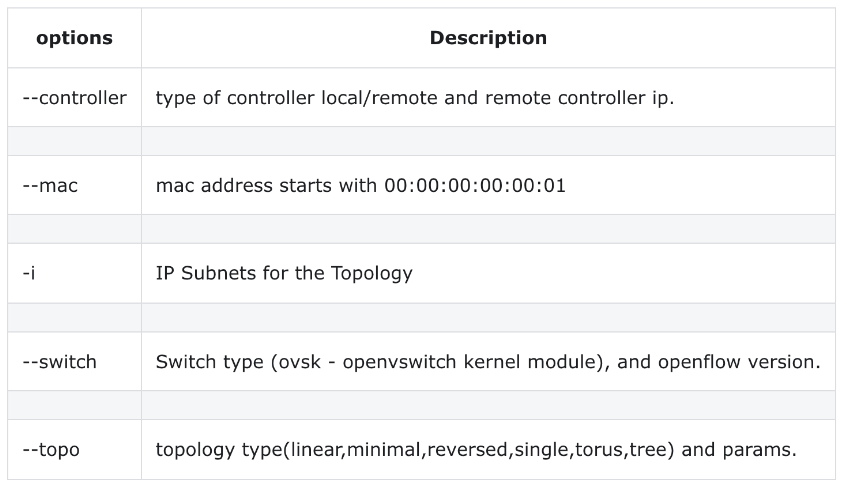
**Build basic topo (mininet):**

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

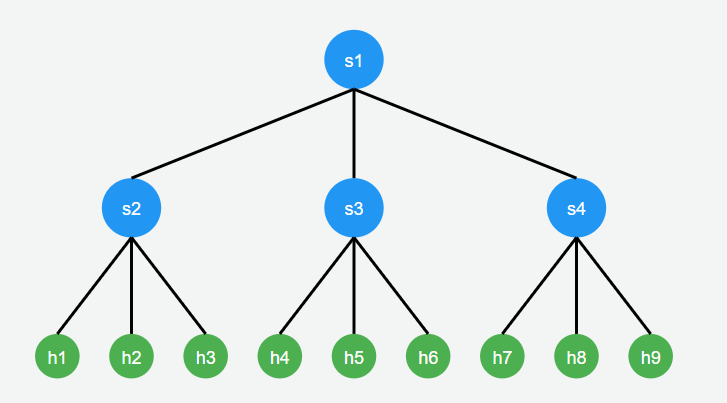
--mac -i (กำหนด ip) 10.1.1.0/24 ….. --topo=linear,4 // Switch 1 ตัว ต่อ Host 1 เครื่อง

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

*sudo mn --controller=remote,ip=127.0.0.1:XXXX --mac … // กำหนด port*



--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 ก็จะยังงานได้ (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

1. 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 ขึ้นมาพร้อมๆ กัน เช่น -P 10 คือ สร้าง Connection ขึ้นมา 10 Connections แล้วส่งข้อมูลออกไปพร้อมๆ กัน

**Ipetf 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/*](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:**

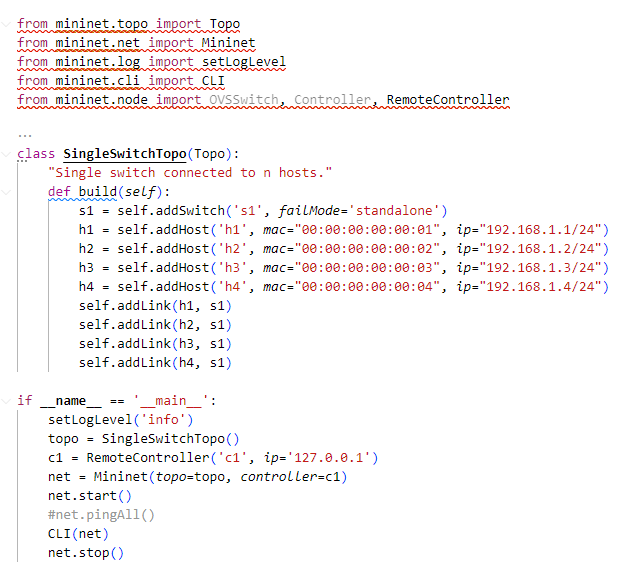
* curl -X DELETE <http://localhost:8080/stats/flowentry/clear/1> //ลบ flow rules เดิมทั้งหมด

**เพิ่ม TABLE แบบ Manual**

* sudo ovs-ofctl -O OpenFlow13 add-flow s1 "table=3,priority=100,icmp,actions=output:s1-eth2"

