Ryu的应用开发（一）Hub实现

一：Hub/集线器 （编程思路）《重点》

（一）明确问题

（二）设计解决方案

（三）确定具体的技术方案

（四）部署实施

（五）验证方案

（六）优化

二：集线器原理---设计解决方案

三：部署实施---Ryu控制器API学习和使用（Hub集线器开发）

（一）代码实现

（二）启动控制器

（三）启动Mininet进行连接测试

（四）使用pingall命令，使得主机向交换机发送数据包---从而实现交换机上传数据包到控制器，实现流表...

四： Hub代码讲解（注释版）

通信流程：《重点》

五：实现整体程序运行了解《重点》

（一）程序入口在哪？------app_manager.RyuApp

1.app_manager.RyuApp是所有Ryu Applications的基类，我们要实现一个控制器应用，必须继承该基类

2.我们自定义的子类（继承于RyuAPP的子类），将在ryu-manager命令加载中被实例化(它是在ryu管理...

3.子类中的__init__方法需要调用父类的__init__方法，并且保持参数一致

（二）设置OpenFlow协议---OFP_VERSIONS

（三）事件监听----装饰器实现set_ev_cls

1.from ryu.controller.handler import set_ev_cls

2.被set_ev_cls装饰的函数将成为一个事件处理器，参数ev_cls是一个事件类，在controller下的ofp_event...

3.装饰器实现代码

（四）协议解析

1.def switch_features_handler(self,ev):自定义函数参数中ev

2.查看事件中的msg类信息

3.获取逻辑设备datapath类信息

4.获取逻辑设备datapath中OpenFlow协议信息

5.获取逻辑设备datapath中OpenFlow协议解析信息

5.通过ofp_parser中获取match类匹配

5.通过ofp_parser中获取actions类匹配

6.match = ofp_parser.OFPMatch()中OFPMatch类协议支持的匹配方式

7. out = ofp_parser.OFPPacketOut(datapath=datapath,buffer_id=msg.buffer_id,in_port=in_port,actio...

8. inst = [ofp_parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS,actions)]

9. mod = ofp_parser.OFPFlowMod(datapath=datapath,priority=priority,match=match,instructions=in...

10. actions = [ofp_parser.OFPActionOutput(ofproto.OFPP_CONTROLLER,ofproto.OFPCML_NO_BUFF...

11.ofproto.OFPP_CONTROLLER,ofproto.OFPCML_NO_BUFFER 一个输出端口，一个是数据包最大长度

12.@set_ev_cls(ofp_event.EventOFPSwitchFeatures,CONFIG_DISPATCHER)中的事件，由ofp_event下...

一：Hub/集线器 （编程思路）《重点》

（一）明确问题

如何实现软件定义的集线器?

（二）设计解决方案

通过控制器来实现集线器算法（泛洪），然后指导数据平面实现集线器操作

（三）确定具体的技术方案

控制器选用Ryu，数据平面通过Mininet模拟

（四）部署实施

在控制器上编程开发集线器应用，创建实验网络为验证方案做准备

（五）验证方案

运行程序，调试程序，验证程序

（六）优化

验证成功后，优化程序

二：集线器原理---设计解决方案

一个数据包从port1进入，会被复制，泛洪转发到其他所有端口发出

三：部署实施---Ryu控制器API学习和使用（Hub集线器开发）

（一）代码实现

Python | 复制代码

```python
from ryu.base import app_manager
from ryu.ofproto import ofproto_v1_3
from ryu.controller import ofp_event
from ryu.controller.handler import MAIN_DISPATCHER,CONFIG_DISPATCHER
from ryu.controller.handler import set_ev_cls


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

    def __init__(self,*args,**kwargs):
        super(Hub,self).__init__(*args,**kwargs)


    @set_ev_cls(ofp_event.EventOFPSwitchFeatures,CONFIG_DISPATCHER)
    def switch_features_handler(self,ev):
        datapath = ev.msg.datapath
        ofproto = datapath.ofproto
        ofp_parser = datapath.ofproto_parser

        match = ofp_parser.OFPMatch()
        actions = [ofp_parser.OFPActionOutput(ofproto.OFPP_CONTROLLER,ofproto.OFPCML_NO_BUFFER)]

        self.add_flow(datapath,0,match,actions,"default flow entry")

    def add_flow(self,datapath,priority,match,actions,remind_content):
        ofproto = datapath.ofproto
        ofp_parser = datapath.ofproto_parser

        inst = [ofp_parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS,
                                                 actions)]

        mod = ofp_parser.OFPFlowMod(datapath=datapath,priority=priority,
                                    match=match,instructions=inst);
        print("install to datapath,"+remind_content)
        datapath.send_msg(mod);


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

```
        ofp_parser = datapath.ofproto_parser

        in_port = msg.match['in_port']

        print("get packet in, install flow entry,and lookback parket to datapath")

        match = ofp_parser.OFPMatch();
        actions = [ofp_parser.OFPActionOutput(ofproto.OFPP_FLOOD)]

        self.add_flow(datapath,1,match,actions,"hub flow entry")

        out = ofp_parser.OFPPacketOut(datapath=datapath,buffer_id=msg.buffer_id,
                                      in_port=in_port,actions=actions)

        datapath.send_msg(out);
```

注意：注释可能为你带来不少错误...尽可能写两个版本，一个不带注释，用于调试。一个写注释，用于学习，回顾

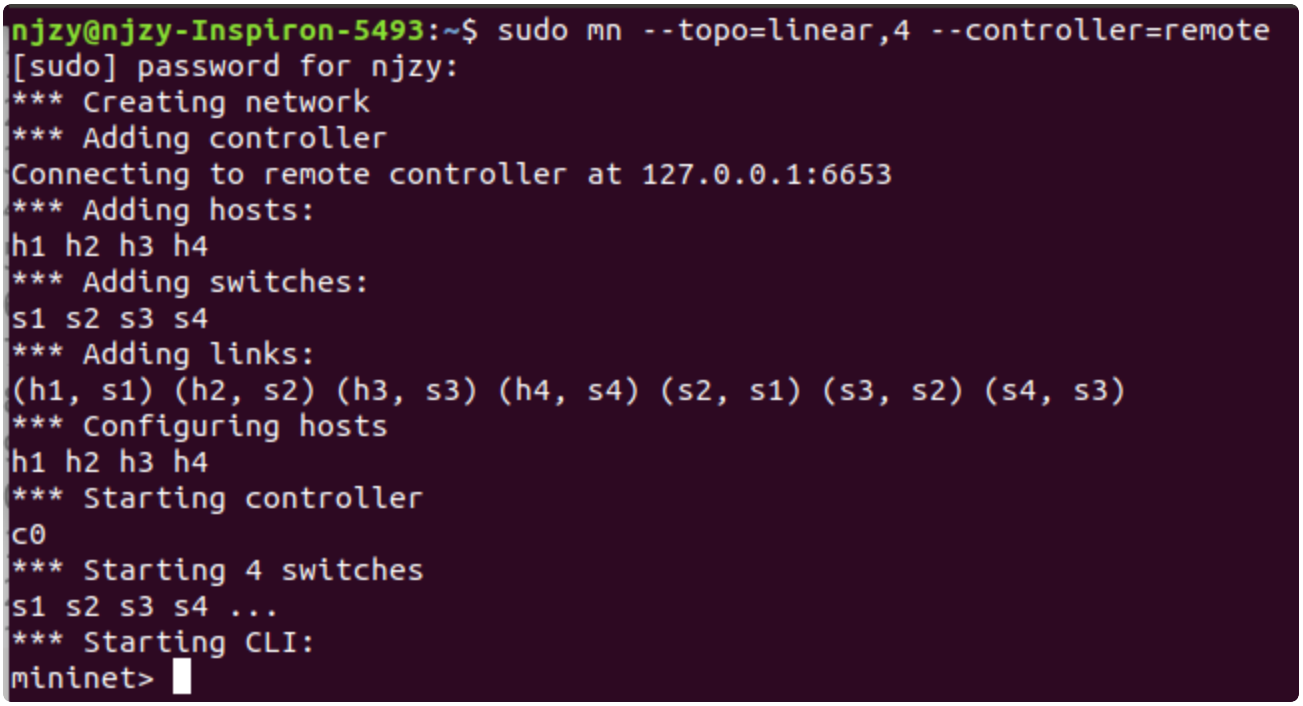
（二）启动控制器

ryu-manager hub.py --verbose #进入目录，在hub.py文件目录下 **--verbose显示调试信息**

```
njzy@njzy-Inspiron-5493:~/CODE/python/SDN_Controller/ryu/ryu/app$ ryu-manager hub.py --verbose
loading app hub.py
loading app ryu.controller.ofp_handler
instantiating app hub.py of Hub
instantiating app ryu.controller.ofp_handler of OFPHandler
BRICK Hub
  CONSUMES EventOFPPacketIn
  CONSUMES EventOFPSwitchFeatures
BRICK ofp_event
  PROVIDES EventOFPPacketIn TO {'Hub': {'main'}}
  PROVIDES EventOFPSwitchFeatures TO {'Hub': {'config'}}
  CONSUMES EventOFPEchoReply
  CONSUMES EventOFPEchoRequest
  CONSUMES EventOFPErrorMsg
  CONSUMES EventOFPHello
  CONSUMES EventOFPPortDescStatsReply
  CONSUMES EventOFPPortStatus
  CONSUMES EventOFPSwitchFeatures
```

（三）启动Mininet进行连接测试

sudo mn ––topo=linear,4 ––controller=remote



Ryu控制器端响应：（注意：启动mininet后，再先关闭Ryu，重新进入，测试效果更好）

```
njzy@njzy-Inspiron-5493:~/CODE/python/SDN_Controller/ryu/ryu/app$ ryu-manager hub.py --verbose
loading app hub.py
loading app ryu.controller.ofp_handler
instantiating app hub.py of Hub
instantiating app ryu.controller.ofp_handler of OFPHandler
BRICK Hub
  CONSUMES EventOFPPacketIn
  CONSUMES EventOFPSwitchFeatures
BRICK ofp_event
  PROVIDES EventOFPPacketIn TO {'Hub': {'main'}}
  PROVIDES EventOFPSwitchFeatures TO {'Hub': {'config'}}
  CONSUMES EventOFPEchoReply
  CONSUMES EventOFPEchoRequest
  CONSUMES EventOFPErrorMsg
  CONSUMES EventOFPHello
  CONSUMES EventOFPPortDescStatsReply
  CONSUMES EventOFPPortStatus
  CONSUMES EventOFPSwitchFeatures
connected socket:<eventlet.greenio.base.GreenSocket object at 0x7f0780a05278> address:('127.0.0.1', 44004)
hello ev <ryu.controller.ofp_event.EventOFPHello object at 0x7f0780a019e8>
move onto config mode
connected socket:<eventlet.greenio.base.GreenSocket object at 0x7f0780a05080> address:('127.0.0.1', 44006)
hello ev <ryu.controller.ofp_event.EventOFPHello object at 0x7f0780a014a8>
move onto config mode
connected socket:<eventlet.greenio.base.GreenSocket object at 0x7f0780a01fd0> address:('127.0.0.1', 44008)
hello ev <ryu.controller.ofp_event.EventOFPHello object at 0x7f07809ebf28>
move onto config mode
connected socket:<eventlet.greenio.base.GreenSocket object at 0x7f0780a01e48> address:('127.0.0.1', 44010)
hello ev <ryu.controller.ofp_event.EventOFPHello object at 0x7f0780a184e0>
move onto config mode
EVENT ofp_event->Hub EventOFPSwitchFeatures
switch features ev version=0x4,msg_type=0x6,msg_len=0x20,xid=0x9f3b923d,OFPSwitchFeatures(auxiliary_id=0,capabilitie
s=79,datapath_id=4,n_buffers=0,n_tables=254)
install to datapath,default flow entry
EVENT ofp_event->Hub EventOFPSwitchFeatures
switch features ev version=0x4,msg_type=0x6,msg_len=0x20,xid=0x337929f0,OFPSwitchFeatures(auxiliary_id=0,capabilitie
s=79,datapath_id=1,n_buffers=0,n_tables=254)
EVENT ofp_event->Hub EventOFPSwitchFeatures
switch features ev version=0x4,msg_type=0x6,msg_len=0x20,xid=0xc0e5b555,OFPSwitchFeatures(auxiliary_id=0,capabilitie
s=79,datapath_id=3,n_buffers=0,n_tables=254)
EVENT ofp_event->Hub EventOFPSwitchFeatures
switch features ev version=0x4,msg_type=0x6,msg_len=0x20,xid=0x7e0a056f,OFPSwitchFeatures(auxiliary_id=0,capabilitie
s=79,datapath_id=2,n_buffers=0,n_tables=254)
install to datapath,default flow entry
install to datapath,default flow entry
install to datapath,default flow entry
move onto main mode
move onto main mode
move onto main mode
move onto main mode
```

openvswitch交换机与Ryu控制器连接，控制器下发默认流表，提示信息**install to datapath,default flow entry**

（四）使用pingall命令，使得主机向交换机发送数据包---从而实现交换机上传数据包到控制器，实现流表获取

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
```

