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

TP2-Configuration d'un réseau OpenFlow sans contrôleur

Partie I :

Objectif pédagogique

Ce TP vise à comprendre le fonctionnement d'un switch OpenFlow sans contrôleur SDN, en créant une topologie simple avec Mininet, en ajoutant des règles de flux manuellement, et en observant l'importance d'un contrôleur SDN pour automatiser la gestion des flux.

Étape 1 – Lancement de la topologie

```
wissal@wissal-v-m:~$ sudo mn --topo minimal --switch ovs --mac --controller none
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller

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

Étape 2 – Test sans flux installés

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

Étape 3 – Vérifier la table de flux

```
v-m:~$ sudo ovs-ofctl dump-flows s1
```

La table est vide car aucun flux n'est encore configuré.

Étape 4 – Ajouter des flux manuellement

On Ajoute un flux **de h1 vers h2** par la cmd :

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

Et on ajoute un flux **de h2 vers h1** par la cmd :

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

Étape 5 – Tester la connectivité

```
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=0.741 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.310 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.161 ms

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 0.161/0.404/0.741/0.245 ms
```

Étape 6 – Vérifier les flux installés

```
wissal@wissal-v-m:~$ sudo ovs-ofctl dump-flows s1
 cookie=0x0, duration=60.589s, table=0, n_packets=6, n_bytes=448, in_port="s1-eth1" actions=
output:"s1-eth2"
 cookie=0x0, duration=46.429s, table=0, n_packets=6, n_bytes=448, in_port="s1-eth2" actions=
output:"s1-eth1"
```

Étape 7 – Supprimer un flux et observer le résultat

On supprime la régle de h1 vers h2 par la cmd :

```
-v-m:~$ sudo ovs-ofctl del-flows s1 "in_port=1"
```

Et on teste a nouveau :

```
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 2035ms
```

Étape 8 – Nettoyage

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

*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 222.067 seconds
wissal@wissal-v-m:~$ sudo mn -c
*** Removing excess controllers/ofprotocols/ofdatapaths/pings/noxes
killall controller ofprotocol ofdatapath ping nox_core lt-nox_core ovs-openflowd ovs-control
ler ovs-testcontroller udpbwtest mnexec ivs ryu-manager  2> /dev/null
killall -9 controller ofprotocol ofdatapath ping nox_core lt-nox_core ovs-openflowd ovs-cont
roller 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.
```

Reponse :

1 : Pourquoi le ping ne passe-t-il pas au départ ?
Parce qu'aucune règle de flux n'est encore installée dans le switch. Sans ces règles, le switch ne sait pas où envoyer les paquets, donc la communication entre les hôtes échoue.

2 : Que fait la commande add-flow ?
Elle permet d'ajouter manuellement une règle de flux dans le switch. Cette règle indique comment traiter les paquets reçus (par exemple : envoyer les paquets du port 1 vers le port 2).

3 : Pourquoi faut-il ajouter deux flux ?
Parce que la communication doit être bidirectionnelle.
Un flux permet d'envoyer les paquets de **h1 vers h2**, et l'autre de **h2 vers h1**.

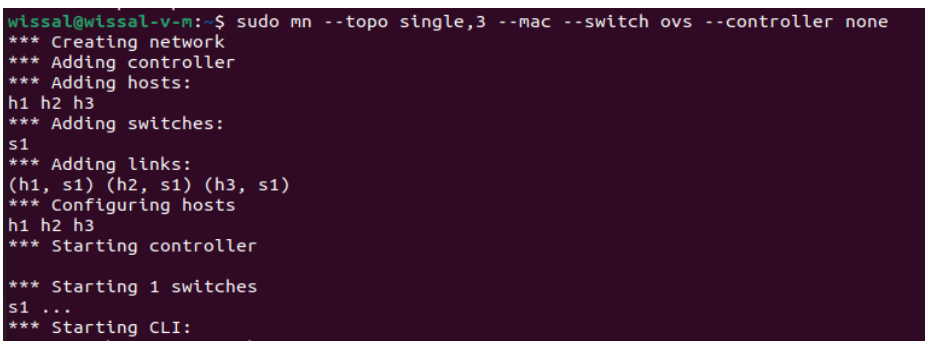
4 : Que se passe-t-il si un hôte change de port ?
Les règles de flux deviennent invalides, car elles dépendent des numéros de ports. Il faut donc ajouter de nouvelles règles adaptées à la nouvelle configuration.

