_vpnv4fs=False*, *enable_vpnv6fs=False*, *enable_l2vpnfs=False*, *enable_enhanced_refresh=False*, *enable_four_octet_as_number=True*, *next_hop=None*, *password=None*, *multi_exit_disc=None*, *site_of_origins=None*, *is_route_server_client=False*, *is_route_reflector_client=False*, *is_next_hop_self=False*, *local_address=None*, *local_port=None*, *local_as=None*, *connect_mode='both'*)
This method registers a new neighbor. The BGP speaker tries to establish a bgp session with the peer (accepts a connection from the peer and also tries to connect to it).

`address` specifies the IP address of the peer. It must be the string representation of an IP address. Only IPv4 is supported now.

`remote_as` specifies the AS number of the peer. It must be an integer between 1 and 65535.

`enable_ipv4` enables IPv4 address family for this neighbor.

`enable_ipv6` enables IPv6 address family for this neighbor.

`enable_vpnv4` enables VPNv4 address family for this neighbor.

`enable_vpnv6` enables VPNv6 address family for this neighbor.

`enable_evpn` enables Ethernet VPN address family for this neighbor.

`enable_ipv4fs` enables IPv4 Flow Specification address family for this neighbor.

`enable_ipv6fs` enables IPv6 Flow Specification address family for this neighbor.

`enable_vpnv4fs` enables VPNv4 Flow Specification address family for this neighbor.

`enable_vpnv6fs` enables VPNv6 Flow Specification address family for this neighbor.

`enable_l2vpnfs` enables L2VPN Flow Specification address family for this neighbor.

`enable_enhanced_refresh` enables Enhanced Route Refresh for this neighbor.

`enable_four_octet_as_number` enables Four-Octet AS Number capability for this neighbor.

`next_hop` specifies the next hop IP address. If not specified, host's ip address to access to a peer is used.

`password` is used for the MD5 authentication if it's specified. By default, the MD5 authentication is disabled.

`multi_exit_disc` specifies multi exit discriminator (MED) value as an int type value. If omitted, MED is not sent to the neighbor.

`site_of_origins` specifies site_of_origin values. This parameter must be a list of string.

`is_route_server_client` specifies whether this neighbor is a router server's client or not.

`is_route_reflector_client` specifies whether this neighbor is a router reflector's client or not.

`is_next_hop_self` specifies whether the BGP speaker announces its own ip address to iBGP neighbor or not as path's next_hop address.

`local_address` specifies Loopback interface address for iBGP peering.

`local_port` specifies source TCP port for iBGP peering.

`local_as` specifies local AS number per-peer. If omitted, the AS number of BGPSpeaker instance is used.

`connect_mode` specifies how to connect to this neighbor. This parameter must be one of the following.

- CONNECT_MODE_ACTIVE = 'active'
- CONNECT_MODE_PASSIVE = 'passive'
- CONNECT_MODE_BOTH (default) = 'both'

`neighbor_del`(*address*)

This method unregister the registered neighbor. If a session with the peer exists, the session will be closed.

`address` specifies the IP address of the peer. It must be the string representation of an IP address.

`neighbor_get`(*route_type*, *address*, *format='json'*)

This method returns the BGP adj-RIB-in/adj-RIB-out information in a json format.

`route_type` This parameter is necessary for only received-routes and sent-routes.

- received-routes : paths received and not withdrawn by given peer
- sent-routes : paths sent and not withdrawn to given peer

`address` specifies the IP address of the peer. It must be the string representation of an IP address.

`format` specifies the format of the response. This parameter must be one of the following.

- 'json' (default)
- 'cli'

`neighbor_reset`(*address*)

This method reset the registered neighbor.

`address` specifies the IP address of the peer. It must be the string representation of an IP address.

`neighbor_state_get`(*address=None*, *format='json'*)

This method returns the state of peer(s) in a json format.

`address` specifies the address of a peer. If not given, the state of all the peers return.

`format` specifies the format of the response. This parameter must be one of the following.

- 'json' (default)
- 'cli'

`neighbor_update`(*address*, *conf_type*, *conf_value*)

This method changes the neighbor configuration.

`address` specifies the IP address of the peer.

`conf_type` specifies configuration type which you want to change. Currently ryu.services.protocols.bgp.bgpspeaker.MULTI_EXIT_DISC can be specified.

`conf_value` specifies value for the configuration type.

neighbors_get (*format='json'*)
This method returns a list of the BGP neighbors.

`format` specifies the format of the response. This parameter must be one of the following.

- 'json' (default)
- 'cli'

out_filter_get (*address*)
This method gets out-filter setting from the specified neighbor.

`address` specifies the IP address of the peer.

Returns a list object containing an instance of Filter sub-class

out_filter_set (*address, filters*)
This method sets out-filter to neighbor.

`address` specifies the IP address of the peer.

`filters` specifies a filter list to filter the path advertisement. The contents must be an instance of Filter sub-class

If you want to define out-filter that send only a particular prefix to neighbor, filters can be created as follows:

```
p = PrefixFilter('10.5.111.0/24',
                 policy=PrefixFilter.POLICY_PERMIT)

all = PrefixFilter('0.0.0.0/0',
                   policy=PrefixFilter.POLICY_DENY)

pList = [p, all]

self.bgpspeaker.out_filter_set(neighbor_address, pList)
```

Note: out-filter evaluates paths in the order of Filter in the pList.

prefix_add (*prefix, next_hop=None, route_dist=None*)
This method adds a new prefix to be advertised.

`prefix` must be the string representation of an IP network (e.g., 10.1.1.0/24).

`next_hop` specifies the next hop address for this prefix. This parameter is necessary for only VPNv4 and VPNv6 address families.

`route_dist` specifies a route distinguisher value. This parameter is necessary for only VPNv4 and VPNv6 address families.

prefix_del (*prefix, route_dist=None*)
This method deletes a advertised prefix.

`prefix` must be the string representation of an IP network.

`route_dist` specifies a route distinguisher value.

rib_get (*family='all', format='json'*)
This method returns the BGP routing information in a json format. This will be improved soon.

`family` specifies the address family of the RIB (e.g. 'ipv4').

`format` specifies the format of the response. This parameter must be one of the following.

- 'json' (default)
- 'cli'

shutdown()
Shutdown BGP speaker

vrf_add(*route_dist*, *import_rts*, *export_rts*, *site_of_origins=None*, *route_family='ipv4'*, *multi_exit_disc=None*)
This method adds a new vrf used for VPN.

`route_dist` specifies a route distinguisher value.

`import_rts` specifies a list of route targets to be imported.

`export_rts` specifies a list of route targets to be exported.

`site_of_origins` specifies site_of_origin values. This parameter must be a list of string.

`route_family` specifies route family of the VRF. This parameter must be one of the following.

- RF_VPN_V4 (default) = 'ipv4'
- RF_VPN_V6 = 'ipv6'
- RF_L2_EVPN = 'evpn'
- RF_VPNV4_FLOWSPEC = 'ipv4fs'
- RF_VPNV6_FLOWSPEC = 'ipv6fs'
- RF_L2VPN_FLOWSPEC = 'l2vpnfs'

`multi_exit_disc` specifies multi exit discriminator (MED) value. It must be an integer.

vrf_del(*route_dist*)
This method deletes the existing vrf.

`route_dist` specifies a route distinguisher value.

vrfs_get(*subcommand='routes'*, *route_dist=None*, *route_family='all'*, *format='json'*)
This method returns the existing vrfs.

`subcommand` specifies one of the following.

- 'routes': shows routes present for vrf
- 'summary': shows configuration and summary of vrf

`route_dist` specifies a route distinguisher value. If route_family is not 'all', this value must be specified.

`route_family` specifies route family of the VRF. This parameter must be one of the following.

- RF_VPN_V4 = 'ipv4'
- RF_VPN_V6 = 'ipv6'
- RF_L2_EVPN = 'evpn'
- 'all' (default)

`format` specifies the format of the response. This parameter must be one of the following.

- 'json' (default)
- 'cli'

class `ryu.services.protocols.bgp.bgpspeaker.`**`EventPrefix`**(*path*, *is_withdraw*)

Used to pass an update on any best remote path to best_path_change_handler.

Attribute	Description
remote_as	The AS number of a peer that caused this change
route_dist	None in the case of IPv4 or IPv6 family
prefix	A prefix was changed
nexthop	The nexthop of the changed prefix
label	MPLS label for VPNv4, VPNv6 or EVPN prefix
path	An instance of `info_base.base.Path` subclass
is_withdraw	True if this prefix has gone otherwise False

class `ryu.services.protocols.bgp.info_base.base.`**`PrefixFilter`**(*prefix*, *policy*, *ge=None*, *le=None*)

Used to specify a prefix for filter.

We can create PrefixFilter object as follows:

```
prefix_filter = PrefixFilter('10.5.111.0/24',
                             policy=PrefixFilter.POLICY_PERMIT)
```

Attribute	Description
prefix	A prefix used for this filter
policy	One of the following values. PrefixFilter.POLICY.PERMIT PrefixFilter.POLICY_DENY
ge	Prefix length that will be applied to this filter. ge means greater than or equal.
le	Prefix length that will be applied to this filter. le means less than or equal.

For example, when PrefixFilter object is created as follows:

```
p = PrefixFilter('10.5.111.0/24',
                 policy=PrefixFilter.POLICY_DENY,
                 ge=26, le=28)
```

Prefixes which match 10.5.111.0/24 and its length matches from 26 to 28 will be filtered. When this filter is used as an out-filter, it will stop sending the path to neighbor because of POLICY_DENY. When this filter is used as in-filter, it will stop importing the path to the global rib because of POLICY_DENY. If you specify POLICY_PERMIT, the path is sent to neighbor or imported to the global rib.

If you don't want to send prefixes 10.5.111.64/26 and 10.5.111.32/27 and 10.5.111.16/28, and allow to send other 10.5.111.0's prefixes, you can do it by specifying as follows:

```
p = PrefixFilter('10.5.111.0/24',
                 policy=PrefixFilter.POLICY_DENY,
                 ge=26, le=28).
```

`clone`()

This method clones PrefixFilter object.

Returns PrefixFilter object that has the same values with the original one.

`evaluate`(*path*)

This method evaluates the prefix.

Returns this object's policy and the result of matching. If the specified prefix matches this object's prefix and ge and le condition, this method returns True as the matching result.

`path` specifies the path that has prefix.

class `ryu.services.protocols.bgp.info_base.base.`**`ASPathFilter`**(*as_number*, *policy*)

Used to specify a prefix for AS_PATH attribute.

We can create ASPathFilter object as follows:

```
as_path_filter = ASPathFilter(65000,policy=ASPathFilter.TOP)
```

Attribute	Description
as_number	A AS number used for this filter
policy	One of the following values. ASPathFilter.POLICY_TOP ASPathFilter.POLICY_END ASPathFilter.POLICY_INCLUDE ASPathFilter.POLICY_NOT_INCLUDE

Meaning of each policy is as follows:

Policy	Description
POLICY_TOP	Filter checks if the specified AS number is at the top of AS_PATH attribute.
POLICY_END	Filter checks is the specified AS number is at the last of AS_PATH attribute.
POLICY_INCLUDE	Filter checks if specified AS number exists in AS_PATH attribute.
POLICY_NOT_INCLUDE	Opposite to POLICY_INCLUDE.

`clone`()

This method clones ASPathFilter object.

Returns ASPathFilter object that has the same values with the original one.

`evaluate`(*path*)

This method evaluates as_path list.

Returns this object's policy and the result of matching. If the specified AS number matches this object's AS number according to the policy, this method returns True as the matching result.

`path` specifies the path.

class `ryu.services.protocols.bgp.info_base.base.`**`AttributeMap`**(*filters*, *attr_type*, *attr_value*)

This class is used to specify an attribute to add if the path matches filters. We can create AttributeMap object as follows:

```
pref_filter = PrefixFilter('192.168.103.0/30',
                           PrefixFilter.POLICY_PERMIT)

attribute_map = AttributeMap([pref_filter],
                             AttributeMap.ATTR_LOCAL_PREF, 250)

speaker.attribute_map_set('192.168.50.102', [attribute_map])
```

AttributeMap.ATTR_LOCAL_PREF means that 250 is set as a local preference value if nlri in the path matches pref_filter.

ASPathFilter is also available as a filter. ASPathFilter checks if AS_PATH attribute in the path matches AS number in the filter.

Attribute	Description
filters	A list of filter. Each object should be a Filter class or its sub-class
attr_type	A type of attribute to map on filters. Currently AttributeMap.ATTR_LOCAL_PREF is available.
attr_value	A attribute value

clone()

This method clones AttributeMap object.

Returns AttributeMap object that has the same values with the original one.

evaluate(*path*)

This method evaluates attributes of the path.

Returns the cause and result of matching. Both cause and result are returned from filters that this object contains.

`path` specifies the path.

2.4.7 MRT file library

Introduction

Ryu MRT file library helps you to read/write MRT (Multi-Threaded Routing Toolkit) Routing Information Export Format [RFC6396].

Reading MRT file

For loading the routing information contained in MRT files, you can use mrtlib.Reader.

class ryu.lib.mrtlib.**Reader**(*f*)

MRT format file reader.

Argument	Description
f	File object which reading MRT format file in binary mode.

Example of Usage:

```
import bz2
from ryu.lib import mrtlib

count = 0
for record in mrtlib.Reader(
        bz2.BZ2File('rib.YYYYMMDD.hhmm.bz2', 'rb')):
    print("%d, %s" % (count, record))
    count += 1
```

Writing MRT file

For dumping the routing information which your RyuApp generated, you can use mrtlib.Writer.

class ryu.lib.mrtlib.**Writer**(*f*)

MRT format file writer.

Argument	Description
f	File object which writing MRT format file in binary mode.

Example of usage:

```
import bz2
import time
from ryu.lib import mrtlib
from ryu.lib.packet import bgp

mrt_writer = mrtlib.Writer(
    bz2.BZ2File('rib.YYYYMMDD.hhmm.bz2', 'wb'))

prefix = bgp.IPAddrPrefix(24, '10.0.0.0')

rib_entry = mrtlib.MrtRibEntry(
    peer_index=0,
    originated_time=int(time.time()),
    bgp_attributes=[bgp.BGPPathAttributeOrigin(0)])

message = mrtlib.TableDump2RibIPv4UnicastMrtMessage(
    seq_num=0,
    prefix=prefix,
    rib_entries=[rib_entry])

record = mrtlib.TableDump2MrtRecord(
    message=message)

mrt_writer.write(record)
```

2.4.8 OVSDB Manager library

Path: `ryu.services.protocols.ovsdb`

Introduction

Ryu OVSDB Manager library allows your code to interact with devices speaking the OVSDB protocol. This enables your code to perform remote management of the devices and react to topology changes on them.

Please note this library will spawn a server listening on the port 6640 (the IANA registered for OVSDB protocol), but does not initiate connections from controller side. Then, to make your devices connect to Ryu, you need to tell the controller IP address and port to your devices.

```
# Show current configuration
$ ovs-vsctl get-manager

# Set manager (controller) address
$ ovs-vsctl set-manager "tcp:127.0.0.1:6640"

# If you want to specify IPv6 address, wrap ip with brackets
$ ovs-vsctl set-manager "tcp:[::1]:6640"
```

Also this library identifies the devices by “system-id” which should be unique, persistent identifier among all devices connecting to a single controller. Please make sure “system-id” is configured before connecting.

```
# Show current configuration
$ ovs-vsctl get Open_vSwitch . external_ids:system-id

# Set system-id manually
$ ovs-vsctl set Open_vSwitch . external_ids:system-id=<SYSTEM-ID>
```

Example

The following logs all new OVSDB connections in "handle_new_ovsdb_connection" and also provides the API "create_port" for creating a port on a bridge.

```
import uuid

from ryu.base import app_manager
from ryu.controller.handler import set_ev_cls
from ryu.services.protocols.ovsdb import api as ovsdb
from ryu.services.protocols.ovsdb import event as ovsdb_event


class MyApp(app_manager.RyuApp):
    @set_ev_cls(ovsdb_event.EventNewOVSDBConnection)
    def handle_new_ovsdb_connection(self, ev):
        system_id = ev.system_id
        address = ev.client.address
        self.logger.info(
            'New OVSDB connection from system-id=%s, address=%s',
            system_id, address)

        # Example: If device has bridge "s1", add port "s1-eth99"
        if ovsdb.bridge_exists(self, system_id, "s1"):
            self.create_port(system_id, "s1", "s1-eth99")

    def create_port(self, system_id, bridge_name, name):
        new_iface_uuid = uuid.uuid4()
        new_port_uuid = uuid.uuid4()

        bridge = ovsdb.row_by_name(self, system_id, bridge_name)

        def _create_port(tables, insert):
            iface = insert(tables['Interface'], new_iface_uuid)
            iface.name = name
            iface.type = 'internal'

            port = insert(tables['Port'], new_port_uuid)
            port.name = name
            port.interfaces = [iface]

            bridge.ports = bridge.ports + [port]

            return new_port_uuid, new_iface_uuid

        req = ovsdb_event.EventModifyRequest(system_id, _create_port)
        rep = self.send_request(req)

        if rep.status != 'success':
            self.logger.error('Error creating port %s on bridge %s: %s',
                              name, bridge, rep.status)
            return None

        return rep.insert_uuids[new_port_uuid]
```

2.5 OpenFlow protocol API Reference

2.5.1 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',
    ]
}
```

`from_jsondict`(*dict_*, *decode_string=<function b64decode>*, ***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	Descrption
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.

`to_jsondict`(*encode_string=<function b64encode>*)

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.

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.

2.5.2 OpenFlow v1.0 Messages and Structures

Controller-to-Switch Messages

Handshake

class `ryu.ofproto.ofproto_v1_0_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_0_parser.`**`OFPSwitchFeatures`**(*datapath*, *datapath_id=None*, *n_buffers=None*, *n_tables=None*, *capabilities=None*, *actions=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.

Attribute	Description
datapath_id	Datapath unique ID.
n_buffers	Max packets buffered at once.
n_tables	Number of tables supported by datapath.
capabilities	Bitmap of capabilities flag. OFPC_FLOW_STATS OFPC_TABLE_STATS OFPC_PORT_STATS OFPC_STP OFPC_RESERVED OFPC_IP_REASM OFPC_QUEUE_STATS OFPC_ARP_MATCH_IP
actions	Bitmap of supported OFPAT_*.
ports	List of `OFPPhyPort` instances.

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": {
      "actions": 2115,
      "capabilities": 169,
      "datapath_id": 1095522080376,
      "n_buffers": 0,
      "n_tables": 255,
      "ports": {
         "6": {
            "OFPPhyPort": {
               "advertised": 640,
               "config": 0,
               "curr": 648,
               "hw_addr": "f2:0b:a4:7d:f8:ea",
               "name": "Port6",
               "peer": 648,
               "port_no": 6,
               "state": 2,
               "supported": 648
            }
         },
         "7": {
            "OFPPhyPort": {
               "advertised": 640,
               "config": 0,
               "curr": 648,
               "hw_addr": "f2:0b:a4:d0:3f:70",
               "name": "Port7",
               "peer": 648,
               "port_no": 7,
               "state": 16,
               "supported": 648
            }
         }
      }
   }
}
```

Switch Configuration

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPSetConfig`**(*datapath*, *flags=None*, *miss_send_len=None*)

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:

```python
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)
```

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPGetConfigRequest`**(*datapath*)

Get config request message

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

Example:

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

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

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPGetConfigReply`**(*datapath*)

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:

```python
@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)
```

Modify State Messages

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPFlowMod`**(*datapath, match=None, cookie=0, command=0, idle_timeout=0, hard_timeout=0, priority=32768, buffer_id=4294967295, out_port=65535, flags=0, actions=None*)

Modify Flow entry message

The controller sends this message to modify the flow table.

Attribute	Description
match	Instance of `OFPMatch`.
cookie	Opaque controller-issued identifier.
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 0xffffffff). Not meaningful for OFPFC_DELETE*.
out_port	For OFPFC_DELETE* commands, require matching entries to include this as an output port. A value of OFPP_NONE indicates no restriction.
flags	One of the following values. OFPFF_SEND_FLOW_REM OFPFF_CHECK_OVERLAP OFPFF_EMERG
actions	List of `OFPAction*` instance.

Example:

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

    match = ofp_parser.OFPMatch(in_port=1)
    cookie = 0
    command = ofp.OFPFC_ADD
    idle_timeout = hard_timeout = 0
    priority = 32768
    buffer_id = 0xffffffff
    out_port = ofproto.OFPP_NONE
    flags = 0
    actions = [ofp_parser.OFPActionOutput(ofp.OFPP_NORMAL, 0)]
    req = ofp_parser.OFPFlowMod(
        datapath, match, cookie, command, idle_timeout, hard_timeout,
        priority, buffer_id, out_port, flags, actions)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPFlowMod": {
      "actions": [
         {
            "OFPActionOutput": {
               "max_len": 65535,
               "port": 6
            }
         }
      ],
      "buffer_id": 65535,
      "command": 0,
      "cookie": 0,
      "flags": 0,
      "hard_timeout": 0,
      "idle_timeout": 0,
      "match": {
         "OFPMatch": {
            "dl_dst": "f2:0b:a4:7d:f8:ea",
            "dl_src": "00:00:00:00:00:00",
            "dl_type": 0,
            "dl_vlan": 0,
            "dl_vlan_pcp": 0,
            "in_port": 0,
            "nw_dst": "0.0.0.0",
            "nw_proto": 0,
            "nw_src": "0.0.0.0",
            "nw_tos": 0,
            "tp_dst": 0,
            "tp_src": 0,
            "wildcards": 4194295
         }
      },
      "out_port": 65532,
      "priority": 123
   }
}
```

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPPortMod`**(*datapath*, *port_no=0*, *hw_addr='00:00:00:00:00:00'*, *config=0*, *mask=0*, *advertise=0*)

Port modification message

The controller send 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 `OFPPhyPort` of `OFPSwitchFeatures`.
config	Bitmap of configuration flags. OFPPC_PORT_DOWN OFPPC_NO_STP OFPPC_NO_RECV OFPPC_NO_RECV_STP OFPPC_NO_FLOOD 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_COPPER OFPPF_FIBER OFPPF_AUTONEG OFPPF_PAUSE OFPPF_PAUSE_ASYM

Example:

```python
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)
```

Queue Configuration Messages

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPQueueGetConfigRequest`**(*datapath*, *port*)

Queue configuration request message

Attribute	Description
port	Port to be queried. Should refer to a valid physical port (i.e. < OFPP_MAX).

Example:

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

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

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPQueueGetConfigReply`**(*datapath*)

Queue configuration reply message

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

Attribute	Description
port	Port to be 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)
```

Read State Messages

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPDescStatsRequest`**(*datapath*, *flags*)

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)
```

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPDescStatsReply`**(*datapath*)

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.EventOFPDescStatsReply, MAIN_DISPATCHER)
def desc_stats_reply_handler(self, ev):
    msg = ev.msg
    ofp = msg.datapath.ofproto
    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)
```

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPFlowStatsRequest`**(*datapath*, *flags*, *match*, *table_id*, *out_port*)

Individual flow statistics request message

The controller uses this message to query individual flow statistics.

Attribute	Description
flags	Zero (none yet defined in the spec).
match	Instance of `OFPMatch`.
table_id	ID of table to read (from ofp_table_stats), 0xff for all tables or 0xfe for emergency.
out_port	Require matching entries to include this as an output port. A value of OFPP_NONE indicates no restriction.

Example:

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

    match = ofp_parser.OFPMatch(in_port=1)
    table_id = 0xff
    out_port = ofp.OFPP_NONE
    req = ofp_parser.OFPFlowStatsRequest(
        datapath, 0, match, table_id, out_port)

    datapath.send_msg(req)
```

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPFlowStatsReply`**(*datapath*)

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.
match	Instance of `OFPMatch`.
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. Only meaningful when this is not an exact-match 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.
actions	List of `OFPAction*` instance

Example:

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

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

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPAggregateStatsRequest`**(*datapath*, *flags*, *match*, *table_id*, *out_port*)

Aggregate flow statistics request message

The controller uses this message to query aggregate flow statictics.

Attribute	Description
flags	Zero (none yet defined in the spec).
match	Fields to match.
table_id	ID of table to read (from ofp_table_stats) 0xff for all tables or 0xfe for emergency.
out_port	Require matching entries to include this as an output port. A value of OFPP_NONE indicates no restriction.

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, match, 0xff, ofp.OFPP_NONE)

    datapath.send_msg(req)
```

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPAggregateStatsReply`**(*datapath*)

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.EventOFPAggregateStatsReply, MAIN_DISPATCHER)
def aggregate_stats_reply_handler(self, ev):
    msg = ev.msg
    ofp = msg.datapath.ofproto
    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)
```

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPTableStatsRequest`**(*datapath, flags*)

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_parser = datapath.ofproto_parser

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

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPTableStatsReply`**(*datapath*)

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.
wildcards	Bitmap of OFPFW_* wildcards that are supported by the table.
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.EventOFPTableStatsReply, MAIN_DISPATCHER)
def stats_reply_handler(self, ev):
    msg = ev.msg
    ofp = msg.datapath.ofproto
    body = ev.msg.body

    tables = []
    for stat in body:
        tables.append('table_id=%d name=%s wildcards=0x%02x '
                      'max_entries=%d active_count=%d '
                      'lookup_count=%d matched_count=%d' %
                      (stat.table_id, stat.name, stat.wildcards,
                       stat.max_entries, stat.active_count,
                       stat.lookup_count, stat.matched_count))
    self.logger.debug('TableStats: %s', tables)
```

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPPortStatsRequest`**(*datapath*, *flags*, *port_no*)

Port statistics request message

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

Attribute	Description
flags	Zero (none yet defined in the spec).
port_no	Port number to read (OFPP_NONE 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)
```

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPPortStatsReply`**(*datapath*)

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.EventOFPPortStatsReply, MAIN_DISPATCHER)
def port_stats_reply_handler(self, ev):
    msg = ev.msg
    ofp = msg.datapath.ofproto
    body = ev.msg.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)
```

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPQueueStatsRequest`**(*datapath*, *flags*, *port_no*, *queue_id*)

Queue statistics request message

The controller uses this message to query queue statictics.

Attribute	Description
flags	Zero (none yet defined in the spec)
port_no	Port number to read (All ports if OFPT_ALL).
queue_id	ID of queue to read (All queues if OFPQ_ALL).

Example:

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

    req = ofp_parser.OFPQueueStatsRequest(datapath, 0, ofp.OFPT_ALL,
                                          ofp.OFPQ_ALL)
    datapath.send_msg(req)
```

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPQueueStatsReply`**(*datapath*)

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.EventOFPQueueStatsReply, MAIN_DISPATCHER)
def stats_reply_handler(self, ev):
    msg = ev.msg
    ofp = msg.datapath.ofproto
    body = ev.msg.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)
```

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPVendorStatsRequest`**(*datapath*, *flags*, *vendor*, *specific_data=None*)

Vendor statistics request message

The controller uses this message to query vendor-specific information of a switch.

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPVendorStatsReply`**(*datapath*)

Vendor statistics reply message

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

Send Packet Message

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPPacketOut`**(*datapath*, *buffer_id=None*, *in_port=None*, *actions=None*, *data=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 (0xffffffff if none).
in_port	Packet's input port (OFPP_NONE if none).
actions	ist of `OFPAction*` instance.
data	Packet data of a binary type value or an instances of packet.Packet.

Example:

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

    buffer_id = 0xffffffff
    in_port = ofp.OFPP_NONE
    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": {
                  "max_len": 65535,
                  "port": 65532
               }
            }
         ],
         "buffer_id": 4294967295,
         "data":
↪"8guk0D9w8gukffjqCABFAABU+BoAAP8Br4sKAAABCgAAAggAAgj3YAAAMdYCAAAAAACrjS0xAAAAABAREhMUFRYXGBkaG
↪",
         "in_port": 65533
   }
}
```

Barrier Message

class `ryu.ofproto.ofproto_v1_0_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)
```

class `ryu.ofproto.ofproto_v1_0_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')
```

Asynchronous Messages

Packet-In Message

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPPacketIn`**(*datapath*, *buffer_id=None*, *total_len=None*, *in_port=None*, *reason=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.
in_port	Port on which frame was received.
reason	Reason packet is being sent. OFPR_NO_MATCH OFPR_ACTION OFPR_INVALID_TTL
data	Ethernet frame.

Example:

```
@set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
def packet_in_handler(self, ev):
    msg = ev.msg
    dp = msg.datapath
    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 in_port=%d, '
                      'reason=%s data=%s',
                      msg.buffer_id, msg.total_len, msg.in_port,
                      reason, utils.hex_array(msg.data))
```

JSON Example:

```
{
   "OFPPacketIn": {
      "buffer_id": 2,
      "data": "////////8gukffjqCAYAAQgABgQAAfILpH346goAAAEAAAAAAAAKAAAD",
      "in_port": 99,
      "reason": 1,
      "total_len": 42
   }
}
```

Flow Removed Message

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPFlowRemoved`**`(`*datapath*`)`

Flow removed message

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

Attribute	Description
match	Instance of `OFPMatch`.
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
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.
packet_count	Number of packets that was associated with the flow.
byte_count	Number of bytes that was associated with the flow.

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: '
                      'match=%s cookie=%d priority=%d reason=%s '
                      'duration_sec=%d duration_nsec=%d '
                      'idle_timeout=%d packet_count=%d byte_count=%d',
                      msg.match, msg.cookie, msg.priority, reason,
                      msg.duration_sec, msg.duration_nsec,
                      msg.idle_timeout, msg.packet_count,
                      msg.byte_count)
```

Port Status Message

class `ryu.ofproto.ofproto_v1_0_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 `OFPPhyPort`

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)
```

Error Message

class `ryu.ofproto.ofproto_v1_0_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_FLOW_MOD_FAILED	OFPFMFC_*
OFPET_PORT_MOD_FAILED	OFPPMFC_*
OFPET_QUEUE_OP_FAILED	OFPQOFC_*

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))
```

Symmetric Messages

Hello

class `ryu.ofproto.ofproto_v1_0_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.

Echo Request

class `ryu.ofproto.ofproto_v1_0_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_parser = datapath.ofproto_parser

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

Echo Reply

class `ryu.ofproto.ofproto_v1_0_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:

```
@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))
```

Vendor

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPVendor`**(*datapath*)

Vendor message

The controller send this message to send the vendor-specific information to a switch.

Port Structures

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPPhyPort`**

Description of a port

Attribute	Description
port_no	Port number and it uniquely identifies a port within a switch.
hw_addr	MAC address for the port.
name	Null-terminated string containing a human-readable name for the interface.
config	Bitmap of port configration flags. OFPPC_PORT_DOWN OFPPC_NO_STP OFPPC_NO_RECV OFPPC_NO_RECV_STP OFPPC_NO_FLOOD OFPPC_NO_FWD OFPPC_NO_PACKET_IN
state	Bitmap of port state flags. OFPPS_LINK_DOWN OFPPS_STP_LISTEN OFPPS_STP_LEARN OFPPS_STP_FORWARD OFPPS_STP_BLOCK OFPPS_STP_MASK
curr	Current features.
advertised	Features being advertised by the port.
supported	Features supported by the port.
peer	Features advertised by peer.

Flow Match Structure

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPMatch`**(*wildcards=None, in_port=None, dl_src=None, dl_dst=None, dl_vlan=None, dl_vlan_pcp=None, dl_type=None, nw_tos=None, nw_proto=None, nw_src=None, nw_dst=None, tp_src=None, tp_dst=None, nw_src_mask=32, nw_dst_mask=32*)

Flow Match Structure

This class is implementation of the flow match structure having compose/query API.

Attribute	Description
wildcards	Wildcard fields.
(match fields)	For the available match fields, please refer to the following.

Argument	Value	Description
in_port	Integer 16bit	Switch input port.
dl_src	MAC address	Ethernet source address.
dl_dst	MAC address	Ethernet destination address.
dl_vlan	Integer 16bit	Input VLAN id.
dl_vlan_pcp	Integer 8bit	Input VLAN priority.
dl_type	Integer 16bit	Ethernet frame type.
nw_tos	Integer 8bit	IP ToS (actually DSCP field, 6 bits).
nw_proto	Integer 8bit	IP protocol or lower 8 bits of ARP opcode.
nw_src	IPv4 address	IP source address.
nw_dst	IPv4 address	IP destination address.
tp_src	Integer 16bit	TCP/UDP source port.
tp_dst	Integer 16bit	TCP/UDP destination port.
nw_src_mask	Integer 8bit	IP source address mask specified as IPv4 address prefix.
nw_dst_mask	Integer 8bit	IP destination address mask specified as IPv4 address prefix.

Example:

```
>>> # compose
>>> match = parser.OFPMatch(
...     in_port=1,
...     dl_type=0x0800,
...     dl_src='aa:bb:cc:dd:ee:ff',
...     nw_src='192.168.0.1')
>>> # query
>>> if 'nw_src' in match:
...     print match['nw_src']
...
'192.168.0.1'
```

Action Structures

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPActionHeader`**(*type_, len_*)

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPAction`**

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPActionOutput`**(*port, max_len=65509*)

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.

Note:: The reason of this magic number (0xffe5) is because there is no good constant in of1.0. The same value as OFPCML_MAX of of1.2 and of1.3 is used.

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPActionVlanVid`**`(`*vlan_vid*`)`

Set the 802.1q VLAN id action

This action indicates the 802.1q VLAN id to be set.

Attribute	Description
vlan_vid	VLAN id.

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPActionVlanPcp`**`(`*vlan_pcp*`)`

Set the 802.1q priority action

This action indicates the 802.1q priority to be set.

Attribute	Description
vlan_pcp	VLAN priority.

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPActionStripVlan`**

Strip the 802.1q header action

This action indicates the 802.1q priority to be striped.

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPActionDlAddr`**`(`*dl_addr*`)`

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPActionSetDlSrc`**`(`*dl_addr*`)`

Set the ethernet source address action

This action indicates the ethernet source address to be set.

Attribute	Description
dl_addr	Ethernet address.

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPActionSetDlDst`**`(`*dl_addr*`)`

Set the ethernet destination address action

This action indicates the ethernet destination address to be set.

Attribute	Description
dl_addr	Ethernet address.

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPActionNwAddr`**`(`*nw_addr*`)`

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPActionSetNwSrc`**`(`*nw_addr*`)`

Set the IP source address action

This action indicates the IP source address to be set.

Attribute	Description
nw_addr	IP address.

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPActionSetNwDst`**`(`*nw_addr*`)`

Set the IP destination address action

This action indicates the IP destination address to be set.

Attribute	Description
nw_addr	IP address.

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPActionSetNwTos`**(*tos*)
Set the IP ToS action

This action indicates the IP ToS (DSCP field, 6 bits) to be set.

Attribute	Description
tos	IP ToS (DSCP field, 6 bits).

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPActionTpPort`**(*tp*)

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPActionSetTpSrc`**(*tp*)
Set the TCP/UDP source port action

This action indicates the TCP/UDP source port to be set.

Attribute	Description
tp	TCP/UDP port.

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPActionSetTpDst`**(*tp*)
Set the TCP/UDP destination port action

This action indicates the TCP/UDP destination port to be set.

Attribute	Description
tp	TCP/UDP port.

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPActionEnqueue`**(*port*, *queue_id*)
Output to queue action

This action indicates send packets to given queue on port.

Attribute	Description
port	Port that queue belongs.
queue_id	Where to enqueue the packets.

class `ryu.ofproto.ofproto_v1_0_parser.`**`OFPActionVendor`**(*vendor=None*)
Vendor action

This action is an extensible action for the vendor.

2.5.3 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=None*)

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:

```
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
                           }
                        },
                        "len": 16,
                        "type": 25
                     }
                  },
                  {
                     "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": 16,
                        "type": 25
                     }
                  }
               ],
               "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=0*, *type_=0*, *group_id=0*, *buckets=None*)

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_2_parser.`**`OFPPortMod`**(*datapath*, *port_no=0*, *hw_addr='00:00:00:00:00:00'*, *config=0*, *mask=0*, *advertise=0*)

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:

```python
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:

```python
@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:

```json
{
   "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:

```python
def send_table_stats_request(self, datapath):
    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
meta-data_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*, *length=None*)

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_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_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 of a binary type value or an instances of packet.Packet.

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+BoAAP8Br4sKAAABCgAAggAAgj3YAAAMdYCAAAAAACrjS0xAAAAABAREhMUFRYXGBkaG
↪",
      "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
    dp = msg.datapath
    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:

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

```json
{
   "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:

```python
@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*, ***kwargs*)

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

If `type == OFPET_EXPERIMENTER`, this message has also the following attributes.

Attribute	Description
exp_type	Experimenter defined type
experimenter	Experimenter ID

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
   }
}
```

```
{
   "OFPErrorMsg": {
      "code": null,
      "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_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_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
   }
}
```

Port Structures

class ryu.ofproto.ofproto_v1_2_parser.**OFPPort**

Description of a port

Attribute	Description
port_no	Port number and it uniquely identifies a port within a switch.
hw_addr	MAC address for the port.
name	Null-terminated string containing a human-readable name for the interface.
config	Bitmap of port configration flags. OFPPC_PORT_DOWN OFPPC_NO_RECV OFPPC_NO_FWD OFPPC_NO_PACKET_IN
state	Bitmap of port state flags. OFPPS_LINK_DOWN OFPPS_BLOCKED OFPPS_LIVE
curr	Current features.
advertised	Features being advertised by the port.
supported	Features supported by the port.
peer	Features advertised by peer.
curr_speed	Current port bitrate in kbps.
max_speed	Max port bitrate in kbps.

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
ipv4_dst	IPv4 address	IPv4 destination address
tcp_src	Integer 16bit	TCP source port

Continued on next page

Table 2.1 – continued from previous page

Argument	Value	Description
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
pbb_uca	Integer 8bit	PBB UCA header field (EXT-256 Old version of ONF Extension)
tcp_flags	Integer 16bit	TCP flags (EXT-109 ONF Extension)
actset_output	Integer 32bit	Output port from action set metadata (EXT-233 ONF Extension)

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::')
```

Note: For the list of the supported Nicira experimenter matches, please refer to *ryu.ofproto.nx_match*.

Note: For VLAN id match field, special values are defined in OpenFlow Spec.

1. Packets with and without a VLAN tag

   - Example:

```
match = parser.OFPMatch()
```

- Packet Matching

non-VLAN-tagged	MATCH
VLAN-tagged(vlan_id=3)	MATCH
VLAN-tagged(vlan_id=5)	MATCH

2. Only packets without a VLAN tag

- Example:

```
match = parser.OFPMatch(vlan_vid=0x0000)
```

- Packet Matching

non-VLAN-tagged	MATCH
VLAN-tagged(vlan_id=3)	x
VLAN-tagged(vlan_id=5)	x

3. Only packets with a VLAN tag regardless of its value

- Example:

```
match = parser.OFPMatch(vlan_vid=(0x1000, 0x1000))
```

- Packet Matching

non-VLAN-tagged	x
VLAN-tagged(vlan_id=3)	MATCH
VLAN-tagged(vlan_id=5)	MATCH

4. Only packets with VLAN tag and VID equal

- Example:

```
match = parser.OFPMatch(vlan_vid=(0x1000 | 3))
```

- Packet Matching

non-VLAN-tagged	x
VLAN-tagged(vlan_id=3)	MATCH
VLAN-tagged(vlan_id=5)	x

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=0*, *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.

The set of keywords available for this is same as OFPMatch.

Example:

```
set_field = OFPActionSetField(eth_src="00:00:00:00:00:00")
```

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

Note: For the list of the supported Nicira experimenter actions, please refer to *ryu.ofproto.nx_actions*.

2.5.4 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	Bitmap of the following flags. OFPC_FRAG_NORMAL OFPC_FRAG_DROP OFPC_FRAG_REASM
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	Bitmap of the following 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
    flags = []

    if msg.flags & ofp.OFPC_FRAG_NORMAL:
        flags.append('NORMAL')
    if msg.flags & ofp.OFPC_FRAG_DROP:
        flags.append('DROP')
    if msg.flags & ofp.OFPC_FRAG_REASM:
        flags.append('REASM')
    self.logger.debug('OFPGetConfigReply received: '
                      'flags=%s miss_send_len=%d',
                      ','.join(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_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=None*)

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	Bitmap of the following flags. OFPFF_SEND_FLOW_REM OFPFF_CHECK_OVERLAP OFPFF_RESET_COUNTS OFPFF_NO_PKT_COUNTS OFPFF_NO_BYT_COUNTS
match	Instance of `OFPMatch`
instructions	list of `OFPInstruction*` instance

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:

```
{
   "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
                           }
                        },
                        "len": 16,
                        "type": 25
                     }
                  },
                  {
                     "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
                    }
                },
                {
                    "OFPActionExperimenterUnknown": {
                        "data": "AAECAwQFBgc=",
                        "experimenter": 101,
                        "len": 16,
                        "type": 65535
                     }
                  },
                  {
                     "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": 176,
               "type": 3
            }
         },
         {
            "OFPInstructionActions": {
               "actions": [
                  {
                     "OFPActionSetField": {
                        "field": {
                           "OXMTlv": {
                              "field": "eth_src",
                              "mask": null,
                              "value": "01:02:03:04:05:06"
                           }
                        },
                        "len": 16,
                        "type": 25
                     }
                  },
                  {
                     "OFPActionSetField": {
                        "field": {
                           "OXMTlv": {
                              "field": "pbb_uca",
                              "mask": null,
                              "value": 1
                           }
                        },
                            "len": 16,
                            "type": 25
                        }
                    }
                ],
                "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,
               "meter_id": 1,
               "type": 6
            }
         },
         {
            "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=0, type_=0, group_id=0, buckets=None*)

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=0*, *hw_addr='00:00:00:00:00:00'*, *config=0*, *mask=0*, *advertise=0*)

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:

```python
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=None*)

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	Bitmap 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_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
                                     }
                                  },
                                  "len": 16,
                                  "type": 25
                               }
                            },
                            {
                               "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
                            }
                        },
                        {
                            "OFPActionExperimenterUnknown": {
                                "len": 16,
                                "data": "ZXhwX2RhdGE=",
                                "experimenter": 98765432,
                                "type": 65535
                            }
                        },
                        {
                            "NXActionUnknown": {
                                "len": 16,
                                "data": "cF9kYXRh",
                                "experimenter": 8992,
                                "type": 65535,
                                "subtype": 25976
                            }
                        }
                    ],
                    "len": 192,
                    "type": 3
                }
            },
            {
                "OFPInstructionActions": {
                    "actions": [
                        {
                            "OFPActionSetField": {
                                "field": {
                                    "OXMTlv": {
                                        "field": "eth_src",
                                        "mask": null,
                                        "value": "01:02:03:04:05:06"
                                    }
                                },
                                "len": 16,
                                "type": 25
                            }
                        },
                        {
                            "OFPActionSetField": {
                                "field": {
                                    "OXMTlv": {
                                       "field": "pbb_uca",
                                       "mask": null,
                                       "value": 1
                                    }
                                 },
                                 "len": 16,
                                 "type": 25
                              }
                           }
                        ],
                        "len": 40,
                        "type": 4
                     }
                  },
                  {
                     "OFPInstructionActions": {
                        "actions": [
                           {
                              "OFPActionOutput": {
                                 "len": 16,
                                 "max_len": 65535,
                                 "port": 4294967293,
                                 "type": 0
                              }
                           }
                        ],
                        "len": 24,
                        "type": 3
                     }
                  }
               ],
               "length": 312,
               "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_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:

```python
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 `OFPPort` 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_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', descs)
```

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_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:

```
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:

```python
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:

```json
{
   "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:

```python
@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_bands=%d '
                        'max_color=%d' %
                        (stat.max_meter, stat.band_types,
                         stat.capabilities, stat.max_bands,
                         stat.max_color))
    self.logger.debug('MeterFeaturesStats: %s', features)
```

JSON Example:

```json
{
   "OFPMeterFeaturesStatsReply": {
      "body": [
         {
            "OFPMeterFeaturesStats": {
               "band_types": 2147483654,
               "capabilities": 15,
                "max_bands": 255,
                "max_color": 0,
                "max_meter": 16777216
            }
        }
    ],
    "flags": 0,
    "type": 11
  }
}
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPTableFeaturesStatsRequest`**(*datapath*, *flags=0*, *body=None*, *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.

Attribute	Description
body	List of `OFPTableFeaturesStats` instance

JSON Example:

See an example in:

`ryu/tests/unit/ofproto/json/of13/4-56-ofp_table_features_reply.packet.json`

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": 64,
               "port": 77,
               "properties": [
                  {
                     "OFPQueuePropMinRate": {
                        "len": 16,
                        "property": 1,
                        "rate": 10
                     }
                  },
                  {
                     "OFPQueuePropMaxRate": {
                        "len": 16,
                        "property": 2,
                        "rate": 900
                     }
                  },
                  {
                     "OFPQueuePropExperimenter": {
                        "data": [],
                        "experimenter": 999,
                        "len": 16,
                        "property": 65535
                     }
```

```
                    }
                ],
                "queue_id": 99
            }
        },
        {
            "OFPPacketQueue": {
                "len": 65,
                "port": 77,
                "properties": [
                    {
                        "OFPQueuePropMinRate": {
                            "len": 16,
                            "property": 1,
                            "rate": 100
                        }
                    },
                    {
                        "OFPQueuePropMaxRate": {
                            "len": 16,
                            "property": 2,
                            "rate": 200
                        }
                    },
                    {
                        "OFPQueuePropExperimenter": {
                            "experimenter": 999,
                            "data": [
                                1
                            ],
                            "len": 17,
                            "property": 65535
                        }
                    }
                ],
                "queue_id": 88
            }
        },
        {
            "OFPPacketQueue": {
                "len": 66,
                "port": 77,
                "properties": [
                    {
                        "OFPQueuePropMinRate": {
                            "len": 16,
                            "property": 1,
                            "rate": 200
                        }
                    },
                    {
                        "OFPQueuePropMaxRate": {
                            "len": 16,
                            "property": 2,
                            "rate": 400
                        }
                    },
                    {
                        "OFPQueuePropExperimenter": {
                           "experimenter": 999,
                           "data": [
                              1,
                              2
                           ],
                           "len": 18,
                           "property": 65535
                        }
                     }
                  ],
                  "queue_id": 77
               }
            }
         ]
      }
}
```

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 of a binary type value or an instances of packet.Packet.

