Routage inter-VLAN avec Open vSwitch

Philippe Latu
philippe.latu(at)inetdoc.net

https://www.inetdoc.net

Résumé

L'usage des commutateurs virtuels se développe très rapidement avec l'augmentation tout aussi rapide du nombre d'instances de machines virtuelles ou de conteneurs hébergés sur un même système hôte physique. En fait, les besoins en commutation de circuits et de paquets sont toujours présents que l'on utilise des équipements physiques ou des systèmes virtuels. Cet article présente deux maquettes réduites au strict minimum qui permettent d'illustrer l'utilisation du commutateur virtuel **Open vSwitch** dans un contexte de routage inter-VLAN. Le parti pris des manipulations est de reproduire à l'identique les opérations que l'on réaliserait sur des équipements réels.

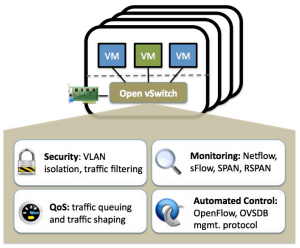


Table des matières

1. Copyright et Licence 2
    1.1. Méta-information 2
2. Le contexte : la maquette et les topologies types 3
3. Préparartion du système hôte 5
4. Routage inter-VLAN dans le système hôte 6
    4.1. Plan d'adressage 6
    4.2. Commutateur et cordons de brassage 7
    4.3. Création des interfaces de type SVI 7
    4.4. Activation du routage sur le système hôte 8
    4.5. Autoconfiguration des hôtes des VLANs Orange et Vert 8
    4.6. Affichage des tables de routage du système hôte 9
    4.7. Traduction d'adresses source sur le système hôte 10
    4.8. Lancement des systèmes virtuels 11
    4.9. Tests d'interconnexion entre réseaux 12
5. Routage inter-VLAN dans un système virtuel 13
    5.1. Plan d'adressage 13
    5.2. Commutateurs et cordons de brassage 13
    5.3. Lancement des systèmes virtuels 15
    5.4. Création des interfaces de routage inter-VLAN 16
    5.5. Activation du routage sur le routeur virtuel 17
    5.6. Autoconfiguration des hôtes des VLANs Orange et Vert 17
    5.7. Affichage des tables de routage du routeur virtuel 18
    5.8. Traduction d'adresses source sur le routeur virtuel 19
    5.9. Tests d'interconnexion entre réseaux 19
6. Applications pratiques 20

1. Copyright et Licence


```
Copyright (c) 2000,2026 Philippe Latu.
Permission is granted to copy, distribute and/or modify this document under the
terms of the GNU Free Documentation License, Version 1.3 or any later version
published by the Free Software Foundation; with no Invariant Sections, no
Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included
in the section entitled "GNU Free Documentation License".
```

```
Copyright (c) 2000,2026 Philippe Latu.
Permission est accordée de copier, distribuer et/ou modifier ce document selon
les termes de la Licence de Documentation Libre GNU (GNU Free Documentation
License), version 1.3 ou toute version ultérieure publiée par la Free Software
Foundation ; sans Sections Invariables ; sans Texte de Première de Couverture,
et sans Texte de Quatrième de Couverture. Une copie de la présente Licence est
incluse dans la section intitulée « Licence de Documentation Libre GNU ».
```


1.1. Méta-information

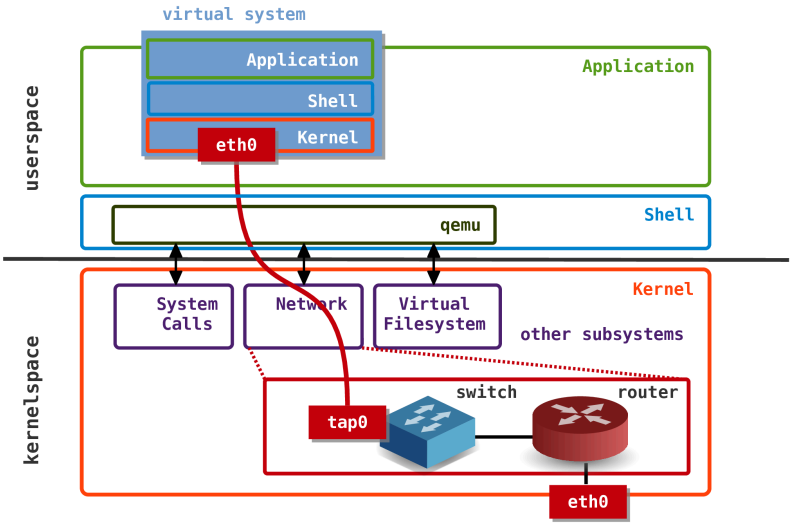
Cet article est écrit avec **DocBook** XML sur un système **Debian GNU/Linux**. Il est disponible en version imprimable au format PDF : inter-vlan-routing-openvswitch.pdf.

2. Le contexte : la maquette et les topologies types

Comme annoncé en introduction, on cherche à illustrer les fonctions réseau classiques de commutation de circuits et de paquets. Par définition, la commutation de circuits est le propre d'un commutateur tandis que la commutation de paquets est le propre d'un routeur. Sur des équipements réels, un routeur se distingue d'un commutateur par l'électronique de ses interfaces. Non seulement cette électronique sert à l'interconnexion entre réseaux hétérogènes mais elle offre aussi des fonctions de qualité de service (QoS) plus sophistiquées. L'électronique d'un commutateur permet de constituer un très grand nombre de circuits *full-duplex* entre interfaces Ethernet. Si un commutateur assure la fonction de routage, celle-ci revient à interconnecter des réseaux homogènes puisque toutes ses interfaces sont de type Ethernet.