获取提示信息get packet in, install flow entry,and lookback parket to datapath

```
get packet in, install flow entry,and lookback parket to datapath
install to datapath,hub flow entry
EVENT ofp_event->Hub EventOFPPacketIn
get packet in, install flow entry,and lookback parket to datapath
install to datapath,hub flow entry
EVENT ofp_event->Hub EventOFPPacketIn
get packet in, install flow entry,and lookback parket to datapath
install to datapath,hub flow entry
EVENT ofp_event->Hub EventOFPPacketIn
EVENT ofp_event->Hub EventOFPPacketIn
get packet in, install flow entry,and lookback parket to datapath
install to datapath,hub flow entry
get packet in, install flow entry,and lookback parket to datapath
install to datapath,hub flow entry
```

四： Hub代码讲解（注释版）

Python | 复制代码

```python
from ryu.base import app_manager
from ryu.ofproto import ofproto_v1_3
from ryu.controller import ofp_event
from ryu.controller.handler import MAIN_DISPATCHER,CONFIG_DISPATCHER
from ryu.controller.handler import set_ev_cls


class Hub(app_manager.RyuApp):
    '''明确控制器所用OpenFlow版本'''
    OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]

    def __init__(self,*args,**kwargs):
        super(Hub,self).__init__(*args,**kwargs)

''' set_ev_cls指定事件类别得以接受消息和交换机状态作为参数
    其中事件类别名称为ryu.controller.ofp_event.EventOFP+<OpenFlow消息名称>
    例如：在 Packet-In 消息的状态下的事件名称为EventOFPPacketIn
    对于交换机的状态来说，可指定以下中的一项
        ryu.controller.handler.HANDSHAKE_DISPATCHER 交换 HELLO 消息
        ryu.controller.handler.CONFIG_DISPATCHER 接收SwitchFeatures消息
        ryu.controller.handler.MAIN_DISPATCHER 一般状态
        ryu.controller.handler.DEAD_DISPATCHER 连线中断'''
    @set_ev_cls(ofp_event.EventOFPSwitchFeatures,CONFIG_DISPATCHER)
    def switch_features_handler(self,ev):
        '''
        在Ryu控制器上，我们需要写一个函数去处理openvswitch的连接
        CONFIG_DISPATCHER : Version negotiated and sent features-request message
        '''
        #对事件进行解析
        # ev.msg 是用来存储对应事件的 OpenFlow 消息类别实体
        # msg.datapath是用来存储OpenFlow交换机的 ryu.controller.controller.Datapath 类别所对应的实体
        datapath = ev.msg.datapath    #从连接中获取数据平面的datapath数据结构
        ofproto = datapath.ofproto    #获取OpenFlow协议信息
        ofp_parser = datapath.ofproto_parser    #获取协议解析
        #解析完成

   '''在连接建立成功以后，需要控制器下发一个默认流表
           来指挥所有匹配不到交换机的数据，把他上传到控制器上
        '''

        # 下发table-miss流表项，让交换机对于不会处理的数据包通过packet-in消息上交给Ryu控制器！！！
        # 匹配数据包
```

```
        # 若数据包没有 match 任何一个普通 Flow Entry 时，则触发 Packet-In

        match = ofp_parser.OFPMatch()        #匹配域

        #OFPActionOutput将数据包发送出去，
        #第一个参数OFPP_CONTROLLER是接收端口，
        #第二个是数据包在交换机上缓存buffer_id,由于我们将数据包全部传送到控制器，所以不在交换机上缓存
        actions = [ofp_parser.OFPActionOutput(ofproto.OFPP_CONTROLLER,ofproto.OFPCML_NO_BUFFER)]

        self.add_flow(datapath,0,match,actions,"default flow entry")    #默认缺省流表项，设置优先级最低即可
    '''
        数据平面是由若干网元（Network Element）组成，每个网元包含一个或多个SDN数据路径（SDN Datapath）。
        SDN Datapath是逻辑上的网络设备，负责转发和处理数据，无控制能力，一个SDN DataPath包含控制数据平面
        接口（Control Data Plane Interface,CDPI）、代理、转发引擎（Forwarding Engine）表和处理功能（Processing Function）
        SDN数据面（转发面）的关键技术:对数据面进行抽象建模。
        '''
    def add_flow(self,datapath,priority,match,actions,remind_content):
        '''构建流表项 : add a flow entry, install it into datapath
        datapath:表示给哪一个逻辑设备下发流表
        priority：表示优先级
        match,actions:匹配域和动作
        '''

        #datapath属性
        ofproto = datapath.ofproto
        ofp_parser = datapath.ofproto_parser

        #在OpenFlow1.3版本中定义了instruct指令集（交换机内部的一些操作）
        #construct a flow msg and send it
        # Apply Actions 是用来设定那些必须立即执行的 action 所使用
        inst = [ofp_parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS,
                                                 actions)]
        # 通过 Flow Mod 消息将 Flow Entry 新增到 Flow table 中
        mod = ofp_parser.OFPFlowMod(datapath=datapath,priority=priority,
                                    match=match,instructions=inst)
        print("install to datapath,"+remind_content)
        #发送出去
        datapath.send_msg(mod)


'''接收数据
```

```
        Ryu控制器通过装饰器去注册监听某些事件，去处理这些事件。
        从而实现从数据平面的消息上传到控制器，再从控制器平面到应用平面，应用程序去处理事件，再逐跳返回到openvswitch
    '''

    '''要处理这个事件，需要先去注册监听他
    EventOFPPacketIn: 是我们要监听的事件
    MAIN_DISPATCHER : 是什么状态下，去监听该事件---Switch-features message received and sent set-config message
    '''
    @set_ev_cls(ofp_event.EventOFPPacketIn,MAIN_DISPATCHER)
    def packet_in_handler(self,ev):
        '''Hub集线器类，所实现的功能：
        1.接收从OpenVSwitch发送过来的数据包
        2.将数据包泛洪到Hub中的其他端口中
        '''

        #解析数据结构
        msg = ev.msg
        datapath = msg.datapath
        ofproto = datapath.ofproto
        ofp_parser = datapath.ofproto_parser

        in_port = msg.match['in_port']        #获取源端口

        print("get packet in, install flow entry,and lookback parket to datapath")

        match = ofp_parser.OFPMatch();        #因为我们是将所有转发，所以不用管匹配，填空表示全部匹配
        actions = [ofp_parser.OFPActionOutput(ofproto.OFPP_FLOOD)]    #注意：FLOOD是OpenFlow协议保留端口－－－泛洪使用

        #调用add_flow,将流表项发送,指导后续数据包转发    install flow entry to avoid packet in next time
        self.add_flow(datapath,1,match,actions,"hub flow entry")    #等级稍微比默认流表项高级

        #注意：我们将流表项下发了，但是我们这次接收的数据包，并没有处理
        #需要再将控制器上的数据包，重新发送给datapath,让他按照流表项处理
        #buffer_id是这个数据包，存放在控制器中的缓冲区位置,是在事件中的buffer_id获取
        out = ofp_parser.OFPPacketOut(datapath=datapath,buffer_id=msg.buffer_id,
                                      in_port=in_port,actions=actions,data=msg.data)

        datapath.send_msg(out);
```

通信流程：《重点》

1.当开始一个Hub集线器时，会先与控制器进行连接，我们需要在Ryu中设置函数去处理连接，设置并下发默认流表----------函数switch_features_handler实现
2.当主机之间通信时，主机上传信息到OpenVSwitch交换机，而交换机无法匹配到流表项时，我们设置将数据全部上传给Ryu控制器，我们在控制器端实现Hub集线器的泛洪功能，即设置流表项（match-actions为所有匹配数据包的动作为ofproto.OFPP_FLOOD，并且将该流变下发给原来datapath,同时我们要将之前交换机发送过来的数据包重新发送给交换机（让其按照新的流表项进行处理）----------函数packet_in_handler实现
3.我们将公共函数add_flow，构建流表项并且下发流表提出

五：实现整体程序运行了解《重点》

（一）程序入口在哪？------app_manager.RyuApp

Python | 复制代码

```
"""
    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.
    """
```

1.app_manager.RyuApp是所有Ryu Applications的基类，我们要实现一个控制器应用，必须继承该基类

2.我们自定义的子类（继承于RyuAPP的子类），将在ryu-manager命令加载中被实例化(它是在ryu管理器加载所有请求的ryu应用程序模块后实例化的)

```
/ryu/ryu/app$ ryu-manager hub.py --verbose
```

即我们执行ryu-manager hub.py --verbose命令开启Ryu控制器时，并且处理了所有请求的ryu应用程序模块，之后Hub子类就被实例化了

3.子类中的__init__方法需要调用父类的__init__方法，并且保持参数一致

Python | 复制代码

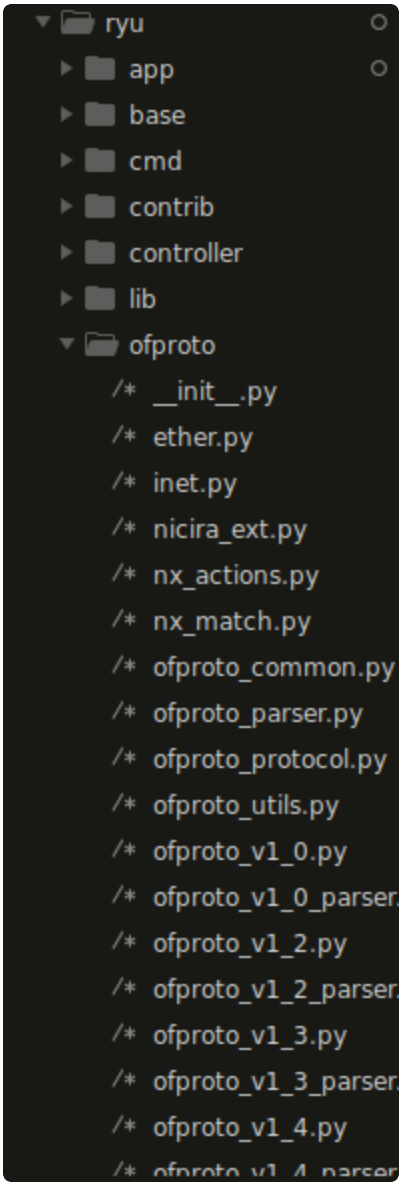
```python
def __init__(self,*args,**kwargs):
        super(Hub,self).__init__(*args,**kwargs)
```

