**CCGC 5002- SDN Lab 5-1 Ryu Group Table and Statistics**

**Compile solutions into a single word/pdf file and upload in the Lab 5 folder under Assignments tab of Blackboard. Lab 5 has got 2 parts – Lab 5-1 Ryu Group Table and Lab 5-2 Meter table and Flow Manager**

**1. Introduction**

The ability for a flow entry to point to a group enables OpenFlow to represent additional methods of forwarding (e.g. select and all).

OpenFlow groups were introduced in OpenFlow 1.1 as a way to perform more complex operations on packets that cannot be defined within a flow alone.

There are different types of groups defined in the OpenFlow specification to serve a variety of purposes

**Example of use cases:**

1) Sniffing/Port mirroring

2) Load balancing

3) fast-failovers to alternative links

**Group table Components**

Diagram

Description automatically generated

1. Group contains a bucket list
2. Bucket - it contains separate lists of actions, and parameters
3. Each bucket or list of buckets can be applied to entering packets; the exact behavior depends on the group type
4. There are four types of groups – ALL, SELECT, INDIRECT, and FAST-FAILOVER.

#### ALL Group

ALL group, will take any packet received as input and duplicate it to be operated on independently by each bucket in the bucket list. In this way, an ALL group can be used to replicate and then operate on separate copies of the packet defined by the actions in each bucket.

Example usecase: Port Mirror

#### SELECT Group

SELECT group, which is primarily designed for load balancing

SELECT group has an assigned weight, and each packet that enters the group is sent to a single bucket. The bucket selection algorithm is undefined and is dependent on the switch’s implementation;

Example usecase: Load Balancer

## 2. Demo -Sniffer

### Objective

Diagram

Description automatically generated

Make h2 as Sniffer machine. it will sniff all the packets passing via S2.

**Steps**

1. run the topology in mininiet

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

1. Run the ryu controller application(simple switch and ofctl)

ryu-manager ryu.app.simple\_switch\_13 ryu.app.ofctl\_rest

​

1. Do pingall from mininiet

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>

1. Check the flows of switch s2

sudo ovs-vsctl show

sudo ovs-ofctl -O OpenFlow13 dump-flows s2

sudo ovs-ofctl -O OpenFlow13 dump-groups s2

1. Configure the Group in S2

Group table1(Group Table ID 50):

Create a Group table with TYPE=ALL(it means, copy a packet for each bucket. and each bucket will be processed). create two buckets. one bucket will send the packet to Port3, another bucket will send the packet to Port1

Group table2(Group ID 51):

Create a Group table with TYPE=ALL(it means, copy a packet for each bucket. and each bucket will be processed). create two buckets. one bucket will send the packet to Port2, another bucket will send the packet to Port1

Method: POST URI: /stats/groupentry/add

*curl -X POST*[*http://localhost:8080/stats/groupentry/add*](http://localhost:8080/stats/groupentry/add)*-d '@group50.json'*

*curl -X POST*[*http://localhost:8080/stats/groupentry/add*](http://localhost:8080/stats/groupentry/add)*-d '@group51.json'*

**Check the group tables**

tanvir@tanvir-vm:~$ sudo ovs-ofctl -O OpenFlow13 dump-groups s2

OFPST\_GROUP\_DESC reply (OF1.3) (xid=0x2):

group\_id=50,type=all,bucket=actions=output:"s2-eth1",bucket=actions=output:"s2-eth2"

group\_id=51,type=all,bucket=actions=output:"s2-eth1",bucket=actions=output:"s2-eth3"

tanvir@tanvir-vm:~$

1. Configure the Flows

* All the packets received from port3 will be forwarded to Group table1(Group table ID 50)
* All the packets received from port2 will be forwarded to Group table2(Group table ID 51)

*curl -X POST*[*http://localhost:8080/stats/flowentry/add*](http://localhost:8080/stats/flowentry/add)*-d '@flow1.json'*

*curl -X POST*[*http://localhost:8080/stats/flowentry/add*](http://localhost:8080/stats/flowentry/add)*-d '@flow2.json'*

**Check the flow tables**

tanvir@tanvir-vm:~$ sudo ovs-ofctl -O OpenFlow13 dump-flows s2

cookie=0x0, duration=51.164s, table=0, n\_packets=8, n\_bytes=784, priority=100,in\_port="s2-eth3" actions=group:50

cookie=0x0, duration=18.320s, table=0, n\_packets=6, n\_bytes=588, priority=100,in\_port="s2-eth2" actions=group:51

....OUTPUT skipped

Now newly added flows have high priority. So this will take control of forwarding the packets.

