Realise par :

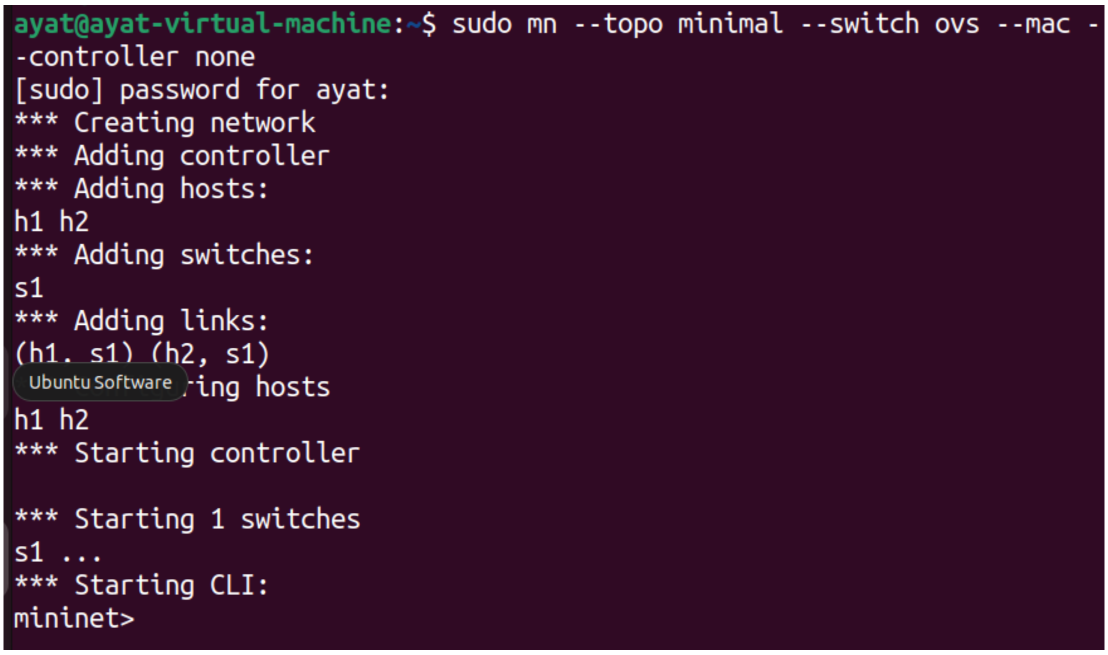
Ayat errahmane LABRIGUI

TP 2 – Configuration d’un réseau OpenFlow sans contrôleur

Partie I

Étape 1 – Lancement de la topologie

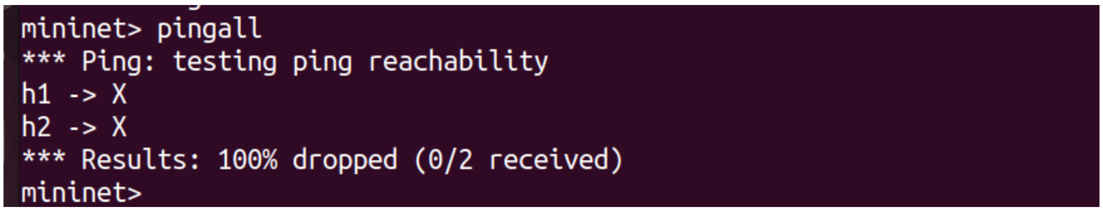
```
sudo mn --topo minimal --switch ovs --mac --controller none
```



```
ayat@ayat-virtual-machine:~$ sudo mn --topo minimal --switch ovs --mac -
-controller none
[sudo] password for ayat:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
Ubuntu Software ing hosts
h1 h2
*** Starting controller

*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

Étape 2 – Test sans flux installés

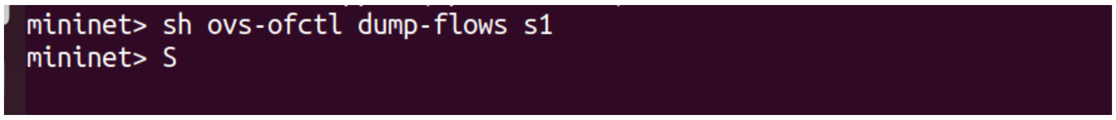
mininet> pingall



```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X
h2 -> X
*** Results: 100% dropped (0/2 received)
mininet>
```

Étape 3 – Vérifier la table de flux

```
sudo ovs-ofctl dump-flows s1
```



```
mininet> sh ovs-ofctl dump-flows s1
mininet> S
```

Table vide → aucun flux n'est encore configuré.

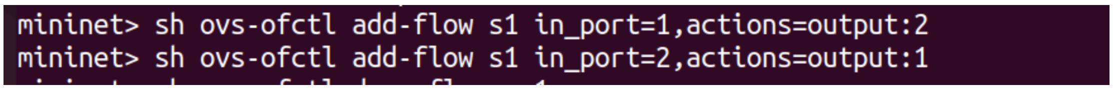
Étape 4 – Ajouter des flux manuellement

Ajoute un flux **de h1 vers h2** :

