PROJET 17

- ## Notre démarche : de la recherche à la réalisation

Après avoir effectué des recherches sur les groupes de discussion Google Onos et sur les chaînes Slack Onos, il nous apparaissait plus évident de **développer une intent** plutôt qu'une application Java (malgré la présence d'un tutoriel sur le wiki).

En effet, nous avons une faible pratique de la programmation Java.
Pourtant, il s'est avéré assez délicat de définir précisément ce qu'était une intent. Nos recherches nous ont amené à trouver d'autres technologies qui pouvaient nous faire parvenir à nos fins.

Les **mots-clés** qui faisaient écho grâce à nos connaissances étaient :
- **QoS**, pour traduire la création de bande passante réservée en pourcentage pour les 3 chemins
- **Load balancing**, pour la même raison que pour la QoS en supposant que le load balancing pourrait être paramétrable.
- **File d'attente ou queue**, nécessaire lorsque l'on parle de QoS
- **MPLS**, technologie simple pour créer les chemins que les flux vont emprunter
- **Mininet**, pour dresser une topologie
- **OVS** pour dresser une topologie
- **Python** pour créer des scripts déployables dans Mininet

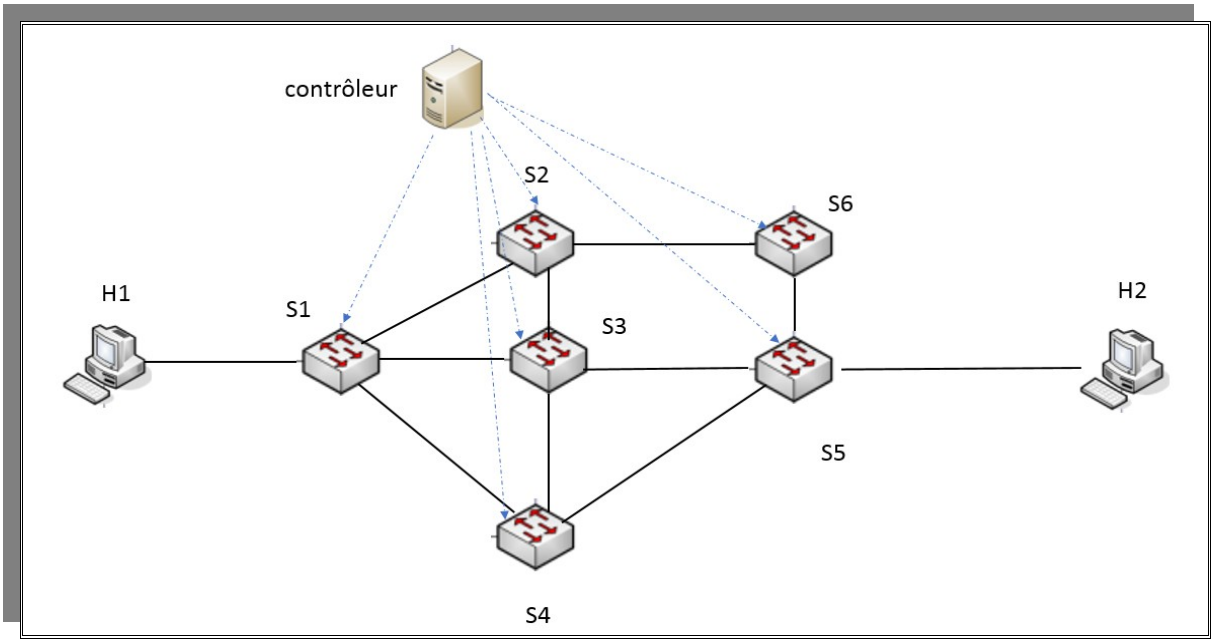
1) La première piste était de créer la topologie avec mininet et Spring-Open

En continuant nos recherches dans les archives des différents projets liés à Onos, un projet a particulièrement retenu notre attention : **Spring-Open**, qui utilisait le ***segment routing***. Ce framework d'Onos inclut quelques packages d'applications permettant aisément de créer des tunnels dans une topologie de switchs, par lesquels les flux de données peuvent passer.

JJanvier 2020

Parallèlement à cela, des fragments de code pour créer des files d'attente et de la **QoS** dans un environnement créé par **OVS**, ainsi que des **namespaces** furent trouvés.

La topologie reposant sur segment routing avec Spring-Open



Difficulté rencontrée :

Spring-Open est un projet opérationnel, utilisable, mais a déjà presque 3 ans d'existence et n'est plus suivi ou maintenu. Aussi, une fois que la configuration pour les 3 tunnels fut faite, **il nous a été impossible d'arriver à mixer les lignes de code pour appliquer la QoS**

```
mininet-vm(config)# tunnel TENPERCENT
mininet-vm(config-tunnel)# node
101  102  103  104  105  106  107  108  109
mininet-vm(config-tunnel)# node 10
101  102  103  104  105  106  107  108  109
mininet-vm(config-tunnel)# node 101
mininet-vm(config-tunnel)# node 102
mininet-vm(config-tunnel)# node 103
mininet-vm(config-tunnel)# node 104
mininet-vm(config-tunnel)# exit
mininet-vm(config)# show tunnel
# Id         Policies Tunnel Path [Head-->Tail] Label Stack [Outer-->Inner]
-|----------|--------|--------------------------|---------------------------
1 TENPERCENT          [101, 102, 103, 104]      [[103, 104]]
mininet-vm(config)#
```

Création de 3 tunnels pour appliquer un custom-bandwith avec la QoS

2) OpenVswitch

Malgré ce premier échec à appliquer la QoS afin de définir 3 débits de trafic-bandwith entre le host 1 et le host 2, nous avons explorés d'autres pistes.
Le TP SDN nous avait montré la simplicité d'utilisation d'OpenVSwitch et des namespace linux.

```
#!/bin/bash
set +xe

# 1. Create 2 namespaces(Simulated clients)
ip netns add h1
ip netns add h2

# 2. Create 5 openvswitches
sudo ovs-vsctl add-br s1
sudo ovs-vsctl add-br s2
sudo ovs-vsctl add-br s3
sudo ovs-vsctl add-br s4
sudo ovs-vsctl add-br s5
sudo ovs-vsctl add-br s6

# 3. creation of link and ports necessary for QOS and Queue
ip link add h1-eth1 type veth peer name s1-eth1
ip link add s1-s2 type veth peer name s2-s1
ip link add s1-s3 type veth peer name s3-s1
ip link add s1-s4 type veth peer name s4-s1
ip link set h1-eth1 netns h1
sudo ovs-vsctl add-port s1 s1-eth1
sudo ovs-vsctl add-port s1 s1-s2
sudo ovs-vsctl add-port s1 s1-s3
sudo ovs-vsctl add-port s1 s1-s4

# 4. creation of qos and queue
sudo ovs-vsctl set interface s1-s2 ofport_request=5 -- set interface s1-s3 ofport_request=6 -- set interface s1-s4 ofport_request=7 -- set port s1-eth1 qos=@newqos -- --id=@newqos create qos typ
e=linux-htb other-config:max-rate=10000000 queues:123=@s1-s2queue queues:234=@s1-s3queue queues:345=@s1-s4queue -- --id=@s1-s2queue create queue other-config:max-rate=1000000 --  --id=@s1-s3queu
e create queue other-config:max-rate=3000000 -- --id=@s1-s4queue create queue other-config:max-rate=6000000

# 5. To direct packets from the port to the queues reserved for them
sudo ovs-ofctl add-flow s1 in_port=5,actions=set_queue:123,normal
sudo ovs-ofctl add-flow s1 in_port=6,actions=set_queue:234,normal
sudo ovs-ofctl add-flow s1 in_port=7,actions=set_queue:345,normal

# 3. Create missing vethernet links
ip link add h2-eth1 type veth peer name s5-eth1
ip link add s2-s3 type veth peer name s3-s2
ip link add s3-s4 type veth peer name s4-s3
ip link add s3-s5 type veth peer name s5-s3
ip link add s4-s5 type veth peer name s5-s4
ip link add s2-s6 type veth peer name s6-s2
ip link add s6-s5 type veth peer name s5-s6
```

