RYU SDN Framework Assignment

Ramaguru R

CB.EN.D*CYS21002-PT

28-Dec-2021

RYU (is the name of the Japanese Fictional Character, meaning "**FLOW**") is a *component-based Open-Source and Python-based Software-Defined Networking (SDN) Framework*. RYU supports various protocols like

- **OF-Config**, which is a protocol designed for OpenFlow switch management. It performs a collection of the settings and status of the logical switches, ports, and queues as defined in NETCONF (RFC 6241) schema.
- **XFlow**, which consists of **NetFlow** and **sFlow**. sFlow is primarily used to parse an sFlow packet from Open vSwitch. NetFlow and sFlow protocols are primarily network traffic management tools used for packet sampling and aggregation.
- **OVSDB**, which is primarily used to permit remote management of network nodes.

RYU Components:

RYU Framework consists of many components like Executables, Base Components, Open Flow Controllers and Encoder/Decoders. For development activities we also have pre-built applications, libraries and third-party library support.

- bin/ryu-manager – Main Executable file
- ryu.base.app_manager – The central management file
- ryu.controller.controller - The main component of OpenFlow Controller handles connections from switches; Generate and route events to appropriate entities.
- ryu.controller.dpset - manages the switches
- ryu.controller.ofp_event – OpenFlow event definitions
- ryu.controller.ofp_handler – OpenFlow Handlings
- ryu.ofproto.ofproto_v1_x – OpenFlow Definitions
- ryu.ofproto.ofproto_v1_x_parser – Encoder/Decoder for OpenFlow

Event Handler:

The different event handlers as available in RYU is shown below in Table 1.

RYU Reference File	Action
ryu.controller.handler.HANDSHAKE_DISPATCHER	Exchange of HELLO message
ryu.controller.handler.CONFIG_DISPATCHER	Waiting to receive Switch Features message
ryu.controller.handler.MAIN_DISPATCHER	Normal status
ryu.controller.handler.DEAD_DISPATCHER	Disconnection of connection

Table. 1: Event Handler in RYU

Match:

Match field defines what fields should be looked for by the RYU application to decide on the actions to be taken. Below table 2 shows some of the parameters used for match [1].

Match field name	Explanation
in_port	Port number of receiving port
in_phy_port	Physical port number of receiving port
metadata	Metadata used to pass information between tables
eth_dst	Destination MAC address of Ethernet
eth_src	Source MAC address of Ethernet
eth_type	Frame type of Ethernet
vlan_vid	VLAN ID
vlan_pcp	VLAN PCP
ip_dscp	IP DSCP
ip_ecn	IP ECN

Table. 2: Match Fields in OpenFlow Protocol

Actions:

Action fields define the packet forwarding to be used in **Packet-Out** and **Flow Mod** messages [1].

Value	Explanation
OFPP_IN_PORT	Forwarded to the receive port
OFPP_TABLE	Applied to the first flow table.
OFPP_NORMAL	Forwarded by the L2/L3 switch function
OFPP_FLOOD	Flooded to all physical ports of the VLAN except blocked ports and receiving ports
OFPP_ALL	Forwarded to all physical ports except receiving ports
OFPP_CONTROLLER	Sent to the controller as a Packet-In message.
OFPP_LOCAL	Indicates a local port of the switch
OFPP_ANY	Meant to be used as a wild card when you select a port using Flow Mod (delete) or Stats Requests messages, and it's not used in packet forwarding.

Table. 3: Actions in OpenFlow Protocol

GUI Topology Viewer:

To provide visual representation of the mininet created, RYU framework supports GUI Topology Viewers.

Command:

```
./bin/ryu run --observe-links ryu/app/gui_topology/gui_topology.py
```

This starts a server and the topology can be viewed at localhost in port number 8080.

