OVS: Lab 2 Construire un routeur avec Open vSwitch

Objet	mise en place d'un routeur distribué à l'aide d'Open vSwitch.
Niveau requis	débutant, avisé
Débutant, à savoir	ovs-openflow-overview
Suivi	:DRAFT:

Topologie de test

- Un hôte dans un réseau externe 172.16.1.0/24
- un hôte dans un réseau interne 10.10.10.0/24
- deux hôtes dans un autre réseau interne 10.10.20.0/24.

En tant que tels, les hôtes de la gamme 10.x.x.x devraient pouvoir se parler, mais ne devraient pas pouvoir parler à des hôtes externes.

L'hôte 10.10.10.2 a une adresse IP flottante de 172.16.1.10 et doit être accessible à cette adresse depuis le réseau externe 172.16.1.0/24. Pour ce faire, nous utiliserons DNAT pour le trafic à partir de 172.16.1.2 → 172.16.1.10 et SNAT pour le trafic à partir de 10.10.10.2 → 172.16.1.2.

Configuration du routeur

Set Bridge to use OpenFlow 1.3

```
sh ovs-vsctl set Bridge s1 "protocols=OpenFlow13"
```

Create Groups

```
sh ovs-ofctl add-group -OOpenFlow13 s1 group_id=1,type=all,bucket=output:1
sh ovs-ofctl add-group -OOpenFlow13 s1 group_id=2,type=all,bucket=output:2,4
sh ovs-ofctl add-group -OOpenFlow13 s1 group_id=3,type=all,bucket=output:3
```

Table 0 - Classificateur

Dans cette table, on détermine le trafic que l'on veut traiter avant de le pousser plus loin dans le pipeline.

```
# Envoyer ARP au répondeur ARP
sh ovs-ofctl add-flow -OOpenFlow13 s1 "table=0, priority=1000,
dl_type=0x0806, actions=goto_table=105"
# Envoi du trafic L3 à la table de réécriture L3
```

```
sh ovs-ofctl add-flow -OOpenFlow13 s1 "table=0, priority=100,
dl_dst=00:00:5E:00:02:01, action=goto_table=5"
sh ovs-ofctl add-flow -OOpenFlow13 s1 "table=0, priority=100,
dl_dst=00:00:5E:00:02:02, action=goto_table=5"
sh ovs-ofctl add-flow -OOpenFlow13 s1 "table=0, priority=100,
dl_dst=00:00:5E:00:02:03, action=goto_table=5"
# Envoyer N3 à la table de réécriture N2
sh ovs-ofctl add-flow -OOpenFlow13 s1 "table=0, priority=0,
action=goto_table=20"
```

Table 5 - Réécriture de L3

Dans cette table, on apporte toutes les modifications de niveau 3 nécessaires avant qu'un paquet ne soit routé

```
# Exclure les sous-réseaux connectés
sh ovs-ofctl add-flow -OOpenFlow13 s1 "table=5, priority=65535,
dl_type=0x0800, nw_dst=10.10.10.0/24 actions=goto_table=10"
sh ovs-ofctl add-flow -OOpenFlow13 s1 "table=5, priority=65535,
dl_type=0x0800, nw_dst=10.10.20.0/24 actions=goto_table=10"
# DNAT
sh ovs-ofctl add-flow -OOpenFlow13 s1 "table=5, priority=100,
dl_type=0x0800,  nw_dst=172.16.1.10 actions=mod_nw_dst=10.10.10.2,
goto_table=10"
# SNAT
sh ovs-ofctl add-flow -OOpenFlow13 s1 "table=5, priority=100,
dl_type=0x0800,  nw_src=10.10.10.2, actions=mod_nw_src=172.16.1.10,
goto_table=10"
# Si aucune réécriture n'est nécessaire, passez au tableau 10.
sh ovs-ofctl add-flow -OOpenFlow13 s1 "table=5, priority=0,
actions=goto_table=10"
```

