Réseaux Datacenters/HPC
TP - Administration et utilisation du SDN

CEA/ENSIIE – 2017-2018

22 mai 2018

Résumé

Le but de ce TP est de prendre en main un contrôleur SDN libre et open-source pour comprendre les apports de ce genre de contrôleur sur la gestion, l'automatisation, la configuration d'un réseau d'entreprise. Nous allons voir au cours de ce TP quelques fonctionalités spécifiques des contrôleurs SDN.

Table des matières

1 Notation et rendu

Comme indiqué en cours, ce module sera noté en grande partie sur le rendu du TP mais aussi sur votre participation. Les éléments de notation qui seront particulièrement observés :

1. La qualité rédactionnelle : forme et fond
2. La qualité des schémas
3. La pertinence des réponses

Les rapports peuvent être faits à maximum 2. Tout nom manquant sur le rendu aura automatiquement 0. Si vous le faites à 2, il n'y a qu'un seul rendu attendu.

1.1 Le rendu

Le rendu final sera une archive (zip) contenant les éléments suivants :

1. Le rapport
2. Les scripts écrits
3. Les graphiques (on limitera à maximum 2MB par graphique, si vous en avez des plus lourds, ne les mettez pas dans l'archive)

Pour rendre, vous devez envoyer par mail l'archive au format NOM1_NOM2.zip à l'adresse damien.gros@cea.fr.

Tout retard dans le rendu entrainera la note de 0 aux personnes du groupe.

Le rendu pour ce TP est fixé au lundi 28 mai 2018 19h.

2 Mise en place de l'environnement

Pour réaliser ce TP, nous allons utiliser un machine virtuelle qui a été spécialement préparée. Elle est sur le calculateur dans le répetoire `/home/shared/grosd/tp-sdn`.

Cette machine virtuelle s'utilise avec l'hyperviseur PCOCC. Comme vous l'avez vu dans un précédent, pour utiliser les machines au travers de PCOCC, il est nécessaire de créer une entrée dans le fichier `~/.pcocc/templates.yaml`. Pensez à adapter le chemin vers l'image.

```
mycentos74-tp-sdn:
    resource-set: ens-cluster
    image: /home/username/.pcocc/images/test-centos-74-tp-sdn
    user-data: ~/.pcocc/cloud-user-data/test-centos74-tp-sdn
```

Listing 1 – Extrait du contenu du fichier templates.yaml

La seconde manipulation est d'ajouter une clé SSH dans le fichier `~/.pcocc/cloud-user-data/test-centos74-tp-sdn`. Pensez à modifier le nom de connexion.

```
#cloud-config

user: gros
ssh_authorized_keys:
 - ssh-rsa cle_ssh.pub
```

Listing 2 – Extrait du fichier sdn.ci

Une fois ces deux fichiers renseignés et que vous avez copié la machine virtuelle dans votre home directory, vous pouvez la lancer avec la commande suivante.

```
\$ pcocc alloc -t 4:00:00 -c 4 mycentos74-tp-sdn:1
```

Listing 3 – Lancement de la machine virtuelle

Petit rappel sur l'utilisation de la commande :

— alloc : permet de lancer une machine virtuelle en mode interactif
— -t : permet de définir la durée pendant laquelle la machine sera exécutée
— -c : définit le nombre de processeur alloué à la machine
— -p : permet de définir sur quelle partition s'exécute la machine virtuelle
— centos7 ;4 :1 : on lance une unique instance de notre machine virtuelle

Une fois la machine lancée, pour vous connectez, il vous suffit de taper la commande suivante :

```
\$ pcocc ssh vm0
```

Listing 4 – Connexion à la machine virtuelle

Vous obtenez donc un shell dans la machine virtuelle. Pour la suite du TP, soit vous ouvrez autant de shell que nécessaire, soit vous utilisez la commande `screen`.