Difficulté rencontrée :

La partie Qos fonctionne et les paquets de données empruntent bien les 3 chemins. Seulement tout cela fonctionne sans Onos. Nous sommes donc hors sujet. En effet si l'on déclare pour les switchs que leur contrôleur est notre instance Onos (sur localhost par exemple), toutes les règles de

JJanvier 2020

direction des flux sont effacées et les instructions pour chaque flux entrant sur chaque port est de demander au contrôleur l'opération à suivre

3)

- Nous avons donc recommencé nos recherches **sur la création d'une app Java à compiler puis à injecter dans Onos.**
Cette fois, les recherches ont été plus fructueuses et nous avons trouvé un script .java qui semblait permettre la création de la QOS, des bridges, des queues, des ports…

Le tutoriel sur le wiki onos n'est pas très intuitif de prime abord, mais nous avons réussi à créer **une appli de type « hello world** », très basique, pour comprendre le principe de création d'app Java.
Pour autant, nous ne disposions pas de trame d'application déjà préparée que nous pourrions customiser.

Il n'existe de pas de tutoriel indiquant pas à pas ce qu'il faut faire pour customiser une application et l'intégrer dans Onos. Les différentes aides trouvées principalement sur le wiki onos nécessitent des prérequis qui ne sont pas apparents.

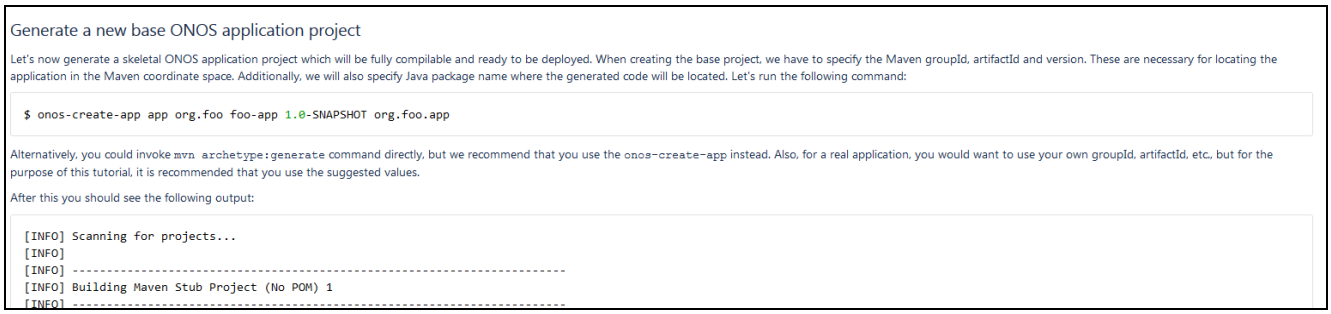
Nous voulions pallier ce manque dans ce rapport, en tâchant d'être le plus exhaustif possible. Voici le début de cette démarche :

- **Machine support** : Oracle VirtualBox a été choisi pour héberger la VM. Cette dernière est celle qui fut déployée pour le TP SDN. Est installé dessus Onos 2.0.0 et plusieurs JDK.

- Nous nous sommes servis de nos recherches sur internet pour créer notre **application Java**. Il faut maintenant distinguer le site sur lequel nous avons pu trouver nos fichiers sources (nous ne sommes pas partis de zéro en ce qui concerne l'écriture du code Java). C'est le **site GitHub** qui nous a le plus aidé. Pour la démarche permettant de compiler le code et le transformer en APP Onos, **le wikiOnos** fut notre support.
Pour customiser les fichiers sources, nous nous sommes juste basés sur **nos connaissances**.

- Nous avons dans un premier temps décidé de compiler tels quels les fichiers sources pour vérifier que toutes les étapes se déroulaient correctement.
La compilation se fait grâce au **logiciel Maven**. Citons qu'il existe d'autres logiciels permettant la construction d'objet java tels que Bazel et Buck. Mais Maven est le logiciel sur lequel nous avons trouvé le plus d'aide.
Il faut untar l'archive onos2.0.0 et se rendre à l'intérieur du répertoire créé.

Pour citer nos sources, voici le site que nous avons suivi.
https://wiki.onosproject.org/display/ONOS/Template+Application+Tutorial


Generate a new base ONOS application project

Let's now generate a skeletal ONOS application project which will be fully compilable and ready to be deployed. When creating the base project, we have to specify the Maven groupId, artifactId and version. These are necessary for locating the application in the Maven coordinate space. Additionally, we will also specify Java package name where the generated code will be located. Let's run the following command:

```
$ onos-create-app app org.foo foo-app 1.0-SNAPSHOT org.foo.app
```

Alternatively, you could invoke `mvn archetype:generate` command directly, but we recommend that you use the `onos-create-app` instead. Also, for a real application, you would want to use your own groupId, artifactId, etc., but for the purpose of this tutorial, it is recommended that you use the suggested values.

After this you should see the following output:

```
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building Maven Stub Project (No POM) 1
[INFO] ------------------------------------------------------------------------
```

Nous allons commenter ce tutoriel lorsqu'une difficulté a été rencontrée.
Nous avons procédé avec le fichier *OvsdbClientService.java* trouvé sur GitHub mais sans y apporter de changements particuliers.

La procédure indique de lancer la commande *onos-create-app*. Ceci permet de démarrer le projet de création d'app Onos
Seulement ce fichier n'est pas présent dans le répertoire. Nous l'avons trouvé sur GitHub, copié dans le répertoire Onos et rendu exécutable. A la fin d'exécution du script, il y a une personnalisation du projet rendue possible. Un répertoire du nom choisi est ainsi créé. Il faut se déplacer dedans et faire une commande pour visualiser l'intérieur. Il y a un fichier *pom.xml* et un répertoire src.

- Le fichier *pom.xml* permet de personnaliser la compilation (notamment en spécifiant les dépendances nécessaire à la bonne compilation de l'app.). Dans *src* se trouve …/main/java puis autant de

répertoires que le nom qui a été donné au projet. Dans notre exemple, *gr17*

```
student@student-VirtualBox:~/onos-2.0.0/org.group17/src/main/java/gr17$
```

C'est à cet endroit que les fichiers *.java doivent être copiés.

