**CCGC 5002- SDN Lab 5-2 Meter table & Flow Manager**

**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

A switch element that can measure and control the rate of packets. The meter triggers a meter band if the packet rate/byte rate passing through the meter exceeds a predefined threshold. If the meter band drops the packet, it is called a Rate Limiter.

A meter table consists of meter entries, defining per-flow meters. Per-flow meters enable OpenFlow to implement various simple QoS operations, such as rate-limiting.

## 2. Test Setup

Meter table feature is released in openvswitch 2.10+ versions. Ubuntu 18.10 OS has support the OVS 2.10 in its repo.

So, To test the meter table, Either we need to install OVS 2.10 version in our OS manually. (or) use ubuntu 18.10 OS.

In Ubuntu 18.10 OS, install the openvswitch as below

sudo apt-get install openvswitch-switch

## 3. Testing without meter

1. Run the mininet simple topology

*sudo mn --controller=remote,ip=127.0.0.1 --mac --switch=ovsk,protocols=OpenFlow13 --topo=single,2*

1. Run the RYU Application

*ryu-manager ryu.app.simple\_switch\_13*

1. start the UDP server in h1 and h2

mininet> h2 iperf -u -s &

mininet> h1 iperf -u -s &

1. Perform the 10Mbps UDP Traffic test from h1 to h2 It means h1 pushing 10Mbps traffic to h2. h2 receives this traffic.

mininet> h1 iperf -u -c h2 -b 10m

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

Client connecting to 10.0.0.2, UDP port 5001

Sending 1470 byte datagrams, IPG target: 1176.00 us (kalman adjust)

UDP buffer size: 208 KByte (default)

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

[ 3] local 10.0.0.1 port 51179 connected with 10.0.0.2 port 5001

[ ID] Interval       Transfer     Bandwidth

[ 3] 0.0-10.0 sec 11.9 MBytes 10.0 Mbits/sec

[ 3] Sent 8505 datagrams

[ 3] Server Report:

[ 3] 0.0-10.0 sec 11.9 MBytes 10.0 Mbits/sec   0.000 ms   0/ 8505 (0%)

[ 3] 0.00-9.96 sec 30 datagrams received out-of-order

mininet>

1. Perform the 10Mbps UDP Traffic test from h2 to h1

It means h2 pushing 10Mbps traffic to h1. h1 receives this traffic.

mininet> h2 iperf -u -c h1 -b 10m

​

[ 3] local 10.0.0.2 port 51337 connected with 10.0.0.1 port 5001

[ ID] Interval       Transfer     Bandwidth

[ 3] 0.0-10.0 sec 11.9 MBytes 10.0 Mbits/sec

[ 3] Sent 8505 datagrams

[ 3] Server Report:

[ 3] 0.0-10.0 sec 11.9 MBytes 10.0 Mbits/sec   0.000 ms   0/ 8505 (0%)

mininet>

​

Now both directions we are able to achieve 10Mbps traffic, as no meter is configured.

## 4. Testing with meter

**Objective**

Apply the rate-limit for incoming traffic of host h1 (MAC adddress - 00:00:00:00:00:01).

**Steps**

1. Run the mininet simple topology

*sudo mn --controller=remote,ip=127.0.0.1 --mac --switch=ovsk,protocols=OpenFlow13 --topo=single,2*

1. Run the RYU Application

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

1. Add the Meter

API Details:

<https://ryu.readthedocs.io/en/latest/app/ofctl_rest.html#add-a-meter-entry>

Configuring 1000Kbps rate limit

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

1. Add the flows

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

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

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

In the switch\_flow1, we are applying the meter rate limit. it means, h1 host is applied with rate limit.

H1 host receives only 1000kbps traffic.

1. Check the flow tables

​

tanvir@tanvir:~$ sudo ovs-ofctl -O OpenFlow13 dump-flows s1

cookie=0x0, duration=36.097s, table=0, n\_packets=0, n\_bytes=0, priority=100,dl\_dst=ff:ff:ff:ff:ff:ff actions=FLOOD

cookie=0x0, duration=28.766s, table=0, n\_packets=0, n\_bytes=0, priority=100,dl\_dst=00:00:00:00:00:01 actions=meter:1,output:"s1-eth1"

cookie=0x0, duration=14.650s, table=0, n\_packets=0, n\_bytes=0, priority=100,dl\_dst=00:00:00:00:00:02 actions=output:"s1-eth2"

tanvir@tanvir:~$

​

​

​

1. check the meter table

tanvir@tanvir:~$ sudo ovs-ofctl -O OpenFlow13 dump-meters s1

OFPST\_METER\_CONFIG reply (OF1.3) (xid=0x2):

meter=1 kbps bands=

type=drop rate=1000

tanvir@tanvir:~$

1. Start the UDP server in h1 and h2

mininet> h2 iperf -u -s &

mininet> h1 iperf -u -s &

1. Perform the 10Mbps UDP Traffic test from h1 to h2

It means h1 pushing 10Mbps traffic to h2. h2 receives this traffic.

mininet> h1 iperf -u -c h2 -b 10m

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

Server listening on UDP port 5001

Receiving 1470 byte datagrams

UDP buffer size: 208 KByte (default)

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

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

Client connecting to 10.0.0.2, UDP port 5001

Sending 1470 byte datagrams, IPG target: 1176.00 us (kalman adjust)

UDP buffer size: 208 KByte (default)

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

[ 3] local 10.0.0.1 port 45492 connected with 10.0.0.2 port 5001

[ ID] Interval       Transfer     Bandwidth

[ 3] 0.0-10.0 sec 11.9 MBytes 10.0 Mbits/sec

[ 3] Sent 8505 datagrams

[ 3] Server Report:

[ 3] 0.0-10.0 sec 11.9 MBytes 10.0 Mbits/sec   0.000 ms   0/ 8505 (0%)

mininet>

​

We have not restricted the h1 outgoing traffic. we restricted only on h1 incoming traffic. hence we see 10mbps going.