Une fois dans la VM, configurez la seconde interface avec l'adresse 10.0.0.20

2.1 Lancement des outils spécifiques

Pour la réalisation ce TP, nous allons utiliser un contrôleur SDN open-source nommé `OpenDaylight`. https ://www.opendaylight.org

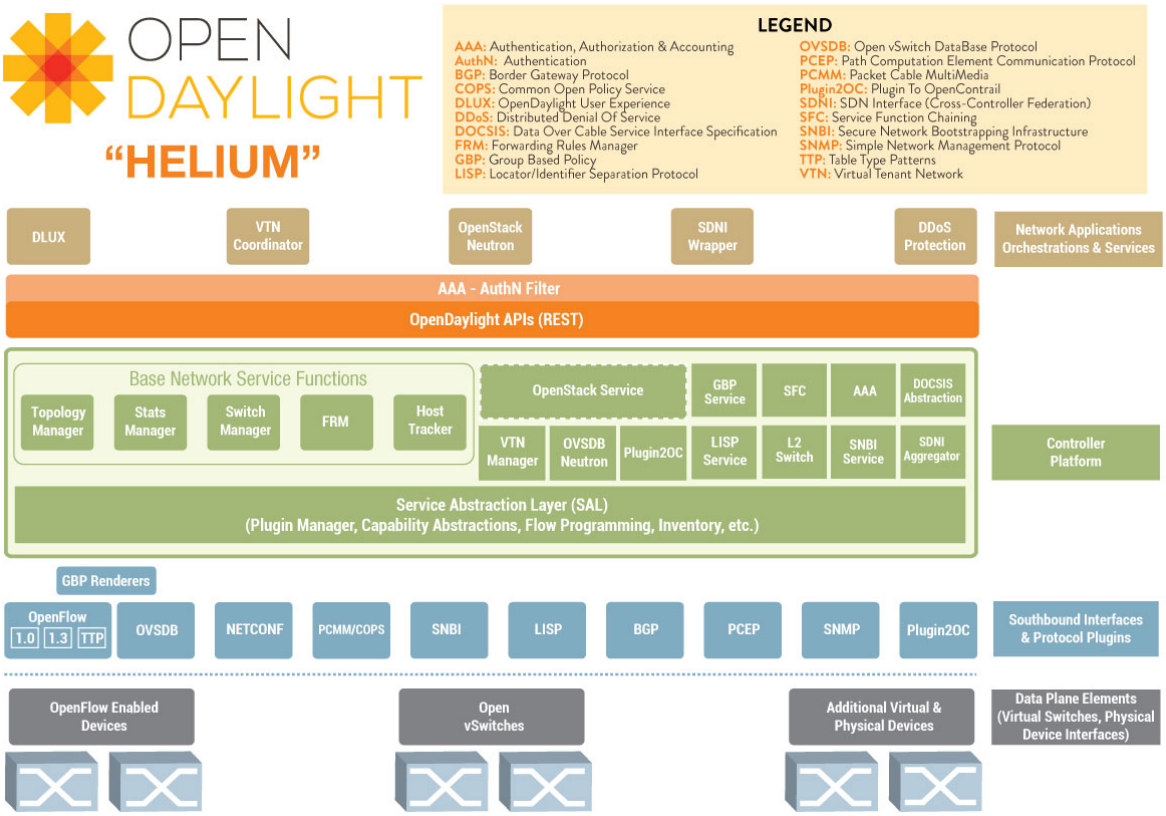
En plus d'être libre et gratuit, il offre aussi un ensemble de logiciels permettant d'ajouter un certain nombre de fonctionnalités à ce dernier.

La simulation du réseau se fera au travers de l'outil `mininet` http ://mininet.org/.

2.1.1 OpenDaylight

Comme une grande partie des contrôleurs SDN présent sur le marché, OpenDaylight est écrit en Java. La version actuelle est **karaf**.

Le schéma 2.1.1 décrit l'architecture fonctionnelle du contrôleur.



Pour lancer le contrôleur, il suffit de se déplacer dans le répertoire `karaf` et de taper : (il faut au préalable se déplacer dans le home de toto, ou récupérer le zip dans /home/shared/grosd/karaf-VERSION.zip)

```
sudo ./bin/karaf
```

Listing 5 – Lancement du contrôleur

Le mot de passe du compte `toto` est `toto`.

Une fois lancé, vous obtenez un shell pour installer, configurer et administrer le contrôleur.

Par défaut, le contrôleur n'a que peu de fonctionnalités. Nous allons donc commencer par installer les fonctionnalités nécessaire à ce TP.

