ที่ต้องติดตั้งกรณีที่อาจารย์ไม่ลงมาให้

sudo apt update
sudo apt install python3 python3-pip xterm iperf hping3 net-tools wireshark apache2-utils curl sudo apt install mininet
sudo pip3 install ryu
sudo pip3 install mininet
sudo cp /usr/bin/python3 /usr/bin/python ryu-manager --version sudo mn --version

ข้อ 1 สร้าง topology ให้เป็นโดยใช้ python

ขั้นตอน 1 หาตำแหน่งไฟล์ที่จารย์เตรียมไว้ให้เจอ (ถ้าจารย์เตรียมให้น่าจะอยู่ path นี้)

```
test@test:~$ cd SDN_Beginners_with_RYU/part2/ofctl/lb/
```

-หากมี ให้เปิด Editor ตัวไหนก้ได้แล้วไปดู *ขั้นตอน 2*

-หากไม่มี ให้ไป Desktop แล้วสร้าง folder มาเก็บโค้ด

cd ~ เพื่อกลับมา home

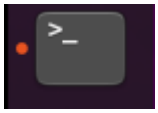
cd Desktop/ เพื่อไป Desktop

mkdir midterm650 เพื่อสร้าง folder

cd midterm650 เพื่อไป midterm650

nano topo.py เพื่อเขียนไฟล์ .py (เมื่อจบ ctrl o, enter, ctrl x)

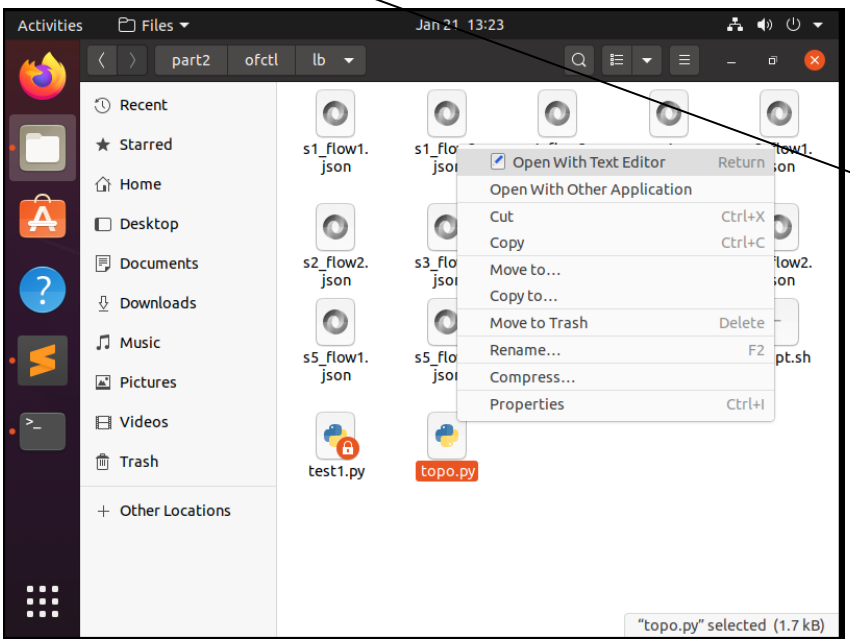
File Explorer



Terminal

Editor

All Apps (Search)



การเปิดไฟล์ด้วย editor แอพ

ไม่ก็ถ้าจารย์เขาเตรียมให้แค่เปิด

อันนี้ก็เจอเลย

ขั้นตอน 2 การเขียนโค้ดไฟล์ topo.py

```
#!/usr/bin/python


"""Grouptable example

          1   Switch2 ----switch4
         /                       \
h1 ---Switch1                     Switch5-----h2
   3      \                       /
           --------Switch3 ------
                   2


# static arp entry addition

h1 arp -s 192.168.1.2 00:00:00:00:00:02
h2 arp -s 192.168.1.1 00:00:00:00:00:01



ryu stuff:

ryu-manager group_table_lb.py

"""

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
from time import sleep


class SingleSwitchTopo(Topo):
    "Single switch connected to n hosts."
    def build(self):
        s1 = self.addSwitch('s1', protocols='OpenFlow13')
        s2 = self.addSwitch('s2', protocols='OpenFlow13')
        s3 = self.addSwitch('s3', protocols='OpenFlow13')
        s4 = self.addSwitch('s4', protocols='OpenFlow13')
        s5 = self.addSwitch('s5', protocols='OpenFlow13')

        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")

        self.addLink(s1,s2,1,1)
        self.addLink(s1,s3,2,1)
        self.addLink(s1,h1,3,1)


        self.addLink(s3,s5,2,2)
        self.addLink(s4,s2,1,2)
        self.addLink(s4,s5,2,1)

        self.addLink(s5,h2,3,1)

if __name__ == '__main__':
    setLogLevel('info')
    topo = SingleSwitchTopo()
    c1 = RemoteController('c1', ip='127.0.0.1')
    net = Mininet(topo=topo, controller=c1)
    net.start()
    #sleep(5)
    #print("Topology is up, lets ping")
    #net.pingAll()
    CLI(net)
    net.stop()
```

นี่คือหน้าตา topo ของโค้ดนี้

คำสั่งเพิ่ม switch ลงใน topo

คำสั่งเพิ่ม pc ลงใน topo

คำสั่งเชื่อมสาย

(อุปกรณ์1,อุปกรณ์2,portของอุปกรณ์1,portของอุปกรณ์2)

s1 port1 เชื่อม s2 port1 (เลข 1 ตรง topo ด้านบน)

s1 port2 เชื่อม s3 port1

ส่วนอื่นของโค้ดไม่ต้องแก้

หากแก้ใน editor แบบแอพ ก้ ctrl s เพื่อ save

หาก nano ก็ ctrl o, enter, ctrl x

ข้อ 2 ใช้ topo ข้อแรกมาใส่ flow และ group

ขั้นตอน 1 เปิด terminal 3 แท็บ (ปุ่มเพิ่มแท็บอยู่ซ้ายบน)

ขั้นตอน 2 tab1 ใช้งาน topo.py (ต้องเข้าไป path ที่ไฟล์อยู่) ผ่านคำสั่ง sudo python topo.py

```
test@test: ~/SDN_Be...   test@test: ~   test@test: ~
test@test:~/SDN_Beginners_with_RYU/part2/ofctl/lb$ sudo python topo.py
[sudo] password for test:
Unable to contact the remote controller at 127.0.0.1:6653
Unable to contact the remote controller at 127.0.0.1:6633
Setting remote controller to 127.0.0.1:6653
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1 s2 s3 s4 s5
*** Adding links:
(s1, h1) (s1, s2) (s1, s3)
*** Configuring hosts
h1 h2
*** Starting controller
c1
*** Starting 5 switches
s1 s2 s3 s4 s5 ...
*** Starting CLI:
mininet>
```

ข้อควรรู้: (ถ้าเป็น vm ตามนี้ user:test, password:test)

-หากต้องการปิด tab ของ topo ให้พิม mininet>quit

-หากต้องการปิด tab ของ ryu ให้ Ctrl c

-ขั้นตอน 2.5 คิดว่าให้ทำใส่ให้ครบรอบตัวถ้ามี host เพิ่มเช่น

h1 arp ใส่ h2 และ h3

h2 arp ใส่ h1 และ h3

h3 arp ใส่ h1 และ h2

ขั้นตอน 2.5

```
mininet> h1 arp -s 192.168.1.2 00:00:00:00:00:02
mininet> h2 arp -s 192.168.1.1 00:00:00:00:00:01
```

ขั้นตอน 3 tab2 ใช้งาน ryu ผ่านคำสั่ง ryu-manager ryu.app.ofctl_rest

```
test@test: ~/SDN_Be...   test@test: ~   test@test: ~
test@test:~$ ryu-manager ryu.app.ofctl_rest
loading app ryu.app.ofctl_rest
loading app ryu.controller.ofp_handler
instantiating app None of DPSet
creating context dpset
creating context wsgi
instantiating app ryu.app.ofctl_rest of RestStatsApi
instantiating app ryu.controller.ofp_handler of OFPHandler
(4558) wsgi starting up on http://0.0.0.0:8080
```

ระวังตรงนี้ให้ดี

ขั้นตอน 4 tab3 ใส่ flow และ group ต่างๆที่จะทำให้ switch ใน topo มันรู้เรื่อง

Add flow ผ่านคำสั่ง curl -X POST http://localhost:8080/stats/flowentry/add -d '@ชื่อโฟลว์.json'

Add group ผ่านคำสั่ง curl -X POST http://localhost:8080/stats/groupentry/add -d '@ชื่อกรุ๊ป.json'

