TP Introduction aux Réseaux et Mininet

Dino Lopez Pacheco dino.lopez@univ-cotedazur.fr

1 Introduction

Dans ce TP, vous trouverez quelques exercices qui vous permettront de réaffirmer vos connaissances acquises à propos de l'architecture physique des réseaux, ainsi que les phénomènes observés lors de l'échange de paquets (messages).

2 Architecture Internet

1. Les switches sont utilisés pour fournir accès réseaux aux clients. Supposez que vous avez des commutateurs (switches) avec 5 ports chacun. Notez que les switch doivent toujours réserver un port pour l'accès Internet (ce qui laisse donc 4 ports pour les clients). Quelle topologie réseau construire afin de donner accès internet à 20 clients ? combien de switches vous faut-il au total ? Dessinez votre topologie.

   On doit donner accès aux clients et distribuer la bande passante au plus équitable possible. C'est donc une topologie en arbre (inversé) symétrique qu'on choisit naturellement. Avec un switch, on a 4 clients et on garde un port pour l'accès Internet. On augmente l'arbre d'un degré, et nous nous retrouvons avec 4 switches (= 16 ports disponibles). Il nous faut augmenter la profondeur de l'arbre, mais uniquement pour ajouter 2 switches de plus. Question extra : Combien de ports nous avons au total pour connecter des clients ?

2. Quelle profondeur doit avoir une topologie arbre pour pouvoir donner service à n clients si on possède de switches à p ports uniquement ?

   Supposez que nos switches ont *p* ports et l'un de ces ports doit être utilisé pour l'interconnexion avec un switch. Donc, nous aurons *c = p - 1* ports disponibles pour les clients.
   G grâce à l'exercice précédent, on voit que, au niveau X de l'arbre, nous avons c^X ports disponibles. Il faut donc que c^X >= n. Il ne nous reste qu'à utiliser les propriétés des logarithmes pour trouver notre bonne valeur de X.

3. Les routeurs sont de dispositifs qui permettent d'accéder à d'autres réseaux. De ce fait, chaque réseau doit posséder au moins 1 routeurs afin d'accéder à l'Internet. Ajoutez un routeur à votre topologie réseau de la question 1.

   Connectons un switch en haut de l'arbre inversé.

4. Supposez que l'architecture réseau faite dans le point 3 doit être répliqué 4 fois, car par exemple, nous avons 4 départements indépendants dans des bâtiments séparés. Si nous savons que chaque routeur doit avoir au moins 2 liaisons vers d'autres routeurs afin d'éviter les isolements en cas de panne, proposez une topologie valable pour l'interconnexions de vos 4 sites.

   Note : l'exercice comporte la phrase « au moins 2 liaisons ». Avec 3 routeurs ou moins, la topologie générée serait une topologie forcement circulaire (*ring*). Avec 4 routeurs ou plus, la topologie peut être circulaire ou maillée (*mesh*), selon vos choix. Pourquoi dit-on donc que la topologie du cœur de l'internet est maillée ? Parce que la

géographie fait que les réseaux circulaires sont souvent difficiles à créer. Puis, 2 liaisons entre routeurs est vraiment le strict minimum demandé ! Voir par exemple la topologie du réseau français RENATER https://www.researchgate.net/figure/RENATER-topology-with-45-sites-and-54-links-between-them_fig3_301413394

5. En partant du schéma entier d'interconnexion des 4 sites, identifiez les topologies réseaux qui ont été utilisés.

   Il y a une topologie mallée, en arbre et étoile (sur chaque feuille de l'arbre)

6. Les réseaux n'ont pas une capacité infinie. Décrivez comment un réseau à commutation de circuit et un réseau à commutation de paquets peuvent expérimenter des phénomènes de congestion.

   Plusieurs clients émettant au total plus des paquets que ce qui peut être transféré par les routeurs ou switches, conduisent à des problèmes de congestion dans un réseau à commutation de paquets.
   Plusieurs clients essayant de créer un circuit virtuel satureront les switches et conduiront également à un problème de saturation.

7. Donnez un avantage et un désavantage des réseaux à commutation par paquet (« packet-switched ») par rapport aux réseaux à commutation par circuit (« circuit-switched »).

   Les réseaux à commutation de paquets établissent rapidement une connexion entre 2 hosts finaux. Mais ils introduisent un délai par paquet, ce que à la fin, peut-être très pénalisant pour l'application

3 L'émulateur réseau Mininet

Mininet est un outil qui permet de déployer un réseau, composé de machines virtuelles (VMs par ses sicles en anglais) ou réelles, exécuter le vrai code des machines, utiliser des vrais commutateurs (matériel ou logiciel), à l'aide d'un nombre réduit de commandes. Lorsque le réseau déployé est composé uniquement par de VMs et de commutateurs logiciel, elle pourrait être déployé à l'intérieur d'une seule machine (un ordinateur portable, par exemple).

