SDN Fundamentals and Techniques

1. Introduction

The main objective of this assignment is to get some intuition behind the basic infrastructure layer operations by creating topologies using Linux namespaces. Finalizing this assignment will allow the understanding of low-level components, i.e., namespaces, related to the infrastructure layer, their advantages and limitations, and their usage in well-known open-source projects, e.g., Kubernetes and Openstack.

2. 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 networking knowledge, familiarity with Linux shell bash, and python languages are considered as prerequisites. Let's start by setting up everything.

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 assignment are available on MyCourses.
- 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.

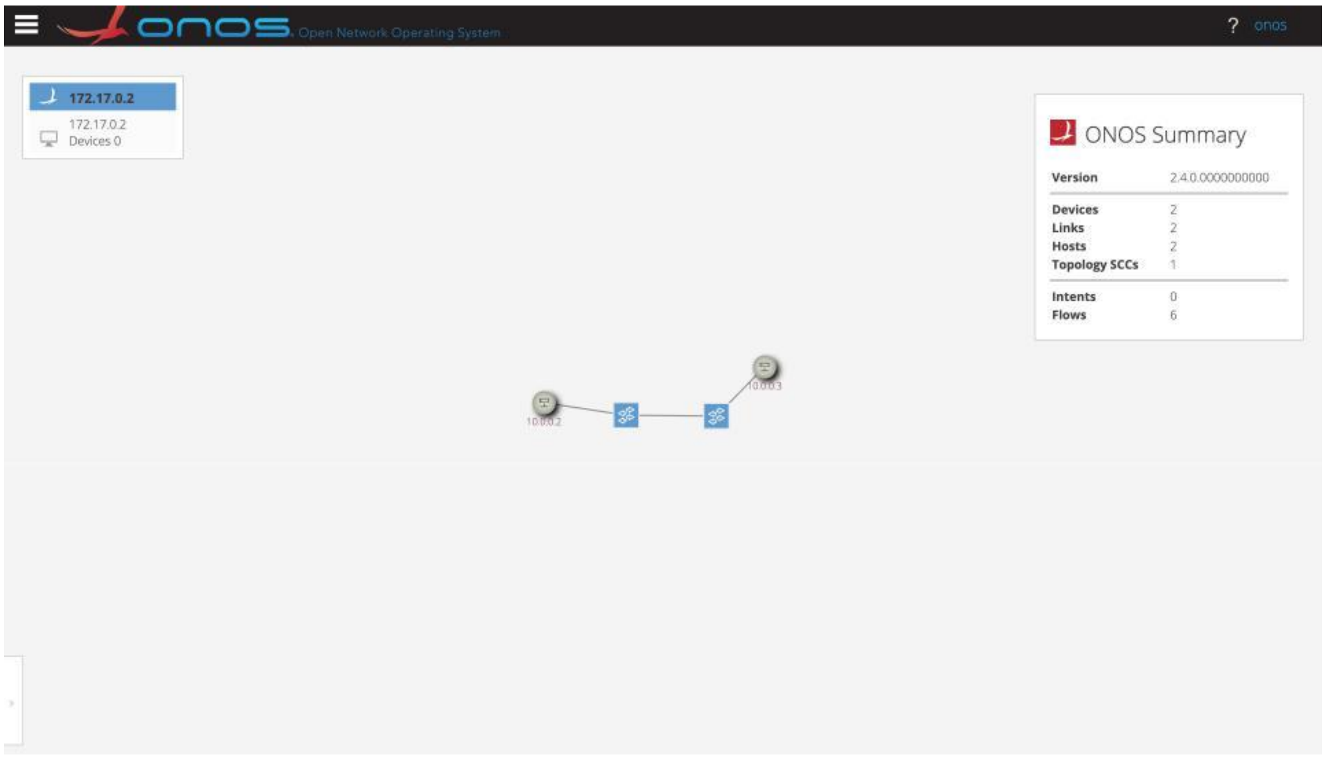
The next step is to start ONOS applications. For this purpose, you may refer to the file "Using ONOS" and follow the instructions therein. Those are activated as this assignment assumes that the application and the controller layers are outside the scope of this demonstration.