Pour voir l'ensemble des applications installées, il vous suffit de taper :

```
feature:list -i
```

Listing 6 – Lancement du contrôleur

Avec la commande suivante, installez les

```
feature:install app
```

Listing 7 – Installation des features

Les applications suivantes permettent de créer, visualiser et administrer les tables de flux présentes sur les switchs contrôlés par OpenDaylight

```
odl-dlux-core
odl-dluxapps-applications
odl-dluxapps-nodes
odl-dluxapps-topology
odl-dluxapps-yangman
odl-dluxapps-yangui
odl-dluxapps-yangutils
odl-dluxapps-yangvisualizer
```

Listing 8 – Application de visualisation des flux

Les applications suivantes permettent au contrôleur d'administrer aussi le niveau 2 des switchs, nécessaire pour l'utilisation de mininet :

```
odl-l2switch-addresstracker
odl-l2switch-all
odl-l2switch-arphandler
odl-l2switch-hosttracker
odl-l2switch-loopremover
odl-l2switch-packethandler
odl-l2switch-switch
```

Listing 9 – Application de contrôle du niveau 2

L'ensemble des applications suivantes permettent de pouvoir communiquer avec les switchs en utilisant OpenFlow :

```
odl-openflowjava-protocol
odl-openflowplugin-app-config-pusher
odl-openflowplugin-app-forwardingrules-manager
odl-openflowplugin-app-reconciliation-framework
odl-openflowplugin-app-topology
odl-openflowplugin-flow-services
odl-openflowplugin-nsf-model
odl-openflowplugin-southbound
```

Listing 10 – Application d'OpenFlow

Ces applications permettent d'avoir une interface Nord pour envoyer des requêtes de type REST et d'interagir avec le contrôleur depuis l'extérieur :

```
odl-restconf
odl-restconf-all
odl-restconf-noauth
```

Listing 11 – Application REST

Les applications suivantes permettent au switch de communiquer sur SNMP avec le contrôleur pour récupérer leurs informations :

```
odl-snmp-plugin
odl-snmp4sdn-snmp4sdn
```

Listing 12 – Application de gestion SNMP

Les applications permettent d'utiliser les API et d'obtenir de la documentation sur ces dernières :

```
odl-yanglib
odl-yangtools-common
odl-yangtools-yang-data
```

```
4 odl-yangtools-yang-parser
```

Listing 13 – Application de contrôle du niveau 2

Pour vous assurer que tous les services nécessaires ont bien été installés et démarrés, il vous suffit d'utiliser la commande `netstat` :

```
netstat -tanp
```

Listing 14 – Commande netstat

— 8181 : interface HTTP/HTTPS de connexion au contrôleur et interface RESTCONF
— 6633 : serveur OpenFlow
— 6653 : serveur OpenFlow

2.1.2 Mininet

Une fois le contrôleur lancé, il nous faut construire un réseau avec des switchs supportants le protocole OpenFlow13. Pour cela, nous allons utiliser le logiciel `mininet`.

Ce logiciel est normalement déjà installé sur votre machine virtuelle, si ce n'est pas le cas, installez-le depuis les dépôts.

```
1 sudo apt install mininet
```

Listing 15 – Installation de mininet

Une fois installé, vous pouvez commencer par supprimer les configurations pré-existantes :

```
1 sudo mn -c
```

Listing 16 – Installation de mininet

Pour lancer mininet, il faut lui définir une topologie de base ainsi que des informations sur le contrôleur :

```
1 sudo mn --topo=linear,4,2 --controller=remote,ip=10.0.0.20:6633 --switch=ovsk,
    protocols=OpenFlow13 --mac
```

Listing 17 – Installation de mininet

— topo : définit la topologie, avec 4 switchs et 2 hôtes par switchs
— controller : définit où est situé le contrôleur
— switch : ovsk, définit le type de switch que l'on veut utiliser, ici c'est des OpenVswitchs
— protocols : on lui spécifie le protocole, ici il faut bien faire attention à la version d'OpenFlow
— mac : on va donner des adresses MAC aux hosts

Une fois dans l'interface avec la topologie chargée, vous pouvez interagir avec les switchs ainsi que les hôtes. En vous basant uniquement sur les informations données par mininet faites l'exercice suivant :

Ex. 1 — Information de topologie

1. Dessinez la topologie que vient de vous créer mininet en spécifiant les noms à chaque fois

Answer (Ex. 1) — Voir plus loin pour la réponse

Pour vérifier que mininet a bien connecté les switchs et qu'il a bien connecté les switchs au contrôleur SDN, vous pouvez taper les deux commandes suivantes :

```
1 mininet> sh ovs-vsctl show
mininet> sh ovs-vsctl list controller
```

Faites un schéma topologique représentant le branchement des hôtes sur les switchs ainsi que l'interconnexion entre les switchs.