Dans le répertoire racine du projet, ici org.group17, il faut exécuter la commande sudo **mvn clean install**

- La compilation ne s'est pas bien déroulée. Le message d'erreur indique que le jdk java n'est pas conforme.
Il faut donc vérifier la version java utilisée en visualisant la variable JAVA_HOME avec la commande *sudo printenv* . Le message d'erreur n'indique pas quelle version est requise.
Dans le répertoire /usr/lib/jvm, figurent les différentes jdk présents sur la machine.

```
student@student-VirtualBox:/usr/lib/jvm$ ls
java-1.12.0-openjdk-amd64  java-12-openjdk-amd64  java-1.8.0-openjdk-amd64  jdk1.8.0_221
java-1.13.0-openjdk-amd64  java-13-openjdk-amd64  java-8-openjdk-amd64      openjdk-12
student@student-VirtualBox:/usr/lib/jvm$
```

```
JAVA_HOME=/usr/lib/jvm/java-13-openjdk-amd64
student@student-VirtualBox:/usr/lib/jvm$
```

La version 13 fut sélectionnée et la compilation put avancer.
Le répertoire target est créé et le fichier .oar y est créé.

```
student@student-VirtualBox:~/onos-2.0.0/org.group17/target$ ls
classes            generated-test-sources  maven-status  org.group17-1.0-SNAPSHOT.jar  org.group17-1.0-SNAPSHOT-tests.jar  test-classes
generated-sources  maven-archiver          oar           org.group17-1.0-SNAPSHOT.oar  surefire-reports
student@student-VirtualBox:~/onos-2.0.0/org.group17/target$
```

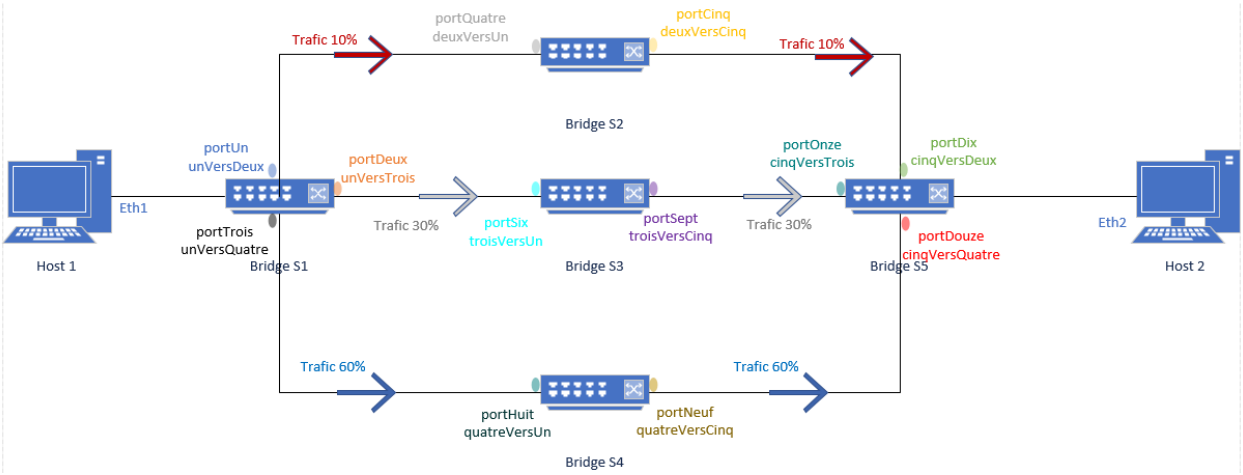
Ici, org.group17-1-0-SNAPSHOT.oar

Il faut ensuite executer la commande *onos-app.* Fichier trouvé sur GitHub. Un nouveau message d'erreur se présente. Il manquait 2 fichiers « *find-node* et *rest-port* ». Il a fallu les chercher encore une fois sur GitHub, les copier dans le répertoire racine du projet aux côtés du fichier pom.xml et relancer *onos-app.*
Onos-app créé l'app dans Onos. Il ne reste plus qu'a l'activer ou pas.

Cette étape nous a permis de valider les étapes nécessaires à la réalisation d'une app d'un fichier source (sans nos arguments !).

Nous nous sommes donc ensuite axé sur la bonne compréhension du fichier source nommé *OvsdbClientService.java*
Il est possible de placer plusieurs fichiers java dans le même répertoire dans lequel **Maven** va compiler .
Le fichier cité ci-dessus sera le fichier principal. Nous allons également utiliser les fichiers *OvsdbPort.java* et *OvsdbBridge.java* . Comme leur nom

l'indique, ils permettront de créer les ports ainsi que les commutateurs de notre topologie dont voici un schéma :



Après plusieurs heures à préparer les fichiers java et lorsque nous pensions qu'ils étaient prêts, de nombreuses erreurs sont apparues lors de la compilation.

Malheureusement il semble que les fichiers sources découverts sur GitHub ne suffisent pas à produire une topologie de switchs, ports et de liens. Les lettres DB dans OVSDB nous avaient un peu alertés.

L'écriture des fichiers .java fut néanmoins une expérience intéressante. Nous allons fournir ces fichiers en annexe. Les erreurs renvoyées par le compilateur permettent de penser que nous n'allons pas avoir le temps nécessaire pour développer la partie database.

4) Tentative avec un contrôleur Onos déjà configuré reposant sur Docker

Nous avons tenté une dernière piste ; Dans une topologie de 4 switchs virtuels créés grâce aux diverses commandes ovs-vsctl (script fourni) et avec un contrôleur Onos obtenu via Docker ou sur le localhost.

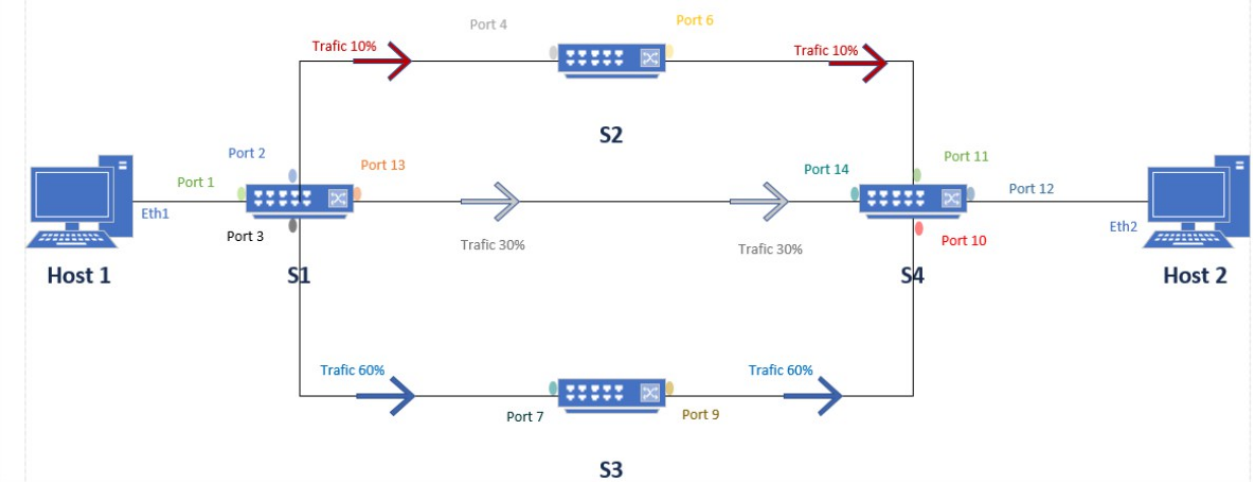
https://wiki.onosproject.org/display/ONOS/Single+Instance+Docker+deployment

Une fois la topologie créée, les switchs sont paramétrés pour avoir comme contrôleur l'instance docker hébergeant Onos. La commande add-single-to-multi-intent permet de diriger les flux.