```
sudo ovs-ofctl add-flow s1 in_port=1,actions=output:2
```

Et un flux **de h2 vers h1** :

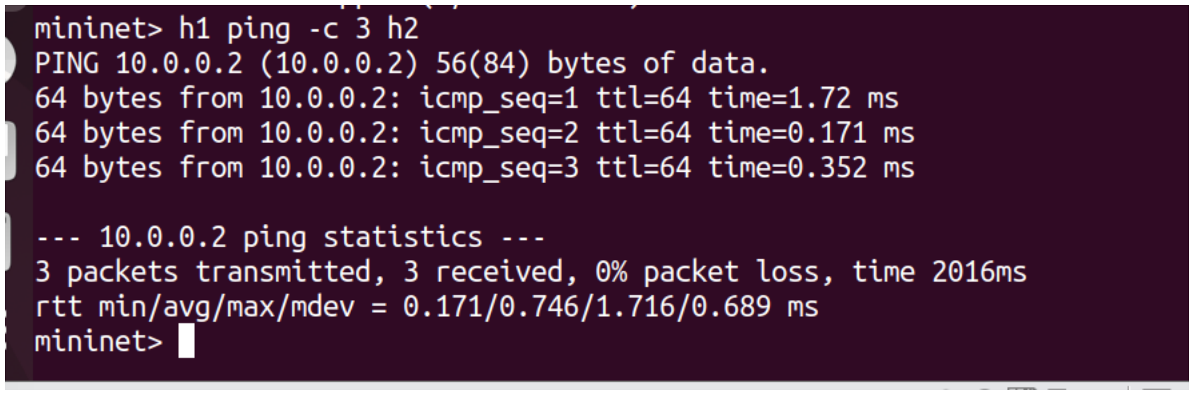
```
sudo ovs-ofctl add-flow s1 in_port=2,actions=output:1
```



```
mininet> sh ovs-ofctl add-flow s1 in_port=1,actions=output:2
mininet> sh ovs-ofctl add-flow s1 in_port=2,actions=output:1
```

Étape 5 – Tester la connectivité

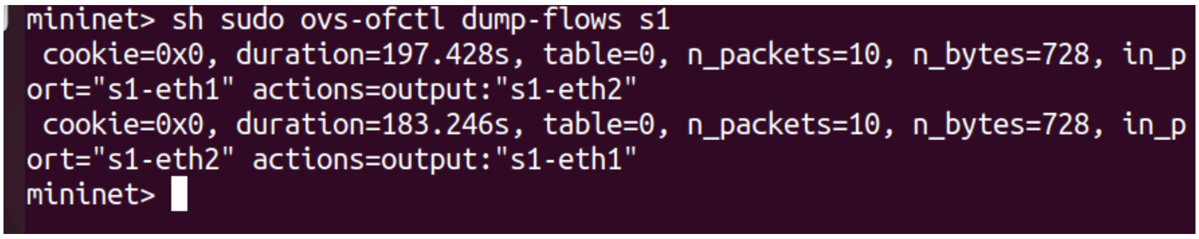
```
mininet> h1 ping -c 3 h2
```



```
mininet> h1 ping -c 3 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=1.72 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.171 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.352 ms

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2016ms
rtt min/avg/max/mdev = 0.171/0.746/1.716/0.689 ms
mininet>
```

Étape 6 – Vérifier les flux installés

```
sudo ovs-ofctl dump-flows s1
```



```
mininet> sh sudo ovs-ofctl dump-flows s1
 cookie=0x0, duration=197.428s, table=0, n_packets=10, n_bytes=728, in_p
ort="s1-eth1" actions=output:"s1-eth2"
 cookie=0x0, duration=183.246s, table=0, n_packets=10, n_bytes=728, in_p
ort="s1-eth2" actions=output:"s1-eth1"
mininet> 
```

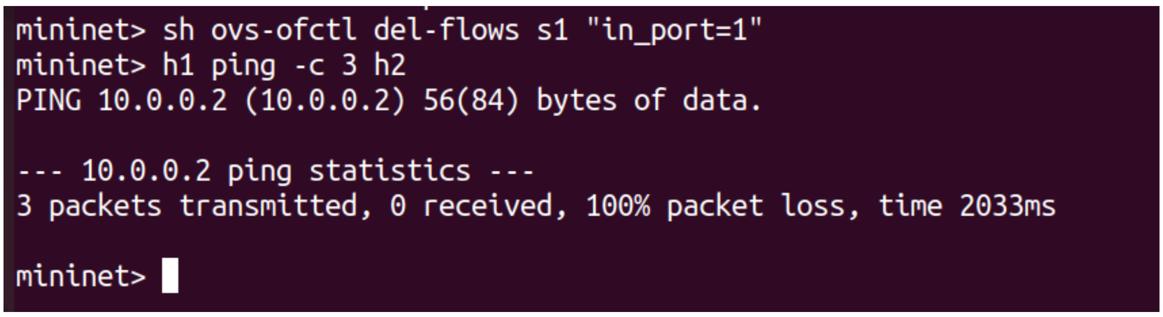
Étape 7 – Supprimer un flux et observer le résultat

Supprime la règle de h1 → h2 :

