Manuel de Travaux Pratiques

Module « Interconnexion LAN/WAN »

Philippe Latu
philippe.latu(at)inetdoc.net

https://www.inetdoc.net

Résumé

Ce document présente la série de travaux pratiques du module sur l'interconnexion des réseaux locaux et étendus en première année de *Master mention Réseaux et télécommunication* de l'Université Paul Sabatier.

Il met l'accent sur les aspects opérationnels (*Ops*) du paradigme *DevOps* dans un contexte « cloud privé ». Il aborde en détail la configuration et la gestion d'une infrastructure réseau virtualisée, illustrant les concepts clés du routage inter-VLAN et de la conteneurisation.

Les manipulations couvrent la mise en place d'une topologie réseau complexe, incluant des réseaux virtuels (VLANs) et des conteneurs. Les étudiants apprennent à configurer des commutateurs virtuels, des interfaces réseau, et à activer le routage sur des systèmes Linux. Ces compétences sont essentielles pour les opérations quotidiennes dans un environnement *DevOps* moderne.

Une attention particulière est portée à la sécurité réseau, avec l'introduction de techniques de filtrage et de traduction d'adresses (NAT). Les participants configurent des règles de pare-feu à l'aide de *nftables*, renforçant ainsi leurs compétences en matière de sécurisation des infrastructures.

Le document guide également les étudiants dans l'installation et la configuration d'Incus, un gestionnaire de conteneurs orienté *Infrastructure as Code* (IaC). Cette partie souligne l'importance de la conteneurisation dans les pratiques *DevOps* actuelles, permettant aux apprenants de comprendre comment déployer et gérer des applications conteneurisées à l'aide de scripts.

Les travaux pratiques incluent des exercices sur l'automatisation des tâches d'administration système, tels que la mise à jour en masse de conteneurs. Ces compétences en *scripting* Bash et en automatisation sont fondamentales dans l'approche *DevOps*, où l'efficacité opérationnelle est primordiale.

Le document aborde également des sujets avancés comme le routage dynamique avec OSPF et l'interconnexion IPv4/IPv6, préparant les étudiants à gérer des infrastructures réseau complexes et évolutives. Ces compétences sont essentielles pour les opérations à grande échelle dans un environnement *DevOps*.

L'objectif de toutes ces manipulations pratiques est d'offrir une formation complète sur les aspects opérationnels du *DevOps*, couvrant la configuration réseau, la virtualisation, la conteneurisation, la sécurité, et l'automatisation. Il s'agit avant tout de préparer les étudiants aux défis quotidiens rencontrés dans la gestion d'infrastructures modernes et agiles.

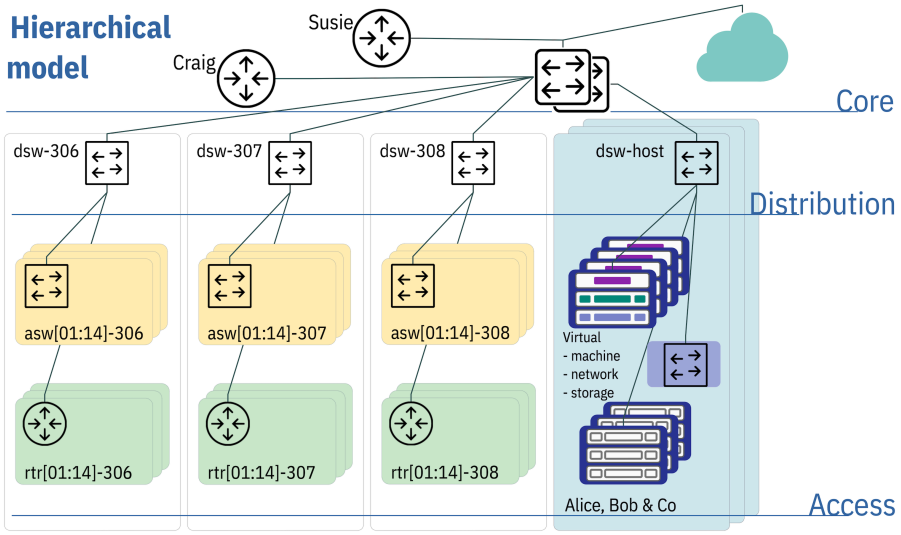


Table des matières

Routage inter-VLAN dans un contexte IaaS

https://www.inetdoc.net

Résumé

Le routage inter-VLAN est très largement utilisé dans l'interconnexion entre les réseaux Ethernet. Les manipulations présentées dans ces travaux pratiques illustrent l'interconnexion entre un réseau appartenant à une infrastructure de type IaaS (*Infrastructure as a Service*) et un réseau d'hébergement de conteneurs Incus raccordés à l'aide de la technologie `macvlan`.

On introduit aussi un premier niveau de filtrage qui assure la traduction d'adresses entre les deux réseaux interconnectés.

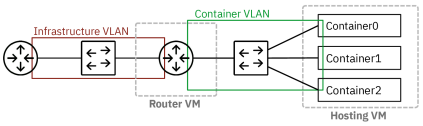


Table des matières

1. Objectifs

Après avoir réalisé les manipulations présentées dans ce document, les étudiants seront capables de :

1. Configurer le routage inter-VLAN sur un système GNU/Linux avec l'activation du routage et la configuration des interfaces réseau.
2. Mettre en place et configurer un service de traduction d'adresses (NAT) pour permettre aux réseaux internes d'accéder à Internet.
3. Installer et configurer le gestionnaire de conteneurs Incus pour créer et gérer des conteneurs système.
4. Configurer l'adressage automatique (DHCP et SLAAC) pour un réseau de conteneurs à l'aide de dnsmasq.
5. Automatiser des tâches de gestion de conteneurs à l'aide de scripts Bash, comme l'exécution de commandes dans plusieurs conteneurs ou la modification de leur configuration réseau.

2. Topologies logiques et virtuelles

Les définitions importantes sur les réseaux locaux virtuels et le routage associé sont présentées dans l'article *Routage Inter-VLAN*

On rappelle simplement que la notion de réseau local virtuel ou VLAN permet de constituer des groupes logiques dans les réseaux Ethernet au niveau liaison de la modélisation. Lors du raccordement entre les équipements (commutateurs, routeurs, serveurs), certaines liaisons doivent véhiculer le trafic de plusieurs réseaux locaux virtuels (VLANs). Ces liaisons sont baptisées ***trunks*** dans le jargon. Pour distinguer le trafic appartenant à chaque réseau local, on ajoute à la trame une balise définie par le standard IEEE 802.1Q. C'est cette étiquetage de trame qui permet la distribution des domaines de diffusion entre plusieurs équipements physiques distincts.

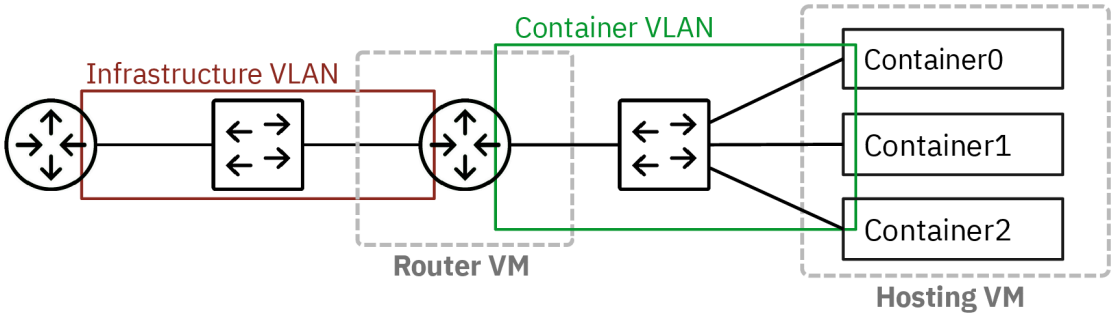
On atteint ainsi un objectif très important. Il est possible de concevoir une topologie logique de réseau totalement indépendante de la topologie physique.

Réseau virtuel ou pas, il ne faut pas oublier les éléments suivants sur la segmentation des réseaux locaux.

- Une interface de ***commutateur*** délimite un domaine de ***collision***.

- Une interface de ***routeur*** délimite à la fois un domaine de ***collision*** et un domaine de ***diffusion***.

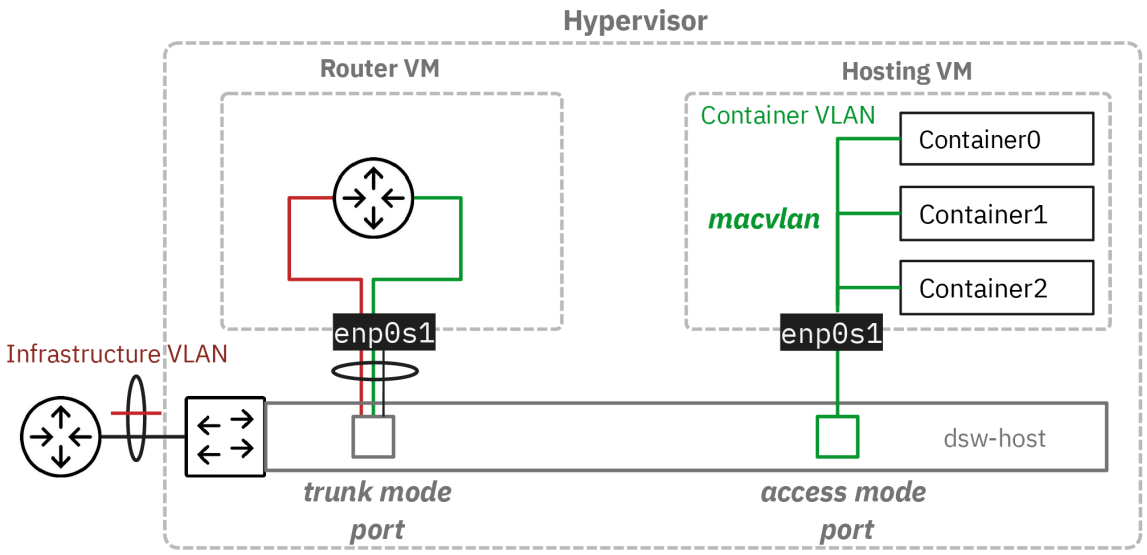
La représentation de la topologie logique ci-dessous montre que le routeur de couleur verte assure l'interconnexion entre un réseau d'infrastructure appelé *Hosting VLAN* et un réseau de conteneurs appelé *Container VLAN*. Les deux rectangles en gris “matérialisent“ les machines virtuelles qui sont utilisées pour les manipulations.



Topologie logique

La représentation de la topologie vue sous l'angle de l'hébergement sur un système hôte hyperviseur montre que les deux VLANs sont présents sur le commutateur virtuel de couche distribution appelé *dsw-host*. Ce commutateur appartient au système hôte. Il assure le raccordement entre les réseaux physiques et virtualisés. On retrouve le routeur de couleur verte raccordé avec un lien unique sur lequel le trafic des deux VLANs doit transiter.

Côté conteneurs, seule l'interface `enp0s1` est raccordé via un lien en mode accès. La technologie macVLAN permet d'utiliser plusieurs adresses MAC sur une même interface réseau.



Topologie hébergée

Tableau 1. Plan d'adressage de la maquette « Routage inter-VLAN dans un contexte IaaS »

Réseau	Numéro VLAN	Adresses de passerelles
Hébergement VLAN rouge	360	`192.168.104.129/29` `2001:678:3fc:168::1/64`
Services VLAN vert	440	`192.0.2.1/24` `fda0:7a62:1b8::1/64`

3. Raccordement au commutateur de distribution

Dans cette section, on étudie le raccordement des deux machines virtuelles au commutateur de distribution sur le système hôte.

Q1. Comment contrôler la configuration des ports du commutateur de distribution sur le système hôte ?

Le commutateur virtuel implanté sur le système hôte est géré par ***Open vSwitch***. On fait donc appel à la commande ovs-vsctl pour accéder aux paramètres de la configuration des ports.

- Pour le port nommé `tap200`, on obtient le paramètre `vlan_mode` avec l'instruction :

```
sudo ovs-vsctl get port tap200 vlan_mode
trunk
```

Le mode `trunk` correspond à un canal de transmission unique dans lequel circule le trafic de plusieurs domaines de diffusion ou VLANs.

- Pour le port nommé `tap2`, on obtient la valeur `access` pour le même paramètre :

```
sudo ovs-vsctl get port tap2 vlan_mode
access
```

Ici, le mode `access` correspond à un canal de transmission dans lequel circule le trafic d'un seul et unique domaine de diffusion ou VLAN.

Q2. Comment afficher le numéro de VLAN attribué au port en mode accès du commutateur de distribution sur le système hôte ?

On reprend la même commande que dans la question précédente avec le mot clé `tag`.

```
sudo ovs-vsctl get port tap2 tag
20
```

Q3. Comment affecter le numéro de VLAN attribué au port en mode accès du commutateur de distribution sur le système hôte ?

On reprend à nouveau la même commande avec l'option `set`.

```
sudo ovs-vsctl set port tap2 tag=440
```

Les valeurs données dans l'exemple ci-dessus sont à changer suivant les attributions du plan d'adressage des réseaux d'hébergement et de conteneurs.

Q4. Comment configurer les ports du commutateur avant le lancement des machines virtuelles ?

On utilise le script de procédure `switch-conf.py` qui applique les déclarations contenues dans un fichier YAML. Le code du script est accessible à partir du dépôt Git *startup-scripts*.

Voici une copie du fichier de configuration des deux ports de commutateur.

```
ovs:
  switches:
    - name: dsw-host
      ports:
        - name: tap5 # Router port
          type: OVSPort
          vlan_mode: trunk
          trunks: [360, 440]
        - name: tap6 # Container hosting VM
          type: OVSPort
          vlan_mode: access
          tag: 440
```

On applique les paramètres définis ci-dessus.

```
$HOME/masters/scripts/switch-conf.py switch.yaml
```

On obtient les résultats suivants.

```
---------------------------------------
Configuring switch dsw-host
>> Port tap5 vlan_mode set to trunk
>> Port tap5 trunks set to [360, 440]
>> Port tap6 vlan_mode is already set to access
>> Port tap6 tag set to 440
---------------------------------------
```

Les numéros de port de commutateur et de VLAN donnés dans les exemples ci-dessus sont à changer suivant le contexte.

Q5. Comment lancer les machines virtuelles associées aux rôles routeur et hébergement de conteneurs ?

On utilise le script de procédure `lab-startup.py` qui applique les déclarations contenues dans un fichier YAML. Le code du script est accessible à partir du dépôt Git *startup-scripts*.

Voici une copie du fichier de déclaration des deux machines virtuelles.

```
kvm:
  vms:
    - vm_name: router
      master_image: debian-testing-amd64.qcow2 # master image to be used
      force_copy: false # do not force copy the master image to the VM image
      memory: 1024
      tapnum: 5
    - vm_name: hosting
      master_image: debian-testing-amd64.qcow2 # master image to be used
      force_copy: false # do not force copy the master image to the VM image
      memory: 1024
      tapnum: 6
```

On lance les deux machines virtuelles avec le script lab-startup.py.

```
$HOME/masters/scripts/lab-startup.py lab1.yaml
```

```
Copying /home/etudianttest/masters/debian-testing-amd64.qcow2 to router.qcow2...done
Creating OVMF_CODE.fd symlink...
Creating router_OVMF_VARS.fd file...
Starting router...
~> Virtual machine filename   : router.qcow2
~> RAM size                   : 1024MB
~> SPICE VDI port number      : 5905
~> telnet console port number : 2305
~> MAC address                : b8:ad:ca:fe:00:05
~> Switch port interface      : tap5, trunk mode
~> IPv6 LL address            : fe80::baad:caff:fefe:5%dsw-host
router started!
Copying /home/etudianttest/masters/debian-testing-amd64.qcow2 to hosting.qcow2...done
Creating hosting_OVMF_VARS.fd file...
Starting hosting...
~> Virtual machine filename   : hosting.qcow2
~> RAM size                   : 1024MB
~> SPICE VDI port number      : 5906
~> telnet console port number : 2306
~> MAC address                : b8:ad:ca:fe:00:06
~> Switch port interface      : tap6, access mode
~> IPv6 LL address            : fe80::baad:caff:fefe:6%vlan440
hosting started!
```

Les deux machines virtuelles sont maintenant disponibles pour la suite des manipulations.

4. Rôle routeur

Dans cette section, on étudie la machine virtuelle qui joue le rôle de routeur entre le réseau d'hébergement et un réseau de conteneurs. Pour traiter les questions, il est nécessaire de mettre en œuvre une maquette avec un adressage indépendant. Voici les choix effectués pour la maquette.

4.1. Configuration des interfaces du routeur

Une fois la machine virtuelle routeur lancée, les premières étapes consistent à lui attribuer un nouveau nom et à configurer les interfaces réseau pour joindre les hôtes voisins.

Q6. Comment changer le nom de la machine virtuelle ?

Il faut éditer le fichier `/etc/hostname` en remplaçant le nom local par le nom voulu. Il est ensuite nécessaire de redémarrer pour que le nouveau nom soit pris en compte par tous les outils du système.

```
etu@localhost:~$ sudo hostnamectl hostname router
etu@vm0:~$ sudo reboot
```

Q7. Comment appliquer les configurations réseau IPv4 et IPv6 à partir de l'unique interface du routeur ?

Consulter la documentation de ***Netplan*** pour obtenir les informations sur la configuration des interfaces réseau à l'adresse *Netplan documentation*.

Il existe plusieurs possibilités pour configurer une interface réseau. Dans le contexte de ces manipulations, on utilise ***Netplan*** dans le but de séparer la partie déclarative du moteur de configuration.

C'est `systemd-networkd` qui joue le rôle de moteur de configuration sur les machines virtuelles utilisées avec ces manipulations.

La configuration de base fournie avec l'image maître suppose que l'interface obtienne un bail DHCP pour la partie IPv4 et une configuration automatique via SLAAC pour la partie IPv6. Cette configuration par défaut doit être éditée et remplacée. Il faut configurer trois interfaces.

- L'interface principale correspond à l'interface "physique" de la machine. Elle est nommée `enp0s1` en fonction de l'ordre des adresses des composants raccordés au bus PCI.

- Une sous-interface doit être créée pour le réseau d'hébergement avec le numéro de VLAN désigné dans le plan d'adressage des réseaux d'hébergement et de conteneurs. Cette interface doit désigner les passerelles IPv4 et IPv6 de façon à joindre l'Internet.

- Une sous-interface doit être créée pour le réseau des conteneurs avec, là encore, le bon numéro de VLAN. Les adresses IPv4 et IPv6 de cette interface deviendront les passerelles du serveur et des conteneurs.

Voici une copie du fichier `/etc/netplan/enp0s1.yaml` de la maquette.

```
network:
    version: 2
    ethernets:
      enp0s1:
        dhcp4: false
        dhcp6: false
        accept-ra: false
        nameservers:
          addresses:
            - 172.16.0.2
            - 2001:678:3fc:3::2

    vlans:
      enp0s1.360:
        id: 360
        link: enp0s1
        addresses:
          - 192.168.104.130/29
          - 2001:678:3fc:168::82/64
        routes:
          - to: default
            via: 192.168.104.129
          - to: "::/0"
            via: "fe80::168:1"
            on-link: true
      enp0s1.440:
        id: 440
        link: enp0s1
        addresses:
          - 192.0.2.1/24
          - fda0:7a62:1b8::1/64
```

Une fois le fichier de configuration en place, il suffit de faire appel à la commande netplan pour appliquer les changements.

```
sudo netplan apply
```

Pour vérifier que l'adressage réseau est correct, on dispose de plusieurs solutions. Voici un exemple avec la commande networkctl qui synthétise l'ensemble de la configuration réseau.

```
networkctl status
```

```
# Interfaces: 1, 2, 3, 4
    State: routable
Online state: online
  Address: 192.168.104.130 on enp0s1.360
           192.0.2.1 on enp0s1.440
           2001:678:3fc:168::82 on enp0s1.360
           2001:678:3fc:168:baad:caff:fefe:5 on enp0s1.360
           fda0:7a62:1b8::1 on enp0s1.440
           fe80::baad:caff:fefe:5 on enp0s1
           fe80::baad:caff:fefe:5 on enp0s1.360
           fe80::baad:caff:fefe:5 on enp0s1.440
  Gateway: 192.168.104.129 on enp0s1.360
           fe80:168::1 on enp0s1.360
      DNS: 172.16.0.2
           2001:678:3fc:3::2
```

Q8. Quels sont les tests de connectivité réalisables après application de la nouvelle configuration des interfaces réseau ?

Relever l'état des trois interfaces et procédez aux tests ICMP et DNS en respectant l'ordre des couches de la modélisation.

Sans la confirmation que la configuration du serveur de conteneurs est prête, c'est du côté hébergement et accès Internet qu'il faut orienter les tests. Classiquement, on cherche à joindre la passerelle en premier puis l'Internet ensuite via des requêtes ICMP. Enfin, on effectue un test de couche application avec une requête DNS.

```
ping -q -c2 192.168.104.129
PING 192.168.104.129 (192.168.104.129) 56(84) bytes of data.

--- 192.168.104.129 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 1.501/1.516/1.531/0.015 ms
```

```
PING 9.9.9.9 (9.9.9.9) 56(84) bytes of data.

--- 9.9.9.9 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 49.304/52.098/54.892/2.794 ms
```

```
ping -q -c2 fe80:168::1%enp0s1.360
PING fe80:168::1%enp0s1.360(fe80:168::1%enp0s1.360) 56 data bytes

--- fe80:168::1%enp0s1.360 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 1.459/13.305/25.152/11.846 ms
```

```
ping -q -c2 2620:fe::fe
PING 2620:fe::fe(2620:fe::fe) 56 data bytes

--- 2620:fe::fe ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 41.812/42.437/43.063/0.625 ms
```

```
host quad9.net
quad9.net has address 216.21.3.77
quad9.net has IPv6 address 2620:0:871:9000::77
quad9.net mail is handled by 5 mx1.quad9.net.
quad9.net mail is handled by 20 mx2.quad9.net.
quad9.net mail is handled by 100 keriomail.pch.net.
```

4.2. Activation de la fonction routage

Sans modification de la configuration par défaut, un système GNU/Linux n'assure pas la fonction de routage du trafic d'une interface réseau à une autre.

L'activation du routage correspond à un réglage de paramètres du sous-système réseau du noyau Linux. L'outil qui permet de consulter et modifier les réglages de paramètre sur le noyau est appelé sysctl.

Q9. Comment activer le routage dans le sous-système réseau du noyau Linux ?

Utiliser la commande sysctl pour effectuer des recherches et identifier les paramètres utiles. Par exemple :

```
sudo sysctl -a -r ".*forward.*"
```

Il est dorénavant recommandé de créer un fichier de configuration spécifique par fonction. C'est la raison pour laquelle on crée un nouveau fichier `/etc/sysctl.d/10-routing.conf`.

Attention ! Il ne faut pas oublier d'appliquer les nouvelles valeurs des paramètres de configuration.

```
cat << EOF | sudo tee /etc/sysctl.d/10-routing.conf
net.ipv4.ip_forward=1
net.ipv6.conf.all.forwarding=1
net.ipv4.conf.all.log_martians=1
EOF
```

Voici un exemple des résultats obtenus après application des nouveaux paramètres.

```
sudo sysctl --system
```

```
* Applique /usr/lib/sysctl.d/10-coredump-debian.conf …
* Applique /etc/sysctl.d/10-routing.conf …
* Applique /usr/lib/sysctl.d/50-default.conf …
* Applique /usr/lib/sysctl.d/50-pid-max.conf …
* Applique /etc/sysctl.conf …
kernel.core_pattern = core
net.ipv4.ip_forward = 1
net.ipv6.conf.all.forwarding = 1
net.ipv4.conf.all.log_martians = 1
kernel.sysrq = 0x01b6
kernel.core_uses_pid = 1
net.ipv4.conf.default.rp_filter = 2
net.ipv4.conf.enp0s1/360.rp_filter = 2
net.ipv4.conf.enp0s1/440.rp_filter = 2
net.ipv4.conf.enp0s1.rp_filter = 2
net.ipv4.conf.lo.rp_filter = 2
net.ipv4.conf.default.accept_source_route = 0
net.ipv4.conf.enp0s1/360.accept_source_route = 0
net.ipv4.conf.enp0s1/440.accept_source_route = 0
net.ipv4.conf.enp0s1.accept_source_route = 0
net.ipv4.conf.lo.accept_source_route = 0
net.ipv4.conf.default.promote_secondaries = 1
net.ipv4.conf.enp0s1/360.promote_secondaries = 1
net.ipv4.conf.enp0s1/440.promote_secondaries = 1
net.ipv4.conf.enp0s1.promote_secondaries = 1
net.ipv4.conf.lo.promote_secondaries = 1
net.ipv4.ping_group_range = 0 2147483647
net.core.default_qdisc = fq_codel
fs.protected_hardlinks = 1
fs.protected_symlinks = 1
fs.protected_regular = 2
fs.protected_fifos = 1
kernel.pid_max = 4194304
```

Q10. Quelles sont les conditions à réunir pour tester le fonctionnement du routage ?

Rechercher comment utiliser l'analyseur réseau tshark pour caractériser l'acheminement du trafic d'un réseau à l'autre.

Le plan d'adressage prévoit d'utiliser des préfixes ayant une portée locale pour les réseaux de conteneurs. Il n'est donc pas possible de passer par une requête ICMP pour caractériser l'accès aux réseaux distants. En effet, l'adresse source n'est pas reconnue par l'hôte distant et les routeurs de l'Internet ne disposent d'aucune solution pour joindre le réseau des conteneurs.

Voici un extrait de capture qui montre que le serveur de conteneur cherche à joindre un hôte sur l'Internet sans succès. Cette capture étant réalisée sur l'interface réseau côté hébergement, elle montre que le trafic est bien acheminé d'un réseau à l'autre.

```
tshark -i enp0s1.360
```

```
Capturing on 'enp0s1.360'
    1 0.000000000    192.0.2.2 → 9.9.9.9      DNS 81 Standard query 0xbdab A 1.debian.pool.ntp.org
    2 0.000056361    192.0.2.2 → 9.9.9.9      DNS 81 Standard query 0xab92 AAAA 1.debian.pool.ntp.org
```

4.3. Activation de la traduction d'adresses

Le résultat de la question ci-dessus montre que les hôtes situés dans le réseau des conteneurs ne peuvent pas joindre l'Internet puisque les préfixes réseau utilisés ont une portée limitée.

Q11. Quels sont les paquets qui fournissent les outils de gestion de la traduction d'adresses ?

Rechercher les paquets relatifs au filtrage et à la gestion des règles de pare-feux.

Dans le contexte des ces manipulations, nous utilisons `nftables` comme outil de gestion du filtrage.

C'est la partie outils de l'espace utilisateur qui nous intéresse ici.

```
apt search ^nftables$
nftables/testing,now 1.1.0-2 amd64  [installé]
  programme de contrôle des règles de filtrage de paquets du projet Netfilter
```

```
sudo apt -y install nftables
```

Q12. Quelles sont les règles à appliquer pour assurer une traduction des adresses sources en sortie sur le réseau d'infrastructure (VLAN rouge) ?

Rechercher dans des exemples de configuration `nftables` avec la fonction `MASQUERADE`.

Voici un exemple de création du fichier `/etc/nftables.conf` avec le jeu d'instructions qui assure la traduction d'adresses sources pour IPv4 et IPv6.

```
cat << EOF | sudo tee /etc/nftables.conf
#!/usr/sbin/nft -f

flush ruleset

table inet nat {
    chain postrouting {
        type nat hook postrouting priority 100;
        oifname "enp0s1.360" counter packets 0 bytes 0 masquerade
    }
}
EOF
```

Avertissement

Il faut impérativement changer le nom d'interface en utilisant le numéro de VLAN attribué dans le plan d'adressage des travaux pratiques.

La création de ce fichier de règles n'est pas suffisante. Il faut appliquer les règles contenues dans le fichier.

```
sudo nft -f /etc/nftables.conf
```

Q13. Comment rendre le chargement des règles de filtrage automatique au redémarrage du système ?

Afficher l'état du service `nftables.service`. Activer ce service si celui est à l'état désactivé (***disabled***).

Pour afficher l'état du service, on utilise la commande suivante.

```
systemctl status nftables.service
```

```
# nftables.service - nftables
     Loaded: loaded (/usr/lib/systemd/system/nftables.service; disabled; preset: enabled)
     Active: inactive (dead)
       Docs: man:nft(8)
             http://wiki.nftables.org
```

On constate qu'il faut activer ce service pour assurer le chargement automatique des règles de filtrage au démarrage.

```
sudo systemctl enable nftables.service
sudo systemctl start nftables.service
sudo systemctl status nftables.service
```

```
# nftables.service - nftables
     Loaded: loaded (/usr/lib/systemd/system/nftables.service; enabled; preset: enabled)
     Active: active (exited) since Sat 2024-09-14 18:35:00 CEST; 7s ago
 Invocation: 1dcc395a33c74606bd7dab7f33b90787
       Docs: man:nft(8)
             http://wiki.nftables.org
    Process: 729 ExecStart=/usr/sbin/nft -f /etc/nftables.conf (code=exited, status=0/SUCCESS)
   Main PID: 729 (code=exited, status=0/SUCCESS)
   Mem peak: 5.9M
        CPU: 44ms

sept. 14 18:34:59 router systemd[1]: Starting nftables.service - nftables...
sept. 14 18:35:00 router systemd[1]: Finished nftables.service - nftables.
```

Q14. Comment caractériser le fonctionnement de la traduction d'adresses sources ?

Rechercher dans les pages de manuel de la commande nftables les options d'affichage du décompte du trafic traité.

Voici un exemple d'affichage des règles actives avec visualisation des compteurs d'utilisation.

```
sudo nft list ruleset
```

```
table inet nat {
        chain postrouting {
                type nat hook postrouting priority srcnat; policy accept;
                oifname "enp0s1.360" counter packets 45 bytes 3424 masquerade
        }
}
```

4.4. Activation de l'adressage automatique pour le réseau de conteneurs

Dans le but mettre en place un adressage automatique des conteneurs hébergés sur l'autre machine virtuelle, on utilise l'outil `dnsmasq`. L'idée est de fournir un service DHCPv4 et SLAAC en un seul et unique fichier de configuration.

On débute par l'installation du paquet.

```
sudo apt -y install dnsmasq
```

Q15. Comment remplacer le fichier de configuration fourni lors de l'installation du paquet par notre propre fichier de configuration ?

Consulter le contenu du fichier `/etc/dnsmasq.conf` et extraire les options de configuration utiles au contexte de ces manipulations.

Voici la commande de copie du fichier issu de l'installation.

```
sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.dist
```

Voici un exemple de configuration adaptée à la maquette.

```
cat << EOF | sudo tee /etc/dnsmasq.conf
# Specify Container VLAN interface
interface=enp0s1.440

# Enable DHCPv4 on Container VLAN
dhcp-range=192.0.2.20,192.0.2.200,3h

# Enable IPv6 router advertisements
enable-ra

# Enable SLAAC
dhcp-range=::,constructor:enp0s1.440,ra-names,slaac

# Optional: Specify DNS servers
dhcp-option=option:dns-server,172.16.0.2,9.9.9.9
dhcp-option=option6:dns-server,[2001:678:3fc:3::2],[260:fe::fe]

# Avoid DNS listen port conflict between dnsmasq and systemd-resolved
port=0
EOF
```

Avertissement

Il faut impérativement changer le numéro de VLAN ainsi que les adresses IPv4 de l'exemple ci-dessus par les informations données dans le plan d'adressage des travaux pratiques.

De plus, une fois le fichier créé, il ne faut pas oublier de redémarrer le service et de contrôler l'état de son fonctionnement.

```
sudo systemctl restart dnsmasq
systemctl status dnsmasq
```

```
# dnsmasq.service - dnsmasq - A lightweight DHCP and caching DNS server
     Loaded: loaded (/usr/lib/systemd/system/dnsmasq.service; enabled; preset: enabled)
     Active: active (running) since Sat 2024-09-14 18:49:02 CEST; 10min ago
 Invocation: 8887da64fc36432db7be668ec71cbfb7
    Process: 1072 ExecStartPre=/usr/share/dnsmasq/systemd-helper checkconfig (code=exited, status=0/SUCCESS)
    Process: 1077 ExecStart=/usr/share/dnsmasq/systemd-helper exec (code=exited, status=0/SUCCESS)
    Process: 1084 ExecStartPost=/usr/share/dnsmasq/systemd-helper start-resolvconf (code=exited, status=0/SUCCESS)
   Main PID: 1083 (dnsmasq)
      Tasks: 1 (limit: 1087)
     Memory: 720K (peak: 2.8M)
        CPU: 94ms
     CGroup: /system.slice/dnsmasq.service
     └─1083 /usr/sbin/dnsmasq -x /run/dnsmasq/dnsmasq.pid -u dnsmasq -r /run/dnsmasq/resolv.conf \
       -7 /etc/dnsmasq.d,.dpkg-dist,.dpkg-old,.dpkg-new --local-service
       --trust-anchor=.,20326,8,2,e06d44b80b8f1d39a95c0b0d7c65d08458e880409bbc683457104237c7f8ec8d

sept. 14 18:49:02 router systemd[1]: Started dnsmasq.service - dnsmasq - A lightweight DHCP and caching DNS server.
```

5. Rôle serveur de conteneurs

5.1. Configuration de l'interface du serveur

Une fois la machine virtuelle serveur de conteneurs lancée, les premières étapes consistent à lui attribuer un nouveau nom et à configurer les interfaces réseau pour joindre le routeur voisin et l'Internet.

Q16. Comment changer le nom de la machine virtuelle ?

Il faut attribuer le nom d'hôte à l'aide la commande hostnamectl et redémarrer le système.

```
sudo hostnamectl hostname hosting
sudo reboot
```

Q17. Comment appliquer les configurations réseau IPv4 et IPv6 à partir de l'unique interface de la machine d'hébergement ?

Consulter la documentation de ***Netplan*** pour obtenir les informations sur la configuration des interfaces réseau à l'adresse *Netplan documentation*.

Il existe plusieurs possibilités pour configurer une interface réseau. Dans le contexte de ces manipulations, on utilise ***Netplan*** dans le but de séparer la partie déclarative du moteur de configuration.

C'est `systemd-networkd` qui joue le rôle de moteur de configuration sur les machines virtuelles utilisées avec ces manipulations.

La configuration de base fournie avec l'image maître suppose que l'interface obtienne un bail DHCP pour la partie IPv4 et une configuration automatique via SLAAC pour la partie IPv6. Cette configuration par défaut doit être éditée et remplacée.

- L'interface réseau appartient un seul et unique domaine de diffusion : le VLAN 440 dans le contexte de cette maquette.
- On fait le choix d'attribuer des adresses statiques à l'interface `enp0s1` pour être en mesure de tester le routage et la traduction d'adresses sources au niveau du routeur. Ainsi, on ne dépend pas du service `dnsmasq` pour les tests de communication.

Voici une copie du fichier `/etc/netplan/enp0s1.yaml` de la maquette.

```
cat << EOF | sudo tee /etc/netplan/enp0s1.yaml
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s1:
      dhcp4: false
      dhcp6: false
      accept-ra: true
      addresses:
        - 192.0.2.2/24
        - fda0:7a62:1b8::2/64
      routes:
        - to: default
          via: 192.0.2.1
        - to: "::/0"
          via: fe80::baad:caff:fefe:5
          on-link: true
      nameservers:
        addresses:
          - 172.16.0.2
          - 2001:678:3fc:3::2
EOF
```

Bien sûr, il ne faut pas oublier d'appliquer les paramètre de configuration de l'interface.

```
sudo netplan apply
```

Q18. Comment valider la configuration réseau du serveur de conteneurs ?

Lancer une série de tests ICMP IPv4 et IPv6.

On reprend les tests usuels avec les commandes ping et host.

```
etu@server:~$ ping -qc2 9.9.9.9
PING 9.9.9.9 (9.9.9.9) 56(84) bytes of data.

--- 9.9.9.9 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 19.433/19.586/19.740/0.153 ms
```

```
etu@server:~$ ping -qc2 2620:fe::fe
PING 2620:fe::fe(2620:fe::fe) 56 data bytes

--- 2620:fe::fe ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 42.983/43.114/43.246/0.131 ms
```

```
etu@server:~$ host kernel.org
kernel.org has address 139.178.84.217
kernel.org has IPv6 address 2604:1380:4641:c500::1
kernel.org mail is handled by 10 smtp1.kernel.org.
kernel.org mail is handled by 10 smtp3.kernel.org.
kernel.org mail is handled by 10 smtp2.kernel.org.
```

5.2. Installation du gestionnaire de conteneurs Incus

Sur l'hôte qui tient le rôle de serveur d'hébergement, la gestion des conteneurs est confiée à *Incus*.

Selon l'exergue du site, ***Incus*** est un gestionnaire de conteneurs et de machines virtuelles puissant, sûr et moderne.

Dans le contexte de ces manipulations, c'est la variété des modes d'interconnexion réseau offerte par ***Incus*** qui est le point le plus déterminant. Ici on utilise le mode *macvlan* qui raccorde les conteneurs au même domaine

de diffusion que la machine hôte. C'est le mode de raccordement le plus simple. Cependant, il n'autorise pas les communications entre les conteneurs du fait de l'isolation des espaces de noms.

Q19. Comment installer le gestionnaire de conteneurs Incus ?

Lancer une recherche dans la liste des paquets Debian.

Le paquet s'appelle tout simplement incus.

```
apt search ^incus
```

```
sudo apt -y install incus
```

Q20. Comment faire pour que l'utilisateur normal `etu` devienne administrateur et gestionnaire des conteneurs ?

Rechercher le nom du groupe système correspondant à l'utilisation des outils Incus.

Il faut que l'utilisateur normal appartienne au groupes systèmes `incus` et `incus-admin` pour qu'il ait tous les droits sur la gestion des conteneurs.

```
grep incus /etc/group
incus:x:990:
incus-admin:x:989:
```

```
sudo adduser etu incus
sudo adduser etu incus-admin
```

Avertissement

Attention ! Il faut se déconnecter/reconnecter pour bénéficier de la nouvelle attribution de groupe. On peut utiliser les commandes groups ou id pour vérifier le résultat.

```
groups
etu adm sudo users incus-admin incus
```

5.3. Configuration du gestionnaire de conteneurs Incus

Q21. Quelle est l'instruction de configuration initiale du gestionnaire Incus ?

Utiliser l'aide de la commande incus.

C'est l'instruction `incus admin init` qui nous intéresse.

Voici une copie d'écran de son exécution.

```
incus admin init
```

```
Would you like to use clustering? (yes/no) [default=no]:
Do you want to configure a new storage pool? (yes/no) [default=yes]:
Name of the new storage pool [default=default]:
Would you like to create a new local network bridge? (yes/no) [default=yes]: no
Would you like to use an existing bridge or host interface? (yes/no) [default=no]: yes
Name of the existing bridge or host interface: enp0s1
Would you like the server to be available over the network? (yes/no) [default=no]:
Would you like stale cached images to be updated automatically? (yes/no) [default=yes]:
Would you like a YAML "init" preseed to be printed? (yes/no) [default=no]:
```

Q22. Quelle est l'instruction qui permet d'afficher le profil par défaut des conteneurs ?

Rechercher dans les options de la commande incus profile.

Voici un exemple d'exécution.

```
incus profile show default
```

```
config: {}
description: Default Incus profile
devices:
  eth0:
    name: eth0
    nictype: macvlan
    parent: enp0s1
    type: nic
  root:
    path: /
    pool: default
    type: disk
name: default
used_by: []
project: default
```

L'affichage de ce profil par défaut permet de vérifier que tous les paramètres voulus sont correctement positionnés.

Q23. Quelle est l'instruction de création et de lancement de nouveaux conteneurs ?

Rechercher dans les options de la commande incus.

Tester son exécution avec un conteneur de type `debian/trixie`.

Voici un exemple d'exécution pour 3 nouveaux conteneurs.

```
for i in {0..2}; do incus launch images:debian/trixie c$i; done
```

```
Launching c0
Launching c1
Launching c2
```

```
incus ls
```

```
+------+---------+--------------------+-----------------------------------------------+-----------+-----------+
| NAME |  STATE  |        IPV4        |                     IPV6                      |   TYPE    | SNAPSHOTS |
+------+---------+--------------------+-----------------------------------------------+-----------+-----------+
| c0   | RUNNING | 192.0.2.52 (eth0)  | fda0:7a62:1b8:0:216:3eff:febd:a233 (eth0) | CONTAINER | 0         |
+------+---------+--------------------+-----------------------------------------------+-----------+-----------+
| c1   | RUNNING | 192.0.2.133 (eth0) | fda0:7a62:1b8:0:216:3eff:fe8e:cc62 (eth0) | CONTAINER | 0         |
+------+---------+--------------------+-----------------------------------------------+-----------+-----------+
| c2   | RUNNING | 192.0.2.187 (eth0) | fda0:7a62:1b8:0:216:3eff:fe7d:6898 (eth0) | CONTAINER | 0         |
+------+---------+--------------------+-----------------------------------------------+-----------+-----------+
```

La copie d'écran ci-dessus montre que l'adressage automatique des conteneurs depuis le routeur a fonctionné.

Q24. Comment tester les communications réseau depuis chaque conteneur ?

Rechercher dans les options de la commande incus celle qui permet de lancer un traitement dans le conteneur.

C'est la commande incus exec qui correspond à notre besoin. Voici une exemple de boucle qui permet de lancer les tests ICMP IPv4 et IPv6 dans les 3 conteneurs actifs.

```
for i in {0..2}
do
    echo ">>>>>>>>>>>>>>>> c$i"
    incus exec c$i -- ping -qc2 9.9.9.9
    incus exec c$i -- ping -qc2 2620:fe::fe
done
```

```
PING 9.9.9.9 (9.9.9.9) 56(84) bytes of data.

--- 9.9.9.9 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 27.887/28.676/29.465/0.789 ms
PING 2620:fe::fe (2620:fe::fe) 56 data bytes

--- 2620:fe::fe ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 41.727/48.212/54.697/6.485 ms
>>>>>>>>>>>>>>>> c1
PING 9.9.9.9 (9.9.9.9) 56(84) bytes of data.

--- 9.9.9.9 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 28.179/34.740/41.302/6.561 ms
PING 2620:fe::fe (2620:fe::fe) 56 data bytes

--- 2620:fe::fe ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 40.002/40.529/41.057/0.527 ms
>>>>>>>>>>>>>>>> c2
PING 9.9.9.9 (9.9.9.9) 56(84) bytes of data.

--- 9.9.9.9 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 28.539/35.408/42.278/6.869 ms
PING 2620:fe::fe (2620:fe::fe) 56 data bytes

--- 2620:fe::fe ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 40.428/47.517/54.606/7.089 ms
```

Q25. Comment exécuter des jeux d'instructions dans les conteneurs depuis le serveur d'hébergement ?

On entre ici dans le domaine de l'automatisation à l'aide de scripts ***Bash***. Même si l'ambition reste très modeste, on peut développer un script qui utilise la liste des conteneurs actifs pour lancer une suite de traitements dans ces mêmes conteneurs.

Comme les conteneurs ***Incus*** appartiennent à la famille des conteneurs système, ils disposent d'une arborescence complète et d'une gestion de paquets. Allons y pour une mise à jour des paquets de chaque conteneur actif.

Voici un exemple de code qui stocke les commandes à lancer dans un tableau et qui les exécute sur chacun des conteneurs actifs.

```
#!/bin/bash

cmds=(
  "apt -y update"
  "apt -y full-upgrade"
  "apt -y clean"
  "apt -y autopurge"
)

clist=$(incus list status=running -c n -f compact | grep -v NAME | tr '\n' ' ' | tr -s ' ')

for c in $clist; do
  echo ">>>>>>>>>>>>>>>>> $c"
  for cmd in "${cmds[@]}"; do
    eval "incus exec $c -- $cmd"
  done
done

exit 0
```

Si ce script est enregistré dans le fichier `run-commands-in-containers.sh`, on peut lancer les traitements comme dans l'exemple ci-dessous.

```
bash run-commands-in-containers.sh
>>>>>>>>>>>>>>>>> c0
Hit:1 http://deb.debian.org/debian trixie InRelease
Hit:2 http://deb.debian.org/debian trixie-updates InRelease
Hit:3 http://deb.debian.org/debian-security trixie-security InRelease
All packages are up to date.
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
>>>>>>>>>>>>>>>>> c1
Hit:1 http://deb.debian.org/debian trixie InRelease
Hit:2 http://deb.debian.org/debian trixie-updates InRelease
Hit:3 http://deb.debian.org/debian-security trixie-security InRelease
All packages are up to date.
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
>>>>>>>>>>>>>>>>> c2
Hit:1 http://deb.debian.org/debian trixie InRelease
Hit:2 http://deb.debian.org/debian trixie-updates InRelease
Hit:3 http://deb.debian.org/debian-security trixie-security InRelease
All packages are up to date.
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
```

Pour compléter, voici une seconde version du script qui utilise les arguments de la ligne de commande pour remplir le tableau des commandes à exécuter.

```
#!/bin/bash

cmds=("$@")

clist=$(incus list status=running -c n -f compact | grep -v NAME | tr '\n' ' ' | tr -s ' ')

for c in $clist; do
  echo ">>>>>>>>>>>>>>>>> $c"
  for cmd in "${cmds[@]}"; do
    eval "incus exec $c -- $cmd"
  done
done
```

Q26. Comment passer d'un adressage automatique à un adressage statique pour chaque conteneur ?

Comme la gestion de la configuration des interfaces est assurée par `systemd-networkd`, il faut s'intéresser à la syntaxe du fichier `/etc/systemd/network/eth0.network` de chaque conteneur.

Cette question est un prétexte pour utiliser le transfert de fichier depuis le serveur d'hébergement vers les conteneurs.

Voici une liste des actions à réaliser sur tous les conteneurs actifs.

1. Installer le paquet netplan.io

2. Générer le fichier de déclaration YAML des paramètres de configuration réseau des interfaces `eth0`

3. Transférer le fichier de déclaration YAML dans le dossier `/etc/netplan/`

4. Effacer le fichier `/etc/systemd/network/eth0.network`

5. Appliquer la nouvelle configuration réseau

Pour connaître les paramètres de configuration réseau d'une interface de conteneur, on peut extraire le fichier `/etc/systemd/network/eth0.network` et consulter son contenu.

```
incus file pull c0/etc/systemd/network/eth0.network .
cat eth0.network
```

```
[Match]
Name=eth0

[Network]
DHCP=true

[DHCPv4]
UseDomains=true

[DHCP]
ClientIdentifier=mac
```

On vérifie ainsi que la configuration réseau issue de la source de tirage des conteneurs implique un adressage automatique au moins en IPv4. On propose donc de remplacer cet adressage automatique par un adressage statique.

Voici une proposition de script qui traite chacun des points définis dans la question.

```
#!/bin/bash

# Préparation -> générer la liste des conteneurs actifs
clist=$(incus list status=running -c n -f compact | grep -v NAME | tr '\n' ' ' | tr -s ' ')

# Étape 1 -> installer le paquet netplan.io
. run-commands-in-containers.sh "apt -y install netplan.io"

addr_idx=0
for c in $clist; do
  echo ">>>>>>>>>>>>>>>> $c"

# Étape 2 -> générer le fichier de configuration réseau YAML
$(cat << EOF > eth0.yaml
network:
  version: 2
  renderer: networkd
  ethernets:
    eth0:
      dhcp4: false
      dhcp6: false
      accept-ra: true
      addresses:
        - 192.0.2.$((addr_idx + 10))/24
        - fda0:7a62:1b8::$(printf "%x" $((addr_idx + 10)))/64
      routes:
        - to: default
          via: 192.0.2.1
        - to: "::/0"
          via: fe80::baad:caff:fefe:5
          on-link: true
      nameservers:
        addresses:
          - 172.16.0.2
          - 2001:678:3fc:3::2
EOF
)

# Étape 3 -> transférer le fichier de déclaration YAML
  incus file push eth0.yaml $c/etc/netplan/eth0.yaml

# Étape 4 -> effacer le fichier /etc/systemd/network/eth0.network
  incus exec $c -- rm /etc/systemd/network/eth0.network

# Étape 5 -> appliquer la nouvelle configuration
  incus exec $c -- netplan apply

  ((addr_idx++))
done

exit 0
```

Si le code du script ci-dessus est placé dans un fichier appelé `set-static-addressing.sh`, on peut l'exécuter directement et relever les résultats.

```
bash set-static-addressing.sh
```

```
incus restart --all
```

```
incus ls
+------+---------+-------------------+----------------------------------------------+-----------+-----------+
| NAME |  STATE  |       IPV4        |                     IPV6                     |   TYPE    | SNAPSHOTS |
+------+---------+-------------------+----------------------------------------------+-----------+-----------+
| c0   | RUNNING | 192.0.2.10 (eth0) | fda0:7a62:1b8::a (eth0)                      | CONTAINER | 0         |
|      |         |                   | fda0:7a62:1b8:0:216:3eff:febd:a233 (eth0)    |           |           |
+------+---------+-------------------+----------------------------------------------+-----------+-----------+
| c1   | RUNNING | 192.0.2.11 (eth0) | fda0:7a62:1b8::b (eth0)                      | CONTAINER | 0         |
|      |         |                   | fda0:7a62:1b8:0:216:3eff:fe8e:cc62 (eth0)    |           |           |
+------+---------+-------------------+----------------------------------------------+-----------+-----------+
| c2   | RUNNING | 192.0.2.12 (eth0) | fda0:7a62:1b8::c (eth0)                      | CONTAINER | 0         |
|      |         |                   | fda0:7a62:1b8:0:216:3eff:fe7d:6898 (eth0)    |           |           |
+------+---------+-------------------+----------------------------------------------+-----------+-----------+
```

6. Pour conclure

Le routage inter-VLAN joue un rôle essentiel dans les environnements de virtualisation et les réseaux en nuage modernes. Cette technique permet une segmentation logique efficace des réseaux, offrant une flexibilité et une sécurité accrues dans des infrastructures complexes. En permettant la communication entre différents VLANs via un routeur, elle facilite la création de topologies réseau évolutives et adaptables.

L'outil Incus se distingue par sa simplicité d'utilisation et sa flexibilité, offrant une gestion efficace des conteneurs et des machines virtuelles. Sa capacité à gérer divers modes d'interconnexion réseau, comme le mode *macvlan* utilisé dans ce document, en fait un choix idéal pour les administrateurs système cherchant à optimiser et automatiser leurs infrastructures.

Enfin, pour ce qui est de l'automatisation, les traitements “atomiques” proposés sont pertinents. Ce qui est nettement moins satisfaisant, c'est l'utilisation des boucles dans des scripts Bash. Si l'automatisation dépasse le cadre des manipulations proposées ici, il ne faut pas oublier la séparation entre l'inventaire des dispositifs (routeur, commutateurs, machines virtuelles, conteneurs) et les procédures est un pilier essentiel d'une automatisation efficace.

Routage interVLAN et protocole PPPoE

https://www.inetdoc.net

Résumé

La généralisation de l'utilisation de la fibre optique dans les réseaux étendus (WAN) jusqu'au raccordement domestique s'est accompagnée d'un changement important au niveau des liaisons de données. La technologie Ethernet est devenue universelle et couvre tous les besoins de commutation de circuits.

Cependant, pour raccorder les sites d'entreprises via des réseaux d'opérateurs, les fonctions historiques du protocole PPP (*Point-to-Point Protocol*) sont toujours utiles. C'est là que le protocole PPPoE intervient. Il permet d'associer un réseau de diffusion Ethernet avec un fonctionnement point à point typique des réseaux étendus.

Le but des manipulations présentées dans ce document est d'illustrer la mise en œuvre d'une session PPPoE entre un routeur virtuel central et un site distant factice (une autre machine virtuelle) qui héberge quelques services.

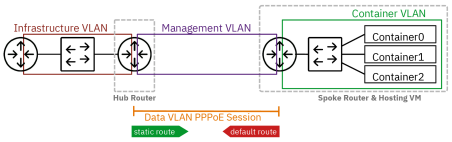


Table des matières

1. Objectifs

Après avoir réalisé les manipulations présentées dans ce document, les étudiants seront capables de :

1. Configurer et mettre en œuvre une topologie réseau complexe combinant routage inter-VLAN et protocole PPPoE.
2. Installer et configurer des conteneurs système Incus, y compris leur adressage réseau statique et dynamique.
3. Mettre en place et dépanner une session PPPoE entre deux routeurs virtuels avec les étapes d'authentification et de négociation du protocole.
4. Utiliser des outils d'automatisation comme les scripts Bash et Netplan pour configurer et gérer efficacement les interfaces réseau et les conteneurs.

5. Implémenter et tester la connectivité IPv4 et IPv6 dans un environnement réseau virtualisé, incluant la configuration de routes statiques et la traduction d'adresses.

2. Interface Ethernet & protocole PPP

Le format de trame historique HDLC est abandonné. Il faut dire que ce format de trame date du développement des liaisons séries asynchrones. Aujourd'hui, les liaisons sur fibres optiques sont ***Full-Duplex*** et on ne se préoccupe plus de synchronisation au niveau de la couche liaison de données. Le format de trame Ethernet devient ainsi une référence universelle.

Le protocole PPP offre depuis l'origine une configuration indépendante de la technologie du réseau étendu aussi bien au niveau de la couche liaison que de la couche réseau.

L'association entre trame Ethernet et PPP se fait grâce à un autre protocole baptisé PPPoE. Ce dernier permet d'encapsuler des trames PPP dans des trames Ethernet. Il est décrit à la page *Point-to-point protocol over Ethernet* qui permet de traiter les questions ci-après.

Q27. Quelle est la raison de l'ajout d'un nouveau protocole entre Ethernet et PPP ?

Consulter la page *Point-to-point protocol over Ethernet*.

Le protocole PPP, initialement destiné aux liaisons point à point, nécessite un mécanisme de découverte des extrémités pour fonctionner sur un réseau local Ethernet, qui est un réseau de diffusion où le canal de transmission est partagé entre tous les hôtes.

Q28. Quels sont les messages de découverte et de session PPPoE ? Préciser qui est l'initiative de la découverte.

Consulter la page *Point-to-point protocol over Ethernet*.

- Client to server: Initiation (PADI)
- Server to client: Offer (PADO)
- Client to server: request (PADR)
- Server to client: session-confirmation (PADS)
- Either end to other end: termination (PADT)

Q29. Quels sont les autres mécanismes de découverte de voisins connus dans un réseau local Ethernet ?

Voici la liste des «grands classiques».

- Address Resolution Protocol (ARP).

  Quelle est l'adresse MAC d'un hôte dont on connaît l'adresse IPv4 ?

- Neighbor Discovery Protocol (NDP).

  Ce protocole est associé à IPv6. Il définit 5 messages ICMPv6 qui couvrent les mêmes opérations que celles réalisées par le protocole ARP sans avoir recours à la diffusion et qui ajoutent de nouvelles fonctions.

- Multicast DNS (mDNS) ou ***Bonjour***.

  Ce protocole entre dans la famille ***zeroconf*** qui a pour but d'annoncer et de fournir des éléments de configuration aux hôtes du réseau sans faire appel à une infrastructure de services de la couche application tels que DNS et DHCP.

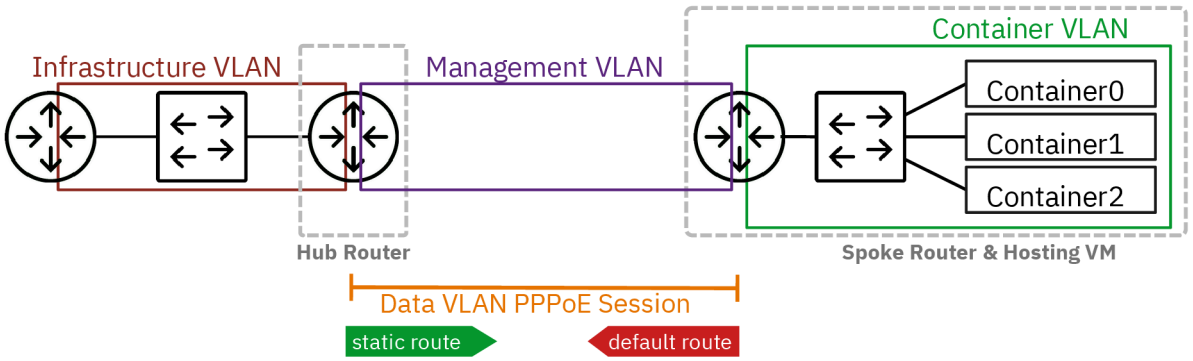
3. Topologies logiques et virtuelles

La représentation de la topologie logique ci-dessous montre deux routeurs. Le routeur placé à gauche assure l'interconnexion entre un réseau d'infrastructure (VLAN rouge) et le réseau opérateur (VLAN violet). Le routeur placé à droite assure l'interconnexion entre le même réseau opérateur et le réseau du site distant (VLAN vert). Les services hébergés sur le site distant sont compris dans le même réseau (VLAN vert).

Sur le réseau étendu factice, on distingue deux autres VLANs : le VLAN violet appelé ***Management VLAN*** est utilisé pour la supervision et le VLAN orange appelé ***Data VLAN*** est utilisé pour acheminer les données entre le site central (***Hub***) et le site distant (***Spoke***).

C'est sur ce dernier réseau (VLAN orange) que la session PPPoE doit être établie pour que le plan d'adressage réseau de l'entreprise soit conforme.

Enfin, les deux rectangles en gris pointillé identifient les machines virtuelles utilisées pour les manipulations.

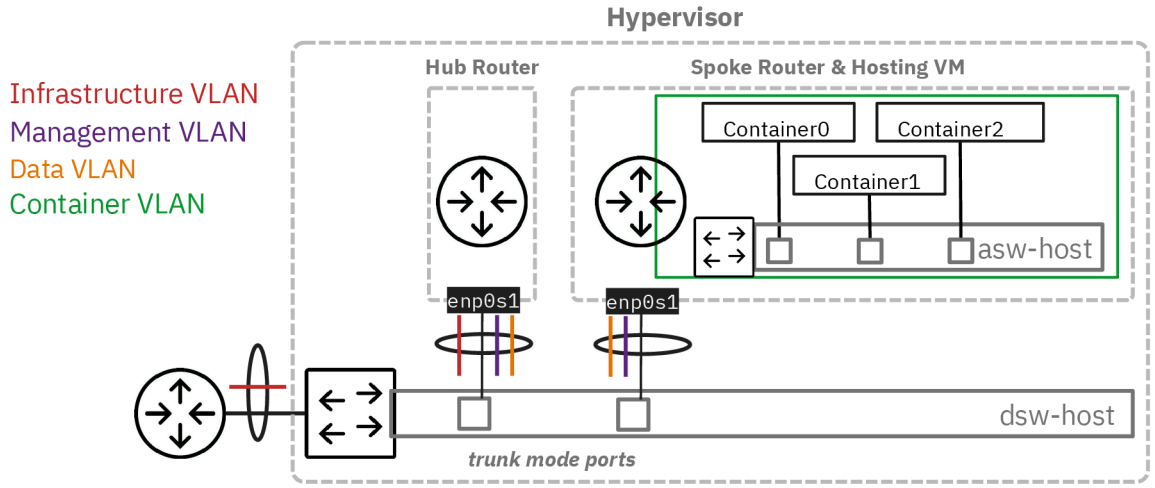


Topologie logique

La représentation de la topologie vue sous l'angle de la virtualisation sur un système hôte hyperviseur montre que le VLAN vert appelé ***Container VLAN*** n'est visible qu'à l'intérieur de la machine virtuelle qui représente le site distant. Ce VLAN est isolé et ses préfixes réseau IPv4 et IPv6 doivent être ***routés***. C'est la raison pour laquelle la machine virtuelle du site distant dispose de son propre commutateur : `asw-host`.

Tous les autres VLANs sont présents sur le commutateur virtuel de couche distribution appelé `dsw-host`. Ce commutateur appartient au système hôte. Il assure le raccordement entre les réseaux physiques et virtualisés. Les deux routeurs virtuels ***Hub*** et ***Spoke*** sont raccordés sur des ports configurés en mode ***trunk*** sur lesquels le trafic de plusieurs VLANs doit transiter.

Côté conteneurs, le raccordement au commutateur `asw-host` sera assuré automatiquement par le gestionnaire *Incus*.