3.1 Préparation de votre machine

Nous supposons ici que vous avez installez la VM mise en ligne par le SI. Dans cette VM, Mininet a été déjà installé. Il ne reste que vérifier la présence de quelques packages.
Cette étape est très importante : l'objectif est de vérifier que votre machine est prête à exécuter Mininet (outil essentiel pour quelques futurs TPs).
Voici une liste de packages qui doivent être installés si nécessaire. **Installez les logiciels uniquement avec APT et non pas par les sources !!** (si vous avez de doutes par rapport à certaines commandes, n'hésitez pas à discuter avec votre encadrant TP).

- Commencer par vérifier que Python2.7 est installé dans votre ordinateur : commande « dpkg -l | grep python2 ». Si ce n'est pas le cas, installez-le avec la commande « $ sudo apt install python2 python2.7 » car Mininet n'est pas encore totalement porté sur Python 3.
- xterm
- git

Depuis votre espace personnel (*home directory*), téléchargez « POX » avec la commande :

$ git clone http://github.com/noxrepo/pox
Cette commande créera un dossier « pox » dans votre espace personnel.

Et un package à désinstaller (s'il est présent -avec la commande dpkg- et si vous avez une version (L)Ubuntu 20.04) : « openvswitch-testcontroller ». Utilisez la commande « $ sudo apt-get purge openvswitch-testcontroller ». Ensuite, redémarrez la VM.

3.2 Premier contact avec Mininet

Les exercices ici proposés sont inspirés du tutoriel du site officiel de Mininet http://mininet.org/walkthrough/

Premièrement, retenez bien que vous devez exécuter Mininet avec les droits d'administrateur (i.e. avec la commande sudo).

```
$ sudo mn
```

Si tout s'est bien passé, à la fin vous devez voir l'invite de commande Mininet (le mode CLI de Mininet) « mininet> », et une topologie réseau devrait être créée. La topologie par défaut est appelée minimal.

Toutes les commandes disponibles dans la CLI mininet peuvent être liste avec la commande « help »

```
mininet> help
```

Les clients qui ont été créés dans notre réseau virtuel peuvent être listés avec la commande « nodes »

```
mininet> nodes
```

Par convention, les clients du réseau sont appelés h1, h2, … hX. Les switches (commutateurs) sont appelés s1, s2, … sX, et, à oublier pour l'instant, les contrôleurs des switches sont appelés c0, c1, … cX.

8. Combien de switches et clients sont disponibles dans votre réseau virtuel ?

   Un switch s1 et deux clients (h1 et h2)

Les informations relatives à chaque nœud créé peuvent être listé avec la commande « dump ».

9. Décrivez ce que vous obtenez comme information des switches et clients du réseau.

   <Host h1: h1-eth0:10.0.0.1 pid=1242>
   <Host h2: h2-eth0:10.0.0.2 pid=1243>
   <OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=1247>
   <OVSController c0: 127.0.0.1:6633 pid=1234>
   2 hosts, on voit les @IP des hosts, PID des namespaces et un switch avec 2 interfaces

Les liens créés peuvent être listés avec la commande « net »

10. Dessinez la topologie créée.

    h1 ----- s1 ------- h2

La puissance de Mininet réside, entre autres, sur le fait que les clients sont des images de la machine hôte (là où Mininet est installé) et ils ont accès à l'espace système de son utilisateur. Il est donc possible d'exécuter « à l'intérieur » des clients plusieurs commandes disponibles dans l'OS. Pour cela, vous devez précéder la commande par le nom de la machine virtuel qui

l'exécutera. Par exemple, pour voir la configuration des interfaces réseaux disponibles dans le client h1, on exécute

```
mininet> h1 ifconfig -a
```

11. Ajoutez à votre dessin de la topologie réseau les adresses IP associées à chaque client, ainsi que le masque de réseau.
12. Le client h2 peut être atteint par le client h1 ? Vérifiez-le par la commande ping. Donnez l'instruction complète que vous avez utilisée.

    mininet> h1 ping -c 3 h2

Pour sortir de mininet, utilisez la commande « exit ».

Pour éviter d'indiquer à chaque fois qu'elle machine doit exécuter quelle commande, et si vous voulez travailler avec une console proche de celle des machines réelles, vous pouvez

1. Exécuter Mininet avec l'option « -x »
   $ sudo mn -x
2. Exécuter la commande xterm et indiquer le client à attacher.
   mininet> xterm h1

La première option ouvrira automatiquement une fenêtre xterm pour chaque client, switch et contrôleur présent dans la topologie.