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+BoAAP8Br4sKAAABCgAAAggAAgj3YAAAMdYCAAAAAACrjS0xAAAAABAREhMUFRYXGBkaG
↪",
        "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
    dp = msg.datapath
    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 = 1 << ofp.OFPR_ACTION | 1 << ofp.OFPR_INVALID_TTL
    port_status_mask = (1 << ofp.OFPPR_ADD
                        | 1 << ofp.OFPPR_DELETE
                        | 1 << ofp.OFPPR_MODIFY)
    flow_removed_mask = (1 << ofp.OFPRR_IDLE_TIMEOUT
                         | 1 << ofp.OFPRR_HARD_TIMEOUT
                         | 1 << 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
    dp = msg.datapath
    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*, ***kwargs*)

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

If `type == OFPET_EXPERIMENTER`, this message has also the following attributes.

Attribute	Description
exp_type	Experimenter defined type
experimenter	Experimenter ID

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=None*)

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_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_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
    }
}
```

Port Structures

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPPort`**

Description of a port

Attribute	Description
port_no	Port number and it uniquely identifies a port within a switch.
hw_addr	MAC address for the port.
name	Null-terminated string containing a human-readable name for the interface.
config	Bitmap of port configration flags. OFPPC_PORT_DOWN OFPPC_NO_RECV OFPPC_NO_FWD OFPPC_NO_PACKET_IN
state	Bitmap of port state flags. OFPPS_LINK_DOWN OFPPS_BLOCKED OFPPS_LIVE
curr	Current features.
advertised	Features being advertised by the port.
supported	Features supported by the port.
peer	Features advertised by peer.
curr_speed	Current port bitrate in kbps.
max_speed	Max port bitrate in kbps.

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

Continued on next page

Table 2.2 – continued from previous page

Argument	Value	Description
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
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
pbb_uca	Integer 8bit	PBB UCA header field (EXT-256 Old version of ONF Extension)
tcp_flags	Integer 16bit	TCP flags (EXT-109 ONF Extension)
actset_output	Integer 32bit	Output port from action set metadata (EXT-233 ONF Extension)

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::')
```

Note: For the list of the supported Nicira experimenter matches, please refer to *ryu.ofproto.nx_match*.

Note: For VLAN id match field, special values are defined in OpenFlow Spec.

1. Packets with and without a VLAN tag
   - Example:

   ```
   match = parser.OFPMatch()
   ```

   - Packet Matching

non-VLAN-tagged	MATCH
VLAN-tagged(vlan_id=3)	MATCH
VLAN-tagged(vlan_id=5)	MATCH

2. Only packets without a VLAN tag
   - Example:

   ```
   match = parser.OFPMatch(vlan_vid=0x0000)
   ```

   - Packet Matching

non-VLAN-tagged	MATCH
VLAN-tagged(vlan_id=3)	x
VLAN-tagged(vlan_id=5)	x

3. Only packets with a VLAN tag regardless of its value
   - Example:

   ```
   match = parser.OFPMatch(vlan_vid=(0x1000, 0x1000))
   ```

   - Packet Matching

non-VLAN-tagged	x
VLAN-tagged(vlan_id=3)	MATCH
VLAN-tagged(vlan_id=5)	MATCH

4. Only packets with VLAN tag and VID equal
   - Example:

   ```
   match = parser.OFPMatch(vlan_vid=(0x1000 | 3))
   ```

   - Packet Matching

non-VLAN-tagged	x
VLAN-tagged(vlan_id=3)	MATCH
VLAN-tagged(vlan_id=5)	x

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=1*, *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=0*, *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.

The set of keywords available for this is same as OFPMatch.

Example:

```
set_field = OFPActionSetField(eth_src="00:00:00:00:00:00")
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`OFPActionExperimenter`**(*experimenter*)

Experimenter action

This action is an extensible action for the experimenter.

Attribute	Description
experimenter	Experimenter ID

Note: For the list of the supported Nicira experimenter actions, please refer to *ryu.ofproto.nx_actions*.

2.5.5 OpenFlow v1.4 Messages and Structures

Controller-to-Switch Messages

Handshake

class `ryu.ofproto.ofproto_v1_4_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_4_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_4_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	Bitmap of the following flags. OFPC_FRAG_NORMAL OFPC_FRAG_DROP OFPC_FRAG_REASM
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_4_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_4_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	Bitmap of the following flags. OFPC_FRAG_NORMAL OFPC_FRAG_DROP OFPC_FRAG_REASM
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
    flags = []

    if msg.flags & ofp.OFPC_FRAG_NORMAL:
        flags.append('NORMAL')
    if msg.flags & ofp.OFPC_FRAG_DROP:
        flags.append('DROP')
    if msg.flags & ofp.OFPC_FRAG_REASM:
        flags.append('REASM')
    self.logger.debug('OFPGetConfigReply received: '
                      'flags=%s miss_send_len=%d',
                      ','.join(flags), msg.miss_send_len)
```

JSON Example:

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

Modify State Messages

class `ryu.ofproto.ofproto_v1_4_parser.`**`OFPTableMod`**(*datapath*, *table_id*, *config*, *properties*)

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 configuration flags. OFPTC_EVICTION OFPTC_VACANCY_EVENTS
properties	List of `OFPTableModProp` subclass instance

Example:

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

    req = ofp_parser.OFPTableMod(datapath, 1, 3)
    flags = ofp.OFPTC_VACANCY_EVENTS
    properties = [ofp_parser.OFPTableModPropEviction(flags)]
    req = ofp_parser.OFPTableMod(datapath, 1, 3, properties)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPTableMod": {
      "config": 0,
      "properties": [
         {
            "OFPTableModPropEviction": {
               "flags": 0,
               "length": 8,
               "type": 2
            }
         },
         {
            "OFPTableModPropVacancy": {
               "length": 8,
               "type": 3,
               "vacancy": 0,
               "vacancy_down": 0,
               "vacancy_up": 0
            }
         },
         {
            "OFPTableModPropExperimenter": {
               "data": [],
               "exp_type": 0,
               "experimenter": 101,
               "length": 12,
               "type": 65535
            }
         },
         {
            "OFPTableModPropExperimenter": {
               "data": [
                  1
               ],
               "exp_type": 1,
               "experimenter": 101,
               "length": 16,
               "type": 65535
            }
         },
         {
            "OFPTableModPropExperimenter": {
               "data": [
                  1,
                  2
               ],
                "exp_type": 2,
                "experimenter": 101,
                "length": 20,
                "type": 65535
            }
        }
    ],
    "table_id": 255
  }
}
```

class `ryu.ofproto.ofproto_v1_4_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*, *importance=0*, *match=None*, *instructions=None*)

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	Bitmap of the following flags. OFPFF_SEND_FLOW_REM OFPFF_CHECK_OVERLAP OFPFF_RESET_COUNTS OFPFF_NO_PKT_COUNTS OFPFF_NO_BYT_COUNTS
importance	Eviction precedence
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
    importance = 0
    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,
                                importance,
                                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,
      "importance": 0,
      "instructions": [
         {
            "OFPInstructionActions": {
               "actions": [
                  {
                     "OFPActionSetField": {
                        "field": {
                           "OXMTlv": {
                              "field": "vlan_vid",
                              "mask": null,
                              "value": 258
                           }
                        },
                        "len": 16,
                        "type": 25
                     }
                  },
                  {
                     "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
                    }
                 },
                 {
                    "OFPActionExperimenterUnknown": {
                       "data": "AAECAwQFBgc=",
                       "experimenter": 101,
                       "len": 16,
                       "type": 65535
                    }
                 },
                 {
                    "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": 176,
              "type": 3
           }
        },
        {
            "OFPInstructionActions": {
               "actions": [
                  {
                     "OFPActionSetField": {
                        "field": {
                           "OXMTlv": {
                              "field": "eth_src",
                              "mask": null,
                              "value": "01:02:03:04:05:06"
                           }
                        },
                        "len": 16,
                        "type": 25
                     }
                  },
                  {
                     "OFPActionSetField": {
                        "field": {
                           "OXMTlv": {
                              "field": "pbb_uca",
                              "mask": null,
                              "value": 1
                           }
                        },
                        "len": 16,
                        "type": 25
                     }
                  }
               ],
               "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,
        "importance": 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,
        "importance": 0,
        "instructions": [
            {
                "OFPInstructionMeter": {
                    "len": 8,
                    "meter_id": 1,
                    "type": 6
                }
            },
            {
                "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
    }
}
```

```
{
    "OFPFlowMod": {
        "buffer_id": 65535,
        "command": 0,
        "cookie": 0,
        "cookie_mask": 0,
        "flags": 0,
        "hard_timeout": 0,
        "idle_timeout": 0,
        "importance": 0,
        "instructions": [],
        "match": {
            "OFPMatch": {
```

```
            "length": 329,
            "oxm_fields": [
               {
                  "OXMTlv": {
                     "field": "in_port",
                     "mask": null,
                     "value": 84281096
                  }
               },
               {
                  "OXMTlv": {
                     "field": "in_phy_port",
                     "mask": null,
                     "value": 16909060
                  }
               },
               {
                  "OXMTlv": {
                     "field": "metadata",
                     "mask": null,
                     "value": 283686952306183
                  }
               },
               {
                  "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": "vlan_vid",
                     "mask": null,
                     "value": 999
                  }
               },
               {
                  "OXMTlv": {
                     "field": "ip_dscp",
                     "mask": null,
                     "value": 9
                  }
               },
               {
                  "OXMTlv": {
                     "field": "ip_ecn",
                     "mask": null,
                     "value": 3
                  }
               },
               {
                  "OXMTlv": {
                     "field": "ip_proto",
                     "mask": null,
                     "value": 99
                  }
               },
               {
                  "OXMTlv": {
                     "field": "ipv4_src",
                     "mask": null,
                     "value": "1.2.3.4"
                  }
               },
               {
                  "OXMTlv": {
                     "field": "ipv4_dst",
                     "mask": null,
                     "value": "1.2.3.4"
                  }
               },
               {
                  "OXMTlv": {
                     "field": "tcp_src",
                     "mask": null,
                     "value": 8080
                  }
               },
               {
                  "OXMTlv": {
                     "field": "tcp_dst",
                     "mask": null,
                     "value": 18080
                  }
               },
               {
                  "OXMTlv": {
                     "field": "udp_src",
                     "mask": null,
                     "value": 28080
                  }
               },
               {
                  "OXMTlv": {
                     "field": "udp_dst",
                     "mask": null,
                     "value": 55936
                  }
               },
               {
                  "OXMTlv": {
```

```
                  "field": "sctp_src",
                  "mask": null,
                  "value": 48080
               }
            },
            {
               "OXMTlv": {
                  "field": "sctp_dst",
                  "mask": null,
                  "value": 59328
               }
            },
            {
               "OXMTlv": {
                  "field": "icmpv4_type",
                  "mask": null,
                  "value": 100
               }
            },
            {
               "OXMTlv": {
                  "field": "icmpv4_code",
                  "mask": null,
                  "value": 101
               }
            },
            {
               "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"
                  }
               },
               {
                  "OXMTlv": {
                     "field": "ipv6_src",
                     "mask": null,
                     "value": "fe80::f00b:a4ff:fe48:28a5"
                  }
               },
               {
                  "OXMTlv": {
                     "field": "ipv6_dst",
                     "mask": null,
                     "value": "fe80::f00b:a4ff:fe05:b7dc"
                  }
               },
               {
                  "OXMTlv": {
                     "field": "ipv6_flabel",
                     "mask": null,
                     "value": 541473
                  }
               },
               {
                  "OXMTlv": {
                     "field": "icmpv6_type",
                     "mask": null,
                     "value": 200
                  }
               },
               {
                  "OXMTlv": {
                     "field": "icmpv6_code",
                     "mask": null,
                     "value": 201
                  }
               },
               {
                  "OXMTlv": {
                     "field": "ipv6_nd_target",
                     "mask": null,
                     "value": "fe80::a60:6eff:fe7f:74e7"
                  }
               },
               {
                  "OXMTlv": {
                     "field": "ipv6_nd_sll",
                     "mask": null,
                     "value": "00:00:00:00:02:9a"
                  }
               },
               {
                  "OXMTlv": {
                     "field": "ipv6_nd_tll",
                     "mask": null,
                     "value": "00:00:00:00:02:2b"
                  }
```

```
                },
                {
                    "OXMTlv": {
                        "field": "mpls_label",
                        "mask": null,
                        "value": 624485
                    }
                },
                {
                    "OXMTlv": {
                        "field": "mpls_tc",
                        "mask": null,
                        "value": 5
                    }
                },
                {
                    "OXMTlv": {
                        "field": "mpls_bos",
                        "mask": null,
                        "value": 1
                    }
                },
                {
                    "OXMTlv": {
                        "field": "pbb_isid",
                        "mask": null,
                        "value": 11259375
                    }
                },
                {
                    "OXMTlv": {
                        "field": "tunnel_id",
                        "mask": null,
                        "value": 651061555542690057
                    }
                },
                {
                    "OXMTlv": {
                        "field": "ipv6_exthdr",
                        "mask": null,
                        "value": 500
                    }
                },
                {
                    "OXMTlv": {
                        "field": "pbb_uca",
                        "mask": null,
                        "value": 1
                    }
                }
            ],
            "type": 1
        }
    },
    "out_group": 4294967295,
    "out_port": 4294967295,
    "priority": 123,
    "table_id": 1
    }
}
```

class `ryu.ofproto.ofproto_v1_4_parser.`**`OFPGroupMod`**(*datapath*, *command=0*, *type_=0*, *group_id=0*, *buckets=None*)

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_4_parser.`**`OFPPortMod`**(*datapath, port_no=0, hw_addr='00:00:00:00:00:00', config=0, mask=0, properties=None*)

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
properties	List of `OFPPortModProp` subclass instance

Example:

```python
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)
    properties = [ofp_parser.OFPPortModPropEthernet(advertise)]
    req = ofp_parser.OFPPortMod(datapath, port_no, hw_addr, config,
                                 mask, properties)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPPortMod": {
      "config": 0,
      "hw_addr": "00:11:00:00:11:11",
      "mask": 0,
      "port_no": 1,
      "properties": [
         {
            "OFPPortModPropEthernet": {
               "advertise": 4096,
               "length": 8,
               "type": 0
            }
         },
         {
            "OFPPortModPropOptical": {
               "configure": 3,
               "fl_offset": 2000,
               "freq_lmda": 1500,
               "grid_span": 3000,
               "length": 24,
               "tx_pwr": 300,
               "type": 1
            }
         },
         {
            "OFPPortModPropExperimenter": {
               "data": [],
               "exp_type": 0,
               "experimenter": 101,
               "length": 12,
               "type": 65535
            }
         },
         {
            "OFPPortModPropExperimenter": {
               "data": [
                  1
               ],
               "exp_type": 1,
               "experimenter": 101,
               "length": 16,
               "type": 65535
            }
         },
         {
            "OFPPortModPropExperimenter": {
               "data": [
                  1,
                  2
```

```
                    ],
                    "exp_type": 2,
                    "experimenter": 101,
                    "length": 20,
                    "type": 65535
                }
            }
        ]
    }
}
```

class `ryu.ofproto.ofproto_v1_4_parser.`**`OFPMeterMod`**(*datapath*, *command=0*, *flags=1*, *meter_id=1*, *bands=None*)

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	Bitmap 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_4_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_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_4_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_4_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_4_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 '
                     'importance=%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.flags, stat.importance,
                      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,
               "importance": 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,
               "importance": 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,
               "importance": 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,
               "importance": 0,
               "instructions": [
                  {
                     "OFPInstructionActions": {
                        "actions": [
                           {
                              "OFPActionSetField": {
                                 "field": {
                                    "OXMTlv": {
                                       "field": "vlan_vid",
                                       "mask": null,
                                       "value": 258
                                    }
                                 },
                                 "len": 16,
                                 "type": 25
                              }
                           },
                           {
                              "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
                              }
                           },
                           {
                              "OFPActionExperimenterUnknown": {
                                 "len": 16,
                                 "data": "ZXhwX2RhdGE=",
                                 "experimenter": 98765432,
                                 "type": 65535
                              }
                           },
                           {
                              "NXActionUnknown": {
                                 "len": 16,
                                 "data": "cF9kYXRh",
                                 "experimenter": 8992,
                                 "type": 65535,
                                 "subtype": 25976
                              }
                           }
                        ],
                        "len": 192,
                        "type": 3
                     }
                  },
                  {
                     "OFPInstructionActions": {
                        "actions": [
                           {
                              "OFPActionSetField": {
                                 "field": {
                                    "OXMTlv": {
                                       "field": "eth_src",
                                       "mask": null,
                                       "value": "01:02:03:04:05:06"
                                    }
                                 },
                                 "len": 16,
                                 "type": 25
                              }
                           },
                           {
                              "OFPActionSetField": {
                                 "field": {
                                    "OXMTlv": {
                                       "field": "pbb_uca",
                                       "mask": null,
                                       "value": 1
                                    }
                                 },
                                 "len": 16,
                                 "type": 25
                              }
                           }
                        ],
                        "len": 40,
                        "type": 4
                     }
                  },
                  {
                     "OFPInstructionActions": {
                        "actions": [
                           {
                              "OFPActionOutput": {
                                 "len": 16,
                                 "max_len": 65535,
                                 "port": 4294967293,
                                 "type": 0
                              }
                           }
                        ],
                        "len": 24,
                        "type": 3
                     }
                  }
               ],
               "length": 312,
               "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_4_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:

```python
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:

```json
{
   "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_4_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_4_parser.`**`OFPTableStatsRequest`**(*datapath*, *flags*, *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_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_4_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_4_parser.`**`OFPTableDescStatsRequest`**(*datapath*, *flags=0*, *type_=None*)

Table description request message

The controller uses this message to query description of all the tables.

Attribute	Description
flags	Zero or `OFPMPF_REQ_MORE`

Example:

```
def send_table_desc_stats_request(self, datapath):
    ofp_parser = datapath.ofproto_parser
    req = ofp_parser.OFPTableDescStatsRequest(datapath, 0)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPTableDescStatsRequest": {
      "flags": 0,
      "type": 14
   }
}
```

class `ryu.ofproto.ofproto_v1_4_parser.`**`OFPTableDescStatsReply`**(*datapath*, *type_=None*, ***kwargs*)

Table description reply message

The switch responds with this message to a table description request.

Attribute	Description
body	List of `OFPTableDesc` instance

Example:

```
@set_ev_cls(ofp_event.EventOFPTableDescStatsReply, MAIN_DISPATCHER)
def table_desc_stats_reply_handler(self, ev):
    tables = []
    for p in ev.msg.body:
        tables.append('table_id=%d config=0x%08x properties=%s' %
                      (p.table_id, p.config, repr(p.properties)))
    self.logger.debug('OFPTableDescStatsReply received: %s', tables)
```

JSON Example:

```
{
   "OFPTableDescStatsReply": {
      "body": [
         {
            "OFPTableDesc": {
               "config": 0,
               "length": 24,
               "properties": [
                  {
                     "OFPTableModPropExperimenter": {
                        "data": [],
                        "exp_type": 0,
                        "experimenter": 101,
                        "length": 12,
                        "type": 65535
                     }
                  }
               ],
               "table_id": 7
            }
         },
         {
            "OFPTableDesc": {
               "config": 0,
               "length": 80,
               "properties": [
                    {
                        "OFPTableModPropEviction": {
                            "flags": 0,
                            "length": 8,
                            "type": 2
                        }
                    },
                    {
                        "OFPTableModPropVacancy": {
                            "length": 8,
                            "type": 3,
                            "vacancy": 0,
                            "vacancy_down": 0,
                            "vacancy_up": 0
                        }
                    },
                    {
                        "OFPTableModPropExperimenter": {
                            "data": [],
                            "exp_type": 0,
                            "experimenter": 101,
                            "length": 12,
                            "type": 65535
                        }
                    },
                    {
                        "OFPTableModPropExperimenter": {
                            "data": [
                                1
                            ],
                            "exp_type": 1,
                            "experimenter": 101,
                            "length": 16,
                            "type": 65535
                        }
                    },
                    {
                        "OFPTableModPropExperimenter": {
                            "data": [
                                1,
                                2
                            ],
                            "exp_type": 2,
                            "experimenter": 101,
                            "length": 20,
                            "type": 65535
                        }
                    }
                ],
                "table_id": 8
            }
        }
    ],
    "flags": 0,
    "type": 14
  }
}
```

class `ryu.ofproto.ofproto_v1_4_parser.`**`OFPTableFeaturesStatsRequest`**(*datapath*, *flags=0*, *body=None*, *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 [].

JSON Example:

See an example in:

`ryu/tests/unit/ofproto/json/of14/5-53-ofp_table_features_request.packet.json`

class `ryu.ofproto.ofproto_v1_4_parser.`**`OFPTableFeaturesStatsReply`**(*datapath*, *type_=None*, ***kwargs*)

Table features statistics reply message

The switch responds with this message to a table features statistics request.

Attribute	Description
body	List of `OFPTableFeaturesStats` instance

JSON Example:

See an example in:

`ryu/tests/unit/ofproto/json/of14/5-54-ofp_table_features_reply.packet.json`

class `ryu.ofproto.ofproto_v1_4_parser.`**`OFPPortStatsRequest`**(*datapath*, *flags*, *port_no*, *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:

```python
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:

```json
{
   "OFPPortStatsRequest": {
      "flags": 0,
      "port_no": 4294967295,
      "type": 4
   }
}
```

class `ryu.ofproto.ofproto_v1_4_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(stat.length, stat.port_no,
                     stat.duration_sec, stat.duration_nsec,
                     stat.rx_packets, stat.tx_packets,
                     stat.rx_bytes, stat.tx_bytes,
                     stat.rx_dropped, stat.tx_dropped,
                     stat.rx_errors, stat.tx_errors,
                     repr(stat.properties))
    self.logger.debug('PortStats: %s', ports)
```

JSON Example:

```
{
   "OFPPortStatsReply": {
      "body": [
         {
            "OFPPortStats": {
               "duration_nsec": 0,
               "duration_sec": 0,
               "length": 224,
               "port_no": 7,
               "properties": [
                  {
                     "OFPPortStatsPropEthernet": {
                        "collisions": 0,
                        "length": 40,
                        "rx_crc_err": 0,
                        "rx_frame_err": 0,
                        "rx_over_err": 0,
                        "type": 0
                     }
                  },
                  {
                     "OFPPortStatsPropOptical": {
                        "bias_current": 300,
                        "flags": 3,
                        "length": 44,
                        "rx_freq_lmda": 1500,
                        "rx_grid_span": 500,
                        "rx_offset": 700,
                        "rx_pwr": 2000,
                        "temperature": 273,
                        "tx_freq_lmda": 1500,
                        "tx_grid_span": 500,
                        "tx_offset": 700,
                        "tx_pwr": 2000,
                        "type": 1
                    }
                },
                {
                    "OFPPortStatsPropExperimenter": {
                        "data": [],
                        "exp_type": 0,
                        "experimenter": 101,
                        "length": 12,
                        "type": 65535
                    }
                },
                {
                    "OFPPortStatsPropExperimenter": {
                        "data": [
                            1
                        ],
                        "exp_type": 1,
                        "experimenter": 101,
                        "length": 16,
                        "type": 65535
                    }
                },
                {
                    "OFPPortStatsPropExperimenter": {
                        "data": [
                            1,
                            2
                        ],
                        "exp_type": 2,
                        "experimenter": 101,
                        "length": 20,
                        "type": 65535
                    }
                }
            ],
            "rx_bytes": 0,
            "rx_dropped": 0,
            "rx_errors": 0,
            "rx_packets": 0,
            "tx_bytes": 336,
            "tx_dropped": 0,
            "tx_errors": 0,
            "tx_packets": 4
        }
    },
    {
        "OFPPortStats": {
            "duration_nsec": 0,
            "duration_sec": 0,
            "length": 120,
            "port_no": 6,
            "properties": [
                {
                    "OFPPortStatsPropEthernet": {
                        "collisions": 0,
                        "length": 40,
                        "rx_crc_err": 0,
                            "rx_frame_err": 0,
                            "rx_over_err": 0,
                            "type": 0
                         }
                      }
                   ],
                   "rx_bytes": 336,
                   "rx_dropped": 0,
                   "rx_errors": 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_4_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:

```python
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_4_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 `OFPPort` 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 properties=%s' %
                     (p.port_no, p.hw_addr,
                      p.name, p.config, p.state, repr(p.properties)))
    self.logger.debug('OFPPortDescStatsReply received: %s', ports)
```

JSON Example:

```
{
   "OFPPortDescStatsReply": {
      "body": [
         {
            "OFPPort": {
               "config": 0,
               "hw_addr": "f2:0b:a4:d0:3f:70",
               "length": 168,
               "name": "Port7",
               "port_no": 7,
               "properties": [
                  {
                     "OFPPortDescPropEthernet": {
                        "advertised": 10240,
                        "curr": 10248,
                        "curr_speed": 5000,
                        "length": 32,
                        "max_speed": 5000,
                        "peer": 10248,
                        "supported": 10248,
                        "type": 0
                     }
                  },
                  {
                     "OFPPortDescPropOptical": {
                        "length": 40,
                        "rx_grid_freq_lmda": 1500,
                        "rx_max_freq_lmda": 2000,
                        "rx_min_freq_lmda": 1000,
                        "supported": 1,
                        "tx_grid_freq_lmda": 1500,
                        "tx_max_freq_lmda": 2000,
                        "tx_min_freq_lmda": 1000,
                        "tx_pwr_max": 2000,
                        "tx_pwr_min": 1000,
                        "type": 1
                     }
                  },
                  {
                     "OFPPortDescPropExperimenter": {
                        "data": [],
                        "exp_type": 0,
                        "experimenter": 101,
                        "length": 12,
                        "type": 65535
                     }
                  },
                  {
                     "OFPPortDescPropExperimenter": {
                        "data": [
                           1
                        ],
                        "exp_type": 1,
                        "experimenter": 101,
                        "length": 16,
                        "type": 65535
                     }
                  },
                  {
                     "OFPPortDescPropExperimenter": {
                        "data": [
                           1,
                           2
                        ],
                        "exp_type": 2,
                        "experimenter": 101,
                        "length": 20,
                        "type": 65535
                     }
                  }
               ],
               "state": 4
            }
         },
         {
            "OFPPort": {
               "config": 0,
               "hw_addr": "f2:0b:a4:7d:f8:ea",
               "length": 72,
               "name": "Port6",
               "port_no": 6,
               "properties": [
                  {
                     "OFPPortDescPropEthernet": {
                        "advertised": 10240,
                        "curr": 10248,
                        "curr_speed": 5000,
                        "length": 32,
                        "max_speed": 5000,
                        "peer": 10248,
                        "supported": 10248,
                        "type": 0
                     }
                  }
               ],
               "state": 4
            }
         }
      ],
      "flags": 0,
      "type": 13
   }
}
```

class `ryu.ofproto.ofproto_v1_4_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_4_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'
                      'properties=%s' %
                      (stat.port_no, stat.queue_id,
                       stat.tx_bytes, stat.tx_packets, stat.tx_errors,
                       stat.duration_sec, stat.duration_nsec,
                       repr(stat.properties)))
    self.logger.debug('QueueStats: %s', queues)
```

JSON Example:

```
{
   "OFPQueueStatsReply": {
      "body": [
         {
            "OFPQueueStats": {
               "duration_nsec": 0,
               "duration_sec": 0,
               "length": 104,
               "port_no": 7,
               "properties": [
                  {
                     "OFPQueueStatsPropExperimenter": {
                        "data": [],
                        "exp_type": 0,
                        "experimenter": 101,
                        "length": 12,
                        "type": 65535
                     }
                  },
                  {
                     "OFPQueueStatsPropExperimenter": {
                        "data": [
                           1
                        ],
                        "exp_type": 1,
                        "experimenter": 101,
                        "length": 16,
                        "type": 65535
                     }
                  },
                  {
                     "OFPQueueStatsPropExperimenter": {
                        "data": [
                           1,
                           2
                        ],
                        "exp_type": 2,
                        "experimenter": 101,
                        "length": 20,
                        "type": 65535
                     }
                  }
               ],
               "queue_id": 1,
               "tx_bytes": 0,
               "tx_errors": 0,
               "tx_packets": 0
            }
         },
         {
            "OFPQueueStats": {
               "duration_nsec": 0,
               "duration_sec": 0,
               "length": 48,
               "port_no": 6,
               "properties": [],
               "queue_id": 1,
               "tx_bytes": 0,
                "tx_errors": 0,
                "tx_packets": 0
             }
          },
          {
             "OFPQueueStats": {
                "duration_nsec": 0,
                "duration_sec": 0,
                "length": 48,
                "port_no": 7,
                "properties": [],
                "queue_id": 2,
                "tx_bytes": 0,
                "tx_errors": 0,
                "tx_packets": 0
             }
          }
       ],
       "flags": 0,
       "type": 5
    }
}
```

class `ryu.ofproto.ofproto_v1_4_parser.`**`OFPQueueDescStatsRequest`**(*datapath*, *flags=0*, *port_no=4294967295*, *queue_id=4294967295*, *type_=None*)

Queue description request message

The controller uses this message to query description of all the queues.

Attribute	Description
flags	Zero or `OFPMPF_REQ_MORE`
port_no	Port number to read (OFPP_ANY for all ports)
queue_id	ID of queue to read (OFPQ_ALL for all queues)

Example:

```
def send_queue_desc_stats_request(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser
    req = ofp_parser.OFPQueueDescStatsRequest(datapath, 0,
                                              ofp.OFPP_ANY,
                                              ofp.OFPQ_ALL)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPQueueDescStatsRequest": {
      "flags": 0,
      "port_no": 7,
      "queue_id": 4294967295,
      "type": 15
   }
}
```

class `ryu.ofproto.ofproto_v1_4_parser.`**`OFPQueueDescStatsReply`**(*datapath*, *type_=None*, ***kwargs*)

Queue description reply message

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

Attribute	Description
body	List of `OFPQueueDesc` instance

Example:

```
@set_ev_cls(ofp_event.EventOFPQueueDescStatsReply, MAIN_DISPATCHER)
def queue_desc_stats_reply_handler(self, ev):
    queues = []
    for q in ev.msg.body:
        queues.append('port_no=%d queue_id=0x%08x properties=%s' %
                      (q.port_no, q.queue_id, repr(q.properties)))
    self.logger.debug('OFPQueueDescStatsReply received: %s', queues)
```

JSON Example:

```
{
   "OFPQueueDescStatsReply": {
      "body": [
         {
            "OFPQueueDesc": {
               "len": 32,
               "port_no": 7,
               "properties": [
                  {
                     "OFPQueueDescPropExperimenter": {
                        "data": [],
                        "exp_type": 0,
                        "experimenter": 101,
                        "length": 12,
                        "type": 65535
                     }
                  }
               ],
               "queue_id": 0
            }
         },
         {
            "OFPQueueDesc": {
               "len": 88,
               "port_no": 8,
               "properties": [
                  {
                     "OFPQueueDescPropMinRate": {
                        "length": 8,
                        "rate": 300,
                        "type": 1
                     }
                  },
                  {
                     "OFPQueueDescPropMaxRate": {
                        "length": 8,
                        "rate": 900,
                        "type": 2
                     }
                  },
                  {
                        "OFPQueueDescPropExperimenter": {
                           "data": [],
                           "exp_type": 0,
                           "experimenter": 101,
                           "length": 12,
                           "type": 65535
                        }
                     },
                     {
                        "OFPQueueDescPropExperimenter": {
                           "data": [
                              1
                           ],
                           "exp_type": 1,
                           "experimenter": 101,
                           "length": 16,
                           "type": 65535
                        }
                     },
                     {
                        "OFPQueueDescPropExperimenter": {
                           "data": [
                              1,
                              2
                           ],
                           "exp_type": 2,
                           "experimenter": 101,
                           "length": 20,
                           "type": 65535
                        }
                     }
                  ],
                  "queue_id": 1
               }
            }
         ],
         "flags": 0,
         "type": 15
   }
}
```

class `ryu.ofproto.ofproto_v1_4_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)
```

JSON Example:

```
{
   "OFPGroupStatsRequest": {
      "flags": 0,
      "group_id": 4294967292,
      "type": 6
   }
}
```

class `ryu.ofproto.ofproto_v1_4_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)
```

JSON Example:

```
{
   "OFPGroupStatsReply": {
      "body": [
         {
            "OFPGroupStats": {
               "bucket_stats": [
                  {
                     "OFPBucketCounter": {
                        "byte_count": 2345,
                        "packet_count": 234
                     }
                  }
               ],
               "byte_count": 12345,
               "duration_nsec": 609036000,
               "duration_sec": 9,
               "group_id": 1,
               "length": 56,
               "packet_count": 123,
               "ref_count": 2
            }
            }
        ],
        "flags": 0,
        "type": 6
    }
}
```

class `ryu.ofproto.ofproto_v1_4_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_4_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', descs)
```

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_4_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:

```python
def send_group_features_stats_request(self, datapath):
    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_4_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_4_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:

```
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_4_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_4_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_4_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_4_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_4_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_bands=%d '
                        'max_color=%d' %
                        (stat.max_meter, stat.band_types,
                         stat.capabilities, stat.max_bands,
                         stat.max_color))
    self.logger.debug('MeterFeaturesStats: %s', features)
```

JSON Example:

```
{
   "OFPMeterFeaturesStatsReply": {
      "body": [
         {
            "OFPMeterFeaturesStats": {
               "band_types": 2147483654,
               "capabilities": 15,
               "max_bands": 255,
               "max_color": 0,
               "max_meter": 16777216
            }
         }
      ],
      "flags": 0,
      "type": 11
   }
}
```

class `ryu.ofproto.ofproto_v1_4_parser.`**`OFPFlowMonitorRequest`**(*datapath*, *flags=0*, *monitor_id=0*, *out_port=4294967295*, *out_group=4294967295*, *monitor_flags=0*, *table_id=255*, *command=0*, *match=None*, *type_=None*)

Flow monitor request message

The controller uses this message to query flow monitors.