```
sudo ovs-ofctl del-flows s1 "in_port=1"
```

Teste à nouveau :

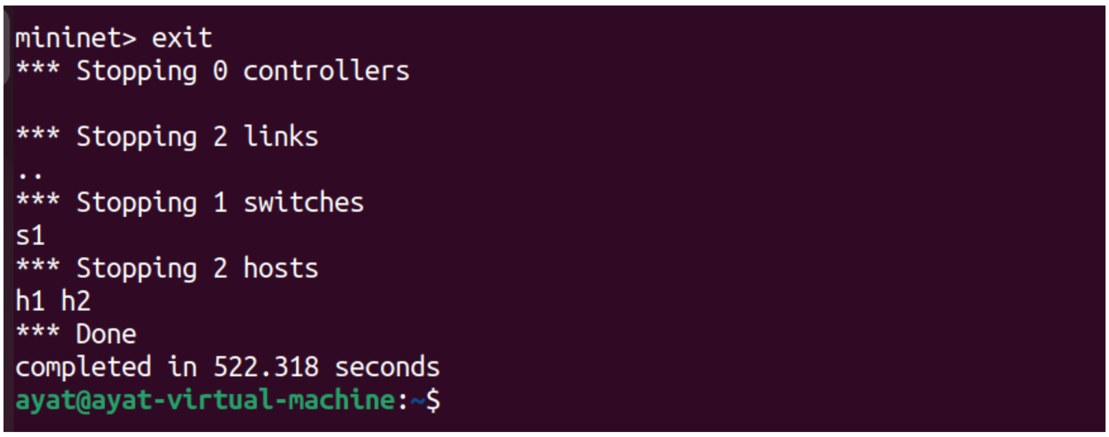
```
mininet> h1 ping -c 3 h2
```



```
mininet> sh ovs-ofctl del-flows s1 "in_port=1"
mininet> h1 ping -c 3 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2033ms

mininet> 
```

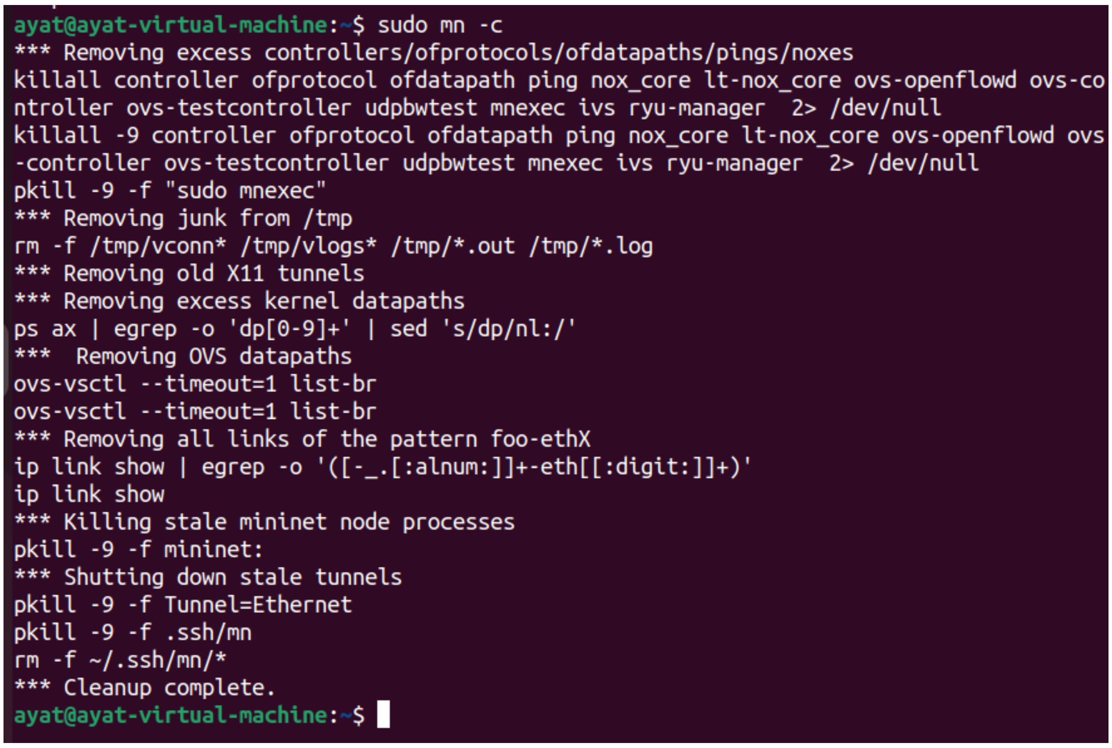
Étape 8 – Nettoyage

mininet> exit



```
mininet> exit
*** Stopping 0 controllers

*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 522.318 seconds
ayat@ayat-virtual-machine:~$
```

sudo mn –c