5 : Quel est l'intérêt d'un contrôleur SDN ?
Le contrôleur SDN automatise la gestion des flux.
Il installe, modifie et supprime les règles dynamiquement sans intervention manuelle, ce qui rend le réseau plus intelligent et plus flexible.

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

Objectif pédagogique :
Comprendre comment gérer la bande passante et prioriser les flux réseau avec **Open vSwitch (OVS)** et **OpenFlow**, en observant leurs effets sur la latence et le débit.

Étape 1 – Lancer Mininet



Étape 2 – Vérifier la connectivité de base

```
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 2076ms
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 2082ms
pipe 3
```

On ajoute une règle simple de forwarding pour autoriser le trafic :

```
-v-m:~$ sudo ovs-ofctl add-flow s1 actions=normal
```

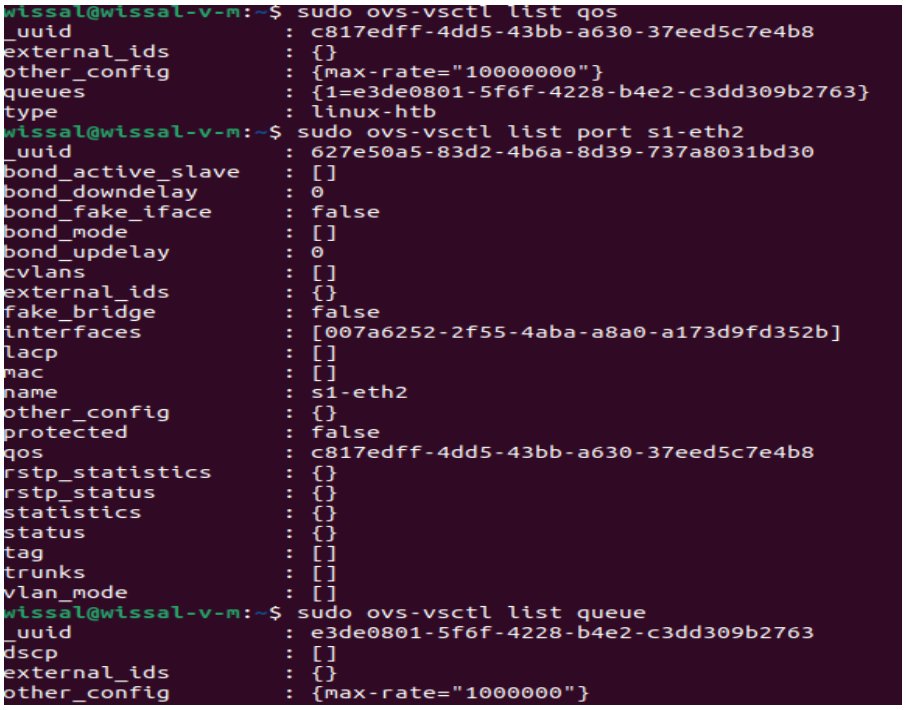
On Re-test :

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
```

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

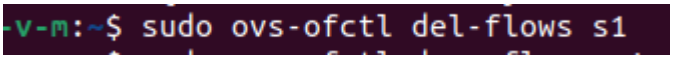
```
wissal@wissal-v-m:~$ sudo ovs-vsctl -- set port s1-eth2 qos=@newqos -- \
--id=@newqos create qos type=linux-htb other-config:max-rate=10000000 queues:1=@q1 -- \
--id=@q1 create queue other-config:max-rate=1000000
c817edff-4dd5-43bb-a630-37eed5c7e4b8
e3de0801-5f6f-4228-b4e2-c3dd309b2763
```

Étape 4 – Vérification de la configuration

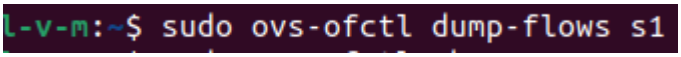


Étape 5 – Supprimer la règle “normal” (elle court-circuite la QoS)

On supprime par la cmd :



Et on vérifier la table de flus par la cmd **:**



Etape 6 — Ajouter des flux OpenFlow avec QoS

Vérifions le mapping des ports par la cmd :