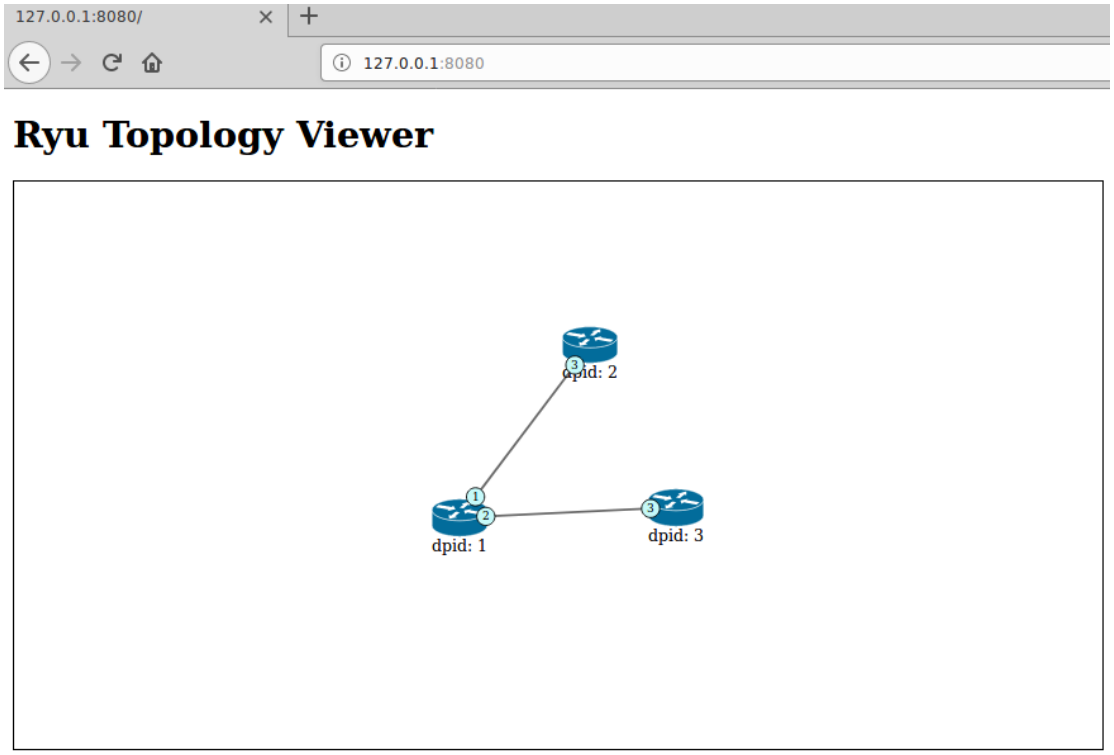

Fig.1: RYU GUI Topology Viewer

Applications:

RYU SDN Libraries allows us to create multiple applications. Some are listed below:

- Hub
- Spanning Tree
- L2 Switch
- L3 Switch
- Router
- Firewall

Programming Model:

The RYU Application Programming Model that serves as basis for any application written using RYU SDN Framework as shown in Fig. 2 [2].

The user logic is written as the application. The communication between the applications is performed through events which is maintained by a queue within each application. RYU uses multi-thread using evenlets. A thread starts the event loop. Whenever an event occurs of specified type, the event handler is called from the application's event loop. Event handlers are responsible for handling events within the application. An Event handler is defined by decorating application class method with an `ryu.controller.handler.set_ev_cls` decorator.

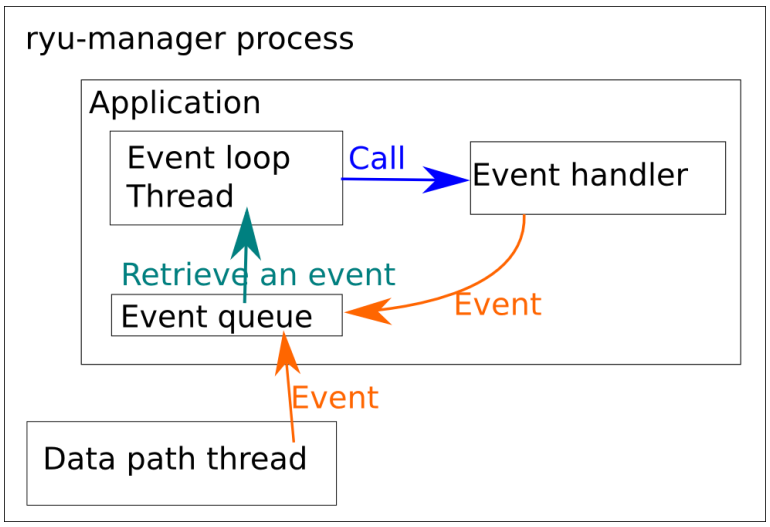

Fig.2: RYU Application Programming Model

REST API Support:

The Framework also supports API over REST.

