

Build basic topo (mininet):

`sudo mn --controller=remote,ip=127.0.0.1 --mac --switch=ovsk,protocols=OpenFlow13 --topo=single,4`

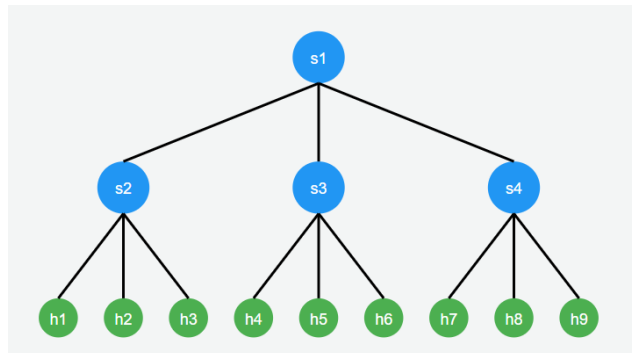
`--mac -i` (กำหนด ip) 10.1.1.0/24 `--topo=linear,4` // Switch 1 ตัว ต่อ Host 1 เครื่อง

-x หลังกำหนด Topo(mininet) = start xterm in all hosts

`sudo mn --controller=remote,ip=127.0.0.1:XXXX --mac ...` // กำหนด port

options	Description
<code>--controller</code>	type of controller local/remote and remote controller ip.
<code>--mac</code>	mac address starts with 00:00:00:00:00:01
<code>-i</code>	IP Subnets for the Topology
<code>--switch</code>	Switch type (ovsk - openvswitch kernel module), and openflow version.
<code>--topo</code>	topology type(linear,minimal,reversed,single,torus,tree) and params.

`--topo=tree,depth=2,fanout=3`



Start controller:

`ryu-manager ryu.app.simple_switch_13`

`ryu-manager --ofp-tcp-listen-port 6634 ryu.app.simple_switch_13` //กำหนด port

`ryu-manager ryu.app.simple_switch_13 ryu.app.ofctl_rest` // REST API สำหรับการควบคุมและจัดการสวิตช์

`ryu.app.simple_monitor_13` //statistics

Mode การทำงานเมื่อ Controller ล่ม

- Standalone: Switch ทำงานเป็น Traditional Switch แม้ว่า controller down ก็ยังงานได้ (Flow rules ยังอยู่)
- Secure: Switch พยายามเชื่อมต่อ Controller ตลอด ไม่ Forward packet (default) (ในกรณี flow timeout set)

1. ในโหมด Secure:

- Flow rules เดิมที่มีอยู่ใน table จะยังทำงานได้ (ถ้าไม่ได้กำหนด timeout)
- แต่ไม่สามารถสร้าง Flow rules ใหม่ได้เมื่อ Controller down
- ดังนั้นถ้ามีการสื่อสารระหว่าง hosts คู่ใหม่ที่ไม่เคยมี flow rules มาก่อน จะไม่สามารถสื่อสารกันได้

2. ในโหมด Standalone:

- Flow rules เดิมยังทำงานได้เหมือนกัน
- แต่สามารถสร้าง Flow rules ใหม่ได้เอง โดยทำงานเป็น traditional switch
- hosts คู่ใหม่สามารถสื่อสารกันได้ปกติ

```
sudo ovs-vsctl set-fail-mode <switch_name> standalone | secure
```

คำสั่งตรวจสอบ:

```
sudo ovs-vsctl show
```

```
sudo ovs-ofctl -O OpenFlow13 dump-flows s1
```

```
sudo ovs-ofctl -O OpenFlow13 dump-ports s1
```

```
sudo ovs-ofctl -O OpenFlow13 show s1
```

```
sudo ovs-appctl fdb/show s1 // Forwarding Database (FDB) หรือตาราง MAC Address
```

```
sudo ovs-ofctl -O OpenFlow13 dump-groups
```

```
sudo ovs-ofctl -O OpenFlow13 del-groups s5 group_id=0 /// ลบ
```

```
sudo ovs-ofctl -O OpenFlow13 dump-meters s1
```

```
sudo ovs-ofctl -O OpenFlow13 meter-stats s1
```

คำสั่งพื้นฐานเกี่ยวกับ arp ประกอบไปด้วย

```
arp -a. แสดงรายละเอียดข้อมูลใน ARP Table
```

```
arp -d 10.1.1.2 ลบข้อมูลใน ARP Table
```

```
arp -s [ip_dest] [mac]
```

Clear config mininet:

```
sudo mn -c
```

mininet commands:

links, pingall, h1 ping h2, h1 arp -a, **net**, xterm h1, h1 ifconfig, h1 ip route, link s1 s2 down
h1 python -m http.server (http server on host 1, use curl to fetch data: curl [ip]:[port] หรือใช้ ApacheBench),
h1 ps aux | grep iperf //ตรวจสอบว่า iperf server ทำงานอยู่บน host1 หรือไม่
h1 kill \$(ps aux | grep '[i]perf' | awk '{print \$2}') // terminate iperf
nodes, dump , h1 arp -a -s 192.168.1.2 00:00:00:00:00:02

iperf:

h4 iperf -s & : -s หมายถึง การกำหนดให้เป็น Server Mode ,& : Run in background
h1 iperf -c -s & เป็นได้ทั้ง server & client
h1 iperf -c h4 -i 10 -P 10 -t 10:
-c หมายถึง การกำหนดให้เป็น Client Mode
-i หมายถึง ช่วงเวลาในการรายงานผล เช่น -i 10 คือ รายงานผลทุก 10 วินาที
-t หมายถึง ช่วงเวลาในการทดสอบในหน่วยวินาที -t 30 คือ ช่วงเวลาทั้งหมดในการทดสอบเท่ากับ 30 วินาที
-b หมายถึง การกำหนดแบนด์วิดท์ เช่น -b 10m คือ แบนด์วิดท์ขนาด 10 Mbps
-P หมายถึง การเชื่อมต่อแบบขนาน (Parallel) คือการสร้าง Connection เชื่อมต่อระหว่าง Client และ Server ขึ้นมา
พร้อมๆ กัน เช่น -P 10 คือ สร้าง Connection ขึ้นมา 10 Connections แล้วส่งข้อมูลออกไปพร้อมๆ กัน

Iperf UDP: (ใช้ keyword optional ได้เหมือน tcp)

h1 iperf -u -s &
h4 iperf -u -c h1

ApacheBench(http test traffic)

ab -n 500 -c 50 <http://10.1.1.4:8000/>
- c 50 หมายถึง จำนวนของ Client ที่ร้องขอไปยัง Server เท่ากับ 50 Clients
- n 500 หมายถึง จำนวนการร้องขอจาก Client 1 ตัว ไปยัง Server 500 ครั้ง

Curl Methods & REST API FOR Controller (before start controller add this `ryu.app.ofctl_rest`)

GET Methos:

- curl http://localhost:8080/stats/switches | jq //ดู switches ทั้งหมด
- curl http://localhost:8080/stats/desc/1 | jq //ดู Details switch 1
- curl http://localhost:8080/stats/flow/1 | jq //ดู flows switch 1
- curl http://localhost:8080/stats/port/1 | jq //ดู port switch 1

POST Methos: // ใน windows ใช้ curl.exe และ Double-quotes

- curl -X POST http://localhost:8080/stats/flowentry/add -d '@hub_flow.json'//ส่ง flowrule ให้ controller
- curl -X POST http://localhost:8080/stats/groupentry/add -d '@hub_flow.json'//ส่ง group ให้ controller

- `curl.exe -X POST http://10.0.0.8:8080/stats/meterentry/add -d "@addmeter.json" //ส่ง meter limit`

DELETE Methods:

- `curl -X DELETE http://localhost:8080/stats/flowentry/clear/1 //ลบ flow rules เดิมทั้งหมด`

เพิ่ม TABLE แบบ Manual

- `sudo ovs-ofctl -O OpenFlow13 add-flow s1 "table=3,priority=100,icmp,actions=output:s1-eth2"`
- `sudo ovs-ofctl -O OpenFlow13 add-flow s1`
`"table=0,priority=100,eth_type=0x0800,nw_src=10.0.0.1,actions=goto_table:1"`

ตัวอย่าง multitable มีอีกด้านล่าง (s1-> h1,h2 ->s2 -> s3 ->h3,h4)