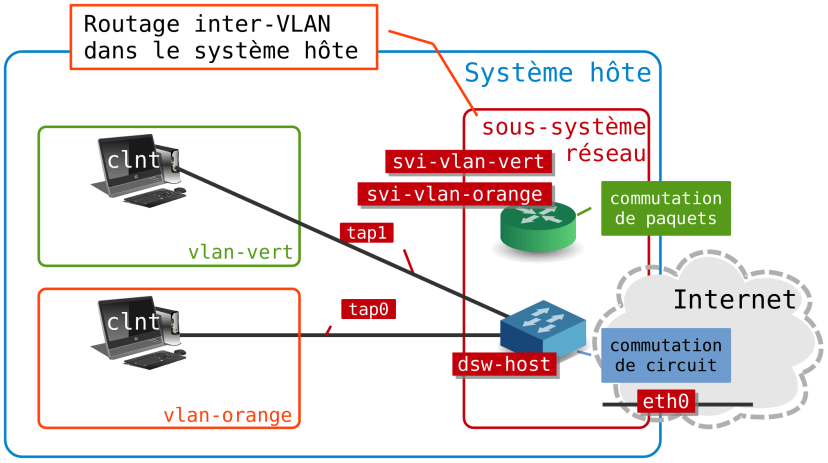
Toujours sur des équipements réels modernes, la commutation de paquets et la commutation de circuits se font grâce à des composants qui manipulent un type de mémoire particulier appelé **Content-addressable memory**. Ces composants permettent d'accélérer considérablement la transmission des données en utilisant un algorithme de hachage des informations contenues dans les en-têtes de trames et de paquets. Plutôt que suivre le processus de désencapsulation normal qui consiste à analyser tous les champs des en-têtes de chaque couche, on se contente de comparer des valeurs binaires calculées à partir du flux réseau entrant et d'un flux antérieur. Si un flux arrive sur une interface avec les mêmes propriétés qu'un flux antérieur (adresses MAC, étiquette IEEE802.1Q, adresses IP, etc.) pour lequel la décision de commutation a déjà été prise, il n'est pas nécessaire de reprendre l'examen des en-têtes.

Dans un contexte de virtualisation, les besoins sont exactement les mêmes. Si le système hôte ne dispose pas (encore) de mémoire TCAM, il est tout à fait possible d'utiliser les mêmes algorithmes de hachage dans le sous-système réseau du noyau pour atteindre le même objectif d'accélération des transmissions entre interfaces. C'est ici qu'il faut mettre un bémol, la comparaison des résultats de hachage et la transmission ne se font plus à la «vitesse du silicium» du composant spécialisé mais à la vitesse d'accès à la mémoire vive (RAM) et aux interfaces réseau du système hôte. Or, une interface réseau de serveur aussi performante soit elle, ne pourra garantir un très grand nombre de circuits *full-duplex*. C'est là que se situe le goulot d'étranglement dans les transmissions réseau entre le monde virtuel et le monde réel.

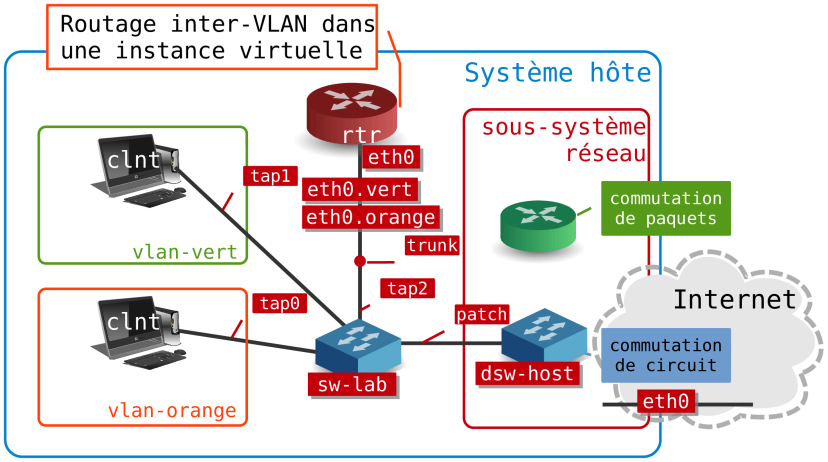


Le rôle du commutateur virtuel **Open vSwitch** est justement de gérer les commutations de circuits et de paquets en coopération avec le sous-système réseau du noyau Linux du système hôte. Ses performances dépendent donc directement des composants disponibles sur le système hôte. Dans le contexte de cet article, il présente un avantage indéniable, il se manipule exactement comme un commutateur réel. Voyons les schémas de topologies types étudiées.

Dans le premier exemple, la fonction de routage (commutation de paquets) est assurée directement par le système hôte.



Dans le second exemple, la fonction de routage (commutation de paquets) est assurée par une instance de système virtuel. De plus, on illustre la cascade (*stacking*) entre commutateurs virtuels.



3. Préparartion du système hôte

Avant de débuter les manipulations, on doit s'assurer que le système hôte dispose de tous les éléments nécessaires.

1. Les fonctions matérielles de virtualisation sur le processeur
2. Les outils utilisateurs pour la virtualisation et la commutation
3. L'attribution des droits d'accès à la virtualisation pour un utilisateur normal

- Pour vérifier que le processeur du système hôte dispose bien des fonctions de paravirtualisation, il faut consulter la liste des attributs du processeur. Il existe au moins deux solutions illustrées dans les deux copies d'écran suivantes.

```
$ egrep -o -m1 '(vmx|svm)' /proc/cpuinfo
vmx
```

```
$ lscpu | grep -i virtualisation
Virtualisation :        VT-x
```

- Pour les outils utilisateurs, au moins deux paquets sont nécessaires pour la virtualisation et la commutation.

```
$ aptitude search '?or(?installed(qemu-system-x86), ?installed(openvswitch-switch))'
i   openvswitch-switch      - Open vSwitch switch implementations
i   qemu-system-x86         - QEMU full system emulation binaries (x86)
```

Si l'exécution de la commande proposée dans la copie d'écran ci-dessus ne produit aucun résultat, il faut donc procéder à l'installation des deux paquets.

```
$ sudo aptitude install openvswitch-switch qemu-system-x86
```

- Pour l'attribution des droits et permissions, il faut s'assurer que l'utilisateur normal soit bien membre des groupes système suivants :
  - kvm pour la virtualisation
  - sudo pour la commutation

```
# adduser etu kvm
```

> **Avertissement**
>
> L'appartenance à un nouveau groupe est conditionnée par une ouverture de nouvelle session. Après avoir utilisé la commande **adduser** de la copie d'écran ci-dessus, il est préférable de déconnecter/reconnecter l'utilisateur normal du système.

Enfin, il faut s'assurer que les fichiers de représentation des périphériques aient aussi les bonnes attributions de groupe système.

```
$ ls -l /dev/kvm
crw-rw---- 1 root kvm 10, 232 mars  13 19:50 /dev/kvm
$ ls -l /dev/vhost-net
crw-rw---- 1 root kvm 10, 238 mars   8 08:21 /dev/vhost-net
```

4. Routage inter-VLAN dans le système hôte

Dans ce premier exemple, toutes les décisions d'acheminement du trafic sont prises dans le sous-système réseau du système hôte. Au niveau commutation de circuits et de paquets, c'est **Open vSwitch** qui joue le rôle le plus important. Si on se réfère au Schéma de la topologie, l'essentiel de la configuration porte sur le commutateur `dsw-host`. Voyons par quelles étapes il faut passer.