3. Tasks

Demo 2 - ONOS and Linux namespace

The shell script basic-net-ns.sh contains a set of shell functions to create a simple topology wherein two switches are connected together, each one of them is connected to one host, i.e., Fig. 1. In our assignment, a host is represented by a network namespace. This code is a simple manner of creating components of the infrastructure layer.

Figure 1: ONOS example.



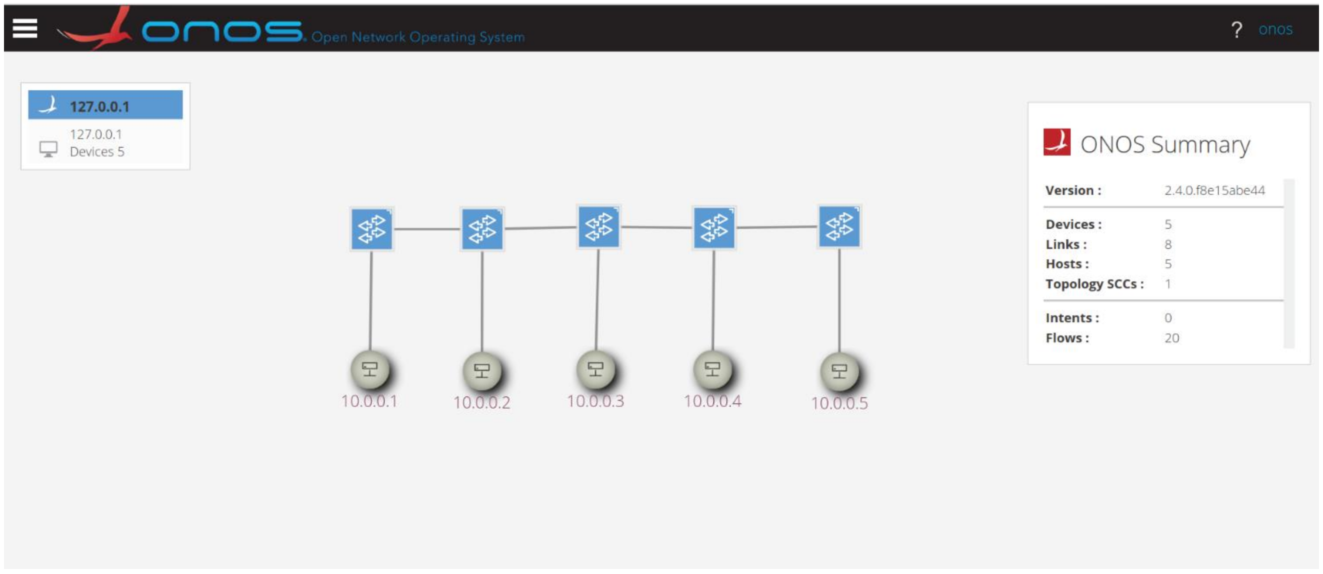
ONOS Summary

Version	2.4.0.0000000000
Devices	2
Links	2
Hosts	2
Topology SCCs	1
Intents	0
Flows	6

In this assignment, you are requested to carry out the following tasks:

Task 1.1

Create a linear topology of 5 switches and 5 hosts. In a linear topology, each switch has two connections with other switches except the first and the last ones. To do that you may write a shell script or a python-based code, your code must be well-commented and following coding principles and naming convention (check the basic-net-ns.sh to get inspired).