```
wissal@wissal-v-m:~$ sudo ovs-ofctl show s1
OFPT_FEATURES_REPLY (xid=0x2): dpid:0000000000000001
n_tables:254, n_buffers:0
capabilities: FLOW_STATS TABLE_STATS PORT_STATS QUEUE_STATS ARP_MATCH_IP
actions: output enqueue set_vlan_vid set_vlan_pcp strip_vlan mod_dl_src mod_dl_dst mod_nw_sr
c mod_nw_dst mod_nw_tos mod_tp_src mod_tp_dst
 1(s1-eth1): addr:3e:f6:1f:dd:c6:bf
     config:     0
     state:      0
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 2(s1-eth2): addr:26:af:73:42:72:2a
     config:     0
     state:      0
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 3(s1-eth3): addr:2a:2e:61:a7:5f:76
     config:     0
     state:      0
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 LOCAL(s1): addr:3e:39:8e:a4:fa:4f
     config:     PORT_DOWN
     state:      LINK_DOWN
     speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (xid=0x4): frags=normal miss_send_len=0
```

6.0-6.1-6.2-6.3-6.4 :

```
wissal@wissal-v-m:~$ sudo ovs-ofctl del-flows s1
wissal@wissal-v-m:~$ sudo ovs-ofctl add-flow s1 "priority=200,arp,actions=normal"
wissal@wissal-v-m:~$ sudo ovs-ofctl add-flow s1 "priority=100,ip,in_port=1,nw_dst=10.0.0.2,a
ctions=set_queue:1,output:2"
wissal@wissal-v-m:~$ sudo ovs-ofctl add-flow s1 "priority=100,ip,in_port=1,nw_dst=10.0.0.3,a
ctions=output:3"
wissal@wissal-v-m:~$ sudo ovs-ofctl add-flow s1 "priority=100,ip,in_port=2,nw_dst=10.0.0.1,a
ctions=output:1"
wissal@wissal-v-m:~$ sudo ovs-ofctl add-flow s1 "priority=100,ip,in_port=3,nw_dst=10.0.0.1,a
ctions=output:1"
```

6.5 :Vérification :

```
wissal@wissal-v-m:~$ sudo ovs-ofctl dump-flows s1
 cookie=0x0, duration=85.706s, table=0, n_packets=0, n_bytes=0, priority=200,arp actions=NOR
MAL
 cookie=0x0, duration=69.982s, 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=50.360s, 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=36.393s, 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=23.646s, 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"
```

Étape 7 — Tests et validation

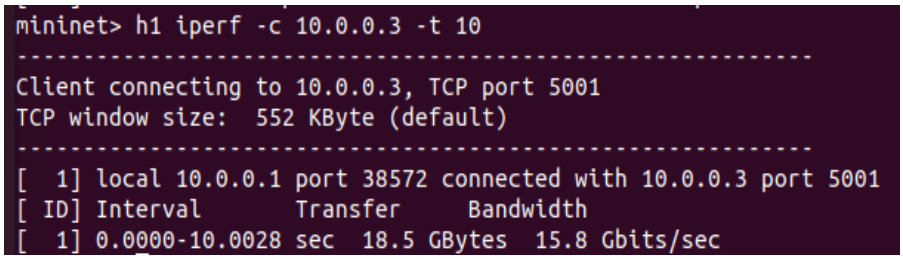
On Nettoye anciens iperf :

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

Flux limité (h1 → h2 ≈ 1 Mbit/s):

```
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 33318 connected with 10.0.0.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-24.7402 sec  3.58 MBytes  1.21 Mbits/sec
```

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



Reponse :

1. Que permet la QoS sur un switch ?
La QoS permet de contrôler la bande passante et de prioriser certains flux réseau pour garantir une meilleure qualité de service.

2. Pourquoi supprimer actions=normal ?
Parce qu'elle fait passer tous les paquets sans appliquer les règles de QoS, annulant ainsi les limitations et priorités définies.

3. Quelle commande applique la limitation de débit ?

La commande ovs-vsctl -- set port ... qos=... avec la configuration des queues limite le débit sur un port.

4. Que représente linux-htb ?
C'est un planificateur de trafic HTB (Hierarchical Token Bucket) qui permet de gérer les limitations de bande passante sous Linux.

5. Peut-on appliquer plusieurs queues par port ?
Oui, un port peut avoir plusieurs queues pour gérer différents flux avec des priorités ou débits différents.