4.1. Plan d'adressage

Tableau 1. plan d'adressage des périmètres

Nom	VLAN	Interface	Préfixe réseau IP
Système hôte	trunk	vlan1	`192.0.2.0/26`
			`2001:db8:fe00:8175::/64`
Orange	10	vlan10	`198.51.100.0/24`
			`fd6e:c073:b4a3:a::/64`
Vert	20	vlan20	`203.0.113.0/24`
			`fd6e:c073:b4a3:14::/64`

Note

Les préfixes réseau IPv4 des trois VLANs appartiennent à la catégorie **TESTNET** définie dans le document RFC5737 **IPv4 Address Blocks Reserved for Documentation**.

Les préfixes réseau IPv6 des deux VLANs `Orange` et `Vert` appartiennent à la famille ULA définie dans le document RFC4193 **Unique Local IPv6 Unicast Addresses**. Le préfixe de départ retenu est le `fd00::/8`. On lui adjoint une chaîne de 40 bits aléatoires obtenus à l'aide de la commande `$ openssl rand -hex 5` pour arriver au préfixe `fd6e:c073:b4a3::/48`. Enfin, on utilise les numéros de VLANs pour obtenir les préfixes réseau sur 64 bits.

4.2. Commutateur et cordons de brassage

Une fois que les paquets nécessaires à la configuration sont installés, on peut passer au changement de configuration de l'interface réseau. On désactive l'interface `eth0` avant de l'associer au commutateur.

1. Désactivation de l'interface physique du système hôte.

```
$ sudo ifdown eth0
```

2. Création du commutateur `dsw-host`.

```
$ sudo ovs-vsctl add-br dsw-host
```

3. Brassage de l'interface physique du système hôte sur le commutateur `dsw-host`.

```
$ sudo ovs-vsctl add-port dsw-host eth0
```

   Activation de l'interface physique au niveau liaison.

```
$ sudo ip link set dev eth0 up
```

4. Création et «activation» des deux cordons de brassage associés aux instances de machines virtuelles.

```
$ sudo ip tuntap add mode tap dev tap0 group kvm multi_queue
$ sudo ip link set dev tap0 up
$ sudo ip tuntap add mode tap dev tap1 group kvm multi_queue
$ sudo ip link set dev tap1 up
```

5. Raccordement des cordons de brassage au commutateur.

```
$ sudo ovs-vsctl add-port dsw-host tap0 tag=10
$ sudo ovs-vsctl set port tap0 vlan_mode=access
$ sudo ovs-vsctl add-port dsw-host tap1 tag=20
$ sudo ovs-vsctl set port tap1 vlan_mode=access
```

4.3. Création des interfaces de type SVI

Une fois la partie commutation de circuits en place, on s'intéresse maintenant au routage des paquets dans le sous-système réseau du système hôte. On utilise ici des interfaces logicielles de routage appelées **Switch virtual interface** que l'on configure sur le commutateur `dsw-host`.

1. Création de l'interface principale du système hôte : `vlan1`.

```
$ sudo ovs-vsctl add-port dsw-host vlan1 -- set interface vlan1 type=internal
$ sudo ovs-vsctl set port vlan1 vlan_mode=native-untagged
$ sudo ip addr add 192.0.2.29/26 brd + dev vlan1
$ sudo ip -6 addr add 2001:db8:fe00:8175::1d/64 dev vlan1
$ sudo ip link set dev vlan1 up
```

   La dernière instruction associe l'interface au VLAN natif d'une interface en mode *trunk*. Toutes les trames sans étiquette IEEE 802.1q circulant dans un *trunk* appartiennent au VLAN natif.

   Une fois configurée au niveau réseau, cette interface fournit les informations suivantes.

```
$ ip addr ls dev vlan1
4: vlan1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN group default
    link/ether 4a:cb:5b:38:d2:c2 brd ff:ff:ff:ff:ff:ff
    inet 192.0.2.29/26 brd 192.0.2.31 scope global vlan1
       valid_lft forever preferred_lft forever
    inet6 2001:db8:fe00:8175::1d/64 scope global mngtmpaddr dynamic
       valid_lft 85994sec preferred_lft 13994sec
    inet6 fe80::48cb:5bff:fe38:d2c2/64 scope link
       valid_lft forever preferred_lft forever
```

2. Création de l'interface SVI du VLAN Orange.

```
$ sudo ovs-vsctl add-port dsw-host vlan10 tag=10 -- set interface vlan10 type=internal
$ sudo ovs-vsctl set port vlan10 vlan_mode=access
$ sudo ip addr add 198.51.100.1/24 brd + dev vlan10
$ sudo ip -6 addr add fd6e:c073:b4a3:a::1/64 dev vlan10
$ sudo ip link set dev vlan10 up
```

3. Création de l'interface SVI du VLAN Vert.

```
$ sudo ovs-vsctl add-port dsw-host vlan20 tag=20 -- set interface vlan20 type=internal
$ sudo ovs-vsctl set port vlan20 vlan_mode=access
$ sudo ip addr add 203.0.113.1/24 brd + dev vlan20
$ sudo ip -6 addr add fd6e:c073:b4a3:14::1/64 dev vlan20
$ sudo ip link set dev vlan20 up
```

4.4. Activation du routage sur le système hôte

Sur un système GNU/Linux, la fonction de routage est contrôlée par les paramètres du sous-système réseau du noyau. Le fichier principal de configuration des paramètres des sous-systèmes du noyau est : `/etc/systcl`. Voici la liste des paramètres à appliquer pour activer le routage IPv4 et le routage IPv6.

```
# egrep -v '(^#|^$)' /etc/sysctl.conf
net.ipv4.conf.default.rp_filter=1
net.ipv4.conf.all.rp_filter=1
net.ipv4.ip_forward=1
net.ipv6.conf.all.forwarding=1
net.ipv4.conf.all.log_martians=1
```

Cette liste de paramètres n'est effective qu'après utilisation de l'instruction `# sysctl -p`.

Il est aussi possible le manipuler les paramètres individuellement à partir de la console à l'aide de commandes du type suivant :

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
# echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
```

4.5. Autoconfiguration des hôtes des VLANs Orange et Vert

Pour la partie IPv4, on fait appel à un démon DHCP sur le système hôte. On place ce démon en écoute sur les deux interfaces `vlan10` et `vlan20`.

Après avoir installé le paquet du serveur DHCP, on restreint son usage aux deux VLANs de la maquette en éditant le fichier `/etc/default/isc-dhcp-server`. On obtient les résultats suivants.