3.3 D'autres topologies réseaux

Mininet fournit, en plus de la topologie « minimal », la topologie « single », « linear » et « tree ». Pour charger l'une de ces topologies, utilisez l'option « --topo ». Par exemple :

```
$ sudo mn ––topo single
```

« single» tout court donne la même topologie que minimal, mais on peut ajouter également comme argument un chiffre, qui indique le nombre de clients à créer. Par exemple single,3.

Exécutez les commandes suivantes et dessinez la topologie créée:

13. $ sudo mn --topo linear

    H1 ---- s1 ---- s2 ---- H2

14. $ sudo mn --topo linear,3

    h1 ---- s1 ---- s2 ---- s3 ---- h3
                |
               h2

15. $ sudo mn --topo tree

    h1 ---- s1 ---- h2

16. $sudo mn --topo tree,2

    h1 ---- s2 ---- s1 ---- s3 ---- h4
            | |
           h2   h3

17. $sudo mn --topo tree,depth=2,fanout=3. À quoi servent les paramètres depth et fanout ?

                S1
            / | \

```
   S2    S3    S4
  / | \ / | \/ | \
  H1 ....
```

3.4 Topologies personnalisées

Et si on souhaitait travailler avec une topologie autre que tree, linear ou simple ? Bien, dans ce cas là, il faut créer une topologie personnalisée à l'aide de l'API Python de Mininet. Supposez que vous voulez créer un client h1 connecté au client h2 par le switch s1. Voici le code Python à écrire :

```
from mininet.topo import Topo

class Test_Topo(Topo):
    def __init__(self):
        "Create P2P topology"

        # Initialize topology
        Topo.__init__(self)

        # Add hosts and switches
        h1 = self.addNode('h1')
        h2 = self.addNode('h2')
        s1 = self.addSwitch('s1')

        # Add links
        self.addLink(h1, s1)
        self.addLink(h2, s1)

topos = {
        'toptest': (lambda: Test_Topo())
}
```

18. Sauvegardez dans un fichier toptest.py le code précédent. Ensuite :
    - Exécutez POX avec la commande « $ ~/pox/pox.py --verbose forwarding.l2_learning ». Laissez tourner l'application.
    - Dans un autre terminal, exécutez le programme Python avec la commande « $ sudo mn --controller=remote --custom /chemin/vers/toptest.py --topo toptest »
19. Si le réseau est créé correctement, ouvrez un terminal pour chaque client et explorez la configuration des cartes réseaux.

    ```
    mininet> h1 ifconfig -a
    mininet> h2 ifconfig -a
    ```

20. Vérifiez par la commande ping la connectivité entre les machines h1 et h2.

    ```
    mininet> h1 ping -c3 10.0.0.2
    ```

21. Sortez de Mininet mais laissez POX tourner.

3.5 Exécution automatiques de commandes

Pouvoir créer un réseau personnalisé est une bonne chose, mais devoir exécuter à chaque fois à la main les différentes commandes pour exécuter nos expériences (par exemple, exécuter un ping pour vérifier la connectivité entre 2 machines), n'est pas très confortable.

Pour résoudre ce problème là, on utilisera encore l'API Python. Le code suivant exécute un ping depuis la machine h1 vers la machine h2. Comme vous pouvez l'apprécier, pour exécuter une commande quelconque sur une machine avec l'API, on fait appel à la méthode cmd() et on donne comme argument la commande à exécuter sous la forme d'une chaine de caractères.

```
from mininet.net import Mininet
from mininet.topo import Topo
from mininet.node import Node
from mininet.cli import CLI
from mininet.link import TCLink
from mininet.log import setLogLevel
from mininet.node import Controller, RemoteController, OVSController

class Test_Topo(Topo):
    def __init__(self):
        "Create P2P topology"

        # Initialize topology
        Topo.__init__(self)

        # Add hosts and switches
        h1 = self.addNode('h1')
        h2 = self.addNode('h2')
        s1 = self.addSwitch('s1')

        # Add links
        self.addLink(h1, s1)
        self.addLink(h2, s1)

topos = {
        'toptest': (lambda: Test_Topo())
}

def topTest():
        topo = Test_Topo()
        net = Mininet(controller=RemoteController, topo=topo,
link=TCLink)
        net.start()
        h1 = net.get('h1')
        print h1.cmd('ping -c 3 10.0.0.2')
        CLI(net) ; # sans cette ligne, on ne verrait jamais le CLI
        net.stop() ; # ne pas oublier de détruire le réseau

if __name__ == '__main__':
        setLogLevel('info')
        topTest()
```

Cependant, notez bien que pour exécuter l'ensemble du code ajouté, on fera appel directement à Python cette fois-ci (rappel : POX doit toujours tourner dans un autre terminal).

```
$ sudo python2.7 /chemin/vers/fichier.py
```