Topologie hébergée

Voici le plan d'adressage utilisé pour la maquette qui sert à la rédaction de ce support de travaux pratiques.

Tableau 1. Plan d'adressage de la maquette « Routage interVLAN et protocole PPPoE »

Planète	VLAN	Numéro	Type	Adresse
Maquette	Rouge	360	Passerelle	`192.168.104.130/29` `2001:678:3fc:168::1/64`
	Violet	440	Adresse	`fe80:1b8::1`
				`fe80:1b8::2`
	Orange	441	Point à point	`10.4.41.1:10.4.41.2`
			Authentifiants	`spoke_site0 / 5p0k3`
	Vert	40	Passerelle	`203.0.113.1/24` `fda0:7a62:28::1/64`

4. Raccordement au commutateur de distribution

Dans cette section, on étudie le raccordement des deux machines virtuelles au commutateur de distribution sur le système hôte.

Q30. Comment contrôler la configuration des ports du commutateur de distribution sur le système hôte ?

Le commutateur virtuel implanté sur le système hôte est géré par ***Open vSwitch***. On fait donc appel à la commande ovs-vsctl pour accéder aux paramètres de la configuration des ports.

- Pour le port nommé `tap200`, on obtient le paramètre `vlan_mode` avec l'instruction :

```
sudo ovs-vsctl get port tap200 vlan_mode
trunk
```

Le mode `trunk` correspond à un canal de transmission unique dans lequel circule le trafic de plusieurs domaines de diffusion ou VLANs.

- Pour le port nommé `tap2`, on obtient la valeur `access` pour le même paramètre :

```
sudo ovs-vsctl get port tap2 vlan_mode
access
```

Ici, le mode `access` correspond à un canal de transmission dans lequel circule le trafic d'un seul et unique domaine de diffusion ou VLAN.

Q31. Comment afficher le numéro de VLAN attribué au port en mode accès du commutateur de distribution sur le système hôte ?

On reprend la même commande que dans la question précédente avec le mot clé `tag`.

```
sudo ovs-vsctl get port tap2 tag
20
```

Q32. Comment affecter le numéro de VLAN attribué au port en mode accès du commutateur de distribution sur le système hôte ?

On reprend à nouveau la même commande avec l'option `set`.

```
sudo ovs-vsctl set port tap2 tag=440
```

Les valeurs données dans l'exemple ci-dessus sont à changer suivant les attributions du plan d'adressage des réseaux d'hébergement et de conteneurs.

Q33. Comment configurer les ports du commutateur avant le lancement des machines virtuelles ?

On utilise le script de procédure `switch-conf.py` qui applique les déclarations contenues dans un fichier YAML. Le code du script est accessible à partir du dépôt Git *startup-scripts*.

Voici une copie du fichier de configuration des deux ports de commutateur.

```
ovs:
  switches:
    - name: dsw-host
      ports:
        - name: tap5 # Hub port
          type: OVSPort
          vlan_mode: trunk
          trunks: [360, 440, 441]
        - name: tap6 # Spoke port
          type: OVSPort
          vlan_mode: trunk
          trunks: [52, 440, 441] # Avec VLAN d'accès temporaire
#         trunks: [440, 441] # Sans VLAN d'accès temporaire
```

On applique les paramètres définis ci-dessus.

```
$HOME/masters/scripts/switch-conf.py switch.yaml
```

On obtient les résultats suivants.

```
----------------------------------------
Configuring switch dsw-host
>> Port tap5 vlan_mode is already set to trunk
>> Port tap5 trunks set to [360, 440, 441]
>> Port tap6 vlan_mode set to trunk
>> Port tap6 trunks set to [52, 440, 441]
----------------------------------------
```

Les numéros de port de commutateur et de VLAN donnés dans les exemples ci-dessus sont à changer suivant le contexte.

Q34. Comment lancer les machines virtuelles associées aux rôles routeur et hébergement de conteneurs ?

On utilise le script de procédure `lab-startup.py` qui applique les déclarations contenues dans un fichier YAML. Le code du script est accessible à partir du dépôt Git *startup-scripts*.

Voici une copie du fichier de déclaration des deux machines virtuelles.

```
kvm:
  vms:
    - vm_name: hub
      master_image: debian-testing-amd64.qcow2 # master image to be used
      force_copy: false # do not force copy the master image to the VM image
      memory: 1024
      tapnum: 5
    - vm_name: spoke
      master_image: debian-testing-amd64.qcow2 # master image to be used
      force_copy: false # do not force copy the master image to the VM image
      memory: 1024
      tapnum: 6
```

On lance les deux machines virtuelles avec le script lab-startup.py.

```
$HOME/masters/scripts/lab-startup.py lab2.yaml
```

```
Copying /home/etudianttest/masters/debian-testing-amd64.qcow2 to hub.qcow2...done
Creating OVMF_CODE.fd symlink...
Creating hub_OVMF_VARS.fd file...
Starting hub...
~> Virtual machine filename   : hub.qcow2
~> RAM size                   : 1024MB
~> SPICE VDI port number      : 5905
~> telnet console port number : 2305
~> MAC address                : b8:ad:ca:fe:00:05
~> Switch port interface      : tap5, trunk mode
~> IPv6 LL address            : fe80::baad:caff:fefe:5%dsw-host
hub started!
Copying /home/etudianttest/masters/debian-testing-amd64.qcow2 to spoke.qcow2...done
Creating spoke_OVMF_VARS.fd file...
Starting spoke...
~> Virtual machine filename   : spoke.qcow2
~> RAM size                   : 1024MB
~> SPICE VDI port number      : 5906
~> telnet console port number : 2306
~> MAC address                : b8:ad:ca:fe:00:06
~> Switch port interface      : tap6, trunk mode
~> IPv6 LL address            : fe80::baad:caff:fefe:6%dsw-host
spoke started!
```

Les deux machines virtuelles sont maintenant disponibles pour la suite des manipulations.

5. Routeur Hub

Dans cette section, on étudie la machine virtuelle qui joue le rôle de routeur entre le réseau d'infrastructure (VLAN rouge) et le réseau étendu (VLANs violet et orange) qui dessert le site distant.

5.1. Configuration des interfaces du routeur

Une fois la machine virtuelle routeur lancée, les premières étapes consistent à lui attribuer un nouveau nom et à configurer les interfaces réseau pour joindre les hôtes voisins.

Q35. Comment changer le nom de la machine virtuelle ?

Il faut éditer le fichier `/etc/hostname` en remplaçant le nom local par le nom voulu. Il est ensuite nécessaire de redémarrer pour que le nouveau nom soit pris en compte par tous les outils du système.

```
etu@localhost:~$ sudo hostnamectl hostname hub
etu@localhost:~$ sudo reboot
```

Q36. Comment appliquer les configurations réseau IPv4 et IPv6 à partir de l'unique interface du routeur ?

Consulter la documentation de ***Netplan*** pour obtenir les informations sur la configuration des interfaces réseau à l'adresse *Netplan documentation*.

Il existe plusieurs possibilités pour configurer une interface réseau. Dans le contexte de ces manipulations, on utilise ***Netplan*** dans le but de séparer la partie déclarative du moteur de configuration.

C'est `systemd-networkd` qui joue le rôle de moteur de configuration sur les machines virtuelles utilisées avec ces manipulations.

La configuration de base fournie avec l'image maître suppose que l'interface obtienne un bail DHCP pour la partie IPv4 et une configuration automatique via SLAAC pour la partie IPv6. Cette configuration par défaut doit être éditée et remplacée. Il faut configurer trois interfaces.

- L'interface principale correspond à l'interface "physique" de la machine. Elle est nommée `enp0s1` en fonction de l'ordre des adresses des composants raccordés au bus PCI.

- Une sous-interface doit être créée pour le réseau d'infrastructure (VLAN rouge). Cette interface doit désigner les passerelles IPv4 et IPv6 de façon à joindre l'Internet.

- Une sous-interface doit être créée pour le réseau étendu de l'exploitant des fourreaux et du câblage en fibres optiques (VLAN violet). Cette interface ne comprend qu'une adresse IPv6 de lien local pour les tests de connectivité ICMPv6 entre les deux sites.

- Une autre sous-interface doit être créée pour le réseau étendu de l'opérateur (VLAN orange). Cette interface ne contient aucune adresse lors de l'initialisation système. C'est le démon pppd qui est responsable de l'attribution des adresses IPv4 et IPv6 lotrs de l'établissement de la session du protocole PPP.

Voici une copie du fichier `/etc/netplan/enp0s1.yaml` de la maquette.

```
network:
  version: 2
  ethernets:
    enp0s1:
      dhcp4: false
      dhcp6: false
      accept-ra: false
      nameservers:
        addresses:
          - 172.16.0.2
          - 2001:678:3fc:3::2

  vlans:
    enp0s1.360: # VLAN rouge
      id: 360
      link: enp0s1
      addresses:
        - 192.168.104.130/29
        - 2001:678:3fc:168::82/64
      routes:
        - to: default
          via: 192.168.104.129
        - to: "::/0"
          via: "fe80:168::1"
          on-link: true
    enp0s1.440: # VLAN violet
      id: 440
      link: enp0s1
      addresses:
        - fe80:1b8::1/64
    enp0s1.441: # VLAN orange
      id: 441
      link: enp0s1
      addresses: []
```

Une fois le fichier de configuration en place, il suffit de faire appel à la commande netplan pour appliquer les changements.

```
sudo netplan apply
```

Pour vérifier que l'adressage réseau est correct, on dispose de plusieurs solutions. Voici un exemple avec netplan qui reprend l'ensemble de la configuration réseau.

```
sudo netplan status
```

```
     Online state: online
    DNS Addresses: 2001:678:3fc:3::2 (compat)
                   172.16.0.2 (compat)
                   2001:678:3fc:3::2 (compat)
       DNS Search: .

#  1: lo ethernet UNKNOWN/UP (unmanaged)
      MAC Address: 00:00:00:00:00:00
        Addresses: 127.0.0.1/8
                   ::1/128

#  2: enp0s1 ethernet UP (networkd: enp0s1)
      MAC Address: b8:ad:ca:fe:00:05 (Red Hat, Inc.)
        Addresses: fe80::baad:caff:fefe:5/64 (link)
    DNS Addresses: 172.16.0.2
                   2001:678:3fc:3::2
           Routes: fe80::/64 metric 256

#  3: enp0s1.441 vlan UP (networkd: enp0s1.441)
      MAC Address: b8:ad:ca:fe:00:05
        Addresses: fe80::baad:caff:fefe:5/64 (link)
           Routes: fe80::/64 metric 256

#  4: enp0s1.440 vlan UP (networkd: enp0s1.440)
      MAC Address: b8:ad:ca:fe:00:05
        Addresses: fe80:1b8::1/64 (link)
                   fe80::baad:caff:fefe:5/64 (link)
           Routes: fe80::/64 metric 256
                   fe80:1b8::/64 metric 256

#  5: enp0s1.360 vlan UP (networkd: enp0s1.360)
      MAC Address: b8:ad:ca:fe:00:05
        Addresses: 192.168.104.130/29
                   2001:678:3fc:168:baad:caff:fefe:5/64 (dynamic, ra)
                   2001:678:3fc:168::82/64 (ra)
                   fe80::baad:caff:fefe:5/64 (link)
    DNS Addresses: 2001:678:3fc:3::2
           Routes: default via 192.168.104.129 (static)
                   192.168.104.128/29 from 192.168.104.130 (link)
                   2001:678:3fc:168::/64 metric 256
                   2001:678:3fc:168::/64 metric 512 (ra)
                   fe80::/64 metric 256
                   default via fe80:168::1 metric 1024 (static)
```

Q37. Quels sont les tests de connectivité réalisables après application de la nouvelle configuration des interfaces réseau ?

Relever l'état des trois interfaces et procédez aux tests ICMP et DNS en respectant l'ordre des couches de la modélisation.

Sans la confirmation que la configuration du serveur de conteneurs est prête, c'est du côté hébergement et accès Internet qu'il faut orienter les tests. Classiquement, on cherche à joindre la passerelle en premier puis l'Internet ensuite via des requêtes ICMP. Enfin, on effectue un test de couche application avec une requête DNS.

```
ping -q -c2 192.168.104.129
PING 192.168.104.129 (192.168.104.129) 56(84) bytes of data.

--- 192.168.104.129 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 1.501/1.516/1.531/0.015 ms
```

```
PING 9.9.9.9 (9.9.9.9) 56(84) bytes of data.

--- 9.9.9.9 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 49.304/52.098/54.892/2.794 ms
```

```
ping -q -c2 fe80:168::1%enp0s1.360
PING fe80:168::1%enp0s1.360(fe80:168::1%enp0s1.360) 56 data bytes

--- fe80:168::1%enp0s1.360 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 1.459/13.305/25.152/11.846 ms
```

```
ping -q -c2 2620:fe::fe
PING 2620:fe::fe(2620:fe::fe) 56 data bytes

--- 2620:fe::fe ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 41.812/42.437/43.063/0.625 ms
```

```
host quad9.net
quad9.net has address 216.21.3.77
quad9.net has IPv6 address 2620:0:871:9000::77
quad9.net mail is handled by 10 mx4.quad9.net.
quad9.net mail is handled by 20 mx2.quad9.net.
quad9.net mail is handled by 5 mx1.quad9.net.
```

5.2. Activation de la fonction routage

Sans modification de la configuration par défaut, un système GNU/Linux n'assure pas la fonction de routage du trafic d'une interface réseau à une autre.

L'activation du routage correspond à un réglage de paramètres du sous-système réseau du noyau Linux. L'outil qui permet de consulter et modifier les réglages de paramètre sur le noyau est appelé sysctl.

Q38. Comment activer le routage dans le sous-système réseau du noyau Linux ?

Utiliser la commande sysctl pour effectuer des recherches et identifier les paramètres utiles. Par exemple :

```
sudo sysctl -a -r ".*forward.*"
```

Il est dorénavant recommandé de créer un fichier de configuration spécifique par fonction. C'est la raison pour laquelle on crée un nouveau fichier `/etc/sysctl.d/10-routing.conf`.

Attention ! Il ne faut pas oublier d'appliquer les nouvelles valeurs des paramètres de configuration.

```
cat << EOF | sudo tee /etc/sysctl.d/10-routing.conf
net.ipv4.ip_forward=1
net.ipv6.conf.all.forwarding=1
net.ipv4.conf.all.log_martians=1
EOF
```

Voici un exemple des résultats obtenus après application des nouveaux paramètres.

```
sudo sysctl --system
```

```
* Applique /usr/lib/sysctl.d/10-coredump-debian.conf …
* Applique /etc/sysctl.d/10-routing.conf …
* Applique /usr/lib/sysctl.d/50-default.conf …
* Applique /usr/lib/sysctl.d/50-pid-max.conf …
* Applique /etc/sysctl.conf …
kernel.core_pattern = core
net.ipv4.ip_forward = 1
net.ipv6.conf.all.forwarding = 1
net.ipv4.conf.all.log_martians = 1
kernel.sysrq = 0x01b6
kernel.core_uses_pid = 1
net.ipv4.conf.default.rp_filter = 2
net.ipv4.conf.enp0s1/360.rp_filter = 2
net.ipv4.conf.enp0s1/440.rp_filter = 2
net.ipv4.conf.enp0s1.rp_filter = 2
net.ipv4.conf.lo.rp_filter = 2
net.ipv4.conf.default.accept_source_route = 0
net.ipv4.conf.enp0s1/360.accept_source_route = 0
net.ipv4.conf.enp0s1/440.accept_source_route = 0
net.ipv4.conf.enp0s1.accept_source_route = 0
net.ipv4.conf.lo.accept_source_route = 0
net.ipv4.conf.default.promote_secondaries = 1
net.ipv4.conf.enp0s1/360.promote_secondaries = 1
net.ipv4.conf.enp0s1/440.promote_secondaries = 1
net.ipv4.conf.enp0s1.promote_secondaries = 1
net.ipv4.conf.lo.promote_secondaries = 1
net.ipv4.ping_group_range = 0 2147483647
net.core.default_qdisc = fq_codel
fs.protected_hardlinks = 1
fs.protected_symlinks = 1
fs.protected_regular = 2
fs.protected_fifos = 1
kernel.pid_max = 4194304
```

Q39. Quelles sont les conditions à réunir pour tester le fonctionnement du routage ?

Rechercher comment utiliser l'analyseur réseau tshark pour caractériser l'acheminement du trafic d'un réseau à l'autre.

Le plan d'adressage prévoit d'utiliser des préfixes ayant une portée locale pour les réseaux de conteneurs. Il n'est donc pas possible de passer par une requête ICMP pour caractériser l'accès aux réseaux distants. En effet, l'adresse source n'est pas reconnue par l'hôte distant et les routeurs de l'Internet ne disposent d'aucune solution pour joindre le réseau des conteneurs.

Voici un extrait de capture qui montre que le serveur de conteneur cherche à joindre un hôte sur l'Internet sans succès. Cette capture étant réalisée sur l'interface réseau côté hébergement, elle montre que le trafic est bien acheminé d'un réseau à l'autre.

```
tshark -i enp0s1.360
Capturing on 'enp0s1.360'
    1 0.000000000    192.0.2.2 → 9.9.9.9      DNS 81 Standard query 0xbdab A 1.debian.pool.ntp.org
    2 0.000056361    192.0.2.2 → 9.9.9.9      DNS 81 Standard query 0xab92 AAAA 1.debian.pool.ntp.org
```

5.3. Activation de la traduction d'adresses

Le résultat de la question ci-dessus montre que les hôtes situés dans le réseau des conteneurs ne peuvent pas joindre l'Internet puisque les préfixes réseau utilisés ont une portée limitée.

Q40. Quels sont les paquets qui fournissent les outils de gestion de la traduction d'adresses ?

Rechercher les paquets relatifs au filtrage et à la gestion des règles de pare-feux.

Dans le contexte des ces manipulations, nous utilisons `nftables` comme outil de gestion du filtrage.

C'est la partie outils de l'espace utilisateur qui nous intéresse ici.

```
apt search ^nftables$
nftables/testing,now 1.1.0-2 amd64  [installé]
  programme de contrôle des règles de filtrage de paquets du projet Netfilter
```

```
sudo apt -y install nftables
```

Q41. Quelles sont les règles à appliquer pour assurer une traduction des adresses sources en sortie sur le réseau d'infrastructure (VLAN rouge) ?

Rechercher dans des exemples de configuration `nftables` avec la fonction `MASQUERADE`.

Voici un exemple de création du fichier `/etc/nftables.conf` avec le jeu d'instructions qui assure la traduction d'adresses sources pour IPv4 et IPv6.

```
cat << 'EOF' | sudo tee /etc/nftables.conf
#!/usr/sbin/nft -f

flush ruleset

# Define variables
define RED_VLAN = enp0s1.360

table inet nat {
    chain postrouting {
        type nat hook postrouting priority 100;
        oifname $RED_VLAN counter packets 0 bytes 0 masquerade
    }
}
EOF
```

Avertissement

Il faut impérativement changer le nom d'interface en utilisant le numéro de VLAN attribué dans le plan d'adressage des travaux pratiques.

La création de ce fichier de règles n'est pas suffisante. Il faut appliquer les règles contenues dans le fichier.

```
sudo nft -f /etc/nftables.conf
```

Q42. Comment rendre le chargement des règles de filtrage automatique au redémarrage du système ?

Afficher l'état du service `nftables.service`. Activer ce service si celui est à l'état désactivé (***disabled***).

Pour afficher l'état du service, on utilise la commande suivante.

```
systemctl status nftables.service
```

On constate qu'il faut activer ce service pour assurer le chargement automatique des règles de filtrage au démarrage.

```
sudo systemctl enable nftables.service
sudo systemctl start nftables.service
sudo systemctl status nftables.service
```

```
Created symlink '/etc/systemd/system/sysinit.target.wants/nftables.service' → '/usr/lib/systemd/system/nftables.service'.
# nftables.service - nftables
     Loaded: loaded (/usr/lib/systemd/system/nftables.service; enabled; preset: enabled)
     Active: active (exited) since Wed 2024-09-18 19:47:08 CEST; 38ms ago
 Invocation: 91e0edd0be824dd89b4f9bed0bef6fce
       Docs: man:nft(8)
             http://wiki.nftables.org
    Process: 1066 ExecStart=/usr/sbin/nft -f /etc/nftables.conf (code=exited, status=0/SUCCESS)
   Main PID: 1066 (code=exited, status=0/SUCCESS)
   Mem peak: 3.9M
        CPU: 31ms

sept. 18 19:47:07 hub systemd[1]: Starting nftables.service - nftables...
sept. 18 19:47:08 hub systemd[1]: Finished nftables.service - nftables.
```

Q43. Comment caractériser le fonctionnement de la traduction d'adresses sources ?

Rechercher dans les pages de manuel de la commande nftables les options d'affichage du décompte du trafic traité.

Voici un exemple d'affichage des règles actives avec visualisation des compteurs d'utilisation.

```
sudo nft list ruleset
```

```
table inet nat {
        chain postrouting {
                type nat hook postrouting priority srcnat; policy accept;
                oifname "enp0s1.360" counter packets 0 bytes 0 masquerade
        }
}
```

5.4. Activation du service PPPoE

Dans le scénario de ces travaux pratiques, il est nécessaire de passer par une authentification pour acheminer le trafic réseau entre les routeurs ***Hub*** et ***Spoke***. Cette fonction est assurée à l'aide du protocole PPP.

Le protocole PPP n'a pas été conçu suivant le modèle client/serveur mais pair à pair. Il suppose que deux les processus pairs de la liaison point à point échangent des informations, dont l'authentification mutuelle.

Dans notre cas, le routeur qui tient le rôle ***Hub*** joue le rôle de serveur dans le sens où il exige que le routeur avec le rôle ***Spoke*** s'authentifie auprès de lui avant de fournir les adresses de couche réseau.

Q44. Quel paquet spécifique à la gestion du dialogue PPPoE à installer sur le routeur ***Hub*** ?

Rechercher dans le catalogue des paquets, la référence pppoe.

```
apt search ^pppoe
```

```
pppoe/testing 4.0-1 amd64
  Pilote PPP sur Ethernet
```

Le résultat de la commande apt show pppoe montre que c'est bien ce paquet qui répond au besoin. On peut donc l'installer.

```
sudo apt -y install pppoe
```

Q45. Quel est l'outil contenu dans le paquet demandé à la question précédente qui assure le rôle de serveur PPPoE ?

Rechercher dans la liste des outils fournis avec le paquet et les pages de manuels.

L'outil pppoe-server gère directement l'encapsulation des trames PPP dans les trames Ethernet. Il communique ensuite les paramètres utiles au démon pppd qui fonctionne de façon totalement transparente vis-à-vis de la technologie du réseau sous-jacent.

Q46. Quels sont les noms des deux sous-couches du protocole PPP qui apparaissent dans les journaux systèmes ?

Quels sont les rôles respectifs de ces deux sous-couches ?

Consulter la page *Point-to-Point Protocol*.

La consultation des journaux système lors du dialogue PPP fait apparaître les différents protocoles utilisés lors de l'ouverture de session.

Exemple de commande de consultation des journaux système.

```
journalctl -n 200 -f -u pppoe-server.service
```

1. LCP : *Link Control Protocol*
2. IPCP : *IP control protocol* pour la version IPv4
3. IPV6CP : *IP control protocol* pour la version IPv6

Des exemples complets de traces d'établissement de connexion PPP sont donnés à l'adresse : *Traces d'une ouverture de session PPPoE*.

Q47. Quels sont les en-têtes du dialogue qui identifient les requêtes (émises|reçues), les rejets et les acquittements ?

Consulter les journaux système contenant les traces d'une connexion PPP.

La copie d'écran donnée ci-dessus fait apparaître les directives `Conf*` pour chaque paramètre négocié.

- `ConfReq` indique une requête.
- `ConfAck` indique un acquittement.
- `ConfNak` indique un rejet.

Q48. Dans quel fichier sont stockés les paramètres d'identité et d'authentification utilisés par le protocole EAP pour la méthode CHAP ?

Consulter les pages de manuels du démon pppd à la section *AUTHENTICATION*.

C'est le fichier `/etc/ppp/chap-secrets` qui contient les couples *login/password* utilisés lors de l'authentification.

Voici un exemple du contenu de ce fichier.

```
# Secrets for authentication using CHAP
# client     server secret       IP addresses
"spoke_site0"   *       "5p0k3"      *
```

On peut le compléter directement à l'aide de la commande suivante.

```
echo '"spoke_site0"   *       "5p0k3"   *' | sudo tee -a /etc/ppp/chap-secrets
```

Q49. Dans quel fichier sont stockés les paramètres passés au démon pppd lors du lancement du serveur PPPoE ?

Consulter les pages de manuels de l'outil pppoe-server.

C'est le fichier `/etc/ppp/pppoe-server-options` qui contient la liste des paramètres utilisés lors du dialogue PPP.

Q50. Quelles sont les options du protocole PPP qui doivent être implantées dans le fichier demandé à la question précédente ?

Consulter les pages de manuels du démon pppd et rechercher les paramètres correspondant à la liste suivante.

- Afficher en détail toutes les étapes d'établissement de session dans les journaux système.
- Référencer l'identifiant du compte utilisateur à utiliser lors de l'authentification du routeur vert. Cette option implique que le compte utilisateur existe sur le système et qu'il soit présent dans le fichier `/etc/ppp/chap-secrets`.
- Imposer au routeur vert une authentification via EAP (*Extensible Authentication Protocol*) sans utiliser les certificats TLS pour simplifier la configuration.
- Préserver la route par défaut, et donc l'accès Internet, du routeur bleu.
- Publier l'adresse IP du serveur DNS à utiliser pour la résolution des noms de domaines.
- Activer l'utilisation des protocoles IPv6CP et IPv6.

Voici une copie de la commande de création du fichier `/etc/ppp/pppoe-server-options` qui contient la liste des paramètres demandés.

```
cat << 'EOF' | sudo tee /etc/ppp/pppoe-server-options
# Gestion de session avec PAM
login
# Authentification EAP
require-eap
# Le Routeur Hub détient déjà une route par défaut
nodefaultroute
# Envoi de l'adresse de résolution DNS avec les adresses IPv4
ms-dns 172.16.0.2
# Ajout du protocole IPv6
+ipv6
# Informations détaillées dans la journalisation
debug
# Options préconisées par la documentation
noaccomp
default-asyncmap
nodeflate
nopcomp
novj
novjccomp
lcp-echo-interval 10
EOF
```

Q51. Comment créer le compte utilisateur local sur le routeur bleu sachant qu'il n'est autorisé ni à se connecter ni à avoir un répertoire personnel ?

Consulter les options de la commande adduser.

Voici un exemple de commande adduser.

```
sudo adduser --gecos 'Spoke Router site0' --disabled-login --no-create-home spoke_site0
```

Avertissement

Cet utilisateur doit porter le même nom que celui défini dans le fichier `/etc/ppp/chap-secrets`.

Q52. Quels sont les paramètres à donner au lancement de l'outil pppoe-server pour qu'il délivre les adresses au routeur vert après authentification de celui-ci ?

Consulter les options de la commande pppoe-server.

Voici un exemple de lancement manuel de la commande pppoe-server.

```
sudo pppoe-server -I enp0s1.441 -C BRAS -L 10.4.41.1 -R 10.4.41.2 -N 1
```

Cette commande individuelle est à utiliser pour faire un tout premier test. Pour rendre la configuration persistante au redémarrage nous avons besoin de créer un service systemd. Il ne faut donc pas oublier d'arrêter le processus avant de passer à la question suivante.

```
sudo killall pppoe-server
```

Q53. Comment créer une nouvelle unité systemd responsable du lancement du processus pppoe-server ?

Consulter la page *systemd Services* et rechercher la procédure à suivre pour ajouter un service au lancement du système.

On commence par la création du fichier de service appelé : `/etc/systemd/system/pppoe-server.service` qui contient toutes les directives de lancement du processus pppoe-server avec les paramètres d'adressage du lien point à point.

Dans l'exemple ci-dessous, les paramètres suivants doivent être édités pour respecter le plan d'adressage défini.

-INom d'interface réseau

Le nom d'interface réseau contient l'identifiant du VLAN sur lequel la session PPP doit être établie.

-LAdresse locale

L'adresse locale correspond à l'adresse attribuée au routeur ***Hub*** par le démon pppd.

-RAdresse distante

L'adresse distante correspond à l'adresse attribuée au routeur ***Spoke*** par le démon pppd après authentification.

Voici un exemple de création du fichier d'unité systemd.

```
cat << 'EOF' | sudo tee /etc/systemd/system/pppoe-server.service
[Unit]
Description=PPPoE Server
After=systemd-networkd.service
Wants=systemd-networkd.service
BindsTo=sys-subsystem-net-devices-enp0s1.441.device
After=sys-subsystem-net-devices-enp0s1.441.device

[Service]
Type=forking
ExecCondition=/bin/sh -c '[ "$(systemctl show --property MainPID --value pppoe-server.service)" = "0" ]'
ExecCondition=/bin/sh -c '[ -z "$(pgrep pppoe-server)" ]'
ExecStart=/usr/sbin/pppoe-server -I enp0s1.441 -C BRAS -L 10.4.41.1 -R 10.4.41.2 -N 1
Restart=on-failure
RestartSec=5

[Install]
WantedBy=multi-user.target
EOF
```

Q54. Comment activer le nouveau service et contrôler son état après lancement ?

Consulter la page *systemd Services* et rechercher la procédure à suivre pour activer et lancer un service.

On commence par la relecture de la liste des services disponibles par le gestionnaire systemd.

```
sudo systemctl daemon-reload
```

On active le nouveau service.

```
sudo systemctl enable pppoe-server.service
```

On lance ce nouveau service.

```
sudo systemctl start pppoe-server.service
```

On vérifie que l'opération s'est déroulée correctement.

```
systemctl status pppoe-server.service
```

```
# pppoe-server.service - PPPoE Server
     Loaded: loaded (/etc/systemd/system/pppoe-server.service; enabled; preset: enabled)
     Active: active (running) since Sat 2024-09-21 15:39:39 CEST; 36min ago
 Invocation: 23332f16c26f4889a9f063870e77da9c
    Process: 2958 ExecCondition=/bin/sh -c [ "$(systemctl show --property MainPID --value pppoe-server.service)" = "0" ] (c
    Process: 2960 ExecCondition=/bin/sh -c [ -z "$(pgrep pppoe-server)" ] (code=exited, status=0/SUCCESS)
    Process: 2964 ExecStart=/usr/sbin/pppoe-server -I enp0s1.441 -C BRAS -L 10.4.41.1 -R 10.4.41.2 -N 1 (code=exited, statu
   Main PID: 2966 (pppoe-server)
      Tasks: 1 (limit: 1086)
     Memory: 208K (peak: 1.8M)
        CPU: 67ms
     CGroup: /system.slice/pppoe-server.service
             └─2966 /usr/sbin/pppoe-server -I enp0s1.441 -C BRAS -L 10.4.41.1 -R 10.4.41.2 -N 1

sept. 21 15:39:39 hub systemd[1]: Starting pppoe-server.service - PPPoE Server...
sept. 21 15:39:39 hub systemd[1]: Started pppoe-server.service - PPPoE Server.
```

la copie d'écran ci-dessus montre que le service pppoe-server est lancé avec le jeu de paramètres de la maquette.

En l'état actuel de la configuration, aucune session PPP n'a encore été établie. Il faut maintenant passer à la configuration réseau du routeur ***Spoke*** pour avancer dans l'utilisation du protocole PPP.

6. Routeur Spoke

Dans cette section, on étudie la machine virtuelle qui joue le rôle de routeur entre le réseau étendu (VLANs violet et orange) et le réseau d'hébergement des conteneurs (VLAN vert) du site distant.

6.1. Configuration des interfaces du routeur

Une fois la machine virtuelle serveur de conteneurs lancée, les premières étapes consistent à lui attribuer un nouveau nom et à configurer les interfaces réseau pour joindre le routeur voisin et l'Internet.

Q55. Comment changer le nom de la machine virtuelle ?

Il faut éditer les deux fichiers `/etc/hosts` et `/etc/hostname` en remplaçant le nom de l'image maître ***vm0*** par le nom voulu. Il est ensuite nécessaire de redémarrer pour que le nouveau nom soit pris en compte par tous les outils du système.

```
etu@localhost:~$ sudo hostnamectl hostname spoke
etu@localhost:~$ sudo reboot
```

Q56. Comment appliquer les configurations réseau IPv4 et IPv6 à partir de l'unique interface du routeur ?

Consulter la documentation de ***Netplan*** pour obtenir les informations sur la configuration des interfaces réseau à l'adresse *Netplan documentation*.

Il existe plusieurs possibilités pour configurer une interface réseau. Dans le contexte de ces manipulations, on utilise ***Netplan*** dans le but de séparer la partie déclarative du moteur de configuration.

C'est `systemd-networkd` qui joue le rôle de moteur de configuration sur les machines virtuelles utilisées avec ces manipulations.

La configuration de base fournie avec l'image maître suppose que l'interface obtienne un bail DHCP pour la partie IPv4 et une configuration automatique via SLAAC pour la partie IPv6. Cette configuration par défaut doit être éditée et remplacée. Il faut configurer trois interfaces.

- L'interface principale correspond à l'interface "physique" de la machine. Elle est nommée `enp0s1` en fonction de l'ordre des adresses des composants raccordés au bus PCI.
- Une sous-interface doit être créée pour le réseau étendu de l'exploitant des fourreaux et du câblage en fibres optiques (VLAN violet). Cette interface ne comprend qu'une adresse IPv6 de lien local pour les tests de connectivité ICMPv6 entre les deux sites.
- Une autre sous-interface doit être créée pour le réseau étendu de l'opérateur (VLAN orange). Cette interface ne contient aucune adresse lors de l'initialisation système. C'est le démon pppd qui est responsable de l'attribution des adresses IPv4 et IPv6 lotrs de l'établissement de la session du protocole PPP.
- Une sous-interface temporaire doit être créée dans le but d'installer les paquets d'outils nécessaires à l'établissement de la session du protocole PPP. Dès que la configuration de session PPP est en place, cette interface doit être détruite pour éviter toute confusion sur l'acheminement du trafic.
- Une sous-interface devra être créée par la suite pour le réseau d'hébergement des conteneurs avec, là encore, le bon numéro de VLAN. Les adresses IPv4 et IPv6 de cette interface deviendront les passerelles du serveur et des conteneurs.

Voici une copie du fichier `/etc/netplan/enp0s1.yaml` de la maquette.

```
network:
  version: 2
  ethernets:
    enp0s1:
      dhcp4: false
      dhcp6: false
      accept-ra: false
      nameservers:
        addresses:
          - 172.16.0.2
          - 2001:678:3fc:3::2

  vlans:
    enp0s1.440: # VLAN violet
      id: 440
      link: enp0s1
      addresses:
        - fe80:1b8::2/64
    enp0s1.441: # VLAN orange
      id: 441
      link: enp0s1
      addresses: []
    enp0s1.52: # VLAN accès temporaire
      id: 52
      link: enp0s1
      dhcp4: true
      dhcp6: false
      accept-ra: true
```

6.2. Activation de la fonction routage

Sans modification de la configuration par défaut, un système GNU/Linux n'assure pas la fonction de routage du trafic d'une interface réseau à une autre.

L'activation du routage correspond à un réglage de paramètres du sous-système réseau du noyau Linux. L'outil qui permet de consulter et modifier les réglages de paramètre sur le noyau est appelé sysctl.

On retrouve ici les mêmes opérations que celles effectuées sur le routeur qui tient le rôle ***Hub***.

Q57. Comment activer le routage dans le sous-système réseau du noyau Linux ?

Utiliser la commande sysctl pour effectuer des recherches et identifier les paramètres utiles. Par exemple :

```
sudo sysctl -a -r ".*forward.*"
```

Il est dorénavant recommandé de créer un fichier de configuration spécifique par fonction. C'est la raison pour laquelle on crée un nouveau fichier `/etc/sysctl.d/10-routing.conf`.

Attention ! Il ne faut pas oublier d'appliquer les nouvelles valeurs des paramètres de configuration.

```
cat << EOF | sudo tee /etc/sysctl.d/10-routing.conf
net.ipv4.ip_forward=1
net.ipv6.conf.all.forwarding=1
net.ipv4.conf.all.log_martians=1
EOF
```

Voici un exemple des résultats obtenus après application des nouveaux paramètres.

```
sudo sysctl --system
```

6.3. Configurer le protocole PPP

Le routeur ***Spoke*** doit utiliser un démon pppd sur le VLAN `Data` (orange) pour établir une session PPP avec le routeur ***Hub***. À la différence de ce dernier, il n'est pas à l'initiative du dialogue PPPoE mais il doit être capable de gérer l'encapsulation des trames PPP sur un réseau local Ethernet.

Q58. Quel paquet fournit le démon de gestion des sessions du protocole PPP sur le routeur ***Spoke*** ?

Rechercher dans le catalogue des paquets, la référence ppp.

```
apt search ^ppp
```

```
ppp/testing 2.5.0-1+2 amd64
  protocole point à point (PPP) - démon

ppp-dev/testing 2.5.0-1+2 all
  protocole point à point (PPP) – fichiers de développement

ppp-gatekeeper/testing 0.1.0-201406111015-1.1 all
  PPP manager for handling balanced, redundant and failover links

pppoe/testing 4.0-1 amd64
  Pilote PPP sur Ethernet

pppoeconf/testing 1.21+nmu3 all
  configures PPPoE/ADSL connections

wmppp.app/testing 1.3.2-2 amd64
  contrôle de connexion et surveillance de la charge réseau avec aspect NeXTStep
```

Le résultat de la commande apt show ppp montre que c'est bien ce paquet qui répond au besoin.

```
sudo apt -y install ppp
```

Q59. Comment utiliser l'encapsulation des trames PPP dans Ethernet à partir du démon pppd fourni avec le paquet ppp ?

Rechercher dans le répertoire de documentation du paquet ppp.

Dans le répertoire `/usr/share/doc/ppp/`, on trouve le fichier `README.pppoe` qui indique que l'appel au module `rp-pppoe.so` permet d'encapsuler des trames PPP sur un réseau local Ethernet.

Toujours à partir du même répertoire, on trouve dans la liste des fichiers d'exemples de configuration un modèle adapté à notre contexte : `peers-pppoe`.

Q60. Dans quel fichier sont stockés les paramètres d'identité et d'authentification utilisés par le protocole CHAP ?

Consulter les pages de manuels du démon pppd à la section *AUTHENTICATION*.

C'est le fichier `/etc/ppp/chap-secrets` qui contient les couples ***login/password*** utilisés lors de l'authentification.

Voici un exemple du contenu de ce fichier. Le nom du client ainsi que son mot de passe secret doivent être identiques à chaque extrémité de la session PPP.

```
# Secrets for authentication using CHAP
# client server secret   IP addresses
"spoke_site0"   *  "5p0k3"   *
```

Q61. Quelles sont les options de configuration du démon pppd à placer dans le fichier `/etc/ppp/peers/pppoe-provider` pour assurer l'établissement de la session PPP entre les routeurs ?

Utiliser le fichier exemple PPPoE fourni avec la documentation du paquet ppp.

Voici comment créer un fichier `/etc/ppp/peers/pppoe-provider` avec les options correspondant au contexte de la maquette du routeur vert.

```
cat << 'EOF' | sudo tee /etc/ppp/peers/pppoe-provider
# Le nom d'utilisateur désigne l'entrée du fichier /etc/ppp/chap-secrets
user spoke_site0

# Chargement du module PPPoE avec les détails dans la journalisation
plugin rp-pppoe.so rp_pppoe_ac BRAS rp_pppoe_verbose 1

# Interface (VLAN) utilisé pour l'établissement de la session PPP
enp0s1.441

# Les adresses sont attribuées par le "serveur" PPPoE
noipdefault
# L'adresse de résolution DNS est aussi fournie par le serveur PPPoE
usepeerdns
# La session PPP devient la route par défaut du routeur Spoke
defaultroute

# Demande de réouverture de session automatique en cas de rupture
persist

# Le routeur Spoke n'exige pas que le routeur Hub s'authentifie
noauth

# Messages d'informations détaillés dans la journalisation
debug

# Utilisation du protocole IPv6
+ipv6

# Options préconisées par la documentation
noaccomp
default-asyncmap
nodeflate
nopcomp
novj
novjccomp
lcp-echo-interval 10
EOF
```

Q62. Comment lancer le démon pppd pour qu'il prenne en compte les paramètres définis dans le fichier complété à la question précédente ?

Consulter les pages de manuels du démon pppd.

C'est l'outil pon qui permet de désigner le fichier de configuration à utiliser. Voici une copie d'écran du lancement du démon pppd.

```
sudo pon pppoe-provider
```

Cette commande individuelle est à utiliser pour faire un tout premier test. Pour rendre la configuration persistante au redémarrage nous avons besoin de créer un service systemd. Il ne faut donc pas oublier d'arrêter le processus avant de passer à la question suivante. Le paquet fournit un outil dédié : poff.

```
sudo poff -a pppoe-provider
```

Q63. Quels sont les noms des deux sous-couches du protocole PPP qui apparaissent dans les journaux systèmes ? Quels sont les rôles respectifs de ces deux sous-couches ?

Consulter la page *Point-to-Point Protocol*.

La consultation des journaux système lors du dialogue PPP fait apparaître tous les détails. Voir les exemples de traces à l'adresse : *Traces d'une ouverture de session PPPoE*.

Q64. Quels sont les en-têtes du dialogue qui identifient les requêtes (émises|reçues), les rejets et les acquittements ?

Consulter les journaux système contenant les traces d'une connexion PPP.

La copie d'écran donnée ci-dessus fait apparaître les directives `Conf*` pour chaque paramètre négocié.

- `ConfReq` indique une requête.
- `ConfAck` indique un acquittement.

- `ConfNak` indique un rejet.

Q65. Comment assurer une ouverture automatique de la session PPP à chaque réinitialisation système ?

Consulter la page *systemd Services* et rechercher la procédure à suivre pour ajouter un service au lancement du système.

On commence par la création du fichier de service appelé : `/etc/systemd/system/ppp.service` qui contient les appels aux outils pon et poff.

Voici l'instruction de création du fichier de service.

```
cat << EOF | sudo tee /etc/systemd/system/ppp.service
[Unit]
Description=PPPoE Client Connection
After=network.target
Wants=network.target
BindsTo=sys-subsystem-net-devices-enp0s1.441.device
After=sys-subsystem-net-devices-enp0s1.441.device

[Service]
Type=forking
ExecStart=/usr/bin/pon pppoe-provider
ExecStop=/usr/bin/poff pppoe-provider
Restart=on-failure
RestartSec=20

[Install]
WantedBy=multi-user.target
EOF
```

Q66. Comment activer le nouveau service et contrôler son état après lancement ?

Consulter la page *systemd Services* et rechercher la procédure à suivre pour activer et lancer un service.

On commence par la relecture de la liste des services disponibles par le gestionnaire systemd.

```
sudo systemctl daemon-reload
```

On active le nouveau service.

```
sudo systemctl enable ppp.service
```

On lance ce nouveau service.

```
sudo systemctl start ppp.service
```

On vérifie que l'opération s'est déroulée correctement.

```
systemctl status ppp.service
```

```
# ppp.service - PPPoE Client Connection
     Loaded: loaded (/etc/systemd/system/ppp.service; enabled; preset: enabled)
     Active: active (running) since Sun 2024-09-22 08:21:32 CEST; 13min ago
 Invocation: 69386a40f7574821b3986ed6c6c242f7
   Main PID: 496 (pppd)
      Tasks: 1 (limit: 1086)
     Memory: 2.8M (peak: 4.9M)
        CPU: 56ms
     CGroup: /system.slice/ppp.service
             └─496 /usr/sbin/pppd call pppoe-provider

sept. 22 08:21:37 spoke pppd[496]: rcvd [IPCP ConfAck id=0x2 <addr 10.4.41.2> <ms-dns1 172.16.0.2> <ms-dns2 172.16.0.2>]
sept. 22 08:21:37 spoke pppd[496]: Script /etc/ppp/ip-pre-up started (pid 510)
sept. 22 08:21:37 spoke pppd[496]: Script /etc/ppp/ip-pre-up finished (pid 510), status = 0x0
sept. 22 08:21:37 spoke pppd[496]: local  IP address 10.4.41.2
sept. 22 08:21:37 spoke pppd[496]: remote IP address 10.4.41.1
sept. 22 08:21:37 spoke pppd[496]: primary   DNS address 172.16.0.2
sept. 22 08:21:37 spoke pppd[496]: secondary DNS address 172.16.0.2
sept. 22 08:21:37 spoke pppd[496]: Script /etc/ppp/ip-up started (pid 514)
sept. 22 08:21:37 spoke pppd[496]: Script /etc/ppp/ipv6-up finished (pid 509), status = 0x0
sept. 22 08:21:37 spoke pppd[496]: Script /etc/ppp/ip-up finished (pid 514), status = 0x0
```

Q67. Comment utiliser la session PPP (le VLAN orange) comme lien unique de raccordement réseau du routeur *Spoke* ?

Maintenant que le fonctionnement de la session PPP est validé, nous n'avons plus besoin du raccordement temporaire sur le routeur ***Spoke***. Il faut donc commenter les entrées du fichier `/etc/netplan/enp0s1.yaml` qui ne sont plus utiles et attribuer l'adresse de résolution DNS de secours.

Une fois ces opérations effectuées, on peut redémarre le routeur ***Spoke*** pour se placer en situation de raccordement distant.

Pour commencer, on commente les entrées inutile du fichier `/etc/netplan/enp0s1.yaml`.

```
cat /etc/netplan/enp0s1.yaml
```

```
network:
  version: 2
  ethernets:
    enp0s1:
      dhcp4: false
      dhcp6: false
      accept-ra: false
#     nameservers:
#       addresses:
#          - 172.16.0.2
#          - 2001:678:3fc:3::2

  vlans:
    enp0s1.440: # VLAN violet
      id: 440
      link: enp0s1
      addresses:
        - fe80:1b8::2/64
    enp0s1.441: # VLAN orange
      id: 441
      link: enp0s1
      addresses: []
#   enp0s1.52: # VLAN accès temporaire
#     id: 52
#     link: enp0s1
#     dhcp4: true
#     dhcp6: false
#     accept-ra: true
```

On peut appliquer directement les modifications à l'aide de la commande netplan.

```
sudo netplan apply
```

```
sudo netplan status
```

Attention

L'affectation de l'adresse IPv4 ou IPv6 de résolution DNS pose problème. En effet, si le démon pppd propose bien deux adresses via l'option `usepeerdns`, ces propositions ne sont pas prises en charge par le service systemd-resolved.

On contourne cette difficulté en affectant une adresse IPv4 directement au service systemd-resolved.

On édite le fichier `/etc/systemd/resolved.conf` pour affecter directement l'adresse de résolution DNS. Voici une copie des lignes utiles du fichier modifié. Toutes les autres lignes sont commentées.

```
grep -Ev '(^#|^$)'  /etc/systemd/resolved.conf
[Resolve]
DNS=172.16.0.2
```

Il ne faut pas oublier de relancer le service pour prendre en compte les modifications du fichier.

```
sudo systemctl restart systemd-resolved
```

Le routeur ***Spoke*** est maintenant prêt à être redémarré pour utiliser le lien de raccordement distant comme seul canal d'accès aux autres réseaux.

```
sudo reboot
```

7. Réseau d'hébergement de conteneurs

À ce stade des manipulations, le routeur ***Spoke*** utilise la session PPP et le routage IPv4 pour accéder à tous les réseaux.

On peut le vérifier en affichant les tables de routage IPv4 et IPv6.

Dans la table de routage IPv4, on trouve la route par défaut.

```
ip route ls
```

```
default dev ppp0 scope link
10.4.41.1 dev ppp0 proto kernel scope link src 10.4.41.2
```

Dans la table de routage IPv6, on ne trouve que des entrées de lien local correspondant au préfixe `fe80::/10`.

```
ip -6 route ls
```

```
fe80::d1b3:8c16:93d1:8370 dev ppp0 proto kernel metric 256 pref medium
fe80::f118:5b7b:cfb5:7b54 dev ppp0 proto kernel metric 256 pref medium
fe80::/64 dev enp0s1 proto kernel metric 256 pref medium
fe80::/64 dev enp0s1.441 proto kernel metric 256 pref medium
fe80::/64 dev enp0s1.440 proto kernel metric 256 pref medium
fe80:1b8::/64 dev enp0s1.440 proto kernel metric 256 pref medium
```

Dans cette partie, nous devons ajouter un commutateur pour raccorder les services hébergés sur le site distant et compléter la configuration du routage pour assurer les accès IPv4 et IPv6.

7.1. Ajouter un commutateur virtuel

Dans le scénario étudié, les services sont hébergés dans un réseau de conteneurs propre au routeur ***Spoke***. La mise en œuvre de cette configuration passe par l'installation d'un commutateur virtuel appelé `asw-host`. On utilise Open vSwitch pour configurer ce commutateur.

Q68. Quel est le paquet à installer pour ajouter un commutateur virtuel au routeur ***Spoke*** ?

Rechercher le mot clé openvswitch dans la liste des paquets.

Voici un exemple de recherche.

```
apt search ^openvswitch
```

C'est le paquet `openvswitch-switch` qui nous intéresse. On l'installe.

```
sudo apt -y install openvswitch-switch
```

Q69. Comment déclarer un commutateur à l'aide de l'outil netplan.io ?

Consulter la documentation de ***Netplan*** pour obtenir les informations sur la configuration des commutateurs virtuels `openvswitch` à l'adresse *Netplan documentation*.

On peut aussi rechercher les informations dans les fichiers exemples fournis avec le paquet netplan.io.

Voici un exemple de recherche.

```
find /usr/share/doc/netplan* -type f -iname "openvswitch*"
/usr/share/doc/netplan/examples/openvswitch.yaml
```

Q70. Quelles sont les modifications à apporter au fichier de déclaration YAML `/etc/netplan/enp0s1.yaml` pour créer le commutateur `asw-host` ?

Voici une copie du fichier `/etc/netplan/enp0s1.yaml` qui contient les instructions de création du commutateur `asw-host` seul.

```
network:
  version: 2
  ethernets:
    enp0s1:
      dhcp4: false
      dhcp6: false
      accept-ra: false

    openvswitch: {}

  bridges:
    asw-host:
      openvswitch: {}

  vlans:
    enp0s1.440: # VLAN violet
      id: 440
      link: enp0s1
      addresses:
        - fe80:1b8::2/64
    enp0s1.441: # VLAN orange
      id: 441
      link: enp0s1
      addresses: []
```

On applique les nouvelles déclarations.

```
sudo netplan apply
```

On vérifie que le nouveau commutateur a bien été créé dans la base `Open vSwitch`.

```
sudo ovs-vsctl show
```

```
e288cc30-e290-44ae-8ed1-5e2a8d184033
    Bridge asw-host
        fail_mode: standalone
        Port asw-host
            Interface asw-host
                type: internal
    ovs_version: "3.4.0"
```

Q71. Comment ajouter une nouvelle interface virtuelle commutée (***Switched Virtual Interface***) qui servira de passerelle par défaut pour tous les hôtes du réseau d'hébergement du site distant ?

Rechercher dans la documentation Netplan des exemples de déclarations d'interfaces de type SVI appartenant à des VLANs.

Voici une nouvelle copie du fichier `/etc/netplan/enp0s1.yaml` auquel on ajouté la déclaration d'une interface `vlan40` avec les adresses IPv4 et IPv6 conformes au contexte de la maquette utilisée pour la rédaction de ce document.

```
network:
  version: 2
  ethernets:
    enp0s1:
      dhcp4: false
      dhcp6: false
      accept-ra: false

  openvswitch: {}

  bridges:
    asw-host:
      openvswitch: {}

  vlans:
    enp0s1.440: # VLAN violet
      id: 440
      link: enp0s1
      addresses:
        - fe80:1b8::2/64
    enp0s1.441: # VLAN orange
      id: 441
      link: enp0s1
      addresses: []
    vlan40:     # VLAN vert
      id: 40
      link: asw-host
      addresses:
        - 203.0.113.1/24
        - fda0:7a62:28::1/64
        - fe80:28::1/64
```

7.2. Routage du réseau d'hébergement

L'objectif de cette section est de rendre le réseau d'hébergement accessible depuis le routeur ***Hub*** et que le protocole IPv6 soit utilisable depuis le routeur ***Spoke***.

Pour rendre le réseau d'hébergement du site distant accessible depuis le routeur ***Hub***, il est nécessaire d'ajouter des routes statiques IPv4 et IPv6 à l'ouverture de la session PPP.

Pour utiliser IPv6 depuis le routeur ***Spoke***, il faut ajouter une route par défaut IPv6 aussi à l'ouverture de session PPP.

On commence par l'ajout de routes statiques IPv4 et IPv6 côté routeur ***Hub***.

Q72. Comment ajouter manuellement les routes IPv4 et IPv6 vers le réseau desservi par le routeur vert ?

Consulter les pages de manuel sur le routage avec la commande : `man ip-route`.

Sachant que le site distant est raccordé via une liaison point à point unique, on choisit de désigner la destination par l'interface de la liaison.

```
sudo ip route add 203.0.113.0/24 dev ppp0
sudo ip -6 route add fda0:7a62:28::/64 dev ppp0
```

Q73. Comment appliquer ces routes statiques dans la configuration système pour qu'elles soient activées à chaque établissement de session PPP ?

Il faut parcourir l'arborescence du répertoire `/etc/ppp/` pour repérer les scripts exécutés lors de l'ouverture de session. Créer un script pour chaque protocole de couche réseau qui ajoute la route statique voulue.

- Pour IPv4, le répertoire est `/etc/ppp/ip-up.d/`. Voici comment créer le script exécutable `staticroute`.

```
cat << 'EOF' | sudo tee /etc/ppp/ip-up.d/staticroute
#!/bin/sh

if [ -z "${CONNECT_TIME}" ]; then
    ip route add 203.0.113.0/24 dev ${PPP_IFACE}
fi
EOF
```

```
sudo chmod +x /etc/ppp/ip-up.d/staticroute
```

- Pour IPv6, le répertoire est `/etc/ppp/ipv6-up.d/`. Voici comment créer le script exécutable `staticroute`.

```
cat << 'EOF' | sudo tee /etc/ppp/ipv6-up.d/staticroute
#!/bin/sh

if [ -z "${CONNECT_TIME}" ]; then
    ip -6 route add fda0:7a62:28::/64 dev ${PPP_IFACE}
fi
EOF
```

```
sudo chmod +x /etc/ppp/ipv6-up.d/staticroute
```

Q74. Comment tester l'ajout de ces routes statiques et les communications vers le réseau d'hébergement depuis le routeur ***Hub*** ?

Afficher les tables de routage après réinitialisation d'une session PPP et lancer des tests ICMP ***vers*** les adresses de l'interface virtuelle commutée du routeur Spoke.

En redémarrant le service `pppoe-server` sur le routeur ***Hub*** ou le service `ppp` sur le routeur ***Spoke***, on provoque un renouvellement de session PPP.

On peut ensuite afficher les tables de routage du routeur ***Hub***.

```
ip route ls
default via 192.168.104.129 dev enp0s1.360 proto static
10.4.41.2 dev ppp0 proto kernel scope link src 10.4.41.1
192.168.104.128/29 dev enp0s1.360 proto kernel scope link src 192.168.104.130
203.0.113.0/24 dev ppp0 scope link
```

```
ip -6 route ls
2001:678:3fc:168::/64 dev enp0s1.360 proto kernel metric 256 pref medium
2001:678:3fc:168::/64 dev enp0s1.360 proto ra metric 512 expires 2591839sec pref high
fda0:7a62:28::/64 dev ppp0 metric 1024 pref medium
fe80::d580:e038:8d05:636e dev ppp0 proto kernel metric 256 pref medium
fe80::dcfe:544d:d6ac:8b0f dev ppp0 proto kernel metric 256 pref medium
fe80::/64 dev enp0s1 proto kernel metric 256 pref medium
fe80::/64 dev enp0s1.441 proto kernel metric 256 pref medium
fe80::/64 dev enp0s1.440 proto kernel metric 256 pref medium
fe80::/64 dev enp0s1.360 proto kernel metric 256 pref medium
fe80:1b8::/64 dev enp0s1.440 proto kernel metric 256 pref medium
default via fe80:168::1 dev enp0s1.360 proto static metric 1024 onlink pref medium
```

On peut aussi afficher la solution de routage pour une adresse destination.

```
ip route get 203.0.113.1
203.0.113.1 dev ppp0 src 10.4.41.1 uid 1000
    cache
```

```
ip -6 route get fda0:7a62:28::1
fda0:7a62:28::1 from :: dev ppp0 src 2001:678:3fc:168:baad:caff:fefe:5 metric 1024 pref medium
```

On peut maintenant passer au routeur ***Spoke*** pour effectuer le même travail sur les routes par défaut.

Q75. Comment ajouter des routes par défaut dans la configuration système pour qu'elles soient activées à chaque établissement de session PPP ?

Il faut parcourir l'arborescence du répertoire `/etc/ppp/` pour repérer les scripts exécutés lors de l'ouverture de session. Créer un script pour chaque protocole de couche réseau qui ajoute la route statique voulue.

- Pour IPv4, le répertoire est `/etc/ppp/ip-up.d/`. Voici comment créer un script exécutable appelé `defaultroute`.

```
cat << 'EOF' | sudo tee /etc/ppp/ip-up.d/defaultroute
#!/bin/sh

if [ -z "${CONNECT_TIME}" ]; then
 ip route add default dev ${PPP_IFACE}
fi
EOF
```

```
sudo chmod +x /etc/ppp/ip-up.d/defaultroute
```

- Pour IPv6, le répertoire est `/etc/ppp/ipv6-up.d/`. Voici comment créer un script exécutable aussi appelé`defaultroute`.

```
cat << 'EOF' | sudo tee /etc/ppp/ipv6-up.d/defaultroute
#!/bin/sh

  if [ -z "${CONNECT_TIME}" ]; then
      ip -6 route add default dev ${PPP_IFACE}
  fi
EOF
```

```
sudo chmod +x /etc/ppp/ipv6-up.d/defaultroute
```

Q76. Comment tester l'ajout des routes par défaut et les communications IPv6 depuis le routeur ***Spoke*** ?

Afficher les tables de routage après réinitialisation d'une session PPP et lancer des tests ICMP ***depuis*** les adresses de l'interface virtuelle commutée du routeur Spoke.

En redémarrant le service `pppoe-server` sur le routeur ***Hub*** ou le service `ppp` sur le routeur ***Spoke***, on provoque un renouvellement de session PPP.

On peut ensuite afficher les tables de routage du routeur ***Spoke***.

```
ip route ls default
default dev ppp0 scope link
```

```
ip -6 route ls default
default dev ppp0 metric 1024 pref medium
```

Il ne reste plus que le test ICMPv6 pour qualifier le routage complet au niveau du routeur ***Spoke***.

```
ping -c2 2620:fe::fe
PING 2620:fe::fe (2620:fe::fe) 56 data bytes
64 bytes from 2620:fe::fe: icmp_seq=1 ttl=58 time=54.3 ms
64 bytes from 2620:fe::fe: icmp_seq=2 ttl=58 time=41.2 ms

--- 2620:fe::fe ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 41.208/47.735/54.262/6.527 ms
```

7.3. Adressage automatique dans le réseau d'hébergement

Pour que les hôtes du réseau de conteneurs obtiennent automatiquement une configuration IPv4 et IPv6, il faut ajouter le service `dnsmasq` sur le routeur ***Spoke***. Il fournit les services DHCPv4 et SLAAC en un seul et unique fichier de configuration.

On débute par l'installation du paquet.

```
sudo apt -y install dnsmasq
```

Q77. Comment remplacer le fichier de configuration fourni lors de l'installation du paquet par notre propre fichier de configuration ?

Consulter le contenu du fichier `/etc/dnsmasq.conf` et extraire les options de configuration utiles au contexte de ces manipulations.