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

1. Simple = single,4



A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generated

**ตัวอย่างการสร้าง Topo แบบกำหนด port การเชื่อมต่อได้**

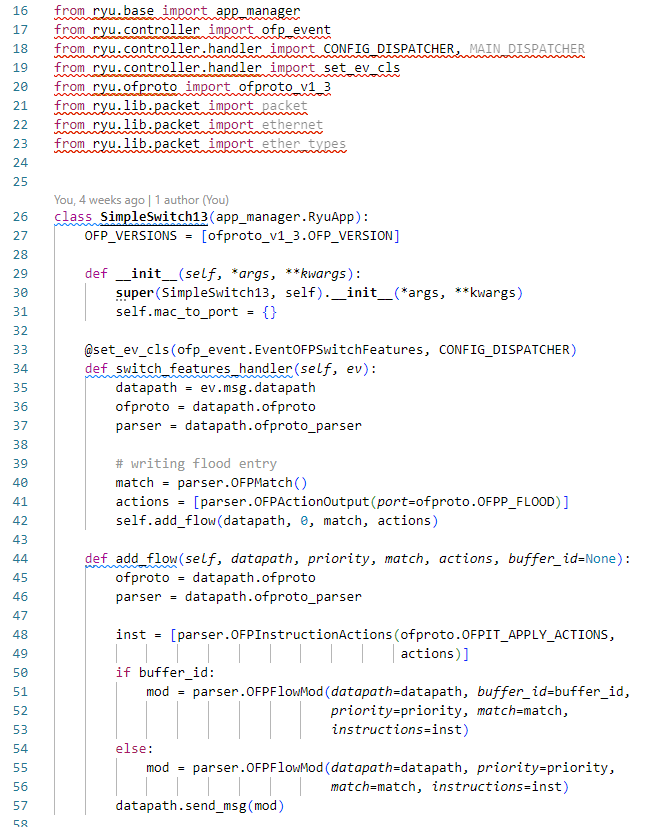