3 Interaction entre le contrôleur et les switchs

L'architecture étant en place, il est temps de passer à la partie OpenFlow.

Dans un navigateur web lancé depuis la VM, connectez-vous au contrôleur SDN.

```
http://10.0.0.20:8181/index.html
```

Listing 18 – Connection au contrôleur SDN

Le login/mdp : admin/admin.

Une fois connecté, rendez-vous sur l'onglet "Topology". Que remarquez-vous ?

Pour résoudre ce problème, exécutez la commande suivante dans mininet :

```
1 mininet> pingall
```

Retournez dans le contrôleur et rafraichissez la page, vous devriez maintenant avoir l'ensemble de la topologie. Pour la bonne continuité du TP, il faut que les hôtes apparaissent dans la topologie.

Pour transmettre et récupérer de l'information vers ou depuis les switchs, vous avez la possibilité d'utiliser :

- Yangam : qui a été installé depuis le contrôleur
- postman : application externe, déjà installé sur la machine virtuelle, permettant de forger des requêtes de type REST

Dans un soucis de lisibilité et de compréhension, nous allons utiliser postman.

4 Création des flux

4.1 Paramétrage de Postman

Dans l'onglet **Authorization**, vérifiez que TYPE soit bien en `Basic Auth`.

Pour valider ces étapes, nous allons définir une configuration basique pour un switch.

Commençons par pousser une nouvelle table de flux. Pour cela, on doit spécifique la bonne url et la bonne action qui est PUT :

4.2 Création d'un premier flux simple

http://10.0.0.20:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:1/table/0/flow/101

- IP :PORT : celui du contrôleur, on se connecte du côté "nord"
- restconf/config/opendaylight-inventory :nodes/node : définit la configuration que l'on veut manipuler/interroger
- openflow :1 : définit le nom du switch sur lequel on veut effectuer l'action
- table : on spécifique que l'on va manipuler une table spécifique
- 0 : précise le numéro de la table
- flow/101 : on va manipuler le flux 101

Pour que la commande soit correctement interprétée par l'équipement, il est nécessaire de vérifier à la cohérence entre l'url entrée et les informations que l'on transmet dans le XML.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<flow xmlns="urn:opendaylight:flow:inventory">
  <id>101</id>
          <instructions>
                  <instruction>
                        <order>0</order>
                        <apply-actions>
                          <action>
                                  <order>1</order>
                                  <output-action>
                                    <output-node-connector>2</output-node-
    connector>
                                  </output-action>
        </action>
      </apply-actions>
                  </instruction>
                </instructions>
  <table_id>0</table_id>
        <priority>100</priority>
        <flow-name>Port1surPort2</flow-name>
        <match>
    <in-port>openflow:1:1</in-port>
  </match>
</flow>
```

Listing 19 – Envoie d'une table OF

Pour vérifier que la table est bien présente sur le switch, il suffit de taper la commande suivante :

```
sh ovs-ofctl -O OpenFlow13 dump-flows s1
```

Listing 20 – Commande permettant de dumper les tables

La nouvelle règle doit apparaitre dans la bonne table.

Pour vous assurez qu'elle est bien prise en compte, vous pouvez relancer la commande `pingall` et vous assurez que les paquets entre les paquets sur le switch OpenFlow1 sont bien autorisés ou bloqués suivant la communication.

Est-ce que le comportement obtenu est bien celui attendu ?

Ce premier exemple nous a permis de bloquer (ou d'autoriser) la communication entre deux ports d'un même switch. Nous allons maintenant aller plus loin en étudiant les différentes possibilités offertes par le contrôleur OpenDayLight.

4.3 Création d'un flux bloquant par adresse MAC

Dans ce second exemple, nous allons bloquer une adresse MAC. Pour cela, nous allons procéder de la même façon :

1. Définir la table cible
2. Définir un identifiant de flux dans cette table
3. Définir une priorité pour ce flux

Pour commencer, nous allons mettre la règle dans une autre table que la 0. Vous allez prendre la table 2.

Commencez par modifier l'URL de connexion au contrôleur pour interagir avec le switch OpenFlow4.

http://10.0.0.20:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:4/table/0/flow/126

Le listing suivant définit l'action de dropper les paquets ayant une adresse MAC spécifique.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<flow xmlns="urn:opendaylight:flow:inventory">
    <strict>false</strict>
    <instructions>
        <instruction>
            <order>0</order>
            <apply-actions>
                <action>
                    <order>0</order>
                    <drop-action/>
                </action>
            </apply-actions>
        </instruction>
    </instructions>
    <table_id>2</table_id>
    <id>126</id>
    <cookie_mask>255</cookie_mask>
    <installHw>false</installHw>
    <match>
        <ethernet-match>
            <ethernet-source>
                <address>00:00:00:00:00:04</address>
            </ethernet-source>
        </ethernet-match>
    </match>
    <hard-timeout>1200</hard-timeout>
    <cookie>3</cookie>
    <idle-timeout>3400</idle-timeout>
    <flow-name>FooXf3</flow-name>
    <priority>2</priority>
    <barrier>false</barrier>
</flow>
```