Voici la commande de copie du fichier issu de l'installation.

```
sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.dist
```

Voici un exemple de configuration adaptée à la maquette.

```
cat << EOF | sudo tee /etc/dnsmasq.conf
# Specify Container VLAN interface
interface=vlan40

# Enable DHCPv4 on Container VLAN
dhcp-range=203.0.113.20,203.0.113.200,3h

# Enable IPv6 router advertisements
enable-ra

# Enable SLAAC
dhcp-range=::,constructor:vlan40,ra-names,slaac

# Optional: Specify DNS servers
dhcp-option=option:dns-server,172.16.0.2,9.9.9.9
dhcp-option=option6:dns-server,[2001:678:3fc:3::2],[2620:fe::fe]

# Avoid DNS listen port conflict between dnsmasq and systemd-resolved
port=0
EOF
```

Avertissement

Il faut impérativement changer le numéro de VLAN ainsi que les adresses IPv4 de l'exemple ci-dessus par les informations données dans le plan d'adressage des travaux pratiques.

De plus, une fois le fichier créé, il ne faut pas oublier de redémarrer le service et de contrôler l'état de son fonctionnement.

```
sudo systemctl restart dnsmasq
systemctl status dnsmasq
```

8. Conteneurs système Incus

Maintenant que la configuration réseau de la topologie étudiée est complète, on peut passer à la gestion des conteneurs de service du site distant.

8.1. Installation du gestionnaire de conteneurs Incus

Sur le routeur ***Spoke***, la gestion des conteneurs est confiée à Incus. Dans le contexte de ces manipulations, nous utilisons le mode ***bridge*** pour le raccordement réseau des conteneurs. Que l'on lance 3 conteneurs ou 300, ceux-ci seront raccordés de façon transparente au commutateur virtuel `asw-host` et bénéficieront d'un adressage automatique.

Q78. Comment installer le gestionnaire de conteneurs Incus ?

Lancer une recherche dans la liste des paquets Debian.

Le paquet s'appelle tout simplement incus.

```
apt search ^incus
```

```
sudo apt -y install incus
```

Q79. Comment faire pour que l'utilisateur normal `etu` devienne administrateur et gestionnaire des conteneurs ?

Rechercher le nom du groupe système correspondant à l'utilisation des outils Incus.

Il faut que l'utilisateur normal appartienne au groupes systèmes `incus` et `incus-admin` pour qu'il ait tous les droits sur la gestion des conteneurs.

```
grep incus /etc/group
incus:x:990:
incus-admin:x:989:
```

```
sudo adduser etu incus
sudo adduser etu incus-admin
```

> **Avertissement**
>
> Attention ! Il faut se déconnecter/reconnecter pour bénéficier de la nouvelle attribution de groupe. On peut utiliser les commandes groups ou id pour vérifier le résultat.

```
groups
etu adm sudo users incus-admin incus
```

8.2. Configuration et lancement des conteneurs

Q80. Quelle est l'instruction de configuration initiale du gestionnaire Incus ?

Utiliser l'aide de la commande incus.

C'est l'instruction `incus admin init` qui nous intéresse.

Voici une copie d'écran de son exécution.

```
incus admin init
```

```
Would you like to use clustering? (yes/no) [default=no]:
Do you want to configure a new storage pool? (yes/no) [default=yes]:
Name of the new storage pool [default=default]:
Where should this storage pool store its data? [default=/var/lib/incus/storage-pools/default]:
Would you like to create a new local network bridge? (yes/no) [default=yes]: no
Would you like to use an existing bridge or host interface? (yes/no) [default=no]: yes
Name of the existing bridge or host interface: asw-host
Would you like the server to be available over the network? (yes/no) [default=no]:
Would you like stale cached images to be updated automatically? (yes/no) [default=yes]:
Would you like a YAML "init" preseed to be printed? (yes/no) [default=no]:
```

Q81. Quelle est l'instruction qui permet d'afficher le profil par défaut des conteneurs ?

Rechercher dans les options de la commande incus profile.

Voici un exemple d'exécution.

```
incus profile show default
```

```
config: {}
description: Default Incus profile
devices:
  eth0:
    name: eth0
    nictype: macvlan
    parent: asw-host
    type: nic
  root:
    path: /
    pool: default
    type: disk
name: default
used_by: []
project: default
```

L'affichage de ce profil par défaut permet d'identifier les clés à modifier ou à ajouter.

Mode de raccordementnictype

La valeur attribuée par défaut est `macvlan`. Elle ne correspond pas à notre besoin puisque les services hébergés sur le site distant doivent pouvoir communiquer entre eux. On remplace cette valeur par `bridged`.

```
incus profile device set default eth0 nictype bridged
```

```
incus profile device get default eth0 nictype
bridged
```

Attribution de VLANvlan

La configuration du profil par défaut n'attribue pas l'appartenance à un VLAN particulier. Dans le contexte de cette maquette, le réseau des conteneurs correspond au VLAN `40`. On ajoute la clé `vlan` au profil par défaut.

```
incus profile device set default eth0 vlan 40
```

```
incus profile device get default eth0 vlan
40
```

Q82. Quelle est l'instruction de création et de lancement de nouveaux conteneurs ?

Rechercher dans les options de la commande incus.

Tester son exécution avec un conteneur de type `debian/trixie`.

Voici un exemple d'exécution pour 3 nouveaux conteneurs.

```
for i in {0..2}; do incus launch images:debian/trixie c$i; done
```

```
Launching c0
Launching c1
Launching c2
```

```
incus ls
```

```
+------+---------+----------------------+-----------------------------------------------+-----------+-----------+
| NAME |  STATE  |         IPV4         |                     IPV6                      |   TYPE    | SNAPSHOTS |
+------+---------+----------------------+-----------------------------------------------+-----------+-----------+
| c0   | RUNNING | 203.0.113.144 (eth0) | fda0:7a62:28:0:216:3eff:fe67:b848 (eth0) | CONTAINER | 0         |
+------+---------+----------------------+-----------------------------------------------+-----------+-----------+
| c1   | RUNNING | 203.0.113.76 (eth0)  | fda0:7a62:28:0:216:3eff:fe0a:c3ba (eth0) | CONTAINER | 0         |
+------+---------+----------------------+-----------------------------------------------+-----------+-----------+
| c2   | RUNNING | 203.0.113.184 (eth0) | fda0:7a62:28:0:216:3eff:febb:bfb9 (eth0) | CONTAINER | 0         |
+------+---------+----------------------+-----------------------------------------------+-----------+-----------+
```

La copie d'écran ci-dessus montre que l'adressage automatique des conteneurs a fonctionné.

Q83. Comment tester les communications réseau depuis chaque conteneur ?

Rechercher dans les options de la commande incus celle qui permet de lancer un traitement dans le conteneur.

C'est la commande incus exec qui correspond à notre besoin. Voici une exemple de boucle qui permet de lancer les tests ICMP IPv4 et IPv6 dans les 3 conteneurs actifs.

```
for i in {0..2}
do
    echo ">>>>>>>>>>>>>>>> c$i"
    incus exec c$i -- ping -qc2 9.9.9.9
    incus exec c$i -- ping -qc2 2620:fe::fe
done
```

```
>>>>>>>>>>>>>>>> c0
PING 9.9.9.9 (9.9.9.9) 56(84) bytes of data.

--- 9.9.9.9 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 29.420/29.666/29.913/0.246 ms
PING 2620:fe::fe (2620:fe::fe) 56 data bytes

--- 2620:fe::fe ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 41.829/47.305/52.782/5.476 ms
>>>>>>>>>>>>>>>> c1
PING 9.9.9.9 (9.9.9.9) 56(84) bytes of data.

--- 9.9.9.9 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 29.477/29.490/29.504/0.013 ms
PING 2620:fe::fe (2620:fe::fe) 56 data bytes

--- 2620:fe::fe ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 41.401/41.546/41.692/0.145 ms
>>>>>>>>>>>>>>>> c2
PING 9.9.9.9 (9.9.9.9) 56(84) bytes of data.

--- 9.9.9.9 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 28.820/29.427/30.034/0.607 ms
PING 2620:fe::fe (2620:fe::fe) 56 data bytes

--- 2620:fe::fe ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 40.708/41.037/41.367/0.329 ms
```

Q84. Comment exécuter des jeux d'instructions dans les conteneurs depuis le serveur d'hébergement ?

On entre ici dans le domaine de l'automatisation à l'aide de scripts ***Bash***. Même si l'ambition reste très modeste, on peut développer un script qui utilise la liste des conteneurs actifs pour lancer une suite de traitements dans ces mêmes conteneurs.

Comme les conteneurs ***Incus*** appartiennent à la famille des conteneurs système, ils disposent d'une arborescence complète et d'une gestion de paquets. Allons y pour une mise à jour des paquets de chaque conteneur actif.

Voici un exemple de code qui stocke les commandes à lancer dans un tableau constitué des arguments de la ligne de commande. Ces commandes sont exécutées sur chacun des conteneurs actifs.

```
#!/bin/bash

cmds=("$@")

clist=$(incus list status=running -c n -f compact | grep -v NAME | tr '\n' ' ' | tr -s ' ')

for c in $clist; do
  echo ">>>>>>>>>>>>>>>> $c"
  for cmd in "${cmds[@]}"; do
    eval "incus exec $c -- $cmd"
  done
done
```

Voici un exemple d'exécution du script `run-commands-in-containers.sh` qui contient les instructions ci-dessus.

```
bash run-commands-in-containers.sh "apt update" "apt -y full-upgrade" "apt clean" "apt -y autopurge"
```

```
>>>>>>>>>>>>>>>> c0
Get:1 http://deb.debian.org/debian trixie InRelease [169 kB]
Get:2 http://deb.debian.org/debian trixie-updates InRelease [49.6 kB]
Get:3 http://deb.debian.org/debian-security trixie-security InRelease [43.5 kB]
Get:4 http://deb.debian.org/debian trixie/main amd64 Packages.diff/Index [27.9 kB]
Get:5 http://deb.debian.org/debian trixie/main amd64 Packages 2024-09-22-0804.28.pdiff [5327 B]
Get:5 http://deb.debian.org/debian trixie/main amd64 Packages 2024-09-22-0804.28.pdiff [5327 B]
Fetched 296 kB in 1s (216 kB/s)
All packages are up to date.
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
>>>>>>>>>>>>>>>> c1
Get:1 http://deb.debian.org/debian trixie InRelease [169 kB]
Get:2 http://deb.debian.org/debian trixie-updates InRelease [49.6 kB]
Get:3 http://deb.debian.org/debian-security trixie-security InRelease [43.5 kB]
Get:4 http://deb.debian.org/debian trixie/main amd64 Packages.diff/Index [27.9 kB]
Get:5 http://deb.debian.org/debian trixie/main amd64 Packages 2024-09-22-0804.28.pdiff [5327 B]
Get:5 http://deb.debian.org/debian trixie/main amd64 Packages 2024-09-22-0804.28.pdiff [5327 B]
Fetched 296 kB in 1s (279 kB/s)
All packages are up to date.
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
>>>>>>>>>>>>>>>> c2
Get:1 http://deb.debian.org/debian trixie InRelease [169 kB]
Get:2 http://deb.debian.org/debian trixie-updates InRelease [49.6 kB]
Get:3 http://deb.debian.org/debian-security trixie-security InRelease [43.5 kB]
Get:4 http://deb.debian.org/debian trixie/main amd64 Packages.diff/Index [27.9 kB]
Get:5 http://deb.debian.org/debian trixie/main amd64 Packages 2024-09-22-0804.28.pdiff [5327 B]
Get:5 http://deb.debian.org/debian trixie/main amd64 Packages 2024-09-22-0804.28.pdiff [5327 B]
Fetched 296 kB in 1s (277 kB/s)
All packages are up to date.
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
```

8.3. Adressage statique des conteneurs

Avec l'adressage automatique des conteneurs système, on sait les administrer depuis le routeur ***Spoke***. Dans un scénario plus réaliste, il faut être capable d'administrer ces mêmes conteneurs depuis le routeur ***Hub*** ou mieux encore depuis l'infrastructure de déploiement CI/CD de l'entreprise. Ce type d'usage suppose que le conteneurs du site distant utilisent des adresses IPv4 et IPv6 statiques.

Q85. Comment passer d'un adressage automatique à un adressage statique pour chaque conteneur ?

Comme la gestion de la configuration des interfaces est assurée par `systemd-networkd`, il faut s'intéresser à la syntaxe du fichier `/etc/systemd/network/eth0.network` de chaque conteneur.

Cette question est un prétexte pour utiliser le transfert de fichier depuis le serveur d'hébergement vers les conteneurs.

Voici une liste des actions à réaliser sur tous les conteneurs actifs.

1. Installer le paquet netplan.io
2. Générer le fichier de déclaration YAML des paramètres de configuration réseau des interfaces `eth0`
3. Transférer le fichier de déclaration YAML dans le dossier `/etc/netplan/`
4. Effacer le fichier `/etc/systemd/network/eth0.network`
5. Appliquer la nouvelle configuration réseau

Pour connaître les paramètres de configuration réseau d'une interface de conteneur, on peut extraire le fichier `/etc/systemd/network/eth0.network` et consulter son contenu.

```
incus file pull c0/etc/systemd/network/eth0.network .
cat eth0.network
```

```
[Match]
Name=eth0

[Network]
DHCP=true

[DHCPv4]
UseDomains=true

[DHCP]
ClientIdentifier=mac
```

On vérifie ainsi que la configuration réseau issue de la source de tirage des conteneurs implique un adressage automatique au moins en IPv4. On propose donc de remplacer cet adressage automatique par un adressage statique.

Voici une proposition de script qui traite chacun des points définis dans la question.

```
#!/bin/bash

# Préparation -> générer la liste des conteneurs actifs
clist=$(incus list status=running -c n -f compact | grep -v NAME | tr '\n' ' ' | tr -s ' ')

# Étape 1 -> installer le paquet netplan.io
. run-commands-in-containers.sh "apt -y install netplan.io"

addr_idx=0
for c in $clist; do
  echo ">>>>>>>>>>>>>>>> $c"

# Étape 2 -> générer le fichier de configuration réseau YAML
$(cat << EOF > eth0.yaml
network:
  version: 2
  renderer: networkd
  ethernets:
    eth0:
      dhcp4: false
      dhcp6: false
      accept-ra: true
      addresses:
        - 203.0.113.$((addr_idx + 10))/24
        - fda0:7a62:28::$(printf "%x" $((addr_idx + 10)))/64
      routes:
        - to: default
          via: 203.0.113.1
        - to: "::/0"
          via: fe80:28::1
          on-link: true
      nameservers:
        addresses:
          - 172.16.0.2
          - 2001:678:3fc:3::2
EOF
)

# Étape 3 -> transférer le fichier de déclaration YAML
  incus file push eth0.yaml $c/etc/netplan/eth0.yaml

# Étape 4 -> effacer le fichier /etc/systemd/network/eth0.network
  incus exec $c -- rm /etc/systemd/network/eth0.network

# Étape 5 -> appliquer la nouvelle configuration
  incus exec $c -- netplan apply

  ((addr_idx++))
done

exit 0
```

Si le code du script ci-dessus est placé dans un fichier appelé `set-static-addressing.sh`, on peut l'exécuter directement et relever les résultats.

```
bash set-static-addressing.sh
```

```
incus ls
```

```
+------+---------+---------------------+-----------------------------------------------+-----------+-----------+
| NAME |  STATE  |        IPV4         |                     IPV6                      |   TYPE    | SNAPSHOTS |
+------+---------+---------------------+-----------------------------------------------+-----------+-----------+
| c0   | RUNNING | 203.0.113.10 (eth0) | fda0:7a62:28::a (eth0)                        | CONTAINER | 0         |
|      |         |                     | fda0:7a62:28:0:216:3eff:fe67:b848 (eth0)      |           |           |
+------+---------+---------------------+-----------------------------------------------+-----------+-----------+
| c1   | RUNNING | 203.0.113.11 (eth0) | fda0:7a62:28::b (eth0)                        | CONTAINER | 0         |
|      |         |                     | fda0:7a62:28:0:216:3eff:fe0a:c3ba (eth0)      |           |           |
+------+---------+---------------------+-----------------------------------------------+-----------+-----------+
| c2   | RUNNING | 203.0.113.12 (eth0) | fda0:7a62:28::c (eth0)                        | CONTAINER | 0         |
|      |         |                     | fda0:7a62:28:0:216:3eff:febb:bfb9 (eth0)      |           |           |
+------+---------+---------------------+-----------------------------------------------+-----------+-----------+
```

Q86. Comment vérifier la connectivité réseau depuis les conteneurs ?

La question précédente montre que la configuration réseau des conteneurs est complète. On doit donc lancer des tests IPv4 et IPv6.

Voici deux exemples de tests ICMP.

```
for i in {0..2}
do
 echo ">>>>>>>>>>>>>>>> c$i"
 incus exec c$i -- ping -c2 2620:fe::fe
done
```

```
for i in {0..2}
do
 echo ">>>>>>>>>>>>>>>> c$i"
 incus exec c$i -- ping -c2 9.9.9.9
done
```

Q87. Comment ouvrir un accès SSH avec un compte utilisateur normal dans les conteneurs ?

Reprendre le script d'exécution des commandes à l'intérieur des conteneurs pour installer les paquets nécessaires, créer le compte utilisateur et autoriser la connexion SSH par mot de passe.

Voici un exemple de script qui assure les traitements demandés.

```
#!/bin/bash

# Function to check if a variable is a non-empty string
is_non_empty_string() {
    if [[ -n "$1" && "$1" == *[!\ ]* ]]; then
        return 0 # True
    else
        return 1 # False
    fi
}

# Check if both arguments are provided
if [ $# -ne 2 ]; then
    echo "Error: This script requires exactly two arguments."
    echo "Usage: $0 <username> <password>"
    exit 1
fi

# Check if $1 is a non-empty string
if ! is_non_empty_string "$1"; then
    echo "Error: First argument is not a valid non-empty string."
    exit 1
fi

# Check if $2 is a non-empty string
if ! is_non_empty_string "$2"; then
    echo "Error: Second argument is not a valid non-empty string."
    exit 1
fi

USER=$1
shift
PASSWD=$1

. run-commands-in-containers.sh \
        "apt -y install ssh sudo" \
        "adduser --gecos \"\" --disabled-password ${USER}" \
        "adduser ${USER} sudo" \
        "adduser ${USER} adm" \
        "sh -c \"echo '${USER}:${PASSWD}' | chpasswd\"" \
        "sed -i 's/#Port 22/Port 22\nPort 2222/' /etc/ssh/sshd_config" \
        "sed -i 's/PasswordAuthentication no/PasswordAuthentication yes/' /etc/ssh/sshd_config" \
        "dpkg-reconfigure openssh-server"

exit 0
```

Voici un exemple de lancement du script avec un nom d'utilisateur et son mot de passe comme arguments de la ligne de commande.

```
bash set-user-and-ssh.sh "etu" "xxxxxxx"
```

Le résultat est évalué par une connexion SSH depuis le routeur ***Hub***.

```
etu@hub:~$ ssh etu@fda0:7a62:28::b
The authenticity of host 'fda0:7a62:28::b (fda0:7a62:28::b)' can't be established.
ED25519 key fingerprint is SHA256:6qkAjHutYP6hbpa4iNh94y1Bk4wRqRD62ahOC+3WC9g.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'fda0:7a62:28::b' (ED25519) to the list of known hosts.
etu@fda0:7a62:28::b's password:
Linux c1 6.10.9-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.10.9-1 (2024-09-08) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Sep 22 14:24:07 2024 from 203.0.113.1
etu@c1:~$
```

9. Traces d'une ouverture de session PPPoE

Pour obtenir la trace des transactions entre les deux routeurs de la maquette, on fait appel à la commande journalctl et on consulte les journaux de l'unité systemd ajoutée sur chaque routeur.

9.1. Journaux du routeur Spoke

Côté routeur ***Spoke***, l'unité systemd s'appelle `ppp.service`. Voici la commande de consultation de l'ouverture de session PPP la plus récente.

```
journalctl -n 100 -f -u ppp.service
```

```
spoke systemd[1]: Starting ppp.service - PPPoE Client Connection...
spoke pppd[629]: Plugin rp-pppoe.so loaded.
spoke pon[629]: Plugin rp-pppoe.so loaded.
spoke pppd[639]: pppd 2.5.0 started by root, uid 0
spoke systemd[1]: Started ppp.service - PPPoE Client Connection.
spoke pppd[639]: Send PPPOE Discovery V1T1 PADI session 0x0 length 12❶
spoke pppd[639]:  dst ff:ff:ff:ff:ff:ff  src b8:ad:ca:fe:00:06
spoke pppd[639]:  [service-name] [host-uniq 7f 02 00 00]
spoke pppd[639]: Recv PPPOE Discovery V1T1 PADO session 0x0 length 44
spoke pppd[639]:  dst b8:ad:ca:fe:00:06  src b8:ad:ca:fe:00:05
spoke pppd[639]:  [AC-name BRAS] [service-name] [AC-cookie eb 9f 92 61 e1 ee 01 a1 5d 8f 11 61 8a fb c8 4b fc 01 00 00] [host-unic
spoke pppd[639]: Access-Concentrator: BRAS
spoke pppd[639]: Cookie: eb 9f 92 61 e1 ee 01 a1 5d 8f 11 61 8a fb c8 4b fc 01 00 00
spoke pppd[639]: AC-Ethernet-Address: b8:ad:ca:fe:00:05
spoke pppd[639]: ----------------------------------------------------
spoke pppd[639]: Send PPPOE Discovery V1T1 PADR session 0x0 length 36
spoke pppd[639]:  dst b8:ad:ca:fe:00:05  src b8:ad:ca:fe:00:06
spoke pppd[639]:  [service-name] [host-uniq 7f 02 00 00] [AC-cookie eb 9f 92 61 e1 ee 01 a1 5d 8f 11 61 8a fb c8 4b fc 01 00 00]
spoke pppd[639]: Recv PPPOE Discovery V1T1 PADS session 0x1 length 12
spoke pppd[639]:  dst b8:ad:ca:fe:00:06  src b8:ad:ca:fe:00:05
spoke pppd[639]:  [service-name] [host-uniq 7f 02 00 00]
spoke pppd[639]: PPP session is 1
spoke pppd[639]: Connected to B8:AD:CA:FE:00:05 via interface enp0s1.441
spoke pppd[639]: using channel 1
spoke pppd[639]: Using interface ppp0
spoke pppd[639]: Connect: ppp0 <--> enp0s1.441❷
spoke pppd[639]: sent [LCP ConfReq id=0x1 <mru 1492> <magic 0xbf826095>]
spoke pppd[639]: rcvd [LCP ConfReq id=0x1 <mru 1492> <auth eap> <magic 0x471e9bda>]
spoke pppd[639]: sent [LCP ConfAck id=0x1 <mru 1492> <auth eap> <magic 0x471e9bda>]
spoke pppd[639]: rcvd [LCP ConfAck id=0x1 <mru 1492> <magic 0xbf826095>]
spoke pppd[639]: sent [LCP EchoReq id=0x0 magic=0xbf826095]
spoke pppd[639]: rcvd [LCP EchoReq id=0x0 magic=0x471e9bda]
spoke pppd[639]: sent [LCP EchoRep id=0x0 magic=0xbf826095]
spoke pppd[639]: rcvd [EAP Request id=0xa1 Identity <Message "Name">]
spoke pppd[639]: EAP: Identity prompt "Name"
spoke pppd[639]: sent [EAP Response id=0xa1 Identity <Name "spoke_site0">]
spoke pppd[639]: rcvd [LCP EchoRep id=0x0 magic=0x471e9bda]
spoke pppd[639]: rcvd [EAP Request id=0xa2 MD5-Challenge <Value 81 3b a7 d1 eb 86 42 15 2c b9 1b 07 83 98 e2 dd b7 c6 57 b4 b5 0f>
spoke pppd[639]: sent [EAP Response id=0xa2 MD5-Challenge <Value 16 e7 3d fa d2 4b 6a 73 41 5f 86 c8 84 97 ed f0> <Name "spoke_sit
spoke pppd[639]: rcvd [EAP Success id=0xa3]
spoke pppd[639]: EAP authentication succeeded
spoke pppd[639]: peer from calling number B8:AD:CA:FE:00:05 authorized
spoke pppd[639]: sent [IPCP ConfReq id=0x1 <addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]❸
spoke pppd[639]: sent [IPV6CP ConfReq id=0x1 <addr fe80::9869:8831:104a:570a>]
spoke pppd[639]: rcvd [CCP ConfReq id=0x1 <bsd v1 15>]
spoke pppd[639]: sent [CCP ConfReq id=0x1]
spoke pppd[639]: sent [CCP ConfRej id=0x1 <bsd v1 15>]
spoke pppd[639]: rcvd [IPCP ConfReq id=0x1 <addr 10.4.41.1>]
spoke pppd[639]: sent [IPCP ConfAck id=0x1 <addr 10.4.41.1>]
spoke pppd[639]: rcvd [IPV6CP ConfReq id=0x1 <addr fe80::61d3:68e0:1e34:e123>]
spoke pppd[639]: sent [IPV6CP ConfAck id=0x1 <addr fe80::61d3:68e0:1e34:e123>]
spoke pppd[639]: rcvd [IPCP ConfNak id=0x1 <addr 10.4.41.2> <ms-dns1 172.16.0.2> <ms-dns2 172.16.0.2>]
spoke pppd[639]: sent [IPCP ConfReq id=0x2 <addr 10.4.41.2> <ms-dns1 172.16.0.2> <ms-dns2 172.16.0.2>]
spoke pppd[639]: rcvd [IPV6CP ConfAck id=0x1 <addr fe80::9869:8831:104a:570a>]
spoke pppd[639]: local  LL address fe80::9869:8831:104a:570a
spoke pppd[639]: remote LL address fe80::61d3:68e0:1e34:e123
spoke pppd[639]: Script /etc/ppp/ipv6-up started (pid 653)
spoke pppd[639]: rcvd [CCP ConfAck id=0x1]
spoke pppd[639]: rcvd [CCP ConfReq id=0x2]
spoke pppd[639]: sent [CCP ConfAck id=0x2]
spoke pppd[639]: rcvd [IPCP ConfAck id=0x2 <addr 10.4.41.2> <ms-dns1 172.16.0.2> <ms-dns2 172.16.0.2>]
spoke pppd[639]: Script /etc/ppp/ip-pre-up started (pid 658)
spoke pppd[639]: Script /etc/ppp/ip-pre-up finished (pid 658), status = 0x0
spoke pppd[639]: local  IP address 10.4.41.2
spoke pppd[639]: remote IP address 10.4.41.1
spoke pppd[639]: primary   DNS address 172.16.0.2
spoke pppd[639]: secondary DNS address 172.16.0.2
spoke pppd[639]: Script /etc/ppp/ip-up started (pid 662)
spoke pppd[639]: Script /etc/ppp/ipv6-up finished (pid 653), status = 0x0
spoke pppd[639]: Script /etc/ppp/ip-up finished (pid 662), status = 0x0
```

❶ Sur un réseau de diffusion il est nécessaire d'identifier les deux hôtes qui doivent établir une session PPP. Cette toute première phase d'identification utilise des trames spécifiques avec les messages décrits dans la Section 2, « Interface Ethernet & protocole PPP ».

❷ La sous-couche *Link Control Protocol* (LCP) assure la configuration automatique des interfaces à chaque extrémité. Les paramètres négociés entre les deux hôtes en communication sont multiples : l'adaptation de la taille de datagramme, les caractères d'échappement, les numéros magiques et la sélection des options d'authentification.

❸ La sous-couche *Network Control Protocol* (NCP) assure l'encapsulation de multiples protocoles de la couche réseau. Dans l'exemple donné, c'est le protocole IPv4 qui est utilisé ; d'où l'acronyme IPCP.

9.2. Journaux du routeur Hub

Côté routeur ***hub***, l'unité systemd s'appelle `pppoe-server.service`. Voici la commande de consultation de l'ouverture de session PPP la plus récente.

```
journalctl -n 100 -f -u pppoe-server.service
```

```
hub pppoe-server[610]: Session 1 created for client b8:ad:ca:fe:00:06 (10.4.41.2) on enp0s1.441 using Service-Name ''
hub pppd[610]: pppd 2.5.0 started by root, uid 0
hub pppd[610]: using channel 2
hub pppd[610]: Using interface ppp0
hub pppd[610]: Connect: ppp0 <--> /dev/pts/0
hub pppd[610]: rcvd [LCP ConfReq id=0x1 <mru 1492> <magic 0xbf826095>]
hub pppd[610]: sent [LCP ConfReq id=0x1 <mru 1492> <auth eap> <magic 0x471e9bda>]
hub pppd[610]: sent [LCP ConfAck id=0x1 <mru 1492> <magic 0xbf826095>]
hub pppd[610]: rcvd [LCP ConfAck id=0x1 <mru 1492> <auth eap> <magic 0x471e9bda>]
hub pppd[610]: sent [LCP EchoReq id=0x0 magic=0x471e9bda]
hub pppd[610]: sent [EAP Request id=0xa1 Identity <Message "Name">]
hub pppd[610]: rcvd [LCP EchoReq id=0x0 magic=0xbf826095]
hub pppd[610]: sent [LCP EchoRep id=0x0 magic=0x471e9bda]
hub pppd[610]: rcvd [LCP EchoRep id=0x0 magic=0xbf826095]
hub pppd[610]: rcvd [EAP Response id=0xa1 Identity <Name "spoke_site0">]
hub pppd[610]: EAP: unauthenticated peer name "spoke_site0"
hub pppd[610]: EAP id=0xa1 'Identify' -> 'MD5Chall'
hub pppd[610]: sent [EAP Request id=0xa2 MD5-Challenge <Value 81 3b a7 d1 eb 86 42 15 2c b9 1b 07 83 98 e2 dd b7 c6 57 b4 b5 0f> <
hub pppd[610]: rcvd [EAP Response id=0xa2 MD5-Challenge <Value 16 e7 3d fa d2 4b 6a 73 41 5f 86 c8 84 97 ed f0> <Name "spoke_site0
hub pppd[610]: sent [EAP Success id=0xa3]
hub pppd[610]: peer from calling number b8:ad:ca:fe:00:06 authorized
hub pppd[610]: sent [CCP ConfReq id=0x1 <bsd v1 15>]
hub pppd[610]: sent [IPCP ConfReq id=0x1 <addr 10.4.41.1>]
hub pppd[610]: sent [IPV6CP ConfReq id=0x1 <addr fe80::61d3:68e0:1e34:e123>]
hub pppd[610]: EAP id=0xa3 'MD5Chall' -> 'Open'
hub pppd[610]: rcvd [IPCP ConfReq id=0x1 <addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
hub pppd[610]: sent [IPCP ConfNak id=0x1 <addr 10.4.41.2> <ms-dns1 172.16.0.2> <ms-dns2 172.16.0.2>]
hub pppd[610]: rcvd [IPV6CP ConfReq id=0x1 <addr fe80::9869:8831:104a:570a>]
hub pppd[610]: sent [IPV6CP ConfAck id=0x1 <addr fe80::9869:8831:104a:570a>]
hub pppd[610]: rcvd [CCP ConfReq id=0x1]
hub pppd[610]: sent [CCP ConfAck id=0x1]
hub pppd[610]: rcvd [CCP ConfRej id=0x1 <bsd v1 15>]
hub pppd[610]: sent [CCP ConfReq id=0x2]
hub pppd[610]: rcvd [IPCP ConfAck id=0x1 <addr 10.4.41.1>]
hub pppd[610]: rcvd [IPV6CP ConfAck id=0x1 <addr fe80::61d3:68e0:1e34:e123>]
hub pppd[610]: local  LL address fe80::61d3:68e0:1e34:e123
hub pppd[610]: remote LL address fe80::9869:8831:104a:570a
hub pppd[610]: Script /etc/ppp/ipv6-up started (pid 616)
hub pppd[610]: rcvd [IPCP ConfReq id=0x2 <addr 10.4.41.2> <ms-dns1 172.16.0.2> <ms-dns2 172.16.0.2>]
hub pppd[610]: sent [IPCP ConfAck id=0x2 <addr 10.4.41.2> <ms-dns1 172.16.0.2> <ms-dns2 172.16.0.2>]
hub pppd[610]: Script /etc/ppp/ip-pre-up started (pid 619)
hub pppd[610]: Script /etc/ppp/ip-pre-up finished (pid 619), status = 0x0
hub pppd[610]: local  IP address 10.4.41.1
hub pppd[610]: remote IP address 10.4.41.2
hub pppd[610]: Script /etc/ppp/ip-up started (pid 623)
hub pppd[610]: Script /etc/ppp/ipv6-up finished (pid 616), status = 0x0
hub pppd[610]: rcvd [CCP ConfAck id=0x2]
hub pppd[610]: Script /etc/ppp/ip-up finished (pid 623), status = 0x0
```

10. Pour conclure...

Ce support de travaux pratiques a permis d'explorer en détail la mise en œuvre d'une topologie réseau complexe combinant routage inter-VLAN, protocole PPPoE et conteneurisation. La démarche proposée permet de configurer pas à pas les différents éléments, des routeurs virtuels jusqu'aux conteneurs système.

L'automatisation des tâches est introduite avec l'utilisation de scripts pour la configuration réseau et le déploiement des conteneurs. Cela permet de découvrir les pratiques actuelles en matière d'administration système et réseau dans un contexte d'infrastructure as code.

Enfin, toutes ces manipulations offrent une expérience pratique des protocoles utilisés sur les réseau étendus comme PPPoE, tout en illustrant l'adressage IPv6 et la virtualisation réseau.

Topologie Hub & Spoke avec le protocole PPPoE

https://www.inetdoc.net

Résumé

Ce support de travaux pratiques est une illustration d'une topologie réseau classique baptisée *Hub & Spoke*. Le *Hub* est un routeur qui concentre tous les flux des routeurs d'extrémités appelés *Spoke*. Les liaisons entre le *Hub* et les routeurs *Spoke* sont point à point et utilisent le protocole PPP. Avec la généralisation de la fibre optique dans les réseaux étendus (WAN), on doit encapsuler les trames PPP dans un VLAN Ethernet à l'aide de la technologie PPPoE.

Les manipulations proposées reprennent en grande partie celles du support *Routage inter-VLAN et protocole PPPoE* en les adaptant à la topologie en triangle.

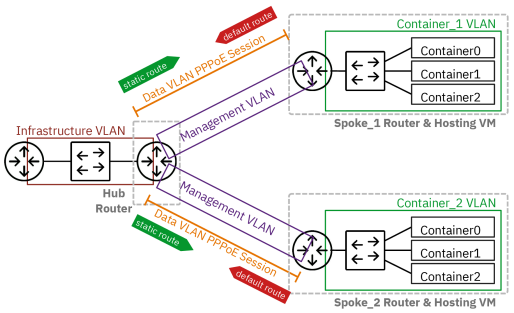


Table des matières

1. Objectifs

Après avoir réalisé les manipulations présentées dans ce document, les étudiants seront capables de :

1. Configurer une topologie réseau Hub & Spoke utilisant le protocole PPPoE entre le routeur concentrateur (rôle *Hub*) et les routeurs d'extrémité (rôle *Spoke*).
2. Établir et gérer des sessions PPP entre les routeurs, en configurant l'authentification et l'attribution d'adresses IP.
3. Mettre en place un routage complet IPv4 et IPv6 entre les différents sites de la topologie, y compris la configuration de routes statiques.
4. Déployer et gérer des conteneurs sur les routeurs *Spoke*, en utilisant le gestionnaire Incus.
5. Tester et valider la connectivité réseau à travers la topologie, en utilisant des outils comme ping, tracepath et en configurant des services web basiques.

2. Topologie Hub & Spoke - Protocole PPPoE

Le Protocole Point à Point (PPP) est utilisé pour établir une communication directe entre deux hôtes. Il relie deux routeurs de façon logique au dessus d'une topologie de réseau physique qui peut comprendre divers composants et différentes technologies. Il permet aux deux extrémités en communication de négocier des paramètres de transmission tels que l'authentification et l'attribution d'adresses de couche réseau. Dans ce document, on utilise le protocole PPP au dessus d'un réseau Ethernet qui représente la technologie mise en œuvre par un opérateur Internet.

Comme un Ethernet est par définition un réseau de diffusion, il est nécessaire d'introduire un protocole intermédiaire appelé PPPoE qui sert à identifier les deux hôtes de la liaison logique point à point.

Ce support met en œuvre une “figure“ classique appelée topologie *Hub & Spoke*. Voici une description des rôles des différents routeurs de cette topologie.

Hub

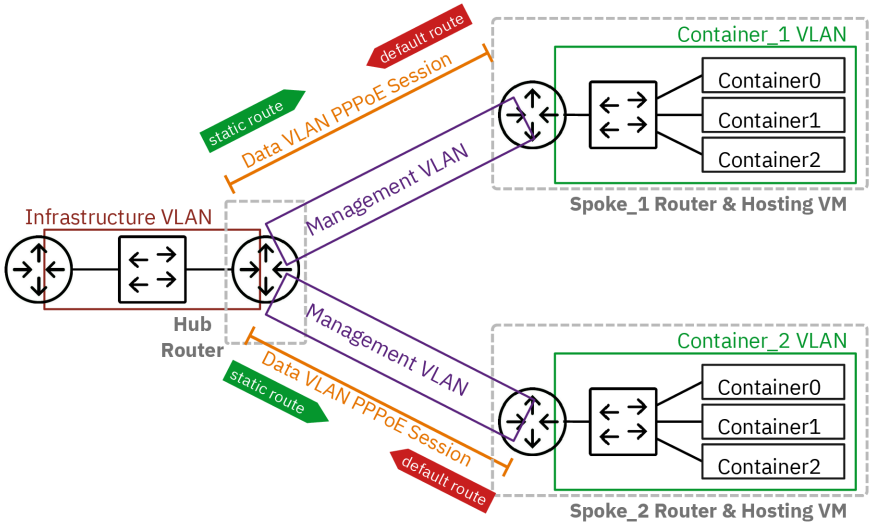
Traduit mot à mot, le rôle *Hub* correspond à un concentrateur. Il concentre tous les flux réseau des routeurs qui ont le rôle *Spoke*. En effet, les échanges entre deux routeurs *Spoke* doivent passer par le routeur *Hub*.

On lui attribue aussi la fonction de ***Broadband Remote Access Server*** ou BRAS. Dans notre contexte, cette fonction se caractérise par le fait que ce routeur détient le plan d'adressage. C'est lui qui a la responsabilité de délivrer les adresses IP lors de l'initiation de la session PPP.

Spoke

Le rôle ***Spoke*** correspond à un réseau d'extrémité au delà duquel on ne trouve aucune interconnexion. Le routeur ***Spoke*** doit s'adresser au routeur Hub dès qu'il veut acheminer un flux réseau. Il s'agit bien d'un routeur d'extrémité qui ne dispose d'aucun chemin alternatif pour joindre l'Internet.

Dans les réseaux domestiques, la «box» correspond bien au rôle ***Spoke*** dans la mesure où elle se voit attribuer des adresses IPv4 et IPv6 publiques par le fournisseur d'accès Internet. Les seules informations qu'elle détient sont les authentifiants du client de l'opérateur.



Topologie entre deux routeurs ***Hub*** et ***Spoke*** avec PPPoE

Comme le montre le graphique ci-dessus, l'opérateur distingue 4 réseaux ou VLANs différents.

Réseau d'infrastructure (VLAN rouge)

Ce raccordement réseau donne l'accès à Internet. Il représente la dorsale de l'opérateur.

Management/Supervision (VLAN violet)

Ce réseau correspond à la gestion des équipements et à la supervision des liens en fibre optique. Il n'utilise que des adresses de lien local IPv6. Il représente le rôle de l'exploitant des chemins de câbles.

Réseau fournisseur d'accès Internet (VLAN orange)

C'est sur ce réseau que la session PPP est établie. Le site distant de l'entreprise cliente (rôle ***Spoke***) s'authentifie auprès du BRAS de l'opérateur et obtient en échange une adresse IPv4 et une adresse IPv6 qui permettent d'accéder au site central (rôle ***Hub***) et à Internet.

Réseau d'hébergement de site distant (VLAN vert)

C'est le réseau des services hébergés sur son propre site par l'entreprise cliente de l'opérateur. Ici, on choisit de déployer plusieurs conteneurs pour illustrer plusieurs hôtes ou serveurs dont le trafic doit transiter uniquement par le site central (rôle ***Hub***) via la session PPP.

Voici le plan d'adressage de la maquette utilisée pour rédiger ce support de travaux pratiques.

Tableau 1. Plan d'adressage de la maquette « Topologie Hub & Spoke et protocole PPPoE »

Rôle	VLAN	Numéro	Type	Destination	Adresse/Authentification
Hub bleu	Rouge	360	Passerelle	-	`192.168.104.129/29` `fe80:168::1/64`
	Violet	440	Lien local	Spoke1	`fe80:1b8::1/64`
	Orange	441	Point à point	Spoke1	`10.44.1.1:10.44.1.2`
	Violet	442	Lien local	Spoke2	`fe80:1ba::1/64`

Rôle	VLAN	Numéro	Type	Destination	Adresse/Authentification
	Orange	443	Point à point	Spoke2	`10.44.3.1:10.44.3.2`
Spoke1 Vert	Violet	440	Lien local	Hub	`fe80:1b8::2/64`
	Orange	441	Authentifiants	Hub	`spoke_site1 / 0r4ng3_1`

	Vert	10	Passerelle	-	`10.0.10.1/24`
					`fda0:7a62:a::1/64`
					`fe80:a::1/64`

Spoke2 Vert	Violet	442	Lien local	Hub	`fe80:1ba::2/64`
	Orange	443	Authentifiants	Hub	`spoke_site2 / 0r4ng3_2`

	Vert	20	Passerelle	-	`10.0.20.1/24`
					`fda0:7a62:14::1/64`
					`fe80:14::1/64`

3. Routeur Hub

Le rôle du routeur ***Hub*** est d'interconnecter un réseau de collecte opérateur (LAN) qui donne accès à l'Internet et plusieurs réseaux étendus (WAN) de raccordement de sites distants. Le routeur ***Hub*** assure aussi la fonction ***Broadband Remote Access Server*** (BRAS). C'est la raison pour laquelle il détient les adresses IPv4 et IPv6 à attribuer aux routeurs d'extrémité appelés ***Spoke***.

Le routeur ***Hub*** doit aussi gérer l'encapsulation des trames PPP sur un réseau de diffusion Ethernet.

Cette section reprend essentiellement la partie ***Routeur Hub*** du support *Routage inter-VLAN et protocole PPPoE*. On doit cependant adapter la configuration des interfaces réseau du routeur ***Hub*** en ajoutant un second jeu d'interfaces pour le deuxième site distant.

Q88. Comment activer le routage dans le sous-système réseau du noyau Linux ?

Utiliser la commande sysctl pour effectuer des recherches et identifier les paramètres utiles. Par exemple :

```
sudo sysctl -a -r ".*forward.*"
```

Il est dorénavant recommandé de créer un fichier de configuration spécifique par fonction. C'est la raison pour laquelle on crée un nouveau fichier `/etc/sysctl.d/10-routing.conf`.

```
cat << EOF | sudo tee /etc/sysctl.d/10-routing.conf
net.ipv4.ip_forward=1
net.ipv6.conf.all.forwarding=1
net.ipv4.conf.all.log_martians=1
EOF
```

Attention ! N'oubliez pas d'appliquer les nouvelles valeurs des paramètres de configuration.

```
sudo sysctl --system
```

Q89. Comment assurer la traduction d'adresses sources pour tous les flux réseaux sortants sur le réseau d'infrastructure (VLAN rouge) ?

Rechercher dans des exemples de configuration `nftables` avec la fonction `MASQUERADE`.

Voici un exemple de création du fichier `/etc/nftables.conf` avec le jeu d'instructions qui assure la traduction d'adresses sources pour IPv4 et IPv6.

```
cat << 'EOF' | sudo tee /etc/nftables.conf
#!/usr/sbin/nft -f

flush ruleset

# Define variables
define RED_VLAN = enp0s1.360

table inet nat {
    chain postrouting {
        type nat hook postrouting priority 100;
        oifname $RED_VLAN counter packets 0 bytes 0 masquerade
    }
}
EOF
```

> **Avertissement**
>
> Il faut impérativement changer le nom d'interface en utilisant le numéro de VLAN attribué dans le plan d'adressage des travaux pratiques.

La création de ce fichier de règles n'est pas suffisante. Il faut appliquer les règles contenues dans le fichier.

```
sudo nft -f /etc/nftables.conf
```

Il faut aussi activer ce service pour assurer le chargement automatique des règles de filtrage au démarrage.

```
sudo systemctl enable --now nftables.service
sudo systemctl status nftables.service
```

Q90. Quel paquet spécifique à la gestion du dialogue PPPoE à installer sur le routeur ***Hub*** ?

Rechercher dans le catalogue des paquets, la référence pppoe.

```
apt search ^pppoe
```

Le résultat de la commande apt show pppoe montre que c'est bien le paquet `pppoe` qui répond au besoin. On peut donc l'installer.

```
sudo apt -y install pppoe
```

Q91. Dans quel fichier sont stockés les paramètres d'identité et d'authentification utilisés par le protocole EAP pour la méthode CHAP ?

Consulter les pages de manuels du démon pppd à la section ***AUTHENTICATION***.

C'est le fichier `/etc/ppp/chap-secrets` qui contient les couples ***login/password*** utilisés lors de l'authentification.

Voici un exemple du contenu de ce fichier.

```
# Secrets for authentication using CHAP
# client     server secret   IP addresses
"spoke_site1" *  "0r4ng3_1"  *
"spoke_site2" *  "0r4ng3_2"  *
```

Q92. Dans quel fichier sont stockés les paramètres passés au démon pppd lors du lancement du serveur PPPoE ?

Consulter les pages de manuels de l'outil pppoe-server.

C'est le fichier `/etc/ppp/pppoe-server-options` qui contient la liste des paramètres utilisés lors du dialogue PPP.

Ce fichier contient tous les paramètres communs aux deux démons `pppd` qui sont lancés via pppoe-server. Voici comment créer le fichier.

```
cat << 'EOF' | sudo tee /etc/ppp/pppoe-server-options
# Gestion de session avec PAM
login
# Authentification EAP
require-eap
# Le Routeur Hub détient déjà une route par défaut
nodefaultroute
# Envoi de l'adresse de résolution DNS avec les adresses IPv4
ms-dns 172.16.0.2
# Ajout du protocole IPv6
+ipv6
# Informations détaillées dans la journalisation
debug
# Options préconisées par la documentation
noaccomp
default-asyncmap
nodeflate
nopcomp
novj
novjccomp
lcp-echo-interval 10
EOF
```

Q93. Comment créer les comptes utilisateurs locaux sur le routeur ***Hub*** sachant qu'ils ne sont autorisés ni à se connecter ni à avoir un répertoire personnel ?

Consulter les options de la commande adduser.

Voici un exemple dans le contexte de la maquette.

```
sudo adduser --gecos 'Spoke 1' --disabled-login --no-create-home spoke_site1
```

```
sudo adduser --gecos 'Spoke 2' --disabled-login --no-create-home spoke_site2
```

Q94. Quel paramètre supplémentaire doit être ajouté à la configuration de la commande pppoe-server pour distinguer les échanges entre les deux routeurs ***Spoke*** ?

Relativement au support *Routage inter-VLAN et protocole PPPoE*, il est essentiel de définir correctement les routes statiques vers les réseaux d'extrémité de chaque routeur ***Spoke***.

Consulter les options de la commande pppoe-server.

L'option `-u` permet de désigner une “unité“ qui sert à nommer l'interface. Par exemple, `-u 0` correspond à l'interface `ppp0`.

Q95. Comment créer deux nouvelles unités systemd responsables du lancement des processus pppoe-server ?

Consulter la page *systemd Services* et rechercher la procédure à suivre pour ajouter un service au lancement du système.

On commence par la création du fichier de service appelé : `/etc/systemd/system/pppoe-server.service` qui contient toutes les directives de lancement du processus pppoe-server avec les paramètres d'adressage du lien point à point.

Voici un exemple de création du fichier d'unité systemd pour le premier service.

```
cat << 'EOF' | sudo tee /etc/systemd/system/pppoe-server1.service
[Unit]
Description=PPPoE Server
After=systemd-networkd.service
Wants=systemd-networkd.service
BindsTo=sys-subsystem-net-devices-enp0s1.441.device
After=sys-subsystem-net-devices-enp0s1.441.device

[Service]
Type=forking
ExecCondition=/bin/sh -c '[ "$(systemctl show --property MainPID --value pppoe-server1.service)" = "0" ]'
ExecStart=/usr/sbin/pppoe-server -I enp0s1.441 -C BRAS -L 10.44.1.1 -R 10.44.1.2 -N 1 -u 0
Restart=on-failure
RestartSec=5

[Install]
WantedBy=multi-user.target
EOF
```

Voici un exemple de création du fichier d'unité systemd pour le second service.

```
cat << 'EOF' | sudo tee /etc/systemd/system/pppoe-server2.service
[Unit]
Description=PPPoE Server
After=systemd-networkd.service
Wants=systemd-networkd.service
BindsTo=sys-subsystem-net-devices-enp0s1.443.device
After=sys-subsystem-net-devices-enp0s1.443.device

[Service]
Type=forking
ExecCondition=/bin/sh -c '[ "$(systemctl show --property MainPID --value pppoe-server2.service)" = "0" ]'
ExecStart=/usr/sbin/pppoe-server -I enp0s1.443 -C BRAS -L 10.44.3.1 -R 10.44.3.2 -N 1 -u 1
Restart=on-failure
RestartSec=5

[Install]
WantedBy=multi-user.target
EOF
```

Q96. Comment activer les deux nouveaux services et contrôler leur état après lancement ?

Consulter la page *systemd Services* et rechercher la procédure à suivre pour activer et lancer un service.

On commence par la relecture de la liste des services disponibles par le gestionnaire systemd.

```
sudo systemctl daemon-reload
```

On active les nouveaux services.

```
for i in {1..2}; do sudo systemctl enable pppoe-server$i.service; done
```

On lance ce nouveau service.

```
for i in {1..2}; do sudo systemctl start pppoe-server$i.service; done
```

On vérifie que l'opération s'est déroulée correctement.

```
for i in {1..2}; do systemctl status pppoe-server$i.service; done
```

En l'état actuel de la configuration, aucune session PPP n'a encore été établie. Il faut maintenant passer à la configuration réseau du routeur ***Spoke*** pour avancer dans l'utilisation du protocole PPP.

4. Routeurs Spoke

Dans le scénario défini dans la Section 2, « Topologie Hub & Spoke - Protocole PPPoE », un routeur de site d'extrémité ou ***Spoke*** ne peut accéder aux autres réseaux que via le routeur ***Hub***. Son interface WAN joue donc le rôle de route par défaut pour le réseau local des hôtes hébergé sur un site distant.

Cette section, comme la précédente, reprend les éléments du support *Routage inter-VLAN et protocole PPPoE* en dédoublant les routeurs d'extrémité.

Les routeurs ***Spoke*** utilisent un démon `pppd` dans le VLAN `Data` (Orange) pour établir une session PPP avec le routeur ***Hub***.

Avant d'aborder les questions ci-dessous, il faut s'assurer que :

- Le nom d'hôte est correctement attribué sur chaque routeur ***Spoke***.
  - `sudo hostnamectl hostname spoke_1`
  - `sudo hostnamectl hostname spoke_2`
- Le routage est configuré et activé.

```
cat << EOF | sudo tee /etc/sysctl.d/10-routing.conf
net.ipv4.ip_forward=1
net.ipv6.conf.all.forwarding=1
net.ipv4.conf.all.log_martians=1
EOF
```

```
sudo sysctl --system
```

- Les paquets `ppp` sont installés sur chaque routeur ***Spoke*** via l'accès réseau temporaire. Il faut donc éditer et appliquer les modifications faites dans le fichier `/etc/netplan/enp0s1.yaml`.

```
sudo apt -y install ppp
```

- Le fichier `/etc/ppp/chap-secrets` contenant les authentifiants pour l'établissement de la session PPP est complété.
  -
```
# Secrets for authentication using CHAP
# client        server  secret          IP addresses
"spoke_site1"   *       "0r4ng3_1"      *
```
  -
```
# Secrets for authentication using CHAP
# client        server  secret          IP addresses
"spoke_site2"   *       "0r4ng3_2"      *
```
- Le fichier `/etc/ppp/peers/pppoe-provider` de définition du profil de session PPP est créé.

> **Avertissement**
>
> Le nom d'utilisateur doit correspondre à l'entrée du fichier `/etc/ppp/chap-secrets` !
>
> Le numéro de VLAN de la sous-interface doit désigner le bon côté du triangle de la topologie !

```
cat << 'EOF' | sudo tee /etc/ppp/peers/pppoe-provider
# Le nom d'utilisateur désigne l'entrée du fichier /etc/ppp/chap-secrets
user spoke_siteX

# Chargement du module PPPoE avec les détails dans la journalisation
plugin rp-pppoe.so rp_pppoe_ac BRAS rp_pppoe_verbose 1

# Interface (VLAN) utilisé pour l'établissement de la session PPP
enp0s1.VVV

# Les adresses sont attribuées par le "serveur" PPPoE
noipdefault
# L'adresse de résolution DNS est aussi fournie par le serveur PPPoE
usepeerdns
# La session PPP devient la route par défaut du routeur Spoke
defaultroute

# Demande de réouverture de session automatique en cas de rupture
persist

# Le routeur Spoke n'exige pas que le routeur Hub s'authentifie
noauth

# Messages d'informations détaillés dans la journalisation
debug

# Utilisation du protocole IPv6
+ipv6

# Options préconisées par la documentation
noaccomp
default-asyncmap
nodeflate
nopcomp
novj
novjccomp
lcp-echo-interval 10
EOF
```

- Les unités `systemd` sont créées et activées sur chaque routeur ***Spoke***.

> Avertissement
>
> Le numéro de VLAN de la sous-interface doit désigner le bon côté du triangle de la topologie !

```
cat << EOF | sudo tee /etc/systemd/system/ppp.service
[Unit]
Description=PPPoE Client Connection
After=network.target
Wants=network.target
BindsTo=sys-subsystem-net-devices-enp0s1.VVV.device
After=sys-subsystem-net-devices-enp0s1.VVV.device

[Service]
Type=forking
ExecStart=/usr/bin/pon pppoe-provider
ExecStop=/usr/bin/poff pppoe-provider
Restart=on-failure
RestartSec=20

[Install]
WantedBy=multi-user.target
EOF
```

On active les nouvelles unités de service avec les instructions suivantes.

```
sudo systemctl daemon-reload
sudo systemctl enable --now ppp.service
```

- Une fois la session PPP établie, n'oubliez pas de désactiver l'accès réseau temporaire et s'assurer que c'est bien cette session qui sert de route par défaut pour accéder à tous les autres réseaux.

  Il faut donc éditer à nouveau et appliquer les modifications faites dans le fichier `/etc/netplan/enp0s1.yaml`.

  Voici un exemple de test lancé sur le second routeur ***Spoke*** de la maquette.

```
ip route get 9.9.9.9
9.9.9.9 dev ppp0 src 10.44.3.2 uid 1000
    cache
```

- On doit éditer le fichier `/etc/systemd/resolved.conf` sur chaque routeur ***Spoke*** pour affecter directement l'adresse de résolution DNS.

> ⚠ Attention
>
> L'affectation de l'adresse IPv4 ou IPv6 de résolution DNS pose problème. En effet, si le démon pppd propose bien deux adresses via l'option `usepeerdns`, ces propositions ne sont pas prises en charge par le service systemd-resolved.
>
> On contourne cette difficulté en affectant une adresse IPv4 directement au service systemd-resolved.

```
grep -Ev '(^#|^$)'  /etc/systemd/resolved.conf
[Resolve]
DNS=172.16.0.2
```

N'oubliez pas de relancer le service pour prendre en compte les modifications du fichier.

```
sudo systemctl restart systemd-resolved
```

À ce stade de la configuration, les sessions PPP des deux routeurs ***Spoke*** sont en place et on peut analyser les messages présents dans les journaux système et identifier les traitements réalisés par les différentes fonctions des protocoles.

Q97. Comment vérifier que le protocole PPPoE a bien permis d'identifier les extrémités en communication dans le réseau de diffusion (VLAN orange) avant de lancer l'ouverture de session PPP ?

Rechercher les options de la commande journalctl pour faire afficher les messages utiles de la journalisation système.

Dans le but de minimiser le nombre de lignes affichées, on peut combiner les commandes journalctl et grep. Les possibilités sont très diverses. Voici un exemple côté routeur Spoke.

```
journalctl --grep pppoe | grep -i pad
```

```
spoke2 pppd[489]: Send PPPOE Discovery V1T1 PADI session 0x0 length 12
spoke2 pppd[489]: Recv PPPOE Discovery V1T1 PADO session 0x0 length 44
spoke2 pppd[489]: Send PPPOE Discovery V1T1 PADR session 0x0 length 36
spoke2 pppd[489]: Recv PPPOE Discovery V1T1 PADS session 0x1 length 12
```

L'extrait ci-dessus montre la séquence spécifique au protocole PPPoE :

- Découverte avec émission de la trame PADI depuis le routeur ***Spoke***.
- Offre avec réception de la trame PADO depuis le routeur ***Hub***.
- Requête avec émission de la trame PADR depuis le routeur ***Spoke*** pour accepter l'offre.
- Ouverture de session avec la trame PADS quand les deux extrémités de la liaison point à point sont en accord.

Côté routeur ***Hub***, les journaux de chaque unité de service `pppoe-serverX` permettent de vérifier que les adresses MAC correspondent bien au bon routeur ***Spoke*** pour chaque côté du triangle de la topologie.

Voici deux exemples pour la maquette de rédaction de ce document.

```
journalctl -u pppoe-server1.service -n 100  | grep created
```

```
hub pppoe-server[621]: Session 1 created for client b8:ad:ca:fe:00:06 (10.44.1.2) on enp0s1.441 using Service-Name ''
```

```
journalctl -u pppoe-server2.service -n 100  | grep created
```

```
hub pppoe-server[673]: Session 1 created for client b8:ad:ca:fe:00:07 (10.44.3.2) on enp0s1.443 using Service-Name ''
```

Q98. Quels sont les messages des journaux système qui montrent que la session PPP a bien été établie ?

Après avoir consulté la page *Point-to-Point Protocol* repérer les messages relatifs aux deux sous-couches LCP et NCP du protocole PPP.

Les messages PPP sont présents sur tous les routeurs de la topologie. Voici deux exemples prélevés côté ***Hub*** et côté ***Spoke***.

- extrait des négociations de paramètres et de l'authentification dans la sous-couche LCP côté ***Hub***.

```
journalctl -u pppoe-server2.service -n 100  | grep -E '(LCP|EAP)'
```

```
hub pppd[673]: sent [LCP ConfReq id=0x1 <mru 1492> <auth eap> <magic 0xef9c0d86>]
hub pppd[673]: rcvd [LCP ConfAck id=0x1 <mru 1492> <auth eap> <magic 0xef9c0d86>]
hub pppd[673]: rcvd [LCP ConfReq id=0x1 <mru 1492> <magic 0x8b9f1855>]
hub pppd[673]: sent [LCP ConfAck id=0x1 <mru 1492> <magic 0x8b9f1855>]
hub pppd[673]: sent [LCP EchoReq id=0x0 magic=0xef9c0d86]
hub pppd[673]: sent [EAP Request id=0x95 Identity <Message "Name">]
hub pppd[673]: rcvd [LCP EchoReq id=0x0 magic=0x8b9f1855]
hub pppd[673]: sent [LCP EchoRep id=0x0 magic=0xef9c0d86]
hub pppd[673]: rcvd [LCP EchoRep id=0x0 magic=0x8b9f1855]
hub pppd[673]: rcvd [EAP Response id=0x95 Identity <Name "spoke_site2">]
hub pppd[673]: EAP: unauthenticated peer name "spoke_site2"
hub pppd[673]: EAP id=0x95 'Identify' -> 'MD5Chall'
hub pppd[673]: sent [EAP Request id=0x96 MD5-Challenge <Value 14 d4 e2 ec 00 a1 26 ca f8 98 c5 7e d6 f4 a7 ef d7> <Name
hub pppd[673]: rcvd [EAP Response id=0x96 MD5-Challenge <Value cd 5d 07 6a 5b 59 2b 1c ec f8 d6 7f 9b 40 44 63> <Name "s
hub pppd[673]: sent [EAP Success id=0x97]
hub pppd[673]: EAP id=0x97 'MD5Chall' -> 'Open'
```

Dans l'exemple ci-dessus, on repère immédiatement le dernier message qui conclue la phase d'authentification du routeur ***Spoke*** auprès du routeur ***Hub*** avec succès.