（二）设置OpenFlow协议---OFP_VERSIONS

Python | 复制代码

```python
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.
    """
```

我们设置的协议类型是1.3版本OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]，其中协议在 from ryu.ofproto import ofproto_v1_3 中



（三）事件监听----装饰器实现set_ev_cls

1.from ryu.controller.handler import set_ev_cls

Python | 复制代码

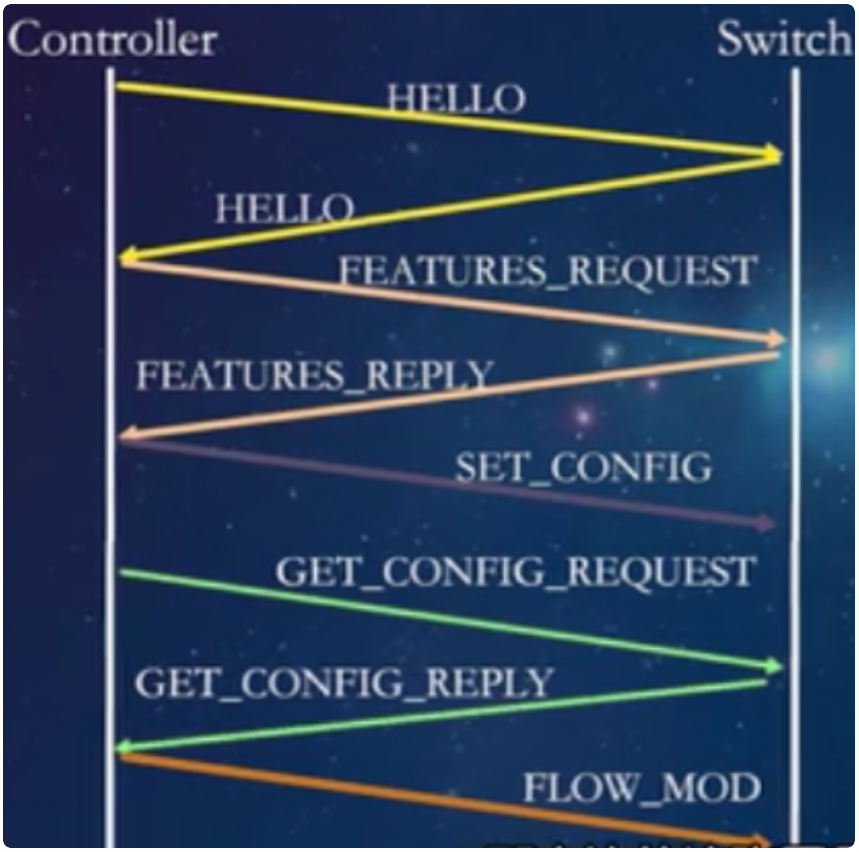
```python
# should be named something like 'observe_event'
def set_ev_cls(ev_cls, dispatchers=None):
    """
    A decorator for Ryu application to declare an event handler.

    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.

    .. tabularcolumns:: |l|L|

    ========================================= ==========================
    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.
    ========================================= ==========================
    """
    def _set_ev_cls_dec(handler):
        if 'callers' not in dir(handler):
            handler.callers = {}
        for e in _listify(ev_cls):
            handler.callers[e] = _Caller(_listify(dispatchers), e.__module__)
        return handler
```

```
36        return _set_ev_cls_dec
```

2.被set_ev_cls装饰的函数将成为一个事件处理器，参数ev_cls是一个事件类，在controller下的ofp_event下（未找到....动态创建类type实现？？），dispatchers参数是事件的协商阶段

@set_ev_cls(ofp_event.EventOFPSwitchFeatures,CONFIG_DISPATCHER)

(1) 协商阶段



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 coverable errors.

发送并等待Hello消息

双方通过握手消息Hello建立安全连接

版本协商并发送功能请求消息

双方建立TLS隧道后，方法发送hello消息进行版本协商

如果协议版本协商成功，则连接建立。否则发送Error消息描述协商失败原因，并终止连接

交换机特征消息接收和发送设置配置消息

协商完成后，控制器和交换机之间发送Features消息,获取交换机参数

参数包括支持的buffer数目、流表数、Actions等

控制器发送SET_CONFIG消息向交换机发送配置参数

通过GET_CONFIG消息得到交换机修改后的配置信息

控制器与OpenFlow交换机之间，发送PACKET_OUT和PACKET_IN消息。通过PACKET_OUT中内置的LLDP包进行网络拓扑的探测

控制器通过FLOW_MOD向控制器下发流表操作

断开与对等方的连接。或者由于一些不可恢复的错误而断开连接

3.装饰器实现代码

Python | 复制代码

```python
def set_ev_cls(ev_cls, dispatchers=None):
    def _set_ev_cls_dec(handler):
        if 'callers' not in dir(handler):
            handler.callers = {}
        for e in _listify(ev_cls):
            handler.callers[e] = _Caller(_listify(dispatchers), e.__module_
_)
        return handler
    return _set_ev_cls_dec
```

外部函数是用来接收事件处理函数和函数参数

内联函数是用来

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

（四）协议解析

1.def switch_features_handler(self,ev):自定义函数参数中ev

```
get ev info:
<ryu.controller.ofp_event.EventOFPSwitchFeatures object at 0x7f659220f668>
```

就是我们监听的事件类实例

```
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__le__', '__lt__', '__module__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', '__weakref__',
 'msg', 'timestamp']
```

2.查看事件中的msg类信息

Python | 复制代码

```
EVENT ofp_event->Hub EventOFPPacketIn
version=0x4,
msg_type=0xa,
msg_len=0x80,
xid=0x0,
OFPPacketIn(
    buffer_id=4294967295,
    cookie=0,
    data=b'33\xff\x9a\xd7\xb1\x1aj\xff\x9a\xd7\xb1\x86\xdd`\x00\x00\x00\x0
0 :\xff\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xf
f\x02\x00\x00\x00\x00\x00\x00\x00\x00\x01\xff\x9a\xd7\xb1\x87\x00~f\x0
0\x00\x00\x00\xfe\x80\x00\x00\x00\x00\x00\x00\x18j\xff\xff\xfe\x9a\xd7\xb1
\x0e\x01\x1c\xddV\xff\xb4\xd8',
    match=OFPMatch(oxm_fields={'in_port': 2}),
    reason=0,
    table_id=0,
    total_len=86)
```

Python | 复制代码

```
['_TYPE', '__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__
eq__', '__format__', '__ge__', '__getattribute__', '__gt__', '__hash__', '_
_init__', '__init_subclass__', '__le__', '__lt__', '__module__', '__ne__',
'__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__siz
eof__', '__str__', '__subclasshook__', '__weakref__', '_base_attributes',
'_class_prefixes', '_class_suffixes', '_decode_value', '_encode_value', '_g
et_decoder', '_get_default_decoder', '_get_default_encoder', '_get_encoder'
, '_get_type', '_is_class', '_opt_attributes', '_reserved', '_restore_args'
, '_serialize_body', '_serialize_header', '_serialize_pre',

'auxiliary_id', 'buf', 'capabilities', 'cls_from_jsondict_key', 'cls_msg_ty
pe', 'datapath', 'datapath_id', 'from_jsondict', 'msg_len', 'msg_type', 'n_
buffers', 'n_tables', 'obj_from_jsondict', 'parser', 'serialize', 'set_buf'
, 'set_classes', 'set_headers', 'set_xid', 'stringify_attrs', 'to_jsondict'
, 'version', 'xid']
```

Python | 复制代码

```
msg所提供的方法和属性

'auxiliary_id', 'buf', 'capabilities', 'cls_from_jsondict_key', 'cls_msg_type',

'datapath', 'datapath_id',

'from_jsondict', 'msg_len', 'msg_type', 'n_buffers', 'n_tables',

'obj_from_jsondict', 'parser', 'serialize',

'set_buf', 'set_classes', 'set_headers', 'set_xid',

'stringify_attrs', 'to_jsondict', 'version', 'xid']
```

3.获取逻辑设备datapath类信息

Python | 复制代码

```
<ryu.controller.controller.Datapath object at 0x7fe1560c6048>
```

Python | 复制代码

```
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__gt__', '__hash__', '__init__',
```

4.获取逻辑设备datapath中OpenFlow协议信息

Python | 复制代码

```
<module 'ryu.ofproto.ofproto_v1_3' from '/usr/local/lib/python3.6/dist-packages/ryu/ofproto/ofproto_v1_3.py'>
```