Nous pouvons utiliser les options suivantes :

add-single-to-multi-intent ingressDevice /port egressDevice/port : permet de spécifier le switch et le port d'entrée d'un flux et le switch et port de sortie.

Voici le script et le schéma :



Script mininet :

```
from mininet.topo import Topo
import sys
import os

class MyTopo( Topo ):
   "Simple topology example."

   def __init__( self ):
      "Create custom topo."

      # Initialize topology
      Topo.__init__( self )
```

Janvier 2020

```
        # Add hosts and switches
        host1 = self.addHost( 'h1' )


       host2 = self.addHost( 'h2' )

        s1 = self.addSwitch('s1')
        s2 = self.addSwitch('s2')
        s3 = self.addSwitch('s3')
        s4 = self.addSwitch('s4')



        # Add links
        self.addLink( host1, s1)
        self.addLink( host2, s4 )
        self.addLink( s1, s2, cls=TCLink,bw=10 )
        self.addLink( s1, s3, cls=TCLink,bw=60)
        self.addLink( s1, s4, cls=TCLink,bw=30  )
        self.addLink( s3, s4, cls=TCLink,bw=60  )
        self.addLink( s2, s4, cls=TCLink,bw=10  )

topos = { 'mytopo': ( lambda: MyTopo() ) }
```

(cette topologie peut également être créée directement avec les commandes ovs-vsctl vues dans le TP)

Commandes dans Onos :

Onos : add-single-to-multi -b 10 S1id/1 S1id/2

Onos : add-single-to-multi -b 10 S2id/4 S2id/6

Onos : add-single-to-multi -b 10 S4id/11 S4id/12

Onos : add-single-to-multi -b 30 S1id/1 S1id/13

Onos : add-single-to-multi -b 30 S4id/14 S4id/12

Onos : add-single-to-multi -b 60 S1id/1 S1id/3

Onos : add-single-to-multi -b 60 S3id/7 S3id/9

Onos : add-single-to-multi -b 60 S4id/10 S4id/1

Il y a peut-être une double utilisation de la fonction bandwidth. Nous l'avons placée dans le script Mininet et avec l'option –b XX avec les commandes Onos. Nous n'avons pas pu tester cette fonctionnalité.

5. Avancement du projet suite aux remarques lors de l'oral de présentation

Environnement :

- Onos Tutorial

- 4Go

Janvier 2020
- Ubuntu 64 bits

Nous avons pu télécharger la VM disponible sur le wiki Onos dans une section tutorial. Ainsi il a été possible d'accéder à la console Onos avec facilité et sans « plantage » dû à un ordinateur n'étant pas assez puissant.

Pour construire la topologie, nous avons repris le script OpenVswitch de la 2ème tentative :

```
#!/bin/bash
set +xe
# 1. Create 2 namespaces(Simulated clients)
sudo ip netns add h1
sudo ip netns add h2
# 2. Create 5 openvswitches
sudo ovs-vsctl add-br s1
sudo ovs-vsctl add-br s2
sudo ovs-vsctl add-br s3
sudo ovs-vsctl add-br s4
sudo ovs-vsctl add-br s5
sudo ovs-vsctl add-br s6
# 3. creation of link and ports necessary for QOS and Queue
sudo ip link add h1-eth1 type veth peer name s1-eth1
sudo ip link add s1-s2 type veth peer name s2-s1
sudo ip link add s1-s3 type veth peer name s3-s1
sudo ip link add s1-s4 type veth peer name s4-s1
sudo ip link set h1-eth1 netns h1
sudo ovs-vsctl add-port s1 s1-eth1
sudo ovs-vsctl add-port s1 s1-s2
sudo ovs-vsctl add-port s1 s1-s3
```

Janvier 2020

```
sudo ovs-vsctl add-port s1 s1-s4

# 4. creation of qos and queue

sudo ovs-vsctl set interface s1-s2 ofport_request=5 -- set interface s1-s3 ofport_request=6 -- set interface s1-s4 ofport_request=7 -- set port s1-eth1 qos=@newqos -- --id=@newqos create qos type=linux-htb other-config:max-rate=10000000 queues:123=@s1-s2queue queues:234=@s1-s3queue queues:345=@s1-s4queue -- --id=@s1-s2queue create queue other-config:max-rate=1000000 --  --id=@s1-s3queue create queue other-config:max-rate=3000000 -- --id=@s1-s4queue create queue other-config:max-rate=6000000

# 5. To direct packets from the port to the queues reserved for them

sudo ovs-ofctl add-flow s1 in_port=5,actions=set_queue:123,normal

sudo ovs-ofctl add-flow s1 in_port=6,actions=set_queue:234,normal

sudo ovs-ofctl add-flow s1 in_port=7,actions=set_queue:345,normal



# 3. Create missing vethernet links

sudo ip link add h2-eth1 type veth peer name s5-eth1

sudo ip link add s2-s3 type veth peer name s3-s2

sudo ip link add s3-s4 type veth peer name s4-s3

sudo ip link add s3-s5 type veth peer name s5-s3

sudo ip link add s4-s5 type veth peer name s5-s4

sudo ip link add s2-s6 type veth peer name s6-s2

sudo ip link add s6-s5 type veth peer name s5-s6

# 4. Move host ports into namespaces

sudo ip link set h2-eth1 netns h2

# 5. Connect switch ports to OVS

sudo ovs-vsctl add-port s5 s5-eth1 # h2-eth1

sudo ovs-vsctl add-port s5 s5-s3

sudo ovs-vsctl add-port s5 s5-s4

sudo ovs-vsctl add-port s5 s5-s6

sudo ovs-vsctl add-port s2 s2-s3

sudo ovs-vsctl add-port s2 s2-s1

sudo ovs-vsctl add-port s2 s2-s6

sudo ovs-vsctl add-port s3 s3-s4

sudo ovs-vsctl add-port s3 s3-s1
```

```
sudo ovs-vsctl add-port s3 s3-s2

sudo ovs-vsctl add-port s3 s3-s5

sudo ovs-vsctl add-port s4 s4-s1

sudo ovs-vsctl add-port s4 s4-s3

sudo ovs-vsctl add-port s4 s4-s5

sudo ovs-vsctl add-port s6 s6-s2

sudo ovs-vsctl add-port s6 s6-s5

# 7. Setup ip

sudo ip netns exec h1 ifconfig h1-eth1 10.1



sudo ip netns exec h1 ifconfig lo up

sudo ip netns exec h2 ifconfig h2-eth1 10.2

sudo ip netns exec h2 ifconfig lo up

# 8. activate ports

sudo ifconfig s1-eth1 up

sudo ifconfig s5-eth1 up

sudo ifconfig s1-s2 up

sudo ifconfig s1-s3 up

sudo ifconfig s1-s4 up

sudo ifconfig s5-s3 up

sudo ifconfig s5-s4 up

sudo ifconfig s5-s6 up

sudo ifconfig s2-s1 up

sudo ifconfig s2-s6 up

sudo ifconfig s3-s1 up

sudo ifconfig s3-s5 up

sudo ifconfig s4-s1 up

sudo ifconfig s4-s5 up

sudo ifconfig s6-s5 up
```