- Table 0: Source IP and packet type verification
- Table 1: ICMP traffic management (block **only** h1 -> h4)
- Table 2: TCP traffic control (ports 80, 443) (สังเกต priority)
- Table 3: Default forwarding

```

table1.json
test > test_multitable_REST2 > sw1 > table1.json
You, 30 minutes ago | 1 author (You)
1 {
2   "_comment": "Block ICMP to h4",
3   "dpid": 1,
4   "table_id": 1,
5   "idle_timeout": 0,
6   "hard_timeout": 0,
7   "priority": 100,
8   "match": {
9     "eth_type": 2048,
10    "ipv4_src": "10.0.0.1",
11    "ipv4_dst": "10.0.0.4",
12    "ip_proto": 1
13  },
14  "actions": [
15  ]
16 }
17

table2_2.json
test > test_multitable_REST2 > sw1 > table2_2.json > ...
You, 30 minutes ago | 1 author (You)
1 {
2   "_comment": "allow tcp 443",
3   "dpid": 1,
4   "table_id": 2,
5   "idle_timeout": 0,
6   "hard_timeout": 0,
7   "priority": 60,
8   "match": {
9     "eth_type": 2048,
10    "ip_proto": 6,
11    "tcp_dst": 443
12  },
13  "actions": [
14    {
15      "type": "GOTO_TABLE",
16      "table_id": 3
17    }
18  ]
19 }
20
  
```

Code python build topo (Run miniter topo by “sudo python [filename.py])

1. Simple = single,4

```
from mininet.topo import Topo
from mininet.net import Mininet
from mininet.log import setLogLevel
from mininet.cli import CLI
from mininet.node import OVSSwitch, Controller, RemoteController

...

class SingleSwitchTopo(Topo):
    "Single switch connected to n hosts."
    def build(self):
        s1 = self.addSwitch('s1', failMode='standalone')
        h1 = self.addHost('h1', mac="00:00:00:00:00:01", ip="192.168.1.1/24")
        h2 = self.addHost('h2', mac="00:00:00:00:00:02", ip="192.168.1.2/24")
        h3 = self.addHost('h3', mac="00:00:00:00:00:03", ip="192.168.1.3/24")
        h4 = self.addHost('h4', mac="00:00:00:00:00:04", ip="192.168.1.4/24")
        self.addLink(h1, s1)
        self.addLink(h2, s1)
        self.addLink(h3, s1)
        self.addLink(h4, s1)

if __name__ == '__main__':
    setLogLevel('info')
    topo = SingleSwitchTopo()
    c1 = RemoteController('c1', ip='127.0.0.1')
    net = Mininet(topo=topo, controller=c1)
    net.start()
    #net.pingAll()
    CLI(net)
    net.stop()

# Topology 1: Ring Topology (วงแหวน)
class RingTopo(Topo):
    "Ring topology with 4 switches and 4 hosts"
    def build(self):
        # Add switches
        switches = []
        for i in range(4):
            switches.append(self.addSwitch(f's{i+1}', failMode='standalone'))

        # Add hosts and connect to switches
        for i in range(4):
            host = self.addHost(f'h{i+1}',
                                mac=f"00:00:00:00:00:0{i+1}",
                                ip=f"192.168.1.{i+1}/24")
            self.addLink(host, switches[i])

        # Create ring connections between switches
        for i in range(4):
            self.addLink(switches[i], switches[(i+1)%4])
```

```

# Topology 2: Hierarchical Tree (ต้นไม้แบบลำดับชั้น)
class HierarchicalTopo(Topo):
    "Hierarchical topology with core, distribution and access layers"
    def build(self):
        # Core layer
        core = self.addSwitch('s1', failMode='standalone')

        # Distribution layer
        dist1 = self.addSwitch('s2', failMode='standalone')
        dist2 = self.addSwitch('s3', failMode='standalone')

        # Access layer
        access1 = self.addSwitch('s4', failMode='standalone')
        access2 = self.addSwitch('s5', failMode='standalone')

        # Connect core to distribution
        self.addLink(core, dist1)
        self.addLink(core, dist2)

        # Connect distribution to access
        self.addLink(dist1, access1)
        self.addLink(dist2, access2)

        # Add hosts
        h1 = self.addHost('h1', mac="00:00:00:00:00:01", ip="192.168.1.1/24")
        h2 = self.addHost('h2', mac="00:00:00:00:00:02", ip="192.168.1.2/24")
        h3 = self.addHost('h3', mac="00:00:00:00:00:03", ip="192.168.1.3/24")
        h4 = self.addHost('h4', mac="00:00:00:00:00:04", ip="192.168.1.4/24")

        # Connect hosts to access switches
        self.addLink(h1, access1)
        self.addLink(h2, access1)
        self.addLink(h3, access2)
        self.addLink(h4, access2)

# Topology 3: Mesh Topology (เมฆ)
class MeshTopo(Topo):
    "Partial mesh topology with 4 switches and 4 hosts"
    def build(self):
        # Add switches
        s1 = self.addSwitch('s1', failMode='standalone')
        s2 = self.addSwitch('s2', failMode='standalone')
        s3 = self.addSwitch('s3', failMode='standalone')
        s4 = self.addSwitch('s4', failMode='standalone')

        # Add hosts
        h1 = self.addHost('h1', mac="00:00:00:00:00:01", ip="192.168.1.1/24")
        h2 = self.addHost('h2', mac="00:00:00:00:00:02", ip="192.168.1.2/24")
        h3 = self.addHost('h3', mac="00:00:00:00:00:03", ip="192.168.1.3/24")
        h4 = self.addHost('h4', mac="00:00:00:00:00:04", ip="192.168.1.4/24")

        # Connect hosts to switches
        self.addLink(h1, s1)
        self.addLink(h2, s2)
        self.addLink(h3, s3)
        self.addLink(h4, s4)

        # Create mesh connections between switches
        self.addLink(s1, s2)
        self.addLink(s1, s3)
        self.addLink(s1, s4)
        self.addLink(s2, s3)
        self.addLink(s2, s4)
        self.addLink(s3, s4)

# Main function to run different topologies
def runTopology(topoClass):
    setLogLevel('info')
    topo = topoClass()
    c1 = RemoteController('c1', ip='127.0.0.1')
    net = Mininet(topo=topo, controller=c1)
    net.start()
    CLI(net)
    net.stop()

if __name__ == '__main__':
    # Uncomment the topology you want to run:
    # runTopology(RingTopo)
    # runTopology(HierarchicalTopo)
    # runTopology(MeshTopo)
    pass

```

ตัวอย่างการสร้าง Topo แบบกำหนด port การเชื่อมต่อได้

สังเกตว่ากำหนด mode = secure

```
class TriangleOfficeTopo(Topo):
    "Triangle topology connecting 3 buildings with 5 hosts"
    def build(self):
        # Add switches (one per building)
        s1 = self.addSwitch('s1', failMode='secure') # Building A
        s2 = self.addSwitch('s2', failMode='secure') # Building B
        s3 = self.addSwitch('s3', failMode='secure') # Building C

        # Add hosts with specific IP and MAC addresses
        # Building A devices
        h1 = self.addHost('h1', mac="00:00:00:00:00:01", ip="192.168.1.1/24") # Security Camera
        h2 = self.addHost('h2', mac="00:00:00:00:00:02", ip="192.168.1.2/24") # Access Control

        # Building B devices
        h3 = self.addHost('h3', mac="00:00:00:00:00:03", ip="192.168.1.3/24") # File Server

        # Building C devices
        h4 = self.addHost('h4', mac="00:00:00:00:00:04", ip="192.168.1.4/24") # Monitoring Syst
        h5 = self.addHost('h5', mac="00:00:00:00:00:05", ip="192.168.1.5/24") # Backup Server

        # Connect hosts to their respective switches
        self.addLink(h1, s1, port2=1)
        self.addLink(h2, s1, port2=2)
        self.addLink(h3, s2, port2=1)
        self.addLink(h4, s3, port2=1)
        self.addLink(h5, s3, port2=2)

        # Create triangle topology between switches
        self.addLink(s1, s2, port1=3, port2=2)
        self.addLink(s2, s3, port1=3, port2=3)
        self.addLink(s3, s1, port1=4, port2=4))
```

Code python build controller (Run controller by “ryu-manager [filename.py])