ไปหา path ที่ไฟล์ .json พวกนั้นอยู่ด้วย (tips: ถ้าสังเกตุดูดีๆ sw แต่ละตัวมี flow เท่าจำนวน port)

```
test@test: ~/SDN_Be...   test@test: ~   test@test: ~/SDN_Be...
test@test:~/SDN_Beginners_with_RYU/part2/ofctl/lb$ curl -X POST http://localhost:8080/stats/flowentry/
add -d '@s2_flow1.json'
test@test:~/SDN_Beginners_with_RYU/part2/ofctl/lb$ curl -X POST http://localhost:8080/stats/flowentry/
add -d '@s2_flow2.json'
test@test:~/SDN_Beginners_with_RYU/part2/ofctl/lb$ curl -X POST http://localhost:8080/stats/flowentry/
add -d '@s3_flow1.json'
test@test:~/SDN_Beginners_with_RYU/part2/ofctl/lb$ curl -X POST http://localhost:8080/stats/flowentry/
add -d '@s3_flow2.json'
test@test:~/SDN_Beginners_with_RYU/part2/ofctl/lb$ curl -X POST http://localhost:8080/stats/flowentry/
add -d '@s4_flow1.json'
test@test:~/SDN_Beginners_with_RYU/part2/ofctl/lb$ curl -X POST http://localhost:8080/stats/flowentry/
add -d '@s4_flow2.json'
test@test:~/SDN_Beginners_with_RYU/part2/ofctl/lb$ curl -X POST  http://localhost:8080/stats/groupentr
y/add -d '@s5_group51.json'
test@test:~/SDN_Beginners_with_RYU/part2/ofctl/lb$ curl -X POST  http://localhost:8080/stats/groupentr
y/add -d '@s1_group50.json'
test@test:~/SDN_Beginners_with_RYU/part2/ofctl/lb$ curl -X POST  http://localhost:8080/stats/flowentry
/add -d '@s1_flow1.json'
test@test:~/SDN_Beginners_with_RYU/part2/ofctl/lb$ curl -X POST  http://localhost:8080/stats/flowentry
/add -d '@s1_flow2.json'
test@test:~/SDN_Beginners_with_RYU/part2/ofctl/lb$ curl -X POST  http://localhost:8080/stats/flowentry
/add -d '@s1_flow3.json'
test@test:~/SDN_Beginners_with_RYU/part2/ofctl/lb$ curl -X POST  http://localhost:8080/stats/flowentry
/add -d '@s5_flow1.json'
test@test:~/SDN_Beginners_with_RYU/part2/ofctl/lb$ curl -X POST  http://localhost:8080/stats/flowentry
/add -d '@s5_flow2.json'
test@test:~/SDN_Beginners_with_RYU/part2/ofctl/lb$ curl -X POST  http://localhost:8080/stats/flowentry
/add -d '@s5_flow3.json'
test@test:~/SDN_Beginners_with_RYU/part2/ofctl/lb$
```

เป็นอันเสร็จขั้นตอนที่นี้ pc ก็สามารถปิงกันได้ตามที่เราเขียนใน .json มาดูตัวอย่าง .json

s1_flow1.json

```
{
    "dpid": 1,
    "table_id": 0,
    "idle_timeout": 0,
    "hard_timeout": 0,
    "priority": 100,
    "match":{
        "in_port":1
    },
    "actions":[
        {
            "type":"OUTPUT",
            "port": 3
        }
    ]
}
```

dpid คือเลขอุปกรณ์switch ในที่นี้คือ sw1

tableid คือหมายเลข table

match คือ packet ตรงกับอะไรไหมถ้าตรงให้ใส่

priority เป็น 100 (หรือตามที่จารย์กำหนด)

match ในที่นี้แปลได้ว่า packet ที่วิ่งเข้า port 1

actions คือการกระทำ ในที่นี้แปลได้ว่า

ให้ packet นั้นๆวิ่งออก port 3

หากสงสัยว่าทำไมต้องเขียนโฟลว์แบบนี้

sw1 เข้าขา 1 ออกขา 3 ให้ย้อนไปดูหน้า topo

ว่าถ้าออกขา 1 หรือขา 2 ตัว packet มันก็จะวน

s1_flow3.json

```
{
    "dpid": 1,
    "table_id": 0,
    "idle_timeout": 0,
    "hard_timeout": 0,
    "priority": 100,
    "match":{
        "in_port":3
    },
    "actions":[
        {
            "type":"GROUP",
            "group_id": 50
        }
    ]
}
```

เมื่อเข้าขา 3 ตัว packet จะถือว่าเป็น group 50

ส่วนโค้ดขวาคือ จับตัว packet ที่เป็นgroup 50

ให้โอกาส 50% ไปออก port1 อีก 50% ออก port2

ส่วนโค้ดล่างลองดู dpid และการเข้าออกเป็น

กรณีศึกษา เทียบกับ s1_flow1.json

s1_group50.json

```
{
    "dpid": 1,
    "type": "SELECT",
    "group_id": 50,
    "buckets": [
        {
            "weight": 50,
            "actions": [
                {
                    "type": "OUTPUT",
                    "port": 1
                }
            ]
        },
        {
            "weight": 50,
            "actions": [
                {
                    "type": "OUTPUT",
                    "port": 2
                }
            ]
        }
    ]
}
```

s2_flow1.json

```
{
    "dpid": 2,
    "table_id": 0,
    "idle_timeout": 0,
    "hard_timeout": 0,
    "priority": 100,
    "match":{
        "in_port":1
    },
    "actions":[
        {
            "type":"OUTPUT",
            "port": 2
        }
    ]
}
```

คำสั่งเช็ค flow

sudo ovs-ofctl -O OpenFlow13 dump-flows s1

คำสั่งเช็ค group

sudo ovs-ofctl -O OpenFlow13 dump-groups s1

อยากดูตัวอื่นก็ s2,s3,s4

คำสั่งนี้ไว้พิมใน tab อื่นๆที่ไม่ใช่ topo กับ ryu

ข้อ 3 ทำ multitble และกำหนด policy(จริงๆแล้วแค่ block การปิงของ pc)

ขั้นตอน 1 สร้าง topo ในที่นี้เป็น 1 switch, 4 pc

```
test@test: ~    test@test: ~    test@test: ~/SDN_Be...
test@test:~$ sudo mn --controller=remote,ip=127.0.0.1 --mac --switch=ovsk,protocols=OpenFlow13 --topo=single,4
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Unable to contact the remote controller at 127.0.0.1:6633
Setting remote controller to 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X X
h2 -> X X X
h3 -> X X X
h4 -> X X X
*** Results: 100% dropped (0/12 received)
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)
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 X
h2 -> h1 h3 X
h3 -> h1 h2 X
h4 -> X X X
*** Results: 50% dropped (6/12 received)
mininet>
```

สร้าง topo

ลองปิงแบบยังไม่ add flow

ลองปิงแบบ add flow

ลองปิงแบบ add policy

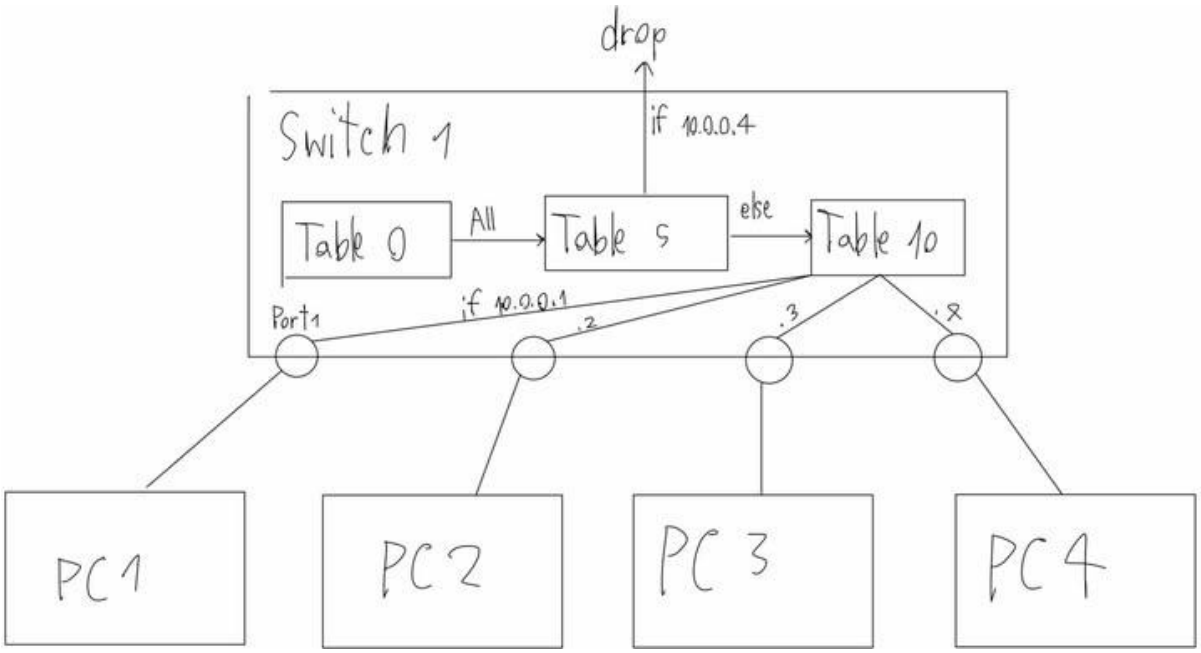
ขั้นตอน 2 ใช้งาน ryu คำสั่งเดิม

```
test@test: ~    test@test: ~    test@test: ~/SDN_Be...
test@test:~$ ryu-manager ryu.app.ofctl_rest
loading app ryu.app.ofctl_rest
loading app ryu.controller.ofp_handler
instantiating app None of DPSet
creating context dpset
creating context wsgi
instantiating app ryu.app.ofctl_rest of RestStatsApi
instantiating app ryu.controller.ofp_handler of OFPHandler
(5827) wsgi starting up on http://0.0.0.0:8080
```

