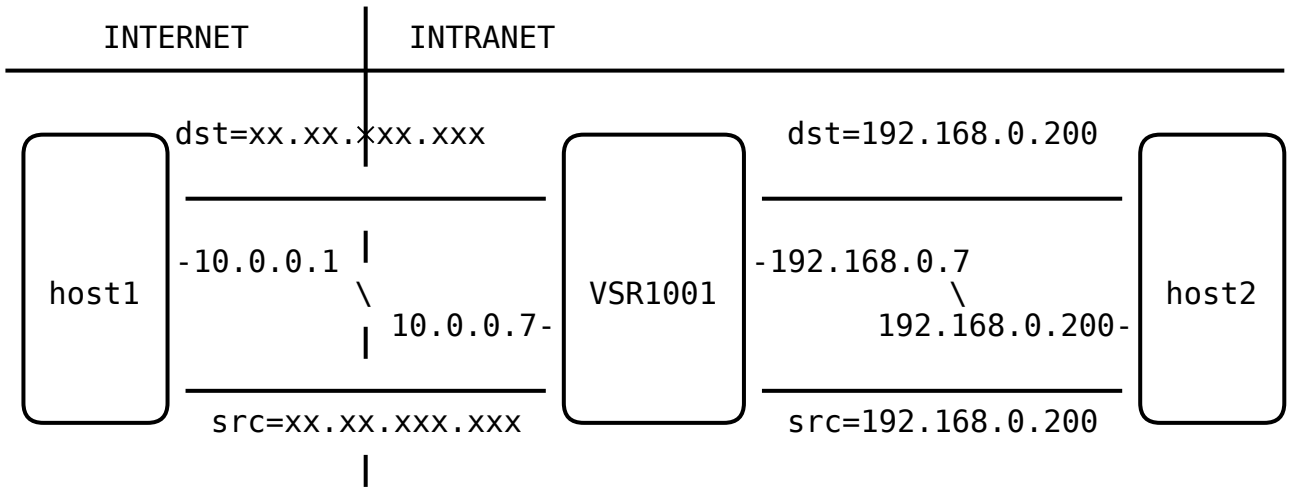
ODL: Lab 4 simple NAT openflow

Objet	NAT avec Opendaylight
Niveau requis	débutant, avisé
Features	odl-openflow-plugin, odl-overview
Suivi	:DONE:

Objectif

Ce didacticiel a pour objet de mettre en oeuvre un nattage simple avec OpenFlow pour que l'hôte 1 (ip adresse 1 10.0.0.1) qui est réputé être sur le NET puisse communiquer avec l'hôte 2 (disposant d'une ip non routable 192.168.0.200) avec une adresse fictive routable (xx.xx.xxx.xxx).



À la fin de ce didacticiel, on doit être familiarisé avec les éléments suivants:

- Openflow
- opendaylight
- Création de flux OpenFlow
  - Matchs OpenFlow
  - Actions OpenFlow

Un routeur fait du network address translation (NAT) (« traduction d'adresse réseau » ou « translation d'adresse réseau ») lorsqu'il fait correspondre des adresses IP à d'autres adresses IP. En particulier, un cas courant est de permettre à des machines disposant d'adresses qui font partie d'un intranet et ne sont ni uniques ni routables à l'échelle d'Internet, de communiquer avec le reste d'Internet en faisant semblant d'utiliser des adresses externes uniques et routables.
Ainsi, il est possible de faire correspondre une seule adresse externe publique visible sur Internet à toutes les adresses d'un réseau privé, afin de pallier l'épuisement des adresses IPv4.
La fonction NAT dans un routeur de service intégré (ISR) traduit une adresse IP source interne en adresse IP globale.
Ce procédé est très largement utilisé par les box internet (ou modem routeur) des fournisseurs d’accès pour cacher les ordinateurs domestiques derrière une seule

identification publique. Il est également utilisé de façon similaire dans des réseaux privés virtuels

Description de la topologie

La topologie de test nécessite la mise en oeuvre:

- de deux hôtes, tous connectés l'un à l'autre via un seul commutateur activé par OpenFlow. Ceci est le plan de données.
- d'un commutateur connecté à un contrôleur. Ceci est le plan de contrôle.