**สังเกตว่ากำหนด mode = secure**

A screenshot of a computer program

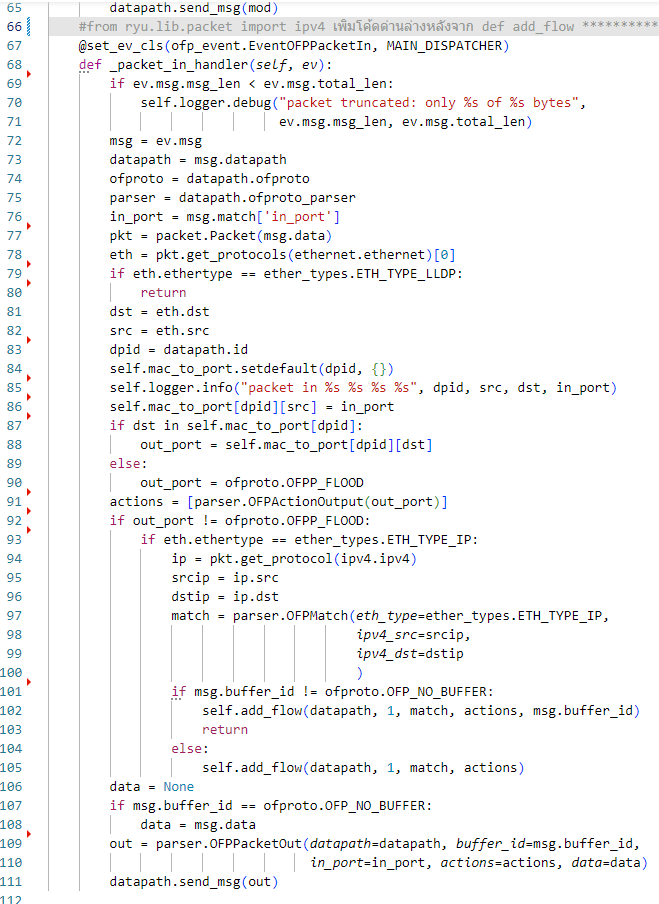
Description automatically generated

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

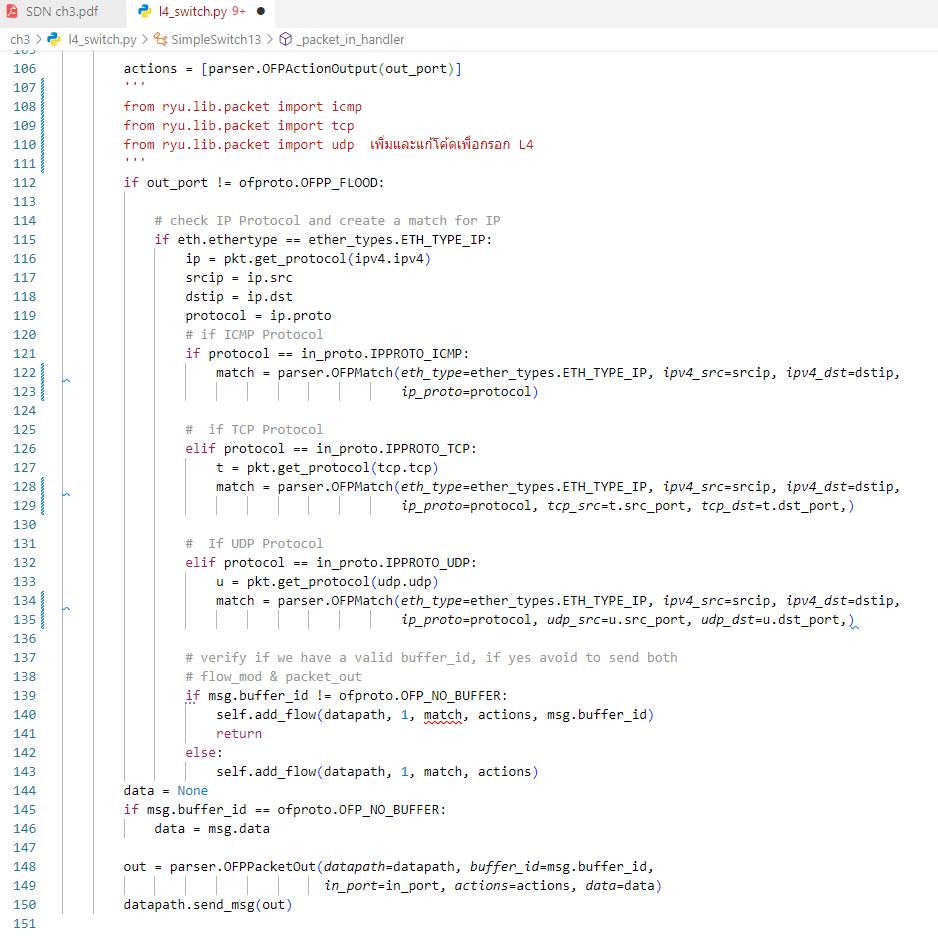
1. Hub controller (For ryu-manager: ryu-manager hub.py)