Attribute	Description
flags	Zero or `OFPMPF_REQ_MORE`
monitor_id	Controller-assigned ID for this monitor
out_port	Require matching entries to include this as an output port
out_group	Require matching entries to include this as an output group
monitor_flags	Bitmap of the following flags. OFPFMF_INITIAL OFPFMF_ADD OFPFMF_REMOVED OFPFMF_MODIFY OFPFMF_INSTRUCTIONS OFPFMF_NO_ABBREV OFPFMF_ONLY_OWN
table_id	ID of table to monitor
command	One of the following values. OFPFMC_ADD OFPFMC_MODIFY OFPFMC_DELETE
match	Instance of `OFPMatch`

Example:

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

    monitor_flags = [ofp.OFPFMF_INITIAL, ofp.OFPFMF_ONLY_OWN]
    match = ofp_parser.OFPMatch(in_port=1)
    req = ofp_parser.OFPFlowMonitorRequest(datapath, 0, 10000,
                                           ofp.OFPP_ANY, ofp.OFPG_ANY,
                                           monitor_flags,
                                           ofp.OFPTT_ALL,
                                           ofp.OFPFMC_ADD, match)
    datapath.send_msg(req)
```

JSON Example:

```json
{
   "OFPFlowMonitorRequest": {
      "command": 0,
      "flags": 0,
      "match": {
         "OFPMatch": {
            "length": 14,
            "oxm_fields": [
               {
                  "OXMTlv": {
                     "field": "eth_dst",
                     "mask": null,
                     "value": "f2:0b:a4:7d:f8:ea"
                  }
               }
            ],
            "type": 1
         }
      },
      "monitor_flags": 15,
      "monitor_id": 100000000,
      "out_group": 4294967295,
      "out_port": 22,
      "table_id": 33,
      "type": 16
   }
}
```

class `ryu.ofproto.ofproto_v1_4_parser.`**`OFPFlowMonitorReply`**(*datapath*, *type_=None*, ***kwargs*)

Flow monitor reply message

The switch responds with this message to a flow monitor request.

Attribute	Description
body	List of list of the following class instance. OFPFlowMonitorFull OFPFlowMonitorAbbrev OFPFlowMonitorPaused

Example:

```
@set_ev_cls(ofp_event.EventOFPFlowMonitorReply, MAIN_DISPATCHER)
def flow_monitor_reply_handler(self, ev):
    msg = ev.msg
    dp = msg.datapath
    ofp = dp.ofproto
    flow_updates = []

    for update in msg.body:
        update_str = 'length=%d event=%d' %
                     (update.length, update.event)
        if (update.event == ofp.OFPFME_INITIAL or
            update.event == ofp.OFPFME_ADDED or
            update.event == ofp.OFPFME_REMOVED or
            update.event == ofp.OFPFME_MODIFIED):
            update_str += 'table_id=%d reason=%d idle_timeout=%d '
                          'hard_timeout=%d priority=%d cookie=%d '
                          'match=%d instructions=%s' %
                          (update.table_id, update.reason,
                           update.idle_timeout, update.hard_timeout,
                           update.priority, update.cookie,
                           update.match, update.instructions)
        elif update.event == ofp.OFPFME_ABBREV:
            update_str += 'xid=%d' % (update.xid)
        flow_updates.append(update_str)
    self.logger.debug('FlowUpdates: %s', flow_updates)
```

JSON Example:

```
{
   "OFPFlowMonitorReply": {
      "body": [
         {
            "OFPFlowUpdateFull": {
               "cookie": 0,
               "event": 0,
               "hard_timeout": 700,
               "idle_timeout": 600,
               "instructions": [
                  {
                     "OFPInstructionActions": {
                        "actions": [
                           {
                              "OFPActionOutput": {
                                 "len": 16,
                                 "max_len": 0,
                                 "port": 4294967290,
                                 "type": 0
                              }
                           }
                        ],
                        "len": 24,
                        "type": 4
                     }
                  }
               ],
               "length": 64,
               "match": {
                  "OFPMatch": {
                     "length": 10,
                     "oxm_fields": [
                        {
                           "OXMTlv": {
                              "field": "eth_type",
                              "mask": null,
                              "value": 2054
                           }
                        }
                     ],
                     "type": 1
                  }
               },
               "priority": 3,
               "reason": 0,
               "table_id": 0
            }
         },
         {
            "OFPFlowUpdateAbbrev": {
               "event": 4,
               "length": 8,
               "xid": 1234
```

```
                }
            },
            {
                "OFPFlowUpdatePaused": {
                    "event": 5,
                    "length": 8
                }
            }
        ],
        "flags": 0,
        "type": 16
    }
}
```

class `ryu.ofproto.ofproto_v1_4_parser.`**`OFPExperimenterStatsRequest`**(*datapath*, *flags*, *experimenter*, *exp_type*, *data*, *type_=None*)

Experimenter multipart request message

Attribute	Description
flags	Zero or `OFPMPF_REQ_MORE`
experimenter	Experimenter ID
exp_type	Experimenter defined
data	Experimenter defined additional data

JSON Example:

```
{
    "OFPExperimenterStatsRequest": {
        "data": "aG9nZWhvZ2U=",
        "exp_type": 3405678728,
        "experimenter": 3735928495,
        "flags": 0,
        "type": 65535
    }
}
```

class `ryu.ofproto.ofproto_v1_4_parser.`**`OFPExperimenterStatsReply`**(*datapath*, *type_=None*, ***kwargs*)

Experimenter multipart reply message

Attribute	Description
body	An `OFPExperimenterMultipart` instance

JSON Example:

```
{
    "OFPExperimenterStatsReply": {
        "body": {
            "OFPExperimenterMultipart": {
                "data": "dGVzdGRhdGE5OTk5OTk5OQ==",
                "exp_type": 3405674359,
                "experimenter": 3735928495
            }
        },
        "flags": 0,
        "type": 65535
    }
}
```

Packet-Out Message

class `ryu.ofproto.ofproto_v1_4_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 of a binary type value or an instances of packet.Packet.

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+BoAAP8Br4sKAAABCgAAAggAAgj3YAAAMdYCAAAAAACrjS0xAAAAABAREhMUFRYXGBkaG
↪",
      "in_port": 4294967293
   }
}
```

Barrier Message

class `ryu.ofproto.ofproto_v1_4_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_4_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_4_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:

```python
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:

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

class `ryu.ofproto.ofproto_v1_4_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:

```python
@set_ev_cls(ofp_event.EventOFPRoleReply, MAIN_DISPATCHER)
def role_reply_handler(self, ev):
    msg = ev.msg
    dp = msg.datapath
    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
   }
}
```

Bundle Messages

class `ryu.ofproto.ofproto_v1_4_parser.`**`OFPBundleCtrlMsg`**(*datapath*, *bundle_id=None*, *type_=None*, *flags=None*, *properties=None*)

Bundle control message

The controller uses this message to create, destroy and commit bundles

Attribute	Description
bundle_id	Id of the bundle
type	One of the following values. OFPBCT_OPEN_REQUEST OFPBCT_OPEN_REPLY OFPBCT_CLOSE_REQUEST OFPBCT_CLOSE_REPLY OFPBCT_COMMIT_REQUEST OFPBCT_COMMIT_REPLY OFPBCT_DISCARD_REQUEST OFPBCT_DISCARD_REPLY
flags	Bitmap of the following flags. OFPBF_ATOMIC OFPBF_ORDERED
properties	List of `OFPBundleProp` subclass instance

Example:

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

    req = ofp_parser.OFPBundleCtrlMsg(datapath, 7,
                                      ofp.OFPBCT_OPEN_REQUEST,
                                      [ofp.OFPBF_ATOMIC], [])
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPBundleCtrlMsg": {
      "bundle_id": 1234,
      "flags": 1,
      "properties": [
            {
               "OFPBundlePropExperimenter": {
                  "data": [],
                  "exp_type": 0,
                  "experimenter": 101,
                  "length": 12,
                  "type": 65535
               }
            },
            {
               "OFPBundlePropExperimenter": {
                  "data": [
                     1
                  ],
                  "exp_type": 1,
                  "experimenter": 101,
                  "length": 16,
                  "type": 65535
               }
            },
            {
               "OFPBundlePropExperimenter": {
                  "data": [
                     1,
                     2
                  ],
                  "exp_type": 2,
                  "experimenter": 101,
                  "length": 20,
                  "type": 65535
               }
            }
         ],
         "type": 0
      }
}
```

class `ryu.ofproto.ofproto_v1_4_parser.`**`OFPBundleAddMsg`**(*datapath*, *bundle_id*, *flags*, *message*, *properties*)

Bundle control message

The controller uses this message to create, destroy and commit bundles

Attribute	Description
bundle_id	Id of the bundle
flags	Bitmap of the following flags. OFPBF_ATOMIC OFPBF_ORDERED
message	`MsgBase` subclass instance
properties	List of `OFPBundleProp` subclass instance

Example:

```
def send_bundle_add_message(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser
    msg = ofp_parser.OFPRoleRequest(datapath, ofp.OFPCR_ROLE_EQUAL, 0)

    req = ofp_parser.OFPBundleAddMsg(datapath, 7, [ofp.OFPBF_ATOMIC],
                                     msg, [])
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPBundleAddMsg": {
      "bundle_id": 1234,
      "flags": 1,
      "message": {
         "OFPEchoRequest": {
            "data": null
         }
      },
      "properties": [
         {
            "OFPBundlePropExperimenter": {
               "data": [],
               "exp_type": 0,
               "experimenter": 101,
               "length": 12,
               "type": 65535
            }
         },
         {
            "OFPBundlePropExperimenter": {
               "data": [
                  1
               ],
               "exp_type": 1,
               "experimenter": 101,
               "length": 16,
               "type": 65535
            }
         },
         {
            "OFPBundlePropExperimenter": {
               "data": [
                  1,
                  2
               ],
               "exp_type": 2,
               "experimenter": 101,
               "length": 20,
               "type": 65535
            }
         }
      ]
   }
}
```

Set Asynchronous Configuration Message

class `ryu.ofproto.ofproto_v1_4_parser.`**`OFPSetAsync`**(*datapath*, *properties=None*)

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
properties	List of `OFPAsyncConfigProp` subclass instances

Example:

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

    properties = [
        ofp_parser.OFPAsyncConfigPropReasons(
            ofp.OFPACPT_PACKET_IN_SLAVE, 8,
            (1 << ofp.OFPR_APPLY_ACTION
             | 1 << ofp.OFPR_INVALID_TTL))]
    req = ofp_parser.OFPSetAsync(datapath, properties)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPSetAsync": {
      "properties": [
         {
            "OFPAsyncConfigPropReasons": {
               "length": 8,
               "mask": 3,
               "type": 0
            }
         },
         {
            "OFPAsyncConfigPropReasons": {
               "length": 8,
               "mask": 3,
               "type": 1
            }
         },
         {
            "OFPAsyncConfigPropReasons": {
               "length": 8,
               "mask": 3,
               "type": 2
            }
         },
         {
            "OFPAsyncConfigPropReasons": {
               "length": 8,
               "mask": 3,
               "type": 3
            }
         },
         {
            "OFPAsyncConfigPropReasons": {
               "length": 8,
               "mask": 3,
               "type": 4
            }
         },
         {
            "OFPAsyncConfigPropReasons": {
               "length": 8,
               "mask": 3,
               "type": 5
            }
         },
         {
            "OFPAsyncConfigPropReasons": {
               "length": 8,
               "mask": 3,
               "type": 6
            }
         },
         {
            "OFPAsyncConfigPropReasons": {
               "length": 8,
               "mask": 3,
               "type": 7
            }
         },
         {
            "OFPAsyncConfigPropReasons": {
               "length": 8,
               "mask": 24,
               "type": 8
            }
         },
         {
            "OFPAsyncConfigPropReasons": {
               "length": 8,
               "mask": 24,
               "type": 9
            }
         },
         {
            "OFPAsyncConfigPropReasons": {
               "length": 8,
               "mask": 3,
               "type": 10
            }
         },
         {
            "OFPAsyncConfigPropReasons": {
               "length": 8,
               "mask": 3,
               "type": 11
            }
         },
         {
            "OFPAsyncConfigPropExperimenter": {
               "data": [],
                    "exp_type": 0,
                    "experimenter": 101,
                    "length": 12,
                    "type": 65534
                }
            },
            {
                "OFPAsyncConfigPropExperimenter": {
                    "data": [
                        1
                    ],
                    "exp_type": 1,
                    "experimenter": 101,
                    "length": 16,
                    "type": 65535
                }
            },
            {
                "OFPAsyncConfigPropExperimenter": {
                    "data": [
                        1,
                        2
                    ],
                    "exp_type": 2,
                    "experimenter": 101,
                    "length": 20,
                    "type": 65535
                }
            }
        ]
    }
}
```

class `ryu.ofproto.ofproto_v1_4_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_4_parser.`**`OFPGetAsyncReply`**(*datapath*, *properties=None*)

Get asynchronous configuration reply message

The switch responds with this message to a get asynchronous configuration request.

Attribute	Description
properties	List of `OFPAsyncConfigProp` subclass instances

Example:

```
@set_ev_cls(ofp_event.EventOFPGetAsyncReply, MAIN_DISPATCHER)
def get_async_reply_handler(self, ev):
    msg = ev.msg

    self.logger.debug('OFPGetAsyncReply received: '
                      'properties=%s', repr(msg.properties))
```

JSON Example:

```
{
   "OFPGetAsyncReply": {
      "properties": [
         {
            "OFPAsyncConfigPropReasons": {
               "length": 8,
               "mask": 3,
               "type": 0
            }
         },
         {
            "OFPAsyncConfigPropReasons": {
               "length": 8,
               "mask": 3,
               "type": 1
            }
         },
         {
            "OFPAsyncConfigPropReasons": {
               "length": 8,
               "mask": 3,
               "type": 2
            }
         },
         {
            "OFPAsyncConfigPropReasons": {
               "length": 8,
               "mask": 3,
               "type": 3
            }
         },
         {
            "OFPAsyncConfigPropReasons": {
               "length": 8,
               "mask": 3,
               "type": 4
            }
         },
         {
            "OFPAsyncConfigPropReasons": {
               "length": 8,
               "mask": 3,
               "type": 5
            }
         },
         {
            "OFPAsyncConfigPropReasons": {
               "length": 8,
                "mask": 3,
                "type": 6
            }
        },
        {
            "OFPAsyncConfigPropReasons": {
                "length": 8,
                "mask": 3,
                "type": 7
            }
        },
        {
            "OFPAsyncConfigPropReasons": {
                "length": 8,
                "mask": 24,
                "type": 8
            }
        },
        {
            "OFPAsyncConfigPropReasons": {
                "length": 8,
                "mask": 24,
                "type": 9
            }
        },
        {
            "OFPAsyncConfigPropReasons": {
                "length": 8,
                "mask": 3,
                "type": 10
            }
        },
        {
            "OFPAsyncConfigPropReasons": {
                "length": 8,
                "mask": 3,
                "type": 11
            }
        },
        {
            "OFPAsyncConfigPropExperimenter": {
                "data": [],
                "exp_type": 0,
                "experimenter": 101,
                "length": 12,
                "type": 65534
            }
        },
        {
            "OFPAsyncConfigPropExperimenter": {
                "data": [
                    1
                ],
                "exp_type": 1,
                "experimenter": 101,
                "length": 16,
                "type": 65535
            }
```

```
          },
          {
             "OFPAsyncConfigPropExperimenter": {
                "data": [
                   1,
                   2
                ],
                "exp_type": 2,
                "experimenter": 101,
                "length": 20,
                "type": 65535
             }
          }
       ]
    }
}
```

Asynchronous Messages

Packet-In Message

class `ryu.ofproto.ofproto_v1_4_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_TABLE_MISS OFPR_APPLY_ACTION OFPR_INVALID_TTL OFPR_ACTION_SET OFPR_GROUP OFPR_PACKET_OUT
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
    dp = msg.datapath
    ofp = dp.ofproto
    if msg.reason == ofp.TABLE_MISS:
        reason = 'TABLE MISS'
    elif msg.reason == ofp.OFPR_APPLY_ACTION:
        reason = 'APPLY ACTION'
    elif msg.reason == ofp.OFPR_INVALID_TTL:
        reason = 'INVALID TTL'
    elif msg.reason == ofp.OFPR_ACTION_SET:
        reason = 'ACTION SET'
    elif msg.reason == ofp.OFPR_GROUP:
        reason = 'GROUP'
    elif msg.reason == ofp.OFPR_PACKET_OUT:
        reason = 'PACKET OUT'
    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": 3,
         "table_id": 1,
         "total_len": 42
      }
}
```

```
{
   "OFPPacketIn": {
      "buffer_id": 4026531840,
      "cookie": 283686884868096,
      "data": "",
      "match": {
         "OFPMatch": {
            "length": 329,
            "oxm_fields": [
               {
                    "OXMTlv": {
                        "field": "in_port",
                        "mask": null,
                        "value": 84281096
                    }
                },
                {
                    "OXMTlv": {
                        "field": "in_phy_port",
                        "mask": null,
                        "value": 16909060
                    }
                },
                {
                    "OXMTlv": {
                        "field": "metadata",
                        "mask": null,
                        "value": 283686952306183
                    }
                },
                {
                    "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": "vlan_vid",
                        "mask": null,
                        "value": 999
                    }
                },
                {
                    "OXMTlv": {
                        "field": "ip_dscp",
                        "mask": null,
                        "value": 9
                    }
                },
                {
                    "OXMTlv": {
                        "field": "ip_ecn",
                        "mask": null,
                        "value": 3
                    }
                },
                {
                    "OXMTlv": {
                        "field": "ip_proto",
                        "mask": null,
                        "value": 99
                    }
                },
                {
                    "OXMTlv": {
                        "field": "ipv4_src",
                        "mask": null,
                        "value": "1.2.3.4"
                    }
                },
                {
                    "OXMTlv": {
                        "field": "ipv4_dst",
                        "mask": null,
                        "value": "1.2.3.4"
                    }
                },
                {
                    "OXMTlv": {
                        "field": "tcp_src",
                        "mask": null,
                        "value": 8080
                    }
                },
                {
                    "OXMTlv": {
                        "field": "tcp_dst",
                        "mask": null,
                        "value": 18080
                    }
                },
                {
                    "OXMTlv": {
                        "field": "udp_src",
                        "mask": null,
                        "value": 28080
                    }
                },
                {
                    "OXMTlv": {
                        "field": "udp_dst",
                        "mask": null,
                        "value": 55936
                    }
                },
                {
                    "OXMTlv": {
                        "field": "sctp_src",
                        "mask": null,
                        "value": 48080
                }
            },
            {
                "OXMTlv": {
                    "field": "sctp_dst",
                    "mask": null,
                    "value": 59328
                }
            },
            {
                "OXMTlv": {
                    "field": "icmpv4_type",
                    "mask": null,
                    "value": 100
                }
            },
            {
                "OXMTlv": {
                    "field": "icmpv4_code",
                    "mask": null,
                    "value": 101
                }
            },
            {
                "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"
                }
            },
                {
                   "OXMTlv": {
                      "field": "ipv6_src",
                      "mask": null,
                      "value": "fe80::f00b:a4ff:fe48:28a5"
                   }
                },
                {
                   "OXMTlv": {
                      "field": "ipv6_dst",
                      "mask": null,
                      "value": "fe80::f00b:a4ff:fe05:b7dc"
                   }
                },
                {
                   "OXMTlv": {
                      "field": "ipv6_flabel",
                      "mask": null,
                      "value": 541473
                   }
                },
                {
                   "OXMTlv": {
                      "field": "icmpv6_type",
                      "mask": null,
                      "value": 200
                   }
                },
                {
                   "OXMTlv": {
                      "field": "icmpv6_code",
                      "mask": null,
                      "value": 201
                   }
                },
                {
                   "OXMTlv": {
                      "field": "ipv6_nd_target",
                      "mask": null,
                      "value": "fe80::a60:6eff:fe7f:74e7"
                   }
                },
                {
                   "OXMTlv": {
                      "field": "ipv6_nd_sll",
                      "mask": null,
                      "value": "00:00:00:00:02:9a"
                   }
                },
                {
                   "OXMTlv": {
                      "field": "ipv6_nd_tll",
                      "mask": null,
                      "value": "00:00:00:00:02:2b"
                   }
                },
                {
                   "OXMTlv": {
                        "field": "mpls_label",
                        "mask": null,
                        "value": 624485
                     }
                  },
                  {
                     "OXMTlv": {
                        "field": "mpls_tc",
                        "mask": null,
                        "value": 5
                     }
                  },
                  {
                     "OXMTlv": {
                        "field": "mpls_bos",
                        "mask": null,
                        "value": 1
                     }
                  },
                  {
                     "OXMTlv": {
                        "field": "pbb_isid",
                        "mask": null,
                        "value": 11259375
                     }
                  },
                  {
                     "OXMTlv": {
                        "field": "tunnel_id",
                        "mask": null,
                        "value": 651061555542690057
                     }
                  },
                  {
                     "OXMTlv": {
                        "field": "ipv6_exthdr",
                        "mask": null,
                        "value": 500
                     }
                  },
                  {
                     "OXMTlv": {
                        "field": "pbb_uca",
                        "mask": null,
                        "value": 1
                     }
                  }
               ],
               "type": 1
            }
         },
         "reason": 0,
         "table_id": 200,
         "total_len": 0
      }
}
```

Flow Removed Message

class `ryu.ofproto.ofproto_v1_4_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 OFPRR_METER_DELETE OFPRR_EVICTION
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_4_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": {
            "config": 0,
            "hw_addr": "f2:0b:a4:d0:3f:70",
            "length": 168,
            "name": "\u79c1\u306e\u30dd\u30fc\u30c8",
            "port_no": 7,
            "properties": [
               {
                  "OFPPortDescPropEthernet": {
                     "advertised": 10240,
                     "curr": 10248,
                     "curr_speed": 5000,
                     "length": 32,
                     "max_speed": 5000,
                     "peer": 10248,
                     "supported": 10248,
                     "type": 0
                  }
               },
               {
                  "OFPPortDescPropOptical": {
                     "length": 40,
                     "rx_grid_freq_lmda": 1500,
                     "rx_max_freq_lmda": 2000,
                        "rx_min_freq_lmda": 1000,
                        "supported": 1,
                        "tx_grid_freq_lmda": 1500,
                        "tx_max_freq_lmda": 2000,
                        "tx_min_freq_lmda": 1000,
                        "tx_pwr_max": 2000,
                        "tx_pwr_min": 1000,
                        "type": 1
                     }
                  },
                  {
                     "OFPPortDescPropExperimenter": {
                        "data": [],
                        "exp_type": 0,
                        "experimenter": 101,
                        "length": 12,
                        "type": 65535
                     }
                  },
                  {
                     "OFPPortDescPropExperimenter": {
                        "data": [
                           1
                        ],
                        "exp_type": 1,
                        "experimenter": 101,
                        "length": 16,
                        "type": 65535
                     }
                  },
                  {
                     "OFPPortDescPropExperimenter": {
                        "data": [
                           1,
                           2
                        ],
                        "exp_type": 2,
                        "experimenter": 101,
                        "length": 20,
                        "type": 65535
                     }
                  }
               ],
               "state": 4
            }
         },
         "reason": 0
      }
}
```

Controller Role Status Message

class `ryu.ofproto.ofproto_v1_4_parser.`**`OFPRoleStatus`**(*datapath*, *role=None*, *reason=None*, *generation_id=None*, *properties=None*)

Role status message

The switch notifies controller of change of role.

Attribute	Description
role	One of the following values. OFPCR_ROLE_NOCHANGE OFPCR_ROLE_EQUAL OFPCR_ROLE_MASTER
reason	One of the following values. OFPCRR_MASTER_REQUEST OFPCRR_CONFIG OFPCRR_EXPERIMENTER
generation_id	Master Election Generation ID
properties	List of `OFPRoleProp` subclass instance

Example:

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

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

    if msg.reason == ofp.OFPCRR_MASTER_REQUEST:
        reason = 'MASTER REQUEST'
    elif msg.reason == ofp.OFPCRR_CONFIG:
        reason = 'CONFIG'
    elif msg.reason == ofp.OFPCRR_EXPERIMENTER:
        reason = 'EXPERIMENTER'
    else:
        reason = 'unknown'

    self.logger.debug('OFPRoleStatus received: role=%s reason=%s '
                      'generation_id=%d properties=%s', role, reason,
                      msg.generation_id, repr(msg.properties))
```

JSON Example:

```
{
   "OFPRoleStatus": {
      "generation_id": 7,
      "properties": [
         {
            "OFPRolePropExperimenter": {
               "data": [],
                    "exp_type": 0,
                    "experimenter": 101,
                    "length": 12,
                    "type": 65535
                 }
              },
              {
                 "OFPRolePropExperimenter": {
                    "data": [
                       1
                    ],
                    "exp_type": 1,
                    "experimenter": 101,
                    "length": 16,
                    "type": 65535
                 }
              },
              {
                 "OFPRolePropExperimenter": {
                    "data": [
                       1,
                       2
                    ],
                    "exp_type": 2,
                    "experimenter": 101,
                    "length": 20,
                    "type": 65535
                 }
              }
           ],
           "reason": 0,
           "role": 2
     }
}
```

Table Status Message

class `ryu.ofproto.ofproto_v1_4_parser.`**`OFPTableStatus`**(*datapath*, *reason=None*, *table=None*)

Table status message

The switch notifies controller of change of table status.

Attribute	Description
reason	One of the following values. OFPTR_VACANCY_DOWN OFPTR_VACANCY_UP
table	`OFPTableDesc` instance

Example:

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

    if msg.reason == ofp.OFPTR_VACANCY_DOWN:
        reason = 'VACANCY_DOWN'
    elif msg.reason == ofp.OFPTR_VACANCY_UP:
        reason = 'VACANCY_UP'
    else:
        reason = 'unknown'

    self.logger.debug('OFPTableStatus received: reason=%s '
                      'table_id=%d config=0x%08x properties=%s',
                      reason, msg.table.table_id, msg.table.config,
                      repr(msg.table.properties))
```

JSON Example:

```
{
   "OFPTableStatus": {
      "reason": 3,
      "table": {
         "OFPTableDesc": {
            "config": 0,
            "length": 80,
            "properties": [
               {
                  "OFPTableModPropEviction": {
                     "flags": 0,
                     "length": 8,
                     "type": 2
                  }
               },
               {
                  "OFPTableModPropVacancy": {
                     "length": 8,
                     "type": 3,
                     "vacancy": 0,
                     "vacancy_down": 0,
                     "vacancy_up": 0
                  }
               },
               {
                  "OFPTableModPropExperimenter": {
                     "data": [],
                     "exp_type": 0,
                     "experimenter": 101,
                     "length": 12,
                     "type": 65535
                  }
               },
               {
                  "OFPTableModPropExperimenter": {
                     "data": [
                        1
                     ],
                     "exp_type": 1,
                     "experimenter": 101,
                     "length": 16,
                     "type": 65535
```

```
                    }
                },
                {
                    "OFPTableModPropExperimenter": {
                        "data": [
                            1,
                            2
                        ],
                        "exp_type": 2,
                        "experimenter": 101,
                        "length": 20,
                        "type": 65535
                    }
                }
            ],
            "table_id": 8
        }
    }
}
}
```

Request Forward Message

class `ryu.ofproto.ofproto_v1_4_parser.`**`OFPRequestForward`**(*datapath*, *request=None*)

Forwarded request message

The swtich forwards request messages from one controller to other controllers.

Attribute	Description
request	`OFPGroupMod` or `OFPMeterMod` instance

Example:

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

    if msg.request.msg_type == ofp.OFPT_GROUP_MOD:
        self.logger.debug(
            'OFPRequestForward received: request=OFPGroupMod('
            'command=%d, type=%d, group_id=%d, buckets=%s)',
            msg.request.command, msg.request.type,
            msg.request.group_id, msg.request.buckets)
    elif msg.request.msg_type == ofp.OFPT_METER_MOD:
        self.logger.debug(
            'OFPRequestForward received: request=OFPMeterMod('
            'command=%d, flags=%d, meter_id=%d, bands=%s)',
            msg.request.command, msg.request.flags,
            msg.request.meter_id, msg.request.bands)
    else:
        self.logger.debug(
            'OFPRequestForward received: request=Unknown')
```

JSON Example:

```
{
   "OFPRequestForward": {
      "request": {
         "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
         }
      }
   }
}
```

Symmetric Messages

Hello

class `ryu.ofproto.ofproto_v1_4_parser.`**`OFPHello`**(*datapath*, *elements=None*)

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_4_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_4_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_4_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_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=="
   }
}
```

Error Message

class `ryu.ofproto.ofproto_v1_4_parser.`**`OFPErrorMsg`**(*datapath*, *type_=None*, *code=None*, *data=None*, ***kwargs*)

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

If `type == OFPET_EXPERIMENTER`, this message has also the following attributes.

Attribute	Description
exp_type	Experimenter defined type
experimenter	Experimenter ID

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
   }
}
```

Experimenter

class `ryu.ofproto.ofproto_v1_4_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
   }
}
```

Port Structures

class `ryu.ofproto.ofproto_v1_4_parser.`**`OFPPort`**(*port_no=None*, *length=None*, *hw_addr=None*, *name=None*, *config=None*, *state=None*, *properties=None*)

Description of a port

Attribute	Description
port_no	Port number and it uniquely identifies a port within a switch.
length	Length of ofp_port (excluding padding).
hw_addr	MAC address for the port.
name	Null-terminated string containing a human-readable name for the interface.
config	Bitmap of port configration flags. OFPPC_PORT_DOWN OFPPC_NO_RECV OFPPC_NO_FWD OFPPC_NO_PACKET_IN
state	Bitmap of port state flags. OFPPS_LINK_DOWN OFPPS_BLOCKED OFPPS_LIVE
properties	List of `OFPPortDescProp` subclass instance

Flow Match Structure

class `ryu.ofproto.ofproto_v1_4_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.

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

Continued on next page

Table 2.3 – continued from previous page

Argument	Value	Description
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
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
pbb_uca	Integer 8bit	PBB UCA header field
tcp_flags	Integer 16bit	TCP flags (EXT-109 ONF Extension)
actset_output	Integer 32bit	Output port from action set metadata (EXT-233 ONF Extension)

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::')
```

Note: For the list of the supported Nicira experimenter matches, please refer to *ryu.ofproto.nx_match*.

Note: For VLAN id match field, special values are defined in OpenFlow Spec.

1. Packets with and without a VLAN tag

   - Example:

   ```
   match = parser.OFPMatch()
   ```

   - Packet Matching

non-VLAN-tagged	MATCH
VLAN-tagged(vlan_id=3)	MATCH
VLAN-tagged(vlan_id=5)	MATCH

2. Only packets without a VLAN tag

   - Example:

   ```
   match = parser.OFPMatch(vlan_vid=0x0000)
   ```

   - Packet Matching

non-VLAN-tagged	MATCH
VLAN-tagged(vlan_id=3)	x
VLAN-tagged(vlan_id=5)	x

3. Only packets with a VLAN tag regardless of its value

   - Example:

   ```
   match = parser.OFPMatch(vlan_vid=(0x1000, 0x1000))
   ```

   - Packet Matching

non-VLAN-tagged	x
VLAN-tagged(vlan_id=3)	MATCH
VLAN-tagged(vlan_id=5)	MATCH

4. Only packets with VLAN tag and VID equal

   - Example:

   ```
   match = parser.OFPMatch(vlan_vid=(0x1000 | 3))
   ```

   - Packet Matching

non-VLAN-tagged	x
VLAN-tagged(vlan_id=3)	MATCH
VLAN-tagged(vlan_id=5)	x

Flow Instruction Structures

class `ryu.ofproto.ofproto_v1_4_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_4_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_4_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_4_parser.`**`OFPInstructionMeter`**(*meter_id=1*, *type_=None*, *len_=None*)

Meter instruction

This instruction applies the meter.

Attribute	Description
meter_id	Meter instance

Action Structures

class `ryu.ofproto.ofproto_v1_4_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_4_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_4_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_4_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_4_parser.`**`OFPActionDecMplsTtl`**(*type_=None*, *len_=None*)
Decrement MPLS TTL action

This action decrements the MPLS TTL.

class `ryu.ofproto.ofproto_v1_4_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_4_parser.`**`OFPActionPopVlan`**(*type_=None*, *len_=None*)
Pop VLAN action

This action pops the outermost VLAN tag from the packet.

class `ryu.ofproto.ofproto_v1_4_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_4_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_4_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_4_parser.`**`OFPActionGroup`**(*group_id=0*, *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_4_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_4_parser.`**`OFPActionDecNwTtl`**(*type_=None*, *len_=None*)

Decrement IP TTL action

This action decrements the IP TTL.

class `ryu.ofproto.ofproto_v1_4_parser.`**`OFPActionSetField`**(*field=None*, ***kwargs*)

Set field action

This action modifies a header field in the packet.

The set of keywords available for this is same as OFPMatch.

Example:

```
set_field = OFPActionSetField(eth_src="00:00:00:00:00:00")
```

class `ryu.ofproto.ofproto_v1_4_parser.`**`OFPActionPushPbb`**(*ethertype*, *type_=None*, *len_=None*)

Push PBB action

This action pushes a new PBB header to the packet.

Attribute	Description
ethertype	Ether type

class `ryu.ofproto.ofproto_v1_4_parser.`**`OFPActionPopPbb`**(*type_=None*, *len_=None*)

Pop PBB action

This action pops the outermost PBB service instance header from the packet.

class `ryu.ofproto.ofproto_v1_4_parser.`**`OFPActionExperimenter`**(*experimenter*)

Experimenter action

This action is an extensible action for the experimenter.

Attribute	Description
experimenter	Experimenter ID

Note: For the list of the supported Nicira experimenter actions, please refer to *ryu.ofproto.nx_actions*.

2.5.6 OpenFlow v1.5 Messages and Structures

Controller-to-Switch Messages

Handshake

class `ryu.ofproto.ofproto_v1_5_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_5_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": 0,
      "capabilities": 79,
      "datapath_id": 1,
      "n_buffers": 255,
        "n_tables": 255
    }
}
```

Switch Configuration

class ryu.ofproto.ofproto_v1_5_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	Bitmap of the following flags. OFPC_FRAG_NORMAL OFPC_FRAG_DROP OFPC_FRAG_REASM
miss_send_len	Max bytes of new flow that datapath should send to the controller

Example:

```python
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_5_parser.**OFPGetConfigRequest**(*datapath*)

Get config request message

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

Example:

```python
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_5_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	Bitmap of the following flags. OFPC_FRAG_NORMAL OFPC_FRAG_DROP OFPC_FRAG_REASM
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
    flags = []

    if msg.flags & ofp.OFPC_FRAG_NORMAL:
        flags.append('NORMAL')
    if msg.flags & ofp.OFPC_FRAG_DROP:
        flags.append('DROP')
    if msg.flags & ofp.OFPC_FRAG_REASM:
        flags.append('REASM')
    self.logger.debug('OFPGetConfigReply received: '
                      'flags=%s miss_send_len=%d',
                      ','.join(flags), msg.miss_send_len)
```

JSON Example:

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

Modify State Messages

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPTableMod`**(*datapath*, *table_id*, *config*, *properties*)

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 configuration flags. OFPTC_EVICTION OFPTC_VACANCY_EVENTS
properties	List of `OFPTableModProp` subclass instance

Example:

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

    req = ofp_parser.OFPTableMod(datapath, 1, 3)
    flags = ofp.OFPTC_VACANCY_EVENTS
    properties = [ofp_parser.OFPTableModPropEviction(flags)]
    req = ofp_parser.OFPTableMod(datapath, 1, 3, properties)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPTableMod": {
      "config": 4,
      "properties": [
         {
            "OFPTableModPropEviction": {
               "flags": 2,
               "length": 8,
               "type": 2
            }
         }
      ],
      "table_id": 255
   }
}
```

class `ryu.ofproto.ofproto_v1_5_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, importance=0, match=None, instructions=None*)

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	Bitmap of the following flags. OFPFF_SEND_FLOW_REM OFPFF_CHECK_OVERLAP OFPFF_RESET_COUNTS OFPFF_NO_PKT_COUNTS OFPFF_NO_BYT_COUNTS
importance	Eviction precedence
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
    importance = 0
    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,
                                importance,
                                match, inst)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPFlowMod": {
      "buffer_id": 0,
      "command": 0,
      "cookie": 1311768467463790320,
      "cookie_mask": 18446744073709551615,
      "flags": 0,
      "hard_timeout": 0,
      "idle_timeout": 0,
      "importance": 39032,
      "instructions": [
         {
            "OFPInstructionActions": {
               "actions": [
                  {
                     "OFPActionPopVlan": {
                        "len": 8,
                        "type": 18
                     }
                  },
                  {
                     "OFPActionSetField": {
                        "field": {
                           "OXMTlv": {
                              "field": "ipv4_dst",
                              "mask": null,
                              "value": "192.168.2.9"
                           }
                        },
                        "len": 16,
                        "type": 25
                     }
                  },
                  {
                     "NXActionLearn": {
                        "cookie": 0,
                        "experimenter": 8992,
                        "fin_hard_timeout": 0,
                        "fin_idle_timeout": 0,
                        "flags": 0,
                        "hard_timeout": 300,
                        "idle_timeout": 0,
                        "len": 96,
                        "priority": 1,
                        "specs": [
                           {
                              "NXFlowSpecMatch": {
                                 "dst": [
                                    "vlan_vid",
                                    0
                                 ],
                                 "n_bits": 12,
                                 "src": [
                                    "vlan_vid",
                                    0
                                 ]
                              }
                           },
                           {
                              "NXFlowSpecMatch": {
                                 "dst": [
                                    "eth_dst_nxm",
                                    0
                                 ],
                                 "n_bits": 48,
                                 "src": [
                                    "eth_src_nxm",
                                    0
                                 ]
                              }
                           },
                           {
                              "NXFlowSpecLoad": {
                                 "dst": [
                                    "vlan_vid",
                                    0
                                 ],
                                 "n_bits": 12,
                                 "src": 0
                              }
                           },
                           {
                              "NXFlowSpecLoad": {
                                 "dst": [
                                    "tunnel_id_nxm",
                                    0
                                 ],
                                 "n_bits": 64,
                                 "src": [
                                    "tunnel_id_nxm",
                                    0
                                 ]
                              }
                           },
                           {
                              "NXFlowSpecOutput": {
                                 "dst": "",
                                 "n_bits": 32,
                                 "src": [
                                    "in_port",
                                    0
                                 ]
                              }
                           }
                        ],
                        "subtype": 16,
                        "table_id": 99,
                        "type": 65535
```

```
                     }
                  }
               ],
               "len": 128,
               "type": 4
            }
         },
         {
            "OFPInstructionGotoTable": {
               "len": 8,
               "table_id": 100,
               "type": 1
            }
         }
      ],
      "match": {
         "OFPMatch": {
            "length": 70,
            "oxm_fields": [
               {
                  "OXMTlv": {
                     "field": "in_port",
                     "mask": null,
                     "value": 43981
                  }
               },
               {
                  "OXMTlv": {
                     "field": "eth_dst",
                     "mask": null,
                     "value": "aa:bb:cc:99:88:77"
                  }
               },
               {
                  "OXMTlv": {
                     "field": "eth_type",
                     "mask": null,
                     "value": 2048
                  }
               },
               {
                  "OXMTlv": {
                     "field": "vlan_vid",
                     "mask": null,
                     "value": 5095
                  }
               },
               {
                  "OXMTlv": {
                     "field": "ipv4_dst",
                     "mask": null,
                     "value": "192.168.2.1"
                  }
               },
               {
                  "OXMTlv": {
                     "field": "tunnel_id",
                     "mask": null,
                        "value": 50000
                     }
                  },
                  {
                     "OXMTlv": {
                        "field": "tun_ipv4_src",
                        "mask": null,
                        "value": "192.168.2.3"
                     }
                  },
                  {
                     "OXMTlv": {
                        "field": "tun_ipv4_dst",
                        "mask": null,
                        "value": "192.168.2.4"
                     }
                  }
               ],
               "type": 1
            }
         },
         "out_group": 0,
         "out_port": 0,
         "priority": 0,
         "table_id": 2
      }
   }
```

```
{
   "OFPFlowMod": {
      "buffer_id": 0,
      "command": 0,
      "cookie": 1311768467463790320,
      "cookie_mask": 18446744073709551615,
      "flags": 0,
      "hard_timeout": 0,
      "idle_timeout": 0,
      "importance": 39032,
      "instructions": [
         {
            "OFPInstructionActions": {
               "actions": [
                  {
                     "NXActionConjunction": {
                        "clause": 1,
                        "experimenter": 8992,
                        "id": 11259375,
                        "len": 16,
                        "n_clauses": 2,
                        "subtype": 34,
                        "type": 65535
                     }
                  }
               ],
               "len": 24,
               "type": 4
            }
         }
      ],
      "match": {
         "OFPMatch": {
            "length": 70,
            "oxm_fields": [
               {
                  "OXMTlv": {
                     "field": "in_port",
                     "mask": null,
                     "value": 43981
                  }
               },
               {
                  "OXMTlv": {
                     "field": "eth_dst",
                     "mask": null,
                     "value": "aa:bb:cc:99:88:77"
                  }
               },
               {
                  "OXMTlv": {
                     "field": "eth_type",
                     "mask": null,
                     "value": 2048
                  }
               },
               {
                  "OXMTlv": {
                     "field": "vlan_vid",
                     "mask": null,
                     "value": 5095
                  }
               },
               {
                  "OXMTlv": {
                     "field": "ipv4_dst",
                     "mask": null,
                     "value": "192.168.2.1"
                  }
               },
               {
                  "OXMTlv": {
                     "field": "tunnel_id",
                     "mask": null,
                     "value": 50000
                  }
               },
               {
                  "OXMTlv": {
                     "field": "tun_ipv4_src",
                     "mask": null,
                     "value": "192.168.2.3"
                  }
               },
               {
                  "OXMTlv": {
                     "field": "tun_ipv4_dst",
                     "mask": null,
                    "value": "192.168.2.4"
                }
              }
            ],
            "type": 1
          }
        },
        "out_group": 0,
        "out_port": 0,
        "priority": 0,
        "table_id": 4
    }
}
```

```
{
   "OFPFlowMod": {
      "buffer_id": 0,
      "command": 0,
      "cookie": 1311768467463790320,
      "cookie_mask": 18446744073709551615,
      "flags": 0,
      "hard_timeout": 0,
      "idle_timeout": 0,
      "importance": 39032,
      "instructions": [
         {
            "OFPInstructionActions": {
               "actions": [
                  {
                     "OFPActionPopVlan": {
                        "len": 8,
                        "type": 18
                     }
                  },
                  {
                     "OFPActionSetField": {
                        "field": {
                           "OXMTlv": {
                              "field": "ipv4_dst",
                              "mask": null,
                              "value": "192.168.2.9"
                           }
                        },
                        "len": 16,
                        "type": 25
                     }
                  }
               ],
               "len": 32,
               "type": 4
            }
         },
         {
            "OFPInstructionGotoTable": {
               "len": 8,
               "table_id": 100,
               "type": 1
            }
            }
        ],
        "match": {
            "OFPMatch": {
                "length": 12,
                "oxm_fields": [
                    {
                        "OXMTlv": {
                            "field": "conj_id",
                            "mask": null,
                            "value": 11259375
                        }
                    }
                ],
                "type": 1
            }
        },
        "out_group": 0,
        "out_port": 0,
        "priority": 0,
        "table_id": 3
    }
}
```

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPGroupMod`**(*datapath*, *command=0*, *type_=0*, *group_id=0*, *command_bucket_id=0*, *buckets=None*, *properties=None*, *bucket_array_len=None*)

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 OFPGC_INSERT_BUCKET OFPGC_REMOVE_BUCKET
type	One of the following values. OFPGT_ALL OFPGT_SELECT OFPGT_INDIRECT OFPGT_FF
group_id	Group identifier.
command_bucket_id	Bucket Id used as part of OFPGC_INSERT_BUCKET and OFPGC_REMOVE_BUCKET commands execution.
buckets	List of `OFPBucket` instance
properties	List of `OFPGroupProp` instance

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

JSON Example:

```
{
    "OFPGroupMod": {
        "bucket_array_len": 56,
        "buckets": [
            {
                "OFPBucket": {
                    "action_array_len": 24,
                    "actions": [
                        {
                            "OFPActionPopVlan": {
                                "len": 8,
                                "type": 18
                            }
                        },
                        {
                            "OFPActionSetField": {
                                "field": {
                                    "OXMTlv": {
                                        "field": "ipv4_dst",
                                        "mask": null,
                                        "value": "192.168.2.9"
                                    }
                                },
                                "len": 16,
                                "type": 25
                            }
                        }
                    ],
                    "bucket_id": 305419896,
                    "len": 56,
                    "properties": [
                        {
                            "OFPGroupBucketPropWeight": {
                                "length": 8,
                                        "type": 0,
                                        "weight": 52428
                                    }
                                },
                                {
                                    "OFPGroupBucketPropWatch": {
                                        "length": 8,
                                        "type": 1,
                                        "watch": 56797
                                    }
                                },
                                {
                                    "OFPGroupBucketPropWatch": {
                                        "length": 8,
                                        "type": 2,
                                        "watch": 4008636142
                                    }
                                }
                            ]
                        }
                    }
                ],
                "command": 3,
                "command_bucket_id": 3149642683,
                "group_id": 2863311530,
                "properties": [],
                "type": 1
        }
}
```

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPPortMod`**(*datapath*, *port_no=0*, *hw_addr='00:00:00:00:00:00'*, *config=0*, *mask=0*, *properties=None*)

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
properties	List of `OFPPortModProp` subclass instance

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)
    properties = [ofp_parser.OFPPortModPropEthernet(advertise)]
    req = ofp_parser.OFPPortMod(datapath, port_no, hw_addr, config,
                                mask, properties)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPPortMod": {
      "config": 0,
      "hw_addr": "00:11:00:00:11:11",
      "mask": 0,
      "port_no": 1,
      "properties": [
         {
            "OFPPortModPropEthernet": {
               "advertise": 4096,
               "length": 8,
               "type": 0
            }
         },
         {
            "OFPPortModPropOptical": {
               "configure": 3,
               "fl_offset": 2000,
               "freq_lmda": 1500,
               "grid_span": 3000,
               "length": 24,
               "tx_pwr": 300,
               "type": 1
            }
         },
         {
            "OFPPortModPropExperimenter": {
               "data": [],
               "exp_type": 0,
               "experimenter": 101,
               "length": 12,
               "type": 65535
            }
         },
         {
            "OFPPortModPropExperimenter": {
               "data": [
                  1
               ],
               "exp_type": 1,
               "experimenter": 101,
               "length": 16,
                "type": 65535
            }
        },
        {
            "OFPPortModPropExperimenter": {
                "data": [
                    1,
                    2
                ],
                "exp_type": 2,
                "experimenter": 101,
                "length": 20,
                "type": 65535
            }
        }
    ]
  }
}
```

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPMeterMod`**(*datapath*, *command=0*, *flags=1*, *meter_id=1*, *bands=None*)

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	Bitmap 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
            }
         }
      ],
      "command": 0,
      "flags": 14,
      "meter_id": 100
   }
}
```

Multipart Messages

class `ryu.ofproto.ofproto_v1_5_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_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_5_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_5_parser.`**`OFPFlowDescStatsRequest`**(*datapath, flags=0, table_id=255, out_port=4294967295, out_group=4294967295, cookie=0, cookie_mask=0, match=None, type_=None*)

Individual flow descriptions request message

The controller uses this message to query individual flow descriptions.