```
ayat@ayat-virtual-machine:~$ sudo mn -c
*** Removing excess controllers/ofprotocols/ofdatapaths/pings/noxes
killall controller ofprotocol ofdatapath ping nox_core lt-nox_core ovs-openflowd ovs-co
ntroller ovs-testcontroller udpbwtest mnexec ivs ryu-manager  2> /dev/null
killall -9 controller ofprotocol ofdatapath ping nox_core lt-nox_core ovs-openflowd ovs
-controller ovs-testcontroller udpbwtest mnexec ivs ryu-manager  2> /dev/null
pkill -9 -f "sudo mnexec"
*** Removing junk from /tmp
rm -f /tmp/vconn* /tmp/vlogs* /tmp/*.out /tmp/*.log
*** Removing old X11 tunnels
*** Removing excess kernel datapaths
ps ax | egrep -o 'dp[0-9]+' | sed 's/dp/nl:/'
***  Removing OVS datapaths
ovs-vsctl --timeout=1 list-br
ovs-vsctl --timeout=1 list-br
*** Removing all links of the pattern foo-ethX
ip link show | egrep -o '([-_.[:alnum:]]+-eth[[:digit:]]+)'
ip link show
*** Killing stale mininet node processes
pkill -9 -f mininet:
*** Shutting down stale tunnels
pkill -9 -f Tunnel=Ethernet
pkill -9 -f .ssh/mn
rm -f ~/.ssh/mn/*
*** Cleanup complete.
ayat@ayat-virtual-machine:~$
```

Questions de réflexion

Question

Pourquoi le ping ne passe-t-il pas au départ ?

➢ Parce qu’aucune règle de flux n’est installée, le switch ne sait pas où envoyer les paquets.

Que fait la commande add-flow ?

- Elle ajoute une règle de flux indiquant comment rediriger les paquets entre les ports.

Pourquoi faut-il ajouter deux flux ?

- Un pour le sens h1→h2 et un autre pour le sens h2→h1.

Que se passe-t-il si un hôte change de port ?

- Les règles deviennent invalides, il faut les recréer avec les nouveaux ports.

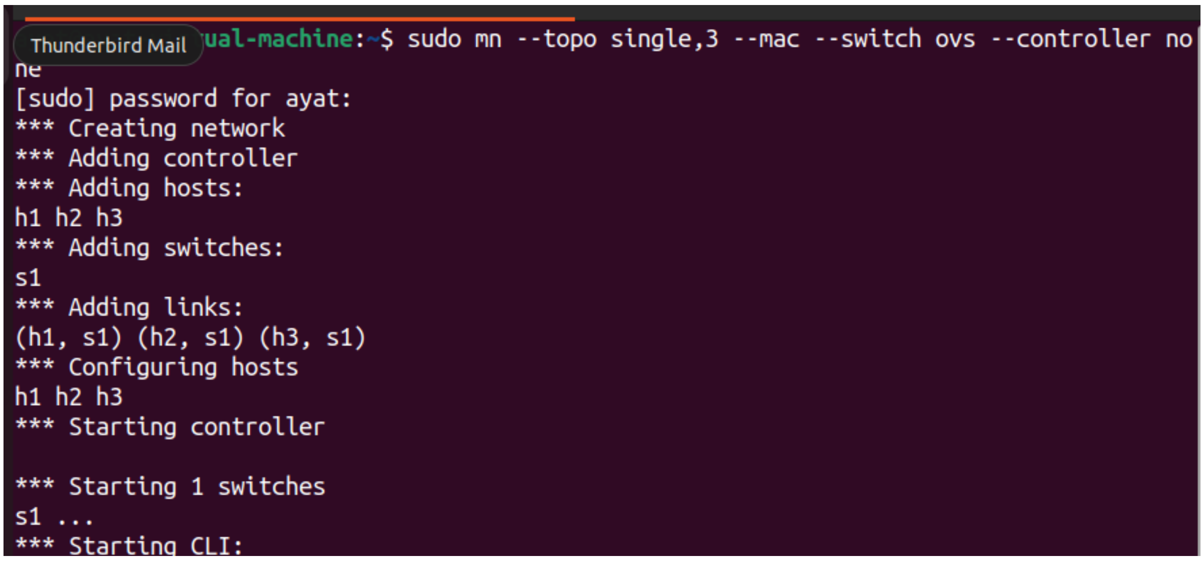
Quel est l’intérêt d’un contrôleur SDN ?

- Il gère automatiquement les flux et adapte le réseau sans configuration manuelle.

Partie II– Gestion de la QoS et Contrôle de la Bande Passante avec OpenFlow

Étape 1 – Lancer Mininet

```
sudo mn --topo single,3 --mac --switch ovs --controller none
```

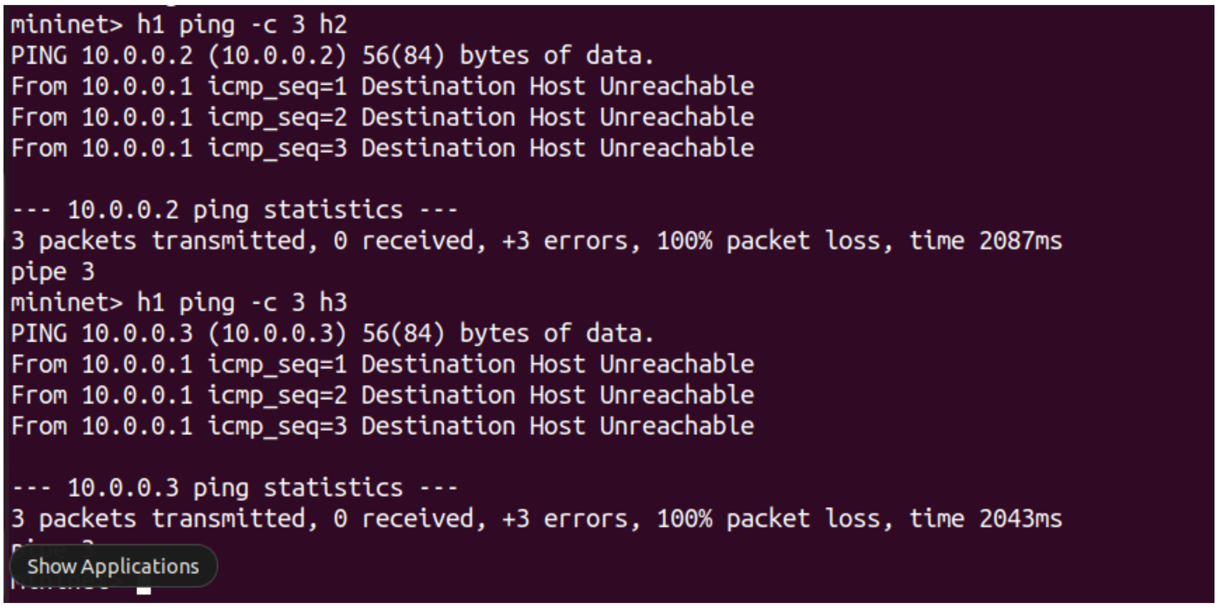