1. Testing

Trigger a continuos ping from h1 to h4 and capture traffic in h2 using tcpdump

a) start the xterm for h2

*mininet> xterm h2*

In the h2 terminal capture tcpdump

*tcpdump -i any icmp -vvv*

b) continuous ping from h1 to h4

mininet> h1 ping h4

PING 10.1.1.4 (10.1.1.4) 56(84) bytes of data.

64 bytes from 10.1.1.4: icmp\_seq=1 ttl=64 time=2.15 ms

64 bytes from 10.1.1.4: icmp\_seq=2 ttl=64 time=0.080 ms

​

c) check the h2 xterm window, you will observe the h1 to h4 traffic

d)

check the group stats

tanvir@tanvir-vm:~$ sudo ovs-ofctl -O OpenFlow13 dump-group-stats s2

OFPST\_GROUP reply (OF1.3) (xid=0x4):

group\_id=50,duration=1171.522s,ref\_count=1,packet\_count=223,byte\_count=20790,bucket0:packet\_count=223,byte\_count=20790,bucket1:packet\_count=223,byte\_count=20790

group\_id=51,duration=1119.832s,ref\_count=1,packet\_count=219,byte\_count=20510,bucket0:packet\_count=219,byte\_count=20510,bucket1:packet\_count=219,byte\_count=20510

tanvir@tanvir-vm:~$

**3. Demo - Multipath Load balancer**

**Objective**

Forward the packet to 1 bucket(out of N buckets) and process it. (Load Balancer)

Diagram

Description automatically generated

**Steps**

1. run the topology in mininiet

sudo python topo.py

1. Run the ryu manager ofctl application

ryu-manager ryu.app.ofctl\_rest

​

1. Add the static ARP Entry in the mininiet hosts

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

1. add the flow entries for s2,s3,s4 (it simple, receive one end forward other side)

*curl -X POST*[*http://localhost:8080/stats/flowentry/add*](http://localhost:8080/stats/flowentry/add)*-d '@s2\_flow1.json'*

*curl -X POST*[*http://localhost:8080/stats/flowentry/add*](http://localhost:8080/stats/flowentry/add)*-d '@s2\_flow2.json'*

*curl -X POST*[*http://localhost:8080/stats/flowentry/add*](http://localhost:8080/stats/flowentry/add)*-d '@s3\_flow1.json'*

*curl -X POST*[*http://localhost:8080/stats/flowentry/add*](http://localhost:8080/stats/flowentry/add)*-d '@s3\_flow2.json'*

*curl -X POST*[*http://localhost:8080/stats/flowentry/add*](http://localhost:8080/stats/flowentry/add)*-d '@s4\_flow1.json'*

*curl -X POST*[*http://localhost:8080/stats/flowentry/add*](http://localhost:8080/stats/flowentry/add)*-d '@s4\_flow2.json'*

Verify the flows

sudo ovs-ofctl -O OpenFlow13 dump-flows s2

sudo ovs-ofctl -O OpenFlow13 dump-flows s3

sudo ovs-ofctl -O OpenFlow13 dump-flows s4

1. Add Group table for S1

Create a Group table with TYPE=SELECT (it means, For each bucket, a bucket will be selected (based on weight - vendor implementation ). and the selected bucket will be processed).

create two buckets. one bucket will send the packet to S2 Port, another bucket will send the packet to S3 Port

*curl -X POST*[*http://localhost:8080/stats/groupentry/add*](http://localhost:8080/stats/groupentry/add)*-d '@s1\_group50.json'*

verify the group table

sudo ovs-ofctl -O OpenFlow13 dump-groups s1

1. Add Group table for S5

Create a Group table with TYPE=SELECT (it means, For each bucket, a bucket will be selected (based on weight - vendor implementation ). and the selected bucket will be processed).

create two buckets. one bucket will send the packet to S4 Port, another bucket will send the packet to S3 Port.