Plus haut, on repère aussi l'identité utilisée pour cette authentification : `spoke_site2`.

- Extrait des mêmes négociations de paramètres et de l'authentification dans la sous-couche LCP côté *Spoke*.

```
journalctl -u ppp -n 100 | grep -E '(LCP|EAP)'
```

```
spoke2 pppd[489]: sent [LCP ConfReq id=0x1 <mru 1492> <magic 0x8b9f1855>]
spoke2 pppd[489]: rcvd [LCP ConfReq id=0x1 <mru 1492> <auth eap> <magic 0xef9c0d86>]
spoke2 pppd[489]: sent [LCP ConfAck id=0x1 <mru 1492> <auth eap> <magic 0xef9c0d86>]
spoke2 pppd[489]: sent [LCP ConfReq id=0x1 <mru 1492> <magic 0x8b9f1855>]
spoke2 pppd[489]: rcvd [LCP ConfAck id=0x1 <mru 1492> <magic 0x8b9f1855>]
spoke2 pppd[489]: sent [LCP EchoReq id=0x0 magic=0x8b9f1855]
spoke2 pppd[489]: rcvd [LCP EchoReq id=0x0 magic=0xef9c0d86]
spoke2 pppd[489]: sent [LCP EchoRep id=0x0 magic=0x8b9f1855]
spoke2 pppd[489]: rcvd [EAP Request id=0x95 Identity <Message "Name">]
spoke2 pppd[489]: EAP: Identity prompt "Name"
spoke2 pppd[489]: sent [EAP Response id=0x95 Identity <Name "spoke_site2">]
spoke2 pppd[489]: rcvd [LCP EchoRep id=0x0 magic=0xef9c0d86]
spoke2 pppd[489]: rcvd [EAP Request id=0x96 MD5-Challenge <Value 14 d4 e2 ec 00 a1 26 ca f8 98 c5 7e d6 f4 a7 ef d7> <Na
spoke2 pppd[489]: sent [EAP Response id=0x96 MD5-Challenge <Value cd 5d 07 6a 5b 59 2b 1c ec f8 d6 7f 9b 40 44 63> <Name
spoke2 pppd[489]: rcvd [EAP Success id=0x97]
spoke2 pppd[489]: EAP authentication succeeded
```

Comme dans l'extrait précédent, on retrouve les traces de l'identité utilisée et du succès de l'authentification.

- On reprend le même travail pour la sous-couche NCP dans laquelle on recherche l'attribution des adresses des liaisons point à point côté *Hub*.

  Dans les journaux, la lettre `N` est remplacée par IP pour le protocole IPv4 et IPV6 pour le protocole IPv6.

```
journalctl -u pppoe-server1.service -n 100  | grep -E '(IP.*CP|CCP)'
```

```
pppd[1055]: peer from calling number b8:ad:ca:fe:00:06 authorized
hub pppd[1055]: sent [CCP ConfReq id=0x1 <bsd v1 15>]
hub pppd[1055]: sent [IPCP ConfReq id=0x1 <addr 10.44.1.1>]
hub pppd[1055]: sent [IPV6CP ConfReq id=0x1 <addr fe80::0421:e270:de27:332c>]
hub pppd[1055]: EAP id=0xaa 'MD5Chall' -> 'Open'
hub pppd[1055]: rcvd [IPCP ConfReq id=0x3 <addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
hub pppd[1055]: sent [IPCP ConfNak id=0x3 <addr 10.44.1.2> <ms-dns1 172.16.0.2> <ms-dns2 172.16.0.2>]
hub pppd[1055]: rcvd [IPV6CP ConfReq id=0x2 <addr fe80::79ea:402e:6158:8922>]
hub pppd[1055]: sent [IPV6CP ConfAck id=0x2 <addr fe80::79ea:402e:6158:8922>]
hub pppd[1055]: rcvd [CCP ConfReq id=0x2]
hub pppd[1055]: sent [CCP ConfAck id=0x2]
hub pppd[1055]: rcvd [CCP ConfRej id=0x1 <bsd v1 15>]
hub pppd[1055]: sent [CCP ConfReq id=0x2]
hub pppd[1055]: rcvd [IPCP ConfAck id=0x1 <addr 10.44.1.1>]
hub pppd[1055]: rcvd [IPV6CP ConfAck id=0x1 <addr fe80::0421:e270:de27:332c>]
hub pppd[1055]: local  LL address fe80::0421:e270:de27:332c
hub pppd[1055]: remote LL address fe80::79ea:402e:6158:8922
hub pppd[1055]: Script /etc/ppp/ipv6-up started (pid 1061)
hub pppd[1055]: rcvd [IPCP ConfReq id=0x4 <addr 10.44.1.2> <ms-dns1 172.16.0.2> <ms-dns2 172.16.0.2>]
hub pppd[1055]: sent [IPCP ConfAck id=0x4 <addr 10.44.1.2> <ms-dns1 172.16.0.2> <ms-dns2 172.16.0.2>]
hub pppd[1055]: Script /etc/ppp/ip-pre-up started (pid 1062)
hub pppd[1055]: Script /etc/ppp/ip-pre-up finished (pid 1062), status = 0x0
hub pppd[1055]: local  IP address 10.44.1.1
hub pppd[1055]: remote IP address 10.44.1.2
hub pppd[1055]: Script /etc/ppp/ip-up started (pid 1066)
hub pppd[1055]: Script /etc/ppp/ipv6-up finished (pid 1061), status = 0x0
hub pppd[1055]: rcvd [CCP ConfAck id=0x2]
hub pppd[1055]: Script /etc/ppp/ip-up finished (pid 1066), status = 0x0
```

- On poursuit la même démarche côté routeurs *Spoke*.

Le point important ici, c'est relever le statut des scripts d'application des routes par défaut vers la liaison PPP pour les routeurs ***Spoke***. En l'état actuel de la configuration, l'attribution de la route par défaut IPv6 ne se fait pas. Il faudra donc corriger ça dans la partie suivante.

```
journalctl -u ppp.service -n 100  -f
```

```
pppd[496]: EAP authentication succeeded
spoke1 pppd[496]: peer from calling number B8:AD:CA:FE:00:05 authorized
spoke1 pppd[496]: sent [IPCP ConfReq id=0x3 <addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
spoke1 pppd[496]: sent [IPV6CP ConfReq id=0x2 <addr fe80::79ea:402e:6158:8922>]
spoke1 pppd[496]: rcvd [CCP ConfReq id=0x1 <bsd v1 15>]
spoke1 pppd[496]: sent [CCP ConfReq id=0x2]
spoke1 pppd[496]: sent [CCP ConfRej id=0x1 <bsd v1 15>]
spoke1 pppd[496]: rcvd [IPCP ConfReq id=0x1 <addr 10.44.1.1>]
spoke1 pppd[496]: sent [IPCP ConfAck id=0x1 <addr 10.44.1.1>]
spoke1 pppd[496]: rcvd [IPV6CP ConfReq id=0x1 <addr fe80::0421:e270:de27:332c>]
spoke1 pppd[496]: sent [IPV6CP ConfAck id=0x1 <addr fe80::0421:e270:de27:332c>]
spoke1 pppd[496]: rcvd [IPCP ConfNak id=0x3 <addr 10.44.1.2> <ms-dns1 172.16.0.2> <ms-dns2 172.16.0.2>]
spoke1 pppd[496]: sent [IPCP ConfReq id=0x4 <addr 10.44.1.2> <ms-dns1 172.16.0.2> <ms-dns2 172.16.0.2>]
spoke1 pppd[496]: rcvd [IPV6CP ConfAck id=0x2 <addr fe80::79ea:402e:6158:8922>]
spoke1 pppd[496]: local  LL address fe80::79ea:402e:6158:8922
spoke1 pppd[496]: remote LL address fe80::0421:e270:de27:332c
spoke1 pppd[496]: Script /etc/ppp/ipv6-up started (pid 1640)
spoke1 pppd[496]: rcvd [CCP ConfAck id=0x2]
spoke1 pppd[496]: rcvd [CCP ConfReq id=0x2]
spoke1 pppd[496]: sent [CCP ConfAck id=0x2]
spoke1 pppd[496]: rcvd [IPCP ConfAck id=0x4 <addr 10.44.1.2> <ms-dns1 172.16.0.2> <ms-dns2 172.16.0.2>]
spoke1 pppd[496]: Script /etc/ppp/ip-pre-up started (pid 1642)
spoke1 pppd[496]: Script /etc/ppp/ip-pre-up finished (pid 1642), status = 0x0
spoke1 pppd[496]: local  IP address 10.44.1.2
spoke1 pppd[496]: remote IP address 10.44.1.1
spoke1 pppd[496]: primary   DNS address 172.16.0.2
spoke1 pppd[496]: secondary DNS address 172.16.0.2
spoke1 pppd[496]: Script /etc/ppp/ip-up started (pid 1645)
spoke1 pppd[496]: Script /etc/ppp/ipv6-up finished (pid 1640), status = 0x0
spoke1 pppd[496]: Script /etc/ppp/ip-up finished (pid 1645), status = 0x0
```

5. Interconnexion IPv4 et IPv6

À ce stade des manipulations, les routeurs ***Spoke*** utilisent leur session PPP et le routage IPv4 pour accéder à tous les réseaux.

L'objectif de cette section est de valider les communications entre les routeurs ***Spoke***, aussi bien avec le protocole réseau IPv4 qu'avec le protocole réseau IPv6.

Il s'agit donc de configurer les routes statiques vers les réseaux d'hébergement des deux sites distants côté routeur ***Hub*** et de valider ou d'ajouter les routes par défaut côté routeurs ***Spoke***.

IPv6 nécessite une attention particulière. En effet, à la suite de l'établissement de session PPP, on ne dispose que d'adresses de lien local. Ces adresses ne permettent pas d'acheminer du trafic vers un autre réseau quel qu'il soit. Il est donc nécessaire de mettre en place des interfaces commutées virtuelles (SVI) avec des adresses IPv6 de type ULA (***Unique Local Address***) pour être en mesure d'échanger du trafic entre des réseaux différents.

Q99. Comment connaître l'état actuel des communications entre les routeurs de la topologie ***Hub and Spoke*** ?

On doit lancer une série de tests ICMP pour chaque protocole de couche réseau et identifier les dysfonctionnements.

Protocole IPv4

Commençons par les tests des routeurs ***Spoke*** vers Internet. Comme les sessions PPP sont établies à la suite de la partie précédente, tous les paquets transitent avec succès.

```
etu@spoke1:~$ ping -qc2 9.9.9.9
PING 9.9.9.9 (9.9.9.9) 56(84) bytes of data.

--- 9.9.9.9 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 29.534/35.666/41.798/6.132 ms
```

```
etu@spoke2:~$ ping -qc2 9.9.9.9
PING 9.9.9.9 (9.9.9.9) 56(84) bytes of data.

--- 9.9.9.9 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 29.515/29.622/29.730/0.107 ms
```

Passons aux communications entre routeurs ***Spoke*** entre les extrémités des liaisons point à point. Là aussi, tout fonctionne puisque le routeur ***Hub*** détient le plan d'adressage.

```
etu@spoke2:~$ ip addr ls dev ppp0
5: ppp0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1492 qdisc fq_codel state UNKNOWN group default qlen 3
    link/ppp
    inet 10.44.3.2 peer 10.44.3.1/32 scope global ppp0
       valid_lft forever preferred_lft forever
    inet6 fe80::5826:d9cb:d027:4cc9 peer fe80::552:a7c:9fba:55af/128 scope link nodad
       valid_lft forever preferred_lft forever
```

```
etu@spoke2:~$ ping -qc2 10.44.1.2
PING 10.44.1.2 (10.44.1.2) 56(84) bytes of data.

--- 10.44.1.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 2.249/2.364/2.479/0.115 ms
```

```
etu@hub:~$ ip route ls
default via 192.168.104.129 dev enp0s1.360 proto static
10.44.1.2 dev ppp0 proto kernel scope link src 10.44.1.1
10.44.3.2 dev ppp1 proto kernel scope link src 10.44.3.1
192.168.104.128/29 dev enp0s1.360 proto kernel scope link src 192.168.104.130
```

Protocole IPv6

Pour IPv6, la configuration est clairement incomplète dans la mesure où il est impossible d'acheminer du trafic au-delà du lien local de chaque routeur ***Spoke***.

```
etu@spoke1:~$ ip -6 route get 2620:fe::fe
RTNETLINK answers: Network is unreachable
```

```
etu@spoke1:~$ ip -6 route ls default
```

La route par défaut n'existe pas.

Pour conclure ces tests, nous devons mettre en place les réseaux d'hébergement de chaque routeur ***Spoke*** ainsi que les routes statiques pour avancer dans l'interconnexion des sites distants.

Q100. Comment créer les routes par défaut IPv4 et IPv6 sur les deux routeurs ***Spoke*** ?

Créer les scripts exécutables `defaultroute` pour chaque protocole réseau et renouveler les sessions PPP.

Pour IPv4, c'est le répertoire est `/etc/ppp/ip-up.d/` qui doit contenir le script exécutable `defaultroute`.

```
cat << 'EOF' | sudo tee /etc/ppp/ip-up.d/defaultroute
#!/bin/sh

if [ -z "${CONNECT_TIME}" ]; then
 ip route add default dev ${PPP_IFACE}
fi
EOF
```

```
sudo chmod +x /etc/ppp/ip-up.d/defaultroute
```

Pour IPv6, c'est le répertoire est `/etc/ppp/ipv6-up.d/` qui doit contenir le script exécutable appelé `defaultroute`.

```
cat << 'EOF' | sudo tee /etc/ppp/ipv6-up.d/defaultroute
#!/bin/sh

if [ -z "${CONNECT_TIME}" ]; then
    ip -6 route add default dev ${PPP_IFACE}
fi
EOF
```

```
sudo chmod +x /etc/ppp/ipv6-up.d/defaultroute
```

Comme demandé, pas oublier de relancer les sessions PPP pour provoquer la création des routes par défaut sur chaque routeur ***Spoke***.

```
sudo systemctl restart ppp.service
```

Une fois les sessions PPP renouvelées, on dispose d'une solution pour router les paquets IPv6 vers les autres réseaux.

```
ip -6 route get 2620:fe::fe
2620:fe::fe from :: dev ppp0 src fda0:7a62:14::1 metric 1024 pref medium
```

Malheureusement, ce n'est pas suffisant pour acheminer du trafic depuis le routeur ***Hub***. Celui-ci n'a aucune connaissance des adresses réseau des réseaux d'hébergement des routeurs ***Spoke***. L'ajout des routes statiques vers ces réseaux 'hébergement est justement l'objet de la question suivante.

Q101. Comment créer les routes statiques vers les réseaux d'hébergement sur le routeur ***Hub*** ?

Créer un script exécutable par protocole réseau dans lequel on utilise les noms d'interfaces PPP dérivés des options `-u` utilisées dans les paramètres de configuration des deux serveurs PPPoE.

Pour IPv4, c'est le répertoire est `/etc/ppp/ip-up.d/` qui doit contenir le script exécutable `staticroute`.

```
cat << 'EOF' | sudo tee /etc/ppp/ip-up.d/staticroute
#!/bin/bash

if [ -z "${CONNECT_TIME}" ]; then
    case "${PPP_IFACE}" in
        "ppp0")
            ip route add 10.0.10.0/24 dev ${PPP_IFACE}
            ;;
        "ppp1")
            ip route add 10.0.20.0/24 dev ${PPP_IFACE}
            ;;
    esac
fi
EOF
```

```
sudo chmod +x /etc/ppp/ip-up.d/staticroute
```

Pour IPv6, c'est le répertoire est `/etc/ppp/ipv6-up.d/` qui doit contenir le script exécutable appelé `staticroute`.

```
cat << 'EOF' | sudo tee /etc/ppp/ipv6-up.d/staticroute
#!/bin/bash

if [ -z "${CONNECT_TIME}" ]; then
    case "${PPP_IFACE}" in
        "ppp0")
            ip -6 route add fda0:7a62:a::/64 dev ${PPP_IFACE}
            ;;
        "ppp1")
            ip -6 route add fda0:7a62:14::/64 dev ${PPP_IFACE}
            ;;
    esac
fi
EOF
```

```
sudo chmod +x /etc/ppp/ipv6-up.d/staticroute
```

Une fois de plus, il faut relancer les sessions PPP pour observer les résultats et lancer les tests ICMP.

Auparavant, on peut afficher les tables de routages IPv4 et IPv6 du routeur ***Hub*** pour vérifier la présence des nouvelles routes statiques.

- En direction du premier routeur ***Spoke***.

```
ip route ls dev ppp0
```

```
10.0.10.0/24 scope link
10.44.1.2 proto kernel scope link src 10.44.1.1
```

```
ip -6 route ls dev ppp0
```

```
fda0:7a62:a::/64 metric 1024 pref medium
fe80::2c4a:3bb9:eead:8fa5 proto kernel metric 256 pref medium
fe80::cca6:346e:80d0:20cf proto kernel metric 256 pref medium
```

- En direction du second routeur ***Spoke***.

```
ip route ls dev ppp1
```

```
10.0.20.0/24 scope link
10.44.3.2 proto kernel scope link src 10.44.3.1
```

```
ip -6 route ls dev ppp1
```

```
fda0:7a62:14::/64 metric 1024 pref medium
fe80::b4d6:f38c:a930:c8f5 proto kernel metric 256 pref medium
fe80::f0fe:10bd:88f0:cb26 proto kernel metric 256 pref medium
```

Q102. Comment créer et configurer les interfaces commutées virtuelles sur chaque routeur ***Spoke*** ?

Après avoir installé le paquet `openvswitchi-switch`, compléter le fichier de configuration netplan.io pour déclarer un commutateur applé `asw-host` et l'interface SVI correspondant au réseau d'hébergement de site (VLAN vert)

On installe le paquet du commutateur virtuel.

```
sudo apt -y install openvswitch-switch
```

On créé le nouveau commutateur et l'interface commutée virtuel en respectant bien le plan d'adressage.

Voici un exemple de fichier de déclaration `/etc/netplan/enp0s1.yaml` pour le premier routeur ***Spoke***.

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s1:
      dhcp4: false
      dhcp6: false
      accept-ra: false

  openvswitch: {}

  bridges:
    asw-host:
      openvswitch: {}

  vlans:
    enp0s1.440: # VLAN violet
      id: 440
      link: enp0s1
      addresses:
        - fe80:1b8::2/64
    enp0s1.441: # VLAN orange
      id: 441
      link: enp0s1
      addresses: []
    vlan10:     # VLAN vert
      id: 10
      link: asw-host
      addresses:
        - 10.0.10.1/24
        - fda0:7a62:a::1/64
        - fe80:a::1/64
```

Le même travail doit être réalisé sur le second routeur ***Spoke*** pour que la configuration soit complète.

Q103. Quels sont les tests ICMP à réaliser pour valider les communications entre les routeurs ***Spoke*** ?

Il faut valider les communications réseau entre les adresses des deux interfaces commutées virtuelles des deux routeurs ***Spoke***.

Il faut aussi vérifier que ces mêmes routeurs ***Spoke*** communiquent avec le protocole IPv6 vers l'Internet.

On commence par tester avec succès les échanges entre le second routeur ***Spoke*** et le premier avec IPv4.

```
etu@spoke2:~$ ping -qc2 10.0.10.1
PING 10.0.10.1 (10.0.10.1) 56(84) bytes of data.

--- 10.0.10.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 1.799/1.943/2.088/0.144 ms
```

On poursuit dans le même sens avec IPv6.

```
etu@spoke2:~$ ping -qc2 fda0:7a62:a::1
PING fda0:7a62:a::1 (fda0:7a62:a::1) 56 data bytes

--- fda0:7a62:a::1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 2.146/2.272/2.398/0.126 ms
```

Enfin, on termine avec les communications IPv6 depuis les deux routeurs ***Spoke***.

```
etu@spoke1:~$ ping -qc2 2620:fe::fe
PING 2620:fe::fe (2620:fe::fe) 56 data bytes

--- 2620:fe::fe ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 41.104/47.623/54.142/6.519 ms
```

```
etu@spoke2:~$ ping -qc2 2620:fe::fe
PING 2620:fe::fe (2620:fe::fe) 56 data bytes

--- 2620:fe::fe ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 40.897/47.183/53.470/6.286 ms
```

Au terme de cette partie, les tables de routage de routeurs de la topologie ***Hub and Spoke*** sont complètes et les communications IPv4 et IPv6 fonctionnent entre les routeurs d'extrémités.

6. Installation et gestion des conteneurs

Maintenant que le routage du trafic réseau est complet, on peut passer à l'hébergement des services sur les réseaux des sites distants.

Dans cette partie, on installe le gestionnaire de conteneurs systèmes Incus et on le configure pour que les conteneurs puissent échanger entre les sites.

Avant d'aborder les questions, on s'assure que les outils et les permissions sont en place.

- Le gestionnaire Incus est installé sur chaque routeur ***Spoke***.

```
sudo apt -y install incus
```

- L'utilisateur normal `etu` a la capacité à gérer les conteneurs en appartenant aux groupes `incus` et `incus-admin`.

```
for grp in incus incus-admin; do sudo adduser etu $grp; done
```

  N'oubliez pas de vous déconnecter/reconnecter pour que l'appartenance aux groupes soit effective.

```
groups
etu adm sudo users incus-admin incus
```

- La configuration par défaut du gestionnaire de conteneurs doit être initialisée.

  Voici un exemple de déclaration de configuration.

```
incus admin init
```

```
Would you like to use clustering? (yes/no) [default=no]:
Do you want to configure a new storage pool? (yes/no) [default=yes]:
Name of the new storage pool [default=default]:
Where should this storage pool store its data? [default=/var/lib/incus/storage-pools/default]:
Would you like to create a new local network bridge? (yes/no) [default=yes]: no
Would you like to use an existing bridge or host interface? (yes/no) [default=no]: yes
Name of the existing bridge or host interface: asw-host
Would you like the server to be available over the network? (yes/no) [default=no]:
Would you like stale cached images to be updated automatically? (yes/no) [default=yes]:
Would you like a YAML "init" preseed to be printed? (yes/no) [default=no]:
```

  La configuration du raccordement réseau doit être complétée avec les instructions suivantes.

```
incus profile device set default eth0 nictype bridged
```

```
incus profile device set default eth0 vlan 20
```

> **Avertissement**
>
> L'exemple ci-dessus doit être adapté au contexte réseau en changeant le numéro de VLAN.

- L'adressage automatique IPv4 et IPv6 doit aussi être installé et configuré sur chaque routeur ***Spoke*** avec l'outil dnsmasq.

  Voici un exemple de fichier de configuration à mettre en place. Bien sûr, le nom d'interface sur laquelle les services sont actifs et les adresses IPv4 doivent être changées.

```
sudo apt -y install dnsmasq
```

```
cat << EOF | sudo tee /etc/dnsmasq.conf
# Specify Container VLAN interface
interface=vlan10

# Enable DHCPv4 on Container VLAN
dhcp-range=10.0.10.100,10.0.10.200,3h

# Enable IPv6 router advertisements
enable-ra

# Enable SLAAC
dhcp-range=::,constructor:vlan10,ra-names,slaac

# Optional: Specify DNS servers
dhcp-option=option:dns-server,172.16.0.2,9.9.9.9
dhcp-option=option6:dns-server,[2001:678:3fc:3::2],[2620:fe::fe]

# Avoid DNS listen port conflict between dnsmasq and systemd-resolved
port=0
EOF
```

```
sudo systemctl restart dnsmasq.service
```

Q104. Comment lancer 3 nouveaux conteneurs sur chaque routeur ***Spoke*** ?

Rechercher dans les options de la commande incus le moyen de créer un nouveau conteneur système.

Tester son exécution avec un conteneur de type `debian/trixie`.

C'est l'option `launch` qui correspond au besoin. Voici un exemple de création de trois nouveaux conteneurs.

```
for i in {0..2}; do incus launch images:debian/trixie c$i; done
```

```
Launching c0
Launching c1
Launching c2
```

```
etu@spoke1:~$ incus ls
+------+---------+--------------------+-----------------------------------------+-----------+-----------+
| NAME |  STATE  |        IPV4        |                  IPV6                   |   TYPE    | SNAPSHOTS |
+------+---------+--------------------+-----------------------------------------+-----------+-----------+
| c0   | RUNNING | 10.0.10.140 (eth0) | fda0:7a62:a:0:216:3eff:fe6a:395c (eth0) | CONTAINER | 0         |
+------+---------+--------------------+-----------------------------------------+-----------+-----------+
| c1   | RUNNING | 10.0.10.196 (eth0) | fda0:7a62:a:0:216:3eff:fe37:f12a (eth0) | CONTAINER | 0         |
+------+---------+--------------------+-----------------------------------------+-----------+-----------+
| c2   | RUNNING | 10.0.10.116 (eth0) | fda0:7a62:a:0:216:3eff:fed9:a25b (eth0) | CONTAINER | 0         |
+------+---------+--------------------+-----------------------------------------+-----------+-----------+
```

Q105. Comment tester les communications réseau depuis les conteneurs des routeurs ***Spoke*** ?

Reprendre les tests ICMP déjà réalisés depuis les routeurs en les exécutant dans les conteneurs.

Pour traiter cette question, on peut utiliser une boucle qui rassemble les tests IPv4 et IPv6 :

```
for i in {0..2}
do
 echo ">>>>>>>>>>>>>>>> c$i"
 incus exec c$i -- ping -qc2 9.9.9.9
 incus exec c$i -- ping -qc2 2620:fe::fe
done
```

On vérifie qu'aucun paquet n'a été perdu avec le message : `0% packet loss`.

On peut aussi tester les services de la couche application avec la gestion de paquets.

```
for i in {0..2}
do
 echo ">>>>>>>>>>>>>>>> c$i"
 incus exec c$i -- apt -y update
 incus exec c$i -- apt -y full-upgrade
 incus exec c$i -- apt clean
done
```

Q106. Comment combiner l'adressage automatique à l'adressage statique pour chaque conteneur ?

On reprend les traitements réalisés dans le support *Routage inter-VLAN et protocole PPPoE* avec la création d'un script qui regroupe les traitements de gestion de paquets et de création du fichier de déclaration des paramètres des interfaces réseau des conteneurs.

Voici un exemple de script pour un routeur ***Spoke***.

```
#!/bin/bash

# Préparation -> générer la liste des conteneurs actifs
clist=$(incus list status=running -c n -f compact | grep -v NAME | tr '\n' ' ' | tr -s ' ')

addr_idx=0
for c in $clist; do
  echo ">>>>>>>>>>>>>>>> $c"

# Étape 1 -> installer le paquet netplan.io
  incus exec $c -- apt -y install netplan.io

# Étape 2 -> générer le fichier de configuration réseau YAML
$(cat << EOF > eth0.yaml
network:
  version: 2
  renderer: networkd
  ethernets:
    eth0:
      dhcp4: false
      dhcp6: false
      accept-ra: true
      addresses:
        - 10.0.10.$((addr_idx + 10))/24
        - fda0:7a62:a::$(printf "%x" $((addr_idx +10)))/64
      routes:
        - to: default
          via: 10.0.10.1
        - to: "::/0"
          via: fe80:a::1
          on-link: true
      nameservers:
        addresses:
          - 172.16.0.2
          - 2001:678:3fc:3::2
EOF
)

# Étape 3 -> transférer le fichier de déclaration YAML
  incus file push eth0.yaml $c/etc/netplan/eth0.yaml

# Étape 4 -> effacer le fichier /etc/systemd/network/eth0.network
  incus exec $c -- rm /etc/systemd/network/eth0.network

# Étape 5 -> appliquer la nouvelle configuration
  incus exec $c -- netplan apply

  ((addr_idx++))
done

exit 0
```

Si le script ci-dessus est enregistré dans le fichier `static-addr.sh`, on peut le lancer.

```
bash static-addr.sh
```

Les résultats des traitements sont visibles en affichant la liste des conteneurs actifs.

```
incus ls
```

```
+------+---------+-------------------+-----------------------------------------+-----------+-----------+
| NAME |  STATE  |       IPV4        |                  IPV6                   |   TYPE    | SNAPSHOTS |
+------+---------+-------------------+-----------------------------------------+-----------+-----------+
| c0   | RUNNING | 10.0.10.10 (eth0) | fda0:7a62:a::a (eth0)                   | CONTAINER | 0         |
|      |         |                   | fda0:7a62:a:0:216:3eff:fe6a:395c (eth0) |           |           |
+------+---------+-------------------+-----------------------------------------+-----------+-----------+
| c1   | RUNNING | 10.0.10.11 (eth0) | fda0:7a62:a::b (eth0)                   | CONTAINER | 0         |
|      |         |                   | fda0:7a62:a:0:216:3eff:fe37:f12a (eth0) |           |           |
+------+---------+-------------------+-----------------------------------------+-----------+-----------+
| c2   | RUNNING | 10.0.10.12 (eth0) | fda0:7a62:a::c (eth0)                   | CONTAINER | 0         |
|      |         |                   | fda0:7a62:a:0:216:3eff:fed9:a25b (eth0) |           |           |
+------+---------+-------------------+-----------------------------------------+-----------+-----------+
```

On doit suivre le même processus pour le second routeur ***Spoke***.

Q107. Comment ouvrir et tester des services Web dans les réseaux d'hébergement ?

On commence par installer un paquet comme nginx dans chaque conteneur et on effectue les tests d'accès ensuite.

pour simuler la consultation de pages Web depuis les conteneurs, on installe aussi un paquet comme wget.

L'installation des paquets se fait à l'aide d'une boucle.

```
for i in {0..2}
do
    echo ">>>>>>>>>>>>>>>>> c$i"
    incus exec c$i -- apt update
    incus exec c$i -- apt -y install nginx wget
    incus exec c$i -- apt clean
done
```

Le test de disponibilité des serveurs Web se fait aussi à l'aide d'une boucle.

```
for i in {0..2}
do
    echo ">>>>>>>>>>>>>>>>> c$i"
    incus exec c$i -- ss -at '( sport = :http )'
done
```

L'affichage produit par la commande ss et ses paramètres montre que le service `HTTP` est en écoute dans chaque conteneur.

Q108. Comment valider l'accès à ces services Web à partir du routeur ***Spoke*** situé à l'autre extrémité de la topologie en triangle ?

Il s'agit de faire un test au niveau de la couche application. À la console, les deux outils adaptés sont wget et curl.

Voici un exemple de test pour chaque protocole de la couche réseau avec wget dans le contexte de la maquette.

```
for addr in {10..12}
do
    wget -O /dev/null http://10.0.20.$addr 2>&1 | grep "HTTP "
done
```

```
requête HTTP transmise, en attente de la réponse… 200 OK
requête HTTP transmise, en attente de la réponse… 200 OK
requête HTTP transmise, en attente de la réponse… 200 OK
```

```
for addr in {10..12}
do
    wget -O /dev/null http://[fda0:7a62:14::$(printf "%x" $addr)] 2>&1 | grep "HTTP "
done
```

```
requête HTTP transmise, en attente de la réponse… 200 OK
requête HTTP transmise, en attente de la réponse… 200 OK
requête HTTP transmise, en attente de la réponse… 200 OK
```

Q109. Comment matérialiser le chemin suivi par les paquets IPv4 ou IPv6 entre les deux routeurs ***Spoke*** ?

On peut utiliser un outil comme tracepath fourni avec le paquet `iputils-tracepath`. Il suffit d'installer ce paquet dans les conteneurs du second routeur ***Spoke*** et de lancer les tests à destination des conteneurs du premier routeur ***Spoke***.

On installe le paquet dans les conteneurs su second routeur ***Spoke***.

```
for i in {0..2}
do
    echo ">>>>>>>>>>>>>>>>> c$i"
    incus exec c$i -- apt -y install iputils-tracepath
done
```

On lance ensuite les tests, toujours à l'aide d'une boucle.

```
for i in {0..2}
do
    echo ">>>>>>>>>>>>>>>>> c$i"
    incus exec c$i -- tracepath -n 10.0.10.$(($i + 10))
done
```

On obtient les résultats suivants dans lesquels apparait la valeur de l'encapsulation de la quantité de données dans la trame : 1492.

```
>>>>>>>>>>>>>>>> c0
 1?: [LOCALHOST]                      pmtu 1500
 1:  10.0.20.1                                             0.868ms
 1:  10.0.20.1                                             0.112ms
 2:  10.0.20.1                                             0.078ms pmtu 1492
 2:  10.44.3.1                                             1.260ms
 3:  10.44.1.2                                             1.658ms
 4:  10.0.10.10                                            2.244ms reached
     Resume: pmtu 1492 hops 4 back 4
>>>>>>>>>>>>>>>> c1
 1?: [LOCALHOST]                      pmtu 1500
 1:  10.0.20.1                                             1.722ms
 1:  10.0.20.1                                             0.134ms
 2:  10.0.20.1                                             0.056ms pmtu 1492
 2:  10.44.3.1                                             1.193ms
 3:  10.44.1.2                                             1.676ms
 4:  10.0.10.11                                            2.569ms reached
     Resume: pmtu 1492 hops 4 back 4
>>>>>>>>>>>>>>>> c2
 1?: [LOCALHOST]                      pmtu 1500
 1:  10.0.20.1                                             2.055ms
 1:  10.0.20.1                                             0.077ms
 2:  10.0.20.1                                             0.074ms pmtu 1492
 2:  10.44.3.1                                             1.094ms
 3:  10.44.1.2                                             1.774ms
 4:  10.0.10.12                                            2.132ms reached
     Resume: pmtu 1492 hops 4 back 4
```

On suit la même démarche avec le protocole IPv6.

```
for i in {0..2}
do
    echo ">>>>>>>>>>>>>>>> c$i"
    incus exec c$i -- tracepath -n fda0:7a62:a::$(printf "%x" $(($i + 10)))
done
```

Les résultats sont identiques aux précédents à l'adressage près.

```
>>>>>>>>>>>>>>>> c0
 1?: [LOCALHOST]                        0.050ms pmtu 1500
 1:  fda0:7a62:14::1                                       1.815ms
 1:  fda0:7a62:14::1                                       0.129ms
 2:  fda0:7a62:14::1                                       3.201ms pmtu 1492
 2:  2001:678:3fc:168:baad:caff:fefe:5                     1.571ms
 3:  fda0:7a62:a:0:9c04:62ff:fe95:f948                     1.742ms
 4:  fda0:7a62:a::a                                        2.239ms reached
     Resume: pmtu 1492 hops 4 back 4
>>>>>>>>>>>>>>>> c1
 1?: [LOCALHOST]                        0.030ms pmtu 1500
 1:  fda0:7a62:14::1                                       0.572ms
 1:  fda0:7a62:14::1                                       0.116ms
 2:  fda0:7a62:14::1                                       0.071ms pmtu 1492
 2:  2001:678:3fc:168:baad:caff:fefe:5                     1.374ms
 3:  fda0:7a62:a:0:9c04:62ff:fe95:f948                     2.088ms
 4:  fda0:7a62:a::b                                        2.396ms reached
     Resume: pmtu 1492 hops 4 back 4
>>>>>>>>>>>>>>>> c2
 1?: [LOCALHOST]                        0.032ms pmtu 1500
 1:  fda0:7a62:14::1                                       0.666ms
 1:  fda0:7a62:14::1                                       0.091ms
 2:  fda0:7a62:14::1                                       0.093ms pmtu 1492
 2:  2001:678:3fc:168:baad:caff:fefe:5                     1.252ms
 3:  fda0:7a62:a:0:9c04:62ff:fe95:f948                     2.177ms
 4:  fda0:7a62:a::c                                        3.664ms reached
     Resume: pmtu 1492 hops 4 back 4
```

7. Pour conclure...

Ce document présente une configuration détaillée d'une topologie réseau ***Hub & Spoke*** utilisant le protocole PPPoE. Il couvre l'installation et la configuration des routeurs ***Hub*** et ***Spoke***, en établissant des connexions PPP et en configurant le routage IPv4 et IPv6.

Ce support de travaux pratiques aborde également la mise en place de conteneurs sur les routeurs ***Spoke***, démontrant l'hébergement de services web et la communication entre les sites distants. Les étapes sont expliquées de manière approfondie, avec des exemples de commandes et de configurations.

Enfin, le document fournit des méthodes pour tester et valider la connectivité réseau, utilisant des outils comme ping, tracepath et wget. Il offre une approche pratique pour comprendre et mettre en œuvre une topologie réseau complexe avec des protocoles modernes.

Filtrage réseau avec netfilter/nftables

https://www.inetdoc.net

Résumé

Ce support de travaux pratiques est une introduction au filtrage réseau. Il reprend la topologie *Hub & Spoke* du support précédent de la série. Les questions débutent par l'identification des outils et passent à l'application des règles de filtrage avec et sans suivi de communication (*stateful vs stateless inspection*). On introduit aussi les fonctions de traduction d'adresses (NAT).

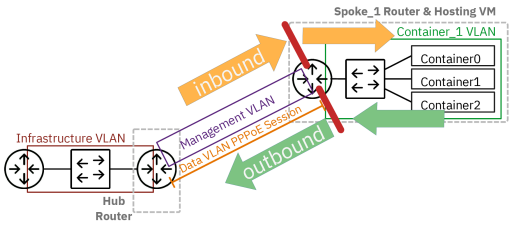


Table des matières

1. Objectifs

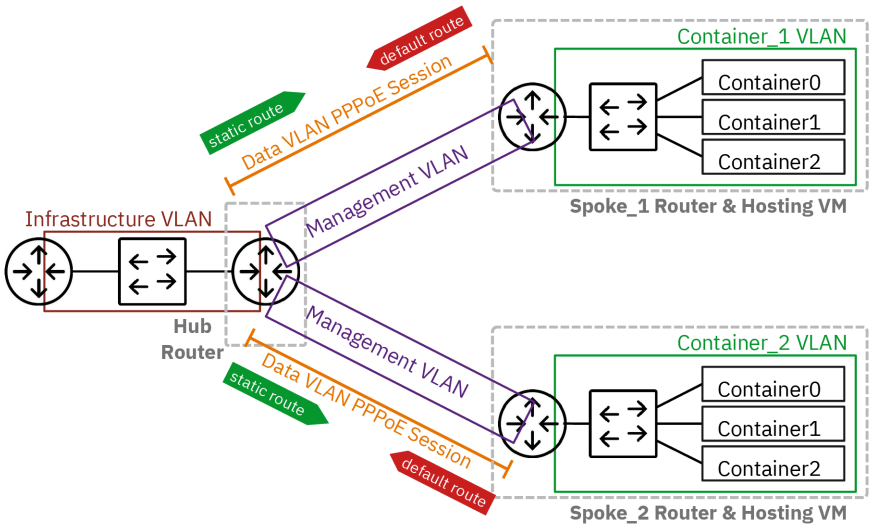
Après avoir réalisé les manipulations présentées dans ce document, les étudiants seront capables de :

1. Configurer et appliquer des règles de filtrage réseau de base sur des routeurs Linux en utilisant nftables.
2. Mettre en place des protections contre l'usurpation d'adresses IP et les attaques par déni de service ICMP.
3. Implémenter un système de protection contre les tentatives d'intrusion sur SSH en utilisant fail2ban.
4. Configurer la traduction d'adresses réseau (NAT) source et destination sur un routeur Linux.
5. Analyser et déboguer les règles de filtrage en utilisant les outils de diagnostic comme conntrack.

2. Architecture réseau étudiée et filtrage existant

Les manipulations sur le système de filtrage réseau présentées ici s'appuient sur la topologie *Hub and Spoke* étudiée dans le support précédent de la série : *Topologie Hub & Spoke avec le protocole PPPoE*.

La topologie étudiée associe trois routeurs qui ont deux rôles distincts.



Topologie entre deux routeurs *Hub* et *Spoke* avec PPPoE

Hub

Traduit mot à mot, le rôle ***Hub*** correspond à un concentrateur. Il concentre tous les flux réseau des routeurs qui ont le rôle ***Spoke***. En effet, les échanges entre deux routeurs ***Spoke*** doivent passer par le routeur ***Hub***.

On lui attribue aussi la fonction de ***Broadband Remote Access Server*** ou BRAS. Dans notre contexte, cette fonction se caractérise par le fait que ce routeur détient le plan d'adressage. C'est lui qui a la responsabilité de délivrer les adresses IP lors de l'initiation de la session PPP.

Spoke

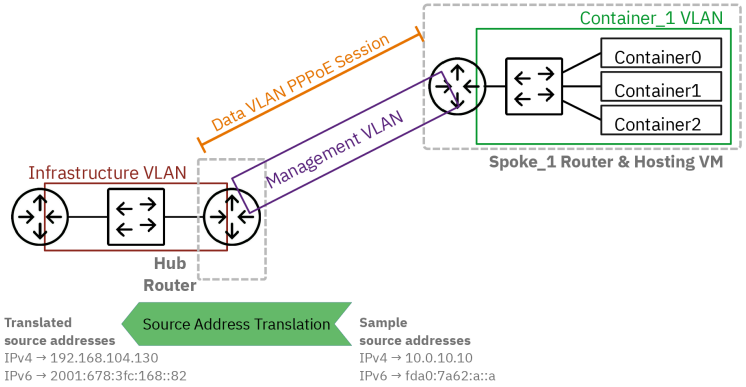
Le rôle ***Spoke*** correspond à un réseau d'extrémité au delà duquel on ne trouve aucune interconnexion. Le routeur ***Spoke*** doit s'adresser au routeur Hub dès qu'il veut acheminer un flux réseau. Il s'agit bien d'un routeur d'extrémité qui ne dispose d'aucun chemin alternatif pour joindre l'Internet.

Dans les réseaux domestiques, la «box» correspond bien au rôle ***Spoke*** dans la mesure où elle se voit attribuer des adresses IPv4 et IPv6 publiques par le fournisseur d'accès Internet. Les seules informations qu'elle détient sont les authentifiants du client de l'opérateur.

Pour commencer, on s'intéresse à l'installation des outils et à la lecture des informations sur le filtrage déjà en place sur le routeur ***Hub***.

2.1. Routage et traduction d'adresses sources (situation de départ)

La situation de départ des manipulations suppose que la topologie ***Hub & Spoke*** est en place et fonctionnelle. On s'appuie sur le support précédent de la série : la *Topologie Hub & Spoke avec le protocole PPPoE* qui comprend déjà un premier niveau de filtrage avec la traduction d'adresses sources de tous les paquets qui sortent vers le réseau d'infrastructure (VLAN rouge).



Traduction d'adresses sources pour les flux sortants du routeur *Hub*.

Q110. Quel est le nom de l'outil de gestion des règles de filtrage réseau ? À quel paquet appartient-il ?

Reprendre la section sur la configuration du routeur ***Hub*** dans le document *Topologie Hub & Spoke avec le protocole PPPoE*.

Il s'agit de la commande nft fournie avec le paquet `nftables`.

```
apt show nftables
```

```
dpkg -L nftables
```

Q111. Quel est l'état du service `systemd` associé ?

Reprendre la section sur la configuration du routeur ***Hub*** dans le document *Topologie Hub & Spoke avec le protocole PPPoE* dans laquelle le service a été activé.

On affiche l'état du service `systemd` fourni avec le paquet `nftables` et on vérifie qu'il est activé (***enabled***).

```
systemctl status nftables
```

Q112. Dans quel fichier sont enregistrés les jeux de règles à appliquer au lancement du système ?

Rechercher dans les informations affichées à la question précédente, la syntaxe de la commande lancée par le service `systemd`.

On identifie le fichier `/etc/nftables` qui sert à stocker les jeux de règles activés lors de l'initialisation du système.

Q113. Comment afficher la liste des règles de filtrage actives ?

Rechercher dans les pages de manuels de la commande nftables l'option qui permet d'afficher la liste du jeu de règles en cours de traitement.

La lecture des pages de manuels conduit à l'option `list` suivie de `ruleset`.

```
sudo nft list ruleset
```

```
table inet nat {
        chain postrouting {
                type nat hook postrouting priority srcnat; policy accept;
                oifname "enp0s1.360" counter packets 89 bytes 7400 masquerade
        }
}
```

2.2. Lecture des règles de traduction d'adresses sources

Dans cette partie, on s'intéresse à l'identification des éléments qui composent le jeu de règles de traduction d'adresses sources appliqué sur le routeur ***Hub*** de la topologie étudiée.

Comme le jeu de règles est déjà présent et actif, il faut comparer les champs des règles du fichier `/etc/nftables.conf` avec la représentation graphique générale *Packet Flow in Netfilter*.

Q114. Quelle est la ***table*** utilisée dans les deux jeux de règles de filtrage appliquées sur le routeur ***Hub*** ?

Il s'agit de la table `nat` que l'on repère en couche liaison de la représentation graphique *Packet Flow in Netfilter*

Q115. Quelle est la ***chaîne*** utilisée dans les deux jeux de règles de filtrage appliquées sur le routeur ***Hub*** ?

Il s'agit de la chaîne `postrouting` que l'on repère en couche liaison à droite de la représentation graphique *Packet Flow in Netfilter*

Comme son nom l'indique, cette chaîne traite les paquets après que la décision de routage ait été prise.

Q116. Comment l'interface réseau sur laquelle les traitements sont appliqués est-elle identifiée ?

Repérer le mot clé placé avant la chaîne de caractère qui contient le nom de l'interface.

La clé placée avant le nom d'interface est `oifname`. Cette clé correspond à ***output interface name***.

La traduction d'adresses sources a lieu en sortie sur l'interface réseau raccordée au réseau d'infrastructure (VLAN rouge).

Q117. Quel est le nom de la ***cible*** utilisée dans les deux jeux de règles de filtrage appliquées sur le routeur ***Hub*** ?

Identifier le mot clé placé à droite en bout de chaîne de traitement après les compteurs.

Le nom utilisé est ***masquerade***. Il caractérise le fait que l'adresse source de tous les paquets sortants par l'interface `enp0s1.360` est remplacée par celle de cette interface, aussi bien avec IPv4 qu'avec IPv6.

2.3. Comptage des paquets et enregistrements des transactions

La traduction d'adresse source entre dans la catégorie du filtrage *Stateful*. Il est nécessaire de conserver un enregistrement de la traduction faite sur un paquet sortant pour réaliser l'opération inverse lors de l'arrivée du paquet retour relatif au paquet sortant.

Dans cette partie, on cherche à afficher la liste des enregistrements en cours dans le routeur *Hub* pour des flux initiés depuis un conteneur hébergé sur un routeur *Spoke*.

Q118. Comment caractériser l'utilisation des jeux de règles de filtrage en l'état actuel de la configuration ?

Rechercher dans les résultats de l'affichage des règles en cours sur le routeur *Hub*, les informations sur le comptage des flux.

La sortie de la commande d'affichage fait apparaître le compte des paquets traités par la chaîne `postrouting` avec l'indication *counter packets*.

```
sudo nft list ruleset
```

```
table inet nat {
        chain postrouting {
                type nat hook postrouting priority srcnat; policy accept;
                oifname "enp0s1.360" counter packets 89 bytes 7400 masquerade
        }
}
```

Q119. Est-ce que les valeurs de comptage affichées correspondent au volume de trafic vu sur l'interface ?

Rechercher, dans les options de la commande ip, le moyen d'afficher les statistiques de l'interface de sortie sur le réseau d'infrastructure (VLAN rouge). Comparer les valeurs obtenues avec celles de l'affichage des règles de filtrage.

C'est l'option `-s` de la commande ip qui donne les informations sur le volume de trafic qui a transité par une interface.

```
ip -s -h addr ls dev enp0s1.360
4: enp0s1.360@enp0s1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether b8:ad:ca:fe:00:05 brd ff:ff:ff:ff:ff:ff
    inet 192.168.104.130/29 brd 192.168.104.135 scope global enp0s1.360
       valid_lft forever preferred_lft forever
    inet6 2001:678:3fc:168:baad:caff:fefe:5/64 scope global dynamic mngtmpaddr noprefixroute
       valid_lft 2591974sec preferred_lft 604774sec
    inet6 2001:678:3fc:168::82/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::baad:caff:fefe:5/64 scope link proto kernel_ll
       valid_lft forever preferred_lft forever
    RX:  bytes packets errors dropped  missed   mcast
         227M   38.1k      0       0       0     137
    TX:  bytes packets errors dropped carrier collsns
         5.26M   71.8k      0       0       0       0
```

On constate que les valeurs ne coïncident pas du tout. En fait, les valeurs annoncées par le système de filtrage correspondent aux *nouveaux flux enregistrés* dans le système d'enregistrement et de suivi des transactions appelé *connection tracking*.

Pour que les valeurs affichées avec la commande nft évoluent, il faut que de nouveaux flux réseaux n'ayant pas encore été enregistrés apparaissent.

Q120. Comment obtenir la liste des enregistrements des transactions traitées par le système de filtrage réseau du routeur *Hub* ?

Rechercher le paquet qui contient la commande conntrack puis rechercher les options de cette commande qui permettent d'afficher les états des enregistrements.

On ouvre une console sur le routeur *Hub* de la maquette et on installe le paquet `conntrack`.

```
sudo apt -y install conntrack
```

Ensuite, on utilise l'option `-L` pour afficher la liste des enregistrements courants. Voici un exemple ponctuel formaté pour les besoins de la copie d'écran.

```
sudo conntrack -L | fmt -w80
conntrack v1.4.8 (conntrack-tools): 1 flow entries have been shown.
tcp      6 299 ESTABLISHED src=172.16.0.6 dst=192.168.104.130 sport=50376
dport=22 src=192.168.104.130 dst=172.16.0.6 sport=22 dport=50376 [ASSURED]
```

On identifie les paramètres suivants :

tcp

L'identification du protocole permet de savoir si on a affaire à un service orienté connexion ou pas.

Dans le cas d'un service orienté connexion, on utilise les trois phases d'établissement, de maintien et de libération pour savoir si l'enregistrement d'un flux doit être conservé.

À l'inverse, si le service n'est pas orienté connexion, on utilise des temporisations prédéfinies pour savoir si un enregistrement est périmé.

ESTABLISHEDASSURED

Ces indicateurs d'état nous informent sur le fait que la communication est dans la phase de maintien et que les échangent sont en cours.

sportdport

Les couples de numéros de ports source et destination en couche transport ainsi que les couples d'adresses IPv4 en couche réseau nous informent sur les permutations réalisées si le trafic réseau est sortant ou entrant.

Q121. Comment provoquer de nouveaux enregistrements IPv4 et IPv4 dans le système de suivi des transactions réseau ?

Pour que me scénario soit plus réaliste, on initie de nouvelles transactions depuis les conteneurs hébergés sur un routeur ***Spoke*** et on observe les résultats sur le routeur ***Hub***.

On commence par installer le paquet qui contient la commande wget sur chaque conteneur du routeur ***Spoke*** utilisé pour les tests.

```
for i in {0..2}
do
   echo ">>>>>>>>>>>>>>>> c$i"
   incus exec c$i -- apt -y install wget
done
```

Toujours sur le même routeur ***Spoke***, on lance une boucle de génération de trafic web. Dans l'exemple ci-dessous, on lance cinq consultations de page web dans chacun des trois conteneurs hébergés.

```
for i in {0..2}
do
   echo ">>>>>>>>>>>>>>>> c$i"
   for j in {0..4}
   do
      incus exec c$i -- wget -4 -O /dev/null https://www.iana.org
      sleep 1
   done
done
```

Sur le routeur ***Hub***, on affiche la liste des enregistrements du système de suivi des transactions. Voici un extrait IPv4 ponctuel non formaté pour afficher correctement les débuts de lignes.

```
sudo conntrack -f ipv4 -L
```

```
tcp      6 16 TIME_WAIT src=10.0.10.10 dst=192.0.46.8 sport=37766 dport=443 src=192.0.46.8 dst=192.168.104.130 sport=443 dp
tcp      6 21 TIME_WAIT src=10.0.10.10 dst=192.0.46.8 sport=49580 dport=443 src=192.0.46.8 dst=192.168.104.130 sport=443 dp
udp      17 24 src=10.0.10.10 dst=172.16.0.2 sport=35685 dport=53 src=172.16.0.2 dst=192.168.104.130 sport=53 dport=35685 m
tcp      6 34 TIME_WAIT src=10.0.10.11 dst=192.0.46.8 sport=56480 dport=443 src=192.0.46.8 dst=192.168.104.130 sport=443 dp
tcp      6 55 TIME_WAIT src=10.0.10.12 dst=192.0.46.8 sport=41652 dport=443 src=192.0.46.8 dst=192.168.104.130 sport=443 dp
tcp      6 45 TIME_WAIT src=10.0.10.12 dst=192.0.46.8 sport=55804 dport=443 src=192.0.46.8 dst=192.168.104.130 sport=443 dp
tcp      6 48 TIME_WAIT src=10.0.10.12 dst=192.0.46.8 sport=55820 dport=443 src=192.0.46.8 dst=192.168.104.130 sport=443 dp
```

Dans l'extrait ci-dessus on retrouve :

- Les adresses IPv4 des trois conteneurs : `10.0.10.10`, `10.0.10.11` et `10.0.10.12`.
- L'adresse du site web sur l'Internet : `192.0.46.8`
- L'adresse du routeur ***Hub*** sur le réseau d'infrastructure : `192.168.104.130`

On lance la même boucle en IPv6 pour afficher les enregistrements propres à ce protocole réseau.

```
for i in {0..2}
do
   echo ">>>>>>>>>>>>>>>> c$i"
   for j in {0..4}
   do
      incus exec c$i -- wget -6 -O /dev/null https://www.iana.org
      sleep 1
   done
done
```

Voici un nouvel extrait de la liste des enregistrements qui montre les trois adresses IPv6 des conteneurs, l'adresse IPv6 du serveur et enfin celle du routeur ***Hub***.

```
sudo conntrack -f ipv6 -L
```

```
sudo conntrack -f ipv6 -L
tcp      6 108 TIME_WAIT src=fda0:7a62:a:0:216:3eff:fe6a:395c dst=2620:0:2830:200::b:8 sport=56654 dport=443 src=2620:0:283
tcp      6 113 TIME_WAIT src=fda0:7a62:a:0:216:3eff:fe6a:395c dst=2620:0:2830:200::b:8 sport=56678 dport=443 src=2620:0:283
tcp      6 110 TIME_WAIT src=fda0:7a62:a:0:216:3eff:fe6a:395c dst=2620:0:2830:200::b:8 sport=56660 dport=443 src=2620:0:283
tcp      6 116 TIME_WAIT src=fda0:7a62:a:0:216:3eff:fe37:f12a dst=2620:0:2830:200::b:8 sport=36964 dport=443 src=2620:0:283
tcp      6 111 TIME_WAIT src=fda0:7a62:a:0:216:3eff:fe6a:395c dst=2620:0:2830:200::b:8 sport=56666 dport=443 src=2620:0:283
tcp      6 98 TIME_WAIT src=fda0:7a62:a:0:216:3eff:fed9:a25b dst=2620:0:2d0:200::b:8 sport=50388 dport=443 src=2620:0:2d0:2
tcp      6 115 TIME_WAIT src=fda0:7a62:a:0:216:3eff:fe6a:395c dst=2620:0:2830:200::b:8 sport=56684 dport=443 src=2620:0:283
tcp      6 96 TIME_WAIT src=fda0:7a62:a:0:216:3eff:fed9:a25b dst=2620:0:2d0:200::b:8 sport=50384 dport=443 src=2620:0:2d0:2
tcp      6 119 FIN_WAIT src=fda0:7a62:a:0:216:3eff:fe37:f12a dst=2620:0:2830:200::b:8 sport=36986 dport=443 src=2620:0:2830
tcp      6 118 TIME_WAIT src=fda0:7a62:a:0:216:3eff:fe37:f12a dst=2620:0:2830:200::b:8 sport=36978 dport=443 src=2620:0:283
```

3. Protection de base des routeurs Hub et Spoke

Du point de vue conception, cette partie est consacrée au filtrage réseau sans état. Les trois sections proposent des règles qui demandent un traitement le plus rapide possible et le moins coûteux possible en ressources.

Le but de cette partie est de mettre en place les fonctions de filtrage de base communes à tous les routeurs de la topologie. Ceci implique que le paquet `nftables` soit installé et que le service `systemd` soit activé sur tous routeurs *Spoke*

```
sudo apt -y install nftables
```

```
sudo systemctl enable --now nftables
```

Voici une description des fonctions à mettre en œuvre dans cette section.

Protection contre l'usurpation des adresses sources

Pour bloquer tous les paquets provenant d'un réseau ***extérieur*** avec des adresses IP sources appartenant à un réseau ***intérieur***, on implante une chaîne `rpfilter` dans la table `raw` qui assure un filtrage sans état.

Les motivations de ces méthodes de filtrage réseau sont présentées dans les documents de référence RFC2827 *Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing* et RFC 3704.

Les tests de validation de ces mécanismes sont faciles à réaliser sur les routeurs ***Spoke***. Tout paquet qui arrive via l'interface PPP et dont l'adresse source IPv4 ou IPv6 appartient au réseau d'hébergement (VLAN vert) doit être jeté. On protège ainsi les routeurs contre les dénis de services.

Protection contre les dénis de services ICMP

Les routeurs doivent s'assurer que le volume de trafic qui est présenté en entrée est compatible avec un fonctionnement nominal des services.

Protection contre les robots de connexion au service SSH

Les routeurs ont besoin d'un accès d'administration à distance via SSH. Pour autant, cet accès doit être protégé contre les tentatives d'intrusion par dictionnaire de couples d'authentifiants.

L'outil fail2ban fourni avec le paquet du même nom introduit une chaîne de filtrage dédiée à ces tentatives d'intrusion.

3.1. Protection contre l'usurpation d'adresse source

Voici le jeu de règles à implanter dans le fichier `/etc/nftables.conf` sur les deux routeurs ***Spoke*** pour activer la protection contre l'usurpation d'adresses réseau source.

```
#!/usr/sbin/nft -f

flush ruleset

table inet raw {
    chain rpfilter {
        type filter hook prerouting priority raw; policy accept;
        iifname "ppp0" fib saddr . iif oif 0 counter packets 0 bytes 0 drop
    }
}
```

> **Important**
>
> Dans le but de faciliter les tests de validation, le trafic “malveillant” est émis depuis le routeur ***Hub***. Sur ce routeur, nous avons la possibilité d'installer tous les outils et de faire toutes les manipulations possibles pour falsifier les adresses sources. De plus, on contrôle le plan d'adressage. Si ce n'était pas le cas, il faudrait utiliser une autre machine virtuelle et les erreurs auraient des conséquences plus beaucoup plus graves.

Q122. Comment définir le rôle de la table `raw` dans le système de filtrage du noyau Linux ?

Rechercher la présentation de cette table dans la documentation `netfilter`.

La table `raw` du système de filtrage Linux `netfilter` est principalement utilisée pour marquer les paquets qui doivent éviter le suivi des connexions. Elle intervient au tout début du traitement des paquets, ce qui permet de traiter certains flux réseau en évitant des opérations qui consomment beaucoup de ressources, comme le suivi des connexions.

Q123. Comment expliquer l'utilisation de la correspondance `fib` dans la règle de filtrage des paquets entrant par l'interface `ppp0` ?

Rechercher le mot clé `fib` dans les pages de manuels de la commande nft.

La règle `fib saddr . iif oif missing` vérifie si l'adresse source correspond à l'interface entrante selon la FIB (***Forwarding Information Base***). Cette règle met en œuvre le filtrage du chemin inverse.

Q124. Comment afficher la liste des règles de filtrage de la table `raw` dédiée au filtrage sans état (***stateless***) ?

Rechercher dans les pages de manuels de la commande nft les options relatives aux listes.

C'est l'option `list` qui permet l'affichage des règles implantées dans les différentes tables.

Voici un exemple dans le contexte de la maquette sur un routeur ***Spoke***. Un jeu de règles a déjà été inséré dans la table `raw`. Elle permet de visualiser les compteurs de correspondance qui montrent que la règle a bien été utilisée.

