Fait par: El M'Rabet Zineb et El Bekkali Wissal

Compte rendu TP Mininet

Objectif du TP

Ce TP a pour but de découvrir Mininet, un émulateur de réseau permettant de créer facilement des topologies virtuelles pour tester des protocoles SDN (Software Defined Networking), souvent en lien avec des contrôleurs comme ONOS.

1. Démarrage de Mininet

La commande de base pour lancer Mininet est : `sudo mn [options]`

Le mot-clé `sudo` est nécessaire car Mininet manipule les interfaces réseau, ce qui demande les privilèges administrateur.

2. Topologie 1 – Single (un seul switch)

Commande :

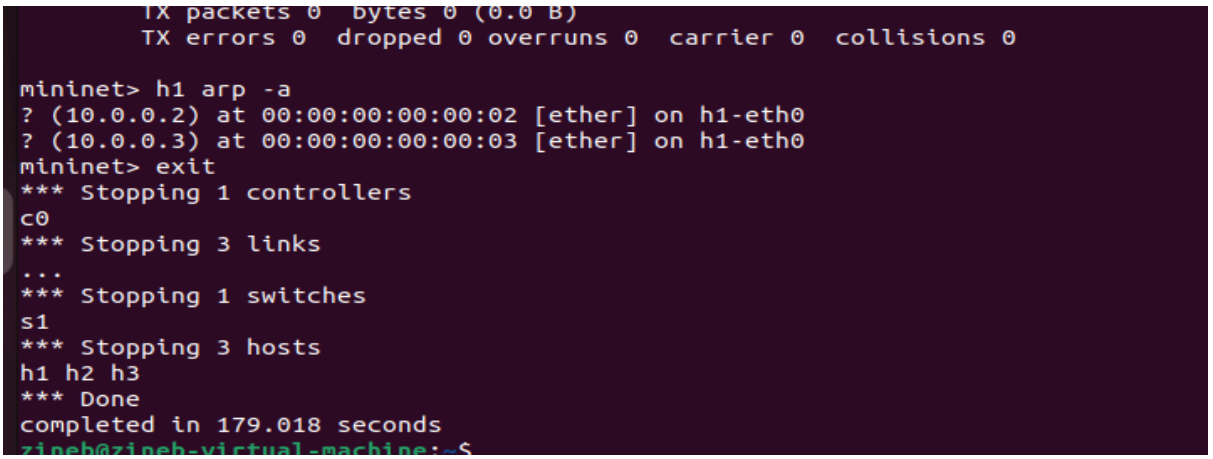
```
sudo mn --topo single,3 --mac --switch ovs
```

Explication :

- `--topo single,3` crée une topologie simple avec 1 switch et 3 hôtes (h1, h2, h3).
- `--mac` assigne automatiquement des adresses MAC uniques.
- `--switch ovs` utilise Open vSwitch comme type de switch virtuel.

```
zineb@zineb-virtual-machine:~$ sudo mn --topo single,3 --mac --switch ovs
[sudo] password for zineb:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
LibreOffice Writer
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> nodes
available nodes are:
c0 h1 h2 h3 s1
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
h3 h3-eth0:s1-eth3
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0 s1-eth3:h3-eth0
c0
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=3105>
<Host h2: h2-eth0:10.0.0.2 pid=3107>
<Host h3: h3-eth0:10.0.0.3 pid=3109>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None pid=3114>
<Controller c0: 127.0.0.1:6653 pid=3098>
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
```