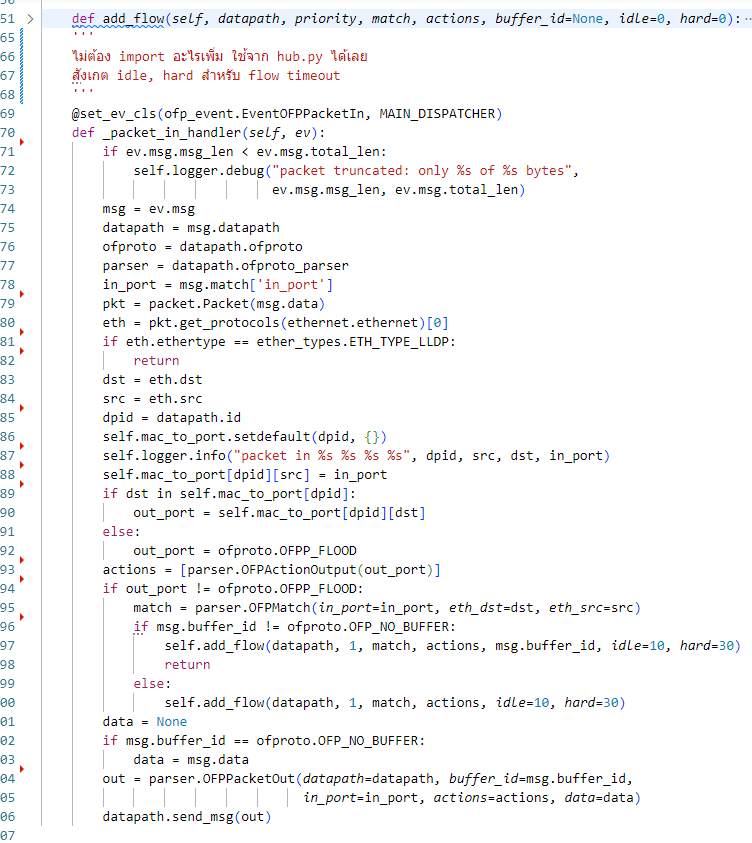


1. L3 switch controller (ryu-manager l3\_switch.py)



1. L4\_Switch (port) สามารถใช้โค้ดจาก L3 แต่แก้โค้ดและ import เพิ่ม



1. Flow timeout : 

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

**ความหมายค่า Port ตรง Action ใน Flow rule สามารถใช้คำแทนได้ "port": "CONTROLLER"**

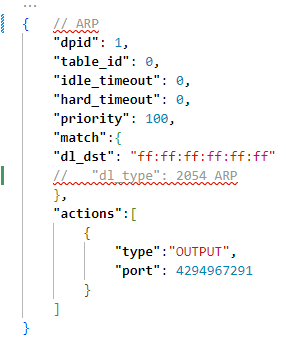
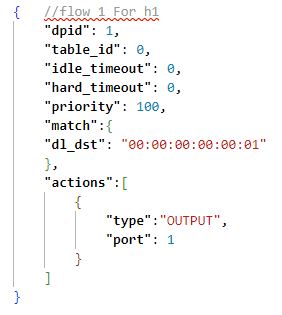
* OFPP\_FLOOD (0xFFFFFFFB) = 4294967291 //packet ออกทุก port ยกเว้น port ที่ packet เข้ามา
* OFPP\_ALL (0xFFFFFFFC) = 4294967292 ส่ง //packet ออกทุก port
* OFPP\_CONTROLLER (0xFFFFFFFD) = 4294967293 //packet ไปที่ controller
* OFPP\_TABLE (0xFFFFFFFA) = 4294967290 //packet เข้า flow table
* OFPP\_IN\_PORT (0xFFFFFFF8) = 4294967288 //packet กลับออกไปทาง port ที่เข้ามา
* OFPP\_NORMAL (0xFFFFFFFA) = 4294967290 //ให้ switch จัดการ packet ตามการทำงานปกติ

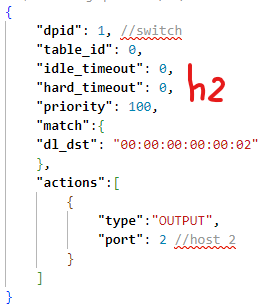
OFPP\_TABLE และ OFPP\_NORMAL มีค่าเท่ากัน (4294967290) เพราะทั้งสองค่านี้ใช้แทนพฤติกรรมเดียวกันคือการให้ switch จัดการ packet ตามการทำงานปกติผ่าน flow table

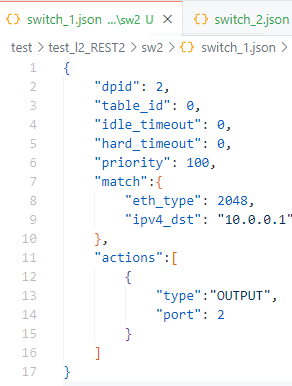
* 1. **Hub.js //วิธีส่งให้ controller ใช้ curl -X POST (stats/flowentry/add) ดูในสรุปด้านบน**

****

* 1. **L2**
  2. **ARP & Flow1 (ของ host1) // ARP ถ้าไม่ได้ให้แก้เป็น 2054 (dl\_type หรือ "eth\_type": 2054**

** **

****

 L3 ต้องระบุ eth\_type ด้วย

* 1. **Multitable**

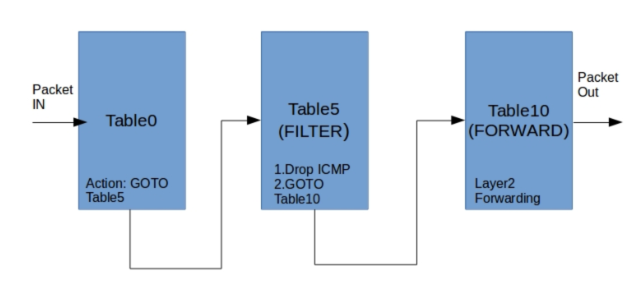
**eth\_type (EtherType) มีค่าหลักๆ ที่ใช้บ่อยดังนี้:**

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)**



A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated ดูตรง actions [port/ table\_id] ดีๆ

* 1. **Group Table**

**Multi-table workflow:**

h1 -> s1 (เข้าสู่ switch)

↓

Table 0 (ตรวจสอบพื้นฐาน)

↓

Table 5 (block ICMP - ถ้าเป็น ICMP จะถูก drop)

↓

Table 10 (forward - ส่งต่อไปยังพอร์ตปลายทาง)

* 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**

h1 -> s1 (เข้าสู่ switch)

↓

Group Table (มี bucket หลายตัว)