Table 10 - Routage IPv4

La table de routage L3 est l'endroit où le routage se produit: 1. on modifie l’adresse MAC source, 2. on décrémente la durée de vie 3. on pousse vers les tables L2 pour le transfert

```
sh  ovs-ofctl add-flow -OOpenFlow13 s1 "table=10, dl_type=0x0800,
nw_dst=10.10.10.0/24, actions=mod_dl_src=00:00:5E:00:02:01, dec_ttl,
goto_table=15"
sh  ovs-ofctl add-flow -OOpenFlow13 s1 "table=10, dl_type=0x0800,
nw_dst=10.10.20.0/24, actions=mod_dl_src=00:00:5E:00:02:02, dec_ttl,
goto_table=15"
sh  ovs-ofctl add-flow -OOpenFlow13 s1 "table=10, dl_type=0x0800,
nw_dst=172.16.1.0/24, actions=mod_dl_src=00:00:5E:00:02:03, dec_ttl,
goto_table=15"
# Drop explicite si impossible de router
```

```
sh  ovs-ofctl add-flow -OOpenFlow13 s1 "table=10, priority=0,
actions=output:0"
```

Table 15 - Transmission (L3 Forwarding)

Dans la table de transfert L3, oun résous une adresse IP de destination en l'adresse MAC correcte pour le transfert L2.

```
sh ovs-ofctl add-flow -OOpenFlow13 s1 "table=15, dl_type=0x0800,
nw_dst=10.10.10.2, actions=mod_dl_dst:00:00:00:00:00:01, goto_table=20"
sh ovs-ofctl add-flow -OOpenFlow13 s1 "table=15, dl_type=0x0800,
nw_dst=10.10.20.2, actions=mod_dl_dst:00:00:00:00:00:02, goto_table=20"
sh ovs-ofctl add-flow -OOpenFlow13 s1 "table=15, dl_type=0x0800,
nw_dst=10.10.20.4, actions=mod_dl_dst:00:00:00:00:00:04, goto_table=20"
sh ovs-ofctl add-flow -OOpenFlow13 s1 "table=15, dl_type=0x0800,
nw_dst=172.16.1.2, actions=mod_dl_dst:00:00:00:00:00:03, goto_table=20"
sh ovs-ofctl add-flow -OOpenFlow13 s1 "table=15, priority=0,
actions=goto_table=20"
```

Table 20 - Écritures L2

Bien qu'on n'utilise pas cette table dans cet exemple, celleci pourrait servir à effacer/servir toutes les encapsulations L2 ici, comme une balise VLAN ou un ID de tunnel VXLAN / GRE.

```
# Go to next table
sh  ovs-ofctl add-flow -OOpenFlow13 s1 "table=20, priority=0,
actions=goto_table=25"
```

Table 25 - Transmission L2

Dans cette table, on effectue la recherche L2 et on transmet le port correct. On gére également le trafic L2 BUM ici à l'aide de groupes OpenFlow - pas de VLAN!

```
# Utiliser des groupes pour le trafic BUM
sh ovs-ofctl add-flow -OOpenFlow13 s1 "table=25, in_port=1,
dl_dst=01:00:00:00:00:00/01:00:00:00:00:00, actions=group=1"
sh ovs-ofctl add-flow -OOpenFlow13 s1 "table=25, in_port=2,
dl_dst=01:00:00:00:00:00/01:00:00:00:00:00, actions=group=2"
sh ovs-ofctl add-flow -OOpenFlow13 s1 "table=25, in_port=3,
dl_dst=01:00:00:00:00:00/01:00:00:00:00:00, actions=group=3"
sh ovs-ofctl add-flow -OOpenFlow13 s1 "table=25, in_port=4,
dl_dst=01:00:00:00:00:00/01:00:00:00:00:00, actions=group=2"
sh ovs-ofctl add-flow -OOpenFlow13 s1 "table=25,
dl_dst=00:00:00:00:00:01,actions=output=1"
```

```
sh ovs-ofctl add-flow -OOpenFlow13 s1 "table=25,
dl_dst=00:00:00:00:00:02,actions=output=2"
sh ovs-ofctl add-flow -OOpenFlow13 s1 "table=25,
dl_dst=00:00:00:00:00:03,actions=output=3"
sh ovs-ofctl add-flow -OOpenFlow13 s1 "table=25,
dl_dst=00:00:00:00:00:04,actions=output=4"
```

Table 100 - LCA

Bien qu'on n'utilise pas cette table aujourd'hui, l'idée serait de filtrer le trafic dans cette table (ou dans une série de tables), puis de le soumettre à nouveau au classificateur une fois nettoyés.

