SDN Fundamentals and Techniques

SDN and Openflow Demos

I. Setting things up

In this course, we will be using ONOS as an SDN controller along with some dependencies: Docker, OVS, and Linux Machine. It is noted that some networking knowledge, familiarity with Linux shell bash, and python languages are considered as prerequisites. Let's start by setting up the environment.

ONOS is officially supported by the Open Networking Foundation (ONF) and runs on Linux machines and containers, i.e., LXC, LXD, and Docker. If you do not have any of them on your machine, you can either set up a virtual machine (using VirtualBox or VMWare) or use one of Aalto servers (kosh.aalto.fi or lyta.aalto.fi). We used Ubuntu versions 18.04 and 20.04 for testing, so it is recommended to use one of these versions.

- All files referred to in this section are available on MyCourses under section Demos.
- We recommend using ONOS as a Docker image and we provide a simple shell script to instantly get ONOS ready. Please refer to the file "Install ONOS".
- Docker is required for the installation of ONOS, please refer to the "Install Docker & OVS" file for installation guidelines. Please change the mode of the shell script to add execution, i.e., chmod +x your_file.sh.
- After successfully installing ONOS, a link should appear in your terminal. To be able to access ONOS, the default ONOS credentials are – username: onos, password: rocks.
- To access the ONOS API, access the following link: http://ONOS-IP:8181/onos/v1/docs/

II. Demos

Demo 5: Using ONOS RESTful API interface to manage hosts, devices, applications and settings

1.1. Introduction

This assignment aims to experience and understand the basic functionalities of the SDN controller, ONOS, through the use of its RESTful API. This demonstration is part of the application layer as it introduces students to general orchestration methods and programmability in SDN-based environments. By doing this assignment, the students will be able to grasp the building blocks allowing the creation of complex ONOS RESTful-based applications and extract different information related to devices, hosts, networking, topologies, settings, and applications.

1.2. Pre-Task

Create a python-based code or use existing python client to use ONOS RESTful API. Please ensure that your code is orthogonal and can handle changes of passwords and user-names.

1.3. Tasks

1.3.1. Task 1

Unlike other demonstrations, in this particular demo, the ONOS applications are not yet activated and you are not allowed to start them manually. Thus, as an initial task, you are asked to start the required ONOS applications using a python-based approach. In your solution, you may use the default ONOS credentials, i.e., username: onos, password: rocks, to access ONOS RESTful API. The list of applications to be started can be found in the following file “start_applications” in demo1.

Task 2

The following task is provided with a pre-defined topology, i.e., use script file “demo5.sh”:

- Create a python program to list all available devices by their IDs.
- Create a python program to get the IP management address and the OpenFlow version used by a given device in the pre-defined architecture.
- Create a python program using the same device id, i.e., used in the previous question, to get the currently active MAC addresses and the Port names.

1.3.2. Task 3

Considering the same pre-defined topology used in Task 1, students are asked to:

- Create a python program to list all available hosts by their id, MAC address, and IP address.

- Create a python program to get the device id and the port used by the host having “10.0.0.130" as an IP address in the pre-defined architecture.
- Create a python program using the same host id, i.e., used in the previous question, to remove the host from the pre-defined architecture.
- Ping the removed host, what do you observe?

1.3.3. Task 4

Considering the same pre-defined topology of Task 1, students are asked to:

- Create a python program to list all ACTIVE links in the pre-defined topology, the output should be a table containing device id source, port source, device id destination, port destination.
- Create a python program to list all the flows applied to a device of your choice, the output may show the flow-id, the application id, the device id, and the instructions.
- Create a python program to list all intents.

Demo 6: Using ONOS RESTful API to filter, mirror, and forward networking traffic based on OpenFlow capabilities

2.1. Introduction

This assignment shall allow students master Openflow flow rules and their applications to practical scenarios. In contrast to the demonstration dubbed “OpenFlow flow tables", i.e., demo 4, this demonstration is part of the application layer as it operates at a higher level while allowing the manipulation of SDN-enabled switches. By doing this assignment, the students will be able to automate the creation, deletion, and updates of OpenFlow rules and understand the advantages and limitations of OpenFlow-based orchestration systems.