*curl -X POST*[*http://localhost:8080/stats/groupentry/add*](http://localhost:8080/stats/groupentry/add)*-d '@s5\_group51.json'*

verify the group table

sudo ovs-ofctl -O OpenFlow13 dump-groups s5

1. Add flows for S1

*curl -X POST*[*http://localhost:8080/stats/flowentry/add*](http://localhost:8080/stats/flowentry/add)*-d '@s1\_flow1.json'*

*curl -X POST*[*http://localhost:8080/stats/flowentry/add*](http://localhost:8080/stats/flowentry/add)*-d '@s1\_flow2.json'*

*curl -X POST*[*http://localhost:8080/stats/flowentry/add*](http://localhost:8080/stats/flowentry/add)*-d '@s1\_flow3.json'*

1. Add flows for S5

*curl -X POST*[*http://localhost:8080/stats/flowentry/add*](http://localhost:8080/stats/flowentry/add)*-d '@s5\_flow1.json'*

*curl -X POST*[*http://localhost:8080/stats/flowentry/add*](http://localhost:8080/stats/flowentry/add)*-d '@s5\_flow2.json'*

*curl -X POST*[*http://localhost:8080/stats/flowentry/add*](http://localhost:8080/stats/flowentry/add)*-d '@s5\_flow3.json'*

1. Testing

Verify the flows

sudo ovs-ofctl -O OpenFlow13 dump-flows s1

sudo ovs-ofctl -O OpenFlow13 dump-flows s2

sudo ovs-ofctl -O OpenFlow13 dump-flows s3

sudo ovs-ofctl -O OpenFlow13 dump-flows s4

sudo ovs-ofctl -O OpenFlow13 dump-flows s5

run iperf test between h1 and h5

mininet> h2 iperf   -s &

mininet> h1 iperf -c h2 -t 60 -P 5

[This is for multiple(5) sessions; For single session - one path such as s3 might not be used much, check number of packets for single session using dump flows]

Check the group stats to see both buckets are utilized

Sudo ovs-ofctl -O OpenFlow13 dump-group-stats s1

Check the flows and groups.

\*\*\*\*\*\*\*take screenshot of the above dumps to show load balancing in the multi path using select group\*\*\*\*\*\*\*

## 4. Statistics Introduction

You may collect the statistics of flow tables, queues, meter, ports etc. OFCTL REST application provides API to collect the stats. Also RYU in built application simple\_monitor\_13.py also collect the statistics at regular interval.

Statistics collection is used to identify various traffic monitoring, threat, security applications, Network wide visualization.

**OpenFlow Messages:**

Some of the Important Statistics Messages are

Flow Statistics Message Aggregate Flow Statistics Message Table Statistics Port Statistics Queue Statistics Group Statistics Meter Statistics

Controller sends the Statistics Request Message. Switch will respond with Statistics Reply Message.

## 5. OFCTL REST API for Statistics

In the Previous sessions, Already we have discussed about the OFCTL REST API for statistics collection.

<https://ryu.readthedocs.io/en/latest/app/ofctl_rest.html>

## 6. Simple\_Monitor Application

This application collects the Flow and PORT Statistics at regular interval and print it.

1) Start the simple mininet topology

sudo mn --controller=remote,ip=127.0.0.1 --mac -i 10.1.1.0/24 --topo=single,2

​

2) Run the RYU simple monitor application

ryu-manager ryu.app.simple\_monitor\_13

​

3) Perform TCP Traffic test between h1 and h2

4) see the ryu manager terminal, which prints the port stats at regular interval

​

datapath         in-port eth-dst           out-port packets bytes

---------------- -------- ----------------- -------- -------- --------

0000000000000001       1 00:00:00:00:00:02       2   701403 33029276718

0000000000000001       2 00:00:00:00:00:01       1   530580 35018712

datapath         port     rx-pkts rx-bytes rx-error tx-pkts tx-bytes tx-error

---------------- -------- -------- -------- -------- -------- -------- --------

0000000000000001       1   701414 33029277624       0   530610 35022624       0

0000000000000001       2   530591 35019610       0   701440 33029281104       0

0000000000000001 fffffffe       0       0       0       0       0       0

​

​

## 7. References

* <https://ryu.readthedocs.io/en/latest/app/ofctl_rest.html>

**Theory about group table**

* <https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/7995427/How+to+Work+with+Fast-Failover+OpenFlow+Groups>
* <https://ryu.readthedocs.io/en/latest/app/ofctl_rest.html#add-a-group-entry>

**RYU Example**

* <https://wildanmsyah.wordpress.com/2018/01/13/multipath-routing-with-load-balancing-using-ryu-openflow-controller/>
* <https://github.com/wildan2711/multipath/blob/master/ryu_multipath.py>