TP1 – Mininet et le contrôleur RYU Lab

1. Introduction :

Le but de ce TP est de vous familiariser avec la plateforme mininet et de découvrir le protocole OpenFlow

2. Mise en place de topologie sous mininet

Dans cette partie, vous allez manipuler les commandes « openflow » pour assurer les communications entre machines au sein d'un réseau composé de « openflow switch ».

Un commutateur OpenFlow est un dispositif réseau qui utilise le protocole OpenFlow pour communiquer avec un contrôleur OpenFlow. Le contrôleur fournit au commutateur un ensemble de règles pour le transfert des paquets, et le commutateur suit alors ces règles pour prendre des décisions de transfert.

Les commutateurs OpenFlow permettent aux administrateurs de réseau de centraliser la gestion du réseau et de contrôler les flux de trafic de manière plus dynamique.

Dans un premier temps, vous lancez mininet avec le contrôleur RYU avec une adresse IP réseau 10.1.1.0/24. La version du protocole Openflow est la version 13, qui est la version la plus utilisée dans la gestion des architectures SDN

Lancez la commande suivante :

Dans un terminal, lancez le contrôleur RYU dans un environnement virtuel (à cause de la version du python 3.10)

```
source ryu-python3.9-venv/bin/activate
```

#Accédez au dossier ryu et lancez la commande suivante pour interagir avec mininet

#vérifiez la version de ryu

```
cd ryu
ryu-manager --version
```

#lancez ryu avec l'analyse des demandes openflow version 13

```
ryu-manager ryu.app.simple_switch_13
```

Maintenant, dans un second terminal lancez mininet, en précisant les paramètres réseau, la version du protocole OpenFlow et le nombre de machines connectées au switch :

```
sudo mn --controller=remote,ip=127.0.0.1 --mac -i 10.1.1.0/24 --switch=ovsk,
protocols=OpenFlow13 --topo=single,4
```

Dans le terminal mininet, affichez les informations sur votre architecture et notez le nom du bridge utilisé :

```
dump
```

Observez la table « Flow table ». Dans un autre terminal lancez la commande suivante :

```
ovs-ofctl -O OpenFlow13 dump-flows le_nom_du_switch
```

Vous constatez, qu'il y a que la table « Miss-table » qui contient que les informations sur le contrôleur. C'est une table qui est vide et que le switch ne peut pas forwarder de message

Pour tester la communication de tous les hôtes, depuis le terminal « mininet » lancez la commande :

```
pingall
```

Re-affichez la « Flow Table » et analysez son contenu. Pour l'instant le plus intéressant est la partie « in-port » et « actions » avec la « priority ». Ces paramètres permettent au switch d'assurer la fonction de forwarding.

Pour tester cette fonctionnalité vous quittez les deux sessions « ryu-manager » «et « mininet ».

Lancez maintenant mininet sans l'option « --controller=remote,ip=127.0.0.1 ». Mininet utilisera son propre controller interne, qui n'est pas tellement performant. Testez la communication avec tous les hôtes.

Que constatez-vous ?

Pour permettre les communications vous mettez à jour le « Flow table » pour assurer la communication :

1. Ajouter un nouveau flux la « Flow table » du switch s1:

```
ovs-ofctl -O OpenFlow13 add-flow s1 in_port=1,actions=output:2
```

2. Affichez cette et voir la nouvelle mise à jour,

3. Complétez pour assurer la fonction Forwarding. Testez la communication entre h1 et h2 depuis le terminal « mininet »:

```
$ h1 ping h2
```

4. Mettre à jour cette table pour assurer la communication entre toutes les machines

5. Affichez la table finale du « flow table ».

6. Testez tous les pings

3. Manipulation de trafic

Vous avez constaté le rôle important de Controller dans une architecture SDN. Vous lancez le Controller « ryu-manager » et « mininet » mais cette fois-ci avec une architecture avec quelques switchs et machines.

```
sudo mn --controller=remote,ip=127.0.0.1 --mac -i 10.1.1.0/24 --switch=ovsk,
protocols=OpenFlow13 --topo=lineare,2,4
```

On souhaite rediriger le trafic udp avec 5MB du hôte « h1s1 » vers « h3s1 » :

Ajouter une règle pour rediriger le trafic de h1 vers h3

```
ovs-ofctl  -O OpenFlow13 add-flow s1
priority=100,ip,nw_src=@IP_h1s1,nw_dst=@IP_h3s1,actions=output:@port_switch_connec
te_h3s1
```

Ajouter une règle pour rediriger le trafic de h3 vers h1

```
ovs-ofctl -O OpenFlow13 add-flow s1
priority=100,ip,nw_src=@IP_h3s1,nw_dst=@IP_h1s1,actions=output:@port_switch_connec
te_h1s1
```

Ensuite, vous ouvrez les « xterm » des deux machines

xterm h1s1 h1s3

sur chaque session, vous utilisez les mêmes commandes « Linux » : ifconfig, ping, iperf, etc.