2.2. Pre-Task

Use the python code created in the previous demonstration to access ONOS RESTful API interface and activate the required ONOS applications. The list of required applications may be found in the following file “start_applications.txt” in demo1.

2.3. Tasks

This assignment is similar to the demonstration related to OpenFlow rules in demo 4. Thus, it will be based on two tasks. The first task is provided with a pre-defined topology in which students are asked to create flow rules to achieve the defined objective using the ONOS RESTful API interface. The second task is a set of queries for listing and deleting OpenFlow flow rules.

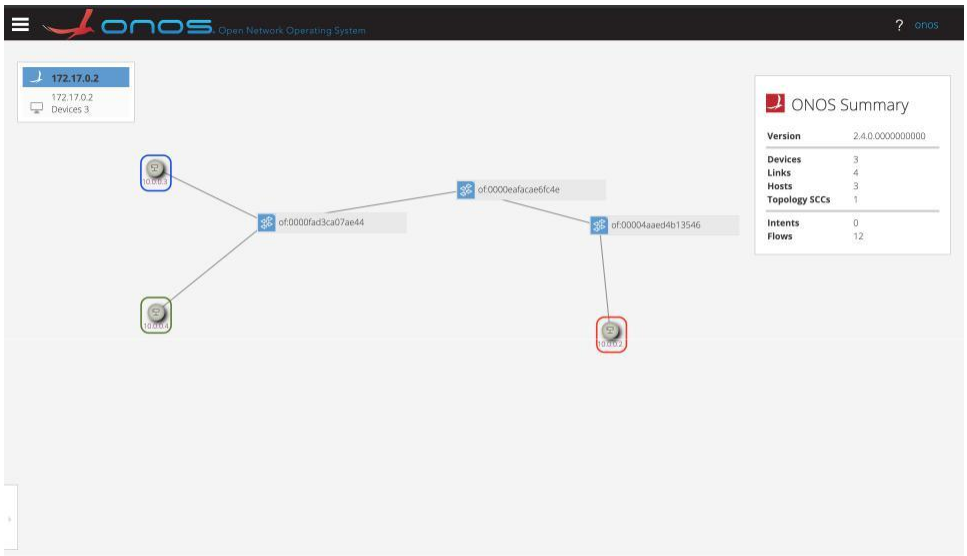

Figure 1: Task1 Topology.

2.3.1. Task 1

The file “demo6.sh” related to Task 1 is available on MyCourses. This shell script creates a topology as depicted in Fig. 1. A host in the ONOS’s GUI is a namespace, however, LXC or LXD containers can be used. In this task you are asked to:

- Create a flow rule to mirror (copy) the traffic between the “Red" and the “Blue" namespaces, i.e., consider only traffic from “Red" to “Blue", to also be sent to the “Green" namespace.
- Create a flow rule to block the ICMP traffic from the “Blue" namespace to the “Red" namespace.
- Create a flow rule to block the traffic to the “Red" namespace.
- Create a flow rule to allow total access to the “Red" namespace from the “Green" and the “Blue" namespaces.
- Do you need to delete the previously created flow rules to activate flow rule number 3?

- Create a flow rule to allow only Http and Https traffic to the “Red" namespace.

Note that for each flow rule, the student is asked to provide details of the entered rule and explanations. Please use only the ONOS RESTful API interface to create the flow rules.

2.3.2. Task 2

The following task is provided with a pre-defined topology, Fig 1. However, the output must be generalized to cover any kind of topologies. Students are asked to:

- Create a python program to list all flow rules per device id, the code must be generic for any given network topology.
- Create a python program to list flow rules by application id.
- Create a python program to delete flow rules knowing the device id and the flow-id.

Please use only the ONOS RESTful API interface to complete Task 2.

Demo 7: Using ONOS RESTful API to filter, mirror, and forward networking traffic based on ONOS’s Intents framework

3.1. Introduction

This assignment aims to examine, test, and understand the ONOS intent framework features, through the use of the ONOS RESTful API. This demonstration is part of the application layer as it introduces students to general orchestration methods and programmability in SDN-based environments. By doing this assignment, the students will be able to differentiate Intents based networking from basic OpenFlow-enabled networks and legacy networking systems. Students will also understand the utility of using abstracted networking approaches for fast and efficient networking development.