ขั้นตอน 3 ใส่ flow, table (7 flow แรกเป็น flow ปกติ, ส่วน flow อันเดี่ยวๆคือ policy)

```
test@test: ~          test@test: ~          test@test: ~/SDN_Be...
test@test:~/SDN_Beginners_with_RYU/part2/ofctl/multitable$ curl -X POST  http://localhost:8080/stats/flowentry/add -d '@table0_flow1.json'
test@test:~/SDN_Beginners_with_RYU/part2/ofctl/multitable$ curl -X POST  http://localhost:8080/stats/flowentry/add -d '@table5_flow1.json'
test@test:~/SDN_Beginners_with_RYU/part2/ofctl/multitable$ curl -X POST  http://localhost:8080/stats/flowentry/add -d '@table10_flow1.json'
test@test:~/SDN_Beginners_with_RYU/part2/ofctl/multitable$ curl -X POST  http://localhost:8080/stats/flowentry/add -d '@table10_flow2.json'
test@test:~/SDN_Beginners_with_RYU/part2/ofctl/multitable$ curl -X POST  http://localhost:8080/stats/flowentry/add -d '@table10_flow3.json'
test@test:~/SDN_Beginners_with_RYU/part2/ofctl/multitable$ curl -X POST  http://localhost:8080/stats/flowentry/add -d '@table10_flow4.json'
test@test:~/SDN_Beginners_with_RYU/part2/ofctl/multitable$ curl -X POST  http://localhost:8080/stats/flowentry/add -d '@table10_arp.json'
test@test:~/SDN_Beginners_with_RYU/part2/ofctl/multitable$
test@test:~/SDN_Beginners_with_RYU/part2/ofctl/multitable$
test@test:~/SDN_Beginners_with_RYU/part2/ofctl/multitable$ curl -X POST  http://localhost:8080/stats/flowentry/add -d '@table5_flow2.json'
test@test:~/SDN_Beginners_with_RYU/part2/ofctl/multitable$
```

เป็นอันเสร็จขั้นตอนที่นี้ pc ก็สามารถปิงกันได้ตามที่เราเขียนใน .json (ปิงกันได้หมดยกเว้น pc4) มาดูตัวอย่าง .json



ข้างบนคือแนวคิด ส่วนข้างล่างคือ table ที่เป็น .json ต่างๆ

table0_flow1.json

```
{
    "dpid": 1,
    "table_id": 0,
    "idle_timeout": 0,
    "hard_timeout": 0,
    "priority": 0,
    "match":{
    },
    "actions":[
        {
            "type":"GOTO_TABLE",
            "table_id": 5
        }
    ]
}
```

table5_flow1.json

```
{
    "dpid": 1,
    "table_id": 5,
    "idle_timeout": 0,
    "hard_timeout": 0,
    "priority": 0,
    "match":{
    },
    "actions":[
        {
            "type":"GOTO_TABLE",
            "table_id": 10
        }
    ]
}
```

ถ้าไม่ต้อง match กับอะไรเลยเราจะกำหนดให้ priority

ในภาพนี้ก็บอกว่าไป table 10 เลย

เป็น 0 ไม่ต้อง match กับอะไรเลยมีความหมายว่า

ทุกๆ packet

ในภาพนี้คือทุก packet เข้ามาแล้วไป table 5 ต่อเลย

สังเกตุ table_id ใต้ dpid เลขเปลี่ยนตาม

หมายเลข table

table5_flow2.json

```
{
    "dpid": 1,
    "table_id": 5,
    "idle_timeout": 0,
    "hard_timeout": 0,
    "priority": 100,
    "match":{
        "eth_type": 2048,
        "ipv4_dst": "10.0.0.4",
        "ip_proto": 1
    },
    "actions":[
    ]
}
```

หากเจอ 10.0.0.4 ให้ drop

สังเกต priority 100 เพราะมีการ match

actions หน้าตาแบบนี้คือ drop

table10_flow1.json

```
{
    "dpid": 1,
    "table_id": 10,
    "idle_timeout": 0,
    "hard_timeout": 0,
    "priority": 0,
    "match":{
    "dl_dst": "00:00:00:00:00:01"
    },
    "actions":[
        {
            "type":"OUTPUT",
            "port": 1
        }
    ]
}
```

หากเจอ mac เบอร์นี้ให้ออก port นี้

2,3,4 ก็ประมาณนี้

table10_arp.json

```
{
    "dpid": 1,
    "table_id": 10,
    "idle_timeout": 0,
    "hard_timeout": 0,
    "priority": 0,
    "match":{
    "dl_dst": "ff:ff:ff:ff:ff:ff"
    },
    "actions":[
        {
            "type":"OUTPUT",
            "port": 4294967291
        }
    ]
}
```

อันนี้คือการ flood ทุกเบอร์

ตรง actions เลขนั้นก็หมายถึง flood

BIGKUMA ปรับปรุง

สร้าง topo มี3แบบ Single กับ Linear กับ Tree

Terminal ที่ 1 สร้าง Topology

อันนี้คือคำสั่งสร้าง topo ด้วย mininet แบบไม่กำหนด IP เริ่มต้นเป็น 10.0.0.1/8

Single

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

Linear

sudo mn --controller=remote,ip=127.0.0.1 --mac --swicth=ovsk,protocols=OpenFlow13 --topo=linear,4

Tree

sudo mn --controller=remote,ip=127.0.0.1 --mac -i 10.1.1.0/24 --topo=tree,depth=2,fanout=3

อันนี้คือคำสั่งสร้าง topo ด้วย mininet แบบกำหนด IP

sudo mn --controller=remote,ip=127.0.0.1 --mac -i 10.1.1.0/24 --switch=ovsk,protocols=OpenFlow13 --topo=linear,4

อันนี้คือคำสั่งรัน topo ที่สร้างจาก pythonเข้าไปใน pathก่อน

Sudo python test.py(ชื่อไฟล์)

Terminal ที่ 2 รันสมอง (ryu)

ryu-manager ryu.app.simple_switch_13 แบบปกคิดว่าไม่น่าใช้

ryu-manager ryu.app.ofctl_rest แบบกำหนด Flow เอง

Terminal ที่ 3 ใส่ Flow และ Group เพื่อกำหนดการส่งข้อมูล

ก่อนใช้คำสั่ง curl ต้องเข้าไปใน path ของไฟล์ .json ก่อน ซึ่ง Flow และ Group ต้องสร้างเองให้ตรงตามโจทย์

curl -X POST http://localhost:8080/stats/flowentry/add -d ‘@ชื่อไฟล์.json’

curl -X POST http://localhost:8080/stats/flowentry/add -d ‘@table20_flow1.json’

curl -X POST http://localhost:8080/stats/groupentry/add -d ‘@ชื่อไฟล์.json’

curl -X POST http://localhost:8080/stats/groupentry/add -d ‘@s5_group51.json’

Terminal ที่ 4 ใส่ Flow และ Group เพื่อกำหนดการส่งข้อมูล

ดู Flow ของ Traffic

sudo ovs-ofctl -O OpenFlow13 dump-flows ชื่อswitch

sudo ovs-ofctl -O OpenFlow13 dump-flows s1

ดู Group ของ Traffic

sudo ovs-ofctl -O OpenFlow13 dump-groups ชื่อswitch

sudo ovs-ofctl -O OpenFlow13 dump-groups s1

Terminal ที่ 1(mininet) คำสั่งทดสอบระบบว่า topo ของเราใช้งาน Flow กับ Group ได้ไหม