Python | 复制代码

```
['DESC_STR_LEN', 'DESC_STR_LEN_STR', 'MAX_XID', 'OFPAT_COPY_TTL_IN', 'OFPAT_COPY_TTL_OUT', 'OFPAT_DEC_MPLS_TTL', 'OFPAT_DEC_NW_TTL', 'OFPAT_EXPERIMENTER', 'OFPAT_GROUP', 'OFPAT_OUTPUT', 'OFPAT_POP_MPLS', 'OFPAT_POP_PBB', 'OFPAT_POP_VLAN', 'OFPAT_PUSH_MPLS', 'OFPAT_PUSH_PBB', 'OFPAT_PUSH_VLAN', 'OFPAT_SET_FIELD', 'OFPAT_SET_MPLS_TTL', 'OFPAT_SET_NW_TTL', 'OFPAT_SET_QUEUE', 'OFPBAC_BAD_ARGUMENT', 'OFPBAC_BAD_EXPERIMENTER', 'OFPBAC_BAD_EXP_TYPE', 'OFPBAC_BAD_LEN', 'OFPBAC_BAD_OUT_GROUP', 'OFPBAC_BAD_OUT_PORT', 'OFPBAC_BAD_QUEUE', 'OFPBAC_BAD_SET_ARGUMENT', 'OFPBAC_BAD_SET_LEN', 'OFPBAC_BAD_SET_TYPE', 'OFPBAC_BAD_TAG', 'OFPBAC_BAD_TYPE', 'OFPBAC_EPERM', 'OFPBAC_MATCH_INCONSISTENT', 'OFPBAC_TOO_MANY', 'OFPBAC_UNSUPPORTED_ORDER', 'OFPBIC_BAD_EXPERIMENTER', 'OFPBIC_BAD_EXP_TYPE', 'OFPBIC_BAD_LEN', 'OFPBIC_BAD_TABLE_ID', 'OFPBIC_EPERM', 'OFPBIC_UNKNOWN_INST', 'OFPBIC_UNSUP_INST', 'OFPBIC_UNSUP_METADATA', 'OFPBIC_UNSUP_METADATA_MASK', 'OFPBMC_BAD_DL_ADDR_MASK', 'OFPBMC_BAD_FIELD', 'OFPBMC_BAD_LEN', 'OFPBMC_BAD_MASK', 'OFPBMC_BAD_NW_ADDR_MASK', 'OFPBMC_BAD_PREREQ', 'OFPBMC_BAD_TAG', 'OFPBMC_BAD_TYPE', 'OFPBMC_BAD_VALUE', 'OFPBMC_BAD_WILDCARDS', 'OFPBMC_DUP_FIELD', 'OFPBMC_EPERM', 'OFPBRC_BAD_EXPERIMENTER', 'OFPBRC_BAD_EXP_TYPE', 'OFPBRC_BAD_LEN', 'OFPBRC_BAD_MULTIPART', 'OFPBRC_BAD_PACKET', 'OFPBRC_BAD_PORT', 'OFPBRC_BAD_TABLE_ID', 'OFPBRC_BAD_TYPE', 'OFPBRC_BAD_VERSION', 'OFPBRC_BUFFER_EMPTY', 'OFPBRC_BUFFER_UNKNOWN', 'OFPBRC_EPERM', 'OFPBRC_IS_SLAVE', 'OFPBRC_MULTIPART_BUFFER_OVERFLOW', 'OFPCML_MAX', 'OFPCML_NO_BUFFER', 'OFPCR_ROLE_EQUAL', 'OFPCR_ROLE_MASTER', 'OFPCR_ROLE_NOCHANGE', 'OFPCR_ROLE_SLAVE', 'OFPC_FLOW_STATS', 'OFPC_FRAG_DROP', 'OFPC_FRAG_MASK', 'OFPC_FRAG_NORMAL', 'OFPC_FRAG_REASM', 'OFPC_GROUP_STATS', 'OFPC_IP_REASM', 'OFPC_PORT_BLOCKED', 'OFPC_PORT_STATS', 'OFPC_QUEUE_STATS', 'OFPC_TABLE_STATS', 'OFPET_BAD_ACTION', 'OFPET_BAD_INSTRUCTION', 'OFPET_BAD_MATCH', 'OFPET_BAD_REQUEST', 'OFPET_EXPERIMENTER', 'OFPET_FLOW_MOD_FAILED', 'OFPET_GROUP_MOD_FAILED', 'OFPET_HELLO_FAILED', 'OFPET_METER_MOD_FAILED', 'OFPET_PORT_MOD_FAILED', 'OFPET_QUEUE_OP_FAILED', 'OFPET_ROLE_REQUEST_FAILED', 'OFPET_SWITCH_CONFIG_FAILED', 'OFPET_TABLE_FEATURES_FAILED', 'OFPET_TABLE_MOD_FAILED', 'OFPFC_ADD', 'OFPFC_DELETE', 'OFPFC_DELETE_STRICT', 'OFPFC_MODIFY', 'OFPFC_MODIFY_STRICT', 'OFPFF_CHECK_OVERLAP', 'OFPFF_NO_BYT_COUNTS', 'OFPFF_NO_PKT_COUNTS', 'OFPFF_RESET_COUNTS', 'OFPFF_SEND_FLOW_REM', 'OFPFMFC_BAD_COMMAND', 'OFPFMFC_BAD_FLAGS', 'OFPFMFC_BAD_TABLE_ID', 'OFPFMFC_BAD_TIMEOUT', 'OFPFMFC_EPERM', 'OFPFMFC_OVERLAP', 'OFPFMFC_TABLE_FULL', 'OFPFMFC_UNKNOWN', 'OFPGC_ADD', 'OFPGC_DELETE', 'OFPGC_MODIFY', 'OFPGFC_CHAINING', 'OFPGFC_CHAINING_CHECKS', 'OFPGFC_SELECT_LIVENESS', 'OFPGFC_SELECT_WEIGHT', 'OFPGMFC_BAD_BUCKET', 'OFPGMFC_BAD_COMMAND', 'OFPGMFC_BAD_TYPE', 'OFPGMFC_BAD_WATCH', 'OFPGMFC_CHAINED_GROUP', 'OFPGMFC_CHAINING_UNSUPPORTED', 'OFPGMFC_EPERM', 'OFPGMFC_GROUP_EXISTS', 'OFPGMFC_INVALID_GROUP', 'OFPGMFC_LOOP', 'OFPGMFC_OUT_OF_BUCKETS', 'OFPGMFC_OUT_OF_GROUPS', 'OFPGMFC_UNKNOWN_GROUP', 'OFPGMFC_WATCH_UNSUPPORTED', 'OFPGMFC_WEIGHT_UNSUPPORTED', 'OFPGT_ALL', 'OFPGT_FF', 'OFPGT_INDIRECT', 'OFPGT_SELECT', 'OFPG_ALL', 'OFPG_ANY', 'OFPG_MAX', 'OFPHET_VERSIONBITMAP', 'OFPHFC_EPERM', 'OFPHFC_INCOMPATIBLE', 'OFPIEH_AUTH', 'OFPIEH_DEST', 'OFPIEH_ESP', 'OFPIEH_FRAG', 'OFPIEH_HOP', 'OFPIEH_NONEXT', 'OFPIEH_ROUTER', 'OFPIEH_UNREP', 'OFPIEH_UNSEQ'
```

```
, 'OFPIT_APPLY_ACTIONS', 'OFPIT_CLEAR_ACTIONS', 'OFPIT_EXPERIMENTER', 'OFPI
T_GOTO_TABLE', 'OFPIT_METER', 'OFPIT_WRITE_ACTIONS', 'OFPIT_WRITE_METADATA'
, 'OFPMBT_DROP', 'OFPMBT_DSCP_REMARK', 'OFPMBT_EXPERIMENTER', 'OFPMC_ADD',
'OFPMC_DELETE', 'OFPMC_MODIFY', 'OFPMF_BURST', 'OFPMF_KBPS', 'OFPMF_PKTPS',
 'OFPMF_STATS', 'OFPMMFC_BAD_BAND', 'OFPMMFC_BAD_BAND_VALUE', 'OFPMMFC_BAD_
BURST', 'OFPMMFC_BAD_COMMAND', 'OFPMMFC_BAD_FLAGS', 'OFPMMFC_BAD_RATE', 'OF
PMMFC_INVALID_METER', 'OFPMMFC_METER_EXISTS', 'OFPMMFC_OUT_OF_BANDS', 'OFPM
MFC_OUT_OF_METERS', 'OFPMMFC_UNKNOWN', 'OFPMMFC_UNKNOWN_METER', 'OFPMPF_REP
LY_MORE', 'OFPMPF_REQ_MORE', 'OFPMP_AGGREGATE', 'OFPMP_DESC', 'OFPMP_EXPERI
MENTER', 'OFPMP_FLOW', 'OFPMP_GROUP', 'OFPMP_GROUP_DESC', 'OFPMP_GROUP_FEAT
URES', 'OFPMP_METER', 'OFPMP_METER_CONFIG', 'OFPMP_METER_FEATURES', 'OFPMP_
PORT_DESC', 'OFPMP_PORT_STATS', 'OFPMP_QUEUE', 'OFPMP_TABLE', 'OFPMP_TABLE_
FEATURES', 'OFPMT_OXM', 'OFPMT_STANDARD', 'OFPM_ALL', 'OFPM_CONTROLLER', 'O
FPM_MAX', 'OFPM_SLOWPATH', 'OFPPC_NO_FWD', 'OFPPC_NO_PACKET_IN', 'OFPPC_NO_
RECV', 'OFPPC_PORT_DOWN', 'OFPPF_100GB_FD', 'OFPPF_100MB_FD', 'OFPPF_100MB_
HD', 'OFPPF_10GB_FD', 'OFPPF_10MB_FD', 'OFPPF_10MB_HD', 'OFPPF_1GB_FD', 'OF
PPF_1GB_HD', 'OFPPF_1TB_FD', 'OFPPF_40GB_FD', 'OFPPF_AUTONEG', 'OFPPF_COPPE
R', 'OFPPF_FIBER', 'OFPPF_OTHER', 'OFPPF_PAUSE', 'OFPPF_PAUSE_ASYM', 'OFPPM
FC_BAD_ADVERTISE', 'OFPPMFC_BAD_CONFIG', 'OFPPMFC_BAD_HW_ADDR', 'OFPPMFC_BA
D_PORT', 'OFPPMFC_EPERM', 'OFPPR_ADD', 'OFPPR_DELETE', 'OFPPR_MODIFY', 'OFP
PS_BLOCKED', 'OFPPS_LINK_DOWN', 'OFPPS_LIVE', 'OFPP_ALL', 'OFPP_ANY', 'OFPP
_CONTROLLER', 'OFPP_FLOOD', 'OFPP_IN_PORT', 'OFPP_LOCAL', 'OFPP_MAX', 'OFPP
_NORMAL', 'OFPP_TABLE', 'OFPQCFC_EPERM', 'OFPQOFC_BAD_PORT', 'OFPQOFC_BAD_Q
UEUE', 'OFPQOFC_EPERM', 'OFPQT_EXPERIMENTER', 'OFPQT_MAX_RATE', 'OFPQT_MIN_
RATE', 'OFPQ_ALL', 'OFPRRFC_BAD_ROLE', 'OFPRRFC_STALE', 'OFPRRFC_UNSUP', 'O
FPRR_DELETE', 'OFPRR_GROUP_DELETE', 'OFPRR_HARD_TIMEOUT', 'OFPRR_IDLE_TIMEO
UT', 'OFPR_ACTION', 'OFPR_INVALID_TTL', 'OFPR_NO_MATCH', 'OFPSCFC_BAD_FLAG
S', 'OFPSCFC_BAD_LEN', 'OFPSCFC_EPERM', 'OFPTFFC_BAD_ARGUMENT', 'OFPTFFC_BA
D_LEN', 'OFPTFFC_BAD_METADATA', 'OFPTFFC_BAD_TABLE', 'OFPTFFC_BAD_TYPE', 'O
FPTFFC_EPERM', 'OFPTFPT_APPLY_ACTIONS', 'OFPTFPT_APPLY_ACTIONS_MISS', 'OFPT
FPT_APPLY_SETFIELD', 'OFPTFPT_APPLY_SETFIELD_MISS', 'OFPTFPT_EXPERIMENTER',
 'OFPTFPT_EXPERIMENTER_MISS', 'OFPTFPT_INSTRUCTIONS', 'OFPTFPT_INSTRUCTIONS
_MISS', 'OFPTFPT_MATCH', 'OFPTFPT_NEXT_TABLES', 'OFPTFPT_NEXT_TABLES_MISS',
 'OFPTFPT_WILDCARDS', 'OFPTFPT_WRITE_ACTIONS', 'OFPTFPT_WRITE_ACTIONS_MISS'
, 'OFPTFPT_WRITE_SETFIELD', 'OFPTFPT_WRITE_SETFIELD_MISS', 'OFPTMFC_BAD_CON
FIG', 'OFPTMFC_BAD_TABLE', 'OFPTMFC_EPERM', 'OFPTT_ALL', 'OFPTT_MAX', 'OFPT
_BARRIER_REPLY', 'OFPT_BARRIER_REQUEST', 'OFPT_ECHO_REPLY', 'OFPT_ECHO_REQU
EST', 'OFPT_ERROR', 'OFPT_EXPERIMENTER', 'OFPT_FEATURES_REPLY', 'OFPT_FEATU
RES_REQUEST', 'OFPT_FLOW_MOD', 'OFPT_FLOW_REMOVED', 'OFPT_GET_ASYNC_REPLY',
 'OFPT_GET_ASYNC_REQUEST', 'OFPT_GET_CONFIG_REPLY', 'OFPT_GET_CONFIG_REQUES
T', 'OFPT_GROUP_MOD', 'OFPT_HELLO', 'OFPT_METER_MOD', 'OFPT_MULTIPART_REPL
Y', 'OFPT_MULTIPART_REQUEST', 'OFPT_PACKET_IN', 'OFPT_PACKET_OUT', 'OFPT_PO
RT_MOD', 'OFPT_PORT_STATUS', 'OFPT_QUEUE_GET_CONFIG_REPLY', 'OFPT_QUEUE_GET
_CONFIG_REQUEST', 'OFPT_ROLE_REPLY', 'OFPT_ROLE_REQUEST', 'OFPT_SET_ASYNC',
 'OFPT_SET_CONFIG', 'OFPT_TABLE_MOD', 'OFPVID_NONE', 'OFPVID_PRESENT', 'OFP
XMC_EXPERIMENTER', 'OFPXMC_NXM_0', 'OFPXMC_NXM_1', 'OFPXMC_OPENFLOW_BASIC',
 'OFPXMT_OFB_ARP_OP', 'OFPXMT_OFB_ARP_SHA', 'OFPXMT_OFB_ARP_SPA', 'OFPXMT_O
FB_ARP_THA', 'OFPXMT_OFB_ARP_TPA', 'OFPXMT_OFB_ETH_DST', 'OFPXMT_OFB_ETH_SR
```

```
C', 'OFPXMT_OFB_ETH_TYPE', 'OFPXMT_OFB_ICMPV4_CODE', 'OFPXMT_OFB_ICMPV4_TYP
E', 'OFPXMT_OFB_ICMPV6_CODE', 'OFPXMT_OFB_ICMPV6_TYPE', 'OFPXMT_OFB_IN_PHY_
PORT', 'OFPXMT_OFB_IN_PORT', 'OFPXMT_OFB_IPV4_DST', 'OFPXMT_OFB_IPV4_SRC',
'OFPXMT_OFB_IPV6_DST', 'OFPXMT_OFB_IPV6_EXTHDR', 'OFPXMT_OFB_IPV6_FLABEL',
'OFPXMT_OFB_IPV6_ND_SLL', 'OFPXMT_OFB_IPV6_ND_TARGET', 'OFPXMT_OFB_IPV6_ND_