```
ual-machine:~$ sudo mn --topo single,3 --mac --switch ovs --controller no
ne
[sudo] password for ayat:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller

*** Starting 1 switches
s1 ...
*** Starting CLI:
```

Étape 2 – Vérifier la connectivité de base

Testez la communication :

```
mininet> h1 ping -c 3 h2

mininet> h1 ping -c 3 h3
```

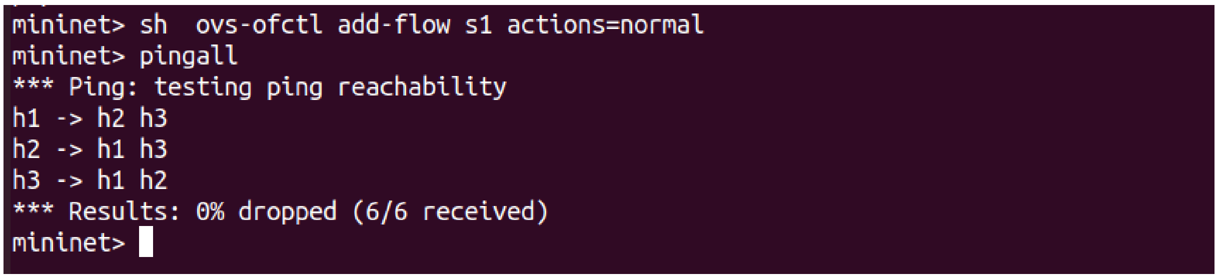


```
mininet> h1 ping -c 3 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=1 Destination Host Unreachable
From 10.0.0.1 icmp_seq=2 Destination Host Unreachable
From 10.0.0.1 icmp_seq=3 Destination Host Unreachable

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2087ms
pipe 3
mininet> h1 ping -c 3 h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=1 Destination Host Unreachable
From 10.0.0.1 icmp_seq=2 Destination Host Unreachable
From 10.0.0.1 icmp_seq=3 Destination Host Unreachable

--- 10.0.0.3 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2043ms
```

```
sudo ovs-ofctl add-flow s1 actions=normal
```

Re-test :

mininet> pingall

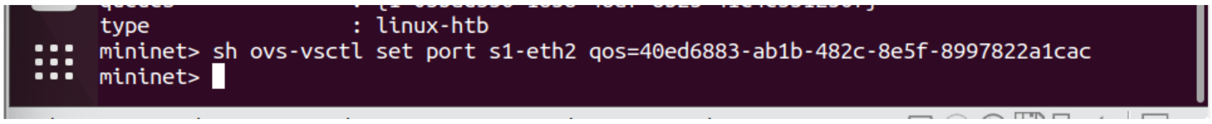


```
mininet> sh  ovs-ofctl add-flow s1 actions=normal
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet>
```

Étape 3 – Création d'une file d'attente QoS

Objectif : limiter le débit de **s1-eth2** (port relié à h2) à **1 Mbit/s**.

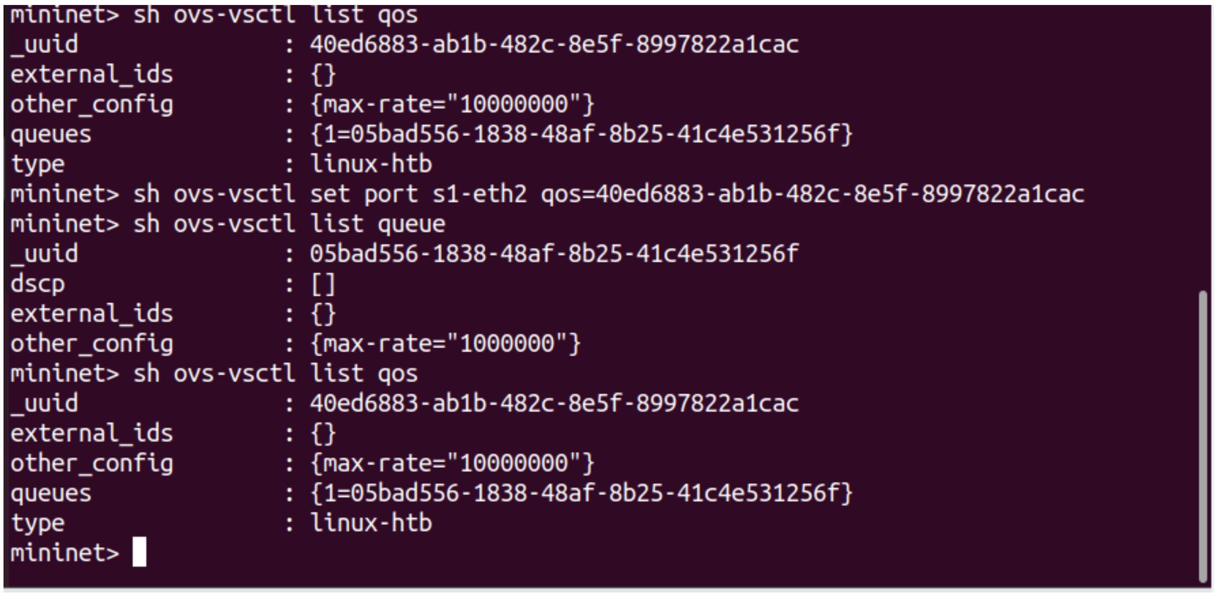
```
sudo ovs-vsctl -- set port s1-eth2 qos=@newqos -- \