```
sudo nft list table inet raw
```

```
table inet raw {
    chain rpfilter {
        type filter hook prerouting priority raw; policy accept;
        iifname "ppp0" fib saddr . iif oif 0 counter packets 10 bytes 280 drop
    }
}
```

Avertissement

Si aucune règle n'a été implantée dans une table, le résultat de la commande produit une erreur.

Q125. Comment valider la fonction de blocage des tentatives d'usurpation d'adresses entre le routeur ***Hub*** et les routeurs ***Spoke*** ?

Pour falsifier les adresses réseau source, nous devons distinguer les deux protocoles.

- Pour IPv4, on installe le paquet `hping3`.

  Rechercher dans les pages de manuels de la commande hping3 les options qui permettent de générer du trafic ICMP avec des adresses source aléatoires à destination d'un conteneur hébergé sur un routeur ***Spoke***.

- Pour IPv6, on doit créer une interface réseau de type `dummy` à laquelle on attribue une adresse du réseau d'hébergement de conteneur.

  Rechercher les instructions de création d'une interface `dummy`.

On débute les tests avec le protocole IPv4 et la commande hping3.

Voici un exemple de test effectué sur le routeur ***Hub*** dans lequel l'option `-a` désigne l'adresse IPv4 source usurpée tandis que l'adresse en bout de ligne désigne la destination. Ici, on cherche à contacter un conteneur avec l'adresse source d'un conteneur voisin en étant placé “à l'extérieur” du VLAN vert.

```
sudo hping3 -1 -a 10.0.10.12 --fast -c 10 10.0.10.10
```

```
HPING 10.0.10.10 (ppp0 10.0.10.10): icmp mode set, 28 headers + 0 data bytes

--- 10.0.10.10 hping statistic ---
10 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

Côté routeur ***Hub***, on constate qu'aucune réponse n'a été reçue.

Côté routeur ***Spoke***, on affiche le jeu des règles de filtrage actif et on relève les valeurs des compteurs de paquets jetés.

```
sudo nft list table inet raw
```

```
table inet raw {
    chain rpfilter {
        type filter hook prerouting priority raw; policy accept;
        iifname "ppp0" fib saddr . iif oif 0 counter packets 20 bytes 560 drop
    }
}
```

Dans l'exemple ci-dessus, 20 paquets ont été jetés.

Pour le protocole IPv6, il n'existe pas de solution équivalente à l'utilisation de la commande hping3. De plus, le fait d'utiliser une session PPP pour acheminer le trafic ne facilite pas les tests.

C'est la raison pour laquelle on utilise un autre artifice : une interface de type `dummy`. Cette interface est factice et nous permet d'émettre des requêtes ICMP6 avec son adresse IPv6.

Voici comment créer une interface et lui ajouter une adresse. On commence par charger le module `dummy` en précisant que nous besoin d'une seule interface.

```
sudo modprobe -v dummy numdummies=1
```

```
sudo ip link set dev dummy0 up
```

On peut maintenant ajouter une adresse IPv6 appartenant au réseau d'hébergement situé au-delà du routeur ***Spoke*** à cette interface.

```
sudo ip -6 addr add fda0:7a62:a::e/64 dev dummy0
```

On modifie la table de routage pour s'assurer que les paquets partiront en direction de la session PPP.

```
sudo ip -6 route del fda0:7a62:a::/64 dev dummy0
sudo ip -6 route add fda0:7a62:a::/64 dev dummy0 metric 2048
```

On vérifie la solution d'acheminement du trafic à destination du réseau d'hébergement des conteneurs.

```
ip -6 route get fda0:7a62:a::a
fda0:7a62:a::a from :: dev ppp0 src fda0:7a62:a::e metric 1024 pref medium
```

Il ne reste plus qu'à lancer des requêtes ICMP6 avec cette adresse source.

```
sudo ping6 -c 10 fda0:7a62:a::a
```

```
PING fda0:7a62:a::a (fda0:7a62:a::a) from fda0:7a62:a::e dummy0: 56 data bytes

--- fda0:7a62:a::a ping statistics ---
10 packets transmitted, 0 received, 100% packet loss, time 9210ms
```

Là encore, aucune réponse n'est revenue et c'est heureux !

Si on relève le compte des paquets jetés côté routeur ***Spoke***, on voit que toutes les requêtes sont tombées dans la règle de la table `raw`.

```
sudo nft list table inet raw
```

```
table inet raw {
    chain rpfilter {
        type filter hook prerouting priority raw; policy accept;
        iifname "ppp0" fib saddr . iif oif 0 counter packets 40 bytes 4160 drop
    }
}
```

Attention

N'oubliez pas de supprimer l'interface `dummy` après les tests.

```
sudo ip link set dev dummy0 down
sudo modprobe -r dummy
```

Q126. Comment utiliser la fonction `rp_filter` du sous-système réseau du noyau Linux pour le protocole IPv4 ?

Rechercher la clé `rp_filter` dans la documentation du noyau Linux : *Kernel IP sysctl*

Il existe un réglage du sous-système réseau du noyau avec la clé `rp_filter`. Voici un extrait de la documentation du noyau.

```
rp_filter - INTEGER
0 - No source validation.
1 - Strict mode as defined in RFC3704 Strict Reverse Path
    Each incoming packet is tested against the FIB and if the interface
    is not the best reverse path the packet check will fail.
    By default failed packets are discarded.
2 - Loose mode as defined in RFC3704 Loose Reverse Path
    Each incoming packet's source address is also tested against the FIB
    and if the source address is not reachable via any interface
    the packet check will fail.

Current recommended practice in RFC3704 is to enable strict mode
to prevent IP spoofing from DDos attacks. If using asymmetric routing
or other complicated routing, then loose mode is recommended.

The max value from conf/{all,interface}/rp_filter is used
when doing source validation on the {interface}.

Default value is 0. Note that some distributions enable it
in startup scripts.
```

On peut fixer la clé à 1 et lancer un test avec une série d'adresses IPv4 source aléatoires.

```
sudo sysctl -w net.ipv4.conf.all.rp_filter=1
```

Voici un exemple de commande qui provoquera un nombre de blocages aléatoire en fonction des correspondances.

```
sudo hping3 -1 --rand-source --fast -c 100 10.0.10.10
```

Dans ce cas, c'est la fonction de protection du noyau qui a détecté des paquets “martiens” pour lesquels il n'existe aucune solution de routage.

```
journalctl -n 500 -f --grep martian
```

```
spoke1 kernel: IPv4: martian source 10.0.10.10 from 239.9.60.83, on dev ppp0
spoke1 kernel: IPv4: martian source 10.0.10.10 from 225.61.235.216, on dev ppp0
spoke1 kernel: IPv4: martian source 10.0.10.10 from 234.137.124.42, on dev ppp0
spoke1 kernel: IPv4: martian source 10.0.10.10 from 225.151.17.117, on dev ppp0
spoke1 kernel: IPv4: martian source 10.0.10.10 from 227.163.239.170, on dev ppp0
spoke1 kernel: IPv4: martian source 10.0.10.10 from 227.27.22.34, on dev ppp0
spoke1 kernel: IPv4: martian source 10.0.10.10 from 233.200.216.31, on dev ppp0
```

3.2. Protection contre les dénis de service ICMP

Dans cette section, on reprend le jeu de règles précédentes et on le complète avec la limitation du nombre de requêtes ICMP entrantes.

Le trafic “malveillant“ est toujours généré sur le routeur ***Hub*** à destination des conteneurs des réseaux d'hébergement des sites distants.

Voici une nouvelle copie du fichier `/etc/nftables.conf` qui contient le jeu de règles à ajouter sur les deux routeurs ***Spoke*** pour assurer la protection contre les dénis de service par saturation de requêtes ICMP.

```
#!/usr/sbin/nft -f

flush ruleset

table inet raw {
    # BCP38 Rules
    chain rpfilter {
        type filter hook prerouting priority raw; policy accept;
        iifname "ppp0" fib saddr . iif oif 0 counter packets 0 bytes 0 drop
    }

    # ICMP Rate Limiting Rules
    chain icmpfilter {
        type filter hook prerouting priority raw; policy accept;
        icmp type echo-request limit rate 10/second burst 5 packets counter accept
        icmp type echo-request counter drop
    }
}
```

Q127. À quelle chaîne prédéfinie de la table `raw` font appels les chaînes personnalisées appelées `icmpfilter` ?

Consulter la représentation graphique *Packet Flow in Netfilter* et repérer les chaînes prédéfinies de la table `raw`.

Il s'agit de la chaîne `prerouting` qui traite les flux réseau “au plus tôt”, avant qu'une décision de routage soit prise.

Q128. Comment qualifier le fonctionnement des règles de limitation du nombre de nouvelles requêtes ICMP avec IPv4 ?

Rechercher les options de la commande hping3 qui permettent de générer un envoi de requêtes ICMP en grand nombre.

Voici un exemple d'envoi de requêtes ICMP en nombre à destination du deuxième conteneur hébergé sur le premier routeur ***Spoke***.

```
sudo hping3 -1 --flood -c 10 10.0.10.11
```

Si le temps d'exécution de ces émission paraît trop long, il ne faut pas hésiter à interrompre l'émission avec Ctrl+C.

```
HPING 10.0.10.11 (ppp0 10.0.10.11): icmp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown

--- 10.0.10.11 hping statistic ---
61492309 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

Les résultats de cette commande montrent que le mécanisme de protection a bien fonctionné. On doit aussi vérifier qu'une émission “raisonnable“ de requêtes ICMP donne des résultats corrects.

```
ping -qc 10 10.0.10.11
```

```
PING 10.0.10.11 (10.0.10.11) 56(84) bytes of data.

--- 10.0.10.11 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9013ms
rtt min/avg/max/mdev = 1.056/1.279/2.034/0.268 ms
```

Côté routeur ***Spoke*** “cible”, on peut relever les compteurs des règles de traitement ICMP et constater qu'un grand nombre de paquets ont été jetés.

```
sudo nft list table inet raw
```

```
table inet raw {
    chain rpfilter {
        type filter hook prerouting priority raw; policy accept;
        iifname "ppp0" fib saddr . iif oif 0 counter packets 0 bytes 0 drop
    }

    chain icmpfilter {
        type filter hook prerouting priority raw; policy accept;
        icmp type echo-request limit rate 10/second burst 5 packets counter packets 5378 bytes 150584 accept
        icmp type echo-request counter packets 90460174 bytes 2532884872 drop
    }
}
```

Q129. Comment qualifier le fonctionnement des règles de limitation du nombre de nouvelles requêtes ICMP avec IPv6 ?

Rechercher les options de la commande ping qui permettent de générer un flux de saturation ICMPv6.

On s'intéresse plus particulièrement aux options `-f` et `-i`.

Voici un exemple de tentative de saturation à destination du deuxième conteneur du réseau d'hébergement du premier routeur ***Spoke***.

```
sudo ping -6 -c100 -i 0.0005 -f fda0:7a62:a::b
```

En réponse à cette commande, on voit que le taux de perte de paquets est important.

```
PING fda0:7a62:a::b (fda0:7a62:a::b) 56 data bytes
.....................................................................
--- fda0:7a62:a::b ping statistics ---
100 packets transmitted, 18 received, 82% packet loss, time 1315ms
rtt min/avg/max/mdev = 0.316/0.845/2.057/0.334 ms, ipg/ewma 13.281/0.931 ms
```

Côte routeur “cible”, on relève à nouveau un grand nombre de paquets jetés.

```
sudo nft list table inet raw
```

```
table inet raw {
        chain rpfilter {
                type filter hook prerouting priority raw; policy accept;
                iifname "ppp0" fib saddr . iif oif 0 counter packets 0 bytes 0 drop
        }

        chain icmpfilter {
                type filter hook prerouting priority raw; policy accept;
                icmpv6 type echo-request limit rate 10/second burst 5 packets counter packets 93 bytes 9672 accept
                icmpv6 type echo-request counter packets 485 bytes 50440 drop
        }
}
```

Comme on l'a fait pour le protocole IPv4, on vérifie qu'on obtient un retour correct suite à des requêtes émises “à un rythme normal”.

```
ping -qc10 fda0:7a62:a::b
```

```
PING fda0:7a62:a::b (fda0:7a62:a::b) 56 data bytes

--- fda0:7a62:a::b ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9018ms
rtt min/avg/max/mdev = 0.839/1.055/1.191/0.097 ms
```

3.3. Protection contre les robots de connexion au service SSH

Comme dans les deux sections précédentes, l'évaluation du mécanisme de protection se joue entre le routeur *Hub* et un routeur *Spoke*. On verra que le cas d'une protection contre les robots SSH, celle-ci doit s'appliquer sur tous les systèmes qui autorisent une connexion par mot de passe.

Le sujet de ce document étant l'étude du filtrage réseau, on s'intéresse ici à la génération automatique de règles suite à la violation des critères définis dans l'application de gestion des tentatives de connexion : fail2ban.

Q130. Quel est la fonction de *fail2ban* ?

Afficher la description du paquet *fail2ban* après l'avoir installé.

```
sudo apt -y install fail2ban
```

```
apt show fail2ban | grep  Description
```

```
Description : ban hosts that cause multiple authentication errors Fail2ban
```

Fail2ban est un outil de sécurité qui analyse les journaux système pour détecter et empêcher les attaques par force brute sur les serveurs. Il bloque automatiquement les adresses IP suspectes après plusieurs tentatives de connexion infructueuses au cours d'une période donnée en mettant à jour les règles du pare-feu. Cette application prend en charge différents services, dont SSH, et offre des paramètres personnalisables pour améliorer la protection des serveurs contre les attaques automatisées.

Q131. Quel est le numéro de port utilisé par le service SSH sur les routeurs ?

Il est important de connaître les caractéristiques du service qui doit être surveillé par fail2ban. Rechercher dans la liste des ports réseau ouverts celui qui concerne le service SSH.

Dans le contexte de la maquette, le service SSH a été paramétré pour utiliser le port numéro 2222. On obtient la liste des ports en écoute avec les commandes lsof ou ss.

```
sudo lsof -i tcp:2222 -sTCP:listen
```

```
COMMAND PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
sshd    672 root    7u  IPv4   7601      0t0  TCP *:2222 (LISTEN)
sshd    672 root    8u  IPv6   7603      0t0  TCP *:2222 (LISTEN)
```

```
ss -tapl '( sport = :2222 )' | fmt -t -w80
```

```
State  Recv-Q Send-Q Local Address:Port Peer Address:PortProcess
LISTEN 0      128          0.0.0.0:2222      0.0.0.0:*
LISTEN 0      128             [::]:2222         [::]:*
```

Ce sont donc les tentatives de connexion au service SSH sur le port numéro ***2222*** que le service fail2ban doit surveiller.

Q132. Comment créer un fichier de configuration des paramètres de gestion du service SSH ?

Rechercher dans la documentation de fail2ban les paramètres de réglage suivants.

- Le numéro de port
- Le service à filtrer
- La règle à appliquer en cas de violation du nombre de tentatives de connexion autorisées
- Le nombre de tentatives de connexion autorisées
- La durée de mise en quarantaine de l'adresse IPv4 ou IPv6
- La durée pendant laquelle on comptabilise les tentatives de connexion

Pour traiter la demande, on créé un fichier spécifique dans le répertoire dédié : `/etc/fail2ban/jail.d/custom-sshd.conf` qui contient tous les paramètres utiles.

```
cat << EOF | sudo tee /etc/fail2ban/jail.d/custom-sshd.conf
[sshd]
enabled = true
port = 2222
filter = sshd
backend = systemd
banaction = nftables-multiport
maxretry = 3
bantime = 1h
findtime = 10m
EOF
```

De manière classique, il convient de redémarrer le service pour que le nouveau fichier de configuration soit pris en compte.

```
sudo systemctl restart fail2ban
```

Q133. Quels sont les outils qui permettent de connaître l'état du service et le fonctionnement du nouveau filtrage des tentatives de connexion ?

Relever l'état du service pour commencer. Ensuite, rechercher dans les outils fournis avec le paquet fail2ban, celui qui affiche les informations sur le filtrage SSH.

L'affichage de l'état du service n'a rien de spécifique à fail2ban, mais c'est le point de départ obligatoire.

```
systemctl status fail2ban.service
```

Les mots clé recherchés sont : ***enabled*** et ***active (running)***.

Pour la partie plus spécifique à fail2ban, on liste les fichiers contenus dans le paquet du même nom.

```
dpkg -L fail2ban | grep bin
```

```
/usr/bin
/usr/bin/fail2ban-client
/usr/bin/fail2ban-regex
/usr/bin/fail2ban-server
/usr/bin/fail2ban-testcases
/usr/bin/fail2ban-python
```

C'est la commande fail2ban-client qui nous intéresse. La recherche du mot clé `status` dans les pages de manuels donne le mode opératoire pour accéder au statut des services traités.

```
sudo fail2ban-client status
```

```
Status
|- Number of jail:      1
`- Jail list:   sshd
```

```
sudo fail2ban-client status sshd
```

```
Status for the jail: sshd
|- Filter
|  |- Currently failed: 0
|  |- Total failed:     0
|  `- Journal matches:  _SYSTEMD_UNIT=ssh.service + _COMM=sshd
`- Actions
   |- Currently banned: 0
   |- Total banned:     0
   `- Banned IP list:
```

Q134. Comment caractériser le fonctionnement du service fail2ban ?

Si le service a été installé et configuré sur un routeur ***Spoke***, il est possible de lancer plusieurs tentatives de connexion SSH depuis le routeur ***Hub*** en se trompant de mot de passe.

On peut alors afficher les règles de filtrage nftables et obtenir la liste des adresses bannies par fail2ban.

On commence par lancer plusieurs tentatives (au moins 3) de connexion SSH à partir du routeur ***Hub***.

```
ssh -p 2222 etu@10.44.1.2
etu@10.44.1.2's password:
Permission denied, please try again.
etu@10.44.1.2's password:
Permission denied, please try again.
etu@10.44.1.2's password:
etu@10.44.1.2: Permission denied (publickey,password)
```

Une fois le mécanisme de blocage activé, il est impossible de tenter une nouvelle connexion.

```
ssh -p 2222 etu@10.44.1.2
ssh: connect to host 10.44.1.2 port 2222: Connection refused
```

On relève ensuite les résultats côté routeur ***Spoke***.

L'état de la quarantaine montre que l'adresse IPv4 du routeur ***Hub*** sur le lien PPP est en quarantaine.

```
sudo fail2ban-client status sshd
```

```
Status for the jail: sshd
|- Filter
|  |- Currently failed: 0
|  |- Total failed:     3
|  `- Journal matches:  _SYSTEMD_UNIT=ssh.service + _COMM=sshd
`- Actions
   |- Currently banned: 1
   |- Total banned:     1
   `- Banned IP list:   10.44.1.1
```

La liste des règles de filtrage montre qu'une nouvelle chaîne a été ajoutée. Dans cette chaîne, on reconnaît l'adresse IPv4 du lien PPP côté ***Hub***.

```
sudo nft list ruleset
```

En limitant l'affichage à la table ajoutée par le service fail2ban, on obtient le jeu de règles suivant.

```
sudo nft list table inet f2b-table
```

```
table inet f2b-table {
    set addr-set-sshd {
        type ipv4_addr
        elements = { 10.44.1.1 }
    }

    chain f2b-chain {
        type filter hook input priority filter - 1; policy accept;
        tcp dport 2222 ip saddr @addr-set-sshd reject with icmp port-unreachable
    }
}
```

Enfin, on répète l'opération avec l'adresse IPv6 du routeur ***Spoke*** sur le lien PPP.

```
ssh -p 2222 etu@fe80::c59:5d57:7476:13cc%ppp0
etu@fe80::c59:5d57:7476:13cc%ppp0's password:
Permission denied, please try again.
etu@fe80::c59:5d57:7476:13cc%ppp0's password:
Permission denied, please try again.
etu@fe80::c59:5d57:7476:13cc%ppp0's password:
etu@fe80::c59:5d57:7476:13cc%ppp0: Permission denied (publickey,password).
```

```
ssh -p 2222 etu@fe80::c59:5d57:7476:13cc%ppp0
ssh: connect to host fe80::c59:5d57:7476:13cc%ppp0 port 2222: Connection refused
```

On voit apparaître une nouvelle adresse dans la liste sur le routeur ***Spoke***.

```
sudo fail2ban-client status sshd
```

```
Status for the jail: sshd
|- Filter
|  |- Currently failed: 0
|  |- Total failed:     6
|  `- Journal matches:  _SYSTEMD_UNIT=ssh.service + _COMM=sshd
`- Actions
   |- Currently banned: 2
   |- Total banned:     2
   `- Banned IP list:   10.44.1.1 fe80::f153:4881:c7c4:f371
```

Les règles de filtrage pour le protocole IPv6 ont aussi été complétées.

```
sudo nft list table inet f2b-table
```

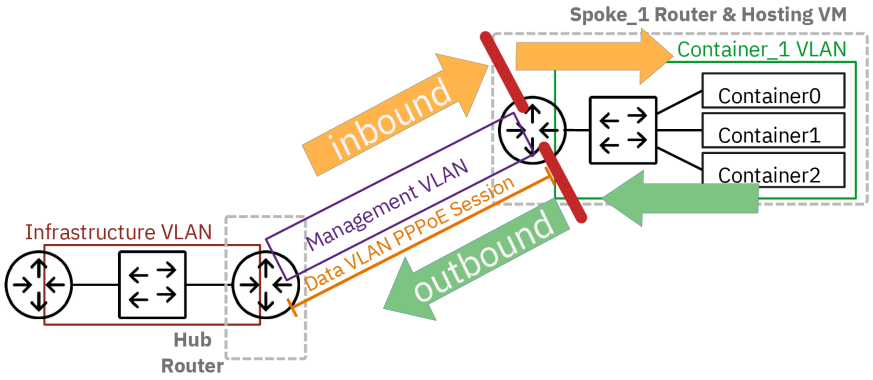
```
table inet f2b-table {
    set addr-set-sshd {
        type ipv4_addr
        elements = { 10.44.1.1 }
    }

    set addr6-set-sshd {
        type ipv6_addr
        elements = { fe80::f153:4881:c7c4:f371 }
    }

    chain f2b-chain {
        type filter hook input priority filter - 1; policy accept;
        tcp dport 2222 ip saddr @addr-set-sshd reject with icmp port-unreachable
        tcp dport 2222 ip6 saddr @addr6-set-sshd reject with icmpv6 port-unreachable
    }
}
```

En appliquant cette configuration du service fail2ban sur les trois routeurs, on dispose d'une base de protection contre les attaques de type ***brute force*** sur l'authentification au service SSH.

4. Filtrage des flux réseaux traversant les routeurs Spoke

Pour rappel, la mise en place du filtrage réseau sur les équipements doit répondre à deux principes.

- On considère que les équipements d'interconnexion mis en œuvre dans ces travaux pratiques délimitent des périmètres de dimension moyenne. Par conséquent, on a une connaissance exhaustive des flux réseaux sur le système. On adopte donc la règle suivante : ***tout trafic réseau non autorisé est interdit***.
- Le filtrage est basé sur le suivi de communication (***stateful inspection***). On cherche donc à écrire des règles qui ***décrivent le plus précisément possible le premier paquet qui doit être enregistré dans la table de suivi de communication***. Ces règles de description du premier paquet doivent être placées après celles qui laissent passer le trafic correspondant ou relatif à une communication déjà enregistrée dans les tables.
- Afin de simplifier l'étude du filtrage, on fait le choix d'autoriser tous les flux sortants émis par les routeurs *Hub* et *Spoke*.



Caractérisation des flux réseau traversant un routeur *Spoke* avec identification du sens entrant ou sortant

On commence par afficher les règles actives sur un routeur *Spoke* à l'issue des questions de la section précédente : Section 3, « Protection de base des routeurs Hub et Spoke ».

> ⚠ Attention
>
> Les noms d'interfaces correspondent à la maquette de test.

```
sudo nft list ruleset
```

```
table inet raw {
        chain rpfilter {
                type filter hook prerouting priority raw; policy accept;
                iifname "ppp0" fib saddr . iif oif 0 counter packets 0 bytes 0 drop
        }

        chain icmpfilter {
                type filter hook prerouting priority raw; policy accept;
                icmp type echo-request limit rate 10/second burst 5 packets counter packets 6821 bytes 193228 accept
                icmp type echo-request counter packets 116414622 bytes 3259609416 drop
        }
}
table inet f2b-table {
        set addr-set-sshd {
                type ipv4_addr
                elements = { 10.44.1.1 }
        }

        set addr6-set-sshd {
                type ipv6_addr
                elements = { fe80::f153:4881:c7c4:f371 }
        }

        chain f2b-chain {
                type filter hook input priority filter - 1; policy accept;
                tcp dport 2222 ip saddr @addr-set-sshd reject with icmp port-unreachable
                tcp dport 2222 ip6 saddr @addr6-set-sshd reject with icmpv6 port-unreachable
        }
}
```

Q135. Quel est le nom de la table du système `netfilter` utilisée par défaut pour le traitement des flux traversant un routeur ?

Consulter la représentation graphique *Packet Flow in Netfilter* et repérer le nom des tables concernées par le transfert des flux réseau.

On commence par identifier la colonne ***FORWARD PATH*** au centre de la représentation.

Dans cette colonne, on trouve les tables `mangle` et `filter`.

Une recherche sur les différences entre ces deux tables nous donne les indications suivantes.

Table filter

Cette table est principalement utilisée pour filtrer les paquets et prendre des décisions concernant l'autorisation ou le blocage du trafic. Elle sert à contrôler quels paquets sont autorisés à entrer dans le système, à le quitter ou à le traverser.

Table mangle

La table d'altération est utilisée pour la modification de paquets. Les altérations principales portent sur les en-têtes, les métadonnées et le marquage, pour permettre un traitement ultérieur par d'autres tables ou les outils de gestion de la qualité de service.

Q136. Quel est le nom de la ***chaîne*** de traitement des flux traversant les routeurs ?

Consulter la représentation graphique *Packet Flow in Netfilter* et repérer les noms des chaînes de la table `filter`.

Comme on se concentre sur la colonne ***FORWARD PATH*** de la représentation graphique, on repère facilement la chaîne `forward`.

Voici une proposition de jeu de règles de filtrage à implanter dans la chaîne `forward` de la table `filter` des routeurs *Spoke*.

```
table inet filter {
    chain forward {
        type filter hook forward priority 0; policy drop;

        # Allow outbound new connections
        oifname "ppp0" ct state new counter accept

        # Allow established and related connections
        ct state established,related accept

        # Allow specific inbound traffic
        # ICMP IPv4 + IPv6
        iifname "ppp0" meta l4proto {icmp, ipv6-icmp} ct state new counter accept
        # SSH
        iifname "ppp0" tcp dport 2222 ct state new counter accept
        # HTTP(S)
        iifname "ppp0" meta l4proto {tcp, udp} th dport {80, 443} ct state new counter accept

        # Count dropped packets
        counter comment "count dropped packets"
    }
}
```

Q137. Quelle est la politique par défaut appliquée par ce nouveau jeu de règles?

Rechercher le mot clé ***policy*** dans la copie ci-dessus.

La politique appliquée à tous les paquets qui passent par la chaîne `forward` est `drop`.

Tous les flux qui ne sont pas explicitement autorisés dans les règles de la chaîne `forward` sont jetés.

Q138. Comment identifier le sens du flux qui traverse le routeur dans le jeu de règles proposées ?

Rechercher les mots clé associés à l'interface réseau qui raccorde le routeur ***Spoke*** au ***Hub***

Dans la topologie étudiée, c'est l'interface `ppp0` du site distant qui donne accès à tous les autres réseaux. Dans les règles proposées, cette interface est précédées des clés `iifname` ou `oifname`.

iifname

La clé se lit ***Input InterFace NAME*** et désigne les flux entrants par l'interface `ppp0`. Vue du routeur, on emploie l'expression ***inbound traffic*** en anglais.

oifname

La clé se lit ***Output InterFace NAME*** et désigne les flux sortants par l'interface `ppp0`. Vue du routeur, on emploie l'expression ***outbound traffic*** en anglais.

Q139. Comment est géré l'enregistrement des états de flux sortants ?

Rechercher les règles dans lesquelles les états `new`, `established` et `related` apparaissent.

Dans le cas des flux sortants, on voit que tous les nouveaux paquets sont autorisés à passer par l'interface `ppp0` en sortie. La règle associe l'état `new` à l'autorisation `accept`.

```
oifname "ppp0" ct state new counter accept
```

Q140. Comment est géré l'enregistrement des états de flux entrants ?

Rechercher à nouveau les règles dans lesquelles l'état `new` est associé à la clé `iifname`.

Dans le cas des flux entrants, on voit qu'une liste fermée de protocoles sont acceptés. Tous les nouveaux flux qui ne correspondent pas à l'une des règles avec la décision `accept` sont jetés.

Prenons l'exemple du protocole HTTP, une règle autorise l'accès aux ports `80` et `443`.

```
iifname "ppp0" meta l4proto {tcp, udp} th dport {80, 443} ct state new counter accept
```

Q141. Comment sont gérés les flux déjà admis dans le système d'enregistrement ?

Rechercher la règle qui contient les états `established` et `related`.

Cette question touche au cœur du filtrage ***stateful***. Tout flux retour relatif à une première transaction admise ou enregistrée est accepté. C'est la raison pour laquelle on doit insister encore et encore sur le fait qu'il faut décrire le premier paquet admis dans le système d'enregistrement de la façon la plus exhaustive possible.

Voici la règle qui laisse passer tous les flux relatifs à un enregistrement déjà effectué.

```
ct state established,related accept
```

Q142. Comment appliquer le jeu de règles proposé sur les routeurs ***Spoke*** ?

On dispose de plusieurs réponse à cette question. Dans le but de rendre l'application des règles de filtrage pérenne, on complète le fichier `/etc/nftables.conf` et on relance le service.

On a une fois de plus recours au ***heredoc***.

```
cat << 'EOF' | sudo tee -a /etc/nftables.conf

table inet filter {
    chain forward {
        type filter hook forward priority 0; policy drop;

        # Allow outbound new connections
        oifname "ppp0" ct state new counter accept

        # Allow established and related connections
        ct state established,related accept

        # Allow specific inbound traffic
        # ICMP IPv4 + IPv6
        iifname "ppp0" meta l4proto {icmp, ipv6-icmp} ct state new counter accept
        # SSH
        iifname "ppp0" tcp dport 2222 ct state new counter accept
        # HTTP(S)
        iifname "ppp0" meta l4proto {tcp, udp} th dport {80, 443} ct state new counter accept

        # Count dropped packets
        counter comment "count dropped packets"
    }
}
EOF
```

Ensuite, on relance le service et on vérifie que le jeu de règles actives est complet.

```
sudo systemctl restart nftables.service
```

```
sudo nft list ruleset
```

Q143. Comment tester les règles relatives aux flux sortants ?

Pour tester la chaîne `forward` dans le sens réseau d'hébergement vers le routeur ***Hub***, il faut initier du trafic depuis les conteneurs.

Voici un exemple qui passe par la couche application : la mise à jour du catalogue de paquets de chaque conteneur.

```
for i in {0..2}
do
    echo ">>>>>>>>>>>>>>>> c$i"
    incus exec c$i -- apt update
done
```

Une fois la mise à jour effectuée dans les conteneurs, on peut relever les compteurs associés aux règles de filtrage.

```
sudo nft list table inet filter
```

```
table inet filter {
        chain forward {
                type filter hook forward priority filter; policy drop;
                oifname "ppp0" ct state new counter packets 41 bytes 3078 accept
                ct state established,related accept
                iifname "ppp0" meta l4proto { icmp, ipv6-icmp } ct state new counter packets 2 bytes 188 accept
                iifname "ppp0" tcp dport 2222 ct state new counter packets 0 bytes 0 accept
                iifname "ppp0" meta l4proto { tcp, udp } th dport { 80, 443 } ct state new counter packets 1 bytes 60 accep
                counter packets 0 bytes 0 comment "count dropped packets"
        }
}
```

Q144. Comment tester les règles relatives aux flux entrants ?

Il suffit de reprendre les séquences de tests présentées à la fin du support de travaux pratiques précédent : *Topologie Hub & Spoke avec le protocole PPPoE* à la question 21.

On reprend l'accès aux pages Web depuis le routeur ***Hub***.

```
for addr in {10..12}
do
    wget -O /dev/null http://10.0.10.$addr 2>&1 | grep "HTTP "
done
```

```
for addr in {10..12}
do
    wget -O /dev/null http://[fda0:7a62:a::$(printf "%x" $addr)] 2>&1 | grep "HTTP "
done
```

Ensuite, on vérifie la nouvelle valeur de comptage sur la règle de filtrage concernée.

```
sudo nft list table inet filter
```

```
table inet filter {
        chain forward {
                type filter hook forward priority filter; policy drop;
                oifname "ppp0" ct state new counter packets 0 bytes 0 accept
                ct state established,related counter packets 81 bytes 13581 accept
                iifname "ppp0" meta l4proto { icmp, ipv6-icmp } ct state new counter packets 0 bytes 0 accept
                iifname "ppp0" tcp dport 2222 ct state new counter packets 0 bytes 0 accept
                iifname "ppp0" meta l4proto { tcp, udp } th dport { 80, 443 } ct state new counter packets 9 bytes 600 acce
                counter packets 0 bytes 0 comment "count dropped packets"
        }
}
```

Une fois ces règles basiques en place, on peut aborder les filtrages réseau spécifiques à la topologie de travaux pratiques.

5. Traduction d'adresses destination sur le routeur Hub

Dans cette partie, on doit compléter les règles de filtrage sur le routeur ***Hub*** en appliquant la même politique que sur les routeurs ***Spoke*** pour les flux traversants : tout ce qui n'est pas autorisé est interdit. En complément à cette politique par défaut, on doit aussi répondre à deux objectifs :

- Autoriser le trafic provenant des routeurs ***Spoke*** vers l'Internet.
- Autoriser l'accès aux services Web hébergés sur les conteneurs des réseaux d'hébergement des sites distants à l'aide de la traduction des adresses destination.

  Voici un exemple de correspondances de numéros de ports pour l'accès aux différents services web.

Tableau 1. Correspondance entre numéro de port et service Web

numéros de port Hub : http,https	conteneur
8010,8453	`10.0.10.10` `fda0:7a62:a::a`
8011,8454	`10.0.10.11` `fda0:7a62:a::b`
8012,8455	`10.0.10.12` `fda0:7a62:a::c`
8020,8463	`10.0.20.10`

numéros de port Hub : http,https	conteneur
	fda0:7a62:14::a
8021,8464	10.0.20.11 fda0:7a62:14::b
8022,8465	10.0.20.12 fda0:7a62:14::c

Avant d'aborder les questions, on commence par afficher l'état courant du jeu de règles de filtrage sur le routeur *Hub*.

```
#!/usr/sbin/nft -f

flush ruleset

# Outbound interface
define RED_VLAN = enp0s1.360

table inet nat {
    chain postrouting {
        type nat hook postrouting priority 100;
        oifname $RED_VLAN counter masquerade
    }
}

table inet raw {
    chain rpfilter {
        type filter hook prerouting priority raw; policy accept;
        iifname $RED_VLAN fib saddr . iif oif 0 counter packets 0 bytes 0 drop
    }

    # ICMP Rate Limiting Rules
    chain icmpfilter {
        type filter hook prerouting priority raw; policy accept;
        icmp type echo-request limit rate 10/second burst 5 packets counter accept
        icmp type echo-request counter drop
    }
}

table inet filter {

    chain forward {
        type filter hook forward priority 0; policy drop;

        # Allow outbound new connections
        iifname "ppp*" ct state new counter accept

        # Allow established and related connections
        ct state established,related counter accept

        # Count dropped packets
        counter comment "count dropped packets"
    }
}
```

Q145. Comment appliquer la politique par défaut, autoriser et enregistrer dans le mécanisme de suivi des états les flux traversants depuis les routeurs *Spoke* ?

Reprendre le jeu de règles de la Section 4, « Filtrage des flux réseaux traversant les routeurs Spoke » et adapter ce jeu au contexte du routeur *Hub*.

Il faut notamment choisir sur quelle(s) interface(s) la règle doit s'appliquer pour ne pas bloquer les flux entre routeurs *Spoke*.

Voici un exemple de règles basées sur l'utilisation des interfaces de raccordement des routeurs *Spoke*.

```
table inet filter {
    chain forward {
        type filter hook forward priority filter; policy drop;
        iifname "ppp*" ct state new counter packets 6 bytes 504 accept
        ct state established,related counter packets 18 bytes 1512 accept
        counter packets 0 bytes 0 comment "count dropped packets"
    }
}
```

Dans ce cas de figure le trafic entrant par les interfaces PPP est autorisé à traverser le routeur *Hub*.

Q146. Comment valider l'utilisation de ces nouvelles règles à partir d'un routeur *Spoke* ?

Il suffit de lancer une mise à jour de catalogue de paquets dans les conteneurs hébergés sur un routeur *Spoke*.

On peut aussi utiliser la commande conntrack pour afficher la liste des enregistrements en cours sur le routeur ***Hub*** pendant les transactions des conteneurs.

Voici un exemple de trace qui montre que les flux initiés par les conteneurs sont correctement acheminés par le routeur ***Hub***.

- Côté routeur ***Spoke***, on initie le trafic.

```
for i in {0..2}
do
    echo ">>>>>>>>>>>>>>>>> c$i"
    incus exec c$i -- apt update
done
```

- Côté routeur ***Hub***, on relève “rapidement” la liste des enregistrements relatif à une adresse de conteneur.

```
sudo conntrack -L -j 10.0.10.1
```

```
tcp      6 36 TIME_WAIT src=10.0.10.10 dst=151.101.2.132 sport=39664 dport=80
    src=151.101.2.132 dst=192.168.104.130 sport=80 dport=39664 [ASSURED] mark=0 use=1
```

Q147. Comment implanter les règles de traduction d'adresses IPv4 et IPv6 destination pour ouvrir l'accès aux services Web placés dans les conteneurs des réseaux d'hébergement des routeurs ***Spoke*** ?

Rechercher la chaîne présente par défaut à utiliser pour les traitements de traduction d'adresses et de ports destination dans la représentation graphique *Packet Flow in Netfilter*.

Une fois la chaîne identifiée, rechercher des exemples de règles de filtrage de type `dnat` et une adaptation au critères de redirection de ports définis dans le tableau ci-dessus.

Bien sûr, les adresses IPv4 et IPv6 des conteneurs hébergés sur les routeur ***Spoke*** doivent être modifiées en fonction du plan d'adressage du document *Topologie Hub & Spoke avec le protocole PPPoE*.

Le nom de chaîne recherchée est `prerouting`. En effet, la traduction d'adresse destination doit se faire avant la décision de routage des paquets IPv4 ou IPv6.

Voici un extrait du fichier `/etc/nftables.conf` qui ne comprend que les règles définies dans la table `nat`.

```
table inet nat {
    chain postrouting {
        type nat hook postrouting priority 100;
        oifname $RED_VLAN counter masquerade
    }

    chain prerouting {
        type nat hook prerouting priority dstnat; policy accept;
        # Spoke1
        iifname $RED_VLAN meta l4proto { tcp, udp } th dport { 8010, 8453 } \
            dnat ip to 10.0.10.10
        iifname $RED_VLAN meta l4proto { tcp, udp } th dport { 8010, 8453 } \
            dnat ip6 to fda0:7a62:a::a
        iifname $RED_VLAN meta l4proto { tcp, udp } th dport { 8011, 8454 } \
            dnat ip to 10.0.10.11
        iifname $RED_VLAN meta l4proto { tcp, udp } th dport { 8011, 8454 } \
            dnat ip6 to fda0:7a62:a::b
        iifname $RED_VLAN meta l4proto { tcp, udp } th dport { 8012, 8455 } \
            dnat ip to 10.0.10.12
        iifname $RED_VLAN meta l4proto { tcp, udp } th dport { 8012, 8455 } \
            dnat ip6 to fda0:7a62:a::c
        # Spoke2
        iifname $RED_VLAN meta l4proto { tcp, udp } th dport { 8020, 8463 } \
            dnat ip to 10.0.20.10
        iifname $RED_VLAN meta l4proto { tcp, udp } th dport { 8020, 8463 } \
            dnat ip6 to fda0:7a62:14::a
        iifname $RED_VLAN meta l4proto { tcp, udp } th dport { 8021, 8464 } \
            dnat ip to 10.0.20.11
        iifname $RED_VLAN meta l4proto { tcp, udp } th dport { 8021, 8464 } \
            dnat ip6 to fda0:7a62:14::b
        iifname $RED_VLAN meta l4proto { tcp, udp } th dport { 8022, 8465 } \
            dnat ip to 10.0.20.12
        iifname $RED_VLAN meta l4proto { tcp, udp } th dport { 8022, 8465 } \
            dnat ip6 to fda0:7a62:14::c
    }
}
```

Relativement à l'unique règle de la chaîne `postrouting`, il est nécessaire d'ajouter deux règles par destination en distinguant les protocoles IPv4 et IPv6. Dans le contexte de la maquette, il y a trois conteneurs hébergés sur chaque routeur ***Spoke***. On doit donc ajouter 6 règles pour chacun d'eux.

Les règles de traduction d'adresses de la chaîne `prerouting` sont structurées de la façon suivante :

- Une même règle traite les protocoles de couche transport TCP et UDP.

- Les numéros de ports sont aussi regroupés. Par exemple : { 8010, 8453 }.
- Relativement aux règles des autres parties du document, les protocoles IPv4 et IPv6 sont séparés et les adresses destination sont explicitement définies pour chaque routeur ***Spoke***.

Q148. Comment tester le ces nouvelles règles de traduction d'adresse destination sur le routeur ***Hub*** ?

Comme il s'agit de flux réseau entrants sur le routeur ***Hub***, il faut nécessairement lancer les tests depuis le réseau d'infrastructure, c'est-à-dire le VLAN rouge.

La connexion au Shell de l'hyperviseur nous permet de lancer nos tests à destination de l'adresse du routeur ***Hub*** dans le réseau d'infrastructure, c'est-à-dire le VLAN rouge. Voici un exemple avec l'adresse définie dans le plan d'adressage de la maquette utilisée pour ce document.

```
wget http://192.168.104.130:8010
--2024-10-13 15:47:25--  http://192.168.104.130:8010/
Connexion à 192.168.104.130:8010…
```

Si l'adresse IPv4 et le numéro de port utilisés pour ce test sont corrects, cela ne suffit toutefois pas.

Q149. Comment autoriser les flux de la traduction d'adresse destination à traverser le routeur ***Hub*** ?

Rechercher à nouveau le nom de la chaîne présente par défaut qui traite les flux réseau qui traversent le routeur ***Hub***.

Une fois la chaîner identifiée, rechercher la syntaxe des règles qui admettent les premiers paquets de traitement de la traduction d'adresse destination dans le mécanisme de suivi des communications du système de filtrage.

C'est la chaîne `forward` qui nous intéresse ici. On a déjà créé des règles dans cette chaîne qui traitent les flux sortants du routeur ***Hub*** provenant des routeurs ***Spoke***.

Voici un exemple de jeu de règles pour la chaîne `forward` adapté au plan d'adressage de la maquette.

```
table inet filter {
    chain forward {
        type filter hook forward priority 0; policy drop;

        # Allow outbound new connections
        iifname "ppp*" ct state new counter accept

        # Allow inbound new connections to the web services hosted on spoke routers
        # Spoke1
        iifname $RED_VLAN ip daddr 10.0.10.10 meta l4proto { tcp, udp } \
            th dport { 8010, 8453 } ct state new counter accept
        iifname $RED_VLAN ip6 daddr fda0:7a62:a::a meta l4proto { tcp, udp } \
            th dport { 8010, 8453 } ct state new counter accept
        iifname $RED_VLAN ip daddr 10.0.10.11 meta l4proto { tcp, udp } \
            th dport { 8011, 8454 } ct state new counter accept
        iifname $RED_VLAN ip6 daddr fda0:7a62:a::b meta l4proto { tcp, udp } \
            th dport { 8011, 8454 } ct state new counter accept
        iifname $RED_VLAN ip daddr 10.0.10.12 meta l4proto { tcp, udp } \
            th dport { 8012, 8455 } ct state new counter accept
        iifname $RED_VLAN ip6 daddr fda0:7a62:a::c meta l4proto { tcp, udp } \
            th dport { 8012, 8455 } ct state new counter accept
        # Spoke2
        iifname $RED_VLAN ip daddr 10.0.20.10 meta l4proto { tcp, udp } \
            th dport { 8020, 8463 } ct state new counter accept
        iifname $RED_VLAN ip6 daddr fda0:7a62:14::a meta l4proto { tcp, udp } \
            th dport { 8020, 8463 } ct state new counter accept
        iifname $RED_VLAN ip daddr 10.0.20.11 meta l4proto { tcp, udp } \
            th dport { 8021, 8464 } ct state new counter accept
        iifname $RED_VLAN ip6 daddr fda0:7a62:14::b meta l4proto { tcp, udp } \
            th dport { 8021, 8464 } ct state new counter accept
        iifname $RED_VLAN ip daddr 10.0.20.12 meta l4proto { tcp, udp } \
            th dport { 8022, 8465 } ct state new counter accept
        iifname $RED_VLAN ip6 daddr fda0:7a62:14::c meta l4proto { tcp, udp } \
            th dport { 8022, 8465 } ct state new counter accept

        # Allow established and related connections
        ct state established,related counter accept

        # Count dropped packets
        counter comment "count dropped packets"
    }
}
```

Comme dans le cas de la chaîne `prerouting`, il est nécessaire de décrire le plus finement possible le premier flux autorisé à entrer dans le mécanisme de suivi des communication. Cela engendre un nombre important de règles même si on parvient à regrouper les deux protocoles de la couches transport et les numéros de ports.

Une fois ce jeu de règles implanté, ce n'est toujours pas suffisant pour accéder aux site web de chaque conteneur du réseau d'hébergement. On peut tout de même qualifier les règles à l'aide des compteurs et de l'affichage des enregistrements du suivi des communications. Voici un exemple.

On commence par lancer une requête qui ne parvient pas à charger la page web du service nginx.

```
wget http://192.168.104.130:8010
--2024-10-13 18:34:00--  http://192.168.104.130:8010/
Connexion à 192.168.104.130:8010… échec : Connexion terminée par expiration du délai d'attente.
```

Côté routeur ***Hub***, on peut examiner les compteurs relatifs à la règle qui autorise l'accès au service Web du conteneur à l'adresse IPv4 `10.0.10.10`.

```
sudo nft list table inet filter | grep '10.0.10.10'
    iifname "enp0s1.360" ip daddr 10.0.10.10 meta l4proto { tcp, udp } th dport { 8010, 8453 }
        ct state new counter packets 22 bytes 1320 accept
```

L'affichage de la liste des enregistrements de suivi des communication au niveau du routeur ***Hub*** montre qu'aucune réponse à la requête n'a été vue avec la clé `UNREPLIED`.

```
 sudo conntrack -f ipv4 -L | grep "10.0.10.10"
conntrack v1.4.8 (conntrack-tools): 2 flow entries have been shown.
tcp      6 87 SYN_SENT src=172.16.0.6 dst=192.168.104.130 sport=48430 dport=8010
    [UNREPLIED] src=10.0.10.10 dst=172.16.0.6 sport=8010 dport=48430 mark=0 use=1
```

Q150. Comment reconfigurer les services Web des conteneurs pour qu'ils soient en écoute sur les numéros de ports définis dans les règles de filtrage ci-dessus ?

On sait depuis l'installation des services nginx qu'ils sont tous en écoute sur le port `80`. On doit donc reconfigurer ces mêmes services avec les numéros de ports définis dans le tableau donné au début de cette partie.

Attention

Pour les besoins de nos tests, on se contente du protocole HTTP et on ne traite pas les ports HTTPS.

On doit réaliser deux tâches sur chaque routeur ***Spoke***.

- Définir le numéro de port en écoute pour chaque service nginx sur chaque conteneur.
- Compléter la liste des ports autorisés dans les règles de filtrage Web.

Voici un exemple des traitements à réaliser sur le routeur ***Spoke1***.

Pour reconfigurer les services Web, on lance un script avec comme celui-ci.

```
for i in {0..2}
do
    echo ">>>>>>>>>>>>>>>> c$i"
    incus exec c$i -- \
        sh -c "sed -i 's/listen 80/listen $((8010 + i))/' /etc/nginx/sites-enabled/default"
    incus exec c$i -- \
        sh -c "sed -i 's/listen \[::\]:80/listen \[::\]:$(( 8010 + i))/' /etc/nginx/sites-enabled/default"
    incus exec c$i -- systemctl restart nginx
done
```

Pour la liste des ports autorisés, on complète la ligne des services Web de la chaîne `forward` de la table `filter`. Voici une copie de la table.

```
table inet filter {
    chain forward {
        type filter hook forward priority 0; policy drop;

        # Allow outbound new connections
        oifname "ppp0" ct state new counter accept

        # Allow established and related connections
        ct state established,related counter accept

        # Allow specific inbound traffic
        # ICMP IPv4 + IPv6
        iifname "ppp0" meta l4proto {icmp, ipv6-icmp} ct state new counter accept
        # SSH
        iifname "ppp0" tcp dport 2222 ct state new counter accept
        # HTTP(S)
        iifname "ppp0" meta l4proto {tcp, udp} th dport \
            {80, 443, 8010, 8011, 8012, 8453, 8454, 8455} ct state new counter accept

        # Count dropped packets
        counter comment "count dropped packets"
    }
}
```

Pour terminer, les tests d'accès depuis l'hyperviseur sont maintenant concluants pour les protocoles IPv4 et IPv6.

```
for p in {0..2}
do
    wget -O /dev/null http://192.168.104.130:$((8010 + $p))
done
```

```
--2024-10-13 19:24:24--  http://192.168.104.130:8010/
Connexion à 192.168.104.130:8010… connecté.
requête HTTP transmise, en attente de la réponse… 200 OK
Taille : 615 [text/html]
Sauvegarde en : « /dev/null »

/dev/null  100%[=============================================>]     615  --.-KB/s    ds 0s

2024-10-13 19:24:24 (33,6 MB/s) – « /dev/null » sauvegardé [615/615]

--2024-10-13 19:24:24--  http://192.168.104.130:8011/
Connexion à 192.168.104.130:8011… connecté.
requête HTTP transmise, en attente de la réponse… 200 OK
Taille : 615 [text/html]
Sauvegarde en : « /dev/null »

/dev/null  100%[=============================================>]     615  --.-KB/s    ds 0s

2024-10-13 19:24:24 (34,4 MB/s) – « /dev/null » sauvegardé [615/615]

--2024-10-13 19:24:24--  http://192.168.104.130:8012/
Connexion à 192.168.104.130:8012… connecté.
requête HTTP transmise, en attente de la réponse… 200 OK
Taille : 615 [text/html]
Sauvegarde en : « /dev/null »

/dev/null  100%[=============================================>]     615  --.-KB/s    ds 0s

2024-10-13 19:24:24 (36,3 MB/s) – « /dev/null » sauvegardé [615/615]
```

```
for p in {0..2}
do
    wget -O /dev/null http://[2001:678:3fc:168::82]:$((8010 + $p))
done
```

```
--2024-10-13 19:50:30--  http://[2001:678:3fc:168::82]:8010/
Connexion à [2001:678:3fc:168::82]:8010… connecté.
requête HTTP transmise, en attente de la réponse… 200 OK
Taille : 615 [text/html]
Sauvegarde en : « /dev/null »

/dev/null  100%[=============================================>]     615  --.-KB/s    ds 0s

2024-10-13 19:50:30 (33,5 MB/s) – « /dev/null » sauvegardé [615/615]

--2024-10-13 19:50:30--  http://[2001:678:3fc:168::82]:8011/
Connexion à [2001:678:3fc:168::82]:8011… connecté.
requête HTTP transmise, en attente de la réponse… 200 OK
Taille : 615 [text/html]
Sauvegarde en : « /dev/null »

/dev/null  100%[=============================================>]     615  --.-KB/s    ds 0s

2024-10-13 19:50:30 (39,0 MB/s) – « /dev/null » sauvegardé [615/615]

--2024-10-13 19:50:30--  http://[2001:678:3fc:168::82]:8012/
Connexion à [2001:678:3fc:168::82]:8012… connecté.
requête HTTP transmise, en attente de la réponse… 200 OK
Taille : 615 [text/html]
Sauvegarde en : « /dev/null »

/dev/null 100%[=============================================>]     615  --.-KB/s    ds 0s

2024-10-13 19:50:30 (42,4 MB/s) – « /dev/null » sauvegardé [615/615]
```

C'est gagné ! La traduction d'adresse destination au niveau du routeur ***Hub*** est validée.

6. Pour conclure...

Ce document a présenté les concepts fondamentaux du filtrage réseau avec `netfilter/nftables` sur Linux. Il a couvert la configuration de base des règles de filtrage, la protection contre les attaques courantes, et la mise en place de traductions d'adresses sur une topologie ***Hub & Spoke***.

Les travaux pratiques ont permis d'explorer concrètement l'implémentation des règles de filtrage ***stateful*** et ***stateless***. Les étudiants ont pu configurer et tester des règles pour sécuriser les routeurs, contrôler le trafic traversant, et mettre en place de la traduction d'adresse source puis destination.

Ce support fournit une base solide pour comprendre et mettre en œuvre le filtrage réseau avec les outils de filtrage des systèmes Linux. Les compétences acquises permettront aux étudiants de concevoir et déployer des politiques de sécurité réseau efficaces dans des environnements réels.

Introduction au routage dynamique OSPF avec FRRouting

https://www.inetdoc.net

Résumé

Le protocole OSPF (*Open Shortest Path First*) permet un routage dynamique efficace et évolutif dans les grands réseaux IP, en calculant automatiquement les meilleures routes en fonction de l'état des liens.

Ce support de travaux pratiques est une introduction au protocole de routage dynamique OSPF. Il détaille la mise en place d'une topologie en triangle utilisant des VLANs, ainsi que la configuration des routeurs avec la suite logicielle FRRouting pour activer et paramétrer OSPF dans une aire unique.

Les manipulations présentées expliquent comment préparer les systèmes, valider les communications entre routeurs et configurer les démons OSPFv2 et OSPFv3 étape par étape.

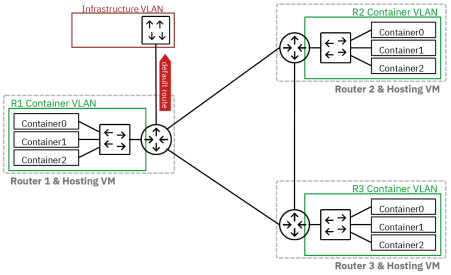


Table des matières

1. Objectifs

Après avoir réalisé les manipulations présentées dans ce document, les étudiants seront capables de :

1. Configurer et activer le protocole de routage dynamique OSPF (versions 2 et 3) sur des routeurs virtuels Linux utilisant FRRouting.
2. Mettre en place une topologie réseau en triangle avec des VLANs et configurer le routage inter-VLAN.
3. Publier et redistribuer des routes par défaut via OSPF dans une aire unique.
4. Créer et configurer des réseaux d'hébergement de conteneurs attachés à chaque routeur, en utilisant des commutateurs virtuels et de l'adressage automatique.
5. Ajuster les métriques OSPF en fonction des débits des liens et optimiser le routage.

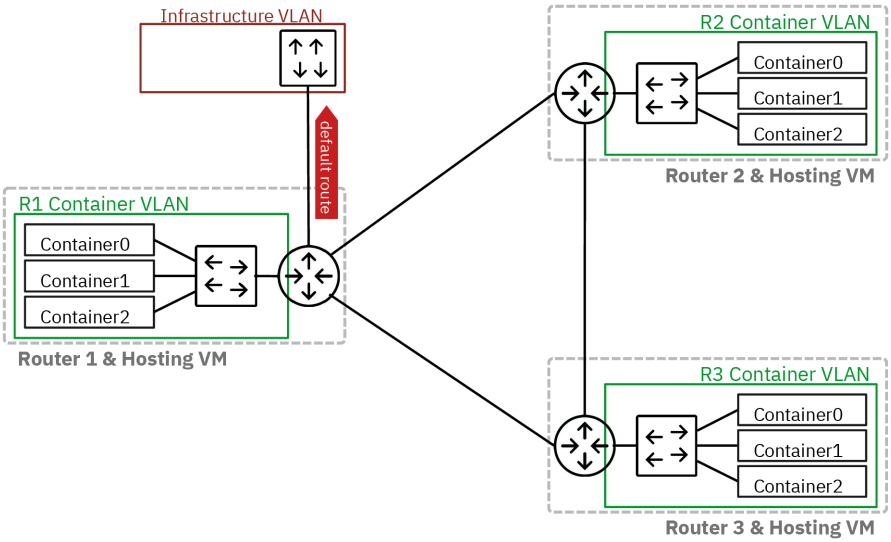
2. Topologie réseau étudiée

La topologie réseau étudiée peut être présentée sous deux formes distinctes : logique et physique.

Topologie logique

On retrouve un grand classique dans l'introduction aux protocoles de routage dynamiques : le triangle dont les trois côtés (liens) sont de type LAN.

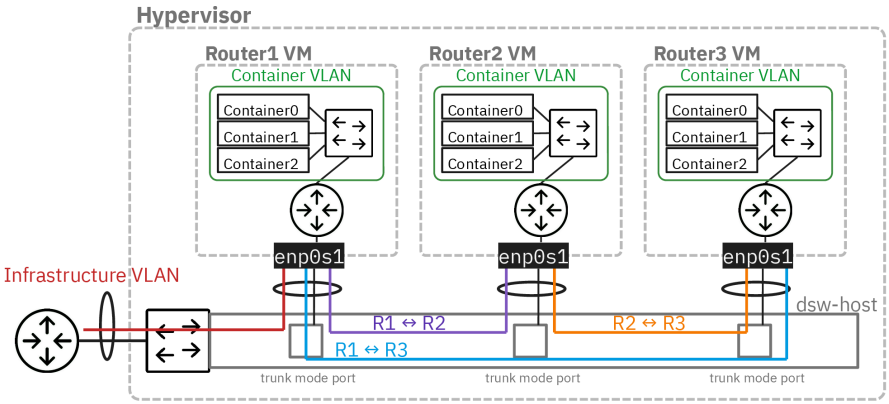
Si on se place sur un sommet du triangle, le côté opposé correspond à un réseau “inconnu” que l'on peut joindre via les routeurs voisins.



Topologie logique en triangle

Topologie physique

On s'appuie sur le support *Routage Inter-VLAN* pour constituer une topologie physique à base de réseaux locaux virtuels ou VLANs. On fait correspondre à chaque lien de la topologie logique en triangle un numéro de VLAN défini.



Topologie physique en étoile

Après avoir mis en œuvre la topologie physique en s'appuyant sur le support de la séance de travaux pratiques précédente : *Routage Inter-VLAN*, on implante les démons de routage OSPF sur les trois routeurs `R1`, `R2` et `R3`.

Ce support se limite à l'étude du routage dynamique à l'intérieur d'une aire unique. La seule “frontière” de communication inter-aires visible est constituée par le lien vers l'Internet. Cette route par défaut sera redistribuée via OSPF par le routeur `R1` aux autres routeurs. On verra alors un exemple de route externe dans les bases de données OSPF.

Voici le plan d'adressage de la maquette qui a été utilisée pour rédiger ce document.

Tableau 1. Plan d'adressage de la maquette « Introduction au routage dynamique OSPF »

Rôle	OSPF router-id	VLAN	Type	Adresses
R1	OSPFv2 : 1.0.0.4 OSPFv3 : 1.0.0.6	360	Passerelle	`192.168.104.129/29` `fe80:168::1`
		440	R1 -> R2	`10.44.0.1/29` `fe80::1b8:1/64`
		441	R1 -> R3	`10.44.1.1/29` `fe80::1b9:1/64`
		10	Hébergement	`10.10.0.1/24`

Rôle	OSPF router-id	VLAN	Type	Adresses
				fd14:ca46:3864:a::1/64
R2	OSPFv2 : 2.0.0.4 OSPFv3 : 2.0.0.6	440	R2 -> R1	10.44.0.2/29 fe80::1b8:2/64
		442	R2 -> R3	10.44.2.2/29 fe80::1ba:2/64
		20	Hébergement	10.20.0.1/24 fd14:ca46:3864:14::1/64
R3	OSPFv2 : 3.0.0.4 OSPFv3 : 3.0.0.6	441	R3 -> R1	10.44.1.3/29 fe80::1b9:3/64
		442	R3 -> R2	10.44.2.3/29 fe80::1ba:3/64
		30	Hébergement	10.30.0.1/24 fd14:ca46:3864:1e::1/64

3. Préparer les systèmes pour le routage IPv4 et IPv6

3.1. Raccorder et lancer les routeurs virtuels

Les trois machines virtuelles de routage doivent être raccordées au commutateur de distribution de l'hyperviseur.

Q151. Comment configurer les ports du commutateur avant le lancement des machines virtuelles ?

On utilise le script de procédure `switch-conf.py` qui applique les déclarations contenues dans un fichier YAML. Le code du script est accessible à partir du dépôt Git *startup-scripts*.

Voici une copie du fichier de configuration `switch.yaml` des trois ports de commutateur.