Table 105 - Répondeur ARP

Dans cette table, on utilise certains OVS-Jitsu pour prendre une requête ARP entrante et la transformer en réponse ARP

```
# Répondre à ARP pour les adresses de routeur
sh ovs-ofctl add-flow -OOpenFlow13 s1 "table=105, dl_type=0x0806,
nw_dst=10.10.10.1, actions=move:NXM_OF_ETH_SRC[]->NXM_OF_ETH_DST[],
mod_dl_src:00:00:5E:00:02:01, load:0x2->NXM_OF_ARP_OP[],
move:NXM_NX_ARP_SHA[]->NXM_NX_ARP_THA[],
move:NXM_OF_ARP_SPA[]->NXM_OF_ARP_TPA[],
load:0x00005e000201->NXM_NX_ARP_SHA[], load:0x0a0a0a01->NXM_OF_ARP_SPA[],
in_port"
sh ovs-ofctl add-flow -OOpenFlow13 s1 "table=105,  dl_type=0x0806,
nw_dst=10.10.20.1, actions=move:NXM_OF_ETH_SRC[]->NXM_OF_ETH_DST[],
mod_dl_src:00:00:5E:00:02:02, load:0x2->NXM_OF_ARP_OP[],
move:NXM_NX_ARP_SHA[]->NXM_NX_ARP_THA[],
move:NXM_OF_ARP_SPA[]->NXM_OF_ARP_TPA[],
load:0x00005e000202->NXM_NX_ARP_SHA[], load:0xa0a1401->NXM_OF_ARP_SPA[],
in_port"
sh ovs-ofctl add-flow -OOpenFlow13 s1 "table=105,  dl_type=0x0806,
nw_dst=172.16.1.1, actions=move:NXM_OF_ETH_SRC[]->NXM_OF_ETH_DST[],
mod_dl_src:00:00:5E:00:02:03, load:0x2->NXM_OF_ARP_OP[],
move:NXM_NX_ARP_SHA[]->NXM_NX_ARP_THA[],
move:NXM_OF_ARP_SPA[]->NXM_OF_ARP_TPA[],
load:0x00005e000203->NXM_NX_ARP_SHA[], load:0xac100101->NXM_OF_ARP_SPA[],
in_port"
# Le proxy ARP pour toutes les IP flottantes va en dessous
sh ovs-ofctl add-flow -OOpenFlow13 s1 "table=105, dl_type=0x0806,
nw_dst=172.16.1.10, actions=move:NXM_OF_ETH_SRC[]->NXM_OF_ETH_DST[],
mod_dl_src:00:00:5E:00:02:03, load:0x2->NXM_OF_ARP_OP[],
move:NXM_NX_ARP_SHA[]->NXM_NX_ARP_THA[],
move:NXM_OF_ARP_SPA[]->NXM_OF_ARP_TPA[],
load:0x00005e000203->NXM_NX_ARP_SHA[], load:0xac10010a->NXM_OF_ARP_SPA[],
in_port"
```

```
# si nous l'avons fait ici, le paquet arp doit être traité comme n'importe
quel autre paquet L2 normal
sh ovs-ofctl add-flow -OOpenFlow13 s1 "table=105, priority=0,
action=resubmit(,20)"
```

Jusqu'ici, on a mis en place les bases du routage, mais il manque un élément crucial: la gestion ICMP. Sans cette fonctionnalité utile, par exemple Path MTU Discovery ne fonctionnera pas, pas plus que les outils de diagnostic tels que Ping et Traceroute.

Vocabulaire

L2 – Couche 2 (liaison de données)

Les switchs de couche 2 (L2) fonctionnent avec les adresse réseau physiques. Les adresses physiques, aussi appelées de couche liaison, matérielles ou de niveau MAC, identifient les appareils individuellement. Durant le processus de fabrication, un numéro permanent a été attribué à chaque dispositif matériel.

L3 – Couche 3 (réseau)

Les switchs L3 utilisent les adresses du réseau ou IP qui identifient des emplacements du réseau. Les adresses physiques identifient des appareils ; les adresses réseau identifient des emplacements. Un emplacement peut être un poste utilisateur LAN, une adresse dans la mémoire d'un ordinateur ou même un paquet de données qui transite sur le réseau.

Adresses Flottantes

Une IP flottante est habituellement une adresse IP publique, routable, qui n'est pas attribuée automatiquement à une instance mais selon le besoin et de manière temporaire.