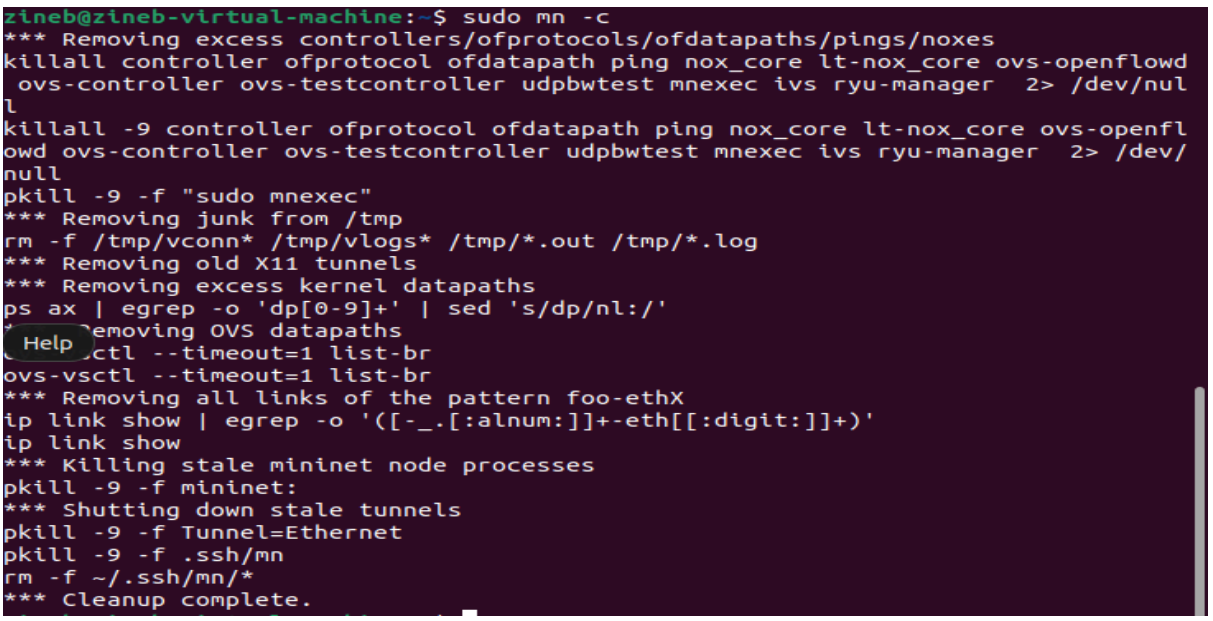
```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet> h1 ping -c 5 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.745 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.127 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.119 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.163 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.124 ms

--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4108ms
rtt min/avg/max/mdev = 0.119/0.255/0.745/0.245 ms
mininet> h1 ifconfig
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.1  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::200:ff:fe00:1  prefixlen 64  scopeid 0x20<link>
        ether 00:00:00:00:00:01  txqueuelen 1000  (Ethernet)
        RX packets 62  bytes 6022 (6.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 26  bytes 2136 (2.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```



Nettoyage :

`sudo mn -c`

Supprime toute topologie encore active avant un nouveau test.



3. Topologie 2 – Linéaire (Linear)

Commande :

`sudo mn --topo linear,3 --mac --switch ovs`

Explication :

Crée une chaîne de 3 switches connectés linéairement (s1 ↔ s2 ↔ s3), avec un hôte à chaque extrémité.

```
zineb@zineb-virtual-machine:~$ sudo mn --topo linear,3 --mac --switch ovs
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (s2, s1) (s3, s2)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
```

Commandes utiles :

- `net` : Vérifie les liens entre les hôtes et les switches.
- `pingall` : Test global de connectivité.
- h1 `ping` h3 : Vérifie la communication bout à bout.
- h3 `ifconfig` et h3 `route` : Informations réseau du dernier hôte.