```
ovs:
  switches:
    - name: dsw-host
      ports:
        - name: tap5 # R1 switch port
          type: OVSPort
          vlan_mode: trunk
          trunks: [360, 440, 441]
        - name: tap6 # R2 switch port
          type: OVSPort
          vlan_mode: trunk
          trunks: [52, 440, 442] # VLAN 52 provides Internet temporary access
#         trunks: [440, 442]
        - name: tap7 # R3 switch port
          type: OVSPort
          vlan_mode: trunk
          trunks: [52, 441, 442] # VLAN 52 provides Internet temporary access
#         trunks: [441, 442]
```

On applique les paramètres définis ci-dessus.

```
$HOME/masters/scripts/switch-conf.py switch.yaml
```

Les numéros de ports de commutateur et de VLAN donnés dans les exemples ci-dessus doivent être adaptés selon le plan d'adressage spécifique à vos travaux pratiques.

Q152. Comment afficher les entrées de la table CAM du commutateur de distribution sur l'hyperviseur ?

Rechercher les options de la commande ovs-appctl à partir de la console de l'hyperviseur.

L'affichage de la table CAM du commutateur `dsw-host` de l'hyperviseur se fait à l'aide de la commande :

```
sudo ovs-appctl fdb/show dsw-host
```

Pour sélectionner un VLAN particulier, on ajoute un appel à grep. Voici des exemples d'affichage pour les VLANs de la maquette *une fois que les routeurs virtuels sont lancés et échangent entre eux*.

- Lien entre `R1` et `R2`.

```
sudo ovs-appctl fdb/show dsw-host | grep 440
   10   440  b8:ad:ca:fe:00:05  274
   18   440  b8:ad:ca:fe:00:06  120
```

- Lien entre R1 et R3.

```
sudo ovs-appctl fdb/show dsw-host | grep 441
   12   441  b8:ad:ca:fe:00:07    1
   10   441  b8:ad:ca:fe:00:05    1
```

- Lien entre R2 et R3.

```
sudo ovs-appctl fdb/show dsw-host | grep 442
   18   442  b8:ad:ca:fe:00:06   51
   12   442  b8:ad:ca:fe:00:07   51
```

Q153. Comment lancer les trois routeurs virtuels ?

On utilise le script de procédure `lab-startup.py` qui applique les déclarations contenues dans un fichier YAML. Le code du script est accessible à partir du dépôt Git *startup-scripts*.

Voici une copie du fichier `frr-lab.yaml` de déclaration des trois routeurs virtuels.

```
kvm:
  vms:
    - vm_name: R1
      os: linux
      master_image: debian-testing-amd64.qcow2 # master image to be used
      force_copy: false # do not force copy the master image to the VM image
      memory: 1024
      tapnum: 5
    - vm_name: R2
      os: linux
      master_image: debian-testing-amd64.qcow2 # master image to be used
      force_copy: false # do not force copy the master image to the VM image
      memory: 1024
      tapnum: 6
    - vm_name: R3
      os: linux
      master_image: debian-testing-amd64.qcow2 # master image to be used
      force_copy: false # do not force copy the master image to the VM image
      memory: 1024
      tapnum: 7
```

Les numéros d'interfaces `tap` sont à changer suivant les attributions du plan d'adressage de travaux pratiques.

On lance les trois routeurs virtuels avec le script lab-startup.py.

```
$HOME/masters/scripts/lab-startup.py frr-lab.yaml
```

Q154. Comment changer le nom d'hôte de chacun des trois routeurs virtuels ?

Utiliser la connexion console avec la commande telnet et définir le nouveau nom d'hôte avec la commande hostnamectl.

On obtient la liste des numéros de ports d'accès console au lancement des routeurs virtuels ou à l'aide de la commande lsof -i par exemple. Ensuite, on ouvre la connexion sur le port ouvert et on change le nom d'hôte et on redémarre le système.

```
telnet localhost 2306
```

```
Trying ::1...
Connected to localhost.
Escape character is '^]'.

localhost login: etu
Mot de passe :
Linux localhost 6.11.4-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.11.4-1 (2024-10-20) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
```

```
etu@localhost:~$ sudo hostnamectl hostname R2
etu@localhost:~$ sudo reboot
```

3.2. Activer le routage sur les routeurs virtuels

Sans modification de la configuration par défaut, un système GNU/Linux n'assure pas la fonction de routage du trafic d'une interface réseau à une autre.

L'activation du routage correspond à un réglage de paramètres du sous-système réseau du noyau Linux. L'outil qui permet de consulter et modifier les réglages de paramètre sur le noyau est appelé sysctl.

Q155. Comment activer le routage dans le sous-système réseau du noyau Linux ?

Utiliser la commande sysctl pour effectuer des recherches et identifier les paramètres utiles.

Il est dorénavant recommandé de créer un fichier de configuration spécifique par fonction. C'est la raison pour laquelle on crée un nouveau fichier `/etc/sysctl.d/10-routing.conf`.

Voici un exemple de création de fichier qui active les clés relatives au routage IPv4 et IPv6.

```
cat << EOF | sudo tee /etc/sysctl.d/10-routing.conf
net.ipv4.ip_forward=1
net.ipv6.conf.all.forwarding=1
net.ipv4.conf.all.log_martians=1
EOF
```

Avertissement

Il ne faut pas oublier d'appliquer les nouvelles valeurs des paramètres de configuration.

```
sudo sysctl --system
```

On utilise la même commande pour vérifier que la fonction routage est bien activée pour les deux protocoles réseau.

```
sudo sysctl net.ipv4.ip_forward net.ipv6.conf.all.forwarding
```

```
net.ipv4.ip_forward = 1
net.ipv6.conf.all.forwarding = 1
```

3.3. Appliquer une première configuration réseau

Q156. Comment appliquer les configurations réseau IPv4 et IPv6 à partir de l'unique interface du routeur ?

Consulter la documentation de ***Netplan*** pour obtenir les informations sur la configuration des interfaces réseau à l'adresse *Netplan documentation*.

Il existe plusieurs possibilités pour configurer une interface réseau. Dans le contexte de ces manipulations, on utilise ***Netplan*** dans le but de séparer la partie déclarative du moteur de configuration.

C'est `systemd-networkd` qui joue le rôle de moteur de configuration sur les machines virtuelles utilisées avec ces manipulations.

La configuration par défaut de l'interface `enp0s1` doit être éditée et remplacée. Il faut configurer autant de sous-interfaces que de VLANs utilisés.

- L'interface principale correspond à l'interface “physique” de la machine. Elle est nommée `enp0s1` en fonction de l'ordre des adresses des composants raccordés au bus PCI.
- Une sous-interface doit être créée pour chaque VLAN désigné dans le plan d'adressage.

  Le routeur `R1` se distingue des deux autres par son raccordement au VLAN d'infrastructure qui lui donne directement accès à l'Internet.

  Les routeurs `R2` et `R3` doivent utiliser un accès temporaire à un VLAN sur lequel l'adressage automatique est actif. Le but de cet accès temporaire est de permettre l'installation des paquets nécessaires à la suite des manipulations.

Voici une copie du fichier `/etc/netplan/enp0s1.yaml` du routeur `R1`.

```
network:
    version: 2
    ethernets:
      enp0s1:
        dhcp4: false
        dhcp6: false
        accept-ra: false
        nameservers:
          addresses:
            - 172.16.0.2
            - 2001:678:3fc:3::2

    vlans:
      # Infrastructure VLAN
      enp0s1.360:
        id: 360
        link: enp0s1
        addresses:
          - 192.168.104.130/29
          - 2001:678:3fc:168::82/64
        routes:
          - to: default
            via: 192.168.104.129
          - to: "::/0"
            via: "fe80::168:1"
            on-link: true
      # R1 -> R2
      enp0s1.440:
        id: 440
        link: enp0s1
        addresses:
          - 10.44.0.1/29
          - fe80::1b8:1/64
      # R1 -> R3
      enp0s1.441:
        id: 441
        link: enp0s1
        addresses:
          - 10.44.1.1/29
          - fe80::1b9:1/64
```

Voici une copie du fichier `/etc/netplan/enp0s1.yaml` du routeur `R2` avec une sous-interface temporaire dans le VLAN `52`.

```
network:
    version: 2
    ethernets:
      enp0s1:
        dhcp4: false
        dhcp6: false
        accept-ra: false
        nameservers:
          addresses:
            - 172.16.0.2
            - 2001:678:3fc:3::2

    vlans:
      # Temporary Internet access
      enp0s1.52:
        id: 52
        link: enp0s1
        dhcp4: true
        dhcp6: false
        accept-ra: true
      # R2 -> R1
      enp0s1.440:
        id: 440
        link: enp0s1
        addresses:
          - 10.44.0.2/29
          - fe80::1b8:2/64
      # R2 -> R3
      enp0s1.442:
        id: 442
        link: enp0s1
        addresses:
          - 10.44.2.2/29
          - fe80::1ba:2/64
```

Ce deuxième fichier de déclaration de la configuration réseau est pratiquement identique à celui du routeur `R3`. C'est pourquoi, on ne fournit pas de copie de ce dernier.

Une fois le fichier de configuration en place, il suffit de faire appel à la commande netplan pour appliquer les changements.

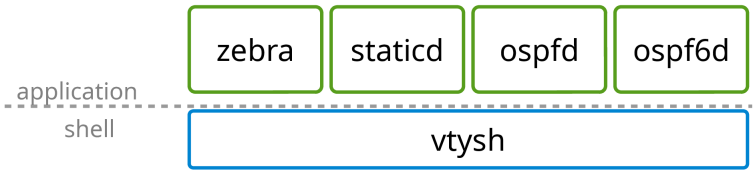
```
sudo netplan apply
```

Pour vérifier l'état courant de la configuration appliquée, on peut utiliser à nouveau la commande netplan.

```
sudo netplan status
```

4. Installer le paquet frr et lancer les démons de routage OSPF

La suite de démons de routage FRRouting couvre la totalité des protocoles de routage dynamiques. Le paquet FRR fournit autant de processus que de protocoles. Ici, on se concentre sur le protocole OSPF. Voici une représentation de l'organisation des démons actifs dans notre contexte.



Les manipulations de ce support s'appuient sur la documentation du projet : *FRRouting User Guide*

> **Note**
>
> Au moment de la rédaction de ce document, FRRouting ne permet pas d'utiliser simultanément les familles d'adresses IPv4 et IPv6 avec un seul démon. C'est pourquoi nous devons activer et configurer deux démons distincts : `ospfd` pour IPv4 et `ospf6d` pour IPv6.

> **Important**
>
> Les questions de cette section doivent être traitées sur les trois routeurs de la topologie étudiée.

Q157. Comment installer le paquet `frr` à partir du dépôt du site *FRRouting Project* ?

Pour installer le paquet FRR, on doit ajouter un nouveau dépôt au système.

On commence par ajouter la clé de signature des paquets à la configuration du gestionnaire.

```
sudo apt -y install curl gpg
```

```
curl -s https://deb.frrouting.org/frr/keys.asc | \
sudo gpg -o /usr/share/keyrings/frr-keyring.gpg --dearmor
```

On créé ensuite un nouveau fichier de liste de sources de paquets qui fait référence à cette clé de signature.

```
echo "deb [signed-by=/usr/share/keyrings/frr-keyring.gpg] \
https://deb.frrouting.org/frr bookworm frr-stable" | \
sudo tee /etc/apt/sources.list.d/frr.list
```

Avant de lancer l'installation des paquets de la suite FRRouting, on doit mettre à jour le catalogue local.

```
sudo apt update
sudo apt -y install frr frr-pythontools
```

On peut afficher les informations sur le paquet `frr`.

```
apt show ^frr$
```

Sans configuration particulière, les services zebra et staticd sont lancés. Aucun protocole de routage dynamique n'est activé.

```
systemctl status frr
```

```
# frr.service - FRRouting
     Loaded: loaded (/usr/lib/systemd/system/frr.service; enabled; preset: enabled)
     Active: active (running) since Sat 2024-11-02 16:17:42 CET; 5min ago
 Invocation: a1d3f0e79647471bb1a17069f3f4c69a
       Docs: https://frrouting.readthedocs.io/en/latest/setup.html
    Process: 2098 ExecStart=/usr/lib/frr/frrinit.sh start (code=exited, status=0/SUCCESS)
   Main PID: 2107 (watchfrr)
     Status: "FRR Operational"
      Tasks: 8 (limit: 1032)
     Memory: 14.9M (peak: 28.2M)
        CPU: 416ms
     CGroup: /system.slice/frr.service
             ├─2107 /usr/lib/frr/watchfrr -d -F traditional zebra mgmtd staticd
             ├─2117 /usr/lib/frr/zebra -d -F traditional -A 127.0.0.1 -s 90000000
             ├─2122 /usr/lib/frr/mgmtd -d -F traditional -A 127.0.0.1
             └─2124 /usr/lib/frr/staticd -d -F traditional -A 127.0.0.1

nov. 02 16:17:42 R2 staticd[2124]: [VTVCM-Y2NW3] Configuration Read in Took: 00:00:00
nov. 02 16:17:42 R2 frrinit.sh[2144]: [2144|staticd] done
nov. 02 16:17:42 R2 zebra[2117]: [VTVCM-Y2NW3] Configuration Read in Took: 00:00:00
nov. 02 16:17:42 R2 frrinit.sh[2128]: [2128|zebra] done
nov. 02 16:17:42 R2 watchfrr[2107]: [QDG3Y-BY5TN] zebra state -> up : connect succeeded
nov. 02 16:17:42 R2 watchfrr[2107]: [QDG3Y-BY5TN] mgmtd state -> up : connect succeeded
nov. 02 16:17:42 R2 watchfrr[2107]: [QDG3Y-BY5TN] staticd state -> up : connect succeeded
nov. 02 16:17:42 R2 watchfrr[2107]: [KWE5Q-QNGFC] all daemons up, doing startup-complete notify
nov. 02 16:17:42 R2 frrinit.sh[2098]: Started watchfrr.
nov. 02 16:17:42 R2 systemd[1]: Started frr.service - FRRouting.
```

Q158. Comment vérifier que la console unifiée du service `frr` est active ?

Afficher le contenu du fichier `/etc/frr/vtysh.conf` et vérifier qu'il contient l'entrée `service integrated-vtysh-config`.

L'accès à cette console unifiée est important puisqu'il permet d'utiliser une console unique pour les trois démons qui sont utilisés dans la suite des manipulations : zebra, ospfd et ospf6d.

Voici un exemple pour un routeur de la maquette.

```
sudo grep "service integrated-vtysh-config" /etc/frr/vtysh.conf
```

```
service integrated-vtysh-config
```

Q159. Comment ajouter l'utilisateur `etu` aux groupes `frr` et `frrvty` ?

Utiliser la commande adduser.

Une fois que l'utilisateur appartient à ces groupes, il a un accès direct à la console de configuration des protocoles actifs.

Comme dans les autres travaux pratiques de la série, on utilise une boucle.

```
for grp in frr frrvty
do
    sudo adduser etu $grp
done
```

Il ne faut pas oublier de déconnecter/reconnecter l'utilisateur pour bénéficier de la nouvelle attribution de groupe.

On vérifie l'appartenance aux groupes avec la commande groups.

```
groups
```

```
etu adm sudo users frrvty frr
```

Q160. Comment activer les deux démons des protocoles OSPFv2 et OSPFv3 ?

Consulter le fichier de configuration : `/etc/frr/daemons`.

Il faut remplacer les clés `no` en `yes` pour les démons des deux versions du protocole OSPF.

```
sudo sed -i 's/ospfd=no/ospfd=yes/' /etc/frr/daemons
sudo sed -i 's/ospf6d=no/ospf6d=yes/' /etc/frr/daemons
sudo systemctl restart frr
```

On peut alors afficher l'état du service `frr` et vérifier que les nouveaux démons de routage dynamique OSPF sont bien activés.

```
systemctl status frr
```

```
# frr.service - FRRouting
     Loaded: loaded (/usr/lib/systemd/system/frr.service; enabled; preset: enabled)
     Active: active (running) since Sat 2024-11-02 17:40:32 CET; 6s ago
 Invocation: 84e33888211f45f297c9135cace76751
       Docs: https://frrouting.readthedocs.io/en/latest/setup.html
    Process: 2467 ExecStart=/usr/lib/frr/frrinit.sh start (code=exited, status=0/SUCCESS)
   Main PID: 2476 (watchfrr)
     Status: "FRR Operational"
      Tasks: 12 (limit: 1032)
     Memory: 23M (peak: 34.8M)
        CPU: 380ms
     CGroup: /system.slice/frr.service
             ├─2476 /usr/lib/frr/watchfrr -d -F traditional zebra mgmtd ospfd ospf6d staticd
             ├─2488 /usr/lib/frr/zebra -d -F traditional -A 127.0.0.1 -s 90000000
             ├─2493 /usr/lib/frr/mgmtd -d -F traditional -A 127.0.0.1
             ├─2495 /usr/lib/frr/ospfd -d -F traditional -A 127.0.0.1
             ├─2498 /usr/lib/frr/ospf6d -d -F traditional -A ::1
             └─2501 /usr/lib/frr/staticd -d -F traditional -A 127.0.0.1

nov. 02 17:40:32 R2 mgmtd[2493]: [VTVCM-Y2NW3] Configuration Read in Took: 00:00:00
nov. 02 17:40:32 R2 frrinit.sh[2504]: [2504|mgmtd] done
nov. 02 17:40:32 R2 watchfrr[2476]: [QDG3Y-BY5TN] zebra state -> up : connect succeeded
nov. 02 17:40:32 R2 watchfrr[2476]: [QDG3Y-BY5TN] mgmtd state -> up : connect succeeded
nov. 02 17:40:32 R2 watchfrr[2476]: [QDG3Y-BY5TN] ospfd state -> up : connect succeeded
nov. 02 17:40:32 R2 watchfrr[2476]: [QDG3Y-BY5TN] ospf6d state -> up : connect succeeded
nov. 02 17:40:32 R2 watchfrr[2476]: [QDG3Y-BY5TN] staticd state -> up : connect succeeded
nov. 02 17:40:32 R2 watchfrr[2476]: [KWE5Q-QNGFC] all daemons up, doing startup-complete notify
nov. 02 17:40:32 R2 frrinit.sh[2467]: Started watchfrr.
nov. 02 17:40:32 R2 systemd[1]: Started frr.service - FRRouting.
```

On peut aussi lister les démons actifs à partir de la console du service.

```
vtysh
```

```
Hello, this is FRRouting (version 10.1.1).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

R2# sh daemons
 mgmtd zebra ospfd ospf6d watchfrr staticd
```

Q161. Comment désactiver les sous-interfaces d'accès temporaire à Internet pour les routeurs R2 et R3 ?

Il existe au moins deux possibilités.

- Utiliser directement la commande ip pour désactiver une sous-interface au niveau liaison.
- Éditer le fichier de déclaration et commenter tous les paramètre de la sous-interface temporaire.

Si on utilise la commande ip, l'interface sera bien désactivée mais elle apparaîtra à nouveau lors du redémarrage du routeur virtuel.

```
sudo ip link set enp0s1.52 down
```

Si on édite le fichier de déclaration en commentant les paramètres de la sous-interface, on s'assure qu'elle n'apparaîtra plus, même lors d'un redémarrage du routeur. Voici un exemple pour le routeur R3.

```
network:
    version: 2
    ethernets:
      enp0s1:
        dhcp4: false
        dhcp6: false
        accept-ra: false
        nameservers:
          addresses:
            - 172.16.0.2
            - 2001:678:3fc:3::2

    vlans:
#      # Temporary Internet access
#      enp0s1.52:
#        id: 52
#        link: enp0s1
#        dhcp4: true
#        dhcp6: false
#        accept-ra: true
      # R3 -> R1
      enp0s1.440:
        id: 440
        link: enp0s1
        addresses:
          - 10.44.0.3/29
          - fe80::1b8:3/64
      # R3 -> R2
      enp0s1.442:
        id: 442
        link: enp0s1
        addresses:
          - 10.44.2.3/29
          - fe80::1ba:3/64
```

Une fois l'ensemble des opérations de cette section réalisées, chaque routeur dispose des outils pour mettre en œuvre la topologie physique et ensuite les protocoles de routage dynamique OSPF.

5. Valider les communications entre routeurs

Avant d'aborder le déploiement du protocole de routage dynamique, il est nécessaire de valider plusieurs éléments :

- Le raccordement des routeurs aux ports de commutateurs désignés
- Les communications entre chaque routeur
- La visualisation des tables de routage pour les interfaces réseau configurées

Q162. Quelles sont les tests à effectuer pour vérifier l'état des différents “côtés” de la topologie triangle ?

Il faut afficher la table de routage de chaque routeur puis la table des voisins. Ainsi, on peut contrôler les correspondances entre les adresses de couche réseau et de couche liaison.

Dans le contexte de la maquette on obtient les résultats suivants pour le routeur `R1`. On se limite à l'affichage des entrées de la table de routage apprises par le noyau avec l'option `proto kernel`.

```
ip route ls proto kernel
```

```
10.44.0.0/29 dev enp0s1.440 scope link src 10.44.0.1
10.44.1.0/29 dev enp0s1.441 scope link src 10.44.1.1
192.168.104.128/29 dev enp0s1.360 scope link src 192.168.104.130
```

```
ip -6 route ls proto kernel
```

```
2001:678:3fc:168::/64 dev enp0s1.360 metric 256 pref medium
fe80::/64 dev enp0s1 metric 256 pref medium
fe80::/64 dev enp0s1.441 metric 256 pref medium
fe80::/64 dev enp0s1.360 metric 256 pref medium
fe80::/64 dev enp0s1.440 metric 256 pref medium
```

Pour les entrées relatives au voisinage réseau, on se limite aussi à l'affichage des voisins directs des interfaces du routeur.

On repère dans la copie d'écran ci-dessous les adresses des deux autres routeurs : `R2` et `R3`.

```
ip nei ls
```

```
10.44.0.2 dev enp0s1.440 lladdr b8:ad:ca:fe:00:06 REACHABLE❶
10.44.1.3 dev enp0s1.441 lladdr b8:ad:ca:fe:00:07 REACHABLE❷
fe80::1b9:3 dev enp0s1.441 lladdr b8:ad:ca:fe:00:07 STALE❸
fe80::baad:caff:fefe:6 dev enp0s1.440 lladdr b8:ad:ca:fe:00:06 router STALE❹
fe80:168::1 dev enp0s1.360 lladdr 80:6a:00:dc:67:53 router STALE
fe80::baad:caff:fefe:7 dev enp0s1.441 lladdr b8:ad:ca:fe:00:07 STALE❺
fe80::1b8:2 dev enp0s1.440 lladdr b8:ad:ca:fe:00:06 STALE❻
```

❶❻❹ Entrées du routeur R2
❷❸❺ Entrées du routeur R3

Important

Les tables de voisinage réseau sont peuplées à partir des échanges entre routeurs. Il est donc nécessaire de lancer les tests de communication ICMP en amont de l'affichage de ces tables.

Q163. Quelles sont les opérations à effectuer pour valider les communications IPv4 et IPv6 entre chacun des routeurs ?

Lancer les tests ICMP usuels entre chaque routeur sur chaque lien actif.

Exemple entre R1 et R2 ; toujours dans le contexte de la maquette.

- Requête IPv4 de R1 vers R2 sur le VLAN 440.

```
ping -qc2 10.44.0.2
```

```
PING 10.44.0.2 (10.44.0.2) 56(84) bytes of data.

--- 10.44.0.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 0.533/7.471/14.409/6.938 ms
```

Aucun paquet n'a été perdu. Le routeur R2 est bien joignable depuis R1.

- Requête IPv6 *multicast* depuis R1 sur le VLAN 480.

```
ping -qc2 ff02::1%enp0s1.440
```

```
PING ff02::1%enp0s1.440 (ff02::1%enp0s1.440) 56 data bytes

--- ff02::1%enp0s1.440 ping statistics ---
2 packets transmitted, 2 received, +1 duplicates, 0% packet loss, time 1031ms
rtt min/avg/max/mdev = 0.065/0.283/0.719/0.307 ms
```

L'opération est à répéter sur chaque lien entre deux routeurs reliés sur le même VLAN.

Q164. Quelles sont les opérations à effectuer pour visualiser les tables de routage IPv4 et IPv6 existantes d'un routeur au niveau de la console unifiée vtysh ?

Afficher les tables de routage à partir de la console vtysh avec les commandes du système Cisco™ IOS show ip route et show ipv6 route.

Dans le contexte de la maquette, on obtient les résultats suivants pour le routeur R2.

```
R2# sh ip route connected
Codes: K - kernel route, C - connected, L - local, S - static,
       R - RIP, O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,
       f - OpenFabric, t - Table-Direct,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup
       t - trapped, o - offload failure

C>* 10.44.0.0/29 is directly connected, enp0s1.440, 00:40:37
C>* 10.44.2.0/29 is directly connected, enp0s1.442, 00:40:37
```

```
R2# sh ipv6 route connected
Codes: K - kernel route, C - connected, L - local, S - static,
       R - RIPng, O - OSPFv3, I - IS-IS, B - BGP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,
       f - OpenFabric, t - Table-Direct,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup
       t - trapped, o - offload failure

C * fe80::/64 is directly connected, enp0s1, 00:41:41
C * fe80::/64 is directly connected, enp0s1.442, 00:41:41
C>* fe80::/64 is directly connected, enp0s1.440, 00:41:41
```

Comme dans le cas de la question précédente, l'affichage des tables de routage doit être fait sur les trois routeurs virtuels pour vérifier la cohérence de la topologie avant de passer au routage dynamique.

6. Configurer les démons OSPFv2 et OSPFv3

Dans cette section, on introduit les premières commandes de configuration du protocole de routage dynamique OSPF qui permettent d'activer le protocole puis d'ajouter des entrées de réseau dans la base de données de ce protocole.

Q165. Comment peut on contrôler que le protocole OSPF est actif ou non sur un routeur ?

Une fois la console vtysh ouverte, lancer les commandes de visualisation de l'état du protocole listées ci-dessous. Ces commandes peuvent être lancées sur chacun des trois routeurs.

show daemons
show ip ospf
show ipv6 ospf6

Dans les informations données dans la copie d'écran ci-dessous, il apparaît qu'aucune configuration du protocole de routage dynamique n'a été activée.

```
R1# sh daemons
 zebra ospfd ospf6d watchfrr staticd
```

```
R1# sh ip ospf
```

```
R1# sh ipv6 ospf
% OSPFv3 instance not found
```

Q166. Quelles sont les opérations à effectuer pour activer les protocoles de routage OSPFv2 et OSPFv3 ? Comment affecter manuellement l'identifiant du routeur ?

Avertissement

Les identifiants à utiliser lors de la séance de travaux pratiques sont donnés dans les tableaux des plans d'adressage.

La liste des commandes utiles en mode configuration dans la console vtysh est la suivante.

router ospf
router ospf6
ospf router-id X.X.X.X
ospf6 router-id X.X.X.X
log detail

Toujours à partir de la console vtysh, on accède au mode configuration à l'aide de la commande conf t. Voici un exemple de séquence sur le troisième routeur.

```
R1# conf t
R1(config)# router ospf
R1(config-router)# ospf router-id 1.0.0.4
R1(config-router)# log detail
R1(config-router)# ^Z
```

```
R1# conf t
R1(config)# router ospf6
R1(config-ospf6)# ospf6 router-id 1.0.0.6
R1(config-ospf6)# log detail
R1(config-ospf6)# ^Z
```

Une fois que chaque démon de routage OSPF possède un identifiant unique, on peut afficher les propriétés du protocole de routage dynamique même si aucun échange de route n'a encore eu lieu.

```
sh run ospfd
```

```
Building configuration...

Current configuration:
!
frr version 10.1.1
frr defaults traditional
hostname R1
log syslog informational
service integrated-vtysh-config
!
router ospf
 ospf router-id 1.0.0.4
 log-adjacency-changes detail
exit
!
end
```

```
sh run ospf6d
```

```
Building configuration...

Current configuration:
!
frr version 10.1.1
frr defaults traditional
hostname R1
log syslog informational
service integrated-vtysh-config
!
router ospf6
 ospf6 router-id 1.0.0.6
 log-adjacency-changes detail
exit
!
end
```

Le choix de codage des identifiants OSPF a pour but d'éviter une confusion avec les adresses des réseaux actifs sur chaque routeur.

Si on reprend les instructions de la question précédente, on obtient l'état de chacun des démons de protocole de routage dynamique.

```
sh ip ospf
```

```
 OSPF Routing Process, Router ID: 1.0.0.4
 Supports only single TOS (TOS0) routes
 This implementation conforms to RFC2328
 RFC1583Compatibility flag is disabled
 OpaqueCapability flag is disabled
 Initial SPF scheduling delay 0 millisec(s)
 Minimum hold time between consecutive SPFs 50 millisec(s)
 Maximum hold time between consecutive SPFs 5000 millisec(s)
 Hold time multiplier is currently 1
 SPF algorithm has not been run
 SPF timer is inactive
 LSA minimum interval 5000 msecs
 LSA minimum arrival 1000 msecs
 Write Multiplier set to 20
 Refresh timer 10 secs
 Maximum multiple paths(ECMP) supported 256
 Administrative distance 110
 Number of external LSA 0. Checksum Sum 0x00000000
 Number of opaque AS LSA 0. Checksum Sum 0x00000000
 Number of areas attached to this router: 0
 All adjacency changes are logged
```

```
sh ipv6 ospf
```

```
 OSPFv3 Routing Process (0) with Router-ID 1.0.0.6
 Running 00:09:20
 LSA minimum arrival 1000 msecs
 Maximum-paths 256
 Administrative distance 110
 Initial SPF scheduling delay 0 millisec(s)
 Minimum hold time between consecutive SPFs 50 millsecond(s)
 Maximum hold time between consecutive SPFs 5000 millsecond(s)
 Hold time multiplier is currently 1
 SPF algorithm has not been run
 SPF timer is inactive
 Number of AS scoped LSAs is 0
 Number of areas in this router is 0
 Authentication Sequence number info
  Higher sequence no 0, Lower sequence no 0
 All adjacency changes are logged
```

L'affectation des identifiants des démons de routage OSPF doit être réalisée sur les trois routeurs de la topologie. Ces identifiants seront très utiles et importants dès qu'il faudra afficher les listes de routeurs voisins au sens du protocole OSPF.

Q167. Comment activer les protocoles de routage OSPFv2 et OSPFv3 pour les réseaux d'interconnexion de chaque routeur ?

Il faut activer le protocole de routage dynamique sur chaque interface de la topologie qui participe à la construction du triangle.

La liste des commandes utiles en mode console et en mode configuration dans vtysh est la suivante.

show ip route connected
show ip route ospf
show ipv6 route connected
show ipv6 route ospf
ip ospf area 0
ipv6 ospf6 area 0

Voici un exemple de séquence d'instructions pour le routeur R1.

On commence par lister les entrées marquées ***C*** ou ***connected*** des tables de routage IPv4 et IPv6 de façon à reconnaître les deux côtés de la topologie triangle connus du “sommet” R1.

```
sh ip route connected
```

```
Codes: K - kernel route, C - connected, L - local, S - static,
       R - RIP, O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,
       f - OpenFabric, t - Table-Direct,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup
       t - trapped, o - offload failure

C>* 10.44.0.0/29 is directly connected, enp0s1.440, 01:26:12
C>* 10.44.1.0/29 is directly connected, enp0s1.441, 01:26:12
C>* 192.168.104.128/29 is directly connected, enp0s1.360, 01:26:12
```

```
sh ipv6 route connected
```

```
Codes: K - kernel route, C - connected, L - local, S - static,
       R - RIPng, O - OSPFv3, I - IS-IS, B - BGP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,
       f - OpenFabric, t - Table-Direct,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup
       t - trapped, o - offload failure

C>* 2001:678:3fc:168::/64 is directly connected, enp0s1.360, 01:30:26
C * fe80::/64 is directly connected, enp0s1.360, 01:30:25
C * fe80::/64 is directly connected, enp0s1.440, 01:30:26
C * fe80::/64 is directly connected, enp0s1, 01:30:26
C>* fe80::/64 is directly connected, enp0s1.441, 01:30:26
```

À partir de ces affichages, on sait que l'on doit activer les protocoles OSPF pour les deux sous-interfaces : enp0s1.440 et enp0s1.441.

```
R1# conf t
R1(config)# int enp0s1.440
R1(config-if)# ip ospf area 0
R1(config-if)# ipv6 ospf6 area 0
R1(config-if)# int enp0s1.441
R1(config-if)# ip ospf area 0
R1(config-if)# ipv6 ospf6 area 0
R1(config-if)# ^Z
```

Ces opérations doivent être répétées sur les trois routeurs de la topologie.

Q168. Comment visualiser l'état des interfaces actives pour chaque processus de protocole de routage dynamique OSPFv2 ou OSPFv3 ?

Les interfaces sont dites actives pour les protocoles OSPFv2 ou OSPFv3 dès qu'elles ont été ajoutées aux processus de routage dynamique en précisant l'aire à laquelle elles appartiennent.

La liste des commandes utiles dans la console vtysh est la suivante.

show ip ospf interface
show ipv6 ospf6 interface

En prenant l'exemple du routeur R2, on obtient les résultats suivants.

```
R2# sh ip ospf interface enp0s1.440
```

```
enp0s1.440 is up❶
  ifindex 4, MTU 1500 bytes, BW 0 Mbit❷ <UP,LOWER_UP,BROADCAST,RUNNING,MULTICAST>
  Internet Address 10.44.0.2/29, Broadcast 10.44.0.7, Area 0.0.0.0
  MTU mismatch detection: enabled
  Router ID 2.0.0.4❸, Network Type BROADCAST, Cost: 10
  Transmit Delay is 1 sec, State Backup, Priority 1
  Designated Router (ID) 1.0.0.4 Interface Address 10.44.0.1/29❹
  Backup Designated Router (ID) 2.0.0.4, Interface Address 10.44.0.2❺
  Multicast group memberships: OSPFAllRouters OSPFDesignatedRouters
  Timer intervals configured, Hello 10s, Dead 40s, Wait 40s, Retransmit 5
    Hello due in 5.086s
  Neighbor Count is 1, Adjacent neighbor count is 1❻
  Graceful Restart hello delay: 10s
```

La copie d'écran ci-dessus permet d'identifier les éléments suivants :

❶ L'indicateur ***is up*** confirme que l'interface est bien active pour le protocole de routage. Cela signifie que les messages du protocole OSPF sont transmis sur cette interface et que des échanges avec des routeurs OSPF voisins sur ce réseau peuvent avoir lieu.

❷ Le fait que la bande passante soit “vue” comme étant nulle montre que le calcul de coût de lien ne prend pas en compte ce facteur. Ce point sera repris dans le calcul des coûts à partir d'une référence définie dans la configuration des démons OSPF.
❸ On retrouve ici l'identifiant du routeur transmis vers les autres routeurs voisins.
❹ Cette ligne identifie le routeur OSPF `R1` comme étant le routeur de référence sur ce réseau IPv4. Dans notre cas, la topologie triangle ne comprend qu'un seul routeur voisin OSPF par réseau, ce qui limite l'utilité d'un routeur de référence pour les calculs des meilleurs routes. Si on avait plusieurs routeurs présents sur un même réseau, le rôle de référent serait essentiel pour limiter les échanges de bases de données de routage lors des calculs de tables de topologie.
❺ Cette ligen identifie le routeur OSPF `R2` comme étant le routeur de référence de secours sur ce réseau. Comme dans le cas précédent, l'utilisation d'une topologie triangle limite l'importance de ce rôle comme il n'y a que deux routeurs pour chaque côté de cette topologie.
❻ Le routeur OSPF a un routeur voisin sur ce réseau. Il s'agit du routeur `R1`.

On reprend la même démarche pour le protocole OSPFv3.

```
R2# sh ipv6 ospf6 interface enp0s1.440
```

```
enp0s1.440 is up❶, type BROADCAST
  Interface ID: 4
  Internet Address:
    inet : 10.44.0.2/29
    inet6: fe80::baad:caff:fefe:6/64
    inet6: fe80::1b8:2/64
  Instance ID 0, Interface MTU 1500 (autodetect: 1500)
  MTU mismatch detection: enabled
  Area ID 0.0.0.0, Cost 10
  State BDR, Transmit Delay 1 sec, Priority 1
  Timer intervals configured:
   Hello 10(8.803), Dead 40, Retransmit 5
  DR: 1.0.0.6 BDR: 2.0.0.4❷
  Number of I/F scoped LSAs is 2
    0 Pending LSAs for LSUpdate in Time 00:00:00 [thread off]
    0 Pending LSAs for LSAck in Time 00:00:00 [thread off]
  Graceful Restart hello delay: 10s
  Authentication Trailer is disabled
```

Relativement, à l'affichage des informations sur l'association entre une interface réseau et le démon de protocole OSPFv2 pour IPv4, l'affichage pour OSPFv3 et IPv6 est plus succinct.

❶ Cette information est identique à celle du démon OSPFv2. L'interface est associée et active. Les messages du protocole OSPFv3 peuvent être échangés sur le réseau auquel cette interface appartient.
❷ Les identifiants des routeurs référence et de secours sont affichés sur une seule ligne.

Q169. Comment vérifier que l'identifiant de routeur a correctement été attribué ?

À partir des commandes proposées et de résultats des questions précédentes, rechercher l'information demandée.

Quelque soit la version du protocole OSPF, l'identifiant de routeur est toujours codé sous la forme d'une adresse IPv4.

Pour valider la conformité entre les identifiants définis dans le plan d'adressage des travaux pratiques et les valeurs effectivement utilisées par les deux démons de routage dynamique, on affiche le contenu des base de topologie du protocole.

Avec le démon OSPFv2, l'identification du routeur est immédiate.

```
R1# sh ip ospf database
```

```
       OSPF Router with ID (1.0.0.4)

                Router Link States (Area 0.0.0.0)

Link ID         ADV Router      Age  Seq#       CkSum  Link count
1.0.0.4         1.0.0.4         884 0x8000001f 0x3bee 2
2.0.0.4         2.0.0.4        1016 0x8000001d 0xab78 2
3.0.0.4         3.0.0.4         843 0x8000001d 0xd34a 2

                Net Link States (Area 0.0.0.0)

Link ID         ADV Router      Age  Seq#       CkSum
10.44.0.1       1.0.0.4         994 0x80000018 0xe61c
10.44.1.1       1.0.0.4         984 0x80000018 0xe61a
10.44.2.3       3.0.0.4         843 0x80000018 0xbc3e
```

En revanche, le démon OSPFv3 ne donne pas d'information sur l'identifiant du processus en cours. L'affichage est cependant beaucoup plus exhaustif avec les informations par interface.

```
R1# sh ipv6 ospf6 database
```

```
       Area Scoped Link State Database (Area 0)

Type LSId            AdvRouter       Age   SeqNum                       Payload
Rtr  0.0.0.0         1.0.0.6        1012 8000001b               1.0.0.6/0.0.0.5
Rtr  0.0.0.0         1.0.0.6        1012 8000001b               1.0.0.6/0.0.0.3
Rtr  0.0.0.0         2.0.0.4        1010 80000019               1.0.0.6/0.0.0.5
Rtr  0.0.0.0         2.0.0.4        1010 80000019               3.0.0.6/0.0.0.4
Rtr  0.0.0.0         3.0.0.6        1010 80000019               1.0.0.6/0.0.0.3
Rtr  0.0.0.0         3.0.0.6        1010 80000019               3.0.0.6/0.0.0.4
Net  0.0.0.3         1.0.0.6        1088 80000017                       1.0.0.6
Net  0.0.0.3         1.0.0.6        1088 80000017                       3.0.0.6
Net  0.0.0.5         1.0.0.6        1012 80000017                       1.0.0.6
Net  0.0.0.5         1.0.0.6        1012 80000017                       2.0.0.4
Net  0.0.0.4         3.0.0.6        1010 80000017                       3.0.0.6
Net  0.0.0.4         3.0.0.6        1010 80000017                       2.0.0.4

       I/F Scoped Link State Database (I/F enp0s1.440 in Area 0)❶

Type LSId            AdvRouter       Age   SeqNum                       Payload
Lnk  0.0.0.5         1.0.0.6         303 8000001a                  fe80::1b8:1
Lnk  0.0.0.4         2.0.0.4        1019 80000017                  fe80::1b8:2

       I/F Scoped Link State Database (I/F enp0s1.441 in Area 0)❷

Type LSId            AdvRouter       Age   SeqNum                       Payload
Lnk  0.0.0.3         1.0.0.6         288 8000001a                  fe80::1b9:1
Lnk  0.0.0.3         3.0.0.6        1090 80000017                  fe80::1b9:3

       AS Scoped Link State Database

Type LSId            AdvRouter       Age   SeqNum                       Payload
```

❶ Sur le côté R1-R2 du triangle, on utilise le VLAN 440 et les identifiants de routeurs correspondent.
❷ Sur le côté R1-R3 du triangle, on utilise le VLAN 441 et les identifiants de routeurs correspondent.

Q170. Comment identifier le type de réseau d'une interface ?

À partir des résultats des questions précédentes, rechercher l'information demandée.

Comme on utilise des liens Ethernet dans ce contexte de travaux pratiques, le type le plus important est le réseau de diffusion ou *BROADCAST*.

```
R3# sh ip ospf interface enp0s1.442
```

```
enp0s1.442 is up
  ifindex 4, MTU 1500 bytes, BW 0 Mbit <UP,LOWER_UP,BROADCAST,RUNNING,MULTICAST>
  Internet Address 10.44.2.3/29, Broadcast 10.44.2.7, Area 0.0.0.0
  MTU mismatch detection: enabled
  Router ID 3.0.0.4, Network Type BROADCAST, Cost: 10
  Transmit Delay is 1 sec, State DR, Priority 1
  Designated Router (ID) 3.0.0.4 Interface Address 10.44.2.3/29
  Backup Designated Router (ID) 2.0.0.4, Interface Address 10.44.2.2
  Saved Network-LSA sequence number 0x80000019
  Multicast group memberships: OSPFAllRouters OSPFDesignatedRouters
  Timer intervals configured, Hello 10s, Dead 40s, Wait 40s, Retransmit 5
    Hello due in 1.361s
  Neighbor Count is 1, Adjacent neighbor count is 1
  Graceful Restart hello delay: 10s
```

```
R3# sh ipv6 ospf6 interface enp0s1.442
```

```
enp0s1.442 is up, type BROADCAST
  Interface ID: 4
  Internet Address:
    inet : 10.44.2.3/29
    inet6: fe80::baad:caff:fefe:7/64
    inet6: fe80::1ba:3/64
  Instance ID 0, Interface MTU 1500 (autodetect: 1500)
  MTU mismatch detection: enabled
  Area ID 0.0.0.0, Cost 10
  State DR, Transmit Delay 1 sec, Priority 1
  Timer intervals configured:
   Hello 10(9.055), Dead 40, Retransmit 5
  DR: 3.0.0.6 BDR: 2.0.0.4
  Number of I/F scoped LSAs is 2
    0 Pending LSAs for LSUpdate in Time 00:00:00 [thread off]
    0 Pending LSAs for LSAck in Time 00:00:00 [thread off]
  Graceful Restart hello delay: 10s
  Authentication Trailer is disabled
```

Q171. Comment obtenir la liste du ou des routeurs voisins pour chaque processus de protocole de routage dynamique OSPFv2 ou OSPFv3 ?

Dès qu'une interface est active, il y a émission de paquets `HELLO` et si un autre routeur avec une interface active envoie aussi des paquets `HELLO` dans le même VLAN, les deux routeurs cherchent à former une adjacence.

La liste des commandes utiles dans la console vtysh est la suivante.

show ip ospf neighbor
show ipv6 ospf6 neighbor

À partir du routeur `R1`, voici un exemple de liste de routeurs OSPF voisins dans laquelle on reconnaît les identifiants des routeurs `R2` et `R3`.

```
R1# sh ip ospf neighbor
```

```
Neighbor ID     Pri State           Up Time         Dead Time Address         Interface                RXmtL RqstL DBsmL
2.0.0.4           1 Full/Backup     11h34m21s         37.712s 10.44.0.2       enp0s1.440:10.44.0.1     0     0     0
3.0.0.4           1 Full/Backup     11h35m30s         33.792s 10.44.1.3       enp0s1.441:10.44.1.1     0     0     0
```

```
R1# sh ipv6 ospf6 neighbor
```

```
Neighbor ID     Pri    DeadTime    State/IfState         Duration I/F[State]
2.0.0.4           1    00:00:37     Full/BDR             11:36:00 enp0s1.440[DR]
3.0.0.6           1    00:00:37     Full/BDR             11:37:15 enp0s1.441[DR]
```

Les deux listes de voisins donnent une information essentielle sur l'état de protocole de routage avec le mot clé `Full`. Cet état indique que deux routeurs adjacents ont entièrement synchronisé leurs bases de données d'état de liens, permettant ainsi un échange complet des informations de routage.

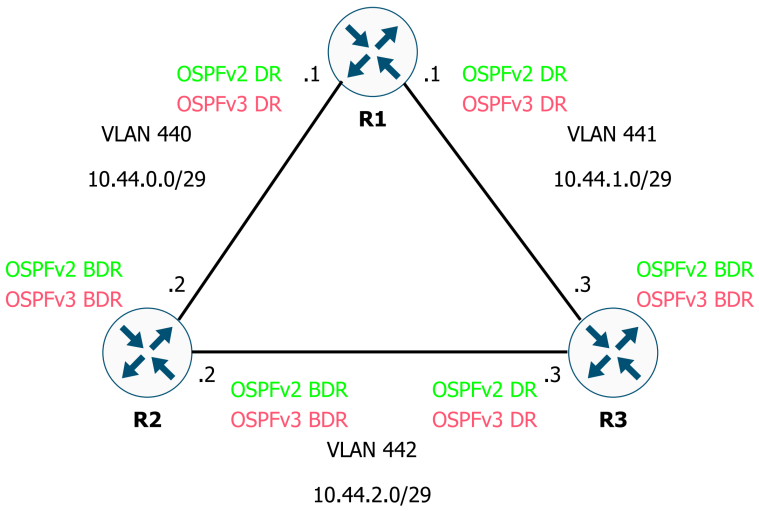
Q172. Comment identifier le rôle des différentes interfaces des routeurs pour chacun des liens du triangle de la topologie logique ?

Avertissement

La réponse à cette question suppose que les démons OSPF des trois routeurs de la topologie logique en triangle aient convergé. On doit repérer l'état `Full` pour les listes de routeurs voisins.

De plus, la réponse varie en fonction de l'ordre d'activation des démons OSPF des différents routeurs. En effet, un routeur peut être élu routeur désigné (DR) en l'absence de routeurs voisins. Cette élection n'est pas remise en cause tant qu'il n'y pas de changement d'état de lien.

À partir des résultats des questions précédentes sur les interfaces actives, il est possible de compléter le schéma de la topologie étudiée avec l'état des interfaces pour chacun des trois liens.



Sur un même réseau de diffusion, il est possible de trouver plusieurs routeurs OSPF. Établir une relation de voisinage et procéder aux échanges de bases de données topologiques entre chaque routeur revient à constituer un réseau de relations complètement maillé. À chaque nouveau calcul de topologie, ce réseau complètement maillé est inefficace. C'est la raison pour laquelle la notion de routeur référent ou ***Designated Router*** a été introduite. Lors d'un recalcul de topologie, tous les routeurs s'adressent au routeur référent qui correspond au cœur d'un réseau en topologie étoile.

Dans le contexte de la topologie triangle étudiée, il y a bien élection d'un routeur référent et d'un routeur référent de secours. Cependant, comme il n'y a que deux routeurs par domaine de diffusion ou VLAN, on ne peut pas caractériser l'utilité de cette élection.

Q173. Quels sont les réseaux IPv4 et IPv6 présents dans la table de topologie du protocole OSPF ?

On cherche a visualiser la liste des préfixes des réseaux connus des deux démons OSPF.

La liste des commandes utiles dans la console vtysh est la suivante.

show ip ospf route
show ipv6 ospf6 route

Une fois que les trois routeurs de la topologie ont convergé, chaque démon connaît les trois préfixes qui correspondent aux trois côtés du triangle. Un routeur correspond à un sommet du triangle et il doit apprendre le préfixe réseau du côté opposé via ses deux routeurs voisins.

Voici la vue depuis le routeur `R1`.

```
R1# sh ip ospf route
```

```
============ OSPF network routing table ============
N    10.44.0.0/29          [10] area: 0.0.0.0
                           directly attached to enp0s1.440
N    10.44.1.0/29          [10] area: 0.0.0.0
                           directly attached to enp0s1.441
N    10.44.2.0/29          [20] area: 0.0.0.0
                           via 10.44.0.2, enp0s1.440
                           via 10.44.1.3, enp0s1.441

============ OSPF router routing table =============

============ OSPF external routing table ===========
```

Les valeurs notées entre crochets correspondent à la métrique du lien pour joindre le réseau noté à gauche.

```
R1# sh ipv6 ospf6 route
```

Cette dernière commande ne produit aucun résultat et c'est normal ! En effet, avec le protocole IPv6 les relations de voisinage entre routeurs (adjacences) se font avec les adresses IPv6 de lien local.

Formulé autrement, les trois côtés de la topologie triangle sont des réseaux de transit qui servent à acheminer le trafic entre routeurs voisins. Il est inutile de gaspiller un préfixe réseau IPv6 pour cette tâche.

Q174. Comment sont calculées les coûts de liens pour joindre les réseaux ?

Rechercher les informations sur les calculs de coûts dans les supports de cours sur le protocole OSPF.

Depuis la première spécification du protocole OSPF, le calcul de métrique peut se faire suivant deux méthodes.

- À partir de l'expression : 10^8 / Bande_Passante_du_lien
- À partir d'une valeur `Cost` définie manuellement

Note

La valeur du numérateur (10^8) correspond à un débit de 100Mbps. À l'époque de la rédaction du standard OSPFv2, ce débit a servi de référence. Aujourd'hui, cette valeur est complètement dépassée. C'est la raison pour laquelle on adapte le calcul de métrique en changeant le coefficient du numérateur. Voir la Section 9, « Adapter de la métrique de lien au débit ».

Dans notre cas, chaque interface de routeur a un coût de `10` et une valeur de “bande passante” de `0` par défaut. Utilisez la commande d'affichage des information OSPF pour revoir ces valeurs.

show ip ospf interface
show ipv6 ospf6 interface

Pour OSPFv2, les deux premiers réseaux de la table sont joignable via un unique lien avec un coût de `10`. Le troisième réseau est joignable via deux liens ; d'où la métrique de `20`.

Pour les préfixes IPv6, aucun préfixe n'est présent sachant que les relations de voisinage entre routeurs utilisent obligatoirement les adresses de lien local appartenant au préfixe `fe80::/10`.

Les préfixes des réseaux d'hébergement de conteneurs apparaitront dès que le protocole de routage aura été activé pour les interfaces SVI.

Q175. Comment visualiser les tables de routage depuis la console vtysh ?

L'affichage demandé illustre les mécanismes de choix entre différentes solutions pour une même destination. Cet affichage est à comparer avec celui demandé à la question suivante.

La liste des commandes utiles dans la console vtysh est la suivante.

show ip route
show ipv6 route

On reprend à nouveau l'exemple du routeur `R1`.

```
R1# sh ip route
```

```
Codes: K - kernel route, C - connected, L - local, S - static,
       R - RIP, O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,
       f - OpenFabric, t - Table-Direct,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup
       t - trapped, o - offload failure

K>* 0.0.0.0/0 [0/0] via 192.168.104.129, enp0s1.360, 15:22:36
O   10.44.0.0/29 [110/10] is directly connected, enp0s1.440, weight 1, 13:49:19
C>* 10.44.0.0/29 is directly connected, enp0s1.440, 15:22:36
L>* 10.44.0.1/32 is directly connected, enp0s1.440, 15:22:36
O   10.44.1.0/29 [110/10] is directly connected, enp0s1.441, weight 1, 13:48:58
C>* 10.44.1.0/29 is directly connected, enp0s1.441, 15:22:36
L>* 10.44.1.1/32 is directly connected, enp0s1.441, 15:22:36
O>* 10.44.2.0/29 [110/20] via 10.44.0.2, enp0s1.440, weight 1, 12:30:55
  *                       via 10.44.1.3, enp0s1.441, weight 1, 12:30:55
C>* 192.168.104.128/29 is directly connected, enp0s1.360, 15:22:36
L>* 192.168.104.130/32 is directly connected, enp0s1.360, 15:22:36
```

```
R1# sh ipv6 route
```

```
Codes: K - kernel route, C - connected, L - local, S - static,
       R - RIPng, O - OSPFv3, I - IS-IS, B - BGP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,
       f - OpenFabric, t - Table-Direct,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup
       t - trapped, o - offload failure

K>* ::/0 [0/1024] via fe80:168::1, enp0s1.360 onlink, 15:23:47
K d 2001:678:3fc:168::/64 [0/512] is directly connected, enp0s1.360, 15:23:44
C>* 2001:678:3fc:168::/64 is directly connected, enp0s1.360, 15:23:46
L>* 2001:678:3fc:168::82/128 is directly connected, enp0s1.360, 15:23:46
C * fe80::/64 is directly connected, enp0s1.360, 15:23:45
C * fe80::/64 is directly connected, enp0s1.440, 15:23:46
C * fe80::/64 is directly connected, enp0s1, 15:23:46
C>* fe80::/64 is directly connected, enp0s1.441, 15:23:46
```

- Les entrées marquées avec le caractère `*` correspondent aux routes retenues et mémorisées par le sous-système réseau du noyau. Les autres entrées sont placées en réserve au cas où la solution initialement retenue serait en défaut.

- L'entrée notée `K` correspond à une route apprise depuis le sous-système réseau du noyau.

- Les entrées notées `C` correspondent à des routes pour lesquelles il existe une interface sur le routeur. Les métriques de ses routes ont la valeur `0`. Ce sont les routes les plus prioritaires.

- Les entrées notées `L` correspondent aux adresses locales du routeur dans un réseau connecté.

- Les entrées notées `O` correspondent aux routes apprises via le protocole OSPF. La métrique de ces routes se décompose en deux parties. La valeur figée à `110` définit le niveau de priorité du protocole OSPF (***Administrative Distance***) relativement aux autres protocoles de routage. Les valeurs notées après le `/` sont les métriques de liens calculées comme indiqué ci-dessus.

Q176. Comment visualiser les tables de routage au niveau système ?

Utiliser une commande usuelle de visualisation de la table de routage.

ip route ls
ip -6 route ls

Avec la commande ip, on voit apparaître les sources d'alimentation de la table de routage finale du système.

- `kernel` pour les entrées connues du sous-système réseau du noyau. Ce sont les entrées avec le caractère `C` dans la console vtysh.

- `ospf` pour les entrées apprises via le protocole de routage dynamique. Le réseau correspond au côté opposé au sommet du triangle est appris via OSPF puisque le sous-système réseau du noyau ne le connaît pas.

```
ip route ls
```

```
default via 192.168.104.129 dev enp0s1.360 proto static
10.44.0.0/29 dev enp0s1.440 proto kernel scope link src 10.44.0.1
10.44.1.0/29 dev enp0s1.441 proto kernel scope link src 10.44.1.1
10.44.2.0/29 nhid 30 proto ospf metric 20
        nexthop via 10.44.1.3 dev enp0s1.441 weight 1
        nexthop via 10.44.0.2 dev enp0s1.440 weight 1
192.168.104.128/29 dev enp0s1.360 proto kernel scope link src 192.168.104.130
```

La table de routage IPv6 ne fait apparaître aucune nouvelle entrée puisque les réseaux de conteneurs desservis par R2 et R3 n'ont pas encore été annoncés à ce stade de la configuration.

```
ip -6 route ls
```

```
2001:678:3fc:168::/64 dev enp0s1.360 proto kernel metric 256 pref medium
2001:678:3fc:168::/64 dev enp0s1.360 proto ra metric 512 expires 2591830sec pref high
fe80::/64 dev enp0s1 proto kernel metric 256 pref medium
fe80::/64 dev enp0s1.441 proto kernel metric 256 pref medium
fe80::/64 dev enp0s1.360 proto kernel metric 256 pref medium
fe80::/64 dev enp0s1.440 proto kernel metric 256 pref medium
default via fe80:168::1 dev enp0s1.360 proto static metric 1024 onlink pref medium
```

7. Publier les routes par défaut via OSPF

Dans la topologie logique étudiée, le routeur R1 dispose d'un lien montant vers l'Internet. On peut donc considérer que ce lien est la route par défaut vers tous les réseaux non connus de l'aire OSPF contenant les trois routeurs.

Il est possible de publier une route par défaut via le protocole OSPF depuis le routeur R1 vers les routeurs R2 et R3.

Pour rappel, revoir la question Q : Q169 et consulter les bases de données OSPFv2 et OSPFv3 avant la mise en place de la publication de route par défaut. On reconnaît les LSAs (***Link State Advertisement)***) de type 1 et 2 qui correspondent respectivement aux annonces de routeurs et de réseaux.

Q177. Quelle est la condition préalable à respecter pour que le routeur R1 soit en mesure de publier une route par défaut via le protocole de routage OSPF ?

À partir des tables de routage relevées dans la Section 6, « Configurer les démons OSPFv2 et OSPFv3 », repérer le routeur qui dispose d'un accès vers un réseau qui n'appartient pas à la topologie triangle.

ip route ls default
ip -6 route ls default
sh ip route kernel
sh ipv6 route kernel

Une route par défaut doit exister avant d'être injectée dans une aire OSPF. Dans notre cas, une route statique par défaut suffit à respecter la condition préalable.

Sur la maquette, on valide la présence des routes par défaut à l'aide la commande ip au niveau système.

```
ip route ls default
```

```
default via 192.168.104.129 dev enp0s1.360 proto static
```

```
ip -6 route ls default
```

```
default via fe80:168::1 dev enp0s1.360 proto static metric 1024 onlink pref medium
```

Au niveau de la console vtysh, ces même routes correspondent aux entrées marquées K pour ***kernel***.

```
R1# sh ip route kernel
```

```
Codes: K - kernel route, C - connected, L - local, S - static,
       R - RIP, O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,
       f - OpenFabric, t - Table-Direct,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup
       t - trapped, o - offload failure

K>* 0.0.0.0/0 [0/0] via 192.168.104.129, enp0s1.360, 15:40:19
```

```
R1# sh ipv6 route kernel
```

```
Codes: K - kernel route, C - connected, L - local, S - static,
       R - RIPng, O - OSPFv3, I - IS-IS, B - BGP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,
       f - OpenFabric, t - Table-Direct,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup
       t - trapped, o - offload failure

K>* ::/0 [0/1024] via fe80:168::1, enp0s1.360 onlink, 15:41:24
K d 2001:678:3fc:168::/64 [0/512] is directly connected, enp0s1.360, 15:41:2
```

Q178. Quelle est l'instruction à utiliser pour publier une route par défaut via le protocole de routage OSPFv2 ?

Parmi toutes les méthodes de redistribution de routes disponibles avec le protocole OSPFv2, il en existe une dédiée à l'injection de route par défaut dans une aire normale. Consulter le guide *FRRouting User Guide*.

Rechercher le mot clé `redistribution` dans la section ***OSPF route-map***.

L'instruction qui correspond à la redistribution de route par défaut à destination des autres routeurs de l'aire OSPF est la suivante.

default-information originate

On doit l'appliquer dans la section ***router ospf*** de la configuration du routeur `R1`.

```
R1# conf t
R1(config)# router ospf
R1(config-router)# default-information originate
R1(config-router)# ^Z
```

Une fois cette instruction exécutée, le rôle du routeur `R1` change. Il devient ***Autonomous System Boundary Router*** ou ASBR. Les bases de données sont complétées avec des LSAs de type 5.