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_desc_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.OFPFlowDescStatsRequest(datapath, 0,
                                             ofp.OFPTT_ALL,
                                             ofp.OFPP_ANY,
                                             ofp.OFPG_ANY,
                                             cookie, cookie_mask,
                                             match)
    datapath.send_msg(req)
```

JSON Example:

```
{
    "OFPFlowDescStatsRequest": {
        "cookie": 1234605616436508552,
        "cookie_mask": 18446744073709551615,
        "flags": 0,
        "match": {
            "OFPMatch": {
                "length": 12,
                "oxm_fields": [
                    {
                        "OXMTlv": {
                            "field": "in_port",
                            "mask": null,
                            "value": 1
                        }
                    }
                ],
                "type": 1
            }
        },
        "out_group": 4294967295,
        "out_port": 4294967295,
        "table_id": 1,
        "type": 1
    }
}
```

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPFlowDescStatsReply`**(*datapath*, *type_=None*, ***kwargs*)

Individual flow descriptions reply message

The switch responds with this message to an individual flow descriptions request.

Attribute	Description
body	List of `OFPFlowDesc` instance

Example:

```
@set_ev_cls(ofp_event.EventOFPFlowDescStatsReply, MAIN_DISPATCHER)
def flow_desc_reply_handler(self, ev):
    flows = []
    for stat in ev.msg.body:
        flows.append('table_id=%s priority=%d '
                     'idle_timeout=%d hard_timeout=%d flags=0x%04x '
                     'importance=%d cookie=%d match=%s '
                     'stats=%s instructions=%s' %
                     (stat.table_id, stat.priority,
                      stat.idle_timeout, stat.hard_timeout,
                      stat.flags, stat.importance,
                      stat.cookie, stat.match,
                      stat.stats, stat.instructions))
    self.logger.debug('FlowDesc: %s', flows)
```

JSON Example:

```
{
    "OFPFlowDescStatsReply": {
        "body": [
            {
                "OFPFlowDesc": {
                    "cookie": 1234605616436508552,
                    "flags": 1,
                    "hard_timeout": 255,
                    "idle_timeout": 255,
                    "importance": 43690,
                    "instructions": [
                        {
                            "OFPInstructionGotoTable": {
                                "len": 8,
                                "table_id": 2,
                                "type": 1
                            }
                        }
                    ],
                    "length": 64,
                    "match": {
                        "OFPMatch": {
                            "length": 12,
                            "oxm_fields": [
                                {
                                    "OXMTlv": {
                                        "field": "in_port",
                                        "mask": null,
                                        "value": 1
                                    }
                                }
                            ],
                            "type": 1
                        }
                    },
                    "priority": 5,
                    "stats": {
                        "OFPStats": {
                            "length": 12,
                            "oxs_fields": [
                                {
                                    "OXSTlv": {
                                        "field": "flow_count",
                                        "value": 1
                                    }
                                }
                            ]
                        }
                    },
                    "table_id": 1
                }
            }
        ],
        "flags": 0,
        "type": 1
    }
}
```

class `ryu.ofproto.ofproto_v1_5_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": 17
```

```
    }
}
```

class `ryu.ofproto.ofproto_v1_5_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 reason=%d priority=%d '
                     'match=%s stats=%s' %
                     (stat.table_id, stat.reason, stat.priority,
                      stat.match, stat.stats))
    self.logger.debug('FlowStats: %s', flows)
```

JSON Example:

```
{
    "OFPFlowStatsReply": {
        "body": [
            {
                "OFPFlowStats": {
                    "length": 40,
                    "match": {
                        "OFPMatch": {
                            "length": 12,
                            "oxm_fields": [
                                {
                                    "OXMTlv": {
                                        "field": "in_port",
                                        "mask": null,
                                        "value": 1
                                    }
                                }
                            ],
                            "type": 1
                        }
                    },
                    "priority": 1,
                    "reason": 0,
                    "stats": {
                        "OFPStats": {
                            "length": 12,
                            "oxs_fields": [
                                {
                                    "OXSTlv": {
                                        "field": "flow_count",
                                        "value": 1
                                    }
                                }
                            ]
                        }
                    },
                    "table_id": 1
                }
            }
        ],
        "flags": 0,
        "type": 17
    }
}
```

class `ryu.ofproto.ofproto_v1_5_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_5_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: stats=%s', body.stats)
```

JSON Example:

```
{
    "OFPAggregateStatsReply": {
        "body": {
            "OFPAggregateStats": {
                "length": 16,
                "stats": {
                    "OFPStats": {
                        "length": 12,
                        "oxs_fields": [
                            {
                                "OXSTlv": {
                                    "field": "flow_count",
                                    "value": 1
                                }
                            }
                        ]
                    }
                }
            }
        },
        "flags": 0,
        "type": 2
    }
}
```

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPPortStatsRequest`**(*datapath*, *flags*, *port_no*, *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_5_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(stat.length, stat.port_no,
                     stat.duration_sec, stat.duration_nsec,
                     stat.rx_packets, stat.tx_packets,
                     stat.rx_bytes, stat.tx_bytes,
                     stat.rx_dropped, stat.tx_dropped,
                     stat.rx_errors, stat.tx_errors,
                     repr(stat.properties))
    self.logger.debug('PortStats: %s', ports)
```

JSON Example:

```
{
   "OFPPortStatsReply": {
      "body": [
         {
            "OFPPortStats": {
               "duration_nsec": 0,
               "duration_sec": 0,
               "length": 224,
               "port_no": 7,
               "properties": [
                  {
                     "OFPPortStatsPropEthernet": {
                        "collisions": 0,
                        "length": 40,
                        "rx_crc_err": 0,
                        "rx_frame_err": 0,
                        "rx_over_err": 0,
                        "type": 0
                     }
                  },
                  {
                     "OFPPortStatsPropOptical": {
                        "bias_current": 300,
                        "flags": 3,
                        "length": 44,
                        "rx_freq_lmda": 1500,
                        "rx_grid_span": 500,
                        "rx_offset": 700,
                        "rx_pwr": 2000,
                        "temperature": 273,
                        "tx_freq_lmda": 1500,
                        "tx_grid_span": 500,
                        "tx_offset": 700,
                        "tx_pwr": 2000,
                        "type": 1
                     }
                  },
                  {
                     "OFPPortStatsPropExperimenter": {
                        "data": [],
                        "exp_type": 0,
                        "experimenter": 101,
                        "length": 12,
                        "type": 65535
                     }
                  },
                  {
                     "OFPPortStatsPropExperimenter": {
                        "data": [
                           1
                        ],
                        "exp_type": 1,
                        "experimenter": 101,
                        "length": 16,
                        "type": 65535
                     }
                  },
                  {
                     "OFPPortStatsPropExperimenter": {
                        "data": [
                           1,
                           2
                        ],
                        "exp_type": 2,
                        "experimenter": 101,
                        "length": 20,
                        "type": 65535
                     }
                  }
```

```
                ],
                "rx_bytes": 0,
                "rx_dropped": 0,
                "rx_errors": 0,
                "rx_packets": 0,
                "tx_bytes": 336,
                "tx_dropped": 0,
                "tx_errors": 0,
                "tx_packets": 4
            }
        },
        {
            "OFPPortStats": {
                "duration_nsec": 0,
                "duration_sec": 0,
                "length": 120,
                "port_no": 6,
                "properties": [
                    {
                        "OFPPortStatsPropEthernet": {
                            "collisions": 0,
                            "length": 40,
                            "rx_crc_err": 0,
                            "rx_frame_err": 0,
                            "rx_over_err": 0,
                            "type": 0
                        }
                    }
                ],
                "rx_bytes": 336,
                "rx_dropped": 0,
                "rx_errors": 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_5_parser.`**`OFPPortDescStatsRequest`**(*datapath*, *flags=0*, *port_no=4294967295*, *type_=None*)

Port description request message

The controller uses this message to query description of one or all the ports.

Attribute	Description
flags	Zero or `OFPMPF_REQ_MORE`
port_no	Port number to read (OFPP_ANY to all ports)

Example:

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

    req = ofp_parser.OFPPortDescStatsRequest(datapath, 0, ofp.OFPP_ANY)
    datapath.send_msg(req)
```

JSON Example:

```
{
    "OFPPortDescStatsRequest": {
        "flags": 0,
        "port_no": 48346,
        "type": 13
    }
}
```

class `ryu.ofproto.ofproto_v1_5_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 `OFPPort` 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 properties=%s' %
                     (p.port_no, p.hw_addr,
                      p.name, p.config, p.state, repr(p.properties)))
    self.logger.debug('OFPPortDescStatsReply received: %s', ports)
```

JSON Example:

```
{
   "OFPPortDescStatsReply": {
      "body": [
         {
            "OFPPort": {
               "config": 0,
               "hw_addr": "f2:0b:a4:d0:3f:70",
               "length": 168,
               "name": "Port7",
               "port_no": 7,
               "properties": [
                  {
                     "OFPPortDescPropEthernet": {
                        "advertised": 10240,
                        "curr": 10248,
                        "curr_speed": 5000,
                        "length": 32,
                        "max_speed": 5000,
                        "peer": 10248,
                        "supported": 10248,
                        "type": 0
                     }
                  },
                  {
                     "OFPPortDescPropOptical": {
                        "length": 40,
                        "rx_grid_freq_lmda": 1500,
                        "rx_max_freq_lmda": 2000,
                        "rx_min_freq_lmda": 1000,
                        "supported": 1,
                        "tx_grid_freq_lmda": 1500,
                        "tx_max_freq_lmda": 2000,
                        "tx_min_freq_lmda": 1000,
                        "tx_pwr_max": 2000,
                        "tx_pwr_min": 1000,
                        "type": 1
                     }
                  },
                  {
                     "OFPPortDescPropExperimenter": {
                        "data": [],
                        "exp_type": 0,
                        "experimenter": 101,
                        "length": 12,
                        "type": 65535
                     }
                  },
                  {
                     "OFPPortDescPropExperimenter": {
                        "data": [
                           1
                        ],
                        "exp_type": 1,
                        "experimenter": 101,
                        "length": 16,
                        "type": 65535
                     }
                  },
                  {
                     "OFPPortDescPropExperimenter": {
                        "data": [
                           1,
                           2
                        ],
                        "exp_type": 2,
                        "experimenter": 101,
                        "length": 20,
                        "type": 65535
                     }
                  }
               ],
               "state": 4
            }
         },
         {
            "OFPPort": {
               "config": 0,
                    "hw_addr": "f2:0b:a4:7d:f8:ea",
                    "length": 72,
                    "name": "Port6",
                    "port_no": 6,
                    "properties": [
                       {
                          "OFPPortDescPropEthernet": {
                             "advertised": 10240,
                             "curr": 10248,
                             "curr_speed": 5000,
                             "length": 32,
                             "max_speed": 5000,
                             "peer": 10248,
                             "supported": 10248,
                             "type": 0
                          }
                       }
                    ],
                    "state": 4
                 }
              }
           ],
           "flags": 0,
           "type": 13
     }
}
```

class `ryu.ofproto.ofproto_v1_5_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": 43981,
      "queue_id": 4294967295,
      "type": 5
```

```
    }
}
```

class ryu.ofproto.ofproto_v1_5_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'
                      'properties=%s' %
                      (stat.port_no, stat.queue_id,
                       stat.tx_bytes, stat.tx_packets, stat.tx_errors,
                       stat.duration_sec, stat.duration_nsec,
                       repr(stat.properties)))
    self.logger.debug('QueueStats: %s', queues)
```

JSON Example:

```
{
   "OFPQueueStatsReply": {
      "body": [
         {
            "OFPQueueStats": {
               "duration_nsec": 0,
               "duration_sec": 0,
               "length": 104,
               "port_no": 7,
               "properties": [
                  {
                     "OFPQueueStatsPropExperimenter": {
                        "data": [],
                        "exp_type": 0,
                        "experimenter": 101,
                        "length": 12,
                        "type": 65535
                     }
                  },
                  {
                     "OFPQueueStatsPropExperimenter": {
                        "data": [
                           1
                        ],
                        "exp_type": 1,
                        "experimenter": 101,
                        "length": 16,
                        "type": 65535
                     }
```

```
                    },
                    {
                        "OFPQueueStatsPropExperimenter": {
                            "data": [
                                1,
                                2
                            ],
                            "exp_type": 2,
                            "experimenter": 101,
                            "length": 20,
                            "type": 65535
                        }
                    }
                ],
                "queue_id": 1,
                "tx_bytes": 0,
                "tx_errors": 0,
                "tx_packets": 0
            }
        },
        {
            "OFPQueueStats": {
                "duration_nsec": 0,
                "duration_sec": 0,
                "length": 48,
                "port_no": 6,
                "properties": [],
                "queue_id": 1,
                "tx_bytes": 0,
                "tx_errors": 0,
                "tx_packets": 0
            }
        },
        {
            "OFPQueueStats": {
                "duration_nsec": 0,
                "duration_sec": 0,
                "length": 48,
                "port_no": 7,
                "properties": [],
                "queue_id": 2,
                "tx_bytes": 0,
                "tx_errors": 0,
                "tx_packets": 0
            }
        }
    ],
    "flags": 0,
    "type": 5
  }
}
```

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPQueueDescStatsRequest`**(*datapath*, *flags=0*, *port_no=4294967295*, *queue_id=4294967295*, *type_=None*)

Queue description request message

The controller uses this message to query description of all the queues.

Attribute	Description
flags	Zero or `OFPMPF_REQ_MORE`
port_no	Port number to read (OFPP_ANY for all ports)
queue_id	ID of queue to read (OFPQ_ALL for all queues)

Example:

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

    req = ofp_parser.OFPQueueDescStatsRequest(datapath, 0,
                                              ofp.OFPP_ANY,
                                              ofp.OFPQ_ALL)
    datapath.send_msg(req)
```

JSON Example:

```
{
    "OFPQueueDescStatsRequest": {
        "flags": 0,
        "port_no": 52651,
        "queue_id": 57020,
        "type": 15
    }
}
```

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPQueueDescStatsReply`**(*datapath*, *type_=None*, ***kwargs*)

Queue description reply message

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

Attribute	Description
body	List of `OFPQueueDesc` instance

Example:

```
@set_ev_cls(ofp_event.EventOFPQueueDescStatsReply, MAIN_DISPATCHER)
def queue_desc_stats_reply_handler(self, ev):
    queues = []
    for q in ev.msg.body:
        queues.append('port_no=%d queue_id=0x%08x properties=%s' %
                      (q.port_no, q.queue_id, repr(q.properties)))
    self.logger.debug('OFPQueueDescStatsReply received: %s', queues)
```

JSON Example:

```
{
   "OFPQueueDescStatsReply": {
      "body": [
         {
            "OFPQueueDesc": {
               "len": 32,
               "port_no": 7,
               "properties": [
                  {
                     "OFPQueueDescPropExperimenter": {
                        "data": [],
                        "exp_type": 0,
                        "experimenter": 101,
                        "length": 12,
                        "type": 65535
                    }
                }
            ],
            "queue_id": 0
        }
    },
    {
        "OFPQueueDesc": {
            "len": 88,
            "port_no": 8,
            "properties": [
                {
                    "OFPQueueDescPropMinRate": {
                        "length": 8,
                        "rate": 300,
                        "type": 1
                    }
                },
                {
                    "OFPQueueDescPropMaxRate": {
                        "length": 8,
                        "rate": 900,
                        "type": 2
                    }
                },
                {
                    "OFPQueueDescPropExperimenter": {
                        "data": [],
                        "exp_type": 0,
                        "experimenter": 101,
                        "length": 12,
                        "type": 65535
                    }
                },
                {
                    "OFPQueueDescPropExperimenter": {
                        "data": [
                            1
                        ],
                        "exp_type": 1,
                        "experimenter": 101,
                        "length": 16,
                        "type": 65535
                    }
                },
                {
                    "OFPQueueDescPropExperimenter": {
                        "data": [
                            1,
                            2
                        ],
                        "exp_type": 2,
                        "experimenter": 101,
                        "length": 20,
                        "type": 65535
                     }
                  }
               ],
               "queue_id": 1
            }
         }
      ],
      "flags": 0,
      "type": 15
   }
}
```

class `ryu.ofproto.ofproto_v1_5_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:

```python
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)
```

JSON Example:

```json
{
   "OFPGroupStatsRequest": {
      "flags": 0,
      "group_id": 4294967292,
      "type": 6
   }
}
```

class `ryu.ofproto.ofproto_v1_5_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:

```python
@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)
```

JSON Example:

```
{
   "OFPGroupStatsReply": {
      "body": [
         {
            "OFPGroupStats": {
               "bucket_stats": [
                  {
                     "OFPBucketCounter": {
                        "byte_count": 2345,
                        "packet_count": 234
                     }
                  }
               ],
               "byte_count": 12345,
               "duration_nsec": 609036000,
               "duration_sec": 9,
               "group_id": 1,
               "length": 56,
               "packet_count": 123,
               "ref_count": 2
            }
         }
      ],
      "flags": 0,
      "type": 6
   }
}
```

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPGroupDescStatsRequest`**(*datapath*, *flags=0*, *group_id=4294967292*, *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`
group_id	ID of group to read (OFPG_ALL to all groups)

Example:

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

    req = ofp_parser.OFPGroupDescStatsRequest(datapath, 0, ofp.OFPG_ALL)
    datapath.send_msg(req)
```

JSON Example:

```
{
    "OFPGroupDescStatsRequest": {
        "flags": 0,
        "group_id": 52651,
        "type": 7
    }
}
```

class `ryu.ofproto.ofproto_v1_5_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 properties=%s' %
                     (stat.length, stat.type, stat.group_id,
                      stat.bucket, repr(stat.properties)))
    self.logger.debug('GroupDescStats: %s', descs)
```

JSON Example:

```
{
    "OFPGroupDescStatsReply": {
        "body": [
            {
                "OFPGroupDescStats": {
                    "bucket_array_len": 32,
                    "buckets": [
                        {
                            "OFPBucket": {
                                "action_array_len": 16,
                                "actions": [
                                    {
                                        "OFPActionOutput": {
                                            "len": 16,
                                            "max_len": 65509,
                                            "port": 1,
                                            "type": 0
                                        }
                                    }
                                ],
                                "bucket_id": 65535,
                                "len": 32,
                                "properties": [
                                    {
                                        "OFPGroupBucketPropWeight": {
                                            "length": 8,
                                            "type": 0,
                                            "weight": 65535
                                        }
                                    }
                                ]
                            }
                        }
                    ],
                    "group_id": 1,
                    "length": 48,
                    "properties": [],
                    "type": 1
                }
            }
        ],
        "flags": 0,
        "type": 7
    }
}
```

class `ryu.ofproto.ofproto_v1_5_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_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_5_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_5_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:

```
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_5_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 ref_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.ref_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,
                    "ref_count": 0,
                    "len": 56,
                    "meter_id": 100,
                    "packet_in_count": 0
                }
            }
        ],
        "flags": 0,
        "type": 9
    }
}
```

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPMeterDescStatsRequest`**(*datapath*, *flags=0*, *meter_id=4294967295*, *type_=None*)

Meter description 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_desc_stats_request(self, datapath):
    ofp = datapath.ofproto
    ofp_parser = datapath.ofproto_parser

    req = ofp_parser.OFPMeterDescStatsRequest(datapath, 0,
                                              ofp.OFPM_ALL)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPMeterDescStatsRequest": {
      "flags": 0,
      "meter_id": 4294967295,
      "type": 10
   }
}
```

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPMeterDescStatsReply`**(*datapath*, *type_=None*, ***kwargs*)

Meter description statistics reply message

The switch responds with this message to a meter description statistics request.

Attribute	Description
body	List of `OFPMeterDescStats` instance

Example:

```
@set_ev_cls(ofp_event.EventOFPMeterDescStatsReply, MAIN_DISPATCHER)
def meter_desc_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('MeterDescStats: %s', configs)
```

JSON Example:

```
{
   "OFPMeterDescStatsReply": {
      "body": [
         {
            "OFPMeterDescStats": {
                "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_5_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_5_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_bands=%d '
                        'max_color=%d' %
                        (stat.max_meter, stat.band_types,
                         stat.capabilities, stat.max_bands,
                         stat.max_color))
    self.logger.debug('MeterFeaturesStats: %s', features)
```

JSON Example:

```
{
   "OFPMeterFeaturesStatsReply": {
      "body": [
         {
            "OFPMeterFeaturesStats": {
               "band_types": 2147483654,
               "capabilities": 15,
               "features": 3,
               "max_bands": 255,
               "max_color": 0,
               "max_meter": 16777216
            }
         }
      ],
      "flags": 0,
      "type": 11
   }
}
```

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPControllerStatusStatsRequest`**(*datapath*, *flags=0*, *type_=None*)

Controller status multipart request message

The controller uses this message to request the status, the roles and the control channels of other controllers configured on the switch.

Attribute	Description
flags	Zero or `OFPMPF_REQ_MORE`

Example:

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

    req = ofp_parser.OFPPortDescStatsRequest(datapath, 0)
    datapath.send_msg(req)
```

JSON Example:

```
{
    "OFPControllerStatusStatsRequest": {
        "flags": 0,
        "type": 18
    }
}
```

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPControllerStatusStatsReply`**(*datapath*, *type_=None*, ***kwargs*)

Controller status multipart reply message

The switch responds with this message to a controller status multipart request.

Attribute	Description
body	List of `OFPControllerStatus` instance

Example:

```
@set_ev_cls(ofp_event.EventOFPControllerStatusStatsReply,
            MAIN_DISPATCHER)
def controller_status_multipart_reply_handler(self, ev):
    status = []
    for s in ev.msg.body:
        status.append('short_id=%d role=%d reason=%d '
                      'channel_status=%d properties=%s' %
                      (s.short_id, s.role, s.reason,
                       s.channel_status, repr(s.properties)))
    self.logger.debug('OFPControllerStatusStatsReply received: %s',
                      status)
```

JSON Example:

```
{
    "OFPControllerStatusStatsReply": {
        "body": [
            {
                "OFPControllerStatusStats": {
                    "channel_status": 1,
                    "length": 48,
                    "properties": [
                        {
                            "OFPControllerStatusPropUri": {
                                "length": 26,
                                "type": 0,
                                "uri": "tls:192.168.34.23:6653"
                            }
                        }
                    ],
                    "reason": 1,
                    "role": 1,
                    "short_id": 65535
                }
            }
        ],
        "flags": 0,
        "type": 18
    }
}
```

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPTableStatsRequest`**(*datapath*, *flags*, *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_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_5_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_5_parser.`**`OFPTableDescStatsRequest`**(*datapath*, *flags=0*, *type_=None*)

Table description request message

The controller uses this message to query description of all the tables.

Attribute	Description
flags	Zero or `OFPMPF_REQ_MORE`

Example:

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

    req = ofp_parser.OFPTableDescStatsRequest(datapath, 0)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPTableDescStatsRequest": {
      "flags": 0,
      "type": 14
   }
}
```

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPTableDescStatsReply`**(*datapath*, *type_=None*, ***kwargs*)

Table description reply message

The switch responds with this message to a table description request.

Attribute	Description
body	List of `OFPTableDesc` instance

Example:

```python
@set_ev_cls(ofp_event.EventOFPTableDescStatsReply, MAIN_DISPATCHER)
def table_desc_stats_reply_handler(self, ev):
    tables = []
    for p in ev.msg.body:
        tables.append('table_id=%d config=0x%08x properties=%s' %
                      (p.table_id, p.config, repr(p.properties)))
    self.logger.debug('OFPTableDescStatsReply received: %s', tables)
```

JSON Example:

```
{
   "OFPTableDescStatsReply": {
      "body": [
         {
            "OFPTableDesc": {
               "config": 0,
               "length": 24,
               "properties": [
                  {
                     "OFPTableModPropExperimenter": {
                        "data": [],
                        "exp_type": 0,
                        "experimenter": 101,
                        "length": 12,
                        "type": 65535
                     }
                  }
               ],
               "table_id": 7
            }
         },
         {
            "OFPTableDesc": {
               "config": 0,
               "length": 80,
               "properties": [
                  {
                     "OFPTableModPropEviction": {
                        "flags": 0,
                        "length": 8,
                        "type": 2
                     }
                  },
                  {
                     "OFPTableModPropVacancy": {
                        "length": 8,
                        "type": 3,
                        "vacancy": 0,
                        "vacancy_down": 0,
                        "vacancy_up": 0
                     }
                  },
                  {
                     "OFPTableModPropExperimenter": {
                        "data": [],
                        "exp_type": 0,
                        "experimenter": 101,
                        "length": 12,
                        "type": 65535
                     }
                  },
                  {
                     "OFPTableModPropExperimenter": {
                        "data": [
                           1
                        ],
                        "exp_type": 1,
                        "experimenter": 101,
                        "length": 16,
                        "type": 65535
                     }
                  },
                  {
                     "OFPTableModPropExperimenter": {
                        "data": [
                           1,
                           2
                        ],
                        "exp_type": 2,
                        "experimenter": 101,
                        "length": 20,
                        "type": 65535
                     }
                  }
               ],
               "table_id": 8
            }
         }
      ],
      "flags": 0,
      "type": 14
   }
}
```

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPTableFeaturesStatsRequest`**(*datapath*, *flags=0*, *body=None*, *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 [].

JSON Example:

```
{
    "OFPTableFeaturesStatsRequest": {
        "body": [
            {
                "OFPTableFeaturesStats": {
                    "capabilities": 4,
                    "command": 1,
                    "features": 1,
                    "length": 80,
                    "max_entries": 255,
                    "metadata_match": 18446744073709551615,
                    "metadata_write": 18446744073709551615,
                    "name": "table1",
                    "properties": [
                        {
                            "OFPTableFeaturePropOxmValues": {
                                "length": 14,
                                "oxm_values": [
                                    {
                                        "OXMTlv": {
```

```
                                                    "field": "eth_src",
                                                    "mask": null,
                                                    "value": "aa:bb:cc:dd:ee:ff"
                                                }
                                            }
                                        ],
                                        "type": 22
                                    }
                                }
                            ],
                            "table_id": 1
                        }
                    }
                ],
                "flags": 0,
                "type": 12
            }
        }
```

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPTableFeaturesStatsReply`**(*datapath*, *type_=None*, ***kwargs*)

Table features statistics reply message

The switch responds with this message to a table features statistics request.

Attribute	Description
body	List of `OFPTableFeaturesStats` instance

JSON Example:

```
{
   "OFPTableFeaturesStatsReply": {
      "body": [
         {
            "OFPTableFeaturesStats": {
               "capabilities": 4,
               "command": 1,
               "features": 1,
               "length": 80,
               "max_entries": 255,
               "metadata_match": 18446744073709551615,
               "metadata_write": 18446744073709551615,
               "name": "table1",
               "properties": [
                  {
                     "OFPTableFeaturePropOxmValues": {
                        "length": 14,
                        "oxm_values": [
                           {
                              "OXMTlv": {
                                 "field": "eth_src",
                                 "mask": null,
                                 "value": "aa:bb:cc:dd:ee:ff"
                              }
                           }
                        ],
                        "type": 22
                     }
```

```
                    }
                ],
                "table_id": 1
            }
        }
    ],
    "flags": 0,
    "type": 12
  }
}
```

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPFlowMonitorRequest`**(*datapath*, *flags=0*, *monitor_id=0*, *out_port=4294967295*, *out_group=4294967295*, *monitor_flags=0*, *table_id=255*, *command=0*, *match=None*, *type_=None*)

Flow monitor request message

The controller uses this message to query flow monitors.

Attribute	Description
flags	Zero or `OFPMPF_REQ_MORE`
monitor_id	Controller-assigned ID for this monitor
out_port	Require matching entries to include this as an output port
out_group	Require matching entries to include this as an output group
monitor_flags	Bitmap of the following flags. OFPFMF_INITIAL OFPFMF_ADD OFPFMF_REMOVED OFPFMF_MODIFY OFPFMF_INSTRUCTIONS OFPFMF_NO_ABBREV OFPFMF_ONLY_OWN
table_id	ID of table to monitor
command	One of the following values. OFPFMC_ADD OFPFMC_MODIFY OFPFMC_DELETE
match	Instance of `OFPMatch`

Example:

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

    monitor_flags = [ofp.OFPFMF_INITIAL, ofp.OFPFMF_ONLY_OWN]
    match = ofp_parser.OFPMatch(in_port=1)
    req = ofp_parser.OFPFlowMonitorRequest(datapath, 0, 10000,
                                           ofp.OFPP_ANY, ofp.OFPG_ANY,
                                           monitor_flags,
                                           ofp.OFPTT_ALL,
                                           ofp.OFPFMC_ADD, match)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPFlowMonitorRequest": {
      "command": 0,
      "flags": 0,
      "match": {
         "OFPMatch": {
            "length": 14,
            "oxm_fields": [
               {
                  "OXMTlv": {
                     "field": "eth_dst",
                     "mask": null,
                     "value": "f2:0b:a4:7d:f8:ea"
                  }
               }
            ],
            "type": 1
         }
      },
      "monitor_flags": 15,
      "monitor_id": 100000000,
      "out_group": 4294967295,
      "out_port": 22,
      "table_id": 33,
      "type": 16
   }
}
```

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPFlowMonitorReply`**(*datapath*, *type_=None*, ***kwargs*)

Flow monitor reply message

The switch responds with this message to a flow monitor request.

Attribute	Description
body	List of list of the following class instance. OFPFlowMonitorFull OFPFlowMonitorAbbrev OFPFlowMonitorPaused

Example:

```
@set_ev_cls(ofp_event.EventOFPFlowMonitorReply, MAIN_DISPATCHER)
def flow_monitor_reply_handler(self, ev):
    msg = ev.msg
    dp = msg.datapath
    ofp = dp.ofproto
    flow_updates = []

    for update in msg.body:
        update_str = 'length=%d event=%d' %
                     (update.length, update.event)
        if (update.event == ofp.OFPFME_INITIAL or
            update.event == ofp.OFPFME_ADDED or
            update.event == ofp.OFPFME_REMOVED or
            update.event == ofp.OFPFME_MODIFIED):
            update_str += 'table_id=%d reason=%d idle_timeout=%d '
                          'hard_timeout=%d priority=%d cookie=%d '
                          'match=%d instructions=%s' %
                          (update.table_id, update.reason,
                           update.idle_timeout, update.hard_timeout,
                           update.priority, update.cookie,
                           update.match, update.instructions)
        elif update.event == ofp.OFPFME_ABBREV:
            update_str += 'xid=%d' % (update.xid)
        flow_updates.append(update_str)
    self.logger.debug('FlowUpdates: %s', flow_updates)
```

JSON Example:

```
{
   "OFPFlowMonitorReply": {
      "body": [
         {
            "OFPFlowUpdateFull": {
               "cookie": 0,
               "event": 0,
               "hard_timeout": 700,
               "idle_timeout": 600,
               "instructions": [
                  {
                     "OFPInstructionActions": {
                        "actions": [
                           {
                              "OFPActionOutput": {
                                 "len": 16,
                                 "max_len": 0,
                                 "port": 4294967290,
                                 "type": 0
                              }
                           }
                        ],
                        "len": 24,
                        "type": 4
                     }
                  }
               ],
               "length": 64,
               "match": {
                  "OFPMatch": {
                     "length": 10,
                     "oxm_fields": [
                        {
                            "OXMTlv": {
                               "field": "eth_type",
                               "mask": null,
                               "value": 2054
                            }
                         }
                      ],
                      "type": 1
                   }
                },
                "priority": 3,
                "reason": 0,
                "table_id": 0
             }
          },
          {
             "OFPFlowUpdateAbbrev": {
                "event": 4,
                "length": 8,
                "xid": 1234
             }
          },
          {
             "OFPFlowUpdatePaused": {
                "event": 5,
                "length": 8
             }
          }
       ],
       "flags": 0,
       "type": 16
    }
}
```

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPBundleFeaturesStatsRequest`**(*datapath*, *flags=0*, *feature_request_flags=0*, *properties=None*, *type_=None*)

Bundle features request message

The controller uses this message to query a switch about its bundle capabilities, including whether it supports atomic bundles, ordered bundles, and scheduled bundles.

Attribute	Description
flags	Zero or `OFPMPF_REQ_MORE`
feature_request_flags	Bitmap of the following flags. OFPBF_TIMESTAMP OFPBF_TIME_SET_SCHED
properties	List of `OFPBundleFeaturesProp` subclass instance

Example:

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

    req = ofp_parser.OFPBundleFeaturesStatsRequest(datapath, 0)
    datapath.send_msg(req)
```

JSON Example:

```
{
    "OFPBundleFeaturesStatsRequest": {
        "feature_request_flags": 3,
        "flags": 0,
        "properties": [
            {
                "OFPBundleFeaturesPropTime": {
                    "length": 72,
                    "sched_accuracy": {
                        "OFPTime": {
                            "nanoseconds": 1717986918,
                            "seconds": 6148914691236517205
                        }
                    },
                    "sched_max_future": {
                        "OFPTime": {
                            "nanoseconds": 2290649224,
                            "seconds": 8608480567731124087
                        }
                    },
                    "sched_max_past": {
                        "OFPTime": {
                            "nanoseconds": 2863311530,
                            "seconds": 11068046444225730969
                        }
                    },
                    "timestamp": {
                        "OFPTime": {
                            "nanoseconds": 3435973836,
                            "seconds": 13527612320720337851
                        }
                    },
                    "type": 1
                }
            }
        ],
        "type": 19
    }
}
```

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPBundleFeaturesStatsReply`**(*datapath*, *type_=None*, ***kwargs*)

Bundle features reply message

The switch responds with this message to a bundle features request.

Attribute	Description
body	Instance of `OFPBundleFeaturesStats`

Example:

```
@set_ev_cls(ofp_event.EventOFPBundleFeaturesStatsReply, MAIN_DISPATCHER)
def bundle_features_stats_reply_handler(self, ev):
    body = ev.msg.body

    self.logger.debug('OFPBundleFeaturesStats: capabilities=%0x%08x '
                      'properties=%s',
                      body.capabilities, repr(body.properties))
```

JSON Example:

```
{
    "OFPBundleFeaturesStatsReply": {
        "body": {
            "OFPBundleFeaturesStats": {
                "capabilities": 7,
                "properties": [
                    {
                        "OFPBundleFeaturesPropTime": {
                            "length": 72,
                            "sched_accuracy": {
                                "OFPTime": {
                                    "nanoseconds": 1717986918,
                                    "seconds": 6148914691236517205
                                }
                            },
                            "sched_max_future": {
                                "OFPTime": {
                                    "nanoseconds": 2290649224,
                                    "seconds": 8608480567731124087
                                }
                            },
                            "sched_max_past": {
                                "OFPTime": {
                                    "nanoseconds": 2863311530,
                                    "seconds": 11068046444225730969
                                }
                            },
                            "timestamp": {
                                "OFPTime": {
                                    "nanoseconds": 3435973836,
                                    "seconds": 13527612320720337851
                                }
                            },
                            "type": 1
                        }
                    }
                ]
            }
        },
        "flags": 0,
        "type": 19
    }
}
```

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPExperimenterStatsRequest`**(*datapath*, *flags*, *experimenter*, *exp_type*, *data*, *type_=None*)

Experimenter multipart request message

Attribute	Description
flags	Zero or `OFPMPF_REQ_MORE`
experimenter	Experimenter ID
exp_type	Experimenter defined
data	Experimenter defined additional data

JSON Example:

```
{
   "OFPExperimenterStatsRequest": {
      "data": "aG9nZWhvZ2U=",
      "exp_type": 3405678728,
      "experimenter": 3735928495,
      "flags": 0,
      "type": 65535
   }
}
```

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPExperimenterStatsReply`**(*datapath*, *type_=None*, ***kwargs*)

Experimenter multipart reply message

Attribute	Description
body	An `OFPExperimenterMultipart` instance

JSON Example:

```
{
   "OFPExperimenterStatsReply": {
      "body": {
         "OFPExperimenterMultipart": {
            "data": "dGVzdGRhdGE5OTk5OTk5OQ==",
            "exp_type": 3405674359,
            "experimenter": 3735928495
         }
      },
      "flags": 0,
      "type": 65535
   }
}
```

Packet-Out Message

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPPacketOut`**(*datapath*, *buffer_id=None*, *match=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)
match	Instance of `OFPMatch` (`in_port` is mandatory in the match field)
actions	list of OpenFlow action class
data	Packet data of a binary type value or an instances of packet.Packet.

Example:

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

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

JSON Example:

```
{
    "OFPPacketOut": {
        "actions": [
            {
                "OFPActionOutput": {
                    "len": 16,
                    "max_len": 65535,
                    "port": 4294967291,
                    "type": 0
                }
            }
        ],
        "actions_len": 16,
        "buffer_id": 4294967295,
        "data": "dGVzdA==",
        "match": {
            "OFPMatch": {
                "length": 12,
                "oxm_fields": [
                    {
                        "OXMTlv": {
                            "field": "in_port",
                            "mask": null,
                            "value": 4294967040
                        }
                    }
                ],
                "type": 1
            }
        }
    }
}
```

Barrier Message

class `ryu.ofproto.ofproto_v1_5_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:

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

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

JSON Example:

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

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPBarrierReply`**(*datapath*)

Barrier reply message

The switch responds with this message to a barrier request.

Example:

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

JSON Example:

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

Role Request Message

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPRoleRequest`**(*datapath*, *role=None*, *short_id=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
short_id	ID number for the controller. The default is OFPCID_UNDEFINED.
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,
                                    ofp.OFPCID_UNDEFINED, 0)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPRoleRequest": {
      "generation_id": 12345678901234567890,
      "role": 1,
      "short_id": 43690
   }
}
```

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPRoleReply`**(*datapath*, *role=None*, *short_id=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
short_id	ID number for the controller. The default is OFPCID_UNDEFINED.
generation_id	Master Election Generation ID

Example:

```
@set_ev_cls(ofp_event.EventOFPRoleReply, MAIN_DISPATCHER)
def role_reply_handler(self, ev):
    msg = ev.msg
    dp = msg.datapath
    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 short_id=%d, generation_id=%d',
                      role, msg.short_id, msg.generation_id)
```

JSON Example:

```
{
   "OFPRoleReply": {
      "generation_id": 1234605616436508552,
      "role": 1,
      "short_id": 43690
   }
}
```

Bundle Messages

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPBundleCtrlMsg`**(*datapath*, *bundle_id=None*, *type_=None*, *flags=None*, *properties=None*)

Bundle control message

The controller uses this message to create, destroy and commit bundles

Attribute	Description
bundle_id	Id of the bundle
type	One of the following values. OFPBCT_OPEN_REQUEST OFPBCT_OPEN_REPLY OFPBCT_CLOSE_REQUEST OFPBCT_CLOSE_REPLY OFPBCT_COMMIT_REQUEST OFPBCT_COMMIT_REPLY OFPBCT_DISCARD_REQUEST OFPBCT_DISCARD_REPLY
flags	Bitmap of the following flags. OFPBF_ATOMIC OFPBF_ORDERED
properties	List of `OFPBundleProp` subclass instance

Example:

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

    req = ofp_parser.OFPBundleCtrlMsg(datapath, 7,
                                      ofp.OFPBCT_OPEN_REQUEST,
                                      [ofp.OFPBF_ATOMIC], [])
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPBundleCtrlMsg": {
      "bundle_id": 99999999,
      "flags": 1,
      "properties": [],
      "type": 1
   }
}
```

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPBundleAddMsg`**(*datapath*, *bundle_id*, *flags*, *message*, *properties*)

Bundle control message

The controller uses this message to create, destroy and commit bundles

Attribute	Description
bundle_id	Id of the bundle
flags	Bitmap of the following flags. OFPBF_ATOMIC OFPBF_ORDERED
message	`MsgBase` subclass instance
properties	List of `OFPBundleProp` subclass instance

Example:

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

    msg = ofp_parser.OFPRoleRequest(datapath, ofp.OFPCR_ROLE_EQUAL, 0)

    req = ofp_parser.OFPBundleAddMsg(datapath, 7, [ofp.OFPBF_ATOMIC],
                                     msg, [])
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPBundleAddMsg": {
      "bundle_id": 99999999,
      "flags": 1,
      "message": {
         "OFPFlowMod": {
            "buffer_id": 0,
            "command": 0,
            "cookie": 1311768467463790320,
            "cookie_mask": 18446744073709551615,
            "flags": 0,
            "hard_timeout": 0,
            "idle_timeout": 0,
            "importance": 39032,
            "instructions": [
               {
                  "OFPInstructionActions": {
                     "actions": [
                        {
                           "OFPActionPopVlan": {
                              "len": 8,
                              "type": 18
                           }
                        },
                        {
                           "OFPActionSetField": {
                              "field": {
                                 "OXMTlv": {
                                    "field": "ipv4_dst",
                                    "mask": null,
                                    "value": "192.168.2.9"
                                 }
                              },
                              "len": 16,
                              "type": 25
                           }
                        },
                        {
                           "NXActionLearn": {
                              "cookie": 0,
                              "experimenter": 8992,
                              "fin_hard_timeout": 0,
                              "fin_idle_timeout": 0,
                              "flags": 0,
                              "hard_timeout": 300,
                              "idle_timeout": 0,
                              "len": 96,
                              "priority": 1,
                              "specs": [
                                 {
                                    "NXFlowSpecMatch": {
                                       "dst": [
                                          "vlan_vid",
                                          0
                                       ],
                                       "n_bits": 12,
                                       "src": [
                                          "vlan_vid",
                                          0
                                       ]
                                    }
                                 },
                                 {
                                    "NXFlowSpecMatch": {
                                       "dst": [
                                          "eth_dst_nxm",
                                          0
                                       ],
                                       "n_bits": 48,
                                       "src": [
                                          "eth_src_nxm",
                                          0
                                       ]
                                    }
                                 },
                                 {
                                    "NXFlowSpecLoad": {
                                       "dst": [
                                          "vlan_vid",
                                          0
                                       ],
                                       "n_bits": 12,
                                       "src": 0
                                    }
                                 },
                                 {
                                    "NXFlowSpecLoad": {
                                       "dst": [
                                          "tunnel_id_nxm",
                                          0
                                       ],
                                       "n_bits": 64,
                                       "src": [
                                          "tunnel_id_nxm",
                                          0
                                       ]
                                    }
                                 },
                                 {
                                    "NXFlowSpecOutput": {
                                       "dst": "",
                                       "n_bits": 32,
                                       "src": [
                                          "in_port",
                                          0
                                       ]
                                    }
                                 }
                              ],
                              "subtype": 16,
                              "table_id": 99,
                              "type": 65535
                           }
                        }
                     ],
                     "len": 128,
                     "type": 4
                  }
               },
               {
                  "OFPInstructionGotoTable": {
                     "len": 8,
                     "table_id": 100,
                     "type": 1
                  }
               }
            ],
            "match": {
               "OFPMatch": {
                  "length": 70,
                  "oxm_fields": [
                     {
                        "OXMTlv": {
                           "field": "in_port",
                           "mask": null,
                        "value": 43981
                    }
                },
                {
                    "OXMTlv": {
                        "field": "eth_dst",
                        "mask": null,
                        "value": "aa:bb:cc:99:88:77"
                    }
                },
                {
                    "OXMTlv": {
                        "field": "eth_type",
                        "mask": null,
                        "value": 2048
                    }
                },
                {
                    "OXMTlv": {
                        "field": "vlan_vid",
                        "mask": null,
                        "value": 5095
                    }
                },
                {
                    "OXMTlv": {
                        "field": "ipv4_dst",
                        "mask": null,
                        "value": "192.168.2.1"
                    }
                },
                {
                    "OXMTlv": {
                        "field": "tunnel_id",
                        "mask": null,
                        "value": 50000
                    }
                },
                {
                    "OXMTlv": {
                        "field": "tun_ipv4_src",
                        "mask": null,
                        "value": "192.168.2.3"
                    }
                },
                {
                    "OXMTlv": {
                        "field": "tun_ipv4_dst",
                        "mask": null,
                        "value": "192.168.2.4"
                    }
                }
            ],
            "type": 1
        }
    },
    "out_group": 0,
    "out_port": 0,
                "priority": 0,
                "table_id": 2
            }
        },
        "properties": []
    }
}
```

Set Asynchronous Configuration Message

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPSetAsync`**(*datapath*, *properties=None*)

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
properties	List of `OFPAsyncConfigProp` subclass instances

Example:

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

    properties = [
        ofp_parser.OFPAsyncConfigPropReasons(
            ofp.OFPACPT_PACKET_IN_SLAVE, 8,
            (1 << ofp.OFPR_APPLY_ACTION
             | 1 << ofp.OFPR_INVALID_TTL))]
    req = ofp_parser.OFPSetAsync(datapath, properties)
    datapath.send_msg(req)
```

JSON Example:

```
{
   "OFPSetAsync": {
      "properties": [
         {
            "OFPAsyncConfigPropReasons": {
               "length": 8,
               "mask": 3,
               "type": 0
            }
         },
         {
            "OFPAsyncConfigPropReasons": {
               "length": 8,
               "mask": 3,
               "type": 1
            }
         },
         {
            "OFPAsyncConfigPropReasons": {
               "length": 8,
               "mask": 3,
               "type": 2
            }
         },
         {
            "OFPAsyncConfigPropReasons": {
               "length": 8,
               "mask": 3,
               "type": 3
            }
         },
         {
            "OFPAsyncConfigPropReasons": {
               "length": 8,
               "mask": 3,
               "type": 4
            }
         },
         {
            "OFPAsyncConfigPropReasons": {
               "length": 8,
               "mask": 3,
               "type": 5
            }
         },
         {
            "OFPAsyncConfigPropReasons": {
               "length": 8,
               "mask": 3,
               "type": 6
            }
         },
         {
            "OFPAsyncConfigPropReasons": {
               "length": 8,
               "mask": 3,
               "type": 7
            }
         },
         {
            "OFPAsyncConfigPropReasons": {
               "length": 8,
               "mask": 24,
               "type": 8
            }
         },
         {
            "OFPAsyncConfigPropReasons": {
               "length": 8,
               "mask": 24,
               "type": 9
            }
         },
         {
            "OFPAsyncConfigPropReasons": {
               "length": 8,
               "mask": 3,
               "type": 10
            }
         },
          {
             "OFPAsyncConfigPropReasons": {
                "length": 8,
                "mask": 3,
                "type": 11
             }
          },
          {
             "OFPAsyncConfigPropExperimenter": {
                "data": [],
                "exp_type": 0,
                "experimenter": 101,
                "length": 12,
                "type": 65534
             }
          },
          {
             "OFPAsyncConfigPropExperimenter": {
                "data": [
                   1
                ],
                "exp_type": 1,
                "experimenter": 101,
                "length": 16,
                "type": 65535
             }
          },
          {
             "OFPAsyncConfigPropExperimenter": {
                "data": [
                   1,
                   2
                ],
                "exp_type": 2,
                "experimenter": 101,
                "length": 20,
                "type": 65535
             }
          }
       ]
    }
}
```

class `ryu.ofproto.ofproto_v1_5_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_5_parser.`**`OFPGetAsyncReply`**(*datapath*, *properties=None*)

Get asynchronous configuration reply message

The switch responds with this message to a get asynchronous configuration request.