2. Hub controller (For ryu-manager: ryu-manager hub.py)

```
16 from ryu.base import app_manager
17 from ryu.controller import ofp_event
18 from ryu.controller.handler import CONFIG_DISPATCHER, MAIN_DISPATCHER
19 from ryu.controller.handler import set_ev_cls
20 from ryu.ofproto import ofproto_v1_3
21 from ryu.lib.packet import packet
22 from ryu.lib.packet import ethernet
23 from ryu.lib.packet import ether_types
24
25
26 You, 4 weeks ago | 1 author (You)
27 class SimpleSwitch13(app_manager.RyuApp):
28     OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]
29
30     def __init__(self, *args, **kwargs):
31         super(SimpleSwitch13, self).__init__(*args, **kwargs)
32         self.mac_to_port = {}
33
34     @set_ev_cls(ofp_event.EventOFPSwitchFeatures, CONFIG_DISPATCHER)
35     def switch_features_handler(self, ev):
36         datapath = ev.msg.datapath
37         ofproto = datapath.ofproto
38         parser = datapath.ofproto_parser
39
40         # writing flood entry
41         match = parser.OFPMatch()
42         actions = [parser.OFPActionOutput(port=ofproto.OFPP_FLOOD)]
43         self.add_flow(datapath, 0, match, actions)
44
45     def add_flow(self, datapath, priority, match, actions, buffer_id=None):
46         ofproto = datapath.ofproto
47         parser = datapath.ofproto_parser
48
49         inst = [parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS,
50                                             actions)]
51         if buffer_id:
52             mod = parser.OFPFlowMod(datapath=datapath, buffer_id=buffer_id,
53                                     priority=priority, match=match,
54                                     instructions=inst)
55         else:
56             mod = parser.OFPFlowMod(datapath=datapath, priority=priority,
57                                     match=match, instructions=inst)
58         datapath.send_msg(mod)
```


3. L3 switch controller (ryu-manager l3_switch.py)

```

65     datapath.send_msg(mod)
66     #from ryu.lib.packet import ipv4 เพิ่มโค้ดด้านล่างหลังจาก def add_flow *****
67 @set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
68 def _packet_in_handler(self, ev):
69     if ev.msg.msg_len < ev.msg.total_len:
70         self.logger.debug("packet truncated: only %s of %s bytes",
71                             ev.msg.msg_len, ev.msg.total_len)
72     msg = ev.msg
73     datapath = msg.datapath
74     ofproto = datapath.ofproto
75     parser = datapath.ofproto_parser
76     in_port = msg.match['in_port']
77     pkt = packet.Packet(msg.data)
78     eth = pkt.get_protocols(ethernet.ethernet)[0]
79     if eth.ethertype == ether_types.ETH_TYPE_LLDP:
80         return
81     dst = eth.dst
82     src = eth.src
83     dpid = datapath.id
84     self.mac_to_port.setdefault(dpid, {})
85     self.logger.info("packet in %s %s %s %s", dpid, src, dst, in_port)
86     self.mac_to_port[dpid][src] = in_port
87     if dst in self.mac_to_port[dpid]:
88         out_port = self.mac_to_port[dpid][dst]
89     else:
90         out_port = ofproto.OFPP_FLOOD
91     actions = [parser.OFPActionOutput(out_port)]
92     if out_port != ofproto.OFPP_FLOOD:
93         if eth.ethertype == ether_types.ETH_TYPE_IP:
94             ip = pkt.get_protocol(ipv4.ipv4)
95             srcip = ip.src
96             dstip = ip.dst
97             match = parser.OFPMatch(eth_type=ether_types.ETH_TYPE_IP,
98                                     ipv4_src=srcip,
99                                     ipv4_dst=dstip
100                                )
101             if msg.buffer_id != ofproto.OFP_NO_BUFFER:
102                 self.add_flow(datapath, 1, match, actions, msg.buffer_id)
103                 return
104             else:
105                 self.add_flow(datapath, 1, match, actions)
106     data = None
107     if msg.buffer_id == ofproto.OFP_NO_BUFFER:
108         data = msg.data
109     out = parser.OFPPacketOut(datapath=datapath, buffer_id=msg.buffer_id,
110                               in_port=in_port, actions=actions, data=data)
111     datapath.send_msg(out)
112

```

4. L4_Switch (port) สามารถใช้โค้ดจาก L3 แต่แก้ไขโค้ดและ import เพิ่ม

```
SDN ch3.pdf I4_switch.py 9+
ch3 > I4_switch.py > SimpleSwitch13 > _packet_in_handler
106 actions = [parser.OFPACTIONOutput(out_port)]
107 ...
108 from ryu.lib.packet import icmp
109 from ryu.lib.packet import tcp
110 from ryu.lib.packet import udp เพิ่มและแก้ไขโค้ดเพื่อกรอก L4
111 ...
112 if out_port != ofproto.OFPP_FLOOD:
113
114     # check IP Protocol and create a match for IP
115     if eth.ethertype == ether_types.ETH_TYPE_IP:
116         ip = pkt.get_protocol(ipv4.ipv4)
117         srcip = ip.src
118         dstip = ip.dst
119         protocol = ip.proto
120         # if ICMP Protocol
121         if protocol == in_proto.IPPROTO_ICMP:
122             match = parser.OFPMatch(eth_type=ether_types.ETH_TYPE_IP, ipv4_src=srcip, ipv4_dst=dstip,
123                                     ip_proto=protocol)
124
125         # if TCP Protocol
126         elif protocol == in_proto.IPPROTO_TCP:
127             t = pkt.get_protocol(tcp.tcp)
128             match = parser.OFPMatch(eth_type=ether_types.ETH_TYPE_IP, ipv4_src=srcip, ipv4_dst=dstip,
129                                     ip_proto=protocol, tcp_src=t.src_port, tcp_dst=t.dst_port,)
130
131         # If UDP Protocol
132         elif protocol == in_proto.IPPROTO_UDP:
133             u = pkt.get_protocol(udp.udp)
134             match = parser.OFPMatch(eth_type=ether_types.ETH_TYPE_IP, ipv4_src=srcip, ipv4_dst=dstip,
135                                     ip_proto=protocol, udp_src=u.src_port, udp_dst=u.dst_port,)
136
137         # verify if we have a valid buffer_id, if yes avoid to send both
138         # flow_mod & packet_out
139         if msg.buffer_id != ofproto.OFP_NO_BUFFER:
140             self.add_flow(datapath, 1, match, actions, msg.buffer_id)
141             return
142         else:
143             self.add_flow(datapath, 1, match, actions)
144     data = None
145     if msg.buffer_id == ofproto.OFP_NO_BUFFER:
146         data = msg.data
147
148     out = parser.OFPPacketOut(datapath=datapath, buffer_id=msg.buffer_id,
149                               in_port=in_port, actions=actions, data=data)
150     datapath.send_msg(out)
151
```

5. Flow timeout :

```

51 > def add_flow(self, datapath, priority, match, actions, buffer_id=None, idle=0, hard=0):..
52     ...
53     ไม่ต้อง import อะไรเพิ่ม ใช้จาก hub.py ได้เลย
54     สิ่งเกิด idle, hard สำหรับ flow timeout
55     ...
56
57 @set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
58 def _packet_in_handler(self, ev):
59     if ev.msg.msg_len < ev.msg.total_len:
60         self.logger.debug("packet truncated: only %s of %s bytes",
61                             ev.msg.msg_len, ev.msg.total_len)
62
63     msg = ev.msg
64     datapath = msg.datapath
65     ofproto = datapath.ofproto
66     parser = datapath.ofproto_parser
67     in_port = msg.match['in_port']
68     pkt = packet.Packet(msg.data)
69     eth = pkt.get_protocols(ethernet.ethernet)[0]
70     if eth.ethertype == ether_types.ETH_TYPE_LLDP:
71         return
72     dst = eth.dst
73     src = eth.src
74     dpid = datapath.id
75     self.mac_to_port.setdefault(dpid, {})
76     self.logger.info("packet in %s %s %s %s", dpid, src, dst, in_port)
77     self.mac_to_port[dpid][src] = in_port
78     if dst in self.mac_to_port[dpid]:
79         out_port = self.mac_to_port[dpid][dst]
80     else:
81         out_port = ofproto.OFPP_FLOOD
82     actions = [parser.OFPACTIONOutput(out_port)]
83     if out_port != ofproto.OFPP_FLOOD:
84         match = parser.OFPMATCH(in_port=in_port, eth_dst=dst, eth_src=src)
85         if msg.buffer_id != ofproto.OFP_NO_BUFFER:
86             self.add_flow(datapath, 1, match, actions, msg.buffer_id, idle=10, hard=30)
87             return
88         else:
89             self.add_flow(datapath, 1, match, actions, idle=10, hard=30)
90     data = None
91     if msg.buffer_id == ofproto.OFP_NO_BUFFER:
92         data = msg.data
93     out = parser.OFPPACKETOut(datapath=datapath, buffer_id=msg.buffer_id,
94                               in_port=in_port, actions=actions, data=data)
95     datapath.send_msg(out)
96

```

Code for REST API to push in flow of controller (ส่ง flow rule ให้ controller)

ความหมายค่า Port ตรง Action ใน Flow rule สามารถใช้ค่าแทนได้ "port": "CONTROLLER"