```
R1# sh ip ospf database external
```

```
       OSPF Router with ID (1.0.0.4)

                AS External Link States

  LS age: 32
  Options: 0x2  : *|-|-|-|-|-|E|-
  LS Flags: 0xb
  LS Type: AS-external-LSA
  Link State ID: 0.0.0.0 (External Network Number)
  Advertising Router: 1.0.0.4
  LS Seq Number: 80000001
  Checksum: 0x269d
  Length: 36

  Network Mask: /0
        Metric Type: 2 (Larger than any link state path)
        TOS: 0
        Metric: 10
        Forward Address: 0.0.0.0
        External Route Tag: 0
```

On voit apparaître une nouvelle rubrique baptisée ***AS External Link States***. Ce nouveau rôle pour le routeur `R1` apparaît aussi lorsque l'on affiche l'état de l'instance de routage OSPFv2.

```
R1# sh ip ospf
```

```
 OSPF Routing Process, Router ID: 1.0.0.4
 Supports only single TOS (TOS0) routes
 This implementation conforms to RFC2328
 RFC1583Compatibility flag is disabled
 OpaqueCapability flag is disabled
 Initial SPF scheduling delay 0 millisec(s)
 Minimum hold time between consecutive SPFs 50 millisec(s)
 Maximum hold time between consecutive SPFs 5000 millisec(s)
 Hold time multiplier is currently 1
 SPF algorithm last executed 2m04s ago
 Last SPF duration 68 usecs
 SPF timer is inactive
 LSA minimum interval 5000 msecs
 LSA minimum arrival 1000 msecs
 Write Multiplier set to 20
 Refresh timer 10 secs
 Maximum multiple paths(ECMP) supported 256
 Administrative distance 110
 This router is an ASBR (injecting external routing information)
 Number of external LSA 1. Checksum Sum 0x0000269d
 Number of opaque AS LSA 0. Checksum Sum 0x00000000
 Number of areas attached to this router: 1
 All adjacency changes are logged
 Area ID: 0.0.0.0 (Backbone)
   Number of interfaces in this area: Total: 2, Active: 2
   Number of fully adjacent neighbors in this area: 2
   Area has no authentication
   SPF algorithm executed 9 times
   Number of LSA 6
   Number of router LSA 3. Checksum Sum 0x0001a8b4
   Number of network LSA 3. Checksum Sum 0x0002727f
   Number of summary LSA 0. Checksum Sum 0x00000000
   Number of ASBR summary LSA 0. Checksum Sum 0x00000000
   Number of NSSA LSA 0. Checksum Sum 0x00000000
   Number of opaque link LSA 0. Checksum Sum 0x00000000
   Number of opaque area LSA 0. Checksum Sum 0x00000000
```

Le routeur R1, est maintenant à la frontière entre deux systèmes autonomes. Il est responsable de l'émission des LSAs de type 5 à destination des autres routeurs de l'aire.

Ici, R1 possède une route statique définie au niveau système vers l'Internet. Cette route statique est redistribuée R2 et R3. Cette route apparaît comme une entrée de type E2 dans les tables de routage de ces routeurs.

L'indicateur E2 correspond au type par défaut des routes apprises par le biais de la redistribution. La métrique est un point important à considérer avec les routes de type E2. Ces routes ne présentent que le coût du chemin allant du routeur ASBR vers le réseau de destination ; ce qui ne correspond pas au coût réel du chemin à l'intérieur de l'aire OSPF.

Q179. Quelle est l'instruction à utiliser pour publier une route par défaut via le protocole de routage OSPFv3 ?

Reprendre la même démarche de la question précédente avec le protocole OSPFv3. Consulter le guide *FRRouting User Guide*.

Rechercher le mot clé `redistribution` dans la section ***OSPF6 route-map***.

L'instruction est identique pour les deux versions du protocole OSPF.

default-information originate

On doit l'appliquer dans la section ***router ospf6*** de la configuration du routeur R1.

```
R1# conf t
R1(config)# router ospf6
R1(config-ospf6)# default-information originate
R1(config-ospf6)# ^Z
```

Une fois cette instruction exécutée, le rôle du routeur R1 change. Il devient ***Autonomous System Boundary Router*** ou ASBR. Les bases de données sont complétées avec des LSAs de type 5.

```
R1# sh ipv6 ospf6 database as-external
```

```
        AS Scoped Link State Database

Type LSId            AdvRouter       Age   SeqNum                       Payload
ASE  0.0.0.1         1.0.0.6          42 80000001                          ::
```

On voit apparaître une nouvelle rubrique baptisée ***AS Scoped Link State Database***. Ce nouveau rôle pour le routeur R1 apparaît aussi lorsque l'on affiche l'état de l'instance de routage OSPFv3.

```
R1# sh ipv6 ospf6
```

```
 OSPFv3 Routing Process (0) with Router-ID 1.0.0.6
 Running 14:44:14
 LSA minimum arrival 1000 msecs
 Maximum-paths 256
 Administrative distance 110
 Initial SPF scheduling delay 0 millisec(s)
 Minimum hold time between consecutive SPFs 50 millsecond(s)
 Maximum hold time between consecutive SPFs 5000 millsecond(s)
 Hold time multiplier is currently 1
 SPF algorithm last executed 00:01:59 ago, reason R+, R-, A
 Last SPF duration 0 sec 145 usec
 SPF timer is inactive
 Number of AS scoped LSAs is 1
 Number of areas in this router is 1
 Authentication Sequence number info
  Higher sequence no 0, Lower sequence no 0
 All adjacency changes are logged

 Area 0
     Number of Area scoped LSAs is 6
     Interface attached to this area: enp0s1.440 enp0s1.441
     SPF last executed 119.712612s ago
```

Q180. Comment la publication de route par défaut apparaît elle sur les autres routeurs de la topologie triangle ?

Relevez, dans la console vtysh, la métrique de la route par défaut sur les routeurs qui n'ont pas une connexion directe vers l'Internet.

Les instructions à utiliser pour traiter cette question entrent dans la liste suivante.

show ip route A.B.C.D/MM
show ipv6 route A:B:C::D/MM
show ip ospf route
show ipv6 ospf6 route

En prenant l'exemple du routeur R2, on retrouve les informations suivantes.

- Vue de la table de routage :

```
R2# sh ip route 0.0.0.0
```

```
Routing entry for 0.0.0.0/0
  Known via "ospf", distance 110, metric 10, best
  Last update 00:07:09 ago
  * 10.44.0.1, via enp0s1.440, weight 1
```

```
R2# sh ipv6 route ::
```

```
Routing entry for ::/0
  Known via "ospf6", distance 110, metric 10, best
  Last update 00:05:09 ago
  * fe80::1b8:1, via enp0s1.440, weight 1
```

- Vue de la base de topologie OSPF :

```
R2# sh ip ospf route
```

```
============ OSPF network routing table ============
N    10.44.0.0/29          [10] area: 0.0.0.0
                           directly attached to enp0s1.440
N    10.44.1.0/29          [20] area: 0.0.0.0
                           via 10.44.0.1, enp0s1.440
                           via 10.44.2.3, enp0s1.442
N    10.44.2.0/29          [10] area: 0.0.0.0
                           directly attached to enp0s1.442

============ OSPF router routing table =============
R    1.0.0.4               [10] area: 0.0.0.0, ASBR
                           via 10.44.0.1, enp0s1.440

============ OSPF external routing table ===========
N E2 0.0.0.0/0             [10/10] tag: 0
                           via 10.44.0.1, enp0s1.440
```

```
R2# sh ipv6 ospf6 route
```

```
*N E2 ::/0                            fe80::1b8:1             enp0s1.440 00:07:18
```

Avec les copies d'écran ci-dessus, on vérifie bien que les routes par défaut ont été apprises via le protocole OSPF.

Q181. Comment assurer la traduction d'adresses source sur l'interface de sortie du routeur `R1` vers l'Internet ?

Reprendre la section ***Activation de la traduction d'adresses*** du support *Routage inter-VLAN et protocole PPPoE*.

Dans le cas de la maquette, c'est l'interface `enp0s1.360` qui assure l'interconnexion avec l'Internet. Voici une copie de l'instruction d'ajout de la règle de traduction des adresses sources et de sa sauvegarde pour IPv4 et IPv6.

- Installer le paquet `nftables`.

```
sudo apt -y install nftables
```

- Créer le fichier de règles `/etc/nftables.conf`.

```
cat << EOF | sudo tee /etc/nftables.conf
#!/usr/sbin/nft -f

flush ruleset

table inet nat {
    chain postrouting {
        type nat hook postrouting priority 100;
        oifname "enp0s1.360" masquerade
    }
}
EOF
```

- Appliquer la règle de traduction d'adresses sources en activant le service.

```
sudo systemctl enable --now nftables.service
```

- Afficher la liste des règles de filtrage actives.

```
sudo nft list ruleset
```

Pour terminer, on peut lancer des tests de communication ICMP IPv4 depuis les routeurs `R2` ou `R3`.

```
etu@R2:~$ ping -qc2 9.9.9.9
PING 9.9.9.9 (9.9.9.9) 56(84) bytes of data.

--- 9.9.9.9 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 28.604/28.676/28.748/0.072 ms
```

Important

À ce stade de la configuration, les tests IPv6 sont impossible à réaliser dans la mesure où toutes les adresses présentes sont des adresses de lien local.

8. Ajouter un réseau d'hébergement à chaque routeur

Dans cette section, on créé un réseau de conteneur attaché à chaque routeur de la topologie triangle. L'objectif est de peupler les tables de routage en ajoutant un lien OSPF au delà des routeurs.

Une fois ces réseaux en place, il est possible de réaliser des tests de connectivité entre conteneurs et d'optimiser les métriques.

Cette partie reprend le contenu de la section ***Réseau d'hébergement de conteneurs*** du support de travaux pratiques *Routage inter-VLAN et protocole PPPoE*.

Q182. Quels sont les paquets à installer pour mettre en place un commutateur d'accès et le gestionnaire de conteneurs ***Incus*** ?

Consulter la section ***Réseau d'hébergement de conteneurs*** du support *Routage inter-VLAN et protocole PPPoE*.

Note

Pour les routeurs `R2` et `R3`, il faut rétablir l'accès temporaire à Internet pour installer les paquets demandés.

openvswitch-switch

Création d'un commutateur de raccordement des conteneurs et d'une interface virtuelle commutée

dnsmasq

Adressage automatique des conteneurs

incus

Gestion des conteneurs

```
sudo apt -y install openvswitch-switch dnsmasq incus
```

Q183. Quelles sont les modifications à apporter au fichier de déclaration YAML `/etc/netplan/enp0s1.yaml` pour créer le commutateur `asw-host` ?

Voici un extrait du fichier `/etc/netplan/enp0s1.yaml` avec les instructions de création du commutateur `asw-host`.

```
    openvswitch: {}

  bridges:
    asw-host:
      openvswitch: {}
```

Q184. Comment ajouter une nouvelle interface virtuelle commutée (***Switched Virtual Interface***) qui servira de passerelle par défaut pour tous les conteneurs hébergés ?

Rechercher dans la documentation Netplan des exemples de déclarations d'interfaces de type SVI appartenant à des VLANs.

Voici une copie complète du fichier `/etc/netplan/enp0s1.yaml` du routeur `R2` auquel on ajouté la déclaration d'une interface `vlan20` avec les adresses IPv4 et IPv6 conformes au contexte de la maquette utilisée pour la rédaction de ce document.

```
network:
    version: 2
    ethernets:
      enp0s1:
        dhcp4: false
        dhcp6: false
        accept-ra: false
        nameservers:
          addresses:
            - 172.16.0.2
            - 2001:678:3fc:3::2

    openvswitch: {}

    bridges:
      asw-host:
        openvswitch: {}

    vlans:
#       # Temporary Internet access
#      enp0s1.52:
#        id: 52
#        link: enp0s1
#        dhcp4: true
#        dhcp6: false
#        accept-ra: true
      # R2 -> R1
      enp0s1.440:
        id: 440
        link: enp0s1
        addresses:
          - 10.44.0.2/29
          - fe80::1b8:2/64
      # R2 -> R3
      enp0s1.442:
        id: 442
        link: enp0s1
        addresses:
          - 10.44.2.2/29
          - fe80::1ba:2/64
      vlan20:
        id: 20
        link: asw-host
        addresses:
          - 10.20.0.1/24
          - fd14:ca46:3864:14::1/64
```

Avec ce fichier de déclaration l'interface d'accès temporaire est commentée et ne servira plus dans la suite des manipulations. À partir de la question suivante, tout le trafic des routeurs R2 et R3 transite par le routeur R1.

Q185. Comment publier le nouveau réseau d'hébergement via OSPF ?

Revoir la question Q : Q167 sur l'activation OSPF au niveau de chaque interface.

On active le protocole OSPF pour l'interface commutée virtuelle de chacun des trois routeurs. Voici un exemple pour le routeur R2.

```
R2# conf t
R2(config)# int vlan20
R2(config-if)# ip ospf area 0
R2(config-if)# ipv6 ospf6 area 0
R2(config-if)# ip ospf passive
R2(config-if)# ipv6 ospf6 passive
R2(config-if)# ^Z
```

Une fois cette opération réalisée, le routage dynamique est complet et le trafic des routeurs R2 et R3 est acheminé via R1. On peut lancer des tests ICMP.

```
ping -qc2 9.9.9.9
```

```
PING 9.9.9.9 (9.9.9.9) 56(84) bytes of data.

--- 9.9.9.9 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 28.502/29.014/29.526/0.512 ms
```

```
ping -qc2 2620:fe::fe
```

```
PING 2620:fe::fe (2620:fe::fe) 56 data bytes

--- 2620:fe::fe ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 40.675/47.384/54.094/6.709 ms
```

Q186. Comment administrer et initialiser le gestionnaire de conteneurs ***Incus*** ?

Consulter la section ***Réseau d'hébergement de conteneurs***.

Pour commencer, l'utilisateur normal doit appartenir aux deux groupes système `incus` et `incus-admin`. On reprend la démarche précédente.

```
for grp in incus incus-admin
do
    sudo adduser etu $grp
done
```

Comme l'affectation de groupe se joue à la connexion, il faut se déconnecter/reconnecter pour rendre l'attribution effective.

```
groups
```

```
etu adm sudo users frrvty frr incus-admin incus
```

Pour l'initialisation du gestionnaire de conteneur, on utilise un fichier de déclaration préparé en amont. Voici une copie du fichier `incus.yaml` pour le routeur `R2`.

```
config: {}
networks: []
storage_pools:
- config: {}
  description: ""
  name: default
  driver: dir
profiles:
- config: {}
  description: ""
  devices:
    eth0:
      name: eth0
      nictype: bridged
      parent: asw-host
      type: nic
      vlan: 20
    root:
      path: /
      pool: default
      type: disk
  name: default
projects: []
cluster: null
```

L'initialisation se fait à partir de ce fichier.

```
cat incus.yaml | incus admin init --preseed
```

On peut afficher le résultat de cette initialisation du gestionnaire de conteneurs avec l'instruction suivante.

```
incus profile show default
```

En cas d'erreur, il est aussi possible d'éditer le profil par défaut.

```
incus profile edit default
```

> **Important**
>
> Les opérations de cette question doivent être réalisées sur les trois routeurs en adaptant les numéros de VLAN de réseau d'hébergement.

Q187. Comment mettre en place l'adressage automatique IPv4 et IPv6 dans le réseau d'hébergement ?

Consulter la section ***Adressage automatique dans le réseau d'hébergement*** du document *Routage inter-VLAN et protocole PPPoE*

On créé un fichier de configuration pour le service dnsmasq en l'adaptant au contexte du plan d'adressage de chaque routeur. Voici un exemple pour le routeur `R1` de la maquette.

```
cat << EOF | sudo tee /etc/dnsmasq.conf
# Specify Container VLAN interface
interface=vlan10

# Enable DHCPv4 on Container VLAN
dhcp-range=10.10.0.20,10.10.0.200,3h

# Enable IPv6 router advertisements
enable-ra

# Enable SLAAC
dhcp-range=::,constructor:vlan10,ra-names,slaac

# Optional: Specify DNS servers
dhcp-option=option:dns-server,172.16.0.2,9.9.9.9
dhcp-option=option6:dns-server,[2001:678:3fc:3::2],[2620:fe::fe]

# Avoid DNS listen port conflict between dnsmasq and systemd-resolved
port=0
EOF
```

Une fois le fichier de configuration en place, on peut relancer le service et afficher son état.

```
sudo systemctl restart dnsmasq
systemctl status dnsmasq
```

Important

Une fois encore, les opérations de cette question doivent être réalisées sur les trois routeurs en adaptant le nom de l'interface et les adresses IPv4 du réseau d'hébergement.

Q188. Comment créer 3 conteneurs avec un adressage ?

Consulter la section ***Configuration et lancement des conteneurs*** du document *Routage inter-VLAN et protocole PPPoE*.

On lance la création des 3 conteneurs demandés avec une boucle.

```
for i in {0..2}; do incus launch images:debian/trixie c$i; done
```

```
incus ls
```

```
+------+---------+---------------------+-----------------------------------------------+-----------+-----------+
| NAME |  STATE  |        IPV4         |                     IPV6                      |   TYPE    | SNAPSHOTS |
+------+---------+---------------------+-----------------------------------------------+-----------+-----------+
| c0   | RUNNING | 10.20.0.33 (eth0)   | fd14:ca46:3864:14:216:3eff:fe2c:c5b9 (eth0) | CONTAINER | 0         |
+------+---------+---------------------+-----------------------------------------------+-----------+-----------+
| c1   | RUNNING | 10.20.0.87 (eth0)   | fd14:ca46:3864:14:216:3eff:fe76:f0d0 (eth0) | CONTAINER | 0         |
+------+---------+---------------------+-----------------------------------------------+-----------+-----------+
| c2   | RUNNING | 10.20.0.153 (eth0)  | fd14:ca46:3864:14:216:3eff:fe2c:d169 (eth0) | CONTAINER | 0         |
+------+---------+---------------------+-----------------------------------------------+-----------+-----------+
```

Q189. Comment vérifier la publication des réseaux de conteneurs dans l'aire OSPF ?

On vérifie que l'interface commutée virtuelle (***Switched Virtual Interface***)est bien active pour chacune des deux versions du protocole de routage dynamique.

Voici un exemple de configuration pour le routeur R2.

```
R2# sh ip ospf interface vlan20
```

```
vlan20 is up
  ifindex 6, MTU 1500 bytes, BW 0 Mbit <UP,LOWER_UP,BROADCAST,RUNNING,MULTICAST>
  Internet Address 10.20.0.1/24, Broadcast 10.20.0.255, Area 0.0.0.0
  MTU mismatch detection: enabled
  Router ID 2.0.0.4, Network Type BROADCAST, Cost: 10
  Transmit Delay is 1 sec, State DR, Priority 1
  Designated Router (ID) 2.0.0.4 Interface Address 10.20.0.1/24
  No backup designated router on this network
  Multicast group memberships: <None>
  Timer intervals configured, Hello 10s, Dead 40s, Wait 40s, Retransmit 5
    No Hellos (Passive interface)
  Neighbor Count is 0, Adjacent neighbor count is 0
  Graceful Restart hello delay: 10s
```

```
R2# sh ipv6 ospf6 interface vlan20
```

```
vlan20 is up, type BROADCAST
  Interface ID: 6
  Internet Address:
    inet : 10.20.0.1/24
    inet6: fd14:ca46:3864:14::1/64
    inet6: fe80::2865:4aff:fe35:fdd0/64
  Instance ID 0, Interface MTU 1500 (autodetect: 1500)
  MTU mismatch detection: enabled
  Area ID 0.0.0.0, Cost 10
  State DR, Transmit Delay 1 sec, Priority 1
  Timer intervals configured:
   No Hellos (Passive interface)
  DR: 2.0.0.4 BDR: 0.0.0.0
  Number of I/F scoped LSAs is 1
    0 Pending LSAs for LSUpdate in Time 00:00:00 [thread off]
    0 Pending LSAs for LSAck in Time 00:00:00 [thread off]
  Graceful Restart hello delay: 10s
  Authentication Trailer is disabled
```

Q190. Comment valider l'accès à l'Internet depuis les conteneurs ?

On reprend la liste des tests ICMP pour valider l'acheminement du trafic au niveau de la couche réseau et on lance une mise à jour du catalogue des paquets pour valider la résolution des noms.

À partir du routeur R2, on utilise les séquences suivantes :

- Tests ICMP au niveau réseau :

```
for i in {0..2}
do
    echo ">>>>>>>>>>>>>>>>>> c$i"
    incus exec c$i -- ping -qc2 9.9.9.9
done
```

```
for i in {0..2}
do
    echo ">>>>>>>>>>>>>>>>>> c$i"
    incus exec c$i -- ping -qc2 2620:fe::fe
done
```

- Tests au niveau application avec résolution des noms :

```
for i in {0..2}
do
    echo ">>>>>>>>>>>>>>>>>> c$i"
    incus exec c$i -- apt update
    incus exec c$i -- apt -y full-upgrade
done
```

Q191. Comment vérifier que tous les réseaux de conteneurs sont bien visibles depuis les routeurs de la topologie triangle ?

Afficher les tables de routage IPv4 et IPv6 de l'un des trois routeurs et vérifier la présence des trois réseaux de conteneurs.

Voici le résultat obtenu depuis la console du routeur R3. On identifie les réseaux d'hébergement des routeurs R1 et R2 pour lesquels on trouve des entrées marquées O>*.

```
R3# sh ip route
```

```
Codes: K - kernel route, C - connected, L - local, S - static,
       R - RIP, O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,
       f - OpenFabric, t - Table-Direct,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup
       t - trapped, o - offload failure

O>* 0.0.0.0/0 [110/10] via 10.44.1.1, enp0s1.441, weight 1, 02:12:47
O>* 10.10.0.0/24 [110/20] via 10.44.1.1, enp0s1.441, weight 1, 02:12:48
O>* 10.20.0.0/24 [110/20] via 10.44.2.2, enp0s1.442, weight 1, 00:11:37
O   10.30.0.0/24 [110/10] is directly connected, vlan30, weight 1, 02:12:36
C>* 10.30.0.0/24 is directly connected, vlan30, 02:12:54
L>* 10.30.0.1/32 is directly connected, vlan30, 02:12:54
O>* 10.44.0.0/29 [110/20] via 10.44.1.1, enp0s1.441, weight 1, 00:11:37
  *                       via 10.44.2.2, enp0s1.442, weight 1, 00:11:37
O   10.44.1.0/29 [110/10] is directly connected, enp0s1.441, weight 1, 02:12:55
C>* 10.44.1.0/29 is directly connected, enp0s1.441, 02:12:55
L>* 10.44.1.3/32 is directly connected, enp0s1.441, 02:12:55
O   10.44.2.0/29 [110/10] is directly connected, enp0s1.442, weight 1, 01:18:48
C>* 10.44.2.0/29 is directly connected, enp0s1.442, 02:12:55
L>* 10.44.2.3/32 is directly connected, enp0s1.442, 02:12:55
```

```
R3# sh ipv6 route
```

```
Codes: K - kernel route, C - connected, L - local, S - static,
       R - RIPng, O - OSPFv3, I - IS-IS, B - BGP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,
       f - OpenFabric, t - Table-Direct,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup
       t - trapped, o - offload failure

O>* ::/0 [110/10] via fe80::1b9:1, enp0s1.441, weight 1, 00:14:55
O>* fd14:ca46:3864:a::/64 [110/20] via fe80::1b9:1, enp0s1.441, weight 1, 02:21:04
O>* fd14:ca46:3864:14::/64 [110/20] via fe80::baad:caff:fefe:6, enp0s1.442, weight 1, 00:14:56
O   fd14:ca46:3864:1e::/64 [110/10] is directly connected, vlan30, weight 1, 02:14:55
C>* fd14:ca46:3864:1e::/64 is directly connected, vlan30, 02:15:27
L>* fd14:ca46:3864:1e::1/128 is directly connected, vlan30, 02:15:27
C * fe80::/64 is directly connected, vlan30, 02:15:28
C * fe80::/64 is directly connected, asw-host, 02:15:29
C * fe80::/64 is directly connected, enp0s1.52, 02:19:13
C * fe80::/64 is directly connected, enp0s1.441, 22:24:07
C * fe80::/64 is directly connected, enp0s1.442, 22:24:08
C>* fe80::/64 is directly connected, enp0s1, 22:24:08
```

Si on reprend, la même démarche sur chaque routeur, on doit trouver deux entrées OSPF relatives aux réseaux d'hébergement raccordés aux deux “sommets du triangle” dans les tables de routage.

9. Adapter de la métrique de lien au débit

Par défaut, le calcul de métrique du le protocole OSPF se fait à partir d'un coût fixe de 10. Dans cette section, on revient au calcul du quotient entre le débit binaire de référence et le débit binaire de l'interface.

À l'origine, le calcul de ce quotient utilise la formule 10^8 / bande passante du lien.

Cette règle a été établie à une époque où l'utilisation d'un lien à 100Mbps était considéré comme une situation futuriste. Aujourd'hui, les liens à 100Mbps sont dépassés.

> **Important**
>
> Pour que les calculs de métriques soient cohérents sur l'ensemble de la topologie triangle, il est essentiel d'appliquer les modifications de configuration sur les trois routeurs.

Q192. Quelle est l'instruction à utiliser pour que le calcul de métrique OSPF se fasse sur la base d'un débit de lien à 40Gbps ?

Rechercher le mot clé ***reference*** dans l'index des instructions de configuration. Consulter la documentation *FRRouting User Guide*.

C'est l'instruction auto-cost reference-bandwidth qui permet de fixer une nouvelle référence de coût de lien en Mbps.

```
R1# conf t
R1(config)# router ospf
R1(config-router)# auto-cost reference-bandwidth ?
  (1-4294967)  The reference bandwidth in terms of Mbits per second
R1(config-router)# auto-cost reference-bandwidth 400000
R1(config-router)# ^Z
```

```
R1# conf t
R1(config)# router ospf6
R1(config-ospf6)# auto-cost reference-bandwidth ?
  (1-4294967)  The reference bandwidth in terms of Mbits per second
R1(config-ospf6)# auto-cost reference-bandwidth 400000
R1(config-ospf6)# ^Z
```

En appliquant les mêmes instructions sur les trois routeurs de la topologie, on fixe le numérateur du quotient de calcul des métriques de routes.

Q193. Comment modifier le débit binaire d'un lien à 10Gbps ?

> **Note**
>
> Sur une machine physique avec des composants réseau matériels, le débit d'un lien est directement extrait des paramètres du composant de l'interface connectée au lien.
>
> Dans le cas d'interfaces qui n'ont aucune réalité physique, ce débit peut être attribué arbitrairement par configuration. On doit rechercher dans les options de la console vtysh le moyen d'attribuer une indication de débit aux sous-interfaces de VLANs.

En affichant les paramètres des interfaces des routeurs virtuels, on constate que le paramètre `speed` est à zéro.

```
R1# sh interface vlan10
```

```
Interface vlan10 is up, line protocol is up
  Link ups:       1    last: 2024/11/03 14:26:00.66
  Link downs:     2    last: 2024/11/03 14:26:00.65
  vrf: default
  index 8 metric 0 mtu 1500 speed 0 txqlen 1000
  flags: <UP,LOWER_UP,BROADCAST,RUNNING,MULTICAST>
  Type: Ethernet
  HWaddr: da:35:df:7e:5d:42
  inet 10.10.0.1/24
  inet6 fd14:ca46:3864:a::1/64
  inet6 fe80::d835:dfff:fe7e:5d42/64
  Interface Type Other
  Interface Slave Type None
  protodown: off
```

Pour assurer un calcul correct de métrique, on doit fixer manuellement le débit binaire des interfaces dans la configuration `frr`.

Q194. Comment peut-on identifier le débit d'un lien dans la configuration OSPF ?

Visualiser les paramètres des interfaces réseau depuis la console vtysh. Les commandes suivantes peuvent être lancées sur chacun des trois routeurs.

show ip ospf interface
show ipv6 ospf6 interface

Voici le résultat obtenu sur le routeur `R1`.

```
R1# conf t
R3(config)# int vlan10
R3(config-if)# bandwidth 10000
R3(config-if)# ^Z
```

```
R1# sh ip ospf interface vlan10
```

```
vlan10 is up
  ifindex 8, MTU 1500 bytes, BW 10000 Mbit <UP,LOWER_UP,BROADCAST,RUNNING,MULTICAST>
  Internet Address 10.10.0.1/24, Broadcast 10.10.0.255, Area 0.0.0.0
  MTU mismatch detection: enabled
  Router ID 1.0.0.4, Network Type BROADCAST, Cost: 4
  Transmit Delay is 1 sec, State DR, Priority 1
  Designated Router (ID) 1.0.0.4 Interface Address 10.10.0.1/24
  No backup designated router on this network
  Multicast group memberships: <None>
  Timer intervals configured, Hello 10s, Dead 40s, Wait 40s, Retransmit 5
    No Hellos (Passive interface)
  Neighbor Count is 0, Adjacent neighbor count is 0
  Graceful Restart hello delay: 10s
```

Q195. Quel est le coût d'accès au réseau de conteneurs attaché au routeur `R3` depuis le routeur `R2` ? Justifier la valeur de métrique obtenue.

À partir des informations de routage sur `R2`, faire la somme des métriques de chaque lien entre les deux extrémités en communication.

Il est possible d'obtenir les informations de calcul de métrique à partir de l'affichage d'une route particulière à la console vtysh.

```
R1# sh ip route 10.20.0.0/24
```

```
Routing entry for 10.20.0.0/24
  Known via "ospf", distance 110, metric 8, best
  Last update 00:21:32 ago
  * 10.44.0.2, via enp0s1.440, weight 1
```

Dans l'exemple ci-dessus, la métrique d'accès au réseau `10.30.0.0/24` correspond à la somme des métriques de deux liens à 10Gbps. La somme donne une métrique de 40/10 + 40/10 = 8.

On obtient un résultat identique avec la table de routage IPv6.

```
R1# sh ipv6 route fd14:ca46:3864:1e::/64
```

```
Routing entry for fd14:ca46:3864:1e::/64
  Known via "ospf6", distance 110, metric 8, best
  Last update 00:25:44 ago
  * fe80::1b9:3, via enp0s1.441, weight 1
```

10. Sauvegarder les fichiers de configuration

Les commandes utiles pour l'affichage et la sauvegarde sont les suivantes.

show run
copy run start

Une fois sorti de la console `frr`, la configuration unifiée des démons est stockée dans le fichier `/etc/frr/frr.conf`.

Voici une copie des fichiers de configuration des trois routeurs une fois toutes les manipulations réalisées.

- Routeur `R1`.

```
frr version 10.1.1
frr defaults traditional
hostname R1
log syslog informational
service integrated-vtysh-config
!
interface enp0s1.440
 bandwidth 10000
 ip ospf area 0
 ipv6 ospf6 area 0
exit
!
interface enp0s1.441
 bandwidth 10000
 ip ospf area 0
 ipv6 ospf6 area 0
exit
!
interface vlan10
 bandwidth 10000
 ip ospf area 0
 ip ospf passive
 ipv6 ospf6 area 0
 ipv6 ospf6 passive
exit
!
router ospf
 ospf router-id 1.0.0.4
 log-adjacency-changes detail
 auto-cost reference-bandwidth 40000
 default-information originate
exit
!
router ospf6
 ospf6 router-id 1.0.0.6
 log-adjacency-changes detail
 auto-cost reference-bandwidth 40000
 default-information originate
exit
!
end
```

- Routeur `R2`.

```
frr version 10.1.1
frr defaults traditional
hostname R2
log syslog informational
service integrated-vtysh-config
!
interface enp0s1.440
 bandwidth 10000
 ip ospf area 0
 ipv6 ospf6 area 0
exit
!
interface enp0s1.442
 bandwidth 10000
 ip ospf area 0
 ipv6 ospf6 area 0
exit
!
interface vlan20
 bandwidth 10000
 ip ospf area 0
 ip ospf passive
 ipv6 ospf6 area 0
 ipv6 ospf6 passive
exit
!
router ospf
 ospf router-id 2.0.0.4
 log-adjacency-changes detail
 auto-cost reference-bandwidth 40000
exit
!
router ospf6
 ospf6 router-id 2.0.0.4
 log-adjacency-changes detail
 auto-cost reference-bandwidth 40000
exit
```

```
!
end
```

- Routeur R3.

```
frr version 10.1.1
frr defaults traditional
hostname R3
log syslog informational
service integrated-vtysh-config
!
interface enp0s1.441
 bandwidth 10000
 ip ospf area 0
 ipv6 ospf6 area 0
exit
!
interface enp0s1.442
 bandwidth 10000
 ip ospf area 0
 ipv6 ospf6 area 0
exit
!
interface vlan30
 bandwidth 10000
 ip ospf area 0
 ip ospf passive
 ipv6 ospf6 area 0
 ipv6 ospf6 passive
exit
!
router ospf
 ospf router-id 3.0.0.4
 log-adjacency-changes detail
 auto-cost reference-bandwidth 40000
exit
!
router ospf6
 ospf6 router-id 3.0.0.6
 log-adjacency-changes detail
 auto-cost reference-bandwidth 40000
exit
!
end
```

11. Pour conclure...

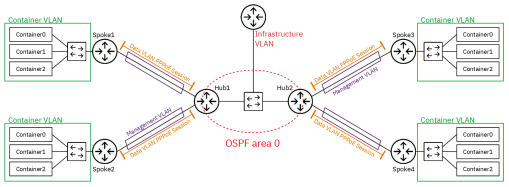
Ce document fournit une introduction détaillée à la configuration du protocole de routage dynamique OSPF avec FRRouting sur des routeurs virtuels Linux. Il couvre la préparation des systèmes, l'installation et l'activation des démons OSPF, ainsi que la validation des communications entre routeurs.

Les manipulations proposées permettent de mettre en place une topologie réseau en triangle et de configurer pas à pas le routage OSPF pour IPv4 et IPv6. Ce guide pratique offre une base solide pour comprendre et implémenter le routage dynamique OSPF dans un environnement virtuel.

Synthèse sur l'interconnexion LAN/WAN

https://www.inetdoc.net

Résumé



L'objectif de ce dernier document de la série de travaux pratiques est de faire la synthèse sur l'interconnexion de réseaux locaux (LAN) et de réseaux étendus (WAN). Côté réseaux étendus, on retrouve les sessions PPPoE vers chaque site distant avec son réseau d'extrémité avec un l'hébergement de services représentés par les conteneurs Incus. Côté réseaux locaux, les routeurs *Hub* échangent leurs routes avec le protocole de routage dynamique OSPF. Ces routeurs constituent ainsi un réseau de “collecte”. Que l'on soit dans le domaine LAN ou WAN, on fait un usage massif des VLANs.

Table des matières

1. Objectifs

Après avoir réalisé les manipulations présentées dans ce document, les étudiants seront capables de :

1. Configurer une topologie réseau Hub & Spoke complexe combinant des interconnexions LAN et WAN.
2. Mettre en place et configurer des sessions PPPoE entre des routeurs Hub et Spoke.
3. Implémenter le routage dynamique OSPF (v2 et v3) entre les routeurs Hub.
4. Configurer la redistribution de routes entre différents protocoles : interfaces connectées, noyau et OSPF.
5. Déployer et configurer des conteneurs pour simuler des réseaux d'hébergement distants.

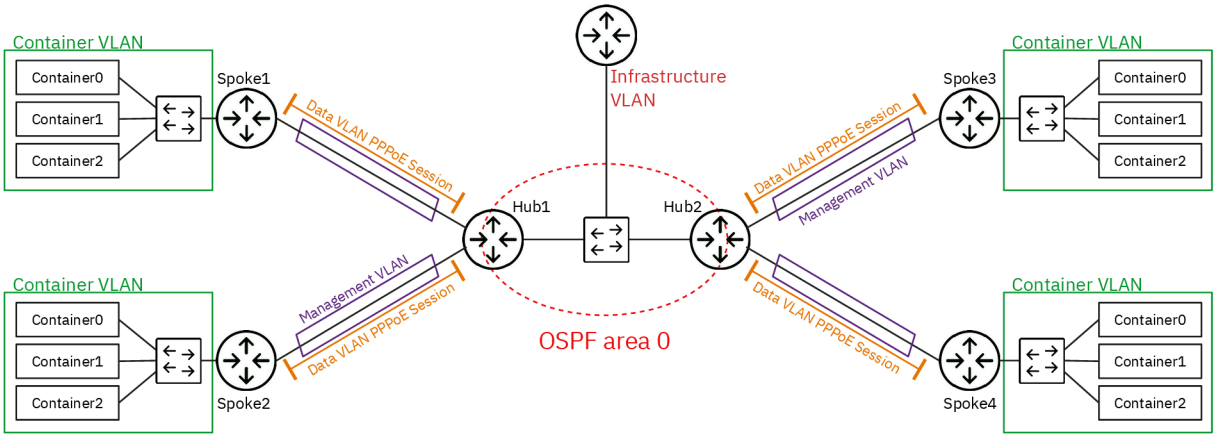
2. Topologie réseau & plan d'adressage

La topologie logique globale se présente comme une association de deux topologies triangulaires *Hub & Spoke* dans laquelle les routeurs *Hub* échangent leurs routes via le protocole de routage dynamique OSPF et fournissent l'accès Internet aux sites distants WAN.Elle couvre les aspects suivants :

- Topologie *Hub & Spoke* avec PPPoE pour l'interconnexion WAN
- Filtrage réseau avec netfilter/nftables

- Routage dynamique avec OSPF

L'objectif est de mettre en place une architecture réseau complète et sécurisée, combinant des techniques d'interconnexion WAN, de filtrage et de routage dynamique.



Routeur Hub

Le routeur ***Hub*** assure l'interconnexion entre les sites distants et le réseau de collecte. Il gère les sessions PPPoE avec les routeurs ***Spoke*** et annonce les routes vers les réseaux distants via le protocole OSPF aux autres routeurs du réseau de collecte.

Note

Les routeurs ***Hub*** partagent la même passerelle vers l'Internet.

Routeur Spoke

Le routeur ***Spoke*** assure l'interconnexion entre son réseau d'extrémité représenté par les conteneurs et le routeur ***Hub*** auprès duquel il s'authentifie pour ouvrir une session PPPoE qui lui permet de joindre tous les autres réseaux dont l'Internet.

Voici un exemple de plan d'adressage associé à la topologie ci-dessus. Il est utilisé pour la maquette de démonstration du fonctionnement de l'interconnexion réseau.

Tableau 1. Plan d'adressage de la maquette « Synthèse sur l'interconnexion LAN/WAN »

Rôle	VLAN	Type	Liaison	Adresse/Authentification
Hub1	120	hosting	-> Internet	`172.20.120.2/23` `2001:678:3fc:78::2/64`
	470	collecte	`OSPFv2 id 1.0.0.4` `OSPFv3 id 1.0.0.6`	`172.20.70.1/29`
	471	mgmt lien local	-> Spoke1	`fe80:1d7::1/64`
	472	data point à point	-> Spoke1	`10.47.2.1:10.47.2.2`
	473	mgmt lien local	-> Spoke2	`fe80:1d9::1/64`
	474	data point à point	-> Spoke2	`10.47.4.1:10.47.4.2`
Spoke1	471	mgmt lien local	-> Hub1	`fe80:1d7::2/64`
	472	authentification	-> Hub1	`spoke1 / 0r4ng3.1`
	1	conteneurs	-	`10.0.1.1/24` `fda0:7a62:1::1/64`

Rôle	VLAN	Type	Liaison	Adresse/Authentification
Spoke2	473	mgmt lien local	-> Hub1	`fe80:1d9::2/64`
	474	authentification	-> Hub1	`spoke2 / 0r4ng3.2`
	2	conteneurs	-	`10.0.2.1/24` `fda0:7a62:2::1/64`
Hub2	120	hosting	-> Internet	`172.20.120.3/23` `2001:678:3fc:78::3/64`
	470	collecte	`OSPFv2 id 2.0.0.4` `OSPFv3 id 2.0.0.6`	`172.20.70.2/29`
	475	mgmt lien local	-> Spoke3	`fe80:1db::1/64`
	476	data point à point	-> Spoke3	`10.47.6.1:10.47.6.2`
	477	mgmt lien local	-> Spoke4	`fe80:1dd::1/64`
	478	data point à point	-> Spoke4	`10.47.8.1:10.47.8.2`
Spoke3	475	mgmt lien local	-> Hub2	`fe80:1db::2/64`
	476	authentification	-> Hub2	`spoke3 / 0r4ng3.3`
	3	conteneurs	-	`10.0.3.1/24` `fda0:7a62:3::1/64`
Spoke4	477	mgmt lien local	-> Hub2	`fe80:1dd::2/64`
	478	authentification	-> Hub2	`spoke4 / 0r4ng3.4`
	4	conteneurs	-	`10.0.4.1/24` `fda0:7a62:4::1/64`

3. Configurer les Routeurs Hub

Dans cette section, on s'intéresse aux rôles et fonctionnalités suivantes :

- Gérer les sessions PPPoE avec les *Spoke*
- Filtrer et traduire les adresses sources (NAT) pour le trafic sortant vers l'Internet
- Annoncer les routes vers les réseaux distants via OSPF

3.1. Configurer les serveurs PPPoE

Les tâches à réaliser sont :

- Activer le routage
- Activer la traduction des adresses sources vers l'Internet
- Installer le paquet pppoe
- Paramétrer le fichier `/etc/ppp/pppoe-server-options`.
- Créer les unités systemd pour le lancement des serveurs PPPoE

Voici un extrait des questions du support de travaux pratiques *Topologie Hub & Spoke avec le protocole PPPoE*.

Q196. Comment activer le routage dans le sous-système réseau du noyau Linux ?

Utiliser la commande sysctl pour effectuer des recherches et identifier les paramètres utiles. Par exemple :

```
sudo sysctl -a -r ".*forward.*"
```

Il est dorénavant recommandé de créer un fichier de configuration spécifique par fonction. C'est la raison pour laquelle on crée un nouveau fichier `/etc/sysctl.d/10-routing.conf`.

```
cat << EOF | sudo tee /etc/sysctl.d/10-routing.conf
net.ipv4.ip_forward=1
net.ipv6.conf.all.forwarding=1
net.ipv4.conf.all.log_martians=1
EOF
```

Attention ! N'oubliez pas d'appliquer les nouvelles valeurs des paramètres de configuration.

```
sudo sysctl --system
```

Q197. Comment assurer la traduction d'adresses sources pour tous les flux réseaux sortants sur le réseau d'infrastructure (VLAN rouge) ?

Rechercher dans des exemples de configuration `nftables` avec la fonction `MASQUERADE`.

Voici un exemple de création du fichier `/etc/nftables.conf` avec le jeu d'instructions qui assure la traduction d'adresses sources pour IPv4 et IPv6.

```
cat << 'EOF' | sudo tee /etc/nftables.conf
#!/usr/sbin/nft -f

flush ruleset

# Define variables
define RED_VLAN = enp0s1.360

table inet nat {
    chain postrouting {
        type nat hook postrouting priority 100;
        oifname $RED_VLAN counter packets 0 bytes 0 masquerade
    }
}
EOF
```

Avertissement

Il faut impérativement changer le nom d'interface en utilisant le numéro de VLAN attribué dans le plan d'adressage des travaux pratiques.

La création de ce fichier de règles n'est pas suffisante. Il faut appliquer les règles contenues dans le fichier.

```
sudo nft -f /etc/nftables.conf
```

Il faut aussi activer ce service pour assurer le chargement automatique des règles de filtrage au démarrage.

```
sudo systemctl enable --now nftables.service
sudo systemctl status nftables.service
```

Q198. Quel paquet spécifique à la gestion du dialogue PPPoE à installer sur le routeur *Hub* ?

Rechercher dans le catalogue des paquets, la référence pppoe.

```
apt search ^pppoe
```

Le résultat de la commande apt show pppoe montre que c'est bien le paquet `pppoe` qui répond au besoin. On peut donc l'installer.

```
sudo apt -y install pppoe
```

Q199. Dans quel fichier sont stockés les paramètres d'identité et d'authentification utilisés par le protocole EAP pour la méthode CHAP ?

Consulter les pages de manuels du démon pppd à la section *AUTHENTICATION*.

C'est le fichier `/etc/ppp/chap-secrets` qui contient les couples *login/password* utilisés lors de l'authentification.

Voici un exemple du contenu de ce fichier.

```
# Secrets for authentication using CHAP
# client     server secret   IP addresses
"spoke_site1" *  "0r4ng3_1"  *
"spoke_site2" *  "0r4ng3_2"  *
```

Q200. Dans quel fichier sont stockés les paramètres passés au démon pppd lors du lancement du serveur PPPoE ?

Consulter les pages de manuels de l'outil pppoe-server.

C'est le fichier `/etc/ppp/pppoe-server-options` qui contient la liste des paramètres utilisés lors du dialogue PPP.

Ce fichier contient tous les paramètres communs aux deux démons `pppd` qui sont lancés via pppoe-server. Voici comment créer le fichier.

```
cat << 'EOF' | sudo tee /etc/ppp/pppoe-server-options
# Gestion de session avec PAM
login
# Authentification EAP
require-eap
# Le Routeur Hub détient déjà une route par défaut
nodefaultroute
# Envoi de l'adresse de résolution DNS avec les adresses IPv4
ms-dns 172.16.0.2
# Ajout du protocole IPv6
+ipv6
# Informations détaillées dans la journalisation
debug
# Options préconisées par la documentation
noaccomp
default-asyncmap
nodeflate
nopcomp
novj
novjccomp
lcp-echo-interval 10
EOF
```

Q201. Comment créer les comptes utilisateurs locaux sur le routeur ***Hub*** sachant qu'ils ne sont autorisés ni à se connecter ni à avoir un répertoire personnel ?

Consulter les options de la commande adduser.

Voici un exemple dans le contexte de la maquette.

```
sudo adduser --gecos 'Spoke 1' --disabled-login --no-create-home spoke_site1
```

```
sudo adduser --gecos 'Spoke 2' --disabled-login --no-create-home spoke_site2
```

Q202. Quel paramètre supplémentaire doit être ajouté à la configuration de la commande pppoe-server pour distinguer les échanges entre les deux routeurs ***Spoke*** ?

Relativement au support *Routage inter-VLAN et protocole PPPoE*, il est essentiel de définir correctement les routes statiques vers les réseaux d'extrémité de chaque routeur ***Spoke***.

Consulter les options de la commande pppoe-server.

L'option `-u` permet de désigner une “unité“ qui sert à nommer l'interface. Par exemple, `-u 0` correspond à l'interface `ppp0`.

Q203. Comment créer deux nouvelles unités systemd responsables du lancement des processus pppoe-server ?

Consulter la page *systemd Services* et rechercher la procédure à suivre pour ajouter un service au lancement du système.

On commence par la création du fichier de service appelé : `/etc/systemd/system/pppoe-server.service` qui contient toutes les directives de lancement du processus pppoe-server avec les paramètres d'adressage du lien point à point.

Voici un exemple de création du fichier d'unité systemd pour le premier service.

```
cat << 'EOF' | sudo tee /etc/systemd/system/pppoe-server1.service
[Unit]
Description=PPPoE Server
After=systemd-networkd.service
Wants=systemd-networkd.service
BindsTo=sys-subsystem-net-devices-enp0s1.441.device
After=sys-subsystem-net-devices-enp0s1.441.device

[Service]
Type=forking
ExecCondition=/bin/sh -c '[ "$(systemctl show --property MainPID --value pppoe-server1.service)" = "0" ]'
ExecStart=/usr/sbin/pppoe-server -I enp0s1.441 -C BRAS -L 10.44.1.1 -R 10.44.1.2 -N 1 -u 0
Restart=on-failure
RestartSec=5

[Install]
WantedBy=multi-user.target
EOF
```

Voici un exemple de création du fichier d'unité systemd pour le second service.

```
cat << 'EOF' | sudo tee /etc/systemd/system/pppoe-server2.service
[Unit]
Description=PPPoE Server
After=systemd-networkd.service
Wants=systemd-networkd.service
BindsTo=sys-subsystem-net-devices-enp0s1.443.device
After=sys-subsystem-net-devices-enp0s1.443.device

[Service]
Type=forking
ExecCondition=/bin/sh -c '[ "$(systemctl show --property MainPID --value pppoe-server2.service)" = "0" ]'
ExecStart=/usr/sbin/pppoe-server -I enp0s1.443 -C BRAS -L 10.44.3.1 -R 10.44.3.2 -N 1 -u 1
Restart=on-failure
RestartSec=5

[Install]
WantedBy=multi-user.target
EOF
```

Q204. Comment activer les deux nouveaux services et contrôler leur état après lancement ?

Consulter la page *systemd Services* et rechercher la procédure à suivre pour activer et lancer un service.

On commence par la relecture de la liste des services disponibles par le gestionnaire systemd.

```
sudo systemctl daemon-reload
```

On active les nouveaux services.

```
for i in {1..2}; do sudo systemctl enable pppoe-server$i.service; done
```

On lance ce nouveau service.

```
for i in {1..2}; do sudo systemctl start pppoe-server$i.service; done
```

On vérifie que l'opération s'est déroulée correctement.

```
for i in {1..2}; do systemctl status pppoe-server$i.service; done
```

En l'état actuel de la configuration, aucune session PPP n'a encore été établie. Il faut maintenant passer à la configuration réseau du routeur ***Spoke*** pour avancer dans l'utilisation du protocole PPP.

3.2. Installer et configurer FRR

Les tâches à réaliser sont :

- Installer et préparer les services FRRouting
- Activer les démons des protocoles OSPFv2 pour IPv4 et OSPFv3 pour IPv6

Voici un extrait des questions du support de travaux pratiques *Introduction au routage dynamique OSPF avec FRRouting*.

Q205. Comment installer le paquet `frr` à partir du dépôt du site *FRRouting Project* ?

Pour installer le paquet FRR, on doit ajouter un nouveau dépôt au système.

On commence par ajouter la clé de signature des paquets à la configuration du gestionnaire.

```
sudo apt -y install curl gpg
```

```
curl -s https://deb.frrouting.org/frr/keys.asc | \
sudo gpg -o /usr/share/keyrings/frr-keyring.gpg --dearmor
```

On créé ensuite un nouveau fichier de liste de sources de paquets qui fait référence à cette clé de signature.

```
echo "deb [signed-by=/usr/share/keyrings/frr-keyring.gpg] \
https://deb.frrouting.org/frr bookworm frr-stable" | \
sudo tee /etc/apt/sources.list.d/frr.list
```

Avant de lancer l'installation des paquets de la suite FRRouting, on doit mettre à jour le catalogue local.

```
sudo apt update
sudo apt -y install frr frr-pythontools
```

On peut afficher les informations sur le paquet `frr`.

```
apt show ^frr$
```

Sans configuration particulière, les services zebra et staticd sont lancés. Aucun protocole de routage dynamique n'est activé.

```
systemctl status frr
```

```
# frr.service - FRRouting
     Loaded: loaded (/usr/lib/systemd/system/frr.service; enabled; preset: enabled)
     Active: active (running) since Sat 2024-11-02 16:17:42 CET; 5min ago
 Invocation: a1d3f0e79647471bb1a17069f3f4c69a
       Docs: https://frrouting.readthedocs.io/en/latest/setup.html
    Process: 2098 ExecStart=/usr/lib/frr/frrinit.sh start (code=exited, status=0/SUCCESS)
   Main PID: 2107 (watchfrr)
     Status: "FRR Operational"
      Tasks: 8 (limit: 1032)
     Memory: 14.9M (peak: 28.2M)
        CPU: 416ms
     CGroup: /system.slice/frr.service
             ├─2107 /usr/lib/frr/watchfrr -d -F traditional zebra mgmtd staticd
             ├─2117 /usr/lib/frr/zebra -d -F traditional -A 127.0.0.1 -s 90000000
             ├─2122 /usr/lib/frr/mgmtd -d -F traditional -A 127.0.0.1
             └─2124 /usr/lib/frr/staticd -d -F traditional -A 127.0.0.1

nov. 02 16:17:42 R2 staticd[2124]: [VTVCM-Y2NW3] Configuration Read in Took: 00:00:00
nov. 02 16:17:42 R2 frrinit.sh[2144]: [2144|staticd] done
nov. 02 16:17:42 R2 zebra[2117]: [VTVCM-Y2NW3] Configuration Read in Took: 00:00:00
nov. 02 16:17:42 R2 frrinit.sh[2128]: [2128|zebra] done
nov. 02 16:17:42 R2 watchfrr[2107]: [QDG3Y-BY5TN] zebra state -> up : connect succeeded
nov. 02 16:17:42 R2 watchfrr[2107]: [QDG3Y-BY5TN] mgmtd state -> up : connect succeeded
nov. 02 16:17:42 R2 watchfrr[2107]: [QDG3Y-BY5TN] staticd state -> up : connect succeeded
nov. 02 16:17:42 R2 watchfrr[2107]: [KWE5Q-QNGFC] all daemons up, doing startup-complete notify
nov. 02 16:17:42 R2 frrinit.sh[2098]: Started watchfrr.
nov. 02 16:17:42 R2 systemd[1]: Started frr.service - FRRouting.
```

Q206. Comment vérifier que la console unifiée du service `frr` est active ?

Afficher le contenu du fichier `/etc/frr/vtysh.conf` et vérifier qu'il contient l'entrée `service integrated-vtysh-config`.

L'accès à cette console unifiée est important puisqu'il permet d'utiliser une console unique pour les trois démons qui sont utilisés dans la suite des manipulations : zebra, ospfd et ospf6d.

Voici un exemple pour un routeur de la maquette.

```
sudo grep "service integrated-vtysh-config" /etc/frr/vtysh.conf
```

```
service integrated-vtysh-config
```

Q207. Comment ajouter l'utilisateur `etu` aux groupes `frr` et `frrvty` ?

Utiliser la commande adduser.

Une fois que l'utilisateur appartient à ces groupes, il a un accès direct à la console de configuration des protocoles actifs.

Comme dans les autres travaux pratiques de la série, on utilise une boucle.

```
for grp in frr frrvty
do
    sudo adduser etu $grp
done
```

Il ne faut pas oublier de déconnecter/reconnecter l'utilisateur pour bénéficier de la nouvelle attribution de groupe.

On vérifie l'appartenance aux groupes avec la commande groups.

```
groups
```

```
etu adm sudo users frrvty frr
```

Q208. Comment activer les deux démons des protocoles OSPFv2 et OSPFv3 ?

Consulter le fichier de configuration : `/etc/frr/daemons`.

Il faut remplacer les clés `no` en `yes` pour les démons des deux versions du protocole OSPF.

```
sudo sed -i 's/ospfd=no/ospfd=yes/' /etc/frr/daemons
sudo sed -i 's/ospf6d=no/ospf6d=yes/' /etc/frr/daemons
sudo systemctl restart frr
```

On peut alors afficher l'état du service `frr` et vérifier que les nouveaux démons de routage dynamique OSPF sont bien activés.

```
systemctl status frr
```

```
# frr.service - FRRouting
     Loaded: loaded (/usr/lib/systemd/system/frr.service; enabled; preset: enabled)
     Active: active (running) since Sat 2024-11-02 17:40:32 CET; 6s ago
 Invocation: 84e33888211f45f297c9135cace76751
       Docs: https://frrouting.readthedocs.io/en/latest/setup.html
    Process: 2467 ExecStart=/usr/lib/frr/frrinit.sh start (code=exited, status=0/SUCCESS)
   Main PID: 2476 (watchfrr)
     Status: "FRR Operational"
      Tasks: 12 (limit: 1032)
     Memory: 23M (peak: 34.8M)
        CPU: 380ms
     CGroup: /system.slice/frr.service
             ├─2476 /usr/lib/frr/watchfrr -d -F traditional zebra mgmtd ospfd ospf6d staticd
             ├─2488 /usr/lib/frr/zebra -d -F traditional -A 127.0.0.1 -s 90000000
             ├─2493 /usr/lib/frr/mgmtd -d -F traditional -A 127.0.0.1
             ├─2495 /usr/lib/frr/ospfd -d -F traditional -A 127.0.0.1
             ├─2498 /usr/lib/frr/ospf6d -d -F traditional -A ::1
             └─2501 /usr/lib/frr/staticd -d -F traditional -A 127.0.0.1

nov. 02 17:40:32 R2 mgmtd[2493]: [VTVCM-Y2NW3] Configuration Read in Took: 00:00:00
nov. 02 17:40:32 R2 frrinit.sh[2504]: [2504|mgmtd] done
nov. 02 17:40:32 R2 watchfrr[2476]: [QDG3Y-BY5TN] zebra state -> up : connect succeeded
nov. 02 17:40:32 R2 watchfrr[2476]: [QDG3Y-BY5TN] mgmtd state -> up : connect succeeded
nov. 02 17:40:32 R2 watchfrr[2476]: [QDG3Y-BY5TN] ospfd state -> up : connect succeeded
nov. 02 17:40:32 R2 watchfrr[2476]: [QDG3Y-BY5TN] ospf6d state -> up : connect succeeded
nov. 02 17:40:32 R2 watchfrr[2476]: [QDG3Y-BY5TN] staticd state -> up : connect succeeded
nov. 02 17:40:32 R2 watchfrr[2476]: [KWE5Q-QNGFC] all daemons up, doing startup-complete notify
nov. 02 17:40:32 R2 frrinit.sh[2467]: Started watchfrr.
nov. 02 17:40:32 R2 systemd[1]: Started frr.service - FRRouting.
```

On peut aussi lister les démons actifs à partir de la console du service.

```
vtysh
```

```
Hello, this is FRRouting (version 10.1.1).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

R2# sh daemons
 mgmtd zebra ospfd ospf6d watchfrr staticd
```

Une fois la partie configuration initiale traitée, passons à la maquette de la synthèse.

Q209. Quel est l'état courant du routage à ce stade de la configuration ?

Afficher les tables de routage des routeurs *Hub* avec les liens PPP actifs. Les échanges via le protocole OSPF ne sont pas encore configurés.

Dans le contexte de la maquette, l'affichage des deux tables de routage du routeur *Hub numéro 1* montre des entrées codées avec les clés suivantes.

```
sh ip route
```

```
Codes: K - kernel route, C - connected, L - local, S - static,
       R - RIP, O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,
       f - OpenFabric, t - Table-Direct,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup
       t - trapped, o - offload failure

K>* 0.0.0.0/0 [0/0] via 172.20.120.1, enp0s1.120, 00:00:36❶
L>* 10.47.2.1/32 is directly connected, ppp0, 00:00:36❷
C>* 10.47.2.2/32 is directly connected, ppp0, 00:00:36
L>* 10.47.4.1/32 is directly connected, ppp1, 00:00:36❸
C>* 10.47.4.2/32 is directly connected, ppp1, 00:00:36
C>* 172.20.70.0/29 is directly connected, enp0s1.470, 00:00:36❹
L>* 172.20.70.1/32 is directly connected, enp0s1.470, 00:00:36
C>* 172.20.120.0/23 is directly connected, enp0s1.120, 00:00:36❺
L>* 172.20.120.2/32 is directly connected, enp0s1.120, 00:00:36
```

```
sh ipv6 route
```

```
Codes: K - kernel route, C - connected, L - local, S - static,
       R - RIPng, O - OSPFv3, I - IS-IS, B - BGP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,
       f - OpenFabric, t - Table-Direct,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup
       t - trapped, o - offload failure

K>* ::/0 [0/1024] via fe80:78::1, enp0s1.120 onlink, 00:31:49❶
K>* 2001:678:3fc:78::/64 [0/512] is directly connected, enp0s1.120, 00:31:49
L>* 2001:678:3fc:78:baad:caff:fefe:5/128 is directly connected, enp0s1.120, 00:31:49
C * fe80::/64 is directly connected, enp0s1.120, 00:31:49
C * fe80::/64 is directly connected, enp0s1.474, 00:31:49
C * fe80::/64 is directly connected, enp0s1.471, 00:31:49
C * fe80::/64 is directly connected, enp0s1.472, 00:31:49
C * fe80::/64 is directly connected, enp0s1.473, 00:31:49
C * fe80::/64 is directly connected, enp0s1.470, 00:31:49
C>* fe80::/64 is directly connected, enp0s1, 00:31:49
C>* fe80::7188:100d:7562:f6d/128 is directly connected, ppp0, 00:31:49❷
C>* fe80::c00f:a3d0:f2c5:b154/128 is directly connected, ppp1, 00:31:49❸
C>* fe80:1d6::/64 is directly connected, enp0s1.470, 00:31:49❹
C>* fe80:1d7::/64 is directly connected, enp0s1.471, 00:31:49
C>* fe80:1d9::/64 is directly connected, enp0s1.473, 00:31:49
```

❶❶ La route par défaut est marquée `K>*` (***kernel***) parce qu'elle est importée de la configuration système. Elle n'est pas gérée par un service FRR.

❷❸❺ Avec IPv4, toutes les entrées des interfaces actives sont marquées `C>*` (***connected***). Le service FRR a importé les entrées du système et peut alimenter les démons de protocole de routage dynamique avec ces informations.

❹❹ Le nouveau réseau de collecte qui va servir aux échanges OSPF doit être présent dès maintenant avant d'activer le protocole de routage dynamique sur les interfaces.

❷❸❹ Avec IPv6, les réseaux d'interconnexion ou de transit n'utilisent que des adresses de lien local.

3.3. Configurer les démons OSPFv2 et OSPFv3

Les tâches à réaliser sont :

- Configurer les identifiants de routeur
- Activer OSPF sur les interfaces du réseau de collecte

Voici un jeu de questions extraites du support de travaux pratiques *Introduction au routage dynamique OSPF avec FRRouting*

Q210. Quelles sont les opérations à effectuer pour activer les protocoles de routage OSPFv2 et OSPFv3 ? Comment affecter manuellement l'identifiant du routeur ?

Avertissement

Les identifiants à utiliser lors de la séance de travaux pratiques sont donnés dans les tableaux des plans d'adressage.

La liste des commandes utiles en mode configuration dans la console vtysh est la suivante.

router ospf
router ospf6
ospf router-id X.X.X.X
ospf6 router-id X.X.X.X
log detail

Toujours à partir de la console vtysh, on accède au mode configuration à l'aide de la commande conf t. Voici un exemple de séquence sur le troisième routeur.