22. Lisez, analysez et testez le code ci-dessus. Vérifiez que le ping depuis le client h1 atteint bien le client h2.

23. Créez un script shell qui reçoit comme argument un chiffre entier N, et qui ensuite écrit N fois sur le terminal « Hello World X », où X est le nombre de fois que la chaine de caractères a été écrite.

```
#!/bin/bash

for ((i=0; i<$1; i++))
do
    echo "Hello World $i"
done
```

24. Faites que le host h1 exécute votre script shell automatiquement (i.e. depuis votre script python)

    h1.cmd("bash hello.sh 5")

3.6 Optionnel : Et c'est la fin

25. Dans un fichier testdel.py, créez la topologie suivante h1 --- s1 --- s2 --- h2 et ajoutez un délai de 10ms entre s1 et s2 (reprenez le code ci-dessous). Vérifiez par la commande ping que le délai est bien respecté.

```
#!/usr/bin/env python

from mininet.topo import Topo
from mininet.net import Mininet
from mininet.node import Node
from mininet.link import TCLink
from mininet.log import setLogLevel
from mininet.cli import CLI
from mininet.node import Controller, RemoteController, OVSController

class TestDel(Topo):
        "Double switch connecting a sender and a receiver"
        def __init__(self):
                Topo.__init__(self)
                s1 = self.addSwitch('s1')
                s2 = self.addSwitch('s2')
                h1 = self.addNode('h1')
                h2 = self.addNode('h2')
                self.addLink(s1, h1)
                self.addLink(s1, s2, delay='10ms')
                self.addLink(s2, h2)

topos = {
        'testdel': (lambda: TestDel())
}

def constTopo():
        topo = TestDel()
        net = Mininet(controller=remote, topo=topo, link=TCLink)
        net.start()
        CLI(net)
        net.stop()

if __name__ == '__main__':
        setLogLevel('info')
        constTopo()
```

La commande ping devrait montrer un temps d'un peu plus de 20ms, ce qui correspond au temps pour que la requête ping arrive au destinataire, plus le temps pour que la réponse arrive à l'émetteur.

26. Copiez votre fichier testdel.py dans un fichier testbw.py afin de mettre un lien de 100Mbps de capacité entre s1 et s2. Ensuite, manuellement, sur h2 exécutez la commande « iperf -s -i 1 » et après sur h1, la commande « iperf -c 10.0.0.2 -t 10 ». Interprétez le résultat obtenu sur h2 et dites si la limite de bande passante a été respectez. Argumentez votre réponse.

```
#!/usr/bin/env python

from mininet.topo import Topo
from mininet.net import Mininet
from mininet.node import Node
from mininet.link import TCLink
from mininet.log import setLogLevel
from mininet.cli import CLI

class TestBw(Topo):
        "Double switch connecting a sender and a receiver"
        def __init__(self):
                Topo.__init__(self)
                s1 = self.addSwitch('s1')
                s2 = self.addSwitch('s2')
                h1 = self.addNode('h1')
                h2 = self.addNode('h2')
                self.addLink(s1, h1)
                self.addLink(s1, s2, bw=100)
                self.addLink(s2, h2)

topos = {
        'testbw': (lambda: TestBw())
}

def constTopo():
        topo = TestBw()
        net = Mininet(topo=topo, link=TCLink)
        net.start()
        CLI(net)
        net.stop()

if __name__ == '__main__':
        setLogLevel('info')
        constTopo()
```

Sur une machine bien dimensionnée en mémoire et CPU, on aura un débit légèrement inférieur aux 100Mbps. Sans le paramètre bw=100 le débit observé par iperf est largement supérieur à 100Mbps.

27. si vous avez réussi l'exercice précédent, nous vous proposons d'expérimenter l'impact du trafic sur le délai observé entre 2 hosts. Pour cela, vous devez créer une topologie avec 2 hosts (h1 et h2) connectés vers un switch s1, d'autres 2 hosts (h3 et h4) connectés vers un switch s2 et une liaison entre ces 2 switches. La liaison entre les 2 switches doit avoir une limite de bande passante de 100Mbps, ainsi qu'un délai de propagation de 10ms. Ensuite, vous démarrez iperf pour effectuer un transfert entre h1 et h3. Ensuite, vous déterminez le délai avec ping entre h2 et h4. Effectuez à nouveau l'expérience ping, mais cette fois-ci sans le transfert entre h1 et h3. Commentez les résultats.

Comportement attendu : si la bande passante de 100Mbps et le délai de propagation sont bien respectés, ping doit trouver un délai (temps) de 20ms, mais ce délai doit augmenter lors que le transfert avec iperf est en marche.