Dans ce didacticiel, on utilise un contrôleur opendaylight.

Mise en oeuvre de la topologie

La topologie GNS3 est composée de 5 noeuds:

- **Noeud 1**: host 1
  - ifconfig eth0 10.0.0.1 netmask 255.255.255.0
  - route add default gateway 10.0.0.7
- **Noeud 2**: host 2
  - ifconfig eth0 192.168.0.200 netmask 255.255.255.0
  - route add default gateway 192.168.0.7
- **Noeud 3**: interface NAT pour accés au réseau via openvswitch et au contrôleur floodlight (réseau 192.168.122.0/24)
- **Noeud 4**: commutateur
  - vlan 10 : ip address 10.0.0.7 24
  - vlan 192 : ip address 192.168.0.7 24
  - vlan 212 : ip address 192.168.122.7 24
  - ge1/0 : port access vlan 10 (host1)
  - ge2/0 : port access vlan 192 (host2)
  - ge3/0 : port access vlan 212 (NAT)

Configuration du contrôleur

Une fois la topologie installée et que toutes les ressources sont en place, lancer le contrôleur opendaylight à partir de la ressource contrôleur.

Se connecter sur l'UI opendaylight avec user/mdp 'karaf/karaf' :

```
ssh -p 8101 karaf@localhost
```

Les features **openflowplugin** suivantes doivent être installées :

Feature	Classe
odl-openflowjava-protocol	OpenDaylight::Openflow Java::Protocol
odl-openflowplugin-flow-services-ui	OpenDaylight::Openflow Plugin::Flow Services:
odl-openflowplugin-flow-services-rest	OpenDaylight::Openflow Plugin::Flow Services:
odl-openflowplugin-flow-services	OpenDaylight::Openflow Plugin::Flow Services
odl-openflowplugin-southbound	OpenDaylight::Openflow Plugin::Li southbound A
odl-openflowplugin-nsf-model	OpenDaylight::OpenflowPlugin::NSF::Model
odl-openflowplugin-app-config-pusher	OpenDaylight::Openflow Plugin::Application - d
odl-openflowplugin-app-topology	OpenDaylight::Openflow Plugin::Application - t
odl-openflowplugin-app-forwardingrules-manager	OpenDaylight::Openflow Plugin::Application - F

Configuration du commutateur

Maintenant que le contrôleur est configuré et fonctionne, il faut configurer le commutateur.

Configurer une instance

Pour configurer OpenFlow, dans la vue système, utiliser la commande `openflow` et spécifier une instance. Ceci amène aux options de configuration globales OpenFlow:

```
system-view
System View: return to User View with Ctrl+Z.
[HPE]openflow instance 1
```

Une instance OpenFlow doit être configurée. Sur les commutateurs ProVision, il existe un mappage un à un entre les instances et les VLAN. En d'autres termes, chaque VLAN nécessite une instance OpenFlow distincte. Ceci n'est pas vrai pour les commutateurs Comware (plusieurs VLAN peuvent être mappés sur une seule instance OpenFlow).

Contrairement aux commutateurs 5900 prenant en charge 4094 instances d'OpenFlow, les commutateurs de la série 5500 ne prennent en charge que 8 instances.