Listing 21 – OpenFlow bloquant une adresse MAC en source

Commencez par vérifier que la table a bien été prise en compte par le switch cible avec la commande 20.

Faites ensuite un ping entre l'hôte s1h4 et un autre. Pour cela, il suffit de taper la commande suivante :

```
h1s4 ping h1s1
```

Listing 22 – Commande permettant ping entre deux hôtes

Est-ce que le ping fonctionne ?

Pour résoudre cela, nous allons commencer par augmenter la priorité de la règle sans modifier la table. Donc, en reprenant le script 21, modifiez la priorité de la table. Appliquez la table sur le même switch OpenFlow4.

Refaites le cheminement pour vérifier que la table est correctement appliquée et refaites un ping 22.

Est-ce que le ping fonctionne ?

Pour résoudre cela, nous allons remettre une priorité à 2 mais cette fois-ci nous allons changer de table. Donc, en reprenant le script 21, modifiez la table. Appliquez la table sur le même switch OpenFlow4.

Refaites le cheminement pour vérifier que la table est correctement appliquée et refaites un ping 22.

Est-ce que le ping fonctionne ?

Pour résoudre cela, nous allons augmenter la priorité de la règle sans modifier la table. Donc, en reprenant le script 21, modifiez la priorité de la table. Appliquez la table sur le même switch OpenFlow4.

Refaites le cheminement pour vérifier que la table est correctement appliquée et refaites un ping 22.

Est-ce que le ping fonctionne ?

Quelles sont donc les conclusions que l'on peut tirer de ces différentes manipulations ?

4.4 Création d'un flux bloquant par adresse IP

Dans ce troisième exemple, nous allons bloquer une adresse IP.

http://10.0.0.20:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:2/table/0/flow/10

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<flow xmlns="urn:opendaylight:flow:inventory">
    <strict>false</strict>
    <instructions>
        <instruction>
            <order>0</order>
            <apply-actions>
                <action>
                    <order>0</order>
                    <drop-action/>
                </action>
            </apply-actions>
        </instruction>
    </instructions>
    <table_id>0</table_id>
    <id>10</id>
    <cookie_mask>255</cookie_mask>
    <installHw>false</installHw>
    <match>
        <ethernet-match>
            <ethernet-type>
                <type>2048</type>
            </ethernet-type>
        </ethernet-match>
        <ipv4-destination>10.0.0.2/32</ipv4-destination>
    </match>
    <hard-timeout>1200</hard-timeout>
    <cookie>1</cookie>
    <idle-timeout>3400</idle-timeout>
    <flow-name>FooXf1</flow-name>
    <priority>200</priority>
    <barrier>false</barrier>
</flow>
```

Listing 23 – OpenFlow bloquant une adresse IP en destination

Appliquez la table. Vérifiez qu'elle soit bien appliquée au niveau du switch OpenFlow2. Essayez de faire un ping mettant en jeu l'adresse IP bloquée en destination.

4.5 Création d'un flux bloquant sur de multiples critères

http://10.0.0.20:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:4/table/2/flow/131

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<flow xmlns="urn:opendaylight:flow:inventory">
    <strict>false</strict>
    <instructions>
        <instruction>
            <order>0</order>
            <apply-actions>
                <action>
                    <order>0</order>
                    <dec-nw-ttl/>
                </action>
            </apply-actions>
        </instruction>
    </instructions>
    <table_id>2</table_id>
    <id>131</id>
    <cookie_mask>255</cookie_mask>
    <match>
        <ethernet-match>
            <ethernet-type>
                <type>2048</type>
            </ethernet-type>
            <ethernet-destination>
                <address>ff:ff:29:01:19:61</address>
            </ethernet-destination>
            <ethernet-source>
                <address>00:00:00:11:23:ae</address>
            </ethernet-source>
        </ethernet-match>
        <ipv4-source>17.1.2.3/32</ipv4-source>
        <ipv4-destination>172.168.5.6/32</ipv4-destination>
        <ip-match>
            <ip-protocol>6</ip-protocol>
            <ip-dscp>2</ip-dscp>
            <ip-ecn>2</ip-ecn>
        </ip-match>
        <tcp-source-port>25364</tcp-source-port>
        <tcp-destination-port>8080</tcp-destination-port>
        <in-port>0</in-port>
    </match>
    <hard-timeout>1200</hard-timeout>
    <cookie>8</cookie>
    <idle-timeout>3400</idle-timeout>
    <flow-name>FooXf8</flow-name>
    <priority>2</priority>
    <barrier>false</barrier>
</flow>
```