pingall ทุก host ping หากัน

h1 ping h2 ทดสอบการping ระหว่างhost

iperf h1 h2 ทดสอบ TCP

links ดูการต่อสายของ topo ว่าต่อกันแบบไหน

nodes ดูว่า มี node อะไรบ้าง

Multi Table ต่อไปของอาจารย์อธิบายละเอียด

ใช้กับ topo นี้

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

```
table0_flow1
{
    "dpid": 1,  // Datapath ID: ระบุสวิตช์ OpenFlow ที่ใช้กฎนี้ (สวิตช์หมายเลข 1)
    "table_id": 0,  // ตารางที่กฎนี้จะถูกใช้ (ตารางหมายเลข 0 คือตารางเริ่มต้น)
    "idle_timeout": 0,  // เวลาที่กฎจะถูกลบหากไม่มีการใช้งาน (0 = ไม่ถูกลบ)
    "hard_timeout": 0,  // เวลาที่กฎจะถูกลบไม่ว่าจะมีการใช้งานหรือไม่ (0 = ไม่ถูกลบ)
    "priority": 0,  // ลำดับความสำคัญของกฎ (0 = ความสำคัญต่ำสุด)
    "match": {},  // เงื่อนไขที่แพ็กเก็ตต้องตรงกับกฎนี้ ({} = ตรงกับทุกแพ็กเก็ต)
    "actions": [  // การกระทำที่สวิตช์จะทำเมื่อแพ็กเก็ตตรงกับกฎนี้
        {
            "type": "GOTO_TABLE",  // ประเภทการกระทำ: ส่งแพ็กเก็ตไปยังตารางอื่นเพื่อประมวลผลต่อไป
            "table_id": 5  // ตารางที่แพ็กเก็ตจะถูกส่งไป (ตารางหมายเลข 5)
        }
    ]
}
```

```
table5_flow1
{
    "dpid": 1,  // Datapath ID: ระบุสวิตช์ OpenFlow ที่ใช้กฎนี้ (สวิตช์หมายเลข 1)
    "table_id": 5,  // ตารางที่กฎนี้จะถูกใช้ (ตารางหมายเลข 5)
    "idle_timeout": 0,  // เวลาที่กฎจะถูกลบหากไม่มีการใช้งาน (0 = ไม่ถูกลบ)
    "hard_timeout": 0,  // เวลาที่กฎจะถูกลบไม่ว่าจะมีการใช้งานหรือไม่ (0 = ไม่ถูกลบ)
    "priority": 0,  // ลำดับความสำคัญของกฎ (0 = ความสำคัญต่ำสุด)
    "match": {},  // เงื่อนไขที่แพ็กเก็ตต้องตรงกับกฎนี้ ({} = ตรงกับทุกแพ็กเก็ต)
    "actions": [  // การกระทำที่สวิตช์จะทำเมื่อแพ็กเก็ตตรงกับกฎนี้
        {
            "type": "GOTO_TABLE",  // ประเภทการกระทำ: ส่งแพ็กเก็ตไปยังตารางอื่นเพื่อประมวลผลต่อไป
            "table_id": 10  // ตารางที่แพ็กเก็ตจะถูกส่งไป (ตารางหมายเลข 10)
        }
    ]
}
```

```
table5_flow2
{
    "dpid": 1,  // Datapath ID: ระบุสวิตช์ OpenFlow ที่ใช้กฎนี้ (สวิตช์หมายเลข 1)
    "table_id": 5,  // ตารางที่กฎนี้จะถูกใช้ (ตารางหมายเลข 5)
    "idle_timeout": 0,  // เวลาที่กฎจะถูกลบหากไม่มีการใช้งาน (0 = ไม่ถูกลบ)
    "hard_timeout": 0,  // เวลาที่กฎจะถูกลบไม่ว่าจะมีการใช้งานหรือไม่ (0 = ไม่ถูกลบ)
    "priority": 100,  // ลำดับความสำคัญของกฎ (100 = มีความสำคัญสูง)
    "match": {  // เงื่อนไขที่แพ็กเก็ตต้องตรงกับกฎนี้
        "eth_type": 2048,  // ตรวจสอบว่า Ethernet Type เป็น IPv4 (ค่า 2048 = IPv4)
        "ipv4_dst": "10.0.0.4",  // ตรวจสอบว่า IP ปลายทาง (Destination IP) เป็น 10.0.0.4
        "ip_proto": 1  // ตรวจสอบว่าโปรโตคอลในเลเยอร์ IP เป็น ICMP (ค่า 1 = ICMP)
    },
    "actions": []  // การกระทำที่สวิตช์จะทำเมื่อแพ็กเก็ตตรงกับกฎนี้ (ในที่นี้ไม่มี action ใดๆ)
}
```

```
table10_arp
{
    "dpid": 1,  // Datapath ID: ระบุสวิตช์ OpenFlow ที่ใช้กฎนี้ (สวิตช์หมายเลข 1)
    "table_id": 10,  // ตารางที่กฎนี้จะถูกใช้ (ตารางหมายเลข 10)
    "idle_timeout": 0,  // เวลาที่กฎจะถูกลบหากไม่มีการใช้งาน (0 = ไม่ถูกลบ)
    "hard_timeout": 0,  // เวลาที่กฎจะถูกลบไม่ว่าจะมีการใช้งานหรือไม่ (0 = ไม่ถูกลบ)
    "priority": 0,  // ลำดับความสำคัญของกฎ (0 = ความสำคัญต่ำสุด)
    "match": {  // เงื่อนไขที่แพ็กเก็ตต้องตรงกับกฎนี้
        "dl_dst": "ff:ff:ff:ff:ff:ff"  // ตรวจสอบว่า MAC Address ปลายทาง (Destination MAC) เป็น Broadcast Address
    },
    "actions": [  // การกระทำที่สวิตช์จะทำเมื่อแพ็กเก็ตตรงกับกฎนี้
        {
            "type": "OUTPUT",  // ประเภทการกระทำ: ส่งแพ็กเก็ตออกทางพอร์ตที่ระบุ
            "port": 4294967291  // พอร์ตที่แพ็กเก็ตจะถูกส่งออก (พอร์ตหมายเลข 4294967291 คือพอร์ต "CONTROLLER")
        }
    ]
}
```

```
table10_flow1
{
    "dpid": 1,  // Datapath ID: ระบุสวิตช์ OpenFlow ที่ใช้กฎนี้ (สวิตช์หมายเลข 1)
    "table_id": 10,  // ตารางที่กฎนี้จะถูกใช้ (ตารางหมายเลข 10)
    "idle_timeout": 0,  // เวลาที่กฎจะถูกลบหากไม่มีการใช้งาน (0 = ไม่ถูกลบ)
    "hard_timeout": 0,  // เวลาที่กฎจะถูกลบไม่ว่าจะมีการใช้งานหรือไม่ (0 = ไม่ถูกลบ)
    "priority": 0,  // ลำดับความสำคัญของกฎ (0 = ความสำคัญต่ำสุด)
    "match": {  // เงื่อนไขที่แพ็กเก็ตต้องตรงกับกฎนี้
        "dl_dst": "00:00:00:00:00:01"  // ตรวจสอบว่า MAC Address ปลายทาง (Destination MAC) เป็น 00:00:00:00:00:01
    },
    "actions": [  // การกระทำที่สวิตช์จะทำเมื่อแพ็กเก็ตตรงกับกฎนี้
        {
            "type": "OUTPUT",  // ประเภทการกระทำ: ส่งแพ็กเก็ตออกทางพอร์ตที่ระบุ
            "port": 1  // พอร์ตที่แพ็กเก็ตจะถูกส่งออก (พอร์ตหมายเลข 1)
        }
    ]
}
```

```
table10_flow2
{
    "dpid": 1,  // Datapath ID: ระบุสวิตช์ OpenFlow ที่ใช้กฎนี้ (สวิตช์หมายเลข 1)
    "table_id": 10,  // ตารางที่กฎนี้จะถูกใช้ (ตารางหมายเลข 10)
    "idle_timeout": 0,  // เวลาที่กฎจะถูกลบหากไม่มีการใช้งาน (0 = ไม่ถูกลบ)
    "hard_timeout": 0,  // เวลาที่กฎจะถูกลบไม่ว่าจะมีการใช้งานหรือไม่ (0 = ไม่ถูกลบ)
    "priority": 0,  // ลำดับความสำคัญของกฎ (0 = ความสำคัญต่ำสุด)
    "match": {  // เงื่อนไขที่แพ็กเก็ตต้องตรงกับกฎนี้
        "dl_dst": "00:00:00:00:00:02"  // ตรวจสอบว่า MAC Address ปลายทาง (Destination MAC) เป็น 00:00:00:00:00:02
    },
    "actions": [  // การกระทำที่สวิตช์จะทำเมื่อแพ็กเก็ตตรงกับกฎนี้
        {
            "type": "OUTPUT",  // ประเภทการกระทำ: ส่งแพ็กเก็ตออกทางพอร์ตที่ระบุ
            "port": 2  // พอร์ตที่แพ็กเก็ตจะถูกส่งออก (พอร์ตหมายเลข 2)
        }
    ]
}
```

```
table10_flow3
{
    "dpid": 1,  // Datapath ID: ระบุสวิตช์ OpenFlow ที่ใช้กฎนี้ (สวิตช์หมายเลข 1)
    "table_id": 10,  // ตารางที่กฎนี้จะถูกใช้ (ตารางหมายเลข 10)
    "idle_timeout": 0,  // เวลาที่กฎจะถูกลบหากไม่มีการใช้งาน (0 = ไม่ถูกลบ)
    "hard_timeout": 0,  // เวลาที่กฎจะถูกลบไม่ว่าจะมีการใช้งานหรือไม่ (0 = ไม่ถูกลบ)
    "priority": 0,  // ลำดับความสำคัญของกฎ (0 = ความสำคัญต่ำสุด)
    "match": {  // เงื่อนไขที่แพ็กเก็ตต้องตรงกับกฎนี้
        "dl_dst": "00:00:00:00:00:03"  // ตรวจสอบว่า MAC Address ปลายทาง (Destination MAC) เป็น 00:00:00:00:00:03
    },
    "actions": [  // การกระทำที่สวิตช์จะทำเมื่อแพ็กเก็ตตรงกับกฎนี้
        {
            "type": "OUTPUT",  // ประเภทการกระทำ: ส่งแพ็กเก็ตออกทางพอร์ตที่ระบุ
            "port": 3  // พอร์ตที่แพ็กเก็ตจะถูกส่งออก (พอร์ตหมายเลข 3)
        }
    ]
}
```

```
table10_flow4
{
    "dpid": 1,  // Datapath ID: ระบุสวิตช์ OpenFlow ที่ใช้กฎนี้ (สวิตช์หมายเลข 1)
    "table_id": 10,  // ตารางที่กฎนี้จะถูกใช้ (ตารางหมายเลข 10)
    "idle_timeout": 0,  // เวลาที่กฎจะถูกลบหากไม่มีการใช้งาน (0 = ไม่ถูกลบ)
    "hard_timeout": 0,  // เวลาที่กฎจะถูกลบไม่ว่าจะมีการใช้งานหรือไม่ (0 = ไม่ถูกลบ)
    "priority": 0,  // ลำดับความสำคัญของกฎ (0 = ความสำคัญต่ำสุด)
    "match": {  // เงื่อนไขที่แพ็กเก็ตต้องตรงกับกฎนี้
        "dl_dst": "00:00:00:00:00:04"  // ตรวจสอบว่า MAC Address ปลายทาง (Destination MAC) เป็น 00:00:00:00:00:04
    },
    "actions": [  // การกระทำที่สวิตช์จะทำเมื่อแพ็กเก็ตตรงกับกฎนี้
        {
            "type": "OUTPUT",  // ประเภทการกระทำ: ส่งแพ็กเก็ตออกทางพอร์ตที่ระบุ
            "port": 4  // พอร์ตที่แพ็กเก็ตจะถูกส่งออก (พอร์ตหมายเลข 4)
        }
    ]
}
```