- ✓ OFPP_FLOOD (0xFFFFFFF8) = 4294967291 //packet ออกทุก port ยกเว้น port ที่ packet เข้ามา
- ✓ OFPP_ALL (0xFFFFFFF9) = 4294967292 ส่ง //packet ออกทุก port
- ✓ OFPP_CONTROLLER (0xFFFFFFF8) = 4294967293 //packet ไปที่ controller
- ✓ OFPP_TABLE (0xFFFFF800) = 4294967290 //packet เข้า flow table
- ✓ OFPP_IN_PORT (0xFFFFF800) = 4294967288 //packet กลับออกไปทาง port ที่เข้ามา
- ✓ OFPP_NORMAL (0xFFFFF800) = 4294967290 //ให้ switch จัดการ packet ตามการทำงานปกติ

OFPP_TABLE และ OFPP_NORMAL มีค่าเท่ากัน (4294967290) เพราะทั้งสองค่านี้ใช้แทนพฤติกรรมเดียวกันคือการให้ switch จัดการ packet ตามการทำงานปกติผ่าน flow table

1. Hub.js //วิธีส่งให้ controller ใช้ curl -X POST (stats/flowentry/add) ดูในสรุปด้านบน

```
1 {
2   "dpid": 1,
3   "table_id": 0,
4   "idle_timeout": 0,
5   "hard_timeout": 0,
6   "priority": 100,
7   "match":{
8   },
9   "actions":[
10    {
11      "type":"OUTPUT",
12      "port": 4294967291
13    }
14  ]
15 }
```

2. L2

2.1 ARP & Flow1 (ของ host1) // ARP ถ้าไม่ได้ให้แก้เป็น 2054 (dl_type หรือ "eth_type": 2054

```
...
{ // ARP
  "dpid": 1,
  "table_id": 0,
  "idle_timeout": 0,
  "hard_timeout": 0,
  "priority": 100,
  "match":{
    "dl_dst": "ff:ff:ff:ff:ff:ff"
    // "dl type": 2054 ARP
  },
  "actions":[
    {
      "type":"OUTPUT",
      "port": 4294967291
    }
  ]
}

{ //flow 1 For h1
  "dpid": 1,
  "table_id": 0,
  "idle_timeout": 0,
  "hard_timeout": 0,
  "priority": 100,
  "match":{
    "dl_dst": "00:00:00:00:00:01"
  },
  "actions":[
    {
      "type":"OUTPUT",
      "port": 1
    }
  ]
}
```

```

{
  "dpid": 1, //switch
  "table_id": 0,
  "idle_timeout": 0,
  "hard_timeout": 0,
  "priority": 100,
  "match":{
    "dl_dst": "00:00:00:00:00:02"
  },
  "actions":[
    {
      "type":"OUTPUT",
      "port": 2 //host 2
    }
  ]
}

```

switch_1.json ...sw2 U × switch_2.json

test > test_l2_REST2 > sw2 > switch_1.json >

```

1  {
2      "dpid": 2,
3      "table_id": 0,
4      "idle_timeout": 0,
5      "hard_timeout": 0,
6      "priority": 100,
7      "match":{
8          "eth_type": 2048,
9          "ipv4_dst": "10.0.0.1"
10     },
11     "actions":[
12         {
13             "type":"OUTPUT",
14             "port": 2
15         }
16     ]
17 }

```

L3 ต้องระบุ eth_type ด้วย

3. Multitable

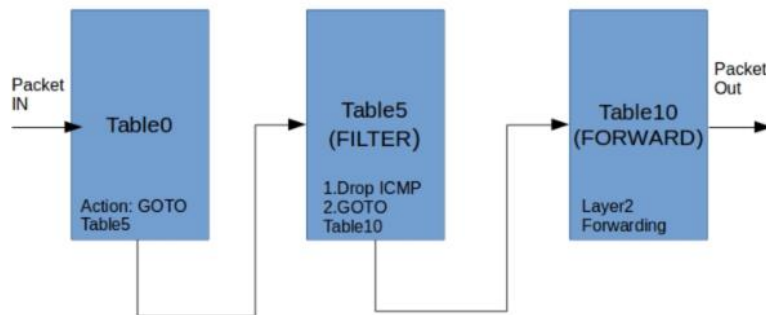
eth_type (EtherType) มีค่าหลักๆ ที่ใช้บ่อยดังนี้:

1. 0x0800 (2048) - IPv4
2. 0x0806 (2054) - ARP
3. 0x86DD (34525) - IPv6
4. 0x8100 (33024) - VLAN-tagged frame (802.1Q)
5. 0x88CC (35020) - LLDP (Link Layer Discovery Protocol)

ip_proto (IP Protocol Number) มีค่าหลักๆ ที่ใช้บ่อยดังนี้:

1. 1 - ICMP (Internet Control Message Protocol)
2. 6 - TCP (Transmission Control Protocol)
3. 17 - UDP (User Datagram Protocol)
4. 2 - IGMP (Internet Group Management Protocol)
5. 50 - ESP (Encapsulating Security Payload - IPsec)
6. 51 - AH (Authentication Header - IPsec)
7. 89 - OSPF (Open Shortest Path First)
8. 47 - GRE (Generic Routing Encapsulation)

ตัวอย่าง (Single,4)



```
SDN ch4.pdf  table0_flow1.json  ...  table5_flow1.json  ...  table5_flow2.json  ...
document > ch4 > ofctl > multitable > table0_flow  document > ch4 > ofctl > multitable > table5_flow  document > ch4 > ofctl > multitable > table5_flow
You, 4 weeks ago | 1 author (You)  You, 4 weeks ago | 1 author (You)  You, 4 weeks ago | 1 author (You)

1 {
2   "dpid": 1,
3   "table_id": 0,
4   "idle_timeout": 0,
5   "hard_timeout": 0,
6   "priority": 0,
7   "match":{
8 },
9   "actions":[
10    {
11      "type": "GOTO_TABLE",
12      "table_id": 5
13    }
14  ]
15 }

1 {
2   "dpid": 1,
3   "table_id": 5,
4   "idle_timeout": 0,
5   "hard_timeout": 0,
6   "priority": 0,
7   "match":{
8 },
9   "actions":[
10    {
11      "type": "GOTO_TABLE",
12      "table_id": 10
13    }
14  ]
15 }

1 {
2   "dpid": 1,
3   "table_id": 5,
4   "idle_timeout": 0,
5   "hard_timeout": 0,
6   "priority": 100,
7   "match":{
8     "eth_type": 2048,
9     "ipv4_dst": "10.0.0.4",
10    "ip_proto": 1
11  },
12   "actions":[
13   ]
14 }
15 }
```

```

document > ch4 > ofctl > multitable > table10_arp.json >
You, 4 weeks ago | 1 author (You)
1 {
2   "dpid": 1,
3   "table_id": 10,
4   "idle_timeout": 0,
5   "hard_timeout": 0,
6   "priority": 0,
7   "match": {
8     "dl_dst": "ff:ff:ff:ff:ff:ff"
9   },
10  "actions": [
11    {
12      "type": "OUTPUT",
13      "port": 4294967291
14    }
15  ]
16 }
17

document > ch4 > ofctl > multitable > table10_flow.json >
You, 4 weeks ago | 1 author (You)
1 {
2   "dpid": 1,
3   "table_id": 10,
4   "idle_timeout": 0,
5   "hard_timeout": 0,
6   "priority": 0,
7   "match": {
8     "dl_dst": "00:00:00:00:00:01"
9   },
10  "actions": [
11    {
12      "type": "OUTPUT",
13      "port": 1
14    }
15  ]
16 }
17

document > ch4 > ofctl > multitable > table10_flow2.json >
You, 4 weeks ago | 1 author (You)
1 {
2   "dpid": 1,
3   "table_id": 10,
4   "idle_timeout": 0,
5   "hard_timeout": 0,
6   "priority": 0,
7   "match": {
8     "dl_dst": "00:00:00:00:00:02"
9   },
10  "actions": [
11    {
12      "type": "OUTPUT",
13      "port": 2
14    }
15  ]
16 }
17

document > ch4 > ofctl > multitable > table10_flow3.json >
You, 4 weeks ago | 1 author (You)
1 {
2   "dpid": 1,
3   "table_id": 10,
4   "idle_timeout": 0,
5   "hard_timeout": 0,
6   "priority": 0,
7   "match": {
8     "dl_dst": "00:00:00:00:00:03"
9   },
10  "actions": [
11    {
12      "type": "OUTPUT",
13      "port": 3
14    }
15  ]
16 }
17

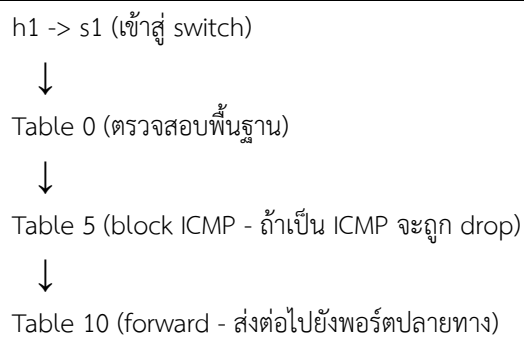
document > ch4 > ofctl > multitable > table10_flow4.json >
You, 4 weeks ago | 1 author (You)
1 {
2   "dpid": 1,
3   "table_id": 10,
4   "idle_timeout": 0,
5   "hard_timeout": 0,
6   "priority": 0,
7   "match": {
8     "dl_dst": "00:00:00:00:00:04"
9   },
10  "actions": [
11    {
12      "type": "OUTPUT",
13      "port": 4
14    }
15  ]
16 }
17