TLL', 'OFPXMT_OFB_IPV6_SRC', 'OFPXMT_OFB_IP_DSCP', 'OFPXMT_OFB_IP_ECN', 'OF
PXMT_OFB_IP_PROTO', 'OFPXMT_OFB_METADATA', 'OFPXMT_OFB_MPLS_BOS', 'OFPXMT_O
FB_MPLS_LABEL', 'OFPXMT_OFB_MPLS_TC', 'OFPXMT_OFB_PBB_ISID', 'OFPXMT_OFB_SC
TP_DST', 'OFPXMT_OFB_SCTP_SRC', 'OFPXMT_OFB_TCP_DST', 'OFPXMT_OFB_TCP_SRC',
 'OFPXMT_OFB_TUNNEL_ID', 'OFPXMT_OFB_UDP_DST', 'OFPXMT_OFB_UDP_SRC', 'OFPXM
T_OFB_VLAN_PCP', 'OFPXMT_OFB_VLAN_VID', 'OFP_ACTION_EXPERIMENTER_HEADER_PAC
K_STR', 'OFP_ACTION_EXPERIMENTER_HEADER_SIZE', 'OFP_ACTION_GROUP_PACK_STR',
 'OFP_ACTION_GROUP_SIZE', 'OFP_ACTION_HEADER_PACK_STR', 'OFP_ACTION_HEADER_
SIZE', 'OFP_ACTION_MPLS_TTL_PACK_STR', 'OFP_ACTION_MPLS_TTL_SIZE', 'OFP_ACT
ION_NW_TTL_PACK_STR', 'OFP_ACTION_NW_TTL_SIZE', 'OFP_ACTION_OUTPUT_PACK_ST
R', 'OFP_ACTION_OUTPUT_SIZE', 'OFP_ACTION_POP_MPLS_PACK_STR', 'OFP_ACTION_P
OP_MPLS_SIZE', 'OFP_ACTION_PUSH_PACK_STR', 'OFP_ACTION_PUSH_SIZE', 'OFP_ACT
ION_SET_FIELD_PACK_STR', 'OFP_ACTION_SET_FIELD_SIZE', 'OFP_ACTION_SET_QUEUE
_PACK_STR', 'OFP_ACTION_SET_QUEUE_SIZE', 'OFP_AGGREGATE_STATS_REPLY_PACK_ST
R', 'OFP_AGGREGATE_STATS_REPLY_SIZE', 'OFP_AGGREGATE_STATS_REQUEST_0_PACK_S
TR', 'OFP_AGGREGATE_STATS_REQUEST_0_SIZE', 'OFP_AGGREGATE_STATS_REQUEST_PAC
K_STR', 'OFP_AGGREGATE_STATS_REQUEST_SIZE', 'OFP_ASYNC_CONFIG_PACK_STR', 'O
FP_ASYNC_CONFIG_SIZE', 'OFP_BUCKET_COUNTER_PACK_STR', 'OFP_BUCKET_COUNTER_S
IZE', 'OFP_BUCKET_PACK_STR', 'OFP_BUCKET_SIZE', 'OFP_DEFAULT_PRIORITY', 'OF
P_DESC_PACK_STR', 'OFP_DESC_SIZE', 'OFP_ERROR_EXPERIMENTER_MSG_PACK_STR',
'OFP_ERROR_EXPERIMENTER_MSG_SIZE', 'OFP_ERROR_MSG_PACK_STR', 'OFP_ERROR_MSG
_SIZE', 'OFP_ETH_ALEN', 'OFP_ETH_ALEN_STR', 'OFP_EXPERIMENTER_HEADER_PACK_S
TR', 'OFP_EXPERIMENTER_HEADER_SIZE', 'OFP_EXPERIMENTER_MULTIPART_HEADER_PAC
K_STR', 'OFP_EXPERIMENTER_MULTIPART_HEADER_SIZE', 'OFP_FLOW_MOD_PACK_STR',
'OFP_FLOW_MOD_PACK_STR0', 'OFP_FLOW_MOD_SIZE', 'OFP_FLOW_REMOVED_PACK_STR',
 'OFP_FLOW_REMOVED_PACK_STR0', 'OFP_FLOW_REMOVED_SIZE', 'OFP_FLOW_STATS_0_P
ACK_STR', 'OFP_FLOW_STATS_0_SIZE', 'OFP_FLOW_STATS_PACK_STR', 'OFP_FLOW_STA
TS_REQUEST_0_PACK_STR', 'OFP_FLOW_STATS_REQUEST_0_SIZE', 'OFP_FLOW_STATS_RE
QUEST_PACK_STR', 'OFP_FLOW_STATS_REQUEST_SIZE', 'OFP_FLOW_STATS_SIZE', 'OFP
_GROUP_DESC_PACK_STR', 'OFP_GROUP_DESC_SIZE', 'OFP_GROUP_DESC_STATS_PACK_ST
R', 'OFP_GROUP_DESC_STATS_SIZE', 'OFP_GROUP_FEATURES_PACK_STR', 'OFP_GROUP_
FEATURES_SIZE', 'OFP_GROUP_MOD_PACK_STR', 'OFP_GROUP_MOD_SIZE', 'OFP_GROUP_
STATS_PACK_STR', 'OFP_GROUP_STATS_REQUEST_PACK_STR', 'OFP_GROUP_STATS_REQUE
ST_SIZE', 'OFP_GROUP_STATS_SIZE', 'OFP_HEADER_PACK_STR', 'OFP_HEADER_SIZE',
 'OFP_HELLO_ELEM_HEADER_PACK_STR', 'OFP_HELLO_ELEM_HEADER_SIZE', 'OFP_HELLO
_ELEM_VERSIONBITMAP_HEADER_PACK_STR', 'OFP_HELLO_ELEM_VERSIONBITMAP_HEADER_
SIZE', 'OFP_HELLO_HEADER_SIZE', 'OFP_INSTRUCTION_ACTIONS_PACK_STR', 'OFP_IN
STRUCTION_ACTIONS_SIZE', 'OFP_INSTRUCTION_GOTO_TABLE_PACK_STR', 'OFP_INSTRU
CTION_GOTO_TABLE_SIZE', 'OFP_INSTRUCTION_METER_PACK_STR', 'OFP_INSTRUCTION_
METER_SIZE', 'OFP_INSTRUCTION_WRITE_METADATA_PACK_STR', 'OFP_INSTRUCTION_WR
ITE_METADATA_SIZE', 'OFP_MATCH_PACK_STR', 'OFP_MATCH_SIZE', 'OFP_MAX_PORT_N
AME_LEN', 'OFP_MAX_TABLE_NAME_LEN', 'OFP_MAX_TABLE_NAME_LEN_STR', 'OFP_METE
R_BAND_DROP_PACK_STR', 'OFP_METER_BAND_DROP_SIZE', 'OFP_METER_BAND_DSCP_REM
```

```
ARK_PACK_STR', 'OFP_METER_BAND_DSCP_REMARK_SIZE', 'OFP_METER_BAND_EXPERIMEN
TER_PACK_STR', 'OFP_METER_BAND_EXPERIMENTER_SIZE', 'OFP_METER_BAND_HEADER_P
ACK_STR', 'OFP_METER_BAND_HEADER_SIZE', 'OFP_METER_BAND_STATS_PACK_STR', 'O
FP_METER_BAND_STATS_SIZE', 'OFP_METER_CONFIG_PACK_STR', 'OFP_METER_CONFIG_S
IZE', 'OFP_METER_FEATURES_PACK_STR', 'OFP_METER_FEATURES_SIZE', 'OFP_METER_
MOD_PACK_STR', 'OFP_METER_MOD_SIZE', 'OFP_METER_MULTIPART_REQUEST_PACK_STR'
, 'OFP_METER_MULTIPART_REQUEST_SIZE', 'OFP_METER_STATS_PACK_STR', 'OFP_METE
R_STATS_SIZE', 'OFP_MULTIPART_REPLY_PACK_STR', 'OFP_MULTIPART_REPLY_SIZE',
 'OFP_MULTIPART_REQUEST_PACK_STR', 'OFP_MULTIPART_REQUEST_SIZE', 'OFP_NO_BUF
FER', 'OFP_OXM_EXPERIMENTER_HEADER_PACK_STR', 'OFP_OXM_EXPERIMENTER_HEADER_
SIZE', 'OFP_PACKET_IN_PACK_STR', 'OFP_PACKET_IN_SIZE', 'OFP_PACKET_OUT_PACK
_STR', 'OFP_PACKET_OUT_SIZE', 'OFP_PACKET_QUEUE_PACK_STR', 'OFP_PACKET_QUEU
E_SIZE', 'OFP_PORT_MOD_PACK_STR', 'OFP_PORT_MOD_SIZE', 'OFP_PORT_PACK_STR',
 'OFP_PORT_SIZE', 'OFP_PORT_STATS_PACK_STR', 'OFP_PORT_STATS_REQUEST_PACK_S
TR', 'OFP_PORT_STATS_REQUEST_SIZE', 'OFP_PORT_STATS_SIZE', 'OFP_PORT_STATUS
_DESC_OFFSET', 'OFP_PORT_STATUS_PACK_STR', 'OFP_PORT_STATUS_SIZE', 'OFP_PRO
P_EXPERIMENTER_PACK_STR', 'OFP_PROP_EXPERIMENTER_SIZE', 'OFP_QUEUE_GET_CONF
IG_REPLY_PACK_STR', 'OFP_QUEUE_GET_CONFIG_REPLY_SIZE', 'OFP_QUEUE_GET_CONFI
G_REQUEST_PACK_STR', 'OFP_QUEUE_GET_CONFIG_REQUEST_SIZE', 'OFP_QUEUE_PROP_E
XPERIMENTER_PACK_STR', 'OFP_QUEUE_PROP_EXPERIMENTER_SIZE', 'OFP_QUEUE_PROP_
HEADER_PACK_STR', 'OFP_QUEUE_PROP_HEADER_SIZE', 'OFP_QUEUE_PROP_MAX_RATE_PA
CK_STR', 'OFP_QUEUE_PROP_MAX_RATE_SIZE', 'OFP_QUEUE_PROP_MIN_RATE_PACK_STR'
, 'OFP_QUEUE_PROP_MIN_RATE_SIZE', 'OFP_QUEUE_STATS_PACK_STR', 'OFP_QUEUE_ST
ATS_REQUEST_PACK_STR', 'OFP_QUEUE_STATS_REQUEST_SIZE', 'OFP_QUEUE_STATS_SIZ
E', 'OFP_ROLE_REQUEST_PACK_STR', 'OFP_ROLE_REQUEST_SIZE', 'OFP_SWITCH_CONFI
G_PACK_STR', 'OFP_SWITCH_CONFIG_SIZE', 'OFP_SWITCH_FEATURES_PACK_STR', 'OFP
_SWITCH_FEATURES_SIZE', 'OFP_TABLE_FEATURES_PACK_STR', 'OFP_TABLE_FEATURES_
SIZE', 'OFP_TABLE_FEATURE_PROP_ACTIONS_PACK_STR', 'OFP_TABLE_FEATURE_PROP_A
CTIONS_SIZE', 'OFP_TABLE_FEATURE_PROP_INSTRUCTIONS_PACK_STR', 'OFP_TABLE_FE
ATURE_PROP_INSTRUCTIONS_SIZE', 'OFP_TABLE_FEATURE_PROP_NEXT_TABLES_PACK_ST
R', 'OFP_TABLE_FEATURE_PROP_NEXT_TABLES_SIZE', 'OFP_TABLE_FEATURE_PROP_OXM_
PACK_STR', 'OFP_TABLE_FEATURE_PROP_OXM_SIZE', 'OFP_TABLE_MOD_PACK_STR', 'OF
P_TABLE_MOD_SIZE', 'OFP_TABLE_STATS_PACK_STR', 'OFP_TABLE_STATS_SIZE', 'OFP
_TCP_PORT', 'OFP_VERSION', 'ONFERR_DUP_INSTRUCTION', 'ONFERR_ET_ASYNC_EPER
M', 'ONFERR_ET_ASYNC_INVALUD', 'ONFERR_ET_ASYNC_UNSUPPORTED', 'ONFERR_ET_BA
D_FLAGS', 'ONFERR_ET_BAD_ID', 'ONFERR_ET_BAD_PRIORITY', 'ONFERR_ET_BAD_TYP
E', 'ONFERR_ET_BUNDLE_CLOSED', 'ONFERR_ET_BUNDLE_EXIST', 'ONFERR_ET_BUNDLE_
IN_PROGRESS', 'ONFERR_ET_CANT_SYNC', 'ONFERR_ET_EPERM', 'ONFERR_ET_FAILED',
 'ONFERR_ET_MPART_REPLY_TIMEOUT', 'ONFERR_ET_MPART_REQUEST_TIMEOUT', 'ONFER
R_ET_MSG_BAD_LEN', 'ONFERR_ET_MSG_BAD_XID', 'ONFERR_ET_MSG_CONFLICT', 'ONFE
RR_ET_MSG_TOO_MANY', 'ONFERR_ET_MSG_UNSUP', 'ONFERR_ET_OUT_OF_BUNDLES', 'ON
FERR_ET_TIMEOUT', 'ONFERR_ET_UNKNOWN', 'ONFFME_ABBREV', 'ONFFME_ADDED', 'ON
FFME_DELETED', 'ONFFME_MODIFIED', 'ONFFMF_ACTIONS', 'ONFFMF_ADD', 'ONFFMF_D
ELETE', 'ONFFMF_INITIAL', 'ONFFMF_MODIFY', 'ONFFMF_OWN', 'ONFMP_FLOW_MONITO
R', 'ONFT_FLOW_MONITOR_CANCEL', 'ONFT_FLOW_MONITOR_PAUSED', 'ONFT_FLOW_MONI
TOR_RESUMED', 'ONF_BCT_CLOSE_REPLY', 'ONF_BCT_CLOSE_REQUEST', 'ONF_BCT_COMM
IT_REPLY', 'ONF_BCT_COMMIT_REQUEST', 'ONF_BCT_DISCARD_REPLY', 'ONF_BCT_DISC
ARD_REQUEST', 'ONF_BCT_OPEN_REPLY', 'ONF_BCT_OPEN_REQUEST', 'ONF_BF_ATOMIC'
```

```
, 'ONF_BF_ORDERED', 'ONF_BUNDLE_ADD_MSG_PACK_STR', 'ONF_BUNDLE_ADD_MSG_SIZ
E', 'ONF_BUNDLE_CTRL_PACK_STR', 'ONF_BUNDLE_CTRL_SIZE', 'ONF_ET_BPT_EXPERIM
ENTER', 'ONF_ET_BUNDLE_ADD_MESSAGE', 'ONF_ET_BUNDLE_CONTROL', 'ONF_FLOW_MON
ITOR_REQUEST_PACK_STR', 'ONF_FLOW_MONITOR_REQUEST_SIZE', 'ONF_FLOW_UPDATE_A
BBREV_PACK_STR', 'ONF_FLOW_UPDATE_ABBREV_SIZE', 'ONF_FLOW_UPDATE_FULL_PACK_
STR', 'ONF_FLOW_UPDATE_FULL_SIZE', 'ONF_FLOW_UPDATE_HEADER_PACK_STR', 'ONF_
FLOW_UPDATE_HEADER_SIZE', 'OXM_OF_ARP_OP', 'OXM_OF_ARP_OP_W', 'OXM_OF_ARP_S
```