```
# aptitude search ~idhcp-server
i   isc-dhcp-server      - ISC DHCP server for automatic IP address assignment

# egrep -v '(^#|^$)' /etc/default/isc-dhcp-server
INTERFACES="vlan10 vlan20"
```

Côté configuration, on applique les éléments suivants dans le fichier `/etc/dhcp/dhcpd.conf`.

```
ddns-update-style none;

default-lease-time 600;
max-lease-time 7200;

authoritative;

log-facility local7;

subnet 198.51.100.0 netmask 255.255.255.0 {
  range 198.51.100.10 198.51.100.30;
  option domain-name-servers 192.0.2.1;
  option routers 198.51.100.1;
  option broadcast-address 198.51.100.255;
}

subnet 203.0.113.0 netmask 255.255.255.0 {
  range 203.0.113.10 203.0.113.30;
  option domain-name-servers 192.0.2.1;
  option routers 203.0.113.1;
  option broadcast-address 203.0.113.255;
}
```

Pour la partie IPv6, on fait appel au paquet radvd pour l'autoconfiguration des hôtes en mode SLAAC (Document RFC4862 **IPv6 Stateless Address Autoconfiguration**). On vérifie que le paquet est bien installé :

```
# aptitude search ~iradvd
i   radvd                - Démon d'information de routeur
```

Voici ensuite une copie du fichier `/etc/radvd.conf` du système hôte.

```
interface vlan10
{
   AdvSendAdvert on;
   prefix fd6e:c073:b4a3:a::/64
   {
       AdvOnLink on;
       AdvAutonomous on;
       AdvRouterAddr on;
   };

   RDNSS 2001:db8:fe00:8175::1
   {
   };
};

interface vlan20
{
   AdvSendAdvert on;
   prefix fd6e:c073:b4a3:14::/64
```

```
    {
        AdvOnLink on;
        AdvAutonomous on;
        AdvRouterAddr on;
    };

    RDNSS 2001:db8:fe00:8175::1
    {
    };
};
```

4.6. Affichage des tables de routage du système hôte

Les tables de routage font apparaître les correspondances entre les préfixes réseau définis plus haut et les interfaces configurées sur le commutateur **Open vSwitch** `dsw-host`.

La table de routage IPv4 est :

```
$ ip route ls
default via 192.0.2.1 dev vlan1
192.0.2.0/26 dev vlan1  proto kernel  scope link  src 192.0.2.29
198.51.100.0/24 dev vlan10  proto kernel  scope link  src 198.51.100.1
203.0.113.0/24 dev vlan20  proto kernel  scope link  src 203.0.113.1
```

La table de routage IPv6 est :

```
$ ip -6 route ls
2001:db8:fe00:8175::/64 dev vlan1  proto kernel  metric 256  pref medium
fd6e:c073:b4a3:a::/64 dev vlan10  proto kernel  metric 256  pref medium
fd6e:c073:b4a3:14::/64 dev vlan20  proto kernel  metric 256  pref medium
fe80::/64 dev vlan1  proto kernel  metric 256  pref medium
fe80::/64 dev eth0  proto kernel  metric 256  pref medium
fe80::/64 dev vlan10  proto kernel  metric 256  pref medium
fe80::/64 dev vlan20  proto kernel  metric 256  pref medium
fe80::/64 dev tap0  proto kernel  metric 256  pref medium
fe80::/64 dev tap1  proto kernel  metric 256  pref medium
default via 2001:db8:fe00:8175::1 dev vlan1  metric 1024  pref medium
```

4.7. Traduction d'adresses source sur le système hôte

Les VLANs Orange et Vert utilisent des préfixes réseau non routables sur l'Internet. On a donc recours à un artifice pour accéder au réseau public : la traduction des adresses sources (S-NAT) avec les deux protocoles IPv4 et IPv6.

Voici une copie des fichiers de règles pour les deux protocoles de couche réseau.

```
#
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
COMMIT
#
*nat
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
-A POSTROUTING -p tcp -m tcp --syn -m tcpmss --mss 1400:1536 -j TCPMSS --clamp-mss-to-pmtu
-A POSTROUTING -o vlan1 -j SNAT --to-source 192.0.2.29
COMMIT
```

```
#
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
COMMIT
#
*nat
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
-A POSTROUTING -p tcp -m tcp --syn -m tcpmss --mss 1400:1536 -j TCPMSS --clamp-mss-to-pmtu
-A POSTROUTING -o vlan1 -s fd00::/8 -j SNAT --to-source 2001:db8:fe00:8175::1d
COMMIT
```

De façon usuelle, il faut «surveiller» les compteurs associés à chaque règle pour en mesurer l'utilisation. À partir des deux jeux de règles donnés ci-dessus, on obtient les résultats suivants.

```
# iptables -t nat -vnL POSTROUTING
Chain POSTROUTING (policy ACCEPT 10 packets, 975 bytes)
 pkts bytes target     prot opt in     out     source       destination
    7   420 TCPMSS     tcp  --  *      *       0.0.0.0/0    0.0.0.0/0    tcp flags:0x17/0x02 tcpmss match 1400:1536 TCPMSS clamp t
  143  9668 SNAT       all  --  *      vlan1   0.0.0.0/0    0.0.0.0/0    to:192.0.2.29
```

```
# ip6tables -t nat -vnL POSTROUTING
Chain POSTROUTING (policy ACCEPT 117 packets, 11199 bytes)
 pkts bytes target     prot opt in     out     source       destination
   94  7520 TCPMSS     tcp      *      *       ::/0         ::/0         tcp flags:0x17/0x02 tcpmss match 1400:1536 TCPMSS clamp t
   12  1112 SNAT       all      *      vlan1   fd00::/8     ::/0         to:2001:db8:fe00:8175::1d
```

4.8. Lancement des systèmes virtuels

On lance deux instances de systèmes virtuels dans chacun des deux VLANs Orange et Vert à l'aide du script suivant :

```
#!/bin/bash

../scripts/ovs-startup.sh orange.qed 1024 0
../scripts/ovs-startup.sh green.qed 1024 1
```

Le code du script de lancement d'une machine virtuelle raccordée à un commutateur Open vSwitch est donné dans le guide **Virtualisation système et enseignement**.

Dans ce script, l'instance `orange.qed` utilise le cordon de brassage `tap0`. Ce cordon a été brassé sur un port en mode accès associé au VLAN `10`. Son adresse MAC est générée automatiquement en fonction du numéro du cordon : `ba:ad:ca:fe:00:00`.

De la même façon, l'instance `green.qed` utilise le cordon de brassage `tap1`. Ce cordon a été brassé sur un autre port en mode accès associé au VLAN `20`. Son adresse MAC est générée automatiquement en fonction du numéro du cordon : `ba:ad:ca:fe:00:01`.

On visualise l'état des connexions en consultant la table CAM (**Content-addressable memory**) du commutateur `dsw-host`.