3.2. Pre-Task

Please use the python code created in the previous demonstration to access ONOS RESTful API interface and activate the required ONOS applications. The list of required applications may be found in the following file “start_applications.txt” in demo1.

3.3. Tasks

This assignment is a series of tasks, each one of them will allow the students to experience the use of Intents in SDN environments. All tasks are performed in a pre-defined topology in which students are asked to create various Intents to achieve the defined objective.

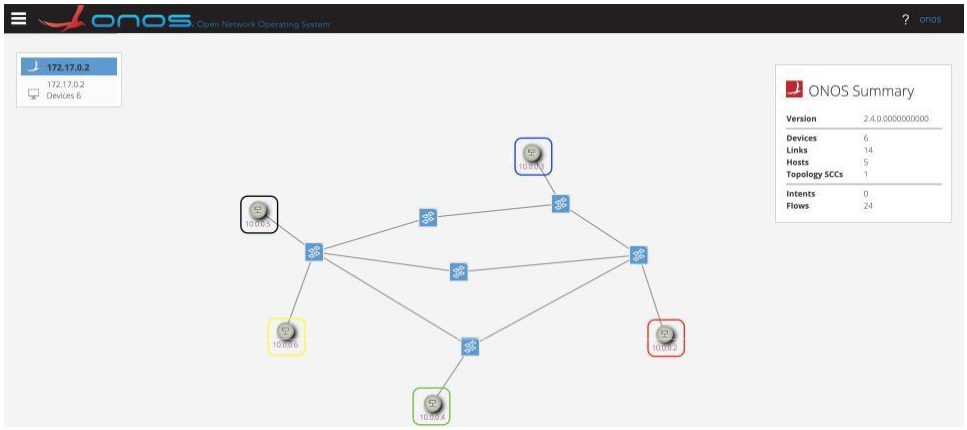

Figure 2: Task Topology.

3.3.1. Task 1

The file "demo7.sh" related to Task 1 is available in Mycourses. This shell script creates a topology as depicted in Fig. 2. A host in the ONOS's GUI is a namespace, however, LXC or LXD containers can be used. In this task you are asked to:

- Create a series of point-to-point Intents, using Python-based codes and ONOS RESTful API, to allow communication between the "RED" network namespace and the "BLACK" network namespace.
- Delete all the created Intents using Python-based code and ONOS RESTful API.
- Create a Host-to-Host Intent to allow communication between the "RED" network namespace and the "BLACK" network namespace.
- Is the Host-to-Host Intent an abstraction of the Point-to-Point Intents? Your answer must be provided with explanations.
- What path is selected by the Host-to-Host Intent for enabling the communication between the "RED" network namespace and the "BLACK" network namespace?
- After specifying the path, you are asked to provide a hypothesis on how the Host-to-Host Intent selects paths.
- Using Host-to-Host Intents, enable the communication between all network namespaces in the topology.
- Without deleting the current Intents, you are asked to create a Single-to-Multi point Intent to allow communication between the "RED" network namespace and "BLACK", "BLUE", "GREEN" network namespaces. Only communication on port 4009 should be allowed. After creating this Intent students are requested to verify using "nc" utility or any similar program.

- Explain the benefits brought by the use of Intent-based networking compared to Open-Flow flow rules.

Note that for each Intent, the student is asked to provide details and explanations. Please use only the ONOS RESTful API interface to enter the Intents.

III. Submission Guidelines

The deadline to submit the solutions through MyCourses are as shown below. Your submission should contain answers to the questions asked in the text, the code with solutions used for the assignment. If you need help or clarification solving the exercises, you are welcome to ask between 10:00 and 12:00 on Thursdays.

- Nait Abbou Aiman aiman.naitabbou@aalto.fi

The deadline to submit your answers for Demo 5 27.03.2022, 11:55 p.m.

The deadline to submit your answers for Demo 6 30.03.2022 11:55 p.m.

The deadline to submit your answers for Demo 7 03.04.2022, 11:55 p.m.

For delayed submissions, 15% of the total score allocated for this assignment will be deducted.

Submissions of Demo 5 made after 30.03.2022 will not be accepted.

Submissions of Demo 6 made after 02.04.2022 will not be accepted.

Submissions of Demo 7 made after 06.04.2022 will not be accepted.

Good luck!