Examples application written:

1. L2 Switch

```
ubuntu@sdnhubvm:~/ryu[20:14] (master)$ ./bin/ryu-manager hub.py --verbose
loading app hub.py
loading app ryu.controller.ofp_handler
instantiating app ryu.controller.ofp_handler of OFPHandler
instantiating app hub.py of L2Switch
BRICK ofp_event
  PROVIDES EventOFPPacketIn TO {'L2Switch': set(['main'])}
  CONSUMES EventOFPHello
  CONSUMES EventOFPErrorMsg
  CONSUMES EventOFPSwitchFeatures
  CONSUMES EventOFPPortDescStatsReply
  CONSUMES EventOFPEchoRequest
BRICK L2Switch
  CONSUMES EventOFPPacketIn
connected socket:<eventlet.greenio.GreenSocket object at 0x7ffb9b9495d0> address:('127.0.0.1', 60462)
hello ev <ryu.controller.ofp_event.EventOFPHello object at 0x7ffb9b949a50>
move onto config mode
connected socket:<eventlet.greenio.GreenSocket object at 0x7ffb9b949650> address:('127.0.0.1', 60463)
hello ev <ryu.controller.ofp_event.EventOFPHello object at 0x7ffb9b949e10>
move onto config mode
connected socket:<eventlet.greenio.GreenSocket object at 0x7ffb9b949710> address:('127.0.0.1', 60464)
hello ev <ryu.controller.ofp_event.EventOFPHello object at 0x7ffb9b961350>
move onto config mode
switch features ev version: 0x1 msg_type 0x6 xid 0x54b896cc OFPSwitchFeatures(actions=4095,capabilities=199,datapath_id=1,n_buffers=256,n_tables=254,ports={1: OFPPhyPort(port_no=1,hw_addr='ce:
f8:a3:80:9c:72',name='s1-eth1',config=0,state=0,curr=192,advertised=0,supported=0,peer=0), 2: OFPPhyPort(port_no=2,hw_addr='6a:02:6e:2f:9b:0f',name='s1-eth2',config=0,state=0,curr=192,advertis
ed=0,supported=0,peer=0), 65534: OFPPhyPort(port_no=65534,hw_addr='16:f4:b4:4c:26:4c',name='s1',config=1,state=1,curr=0,advertised=0,supported=0,peer=0)})
move onto main mode
```

Fig.3: L2 Switch Running in Verbose Mode

2. On Pinging the node h2 from h1 in the mininet.
3. Events can be observed.

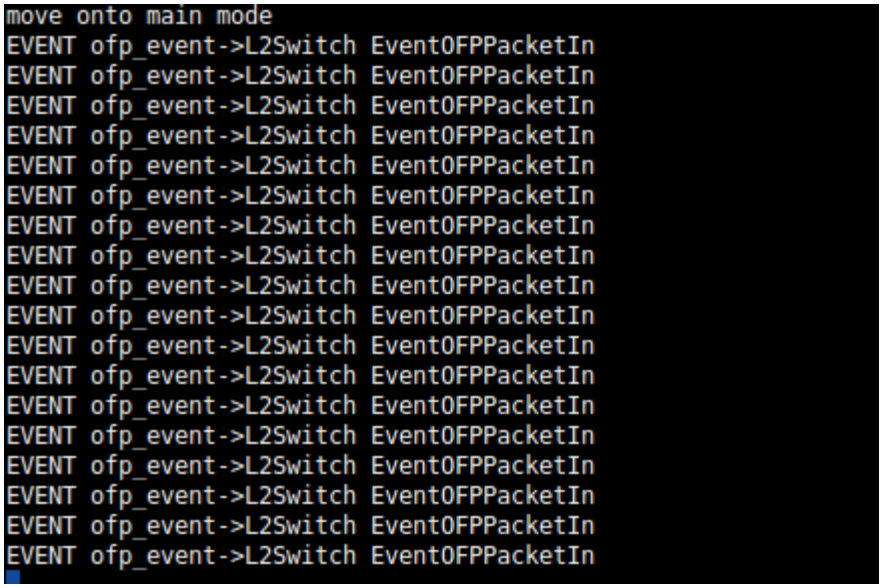

Fig.4: L2 Switch Event – OFPPacketIn

References:

[1] OpenFlow Protocol Action. https://osrg.github.io/ryu-book/en/html/openflow_protocol.html#match

[1] RYU SDN Framework Architecture. https://osrg.github.io/ryu-book/en/html/arch.html