```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s2-eth1
h3 h3-eth0:s3-eth1
s1 lo:  s1-eth1:h1-eth0 s1-eth2:s2-eth2
s2 lo:  s2-eth1:h2-eth0 s2-eth2:s1-eth2 s2-eth3:s3-eth2
s3 lo:  s3-eth1:h3-eth0 s3-eth2:s2-eth3
c0
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
```

```
mininet> h1 ping h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=0.126 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.139 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.143 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=0.141 ms
^C
--- 10.0.0.3 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3072ms
rtt min/avg/max/mdev = 0.126/0.137/0.143/0.006 ms
mininet> h3 ifconfig
h3-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.3  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::200:ff:fe00:3  prefixlen 64  scopeid 0x20<link>
        ether 00:00:00:00:00:03  txqueuelen 1000  (Ethernet)
        RX packets 162  bytes 16887 (16.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 82  bytes 7368 (7.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
mininet> h3 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        0.0.0.0         255.0.0.0       U     0      0        0 h3-eth0
mininet> links
h1-eth0<->s1-eth1 (OK OK)
h2-eth0<->s2-eth1 (OK OK)
h3-eth0<->s3-eth1 (OK OK)
s2-eth2<->s1-eth2 (OK OK)
s3-eth2<->s2-eth3 (OK OK)
```

4. Topologie 3 – Arborescente (Tree)

Commande :

`sudo mn --topo tree,depth=2,fanout=2 --mac --switch ovs`

Explication :

- `depth=2` → deux niveaux de hiérarchie (un switch racine, deux intermédiaires).
- `fanout=2` → chaque switch est connecté à deux nœuds (soit deux hôtes ou deux switches).

```
zineb@zineb-virtual-machine:~$ sudo mn --topo tree,depth=2,fanout=2 --mac --swit
ch ovs
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3
*** Adding links:
(s1, s2) (s1, s3) (s2, h1) (s2, h2) (s3, h3) (s3, h4)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
```

```
mininet> h1 ping h4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=0.802 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=0.142 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=0.147 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=64 time=0.420 ms
64 bytes from 10.0.0.4: icmp_seq=5 ttl=64 time=0.139 ms
^C
--- 10.0.0.4 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4070ms
rtt min/avg/max/mdev = 0.139/0.330/0.802/0.259 ms
mininet> iperf h1 h4
*** Iperf: testing TCP bandwidth between h1 and h4
*** Results: ['16.5 Gbits/sec', '16.5 Gbits/sec']
```

5. Topologie 4 – Personnalisée (Custom)

Lancement :

`sudo mn --custom mytopo.py --topo mytopo --mac --swi`