5.获取逻辑设备datapath中OpenFlow协议解析信息

Python | 复制代码

```
1  <module 'ryu.ofproto.ofproto_v1_3_parser' from '/usr/local/lib/python3.6/di
   st-packages/ryu/ofproto/ofproto_v1_3_parser.py'>
```

Python | 复制代码

```
['Flow', 'FlowWildcards', 'LOG', 'MTArpOp', 'MTArpSha', 'MTArpSpa', 'MTArpTha', 'MTArpTpa', 'MTEthDst', 'MTEthSrc', 'MTEthType', 'MTICMPV4Code', 'MTICMPV4Type', 'MTICMPV6Code', 'MTICMPV6Type', 'MTIPDscp', 'MTIPECN', 'MTIPProto', 'MTIPV4Dst', 'MTIPV4Src', 'MTIPv6', 'MTIPv6Dst', 'MTIPv6ExtHdr', 'MTIPv6Flabel', 'MTIPv6NdSll', 'MTIPv6NdTarget', 'MTIPv6NdTll', 'MTIPv6Src', 'MTInPhyPort', 'MTInPort', 'MTMetadata', 'MTMplsBos', 'MTMplsLabel', 'MTMplsTc', 'MTPbbIsid', 'MTSCTPDst', 'MTSCTPSrc', 'MTTCPDst', 'MTTCPSrc', 'MTTunnelId', 'MTUDPDst', 'MTUDPSrc', 'MTVlanPcp', 'MTVlanVid', 'MsgBase', 'NXAction', 'NXActionBundle', 'NXActionBundleLoad', 'NXActionCT', 'NXActionCTClear', 'NXActionConjunction', 'NXActionController', 'NXActionController2', 'NXActionDecMplsTtl', 'NXActionDecNshTtl', 'NXActionDecTtl', 'NXActionDecTtlCntIds', 'NXActionEncapEther', 'NXActionEncapNsh', 'NXActionExit', 'NXActionFinTimeout', 'NXActionLearn', 'NXActionMultipath', 'NXActionNAT', 'NXActionNote', 'NXActionOutputReg', 'NXActionOutputReg2', 'NXActionOutputTrunc', 'NXActionPopMpls', 'NXActionPopQueue', 'NXActionPushMpls', 'NXActionRegLoad', 'NXActionRegLoad2', 'NXActionRegMove', 'NXActionResubmit', 'NXActionResubmitTable', 'NXActionSample', 'NXActionSample2', 'NXActionSetMplsLabel', 'NXActionSetMplsTc', 'NXActionSetMplsTtl', 'NXActionSetQueue', 'NXActionSetTunnel', 'NXActionSetTunnel64', 'NXActionStackPop', 'NXActionStackPush', 'NXActionUnknown', 'NXFlowSpecLoad', 'NXFlowSpecMatch', 'NXFlowSpecOutput', 'OFPAction', 'OFPActionCopyTtlIn', 'OFPActionCopyTtlOut', 'OFPActionDecMplsTtl', 'OFPActionDecNwTtl', 'OFPActionExperimenter', 'OFPActionExperimenterUnknown', 'OFPActionGroup', 'OFPActionHeader', 'OFPActionId', 'OFPActionOutput', 'OFPActionPopMpls', 'OFPActionPopPbb', 'OFPActionPopVlan', 'OFPActionPushMpls', 'OFPActionPushPbb', 'OFPActionPushVlan', 'OFPActionSetField', 'OFPActionSetMplsTtl', 'OFPActionSetNwTtl', 'OFPActionSetQueue', 'OFPAggregateStats', 'OFPAggregateStatsReply', 'OFPAggregateStatsRequest', 'OFPBarrierReply', 'OFPBarrierRequest', 'OFPBucket', 'OFPBucketCounter', 'OFPDescStats', 'OFPDescStatsReply', 'OFPDescStatsRequest', 'OFPEchoReply', 'OFPEchoRequest', 'OFPErrorExperimenterMsg', 'OFPErrorMsg', 'OFPExperimenter', 'OFPExperimenterMultipart', 'OFPExperimenterOxmId', 'OFPExperimenterStatsReply', 'OFPExperimenterStatsRequest', 'OFPExperimenterStatsRequestBase', 'OFPFeaturesRequest', 'OFPFlowMod', 'OFPFlowRemoved', 'OFPFlowStats', 'OFPFlowStatsReply', 'OFPFlowStatsRequest', 'OFPFlowStatsRequestBase', 'OFPGetAsyncReply', 'OFPGetAsyncRequest', 'OFPGetConfigReply', 'OFPGetConfigRequest', 'OFPGroupDescStats', 'OFPGroupDescStatsReply', 'OFPGroupDescStatsRequest', 'OFPGroupFeaturesStats', 'OFPGroupFeaturesStatsReply', 'OFPGroupFeaturesStatsRequest', 'OFPGroupMod', 'OFPGroupStats', 'OFPGroupStatsReply', 'OFPGroupStatsRequest', 'OFPHello', 'OFPHelloElemVersionBitmap', 'OFPInstruction', 'OFPInstructionActions', 'OFPInstructionGotoTable', 'OFPInstructionId', 'OFPInstructionMeter', 'OFPInstructionWriteMetadata', 'OFPMatch', 'OFPMatchField', 'OFPMeterBand', 'OFPMeterBandDrop', 'OFPMeterBandDscpRemark', 'OFPMeterBandExperimenter', 'OFPMeterBandHeader', 'OFPMeterBandStats', 'OFPMeterConfigStats', 'OFPMeterConfigStatsReply', 'OFPMeterConfigStatsRequest', 'OFPMeterFeaturesStats', 'OFPMeterFeaturesStatsReply', 'OFPMeterFeaturesStatsRequest', 'OFPMeterMod'
```

```
, 'OFPMeterStats', 'OFPMeterStatsReply', 'OFPMeterStatsRequest', 'OFPMultip
artReply', 'OFPMultipartRequest', 'OFPOxmId', 'OFPPacketIn', 'OFPPacketOut'
, 'OFPPacketQueue', 'OFPPort', 'OFPPortDescStatsReply', 'OFPPortDescStatsRe
quest', 'OFPPortMod', 'OFPPortStats', 'OFPPortStatsReply', 'OFPPortStatsReq
uest', 'OFPPortStatus', 'OFPPropBase', 'OFPPropCommonExperimenter4ByteData'
, 'OFPPropUnknown', 'OFPQueueGetConfigReply', 'OFPQueueGetConfigRequest',
'OFPQueueProp', 'OFPQueuePropExperimenter', 'OFPQueuePropHeader', 'OFPQueue
PropMaxRate', 'OFPQueuePropMinRate', 'OFPQueueStats', 'OFPQueueStatsReply',
 'OFPQueueStatsRequest', 'OFPRoleReply', 'OFPRoleRequest', 'OFPSetAsync',
'OFPSetConfig', 'OFPSwitchFeatures', 'OFPTableFeatureProp', 'OFPTableFeatur
ePropActions', 'OFPTableFeaturePropExperimenter', 'OFPTableFeaturePropInstr
uctions', 'OFPTableFeaturePropNextTables', 'OFPTableFeaturePropOxm', 'OFPTa
bleFeaturePropUnknown', 'OFPTableFeaturesStats', 'OFPTableFeaturesStatsRepl
y', 'OFPTableFeaturesStatsRequest', 'OFPTableMod', 'OFPTableStats', 'OFPTab
leStatsReply', 'OFPTableStatsRequest', 'ONFBundleAddMsg', 'ONFBundleCtrlMs
g', 'ONFFlowMonitorRequest', 'ONFFlowMonitorStatsRequest', 'StringifyMixin'
, 'UINT16_MAX', 'UINT32_MAX', 'UINT64_MAX', '_MSG_PARSERS', '_NXFlowSpec',
'__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name_
_', '__package__', '__spec__', '_register_exp_type', '_register_parser', '_
set_msg_type', '_set_stats_type', 'addrconv', 'base64', 'ether', 'exceptio
n', 'logging', 'mac', 'msg_pack_into', 'msg_parser', 'nx_actions', 'ofprot
o', 'ofproto_common', 'ofproto_parser', 'packet', 'six', 'struct', 'utils']
```