Attribute	Description
properties	List of `OFPAsyncConfigProp` subclass instances

Example:

```
@set_ev_cls(ofp_event.EventOFPGetAsyncReply, MAIN_DISPATCHER)
def get_async_reply_handler(self, ev):
    msg = ev.msg

    self.logger.debug('OFPGetAsyncReply received: '
                      'properties=%s', repr(msg.properties))
```

JSON Example:

```
{
   "OFPGetAsyncReply": {
      "properties": [
         {
            "OFPAsyncConfigPropReasons": {
               "length": 8,
               "mask": 3,
               "type": 0
            }
         },
         {
            "OFPAsyncConfigPropReasons": {
               "length": 8,
               "mask": 3,
               "type": 1
            }
         },
         {
            "OFPAsyncConfigPropReasons": {
               "length": 8,
               "mask": 3,
               "type": 2
            }
         },
         {
            "OFPAsyncConfigPropReasons": {
               "length": 8,
               "mask": 3,
               "type": 3
            }
         },
         {
            "OFPAsyncConfigPropReasons": {
               "length": 8,
               "mask": 3,
               "type": 4
```

```
                }
            },
            {
                "OFPAsyncConfigPropReasons": {
                    "length": 8,
                    "mask": 3,
                    "type": 5
                }
            },
            {
                "OFPAsyncConfigPropReasons": {
                    "length": 8,
                    "mask": 3,
                    "type": 6
                }
            },
            {
                "OFPAsyncConfigPropReasons": {
                    "length": 8,
                    "mask": 3,
                    "type": 7
                }
            },
            {
                "OFPAsyncConfigPropReasons": {
                    "length": 8,
                    "mask": 24,
                    "type": 8
                }
            },
            {
                "OFPAsyncConfigPropReasons": {
                    "length": 8,
                    "mask": 24,
                    "type": 9
                }
            },
            {
                "OFPAsyncConfigPropReasons": {
                    "length": 8,
                    "mask": 3,
                    "type": 10
                }
            },
            {
                "OFPAsyncConfigPropReasons": {
                    "length": 8,
                    "mask": 3,
                    "type": 11
                }
            },
            {
                "OFPAsyncConfigPropExperimenter": {
                    "data": [],
                    "exp_type": 0,
                    "experimenter": 101,
                    "length": 12,
                    "type": 65534
                }
            },
            {
                "OFPAsyncConfigPropExperimenter": {
                    "data": [
                        1
                    ],
                    "exp_type": 1,
                    "experimenter": 101,
                    "length": 16,
                    "type": 65535
                }
            },
            {
                "OFPAsyncConfigPropExperimenter": {
                    "data": [
                        1,
                        2
                    ],
                    "exp_type": 2,
                    "experimenter": 101,
                    "length": 20,
                    "type": 65535
                }
            }
        ]
    }
}
```

Asynchronous Messages

Packet-In Message

class `ryu.ofproto.ofproto_v1_5_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_TABLE_MISS OFPR_APPLY_ACTION OFPR_INVALID_TTL OFPR_ACTION_SET OFPR_GROUP OFPR_PACKET_OUT
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
    dp = msg.datapath
    ofp = dp.ofproto

    if msg.reason == ofp.TABLE_MISS:
        reason = 'TABLE MISS'
    elif msg.reason == ofp.OFPR_APPLY_ACTION:
        reason = 'APPLY ACTION'
    elif msg.reason == ofp.OFPR_INVALID_TTL:
        reason = 'INVALID TTL'
    elif msg.reason == ofp.OFPR_ACTION_SET:
        reason = 'ACTION SET'
    elif msg.reason == ofp.OFPR_GROUP:
        reason = 'GROUP'
    elif msg.reason == ofp.OFPR_PACKET_OUT:
        reason = 'PACKET OUT'
    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": 200,
      "cookie": 0,
      "data": "aG9nZQ==",
      "match": {
         "OFPMatch": {
            "length": 40,
                "oxm_fields": [
                    {
                        "OXMTlv": {
                            "field": "in_port",
                            "mask": null,
                            "value": 43981
                        }
                    },
                    {
                        "OXMTlv": {
                            "field": "tunnel_id",
                            "mask": null,
                            "value": 50000
                        }
                    },
                    {
                        "OXMTlv": {
                            "field": "tun_ipv4_src",
                            "mask": null,
                            "value": "192.168.2.3"
                        }
                    },
                    {
                        "OXMTlv": {
                            "field": "tun_ipv4_dst",
                            "mask": null,
                            "value": "192.168.2.4"
                        }
                    }
                ],
                "type": 1
            }
        },
        "reason": 0,
        "table_id": 100,
        "total_len": 1000
    }
}
```

Flow Removed Message

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPFlowRemoved`**`(`*datapath*, *table_id=None*, *reason=None*, *priority=None*, *idle_timeout=None*, *hard_timeout=None*, *cookie=None*, *match=None*, *stats=None*`)`

Flow removed message

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

Attribute	Description
table_id	ID of the table
reason	One of the following values. OFPRR_IDLE_TIMEOUT OFPRR_HARD_TIMEOUT OFPRR_DELETE OFPRR_GROUP_DELETE OFPRR_METER_DELETE OFPRR_EVICTION
priority	Priority level of flow entry
idle_timeout	Idle timeout from original flow mod
hard_timeout	Hard timeout from original flow mod
cookie	Opaque controller-issued identifier
match	Instance of `OFPMatch`
stats	Instance of `OFPStats`

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'
    elif msg.reason == ofp.OFPRR_METER_DELETE:
        reason = 'METER DELETE'
    elif msg.reason == ofp.OFPRR_EVICTION:
        reason = 'EVICTION'
    else:
        reason = 'unknown'

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

JSON Example:

```
{
    "OFPFlowRemoved": {
        "cookie": 1234605616436508552,
        "hard_timeout": 255,
        "idle_timeout": 255,
        "match": {
            "OFPMatch": {
                "length": 12,
                "oxm_fields": [
                    {
                        "OXMTlv": {
                            "field": "in_port",
                            "mask": null,
                            "value": 1
                        }
                    }
                ],
                "type": 1
            }
        },
        "priority": 1,
        "reason": 0,
        "stats": {
            "OFPStats": {
                "length": 12,
                "oxs_fields": [
                    {
                        "OXSTlv": {
                            "field": "flow_count",
                            "value": 1
                        }
                    }
                ]
            }
        },
        "table_id": 1
    }
}
```

Port Status Message

class `ryu.ofproto.ofproto_v1_5_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": {
            "config": 0,
            "hw_addr": "f2:0b:a4:d0:3f:70",
            "length": 168,
            "name": "\u79c1\u306e\u30dd\u30fc\u30c8",
            "port_no": 7,
            "properties": [
               {
                  "OFPPortDescPropEthernet": {
                     "advertised": 10240,
                     "curr": 10248,
                     "curr_speed": 5000,
                     "length": 32,
                     "max_speed": 5000,
                     "peer": 10248,
                     "supported": 10248,
                     "type": 0
                  }
               },
               {
                  "OFPPortDescPropOptical": {
                     "length": 40,
                     "rx_grid_freq_lmda": 1500,
                     "rx_max_freq_lmda": 2000,
                     "rx_min_freq_lmda": 1000,
                     "supported": 1,
                     "tx_grid_freq_lmda": 1500,
                     "tx_max_freq_lmda": 2000,
                     "tx_min_freq_lmda": 1000,
                     "tx_pwr_max": 2000,
                     "tx_pwr_min": 1000,
                     "type": 1
                  }
               },
               {
                  "OFPPortDescPropExperimenter": {
                     "data": [],
                     "exp_type": 0,
                     "experimenter": 101,
                            "length": 12,
                            "type": 65535
                        }
                    },
                    {
                        "OFPPortDescPropExperimenter": {
                            "data": [
                                1
                            ],
                            "exp_type": 1,
                            "experimenter": 101,
                            "length": 16,
                            "type": 65535
                        }
                    },
                    {
                        "OFPPortDescPropExperimenter": {
                            "data": [
                                1,
                                2
                            ],
                            "exp_type": 2,
                            "experimenter": 101,
                            "length": 20,
                            "type": 65535
                        }
                    }
                ],
                "state": 4
            }
        },
        "reason": 0
    }
}
```

Controller Role Status Message

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPRoleStatus`**(*datapath*, *role=None*, *reason=None*, *generation_id=None*, *properties=None*)

Role status message

The switch notifies controller of change of role.

Attribute	Description
role	One of the following values. OFPCR_ROLE_NOCHANGE OFPCR_ROLE_EQUAL OFPCR_ROLE_MASTER
reason	One of the following values. OFPCRR_MASTER_REQUEST OFPCRR_CONFIG OFPCRR_EXPERIMENTER
generation_id	Master Election Generation ID
properties	List of `OFPRoleProp` subclass instance

Example:

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

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

    if msg.reason == ofp.OFPCRR_MASTER_REQUEST:
        reason = 'MASTER REQUEST'
    elif msg.reason == ofp.OFPCRR_CONFIG:
        reason = 'CONFIG'
    elif msg.reason == ofp.OFPCRR_EXPERIMENTER:
        reason = 'EXPERIMENTER'
    else:
        reason = 'unknown'

    self.logger.debug('OFPRoleStatus received: role=%s reason=%s '
                      'generation_id=%d properties=%s', role, reason,
                      msg.generation_id, repr(msg.properties))
```

JSON Example:

```
{
   "OFPRoleStatus": {
      "generation_id": 17356517385562371090,
      "properties": [],
      "reason": 0,
      "role": 3
   }
}
```

Table Status Message

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPTableStatus`**(*datapath*, *reason=None*, *table=None*)

Table status message

The switch notifies controller of change of table status.

Attribute	Description
reason	One of the following values. OFPTR_VACANCY_DOWN OFPTR_VACANCY_UP
table	`OFPTableDesc` instance

Example:

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

    if msg.reason == ofp.OFPTR_VACANCY_DOWN:
        reason = 'VACANCY_DOWN'
    elif msg.reason == ofp.OFPTR_VACANCY_UP:
        reason = 'VACANCY_UP'
    else:
        reason = 'unknown'

    self.logger.debug('OFPTableStatus received: reason=%s '
                      'table_id=%d config=0x%08x properties=%s',
                      reason, msg.table.table_id, msg.table.config,
                      repr(msg.table.properties))
```

JSON Example:

```
{
   "OFPTableStatus": {
      "reason": 3,
      "table": {
         "OFPTableDesc": {
            "config": 0,
            "length": 80,
            "properties": [
               {
                  "OFPTableModPropEviction": {
                     "flags": 0,
                     "length": 8,
                     "type": 2
                  }
               },
               {
                  "OFPTableModPropVacancy": {
                     "length": 8,
                     "type": 3,
                     "vacancy": 0,
                     "vacancy_down": 0,
                     "vacancy_up": 0
                  }
               },
               {
                  "OFPTableModPropExperimenter": {
                     "data": [],
                     "exp_type": 0,
                     "experimenter": 101,
                     "length": 12,
                     "type": 65535
                  }
               },
               {
                  "OFPTableModPropExperimenter": {
                     "data": [
                        1
                     ],
                     "exp_type": 1,
                     "experimenter": 101,
                     "length": 16,
                     "type": 65535
                  }
               },
               {
                  "OFPTableModPropExperimenter": {
                     "data": [
                        1,
                        2
                     ],
                     "exp_type": 2,
                     "experimenter": 101,
                     "length": 20,
                     "type": 65535
                  }
               }
            ],
            "table_id": 8
         }
      }
   }
}
```

Request Forward Message

class ryu.ofproto.ofproto_v1_5_parser.**OFPRequestForward**(*datapath*, *request=None*)

Forwarded request message

The swtich forwards request messages from one controller to other controllers.

Attribute	Description
request	OFPGroupMod or OFPMeterMod instance

Example:

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

    if msg.request.msg_type == ofp.OFPT_GROUP_MOD:
        self.logger.debug(
            'OFPRequestForward received: request=OFPGroupMod('
            'command=%d, type=%d, group_id=%d, command_bucket_id=%d, '
            'buckets=%s, properties=%s)',
            msg.request.command, msg.request.type,
            msg.request.group_id, msg.request.command_bucket_id,
            msg.request.buckets, repr(msg.request.properties))
    elif msg.request.msg_type == ofp.OFPT_METER_MOD:
        self.logger.debug(
            'OFPRequestForward received: request=OFPMeterMod('
            'command=%d, flags=%d, meter_id=%d, bands=%s)',
            msg.request.command, msg.request.flags,
            msg.request.meter_id, msg.request.bands)
    else:
        self.logger.debug(
            'OFPRequestForward received: request=Unknown')
```

JSON Example:

```
{
   "OFPRequestForward": {
      "request": {
         "OFPGroupMod": {
            "bucket_array_len": 56,
            "buckets": [
               {
                  "OFPBucket": {
                     "action_array_len": 24,
                     "actions": [
                        {
                           "OFPActionPopVlan": {
                              "len": 8,
                              "type": 18
                           }
                        },
                        {
                           "OFPActionSetField": {
                              "field": {
                                 "OXMTlv": {
                                    "field": "ipv4_dst",
                                    "mask": null,
                                    "value": "192.168.2.9"
                                 }
                              },
                              "len": 16,
                              "type": 25
                           }
                        }
                     ],
                     "bucket_id": 305419896,
                     "len": 56,
                     "properties": [
                        {
                           "OFPGroupBucketPropWeight": {
                              "length": 8,
                              "type": 0,
                              "weight": 52428
                           }
                        },
                        {
                           "OFPGroupBucketPropWatch": {
                              "length": 8,
                              "type": 1,
                              "watch": 56797
                           }
                        },
                        {
                           "OFPGroupBucketPropWatch": {
                              "length": 8,
                              "type": 2,
                              "watch": 4008636142
                           }
                        }
                     ]
                  }
               }
            ],
            "command": 3,
            "command_bucket_id": 3149642683,
            "group_id": 2863311530,
            "properties": [],
            "type": 1
         }
      }
   }
}
```

Controller Status Message

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPControllerStatus`**(*datapath*, *status=None*)

Controller status message

The switch informs the controller about the status of the control channel it maintains with each controller.

Attribute	Description
status	`OFPControllerStatusStats` instance

Example:

```
@set_ev_cls(ofp_event.EventOFPControllerStatus, MAIN_DISPATCHER)
def table(self, ev):
    msg = ev.msg
    dp = msg.datapath
    ofp = dp.ofproto
    status = msg.status

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

    if status.reason == ofp.OFPCSR_REQUEST:
        reason = 'REQUEST'
    elif status.reason == ofp.OFPCSR_CHANNEL_STATUS:
        reason = 'CHANNEL_STATUS'
    elif status.reason == ofp.OFPCSR_ROLE:
        reason = 'ROLE'
    elif status.reason == ofp.OFPCSR_CONTROLLER_ADDED:
        reason = 'CONTROLLER_ADDED'
    elif status.reason == ofp.OFPCSR_CONTROLLER_REMOVED:
        reason = 'CONTROLLER_REMOVED'
    elif status.reason == ofp.OFPCSR_SHORT_ID:
        reason = 'SHORT_ID'
    elif status.reason == ofp.OFPCSR_EXPERIMENTER:
        reason = 'EXPERIMENTER'
    else:
        reason = 'unknown'

    if status.channel_status == OFPCT_STATUS_UP:
        channel_status = 'UP'
    if status.channel_status == OFPCT_STATUS_DOWN:
        channel_status = 'DOWN'
    else:
        channel_status = 'unknown'

    self.logger.debug('OFPControllerStatus received: short_id=%d'
                      'role=%s reason=%s channel_status=%s '
                      'properties=%s',
                      status.short_id, role, reason, channel_status,
                      repr(status.properties))
```

JSON Example:

```
{
    "OFPControllerStatus": {
        "status": {
            "OFPControllerStatusStats": {
                "channel_status": 1,
                "length": 48,
                "properties": [
                    {
                        "OFPControllerStatusPropUri": {
                            "length": 26,
                            "type": 0,
                            "uri": "tls:192.168.34.23:6653"
                        }
                    }
                ],
                "reason": 1,
                "role": 1,
                "short_id": 65535
            }
        }
    }
}
```

Symmetric Messages

Hello

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPHello`**(*datapath*, *elements=None*)

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": [
                  6
               ]
            }
         }
      ]
   }
}
```

class `ryu.ofproto.ofproto_v1_5_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_5_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_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": ""
   }
}
```

Echo Reply

class `ryu.ofproto.ofproto_v1_5_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_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": ""
   }
}
```

Error Message

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPErrorMsg`**(*datapath*, *type_=None*, *code=None*, *data=None*, ***kwargs*)

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

If `type == OFPET_EXPERIMENTER`, this message has also the following attributes.

Attribute	Description
exp_type	Experimenter defined type
experimenter	Experimenter ID

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": 6,
      "data": "Bg4ACAAAAAA=",
      "type": 4
```

```
    }
}
```

Experimenter

class `ryu.ofproto.ofproto_v1_5_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:

```
{
   "OFPErrorMsg": {
      "code": null,
      "data": "amlra2VuIGRhdGE=",
      "exp_type": 60000,
      "experimenter": 999999,
      "type": 65535
   }
}
```

Port Structures

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPPort`**(*port_no=None*, *length=None*, *hw_addr=None*, *name=None*, *config=None*, *state=None*, *properties=None*)

Description of a port

Attribute	Description
port_no	Port number and it uniquely identifies a port within a switch.
length	Length of ofp_port (excluding padding).
hw_addr	MAC address for the port.
name	Null-terminated string containing a human-readable name for the interface.
config	Bitmap of port configration flags. OFPPC_PORT_DOWN OFPPC_NO_RECV OFPPC_NO_FWD OFPPC_NO_PACKET_IN
state	Bitmap of port state flags. OFPPS_LINK_DOWN OFPPS_BLOCKED OFPPS_LIVE
properties	List of `OFPPortDescProp` subclass instance

Flow Match Structure

class `ryu.ofproto.ofproto_v1_5_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.

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
udp_dst	Integer 16bit	UDP destination port
sctp_src	Integer 16bit	SCTP source port
sctp_dst	Integer 16bit	SCTP destination port

Continued on next page

Table 2.4 – continued from previous page

Argument	Value	Description
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
pbb_uca	Integer 8bit	PBB UCA header field
tcp_flags	Integer 16bit	TCP flags
actset_output	Integer 32bit	Output port from action set metadata
packet_type	Integer 32bit	Packet type value

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::')
```

Note: For the list of the supported Nicira experimenter matches, please refer to *ryu.ofproto.nx_match*.

Note: For VLAN id match field, special values are defined in OpenFlow Spec.

1. Packets with and without a VLAN tag

   - Example:

```
match = parser.OFPMatch()
```

- Packet Matching

non-VLAN-tagged	MATCH
VLAN-tagged(vlan_id=3)	MATCH
VLAN-tagged(vlan_id=5)	MATCH

2. Only packets without a VLAN tag

- Example:

```
match = parser.OFPMatch(vlan_vid=0x0000)
```

- Packet Matching

non-VLAN-tagged	MATCH
VLAN-tagged(vlan_id=3)	x
VLAN-tagged(vlan_id=5)	x

3. Only packets with a VLAN tag regardless of its value

- Example:

```
match = parser.OFPMatch(vlan_vid=(0x1000, 0x1000))
```

- Packet Matching

non-VLAN-tagged	x
VLAN-tagged(vlan_id=3)	MATCH
VLAN-tagged(vlan_id=5)	MATCH

4. Only packets with VLAN tag and VID equal

- Example:

```
match = parser.OFPMatch(vlan_vid=(0x1000 | 3))
```

- Packet Matching

non-VLAN-tagged	x
VLAN-tagged(vlan_id=3)	MATCH
VLAN-tagged(vlan_id=5)	x

Flow Stats Structures

class ryu.ofproto.ofproto_v1_5_parser.**OFPStats**(*length=None*, *_ordered_fields=None*, ***kwargs*)

Flow Stats Structure

This class is implementation of the flow stats structure having compose/query API.

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

Argument	Value	Description
duration	Integer 32bit*2	Time flow entry has been alive. This field is a tuple of two Integer 32bit. The first value is duration_sec and the second is duration_nsec.
idle_time	Integer 32bit*2	Time flow entry has been idle.
flow_count	Integer 32bit	Number of aggregated flow entries.
packet_count	Integer 64bit	Number of packets matched by a flow entry.
byte_count	Integer 64bit	Number of bytes matched by a flow entry.

Example:

```
>>> # compose
>>> stats = parser.OFPStats(
...     packet_count=100,
...     duration=(100, 200)
>>> # query
>>> if 'duration' in stats:
...     print stats['duration']
...
(100, 200)
```

Flow Instruction Structures

class `ryu.ofproto.ofproto_v1_5_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_5_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_5_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_5_parser.`**`OFPInstructionStatTrigger`**(*flags*, *thresholds*, *type_=None*, *len_=None*)

Statistics triggers instruction

This instruction defines a set of statistics thresholds using OXS.

Attribute	Description
flags	Bitmap of the following flags. OFPSTF_PERIODIC OFPSTF_ONLY_FIRST
thresholds	Instance of `OFPStats`

Action Structures

class `ryu.ofproto.ofproto_v1_5_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_5_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_5_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_5_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_5_parser.`**`OFPActionDecMplsTtl`**(*type_=None*, *len_=None*)

Decrement MPLS TTL action

This action decrements the MPLS TTL.

class `ryu.ofproto.ofproto_v1_5_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_5_parser.`**`OFPActionPopVlan`**(*type_=None*, *len_=None*)

Pop VLAN action

This action pops the outermost VLAN tag from the packet.

class `ryu.ofproto.ofproto_v1_5_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_5_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_5_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_5_parser.`**`OFPActionGroup`**(*group_id=0*, *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_5_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_5_parser.`**`OFPActionDecNwTtl`**(*type_=None*, *len_=None*)

Decrement IP TTL action

This action decrements the IP TTL.

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPActionSetField`**(*field=None*, ***kwargs*)

Set field action

This action modifies a header field in the packet.

The set of keywords available for this is same as OFPMatch which including with/without mask.

Example:

```
set_field = OFPActionSetField(eth_src="00:00:00:00:00:00")
set_field = OFPActionSetField(ipv4_src=("192.168.100.0",
                                        "255.255.255.0"))
```

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPActionPushPbb`**(*ethertype*, *type_=None*, *len_=None*)

Push PBB action

This action pushes a new PBB header to the packet.

Attribute	Description
ethertype	Ether type

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPActionPopPbb`**(*type_=None*, *len_=None*)

Pop PBB action

This action pops the outermost PBB service instance header from the packet.

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPActionCopyField`**(*n_bits=0*, *src_offset=0*, *dst_offset=0*, *oxm_ids=None*, *type_=None*, *len_=None*)

Copy Field action

This action copy value between header and register.

Attribute	Description
n_bits	Number of bits to copy.
src_offset	Starting bit offset in source.
dst_offset	Starting bit offset in destination.
oxm_ids	List of `OFPOxmId` instances. The first element of this list, src_oxm_id, identifies the field where the value is copied from. The second element of this list, dst_oxm_id, identifies the field where the value is copied to. The default is [].

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPActionMeter`**(*meter_id*, *type_=None*, *len_=None*)

Meter action

This action applies meter (rate limiter)

Attribute	Description
meter_id	Meter instance

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPActionExperimenter`**(*experimenter*)

Experimenter action

This action is an extensible action for the experimenter.

Attribute	Description
experimenter	Experimenter ID

Note: For the list of the supported Nicira experimenter actions, please refer to *ryu.ofproto.nx_actions*.

Controller Status Structure

class `ryu.ofproto.ofproto_v1_5_parser.`**`OFPControllerStatusStats`**(*short_id=None, role=None, reason=None, channel_status=None, properties=None, length=None*)

Controller status structure

Attribute	Description
length	Length of this entry.
short_id	ID number which identifies the controller.
role	Bitmap of controller's role flags. OFPCR_ROLE_NOCHANGE OFPCR_ROLE_EQUAL OFPCR_ROLE_MASTER OFPCR_ROLE_SLAVE
reason	Bitmap of controller status reason flags. OFPCSR_REQUEST OFPCSR_CHANNEL_STATUS OFPCSR_ROLE OFPCSR_CONTROLLER_ADDED OFPCSR_CONTROLLER_REMOVED OFPCSR_SHORT_ID OFPCSR_EXPERIMENTER
channel_status	Bitmap of control channel status flags. OFPCT_STATUS_UP OFPCT_STATUS_DOWN
properties	List of `OFPControllerStatusProp` subclass instance

2.6 Nicira Extension Structures

2.6.1 Nicira Extension Actions Structures

The followings shows the supported NXAction classes only in OpenFlow1.0

class `ryu.ofproto.ofproto_v1_0_parser.`**`NXActionSetQueue`**(*queue_id, type_=None, len_=None, vendor=None, subtype=None*)

Set queue action

This action sets the queue that should be used to queue when packets are output.

And equivalent to the followings action of ovs-ofctl command.

set_queue:*queue*

Attribute	Description
queue_id	Queue ID for the packets

Note: This actions is supported by `OFPActionSetQueue` in OpenFlow1.2 or later.

Example:

```
actions += [parser.NXActionSetQueue(queue_id=10)]
```

class `ryu.ofproto.ofproto_v1_0_parser.`**`NXActionDecTtl`**(*type_=None*, *len_=None*, *vendor=None*, *subtype=None*)

Decrement IP TTL action

This action decrements TTL of IPv4 packet or hop limit of IPv6 packet.

And equivalent to the followings action of ovs-ofctl command.

dec_ttl

Note: This actions is supported by `OFPActionDecNwTtl` in OpenFlow1.2 or later.

Example:

```
actions += [parser.NXActionDecTtl()]
```

class `ryu.ofproto.ofproto_v1_0_parser.`**`NXActionPushMpls`**(*ethertype*, *type_=None*, *len_=None*, *vendor=None*, *subtype=None*)

Push MPLS action

This action pushes a new MPLS header to the packet.

And equivalent to the followings action of ovs-ofctl command.

push_mpls:*ethertype*

Attribute	Description
ethertype	Ether type(The value must be either 0x8847 or 0x8848)

Note: This actions is supported by `OFPActionPushMpls` in OpenFlow1.2 or later.

Example:

```
match = parser.OFPMatch(dl_type=0x0800)
actions += [parser.NXActionPushMpls(ethertype=0x8847)]
```

class `ryu.ofproto.ofproto_v1_0_parser.`**`NXActionPopMpls`**(*ethertype*, *type_=None*, *len_=None*, *vendor=None*, *subtype=None*)

Pop MPLS action

This action pops the MPLS header from the packet.

And equivalent to the followings action of ovs-ofctl command.

pop_mpls:*ethertype*

Attribute	Description
ethertype	Ether type

Note: This actions is supported by `OFPActionPopMpls` in OpenFlow1.2 or later.

Example:

```
match = parser.OFPMatch(dl_type=0x8847)
actions += [parser.NXActionPushMpls(ethertype=0x0800)]
```

class `ryu.ofproto.ofproto_v1_0_parser.`**`NXActionSetMplsTtl`**(*ttl*, *type_=None*, *len_=None*, *vendor=None*, *subtype=None*)

Set MPLS TTL action

This action sets the MPLS TTL.

And equivalent to the followings action of ovs-ofctl command.

set_mpls_ttl:*ttl*

Attribute	Description
ttl	MPLS TTL

Note: This actions is supported by `OFPActionSetMplsTtl` in OpenFlow1.2 or later.

Example:

```
actions += [parser.NXActionSetMplsTil(ttl=128)]
```

class `ryu.ofproto.ofproto_v1_0_parser.`**`NXActionDecMplsTtl`**(*type_=None*, *len_=None*, *vendor=None*, *subtype=None*)

Decrement MPLS TTL action

This action decrements the MPLS TTL.

And equivalent to the followings action of ovs-ofctl command.

dec_mpls_ttl

Note: This actions is supported by `OFPActionDecMplsTtl` in OpenFlow1.2 or later.

Example:

```
actions += [parser.NXActionDecMplsTil()]
```

class `ryu.ofproto.ofproto_v1_0_parser.`**`NXActionSetMplsLabel`**(*label*, *type_=None*, *len_=None*, *vendor=None*, *subtype=None*)

Set MPLS Lavel action

This action sets the MPLS Label.

And equivalent to the followings action of ovs-ofctl command.

set_mpls_label:*label*

Attribute	Description
label	MPLS Label

Note: This actions is supported by `OFPActionSetField(mpls_label=label)` in OpenFlow1.2 or later.

Example:

```
actions += [parser.NXActionSetMplsLabel(label=0x10)]
```

class `ryu.ofproto.ofproto_v1_0_parser.`**`NXActionSetMplsTc`**(*tc*, *type_=None*, *len_=None*, *vendor=None*, *subtype=None*)

Set MPLS Tc action

This action sets the MPLS Tc.

And equivalent to the followings action of ovs-ofctl command.

set_mpls_tc:*tc*

Attribute	Description
tc	MPLS Tc

Note: This actions is supported by `OFPActionSetField(mpls_label=tc)` in OpenFlow1.2 or later.

Example:

```
actions += [parser.NXActionSetMplsLabel(tc=0x10)]
```

The followings shows the supported NXAction classes in OpenFlow1.0 or later

class `ryu.ofproto.ofproto_v1_3_parser.`**`NXActionPopQueue`**(*type_=None*, *len_=None*, *experimenter=None*, *subtype=None*)

Pop queue action

This action restors the queue to the value it was before any set_queue actions were applied.

And equivalent to the followings action of ovs-ofctl command.

pop_queue

Example:

```
actions += [parser.NXActionPopQueue()]
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`NXActionRegLoad`**(*ofs_nbits*, *dst*, *value*, *type_=None*, *len_=None*, *experimenter=None*, *subtype=None*)

Load literal value action

This action loads a literal value into a field or part of a field.

And equivalent to the followings action of ovs-ofctl command.

load:*value*->*dst*[*start*..*end*]

Attribute	Description
ofs_nbits	Start and End for the OXM/NXM field. Setting method refer to the `nicira_ext.ofs_nbits`
dst	OXM/NXM header for destination field
value	OXM/NXM value to be loaded

Example:

```
actions += [parser.NXActionRegLoad(
                ofs_nbits=nicira_ext.ofs_nbits(4, 31),
                dst="eth_dst",
                value=0x112233)]
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`NXActionRegLoad2`**(*dst*, *value*, *mask=None*, *type_=None*, *len_=None*, *experimenter=None*, *subtype=None*)

Load literal value action

This action loads a literal value into a field or part of a field.

And equivalent to the followings action of ovs-ofctl command.

set_field:*value*[/*mask*]->*dst*

Attribute	Description
value	OXM/NXM value to be loaded
mask	Mask for destination field
dst	OXM/NXM header for destination field

Example:

```
actions += [parser.NXActionRegLoad2(dst="tun_ipv4_src",
                                    value="192.168.10.0",
                                    mask="255.255.255.0")]
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`NXActionNote`**(*note*, *type_=None*, *len_=None*, *experimenter=None*, *subtype=None*)

Note action

This action does nothing at all.

And equivalent to the followings action of ovs-ofctl command.

note:[*hh*]..

Attribute	Description
note	A list of integer type values

Example:

```
actions += [parser.NXActionNote(note=[0xaa,0xbb,0xcc,0xdd])]
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`NXActionSetTunnel`**(*tun_id*, *type_=None*, *len_=None*, *experimenter=None*, *subtype=None*)

Set Tunnel action

This action sets the identifier (such as GRE) to the specified id.

And equivalent to the followings action of ovs-ofctl command.

Note: ovs-ofctl command of the OpenFlow1.0 is different from that of OpenFlow1.2 or later.

OpenFlow1.0

set_tunnel:*id*

OpenFlow1.2 or later

set_field:*value*->tun_id

Attribute	Description
tun_id	Tunnel ID(32bits)

Example:

```
actions += [parser.NXActionSetTunnel(tun_id=0xa)]
```

class ryu.ofproto.ofproto_v1_3_parser.**NXActionSetTunnel64**(*tun_id*, *type_=None*, *len_=None*, *experimenter=None*, *subtype=None*)

Set Tunnel action

This action outputs to a port that encapsulates the packet in a tunnel.

And equivalent to the followings action of ovs-ofctl command.

Note: ovs-ofctl command of the OpenFlow1.0 is different from that of OpenFlow1.2 or later.

OpenFlow1.0

set_tunnel64:*id*

OpenFlow1.2 or later

set_field:*value*->tun_id

Attribute	Description
tun_id	Tunnel ID(64bits)

Example:

```
actions += [parser.NXActionSetTunnel64(tun_id=0xa)]
```

class ryu.ofproto.ofproto_v1_3_parser.**NXActionRegMove**(*src_field*, *dst_field*, *n_bits*, *src_ofs=0*, *dst_ofs=0*, *type_=None*, *len_=None*, *experimenter=None*, *subtype=None*)

Move register action

This action copies the src to dst.

And equivalent to the followings action of ovs-ofctl command.

move:*src*[*start*..*end*]->*dst*[*start*..*end*]

Attribute	Description
src_field	OXM/NXM header for source field
dst_field	OXM/NXM header for destination field
n_bits	Number of bits
src_ofs	Starting bit offset in source
dst_ofs	Starting bit offset in destination

Caution: **src_start** and **src_end** difference and **dst_start** and **dst_end** difference must be the same.

Example:

```
actions += [parser.NXActionRegMove(src_field="reg0",
                                   dst_field="reg1",
                                   n_bits=5,
                                   src_ofs=0
                                   dst_ofs=10)]
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`NXActionResubmit`**(*in_port=65528*, *type_=None*, *len_=None*, *experimenter=None*, *subtype=None*)

Resubmit action

This action searches one of the switch's flow tables.

And equivalent to the followings action of ovs-ofctl command.

resubmit:*port*

Attribute	Description
in_port	New in_port for checking flow table

Example:

```
actions += [parser.NXActionResubmit(in_port=8080)]
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`NXActionResubmitTable`**(*in_port=65528*, *table_id=255*, *type_=None*, *len_=None*, *experimenter=None*, *subtype=None*)

Resubmit action

This action searches one of the switch's flow tables.

And equivalent to the followings action of ovs-ofctl command.

resubmit([*port*],[*table*])

Attribute	Description
in_port	New in_port for checking flow table
table_id	Checking flow tables

Example:

```
actions += [parser.NXActionResubmit(in_port=8080,
                                    table_id=10)]
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`NXActionOutputReg`**(*ofs_nbits*, *src*, *max_len*, *type_=None*, *len_=None*, *experimenter=None*, *subtype=None*)

Add output action

This action outputs the packet to the OpenFlow port number read from src.

And equivalent to the followings action of ovs-ofctl command.

output:*src*[*start*...*end*]

Attribute	Description
ofs_nbits	Start and End for the OXM/NXM field. Setting method refer to the `nicira_ext.ofs_nbits`
src	OXM/NXM header for source field
max_len	Max length to send to controller

Example:

```
actions += [parser.NXActionOutputReg(
                ofs_nbits=nicira_ext.ofs_nbits(4, 31),
                src="reg0",
                max_len=1024)]
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`NXActionOutputReg2`**(*ofs_nbits*, *src*, *max_len*, *type_=None*, *len_=None*, *experimenter=None*, *subtype=None*)

Add output action

This action outputs the packet to the OpenFlow port number read from src.

And equivalent to the followings action of ovs-ofctl command.

output:*src*[*start*...*end*]

Note: Like the `NXActionOutputReg` but organized so that there is room for a 64-bit experimenter OXM as ‘src’.

Attribute	Description
ofs_nbits	Start and End for the OXM/NXM field. Setting method refer to the `nicira_ext.ofs_nbits`
src	OXM/NXM header for source field
max_len	Max length to send to controller

Example:

```
actions += [parser.NXActionOutputReg2(
                ofs_nbits=nicira_ext.ofs_nbits(4, 31),
                src="reg0",
                max_len=1024)]
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`NXActionLearn`**(*table_id*, *specs*, *idle_timeout=0*, *hard_timeout=0*, *priority=32768*, *cookie=0*, *flags=0*, *fin_idle_timeout=0*, *fin_hard_timeout=0*, *type_=None*, *len_=None*, *experimenter=None*, *subtype=None*)

Adds or modifies flow action

This action adds or modifies a flow in OpenFlow table.

And equivalent to the followings action of ovs-ofctl command.

learn(*argument*[,*argument*]...)

Attribute	Description
table_id	The table in which the new flow should be inserted
specs	Adds a match criterion to the new flow Please use the `NXFlowSpecMatch` in order to set the following format *field*=*value* *field*[*start*..*end*] = *src*[*start*..*end*] *field*[*start*..*end*] Please use the `NXFlowSpecLoad` in order to set the following format **load**:*value*->*dst* [*start*..*end*] **load**:*src*[*start*..*end*] ->*dst*[*start*..*end*] Please use the `NXFlowSpecOutput` in order to set the following format **output:**field[*start*..*end*]
idle_timeout	Idle time before discarding(seconds)
hard_timeout	Max time before discarding(seconds)
priority	Priority level of flow entry
cookie	Cookie for new flow
flags	send_flow_rem
fin_idle_timeout	Idle timeout after FIN(seconds)
fin_hard_timeout	Hard timeout after FIN(seconds)

Caution: The arguments specify the flow's match fields, actions, and other properties, as follows. At least one match criterion and one action argument should ordinarily be specified.

Example:

```
actions += [
    parser.NXActionLearn(able_id=10,
         specs=[parser.NXFlowSpecMatch(src=0x800,
                                       dst=('eth_type_nxm', 0),
                                       n_bits=16),
                parser.NXFlowSpecMatch(src=('reg1', 1),
                                       dst=('reg2', 3),
                                       n_bits=5),
                parser.NXFlowSpecMatch(src=('reg3', 1),
                                       dst=('reg3', 1),
                                       n_bits=5),
                parser.NXFlowSpecLoad(src=0,
                                      dst=('reg4', 3),
                                      n_bits=5),
                parser.NXFlowSpecLoad(src=('reg5', 1),
                                      dst=('reg6', 3),
                                      n_bits=5),
                parser.NXFlowSpecOutput(src=('reg7', 1),
                                        dst="",
                                        n_bits=5)],
         idle_timeout=180,
         hard_timeout=300,
         priority=1,
         cookie=0x64,
         flags=ofproto.OFPFF_SEND_FLOW_REM,
         fin_idle_timeout=180,
         fin_hard_timeout=300)]
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`NXActionExit`**(*type_=None*, *len_=None*, *experimenter=None*, *subtype=None*)

Halt action

This action causes OpenvSwitch to immediately halt execution of further actions.

And equivalent to the followings action of ovs-ofctl command.

exit

Example:

```
actions += [parser.NXActionExit()]
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`NXActionController`**(*max_len*, *controller_id*, *reason*, *type_=None*, *len_=None*, *experimenter=None*, *subtype=None*)

Send packet in message action

This action sends the packet to the OpenFlow controller as a packet in message.

And equivalent to the followings action of ovs-ofctl command.

controller(*key=value...***)**

Attribute	Description
max_len	Max length to send to controller
controller_id	Controller ID to send packet-in
reason	Reason for sending the message

Example:

```
actions += [
    parser.NXActionController(max_len=1024,
                              controller_id=1,
                              reason=ofproto.OFPR_INVALID_TTL)]
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`NXActionController2`**(*type_=None*, *len_=None*, *vendor=None*, *subtype=None*, ***kwargs*)

Send packet in message action

This action sends the packet to the OpenFlow controller as a packet in message.

And equivalent to the followings action of ovs-ofctl command.

controller(*key*=*value*...)

