

21/01/2024

Telecommunication Software
Report 6

Jehanne Sarrazin

230AEM053

For this final practical work, we had to install Mininet, OpenvSwitch (OVS) and Floodlight on our Linux system.

Unfortunately I ran into some trouble with the installation of OVS, as it wouldn't recognize python3 as python and I spent a lot of time trying to fix it by redirecting the python command to python3 as well as the files of the module but after a while, I still couldn't get it to work so I decided to just do this work as best as I could without these softwares. Consequently, Mininet didn't work and I couldn't install Floodlight.

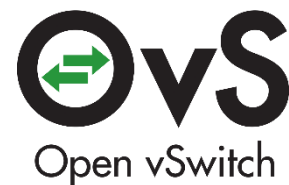
In this report you will find explanation on how these software works and how to build the expected topology on a Linux system.

1. The softwares :



Mininet is a tool used to easily create a virtual network on a machine using a command line system. It uses OpenvSwitch.

Open vSwitch is a software that implements a virtual multilayer switch. It allows to create and manage complex network topologies on a machine or across servers.

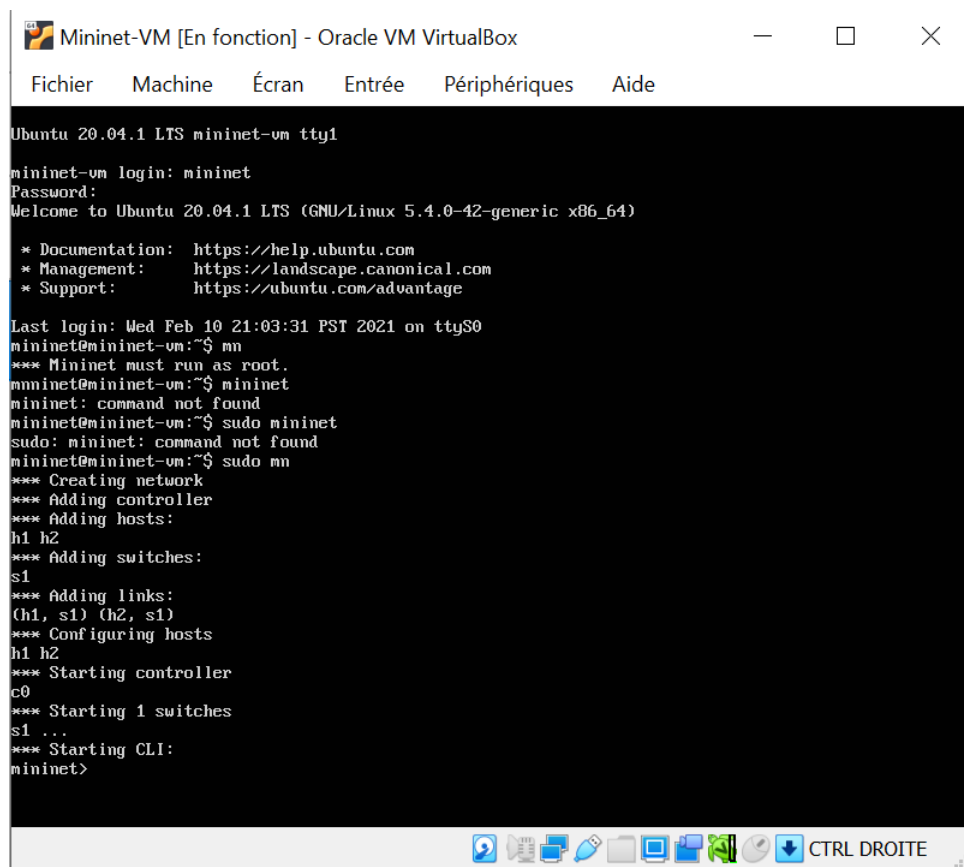


Floodlight Controller implements functions for controlling an OpenFlow network. It also offers a cleaner view of the network for the user.

All these softwares are used in SDN, which stands for software defined networking. This means using a software-based approach to controlling a network. Usually, networks are controlled from the hardware constituting the network (switches, routers, etc). SDN have the advantages of being more robust and secure, and more importantly more customizable to the user's need. Virtual network can be created inside physical ones and can be much easier to control and manage.

2. **Creating a virtual network**

In this work, Mininet can be used to create custom network easily. After downloading Mininet, we can create a network from its terminal. Again, I couldn't get it to work on my virtual machine, so I downloaded a virtual machine of Mininet to take a few screenshots, despite not being able to dive more into OVS or the python view of what can be used to create networks:



```
Mininet-VM [En fonction] - Oracle VM VirtualBox
Fichier  Machine  Écran  Entrée  Périphériques  Aide

Ubuntu 20.04.1 LTS mininet-vm tty1
mininet-vm login: mininet
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-42-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

Last login: Wed Feb 10 21:03:31 PST 2021 on ttyS0
mininet@mininet-vm:~$ mn
*** Mininet must run as root.
mininet@mininet-vm:~$ mininet
mininet: command not found
mininet@mininet-vm:~$ sudo mininet
sudo: mininet: command not found
mininet@mininet-vm:~$ sudo mn
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

Here for example, I created a default network using “sudo mn”. I soon realized that this would be harder than intended as my keyboard is in French (AZERTY instead of QWERTY) and this virtual machine assumed I was using QWERTY one, meaning that I was typing blindly for the letters I didn’t have in common with a QWERTY keyboard.

Normally with a Linux system, one can create the networks in mininet using Python. For example in a network system, hosts can be added using “name1Host = self.addHost(‘h1’)” and switches using “name1switch = self.addSwitch(‘s1’)”.

The links within the networks can be made using “self.addLink(‘name1Host’, ‘name1switch’)”.

Using python to create a topology has the advantage of being able to create the desired topology more easily (for example using “for” loops to link the different devices of the network).

It can then be created using “sudo mn --custom mesh.py --topo mytopo --controller=remote” (since we’re using remote SDN controllers in this task).

“dump” will show all nodes, “net” will show all the links, and “pingall” will ping all devices with each other to verify the connectivity. One can also use “name1Host ping name2Host” to ping manually each device.

“name1Host ifconfig” to read all about its configuration.

“exit” to quit a network and turn off all devices.

A default network can be created with **"sudo mn"** using mininet using a single SDN controller. **"sudo mn --controller=remote"**, which can be now connected with Floodlight using **"cd floodlight"** and then **"java -jar target/floodlight.jar"**. It will start floodlight and floodlight will connect to the network created with Mininet.

With the SDN session open, **"pingall"** in Mininet should be able to ping all the devices of the network.

Pox is also a SDN controller that can be used instead of floodlight and is python based. It can be used to control the Mininet network from a different virtual machine. It can be turned on with **"cd pox"** and then **"python ./pox.py forwarding.l2_learning"** and will listen for networks. Then in Mininet, we need to specify the ip of the controller using **"sudo mn --controller=remote, ip=10.0.0.12, port=6633"** (change for the correct ip), and the two sessions will connect.

To conclude this report, I'm a bit disappointed I couldn't get to develop the network as expected nor get Floodlight to work on my machine, but I still learned a lot about SDN by reading a lot of articles and videos showing how to develop a network using Mininet, OVS and Floodlight.