```

ดูตรง actions [port/ table_id] ดีๆ

4. Group Table

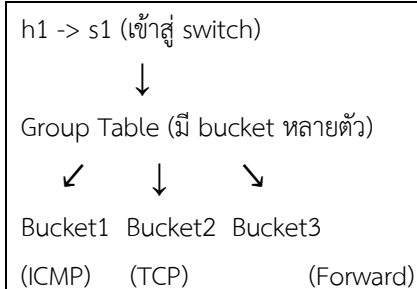
Multi-table workflow:



- ทำงานเป็นลำดับขั้น (pipeline)
- แต่ละ table มีหน้าที่เฉพาะ
- packet ต้องผ่านทุก table ตามลำดับ

Group Table workflow

`sudo ovs-ofctl -O OpenFlow13 del-groups s5 group_id=0 // ลบ group id ในกรณี /add ผิด`



กรณี ALL Group:

- packet จะถูกทำสำเนาและส่งไปทุก bucket
- แต่ละ bucket ทำงานแยกกัน

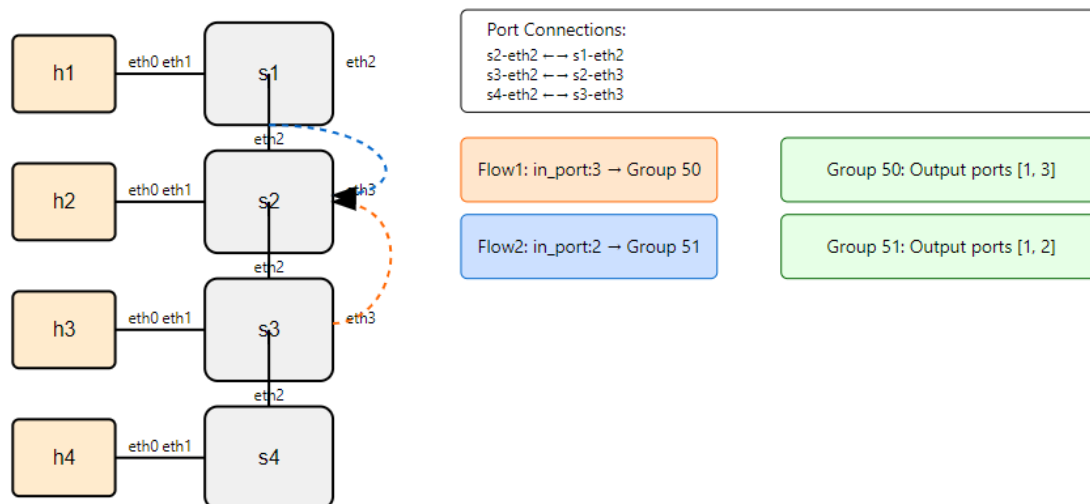
กรณี SELECT Group:

- packet จะถูกส่งไปเพียง bucket เดียว
- เลือก bucket ตามเงื่อนไขที่กำหนด (เช่น load balancing)

ความแตกต่างหลัก:

1. Multi-table ทำงานเป็นลำดับขั้นตอน แต่ Group Table ทำงานแบบขนานหรือเลือกทำอันใดอันหนึ่ง
2. Multi-table แยกตามประเภทการทำงาน แต่ Group Table รวมกลุ่มการทำงานที่เกี่ยวข้องกัน

4.1 Sniffer (linear,4 & ryu.app.simple_switch_13 ryu.app.ofctl_rest) ต้องส่ง Group ก่อน



1. Flow1 (เมื่อ packet เข้ามาที่ port 3 sw2):
 - Match condition: in_port = 3 (packet มาจาก s3)
 - Action: ส่งไปที่ Group 50
 - Group 50 จะทำการ multicast packet ไปที่:

- Output port 1 (ไปที่ h2)
 - Output port 3 (ส่งกลับไปที่ s3)
2. Flow2 (เมื่อ packet เข้ามาที่ port 2):
- Match condition: in_port = 2 (packet มาจาก s1)
 - Action: ส่งไปที่ Group 51
 - Group 51 จะทำการ multicast packet ไปที่:
 - Output port 1 (ไปที่ h2)
 - Output port 2 (ส่งกลับไปที่ s1)
1. เมื่อ packet เข้ามาที่ s2 ระบบจะเช็ค incoming port
 2. ถ้าเข้ามาที่ port 3 จะเข้า Flow1 -> Group 50
 3. ถ้าเข้ามาที่ port 2 จะเข้า Flow2 -> Group 51
 4. จากนั้น Group จะทำการ multicast packet ไปยัง output ports ที่กำหนดไว้
- หมายเหตุ: ทั้ง Group 50 และ 51 เป็น type "ALL" ซึ่งหมายถึงจะทำการ multicast packet ไปยังทุก output ports ที่กำหนดไว้ในกลุ่มนั้นๆ

group50.json M X

```

document > ch4 > ofctl > sniffer > group50.json > ...
You, 50 minutes ago | 1 author (You)
1 {
2   "dpid": 2,
3   "type": "ALL",
4   "group_id": 50,
5   "buckets": [
6     {
7       "actions": [
8         {
9           "type": "OUTPUT",
10          "port": 1
11        }
12      ],
13    },
14    {
15      "actions": [
16        {
17          "type": "OUTPUT",
18          "port": 3
19        }
20      ]
21    }
22  ]
23 }
```

group51.json M X

```

document > ch4 > ofctl > sniffer > group51.json > ...
You, 50 minutes ago | 1 author (You)
1 {
2   "dpid": 2,
3   "type": "ALL",
4   "group_id": 51,
5   "buckets": [
6     {
7       "actions": [
8         {
9           "type": "OUTPUT",
10          "port": 1
11        }
12      ],
13    },
14    {
15      "actions": [
16        {
17          "type": "OUTPUT",
18          "port": 2
19        }
20      ]
21    }
22  ]
23 }
```

flow1.json X

```

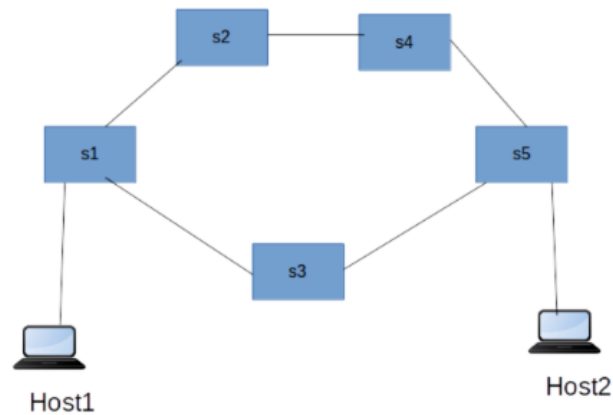
document > ch4 > ofctl > sniffer > flow1.json > ...
You, 4 weeks ago | 1 author (You)
1 {
2   "dpid": 2,
3   "table_id": 0,
4   "idle_timeout": 0,
5   "hard_timeout": 0,
6   "priority": 100,
7   "match": {
8     "in_port": 3
9   },
10  "actions": [
11    {
12      "type": "GROUP",
13      "group_id": 50
14    }
15  ]
16 }
```

flow2.json X

```

document > ch4 > ofctl > sniffer > flow2.json > ...
You, 4 weeks ago | 1 author (You)
1 {
2   "dpid": 2,
3   "table_id": 0,
4   "idle_timeout": 0,
5   "hard_timeout": 0,
6   "priority": 100,
7   "match": {
8     "in_port": 2
9   },
10  "actions": [
11    {
12      "type": "GROUP",
13      "group_id": 51
14    }
15  ]
16 }
```

