

Lab 10 (Week 11)

Ryu Controller Application

(a brief tutorial for CW Part II)

CAN201

Dr. Wenjun Fan

Outline

- A Brief of Ryu Controller Framework
- Demo of the CW Part II

A Brief of Ryu Controller Framework

- **Useful Link for learning the Ryu SDN framework:**
 - https://osrg.github.io/ryu-book/en/html/switching_hub.html

A Brief of Ryu Controller Framework

- 'set_ev_cls' decorator tells Ryu when the decorated function should be called.

Install the table-miss flow entry

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

    # install the table-miss flow entry.
    match = parser.OFPMatch()
    actions = [parser.OFPActionOutput(ofproto.OFPP_CONTROLLER,
                                     ofproto.OFPCML_NO_BUFFER)]

    self.add_flow(datapath, 0, match, actions)
```

Install the table-miss flow entry

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

    # get Datapath ID to identify OpenFlow switches.
    dpid = datapath.id
    self.mac_to_port.setdefault(dpid, {})
```

To understand MAIN_DISPATCHER, ev.msg, msg.datapath, datapath.ofproto, datapath.ofproto_parser, please refer to:
https://ryu.readthedocs.io/en/latest/writing_ryu_app.html

A Brief of Ryu Controller Framework

- Analyse the received packets using the packet library

```
# analyse the received packets using the packet library.  
pkt = packet.Packet(msg.data)  
eth_pkt = pkt.get_protocol(ethernet.ethernet)  
dst = eth_pkt.dst  
src = eth_pkt.src
```

Library for other layers:

```
ipv4_pkt = pkt.get_protocol(ipv4.ipv4)  
tcp_pkt = pkt.get_protocol(tcp.tcp)
```

```
# you can further get the src and dst ip address of the IP pkt  
ip_src = ipv4_pkt.src  
ip_dst = ipv4_pkt.dst
```

A Brief of Ryu Controller Framework

- Build the switch forwarding table (i.e., mac to port table)

```
# get the received port number from packet_in message.
in_port = msg.match['in_port']

self.logger.info("packet in %s %s %s %s", dpid, src, dst, in_port)

# learn a mac address to avoid FLOOD next time.
self.mac_to_port[dpid][src] = in_port

# if the destination mac address is already learned,
# decide which port to output the packet, otherwise FLOOD.
if dst in self.mac_to_port[dpid]:
    out_port = self.mac_to_port[dpid][dst]
else:
    out_port = ofproto.OFPP_FLOOD
```

“src” here denotes the source MAC address where the pkt comes from.

A Brief of Ryu Controller Framework

- Set up the redirection action

```
# select server2's port
if SERVER2['mac'] in self.mac_to_port[dpid]:
    out_port = self.mac_to_port[dpid][SERVER2['mac']]
else:
    out_port = ofproto.OFPP_FLOOD

# flow entry for redirecting
match = parser.OFPMatch(eth_type=ether_types.ETH_TYPE_IP,
                        ipv4_src=pkt_ipv4.src, ipv4_dst=pkt_ipv4.dst)
actions = [parser.OFPACTIONSetField(eth_dst=SERVER2['mac']),
           parser.OFPACTIONSetField(ipv4_dst=SERVER2['ip']),
           parser.OFPACTIONOutput(port=out_port)]
```

The actions over here is for redirecting the pkt from Client to Server2.

So, think about how to set up the actions for forwarding the pkt replied by Server2 to Client

- Demo of CW Part II
- (The demo video is also uploaded under CW Part II materials in LMO)

CAN201 CW Part II

Demonstration