--id=@newqos create qos type=linux-htb other-config:max-rate=10000000 queues:1=@q1 -- \
--id=@q1 create queue other-config:max-rate=1000000
```



```
type                  : linux-htb
mininet> sh ovs-vsctl set port s1-eth2 qos=40ed6883-ab1b-482c-8e5f-8997822a1cac
mininet>
```

Étape 4 – Vérification de la configuration

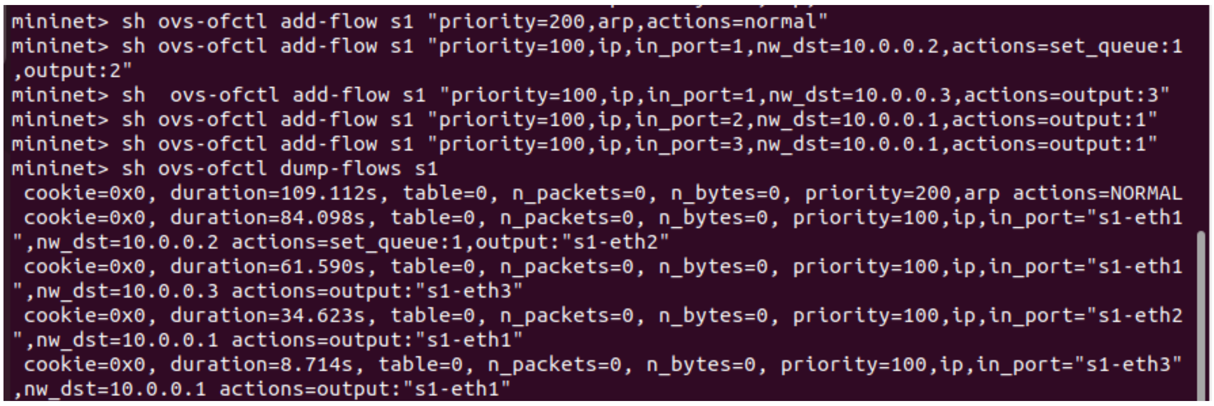
```
sudo ovs-vsctl list qos

sudo ovs-vsctl list port s1-eth2

sudo ovs-vsctl list queue
```



```
mininet> sh ovs-vsctl list qos
_uuid               : 40ed6883-ab1b-482c-8e5f-8997822a1cac
external_ids        : {}
other_config        : {max-rate="10000000"}
queues              : {1=05bad556-1838-48af-8b25-41c4e531256f}
type                : linux-htb
mininet> sh ovs-vsctl set port s1-eth2 qos=40ed6883-ab1b-482c-8e5f-8997822a1cac
mininet> sh ovs-vsctl list queue
_uuid               : 05bad556-1838-48af-8b25-41c4e531256f
dscp                : []
external_ids        : {}
other_config        : {max-rate="1000000"}
mininet> sh ovs-vsctl list qos
_uuid               : 40ed6883-ab1b-482c-8e5f-8997822a1cac
external_ids        : {}
other_config        : {max-rate="10000000"}
queues              : {1=05bad556-1838-48af-8b25-41c4e531256f}
type                : linux-htb
mininet>
```

Étape 5 – Supprimer la règle “normal” (elle court-circuite la QoS)

```
sudo ovs-ofctl del-flows s1
```



```
mininet> sh ovs-ofctl add-flow s1 "priority=200,arp,actions=normal"
mininet> sh ovs-ofctl add-flow s1 "priority=100,ip,in_port=1,nw_dst=10.0.0.2,actions=set_queue:1
,output:2"
mininet> sh  ovs-ofctl add-flow s1 "priority=100,ip,in_port=1,nw_dst=10.0.0.3,actions=output:3"
mininet> sh ovs-ofctl add-flow s1 "priority=100,ip,in_port=2,nw_dst=10.0.0.1,actions=output:1"
mininet> sh ovs-ofctl add-flow s1 "priority=100,ip,in_port=3,nw_dst=10.0.0.1,actions=output:1"
mininet> sh ovs-ofctl dump-flows s1
 cookie=0x0, duration=109.112s, table=0, n_packets=0, n_bytes=0, priority=200,arp actions=NORMAL
 cookie=0x0, duration=84.098s, table=0, n_packets=0, n_bytes=0, priority=100,ip,in_port="s1-eth1
",nw_dst=10.0.0.2 actions=set_queue:1,output:"s1-eth2"
 cookie=0x0, duration=61.590s, table=0, n_packets=0, n_bytes=0, priority=100,ip,in_port="s1-eth1
",nw_dst=10.0.0.3 actions=output:"s1-eth3"
 cookie=0x0, duration=34.623s, table=0, n_packets=0, n_bytes=0, priority=100,ip,in_port="s1-eth2
",nw_dst=10.0.0.1 actions=output:"s1-eth1"
 cookie=0x0, duration=8.714s, table=0, n_packets=0, n_bytes=0, priority=100,ip,in_port="s1-eth3"
,nw_dst=10.0.0.1 actions=output:"s1-eth1"
```

Vérifier la table de flux :