1. Perform the 10Mbps UDP Traffic test from h2 to h1

It means h2 pushing 10Mbps traffic to h1. h1 receives this traffic.

This has to be rate limited by the Meter , as this traffic is target to h1(incoming trafffic)

mininet> h2 iperf -u -c h1 -b 10m

[ 3] Server Report:

[ 3] 0.0-10.3 sec 1.39 MBytes 1.14 Mbits/sec   0.000 ms 7512/ 8504 (0%)

​

​

1. Check the meter table

*ovs-ofctl -O OpenFlow13 meter-stats s1*

tanvir@tanvir:~$ sudo ovs-ofctl -O OpenFlow13 meter-stats s1

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

meter:1 flow\_count:1 packet\_in\_count:19531 byte\_in\_count:29520582 duration:1093.735s bands:

0: packet\_count:15715 byte\_count:23761080

**\*\*\*\*\*\*\*\*Take screenshot of both above and mininet to show that traffic has been blocked/dropped****. If you do not see appropriate result, run ofctl rest app with simple switch app.**

# Flow Manager Application

## 1. Introduction

<https://github.com/martimy/flowmanager> The FlowManager is a RYU controller application that gives the user manual control over the flow tables in an OpenFlow network. The user can create, modify, or delete flows directly from the application. The user can also monitor the OpenFlow switches and view statistics. Author: Martimy.

Flowmanager does not have inbuilt switching application. So, if user want switching operations(reactive switching), run switching application(simple switch, l3, l4 switch, .. etc) along with flowmanger.

## 2. Installation

Just clone or download the flowmanger application from github.

sudo apt install git

*git clone*[*https://github.com/martimy/flowmanager*](https://github.com/martimy/flowmanager)

## 3. Quick Demo

### Demo

1. Start the RYU flow manager application

*ryu-manager --observe-links ~/flowmanager/flowmanager.py ryu.app.simple\_switch\_13*

1. Run Mininiet topology

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

1. Open the flow manager application UI in Google Chrome Browser (use Firefox if chrome is not installed)

[*http://localhost:8080/home/index.html*](http://localhost:8080/home/index.html)

1. do pingall from mininet

*mininet>pingall*

1. Check the topology diagram

A picture containing map

Description automatically generated

1. Check the Flows (you may need to refresh the browser)

Graphical user interface

Description automatically generated

1. Check the Switch & Ports

Graphical user interface, application, table

Description automatically generated

## 4. Flow Operations

Flow control tab provides the Add, Modify , Delete Flow operations.

### Add a flow

Lets add a flow S1 to drop all packets generated by 10.1.1.1 host.

Select Add Operation -Select Match ipv4\_src 10.1.1.1 eth\_type 0x0800 (using Flow Control tab, 0x0800 is ethernet type for IP protocol)

Eth\_type is 2048 from the ryu ofctl documentation. The hex equivalent is 800.

* priority 1000
* action  
  output port -1 (default port)

Graphical user interface, website

Description automatically generated

After you submit, check the flows. You won’t be able to ping h3 or h4 from h1 since s1 will drop them. Ping h2 from h1 in mininet should work. Show that packets are being dropped for h1 from h3 or h4 since the root switch will drop them. \*\*\*\*\*\*\*Take screenshot of the mininet screen.

### Delete a Flow with exact match

* Select the Delete operation
* Provide the Match fields (all fields)
* Provide the output Port
* submit it

Now check the flows. You may delete the exact flow you added before and show that the h1 can ping h3 now. \*\*\*\*\*Take screenshot of mininet.

### Delete flows with only output port number

* Select the Delete operation
* Select Match Any
* Provide the output Port
* submit it

Graphical user interface, application

Description automatically generated

* Now check the flows. \*\*\*\*\*take screenshot of Flows to show that the Instructions column does not have any flow OUTPUT:1.

### Delete all flows

Select the Delete operation

* Select Match Any
* Provide the output Port as (-1)
* submit it

Graphical user interface

Description automatically generated

* Now check the flows. \*\*\*\*\*take screenshot of Flows to show that all flows are removed (-1 represents all).

Table Miss entry also will be removed. So , we need to destroy the topology and start again to continue the exercises.

## 5. References

<https://martimy.github.io/flowmanager/>

<http://adhocnode.com/building-openflow-lab/>

<https://ryu.readthedocs.io/en/latest/app/ofctl_rest.html#add-a-meter-entry>

<https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/15040523/How+to+Use+OpenFlow+Meters>

<https://www.talentica.com/blogs/using-meters-to-implement-qos-in-opendaylight/>