```
$ sudo ovs-appctl fdb/show dsw-host
 port  VLAN  MAC                Age
    4    20 ba:ad:ca:fe:00:01    3
    5    10  66:d2:01:39:4a:21    3
    3    10 ba:ad:ca:fe:00:00    3
    6    20  ee:8f:84:8e:cb:a8    3
    1     0  de:56:80:40:1d:ed    0
    2     0  42:d9:5d:6c:89:d8    0
```

4.9. Tests d'interconnexion entre réseaux

Depuis le système virtuel dans le VLAN Orange, on effectue une série de tests ICMP et HTTP.

1. Vers le système virtuel du VLAN Vert :

```
$ ping6 -c 2 fd6e:c073:b4a3:14:b8ad:caff:fefe:1
PING fd6e:c073:b4a3:14:b8ad:caff:fefe:1(fd6e:c073:b4a3:14:b8ad:caff:fefe:1) 56 data bytes
64 bytes from fd6e:c073:b4a3:14:b8ad:caff:fefe:1: icmp_seq=1 ttl=63 time=0.986 ms
64 bytes from fd6e:c073:b4a3:14:b8ad:caff:fefe:1: icmp_seq=2 ttl=63 time=1.10 ms

--- fd6e:c073:b4a3:14:b8ad:caff:fefe:1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.986/1.047/1.108/0.061 ms
```

2. Vers un hôte situé sur l'Internet :

```
$ ping6 -nc 2 www.iana.org
PING www.iana.org(2620:0:2d0:200::8) 56 data bytes
64 bytes from 2620:0:2d0:200::8: icmp_seq=1 ttl=51 time=183 ms
64 bytes from 2620:0:2d0:200::8: icmp_seq=2 ttl=51 time=182 ms

--- www.iana.org ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 182.993/183.289/183.585/0.296 ms
```

3. Chargement de page web.

```
$ wget -6 -O /dev/null http://inetdoc.net
--2015-10-21 14:03:05--  http://inetdoc.net/
Résolution de inetdoc.net (inetdoc.net)… 2a01:6600:8083:c300::1
Connexion à inetdoc.net (inetdoc.net)|2a01:6600:8083:c300::1|:80… connecté.
requête HTTP transmise, en attente de la réponse… 200 OK
Taille : 27201 (27K) [text/html]
Sauvegarde en : « /dev/null »

/dev/null          100%[=================>]  26,56K  --.-KB/s   ds 0,06s

2015-10-21 14:03:06 (416 KB/s) — « /dev/null » sauvegardé [27201/27201]
```

Enfin, il est possible de faire des tests de performances réseau avec l'outil iperf. Voici un échantillon de mesures entre les deux VLANs Orange et Vert.

- Serveur côté VLAN Vert :

```
etu@green-clnt:~$ iperf -w 320k -V -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size:  320 KByte
------------------------------------------------------------
[  4] local fd6e:c073:b4a3:14:b8ad:caff:fefe:1 port 5001 connected with
  fd6e:c073:b4a3:a:b8ad:caff:fefe:0 port 48032
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-30.0 sec  6.06 GBytes  1.74 Gbits/sec
```

- Client côté VLAN Orange :

```
etu@orange-clnt:~$ iperf -w 320k -P 8 -i 2 -t 30 -V \
 -c fd6e:c073:b4a3:14:b8ad:caff:fefe:1
------------------------------------------------------------
Client connecting to fd6e:c073:b4a3:14:b8ad:caff:fefe:1, TCP port 5001
TCP window size:  320 KByte
------------------------------------------------------------
[  3] local fd6e:c073:b4a3:a:b8ad:caff:fefe:0 port 48032 connected with
  fd6e:c073:b4a3:14:b8ad:caff:fefe:1 port 5001
[ ID] Interval       Transfer     Bandwidth
<snip/>
[  3]  0.0-30.0 sec  6.06 GBytes  1.74 Gbits/sec
```

5. Routage inter-VLAN dans un système virtuel

Dans ce second exemple, toutes les décisions d'acheminement du trafic sont prises dans le sous-système réseau d'une instance de système virtuel. Au niveau commutation de circuits, **Open vSwitch** joue le rôle important, tandis qu'au niveau commutation de paquets, la table de routage du sous-système réseau du routeur virtuel pointe vers les interfaces des différents VLANs. Si on se réfère au Schéma de la topologie, le travail de configuration porte sur les commutateurs `dsw-host`, `sw-lab` et sur une nouvelle instance de système virtuel. Voyons par quelles étapes il faut passer.

5.1. Plan d'adressage

On utilise à nouveau les mêmes préfixes réseau que ceux choisis pour la première maquette.

On ajoute dans le tableau ci-dessous le routeur virtuel dont les interfaces servent à acheminer le trafic entre les réseaux de la maquette.

Tableau 2. plan d'adressage des périmètres

Nom	VLAN	Interface	Préfixe réseau IP
Système hôte	trunk	vlan1	`192.0.2.0/26`
			`2001:db8:fe00:8175::/64`
Routeur virtuel	trunk	eth0	`192.0.2.0/26`
			`2001:db8:fe00:8175::/64`
Orange	10	eth0.10	`198.51.100.0/24`
			`fd6e:c073:b4a3:a::/64`
Vert	20	eth0.20	`203.0.113.0/24`
			`fd6e:c073:b4a3:14::/64`

5.2. Commutateurs et cordons de brassage

Relativement à la configuration de la première maquette, une partie de la configuration du commutateur `dsw-host` est conservée. L'interface `vlan1` reste l'interface principale du système hôte. C'est elle qui permet d'acheminer le trafic des machines virtuelles vers l'Internet.

En revanche, les cordons de brassage doivent tous être raccordés à un nouveau commutateur appelé `sw-lab`. Ce commutateur est lui-même cascadé au commutateur principal du système hôte `dsw-host`.

1. Rappel de la configuration du commutateur principal du système hôte.