Attribute	Description
max_len	Max length to send to controller
controller_id	Controller ID to send packet-in
reason	Reason for sending the message
userdata	Additional data to the controller in the packet-in message
pause	Flag to pause pipeline to resume later

Example:

```
actions += [
    parser.NXActionController(max_len=1024,
                              controller_id=1,
                              reason=ofproto.OFPR_INVALID_TTL,
                              userdata=[0xa,0xb,0xc],
                              pause=True)]
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`NXActionDecTtlCntIds`**(*cnt_ids*, *type_=None*, *len_=None*, *experimenter=None*, *subtype=None*)

Decrement TTL action

This action decrements TTL of IPv4 packet or hop limits of IPv6 packet.

And equivalent to the followings action of ovs-ofctl command.

dec_ttl(*id1*[,*id2*]...)

Attribute	Description
cnt_ids	Controller ids

Example:

```
actions += [parser.NXActionDecTtlCntIds(cnt_ids=[1,2,3])]
```

Note: If you want to set the following ovs-ofctl command. Please use `OFPActionDecNwTtl`.

dec_ttl

class `ryu.ofproto.ofproto_v1_3_parser.`**`NXActionStackPush`**(*field*, *start*, *end*, *type_=None*, *len_=None*, *experimenter=None*, *subtype=None*)

Push field action

This action pushes field to top of the stack.

And equivalent to the followings action of ovs-ofctl command.

pop:*dst*[*start*...*end*]

Attribute	Description
field	OXM/NXM header for source field
start	Start bit for source field
end	End bit for source field

Example:

```
actions += [parser.NXActionStackPush(field="reg2",
                                     start=0,
                                     end=5)]
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`NXActionStackPop`**(*field*, *start*, *end*, *type_=None*, *len_=None*, *experimenter=None*, *subtype=None*)

Pop field action

This action pops field from top of the stack.

And equivalent to the followings action of ovs-ofctl command.

pop:*src*[*start*...*end*]

Attribute	Description
field	OXM/NXM header for destination field
start	Start bit for destination field
end	End bit for destination field

Example:

```
actions += [parser.NXActionStackPop(field="reg2",
                                    start=0,
                                    end=5)]
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`NXActionSample`**(*probability*, *collector_set_id=0*, *obs_domain_id=0*, *obs_point_id=0*, *type_=None*, *len_=None*, *experimenter=None*, *subtype=None*)

Sample packets action

This action samples packets and sends one sample for every sampled packet.

And equivalent to the followings action of ovs-ofctl command.

sample(*argument*[,*argument*]...)

Attribute	Description
probability	The number of sampled packets
collector_set_id	The unsigned 32-bit integer identifier of the set of sample collectors to send sampled packets to
obs_domain_id	The Unsigned 32-bit integer Observation Domain ID
obs_point_id	The unsigned 32-bit integer Observation Point ID

Example:

```
actions += [parser.NXActionSample(probability=3,
                                  collector_set_id=1,
                                  obs_domain_id=2,
                                  obs_point_id=3,)]
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`NXActionSample2`**(*probability*, *collector_set_id=0*, *obs_domain_id=0*, *obs_point_id=0*, *sampling_port=0*, *type_=None*, *len_=None*, *experimenter=None*, *subtype=None*)

Sample packets action

This action samples packets and sends one sample for every sampled packet. 'sampling_port' can be equal to ingress port or one of egress ports.

And equivalent to the followings action of ovs-ofctl command.

sample(*argument*[,*argument*]...)

Attribute	Description
probability	The number of sampled packets
collector_set_id	The unsigned 32-bit integer identifier of the set of sample collectors to send sampled packets to
obs_domain_id	The Unsigned 32-bit integer Observation Domain ID
obs_point_id	The unsigned 32-bit integer Observation Point ID
sampling_port	Sampling port number

Example:

```
actions += [parser.NXActionSample2(probability=3,
                                   collector_set_id=1,
                                   obs_domain_id=2,
                                   obs_point_id=3,
                                   sampling_port=8080)]
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`NXActionFinTimeout`**(*fin_idle_timeout*, *fin_hard_timeout*, *type_=None*, *len_=None*, *experimenter=None*, *subtype=None*)

Change TCP timeout action

This action changes the idle timeout or hard timeout or both, of this OpenFlow rule when the rule matches a TCP packet with the FIN or RST flag.

And equivalent to the followings action of ovs-ofctl command.

fin_timeout(*argument*[,*argument*]...)

Attribute	Description
fin_idle_timeout	Causes the flow to expire after the given number of seconds of inactivity
fin_idle_timeout	Causes the flow to expire after the given number of second, regardless of activity

Example:

```
match = parser.OFPMatch(ip_proto=6, eth_type=0x0800)
actions += [parser.NXActionFinTimeout(fin_idle_timeout=30,
                                      fin_hard_timeout=60)]
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`NXActionConjunction`**(*clause*, *n_clauses*, *id_*, *type_=None*, *len_=None*, *experimenter=None*, *subtype=None*)

Conjunctive matches action

This action ties groups of individual OpenFlow flows into higher-level conjunctive flows. Please refer to the ovs-ofctl command manual for details.

And equivalent to the followings action of ovs-ofctl command.

conjunction(*id*,*k*/*n*)

Attribute	Description
clause	Number assigned to the flow's dimension
n_clauses	Specify the conjunctive flow's match condition
id_	Conjunction ID

Example:

```
actions += [parser.NXActionConjunction(clause=1,
                                       n_clauses=2,
                                       id_=10)]
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`NXActionMultipath`**(*fields*, *basis*, *algorithm*, *max_link*, *arg*, *ofs_nbits*, *dst*, *type_=None*, *len_=None*, *experimenter=None*, *subtype=None*)

Select multipath link action

This action selects multipath link based on the specified parameters. Please refer to the ovs-ofctl command manual for details.

And equivalent to the followings action of ovs-ofctl command.

multipath(*fields*, *basis*, *algorithm*, *n_links*, *arg*, *dst*[*start*..*end*])

Attribute	Description
fields	One of NX_HASH_FIELDS_*
basis	Universal hash parameter
algo-rithm	One of NX_MP_ALG_*.
max_link	Number of output links
arg	Algorithm-specific argument
ofs_nbits	Start and End for the OXM/NXM field. Setting method refer to the `nicira_ext.ofs_nbits`
dst	OXM/NXM header for source field

Example:

```
actions += [parser.NXActionMultipath(
                fields=nicira_ext.NX_HASH_FIELDS_SYMMETRIC_L4,
                basis=1024,
                algorithm=nicira_ext.NX_MP_ALG_HRW,
                max_link=5,
                arg=0,
                ofs_nbits=nicira_ext.ofs_nbits(4, 31),
                dst="reg2")]
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`NXActionBundle`**(*algorithm*, *fields*, *basis*, *slave_type*, *n_slaves*, *ofs_nbits*, *dst*, *slaves*)

Select bundle link action

This action selects bundle link based on the specified parameters. Please refer to the ovs-ofctl command manual for details.

And equivalent to the followings action of ovs-ofctl command.

bundle(*fields*, *basis*, *algorithm*, *slave_type*, *slaves*:[*s1*, *s2*,...])

Attribute	Description
algorithm	One of NX_MP_ALG_*.
fields	One of NX_HASH_FIELDS_*
basis	Universal hash parameter
slave_type	Type of slaves(must be NXM_OF_IN_PORT)
n_slaves	Number of slaves
ofs_nbits	Start and End for the OXM/NXM field. (must be zero)
dst	OXM/NXM header for source field(must be zero)
slaves	List of slaves

Example:

```
actions += [parser.NXActionBundle(
                algorithm=nicira_ext.NX_MP_ALG_HRW,
                fields=nicira_ext.NX_HASH_FIELDS_ETH_SRC,
                basis=0,
                slave_type=nicira_ext.NXM_OF_IN_PORT,
                n_slaves=2,
                ofs_nbits=0,
                dst=0,
                slaves=[2, 3])]
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`NXActionBundleLoad`**(*algorithm*, *fields*, *basis*, *slave_type*, *n_slaves*, *ofs_nbits*, *dst*, *slaves*)

Select bundle link action

This action has the same behavior as the bundle action, with one exception. Please refer to the ovs-ofctl command manual for details.

And equivalent to the followings action of ovs-ofctl command.

bundle_load(*fields*, *basis*, *algorithm*, *slave_type*, *dst*[*start*... *emd*], *slaves*:[*s1*, *s2*,...]) |

Attribute	Description
algorithm	One of NX_MP_ALG_*.
fields	One of NX_HASH_FIELDS_*
basis	Universal hash parameter
slave_type	Type of slaves(must be NXM_OF_IN_PORT)
n_slaves	Number of slaves
ofs_nbits	Start and End for the OXM/NXM field. Setting method refer to the `nicira_ext.ofs_nbits`
dst	OXM/NXM header for source field
slaves	List of slaves

Example:

```
actions += [parser.NXActionBundleLoad(
                algorithm=nicira_ext.NX_MP_ALG_HRW,
                fields=nicira_ext.NX_HASH_FIELDS_ETH_SRC,
                basis=0,
                slave_type=nicira_ext.NXM_OF_IN_PORT,
                n_slaves=2,
                ofs_nbits=nicira_ext.ofs_nbits(4, 31),
                dst="reg0",
                slaves=[2, 3])]
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`NXActionCT`**(*flags*, *zone_src*, *zone_ofs_nbits*, *recirc_table*, *alg*, *actions*, *type_=None*, *len_=None*, *experimenter=None*, *subtype=None*)

Pass traffic to the connection tracker action

This action sends the packet through the connection tracker.

And equivalent to the followings action of ovs-ofctl command.

ct(*argument*[,*argument*]...)

Attribute	Description
flags	Zero or more(Unspecified flag bits must be zero.)
zone_src	OXM/NXM header for source field
zone_ofs_nbits	Start and End for the OXM/NXM field. Setting method refer to the `nicira_ext.ofs_nbits`. If you need set the Immediate value for zone, zone_src must be set to None or empty character string.
recirc_table	Recirculate to a specific table
alg	Well-known port number for the protocol
actions	Zero or more actions may immediately follow this action

Note: If you set number to zone_src, Traceback occurs when you run the to_jsondict.

Example:

```
match = parser.OFPMatch(eth_type=0x0800, ct_state=(0,32))
actions += [parser.NXActionCT(
                flags = 1,
                zone_src = "reg0",
                zone_ofs_nbits = nicira_ext.ofs_nbits(4, 31),
                recirc_table = 4,
                alg = 0,
                actions = [])]
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`NXActionNAT`**(*flags*, *range_ipv4_min=''*, *range_ipv4_max=''*, *range_ipv6_min=''*, *range_ipv6_max=''*, *range_proto_min=None*, *range_proto_max=None*, *type_=None*, *len_=None*, *experimenter=None*, *subtype=None*)

Network address translation action

This action sends the packet through the connection tracker.

And equivalent to the followings action of ovs-ofctl command.

Note: The following command image does not exist in ovs-ofctl command manual and has been created from the command response.

nat(src=*ip_min***-***ip_max* **:** *proto_min***-***proto-max***)**

Attribute	Description
flags	Zero or more(Unspecified flag bits must be zero.)
range_ipv4_min	Range ipv4 address minimun
range_ipv4_max	Range ipv4 address maximun
range_ipv6_min	Range ipv6 address minimun
range_ipv6_max	Range ipv6 address maximun
range_proto_min	Range protocol minimum
range_proto_max	Range protocol maximun

Caution: `NXActionNAT` must be defined in the actions in the `NXActionCT`.

Example:

```
match = parser.OFPMatch(eth_type=0x0800)
actions += [
    parser.NXActionCT(
        flags = 1,
        zone_src = "reg0",
        zone_ofs_nbits = nicira_ext.ofs_nbits(4, 31),
        recirc_table = 255,
        alg = 0,
        actions = [
            parser.NXActionNAT(
                flags = 1,
                range_ipv4_min = "10.1.12.0",
                range_ipv4_max = "10.1.13.255",
                range_ipv6_min = "",
                range_ipv6_max = "",
                range_proto_min = 1,
                range_proto_max = 1023
            )
        ]
    )
]
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`NXActionOutputTrunc`**(*port*, *max_len*, *type_=None*, *len_=None*, *experimenter=None*, *subtype=None*)

Truncate output action

This action truncate a packet into the specified size and outputs it.

And equivalent to the followings action of ovs-ofctl command.

output(port=*port***,max_len=***max_len***)**

Attribute	Description
port	Output port
max_len	Max bytes to send

Example:

```
actions += [parser.NXActionOutputTrunc(port=8080,
                                        max_len=1024)]
```

class `ryu.ofproto.ofproto_v1_3_parser.`**`NXFlowSpecMatch`**(*src*, *dst*, *n_bits*)
Specification for adding match criterion

This class is used by `NXActionLearn`.

For the usage of this class, please refer to `NXActionLearn`.

Attribute	Description
src	OXM/NXM header and Start bit for source field
dst	OXM/NXM header and Start bit for destination field
n_bits	The number of bits from the start bit

class `ryu.ofproto.ofproto_v1_3_parser.`**`NXFlowSpecLoad`**(*src*, *dst*, *n_bits*)
Add NXAST_REG_LOAD actions

This class is used by `NXActionLearn`.

For the usage of this class, please refer to `NXActionLearn`.

Attribute	Description
src	OXM/NXM header and Start bit for source field
dst	OXM/NXM header and Start bit for destination field
n_bits	The number of bits from the start bit

class `ryu.ofproto.ofproto_v1_3_parser.`**`NXFlowSpecOutput`**(*src*, *n_bits*, *dst=''*)
Add an OFPAT_OUTPUT action

This class is used by `NXActionLearn`.

For the usage of this class, please refer to `NXActionLearn`.

Attribute	Description
src	OXM/NXM header and Start bit for source field
dst	Must be ‘’
n_bits	The number of bits from the start bit

`ryu.ofproto.nicira_ext.`**`ofs_nbits`**(*start*, *end*)
The utility method for ofs_nbits

This method is used in the class to set the ofs_nbits.

This method converts start/end bits into ofs_nbits required to specify the bit range of OXM/NXM fields.

ofs_nbits can be calculated as following:

```
ofs_nbits = (start << 6) + (end - start)
```

The parameter start/end means the OXM/NXM field of ovs-ofctl command.

field[*start*..*end*]

Attribute	Description
start	Start bit for OXM/NXM field
end	End bit for OXM/NXM field

2.6.2 Nicira Extended Match Structures

The API of this class is the same as `OFPMatch`.

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

Argument	Value	Description
in_port_nxm	Integer 16bit	OpenFlow port number.
eth_dst_nxm	MAC address	Ethernet destination address.
eth_src_nxm	MAC address	Ethernet source address.
eth_type_nxm	Integer 16bit	Ethernet type. Needed to support Nicira extensions that require the eth_type to be set. (i.e. tcp_
vlan_tci	Integer 16bit	VLAN TCI. Basically same as vlan_vid plus vlan_pcp.
nw_tos	Integer 8bit	IP ToS or IPv6 traffic class field dscp. Requires setting fields: eth_type_nxm = [0x0800 (IPv4)
ip_proto_nxm	Integer 8bit	IP protocol. Needed to support Nicira extensions that require the ip_proto to be set. (i.e. tcp_fl
ipv4_src_nxm	IPv4 address	IPv4 source address. Requires setting fields: eth_type_nxm = 0x0800 (IPv4)
ipv4_dst_nxm	IPv4 address	IPv4 destination address. Requires setting fields: eth_type_nxm = 0x0800 (IPv4)
tcp_src_nxm	Integer 16bit	TCP source port. Requires setting fields: eth_type_nxm = [0x0800 (IPv4)|0x86dd (IPv6)] and
tcp_dst_nxm	Integer 16bit	TCP destination port. Requires setting fields: eth_type_nxm = [0x0800 (IPv4)|0x86dd (IPv6)]
udp_src_nxm	Integer 16bit	UDP source port. Requires setting fields: eth_type_nxm = [0x0800 (IPv4)|0x86dd (IPv6)] and
udp_dst_nxm	Integer 16bit	UDP destination port. eth_type_nxm = [0x0800 (IPv4)|0x86dd (IPv6)] and ip_proto_nxm = 17
icmpv4_type_nxm	Integer 8bit	Type matches the ICMP type and code matches the ICMP code. Requires setting fields: eth_ty
icmpv4_code_nxm	Integer 8bit	Type matches the ICMP type and code matches the ICMP code. Requires setting fields: eth_ty
arp_op_nxm	Integer 16bit	Only ARP opcodes between 1 and 255 should be specified for matching. Requires setting field
arp_spa_nxm	IPv4 address	An address may be specified as an IP address or host name. Requires setting fields: eth_type_n
arp_tpa_nxm	IPv4 address	An address may be specified as an IP address or host name. Requires setting fields: eth_type_n
tunnel_id_nxm	Integer 64bit	Tunnel identifier.
arp_sha_nxm	MAC address	An address is specified as 6 pairs of hexadecimal digits delimited by colons. Requires setting f
arp_tha_nxm	MAC address	An address is specified as 6 pairs of hexadecimal digits delimited by colons. Requires setting f
ipv6_src_nxm	IPv6 address	IPv6 source address. Requires setting fields: eth_type_nxm = 0x86dd (IPv6)
ipv6_dst_nxm	IPv6 address	IPv6 destination address. Requires setting fields: eth_type_nxm = 0x86dd (IPv6)
icmpv6_type_nxm	Integer 8bit	Type matches the ICMP type and code matches the ICMP code. Requires setting fields: eth_ty
icmpv6_code_nxm	Integer 8bit	Type matches the ICMP type and code matches the ICMP code. Requires setting fields: eth_ty
nd_target	IPv6 address	The target address ipv6. Requires setting fields: eth_type_nxm = 0x86dd (IPv6) and ip_proto_
nd_sll	MAC address	The source link-layer address option. Requires setting fields: eth_type_nxm = 0x86dd (IPv6) a
nd_tll	MAC address	The target link-layer address option. Requires setting fields: eth_type_nxm = 0x86dd (IPv6) ar
ip_frag	Integer 8bit	frag_type specifies what kind of IP fragments or non-fragments to match. Requires setting fiel
ipv6_label	Integer 32bit	Matches IPv6 flow label. Requires setting fields: eth_type_nxm = 0x86dd (IPv6)
ip_ecn_nxm	Integer 8bit	Matches ecn bits in IP ToS or IPv6 traffic class fields. Requires setting fields: eth_type_nxm =
nw_ttl	Integer 8bit	IP TTL or IPv6 hop limit value ttl. Requires setting fields: eth_type_nxm = [0x0800 (IPv4)|0x
mpls_ttl	Integer 8bit	The TTL of the outer MPLS label stack entry of a packet. Requires setting fields: eth_type_nx
tun_ipv4_src	IPv4 address	Tunnel IPv4 source address. Requires setting fields: eth_type_nxm = 0x0800 (IPv4)
tun_ipv4_dst	IPv4 address	Tunnel IPv4 destination address. Requires setting fields: eth_type_nxm = 0x0800 (IPv4)
pkt_mark	Integer 32bit	Packet metadata mark.
tcp_flags_nxm	Integer 16bit	TCP Flags. Requires setting fields: eth_type_nxm = [0x0800 (IP)|0x86dd (IPv6)] and ip_proto
conj_id	Integer 32bit	Conjunction ID used only with the conjunction action
tun_gbp_id	Integer 16bit	The group policy identifier in the VXLAN header.
tun_gbp_flags	Integer 8bit	The group policy flags in the VXLAN header.
tun_flags	Integer 16bit	Flags indicating various aspects of the tunnel encapsulation.
ct_state	Integer 32bit	Conntrack state.
ct_zone	Integer 16bit	Conntrack zone.
ct_mark	Integer 32bit	Conntrack mark.
ct_label	Integer 128bit	Conntrack label.

Table 2.5 – continued from previous pa

Argument	Value	Description
tun_ipv6_src	IPv6 address	Tunnel IPv6 source address. Requires setting fields: eth_type_nxm = 0x86dd (IPv6)
tun_ipv6_dst	IPv6 address	Tunnel IPv6 destination address. Requires setting fields: eth_type_nxm = 0x86dd (IPv6)
_recirc_id	Integer 32bit	ID for recirculation.
_dp_hash	Integer 32bit	Flow hash computed in Datapath.
reg<idx>	Integer 32bit	Packet register. <idx> is register number 0-15.
xxreg<idx>	Integer 128bit	Packet extended-extended register. <idx> is register number 0-3.

Note: Setting the TCP flags via the nicira extensions. This is required when using OVS version < 2.4. When using the nxm fields, you need to use any nxm prereq fields as well or you will receive a OFPBMC_BAD_PREREQ error

Example:

```
# WILL NOT work
flag = tcp.TCP_ACK
match = parser.OFPMatch(
    tcp_flags_nxm=(flag, flag),
    ip_proto=inet.IPPROTO_TCP,
    eth_type=eth_type)

# Works
flag = tcp.TCP_ACK
match = parser.OFPMatch(
    tcp_flags_nxm=(flag, flag),
    ip_proto_nxm=inet.IPPROTO_TCP,
    eth_type_nxm=eth_type)
```

2.7 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**(**args*, ***kwargs*)
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.

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 3

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
```

3.2 Topology Viewer

ryu.app.gui_topology.gui_topology provides topology visualization.

This depends on following ryu applications.

ryu.app.rest_topology	Get node and link data.
ryu.app.ws_topology	Being notified change of link up/down.
ryu.app.ofctl_rest	Get flows of datapaths.

3.2.1 Usage

Run mininet (or join your real environment):

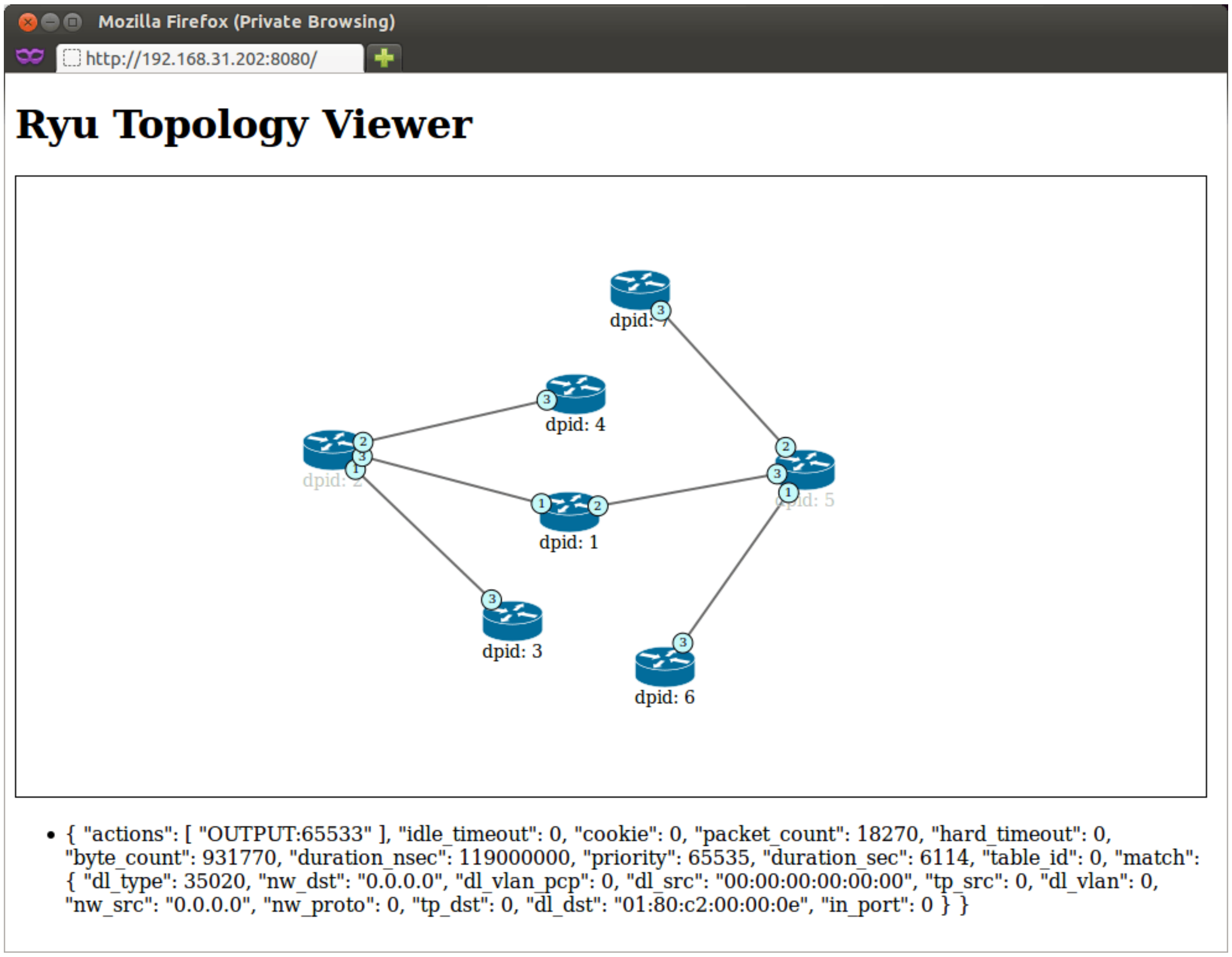
```
$ sudo mn --controller remote --topo tree,depth=3
```

Run GUI application:

```
$ PYTHONPATH=. ./bin/ryu run --observe-links ryu/app/gui_topology/gui_topology.py
```

Access http://<ip address of ryu host>:8080 with your web browser.

3.2.2 Screenshot


Mozilla Firefox (Private Browsing)
http://192.168.31.202:8080/
dpid: 7
dpid: 4
dpid: 2
dpid: 1
dpid: 5
dpid: 3
dpid: 6

Ryu Topology Viewer

- { "actions": ["OUTPUT:65533"], "idle_timeout": 0, "cookie": 0, "packet_count": 18270, "hard_timeout": 0, "byte_count": 931770, "duration_nsec": 119000000, "priority": 65535, "duration_sec": 6114, "table_id": 0, "match": { "dl_type": 35020, "nw_dst": "0.0.0.0", "dl_vlan_pcp": 0, "dl_src": "00:00:00:00:00:00", "tp_src": 0, "dl_vlan": 0, "nw_src": "0.0.0.0", "nw_proto": 0, "tp_dst": 0, "dl_dst": "01:80:c2:00:00:0e", "in_port": 0 } }

CHAPTER 4

Tests

4.1 Testing VRRP Module

This page describes how to test Ryu VRRP service

4.1.1 Running integrated tests

Some testing scripts are available.

- ryu/tests/integrated/test_vrrp_linux_multi.py
- ryu/tests/integrated/test_vrrp_multi.py

Each files include how to run in the comment. Please refer to it.

4.1.2 Running multiple Ryu VRRP in network namespace

The following command lines set up necessary bridges and interfaces.

And then run RYU-VRRP:

```
# ip netns add gateway1
# ip netns add gateway2

# brctl addbr vrrp-br0
# brctl addbr vrrp-br1

# ip link add veth0 type veth peer name veth0-br0
# ip link add veth1 type veth peer name veth1-br0
# ip link add veth2 type veth peer name veth2-br0
# ip link add veth3 type veth peer name veth3-br1
# ip link add veth4 type veth peer name veth4-br1
# ip link add veth5 type veth peer name veth5-br1
# brctl addif vrrp-br0 veth0-br0
# brctl addif vrrp-br0 veth1-br0
# brctl addif vrrp-br0 veth2-br0
# brctl addif vrrp-br1 veth3-br1
# brctl addif vrrp-br1 veth4-br1
# brctl addif vrrp-br1 veth5-br1

# ip link set vrrp-br0 up
# ip link set vrrp-br1 up

# ip link set veth0 up
# ip link set veth0-br0 up
# ip link set veth1-br0 up
# ip link set veth2-br0 up
# ip link set veth3-br1 up
# ip link set veth4-br1 up
# ip link set veth5 up
# ip link set veth5-br1 up

# ip link set veth1 netns gateway1
# ip link set veth2 netns gateway2
# ip link set veth3 netns gateway1
# ip link set veth4 netns gateway2

# ip netns exec gateway1 ip link set veth1 up
# ip netns exec gateway2 ip link set veth2 up
# ip netns exec gateway1 ip link set veth3 up
# ip netns exec gateway2 ip link set veth4 up

# ip netns exec gateway1 .ryu-vrrp veth1 '10.0.0.2' 254
# ip netns exec gateway2 .ryu-vrrp veth2 '10.0.0.3' 100
```

Caveats

Please make sure that all interfaces and bridges are UP. Don't forget interfaces in netns gateway1/gateway2.

```
                ^ veth5
                |
                V veth5-br1
        -----------------------
        |Linux Brirge vrrp-br1|
        -----------------------
veth3-br1^             ^ veth4-br1
         |             |
    veth3V             V veth4
    ----------        ----------
    |netns   |        |netns   |
    |gateway1|        |gateway2|
    |ryu-vrrp|        |ryu-vrrp|
    ----------        ----------
    veth1^             ^ veth2
         |             |
veth1-br0V             V veth2-br0
        -----------------------
        |Linux Brirge vrrp-br0|
        -----------------------
```

```
                ^ veth0-br0
                |
                V veth0
```

Here's the helper executable, ryu-vrrp:

```
#!/usr/bin/env python
#
# Copyright (C) 2013 Nippon Telegraph and Telephone Corporation.
# Copyright (C) 2013 Isaku Yamahata <yamahata at valinux co jp>
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#    http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
# implied.
# See the License for the specific language governing permissions and
# limitations under the License.

from ryu.lib import hub
hub.patch()

# TODO:
#   Right now, we have our own patched copy of ovs python bindings
#   Once our modification is upstreamed and widely deployed,
#   use it
#
# NOTE: this modifies sys.path and thus affects the following imports.
# eg. oslo.config.cfg.
import ryu.contrib

from oslo.config import cfg
import logging
import netaddr
import sys
import time

from ryu import log
log.early_init_log(logging.DEBUG)

from ryu import flags
from ryu import version
from ryu.base import app_manager
from ryu.controller import controller
from ryu.lib import mac as lib_mac
from ryu.lib.packet import vrrp
from ryu.services.protocols.vrrp import api as vrrp_api
from ryu.services.protocols.vrrp import event as vrrp_event


CONF = cfg.CONF

_VRID = 7
```

```
_IP_ADDRESS = '10.0.0.1'
_PRIORITY = 100


class VRRPTestRouter(app_manager.RyuApp):
    def __init__(self, *args, **kwargs):
        super(VRRPTestRouter, self).__init__(*args, **kwargs)
        print args
        self.logger.debug('vrrp_config %s', args)
        self._ifname = args[0]
        self._primary_ip_address = args[1]
        self._priority = int(args[2])

    def start(self):
        print 'start'
        hub.spawn(self._main)

    def _main(self):
        print self
        interface = vrrp_event.VRRPInterfaceNetworkDevice(
            lib_mac.DONTCARE, self._primary_ip_address, None, self._ifname)
        self.logger.debug('%s', interface)

        ip_addresses = [_IP_ADDRESS]
        config = vrrp_event.VRRPConfig(
            version=vrrp.VRRP_VERSION_V3, vrid=_VRID, priority=self._priority,
            ip_addresses=ip_addresses)
        self.logger.debug('%s', config)

        rep = vrrp_api.vrrp_config(self, interface, config)
        self.logger.debug('%s', rep)


def main():
    vrrp_config = sys.argv[-3:]
    sys.argv = sys.argv[:-3]
    CONF(project='ryu', version='ryu-vrrp %s' % version)

    log.init_log()
    # always enable ofp for now.
    app_lists = ['ryu.services.protocols.vrrp.manager',
                 'ryu.services.protocols.vrrp.dumper',
                 'ryu.services.protocols.vrrp.sample_manager']

    app_mgr = app_manager.AppManager.get_instance()
    app_mgr.load_apps(app_lists)
    contexts = app_mgr.create_contexts()
    app_mgr.instantiate_apps(**contexts)
    vrrp_router = app_mgr.instantiate(VRRPTestRouter, *vrrp_config, **contexts)
    vrrp_router.start()

    while True:
        time.sleep(999999)

    app_mgr.close()


if __name__ == "__main__":
    main()
```

4.2 Testing OF-config support with LINC

This page describes how to setup LINC and test Ryu OF-config with it.

The procedure is as follows. Although all the procedure is written for reader's convenience, please refer to LINC document for latest informations of LINC.

> https://github.com/FlowForwarding/LINC-Switch

The test procedure

- install Erlang environment
- build LINC
- configure LINC switch
- setup for LINC
- run LINC switch
- run Ryu test_of_config app

For getting/installing Ryu itself, please refer to http://osrg.github.io/ryu/

4.2.1 Install Erlang environment

Since LINC is written in Erlang, you need to install Erlang execution environment. Required version is R15B+.

The easiest way is to use binary package from https://www.erlang-solutions.com/downloads/download-erlang-otp

The distribution may also provide Erlang package.

4.2.2 build LINC

install necessary packages for build

install necessary build tools

On Ubuntu:

```
# apt-get install git-core bridge-utils libpcap0.8 libpcap-dev libcap2-bin uml-
↪utilities
```

On RedHat/CentOS:

```
# yum install git sudo bridge-utils libpcap libpcap-devel libcap tunctl
```

Note that on RedHat/CentOS 5.x you need a newer version of libpcap:

```
# yum erase libpcap libpcap-devel
# yum install flex byacc
# wget http://www.tcpdump.org/release/libpcap-1.2.1.tar.gz
# tar xzf libpcap-1.2.1.tar.gz
# cd libpcap-1.2.1
# ./configure
# make && make install
```

get LINC repo and built

Clone LINC repo:

```
% git clone git://github.com/FlowForwarding/LINC-Switch.git
```

Then compile everything:

```
% cd LINC-Switch
% make
```

Note: At the time of this writing, test_of_config fails due to a bug of LINC. You can try this test with LINC which is built by the following methods.

```
% cd LINC-Switch
% make
% cd deps/of_config
% git reset --hard f772af4b765984381ad024ca8e5b5b8c54362638
% cd ../..
% make offline
```

4.2.3 Setup LINC

edit LINC switch configuration file. `rel/linc/releases/0.1/sys.config` Here is the sample sys.config for test_of_config.py to run.

```
[{linc,
     [{of_config,enabled},
      {capable_switch_ports,
          [{port,1,[{interface,"linc-port"}]},
           {port,2,[{interface,"linc-port2"}]},
           {port,3,[{interface,"linc-port3"}]},
           {port,4,[{interface,"linc-port4"}]}]},
      {capable_switch_queues,
          [
            {queue,991,[{min_rate,10},{max_rate,120}]},
            {queue,992,[{min_rate,10},{max_rate,130}]},
            {queue,993,[{min_rate,200},{max_rate,300}]},
            {queue,994,[{min_rate,400},{max_rate,900}]}
            ]},
      {logical_switches,
          [{switch,0,
               [{backend,linc_us4},
                {controllers,[{"Switch0-Default-Controller","127.0.0.1",6633,tcp}]},
                {controllers_listener,{"127.0.0.1",9998,tcp}},
                {queues_status,enabled},
                {ports,[{port,1,{queues,[]}},{port,2,{queues,[991,992]}}]}]}
                ,
          {switch,7,
              [{backend,linc_us3},
               {controllers,[{"Switch7-Controller","127.0.0.1",6633,tcp}]},
               {controllers_listener,disabled},
               {queues_status,enabled},
               {ports,[{port,4,{queues,[]}},{port,3,{queues,[993,994]}}]}]}
       ]}]},
 {enetconf,
     [{capabilities,
          [{base,{1,0}},
           {base,{1,1}},
           {startup,{1,0}},
           {'writable-running',{1,0}}]},
      {callback_module,linc_ofconfig},
      {sshd_ip,{127,0,0,1}},
      {sshd_port,1830},
      {sshd_user_passwords,[{"linc","linc"}]}]},
 {lager,
     [{handlers,
          [{lager_console_backend,debug},
           {lager_file_backend,
              [{"log/error.log",error,10485760,"$D0",5},
               {"log/console.log",info,10485760,"$D0",5}]}]}]},
 {sasl,
     [{sasl_error_logger,{file,"log/sasl-error.log"}},
      {errlog_type,error},
      {error_logger_mf_dir,"log/sasl"},
      {error_logger_mf_maxbytes,10485760},
      {error_logger_mf_maxfiles,5}]},
 {sync,[{excluded_modules,[procket]}]}].
```

4.2.4 setup for LINC

As the above sys.config requires some network interface, create them:

```
# ip link add linc-port type veth peer name linc-port-peer
# ip link set linc-port up
# ip link add linc-port2 type veth peer name linc-port-peer2
# ip link set linc-port2 up
# ip link add linc-port3 type veth peer name linc-port-peer3
# ip link set linc-port3 up
# ip link add linc-port4 type veth peer name linc-port-peer4
# ip link set linc-port4 up
```

After stopping LINC, those created interfaces can be deleted:

```
# ip link delete linc-port
# ip link delete linc-port2
# ip link delete linc-port3
# ip link delete linc-port4
```

4.2.5 Starting LINC OpenFlow switch

Then run LINC:

```
# rel/linc/bin/linc console
```

4.2.6 Run Ryu test_of_config app

Run test_of_config app:

```
# ryu-manager --verbose ryu.tests.integrated.test_of_config ryu.app.rest
```

If you don’t install ryu and are working in the git repo directly:

```
# PYTHONPATH=. ./bin/ryu-manager --verbose ryu.tests.integrated.test_of_config ryu.
↪app.rest
```

CHAPTER 5

Snort Intergration

This document describes how to integrate Ryu with Snort.

5.1 Overview

There are two options can send alert to Ryu controller. The Option 1 is easier if you just want to demonstrate or test. Since Snort need very large computation power for analyzing packets you can choose Option 2 to separate them.

[Option 1] Ryu and Snort are on the same machine

```
      +---------------------+
      |      unixsock       |
      |    Ryu  ==  snort   |
      +----eth0-----eth1----+
             |       |
+-------+   +----------+   +-------+
| HostA |---| OFSwitch |---| HostB |
+-------+   +----------+   +-------+
```

The above depicts Ryu and Snort architecture. Ryu receives Snort alert packet via **Unix Domain Socket** . To monitor packets between HostA and HostB, installing a flow that mirrors packets to Snort.

[Option 2] Ryu and Snort are on the different machines

```
          +---------------+
          |    Snort     eth0--|
          |   Sniffer     |    |
          +-----eth1------+    |
                 |             |
+-------+   +----------+   +-----------+
| HostA |---| OFSwitch |---| LAN (*CP) |
+-------+   +----------+   +-----------+
                 |             |
            +----------+   +----------+
            |  HostB    |    |    Ryu    |
            +-----------+    +-----------+
```

***CP: Control Plane**

The above depicts Ryu and Snort architecture. Ryu receives Snort alert packet via **Network Socket** . To monitor packets between HostA and HostB, installing a flow that mirrors packets to Snort.

5.2 Installation Snort

Snort is an open source network intrusion prevention and detectionsystem developed by Sourcefire. If you are not familiar with installing/setting up Snort, please referto snort setup guides.

http://www.snort.org/documents

5.3 Configure Snort

The configuration example is below:

- Add a snort rules file into `/etc/snort/rules` named `Myrules.rules`

```
alert icmp any any -> any any (msg:"Pinging...";sid:1000004;)
alert tcp any any -> any 80 (msg:"Port 80 is accessing"; sid:1000003;)
```

- Add the custom rules in `/etc/snort/snort.conf`

```
include $RULE_PATH/Myrules.rules
```

Configure NIC as a promiscuous mode.