Listing 24 – OpenFlow bloquant un ensemble de critères

On ne peut pas vérifier que ça fonctionne dans Mininet, mais on peut vérifier que la table est bien appliquée.

4.6 Nettoyage des tables

Une fois une table dans un contrôlleur, cette reste tant qu'elle n'est pas supprimée du contrôleur. L'inconvénient majeur est qu'il est nécessaire de faire règle par règle.

Pour cela, il vous suffit, à partir de PostMan de modifier le type de requête HTTP en passant de PUT à DELETE et d'envoyer sur le contrôleur. Vous pouvez ensuite vous assurer que l'entrée dans la table a bien été supprimé en passant la commande 20.

Il est nécessaire de faire cela sur chaque entrée avant de passer à la partie suivante sinon vous aurez indésirables.

5 Autres applications

Dans cette seconde partie, nous allons changer de topologie mais aussi introduire les VLAN, qui sont des éléments que l'on retrouve aussi dans les réseaux d'entreprises.

Pour pouvoir générer des VLAN dans mininet, il est nécessaire d'utiliser un script python, qui est nommé vlan.py.

```
sudo mn -c
sudo mn --custom  vlan.py --topo mytopo --controller=remote,ip=10.0.0.20:6633 --
    switch=ovsk,protocols=OpenFlow13 --mac
```

Listing 25 – OpenFlow bloquant un ensemble de critères

Pour voir la nouvelle topologie, lancez un `pingall` depuis mininet pour que le contrôleur puisse découvrir les hôtes connectés et ensuite, regardez dans l'interface web du contrôleur pour visualiser la nouvelle topologie.

5.1 Création d'un flux basé sur les identifiants de VLAN

Sur les mêmes exemples que pour les adresses MAC ainsi que pour les adresses, on peut mettre des filtres sur des identifiants de VLAN. Pour éviter de tout bloquer, nous allons laisser la partie filtre sur les adresses MAC pour pouvoir vérifier que les autres communications au sein de ce VLAN sont toujours fonctionnelles.

L'application se fait exactement de la même façon que précédemment.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<flow xmlns="urn:opendaylight:flow:inventory">
    <strict>false</strict>
    <instructions>
        <instruction>
            <order>0</order>
            <apply-actions>
                <action>
                    <order>0</order>
                    <dec-nw-ttl/>
                </action>
            </apply-actions>
        </instruction>
    </instructions>
    <table_id>2</table_id>
    <id>138</id>
    <cookie_mask>255</cookie_mask>
```