JJanvier 2020

```
sudo ifconfig s6-s2 up



# 9. setting the 3 paths in the direction of host2

sudo ovs-ofctl add-flow s2 in_port=2,actions=output:3

sudo ovs-ofctl add-flow s6 in_port=1,actions=output:2

sudo ovs-ofctl add-flow s5 in_port=4,actions=output:1

sudo ovs-ofctl add-flow s3 in_port=2,actions=output:4

sudo ovs-ofctl add-flow s5 in_port=2,actions=output:1

sudo ovs-ofctl add-flow s4 in_port=1,actions=output:3

sudo ovs-ofctl add-flow s5 in_port=3,actions=output:1
```

Il y a des commandes effacées par rapport à la tentative 2, notamment concernant la direction des flux car nous allons le commander depuis la console Onos et pas grâce à OpenVswitch.

Il faut ensuite configurer les switchs pour être controlé par Onos :

Sudo ovs-vsctl set-controller s1 tcp:172.17.0.5

La commande que nous allons utiliser afin de lier les interfaces créées avec OpenVswitch à des ports OpenFlow est **interface-add :**

```
onos> interface-add --help
DESCRIPTION
        onos:interface-add

        Adds a new configured interface

SYNTAX
        onos:interface-add [options] port name

ARGUMENTS
        port
                Device port that the interface is associated with
        name
                Interface name

OPTIONS
        -i, --ip
                IP address configured on the interface
                (e.g. 10.0.1.1/24). Can be specified multiple times.
        -j, --json
                Output JSON
        -m, --mac
                MAC address of the interface
        -v, --vlan
                VLAN configured on the interface
        --help
                Display this help message

onos>
```

Pour compartimenter un peu plus, nous avons créé également 3 vlans pour les 3 chemins qui doivent être empruntés

```
onos> interface-add -v 100 of:0000000000000001/2 s1-s2
Interface added
onos> interface-add -v 100 of:0000000000000002/1 s2-s1
Interface added
onos> interface-add -v 100 of:0000000000000002/2 s2-s6
Interface added
onos> interface-add -v 100 of:0000000000000006/1 s6-s2
Interface added
onos> interface-add -v 100 of:0000000000000006/2 s6-s5
Interface added
onos> interface-add -v 100 of:0000000000000005/1 s5-s6
Interface added
onos> interfaces
s1-s2: port=of:0000000000000001/2 vlan=100
s2-s1: port=of:0000000000000002/1 vlan=100
s2-s6: port=of:0000000000000002/2 vlan=100
s5-s6: port=of:0000000000000005/1 vlan=100
s6-s2: port=of:0000000000000006/1 vlan=100
s6-s5: port=of:0000000000000006/2 vlan=100
onos>
```

Les ports s1-s2, s2-s1…sont ceux créés sous OpenVswitch. Toutes les interfaces OF créées ici sont celles permettant le trajet du flux de 10% (et vlan 100).

La commande Onos « interfaces », permet de vérifier la création et la correspondance.

Même travail pour le trajet de 30% :

```
onos> interface-add -v 300 of:0000000000000001/3 s1-s3
Interface added
onos> interface-add -v 300 of:0000000000000003/1 s3-s1
Interface added
onos> interface-add -v 300 of:0000000000000003/2 s3-s5
Interface added
onos> interface-add -v 300 of:0000000000000005/2 s5-s3
Interface added
onos> interfaces
s1-s2: port=of:0000000000000001/2 vlan=100
s1-s3: port=of:0000000000000001/3 vlan=300
s2-s1: port=of:0000000000000002/1 vlan=100
s2-s6: port=of:0000000000000002/2 vlan=100
s3-s1: port=of:0000000000000003/1 vlan=300
s3-s5: port=of:0000000000000003/2 vlan=300
s5-s6: port=of:0000000000000005/1 vlan=100
s5-s3: port=of:0000000000000005/2 vlan=300
s6-s2: port=of:0000000000000006/1 vlan=100
s6-s5: port=of:0000000000000006/2 vlan=100
onos>
```

Et pour celui à 60% :

```
onos> interface-add -v 600 of:0000000000000001/4 s1-s4
Interface added
onos> interface-add -v 600 of:0000000000000004/1 s4-s1
Interface added
onos> interface-add -v 600 of:0000000000000004/2 s4-s5
Interface added
onos> interface-add -v 600 of:0000000000000005/3 s5-s4
Interface added
onos> interfaces
s1-s2: port=of:0000000000000001/2 vlan=100
s1-s3: port=of:0000000000000001/3 vlan=300
s1-s4: port=of:0000000000000001/4 vlan=600
s2-s1: port=of:0000000000000002/1 vlan=100
s2-s6: port=of:0000000000000002/2 vlan=100
s3-s1: port=of:0000000000000003/1 vlan=300
s3-s5: port=of:0000000000000003/2 vlan=300
s4-s1: port=of:0000000000000004/1 vlan=600
s4-s5: port=of:0000000000000004/2 vlan=600
s5-s6: port=of:0000000000000005/1 vlan=100
s5-s3: port=of:0000000000000005/2 vlan=300
s5-s4: port=of:0000000000000005/3 vlan=600
s6-s2: port=of:0000000000000006/1 vlan=100
s6-s5: port=of:0000000000000006/2 vlan=100
onos>
```

Définition du port OF 0005/4 comme port reliant host 2 au switch 5 :

```
onos> interface-add of:0000000000000005/4 s5-eth1
Interface added
onos>
```

Ajout de l'interface sur s1 sur laquelle est connectée le host 1 et division des flux arrivant de ce port sur les 3 ports de sortie vers les switchs 2, 3 et 4 :

```
onos> interface-add of:0000000000000001/1 s1-eth1
Interface added
onos> add-single-to-multi-intent 0000000000000001/1 0000000000000001/2 0000000000000001/3 0000000000000001/4
Single point to multipoint intent submitted:
SinglePointToMultiPointIntent{id=0x3, key=0x3, appId=DefaultApplicationId{id=3, name=org.onosproject.cli}, priority=100, resources=[], s
ltTrafficSelector{criteria=[]}, treatment=DefaultTrafficTreatment{immediate=[NOACTION], deferred=[], transition=None, meter=[], cleared=
igger=null, metadata=null}, ingress=FilteredConnectPoint{connectPoint=0000000000000001/1, trafficSelector=DefaultTrafficSelector{criteri
s=[FilteredConnectPoint{connectPoint=0000000000000001/3, trafficSelector=DefaultTrafficSelector{criteria=[]}}, FilteredConnectPoint{conn
000000000001/2, trafficSelector=DefaultTrafficSelector{criteria=[]}}, FilteredConnectPoint{connectPoint=0000000000000001/4, trafficSelec
afficSelector{criteria=[]}}], filteredIngressCPs=FilteredConnectPoint{connectPoint=0000000000000001/1, trafficSelector=DefaultTrafficSel
a=[]}}, filteredEgressCP=[FilteredConnectPoint{connectPoint=0000000000000001/3, trafficSelector=DefaultTrafficSelector{criteria=[]}}, Fi
Point{connectPoint=0000000000000001/2, trafficSelector=DefaultTrafficSelector{criteria=[]}}, FilteredConnectPoint{connectPoint=000000000
afficSelector=DefaultTrafficSelector{criteria=[]}}], constraints=[], resourceGroup=null}
onos>
```

0000000000000001 est le switch s1. Le /1 signifie port 1 du switch.