```
$ sudo ovs-vsctl show
0fb1e80b-37cf-4dce-9a19-17b9fb989610
    Bridge dsw-host
        Port "eth0"
            Interface "eth0"
        Port "vlan1"
            Interface "vlan1"
                type: internal
        Port dsw-host
            Interface dsw-host
                type: internal
    ovs_version: "2.3.0"
```

2. Création du nouveau commutateur `sw-lab`.

```
$ sudo ovs-vsctl add-br sw-lab
```

3. Mise en place de la cascade (*stacking*) entre les deux commutateurs.

```
$ sudo ovs-vsctl add-port dsw-host patch2sw-lab -- \
 set interface patch2sw-lab type=patch options:peer=patch2dsw-host
$ sudo ovs-vsctl add-port sw-lab patch2dsw-host -- \
 set interface patch2dsw-host type=patch options:peer=patch2sw-lab
```

   Visualisation du résultat.

```
$ sudo ovs-vsctl show
0fb1e80b-37cf-4dce-9a19-17b9fb989610
    Bridge sw-lab
        Port "patch2dsw-host"
            Interface "patch2dsw-host"
                type: patch
                options: {peer="patch2sw-lab"}
        Port sw-lab
            Interface sw-lab
                type: internal
    Bridge dsw-host
        Port "eth0"
            Interface "eth0"
        Port "vlan1"
            Interface "vlan1"
                type: internal
        Port dsw-host
            Interface dsw-host
                type: internal
        Port "patch2sw-lab"
            Interface "patch2sw-lab"
                type: patch
                options: {peer="patch2dsw-host"}
    ovs_version: "2.3.0"
```

4. Création et activation des cordons de brassage des VLANs Orange et Vert. Les ports sur lesquels ces cordons sont raccordés sont placés en mode accès.

   VLAN Orange :

```
$ sudo ip tuntap add mode tap dev tap0 group kvm multi_queue
$ sudo ip link set dev tap0 up
$ sudo ovs-vsctl add-port sw-lab tap0 tag=10
$ sudo ovs-vsctl set port tap0 vlan_mode=access
```

   VLAN Vert :

```
$ sudo ip tuntap add mode tap dev tap1 group kvm multi_queue
$ sudo ip link set dev tap1 up
$ sudo ovs-vsctl add-port sw-lab tap1 tag=20
$ sudo ovs-vsctl set port tap1 vlan_mode=access
```

5. Création et activation du cordon de brassage vers le routeur virtuel. Le port du commutateur `sw-lab` sur lequel il est raccordé est configuré en mode ***trunk***. Le commutateur accepte de traiter des trames avec les étiquettes IEEE802.1Q dans ce mode.

```
$ sudo ip tuntap add mode tap dev tap2 group kvm multi_queue
$ sudo ip link set dev tap2 up
$ sudo ovs-vsctl add-port sw-lab tap2
$ sudo ovs-vsctl set port tap2 vlan_mode=trunk
```

5.3. Lancement des systèmes virtuels

Comme le routage du trafic se fait dans une instance de machine virtuelle, il faut lancer les instances dès maintenant avant de configurer les interfaces réseau. Cette opération se fait à l'aide du script suivant :

```
#!/bin/bash

../scripts/ovs-startup.sh orange.qed 1024 0
../scripts/ovs-startup.sh green.qed 1024 1
../scripts/ovs-startup.sh red.qed 1024 2
```

Le code du script de lancement d'une machine virtuelle raccordée à un commutateur Open vSwitch est donné dans le guide **Virtualisation système et enseignement**.

D'après le schéma de la topologie, la seule instance de machine virtuelle joignable depuis le système hôte est le routeur (avec le fichier image `red.qed`) dont l'adresse MAC est `ba:ad:ca:fe:00:02`. On peut le vérifier en visualisant les tables CAM des deux commutateurs puis la table du voisinage réseau du système hôte.

```
$ sudo ovs-appctl fdb/show dsw-host
 port  VLAN  MAC                Age
    8    10  ba:ad:ca:fe:00:00  221
    8    20  ba:ad:ca:fe:00:01   34
    8     0  ba:ad:ca:fe:00:02   13
    1     0  de:56:80:40:1d:ed    0
    2     0  42:d9:5d:6c:89:d8    0
```

```
$ sudo ovs-appctl fdb/show sw-lab
 port  VLAN  MAC                Age
    3    20  ba:ad:ca:fe:00:01   51
    1     0  de:56:80:40:1d:ed   31
    1     0  42:d9:5d:6c:89:d8   30
    4     0  ba:ad:ca:fe:00:02   30
    2    10  ba:ad:ca:fe:00:00    4
```

L'affichage des deux tables CAM montre que la base de données des VLANs est bien partagée entre commutateurs. La fonction de commutation de circuits est donc bien assurée.

```
$ ip neighbor ls
192.0.2.1 dev vlan1 lladdr de:56:80:40:1d:ed STALE
2001:db8:fe00:8175:21c:23ff:fed2:d105 dev vlan1 lladdr 00:1c:23:d2:d1:05 STALE
fe80::dc56:80ff:fe40:1ded dev vlan1 lladdr de:56:80:40:1d:ed router DELAY
2001:db8:fe00:8175::1 dev vlan1 lladdr de:56:80:40:1d:ed router REACHABLE
fe80::b8ad:caff:fefe:2 dev vlan1 lladdr ba:ad:ca:fe:00:02 STALE
```

L'affichage de la table du voisinage réseau du système hôte montre que seule l'interface `vlan1` est concernée et que seul le routeur virtuel est joignable.

Pour accéder au routeur virtuel, on utilise l'adresse IPv6 de lien local avec le protocole SSH. Cette adresse est indépendante de la configuration des interfaces de l'instance de système virtuel. Elle est configurée automatiquement dès que l'interface est active.