```
    <match>
        <ethernet-match>
            <ethernet-type>
                <type>2048</type>
            </ethernet-type>
            <ethernet-destination>
                <address>00:00:00:00:00:01</address>
            </ethernet-destination>
            <ethernet-source>
                <address>00:00:00:00:00:03</address>
            </ethernet-source>
        </ethernet-match>
        <vlan-match>
            <vlan-id>
                <vlan-id>300</vlan-id>
                <vlan-id-present>true</vlan-id-present>
            </vlan-id>
      </vlan-match>
    </match>
    <hard-timeout>1200</hard-timeout>
    <cookie>15</cookie>
    <idle-timeout>3400</idle-timeout>
    <flow-name>FooXf15</flow-name>
    <priority>2</priority>
    <barrier>false</barrier>
</flow>
```

Listing 26 – OpenFlow bloquant basé sur un identifiant de vlan

5.2 Création d'un flux basé sur les identifiants de VLAN

On peut réaliser un certain nombre d'action sur les switch. Nous avons essentiellement étudié la partie blocage mais on peut aussi dupliqué le trafic.

La règle suivante permet de dupliquer le trafic lorsque l'on matche sur ces deux adresses MAC.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<flow xmlns="urn:opendaylight:flow:inventory">
    <strict>false</strict>
    <flow-name>FooXf108</flow-name>
    <id>101</id>
    <cookie_mask>255</cookie_mask>
    <cookie>108</cookie>
    <table_id>0</table_id>
    <priority>200</priority>
    <hard-timeout>1200</hard-timeout>
    <idle-timeout>3400</idle-timeout>
    <installHw>false</installHw>
    <instructions>
        <instruction>
            <order>0</order>
            <apply-actions>
                <action>
                    <order>0</order>
                    <output-action>
                        <output-node-connector>FLOOD</output-node-connector>
                        <max-length>60</max-length>
                    </output-action>
                </action>
            </apply-actions>
```

```
        </instruction>
    </instructions>
    <match>
        <ethernet-match>
            <ethernet-type>
                <type>2048</type>
            </ethernet-type>
            <ethernet-destination>
                <address>00:00:00:00:00:02</address>
            </ethernet-destination>
            <ethernet-source>
                <address>00:00:00:00:00:04</address>
            </ethernet-source>
        </ethernet-match>
    </match>
</flow>
```

Listing 27 – OpenFlow broadcastant le trafic

Appliquez cette règle sur le switch OpenFlow2.

Pour vérifier que cela fonctionne, il est nécessaire de réaliser les actions suivantes dans mininet :

```
mn> h4 ping h2 &
mn> h3 tcpdump -v
```

Listing 28 – OpenFlow bloquant un ensemble de critères

Vous devriez voir le trafic ICMP entre les deux hôtes, mais uniquement dans un sens.

5.3 Autres fonctionnalités

Il est aussi possible de rediriger vers une autre table pour que le traitement soit effectué en espace utilisateur et non plus en mode noyau. On peut privilégier ce gens de traitements dans les cas où la performance n'est pas le premier objectif.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
    <flow xmlns="urn:opendaylight:flow:inventory">
        <strict>false</strict>
        <instructions>
            <instruction>
                <order>0</order>
                <go-to-table>
                    <table_id>1</table_id>
                </go-to-table>
            </instruction>
        </instructions>
        <table_id>0</table_id>
        <id>1</id>
        <cookie_mask>10</cookie_mask>
        <installHw>false</installHw>
           <match>
            <ethernet-match>
                <ethernet-type>
                    <type>2048</type>
                </ethernet-type>
              <ethernet-destination>
                    <address>00:00:00:00:00:02</address>
                </ethernet-destination>
                <ethernet-source>
                    <address>00:00:00:00:00:01</address>
                </ethernet-source>
```

```
            </ethernet-match>
        </match>
        <hard-timeout>1800</hard-timeout>
        <cookie>10</cookie>
        <idle-timeout>1800</idle-timeout>
        <flow-name>flow-instruction-go-to-table</flow-name>
        <priority>2000</priority>
        <barrier>false</barrier>
    </flow>


----

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<flow xmlns="urn:opendaylight:flow:inventory">
    <strict>false</strict>
    <instructions>
        <instruction>
            <order>0</order>
            <apply-actions>
                <action>
                    <order>0</order>
                    <drop-action/>
                </action>
            </apply-actions>
        </instruction>
    </instructions>
    <table_id>1</table_id>
    <id>1</id>
    <cookie_mask>255</cookie_mask>
    <installHw>false</installHw>
    <match>
        <ethernet-match>
                <ethernet-type>
                                <type>2048</type>
                </ethernet-type>
            <ethernet-source>
                <address>00:00:00:00:00:01</address>
            </ethernet-source>
            <ethernet-destination>
                <address>00:00:00:00:00:02</address>
            </ethernet-destination>
        </ethernet-match>
    </match>
    <hard-timeout>1200</hard-timeout>
    <cookie>3</cookie>
    <idle-timeout>3400</idle-timeout>
    <flow-name>Block-MAC-ODL-2</flow-name>
    <priority>2000</priority>
    <barrier>false</barrier>
</flow>
```

Listing 29 – OpenFlow redirection vers une table spécifique