Configurer le contrôleur

Un ID de contrôleur et une adresse IP doivent être spécifiés. On peut configurer plusieurs contrôleurs pour la redondance et l'équilibrage de la charge. Dans ce cas, un seul contrôleur est configuré.

```
[HPE-of-inst-1]controller 1 address ip 192.168.56.7
```

Associer les vlans

Les commutateurs hybrides OpenFlow prennent en charge le fonctionnement OpenFlow et le fonctionnement de commutation Ethernet normal. Certains VLAN utilisent la commutation Ethernet L2 traditionnelle, l'isolation de VLAN, le routage L3, le traitement des listes de contrôle d'accès et de qualité de service, etc. Ce type de commutateur doit fournir un mécanisme de classification qui achemine le trafic vers le pipeline OpenFlow ou le pipeline normal. Les commutateurs HP utilisent des VLAN (un ou plusieurs) à cette fin.

```
[5500-1-of-inst-1]classification vlan 10
This command isn't effective until the active instance command is issued.
[5500-1-of-inst-1]
```

Activer l'instance

La dernière étape consiste à activer l'instance:

```
[5500-1-of-inst-1] active instance
```

C'est tout. Il n’est pas trop difficile de configurer une seule instance de base d’OpenFlow sur les commutateurs HP ProVision

On doit voir le commutateur se connecter au contrôleur.

Pour s'assurer de pouvoir communiquer avec tout le monde:

- sur l'hôte 1 passer la commande:

```
ping 192.168.0.200
```

- sur l'hôte 2 passer la commande:

```
ping 10.0.0.1
```

A ce stade les hôtes ne peuvent pas communiquer entre eux, car la seule règle installée est la règle da la table MISS qui permet de dropper les flux qui ne correspondent à aucune règle.

Recherche des informations sur les

périphériques connectés

Sur le commutateur récupérer le DPID de l'instance OPENFLOW

```
show openflow instance 1
```

Puis demander au contrôleur quels périphériques sont connectés. Cela donnera toutes les informations dont on aura besoin, par exemple. ports, adresses MAC, adresses IP. Pour cela avec postman faire un GET à l'url:

http://x.x.x.x:8181/restconf/operational/opendaylight-inventory:nodes/node/openflow:#

où # est le DPID valide du commutateur

Dans la sortie, on obtiendra une liste des périphériques que Floodlight a appris, récupérer les informations des vlans 10 et 20 et des ports GE1/0 (host1) et GE2/0 (host 2).

Par exemple pour élaborer ce lab, la sortie pour le VLAN 10 ressemble à ceci:

```
                {
                    "id": "openflow:506043239107447:404",
                    "opendaylight-port-statistics:flow-capable-node-
connector-statistics": {
                        "transmit-drops": 18446744073709551615,
                        "receive-frame-error": 18446744073709551615,
                        "receive-drops": 18446744073709551615,
                        "receive-crc-error": 18446744073709551615,
                        "bytes": {
                            "transmitted": 18446744073709551615,
                            "received": 18446744073709551615
                        },
                        "duration": {
                            "second": 4719,
                            "nanosecond": 4294967295
                        },
                        "receive-errors": 18446744073709551615,
                        "transmit-errors": 18446744073709551615,
                        "receive-over-run-error": 18446744073709551615,
                        "collision-count": 18446744073709551615,
                        "packets": {
                            "transmitted": 18446744073709551615,
                            "received": 18446744073709551615
                        }
                    },
                    "flow-node-inventory:peer-features": "",
                    "flow-node-inventory:advertised-features": "",
                    "flow-node-inventory:port-number": "404",
                    "flow-node-inventory:hardware-address":
```

```
"cc:3e:5f:82:0b:77",
                    "flow-node-inventory:supported": "",
                    "flow-node-inventory:current-speed": 0,
                    "flow-node-inventory:current-feature": "",
                    "flow-node-inventory:state": {
                        "live": true,
                        "link-down": false,
                        "blocked": false
                    },
                    "flow-node-inventory:maximum-speed": 0,
                    "flow-node-inventory:name": "Vlan10",
                    "flow-node-inventory:configuration": ""
                },
```

Les informations importantes à récupérer sont le flow-node-inventory:name (le nom canique de l'interface), l'id et le flow-node-inventory:hardware-address (l'adresse MAC).