5.通过ofp_parser中获取match类匹配

Python | 复制代码

```
1  OFPMatch(oxm_fields={})
```

```python
['_TYPE', '__class__', '__contains__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__le__', '__lt__', '__module__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', '__weakref__', '_base_attributes', '_class_prefixes', '_class_suffixes', '_composed_with_old_api', '_decode_value', '_encode_value', '_fields2', '_flow', '_get_decoder', '_get_default_decoder', '_get_default_encoder', '_get_encoder', '_get_type', '_is_class', '_opt_attributes', '_restore_args', '_wc', 'append_field', 'cls_from_jsondict_key', 'fields', 'from_jsondict', 'get', 'items', 'iteritems', 'length', 'obj_from_jsondict', 'parser', 'parser_old', 'serialize', 'serialize_old', 'set_arp_opcode', 'set_arp_sha', 'set_arp_sha_masked', 'set_arp_spa', 'set_arp_spa_masked', 'set_arp_tha', 'set_arp_tha_masked', 'set_arp_tpa', 'set_arp_tpa_masked', 'set_classes', 'set_dl_dst', 'set_dl_dst_masked', 'set_dl_src', 'set_dl_src_masked', 'set_dl_type', 'set_icmpv4_code', 'set_icmpv4_type', 'set_icmpv6_code', 'set_icmpv6_type', 'set_in_phy_port', 'set_in_port', 'set_ip_dscp', 'set_ip_ecn', 'set_ip_proto', 'set_ipv4_dst', 'set_ipv4_dst_masked', 'set_ipv4_src', 'set_ipv4_src_masked', 'set_ipv6_dst', 'set_ipv6_dst_masked', 'set_ipv6_exthdr', 'set_ipv6_exthdr_masked', 'set_ipv6_flabel', 'set_ipv6_flabel_masked', 'set_ipv6_nd_sll', 'set_ipv6_nd_target', 'set_ipv6_nd_tll', 'set_ipv6_src', 'set_ipv6_src_masked', 'set_metadata', 'set_metadata_masked', 'set_mpls_bos', 'set_mpls_label', 'set_mpls_tc', 'set_pbb_isid', 'set_pbb_isid_masked', 'set_sctp_dst', 'set_sctp_src', 'set_tcp_dst', 'set_tcp_src', 'set_tunnel_id', 'set_tunnel_id_masked', 'set_udp_dst', 'set_udp_src', 'set_vlan_pcp', 'set_vlan_vid', 'set_vlan_vid_masked', 'set_vlan_vid_none', 'stringify_attrs', 'to_jsondict', 'type']
```

5.通过ofp_parser中获取actions类匹配

```python
[OFPActionOutput(len=16,max_len=65535,port=4294967293,type=0)]
```

6.match = ofp_parser.OFPMatch()中OFPMatch类协议支持的匹配方式

Python | 复制代码

```python
class OFPMatch(StringifyMixin):
    """
    Flow Match Structure

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

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

    ================ =============== ==================================
    Argument         Value           Description
    ================ =============== ==================================
    in_port          Integer 32bit   Switch input port
    in_phy_port      Integer 32bit   Switch physical input port
    metadata         Integer 64bit   Metadata passed between tables
    eth_dst          MAC address     Ethernet destination address
    eth_src          MAC address     Ethernet source address
    eth_type         Integer 16bit   Ethernet frame type
    vlan_vid         Integer 16bit   VLAN id
    vlan_pcp         Integer 8bit    VLAN priority
    ip_dscp          Integer 8bit    IP DSCP (6 bits in ToS field)
    ip_ecn           Integer 8bit    IP ECN (2 bits in ToS field)
    ip_proto         Integer 8bit    IP protocol
    ipv4_src         IPv4 address    IPv4 source address
    ipv4_dst         IPv4 address    IPv4 destination address
    tcp_src          Integer 16bit   TCP source port
    tcp_dst          Integer 16bit   TCP destination port
    udp_src          Integer 16bit   UDP source port
    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 Extensio
n)
    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 :ref:`ryu.ofproto.nx_match <nx_match_structures>
`.

    .. Note::

        For VLAN id match field, special values are defined in OpenFlow S
pec.

        1) Packets with and without a VLAN tag

            - Example::
```

```
91
92                 match = parser.OFPMatch()
93
94             - Packet Matching
95
96                 ===================== =====
97                 non-VLAN-tagged        MATCH
98                 VLAN-tagged(vlan_id=3) MATCH
99                 VLAN-tagged(vlan_id=5) MATCH
100                ===================== =====
101
102        2) Only packets without a VLAN tag
103
104            - Example::
105
106                match = parser.OFPMatch(vlan_vid=0x0000)
107
108            - Packet Matching
109
110                ===================== =====
111                non-VLAN-tagged        MATCH
112                VLAN-tagged(vlan_id=3)   x
113                VLAN-tagged(vlan_id=5)   x
114                ===================== =====
115
116        3) Only packets with a VLAN tag regardless of its value
117
118            - Example::
119
120                match = parser.OFPMatch(vlan_vid=(0x1000, 0x1000))
121
122            - Packet Matching
123
124                ===================== =====
125                non-VLAN-tagged          x
126                VLAN-tagged(vlan_id=3) MATCH
127                VLAN-tagged(vlan_id=5) MATCH
128                ===================== =====
129
130        4) Only packets with VLAN tag and VID equal
131
132            - Example::
133
134                match = parser.OFPMatch(vlan_vid=(0x1000 | 3))
135
136            - Packet Matching
137
138                ===================== =====
```

```
                non-VLAN-tagged          x
                VLAN-tagged(vlan_id=3) MATCH
                VLAN-tagged(vlan_id=5)   x
                ======================= =====
    """
```

7. out = ofp_parser.OFPPacketOut(datapath=datapath,buffer_id=msg.buffer_id,in_port=in_port,actions=actions,data=msg.data)

Python | 复制代码

```
    out = ofp_parser.OFPPacketOut(datapath=datapath,buffer_id=msg.buffer_id
,
                                  in_port=in_port,actions=actions
)

    datapath.send_msg(out);
```

控制器使用此消息通过交换机发送数据包（存在控制器的消息缓存，我们需要发送给交换机，让他进行发送）---注意我们不仅要传递buffer_id还要传递data才可以实现将数据返回

Python | 复制代码

```python
@_set_msg_type(ofproto.OFPT_PACKET_OUT)
class OFPPacketOut(MsgBase):
    """
    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)
    """

    def __init__(self, datapath, buffer_id=None, in_port=None, actions=None,
                 data=None, actions_len=None):
        assert in_port is not None

        super(OFPPacketOut, self).__init__(datapath)
        self.buffer_id = buffer_id
        self.in_port = in_port
        self.actions_len = 0
        self.actions = actions
        self.data = data

```

```
    def _serialize_body(self):
        self.actions_len = 0
        offset = ofproto.OFP_PACKET_OUT_SIZE
        for a in self.actions:
            a.serialize(self.buf, offset)
            offset += a.len
            self.actions_len += a.len

        if self.data is not None:
            assert self.buffer_id == 0xffffffff
            if isinstance(self.data, packet.Packet):
                self.data.serialize()
                self.buf += self.data.data
            else:
                self.buf += self.data

        msg_pack_into(ofproto.OFP_PACKET_OUT_PACK_STR,
                      self.buf, ofproto.OFP_HEADER_SIZE,
                      self.buffer_id, self.in_port, self.actions_len)

    @classmethod
    def from_jsondict(cls, dict_, decode_string=base64.b64decode,
                      **additional_args):
        if isinstance(dict_['data'], dict):
            data = dict_.pop('data')
            ins = super(OFPPacketOut, cls).from_jsondict(dict_,
                                                         decode_string,
                                                         **additional_args
)
            ins.data = packet.Packet.from_jsondict(data['Packet'])
            dict_['data'] = data
        else:
            ins = super(OFPPacketOut, cls).from_jsondict(dict_,
                                                         decode_string,
                                                         **additional_args
)

        return ins
```

8. inst = [ofp_parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS,*actions)]*

```python
    inst = [ofp_parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS,
                                             actions)]

    mod = ofp_parser.OFPFlowMod(datapath=datapath,priority=priority,
                                match=match,instructions=inst)
    datapath.send_msg(mod);
```

此指令写入/应用/清除操作。就是说通过这个类实现我们对声明的actions行为进行应用（应用这个动作）、还是清除（不需要这个动作了）、还是写入操作

```python
class OFPInstruction(StringifyMixin):
    _INSTRUCTION_TYPES = {}

    @staticmethod
    def register_instruction_type(types):
        def _register_instruction_type(cls):
            for type_ in types:
                OFPInstruction._INSTRUCTION_TYPES[type_] = cls
            return cls
        return _register_instruction_type

    @classmethod
    def parser(cls, buf, offset):
        (type_, len_) = struct.unpack_from('!HH', buf, offset)
        cls_ = cls._INSTRUCTION_TYPES.get(type_)
        return cls_.parser(buf, offset)
```