Flow ต่อไปของอาจารย์อธิบายละเอียด

ใช้กับ topo นี้

Topo.py ตามหน้าที่ 2 เปิดดู อันนี้จะบอกวิธีการใช้ Group

```
s1_flow1
{
    "dpid": 1,  // Datapath ID: ระบุสวิตช์ OpenFlow ที่ใช้กฎนี้ (สวิตช์หมายเลข 1)
    "table_id": 0,  // ตารางที่กฎนี้จะถูกใช้ (ตารางหมายเลข 0 คือตารางเริ่มต้น)
    "idle_timeout": 0,  // เวลาที่กฎจะถูกลบหากไม่มีการใช้งาน (0 = ไม่ถูกลบ)
    "hard_timeout": 0,  // เวลาที่กฎจะถูกลบไม่ว่าจะมีการใช้งานหรือไม่ (0 = ไม่ถูกลบ)
    "priority": 100,  // ลำดับความสำคัญของกฎ (100 = มีความสำคัญสูง)
    "match": {  // เงื่อนไขที่แพ็กเก็ตต้องตรงกับกฎนี้
        "in_port": 1  // ตรวจสอบว่าแพ็กเก็ตเข้ามาทางพอร์ตหมายเลข 1
    },
    "actions": [  // การกระทำที่สวิตช์จะทำเมื่อแพ็กเก็ตตรงกับกฎนี้
        {
            "type": "OUTPUT",  // ประเภทการกระทำ: ส่งแพ็กเก็ตออกทางพอร์ตที่ระบุ
            "port": 3  // พอร์ตที่แพ็กเก็ตจะถูกส่งออก (พอร์ตหมายเลข 3)
        }
    ]
}
```

```
s1_flow2
{
    "dpid": 1,  // Datapath ID: ระบุสวิตช์ OpenFlow ที่ใช้กฎนี้ (สวิตช์หมายเลข 1)
    "table_id": 0,  // ตารางที่กฎนี้จะถูกใช้ (ตารางหมายเลข 0 คือตารางเริ่มต้น)
    "idle_timeout": 0,  // เวลาที่กฎจะถูกลบหากไม่มีการใช้งาน (0 = ไม่ถูกลบ)
    "hard_timeout": 0,  // เวลาที่กฎจะถูกลบไม่ว่าจะมีการใช้งานหรือไม่ (0 = ไม่ถูกลบ)
    "priority": 100,  // ลำดับความสำคัญของกฎ (100 = มีความสำคัญสูง)
    "match": {  // เงื่อนไขที่แพ็กเก็ตต้องตรงกับกฎนี้
        "in_port": 2  // ตรวจสอบว่าแพ็กเก็ตเข้ามาทางพอร์ตหมายเลข 2
    },
    "actions": [  // การกระทำที่สวิตช์จะทำเมื่อแพ็กเก็ตตรงกับกฎนี้
        {
            "type": "OUTPUT",  // ประเภทการกระทำ: ส่งแพ็กเก็ตออกทางพอร์ตที่ระบุ
            "port": 3  // พอร์ตที่แพ็กเก็ตจะถูกส่งออก (พอร์ตหมายเลข 3)
        }
    ]
}
```

```
s1_flow3
{
    "dpid": 1,  // Datapath ID: ระบุสวิตช์ OpenFlow ที่ใช้กฎนี้ (สวิตช์หมายเลข 1)
    "table_id": 0,  // ตารางที่กฎนี้จะถูกใช้ (ตารางหมายเลข 0 คือตารางเริ่มต้น)
    "idle_timeout": 0,  // เวลาที่กฎจะถูกลบหากไม่มีการใช้งาน (0 = ไม่ถูกลบ)
    "hard_timeout": 0,  // เวลาที่กฎจะถูกลบไม่ว่าจะมีการใช้งานหรือไม่ (0 = ไม่ถูกลบ)
    "priority": 100,  // ลำดับความสำคัญของกฎ (100 = มีความสำคัญสูง)
    "match": {  // เงื่อนไขที่แพ็กเก็ตต้องตรงกับกฎนี้
        "in_port": 3  // ตรวจสอบว่าแพ็กเก็ตเข้ามาทางพอร์ตหมายเลข 3
    },
    "actions": [  // การกระทำที่สวิตช์จะทำเมื่อแพ็กเก็ตตรงกับกฎนี้
        {
            "type": "GROUP",  // ประเภทการกระทำ: ส่งแพ็กเก็ตไปยังกลุ่ม (Group)
            "group_id": 50  // Group ID: ตัวระบุของกลุ่มที่แพ็กเก็ตจะถูกส่งไป (กลุ่มหมายเลข 50)
        }
    ]
}
```

```
s1_group50
{
    "dpid": 1,  // Datapath ID: ระบุสวิตช์ OpenFlow ที่ใช้ Group นี้ (สวิตช์หมายเลข 1)
    "type": "SELECT",  // ประเภทของ Group: "SELECT" หมายถึงส่งแพ็กเก็ตไปยังบัคเก็ตเดียวโดยใช้การเลือกตามน้ำหนักหรือแบบสุ่ม
    "group_id": 50,  // Group ID: ตัวระบุของ Group นี้ (Group หมายเลข 50)
    "buckets": [  // บัคเก็ต (Buckets) ที่กำหนดการกระทำสำหรับ Group นี้
        {
            "weight": 50,  // น้ำหนักของบัคเก็ตนี้ (50 = 50% ของการเลือก)
            "actions": [  // การกระทำที่สวิตช์จะทำในบัคเก็ตนี้
                {
                    "type": "OUTPUT",  // ประเภทการกระทำ: ส่งแพ็กเก็ตออกทางพอร์ตที่ระบุ
                    "port": 1  // พอร์ตที่แพ็กเก็ตจะถูกส่งออก (พอร์ตหมายเลข 1)
                }
            ]
        },
        {
            "weight": 50,  // น้ำหนักของบัคเก็ตนี้ (50 = 50% ของการเลือก)
            "actions": [  // การกระทำที่สวิตช์จะทำในบัคเก็ตนี้
                {
                    "type": "OUTPUT",  // ประเภทการกระทำ: ส่งแพ็กเก็ตออกทางพอร์ตที่ระบุ
                    "port": 2  // พอร์ตที่แพ็กเก็ตจะถูกส่งออก (พอร์ตหมายเลข 2)
                }
            ]
        }
    ]
}
```

```
s2_flow1
{
    "dpid": 2,  // Datapath ID: ระบุสวิตช์ OpenFlow ที่ใช้กฎนี้ (สวิตช์หมายเลข 2)
    "table_id": 0,  // ตารางที่กฎนี้จะถูกใช้ (ตารางหมายเลข 0 คือตารางเริ่มต้น)
    "idle_timeout": 0,  // เวลาที่กฎจะถูกลบหากไม่มีการใช้งาน (0 = ไม่ถูกลบ)
    "hard_timeout": 0,  // เวลาที่กฎจะถูกลบไม่ว่าจะมีการใช้งานหรือไม่ (0 = ไม่ถูกลบ)
    "priority": 100,  // ลำดับความสำคัญของกฎ (100 = มีความสำคัญสูง)
    "match": {  // เงื่อนไขที่แพ็กเก็ตต้องตรงกับกฎนี้
        "in_port": 1  // ตรวจสอบว่าแพ็กเก็ตเข้ามาทางพอร์ตหมายเลข 1
    },
    "actions": [  // การกระทำที่สวิตช์จะทำเมื่อแพ็กเก็ตตรงกับกฎนี้
        {
            "type": "OUTPUT",  // ประเภทการกระทำ: ส่งแพ็กเก็ตออกทางพอร์ตที่ระบุ
            "port": 2  // พอร์ตที่แพ็กเก็ตจะถูกส่งออก (พอร์ตหมายเลข 2)
        }
    ]
}
```

```
s2_flow2
{
    "dpid": 2,  // Datapath ID: ระบุสวิตช์ OpenFlow ที่ใช้กฎนี้ (สวิตช์หมายเลข 2)
    "table_id": 0,  // ตารางที่กฎนี้จะถูกใช้ (ตารางหมายเลข 0 คือตารางเริ่มต้น)
    "idle_timeout": 0,  // เวลาที่กฎจะถูกลบหากไม่มีการใช้งาน (0 = ไม่ถูกลบ)
    "hard_timeout": 0,  // เวลาที่กฎจะถูกลบไม่ว่าจะมีการใช้งานหรือไม่ (0 = ไม่ถูกลบ)
    "priority": 100,  // ลำดับความสำคัญของกฎ (100 = มีความสำคัญสูง)
    "match": {  // เงื่อนไขที่แพ็กเก็ตต้องตรงกับกฎนี้
        "in_port": 2  // ตรวจสอบว่าแพ็กเก็ตเข้ามาทางพอร์ตหมายเลข 2
    },
    "actions": [  // การกระทำที่สวิตช์จะทำเมื่อแพ็กเก็ตตรงกับกฎนี้
        {
            "type": "OUTPUT",  // ประเภทการกระทำ: ส่งแพ็กเก็ตออกทางพอร์ตที่ระบุ
            "port": 1  // พอร์ตที่แพ็กเก็ตจะถูกส่งออก (พอร์ตหมายเลข 1)
        }
    ]
}
```

```
s3_flow1
{
    "dpid": 3,  // Datapath ID: ระบุสวิตช์ OpenFlow ที่ใช้กฎนี้ (สวิตช์หมายเลข 3)
    "table_id": 0,  // ตารางที่กฎนี้จะถูกใช้ (ตารางหมายเลข 0 คือตารางเริ่มต้น)
    "idle_timeout": 0,  // เวลาที่กฎจะถูกลบหากไม่มีการใช้งาน (0 = ไม่ถูกลบ)
    "hard_timeout": 0,  // เวลาที่กฎจะถูกลบไม่ว่าจะมีการใช้งานหรือไม่ (0 = ไม่ถูกลบ)
    "priority": 100,  // ลำดับความสำคัญของกฎ (100 = มีความสำคัญสูง)
    "match": {  // เงื่อนไขที่แพ็กเก็ตต้องตรงกับกฎนี้
        "in_port": 1  // ตรวจสอบว่าแพ็กเก็ตเข้ามาทางพอร์ตหมายเลข 1
    },
    "actions": [  // การกระทำที่สวิตช์จะทำเมื่อแพ็กเก็ตตรงกับกฎนี้
        {
            "type": "OUTPUT",  // ประเภทการกระทำ: ส่งแพ็กเก็ตออกทางพอร์ตที่ระบุ
            "port": 2  // พอร์ตที่แพ็กเก็ตจะถูกส่งออก (พอร์ตหมายเลข 2)
        }
    ]
}
```

```
s3_flow2
{
    "dpid": 3,  // Datapath ID: ระบุสวิตช์ OpenFlow ที่ใช้กฎนี้ (สวิตช์หมายเลข 3)
    "table_id": 0,  // ตารางที่กฎนี้จะถูกใช้ (ตารางหมายเลข 0 คือตารางเริ่มต้น)
    "idle_timeout": 0,  // เวลาที่กฎจะถูกลบหากไม่มีการใช้งาน (0 = ไม่ถูกลบ)
    "hard_timeout": 0,  // เวลาที่กฎจะถูกลบไม่ว่าจะมีการใช้งานหรือไม่ (0 = ไม่ถูกลบ)
    "priority": 100,  // ลำดับความสำคัญของกฎ (100 = มีความสำคัญสูง)
    "match": {  // เงื่อนไขที่แพ็กเก็ตต้องตรงกับกฎนี้
        "in_port": 2  // ตรวจสอบว่าแพ็กเก็ตเข้ามาทางพอร์ตหมายเลข 2
    },
    "actions": [  // การกระทำที่สวิตช์จะทำเมื่อแพ็กเก็ตตรงกับกฎนี้
        {
            "type": "OUTPUT",  // ประเภทการกระทำ: ส่งแพ็กเก็ตออกทางพอร์ตที่ระบุ
            "port": 1  // พอร์ตที่แพ็กเก็ตจะถูกส่งออก (พอร์ตหมายเลข 1)
        }
    ]
}
```

```
s4_flow1
{
    "dpid": 4,  // Datapath ID: ระบุสวิตช์ OpenFlow ที่ใช้กฎนี้ (สวิตช์หมายเลข 4)
    "table_id": 0,  // ตารางที่กฎนี้จะถูกใช้ (ตารางหมายเลข 0 คือตารางเริ่มต้น)
    "idle_timeout": 0,  // เวลาที่กฎจะถูกลบหากไม่มีการใช้งาน (0 = ไม่ถูกลบ)
    "hard_timeout": 0,  // เวลาที่กฎจะถูกลบไม่ว่าจะมีการใช้งานหรือไม่ (0 = ไม่ถูกลบ)
    "priority": 100,  // ลำดับความสำคัญของกฎ (100 = มีความสำคัญสูง)
    "match": {  // เงื่อนไขที่แพ็กเก็ตต้องตรงกับกฎนี้
        "in_port": 1  // ตรวจสอบว่าแพ็กเก็ตเข้ามาทางพอร์ตหมายเลข 1
    },
    "actions": [  // การกระทำที่สวิตช์จะทำเมื่อแพ็กเก็ตตรงกับกฎนี้
        {
            "type": "OUTPUT",  // ประเภทการกระทำ: ส่งแพ็กเก็ตออกทางพอร์ตที่ระบุ
            "port": 2  // พอร์ตที่แพ็กเก็ตจะถูกส่งออก (พอร์ตหมายเลข 2)
        }
    ]
}
```