4.2 Load balancing



h1 arp -a -s 192.168.1.2 00:00:00:00:00:02 // set static host

```
SDN ch4.pdf  s2_flow1.json  s2_flow2.json
document > ch4 > ofctl > lb > s2_flow1.json > ...
You, 4 weeks ago | 1 author (You)
1 {
2   "dpid": 2,
3   "table_id": 0,
4   "idle_timeout": 0,
5   "hard_timeout": 0,
6   "priority": 100,
7   "match":{
8     "in_port":1
9   },
10  "actions":[
11    {
12      "type":"OUTPUT",
13      "port": 2
14    }
15  ]
16 }

document > ch4 > ofctl > lb > s2_flow2.json > ...
You, 2 days ago | 1 author (You)
1 {
2   "dpid": 2,
3   "table_id": 0,
4   "idle_timeout": 0,
5   "hard_timeout": 0,
6   "priority": 100,
7   "match":{
8     "in_port":2
9   },
10  "actions":[
11    {
12      "type":"OUTPUT",
13      "port": 1
14    }
15  ]
16 }
```

สร้าง Flow rules ไปกลับของ sw2, sw3, sw4

```
SDN ch4.pdf  s1_flow1.json  s1_flow2.json
document > ch4 > ofctl > lb > s1_flow1.json > [ ] actions >
You, 2 days ago | 1 author (You)
1 {
2   "dpid": 1,
3   "table_id": 0,
4   "idle_timeout": 0,
5   "hard_timeout": 0,
6   "priority": 100,
7   "match":{
8     "in_port":1 //sw2
9   },
10  "actions":[
11    {
12      "type":"OUTPUT",
13      "port": 3 //h1
14    }
15  ]
16 }

document > ch4 > ofctl > lb > s1_flow2.json > [ ] actions >
...
1 {
2   "dpid": 1,
3   "table_id": 0,
4   "idle_timeout": 0,
5   "hard_timeout": 0,
6   "priority": 100,
7   "match":{
8     "in_port":2 //sw3
9   },
10  "actions":[
11    {
12      "type":"OUTPUT",
13      "port": 3 //h1
14    }
15  ]
16 }
```

Flow rules sw1 สำหรับรับมาจาก sw2,sw3 (ทำสำหรับ switch5 ด้วย)

```
document > ch4 > ofctl > lb > s1_flow3.json > ... :tl > lb > s1_group50.json > [ ] buckets > { } 1 > [ ] actions > .
You, 2 days ago | 1 author (You)
1 {
2   "dpid": 1,
3   "table_id": 0,
4   "idle_timeout": 0,
5   "hard_timeout": 0,
6   "priority": 100,
7   "match": {
8     "in_port": 3 //h1
9   },
10  "actions": [
11    {
12      "type": "GROUP",
13      "group_id": 50
14    }
15  ]
16 }
You, 4 weeks ago * add

document > ch4 > ofctl > lb > s1_group50.json > [ ] buckets > { } 1 > [ ] actions > .
You, 2 days ago | 1 author (You)
1 {
2   "dpid": 1,
3   "type": "SELECT",
4   "group_id": 50,
5   "buckets": [
6     {
7       "weight": 50,
8       "actions": [
9         {
10          "type": "OUTPUT",
11          "port": 1 //sw2
12        }
13      ]
14     },
15     {
16       "weight": 50,
17       "actions": [
18         {
19          "type": "OUTPUT",
20          "port": 2 //sw3
21        }
22      ]
23     }
24   ]
25 }
26 }
```

Flow rules sw1 สำหรับขาออกจาก h1 โดยส่งไปให้ group 50 เพื่อทำ load balance (ทำสำหรับ sw5 ด้วย)

5. Statistics

- 5.1 Create simple TOPO (single,2 with -I [IP])
- 5.2 `ryu-manager ryu.app.simple_monitor_13`
- 5.3 perform TCP traffic
- 5.4 result will show in ryu terminal

6. Meter Table

- 6.1 `sudo ovs-ofctl -O OpenFlow13 meter-stats s1`
- 6.2 `curl /stats/meterentry/add -d "....."`

```
SDN ch4.pdf addmeter.json x
document > ch4 > ofctl > meter > addmeter.json > [ ] banc
You, 4 weeks ago | 1 author (You)
1 {
2   "dpid": 1,
3   "flags": "KBPS",
4   "meter_id": 1,
5   "bands": [
6     {
7       "type": "DROP",
8       "rate": 1000
9     }
10  ]
11 }
```

Add flow ตามปกติ (arp, des_mac,)

Xterm commands

- tcpdump -i any -v หรือ tcpdump -i s1-eth3 ผ่าน terminal ไม่ต้อง xterm #####
- tcpdump -i any -v udp, tcpdump -i any -v host 10.0.0.1, tcpdump -i any -v port 5001
- tcpdump -i h1-eth0 -v, tcpdump -i any -vvv

Flow Manager:

1. clone git repo,
2. ryu-manager -- observe-links ~/flowmanager/flowmanager.py ryu.app.simple_switch_13
3. sudo mn -- controller=remote,ip=127.0.0.1 -- mac -i 10.1.1.0/24 -- topo=tree,depth=2,fanout=2
4. <http://localhost:8080/home/index.html>
5. Pingall

Statistics Collection (ch5)

1. flow statistics

```
from ryu.base import app_manager
from ryu.controller import ofp_event
from ryu.controller.handler import CONFIG_DISPATCHER, MAIN_DISPATCHER, set_ev_cls
from ryu.ofproto import ofproto_v1_3
from ryu.lib.packet import packet, ethernet, ether_types
from ryu.lib import hub

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 = {}
        self.datapaths = {}
        hub.spawn(self, self._monitor)

    def _monitor(self):
        while True:
            hub.sleep(10)
            [dp.send_msg(dp.ofproto_parser.OFPFlowStatsRequest(dp))
             for dp in self.datapaths.values()]

    @set_ev_cls(ofp_event.EventOFPFlowStatsReply, MAIN_DISPATCHER)
    def _stats_reply(self, ev):
        [self.logger.info(f'Flow: {stat}, bytes: {stat.byte_count}, '
                        f'packets: {stat.packet_count}') for stat in ev.msg.body]

    @set_ev_cls(ofp_event.EventOFPSwitchFeatures, CONFIG_DISPATCHER)
    def _switch_features(self, ev):
        dp = ev.msg.datapath
        self.datapaths[dp.id] = dp
        parser = dp.ofproto_parser
        match = parser.OFPMatch()
        actions = [parser.OFPActionOutput(dp.ofproto.OFPP_CONTROLLER,
                                          dp.ofproto.OFPCML_NO_BUFFER)]
        self._add_flow(dp, 0, match, actions)

    def _add_flow(self, dp, priority, match, actions, buffer_id=None):
        ofproto = dp.ofproto
        parser = dp.ofproto_parser
        inst = [parser.OFPInstructionActions(ofproto.OFPT_APPLY_ACTIONS, actions)]
        mod = parser.OFPFlowMod(
            datapath=dp, priority=priority, match=match, instructions=inst,
            buffer_id=buffer_id if buffer_id is not None else ofproto.OFP_NO_BUFFER
        )
        dp.send_msg(mod)

    @set_ev_cls(ofp_event.EventOFPacketIn, MAIN_DISPATCHER)
    def _packet_in(self, ev):
        msg = ev.msg
        dp = msg.datapath
        ofproto = dp.ofproto
        parser = dp.ofproto_parser
        in_port = msg.match['in_port']

        pkt = packet.Packet(msg.data)
        eth = pkt.get_protocols(ethernet.ethernet)[0]
        if eth.ethertype == ether_types.ETH_TYPE_LLDP:
            return

        dst, src = eth.dst, eth.src
        dpid = dp.id

        self.mac_to_port.setdefault(dpid, {})
        self.mac_to_port[dpid][src] = in_port
        out_port = self.mac_to_port[dpid].get(dst, ofproto.OFPP_FLOOD)
        actions = [parser.OFPActionOutput(out_port)]

        if out_port != ofproto.OFPP_FLOOD:
            match = parser.OFPMatch(in_port=in_port, eth_dst=dst, eth_src=src)
            if msg.buffer_id != ofproto.OFP_NO_BUFFER:
                self._add_flow(dp, 1, match, actions, msg.buffer_id)
                return
            self._add_flow(dp, 1, match, actions)

        data = None if msg.buffer_id != ofproto.OFP_NO_BUFFER else msg.data
        out = parser.OFPActionOutput(datapath=dp, buffer_id=msg.buffer_id,
                                     in_port=in_port, actions=actions, data=data)
        dp.send_msg(out)
```

2. Aggregate Stats (ใช้ของเดิม flow statistics)

```
1 def _monitor(self):
2     while True:
3         hub.sleep(10)
4         [dp.send_msg(dp.ofproto_parser.OFPAggregateStatsRequest(
5             dp, 0, dp.ofproto.OFPTT_ALL,
6             dp.ofproto.OFPP_ANY, dp.ofproto.OFPG_ANY,
7             0, 0, dp.ofproto_parser.OFPMatch()
8         )) for dp in self.datapaths.values()]
9
10 @set_ev_cls(ofp_event.EventOFPAggregateStatsReply, MAIN_DISPATCHER)
11 def _stats_reply(self, ev):
12     body = ev.msg.body
13     self.logger.info(f'AggregateStats: {body}, Flows: {body.flow_count}')
```