```
zineb@zineb-virtual-machine:~$ sudo mn --custom mytopo.py --topo mytopo --mac
switch ovs
[sudo] password for zineb:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1 s2
*** Adding links:
(h1, s1) (h2, s1) (h3, s2) (s1, s2)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...
*** Starting CLI:
mininet> nodes
available nodes are:
c0 h1 h2 h3 s1 s2
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
h3 h3-eth0:s2-eth2
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0 s1-eth3:s2-eth1
s2 lo:  s2-eth1:s1-eth3 s2-eth2:h3-eth0
c0
```

Commandes dans Mininet :

- `nodes` : Liste des hôtes et switches.
- `net` : Affiche les liens.
- `dump` : Détails sur chaque nœud.
- `pingall` : Vérifie la connectivité complète.
- `links` : Montre les liaisons entre éléments.

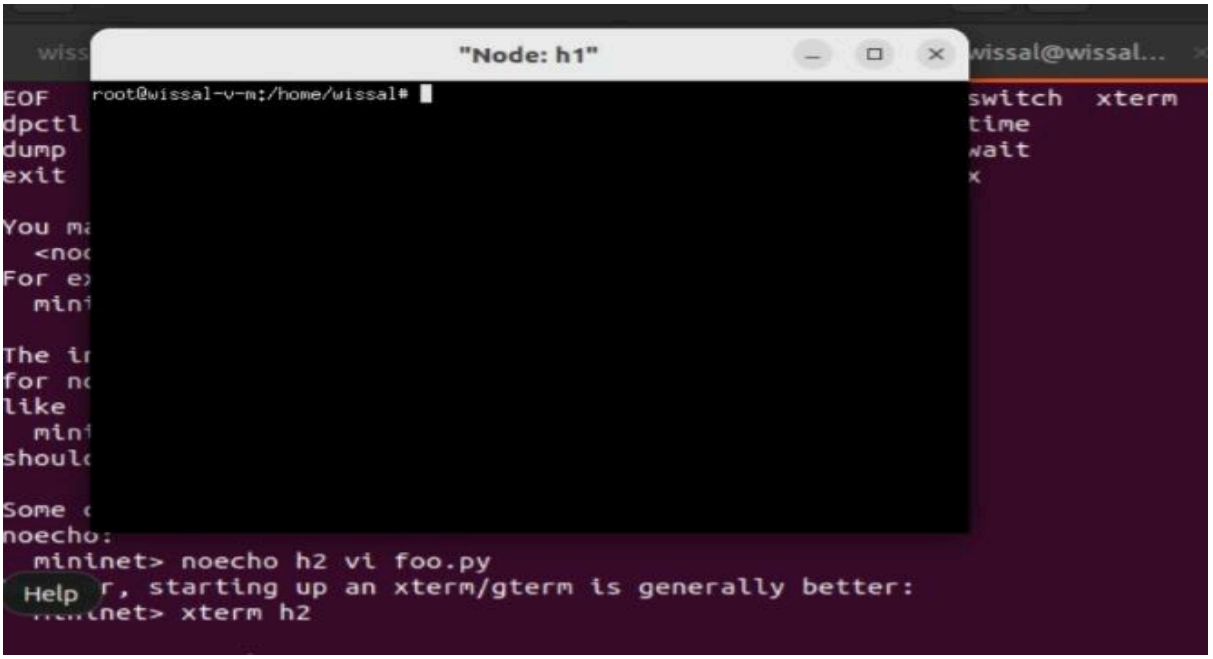
```
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=3872>
<Host h2: h2-eth0:10.0.0.2 pid=3874>
<Host h3: h3-eth0:10.0.0.3 pid=3876>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None pid=3881>
<OVSSwitch s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None pid=3884>
<Controller c0: 127.0.0.1:6653 pid=3865>
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet> links
h1-eth0<->s1-eth1 (OK OK)
h2-eth0<->s1-eth2 (OK OK)
h3-eth0<->s2-eth2 (OK OK)
s1-eth3<->s2-eth1 (OK OK)
```

```
mininet> h1 route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        0.0.0.0         255.0.0.0       U     0      0        0 h1-eth0
```

```
mininet> h1 ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: h1-eth0@if37: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
state UP group default qlen 1000
    link/ether 00:00:00:00:00:01 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.0.0.1/8 brd 10.255.255.255 scope global h1-eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::200:ff:fe00:1/64 scope link
       valid_lft forever preferred_lft forever
```

```
mininet> pingpair
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
```

```
mininet> help

Documented commands (type help <topic>):
========================================
EOF    gterm  iperfudp  nodes        pingpair      py      switch  xterm
dpctl  help   link      noecho       pingpairfull  quit    time
dump   intfs  links     pingall      ports         sh      wait
exit   iperf  net       pingallfull  px            source  x

You may also send a command to a node using:
  <node> command {args}
For example:
  mininet> h1 ifconfig

The interpreter automatically substitutes IP addresses
for node names when a node is the first arg, so commands
like
  mininet> h2 ping h3
should work.

Some character-oriented interactive commands require
noecho:
  mininet> noecho h2 vi foo.py
However, starting up an xterm/gterm is generally better:
  mininet> xterm h2
```



Conclusion

Ce TP nous a permis de :

- Créer différentes topologies virtuelles (simple, linéaire, arborescente, personnalisée).
- Explorer les commandes de diagnostic réseau (ping, ifconfig, iperf, arp).
- Comprendre comment émuler un réseau SDN localement avec Open vSwitch.
- Préparer l’environnement pour interagir avec un contrôleur SDN c'est-à- dire ONOS.