s4_flow2

```
{
    "dpid": 4,  // Datapath ID: ระบุสวิตช์ OpenFlow ที่ใช้กฎนี้ (สวิตช์หมายเลข 4)
    "table_id": 0,  // ตารางที่กฎนี้จะถูกใช้ (ตารางหมายเลข 0 คือตารางเริ่มต้น)
    "idle_timeout": 0,  // เวลาที่กฎจะถูกลบหากไม่มีการใช้งาน (0 = ไม่ถูกลบ)
    "hard_timeout": 0,  // เวลาที่กฎจะถูกลบไม่ว่าจะมีการใช้งานหรือไม่ (0 = ไม่ถูกลบ)
    "priority": 100,  // ลำดับความสำคัญของกฎ (100 = มีความสำคัญสูง)
    "match": {  // เงื่อนไขที่แพ็กเก็ตต้องตรงกับกฎนี้
        "in_port": 2  // ตรวจสอบว่าแพ็กเก็ตเข้ามาทางพอร์ตหมายเลข 2
    },
    "actions": [  // การกระทำที่สวิตช์จะทำเมื่อแพ็กเก็ตตรงกับกฎนี้
        {
            "type": "OUTPUT",  // ประเภทการกระทำ: ส่งแพ็กเก็ตออกทางพอร์ตที่ระบุ
            "port": 1  // พอร์ตที่แพ็กเก็ตจะถูกส่งออก (พอร์ตหมายเลข 1)
        }
    ]
}
```

s5_flow1

```
{
    "dpid": 5,  // Datapath ID: ระบุสวิตช์ OpenFlow ที่ใช้กฎนี้ (สวิตช์หมายเลข 5)
    "table_id": 0,  // ตารางที่กฎนี้จะถูกใช้ (ตารางหมายเลข 0 คือตารางเริ่มต้น)
    "idle_timeout": 0,  // เวลาที่กฎจะถูกลบหากไม่มีการใช้งาน (0 = ไม่ถูกลบ)
    "hard_timeout": 0,  // เวลาที่กฎจะถูกลบไม่ว่าจะมีการใช้งานหรือไม่ (0 = ไม่ถูกลบ)
    "priority": 100,  // ลำดับความสำคัญของกฎ (100 = มีความสำคัญสูง)
    "match": {  // เงื่อนไขที่แพ็กเก็ตต้องตรงกับกฎนี้
        "in_port": 1  // ตรวจสอบว่าแพ็กเก็ตเข้ามาทางพอร์ตหมายเลข 1
    },
    "actions": [  // การกระทำที่สวิตช์จะทำเมื่อแพ็กเก็ตตรงกับกฎนี้
        {
            "type": "OUTPUT",  // ประเภทการกระทำ: ส่งแพ็กเก็ตออกทางพอร์ตที่ระบุ
            "port": 3  // พอร์ตที่แพ็กเก็ตจะถูกส่งออก (พอร์ตหมายเลข 3)
        }
    ]
}
```

s5_flow2

```
{
    "dpid": 5,  // Datapath ID: ระบุสวิตช์ OpenFlow ที่ใช้กฎนี้ (สวิตช์หมายเลข 5)
    "table_id": 0,  // ตารางที่กฎนี้จะถูกใช้ (ตารางหมายเลข 0 คือตารางเริ่มต้น)
    "idle_timeout": 0,  // เวลาที่กฎจะถูกลบหากไม่มีการใช้งาน (0 = ไม่ถูกลบ)
    "hard_timeout": 0,  // เวลาที่กฎจะถูกลบไม่ว่าจะมีการใช้งานหรือไม่ (0 = ไม่ถูกลบ)
    "priority": 100,  // ลำดับความสำคัญของกฎ (100 = มีความสำคัญสูง)
    "match": {  // เงื่อนไขที่แพ็กเก็ตต้องตรงกับกฎนี้
        "in_port": 2  // ตรวจสอบว่าแพ็กเก็ตเข้ามาทางพอร์ตหมายเลข 2
    },
    "actions": [  // การกระทำที่สวิตช์จะทำเมื่อแพ็กเก็ตตรงกับกฎนี้
        {
            "type": "OUTPUT",  // ประเภทการกระทำ: ส่งแพ็กเก็ตออกทางพอร์ตที่ระบุ
            "port": 3  // พอร์ตที่แพ็กเก็ตจะถูกส่งออก (พอร์ตหมายเลข 3)
        }
    ]
}
```