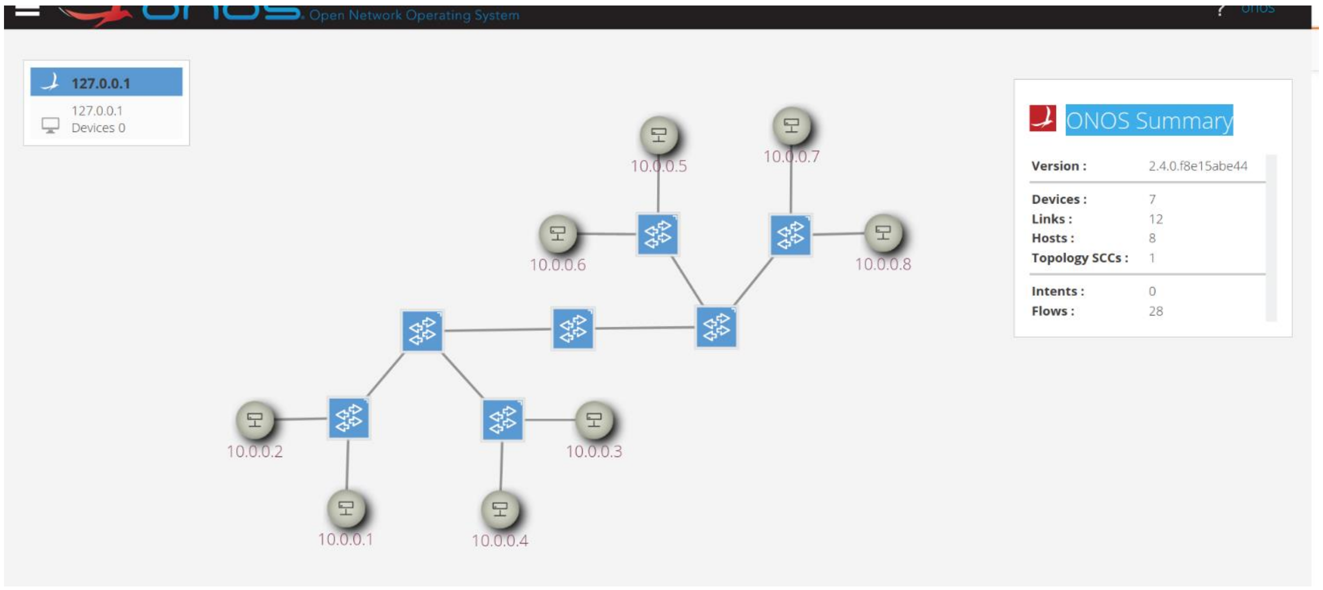
ONOS Summary

Version :	2.4.0.f8e15abe44
Devices :	5
Links :	8
Hosts :	5
Topology SCCs :	1
Intents :	0
Flows :	20

Figure 2: Example of a Linear topology with 5 switches and 5 hosts.

Task 1.2

Create a tree topology depth 3 fanout 2. In a tree topology, the depth refers to the number of layers in a tree and the fanout is the number of children each node has. For instance, in a tree where depth and fanout are equal to 2, the tree will have a node 0, i.e., switch 0, that has two other switches connected to it, switches 1 and 2.



ONOS Summary

Version :	2.4.0.f8e15abe44
Devices :	7
Links :	12
Hosts :	8
Topology SCCs :	1
Intents :	0
Flows :	28

Figure 3: Example of a Tree topology with a depth of 3 and a fanout of 2.

Task 1.3

Try to automate the precedent tasks to be able to create a topology given the type, i.e., tree or linear, and for each type its specifications (i.e., depth and fanout for the tree and number of switches for linear). It is worth noticing that you can use Python as it is preferred because of the available tools that allow the exploitation of networking concepts.