3. port statistics monitoring

```
1 # ส่วนที่แตกต่างสำหรับ Port Stats:
2 def _monitor(self):
3     while True:
4         hub.sleep(10)
5         [dp.send_msg(dp.ofproto_parser.OFPPortStatsRequest(dp))
6          for dp in self.datapaths.values()]
7
8 @set_ev_cls(ofp_event.EventOFPPortStatsReply, MAIN_DISPATCHER)
9 def _stats_reply(self, ev):
10    [self.logger.info(f"Port stats: {stat}") for stat in ev.msg.body]
```

Params (ch5) เรียกใช้ไฟล์ config จากข้างนอก

1. สร้างไฟล์ params.conf [DEFAULT] ขึ้นบรรทัดใหม่ INTERVAL = 5
2. Import ในโค้ด controller (ryu) from ryu import cfg
3. เรียกใช้ ryu-manager --config-file params.conf flow_stats_param.py

Group (ch5) ใช้โค้ด python (ryu) สร้าง Group แทน REST API บทที่ 4 (linear,4) (sniffer)

```
1 from ryu.base import app_manager
2 from ryu.controller import ofp_event
3 from ryu.controller.handler import CONFIG_DISPATCHER, MAIN_DISPATCHER, set_ev_cls
4 from ryu.ofproto import ofproto_v1_3
5 from ryu.lib.packet import packet, ethernet, ether_types
6
7 class SimpleSwitch13(app_manager.RyuApp):
8     OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]
9
10    def __init__(self, *args, **kwargs):
11        super(SimpleSwitch13, self).__init__(*args, **kwargs)
12        self.mac_to_port = {}
13
14    def _add_flow(self, dp, priority, match, actions, buffer_id=None):
15        inst = [dp.ofproto_parser.OFInstructionActions(
16                dp.ofproto.OFPT_APPLY_ACTIONS, actions)]
17        dp.send_msg(dp.ofproto_parser.OFFlowMod(
18                datapath=dp, priority=priority, match=match,
19                instructions=inst, buffer_id=buffer_id if buffer_id
20                else dp.ofproto.OFP_NO_BUFFER))
21
22    def _setup_group_tables(self, dp):
23        p = dp.ofproto_parser
24        groups = [
25            (50, [(1,), (3,)]), # group_id 50: output to ports 1 and 3
26            (51, [(1,), (2,)]), # group_id 51: output to ports 1 and 2
27        ]
28        for group_id, ports in groups:
29            buckets = [p.OFPPBucket(actions=[p.OFPActionOutput(port)])
30                       for port in p[0] for p in ports]
31            dp.send_msg(p.OFPGGroupMod(dp, dp.ofproto.OFPGC_ADD,
32                                     dp.ofproto.OFPGT_ALL, group_id, buckets))
33            self._add_flow(dp, 10, p.OFPMatch(in_port=ports[1][0]),
34                           [p.OFPActionGroup(group_id=group_id)])
35
36    @set_ev_cls(ofp_event.EventOFPSwitchFeatures, CONFIG_DISPATCHER)
37    def _switch_features(self, ev):
38        dp = ev.msg.datapath
39        p = dp.ofproto_parser
40        self._add_flow(dp, 0, p.OFPMatch(),
41                       [p.OFPActionOutput(dp.ofproto.OFPP_CONTROLLER,
42                                         dp.ofproto.OFPCML_NO_BUFFER)])
43        if dp.id == 2:
44            self._setup_group_tables(dp)
45
46    self._setup_group_tables(dp)
47
48    @set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
49    def _packet_in(self, ev):
50        msg = ev.msg
51        dp = msg.datapath
52        p = dp.ofproto_parser
53        in_port = msg.match['in_port']
54        pkt = packet.Packet(msg.data)
55        eth = pkt.get_protocols(ethernet.ethernet)[0]
56
57        if eth.ethertype == ether_types.ETH_TYPE_LLDP:
58            return
59
60        dpid = dp.id
61        self.mac_to_port.setdefault(dpid, {})
62        self.mac_to_port[dpid][eth.src] = in_port
63        out_port = self.mac_to_port[dpid].get(eth.dst, dp.ofproto.OFPP_FLOOD)
64
65        actions = [p.OFPActionOutput(out_port)]
66        if out_port != dp.ofproto.OFPP_FLOOD:
67            match = p.OFPMatch(in_port=in_port, eth_dst=eth.dst, eth_src=eth.src)
68            if msg.buffer_id != dp.ofproto.OFP_NO_BUFFER:
69                self._add_flow(dp, 1, match, actions, msg.buffer_id)
70                return
71            self._add_flow(dp, 1, match, actions)
72
73        data = None if msg.buffer_id != dp.ofproto.OFP_NO_BUFFER else msg.data
74        dp.send_msg(p.OFPPacketOut(datapath=dp, buffer_id=msg.buffer_id,
75                                  in_port=in_port, actions=actions, data=data))
```

Group - Loadbalance

```
1 from ryu.base import app_manager
2 from ryu.controller import ofp_event
3 from ryu.controller.handler import CONFIG_DISPATCHER, MAIN_DISPATCHER, set_ev_cls
4 from ryu.ofproto import ofproto_v1_3
5 from ryu.lib.packet import packet, ethernet, ether_types
6
7 class SimpleSwitch13(app_manager.RyuApp):
8     OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]
9
10     def __init__(self, *args, **kwargs):
11         super(SimpleSwitch13, self).__init__(*args, **kwargs)
12         self.mac_to_port = {}
13
14     def add_flow(self, dp, priority, match, actions, buffer_id=None):
15         inst = [dp.ofproto_parser.OFPInstructionActions(
16             dp.ofproto.OFPIT_APPLY_ACTIONS, actions)]
17         dp.send_msg(dp.ofproto_parser.OFPFlowMod(
18             datapath=dp, priority=priority, match=match,
19             instructions=inst, buffer_id=buffer_id if buffer_id
20             else dp.ofproto.OFP_NO_BUFFER))
21
22     def setup_lb(self, dp):
23         p = dp.ofproto_parser
24         # Create weighted group table (30% to port1, 70% to port2)
25         buckets = [
26             p.OFPBucket(30, dp.ofproto.OFPP_ANY, dp.ofproto.OFPQ_ALL,
27                 actions=[p.OFPActionOutput(1)]),
28             p.OFPBucket(70, dp.ofproto.OFPP_ANY, dp.ofproto.OFPQ_ALL,
29                 actions=[p.OFPActionOutput(2)])
30         ]
31
32         dp.send_msg(p.OFPGroupMod(dp, dp.ofproto.OFPGC_ADD,
33             dp.ofproto.OFPGT_SELECT, 50, buckets))
34
35         # Add flows for switch
36         self.add_flow(dp, 10, p.OFPMatch(in_port=3),
37             [p.OFPActionGroup(group_id=50)])
38
39         # Return flows from switches to host
40         for in_port in [1, 2]:
41             self.add_flow(dp, 10, p.OFPMatch(in_port=in_port),
42                 [p.OFPActionOutput(3)])
43
44         [p.OFPActionOutput(3)]
45
46     @set_ev_cls(ofp_event.EventOFPSwitchFeatures, CONFIG_DISPATCHER)
47     def switch_features(self, ev):
48         dp = ev.msg.datapath
49         p = dp.ofproto_parser
50         self.add_flow(dp, 0, p.OFPMatch(),
51             [p.OFPActionOutput(dp.ofproto.OFPP_CONTROLLER,
52                 dp.ofproto.OFPCML_NO_BUFFER)])
53
54         if dp.id in [1, 5]: # Setup load balancing for switches 1 and 5
55             self.setup_lb(dp)
56
57     @set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
58     def packet_in(self, ev):
59         msg = ev.msg
60         dp = msg.datapath
61         p = dp.ofproto_parser
62         in_port = msg.match['in_port']
63         pkt = packet.Packet(msg.data)
64         eth = pkt.get_protocols(ethernet.ethernet)[0]
65
66         if eth.ethertype == ether_types.ETH_TYPE_LLDP:
67             return
68
69         dpid = dp.id
70         self.mac_to_port.setdefault(dpid, {})
71         self.mac_to_port[dpid][eth.src] = in_port
72         out_port = self.mac_to_port[dpid].get(eth.dst, dp.ofproto.OFPP_FLOOD)
73
74         actions = [p.OFPActionOutput(out_port)]
75         if out_port != dp.ofproto.OFPP_FLOOD:
76             match = p.OFPMatch(in_port=in_port, eth_dst=eth.dst, eth_src=eth.src)
77             if msg.buffer_id != dp.ofproto.OFP_NO_BUFFER:
78                 self.add_flow(dp, 1, match, actions, msg.buffer_id)
79                 return
80             self.add_flow(dp, 1, match, actions)
81
82         data = None if msg.buffer_id != dp.ofproto.OFP_NO_BUFFER else msg.data
83         dp.send_msg(p.OFPPacketOut(datapath=dp, buffer_id=msg.buffer_id,
84             in_port=in_port, actions=actions, data=data))
```