↙ ↓ ↘

Bucket1 Bucket2 Bucket3

(ICMP) (TCP) (Forward)

กรณี ALL Group:

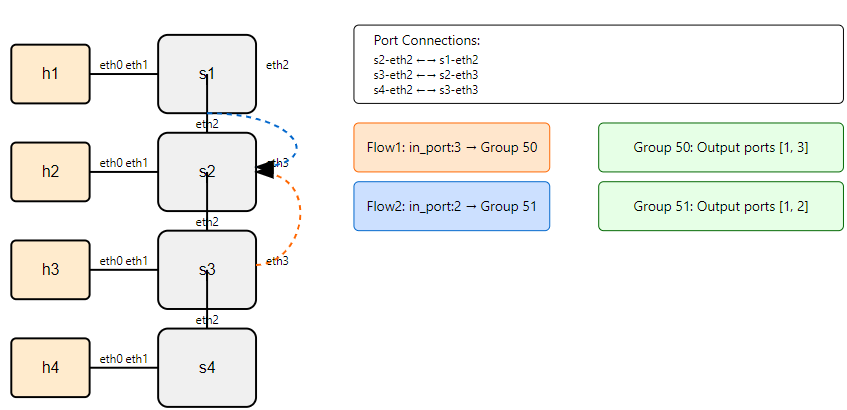
* packet จะถูกทำสำเนาและส่งไปทุก bucket
* แต่ละ bucket ทำงานแยกกัน

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
   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)
3. เมื่อ packet เข้ามาที่ s2 ระบบจะเช็ค incoming port
4. ถ้าเข้ามาที่ port 3 จะเข้า Flow1 -> Group 50
5. ถ้าเข้ามาที่ port 2 จะเข้า Flow2 -> Group 51
6. 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

A screenshot of a computer code

Description automatically generatedA screenshot of a computer code

Description automatically generated

A screenshot of a computer program

Description automatically generated

* 1. **Load balancing**

**A diagram of a diagram

Description automatically generated**

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

Screens screenshot of a computer program

Description automatically generated

สร้าง Flow rules ไปกลับของ sw2, sw3, sw4

A screenshot of a computer screen

Description automatically generated

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

A screenshot of a computer screen

Description automatically generated

Flow rules sw1 สำหรับขาออกจาก h1 โดยส่งไปให้ group 50 เพื่อทำ load balance (ทำสำหรับ sw5ด้วย)

* 1. **Statistics**
  2. Create simple TOPO (single,2 with -I [IP])
  3. ryu-manager ryu.app.simple\_monitor\_13
  4. perform TCP traffic
  5. result will show in ryu terminal
  6. **Meter Table**
  7. sudo ovs-ofctl -O OpenFlow13 meter-stats s1
  8. curl ….. /stats/meterentry/add -d “…….”

A screenshot of a computer program

Description automatically generated Add flow ตามปกติ (arp, des\_mac,)

**Xterm commands**

* tcpdump -i any -vหรือ **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

A screenshot of a computer program

Description automatically generated A screenshot of a computer program

Description automatically generated

* 1. Aggregate Stats (ใช้ของเดิม flow statistics)

A screen shot of a computer code

Description automatically generated

* 1. port statistics monitoring

A screenshot of a computer code

Description automatically generated

**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)**

**A screenshot of a computer program

Description automatically generated** **A screenshot of a computer program

Description automatically generated**

**Group - Loadbalance**

A screenshot of a computer program

Description automatically generated A screenshot of a computer program

Description automatically generated

**Arp\_proxy (เรียกใช้ arp dict python แทน)**

A screenshot of a computer code

Description automatically generated A screenshot of a computer code

Description automatically generated

A screenshot of a computer program

Description automatically generated เรียกใช้

**Topodiscovery (ryu python)**

* 1. Linear,4
  2. ryu-manager **--observe-links** topology\_discovery.py

A screenshot of a computer program

Description automatically generated A screenshot of a computer program

Description automatically generated

**Multicontroller (ryu python) (equal role)**

* 1. สร้าง Topo ด้วยโค้ดปกติ แต่ว่าชี้ controller 2 ที่

**A computer code with text

Description automatically generated with medium confidence**

* 1. กำหนด Port ตอนรัน controller ryu-manager --ofp-tcp-listen-port 6554 ……py

**Multicontroller (ryu python) (Master/Slave Role)**

* 1. สร้าง Topo ด้วยโค้ดปกติ แต่ว่าชี้ controller 2 ที่
  2. Master



* 1. Slave

A screenshot of a computer program

Description automatically generated