Les valeurs collectées en sortie de liste sont présentées dans le tableau suivant :

flow-node-inventory:name	**id**	**flow-node-inventory:hardware-address**
Vlan10	openflow:506043239114435:404	cc:3e:5f:82:0b:77
Vlan192	openflow:506043239114435:406	cc:3e:5f:82:0b:77
GE1/0	openflow:506043239114435:17	0c:8b:f6:6a:0b:00
GE2/0	openflow:506043239114435:33	0c:8b:f6:6a:0b:01

Fabrication des flux statiques

Flux statiques pour permettre la communication entre host1 et host2

Afin de faire communiquer les hôtes installer deux règles supplémentaires sur ODL sur la table-0 (à l'aide de son API REST et de Postman), comme suit (cf. odl-lab-1-openflow) :

Flux statique de host1 vers host2

Méthode	PUT
URI	/restconf/config/opendaylight-inventory:nodes/node/openflow:506043239114435/table/0/flow/160

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<flow xmlns="urn:opendaylight:flow:inventory">
<strict>false</strict>
<flow-name>host1-host2</flow-name>
<id>160</id>
<cookie_mask>255</cookie_mask>
<cookie>105</cookie>
<table_id>0</table_id>
<priority>1</priority>
```

```
<hard-timeout>0</hard-timeout>
<idle-timeout>0</idle-timeout>
<match>
<ethernet-match>
<ethernet-type>
<type>2048</type>
</ethernet-type>
</ethernet-match>
<in-port>openflow:506043239114435:404</in-port>
</match>
<instructions>
<instruction>
<order>0</order>
<apply-actions>
<action>
<order>0</order>
<output-action>
<output-node-connector>openflow:506043239114435:406</output-node-connector>
<max-length>65535</max-length>
</output-action>
</action>
</apply-actions>
</instruction>
</instructions>
</flow>
```

Flux statique de host2 vers host1

Méthode	PUT
URI	/restconf/config/opendaylight-inventory:nodes/node/openflow:506043239114435/table/0/flow/161

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<flow xmlns="urn:opendaylight:flow:inventory">
<strict>false</strict>
<flow-name>host2-host1</flow-name>
<id>161</id>
<cookie_mask>255</cookie_mask>
<cookie>105</cookie>
<table_id>0</table_id>
<priority>1</priority>
<hard-timeout>0</hard-timeout>
<idle-timeout>0</idle-timeout>
<match>
<ethernet-match>
<ethernet-type>
<type>2048</type>
</ethernet-type>
</ethernet-match>
<in-port>openflow:506043239114435:406</in-port>
</match>
```

```
<instructions>
<instruction>
<order>0</order>
<apply-actions>
<action>
<order>0</order>
<output-action>
<output-node-connector>openflow:506043239114435:404</output-node-connector>
<max-length>65535</max-length>
</output-action>
</action>
</apply-actions>
</instruction>
</instructions>
</flow>
```

Vérification des communications

Les communications entre l'hôte1 et l'hôte2 sont possibles.

Flux pour permettre le redirection*

Afin de permettre la communication vers l'hôte 2 depuis et vers internet nous allons ajouter deux flux.

Flux pour modifier l'adresse ip de destination

Méthode	PUT
URI	/restconf/config/opendaylight-inventory:nodes/node/openflow:506043239114435/table/0/flow/260

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<flow xmlns="urn:opendaylight:flow:inventory">
<strict>false</strict>
<flow-name>dst4</flow-name>
<id>260</id>
<cookie_mask>255</cookie_mask>
<cookie>105</cookie>
<table_id>0</table_id>
<priority>10</priority>
<hard-timeout>0</hard-timeout>
<idle-timeout>0</idle-timeout>
<match>
<ethernet-match>
<ethernet-type>
<type>2048</type>
</ethernet-type>
</ethernet-match>
<ipv4-destination>xx.xx.xxx.xxx/32</ipv4-destination>
```