```
$ sudo ifconfig eth1 promisc
```

5.4 Usage

[Option 1]

1. Modify the `simple_switch_snort.py`:

```
socket_config = {'unixsock': True}
# True: Unix Domain Socket Server [Option1]
# False: Network Socket Server [Option2]
```

2. Run Ryu with sample application:

```
$ sudo ./bin/ryu-manager ryu/app/simple_switch_snort.py
```

The incoming packets will all mirror to **port 3** which should be connect to Snort network interface. You can modify the mirror port by assign a new value in the `self.snort_port = 3` of `simple_switch_snort.py`

3. Run Snort:

```
$ sudo -i
$ snort -i eth1 -A unsock -l /tmp -c /etc/snort/snort.conf
```

4. Send an ICMP packet from HostA (192.168.8.40) to HostB (192.168.8.50):

```
$ ping 192.168.8.50
```

5. You can see the result under next section.

[Option 2]

1. Modify the `simple_switch_snort.py`:

```
socket_config = {'unixsock': False}
# True: Unix Domain Socket Server [Option1]
# False: Network Socket Server [Option2]
```

2. Run Ryu with sample application (On the Controller):

```
$ ./bin/ryu-manager ryu/app/simple_switch_snort.py
```

3. Run Snort (On the Snort machine):

```
$ sudo -i
$ snort -i eth1 -A unsock -l /tmp -c /etc/snort/snort.conf
```

4. Run `pigrelay.py` (On the Snort machine):

```
$ sudo python pigrelay.py
```

This program listening snort alert messages from unix domain socket and sending it to Ryu using network socket.

You can clone the source code from this repo. https://github.com/John-Lin/pigrelay

5. Send an ICMP packet from HostA (192.168.8.40) to HostB (192.168.8.50):

```
$ ping 192.168.8.50
```

6. You can see the alert message below:

```
alertmsg: Pinging...
icmp(code=0,csum=19725,data=echo(data=array('B', [97, 98, 99, 100, 101, 102, 103,␣
↪104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119,␣
↪97, 98, 99, 100, 101, 102, 103, 104, 105]),id=1,seq=78),type=8)

ipv4(csum=42562,dst='192.168.8.50',flags=0,header_length=5,identification=724,
↪offset=0,option=None,proto=1,src='192.168.8.40',tos=0,total_length=60,ttl=128,
↪version=4)

ethernet(dst='00:23:54:5a:05:14',ethertype=2048,src='00:23:54:6c:1d:17')


alertmsg: Pinging...
icmp(code=0,csum=21773,data=echo(data=array('B', [97, 98, 99, 100, 101, 102, 103,␣
↪104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119,␣
↪97, 98, 99, 100, 101, 102, 103, 104, 105]),id=1,seq=78),type=0)

ipv4(csum=52095,dst='192.168.8.40',flags=0,header_length=5,identification=7575,
↪offset=0,option=None,proto=1,src='192.168.8.50',tos=0,total_length=60,ttl=64,
↪version=4)
```

CHAPTER 6

Built-in Ryu applications

Ryu has some built-in Ryu applications. Some of them are examples. Others provide some functionalities to other Ryu applications.

6.1 ryu.app.ofctl

ryu.app.ofctl provides a convenient way to use OpenFlow messages synchronously.

OfctlService ryu application is automatically loaded if your Ryu application imports ofctl.api module.

Example:

```
import ryu.app.ofctl.api
```

OfctlService application internally uses OpenFlow barrier messages to ensure message boundaries. As OpenFlow messages are asynchronous and some of messages does not have any replies on success, barriers are necessary for correct error handling.

6.1.1 api module

`ryu.app.ofctl.api.`**`get_datapath`**(*app*, *dpid*)

Get datapath object by dpid.

Parameters

- **`app`** – Client RyuApp instance
- **`dpid`** – Datapath-id (in integer)

Returns None on error.

`ryu.app.ofctl.api.`**`send_msg`**(*app*, *msg*, *reply_cls=None*, *reply_multi=False*)

Send an OpenFlow message and wait for reply messages.

Parameters

- **app** – Client RyuApp instance
- **msg** – An OpenFlow controller-to-switch message to send
- **reply_cls** – OpenFlow message class for expected replies. None means no replies are expected. The default is None.
- **reply_multi** – True if multipart replies are expected. The default is False.

If no replies, returns None. If reply_multi=False, returns OpenFlow switch-to-controller message. If reply_multi=True, returns a list of OpenFlow switch-to-controller messages.

Raise an exception on error.

Example:

```
import ryu.app.ofctl.api as api

msg = parser.OFPPortDescStatsRequest(datapath=datapath)
result = api.send_msg(self, msg,
                      reply_cls=parser.OFPPortDescStatsReply,
                      reply_multi=True)
```

6.1.2 exceptions

exception `ryu.app.ofctl.exception.`**`InvalidDatapath`**(*result*)

Datapath is invalid.

This can happen when the bridge disconnects.

exception `ryu.app.ofctl.exception.`**`OFError`**(*result*)

OFPErrorMsg is received.

exception `ryu.app.ofctl.exception.`**`UnexpectedMultiReply`**(*result*)

Two or more replies are received for reply_muiti=False request.

6.2 ryu.app.ofctl_rest

ryu.app.ofctl_rest provides REST APIs for retrieving the switch stats and Updating the switch stats. This application helps you debug your application and get various statistics.

This application supports OpenFlow version 1.0, 1.2, 1.3, 1.4 and 1.5.

Contents

* *Description of Actions on request messages*

6.2.1 Retrieve the switch stats

Get all switches

Get the list of all switches which connected to the controller.

Usage:

Method	GET
URI	/stats/switches

Response message body:

Attribute	Description	Example
dpid	Datapath ID	1

Example of use:

```
$ curl -X GET http://localhost:8080/stats/switches
```

```
[
  1,
  2,
  3
]
```

Note: The result of the REST command is formatted for easy viewing.

Get the desc stats

Get the desc stats of the switch which specified with Datapath ID in URI.

Usage:

Method	GET
URI	/stats/desc/<dpid>

Response message body:

Attribute	Description	Example
dpid	Datapath ID	"1"
mfr_desc	Manufacturer description	"Nicira, Inc.",
hw_desc	Hardware description	"Open vSwitch",
sw_desc	Software description	"2.3.90",
serial_num	Serial number	"None",
dp_desc	Human readable description of datapath	"None"

Example of use:

```
$ curl -X GET http://localhost:8080/stats/desc/1
```

```
{
  "1": {
    "mfr_desc": "Nicira, Inc.",
    "hw_desc": "Open vSwitch",
    "sw_desc": "2.3.90",
    "serial_num": "None",
    "dp_desc": "None"
  }
}
```

Get all flows stats

Get all flows stats of the switch which specified with Datapath ID in URI.

Usage:

Method	GET
URI	/stats/flow/<dpid>

Response message body(OpenFlow1.3 or earlier):

Attribute	Description	Example
dpid	Datapath ID	"1"
length	Length of this entry	88
table_id	Table ID	0
duration_sec	Time flow has been alive in seconds	2
duration_nsec	Time flow has been alive in nanoseconds beyond duration_sec	6.76e+08
priority	Priority of the entry	11111
idle_timeout	Number of seconds idle before expiration	0
hard_timeout	Number of seconds before expiration	0
flags	Bitmap of OFPFF_* flags	1
cookie	Opaque controller-issued identifier	1
packet_count	Number of packets in flow	0
byte_count	Number of bytes in flow	0
match	Fields to match	{"in_port": 1}
actions	Instruction set	["OUTPUT:2"]

Response message body(OpenFlow1.4 or later):

Attribute	Description	Example
dpid	Datapath ID	"1"
length	Length of this entry	88
table_id	Table ID	0
duration_sec	Time flow has been alive in seconds	2
duration_nsec	Time flow has been alive in nanoseconds beyond duration_sec	6.76e+08
priority	Priority of the entry	11111
idle_timeout	Number of seconds idle before expiration	0
hard_timeout	Number of seconds before expiration	0
flags	Bitmap of OFPFF_* flags	1
cookie	Opaque controller-issued identifier	1
packet_count	Number of packets in flow	0
byte_count	Number of bytes in flow	0
importance	Eviction precedence	0
match	Fields to match	{"eth_type": 2054}
instructions	struct ofp_instruction_header	[{"type":GOTO_TABLE", "table_id":1}]

Example of use:

```
$ curl -X GET http://localhost:8080/stats/flow/1
```

Response (OpenFlow1.3 or earlier):

```
{
  "1": [
    {
      "length": 88,
      "table_id": 0,
      "duration_sec": 2,
      "duration_nsec": 6.76e+08,
      "priority": 11111,
      "idle_timeout": 0,
      "hard_timeout": 0,
      "flags": 1,
      "cookie": 1,
      "packet_count": 0,
      "byte_count": 0,
      "match": {
        "in_port": 1
      },
      "actions": [
        "OUTPUT:2"
      ]
    }
  ]
}
```

Response (OpenFlow1.4 or later):

```
 {
    "1": [
      {
        "length": 88,
        "table_id": 0,
        "duration_sec": 2,
        "duration_nsec": 6.76e+08,
        "priority": 11111,
        "idle_timeout": 0,
        "hard_timeout": 0,
        "flags": 1,
        "cookie": 1,
        "packet_count": 0,
        "byte_count": 0,
        "match": {
          "eth_type": 2054
        },
        "importance": 0,
        "instructions": [
          {
            "type": "APPLY_ACTIONS",
            "actions": [
              {
                "port": 2,
                "max_len": 0,
                "type": "OUTPUT"
              }
            ]
          }
        ]
      }
    ]
}
```

Get flows stats filtered by fields

Get flows stats of the switch filtered by the OFPFlowStats fields. This is POST method version of *Get all flows stats*.

Usage:

Method	POST
URI	/stats/flow/<dpid>

Request message body:

Attribute	Description	Example	Default
table_id	Table ID (int)	0	OF-PTT_ALL
out_port	Require matching entries to include this as an output port (int)	2	OFPP_ANY
out_group	Require matching entries to include this as an output group (int)	1	OFPG_ANY
cookie	Require matching entries to contain this cookie value (int)	1	0
cookie_mask	Mask used to restrict the cookie bits that must match (int)	1	0
match	Fields to match (dict)	{"in_port": 1}	{} #wild-carded
priority	Priority of the entry (int) (See Note)	11111	#wild-carded

Note: OpenFlow Spec does not allow to filter flow entries by priority, but when with a large amount of flow entries, filtering by priority is convenient to get statistics efficiently. So, this app provides priority field for filtering.

Response message body: The same as *Get all flows stats*

Example of use:

```
$ curl -X POST -d '{
     "table_id": 0,
     "out_port": 2,
     "cookie": 1,
     "cookie_mask": 1,
     "match":{
         "in_port":1
     }
 }' http://localhost:8080/stats/flow/1
```

Response (OpenFlow1.3 or earlier):

```
{
  "1": [
    {
      "length": 88,
      "table_id": 0,
      "duration_sec": 2,
      "duration_nsec": 6.76e+08,
      "priority": 11111,
      "idle_timeout": 0,
      "hard_timeout": 0,
      "flags": 1,
      "cookie": 1,
      "packet_count": 0,
      "byte_count": 0,
      "match": {
        "in_port": 1
      },
      "actions": [
        "OUTPUT:2"
      ]
    }
  ]
}
```

Response (OpenFlow1.4 or later):

```
{
   "1": [
     {
       "length": 88,
       "table_id": 0,
       "duration_sec": 2,
       "duration_nsec": 6.76e+08,
       "priority": 11111,
       "idle_timeout": 0,
       "hard_timeout": 0,
       "flags": 1,
       "cookie": 1,
       "packet_count": 0,
       "byte_count": 0,
       "match": {
         "eth_type": 2054
       },
       "importance": 0,
       "instructions": [
         {
           "type": "APPLY_ACTIONS",
           "actions": [
             {
               "port": 2,
               "max_len": 0,
               "type": "OUTPUT"
             }
           ]
         }
       ]
     }
   ]
}
```

Get aggregate flow stats

Get aggregate flow stats of the switch which specified with Datapath ID in URI.

Usage:

Method	GET
URI	/stats/aggregateflow/<dpid>

Response message body:

Attribute	Description	Example
dpid	Datapath ID	“1”
packet_count	Number of packets in flows	18
byte_count	Number of bytes in flows	756
flow_count	Number of flows	3

Example of use:

```
$ curl -X GET http://localhost:8080/stats/aggregateflow/1
```

```
{
  "1": [
    {
      "packet_count": 18,
      "byte_count": 756,
      "flow_count": 3
    }
  ]
}
```

Get aggregate flow stats filtered by fields

Get aggregate flow stats of the switch filtered by the OFPAggregateStats fields. This is POST method version of *Get aggregate flow stats*.

Usage:

Method	POST
URI	/stats/aggregateflow/<dpid>

Request message body:

Attribute	Description	Example	Default
table_id	Table ID (int)	0	OF-PTT_ALL
out_port	Require matching entries to include this as an output port (int)	2	OFPP_ANY
out_group	Require matching entries to include this as an output group (int)	1	OFPG_ANY
cookie	Require matching entries to contain this cookie value (int)	1	0
cookie_mask	Mask used to restrict the cookie bits that must match (int)	1	0
match	Fields to match (dict)	{"in_port": 1}	{} #wild-carded

Response message body: The same as *Get aggregate flow stats*

Example of use:

```
$ curl -X POST -d '{
     "table_id": 0,
     "out_port": 2,
     "cookie": 1,
     "cookie_mask": 1,
     "match":{
         "in_port":1
     }
 }' http://localhost:8080/stats/aggregateflow/1
```

```
{
  "1": [
    {
      "packet_count": 18,
      "byte_count": 756,
      "flow_count": 3
    }
  ]
}
```

Get table stats

Get table stats of the switch which specified with Datapath ID in URI.

Usage:

Method	GET
URI	/stats/table/<dpid>

Response message body(OpenFlow1.0):

Attribute	Description	Example
dpid	Datapath ID	"1"
table_id	Table ID	0
name	Name of Table	"classifier"
max_entries	Max number of entries supported	1e+06
wildcards	Bitmap of OFPFW_* wildcards that are supported by the table	["IN_PORT","DL_VLAN"]
active_count	Number of active entries	0
lookup_count	Number of packets looked up in table	8
matched_count	Number of packets that hit table	0

Response message body(OpenFlow1.2):

Attribute	Description	Example
dpid	Datapath ID	"1"
table_id	Table ID	0
name	Name of Table	"classifier"
match	Bitmap of (1 << OFPXMT_*) that indicate the fields the table can match on	["OFB_IN_PORT","OFB_METADATA"]
wildcards	Bitmap of (1 << OFPXMT_*) wildcards that are supported by the table	["OFB_IN_PORT","OFB_METADATA"]
write_actions	Bitmap of OFPAT_* that are supported by the table with OFPIT_WRITE_ACTIONS	["OUTPUT","SET_MPLS_TTL"]
apply_actions	Bitmap of OFPAT_* that are supported by the table with OFPIT_APPLY_ACTIONS	["OUTPUT","SET_MPLS_TTL"]
write_setfields	Bitmap of (1 << OFPXMT_*) header fields that can be set with OFPIT_WRITE_ACTIONS	["OFB_IN_PORT","OFB_METADATA"]
apply_setfields	Bitmap of (1 << OFPXMT_*) header fields that can be set with OFPIT_APPLY_ACTIONS	["OFB_IN_PORT","OFB_METADATA"]
metadata_match	Bits of metadata table can match	18446744073709552000
metadata_write	Bits of metadata table can write	18446744073709552000
instructions	Bitmap of OFPIT_* values supported	["GOTO_TABLE","WRITE_METADATA"]
config	Bitmap of OFPTC_* values	[]
max_entries	Max number of entries supported	1e+06
active_count	Number of active entries	0
lookup_count	Number of packets looked up in table	0
matched_count	Number of packets that hit table	8

Response message body(OpenFlow1.3):

Attribute	Description	Example
dpid	Datapath ID	"1"
table_id	Table ID	0
active_count	Number of active entries	0
lookup_count	Number of packets looked up in table	8
matched_count	Number of packets that hit table	0

Example of use:

```
$ curl -X GET http://localhost:8080/stats/table/1
```

Response (OpenFlow1.0):

```
{
  "1": [
    {
      "table_id": 0,
      "lookup_count": 8,
      "max_entries": 1e+06,
      "active_count": 0,
      "name": "classifier",
      "matched_count": 0,
      "wildcards": [
       "IN_PORT",
       "DL_VLAN"
      ]
    },
    ...
    {
      "table_id": 253,
      "lookup_count": 0,
      "max_entries": 1e+06,
      "active_count": 0,
      "name": "table253",
      "matched_count": 0,
      "wildcards": [
       "IN_PORT",
       "DL_VLAN"
      ]
    }
  ]
}
```

Response (OpenFlow1.2):

```
{
  "1": [
    {
      "apply_setfields": [
       "OFB_IN_PORT",
       "OFB_METADATA"
      ],
      "match": [
       "OFB_IN_PORT",
       "OFB_METADATA"
      ],
      "metadata_write": 18446744073709552000,
      "config": [],
      "instructions":[
       "GOTO_TABLE",
       "WRITE_METADATA"
      ],
      "table_id": 0,
      "metadata_match": 18446744073709552000,
      "lookup_count": 8,
      "wildcards": [
       "OFB_IN_PORT",
       "OFB_METADATA"
      ],
      "write_setfields": [
       "OFB_IN_PORT",
       "OFB_METADATA"
      ],
      "write_actions": [
       "OUTPUT",
       "SET_MPLS_TTL"
      ],
      "name": "classifier",
      "matched_count": 0,
      "apply_actions": [
       "OUTPUT",
       "SET_MPLS_TTL"
      ],
      "active_count": 0,
      "max_entries": 1e+06
    },
    ...
    {
      "apply_setfields": [
       "OFB_IN_PORT",
       "OFB_METADATA"
      ],
      "match": [
       "OFB_IN_PORT",
       "OFB_METADATA"
      ],
      "metadata_write": 18446744073709552000,
      "config": [],
      "instructions": [
       "GOTO_TABLE",
       "WRITE_METADATA"
      ],
      "table_id": 253,
      "metadata_match": 18446744073709552000,
      "lookup_count": 0,
      "wildcards": [
       "OFB_IN_PORT",
       "OFB_METADATA"
      ],
      "write_setfields": [
       "OFB_IN_PORT",
       "OFB_METADATA"
      ],
      "write_actions": [
       "OUTPUT",
       "SET_MPLS_TTL"
      ],
      "name": "table253",
      "matched_count": 0,
      "apply_actions": [
       "OUTPUT",
       "SET_MPLS_TTL"
      ],
      "active_count": 0,
      "max_entries": 1e+06
    }
  ]
}
```

Response (OpenFlow1.3):

```
{
  "1": [
    {
      "active_count": 0,
      "table_id": 0,
      "lookup_count": 8,
      "matched_count": 0
    },
    ...
    {
      "active_count": 0,
      "table_id": 253,
      "lookup_count": 0,
      "matched_count": 0
    }
  ]
}
```

Get table features

Get table features of the switch which specified with Datapath ID in URI.

Usage:

Method	GET
URI	/stats/tablefeatures/<dpid>

Response message body:

Attribute	Description	Example
dpid	Datapath ID	"1"
table_id	Table ID	0
name	Name of Table	"table_0"
meta-data_match	Bits of metadata table can match	18446744073709552000
meta-data_write	Bits of metadata table can write	18446744073709552000
config	Bitmap of OFPTC_* values	0
max_entries	Max number of entries supported	4096
properties	struct ofp_table_feature_prop_header	[{"type": "INSTRUCTIONS","instruction_ids": [...]},...]

Example of use:

```
$ curl -X GET http://localhost:8080/stats/tablefeatures/1
```

```
{
  "1": [
    {
      "metadata_write": 18446744073709552000,
      "config": 0,
      "table_id": 0,
      "metadata_match": 18446744073709552000,
      "max_entries": 4096,
      "properties": [
        {
          "type": "INSTRUCTIONS",
          "instruction_ids": [
           {
           "len": 4,
           "type": 1
           },
           ...
          ]
        },
        ...
      ],
      "name": "table_0"
    },
    {
      "metadata_write": 18446744073709552000,
      "config": 0,
      "table_id": 1,
      "metadata_match": 18446744073709552000,
      "max_entries": 4096,
      "properties": [
        {
          "type": "INSTRUCTIONS",
          "instruction_ids": [
           {
           "len": 4,
           "type": 1
           },
           ...
          ]
        },
        ...
      ],
      "name": "table_1"
    },
    ...
  ]
}
```

Get ports stats

Get ports stats of the switch which specified with Datapath ID in URI.

Usage:

Method	GET
URI	/stats/port/<dpid>[/<port>]

Note: Specification of port number is optional.

Response message body(OpenFlow1.3 or earlier):

Attribute	Description	Example
dpid	Datapath ID	"1"
port_no	Port number	1
rx_packets	Number of received packets	9
tx_packets	Number of transmitted packets	6
rx_bytes	Number of received bytes	738
tx_bytes	Number of transmitted bytes	252
rx_dropped	Number of packets dropped by RX	0
tx_dropped	Number of packets dropped by TX	0
rx_errors	Number of receive errors	0
tx_errors	Number of transmit errors	0
rx_frame_err	Number of frame alignment errors	0
rx_over_err	Number of packets with RX overrun	0
rx_crc_err	Number of CRC errors	0
collisions	Number of collisions	0
duration_sec	Time port has been alive in seconds	12
duration_nsec	Time port has been alive in nanoseconds beyond duration_sec	9.76e+08

Response message body(OpenFlow1.4 or later):

Attribute	Description	Example
dpid	Datapath ID	"1"
port_no	Port number	1
rx_packets	Number of received packets	9
tx_packets	Number of transmitted packets	6
rx_bytes	Number of received bytes	738
tx_bytes	Number of transmitted bytes	252
rx_dropped	Number of packets dropped by RX	0
tx_dropped	Number of packets dropped by TX	0
rx_errors	Number of receive errors	0
tx_errors	Number of transmit errors	0
duration_sec	Time port has been alive in seconds	12
duration_nsec	Time port has been alive in nanoseconds beyond duration_sec	9.76e+08
properties	struct ofp_port_desc_prop_header	[{"rx_frame_err": 0, "rx_over_err": 0, "rx_crc_err": 0, "collisions": 0,...},...]

Example of use:

```
$ curl -X GET http://localhost:8080/stats/port/1
```

Response (OpenFlow1.3 or earlier):

```
{
  "1": [
    {
      "port_no": 1,
      "rx_packets": 9,
      "tx_packets": 6,
      "rx_bytes": 738,
      "tx_bytes": 252,
      "rx_dropped": 0,
      "tx_dropped": 0,
      "rx_errors": 0,
      "tx_errors": 0,
      "rx_frame_err": 0,
      "rx_over_err": 0,
      "rx_crc_err": 0,
      "collisions": 0,
      "duration_sec": 12,
      "duration_nsec": 9.76e+08
    },
    {
      :
      :
    }
  ]
}
```

Response (OpenFlow1.4 or later):

```
{
   "1": [
     {
       "port_no": 1,
       "rx_packets": 9,
       "tx_packets": 6,
       "rx_bytes": 738,
       "tx_bytes": 252,
       "rx_dropped": 0,
       "tx_dropped": 0,
       "rx_errors": 0,
       "tx_errors": 0,
       "duration_nsec": 12,
       "duration_sec": 9.76e+08,
       "properties": [
         {
           "rx_frame_err": 0,
           "rx_over_err": 0,
           "rx_crc_err": 0,
           "collisions": 0,
           "type": "ETHERNET"
         },
         {
           "bias_current": 300,
           "flags": 3,
           "rx_freq_lmda": 1500,
           "rx_grid_span": 500,
           "rx_offset": 700,
           "rx_pwr": 2000,
           "temperature": 273,
           "tx_freq_lmda": 1500,
           "tx_grid_span": 500,
           "tx_offset": 700,
           "tx_pwr": 2000,
           "type": "OPTICAL"
         },
         {
           "data": [],
           "exp_type": 0,
           "experimenter": 101,
           "type": "EXPERIMENTER"
         },
         {
           :

           :
         }
       ]
     }
   ]
 }
```

Get ports description

Get ports description of the switch which specified with Datapath ID in URI.

Usage(OpenFlow1.4 or earlier):

Method	GET
URI	/stats/portdesc/<dpid>

Usage(OpenFlow1.5 or later):

Method	GET
URI	/stats/portdesc/<dpid>/[<port>]

Note: Specification of port number is optional.

Response message body(OpenFlow1.3 or earlier):

Attribute	Description	Example
dpid	Datapath ID	"1"
port_no	Port number	1
hw_addr	Ethernet hardware address	"0a:b6:d0:0c:e1:d7"
name	Name of port	"s1-eth1"
config	Bitmap of OFPPC_* flags	0
state	Bitmap of OFPPS_* flags	0
curr	Current features	2112
advertised	Features being advertised by the port	0
supported	Features supported by the port	0
peer	Features advertised by peer	0
curr_speed	Current port bitrate in kbps	1e+07
max_speed	Max port bitrate in kbps	0

Response message body(OpenFlow1.4 or later):

Attribute	Description	Example
dpid	Datapath ID	“1”
port_no	Port number	1
hw_addr	Ethernet hardware address	“0a:b6:d0:0c:e1:d7”
name	Name of port	“s1-eth1”
config	Bitmap of OFPPC_* flags	0
state	Bitmap of OFPPS_* flags	0
length	Length of this entry	168
properties	struct ofp_port_desc_prop_header	[{“length”: 32, “curr”: 10248,...}...]

Example of use:

```
$ curl -X GET http://localhost:8080/stats/portdesc/1
```

Response (OpenFlow1.3 or earlier):

```
{
  "1": [
    {
      "port_no": 1,
      "hw_addr": "0a:b6:d0:0c:e1:d7",
      "name": "s1-eth1",
      "config": 0,
      "state": 0,
      "curr": 2112,
      "advertised": 0,
      "supported": 0,
      "peer": 0,
      "curr_speed": 1e+07,
      "max_speed": 0
    },
    {
      :
      :
    }
  ]
}
```

Response (OpenFlow1.4 or later):

```
{
   "1": [
     {
       "port_no": 1,
       "hw_addr": "0a:b6:d0:0c:e1:d7",
       "name": "s1-eth1",
       "config": 0,
       "state": 0,
       "length": 168,
       "properties": [
         {
           "length": 32,
           "curr": 10248,
           "advertised": 10240,
           "supported": 10248,
           "peer": 10248,
           "curr_speed": 5000,
           "max_speed": 5000,
           "type": "ETHERNET"
         },
         {
           "length": 40,
           "rx_grid_freq_lmda": 1500,
           "tx_grid_freq_lmda": 1500,
           "rx_max_freq_lmda": 2000,
           "tx_max_freq_lmda": 2000,
           "rx_min_freq_lmda": 1000,
           "tx_min_freq_lmda": 1000,
           "tx_pwr_max": 2000,
           "tx_pwr_min": 1000,
           "supported": 1,
           "type": "OPTICAL"
         },
         {
           "data": [],
           "exp_type": 0,
           "experimenter": 101,
           "length": 12,
           "type": "EXPERIMENTER"
         },
         {
           :

           :
         }
       ]
     }
   ]
}
```

Get queues stats

Get queues stats of the switch which specified with Datapath ID in URI.

Usage:

Method	GET
URI	/stats/queue/<dpid>[/<port>[/<queue_id>]]

Note: Specification of port number and queue id are optional.

If you want to omitting the port number and setting the queue id, please specify the keyword "ALL" to the port number.

e.g. GET http://localhost:8080/stats/queue/1/ALL/1

Response message body(OpenFlow1.3 or earlier):

Attribute	Description	Example
dpid	Datapath ID	“1”
port_no	Port number	1
queue_id	Queue ID	0
tx_bytes	Number of transmitted bytes	0
tx_packets	Number of transmitted packets	0
tx_errors	Number of packets dropped due to overrun	0
duration_sec	Time queue has been alive in seconds	4294963425
duration_nsec	Time queue has been alive in nanoseconds beyond duration_sec	3912967296

Response message body(OpenFlow1.4 or later):

Attribute	Description	Example
dpid	Datapath ID	“1”
port_no	Port number	1
queue_id	Queue ID	0
tx_bytes	Number of transmitted bytes	0
tx_packets	Number of transmitted packets	0
tx_errors	Number of packets dropped due to overrun	0
duration_sec	Time queue has been alive in seconds	4294963425
duration_nsec	Time queue has been alive in nanoseconds beyond duration_sec	3912967296
length	Length of this entry	104
properties	struct ofp_queue_stats_prop_header	[{“type”: 65535,”length”: 12,...},...]

Example of use:

```
$ curl -X GET http://localhost:8080/stats/queue/1
```

Response (OpenFlow1.3 or earlier):

```
{
  "1": [
    {
      "port_no": 1,
      "queue_id": 0,
      "tx_bytes": 0,
      "tx_packets": 0,
      "tx_errors": 0,
      "duration_sec": 4294963425,
      "duration_nsec": 3912967296
    },
    {
      "port_no": 1,
      "queue_id": 1,
      "tx_bytes": 0,
      "tx_packets": 0,
      "tx_errors": 0,
      "duration_sec": 4294963425,
      "duration_nsec": 3912967296
    }
  ]
}
```

Response (OpenFlow1.4 or later):

```
{
  "1": [
    {
      "port_no": 1,
      "queue_id": 0,
      "tx_bytes": 0,
      "tx_packets": 0,
      "tx_errors": 0,
      "duration_sec": 4294963425,
      "duration_nsec": 3912967296,
      "length": 104,
      "properties": [
         {
            "OFPQueueStatsPropExperimenter": {
               "type": 65535,
               "length": 16,
               "data": [
                  1
               ],
               "exp_type": 1,
               "experimenter": 101
            }
         },
         {
            :

            :
         }
      ]
    },
    {
      "port_no": 2,
      "queue_id": 1,
      "tx_bytes": 0,
      "tx_packets": 0,
      "tx_errors": 0,
      "duration_sec": 4294963425,
      "duration_nsec": 3912967296,
      "length": 48,
      "properties": []
    }
  ]
}
```

Get queues config

Get queues config of the switch which specified with Datapath ID and Port in URI.

Usage:

Method	GET
URI	/stats/queueconfig/<dpid>/[<port>]

Note: Specification of port number is optional.

Caution: This message is deprecated in Openflow1.4. If OpenFlow 1.4 or later is in use, please refer to *Get queues description* instead.

Response message body:

Attribute	Description	Example
dpid	Datapath ID	“1”
port	Port which was queried	1
queues	struct ofp_packet_queue	
– queue_id	ID for the specific queue	2
– port	Port this queue is attached to	0
– properties	struct ofp_queue_prop_header properties	[{“property”: “MIN_RATE”,”rate”: 80}]

Example of use:

```
$ curl -X GET http://localhost:8080/stats/queueconfig/1/1
```

```
{
  "1": [
    {
      "port": 1,
      "queues": [
        {
          "properties": [
            {
              "property": "MIN_RATE",
              "rate": 80
            }
          ],
          "port": 0,
          "queue_id": 1
        },
        {
          "properties": [
            {
              "property": "MAX_RATE",
              "rate": 120
            }
          ],
          "port": 2,
          "queue_id": 2
        },
        {
          "properties": [
            {
              "property": "EXPERIMENTER",
              "data": [],
              "experimenter": 999
            }
          ],
          "port": 3,
          "queue_id": 3
        }
      ]
    }
  ]
}
```

Get queues description

Get queues description of the switch which specified with Datapath ID, Port and Queue_id in URI.

Usage:

Method	GET
URI	/stats/queuedesc/<dpid>[/<port>/[<queue_id>]]

Note: Specification of port number and queue id are optional.

If you want to omitting the port number and setting the queue id, please specify the keyword "ALL" to the port number.

e.g. GET http://localhost:8080/stats/queuedesc/1/ALL/1

Caution: This message is available in OpenFlow1.4 or later. If Openflow1.3 or earlier is in use, please refer to *Get queues config* instead.

Response message body:

Attribute	Description	Example
dpid	Datapath ID	"1"
len	Length in bytes of this queue desc	88
port_no	Port which was queried	1
queue_id	Queue ID	1
properties	struct ofp_queue_desc_prop_header	[{"length": 8, ...},...]

Example of use:

```
$ curl -X GET http://localhost:8080/stats/queuedesc/1/1/1
```

```
{
 "1": [
     {
       "len": 88,
       "port_no": 1,
       "queue_id": 1,
       "properties": [
         {
           "length": 8,
           "rate": 300,
           "type": "MIN_RATE"
         },
         {
           "length": 8,
           "rate": 900,
           "type": "MAX_RATE"
         },
         {
           "length": 16,
           "exp_type": 0,
           "experimenter": 101,
           "data": [1],
           "type": "EXPERIMENTER"
         },
         {
           :

           :
         }
       ]
     }
   ]
 }
```

Get groups stats

Get groups stats of the switch which specified with Datapath ID in URI.

Usage:

Method	GET
URI	/stats/group/<dpid>[/<group_id>]

Note: Specification of group id is optional.

Response message body:

Attribute	Description	Example
dpid	Datapath ID	"1"
length	Length of this entry	56
group_id	Group ID	1
ref_count	Number of flows or groups that directly forward to this group	1
packet_count	Number of packets processed by group	0
byte_count	Number of bytes processed by group	0
duration_sec	Time group has been alive in seconds	161
duration_nsec	Time group has been alive in nanoseconds beyond duration_sec	3.03e+08
bucket_stats	struct ofp_bucket_counter	
– packet_count	Number of packets processed by bucket	0
– byte_count	Number of bytes processed by bucket	0

Example of use:

```
$ curl -X GET http://localhost:8080/stats/group/1
```

```
{
  "1": [
    {
      "length": 56,
      "group_id": 1,
      "ref_count": 1,
      "packet_count": 0,
      "byte_count": 0,
      "duration_sec": 161,
      "duration_nsec": 3.03e+08,
      "bucket_stats": [
        {
          "packet_count": 0,
          "byte_count": 0
        }
      ]
    }
  ]
}
```

Get group description stats

Get group description stats of the switch which specified with Datapath ID in URI.

Usage(Openflow1.4 or earlier):

Method	GET
URI	/stats/groupdesc/<dpid>

Usage(Openflow1.5 or later):

Method	GET
URI	/stats/groupdesc/<dpid>/[<group_id>]

Note: Specification of group id is optional.

Response message body(Openflow1.3 or earlier):

Attribute	Description	Example
dpid	Datapath ID	“1”
type	One of OFPGT_*	“ALL”
group_id	Group ID	1
buckets	struct ofp_bucket	
– weight	Relative weight of bucket (Only defined for select groups)	0
– watch_port	Port whose state affects whether this bucket is live (Only required for fast failover groups)	4294967295
– watch_group	Group whose state affects whether this bucket is live (Only required for fast failover groups)	4294967295
– actions	0 or more actions associated with the bucket	[”OUTPUT:1”]

Response message body(Openflow1.4 or later):

Attribute	Description	Example
dpid	Datapath ID	"1"
type	One of OFPGT_*	"ALL"
group_id	Group ID	1
length	Length of this entry	40
buckets	struct ofp_bucket	
– weight	Relative weight of bucket (Only defined for select groups)	0
– watch_port	Port whose state affects whether this bucket is live (Only required for fast failover groups)	4294967295
– watch_group	Group whose state affects whether this bucket is live (Only required for fast failover groups)	4294967295
– len	Length the bucket in bytes, including this header and any adding to make it 64-bit aligned.	32
– actions	0 or more actions associated with the bucket	[{"OUTPUT:1", "max_len": 65535,...}]

Example of use:

```
$ curl -X GET http://localhost:8080/stats/groupdesc/1
```

Response (Openflow1.3 or earlier):

```
{
  "1": [
    {
      "type": "ALL",
      "group_id": 1,
      "buckets": [
        {
          "weight": 0,
          "watch_port": 4294967295,
          "watch_group": 4294967295,
          "actions": [
            "OUTPUT:1"
          ]
        }
      ]
    }
  ]
}
```

Response (Openflow1.4 or later):

```
{
   "1": [
     {
       "type": "ALL",
       "group_id": 1,
       "length": 40,
       "buckets": [
         {
           "weight": 1,
           "watch_port": 1,
           "watch_group": 1,
           "len": 32,
           "actions": [
             {
                 "type": "OUTPUT",
                 "max_len": 65535,
                 "port": 2
             }
           ]
         }
       ]
     }
   ]
}
```

Get group features stats

Get group features stats of the switch which specified with Datapath ID in URI.

Usage:

Method	GET
URI	/stats/groupfeatures/<dpid>

Response message body:

Attribute	Description	Example
dpid	Datapath ID	"1"
types	Bitmap of (1 << OFPGT_*) values supported	[]
capabilities	Bitmap of OFPGFC_* capability supported	["SELECT_WEIGHT","SELECT_LIVENESS","CHAINING"]
max_groups	Maximum number of groups for each type	[{"ALL": 4294967040},...]
actions	Bitmaps of (1 << OFPAT_*) values supported	[{"ALL": ["OUTPUT",...]},...]

Example of use:

```
$ curl -X GET http://localhost:8080/stats/groupfeatures/1
```

```
{
  "1": [
    {
      "types": [],
      "capabilities": [
        "SELECT_WEIGHT",
        "SELECT_LIVENESS",
        "CHAINING"
      ],
      "max_groups": [
        {
          "ALL": 4294967040
        },
        {
          "SELECT": 4294967040
        },
        {
          "INDIRECT": 4294967040
        },
        {
          "FF": 4294967040
        }
      ],
      "actions": [
        {
          "ALL": [
            "OUTPUT",
            "COPY_TTL_OUT",
            "COPY_TTL_IN",
            "SET_MPLS_TTL",
            "DEC_MPLS_TTL",
            "PUSH_VLAN",
            "POP_VLAN",
            "PUSH_MPLS",
            "POP_MPLS",
            "SET_QUEUE",
            "GROUP",
            "SET_NW_TTL",
            "DEC_NW_TTL",
            "SET_FIELD"
          ]
        },
        {
          "SELECT": []
        },
        {
          "INDIRECT": []
        },
        {
          "FF": []
        }
      ]
    }
  ]
}
```

Get meters stats

Get meters stats of the switch which specified with Datapath ID in URI.

Usage:

Method	GET
URI	/stats/meter/<dpid>[/<meter_id>]

Note: Specification of meter id is optional.

Response message body:

Attribute	Description	Example
dpid	Datapath ID	"1"
meter_id	Meter ID	1
len	Length in bytes of this stats	56
flow_count	Number of flows bound to meter	0
packet_in_count	Number of packets in input	0
byte_in_count	Number of bytes in input	0
duration_sec	Time meter has been alive in seconds	37
duration_nsec	Time meter has been alive in nanoseconds beyond duration_sec	988000
band_stats	struct ofp_meter_band_stats	
– packet_band_count	Number of packets in band	0
– byte_band_count	Number of bytes in band	0

Example of use:

```
$ curl -X GET http://localhost:8080/stats/meter/1
```

```
{
  "1": [
    {
      "meter_id": 1,
      "len": 56,
      "flow_count": 0,
      "packet_in_count": 0,
      "byte_in_count": 0,
      "duration_sec": 37,
      "duration_nsec": 988000,
      "band_stats": [
        {
          "packet_band_count": 0,
          "byte_band_count": 0
        }
      ]
    }
  ]
}
```

Get meter config stats

Get meter description stats

Get meter config stats of the switch which specified with Datapath ID in URI.

Caution: This message has been renamed in openflow 1.5. If Openflow 1.4 or earlier is in use, please used as Get meter description stats. If Openflow 1.5 or later is in use, please used as Get meter description stats.

Usage(Openflow1.4 or earlier):

Method	GET
URI	/stats/meterconfig/<dpid>[/<meter_id>]

Usage(Openflow1.5 or later):

Method	GET
URI	/stats/meterdesc/<dpid>[/<meter_id>]

Note: Specification of meter id is optional.

Response message body:

Attribute	Description	Example
dpid	Datapath ID	"1"
flags	All OFPMC_* that apply	"KBPS"
meter_id	Meter ID	1
bands	struct ofp_meter_band_header	
– type	One of OFPMBT_*	"DROP"
– rate	Rate for this band	1000
– burst_size	Size of bursts	0

Example of use:

```
$ curl -X GET http://localhost:8080/stats/meterconfig/1
```

```
{
  "1": [
    {
      "flags": [
        "KBPS"
      ],
      "meter_id": 1,
      "bands": [
        {
          "type": "DROP",
          "rate": 1000,
          "burst_size": 0
        }
      ]
    }
  ]
}
```

Get meter features stats

Get meter features stats of the switch which specified with Datapath ID in URI.

Usage:

Method	GET
URI	/stats/meterfeatures/<dpid>

Response message body:

Attribute	Description	Example
dpid	Datapath ID	“1”
max_meter	Maximum number of meters	256
band_types	Bitmaps of (1 << OFPMBT_*) values supported	[”DROP”]
capabilities	Bitmaps of “ofp_meter_flags”	[”KBPS”, “BURST”, “STATS”]
max_bands	Maximum bands per meters	16
max_color	Maximum color value	8

Example of use:

```
$ curl -X GET http://localhost:8080/stats/meterfeatures/1
```

```
{
  "1": [
    {
      "max_meter": 256,
      "band_types": [
        "DROP"
      ],
      "capabilities": [
        "KBPS",
        "BURST",
        "STATS"
      ],
      "max_bands": 16,
      "max_color": 8
    }
  ]
}
```

Get role

Get the current role of the controller from the switch.

Usage:

Method	GET
URI	/stats/role/<dpid>

Response message body(Openflow1.4 or earlier):

Attribute	Description	Example
dpid	Datapath ID	1
role	One of OFPCR_ROLE_*	“EQUAL”
generation_id	Master Election Generation Id	0

Response message body(Openflow1.5 or later):

Attribute	Description	Example
dpid	Datapath ID	1
role	One of OFPCR_ROLE_*	“EQUAL”
short_id	ID number for the controller	0
generation_id	Master Election Generation Id	0

Example of use:

```
$ curl -X GET http://localhost:8080/stats/role/1
```

Response (Openflow1.4 or earlier):

```
{
    "1": [
        {
            "generation_id": 0,
            "role": "EQUAL"
        }
    ]
}
```

Response (Openflow1.5 or later):

```
{
    "1": [
        {
            "generation_id": 0,
            "role": "EQUAL",
            "short_id": 0
        }
    ]
}
```

6.2.2 Update the switch stats

Add a flow entry

Add a flow entry to the switch.

Usage:

Method	POST
URI	/stats/flowentry/add

Request message body(Openflow1.3 or earlier):

Attribute	Description	Example	Default
dpid	Datapath ID (int)	1	(Mandatory)
cookie	Opaque controller-issued identifier (int)	1	0
cookie_mask	Mask used to restrict the cookie bits (int)	1	0
table_id	Table ID to put the flow in (int)	0	0
idle_timeout	Idle time before discarding (seconds) (int)	30	0
hard_timeout	Max time before discarding (seconds) (int)	30	0
priority	Priority level of flow entry (int)	11111	0
buffer_id	Buffered packet to apply to, or OFP_NO_BUFFER (int)	1	OFP_NO_BUFFER
flags	Bitmap of OFPFF_* flags (int)	1	0
match	Fields to match (dict)	{“in_port”:1}	{} #wildcarded
actions	Instruction set (list of dict)	[{“type”:”OUTPUT”, “port”:2}]	[] #DROP

Request message body(Openflow1.4 or later):

Attribute	Description	Example	Default
dpid	Datapath ID (int)	1	(Mandatory)
cookie	Opaque controller-issued identifier (int)	1	0
cookie_mask	Mask used to restrict the cookie bits (int)	1	0
table_id	Table ID to put the flow in (int)	0	0
idle_timeout	Idle time before discarding (seconds) (int)	30	0
hard_timeout	Max time before discarding (seconds) (int)	30	0
priority	Priority level of flow entry (int)	11111	0
buffer_id	Buffered packet to apply to, or OFP_NO_BUFFER (int)	1	OFP_NO_BUFFER
flags	Bitmap of OFPFF_* flags (int)	1	0
match	Fields to match (dict)	{“in_port”:1}	{} #wildcarded
instructions	Instruction set (list of dict)	[{“type”:”METER”, “meter_id”:2}]	[] #DROP

Note: For description of match and actions, please see *Reference: Description of Match and Actions*.

Example of use(Openflow1.3 or earlier):

```
$ curl -X POST -d '{
    "dpid": 1,
    "cookie": 1,
    "cookie_mask": 1,
    "table_id": 0,
    "idle_timeout": 30,
    "hard_timeout": 30,
    "priority": 11111,
    "flags": 1,
    "match":{
        "in_port":1
    },
    "actions":[
        {
            "type":"OUTPUT",
            "port": 2
        }
    ]
 }' http://localhost:8080/stats/flowentry/add
```

```
$ curl -X POST -d '{
    "dpid": 1,
    "priority": 22222,
    "match":{
        "in_port":1
    },
    "actions":[
        {
            "type":"GOTO_TABLE",
            "table_id": 1
        }
    ]
 }' http://localhost:8080/stats/flowentry/add
```

```
$ curl -X POST -d '{
    "dpid": 1,
    "priority": 33333,
    "match":{
        "in_port":1
    },
    "actions":[
        {
            "type":"WRITE_METADATA",
            "metadata": 1,
            "metadata_mask": 1
        }
    ]
 }' http://localhost:8080/stats/flowentry/add
```

```
$ curl -X POST -d '{
    "dpid": 1,
    "priority": 44444,
    "match":{
        "in_port":1
    },
    "actions":[
        {
            "type":"METER",
            "meter_id": 1
        }
    ]
 }' http://localhost:8080/stats/flowentry/add
```

Example of use(Openflow1.4 or later):

```
$ curl -X POST -d '{
    "dpid": 1,
    "cookie": 1,
    "cookie_mask": 1,
    "table_id": 0,
    "idle_timeout": 30,
    "hard_timeout": 30,
    "priority": 11111,
    "flags": 1,
    "match":{
        "in_port":1
    },
    "instructions": [
        {
            "type": "APPLY_ACTIONS",
            "actions": [
                {
                    "max_len": 65535,
                    "port": 2,
                    "type": "OUTPUT"
                }
            ]
        }
    ]
 }' http://localhost:8080/stats/flowentry/add
```