Sur le switch de sortie, s5, regroupement des 3 flux sur un seul en direction du host 2

```
onos> interface-add of:0000000000000001/1 s1-eth1
Interface added
onos> add-single-to-multi-intent 0000000000000001/1 0000000000000001/2 0000000000000001/3 0000000000000001/4
Single point to multipoint intent submitted:
SinglePointToMultiPointIntent{id=0x3, key=0x3, appId=DefaultApplicationId{id=3, name=org.onosproject.cli}, priority=100, resources=[], s
ltTrafficSelector{criteria=[]}, treatment=DefaultTrafficTreatment{immediate=[NOACTION], deferred=[], transition=None, meter=[], cleared=
igger=null, metadata=null}, ingress=FilteredConnectPoint{connectPoint=0000000000000001/1, trafficSelector=DefaultTrafficSelector{criteri
s=[FilteredConnectPoint{connectPoint=0000000000000001/3, trafficSelector=DefaultTrafficSelector{criteria=[]}}, FilteredConnectPoint{conn
000000000001/2, trafficSelector=DefaultTrafficSelector{criteria=[]}}, FilteredConnectPoint{connectPoint=0000000000000001/4, trafficSelec
afficSelector{criteria=[]}}], filteredIngressCPs=FilteredConnectPoint{connectPoint=0000000000000001/1, trafficSelector=DefaultTrafficSel
a=[]}}, filteredEgressCP=[FilteredConnectPoint{connectPoint=0000000000000001/3, trafficSelector=DefaultTrafficSelector{criteria=[]}}, Fi
Point{connectPoint=0000000000000001/2, trafficSelector=DefaultTrafficSelector{criteria=[]}}, FilteredConnectPoint{connectPoint=000000000
afficSelector=DefaultTrafficSelector{criteria=[]}}], constraints=[], resourceGroup=null}
onos>
```

Ajout de l'intent bande passante du port menant s1 à s2

```
onos> add-single-to-multi-intent -b 10 s1-s2 s2-s1
Error executing command: Connect point must be in "deviceUri/portNumber" format
onos> add-single-to-multi-intent -b 10 0000000000000001/2 0000000000000002/1
Single point to multipoint intent submitted:
SinglePointToMultiPointIntent{id=0x0, key=0x0, appId=DefaultApplicationId{id=3,
name=org.onosproject.cli}, priority=100, resources=[], selector=DefaultTrafficSe
lector{criteria=[]}, treatment=DefaultTrafficTreatment{immediate=[NOACTION], def
erred=[], transition=None, meter=[], cleared=false, StatTrigger=null, metadata=n
ull}, ingress=FilteredConnectPoint{connectPoint=0000000000000001/2, trafficSelec
tor=DefaultTrafficSelector{criteria=[]}}, egress=[FilteredConnectPoint{connectPo
int=0000000000000002/1, trafficSelector=DefaultTrafficSelector{criteria=[]}}], f
ilteredIngressCPs=FilteredConnectPoint{connectPoint=0000000000000001/2, trafficS
elector=DefaultTrafficSelector{criteria=[]}}, filteredEgressCP=[FilteredConnectP
oint{connectPoint=0000000000000002/1, trafficSelector=DefaultTrafficSelector{cri
teria=[]}}], constraints=[BandwidthConstraint{bandwidth=10}], resourceGroup=null
}
onos>
onos>
```

Ajout des 2 intents bande passante du port menant s1 à s3 et s1 à s4

```
onos> add-single-to-multi-intent -b 30 0000000000000001/3 0000000000000003/1
Single point to multipoint intent submitted:
SinglePointToMultiPointIntent{id=0x1, key=0x1, appId=DefaultApplicationId{id=3, name=org.onosproject.cli}, priority=100, resource
s=[], selector=DefaultTrafficSelector{criteria=[]}, treatment=DefaultTrafficTreatment{immediate=[NOACTION], deferred=[], transiti
on=None, meter=[], cleared=false, StatTrigger=null, metadata=null}, ingress=FilteredConnectPoint{connectPoint=0000000000000001/3,
 trafficSelector=DefaultTrafficSelector{criteria=[]}}, egress=[FilteredConnectPoint{connectPoint=0000000000000003/1, trafficSelec
tor=DefaultTrafficSelector{criteria=[]}}], filteredIngressCPs=FilteredConnectPoint{connectPoint=0000000000000001/3, trafficSelect
or=DefaultTrafficSelector{criteria=[]}}, filteredEgressCP=[FilteredConnectPoint{connectPoint=0000000000000003/1, trafficSelector=
DefaultTrafficSelector{criteria=[]}}], constraints=[BandwidthConstraint{bandwidth=30}], resourceGroup=null}
onos> add-single-to-multi-intent -b 60 0000000000000001/4 0000000000000004/1
Single point to multipoint intent submitted:
SinglePointToMultiPointIntent{id=0x2, key=0x2, appId=DefaultApplicationId{id=3, name=org.onosproject.cli}, priority=100, resource
s=[], selector=DefaultTrafficSelector{criteria=[]}, treatment=DefaultTrafficTreatment{immediate=[NOACTION], deferred=[], transiti
on=None, meter=[], cleared=false, StatTrigger=null, metadata=null}, ingress=FilteredConnectPoint{connectPoint=0000000000000001/4,
 trafficSelector=DefaultTrafficSelector{criteria=[]}}, egress=[FilteredConnectPoint{connectPoint=0000000000000004/1, trafficSelec
tor=DefaultTrafficSelector{criteria=[]}}], filteredIngressCPs=FilteredConnectPoint{connectPoint=0000000000000001/4, trafficSelect
or=DefaultTrafficSelector{criteria=[]}}, filteredEgressCP=[FilteredConnectPoint{connectPoint=0000000000000004/1, trafficSelector=
DefaultTrafficSelector{criteria=[]}}], constraints=[BandwidthConstraint{bandwidth=60}], resourceGroup=null}
onos>
```

Récapitulatif des commandes Onos :

interface-add -v 100 of:0000000000000001/2 s1-s2
interface-add -v 100 of:0000000000000002/1 s2-s1
interface-add -v 100 of:0000000000000002/2 s2-s6
interface-add -v 100 of:0000000000000006/1 s6-s2
interface-add -v 100 of:0000000000000006/2 s6-s5
interface-add -v 100 of:0000000000000001/2 s5-s6

interface-add -v 300 of:0000000000000001/3 s1-s3

interface-add -v 300 of:0000000000000003/1 s3-s1
interface-add -v 300 of:0000000000000003/2 s3-s5
interface-add -v 300 of:0000000000000005/2 s5-s3

interface-add -v 600 of:0000000000000001/4 s1-s4
interface-add -v 600 of:0000000000000004/1 s4-s1
interface-add -v 600 of:0000000000000004/2 s4-s5
interface-add -v 600 of:0000000000000005/3 s5-s4

interface-add of:0000000000000005/4 s5-eth1
interface-add of:0000000000000001/1 s1-eth1