```
</match>
<instructions>
<instruction>
<order>0</order>
<apply-actions>
<action>
<order>0</order>
<set-nw-dst-action>
<ipv4-address>192.168.0.200/32</ipv4-address>
</set-nw-dst-action>
</action>
<action>
<order>1</order>
<output-action>
<output-node-connector>openflow:506043239114435:406</output-node-connector>
<max-length>65535</max-length>
</output-action>
</action>
</apply-actions>
</instruction>
</instructions>
</flow>
```

Flux pour cacher l'adresse ip de l'hôte 2

Méthode	PUT
URI	/restconf/config/opendaylight-inventory:nodes/node/openflow:506043239114435/table/0/flow/261

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<flow xmlns="urn:opendaylight:flow:inventory">
<strict>false</strict>
<flow-name>src4</flow-name>
<id>261</id>
<cookie_mask>255</cookie_mask>
<cookie>105</cookie>
<table_id>0</table_id>
<priority>10</priority>
<hard-timeout>0</hard-timeout>
<idle-timeout>0</idle-timeout>
<match>
<ethernet-match>
<ethernet-type>
<type>2048</type>
</ethernet-type>
</ethernet-match>
<ipv4-source>192.168.0.200/32</ipv4-source>
</match>
<instructions>
<instruction>
<order>0</order>
```

```
<apply-actions>
<action>
<order>0</order>
<set-nw-src-action>
<ipv4-address>xx.xx.xxx.xxx/32</ipv4-address>
</set-nw-src-action>
</action>
<action>
<order>1</order>
<output-action>
<output-node-connector>openflow:506043239114435:405</output-node-connector>
<max-length>65535</max-length>
</output-action>
</action>
</apply-actions>
</instruction>
</instructions>
</flow>
```

Après l'ajout de cette règle lorsqu'on fait un ping depuis l'hôte1 vers l'adresse 192.168.0.200 on a plus de retour, ça indique que les règles de nattage fonctionnent.

Mais si on fait un ping à l'adresse xx.xx.xxx.xxx on n'a pas de retour non plus, qu'est ce qu'il se passe?

Regardons sous le capot en faisant une trace avec wireshark, sur l'interface ge1/0 on voit bien des paquets icmp passer

```
No.     Time           Source                Destination           Protocol
Length Info
      2 0.308031       10.0.0.1              xx.xx.xxx.xxx         ICMP
98     Echo (ping) request  id=0xbb12, seq=21/5376, ttl=64 (no response
found!)

Frame 2: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on
interface 0
Ethernet II, Src: 0c:fb:b8:b4:93:00 (0c:fb:b8:b4:93:00), Dst:
HewlettP_82:26:c3 (cc:3e:5f:82:26:c3)
Internet Protocol Version 4, Src: 10.0.0.1, Dst: xx.xx.xxx.xxx
Internet Control Message Protocol
```

Mais sur les autres interfaces on ne voit rien, c'est logique, aucune indication à ce stade ne permet au commutateur de choisir quelle route prendre pour atteindre cette addresse:

- ni dans la table arp
- ni par une route statiqu

Ajouttons une route statique dans la configuration du commutateur:

```
ip route-static xx.xx.xxx.xxx 255.255.255.255 192.168.0.200
```

Cette fois un ping depuis l'hôte 1 vers l'adresse fictive aboutit

```
ping xx.xx.xxx.xxx
```

Pour s'assurer que les flux sont bien rediriger vers l'hôte 2, mettre sur celui-ci le port 8000 à l'écoute avec netcat:

```
nc -lp 8000
```

Depuis l'hôte 1 se connecter avec nectcat à l'adresse xx.xx.xxx.xxx sur le port 8000 (avec un message) :

```
echo "bonjour"|nc xx.xx.xxx.xxx 8000
```

On doit voir le message “bonjour” sur l'hôte 2.