Demo 3 - ONOS and Linux Containers

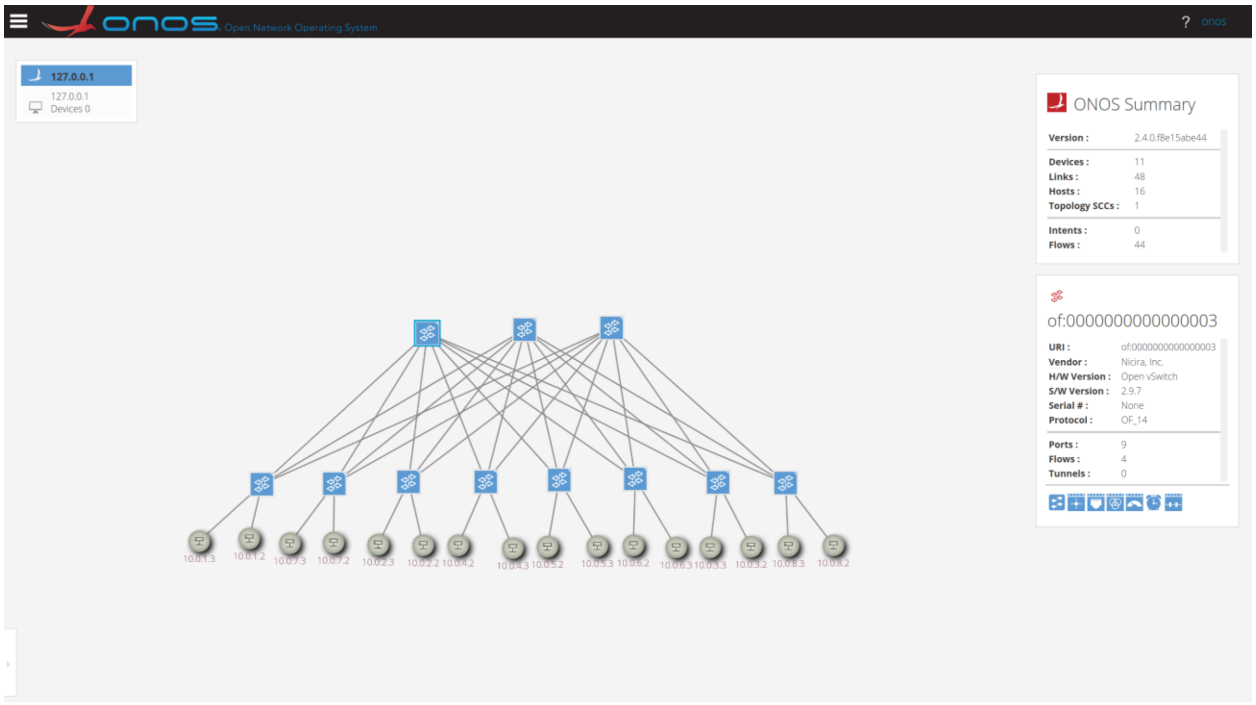
This assignment contains example using LXC as a container implementation, The file related to LXC containers is stored in ONOS_LXC.

The python script creates a simple topology wherein two switches are connected together, each one of them is connected to one host, i.e., Fig. 1. A host in the ONOS's GUI a LXC container. This code is a simple manner of creating components of the infrastructure layer.

In this assignment, you are requested to carry out the following tasks:

Task 2.1

Create a leaf and spine topology in which the number of switches in the leaf layer is equal to eight, i.e., 8, and the number of switches in the spine is three, i.e., 3 (See Fig. 4). Each leaf node should contain two, i.e., 2, Linux Containers. It is noticed that a Leaf-spine topology is a two-layer network topology composed of leaf switches and spine switches. Each switch in the spine layer is connected to all the leaf switches. This topology is highly considered in data-centers. The task should be done using LXC containers.



ONOS Summary

Version :	2.4.0.f8e15abe44
Devices :	11
Links :	48
Hosts :	16
Topology SCCs :	1
Intents :	0
Flows :	44

of:0000000000000003

URI :	of:0000000000000003
Vendor :	Nicira, Inc.
H/W Version :	Open vSwitch
S/W Version :	2.9.7
Serial # :	None
Protocol :	OF_14
Ports :	9
Flows :	4
Tunnels :	0

Figure 4: Example of a Leaf Spine topology with eight leaf and three spine switches.

Task 2.2

Work on the automation aspect of Task 2.1. The final results should be a code asking for the number of leaves, spine, and host nodes while the output must generate, in an automated way, the expected topology.

Task 2.3

Compared to namespaces (Demo 2), do you find manipulating low-level containers hard to implement and understand or not? Please elaborate why?

4. Submission Guidelines

The deadline to submit the solutions through MyCourses is on **17.03.2021** at 23:55 for Demo 2 and Demo 3. 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, between 10:00 and 12:00 on Thursdays (17 of march).

- Nait Abbou Aiman aiman.naitabbou@aalto.fi

The deadline to submit your answers for Demo 2 and Demo 3 is 17.03.2021, 11:55 p.m.

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

Submissions of Demo 1 and Demo 2 made after 20.03.2022 will not be accepted.

Good luck!