```
$ ssh etu@fe80::b8ad:caff:fefe:2%vlan1
Warning: Permanently added 'fe80::b8ad:caff:fefe:2%vlan1' (RSA) to the list of known hosts.
etu@fe80::b8ad:caff:fefe:2%vlan1's password:
No mail.
Last login: Sat Oct 24 09:44:54 2015 from fe80::40d9:5dff:fe6c:89d8%eth0
```

5.4. Création des interfaces de routage inter-VLAN

Une fois la partie commutation de circuits en place, on s'intéresse maintenant au routage des paquets dans le sous-système réseau du routeur virtuel. On utilise ici des sous-interfaces auxquelles on associe les étiquettes IEEE802.1Q.

1. On commence par la configuration de l'interface `eth0` à laquelle correspond le VLAN natif. On rappelle ici que le VLAN natif est celui auquel appartiennent toutes les trames Ethernet sans étiquette IEEE802.1Q qui circulent dans un ***trunk***. Or, on a vu ci-dessus que le canal de communication entre le commutateur `dsw-host` du système hôte et le routeur virtuel est justement en mode ***trunk***. Voici un extrait du fichier `/etc/network/interfaces` du routeur virtuel.

```
# The primary network interface
allow-hotplug eth0
iface eth0 inet static
 address 192.0.2.28/26
 gateway 192.0.2.1
 dns-nameservers 192.0.2.1

iface eth0 inet6 static
 address 2001:db8:fe00:8175::1c/64
 gateway 2001:db8:fe00:8175::1
 dns-nameservers 2001:db8:fe00:8175::1
```

2. Pour le VLAN Orange, on commence par créer la sous-interface `eth0.10` avant de lui affecter ses adresses IPv4 et IPv6.

   La commande manuelle de création de sous-interface est la suivante :

```
# ip link add link eth0 name eth0.10 type vlan id 10
```

   Il est aussi possible de placer directement les paramètres de configuration dans le fichier `/etc/network/interfaces` et de faire appel aux commandes **ifup** et **ifdown** ensuite. Voici l'extrait du fichier de configuration correspondant.

```
auto eth0.10
iface eth0.10 inet static
 hwaddress ba:ad:ca:fe:0a:02
 address 198.51.100.1/24

iface eth0.10 inet6 static
 address fd6e:c073:b4a3:a::1/64
```

   L'activation de la sous-interface produit le résultat suivtant :

```
root@red-rtr:~# ifup eth0.10
Waiting for DAD... Done
root@red-rtr:~# ip addr ls dev eth0.10
3: eth0.10@eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether ba:ad:ca:fe:0a:02 brd ff:ff:ff:ff:ff:ff
    inet 198.51.100.1/24 brd 198.51.100.255 scope global eth0.10
       valid_lft forever preferred_lft forever
    inet6 fd6e:c073:b4a3:a::1/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::b8ad:caff:fefe:2/64 scope link
       valid_lft forever preferred_lft forever
```

   Note

   On a fixé manuellement l'adresse MAC de la sous-interface dans le but de l'identifier plus facilement par la suite lors des échanges avec les autres systèmes virtuels.

3. On procède exactement de la même façon pour le VLAN Vert que pour le VLAN Orange.

   La commande manuelle de création de sous-interface est la suivante :

```
# ip link add link eth0 name eth0.20 type vlan id 20
```

   Voici l'extrait du fichier de configuration `/etc/network/interfaces` pour la sous-interface `eth0.20`.

```
auto eth0.20
iface eth0.20 inet static
 hwaddress ba:ad:ca:fe:14:02
 address 203.0.113.1/24

iface eth0.20 inet6 static
 address fd6e:c073:b4a3:14::1/64
```

   L'activation de la sous-interface produit le résultat suivant :

```
root@red-rtr:~# ifup eth0.20
Waiting for DAD... Done
root@red-rtr:~# ip addr ls dev eth0.20
4: eth0.20@eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether ba:ad:ca:fe:14:02 brd ff:ff:ff:ff:ff:ff
    inet 203.0.113.1/24 brd 203.0.113.255 scope global eth0.20
       valid_lft forever preferred_lft forever
    inet6 fd6e:c073:b4a3:14::1/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::b8ad:caff:fefe:2/64 scope link
       valid_lft forever preferred_lft forever
```

5.5. Activation du routage sur le routeur virtuel

Cette section est identique à celle sur la première maquette.

On reprend ici la liste des paramètres du fichier `/etc/systcl` pour activer le routage IPv4 ainsi que le routage IPv6.

```
# egrep -v '(^#|^$)' /etc/sysctl.conf
net.ipv4.conf.default.rp_filter=1
net.ipv4.conf.all.rp_filter=1
net.ipv4.ip_forward=1
net.ipv6.conf.all.forwarding=1
net.ipv4.conf.all.log_martians=1
```

Cette liste de paramètres n'est effective qu'après utilisation de l'instruction `# sysctl -p`.

5.6. Autoconfiguration des hôtes des VLANs Orange et Vert

Pour la partie IPv4, on fait appel à un démon DHCP sur le routeur virtuel. On place ce démon en écoute sur les deux sous-interfaces `eth0.10` et `eth0.20`.

Après avoir installé le paquet du serveur DHCP, on restreint son usage aux deux VLANs de la maquette en éditant le fichier `/etc/default/isc-dhcp-server`. On obtient les résultats suivants.

```
# aptitude search ~idhcp-server
i   isc-dhcp-server      - ISC DHCP server for automatic IP address assignment

# egrep -v '(^#|^$)' /etc/default/isc-dhcp-server
INTERFACES="eth0.10 eth0.20"
```

Côté configuration, on applique les éléments suivants dans le fichier `/etc/dhcp/dhcpd.conf`.

```
ddns-update-style none;

default-lease-time 600;
max-lease-time 7200;

authoritative;

log-facility local7;

subnet 198.51.100.0 netmask 255.255.255.0 {
  range 198.51.100.10 198.51.100.30;
  option domain-name-servers 192.0.2.1;
  option routers 198.51.100.1;
  option broadcast-address 198.51.100.255;
}

subnet 203.0.113.0 netmask 255.255.255.0 {
  range 203.0.113.10 203.0.113.30;
  option domain-name-servers 192.0.2.1;
  option routers 203.0.113.1;
  option broadcast-address 203.0.113.255;
}
```

Pour la partie IPv6, on fait appel au paquet radvd pour l'autoconfiguration des hôtes en mode SLAAC (Document RFC4862 **IPv6 Stateless Address Autoconfiguration**). On vérifie que le paquet est bien installé.

```
# aptitude search ~iradvd
i   radvd                - Démon d'information de routeur
```

Voici ensuite une copie du fichier `/etc/radvd.conf` du routeur virtuel.

