Software-Defined Networks Lab

Experiment 3 - Writing SDN Application Modules over Ryu

Release Date: April 11, 2018

1. Objective

In this experiment, you will learn:

- How to create a Ryu application

2. Tutorial

Note:

- **It's strongly recommended to run the experiments in virtual machine, e.g., Oracle VM VirtualBox[1], or VMware Workstation[2].**
- **Make sure you have Internet connection.**
- **Make sure you have super user permission.**

In the example, we will use Ubuntu 16.04[3] as our Operating System, but you can also choose other Linux OS.

[1] https://www.virtualbox.org/
[2] https://www.vmware.com/
[3] http://releases.ubuntu.com/16.04/

We will explain the basic behavior you need to know in `simple_switch.py`, and then you need to modify the code in `simple_switch_stp_13.py`.

Note: the script can be found under `ryu/app/` directory, or you can see it from this link: https://github.com/osrg/ryu/blob/master/ryu/app/simple_switch.py

simple_switch.py (part1)

```
53      @set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
54      def _packet_in_handler(self, ev):
55          msg = ev.msg
56          datapath = msg.datapath
57          ofproto = datapath.ofproto
58
59          pkt = packet.Packet(msg.data)
60          eth = pkt.get_protocol(ethernet.ethernet)
61
62          if eth.ethertype == ether_types.ETH_TYPE_LLDP:
63              # ignore lldp packet
64              return
65          dst = eth.dst
66          src = eth.src
67
68          dpid = datapath.id
69          self.mac_to_port.setdefault(dpid, {})
70
71          self.logger.info("packet in %s %s %s %s", dpid, src, dst, msg.in_port)
72
73          # learn a mac address to avoid FLOOD next time.
74          self.mac_to_port[dpid][src] = msg.in_port
75
76          if dst in self.mac_to_port[dpid]:
77              out_port = self.mac_to_port[dpid][dst]
78          else:
79              out_port = ofproto.OFPP_FLOOD
```

- line 60: `eth`: get the Ethernet information in the message (e.g., MAC).
- line 68: `dpid`: record the switch ID where this message came from.
- line 71: `in_port`: the port from which the message is received at that switch.
- line 77: `out_port`: the port we tell the switch that it needs to send the packet out.
- line 79: `OFPP_FLOOD`: a special action that tells the switch to send the packet to all ports.

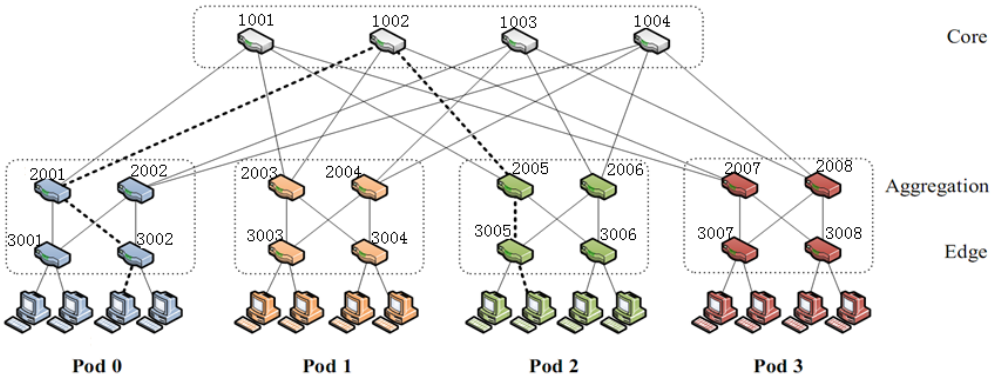
simple_switch.py (part2)

```
81          actions = [datapath.ofproto_parser.OFPActionOutput(out_port)]
82
83          # install a flow to avoid packet_in next time
84          if out_port != ofproto.OFPP_FLOOD:
85              self.add_flow(datapath, msg.in_port, dst, src, actions)
86
87          data = None
88          if msg.buffer_id == ofproto.OFP_NO_BUFFER:
89              data = msg.data
90
91          out = datapath.ofproto_parser.OFPPacketOut(
92              datapath=datapath, buffer_id=msg.buffer_id, in_port=msg.in_port,
93              actions=actions, data=data)
94          datapath.send_msg(out)
```

- line 81: `OFPActionOutput(out_port)`: action indicating that output a packet to the switch out_port.
- line 85: `add_flow`: add a new flow entry to the switch (see the function in the same file for detail)
- line 91: `OFPPacketOut`: create the PacketOut message.

3. Experiment

In this experiment, we will use Fat Tree[4] topology that you've built in Experiment 1.



- You create a fat tree topology with 4 pods. Each has 4 hosts, 2 edge switches, and 2 aggregation switches.
- Your job is to:
  1. Redirect all packets from their original destination host to a particular host. In this lab, we will provide a simple UDP client and server. The server listens on port 5134 to receive packets from the client.

[4] See page 8 of https://www.slideshare.net/AnkitaMahajan2/fattree-a-scalable-fauult-tolerant

2. Start the UDP server on host 1 (on the leftmost) in this topology.
3. Redirect any UDP packet that is originally sent from any UDP client to any IP address with port 5134 to the leftmost UDP server.
4. For example, assume that the leftmost host in this topology has an IP address of 10.0.0.1, and the rightmost host has an IP address of 10.0.0.16.
   If the provided UDP client running on the rightmost host wants to send a packet to one UDP server running on a host whose IP address is 10.0.0.2, you should redirect this packet to the leftmost host (by changing the destination IP address in the packet header from 10.0.0.2 to 10.0.0.1) and let the leftmost server to receive this packet.
   You don't need to consider the case of sending a packet from 10.0.0.1 to 10.0.0.1.

- Useful tips:
  1. You may need to use other actions in addition to `OFPActionOutput`.
  2. Use Mininet to create the topology. You need to make sure that the creation of hosts and switches have all completed before you run up the Ryu controller.
  3. You may want to use IP address instead of MAC address to verify the information.
  4. How you run Mininet is not specified, you can use either method mentioned in Experiment 1.
  5. To run ryu app, use `ryu run your-app.py`.

4. Submission

1. A pdf file (**report.pdf**) containing:
   a. report that tells how you do it, and
   b. screenshots to prove that you have done the redirection of packet.
2. Your module file (**app.py**) and a text file (**HOWTO.txt**) to tell TA how to run your application. You may include other files if needed.
3. Compress all the files into a zip file.
4. Name your zip file as "<studentID>.zip", e.g. 0556519.zip.
5. Upload the file onto e3.
6. Due date: 2018/05/01.

5. Useful Links

Ryubook

[1] https://osrg.github.io/ryu-book/zh_tw/html/switching_hub.html

Ryu doc

[2] http://ryu.readthedocs.io/en/latest/index.html
[3] http://ryu.readthedocs.io/en/latest/ofproto_v1_3_ref.html