Technical University of Munich, Chair of Communication Networks
Prof. Dr.-Ing. W. Kellerer, Amaury Van Bemten, Nemanja Djeric

Software Defined Networking Lab
SS 2019

Tutorial 2

Tutorial 2 - SDN Firewall

The goal of the tutorial is to get familiar with writing a ryu application and learning about different types of openflow messages and their usage.

For more information about writing ryu application please refer to Writing RYU Application. Furthermore, it is also useful to check implementations of other ryu applications at Examples of different ryu applications.

In this tutorial, you will use the topology shown in Fig. 1. It represents a simplified topology of TUM, where faculty buildings in Munich and Garching are connected via one backbone link connecting main OpenFlow enabled switch in Munich 'm' and main OpenFlow enabled switch in Garching 'g'. Furthermore, two more OpenFlow enabled switches are installed at each cite, one for student pcs and one for professor pcs. Since the switches are OpenFlow enabled, you will use ryu-app for implementation of the functionalities specified in Problems section. Example of one host notation in our topology is 'g_p1' where 'g' means that host is located in Garching, while 'p' means that the host is used by professor.

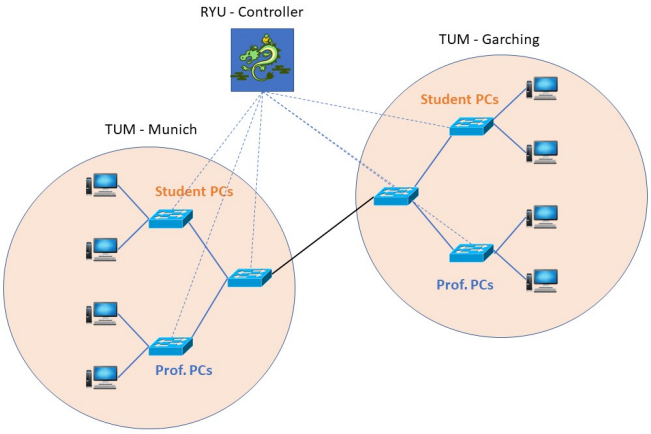

Figure 1: Amazing SDN lab topology.

On moodle page, you can find the compressed files (topology and application). Download the file to ryu/ryu/app and uncompress them with:

```
tar -xvf
```

You can boot up the topology by running:

```
sudo mn --custom PATH_TO_TOPO/sdn_firewall_topo.py --topo mytopo
--controller remote --mac --link tc
```

You should edit sdn_firewall_app.py (application is using OF1.0) and you can run it by doing:

```
ryu-manager PATH_TO_APP/sdn_firewall_app.py
```

Useful links and materials:

- Writing first ryu-app
- Ryu protocol API for OF1.0
- Ryu Monitor for OF1.3

1 Separating networks (Flow Mod)

In order to increase the security we would like to separate students and professors networks. The goal here is to edit ryu sdn_firewall_app.py so that students cannot communicate with professors.

You should write a function which checks if a packet should be able to reach the end host in sdn_firewall_app.py at comment line #Task 1: Write a function which checks if hosts have permission to communicate with each other. Call it correctly at the comment line #Task 1: Call the function. Hint: you can use MAC addresses for differentiation between hosts.

Checking the code:

- Run the mininet topology and your edited ryu app,
- Make sure hosts are only able to ping the desired hosts,
- Run iperf with default settings between a few randomly selected hosts,
- Dump rules on corresponding switches using ovs-ofctl.

2 Blocking specific traffic (Drop rule, Priority)

In this task you will have to write a function that blocks the traffic based on the used tcp port. Students should not be able to use http traffic between them, while professors should not be able to use SSH traffic in their network. In order to achieve this you will have to expand the application with writing a function which drops all packets with a specific TCP port. The definition of a function is provided at the comment line name #Task 2: Blocking traffic based on TCP port. Don't forget to call a function in the appropriate place in application and for appropriate switches / users.

Useful hints:

- Default match action is drop,

- In order to have a match on TCP port we also have to specify network protocol (IANA Protocol numbers,
- Differentiate students and professors traffic with either IPs/MACs/switch datapath,
- Make sure that you are using a higher priority for matching on TCP port than for the rest of the traffic.

Checking the code:

- Run the mininet topology and your edited ryu app,
- (Show that traffic on specific port is blocked) Run iperf with a http port on one student host, try connecting from another student host,
- (Show that other traffic is not blocked) Use iperf with default settings,
- Repeat this also for different TCP port on prof. hosts,
- Dump rules on corresponding switches.

3 Monitoring and Limiting data usage (Drop rule, Priority, Timeout, PortStats, FlowStats)

Unfortunately, networks do have a limited amount of resources, so you will have to make sure that students and professors are not using too much network resources. Students are allowed only to send 1GB of data every 1 minute, while professors are able to send 5GB every 2 minutes.

In order to do this you have to write a function which request switch statistics (PortStats). Find the comment line #Task 3a: Requesting port stats for a specific datapath and implement sending Port Stats message to a specific switches. You will also have to process the answers and store received statistics in the function #Task 3b: Process port stats reply. Furthermore, in this function you should also check if someone is using more resources than allowed.

Monitoring and blocking should be done every 5s and called in _monitor function.

Simplified version of this task would be to only limit the traffic per switch and not per user.

Useful hints:

- Use a hard timer function for drop rules so that user can send data in the next time interval,
- Gathering port statistics is sufficient, for more granularity you can use flow stats.

Checking the code:

- Run the mininet topology and your edited ryu app,
- Using logger to print gathered monitoring information,
- Run iperf and check if the user is blocked after using too much resources,

- Dump rules on switches via ovs-ofctl,
- Check if the user is allowed to send in the next time interval.