Arp_proxy (เรียกใช้ arp dict python แทน)

```
from ryu.lib.packet import arp
```

```
arp_table = {"10.1.1.1": "00:00:00:00:00:01",
             "10.1.1.2": "00:00:00:00:00:02",
             "10.1.1.3": "00:00:00:00:00:03",
             "10.1.1.4": "00:00:00:00:00:04"
            }
```

```
def arp_process(self, datapath, eth, a, in_port):
    r = arp_table.get(a.dst_ip)
    if r:
        self.logger.info("Matched MAC %s ", r)
        arp_resp = packet.Packet()
        arp_resp.add_protocol(ethernet.ethernet(ethertype=eth.ethertype,
            dst=eth.src, src=r))
        arp_resp.add_protocol(arp.arp(opcode=arp.ARP_REPLY,
            src_mac=r, src_ip=a.dst_ip,
            dst_mac=a.src_mac,
            dst_ip=a.src_ip))
        arp_resp.serialize()
        actions = []
        actions.append(datapath.ofproto_parser.OFPActionOutput(in_port))
        parser = datapath.ofproto_parser
        ofproto = datapath.ofproto
        out = parser.OFPPacketOut(datapath=datapath, buffer_id=ofproto.OFP_NO_BUFFER,
            in_port=ofproto.OFPP_CONTROLLER, actions=actions, data=arp_resp)
        datapath.send_msg(out)
        self.logger.info("Proxied ARP Response packet")
```

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

# Check whether is it arp packet
if eth.ethertype == ether_types.ETH_TYPE_ARP:
    self.logger.info("Received ARP Packet %s %s %s ", dpid, src, dst)
    a = pkt.get_protocol(arp.arp)
    self.arp_process(datapath, eth, a, in_port)
    return

if dst in self.mac_to_port[dpid]:
    out_port = self.mac_to_port[dpid][dst]
```

เรียกใช้

Topodiscovery (ryu python)

1. Linear,4
2. ryu-manager --observe-links topology_discovery.py

```
from ryu.base import app_manager
from ryu.controller import ofp_event
from ryu.controller.handler import import CONFIG_DISPATCHER, MAIN_DISPATCHER, set_ev_cls
from ryu.ofproto import ofproto_v1_3
from ryu.lib.packet import packet, ethernet, ether_types
from ryu.topology.api import get_switch, get_link, get_host
from ryu.lib import hub

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 = {}
        self.topology_api_app = self
        hub.spawn(self._discover_topology)

    def _discover_topology(self):
        hub.sleep(10) # Wait for switches to connect
        switches = [switch.dpid for switch in get_switch(self, None)]
        links = [(link.src.dpid, link.dst.dpid, {'port': link.src.port_no}),
                 for link in get_link(self, None)]
        hosts = [(host.mac, host.port.dpid, {'port': host.port.port_no}),
                 for host in get_host(self, None)]
        self.logger.info("Topology Info - Switches: {switches}, "
                        "Links: {links}, Hosts: {hosts}")

    def _add_flow(self, dp, priority, match, actions, buffer_id=None):
        inst = [dp.ofproto_parser.OFPInstructionActions(
            dp.ofproto.OFPIT_APPLY_ACTIONS, actions)]
        dp.send_msg(dp.ofproto_parser.OFPFlowMod(
            datapath=dp, priority=priority, match=match,
            instructions=inst, buffer_id=buffer_id if buffer_id
            else dp.ofproto.OFP_NO_BUFFER))

    @set_ev_cls(ofp_event.EventOFPSwitchFeatures, CONFIG_DISPATCHER)
    def _switch_features(self, ev):
        dp = ev.msg.datapath
        self._add_flow(dp, 0, dp.ofproto_parser.OFPMatch(),
                       [dp.ofproto_parser.OFPActionOutput(
                           dp.ofproto.OFPP_CONTROLLER,
                           dp.ofproto.OFPCML_NO_BUFFER)])

    @set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
    def _packet_in(self, ev):
        msg = ev.msg
        dp = msg.datapath
        p = dp.ofproto_parser
        in_port = msg.match['in_port']
        pkt = packet.Packet(msg.data)
        eth = pkt.get_protocols(ethernet.ethernet)[0]

        if eth.ethertype == ether_types.ETH_TYPE_LLDP:
            return

        dpid = dp.id
        self.mac_to_port.setdefault(dpid, {})
        self.mac_to_port[dpid][eth.src] = in_port
        out_port = self.mac_to_port[dpid].get(eth.dst, dp.ofproto.OFPP_FLOOD)

        actions = [p.OFPActionOutput(out_port)]
        if out_port != dp.ofproto.OFPP_FLOOD:
            match = p.OFPMatch(in_port=in_port, eth_dst=eth.dst, eth_src=eth.src)
            if msg.buffer_id != dp.ofproto.OFP_NO_BUFFER:
                self._add_flow(dp, 1, match, actions, msg.buffer_id)
                return
            self._add_flow(dp, 1, match, actions)

        data = None if msg.buffer_id != dp.ofproto.OFP_NO_BUFFER else msg.data
        dp.send_msg(p.OFPActionOutput(datapath=dp, buffer_id=msg.buffer_id,
                                      in_port=in_port, actions=actions, data=data))
```

Multicontroller (ryu python) (equal role)

1. สร้าง Topo ด้วยโค้ดปกติ แต่ว่าใช้ controller 2 ที่

```
if __name__ == '__main__':
    setLogLevel('info')
    topo = MyTopo()
    net = Mininet(topo=topo, build=False)
    c0 = RemoteController('c0', ip='127.0.0.1', port=6653)
    c1 = RemoteController('c1', ip='127.0.0.1', port=6654)
    net.addController(c0)
    net.addController(c1)
    net.build()
    net.start()
    CLI(net)
    net.stop()
```

2. กำหนด Port ตอนรัน controller ryu-manager --ofp-tcp-listen-port 6554

Multicontroller (ryu python) (Master/Slave Role)

1. สร้าง Topo ด้วยโค้ดปกติ แต่ว่ามี controller 2 ที่
2. Master

```
from ryu.base import app_manager
from ryu.controller import ofp_event
from ryu.controller.handler import CONFIG_DISPATCHER, MAIN_DISPATCHER, set_ev_cls
from ryu.ofproto import ofproto_v1_3
from ryu.lib.packet import packet, ethernet, ether_types, ipv4

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

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

    def _send_role_request(self, dp):
        dp.send_msg(dp.ofproto_parser.OFPRoleRequest(dp, dp.ofproto.OFPCR_ROLE_MASTER, 0))

    @set_ev_cls(ofp_event.EventOFPRoleReply, MAIN_DISPATCHER)
    def _role_reply(self, ev):
        if ev.msg.role == ev.msg.datapath.ofproto.OFPCR_ROLE_MASTER:
            self.logger.info('Controller is master')

    @set_ev_cls(ofp_event.EventOFPSwitchFeatures, CONFIG_DISPATCHER)
    def _switch_features(self, ev):
        dp = ev.msg.datapath
        self._send_role_request(dp)
        self._add_flow(dp, 0, dp.ofproto_parser.OFPMatch(),
                       [dp.ofproto_parser.OFPActionOutput(
                           dp.ofproto.OFPP_CONTROLLER,
                           dp.ofproto.OFPCML_NO_BUFFER)])

    def _add_flow(self, dp, priority, match, actions, buffer_id=None):
        inst = [dp.ofproto_parser.OFPInstructionActions(
            dp.ofproto.OFPIIT_APPLY_ACTIONS, actions)]
        dp.send_msg(dp.ofproto_parser.OFPFlowMod(
            datapath=dp, priority=priority, match=match,
            instructions=inst, buffer_id=buffer_id if buffer_id
            else dp.ofproto.OFP_NO_BUFFER))
```

3. Slave

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

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

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

    def _send_role_request(self, dp):
        dp.send_msg(dp.ofproto_parser.OFPRoleRequest(dp, dp.ofproto.OFPCR_ROLE_SLAVE, 0))

    @set_ev_cls(ofp_event.EventOFPRoleReply, MAIN_DISPATCHER)
    def _role_reply(self, ev):
        if ev.msg.role == ev.msg.datapath.ofproto.OFPCR_ROLE_SLAVE:
            self.logger.info('Controller is slave')
```