JJanvier 2020

add-single-to-multi-intent 0000000000000001/1 0000000000000001/2 0000000000000001/3 0000000000000001/4

add-single-to-multi-intent -b 10 0000000000000001/2 0000000000000002/1

add-single-to-multi-intent -b 30 0000000000000001/3 0000000000000003/1

add-single-to-multi-intent -b 60 0000000000000001/4 0000000000000004/1

Etat des liens :

```
onos> links
src=of:0000823edd057945/5, dst=of:0000fab98e3ba848/2, type=DIRECT, state=ACTIVE, expected=false
src=of:0000823edd057945/6, dst=of:0000c29d1aaff046/2, type=DIRECT, state=ACTIVE, expected=false
src=of:0000823edd057945/7, dst=of:0000825713a9754b/1, type=DIRECT, state=ACTIVE, expected=false
src=of:0000825713a9754b/1, dst=of:0000823edd057945/7, type=DIRECT, state=ACTIVE, expected=false
src=of:0000825713a9754b/2, dst=of:0000c29d1aaff046/1, type=DIRECT, state=ACTIVE, expected=false
src=of:0000825713a9754b/3, dst=of:000092a910baff42/3, type=DIRECT, state=ACTIVE, expected=false
src=of:000092a910baff42/2, dst=of:0000c29d1aaff046/4, type=DIRECT, state=ACTIVE, expected=false
src=of:000092a910baff42/3, dst=of:0000825713a9754b/3, type=DIRECT, state=ACTIVE, expected=false
src=of:000092a910baff42/4, dst=of:0000fa9f7c628442/2, type=DIRECT, state=ACTIVE, expected=false
src=of:0000c29d1aaff046/1, dst=of:0000825713a9754b/2, type=DIRECT, state=ACTIVE, expected=false
src=of:0000c29d1aaff046/2, dst=of:0000823edd057945/6, type=DIRECT, state=ACTIVE, expected=false
src=of:0000c29d1aaff046/3, dst=of:0000fab98e3ba848/1, type=DIRECT, state=ACTIVE, expected=false
src=of:0000c29d1aaff046/4, dst=of:000092a910baff42/2, type=DIRECT, state=ACTIVE, expected=false
src=of:0000fa9f7c628442/1, dst=of:0000fab98e3ba848/3, type=DIRECT, state=ACTIVE, expected=false
src=of:0000fa9f7c628442/2, dst=of:000092a910baff42/4, type=DIRECT, state=ACTIVE, expected=false
src=of:0000fab98e3ba848/1, dst=of:0000c29d1aaff046/3, type=DIRECT, state=ACTIVE, expected=false
src=of:0000fab98e3ba848/2, dst=of:0000823edd057945/5, type=DIRECT, state=ACTIVE, expected=false
src=of:0000fab98e3ba848/3, dst=of:0000fa9f7c628442/1, type=DIRECT, state=ACTIVE, expected=false
onos>
```

Les hosts précédemment créés sous openvswitch

```
onos> hosts
id=22:2C:00:7F:12:D7/None, mac=22:2C:00:7F:12:D7, locations=[of:000092a910baff42/1], vlan=None, ip(s)=[10.0.0.2], innerVlan=None,
 outerTPID=unknown, provider=of:org.onosproject.provider.host, configured=false
id=2A:47:1C:01:57:DC/None, mac=2A:47:1C:01:57:DC, locations=[of:0000823edd057945/1], vlan=None, ip(s)=[10.0.0.1], innerVlan=None,
 outerTPID=unknown, provider=of:org.onosproject.provider.host, configured=false
```

Les ports :

```
onos> ports
id=of:0000823edd057945, available=true, local-status=connected 1h54m ago, role=MASTER, type=SWITCH, mfr=Nicira, Inc., hw=Open vSw
itch, sw=2.5.5, serial=None, chassis=823edd057945, driver=ovs, channelId=172.17.0.1:43186, managementAddress=172.17.0.1, protocol
=OF_13
  port=LOCAL, state=disabled, type=copper, speed=0 , adminState=disabled, portMac=82:3e:dd:05:79:45, portName=s1
  port=1, state=enabled, type=copper, speed=10000 , adminState=enabled, portMac=ca:eb:ec:59:f9:e9, portName=s1-eth1
  port=5, state=enabled, type=copper, speed=10000 , adminState=enabled, portMac=4e:4c:09:f0:b0:f6, portName=s1-s2
  port=6, state=enabled, type=copper, speed=10000 , adminState=enabled, portMac=86:71:02:64:a9:cd, portName=s1-s3
  port=7, state=enabled, type=copper, speed=10000 , adminState=enabled, portMac=1a:90:49:4a:d2:d2, portName=s1-s4
id=of:0000825713a9754b, available=true, local-status=connected 11m49s ago, role=MASTER, type=SWITCH, mfr=Nicira, Inc., hw=Open vS
witch, sw=2.5.5, serial=None, chassis=825713a9754b, driver=ovs, channelId=172.17.0.1:43202, managementAddress=172.17.0.1, protoco
l=OF_13
  port=LOCAL, state=disabled, type=copper, speed=0 , adminState=disabled, portMac=82:57:13:a9:75:4b, portName=s4
  port=1, state=enabled, type=copper, speed=10000 , adminState=enabled, portMac=be:2d:7c:53:93:1a, portName=s4-s1
  port=2, state=enabled, type=copper, speed=10000 , adminState=enabled, portMac=de:f9:2c:8e:85:e2, portName=s4-s3
  port=3, state=enabled, type=copper, speed=10000 , adminState=enabled, portMac=e6:7f:d0:06:1f:16, portName=s4-s5
id=of:000092a910baff42, available=true, local-status=connected 11m39s ago, role=MASTER, type=SWITCH, mfr=Nicira, Inc., hw=Open vS
witch, sw=2.5.5, serial=None, chassis=92a910baff42, driver=ovs, channelId=172.17.0.1:43204, managementAddress=172.17.0.1, protoco
l=OF_13
  port=LOCAL, state=disabled, type=copper, speed=0 , adminState=disabled, portMac=92:a9:10:ba:ff:42, portName=s5
  port=1, state=enabled, type=copper, speed=10000 , adminState=enabled, portMac=ba:9b:88:62:49:f2, portName=s5-eth1
  port=2, state=enabled, type=copper, speed=10000 , adminState=enabled, portMac=c6:08:97:3f:90:89, portName=s5-s3
  port=3, state=enabled, type=copper, speed=10000 , adminState=enabled, portMac=f2:98:a8:f6:68:06, portName=s5-s4
  port=4, state=enabled, type=copper, speed=10000 , adminState=enabled, portMac=c6:7d:04:7c:19:cd, portName=s5-s6
id=of:0000c29d1aaff046, available=true, local-status=connected 1h50m ago, role=MASTER, type=SWITCH, mfr=Nicira, Inc., hw=Open vSw
itch, sw=2.5.5, serial=None, chassis=c29d1aaff046, driver=ovs, channelId=172.17.0.1:43190, managementAddress=172.17.0.1, protocol
=OF_13
  port=LOCAL, state=disabled, type=copper, speed=0 , adminState=disabled, portMac=c2:9d:1a:af:f0:46, portName=s3
  port=1, state=enabled, type=copper, speed=10000 , adminState=enabled, portMac=ee:fd:5b:23:4c:0d, portName=s3-s4
  port=2, state=enabled, type=copper, speed=10000 , adminState=enabled, portMac=42:69:14:4d:d0:16, portName=s3-s1
  port=3, state=enabled, type=copper, speed=10000 , adminState=enabled, portMac=06:f8:55:6a:8d:6f, portName=s3-s2
  port=4, state=enabled, type=copper, speed=10000 , adminState=enabled, portMac=1a:f8:c0:20:97:88, portName=s3-s5
id=of:0000fa9f7c628442, available=true, local-status=connected 11m28s ago, role=MASTER, type=SWITCH, mfr=Nicira, Inc., hw=Open vS
witch, sw=2.5.5, serial=None, chassis=fa9f7c628442, driver=ovs, channelId=172.17.0.1:43206, managementAddress=172.17.0.1, protoco
l=OF_13
  port=LOCAL, state=disabled, type=copper, speed=0 , adminState=disabled, portMac=fa:9f:7c:62:84:42, portName=s6
  port=1, state=enabled, type=copper, speed=10000 , adminState=enabled, portMac=f2:18:aa:ce:e1:b3, portName=s6-s2
  port=2, state=enabled, type=copper, speed=10000 , adminState=enabled, portMac=06:d9:a6:50:39:78, portName=s6-s5
id=of:0000fab98e3ba848, available=true, local-status=connected 12m3s ago, role=MASTER, type=SWITCH, mfr=Nicira, Inc., hw=Open vSw
itch, sw=2.5.5, serial=None, chassis=fab98e3ba848, driver=ovs, channelId=172.17.0.1:43200, managementAddress=172.17.0.1, protocol
=OF_13
  port=LOCAL, state=disabled, type=copper, speed=0 , adminState=disabled, portMac=fa:b9:8e:3b:a8:48, portName=s2
  port=1, state=enabled, type=copper, speed=10000 , adminState=enabled, portMac=f2:9c:02:ea:4b:4f, portName=s2-s3
  port=2, state=enabled, type=copper, speed=10000 , adminState=enabled, portMac=76:7d:d3:84:7f:70, portName=s2-s1
  port=3, state=enabled, type=copper, speed=10000 , adminState=enabled, portMac=b2:ff:8b:3c:e3:9d, portName=s2-s6
onos>
```