```
sudo ovs-ofctl dump-flows s1
```

Etape 6 — Ajouter des flux OpenFlow avec QoS

Objectif : **limiter h1 → h2 à ~1 Mbit/s** (queue 1) et laisser **h1 → h3 non limité**.

Pré-requis : la QoS **est déjà attachée** au port s1-eth2 (voir Étape 3) et il faut vérifier le **mapping des ports** :

```
sudo ovs-ofctl show s1
```

# Attendu : 1(s1-eth1) 2(s1-eth2) 3(s1-eth3)

6.0 Réinitialiser proprement la table de flux

```
sudo ovs-ofctl del-flows s1
```

6.1 Laisser passer l'ARP (sinon pas de résolution IP)

```
sudo ovs-ofctl add-flow s1 "priority=200,arp,actions=normal"
```

6.2 Trafic IP h1 → h2 limité à 1 Mbit/s

```
sudo ovs-ofctl add-flow s1
"priority=100,ip,in_port=1,nw_dst=10.0.0.2,actions=set_queue:1,output:2"
```

6.3 Trafic IP h1 → h3 non limité

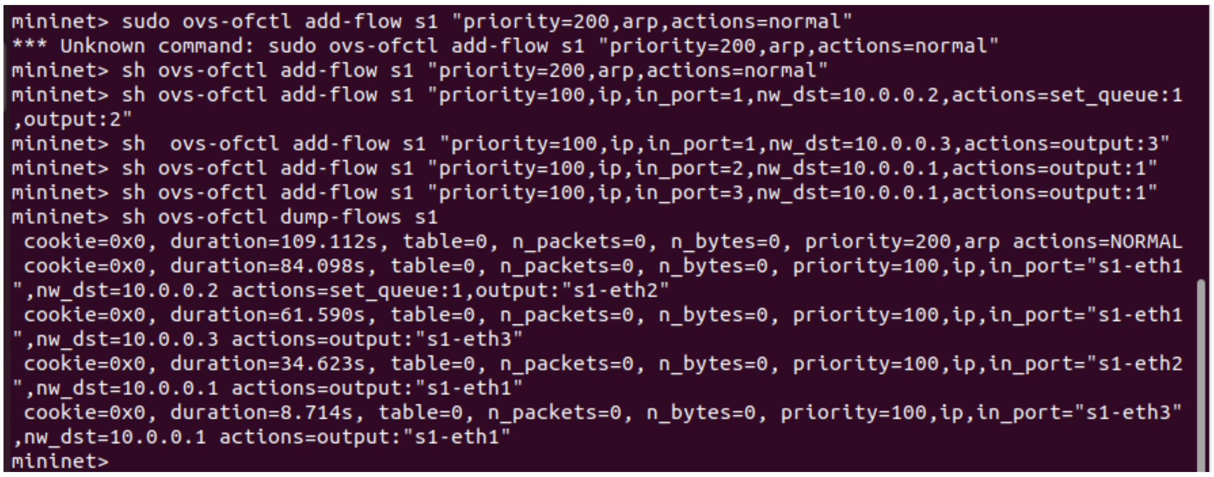
```
sudo ovs-ofctl add-flow s1 "priority=100,ip,in_port=1,nw_dst=10.0.0.3,actions=output:3"
```

6.4 Flux retour (nécessaires pour TCP/ICMP)

```
sudo ovs-ofctl add-flow s1 "priority=100,ip,in_port=2,nw_dst=10.0.0.1,actions=output:1"

sudo ovs-ofctl add-flow s1 "priority=100,ip,in_port=3,nw_dst=10.0.0.1,actions=output:1"
```

6.5 Vérification