s5_flow3

```
{
    "dpid": 5,  // Datapath ID: ระบุสวิตช์ OpenFlow ที่ใช้กฎนี้ (สวิตช์หมายเลข 5)
    "table_id": 0,  // ตารางที่กฎนี้จะถูกใช้ (ตารางหมายเลข 0 คือตารางเริ่มต้น)
    "idle_timeout": 0,  // เวลาที่กฎจะถูกลบหากไม่มีการใช้งาน (0 = ไม่ถูกลบ)
    "hard_timeout": 0,  // เวลาที่กฎจะถูกลบไม่ว่าจะมีการใช้งานหรือไม่ (0 = ไม่ถูกลบ)
    "priority": 100,  // ลำดับความสำคัญของกฎ (100 = มีความสำคัญสูง)
    "match": {  // เงื่อนไขที่แพ็กเก็ตต้องตรงกับกฎนี้
        "in_port": 3  // ตรวจสอบว่าแพ็กเก็ตเข้ามาทางพอร์ตหมายเลข 3
    },
    "actions": [  // การกระทำที่สวิตช์จะทำเมื่อแพ็กเก็ตตรงกับกฎนี้
        {
            "type": "GROUP",  // ประเภทการกระทำ: ส่งแพ็กเก็ตไปยังกลุ่ม (Group)
            "group_id": 51  // Group ID: ตัวระบุของกลุ่มที่แพ็กเก็ตจะถูกส่งไป (กลุ่มหมายเลข 51)
        }
    ]
}
```

s5_group51

```
{
    "dpid": 5,  // Datapath ID: ระบุสวิตช์ OpenFlow ที่ใช้ Group นี้ (สวิตช์หมายเลข 5)
    "type": "SELECT",  // ประเภทของ Group: "SELECT" หมายถึงส่งแพ็กเก็ตไปยังบัคเก็ตเดียวโดยใช้การเลือกตามน้ำหนักหรือแบบสุ่ม
    "group_id": 51,  // Group ID: ตัวระบุของ Group นี้ (Group หมายเลข 51)
    "buckets": [  // บัคเก็ต (Buckets) ที่กำหนดการกระทำสำหรับ Group นี้
        {
            "weight": 50,  // น้ำหนักของบัคเก็ตนี้ (50 = 50% ของการเลือก)
            "actions": [  // การกระทำที่สวิตช์จะทำในบัคเก็ตนี้
                {
                    "type": "OUTPUT",  // ประเภทการกระทำ: ส่งแพ็กเก็ตออกทางพอร์ตที่ระบุ
                    "port": 1  // พอร์ตที่แพ็กเก็ตจะถูกส่งออก (พอร์ตหมายเลข 1)
                }
            ]
        },
        {
            "weight": 50,  // น้ำหนักของบัคเก็ตนี้ (50 = 50% ของการเลือก)
            "actions": [  // การกระทำที่สวิตช์จะทำในบัคเก็ตนี้
                {
                    "type": "OUTPUT",  // ประเภทการกระทำ: ส่งแพ็กเก็ตออกทางพอร์ตที่ระบุ
                    "port": 2  // พอร์ตที่แพ็กเก็ตจะถูกส่งออก (พอร์ตหมายเลข 2)
                }
            ]
        }
    ]
}
```

```
curl -X POST http://localhost:8080/stats/flowentry/add -d '@s2_flow1.json'
curl -X POST http://localhost:8080/stats/flowentry/add -d '@s2_flow2.json'
curl -X POST http://localhost:8080/stats/flowentry/add -d '@s3_flow1.json'
curl -X POST http://localhost:8080/stats/flowentry/add -d '@s3_flow2.json'
curl -X POST http://localhost:8080/stats/flowentry/add -d '@s4_flow1.json'
curl -X POST http://localhost:8080/stats/flowentry/add -d '@s4_flow2.json'
curl -X POST  http://localhost:8080/stats/groupentry/add -d '@s5_group51.json'
curl -X POST  http://localhost:8080/stats/groupentry/add -d '@s1_group50.json'
curl -X POST  http://localhost:8080/stats/flowentry/add -d '@s1_flow1.json'
curl -X POST  http://localhost:8080/stats/flowentry/add -d '@s1_flow2.json'
curl -X POST  http://localhost:8080/stats/flowentry/add -d '@s1_flow3.json'
curl -X POST  http://localhost:8080/stats/flowentry/add -d '@s5_flow1.json'
curl -X POST  http://localhost:8080/stats/flowentry/add -d '@s5_flow2.json'
curl -X POST  http://localhost:8080/stats/flowentry/add -d '@s5_flow3.json'
```

Flow ต่อไปของเหลี่ยมจุ๊ๆ

ใช้กับ topo นี้

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

หรือ

Topo6.py ตามหน้าที่ 18 เปิดดู อันนี้จะบอกวิธีการใช้

1. ในเรื่อง **Multitables** ให้นักศึกษาทำการกำหนด **Flow** เงื่อนไข สัมพันธ์กับรูปภาพข้างล่างโดยมีเงื่อนไขดังนี้
   a. ทำการ **run ryu manager ofctl**
   b. ทำการ **run mininet** ให้มีการสร้าง **switch 1** ตัว และมี **host** อยู่ **6** เครื่อง
   c. ให้ทำการ **block TCP** จาก **h2** ที่ **table 20**
   d. ให้ทำการ **block icmp** จาก **h5** ที่ **table 30**
   e. ที่เหลือสามารถ **ping** หากันได้ตามปกติ และรับส่งข้อมูล **TCP** ผ่าน **Iperf** ได้ตามปกติ

table0_flow1.json
```
{
    "dpid": 1,
    "table_id": 0,
    "idle_timeout": 0,
    "hard_timeout": 0,
    "priority": 0,
    "match":{
    },
    "actions":[
        {
            "type":"GOTO_TABLE",
            "table_id": 10
        }
    ]
}
```

table10_flow1.json
```
{
    "dpid": 1,
    "table_id": 10,
    "idle_timeout": 0,
    "hard_timeout": 0,
    "priority": 0,
    "match":{
    },
    "actions":[
        {
            "type":"GOTO_TABLE",
            "table_id": 20
        }
    ]
}
```

table20_flow1.json
```
{
    "dpid": 1,
    "table_id": 20,
    "idle_timeout": 0,
    "hard_timeout": 0,
    "priority": 0,
    "match":{
    },
    "actions":[
        {
            "type":"GOTO_TABLE",
            "table_id": 30
        }
    ]
}
```

table20_flow2.json
```
{
    "dpid": 1,
    "table_id": 20,
    "idle_timeout": 0,
    "hard_timeout": 0,
    "priority": 100,
    "match":{
        "eth_type": 2048,
        "ipv4_src": "10.0.0.2",
        "ip_proto": 6
    },
    "actions":[
    ]
}
```

table30_arp.json
```
{
    "dpid": 1,
    "table_id": 30,
    "idle_timeout": 0,
    "hard_timeout": 0,
    "priority": 0,
    "match":{
    "dl_dst": "ff:ff:ff:ff:ff:ff"
    },
    "actions":[
        {
            "type":"OUTPUT",
            "port": 4294967291
        }
    ]
}
```

table30_flow1.json
```
{
    "dpid": 1,
    "table_id": 30,
    "idle_timeout": 0,
    "hard_timeout": 0,
    "priority": 0,
    "match":{
    "dl_dst": "00:00:00:00:00:01"
    },
    "actions":[
        {
            "type":"OUTPUT",
            "port": 1
        }
    ]
}
```

table30_flow2.json
```
{
    "dpid": 1,
    "table_id": 30,
    "idle_timeout": 0,
    "hard_timeout": 0,
    "priority": 0,
    "match":{
    "dl_dst": "00:00:00:00:00:02"
    },
    "actions":[
        {
            "type":"OUTPUT",
            "port": 2
        }
    ]
}
```

table30_flow3.json
```
{
    "dpid": 1,
    "table_id": 30,
    "idle_timeout": 0,
    "hard_timeout": 0,
    "priority": 0,
    "match":{
    "dl_dst": "00:00:00:00:00:03"
    },
    "actions":[
        {
            "type":"OUTPUT",
            "port": 3
        }
    ]
}
```

table30_flow4.json
```
{
    "dpid": 1,
    "table_id": 30,
    "idle_timeout": 0,
    "hard_timeout": 0,
    "priority": 0,
    "match":{
    "dl_dst": "00:00:00:00:00:04"
    },
    "actions":[
        {
            "type":"OUTPUT",
            "port": 4
        }
    ]
}
```

table30_flow5.json > ...

```json
{
    "dpid": 1,
    "table_id": 30,
    "idle_timeout": 0,
    "hard_timeout": 0,
    "priority": 0,
    "match":{
    "dl_dst": "00:00:00:00:00:05"
    },
    "actions":[
        {
            "type":"OUTPUT",
            "port": 5
        }
    ]
}
```

table30_flow6.json > ...

```json
{
    "dpid": 1,
    "table_id": 30,
    "idle_timeout": 0,
    "hard_timeout": 0,
    "priority": 0,
    "match":{
    "dl_dst": "00:00:00:00:00:06"
    },
    "actions":[
        {
            "type":"OUTPUT",
            "port": 6
        }
    ]
}
```