Pour tester , un flux UDP, vous lancez « iperf » en mode UDP et avec la bande passante demandée

Observez le résultat et vérifiez si la bande passante a été respectée.
Affichez la table « Flow table »

Pour que le Controller puisse assurer le forwarding, vous lancez sur le terminal « mininet » « pingall »

Sur le même terminal testez les commandes suivantes :

```
h2s1 ifconfig
h4s1 ping h2s2
h1s2 ip route
```

vous pouvez changer l'adresse IP d'une machine dans le même réseau, par exemple, on souhaite affecter l'adresse IP 10.1.1.10/24 à la machine « h1s2 » :

```
h1s2 ifconfig h1s2-eth0 10.1.1.100 netmask 255.255.255.0

h2s1 ping h1s2
```

Modifiez une machine au choix son adresse IP avec 10.1.1.200/24 et testez un ping avec cette machine

Enfin, Si vous avez une entrée de flux particulière que vous souhaitez supprimer et que vous en connaissez les détails, vous pouvez spécifier ces détails pour la suppression. Par exemple :

```
sudo ovs-ofctl del-flows s1 "priority=100,in_port=2,actions=output:4"
```

on souhaite interdire la communication entre « h3s1 » et « h4s2 »
Faîtes un test de ping

- **Gestion de trafic**

Avec mininet vous pouvez générer du trafic comme dans un réseau physique :

<u>Gestion du trafic TCP</u>
Sélectionnez deux machines sur deux switchs différents et lancez les sessions avec « xterm »
Sur le premier lancez « iperf -s »

Sur le deuxième lancez « iperf -c @IP_du_server »
Pour avoir un report sur la consommation de bande passante ajoutez de côté client « -i 5 » et testez
On souhaite simuler le trafic sur une période bien définie par exemple « 50s », toujours de côté clients ajoutez en plus l'option « -t 50 ». Observez le résultat obtenu. Que pensez-vous de la consommation de la bande passante.
Enfin, si on souhaite étudier l'impact de flux TCP sur le réseau, on génère plusieurs flux avec l'option « -P 10 » pour dix connexions simultanées
Quel est impact de TCP sur la bande passante, est-elle équivalente pour tous les flux.

Gestion dy trafic UDP

Refaites la même manipulation mais avec l'option « -u » pour simuler le trafic UDP

Quel est l'impact de UDP sur la bande passante

Gestion du trafic HTTP

Pour tester le trafic http :

Sur un « xterm » d'une machine quelconque, lancez commande suivante :

Par exemple :

h4s1 :

```
python3 -m http.server 80
```

dans un autre terminal « xterm »

exemple :

h1s2 :

```
curl http://@IP_http_server/
```

Si on veut simuler 50 utilisateurs avec 500 demandes d'établissement de connexion au serveur web simultanément

```
ab -n 500 -c 50 http://@IP_http_server:80/
```

4. Création de topologie propriétaire

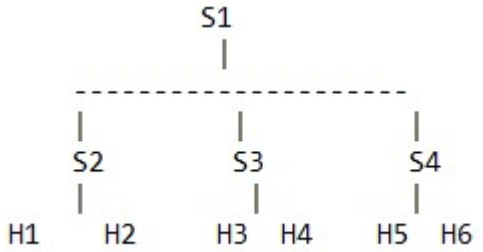
Jusqu'à présent, c'est mininet qui génère la topologie du réseau. Dans cette section, vous allez générer votre propre topologie avec votre plan d'adressage.

Sur le dossier /home/toto/Téléchargement, copiez le fichier « .py » et exécutez-le et n'oubliez pas de lancer votre « ryu-manager »

Exemple

```
python3 fichier.py
```

Créez un nouveau script pour une topologie hiérarchique à deux niveaux, l'adresse réseau est de classe B : 172.16.0.0/16



- Générez du trafic TCP entre H3 et H5
- En simultané, générez un trafic UDP entre H1 et H6 avec une bande passante de 20M

Qu'observez-vous ?