```
$ curl -X POST -d '{
    "dpid": 1,
    "priority": 22222,
    "match":{
        "in_port":1
    },
    "instructions": [
        {
            "type":"GOTO_TABLE",
            "table_id": 1
        }
    ]
 }' http://localhost:8080/stats/flowentry/add
```

```
$ curl -X POST -d '{
    "dpid": 1,
    "priority": 33333,
    "match":{
        "in_port":1
    },
    "instructions": [
        {
            "type":"WRITE_METADATA",
            "metadata": 1,
            "metadata_mask": 1
        }
    ]
 }' http://localhost:8080/stats/flowentry/add
```

```
$ curl -X POST -d '{
    "dpid": 1,
    "priority": 44444,
    "match":{
        "in_port":1
    },
    "instructions": [
        {
            "type":"METER",
            "meter_id": 1
        }
    ]
 }' http://localhost:8080/stats/flowentry/add
```

Note: To confirm flow entry registration, please see *Get all flows stats* or *Get flows stats filtered by fields*.

Modify all matching flow entries

Modify all matching flow entries of the switch.

Usage:

Method	POST
URI	/stats/flowentry/modify

Request message body:

Attribute	Description	Example	Default
dpid	Datapath ID (int)	1	(Mandatory)
cookie	Opaque controller-issued identifier (int)	1	0
cookie_mask	Mask used to restrict the cookie bits (int)	1	0
table_id	Table ID to put the flow in (int)	0	0
idle_timeout	Idle time before discarding (seconds) (int)	30	0
hard_timeout	Max time before discarding (seconds) (int)	30	0
priority	Priority level of flow entry (int)	11111	0
buffer_id	Buffered packet to apply to, or OFP_NO_BUFFER (int)	1	OFP_NO_BUFFER
flags	Bitmap of OFPFF_* flags (int)	1	0
match	Fields to match (dict)	{"in_port":1}	{} #wildcarded
actions	Instruction set (list of dict)	[{"type":"OUTPUT", "port":2}]	[] #DROP

Example of use:

```
$ curl -X POST -d '{
    "dpid": 1,
    "cookie": 1,
    "cookie_mask": 1,
    "table_id": 0,
    "idle_timeout": 30,
    "hard_timeout": 30,
    "priority": 11111,
    "flags": 1,
    "match":{
        "in_port":1
    },
    "actions":[
        {
            "type":"OUTPUT",
            "port": 2
        }
    ]
 }' http://localhost:8080/stats/flowentry/modify
```

Modify flow entry strictly

Modify flow entry strictly matching wildcards and priority

Usage:

Method	POST
URI	/stats/flowentry/modify_strict

Request message body:

Attribute	Description	Example	Default
dpid	Datapath ID (int)	1	(Mandatory)
cookie	Opaque controller-issued identifier (int)	1	0
cookie_mask	Mask used to restrict the cookie bits (int)	1	0
table_id	Table ID to put the flow in (int)	0	0
idle_timeout	Idle time before discarding (seconds) (int)	30	0
hard_timeout	Max time before discarding (seconds) (int)	30	0
priority	Priority level of flow entry (int)	11111	0
buffer_id	Buffered packet to apply to, or OFP_NO_BUFFER (int)	1	OFP_NO_BUFFER
flags	Bitmap of OFPFF_* flags (int)	1	0
match	Fields to match (dict)	{“in_port”:1}	{} #wildcarded
actions	Instruction set (list of dict)	[{“type”:”OUTPUT”, “port”:2}]	[] #DROP

Example of use:

```
$ curl -X POST -d '{
    "dpid": 1,
    "cookie": 1,
    "cookie_mask": 1,
    "table_id": 0,
    "idle_timeout": 30,
    "hard_timeout": 30,
    "priority": 11111,
    "flags": 1,
    "match":{
        "in_port":1
    },
    "actions":[
        {
            "type":"OUTPUT",
            "port": 2
        }
    ]
 }' http://localhost:8080/stats/flowentry/modify_strict
```

Delete all matching flow entries

Delete all matching flow entries of the switch.

Usage:

Method	POST
URI	/stats/flowentry/delete

Request message body:

Attribute	Description	Example	Default
dpid	Datapath ID (int)	1	(Mandatory)
cookie	Opaque controller-issued identifier (int)	1	0
cookie_mask	Mask used to restrict the cookie bits (int)	1	0
table_id	Table ID to put the flow in (int)	0	0
idle_timeout	Idle time before discarding (seconds) (int)	30	0
hard_timeout	Max time before discarding (seconds) (int)	30	0
priority	Priority level of flow entry (int)	11111	0
buffer_id	Buffered packet to apply to, or OFP_NO_BUFFER (int)	1	OFP_NO_BUFFER
out_port	Output port (int)	1	OFPP_ANY
out_group	Output group (int)	1	OFPG_ANY
flags	Bitmap of OFPFF_* flags (int)	1	0
match	Fields to match (dict)	{“in_port”:1}	{} #wildcarded
actions	Instruction set (list of dict)	[{“type”:”OUTPUT”, “port”:2}]	[] #DROP

Example of use:

```
$ curl -X POST -d '{
    "dpid": 1,
    "cookie": 1,
    "cookie_mask": 1,
    "table_id": 0,
    "idle_timeout": 30,
    "hard_timeout": 30,
    "priority": 11111,
    "flags": 1,
    "match":{
        "in_port":1
    },
    "actions":[
        {
            "type":"OUTPUT",
            "port": 2
        }
    ]
}' http://localhost:8080/stats/flowentry/delete
```

Delete flow entry strictly

Delete flow entry strictly matching wildcards and priority.

Usage:

Method	POST
URI	/stats/flowentry/delete_strict

Request message body:

Attribute	Description	Example	Default
dpid	Datapath ID (int)	1	(Mandatory)
cookie	Opaque controller-issued identifier (int)	1	0
cookie_mask	Mask used to restrict the cookie bits (int)	1	0
table_id	Table ID to put the flow in (int)	0	0
idle_timeout	Idle time before discarding (seconds) (int)	30	0
hard_timeout	Max time before discarding (seconds) (int)	30	0
priority	Priority level of flow entry (int)	11111	0
buffer_id	Buffered packet to apply to, or OFP_NO_BUFFER (int)	1	OFP_NO_BUFFER
out_port	Output port (int)	1	OFPP_ANY
out_group	Output group (int)	1	OFPG_ANY
flags	Bitmap of OFPFF_* flags (int)	1	0
match	Fields to match (dict)	{“in_port”:1}	{} #wildcarded
actions	Instruction set (list of dict)	[{“type”:”OUTPUT”, “port”:2}]	[] #DROP

Example of use:

```
$ curl -X POST -d '{
    "dpid": 1,
    "cookie": 1,
    "cookie_mask": 1,
    "table_id": 0,
    "idle_timeout": 30,
    "hard_timeout": 30,
    "priority": 11111,
    "flags": 1,
    "match":{
        "in_port":1
    },
    "actions":[
        {
            "type":"OUTPUT",
            "port": 2
        }
    ]
 }' http://localhost:8080/stats/flowentry/delete_strict
```

Delete all flow entries

Delete all flow entries of the switch which specified with Datapath ID in URI.

Usage:

Method	DELETE
URI	/stats/flowentry/clear/<dpid>

Example of use:

```
$ curl -X DELETE http://localhost:8080/stats/flowentry/clear/1
```

Add a group entry

Add a group entry to the switch.

Usage:

Method	POST
URI	/stats/groupentry/add

Request message body:

Attribute	Description	Example	Default
dpid	Datapath ID (int)	1	(Mandatory)
type	One of OFPGT_* (string)	"ALL"	"ALL"
group_id	Group ID (int)	1	0
buckets	struct ofp_bucket		
– weight	Relative weight of bucket (Only defined for select groups)	0	0
– watch_port	Port whose state affects whether this bucket is live (Only required for fast failover groups)	4294967295	OFPP_ANY
– watch_group	Group whose state affects whether this bucket is live (Only required for fast failover groups)	4294967295	OFPG_ANY
– actions	0 or more actions associated with the bucket (list of dict)	[{"type": "OUTPUT", "port": 1}]	[] #DROP

Example of use:

```
$ curl -X POST -d '{
    "dpid": 1,
    "type": "ALL",
    "group_id": 1,
    "buckets": [
        {
            "actions": [
                {
                    "type": "OUTPUT",
                    "port": 1
                }
            ]
        }
    ]
 }' http://localhost:8080/stats/groupentry/add
```

Note: To confirm group entry registration, please see *Get group description stats*.

Modify a group entry

Modify a group entry to the switch.

Usage:

Method	POST
URI	/stats/groupentry/modify

Request message body:

Attribute	Description	Example	Default
dpid	Datapath ID (int)	1	(Mandatory)
type	One of OFPGT_* (string)	“ALL”	“ALL”
group_id	Group ID (int)	1	0
buckets	struct ofp_bucket		
– weight	Relative weight of bucket (Only defined for select groups)	0	0
– watch_port	Port whose state affects whether this bucket is live (Only required for fast failover groups)	4294967295	OFPP_ANY
– watch_group	Group whose state affects whether this bucket is live (Only required for fast failover groups)	4294967295	OFPG_ANY
– actions	0 or more actions associated with the bucket (list of dict)	[{“type”: “OUTPUT”, “port”: 1}]	[] #DROP

Example of use:

```
$ curl -X POST -d '{
    "dpid": 1,
    "type": "ALL",
    "group_id": 1,
    "buckets": [
        {
            "actions": [
                {
                    "type": "OUTPUT",
                    "port": 1
                }
            ]
        }
    ]
}' http://localhost:8080/stats/groupentry/modify
```

Delete a group entry

Delete a group entry to the switch.

Usage:

Method	POST
URI	/stats/groupentry/delete

Request message body:

Attribute	Description	Example	Default
dpid	Datapath ID (int)	1	(Mandatory)
group_id	Group ID (int)	1	0

Example of use:

```
$ curl -X POST -d '{
    "dpid": 1,
    "group_id": 1
 }' http://localhost:8080/stats/groupentry/delete
```

Modify the behavior of the port

Modify the behavior of the physical port.

Usage:

Method	POST
URI	/stats/portdesc/modify

Request message body:

Attribute	Description	Example	Default
dpid	Datapath ID (int)	1	(Mandatory)
port_no	Port number (int)	1	0
config	Bitmap of OFPPC_* flags (int)	1	0
mask	Bitmap of OFPPC_* flags to be changed (int)	1	0

Example of use:

```
$ curl -X POST -d '{
    "dpid": 1,
    "port_no": 1,
    "config": 1,
    "mask": 1
    }' http://localhost:8080/stats/portdesc/modify
```

Note: To confirm port description, please see *Get ports description*.

Add a meter entry

Add a meter entry to the switch.

Usage:

Method	POST
URI	/stats/meterentry/add

Request message body:

Attribute	Description	Example	Default
dpid	Datapath ID (int)	1	(Mandatory)
flags	Bitmap of OFPMF_* flags (list)	["KBPS"]	[] #Empty
meter_id	Meter ID (int)	1	0
bands	struct ofp_meter_band_header		
– type	One of OFPMBT_* (string)	"DROP"	None
– rate	Rate for this band (int)	1000	None
– burst_size	Size of bursts (int)	100	None

Example of use:

```
$ curl -X POST -d '{
    "dpid": 1,
    "flags": "KBPS",
    "meter_id": 1,
    "bands": [
        {
            "type": "DROP",
            "rate": 1000
        }
    ]
 }' http://localhost:8080/stats/meterentry/add
```

Note: To confirm meter entry registration, please see *Get meter config stats*.

Modify a meter entry

Modify a meter entry to the switch.

Usage:

Method	POST
URI	/stats/meterentry/modify

Request message body:

Attribute	Description	Example	Default
dpid	Datapath ID (int)	1	(Mandatory)
flags	Bitmap of OFPMF_* flags (list)	[”KBPS”]	[] #Empty
meter_id	Meter ID (int)	1	0
bands	struct ofp_meter_band_header		
– type	One of OFPMBT_* (string)	“DROP”	None
– rate	Rate for this band (int)	1000	None
– burst_size	Size of bursts (int)	100	None

Example of use:

```
$ curl -X POST -d '{
    "dpid": 1,
    "meter_id": 1,
    "flags": "KBPS",
    "bands": [
        {
            "type": "DROP",
            "rate": 1000
        }
    ]
 }' http://localhost:8080/stats/meterentry/modify
```

Delete a meter entry

Delete a meter entry to the switch.

Usage:

Method	POST
URI	/stats/meterentry/delete

Request message body:

Attribute	Description	Example	Default
dpid	Datapath ID (int)	1	(Mandatory)
meter_id	Meter ID (int)	1	0

Example of use:

```
$ curl -X POST -d '{
    "dpid": 1,
    "meter_id": 1
 }' http://localhost:8080/stats/meterentry/delete
```

Modify role

modify the role of the switch.

Usage:

Method	POST
URI	/stats/role

Request message body:

Attribute	Description	Example	Default
dpid	Datapath ID (int)	1	(Mandatory)
role	One of OFPCR_ROLE_*(string)	“MASTER”	OFPCR_ROLE_EQUAL

Example of use:

```
$ curl -X POST -d '{
    "dpid": 1,
    "role": "MASTER"
 }' http://localhost:8080/stats/role
```

6.2.3 Support for experimenter multipart

Send a experimenter message

Send a experimenter message to the switch which specified with Datapath ID in URI.

Usage:

Method	POST
URI	/stats/experimenter/<dpid>

Request message body:

Attribute	Description	Example	Default
dpid	Datapath ID (int)	1	(Mandatory)
experimenter	Experimenter ID (int)	1	0
exp_type	Experimenter defined (int)	1	0
data_type	Data format type (“ascii” or “base64”)	“ascii”	“ascii”
data	Data to send (string)	“data”	“” #Empty

Example of use:

```
$ curl -X POST -d '{
    "dpid": 1,
    "experimenter": 1,
    "exp_type": 1,
    "data_type": "ascii",
    "data": "data"
    }' http://localhost:8080/stats/experimenter/1
```

6.2.4 Reference: Description of Match and Actions

Description of Match on request messages

List of Match fields (OpenFlow1.0):

Match field	Description	Example
in_port	Input switch port (int)	{"in_port": 7}
dl_src	Ethernet source address (string)	{"dl_src": "aa:bb:cc:11:22:33"}
dl_dst	Ethernet destination address (string)	{"dl_dst": "aa:bb:cc:11:22:33"}
dl_vlan	Input VLAN id (int)	{"dl_vlan": 5}
dl_vlan_pcp	Input VLAN priority (int)	{"dl_vlan_pcp": 3, "dl_vlan": 3}
dl_type	Ethernet frame type (int)	{"dl_type": 123}
nw_tos	IP ToS (int)	{"nw_tos": 16, "dl_type": 2048}
nw_proto	IP protocol or lower 8 bits of ARP opcode (int)	{"nw_proto": 5, "dl_type": 2048}
nw_src	IPv4 source address (string)	{"nw_src": "192.168.0.1", "dl_type": 2048}
nw_dst	IPv4 destination address (string)	{"nw_dst": "192.168.0.1/24", "dl_type": 2048}
tp_src	TCP/UDP source port (int)	{"tp_src": 1, "nw_proto": 6, "dl_type": 2048}
tp_dst	TCP/UDP destination port (int)	{"tp_dst": 2, "nw_proto": 6, "dl_type": 2048}

Note: IPv4 address field can be described as IP Prefix like as follows.

IPv4 address:

```
"192.168.0.1"
"192.168.0.2/24"
```

List of Match fields (OpenFlow1.2 or later):

Match field	Description	Example
in_port	Switch input port (int)	{"in_port": 7}
in_phy_port	Switch physical input port (int)	{"in_phy_port": 5, "in_port": 3}
metadata	Metadata passed between tables (int or string)	{"metadata": 12345} or {"metadata": "0x121
eth_dst	Ethernet destination address (string)	{"eth_dst": "aa:bb:cc:11:22:33/00:00:00:00:f
eth_src	Ethernet source address (string)	{"eth_src": "aa:bb:cc:11:22:33"}
eth_type	Ethernet frame type (int)	{"eth_type": 2048}
vlan_vid	VLAN id (int or string)	See *Example of VLAN ID match field*
vlan_pcp	VLAN priority (int)	{"vlan_pcp": 3, "vlan_vid": 3}
ip_dscp	IP DSCP (6 bits in ToS field) (int)	{"ip_dscp": 3, "eth_type": 2048}
ip_ecn	IP ECN (2 bits in ToS field) (int)	{"ip_ecn": 0, "eth_type": 34525}
ip_proto	IP protocol (int)	{"ip_proto": 5, "eth_type": 34525}
ipv4_src	IPv4 source address (string)	{"ipv4_src": "192.168.0.1", "eth_type": 2048
ipv4_dst	IPv4 destination address (string)	{"ipv4_dst": "192.168.10.10/255.255.255.0",
tcp_src	TCP source port (int)	{"tcp_src": 3, "ip_proto": 6, "eth_type": 204
tcp_dst	TCP destination port (int)	{"tcp_dst": 5, "ip_proto": 6, "eth_type": 204
udp_src	UDP source port (int)	{"udp_src": 2, "ip_proto": 17, "eth_type": 20
udp_dst	UDP destination port (int)	{"udp_dst": 6, "ip_proto": 17, "eth_type": 20
sctp_src	SCTP source port (int)	{"sctp_src": 99, "ip_proto": 132, "eth_type":
sctp_dst	SCTP destination port (int)	{"sctp_dst": 99, "ip_proto": 132, "eth_type":
icmpv4_type	ICMP type (int)	{"icmpv4_type": 5, "ip_proto": 1, "eth_type"
icmpv4_code	ICMP code (int)	{"icmpv4_code": 6, "ip_proto": 1, "eth_type"
arp_op	ARP opcode (int)	{"arp_op": 3, "eth_type": 2054}

Table 6.1 – continued from previous page

Match field	Description	Example
arp_spa	ARP source IPv4 address (string)	{"arp_spa": "192.168.0.11", "eth_type": 2054
arp_tpa	ARP target IPv4 address (string)	{"arp_tpa": "192.168.0.44/24", "eth_type": 2
arp_sha	ARP source hardware address (string)	{"arp_sha": "aa:bb:cc:11:22:33", "eth_type":
arp_tha	ARP target hardware address (string)	{"arp_tha": "aa:bb:cc:11:22:33/00:00:00:00:f
ipv6_src	IPv6 source address (string)	{"ipv6_src": "2001::aaaa:bbbb:cccc:1111", "e
ipv6_dst	IPv6 destination address (string)	{"ipv6_dst": "2001::ffff:cccc:bbbb:1111/64",
ipv6_flabel	IPv6 Flow Label (int)	{"ipv6_flabel": 2, "eth_type": 34525}
icmpv6_type	ICMPv6 type (int)	{"icmpv6_type": 3, "ip_proto": 58, "eth_type
icmpv6_code	ICMPv6 code (int)	{"icmpv6_code": 4, "ip_proto": 58, "eth_type
ipv6_nd_target	Target address for Neighbor Discovery (string)	{"ipv6_nd_target": "2001::ffff:cccc:bbbb:111
ipv6_nd_sll	Source link-layer for Neighbor Discovery (string)	{"ipv6_nd_sll": "aa:bb:cc:11:22:33", "icmpv6
ipv6_nd_tll	Target link-layer for Neighbor Discovery (string)	{"ipv6_nd_tll": "aa:bb:cc:11:22:33", "icmpv6
mpls_label	MPLS label (int)	{"mpls_label": 3, "eth_type": 34888}
mpls_tc	MPLS Traffic Class (int)	{"mpls_tc": 2, "eth_type": 34888}
mpls_bos	MPLS BoS bit (int) (Openflow1.3+)	{"mpls_bos": 1, "eth_type": 34888}
pbb_isid	PBB I-SID (int or string) (Openflow1.3+)	{"pbb_isid": 5, "eth_type": 35047} or{"pbb_
tunnel_id	Logical Port Metadata (int or string) (Openflow1.3+)	{"tunnel_id": 7} or {"tunnel_id": "0x07/0xff"
ipv6_exthdr	IPv6 Extension Header pseudo-field (int or string) (Openflow1.3+)	{"ipv6_exthdr": 3, "eth_type": 34525} or {"i
pbb_uca	PBB UCA hander field(int) (Openflow1.4+)	{"pbb_uca": 1, "eth_type": 35047}
tcp_flags	TCP flags(int) (Openflow1.5+)	{"tcp_flags": 2, "ip_proto": 6, "eth_type": 20
actset_output	Output port from action set metadata(int) (Openflow1.5+)	{"actset_output": 3}
packet_type	Packet type value(int) (Openflow1.5+)	{"packet_type": [1, 2048]}

Note: Some field can be described with mask like as follows.

Ethernet address:

```
"aa:bb:cc:11:22:33"
"aa:bb:cc:11:22:33/00:00:00:00:ff:ff"
```

IPv4 address:

```
"192.168.0.11"
"192.168.0.44/24"
"192.168.10.10/255.255.255.0"
```

IPv6 address:

```
"2001::ffff:cccc:bbbb:1111"
"2001::ffff:cccc:bbbb:2222/64"
"2001::ffff:cccc:bbbb:2222/ffff:ffff:ffff:ffff::0"
```

Metadata:

```
"0x1212121212121212"
"0x3434343434343434/0x0101010101010101"
```

Example of VLAN ID match field

The following is available in OpenFlow1.0 or later.

- To match only packets with VLAN tag and VLAN ID equal value 5:

```
$ curl -X POST -d '{
    "dpid": 1,
    "match":{
        "dl_vlan": 5
    },
    "actions":[
        {
            "type":"OUTPUT",
            "port": 1
        }
    ]
 }' http://localhost:8080/stats/flowentry/add
```

Note: When "dl_vlan" field is described as decimal int value, OFPVID_PRESENT(0x1000) bit is automatically applied.

The following is available in OpenFlow1.2 or later.

- To match only packets without a VLAN tag:

```
$ curl -X POST -d '{
    "dpid": 1,
    "match":{
        "dl_vlan": "0x0000"   # Describe OFPVID_NONE(0x0000)
    },
    "actions":[
        {
            "type":"OUTPUT",
            "port": 1
        }
    ]
 }' http://localhost:8080/stats/flowentry/add
```

- To match only packets with a VLAN tag regardless of its value:

```
$ curl -X POST -d '{
    "dpid": 1,
    "match":{
        "dl_vlan": "0x1000/0x1000"   # Describe OFPVID_PRESENT(0x1000/
↪0x1000)
    },
    "actions":[
        {
            "type":"OUTPUT",
            "port": 1
        }
    ]
 }' http://localhost:8080/stats/flowentry/add
```

- To match only packets with VLAN tag and VLAN ID equal value 5:

```
$ curl -X POST -d '{
    "dpid": 1,
    "match":{
        "dl_vlan": "0x1005"   # Describe sum of VLAN-ID(e.g. 5) | OFPVID_
↪PRESENT(0x1000)
    },
    "actions":[
        {
            "type":"OUTPUT",
            "port": 1
        }
    ]
 }' http://localhost:8080/stats/flowentry/add
```

Note: When using the descriptions for OpenFlow1.2 or later, please describe "dl_vlan" field as hexadecimal string value, and OFPVID_PRESENT(0x1000) bit is NOT automatically applied.

Description of Actions on request messages

List of Actions (OpenFlow1.0):

Actions	Description	Example
OUTPUT	Output packet from "port"	{"type": "OUTPUT", "port": 3}
SET_VLAN_VID	Set the 802.1Q VLAN ID using "vlan_vid"	{"type": "SET_VLAN_VID", "vlan_vid": 5}
SET_VLAN_PCP	Set the 802.1Q priority using "vlan_pcp"	{"type": "SET_VLAN_PCP", "vlan_pcp": 3}
STRIP_VLAN	Strip the 802.1Q header	{"type": "STRIP_VLAN"}
SET_DL_SRC	Set ethernet source address using "dl_src"	{"type": "SET_DL_SRC", "dl_src": "aa:bb:cc:11:22:33"}
SET_DL_DST	Set ethernet destination address using "dl_dst"	{"type": "SET_DL_DST", "dl_dst": "aa:bb:cc:11:22:33"}
SET_NW_SRC	IP source address using "nw_src"	{"type": "SET_NW_SRC", "nw_src": "10.0.0.1"}
SET_NW_DST	IP destination address using "nw_dst"	{"type": "SET_NW_DST", "nw_dst": "10.0.0.1"}
SET_NW_TOS	Set IP ToS (DSCP field, 6 bits) using "nw_tos"	{"type": "SET_NW_TOS", "nw_tos": 184}
SET_TP_SRC	Set TCP/UDP source port using "tp_src"	{"type": "SET_TP_SRC", "tp_src": 8080}
SET_TP_DST	Set TCP/UDP destination port using "tp_dst"	{"type": "SET_TP_DST", "tp_dst": 8080}
ENQUEUE	Output to queue with "queue_id" attached to "port"	{"type": "ENQUEUE", "queue_id": 3, "port": 1}

List of Actions (OpenFlow1.2 or later):

Actions	Description	Example
OUTPUT	Output packet from “port”	{“type”: “OUTPUT”, “port”: 3}
COPY_TTL_OUT	Copy TTL outwards	{“type”: “COPY_TTL_OUT”}
COPY_TTL_IN	Copy TTL inwards	{“type”: “COPY_TTL_IN”}
SET_MPLS_TTL	Set MPLS TTL using “mpls_ttl”	{“type”: “SET_MPLS_TTL”, “mpls_ttl”: 64}
DEC_MPLS_TTL	Decrement MPLS TTL	{“type”: “DEC_MPLS_TTL”}
PUSH_VLAN	Push a new VLAN tag with “ethertype”	{“type”: “PUSH_VLAN”, “ethertype”: 33024}
POP_VLAN	Pop the outer VLAN tag	{“type”: “POP_VLAN”}
PUSH_MPLS	Push a new MPLS tag with “ethertype”	{“type”: “PUSH_MPLS”, “ethertype”: 34887}
POP_MPLS	Pop the outer MPLS tag with “ethertype”	{“type”: “POP_MPLS”, “ethertype”: 2054}
SET_QUEUE	Set queue id using “queue_id” when outputting to a port	{“type”: “SET_QUEUE”, “queue_id”: 7}
GROUP	Apply group identified by “group_id”	{“type”: “GROUP”, “group_id”: 5}
SET_NW_TTL	Set IP TTL using “nw_ttl”	{“type”: “SET_NW_TTL”, “nw_ttl”: 64}
DEC_NW_TTL	Decrement IP TTL	{“type”: “DEC_NW_TTL”}
SET_FIELD	Set a “field” using “value” (The set of keywords available for “field” is the same as match field)	See *Example of set-field action*
PUSH_PBB	Push a new PBB service tag with “ethertype” (Openflow1.3+)	{“type”: “PUSH_PBB”, “ethertype”: 35047}
POP_PBB	Pop the outer PBB service tag (Openflow1.3+)	{“type”: “POP_PBB”}
COPY_FIELD	Copy value between header and register (Openflow1.5+)	{“type”: “COPY_FIELD”, “n_bits”: 32, “src_offset”: 1, “dst_offset”: 2, “src_oxm_id”: “eth_src”, “dst_oxm_id”: “eth_dst”}
METER	Apply meter identified by “meter_id” (Openflow1.5+)	{“type”: “METER”, “meter_id”: 3}
EXPERIMENTER	Extensible action for the experimenter (Set “base64” or “ascii” to “data_type” field)	{“type”: “EXPERIMENTER”, “experimenter”: 101, “data”: “AAECAwQFBgc=”, “data_type”: “base64”}
GOTO_TABLE	(Instruction) Setup the next table identified by “table_id”	{“type”: “GOTO_TABLE”, “table_id”: 8}
WRITE_METADATA	(Instruction) Setup the metadata field using “metadata” and “metadata_mask”	{“type”: “WRITE_METADATA”, “metadata”: 0x3, “metadata_mask”: 0x3}
METER	(Instruction) Apply meter identified by “meter_id” (deprecated in Openflow1.5)	{“type”: “METER”, “meter_id”: 3}
WRITE_ACTIONS	(Instruction) Write the action(s) onto the datapath action set	{“type”: “WRITE_ACTIONS”, actions”:[{“type”:”POP_VLAN”,},{ “type”:”OUTPUT”, “port”: 2}]}
CLEAR_ACTIONS	(Instruction) Clears all actions from the datapath action set	{“type”: “CLEAR_ACTIONS”}

Example of set-field action

To set VLAN ID to non-VLAN-tagged frame:

```
$ curl -X POST -d '{
    "dpid": 1,
    "match":{
        "dl_type": "0x8000"
    },
    "actions":[
        {
            "type": "PUSH_VLAN",     # Push a new VLAN tag if a input frame␣
↪is non-VLAN-tagged
            "ethertype": 33024       # Ethertype 0x8100(=33024): IEEE 802.1Q␣
↪VLAN-tagged frame
        },
        {
            "type": "SET_FIELD",
            "field": "vlan_vid",     # Set VLAN ID
            "value": 4102            # Describe sum of vlan_id(e.g. 6) |␣
↪OFPVID_PRESENT(0x1000=4096)
        },
        {
            "type": "OUTPUT",
            "port": 2
        }
    ]
 }' http://localhost:8080/stats/flowentry/add
```

6.3 ryu.app.rest_vtep

This sample application performs as VTEP for EVPN VXLAN and constructs a Single Subnet per EVI corresponding to the VLAN Based service in [RFC7432].

Note: This app will invoke OVSDB request to the switches. Please set the manager address before calling the API of this app.

```
$ sudo ovs-vsctl set-manager ptcp:6640
$ sudo ovs-vsctl show
    ...(snip)
    Manager "ptcp:6640"
    ...(snip)
```

6.3.1 Usage Example

Environment

This example supposes the following environment:

```
 Host A (172.17.0.1)                     Host B (172.17.0.2)
+---------------------+                 +---------------------+
|   Ryu1               | --- BGP(EVPN) --- |   Ryu2               |
+----------------------+                   +----------------------+
        |                                          |
+----------------------+                   +----------------------+
|   s1 (OVS)           | ===== vxlan ===== |   s2 (OVS)           |
+----------------------+                   +----------------------+
(s1-eth1)     (s1-eth2)                    (s2-eth1)     (s2-eth2)
   |             |                            |             |
+---------+  +---------+                   +---------+  +---------+
| s1h1    |  | s1h2    |                   | s2h1    |  | s2h2    |
+---------+  +---------+                   +---------+  +---------+
```

Configuration steps

1. Creates a new BGPSpeaker instance on each host.

   On Host A:

```
(Host A)$ curl -X POST -d '{
 "dpid": 1,
 "as_number": 65000,
 "router_id": "172.17.0.1"
 }' http://localhost:8080/vtep/speakers | python -m json.tool
```

   On Host B:

```
(Host B)$ curl -X POST -d '{
 "dpid": 1,
 "as_number": 65000,
 "router_id": "172.17.0.2"
 }' http://localhost:8080/vtep/speakers | python -m json.tool
```

2. Registers the neighbor for the speakers on each host.

   On Host A:

```
(Host A)$ curl -X POST -d '{
 "address": "172.17.0.2",
 "remote_as": 65000
 }' http://localhost:8080/vtep/neighbors |
 python -m json.tool
```

   On Host B:

```
(Host B)$ curl -X POST -d '{
 "address": "172.17.0.1",
 "remote_as": 65000
 }' http://localhost:8080/vtep/neighbors |
 python -m json.tool
```

3. Defines a new VXLAN network(VNI=10) on the Host A/B.

   On Host A:

```
(Host A)$ curl -X POST -d '{
 "vni": 10
 }' http://localhost:8080/vtep/networks | python -m json.tool
```

On Host B:

```
(Host B)$ curl -X POST -d '{
 "vni": 10
 }' http://localhost:8080/vtep/networks | python -m json.tool
```

4. Registers the clients to the VXLAN network.

   For "s1h1"(ip="10.0.0.11", mac="aa:bb:cc:00:00:11") on Host A:

```
(Host A)$ curl -X POST -d '{
 "port": "s1-eth1",
 "mac": "aa:bb:cc:00:00:11",
 "ip": "10.0.0.11"
 } ' http://localhost:8080/vtep/networks/10/clients |
 python -m json.tool
```

   For "s2h1"(ip="10.0.0.21", mac="aa:bb:cc:00:00:21") on Host B:

```
(Host B)$ curl -X POST -d '{
 "port": "s2-eth1",
 "mac": "aa:bb:cc:00:00:21",
 "ip": "10.0.0.21"
 } ' http://localhost:8080/vtep/networks/10/clients |
 python -m json.tool
```

Testing

If BGP (EVPN) connection between Ryu1 and Ryu2 has been established, pings between the client s1h1 and s2h1 should work.

```
(s1h1)$ ping 10.0.0.21
```

Troubleshooting

If connectivity between s1h1 and s2h1 isn't working, please check the followings.

1. Make sure that Host A and Host B have full network connectivity.

```
(Host A)$ ping 172.17.0.2
```

2. Make sure that BGP(EVPN) connection has been established.

```
(Host A)$ curl -X GET http://localhost:8080/vtep/neighbors |
 python -m json.tool

...
{
    "172.17.0.2": {
        "EvpnNeighbor": {
            "address": "172.17.0.2",
            "remote_as": 65000,
            "state": "up"  # "up" shows the connection established
        }
    }
}
```

3. Make sure that BGP(EVPN) routes have been advertised.

```
(Host A)$ curl -X GET http://localhost:8080/vtep/networks |
 python -m json.tool

 ...
{
    "10": {
        "EvpnNetwork": {
            "clients": {
                "aa:bb:cc:00:00:11": {
                    "EvpnClient": {
                        "ip": "10.0.0.11",
                        "mac": "aa:bb:cc:00:00:11",
                        "next_hop": "172.17.0.1",
                        "port": 1
                    }
                },
                "aa:bb:cc:00:00:21": {  # route for "s2h1" on Host B
                    "EvpnClient": {
                        "ip": "10.0.0.21",
                        "mac": "aa:bb:cc:00:00:21",
                        "next_hop": "172.17.0.2",
                        "port": 3
                    }
                }
            },
            "ethernet_tag_id": 0,
            "route_dist": "65000:10",
            "vni": 10
        }
    }
}
```

6.3.2 REST API

class `ryu.app.rest_vtep.`**`RestVtepController`**(*req*, *link*, *data*, ***config*)

`add_speaker`(*req*, ***kwargs*)

Creates a new BGPSpeaker instance.

Usage:

Method	URI
POST	/vtep/speakers

Request parameters:

Attribute	Description
dpid	ID of Datapath binding to speaker. (e.g. 1)
as_number	AS number. (e.g. 65000)
router_id	Router ID. (e.g. "172.17.0.1")

Example:

```
$ curl -X POST -d '{
 "dpid": 1,
"as_number": 65000,
"router_id": "172.17.0.1"
}' http://localhost:8080/vtep/speakers | python -m json.tool
```

```
{
    "172.17.0.1": {
        "EvpnSpeaker": {
            "as_number": 65000,
            "dpid": 1,
            "neighbors": {},
            "router_id": "172.17.0.1"
        }
    }
}
```

get_speakers(_, ***kwargs*)

Gets the info of BGPSpeaker instance.

Usage:

Method	URI
GET	/vtep/speakers

Example:

```
$ curl -X GET http://localhost:8080/vtep/speakers |
 python -m json.tool
```

```
{
    "172.17.0.1": {
        "EvpnSpeaker": {
            "as_number": 65000,
            "dpid": 1,
            "neighbors": {
                "172.17.0.2": {
                    "EvpnNeighbor": {
                        "address": "172.17.0.2",
                        "remote_as": 65000,
                        "state": "up"
                    }
                }
            },
            "router_id": "172.17.0.1"
        }
    }
}
```

del_speaker(_, ***kwargs*)

Shutdowns BGPSpeaker instance.

Usage:

Method	URI
DELETE	/vtep/speakers

Example:

```
$ curl -X DELETE http://localhost:8080/vtep/speakers |
 python -m json.tool
```

```
{
    "172.17.0.1": {
        "EvpnSpeaker": {
            "as_number": 65000,
            "dpid": 1,
            "neighbors": {},
            "router_id": "172.17.0.1"
        }
    }
}
```

add_neighbor(*req*, ***kwargs*)

Registers a new neighbor to the speaker.

Usage:

Method	URI
POST	/vtep/neighbors

Request parameters:

Attribute	Description
address	IP address of neighbor. (e.g. “172.17.0.2”)
remote_as	AS number of neighbor. (e.g. 65000)

Example:

```
$ curl -X POST -d '{
 "address": "172.17.0.2",
 "remote_as": 65000
 }' http://localhost:8080/vtep/neighbors |
 python -m json.tool
```

```
{
    "172.17.0.2": {
        "EvpnNeighbor": {
            "address": "172.17.0.2",
            "remote_as": 65000,
            "state": "down"
        }
    }
}
```

get_neighbors(*_*, ***kwargs*)

Gets a list of all neighbors.

Usage:

Method	URI
GET	/vtep/neighbors

Example:

```
$ curl -X GET http://localhost:8080/vtep/neighbors |
 python -m json.tool
```

```
{
    "172.17.0.2": {
        "EvpnNeighbor": {
            "address": "172.17.0.2",
            "remote_as": 65000,
            "state": "up"
        }
    }
}
```

get_neighbor(_, ***kwargs*)

Gets the neighbor for the specified address.

Usage:

Method	URI
GET	/vtep/neighbors/{address}

Request parameters:

Attribute	Description
address	IP address of neighbor. (e.g. "172.17.0.2")

Example:

```
$ curl -X GET http://localhost:8080/vtep/neighbors/172.17.0.2 |
 python -m json.tool
```

```
{
    "172.17.0.2": {
        "EvpnNeighbor": {
            "address": "172.17.0.2",
            "remote_as": 65000,
            "state": "up"
        }
    }
}
```

del_neighbor(_, ***kwargs*)

Unregister the specified neighbor from the speaker.

Usage:

Method	URI
DELETE	/vtep/speaker/neighbors/{address}

Request parameters:

Attribute	Description
address	IP address of neighbor. (e.g. "172.17.0.2")

Example:

```
$ curl -X DELETE http://localhost:8080/vtep/speaker/neighbors/172.17.0.2 |
 python -m json.tool
```

```
{
    "172.17.0.2": {
        "EvpnNeighbor": {
            "address": "172.17.0.2",
            "remote_as": 65000,
            "state": "up"
        }
    }
}
```

add_network (*req*, ***kwargs*)

Defines a new network.

Usage:

Method	URI
POST	/vtep/networks

Request parameters:

Attribute	Description
vni	Virtual Network Identifier. (e.g. 10)

Example:

```
$ curl -X POST -d '{
 "vni": 10
 }' http://localhost:8080/vtep/networks | python -m json.tool
```

```
{
    "10": {
        "EvpnNetwork": {
            "clients": {},
            "ethernet_tag_id": 0,
            "route_dist": "65000:10",
            "vni": 10
        }
    }
}
```

get_networks (*_*, ***kwargs*)

Gets a list of all networks.

Usage:

Method	URI
GET	/vtep/networks

Example:

```
$ curl -X GET http://localhost:8080/vtep/networks |
 python -m json.tool
```

```
{
    "10": {
        "EvpnNetwork": {
            "clients": {
                "aa:bb:cc:dd:ee:ff": {
                    "EvpnClient": {
                        "ip": "10.0.0.1",
                        "mac": "aa:bb:cc:dd:ee:ff",
                        "next_hop": "172.17.0.1",
                        "port": 1
                    }
                }
            },
            "ethernet_tag_id": 0,
            "route_dist": "65000:10",
            "vni": 10
        }
    }
}
```

get_network (_, ***kwargs*)

Gets the network for the specified VNI.

Usage:

Method	URI
GET	/vtep/networks/{vni}

Request parameters:

Attribute	Description
vni	Virtual Network Identifier. (e.g. 10)

Example:

```
$ curl -X GET http://localhost:8080/vtep/networks/10 |
 python -m json.tool
```

```
{
    "10": {
        "EvpnNetwork": {
            "clients": {
                "aa:bb:cc:dd:ee:ff": {
                    "EvpnClient": {
                        "ip": "10.0.0.1",
                        "mac": "aa:bb:cc:dd:ee:ff",
                        "next_hop": "172.17.0.1",
                        "port": 1
                    }
                }
            },
            "ethernet_tag_id": 0,
            "route_dist": "65000:10",
            "vni": 10
        }
    }
}
```

del_network (_, ***kwargs*)

Deletes the network for the specified VNI.

Usage:

Method	URI
DELETE	/vtep/networks/{vni}

Request parameters:

Attribute	Description
vni	Virtual Network Identifier. (e.g. 10)

Example:

```
$ curl -X DELETE http://localhost:8080/vtep/networks/10 |
 python -m json.tool
```

```
{
    "10": {
        "EvpnNetwork": {
            "ethernet_tag_id": 10,
            "clients": [
                {
                    "EvpnClient": {
                        "ip": "10.0.0.11",
                        "mac": "e2:b1:0c:ba:42:ed",
                        "port": 1
                    }
                }
            ],
            "route_dist": "65000:100",
            "vni": 10
        }
    }
}
```

add_client (*req*, ***kwargs*)

Registers a new client to the specified network.

Usage:

Method	URI
POST	/vtep/networks/{vni}/clients

Request parameters:

Attribute	Description
vni	Virtual Network Identifier. (e.g. 10)
port	Port number to connect client. For convenience, port name can be specified and automatically translated to port number. (e.g. “s1-eth1” or 1)
mac	Client MAC address to register. (e.g. “aa:bb:cc:dd:ee:ff”)
ip	Client IP address. (e.g. “10.0.0.1”)

Example:

```
$ curl -X POST -d '{
 "port": "s1-eth1",
 "mac": "aa:bb:cc:dd:ee:ff",
 "ip": "10.0.0.1"
 }' http://localhost:8080/vtep/networks/10/clients |
 python -m json.tool
```

```
{
    "10": {
        "EvpnClient": {
            "ip": "10.0.0.1",
            "mac": "aa:bb:cc:dd:ee:ff",
            "next_hop": "172.17.0.1",
            "port": 1
        }
    }
}
```

del_client(_, ***kwargs*)

Registers a new client to the specified network.

Usage:

Method	URI
DELETE	/vtep/networks/{vni}/clients/{mac}

Request parameters:

Attribute	Description
vni	Virtual Network Identifier. (e.g. 10)
mac	Client MAC address to register.

Example:

```
$ curl -X DELETE http://localhost:8080/vtep/networks/10/clients/
↪aa:bb:cc:dd:ee:ff |
 python -m json.tool
```

```
{
    "10": {
        "EvpnClient": {
            "ip": "10.0.0.1",
            "mac": "aa:bb:cc:dd:ee:ff",
            "next_hop": "172.17.0.1",
            "port": 1
        }
    }
}
```

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index