Python | 复制代码

```python
@OFPInstruction.register_instruction_type([ofproto.OFPIT_WRITE_ACTIONS,
                                           ofproto.OFPIT_APPLY_ACTIONS,
                                           ofproto.OFPIT_CLEAR_ACTIONS])
class OFPInstructionActions(OFPInstruction):
    """
    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__.
    """

    def __init__(self, type_, actions=None, len_=None):
        super(OFPInstructionActions, self).__init__()
        self.type = type_
        for a in actions:
            assert isinstance(a, OFPAction)
        self.actions = actions

    @classmethod
    def parser(cls, buf, offset):
        (type_, len_) = struct.unpack_from(
            ofproto.OFP_INSTRUCTION_ACTIONS_PACK_STR,
            buf, offset)

        offset += ofproto.OFP_INSTRUCTION_ACTIONS_SIZE
        actions = []
        actions_len = len_ - ofproto.OFP_INSTRUCTION_ACTIONS_SIZE
        exc = None
        try:
            while actions_len > 0:
```

```
                a = OFPAction.parser(buf, offset)
                actions.append(a)
                actions_len -= a.len
                offset += a.len
        except struct.error as e:
            exc = e

        inst = cls(type_, actions)
        inst.len = len_
        if exc is not None:
            raise exception.OFPTruncatedMessage(ofpmsg=inst,
                                                residue=buf[offset:],
                                                original_exception=exc)
        return inst

    def serialize(self, buf, offset):
        action_offset = offset + ofproto.OFP_INSTRUCTION_ACTIONS_SIZE
        if self.actions:
            for a in self.actions:
                a.serialize(buf, action_offset)
                action_offset += a.len

        self.len = action_offset - offset
        pad_len = utils.round_up(self.len, 8) - self.len
        msg_pack_into("%dx" % pad_len, buf, action_offset)
        self.len += pad_len

        msg_pack_into(ofproto.OFP_INSTRUCTION_ACTIONS_PACK_STR,
                      buf, offset, self.type, self.len)
```

9. mod = ofp_parser.OFPFlowMod(datapath=datapath,priority=priority,match=match,instructions=inst); 修改流表项消息 控制器发送此消息以修改流表。

Python | 复制代码

```python
@_register_parser
@_set_msg_type(ofproto.OFPT_FLOW_MOD)
class OFPFlowMod(MsgBase):
    """
    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::

        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:f
f:ff')
            actions = [ofp_parser.OFPActionOutput(ofp.OFPP_NORMAL, 0)]
            inst = [ofp_parser.OFPInstructionActions(ofp.OFPIT_APPLY_ACTI
ONS,
                                                     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)
    """

    def __init__(self, datapath, cookie=0, cookie_mask=0, table_id=0,
                 command=ofproto.OFPFC_ADD,
                 idle_timeout=0, hard_timeout=0,
                 priority=ofproto.OFP_DEFAULT_PRIORITY,
                 buffer_id=ofproto.OFP_NO_BUFFER,
                 out_port=0, out_group=0, flags=0,
                 match=None,
                 instructions=None):
        instructions = instructions if instructions else []
        super(OFPFlowMod, self).__init__(datapath)
        self.cookie = cookie
        self.cookie_mask = cookie_mask
        self.table_id = table_id
        self.command = command
        self.idle_timeout = idle_timeout
        self.hard_timeout = hard_timeout
        self.priority = priority
        self.buffer_id = buffer_id
        self.out_port = out_port
        self.out_group = out_group
```

```
88          self.flags = flags
89          if match is None:
90              match = OFPMatch()
91          assert isinstance(match, OFPMatch)
92          self.match = match
93          for i in instructions:
94              assert isinstance(i, OFPInstruction)
95          self.instructions = instructions
96
97      def _serialize_body(self):
98          msg_pack_into(ofproto.OFP_FLOW_MOD_PACK_STR0, self.buf,
99                        ofproto.OFP_HEADER_SIZE,
100                       self.cookie, self.cookie_mask, self.table_id,
101                       self.command, self.idle_timeout, self.hard_timeout,
102                       self.priority, self.buffer_id, self.out_port,
103                       self.out_group, self.flags)
104
105         offset = (ofproto.OFP_FLOW_MOD_SIZE -
106                   ofproto.OFP_MATCH_SIZE)
107
108         match_len = self.match.serialize(self.buf, offset)
109         offset += match_len
110
111         for inst in self.instructions:
112             inst.serialize(self.buf, offset)
113             offset += inst.len
114
115     @classmethod
116     def parser(cls, datapath, version, msg_type, msg_len, xid, buf):
117         msg = super(OFPFlowMod, cls).parser(
118             datapath, version, msg_type, msg_len, xid, buf)
119
120         (msg.cookie, msg.cookie_mask, msg.table_id,
121          msg.command, msg.idle_timeout, msg.hard_timeout,
122          msg.priority, msg.buffer_id, msg.out_port,
123          msg.out_group, msg.flags) = struct.unpack_from(
124             ofproto.OFP_FLOW_MOD_PACK_STR0, msg.buf,
125             ofproto.OFP_HEADER_SIZE)
126         offset = ofproto.OFP_FLOW_MOD_SIZE - ofproto.OFP_HEADER_SIZE
127
128         try:
129             msg.match = OFPMatch.parser(buf, offset)
130         except exception.OFPTruncatedMessage as e:
131             msg.match = e.ofpmsg
132             e.ofpmsg = msg
133             raise e
134
135         offset += utils.round_up(msg.match.length, 8)
```

```
136
137         instructions = []
138         try:
139             while offset < msg_len:
140                 i = OFPInstruction.parser(buf, offset)
141                 instructions.append(i)
142                 offset += i.len
143         except exception.OFPTruncatedMessage as e:
144             instructions.append(e.ofpmsg)
145             msg.instructions = instructions
146             e.ofpmsg = msg
147             raise e
148         except struct.error as e:
149             msg.instructions = instructions
150             raise exception.OFPTruncatedMessage(ofpmsg=msg,
151                                                 residue=buf[offset:],
152                                                 original_exception=e)
153         msg.instructions = instructions
154
155         return msg
```

10. *actions = [ofp_parser.OFPActionOutput(ofproto.OFPP_CONTROLLER,ofproto.OFPCML_NO_BUFFER)]———此操作表示将数据包输出到交换机端口*

```python
@OFPAction.register_action_type(ofproto.OFPAT_OUTPUT,
                                ofproto.OFP_ACTION_OUTPUT_SIZE)
class OFPActionOutput(OFPAction):
    """
    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
    ================ ======================================================
    """

    def __init__(self, port, max_len=ofproto.OFPCML_MAX,
                 type_=None, len_=None):
        super(OFPActionOutput, self).__init__()
        self.port = port
        self.max_len = max_len

    @classmethod
    def parser(cls, buf, offset):
        type_, len_, port, max_len = struct.unpack_from(
            ofproto.OFP_ACTION_OUTPUT_PACK_STR, buf, offset)
        return cls(port, max_len)

    def serialize(self, buf, offset):
        msg_pack_into(ofproto.OFP_ACTION_OUTPUT_PACK_STR, buf,
                      offset, self.type, self.len, self.port, self.max_len
)
```

11.*ofproto.OFPP_CONTROLLER,ofproto.OFPCML_NO_BUFFER 一个输出端口，一个是数据包最大长度*

```
<module 'ryu.ofproto.ofproto_v1_3' from '/usr/local/lib/python3.6/dist-packages/ryu/ofproto/ofproto_v1_3.py'>
```

```python
# enum ofp_port_no
OFPP_MAX = 0xffffff00
OFPP_IN_PORT = 0xfffffff8        # Send the packet out the input port. This
                                 # virtual port must be explicitly used
                                 # in order to send back out of the input
                                 # port.
OFPP_TABLE = 0xfffffff9          # Perform actions in flow table.
                                 # NB: This can only be the destination
                                 # port for packet-out messages.
OFPP_NORMAL = 0xfffffffa         # Process with normal L2/L3 switching.
OFPP_FLOOD = 0xfffffffb          # All physical ports except input port and
                                 # those disabled by STP.
OFPP_ALL = 0xfffffffc            # All physical ports except input port.
OFPP_CONTROLLER = 0xfffffffd     # Send to controller.    这是让交换机将数据包发送给控制器《重点》
OFPP_LOCAL = 0xfffffffe          # Local openflow "port".
OFPP_ANY = 0xffffffff              # Not associated with a physical port.
```

```python
actions = [ofp_parser.OFPActionOutput(ofproto.OFPP_CONTROLLER,ofproto.OFPCML_NO_BUFFER)]  流表项动作表示将数据包发送给控制器
```

```python
# enum ofp_controller_max_len
OFPCML_MAX = 0xffe5          # maximum max_len value which can be used to request a specific byte length.    可用于请求特定字节长度的最大Max值。（是向控制器请求的最大数据长度）
OFPCML_NO_BUFFER = 0xffff    # indicates that no buffering should be applied and the whole packet is to be sent to the controller.    指示不应应用缓冲，并且将整个数据包发送到控制器
```

```python
actions = [ofp_parser.OFPActionOutput(ofproto.OFPP_CONTROLLER,ofproto.OFPCML_NO_BUFFER)]  表示交换机中不用缓存数据包，全部发送给控制器即可
```

12.@set_ev_cls(ofp_event.EventOFPSwitchFeatures,CONFIG_DISPATCHER)中的事件，由ofp_event下面的代码动态生成

Python | 复制代码

```python
NAME = 'ofp_event'


class EventOFPMsgBase(event.EventBase):
    """
    The base class of OpenFlow event class.

    OpenFlow event classes have at least the following attributes.

    .. tabularcolumns:: |l|L|

    ============ ============================================================
    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.
    """

    def __init__(self, msg):
        self.timestamp = time.time()
        super(EventOFPMsgBase, self).__init__()
        self.msg = msg


#
# Create ofp_event type corresponding to OFP Msg
#

_OFP_MSG_EVENTS = {}


def _ofp_msg_name_to_ev_name(msg_name):
    return 'Event' + msg_name
```

```
41
42
43  def ofp_msg_to_ev(msg):
44      return ofp_msg_to_ev_cls(msg.__class__)(msg)
45
46
47  def ofp_msg_to_ev_cls(msg_cls):
48      name = _ofp_msg_name_to_ev_name(msg_cls.__name__)
49      return _OFP_MSG_EVENTS[name]
50
51
52  def _create_ofp_msg_ev_class(msg_cls):
53      name = _ofp_msg_name_to_ev_name(msg_cls.__name__)
54      # print 'creating ofp_event %s' % name
55
56      if name in _OFP_MSG_EVENTS:
57          return
58
59      cls = type(name, (EventOFPMsgBase,),
60                 dict(__init__=lambda self, msg:
61                      super(self.__class__, self).__init__(msg)))
62      globals()[name] = cls
63      _OFP_MSG_EVENTS[name] = cls
64
65
66  def _create_ofp_msg_ev_from_module(ofp_parser):
67      # print mod
68      for _k, cls in inspect.getmembers(ofp_parser, inspect.isclass):
69          if not hasattr(cls, 'cls_msg_type'):
70              continue
71          _create_ofp_msg_ev_class(cls)
72
73
74  for ofp_mods in ofproto.get_ofp_modules().values():
75      ofp_parser = ofp_mods[1]
76      # print 'loading module %s' % ofp_parser
77      _create_ofp_msg_ev_from_module(ofp_parser)
```