```
sudo ovs-ofctl dump-flows s1
```

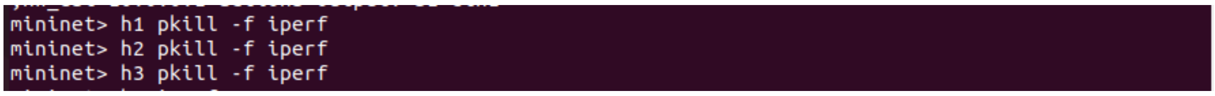


```
mininet> sudo ovs-ofctl add-flow s1 "priority=200,arp,actions=normal"
*** Unknown command: sudo ovs-ofctl add-flow s1 "priority=200,arp,actions=normal"
mininet> sh ovs-ofctl add-flow s1 "priority=200,arp,actions=normal"
mininet> sh ovs-ofctl add-flow s1 "priority=100,ip,in_port=1,nw_dst=10.0.0.2,actions=set_queue:1
,output:2"
mininet> sh  ovs-ofctl add-flow s1 "priority=100,ip,in_port=1,nw_dst=10.0.0.3,actions=output:3"
mininet> sh ovs-ofctl add-flow s1 "priority=100,ip,in_port=2,nw_dst=10.0.0.1,actions=output:1"
mininet> sh ovs-ofctl add-flow s1 "priority=100,ip,in_port=3,nw_dst=10.0.0.1,actions=output:1"
mininet> sh ovs-ofctl dump-flows s1
 cookie=0x0, duration=109.112s, table=0, n_packets=0, n_bytes=0, priority=200,arp actions=NORMAL
 cookie=0x0, duration=84.098s, table=0, n_packets=0, n_bytes=0, priority=100,ip,in_port="s1-eth1
",nw_dst=10.0.0.2 actions=set_queue:1,output:"s1-eth2"
 cookie=0x0, duration=61.590s, table=0, n_packets=0, n_bytes=0, priority=100,ip,in_port="s1-eth1
",nw_dst=10.0.0.3 actions=output:"s1-eth3"
 cookie=0x0, duration=34.623s, table=0, n_packets=0, n_bytes=0, priority=100,ip,in_port="s1-eth2
",nw_dst=10.0.0.1 actions=output:"s1-eth1"
 cookie=0x0, duration=8.714s, table=0, n_packets=0, n_bytes=0, priority=100,ip,in_port="s1-eth3"
,nw_dst=10.0.0.1 actions=output:"s1-eth1"
mininet>
```

Étape 7 — Tests et validation

7.1 Nettoyer d'anciens iperf (si besoin)

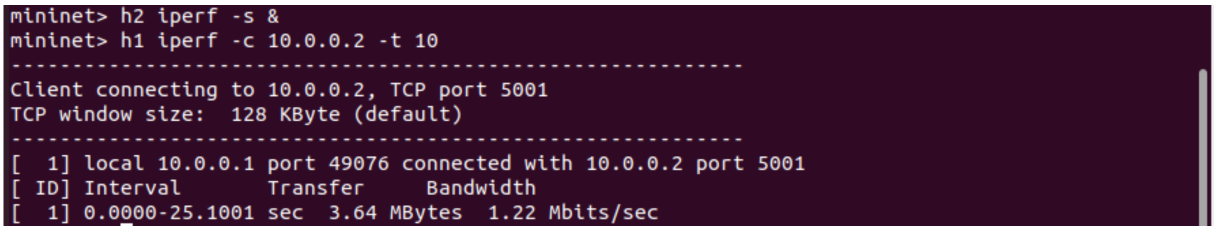
```
mininet> h1 pkill -f iperf

mininet> h2 pkill -f iperf

mininet> h3 pkill -f iperf
```



```
mininet> h1 pkill -f iperf
mininet> h2 pkill -f iperf
mininet> h3 pkill -f iperf
```

7.2 Flux limité (h1 → h2 ≈ 1 Mbit/s)

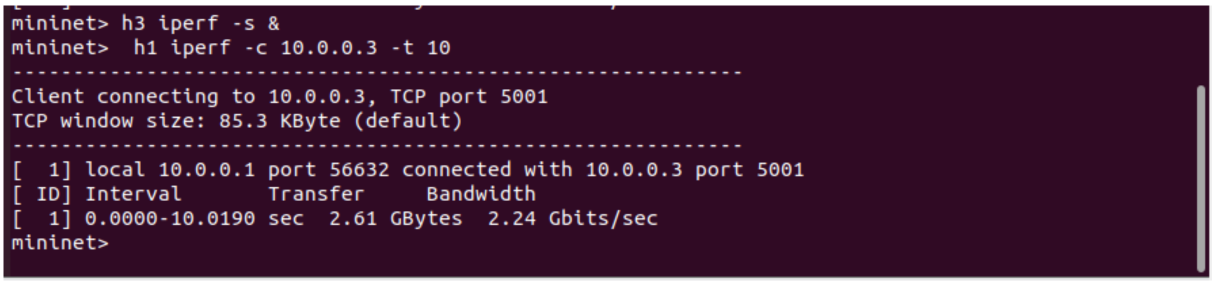
```
mininet> h2 iperf -s &

mininet> h1 iperf -c 10.0.0.2 -t 10
```



```
mininet> h2 iperf -s &
mininet> h1 iperf -c 10.0.0.2 -t 10
------------------------------------------------------------
Client connecting to 10.0.0.2, TCP port 5001
TCP window size:  128 KByte (default)
------------------------------------------------------------
[  1] local 10.0.0.1 port 49076 connected with 10.0.0.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-25.1001 sec  3.64 MBytes  1.22 Mbits/sec
```

7.3 Flux non limité (h1 → h3 ≈ 9–10 Mbit/s)

```
mininet> h3 iperf -s &

mininet> h1 iperf -c 10.0.0.3 -t 10
```


```
mininet> h3 iperf -s &
mininet>  h1 iperf -c 10.0.0.3 -t 10
------------------------------------------------------------
Client connecting to 10.0.0.3, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  1] local 10.0.0.1 port 56632 connected with 10.0.0.3 port 5001
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-10.0190 sec  2.61 GBytes  2.24 Gbits/sec
mininet>
```

Questions pédagogiques

Question	**Réponse**
Que permet la QoS sur un switch ?	
Pourquoi supprimer actions=normal ?	
Quelle commande applique la limitation de débit ?	
Que représente linux-htb ?	
Peut-on appliquer plusieurs queues par port ?	