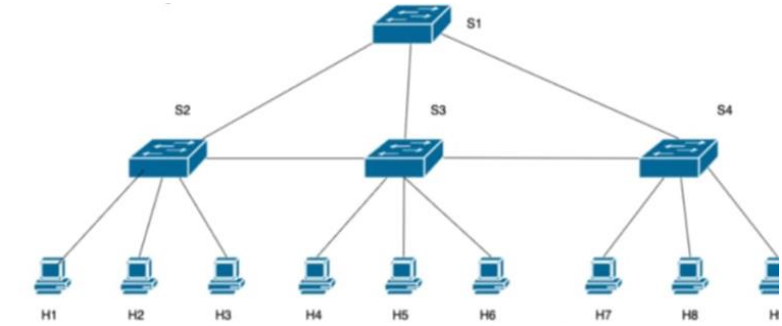
table30_flow7.json > ...

```json
{
    "dpid": 1,
    "table_id": 20,
    "idle_timeout": 0,
    "hard_timeout": 0,
    "priority": 100,
    "match":{
        "eth_type": 2048,
        "ipv4_src": "10.0.0.5",
        "ip_proto": 1
    },
    "actions":[
    ]
}
```

ค่า `ip_proto` ที่พบบ่อย

ค่า	โปรโตคอล	คำอธิบาย
1	ICMP	Internet Control Message Protocol (ใช้สำหรับ Ping และการตรวจสอบเครือข่าย)
6	TCP	Transmission Control Protocol (ใช้สำหรับการเชื่อมต่อที่เชื่อถือได้ เช่น HTTP)
17	UDP	User Datagram Protocol (ใช้สำหรับการส่งข้อมูลแบบไม่เชื่อถือได้ เช่น DNS)
2	IGMP	Internet Group Management Protocol (ใช้สำหรับการจัดการ Multicast Group)
47	GRE	Generic Routing Encapsulation (ใช้สำหรับการสร้าง Tunnel)
50	ESP	Encapsulating Security Payload (ใช้ใน IPSec สำหรับการเข้ารหัส)
51	AH	Authentication Header (ใช้ใน IPSec สำหรับการตรวจสอบความถูกต้อง)
89	OSPF	Open Shortest Path First (โปรโตคอล Routing)
132	SCTP	Stream Control Transmission Protocol (คล้าย TCP แต่มีฟีเจอร์เพิ่มเติม)

อันนี้โค้ดเผื่อใช้สร้าง Topo นี้เพื่อโจทย์ออก



Network_Topo.py 5 ×

HomeWork > Network_Topo.py > SingleSwitchTopo > build

```python
#!/usr/bin/python


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):
    def build(self):
        s1 = self.addSwitch('s1')
        s2 = self.addSwitch('s2')
        s3 = self.addSwitch('s3')
        s4 = self.addSwitch('s4')
        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")
        h5 = self.addHost('h5', mac="00:00:00:00:00:05", ip="192.168.1.5/24")
        h6 = self.addHost('h6', mac="00:00:00:00:00:06", ip="192.168.1.6/24")
        h7 = self.addHost('h7', mac="00:00:00:00:00:07", ip="192.168.1.7/24")
        h8 = self.addHost('h8', mac="00:00:00:00:00:08", ip="192.168.1.8/24")
        h9 = self.addHost('h9', mac="00:00:00:00:00:09", ip="192.168.1.9/24")


        self.addLink(h1, s2)   # h1 ต่อกับ s2
        self.addLink(h2, s2)
        self.addLink(h3, s2)
        self.addLink(h4, s3)
        self.addLink(h5, s3)
        self.addLink(h6, s3)
        self.addLink(h7, s4)
        self.addLink(h8, s4)
        self.addLink(h9, s4)

        self.addLink(s1, s2)
        self.addLink(s1, s3)
        self.addLink(s1, s4)
        self.addLink(s2, s3)
        self.addLink(s3, s4)


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()
```

อันนี้ใช้สร้างแบบ 1 Switch 6 Host เผื่อใช้งาน

```python
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 MyTopo(Topo):
    def __init__(self):
        Topo.__init__(self)

        # Add switch
        s1 = self.addSwitch('s1')

        # Add hosts
        h1 = self.addHost('h1', ip="10.0.0.1/24")
        h2 = self.addHost('h2', ip="10.0.0.2/24")
        h3 = self.addHost('h3', ip="10.0.0.3/24")
        h4 = self.addHost('h4', ip="10.0.0.4/24")
        h5 = self.addHost('h5', ip="10.0.0.5/24")
        h6 = self.addHost('h6', ip="10.0.0.6/24")

        # Add links
        self.addLink(h1, s1)
        self.addLink(h2, s1)
        self.addLink(h3, s1)
        self.addLink(h4, s1)
        self.addLink(h5, s1)
        self.addLink(h6, s1)

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

ต่อตัวอย่างของ json ที่เกี่ยวกับ Group เผื่อเอามาแปลง

```
{
  "dpid": 1,                // Datapath ID ของสวิตช์
  "group_id": 1,            // ID ของ Group
  "type": "ALL",            // ประเภทของ Group
  "buckets": [              // กลุ่มของ Buckets ที่กำหนดการกระทำ (Actions)
    {
      "actions": [          // การกระทำที่ต้องทำใน Bucket นี้
        {
          "type": "OUTPUT",  // ส่งแพ็กเก็ตออกทางพอร์ต
          "port": 2          // พอร์ตที่ส่งแพ็กเก็ตออก
        }
      ]
    },
    {
      "actions": [          // การกระทำที่ต้องทำใน Bucket นี้
        {
          "type": "OUTPUT",  // ส่งแพ็กเก็ตออกทางพอร์ต
          "port": 3          // พอร์ตที่ส่งแพ็กเก็ตออก
        }
      ]
    }
  ]
}
```

1. Group Type: ALL

- **วัตถุประสงค์**: ส่งแพ็กเก็ตไปยังทุก Bucket ใน Group
- **ตัวอย่างการใช้งาน**: ส่งแพ็กเก็ตไปยังหลายพอร์ตพร้อมกัน (Multicast)

json Copy

```
{
  "dpid": 1,
  "group_id": 1,
  "type": "ALL",
  "buckets": [
    {
      "actions": [
        {
          "type": "OUTPUT",
          "port": 2
        }
      ]
    },
    {
      "actions": [
        {
          "type": "OUTPUT",
          "port": 3
        }
      ]
    }
  ]
}
```

- **ผลลัพธ์**: แพ็กเก็ตจะถูกส่งออกทางพอร์ต 2 และพอร์ต 3 พร้อมกัน

2. Group Type: SELECT

- **วัตถุประสงค์**: ส่งแพ็กเก็ตไปยัง Bucket ใด Bucket หนึ่งใน Group แบบสุ่มหรือตามอัลกอริทึม
- **ตัวอย่างการใช้งาน**: Load Balancing

json Copy

```
{
  "dpid": 1,
  "group_id": 2,
  "type": "SELECT",
  "buckets": [
    {
      "actions": [
        {
          "type": "OUTPUT",
          "port": 2
        }
      ]
    },
    {
      "actions": [
        {
          "type": "OUTPUT",
          "port": 3
        }
      ]
    }
  ]
}
```

- **ผลลัพธ์**: แพ็กเก็ตจะถูกส่งออกทางพอร์ต 2 หรือพอร์ต 3 แบบสุ่ม

3. Group Type: INDIRECT

- **วัตถุประสงค์**: ส่งแพ็กเก็ตไปยัง Bucket เดียวใน Group
- **ตัวอย่างการใช้งาน**: ส่งแพ็กเก็ตไปยัง Firewall หรือเครื่องมือตรวจสอบ

json Copy

```
{
  "dpid": 1,
  "group_id": 3,
  "type": "INDIRECT",
  "buckets": [
    {
      "actions": [
        {
          "type": "OUTPUT",
          "port": 4
        }
      ]
    }
  ]
}
```

- **ผลลัพธ์**: แพ็กเก็ตจะถูกส่งออกทางพอร์ต 4 เท่านั้น

4. Group Type: FF (Fast Failover)

- **วัตถุประสงค์**: ส่งแพ็กเก็ตไปยัง Bucket แรกที่พร้อมทำงาน (Active)
- **ตัวอย่างการใช้งาน**: Failover (สลับไปใช้ลิงก์สำรองเมื่อลิงก์หลักล้มเหลว)

json Copy

```
{
  "dpid": 1,
  "group_id": 4,
  "type": "FF",
  "buckets": [
    {
      "watch_port": 2,
      "actions": [
        {
          "type": "OUTPUT",
          "port": 2
        }
      ]
    },
    {
      "watch_port": 3,
      "actions": [
        {
          "type": "OUTPUT",
          "port": 3
        }
      ]
    }
  ]
}
```

- **ผลลัพธ์**: แพ็กเก็ตจะถูกส่งออกทางพอร์ต 2 หากพอร์ต 2 พร้อมทำงาน มิฉะนั้นจะส่งออกทางพอร์ต 3

สรุป

- **Group**: ใช้เพื่อจัดการแพ็กเก็ตแบบกลุ่ม (Group Processing)
- **ประเภทของ Group**:
  - `ALL` : ส่งแพ็กเก็ตไปยังทุก Bucket
  - `SELECT` : ส่งแพ็กเก็ตไปยัง Bucket ใด Bucket หนึ่ง
  - `INDIRECT` : ส่งแพ็กเก็ตไปยัง Bucket เดียว
  - `FF` **(Fast Failover)**: ส่งแพ็กเก็ตไปยัง Bucket แรกที่พร้อมทำงาน

curl -L -o Bigkuma.zip https://github.com/thanormsaksudsee/SDN_MIDTERM/archive/refs/heads/main.zip