Test de flux avec commande add-test-flows

```
..cleaning up
Run 146:
..batch add request
..completed 4425 ± 100 ms
..cleaning up
Run 147:
..batch add request
..completed 4413 ± 100 ms
..cleaning up
Run 148:
..batch add request
..completed 4525 ± 100 ms
..cleaning up
Run 149:
..batch add request
..completed 4438 ± 100 ms
..cleaning up
Run is success.
  Run 0 : 62 ms
  Run 1 : 48 ms
  Run 2 : 46 ms
  Run 3 : 43 ms
  Run 4 : 27 ms
  Run 5 : 52 ms
  Run 6 : 31 ms
  Run 7 : 34 ms
  Run 8 : 29 ms
  Run 9 : 29 ms
  Run 10 : 22 ms
  Run 11 : 24 ms
  Run 12 : 25 ms
  Run 13 : 25 ms
  Run 14 : 25 ms
  Run 15 : 16 ms
  Run 16 : 267 ms
  Run 17 : 20 ms
  Run 18 : 25 ms
  Run 19 : 17 ms
  Run 20 : 33 ms
  Run 21 : 29 ms
  Run 22 : 18 ms
  Run 23 : 24 ms
  Run 24 : 37 ms
```

Apres un test add-test-flow :

```
sdn@onos-tutorial:~/Downloads$ sudo ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x100003bdf6857, duration=10311.588s, table=0, n_packets=5898, n_bytes=460044, idle_age=0, priority=40000,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x100000c42ec3f, duration=10311.579s, table=0, n_packets=9699, n_bytes=756522, idle_age=0, priority=40000,dl_type=0x8942 actions=CONTROLLER:65535
 cookie=0x100001b2f5f26, duration=10311.372s, table=0, n_packets=72, n_bytes=3024, idle_age=7, priority=40000,arp actions=CONTROLLER:65535
 cookie=0x10000e5dd8485, duration=4023.771s, table=0, n_packets=2228, n_bytes=218344, idle_age=3, priority=5,ip actions=CONTROLLER:65535
```

Nous pouvons noter que les flux sont répartis différemment :

2228 packets pour le bandwidth à 10 ;

5898 pour le bandwidth à 30 ;

9699 pour le bandwidth à 60 ;

Malheureusement nous n'avons réussi cette mise en situation qu'une seule fois. Les tentatives suivantes, dès l'activation des interfaces dans le script openvswitch, la console Onos se met à ralentir, jusqu'à nous faire perdre la main. Il faut alors tuer le process.

Annexes

https://github.com/opennetworkinglab/onos/tree/master/protocols/ovsdb/api/src/main/java/org/onosproject/ovsdb/controller

https://github.com/opennetworkinglab/onos/blob/master/protocols/ovsdb/api/src/main/java/org/onosproject/ovsdb/controller/OvsdbClientService.java

https://github.com/opennetworkinglab/onos/blob/master/protocols/ovsdb/api/src/main/java/org/onosproject/ovsdb/controller/OvsdbBridge.java

https://github.com/opennetworkinglab/onos/blob/master/protocols/ovsdb/api/src/main/java/org/onosproject/ovsdb/controller/OvsdbPort.java

https://github.com/opennetworkinglab/onos/blob/master/protocols/ovsdb/api/src/main/java/org/onosproject/ovsdb/controller/OvsdbQueue.java

https://www.mvndoc.com/c/org.onosproject/onos-ovsdb-api/org/onosproject/ovsdb/controller/OvsdbClientService.html

https://github.com/opennetworkinglab/onos/tree/master/apps

https://wiki.onosproject.org/

https://wiki.onosproject.org/display/ONOS/Tutorials

https://wiki.onosproject.org/display/ONOS/Basic+ONOS+Tutorial

https://wiki.onosproject.org/display/ONOS/Appendix+A+%3A+CLI+commands

https://wiki.onosproject.org/pages/viewpage.action?pageId=2130908

https://wiki.onosproject.org/display/ONOS/Intent+Framework

http://csie.nqu.edu.tw/smallko/sdn/ingress_policing_queue.htm

https://www.programcreek.com/java-api-examples/?code=shlee89/athena/athena-master/protocols/ovsdb/api/src/main/java/org/onosproject/ovsdb/controller/OvsdbClientService.java

http://api.onosproject.org/1.13.2/org/onosproject/net/behaviour/QosDescription.html

JJanvier 2020

http://api.onosproject.org/1.13.1/org/onosproject/net/behaviour/QueueDescription.Builder.html

http://api.onosproject.org/1.13.1/org/onlab/util/Bandwidth.html

https://www.theseus.fi/bitstream/handle/10024/124568/Litmanen_Ilpo.pdf?sequence=1

http://docs.openvswitch.org/en/latest/faq/qos/