TP Réseaux Programmables

Paul Chaignon 28/01/2019

1 Introduction

L'objectif de ce TP est d'observer un ensemble de comportements de réseaux SDN. Pour cela nous allons émuler un réseau OpenFlow sur une seule machine et tester différents programmes au niveau du contrôleur.

1.1 Installation

Nous allons émuler notre réseau OpenFlow à l'aide de Mininet. Mininet émule des hôtes et switch sur une même machine en utilisant les namespaces Linux et le switch software Open vSwitch respectivement.

Listing 1: Installation de Mininet.

```
wget -O mininet.tar.gz https://github.com/mininet/mininet/archive/2.2.2.\
    tar.gz
tar xvzf mininet.tar.gz
mv mininet-2.2.2 mininet
cd mininet
./util/install.sh -knwv
sudo su
mn --test=pingall && (which /usr/local/bin/controller || echo "Woohoo!")
```

Si "Woohoo!" ne s'affiche pas à la fin de l'installation, appelez le prof.

Pour le contrôleur OpenFlow, nous utiliserons le contrôleur POX qui permet de programmer le comportement du réseau en Python. Vous n'allez pas avoir besoin de programmer en Python pour autant.

Listing 2: Installation du contrôleur POX.

```
wget -O pox.zip https://github.com/noxrepo/pox/archive/eel.zip
unzip pox.zip
cd pox-eel
./pox.py
```

Listing 3: Téléchargement des scripts pour le TP.

```
wget https://gist.githubusercontent.com/pchaigno/51\
    b1e651c6b8504e15a0b2193759fd82/raw/1\
    bafe748fbf66c4861a9c5b937bbb171c16747b6/tc.mn
wget https://gist.githubusercontent.com/pchaigno/51\
    b1e651c6b8504e15a0b2193759fd82/raw/1\
    bafe748fbf66c4861a9c5b937bbb171c16747b6/tc-and-ip.mn
```

1.2 Prise en Main

Quelques commandes utiles pour le TP :

Listing 4: Lancer et arrêter Mininet.

```
# Lancer Mininet:
# sudo mn --topo=linear,[nb switch],[nb host par switch]
sudo mn --topo=linear,4
# Quitter Mininet (2 commandes possibles)
mininet> [CTRL+D]
mininet> quit
# Nettoyer l'environnement en cas de bug:
sudo mn -c
```

Listing 5: Vérifier la config. Mininet

```
# Voir l'aide :
mininet> help
# Voir la liste des liens reseaux :
mininet> links
# Voir les IP et les interfaces des hotes et switch :
mininet> dump
# Voir la config. des interfaces de h1 :
mininet> h1 ifconfig
# Voir les regles de forwarding dans s1 :
ovs-ofctl dump-flows s1
# Pour retrouver une commande de l'historique Mininet faite CTRL+R et \
    commencez a taper !
mininet> [CTRL+R]
```

Listing 6: Pings avec Mininet

```
# Pinguer h2 a partir de h1 :
mininet> h1 ping h2
# Ping party!
mininet> pingall
```

Question : Que fait `pingall`?

Assurez-vous de toujours avoir un seul Mininet et un seul contrôleur en cours d'exécution pour la suite du TP. Lorsqu'une commande `sudo mn` est donnée, il faut donc relancer Mininet avec cette commande.

Si vous obtenez une erreur à un quelconque moment, appelez le prof !

1.3 Sans contrôleur

Nous allons d'abord simuler un réseau sans contrôleur OpenFlow et avec une latence de 10ms sur chaque lien (l'enfer !).

Listing 7: Lancer Mininet sans contrôleur

```
sudo mn --topo=linear,4 --link tc,bw=10,delay=10ms --switch=ovsbr
```

Question : Dessinez un schéma de la topologie réseau (grossièrement).

Listing 8: Visualiser les règles de forwarding

```
sudo ovs-ofctl dump-flows s1
mininet> pingall
sudo ovs-ofctl dump-flows s1
```

Question : Comment se comportent les switch ? Les règles de forwarding changent-elles ? Pouvez-vous deviner ce que fait `action=NORMAL` ?

Question : Mesurez, avec ping, le RTT (Round Trip Time) entre h1 et h2, puis entre h1 et h4. Est-ce cohérent avec votre schéma du réseau ?

2 Réseaux OpenFlow Réactifs et Proactifs

2.1 Learning Switch

Lançons maintenant un réseau avec un contrôleur qui implémente un learning switch !

Listing 9: Lancer Mininet avec un contrôleur "learning switch"

```
sudo tc qdisc add dev lo root netem delay 10ms
./pox.py forwarding.l2_learning
sudo mn --topo=linear,4 --controller=remote --link tc,bw=10,delay=10ms --\
   pre=tc.mn
```

Observez les règles sur `s1` avant et après un ping entre h1 et h2.

Question : Comment les règles évoluent-elles après le ping ?

Les switch communiquent avec le contrôleur via l'interface `lo`, là aussi avec une latence de 10ms sur le lien. Observez les communications entre les switch et le contrôleur à l'aide de Wireshark :

Listing 10: Observer les communications switch-contrôleur avec Wireshark.

```
sudo wireshark -i lo
# Entrez le filtre suivant apres avoir demarre la capture :
wireshark> openflow_v1
```

Question : Quels types de messages le contrôleur et les switch s'échangent-ils ? Que contiennent les messages `PACKET_IN` et `PACKET_OUT` ? À quoi servent-ils ?