```
interface eth0.10
{
   AdvSendAdvert on;
   prefix fd6e:c073:b4a3:a::/64
   {
       AdvOnLink on;
       AdvAutonomous on;
       AdvRouterAddr on;
   };

   RDNSS 2001:db8:fe00:8175::1
```

```
    {
    };
};

interface eth0.20
{
    AdvSendAdvert on;
    prefix fd6e:c073:b4a3:14::/64
    {
        AdvOnLink on;
        AdvAutonomous on;
        AdvRouterAddr on;
    };

    RDNSS 2001:db8:fe00:8175::1
    {
    };
};
```

5.7. Affichage des tables de routage du routeur virtuel

Les tables de routage font apparaître les correspondances entre les préfixes réseau définis plus haut et les interfaces configurées sur l'instance de routeur virtuel.

La table de routage IPv4 est :

```
$ ip route ls
default via 192.0.2.1 dev eth0
192.0.2.0/26 dev eth0  proto kernel  scope link  src 192.0.2.29
198.51.100.0/24 dev eth0.10  proto kernel  scope link  src 198.51.100.1
203.0.113.0/24 dev eth0.20  proto kernel  scope link  src 203.0.113.1
```

La table de routage IPv6 est :

```
$ ip -6 route ls
2001:db8:fe00:8175::/64 dev eth0  proto kernel  metric 256  pref medium
fd6e:c073:b4a3:a::/64 dev eth0.10  proto kernel  metric 256  pref medium
fd6e:c073:b4a3:14::/64 dev eth0.20  proto kernel  metric 256  pref medium
fe80::/64 dev eth0  proto kernel  metric 256  pref medium
fe80::/64 dev eth0.20  proto kernel  metric 256  pref medium
fe80::/64 dev eth0.10  proto kernel  metric 256  pref medium
default via 2001:db8:fe00:8175::1 dev eth0  metric 1024  pref medium
```

5.8. Traduction d'adresses source sur le routeur virtuel

Relativement à la première maquette sur laquelle la traduction des adresses source a lieu sur le système hôte, seule l'interface change.

Les VLANs Orange et Vert utilisent les mêmes préfixes réseau non routables sur l'Internet.

Voici une copie des fichiers de règles pour les deux protocoles de couche réseau : IPv4 et IPv6.

```
#
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
COMMIT
#
*nat
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
-A POSTROUTING -p tcp -m tcp --syn -m tcpmss --mss 1400:1536 -j TCPMSS --clamp-mss-to-pmtu
-A POSTROUTING -o eth0 -j SNAT --to-source 192.0.2.28
COMMIT
```

```
#
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
COMMIT
#
*nat
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
-A POSTROUTING -p tcp -m tcp --syn -m tcpmss --mss 1400:1536 -j TCPMSS --clamp-mss-to-pmtu
-A POSTROUTING -o eth0 -s fd00::/8 -j SNAT --to-source 2001:db8:fe00:8175::1c
COMMIT
```

Il faut à nouveau «surveiller» les compteurs associés à chaque règle pour en mesurer l'utilisation. À partir des deux jeux de règles donnés ci-dessus, on obtient les résultats suivants.

```
# iptables -t nat -vnL POSTROUTING
Chain POSTROUTING (policy ACCEPT 2 packets, 96 bytes)
 pkts bytes target     prot opt in  out     source     destination
    8   448 TCPMSS     tcp  --  *   *       0.0.0.0/0  0.0.0.0/0   tcp flags:0x17/0x02 tcpmss match 1400:1536 TCPMSS clamp to PMTU
  237 15186 SNAT       all  --  *   eth0    0.0.0.0/0  0.0.0.0/0   to:192.0.2.28
```

```
# ip6tables -t nat -vnL POSTROUTING
Chain POSTROUTING (policy ACCEPT 136 packets, 11260 bytes)
 pkts bytes target     prot opt in  out     source     destination
   46  3680 TCPMSS     tcp      *   *       ::/0       ::/0        tcp flags:0x17/0x02 tcpmss match 1400:1536 TCPMSS clamp to PMTU
    3   251 SNAT       all      *   eth0    fd00::/8   ::/0        to:2001:db8:fe00:8175::1c
```

5.9. Tests d'interconnexion entre réseaux

Les résultats des tests d'interconnexion réseau entre les VLANs Orange et Vert ainsi que vers l'Internet sont identiques à ceux obtenus avec la première maquette. Il est donc inutile de reproduire les mêmes copies d'écran ici.

En revanche, les tests de performances diffèrent. Les performances sont nettement moins bonnes.

Voici un échantillon de mesures iperf entre les deux VLANs Orange et Vert.

- Serveur côté VLAN Vert :

```
etu@green-clnt:~$ iperf -w 320k -V -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size:  320 KByte
------------------------------------------------------------
[  4] local fd6e:c073:b4a3:14:b8ad:caff:fefe:1 port 5001 connected with
  fd6e:c073:b4a3:a:b8ad:caff:fefe:0 port 37210
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-30.0 sec  2.98 GBytes   852 Mbits/sec
```

- Client côté VLAN Orange :

```
etu@orange-clnt:~$ iperf -w 320k -P 8 -i 2 -t 30 -V \
 -c fd6e:c073:b4a3:14:b8ad:caff:fefe:1
------------------------------------------------------------
Client connecting to fd6e:c073:b4a3:14:b8ad:caff:fefe:1, TCP port 5001
TCP window size:  320 KByte
------------------------------------------------------------
[  3] local fd6e:c073:b4a3:a:b8ad:caff:fefe:0 port 37210 connected with
  fd6e:c073:b4a3:14:b8ad:caff:fefe:1 port 5001
[ ID] Interval       Transfer     Bandwidth
<snip/>
[  3]  0.0-30.0 sec  2.98 GBytes   853 Mbits/sec
```

Il apparaît clairement que le routage dans le sous-système réseau du système hôte est nettement plus performant que le routage dans une instance de système virtuel.

6. Applications pratiques

Le routage inter-VLAN est utilisé à plusieurs niveaux dans les documents du site **inetdoc**.

Introduction au routage inter-VLAN

Le support de travaux pratiques **Routage inter-VLAN dans un contexte IaaS** est une introduction aux opérations de configuration. Il est basé sur une séparation entre la fonction de routage au niveau réseau assurée par un système GNU/Linux et la fonction de commutation des trames au niveau liaison assurée par un commutateur Cisco.

Introduction au routage dynamique avec OSPF

Le support de travaux pratique **Routage dynamique avec OSPF (Bird)** est une excellente occasion de caractériser l'indépendance entre topologie logique et topologie physique. En effet, la topologie physique est de type étoile puisque l'on utilise des connexions filaires cuivre Ethernet alors que la topologie logique est de type triangle grâce à l'utilisation des VLANs.