Mesurez le RTT entre h1 et h2, puis entre h1 et h4. Observez les variations sur une période de plus de 1 minute.

Question : Qu'observez-vous ? Est-ce cohérent avec les règles de forwarding installée dans les switch ?

Vous pouvez trouver le code s'exécutant au contrôleur dans le fichier `l2_learning.py` dans `pox-eel/pox/forwarding/`. Modifiez le code afin d'éviter les variations de RTT. Relancez le contrôleur pour vérifier.

Be smart! Il n'est pas attendu que vous lisiez tout le code du contrôleur.

Question : Combien de règles de forwarding observez-vous au niveau des différents switch si vous faîte `pingall` ? Pouvez-vous en déduire une faiblesse de cette approche de la programmation du réseau ? Quel est l'intérêt du champs `idle_timeout=10` des règles de forwarding ?

Lancez un ping entre deux hôtes. Terminez le processus du contrôleur.

Question : Que se passe-t-il ?

2.2 Switch L2

Nous allons maintenant lancer un contrôleur qui implémente une variation du comportement "learning switch" précédant.

Listing 11: Lancer Mininet avec un contrôleur

```
./pox.py forwarding.l2_pairs
sudo mn --topo=linear,4 --controller=remote --link tc,bw=10,delay=10ms --\
   pre=tc.mn
```

Question : Comment les règles dans les switch sont-elles changées ?

Mesurez le RTT entre h1 et h2.

Question : Y a-t-il un saut périodique ? Que se passe-t-il si vous pinguez h2 depuis un autre hôte ?

Lancez un ping entre deux hôtes. Terminez le processus du contrôleur.

Question : Que se passe-t-il ? Quelle différence avec le comportement précédant du contrôleur ?

2.3 Contrôleur Proactif

Listing 12: Lancer Mininet avec un contrôleur proactif

```
./pox.py openflow.discovery forwarding.topo_proactive
sudo mn --topo=linear,4 --controller=remote --link tc,bw=10,delay=10ms --\
    pre=tc-and-ip.mn
```

Attendez que les 6 liens entre les switch soient découverts.

Question : Comment la configuration IP a-t-elle changée par rapport au réseau précédant ?

Observez les règles de forwarding du switch s1. Faîte un ping de h1 à h2.

Question : Comment les règles de forwarding ont-elles changé dans s1 ? Peut-on dire qu'il s'agit réellement d'un contrôleur proactif ?

3 Open vSwitch

Vous allez maintenant configurer le réseau vous-mêmes. Voyons ce que ces switch ont dans le ventre !

Supprimez d'abord les règles de forwarding et vérifiez que le réseau ne fonctionne plus.

Listing 13: Remettre le réseau à plat.

```
sudo mn --topo=linear,4 --link tc,bw=10,delay=10ms --pre=tc.mn --switch=\
    ovsbr
mininet> pingall
for s in s1 s2 s3 s4; do sudo ovs-ofctl del-flows $s; done
sudo ovs-ofctl dump-flows s1
mininet> h1 ping h2
```

Quelques exemples de commandes utiles pour configurer Open vSwitch :

Listing 14: Ajouter des règles Open vSwitch

```
# Voir tous les ports et switch Open vSwitch :
```

```
sudo ovs-vsctl show

# Ajouter une regle de forwarding sur s1 :
sudo ovs-ofctl add-flow s1 [match],actions=[actions]

# Supprimer une regle de forwarding sur s1 :
sudo ovs-ofctl del-flows s1 [match]

# Rediriger tous les paquets provenant de h1 vers h2
sudo ovs-ofctl add-flow s1 in_port="s1-eth1",actions=output:"s1-eth2"

# 2 facon de dropper tous les paquets IP a destination de h1 :
sudo ovs-ofctl add-flow s1 ip,nw_dst=10.0.0.1,actions=drop
sudo ovs-ofctl add-flow s1 dl_dst=[MAC h2],actions=drop

# Rediriger tous les paquets a destination de h1 vers h2 :
sudo ovs-ofctl add-flow s3 ip,nw_dst=10.0.0.1,actions=mod_dl_dst:[MAC h2],\
   mod_nw_dst:10.0.0.2,output:"s1-eth2"

# Pour voir la documentation :
man ovs-ofctl
```

La documentation `ovs-ofctl` est aussi disponible ici : `http://www.openvswitch.org/support/dist-docs/ovs-ofctl.8.txt`

Installez les règles de forwarding permettant à h1 et h2 de se parler. Faîte au plus simple.

Complétez ou modifiez vos règles afin que tous les hôtes puissent se parler. Vérifiez avec `pingall`.

Modifiez vos règles afin de bloquer le port TCP/80 entre h1 et h2. Vérifiez avec netcat :

Listing 15: Ajouter des règles Open vSwitch

```
# Pour lancer un serveur avec netcat :
sudo nc -vl -p [port]
# Pour se connecter au serveur netcat :
sudo nc -v [ip] [port]
```

Modifiez vos règles pour bloquer le port TCP/80 de h1 vers h2 seulement pour les connexions entrantes. Vérifiez avec netcat.