```
R1# conf t
R1(config)# router ospf
R1(config-router)# ospf router-id 1.0.0.4
R1(config-router)# log detail
R1(config-router)# ^Z
```

```
R1# conf t
R1(config)# router ospf6
R1(config-ospf6)# ospf6 router-id 1.0.0.6
R1(config-ospf6)# log detail
R1(config-ospf6)# ^Z
```

Une fois que chaque démon de routage OSPF possède un identifiant unique, on peut afficher les propriétés du protocole de routage dynamique même si aucun échange de route n'a encore eu lieu.

```
sh run ospfd
```

```
Building configuration...

Current configuration:
!
frr version 10.1.1
frr defaults traditional
hostname R1
log syslog informational
service integrated-vtysh-config
!
router ospf
 ospf router-id 1.0.0.4
 log-adjacency-changes detail
exit
!
end
```

```
sh run ospf6d
```

```
Building configuration...

Current configuration:
!
frr version 10.1.1
frr defaults traditional
hostname R1
log syslog informational
service integrated-vtysh-config
!
router ospf6
 ospf6 router-id 1.0.0.6
 log-adjacency-changes detail
exit
!
end
```

Le choix de codage des identifiants OSPF a pour but d'éviter une confusion avec les adresses des réseaux actifs sur chaque routeur.

Si on reprend les instructions de la question précédente, on obtient l'état de chacun des démons de protocole de routage dynamique.

```
sh ip ospf
```

```
 OSPF Routing Process, Router ID: 1.0.0.4
 Supports only single TOS (TOS0) routes
 This implementation conforms to RFC2328
 RFC1583Compatibility flag is disabled
 OpaqueCapability flag is disabled
 Initial SPF scheduling delay 0 millisec(s)
 Minimum hold time between consecutive SPFs 50 millisec(s)
 Maximum hold time between consecutive SPFs 5000 millisec(s)
 Hold time multiplier is currently 1
 SPF algorithm has not been run
 SPF timer is inactive
 LSA minimum interval 5000 msecs
 LSA minimum arrival 1000 msecs
 Write Multiplier set to 20
 Refresh timer 10 secs
 Maximum multiple paths(ECMP) supported 256
 Administrative distance 110
 Number of external LSA 0. Checksum Sum 0x00000000
 Number of opaque AS LSA 0. Checksum Sum 0x00000000
 Number of areas attached to this router: 0
 All adjacency changes are logged
```

```
sh ipv6 ospf
```

```
 OSPFv3 Routing Process (0) with Router-ID 1.0.0.6
 Running 00:09:20
 LSA minimum arrival 1000 msecs
 Maximum-paths 256
 Administrative distance 110
 Initial SPF scheduling delay 0 millisec(s)
 Minimum hold time between consecutive SPFs 50 millsecond(s)
 Maximum hold time between consecutive SPFs 5000 millsecond(s)
 Hold time multiplier is currently 1
 SPF algorithm has not been run
 SPF timer is inactive
 Number of AS scoped LSAs is 0
 Number of areas in this router is 0
 Authentication Sequence number info
  Higher sequence no 0, Lower sequence no 0
 All adjacency changes are logged
```

L'affectation des identifiants des démons de routage OSPF doit être réalisée sur les trois routeurs de la topologie. Ces identifiants seront très utiles et importants dès qu'il faudra afficher les listes de routeurs voisins au sens du protocole OSPF.

Q211. Comment activer les protocoles de routage OSPFv2 et OSPFv3 pour les réseaux d'interconnexion de chaque routeur ?

Il faut activer le protocole de routage dynamique sur chaque interface de la topologie qui participe à la construction du triangle.

La liste des commandes utiles en mode console et en mode configuration dans vtysh est la suivante.

show ip route connected
show ip route ospf
show ipv6 route connected
show ipv6 route ospf
ip ospf area 0
ipv6 ospf6 area 0

Voici un exemple de séquence d'instructions pour le routeur R1.

On commence par lister les entrées marquées ***C*** ou ***connected*** des tables de routage IPv4 et IPv6 de façon à reconnaître les deux côtés de la topologie triangle connus du “sommet” R1.

```
sh ip route connected
```

```
Codes: K - kernel route, C - connected, L - local, S - static,
       R - RIP, O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,
       f - OpenFabric, t - Table-Direct,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup
       t - trapped, o - offload failure

C>* 10.44.0.0/29 is directly connected, enp0s1.440, 01:26:12
C>* 10.44.1.0/29 is directly connected, enp0s1.441, 01:26:12
C>* 192.168.104.128/29 is directly connected, enp0s1.360, 01:26:12
```

```
sh ipv6 route connected
```

```
Codes: K - kernel route, C - connected, L - local, S - static,
       R - RIPng, O - OSPFv3, I - IS-IS, B - BGP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,
       f - OpenFabric, t - Table-Direct,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup
       t - trapped, o - offload failure

C>* 2001:678:3fc:168::/64 is directly connected, enp0s1.360, 01:30:26
C * fe80::/64 is directly connected, enp0s1.360, 01:30:25
C * fe80::/64 is directly connected, enp0s1.440, 01:30:26
C * fe80::/64 is directly connected, enp0s1, 01:30:26
C>* fe80::/64 is directly connected, enp0s1.441, 01:30:26
```

À partir de ces affichages, on sait que l'on doit activer les protocoles OSPF pour les deux sous-interfaces : enp0s1.440 et enp0s1.441.

```
R1# conf t
R1(config)# int enp0s1.440
R1(config-if)# ip ospf area 0
R1(config-if)# ipv6 ospf6 area 0
R1(config-if)# int enp0s1.441
R1(config-if)# ip ospf area 0
R1(config-if)# ipv6 ospf6 area 0
R1(config-if)# ^Z
```

Ces opérations doivent être répétées sur les trois routeurs de la topologie.

Q212. Comment visualiser l'état des interfaces actives pour chaque processus de protocole de routage dynamique OSPFv2 ou OSPFv3 ?

Les interfaces sont dites actives pour les protocoles OSPFv2 ou OSPFv3 dès qu'elles ont été ajoutées aux processus de routage dynamique en précisant l'aire à laquelle elles appartiennent.

La liste des commandes utiles dans la console vtysh est la suivante.

show ip ospf interface
show ipv6 ospf6 interface

En prenant l'exemple du routeur R2, on obtient les résultats suivants.

```
R2# sh ip ospf interface enp0s1.440
```

```
enp0s1.440 is up❶
  ifindex 4, MTU 1500 bytes, BW 0 Mbit❷ <UP,LOWER_UP,BROADCAST,RUNNING,MULTICAST>
  Internet Address 10.44.0.2/29, Broadcast 10.44.0.7, Area 0.0.0.0
  MTU mismatch detection: enabled
  Router ID 2.0.0.4❸, Network Type BROADCAST, Cost: 10
  Transmit Delay is 1 sec, State Backup, Priority 1
  Designated Router (ID) 1.0.0.4 Interface Address 10.44.0.1/29❹
  Backup Designated Router (ID) 2.0.0.4, Interface Address 10.44.0.2❺
  Multicast group memberships: OSPFAllRouters OSPFDesignatedRouters
  Timer intervals configured, Hello 10s, Dead 40s, Wait 40s, Retransmit 5
    Hello due in 5.086s
  Neighbor Count is 1, Adjacent neighbor count is 1❻
  Graceful Restart hello delay: 10s
```

La copie d'écran ci-dessus permet d'identifier les éléments suivants :

❶ L'indicateur ***is up*** confirme que l'interface est bien active pour le protocole de routage. Cela signifie que les messages du protocole OSPF sont transmis sur cette interface et que des échanges avec des routeurs OSPF voisins sur ce réseau peuvent avoir lieu.

❷ Le fait que la bande passante soit “vue” comme étant nulle montre que le calcul de coût de lien ne prend pas en compte ce facteur. Ce point sera repris dans le calcul des coûts à partir d'une référence définie dans la configuration des démons OSPF.

❸ On retrouve ici l'identifiant du routeur transmis vers les autres routeurs voisins.

❹ Cette ligne identifie le routeur OSPF R1 comme étant le routeur de référence sur ce réseau IPv4. Dans notre cas, la topologie triangle ne comprend qu'un seul routeur voisin OSPF par réseau, ce qui limite l'utilité d'un routeur de référence pour les calculs des meilleurs routes. Si on avait plusieurs routeurs présents sur un même réseau, le rôle de référent serait essentiel pour limiter les échanges de bases de données de routage lors des calculs de tables de topologie.

❺ Cette ligen identifie le routeur OSPF R2 comme étant le routeur de référence de secours sur ce réseau. Comme dans le cas précédent, l'utilisation d'une topologie triangle limite l'importance de ce rôle comme il n'y a que deux routeurs pour chaque côté de cette topologie.

❻ Le routeur OSPF a un routeur voisin sur ce réseau. Il s'agit du routeur R1.

On reprend la même démarche pour le protocole OSPFv3.

```
R2# sh ipv6 ospf6 interface enp0s1.440
```

```
enp0s1.440 is up❶, type BROADCAST
  Interface ID: 4
  Internet Address:
    inet : 10.44.0.2/29
    inet6: fe80::baad:caff:fefe:6/64
    inet6: fe80::1b8:2/64
  Instance ID 0, Interface MTU 1500 (autodetect: 1500)
  MTU mismatch detection: enabled
  Area ID 0.0.0.0, Cost 10
  State BDR, Transmit Delay 1 sec, Priority 1
  Timer intervals configured:
   Hello 10(8.803), Dead 40, Retransmit 5
  DR: 1.0.0.6 BDR: 2.0.0.4❷
  Number of I/F scoped LSAs is 2
    0 Pending LSAs for LSUpdate in Time 00:00:00 [thread off]
    0 Pending LSAs for LSAck in Time 00:00:00 [thread off]
  Graceful Restart hello delay: 10s
  Authentication Trailer is disabled
```

Relativement, à l'affichage des informations sur l'association entre une interface réseau et le démon de protocole OSPFv2 pour IPv4, l'affichage pour OSPFv3 et IPv6 est plus succinct.

❶ Cette information est identique à celle du démon OSPFv2. L'interface est associée et active. Les messages du protocole OSPFv3 peuvent être échangés sur le réseau auquel cette interface appartient.

❷ Les identifiants des routeurs référence et de secours sont affichés sur une seule ligne.

Q213. Comment vérifier que l'identifiant de routeur a correctement été attribué ?

À partir des commandes proposées et de résultats des questions précédentes, rechercher l'information demandée.

Quelque soit la version du protocole OSPF, l'identifiant de routeur est toujours codé sous la forme d'une adresse IPv4.

Pour valider la conformité entre les identifiants définis dans le plan d'adressage des travaux pratiques et les valeurs effectivement utilisées par les deux démons de routage dynamique, on affiche le contenu des base de topologie du protocole.

Avec le démon OSPFv2, l'identification du routeur est immédiate.

```
R1# sh ip ospf database
```

```
       OSPF Router with ID (1.0.0.4)

                Router Link States (Area 0.0.0.0)

Link ID         ADV Router      Age  Seq#       CkSum  Link count
1.0.0.4        1.0.0.4          884 0x8000001f 0x3bee 2
2.0.0.4        2.0.0.4         1016 0x8000001d 0xab78 2
3.0.0.4        3.0.0.4          843 0x8000001d 0xd34a 2

                Net Link States (Area 0.0.0.0)

Link ID         ADV Router      Age  Seq#       CkSum
10.44.0.1      1.0.0.4          994 0x80000018 0xe61c
10.44.1.1      1.0.0.4          984 0x80000018 0xe61a
10.44.2.3      3.0.0.4          843 0x80000018 0xbc3e
```

En revanche, le démon OSPFv3 ne donne pas d'information sur l'identifiant du processus en cours. L'affichage est cependant beaucoup plus exhaustif avec les informations par interface.

```
R1# sh ipv6 ospf6 database
```

```
        Area Scoped Link State Database (Area 0)

Type LSId           AdvRouter       Age   SeqNum                     Payload
Rtr  0.0.0.0        1.0.0.6        1012 8000001b               1.0.0.6/0.0.0.5
Rtr  0.0.0.0        1.0.0.6        1012 8000001b               1.0.0.6/0.0.0.3
Rtr  0.0.0.0        2.0.0.4        1010 80000019               1.0.0.6/0.0.0.5
Rtr  0.0.0.0        2.0.0.4        1010 80000019               3.0.0.6/0.0.0.4
Rtr  0.0.0.0        3.0.0.6        1010 80000019               1.0.0.6/0.0.0.3
Rtr  0.0.0.0        3.0.0.6        1010 80000019               3.0.0.6/0.0.0.4
Net  0.0.0.3        1.0.0.6        1088 80000017                      1.0.0.6
Net  0.0.0.3        1.0.0.6        1088 80000017                      3.0.0.6
Net  0.0.0.5        1.0.0.6        1012 80000017                      1.0.0.6
Net  0.0.0.5        1.0.0.6        1012 80000017                      2.0.0.4
Net  0.0.0.4        3.0.0.6        1010 80000017                      3.0.0.6
Net  0.0.0.4        3.0.0.6        1010 80000017                      2.0.0.4

        I/F Scoped Link State Database (I/F enp0s1.440 in Area 0)❶

Type LSId           AdvRouter       Age   SeqNum                     Payload
Lnk  0.0.0.5        1.0.0.6         303 8000001a                 fe80::1b8:1
Lnk  0.0.0.4        2.0.0.4        1019 80000017                 fe80::1b8:2

        I/F Scoped Link State Database (I/F enp0s1.441 in Area 0)❷

Type LSId           AdvRouter       Age   SeqNum                     Payload
Lnk  0.0.0.3        1.0.0.6         288 8000001a                 fe80::1b9:1
Lnk  0.0.0.3        3.0.0.6        1090 80000017                 fe80::1b9:3

        AS Scoped Link State Database

Type LSId           AdvRouter       Age   SeqNum                     Payload
```

❶ Sur le côté R1-R2 du triangle, on utilise le VLAN 440 et les identifiants de routeurs correspondent.
❷ Sur le côté R1-R3 du triangle, on utilise le VLAN 441 et les identifiants de routeurs correspondent.

Q214. Comment identifier le type de réseau d'une interface ?

À partir des résultats des questions précédentes, rechercher l'information demandée.

Comme on utilise des liens Ethernet dans ce contexte de travaux pratiques, le type le plus important est le réseau de diffusion ou ***BROADCAST***.

```
R3# sh ip ospf interface enp0s1.442
```

```
enp0s1.442 is up
  ifindex 4, MTU 1500 bytes, BW 0 Mbit <UP,LOWER_UP,BROADCAST,RUNNING,MULTICAST>
  Internet Address 10.44.2.3/29, Broadcast 10.44.2.7, Area 0.0.0.0
  MTU mismatch detection: enabled
  Router ID 3.0.0.4, Network Type BROADCAST, Cost: 10
  Transmit Delay is 1 sec, State DR, Priority 1
  Designated Router (ID) 3.0.0.4 Interface Address 10.44.2.3/29
  Backup Designated Router (ID) 2.0.0.4, Interface Address 10.44.2.2
  Saved Network-LSA sequence number 0x80000019
  Multicast group memberships: OSPFAllRouters OSPFDesignatedRouters
  Timer intervals configured, Hello 10s, Dead 40s, Wait 40s, Retransmit 5
    Hello due in 1.361s
  Neighbor Count is 1, Adjacent neighbor count is 1
  Graceful Restart hello delay: 10s
```

```
R3# sh ipv6 ospf6 interface enp0s1.442
```

```
enp0s1.442 is up, type BROADCAST
  Interface ID: 4
  Internet Address:
    inet : 10.44.2.3/29
    inet6: fe80::baad:caff:fefe:7/64
    inet6: fe80::1ba:3/64
  Instance ID 0, Interface MTU 1500 (autodetect: 1500)
  MTU mismatch detection: enabled
  Area ID 0.0.0.0, Cost 10
  State DR, Transmit Delay 1 sec, Priority 1
  Timer intervals configured:
   Hello 10(9.055), Dead 40, Retransmit 5
  DR: 3.0.0.6 BDR: 2.0.0.4
  Number of I/F scoped LSAs is 2
    0 Pending LSAs for LSUpdate in Time 00:00:00 [thread off]
    0 Pending LSAs for LSAck in Time 00:00:00 [thread off]
  Graceful Restart hello delay: 10s
  Authentication Trailer is disabled
```

Q215. Comment obtenir la liste du ou des routeurs voisins pour chaque processus de protocole de routage dynamique OSPFv2 ou OSPFv3 ?

Dès qu'une interface est active, il y a émission de paquets HELLO et si un autre routeur avec une interface active envoie aussi des paquets HELLO dans le même VLAN, les deux routeurs cherchent à former une adjacence.

La liste des commandes utiles dans la console vtysh est la suivante.

show ip ospf neighbor
show ipv6 ospf6 neighbor

À partir du routeur R1, voici un exemple de liste de routeurs OSPF voisins dans laquelle on reconnaît les identifiants des routeurs R2 et R3.

```
R1# sh ip ospf neighbor
```

```
Neighbor ID     Pri State           Up Time         Dead Time Address         Interface                RXmtL RqstL DBsmL
2.0.0.4           1 Full/Backup     11h34m21s         37.712s 10.44.0.2       enp0s1.440:10.44.0.1     0     0     0
3.0.0.4           1 Full/Backup     11h35m30s         33.792s 10.44.1.3       enp0s1.441:10.44.1.1     0     0     0
```

```
R1# sh ipv6 ospf6 neighbor
```

```
Neighbor ID     Pri    DeadTime    State/IfState         Duration I/F[State]
2.0.0.4           1    00:00:37     Full/BDR             11:36:00 enp0s1.440[DR]
3.0.0.6           1    00:00:37     Full/BDR             11:37:15 enp0s1.441[DR]
```

Les deux listes de voisins donnent une information essentielle sur l'état de protocole de routage avec le mot clé Full. Cet état indique que deux routeurs adjacents ont entièrement synchronisé leurs bases de données d'état de liens, permettant ainsi un échange complet des informations de routage.

Pour achever cette partie de la synthèse, il reste à étudier la redistribution des routes connectées dans les démons de routage OSPFv2 et OSPFv3.

Q216. Quel est l'état des voisins OSPF à ce stade de la configuration ?

Afficher la liste des voisins OSPFv2 OSPFv3 sur l'un des deux routeurs ***Hub***.

En se plaçant sur le second routeur ***Hub*** de la maquette, on obtient les informations suivantes.

```
sh ip ospf neighbor
```

```
Neighbor ID     Pri State     Up Time    Dead Time  Address      Interface              RXmtL RqstL DBsmL
1.0.0.4           1 Full/DR   23.841s      36.142s  172.20.70.1  enp0s1.470:172.20.70.2     0     0     0
```

```
sh ipv6 ospf6 neighbor
```

```
Neighbor ID     Pri    DeadTime    State/IfState         Duration I/F[State]
1.0.0.6           1    00:00:30     Full/DR              00:00:29 enp0s1.470[BDR]
```


Note

En l'état actuel des échanges OSPF, l'affichage des tables de routage ne montre aucune route apprise via routage dynamique. Il est donc nécessaire d'importer les routes des liaisons PPPoE dans la configuration des démons OSPF.

3.4. Redistribuer les routes connectées dans OSPF

Les tâches à réaliser sont :

- Configurer la redistribution des routes connectées dans OSPF
- Valider la présence des routes IPv4 de transit vers les routeurs ***Spoke***

Q217. Quelle est l'instruction qui permet de redistribuer les routes connectées dans les démons OSPF ?

Rechercher les options de l'instruction redistribute dans la documentation FRR à l'adresse *FRRouting User Guide*.

L'instruction redistribute est générique et s'emploie dans la configuration de tous les protocoles de routage. Voici un extrait de configuration pour chacun des deux démons OSPF.

```
sh run ospfd
```

```
Building configuration...

Current configuration:
!
frr version 10.1.1
frr defaults traditional
hostname hub1
log syslog informational
service integrated-vtysh-config
!
interface enp0s1.470
 ip ospf area 0
exit
!
router ospf
 ospf router-id 1.0.0.4
 log-adjacency-changes detail
 redistribute connected
exit
!
end
```

```
sh run ospf6d
```

```
Building configuration...

Current configuration:
!
frr version 10.1.1
frr defaults traditional
hostname hub1
log syslog informational
service integrated-vtysh-config
!
interface enp0s1.470
 ipv6 ospf6 area 0
exit
!
router ospf6
 ospf6 router-id 1.0.0.6
 log-adjacency-changes detail
 redistribute connected
exit
!
end
```

Q218. Pourquoi choisir la redistribution des routes connectées ?

Il existe de nombreuses solutions pour intégrer des réseaux dans OSPF. Voici deux exemples :

- Activer OSPF sur les interfaces PPP
- Ajouter les adresses des réseaux de transit vers les routeurs ***Spoke***.

Cela s'explique par les principes de la topologie ***Hub & Spoke***. Un lien WAN sur lequel on établit une session PPPoE n'a pas de caractère permanent. Il faut donc trouver une solution qui permette au protocole de routage dynamique OSPF de fonctionner de façon optimale avec un nombre de routeurs ***Spoke*** variable.

Avec la maquette des travaux pratiques, chaque routeur ***Hub*** est relié à deux routeurs ***Spoke***. Dans un contexte plus réaliste, le nombre de routeurs ***Spoke*** peut varier en fonction de l'ouverture ou de la fermeture de sites distants, et le fonctionnement du réseau de collecte ne doit pas être impacté par ces variations.

C'est la raison pour laquelle les variations du nombre de routes connectées sont gérées au niveau système avec les services PPP. Les changements sont automatiquement répercutés dans la base d'information FRR puis publiés par le protocole de routage dynamique OSPF. La redistribution des routes connectées répond parfaitement à ce besoin.

Q219. Comment vérifier que la redistribution de route est effective ?

Rechercher les entrées OSPF dans les tables de routage des deux routeurs ***Hub***.

Voici ce que l'on observe sur le premier routeur de la maquette.

```
sh ip route ospf
```

```
Codes: K - kernel route, C - connected, L - local, S - static,
       R - RIP, O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,
       f - OpenFabric, t - Table-Direct,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup
       t - trapped, o - offload failure

O>* 10.47.6.2/32 [110/20] via 172.20.70.2, enp0s1.470, weight 1, 00:46:51
O>* 10.47.8.2/32 [110/20] via 172.20.70.2, enp0s1.470, weight 1, 00:46:51
O   172.20.70.0/29 [110/10] is directly connected, enp0s1.470, weight 1, 03:43:32
O   172.20.120.0/23 [110/20] via 172.20.70.2, enp0s1.470, weight 1, 00:46:51
```

Dans le cas du protocole IPv6, la redistribution n'est pas visible puisque les réseaux de transit utilisent uniquement des adresses de lien local pour former les adjacences entre routeurs. La commande ci-dessous ne produit aucune sortie. Il faut attendre la mise en place des réseaux d'hébergement au delà des routeurs ***Spoke*** pour voir apparaître de nouvelles entrées dans la table de routage.

```
sh ipv6 route ospf
```

4. Configurer les routeurs Spoke

Les quatre routeurs ***Spoke*** de la topologie jouent toujours le rôle de routeur d'extrémité.

Pour les manipulations de configuration de ce rôle, on s'appuie sur le support de travaux pratiques : *Routage inter-VLAN et protocole PPPoE*.

Dans cette section, on s'intéresse aux rôles et fonctionnalités suivantes :

- Gérer les sessions PPP avec les routeurs ***Hub***
- Valider les communications entre routeurs ***Spoke***

4.1. Configurer les clients PPP

Les tâches à réaliser sont :

- Activer le routage
- Configurer le protocole PPP

Pour l'activation du routage, on retrouve les mêmes opérations que sur les routeurs ***Hub***.

Q220. Comment activer le routage dans le sous-système réseau du noyau Linux ?

Utiliser la commande sysctl pour effectuer des recherches et identifier les paramètres utiles. Par exemple :

```
sudo sysctl -a -r ".*forward.*"
```

Il est dorénavant recommandé de créer un fichier de configuration spécifique par fonction. C'est la raison pour laquelle on crée un nouveau fichier `/etc/sysctl.d/10-routing.conf`.

Attention ! Il ne faut pas oublier d'appliquer les nouvelles valeurs des paramètres de configuration.

```
cat << EOF | sudo tee /etc/sysctl.d/10-routing.conf
net.ipv4.ip_forward=1
net.ipv6.conf.all.forwarding=1
net.ipv4.conf.all.log_martians=1
EOF
```

Voici un exemple des résultats obtenus après application des nouveaux paramètres.

```
sudo sysctl --system
```

La configuration des “clients” PPP diffère de celle des routeurs ***Hub***.

Q221. Quel paquet fournit le démon de gestion des sessions du protocole PPP sur le routeur ***Spoke*** ?

Rechercher dans le catalogue des paquets, la référence ppp.

```
apt search ^ppp
```

```
ppp/testing 2.5.0-1+2 amd64
  protocole point à point (PPP) - démon

ppp-dev/testing 2.5.0-1+2 all
  protocole point à point (PPP) – fichiers de développement

ppp-gatekeeper/testing 0.1.0-201406111015-1.1 all
  PPP manager for handling balanced, redundant and failover links

pppoe/testing 4.0-1 amd64
  Pilote PPP sur Ethernet

pppoeconf/testing 1.21+nmu3 all
  configures PPPoE/ADSL connections

wmppp.app/testing 1.3.2-2 amd64
  contrôle de connexion et surveillance de la charge réseau avec aspect NeXTStep
```

Le résultat de la commande apt show ppp montre que c'est bien ce paquet qui répond au besoin.

```
sudo apt -y install ppp
```

Q222. Comment utiliser l'encapsulation des trames PPP dans Ethernet à partir du démon pppd fourni avec le paquet ppp ?

Rechercher dans le répertoire de documentation du paquet ppp.

Dans le répertoire `/usr/share/doc/ppp/`, on trouve le fichier `README.pppoe` qui indique que l'appel au module `rp-pppoe.so` permet d'encapsuler des trames PPP sur un réseau local Ethernet.

Toujours à partir du même répertoire, on trouve dans la liste des fichiers d'exemples de configuration un modèle adapté à notre contexte : `peers-pppoe`.

Q223. Dans quel fichier sont stockés les paramètres d'identité et d'authentification utilisés par le protocole CHAP ?

Consulter les pages de manuels du démon pppd à la section ***AUTHENTICATION***.

C'est le fichier `/etc/ppp/chap-secrets` qui contient les couples ***login/password*** utilisés lors de l'authentification.

Voici un exemple du contenu de ce fichier. Le nom du client ainsi que son mot de passe secret doivent être identiques à chaque extrémité de la session PPP.

```
# Secrets for authentication using CHAP
# client server secret   IP addresses
"spoke_site0"   *  "5p0k3"   *
```

Q224. Quelles sont les options de configuration du démon pppd à placer dans le fichier `/etc/ppp/peers/pppoe-provider` pour assurer l'établissement de la session PPP entre les routeurs ?

Utiliser le fichier exemple PPPoE fourni avec la documentation du paquet ppp.

Voici comment créer un fichier `/etc/ppp/peers/pppoe-provider` avec les options correspondant au contexte de la maquette du routeur vert.

```
cat << 'EOF' | sudo tee /etc/ppp/peers/pppoe-provider
# Le nom d'utilisateur désigne l'entrée du fichier /etc/ppp/chap-secrets
user spoke_site0

# Chargement du module PPPoE avec les détails dans la journalisation
plugin rp-pppoe.so rp_pppoe_ac BRAS rp_pppoe_verbose 1

# Interface (VLAN) utilisé pour l'établissement de la session PPP
enp0s1.441

# Les adresses sont attribuées par le "serveur" PPPoE
noipdefault
# L'adresse de résolution DNS est aussi fournie par le serveur PPPoE
usepeerdns
# La session PPP devient la route par défaut du routeur Spoke
defaultroute

# Demande de réouverture de session automatique en cas de rupture
persist

# Le routeur Spoke n'exige pas que le routeur Hub s'authentifie
noauth

# Messages d'informations détaillés dans la journalisation
debug

# Utilisation du protocole IPv6
+ipv6

# Options préconisées par la documentation
noaccomp
default-asyncmap
nodeflate
nopcomp
novj
novjccomp
lcp-echo-interval 10
EOF
```

Q225. Comment lancer le démon pppd pour qu'il prenne en compte les paramètres définis dans le fichier complété à la question précédente ?

Consulter les pages de manuels du démon pppd.

C'est l'outil pon qui permet de désigner le fichier de configuration à utiliser. Voici une copie d'écran du lancement du démon pppd.

```
sudo pon pppoe-provider
```

Cette commande individuelle est à utiliser pour faire un tout premier test. Pour rendre la configuration persistante au redémarrage nous avons besoin de créer un service systemd. Il ne faut donc pas oublier d'arrêter le processus avant de passer à la question suivante. Le paquet fournit un outil dédié : poff.

```
sudo poff -a pppoe-provider
```

Q226. Quels sont les noms des deux sous-couches du protocole PPP qui apparaissent dans les journaux systèmes ? Quels sont les rôles respectifs de ces deux sous-couches ?

Consulter la page *Point-to-Point Protocol*.

La consultation des journaux système lors du dialogue PPP fait apparaître tous les détails. Voir les exemples de traces à l'adresse : *Traces d'une ouverture de session PPPoE*.

Q227. Quels sont les en-têtes du dialogue qui identifient les requêtes (émises|reçues), les rejets et les acquittements ?

Consulter les journaux système contenant les traces d'une connexion PPP.

La copie d'écran donnée ci-dessus fait apparaître les directives `Conf*` pour chaque paramètre négocié.

- `ConfReq` indique une requête.
- `ConfAck` indique un acquittement.
- `ConfNak` indique un rejet.

Q228. Comment assurer une ouverture automatique de la session PPP à chaque réinitialisation système ?

Consulter la page *systemd Services* et rechercher la procédure à suivre pour ajouter un service au lancement du système.

On commence par la création du fichier de service appelé : `/etc/systemd/system/ppp.service` qui contient les appels aux outils pon et poff.

Voici l'instruction de création du fichier de service.

```
cat << EOF | sudo tee /etc/systemd/system/ppp.service
[Unit]
Description=PPPoE Client Connection
After=network.target
Wants=network.target
BindsTo=sys-subsystem-net-devices-enp0s1.441.device
After=sys-subsystem-net-devices-enp0s1.441.device

[Service]
Type=forking
ExecStart=/usr/bin/pon pppoe-provider
ExecStop=/usr/bin/poff pppoe-provider
Restart=on-failure
RestartSec=20

[Install]
WantedBy=multi-user.target
EOF
```

Q229. Comment activer le nouveau service et contrôler son état après lancement ?

Consulter la page *systemd Services* et rechercher la procédure à suivre pour activer et lancer un service.

On commence par la relecture de la liste des services disponibles par le gestionnaire systemd.

```
sudo systemctl daemon-reload
```

On active le nouveau service.

```
sudo systemctl enable ppp.service
```

On lance ce nouveau service.

```
sudo systemctl start ppp.service
```

On vérifie que l'opération s'est déroulée correctement.

```
systemctl status ppp.service
```

```
# ppp.service - PPPoE Client Connection
     Loaded: loaded (/etc/systemd/system/ppp.service; enabled; preset: enabled)
     Active: active (running) since Sun 2024-09-22 08:21:32 CEST; 13min ago
 Invocation: 69386a40f7574821b3986ed6c6c242f7
   Main PID: 496 (pppd)
      Tasks: 1 (limit: 1086)
     Memory: 2.8M (peak: 4.9M)
        CPU: 56ms
     CGroup: /system.slice/ppp.service
             └─496 /usr/sbin/pppd call pppoe-provider

sept. 22 08:21:37 spoke pppd[496]: rcvd [IPCP ConfAck id=0x2 <addr 10.4.41.2> <ms-dns1 172.16.0.2> <ms-dns2 172.16.0.2>]
sept. 22 08:21:37 spoke pppd[496]: Script /etc/ppp/ip-pre-up started (pid 510)
sept. 22 08:21:37 spoke pppd[496]: Script /etc/ppp/ip-pre-up finished (pid 510), status = 0x0
sept. 22 08:21:37 spoke pppd[496]: local  IP address 10.4.41.2
sept. 22 08:21:37 spoke pppd[496]: remote IP address 10.4.41.1
sept. 22 08:21:37 spoke pppd[496]: primary   DNS address 172.16.0.2
sept. 22 08:21:37 spoke pppd[496]: secondary DNS address 172.16.0.2
sept. 22 08:21:37 spoke pppd[496]: Script /etc/ppp/ip-up started (pid 514)
sept. 22 08:21:37 spoke pppd[496]: Script /etc/ppp/ipv6-up finished (pid 509), status = 0x0
sept. 22 08:21:37 spoke pppd[496]: Script /etc/ppp/ip-up finished (pid 514), status = 0x0
```

Q230. Comment utiliser la session PPP (le VLAN orange) comme lien unique de raccordement réseau du routeur *Spoke* ?

Maintenant que le fonctionnement de la session PPP est validé, nous n'avons plus besoin du raccordement temporaire sur le routeur ***Spoke***. Il faut donc commenter les entrées du fichier `/etc/netplan/enp0s1.yaml` qui ne sont plus utiles et attribuer l'adresse de résolution DNS de secours.

Une fois ces opérations effectuées, on peut redémarre le routeur ***Spoke*** pour se placer en situation de raccordement distant.

Pour commencer, on commente les entrées inutile du fichier `/etc/netplan/enp0s1.yaml`.

```
cat /etc/netplan/enp0s1.yaml
```

```
network:
  version: 2
  ethernets:
    enp0s1:
      dhcp4: false
      dhcp6: false
      accept-ra: false
#     nameservers:
#       addresses:
#          - 172.16.0.2
#          - 2001:678:3fc:3::2

  vlans:
    enp0s1.440: # VLAN violet
      id: 440
      link: enp0s1
      addresses:
        - fe80:1b8::2/64
    enp0s1.441: # VLAN orange
      id: 441
      link: enp0s1
      addresses: []
#   enp0s1.52: # VLAN accès temporaire
#     id: 52
#     link: enp0s1
#     dhcp4: true
#     dhcp6: false
#     accept-ra: true
```

On peut appliquer directement les modifications à l'aide de la commande netplan.

```
sudo netplan apply
```

```
sudo netplan status
```

Attention

L'affectation de l'adresse IPv4 ou IPv6 de résolution DNS pose problème. En effet, si le démon pppd propose bien deux adresses via l'option `usepeerdns`, ces propositions ne sont pas prises en charge par le service systemd-resolved.

On contourne cette difficulté en affectant une adresse IPv4 directement au service systemd-resolved.

On édite le fichier `/etc/systemd/resolved.conf` pour affecter directement l'adresse de résolution DNS. Voici une copie des lignes utiles du fichier modifié. Toutes les autres lignes sont commentées.

```
grep -Ev '(^#|^$)'  /etc/systemd/resolved.conf
[Resolve]
DNS=172.16.0.2
```

Il ne faut pas oublier de relancer le service pour prendre en compte les modifications du fichier.

```
sudo systemctl restart systemd-resolved
```

Le routeur ***Spoke*** est maintenant prêt à être redémarré pour utiliser le lien de raccordement distant comme seul canal d'accès aux autres réseaux.

```
sudo reboot
```

Pour visualiser les détails du fonctionnement du protocole point à point, voir la section « Traces d'une ouverture de session PPPoE » du document « *Routage inter-VLAN et protocole PPPoE* ».

4.2. Valider les communications

Après avoir vérifié que les quatre sessions PPP de la maquette sont actives, on peut lancer une série de tests ICMP suivant deux axes.

1. Accéder à l'Internet depuis chaque routeur ***Spoke*** avec IPv4.

   L'accès via IPv6 n'est pas possible sachant qu'aucune adresse de type ULA ou GUA n'est présente sur les routeurs ***Spoke***.

```
etu@spoke4:~$ ping -qc2 9.9.9.9
```

```
PING 9.9.9.9 (9.9.9.9) 56(84) bytes of data.

--- 9.9.9.9 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 28.778/29.342/29.907/0.564 ms
```

2. Accéder aux autres routeurs ***Spoke***.

Le script ci-dessous peut être exécuté sur n'importe quel routeur ***Spoke*** et permet de vérifier qu'aucun paquet n'est perdu.

```
for addr in "10.47.2.2" "10.47.4.2" "10.47.6.2" "10.47.8.2"
do
    ping -qc2 $addr
done
```

5. Ajouter les réseaux de services distants

Dans cette section, on s'intéresse aux rôles et fonctionnalités suivantes :

- Configurer un commutateur virtuel avec une interface commutée
- Router les réseaux d'hébergement via OSPF

5.1. Ajouter un commutateur virtuel

Voici la liste des questions traitées dans le document *Routage inter-VLAN et protocole PPPoE*

Q231. Quel est le paquet à installer pour ajouter un commutateur virtuel au routeur ***Spoke*** ?

Rechercher le mot clé openvswitch dans la liste des paquets.

Voici un exemple de recherche.

```
apt search ^openvswitch
```

C'est le paquet `openvswitch-switch` qui nous intéresse. On l'installe.

```
sudo apt -y install openvswitch-switch
```

Q232. Comment déclarer un commutateur à l'aide de l'outil netplan.io ?

Consulter la documentation de ***Netplan*** pour obtenir les informations sur la configuration des commutateurs virtuels `openvswitch` à l'adresse *Netplan documentation*.

On peut aussi rechercher les informations dans les fichiers exemples fournis avec le paquet netplan.io.

Voici un exemple de recherche.

```
find /usr/share/doc/netplan* -type f -iname "openvswitch*"
/usr/share/doc/netplan/examples/openvswitch.yaml
```

Q233. Quelles sont les modifications à apporter au fichier de déclaration YAML `/etc/netplan/enp0s1.yaml` pour créer le commutateur `asw-host` ?

Voici une copie du fichier `/etc/netplan/enp0s1.yaml` qui contient les instructions de création du commutateur `asw-host` seul.

```
network:
  version: 2
  ethernets:
    enp0s1:
      dhcp4: false
      dhcp6: false
      accept-ra: false

    openvswitch: {}

  bridges:
    asw-host:
      openvswitch: {}

  vlans:
    enp0s1.440: # VLAN violet
      id: 440
      link: enp0s1
      addresses:
        - fe80:1b8::2/64
    enp0s1.441: # VLAN orange
      id: 441
      link: enp0s1
      addresses: []
```

On applique les nouvelles déclarations.

```
sudo netplan apply
```

On vérifie que le nouveau commutateur a bien été créé dans la base `Open vSwitch`.

```
sudo ovs-vsctl show
```

```
e288cc30-e290-44ae-8ed1-5e2a8d184033
    Bridge asw-host
        fail_mode: standalone
        Port asw-host
            Interface asw-host
                type: internal
    ovs_version: "3.4.0"
```

Q234. Comment ajouter une nouvelle interface virtuelle commutée (***Switched Virtual Interface***) qui servira de passerelle par défaut pour tous les hôtes du réseau d'hébergement du site distant ?

Rechercher dans la documentation Netplan des exemples de déclarations d'interfaces de type SVI appartenant à des VLANs.

Voici une nouvelle copie du fichier `/etc/netplan/enp0s1.yaml` auquel on ajouté la déclaration d'une interface `vlan40` avec les adresses IPv4 et IPv6 conformes au contexte de la maquette utilisée pour la rédaction de ce document.

```
network:
  version: 2
  ethernets:
    enp0s1:
      dhcp4: false
      dhcp6: false
      accept-ra: false

  openvswitch: {}

  bridges:
    asw-host:
      openvswitch: {}

  vlans:
    enp0s1.440: # VLAN violet
      id: 440
      link: enp0s1
      addresses:
        - fe80:1b8::2/64
    enp0s1.441: # VLAN orange
      id: 441
      link: enp0s1
      addresses: []
    vlan40:     # VLAN vert
      id: 40
      link: asw-host
      addresses:
        - 203.0.113.1/24
        - fda0:7a62:28::1/64
        - fe80:28::1/64
```

5.2. Router les réseaux d'hébergement depuis les Hubs

La configuration mise en place dans cette section conditionne le routage des réseaux d'hébergement des services des sites distants à l'ouverture de session PPP.

Q235. Comment créer les routes statiques vers les réseaux d'hébergement sur le routeur ***Hub*** ?

Créer un script exécutable par protocole réseau dans lequel on utilise les noms d'interfaces PPP dérivés des options `-u` utilisées dans les paramètres de configuration des deux serveurs PPPoE.

Pour IPv4, c'est le répertoire est `/etc/ppp/ip-up.d/` qui doit contenir le script exécutable `staticroute`.

```
cat << 'EOF' | sudo tee /etc/ppp/ip-up.d/staticroute
#!/bin/bash

if [ -z "${CONNECT_TIME}" ]; then
    case "${PPP_IFACE}" in
        "ppp0")
            ip route add 10.0.10.0/24 dev ${PPP_IFACE}
            ;;
        "ppp1")
            ip route add 10.0.20.0/24 dev ${PPP_IFACE}
            ;;
    esac
fi
EOF
```

```
sudo chmod +x /etc/ppp/ip-up.d/staticroute
```

Pour IPv6, c'est le répertoire est `/etc/ppp/ipv6-up.d/` qui doit contenir le script exécutable appelé `staticroute`.

```
cat << 'EOF' | sudo tee /etc/ppp/ipv6-up.d/staticroute
#!/bin/bash

if [ -z "${CONNECT_TIME}" ]; then
    case "${PPP_IFACE}" in
        "ppp0")
            ip -6 route add fda0:7a62:a::/64 dev ${PPP_IFACE}
            ;;
        "ppp1")
            ip -6 route add fda0:7a62:14::/64 dev ${PPP_IFACE}
            ;;
    esac
fi
EOF
```

```
sudo chmod +x /etc/ppp/ipv6-up.d/staticroute
```

Une fois de plus, il faut relancer les sessions PPP pour observer les résultats et lancer les tests ICMP.

Auparavant, on peut afficher les tables de routages IPv4 et IPv6 du routeur ***Hub*** pour vérifier la présence des nouvelles routes statiques.

- En direction du premier routeur ***Spoke***.

```
ip route ls dev ppp0
```

```
10.0.10.0/24 scope link
10.44.1.2 proto kernel scope link src 10.44.1.1
```

```
ip -6 route ls dev ppp0
```

```
fda0:7a62:a::/64 metric 1024 pref medium
fe80::2c4a:3bb9:eead:8fa5 proto kernel metric 256 pref medium
fe80::cca6:346e:80d0:20cf proto kernel metric 256 pref medium
```

- En direction du second routeur ***Spoke***.

```
ip route ls dev ppp1
```

```
10.0.20.0/24 scope link
10.44.3.2 proto kernel scope link src 10.44.3.1
```

```
ip -6 route ls dev ppp1
```

```
fda0:7a62:14::/64 metric 1024 pref medium
fe80::b4d6:f38c:a930:c8f5 proto kernel metric 256 pref medium
fe80::f0fe:10bd:88f0:cb26 proto kernel metric 256 pref medium
```

Q236. Comment l'ajout de route vers les réseaux d'hébergement est-il répercuté par FRR ?

Afficher les tables de routage des routeurs ***Hub*** après avoir relancé les ouvertures de sessions PPP sur les quatre routeurs ***Spoke***.

Après redémarrage des services PPP sur les routeurs ***Spoke***, les routes vers les réseaux d'hébergement sont marquées `K>*`. Ce sont des entrées appartenant à la catégorie ***kernel***.

Voici deux extraits des tables de routage du routeur ***Hub numéro 1***.

```
sh ip route kernel
```

```
Codes: K - kernel route, C - connected, L - local, S - static,
       R - RIP, O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,
       f - OpenFabric, t - Table-Direct,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup
       t - trapped, o - offload failure

K>* 0.0.0.0/0 [0/0] via 172.20.120.1, enp0s1.120, 07:19:49
K>* 10.0.1.0/24 [0/0] is directly connected, ppp0, 00:19:42
K>* 10.0.2.0/24 [0/0] is directly connected, ppp1, 00:18:21
```

```
sh ipv6 route kernel
```

```
Codes: K - kernel route, C - connected, L - local, S - static,
       R - RIPng, O - OSPFv3, I - IS-IS, B - BGP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,
       f - OpenFabric, t - Table-Direct,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup
       t - trapped, o - offload failure

K>* ::/0 [0/1024] via fe80:78::1, enp0s1.120 onlink, 07:21:03
K>* 2001:678:3fc:78::/64 [0/512] is directly connected, enp0s1.120, 07:21:03
K>* fda0:7a62:1::/64 [0/1024] is directly connected, ppp0, 00:20:56
K>* fda0:7a62:2::/64 [0/1024] is directly connected, ppp1, 00:19:35
```

Q237. Comment redistribuer ces nouvelles routes dans le protocole OSPF ?

Reprendre la section sur la redistribution des routes connectées en l'adaptant à la catégorie ***kernel***.

On ajoute une nouvelle instruction redistribute dans les configurations des démons OSPF de chacun des routeurs ***Hub***.

Voici deux extraits de configuration pour le routeur ***Hub numéro 2***.

```
sh run ospfd
```

```
Building configuration...

Current configuration:
!
frr version 10.1.1
frr defaults traditional
hostname hub2
log syslog informational
service integrated-vtysh-config
!
interface enp0s1.470
 ip ospf area 0
exit
!
router ospf
 ospf router-id 2.0.0.4
 log-adjacency-changes detail
 redistribute kernel
 redistribute connected
exit
!
end
```

```
sh run ospf6d
```

```
Building configuration...

Current configuration:
!
frr version 10.1.1
frr defaults traditional
hostname hub2
log syslog informational
service integrated-vtysh-config
!
interface enp0s1.470
 ipv6 ospf6 area 0
exit
!
router ospf6
 ospf6 router-id 2.0.0.6
 log-adjacency-changes
 redistribute kernel
 redistribute connected
exit
!
end
```

Q238. Comment vérifier que la redistribution de route est effective ?

En affichant les entrées OSPF des tables de routage d'un routeur ***Hub***, on doit voir apparaître les réseaux d'hébergement accessibles via le ***Hub*** opposé.

Voici deux exemples sur le routeurs ***Hub numéro 2***.

```
sh ip route ospf
```

```
Codes: K - kernel route, C - connected, L - local, S - static,
       R - RIP, O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,
       f - OpenFabric, t - Table-Direct,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup
       t - trapped, o - offload failure

O>* 10.0.1.0/24 [110/20] via 172.20.70.1, enp0s1.470, weight 1, 00:27:03
O>* 10.0.2.0/24 [110/20] via 172.20.70.1, enp0s1.470, weight 1, 00:27:03
O>* 10.47.2.2/32 [110/20] via 172.20.70.1, enp0s1.470, weight 1, 00:34:51
O>* 10.47.4.2/32 [110/20] via 172.20.70.1, enp0s1.470, weight 1, 00:33:30
O   172.20.70.0/29 [110/10] is directly connected, enp0s1.470, weight 1, 06:22:48
O   172.20.120.0/23 [110/20] via 172.20.70.1, enp0s1.470, weight 1, 03:28:25
```

```
sh ipv6 route ospf6
```

```
Codes: K - kernel route, C - connected, L - local, S - static,
       R - RIPng, O - OSPFv3, I - IS-IS, B - BGP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,
       f - OpenFabric, t - Table-Direct,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup
       t - trapped, o - offload failure

O   2001:678:3fc:78::/64 [110/20] via fe80::baad:caff:fefe:5, enp0s1.470, weight 1, 00:29:02
O>* fda0:7a62:1::/64 [110/20] via fe80::baad:caff:fefe:5, enp0s1.470, weight 1, 00:29:02
O>* fda0:7a62:2::/64 [110/20] via fe80::baad:caff:fefe:5, enp0s1.470, weight 1, 00:29:02
```

5.3. Installer et configurer Incus

Dans cette section, on s'intéresse aux rôles et fonctionnalités suivantes :

- Installer le gestionnaire de conteneurs système Incus
- Appliquer une configuration initiale
- Configurer l'adressage automatique pour les conteneurs

Pour la partie installation, on reprend les questions du document *Routage inter-VLAN et protocole PPPoE*.

Q239. Comment installer le gestionnaire de conteneurs Incus ?

Lancer une recherche dans la liste des paquets Debian.

Le paquet s'appelle tout simplement incus.

```
apt search ^incus
```

```
sudo apt -y install incus
```

Q240. Comment faire pour que l'utilisateur normal `etu` devienne administrateur et gestionnaire des conteneurs ?

Rechercher le nom du groupe système correspondant à l'utilisation des outils Incus.

Il faut que l'utilisateur normal appartienne au groupes systèmes `incus` et `incus-admin` pour qu'il ait tous les droits sur la gestion des conteneurs.

```
grep incus /etc/group
incus:x:990:
incus-admin:x:989:
```

```
sudo adduser etu incus
sudo adduser etu incus-admin
```


Avertissement

Attention ! Il faut se déconnecter/reconnecter pour bénéficier de la nouvelle attribution de groupe. On peut utiliser les commandes groups ou id pour vérifier le résultat.

```
groups
etu adm sudo users incus-admin incus
```

Pour la partie configuration initiale, on reprend l'utilisation d'un fichier YAML proposée dans le document *Introduction au routage dynamique OSPF avec FRRouting*.

Q241. Comment administrer et initialiser le gestionnaire de conteneurs *Incus* ?

Consulter la section *Réseau d'hébergement de conteneurs*.

Pour commencer, l'utilisateur normal doit appartenir aux deux groupes système `incus` et `incus-admin`. On reprend la démarche précédente.

```
for grp in incus incus-admin
do
    sudo adduser etu $grp
done
```

Comme l'affectation de groupe se joue à la connexion, il faut se déconnecter/reconnecter pour rendre l'attribution effective.

```
groups
```

```
etu adm sudo users frrvty frr incus-admin incus
```

Pour l'initialisation du gestionnaire de conteneur, on utilise un fichier de déclaration préparé en amont. Voici une copie du fichier `incus.yaml` pour le routeur `R2`.

```
config: {}
networks: []
storage_pools:
- config: {}
  description: ""
  name: default
  driver: dir
profiles:
- config: {}
  description: ""
  devices:
    eth0:
      name: eth0
      nictype: bridged
      parent: asw-host
      type: nic
      vlan: 20
    root:
      path: /
      pool: default
      type: disk
  name: default
projects: []
cluster: null
```

L'initialisation se fait à partir de ce fichier.

```
cat incus.yaml | incus admin init --preseed
```

On peut afficher le résultat de cette initialisation du gestionnaire de conteneurs avec l'instruction suivante.

```
incus profile show default
```

En cas d'erreur, il est aussi possible d'éditer le profil par défaut.

```
incus profile edit default
```

> **Important**
>
> Les opérations de cette question doivent être réalisées sur les trois routeurs en adaptant les numéros de VLAN de réseau d'hébergement.

Q242. Comment mettre en place l'adressage automatique IPv4 et IPv6 dans le réseau d'hébergement ?

Consulter la section ***Adressage automatique dans le réseau d'hébergement*** du document *Routage inter-VLAN et protocole PPPoE*

On créé un fichier de configuration pour le service dnsmasq en l'adaptant au contexte du plan d'adressage de chaque routeur. Voici un exemple pour le routeur `R1` de la maquette.

```
cat << EOF | sudo tee /etc/dnsmasq.conf
# Specify Container VLAN interface
interface=vlan10

# Enable DHCPv4 on Container VLAN
dhcp-range=10.10.0.20,10.10.0.200,3h

# Enable IPv6 router advertisements
enable-ra

# Enable SLAAC
dhcp-range=::,constructor:vlan10,ra-names,slaac

# Optional: Specify DNS servers
dhcp-option=option:dns-server,172.16.0.2,9.9.9.9
dhcp-option=option6:dns-server,[2001:678:3fc:3::2],[2620:fe::fe]

# Avoid DNS listen port conflict between dnsmasq and systemd-resolved
port=0
EOF
```

Une fois le fichier de configuration en place, on peut relancer le service et afficher son état.

```
sudo systemctl restart dnsmasq
systemctl status dnsmasq
```

> **Important**
>
> Une fois encore, les opérations de cette question doivent être réalisées sur les trois routeurs en adaptant le nom de l'interface et les adresses IPv4 du réseau d'hébergement.

Q243. Comment créer 3 conteneurs avec un adressage ?

Consulter la section ***Configuration et lancement des conteneurs*** du document *Routage inter-VLAN et protocole PPPoE*.

On lance la création des 3 conteneurs demandés avec une boucle.

```
for i in {0..2}; do incus launch images:debian/trixie c$i; done
```

```
incus ls
```

```
+------+---------+--------------------+-----------------------------------------------+-----------+-----------+
| NAME |  STATE  |        IPV4        |                     IPV6                      |   TYPE    | SNAPSHOTS |
+------+---------+--------------------+-----------------------------------------------+-----------+-----------+
| c0   | RUNNING | 10.20.0.33 (eth0)  | fd14:ca46:3864:14:216:3eff:fe2c:c5b9 (eth0) | CONTAINER | 0         |
+------+---------+--------------------+-----------------------------------------------+-----------+-----------+
| c1   | RUNNING | 10.20.0.87 (eth0)  | fd14:ca46:3864:14:216:3eff:fe76:f0d0 (eth0) | CONTAINER | 0         |
+------+---------+--------------------+-----------------------------------------------+-----------+-----------+
| c2   | RUNNING | 10.20.0.153 (eth0) | fd14:ca46:3864:14:216:3eff:fe2c:d169 (eth0) | CONTAINER | 0         |
+------+---------+--------------------+-----------------------------------------------+-----------+-----------+
```

6. Bilan et conclusion

Arrivé à cette étape, la totalité des outils nécessaires à l'interconnexion des réseaux LAN et WAN est en place et il est temps de dresser le bilan avec les ultimes tests de communication entre les services des réseaux distants.

6.1. Tester toutes les communications

Après avoir relevé les adresses attribuées aux conteneurs des quatre routeurs ***Spoke***, on peut lancer une boucle de tests ICMP pour qualifier les communications.

Voici un exemple réalisé dans le contexte de la maquette utilisée pour rédiger ce document.

La commande suivante permet de liste les adresses attribuées au conteneurs sur chaque réseau d'extrémité.

```
incus ls -c 46 -f csv
```

Les résultats au format CSV peuvent être accumulés dans un fichier d'adresses qui sert pour les tests ICMP. Voici un extrait du fichier des adresses de conteneurs :

```
10.0.1.43 (eth0),fda0:7a62:1:0:216:3eff:fec9:2051 (eth0)
10.0.1.57 (eth0),fda0:7a62:1:0:216:3eff:fe4e:167c (eth0)
10.0.1.174 (eth0),fda0:7a62:1:0:216:3eff:fe18:3562 (eth0)
```

Pour qualifier l'accord de niveau de service ou ***Service Level Agreement*** (SLA), on peut utiliser un code comme celui du script ci-dessous qui teste la connectivité à partir de toutes les adresses fournies.

```
#!/bin/bash

# Test if input file exists
if [[ ! -f "$1" ]]; then
    echo "Error: missing or non-existent input file"
    exit 1
fi

file=$1
echo "-------------------------------------------------------------"
echo "CSV input file must be generated from the following command:"
echo "   incus ls -c 46 -f csv"
echo "-------------------------------------------------------------"

# Declare IPv4 and IPv6 array addresses
declare -a ipv4_addresses
declare -a ipv6_addresses

# Read CSV file content
mapfile -t lines < "$file"

# Extract IP addresses from each line
for line in "${lines[@]}"; do
     IFS=',' read -r ipv4 ipv6 <<< "$line"
     ipv4_addresses+=("${ipv4%% *}")
     ipv6_addresses+=("${ipv6%% *}")
done

echo "Launch pings..."
fail=false
for address in "${ipv4_addresses[@]}" "${ipv6_addresses[@]}"
do
     # Capture ping output
     ping_output=$(ping -qc2 "$address" | tr '\n' ' ')
     # Get packet loss number
     packets_lost=$(echo "$ping_output" |\
        grep -Eo '[[:digit:]]+\% packet loss' |\
        grep -Eo '[[:digit:]]+')

     if [[ $packets_lost != 0 ]]; then
        echo "Test failed : $packets_lost% lost packets for $address"
        fail=true
     else
        echo -n "."
     fi
done

if [[ fail==false ]]; then
        echo "Success!"
fi

exit 0
```

Voici un exemple du résultat d'exécution du script sur la maquette.

```
bash icmp-tests.sh addresses.csv
-------------------------------------------------------------
CSV input file must be generated from the following command:
   incus ls -c 46 -f csv
-------------------------------------------------------------
Launch pings...
.......................Success!
```

6.2. Afficher les tables de routage complètes

Pour faire le bilan sur le routage dynamique avec les protocoles OSPFv2 et OSPFv3, on affiche les tables de routage dans lesquelles tous les réseaux doivent figurer.

En se plaçant sur le routeur *Hub numéro 2*, on obtient les entrées suivantes :

```
sh ip route
```

```
Codes: K - kernel route, C - connected, L - local, S - static,
       R - RIP, O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,
       f - OpenFabric, t - Table-Direct,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup
       t - trapped, o - offload failure

K>* 0.0.0.0/0 [0/0] via 172.20.120.1, enp0s1.120, 20:50:24
O>* 10.0.1.0/24 [110/20] via 172.20.70.1, enp0s1.470, weight 1, 14:53:17
O>* 10.0.2.0/24 [110/20] via 172.20.70.1, enp0s1.470, weight 1, 14:53:17
K>* 10.0.3.0/24 [0/0] is directly connected, ppp0, 14:48:53
K>* 10.0.4.0/24 [0/0] is directly connected, ppp1, 14:48:25
O>* 10.47.2.2/32 [110/20] via 172.20.70.1, enp0s1.470, weight 1, 15:01:05
O>* 10.47.4.2/32 [110/20] via 172.20.70.1, enp0s1.470, weight 1, 14:59:44
L>* 10.47.6.1/32 is directly connected, ppp0, 14:48:53
C>* 10.47.6.2/32 is directly connected, ppp0, 14:48:53
L>* 10.47.8.1/32 is directly connected, ppp1, 14:48:25
C>* 10.47.8.2/32 is directly connected, ppp1, 14:48:25
O   172.20.70.0/29 [110/10] is directly connected, enp0s1.470, weight 1, 20:49:02
C>* 172.20.70.0/29 is directly connected, enp0s1.470, 20:50:24
L>* 172.20.70.2/32 is directly connected, enp0s1.470, 20:50:24
O   172.20.120.0/23 [110/20] via 172.20.70.1, enp0s1.470, weight 1, 17:54:39
C>* 172.20.120.0/23 is directly connected, enp0s1.120, 20:50:24
L>* 172.20.120.3/32 is directly connected, enp0s1.120, 20:50:24
```

Les entrées marquées `O>*` de la table de routage ci-dessus correspondent aux routes apprises via le protocole OSPF qui ont été retenues par le sous-système réseau du noyau du routeur. Vu du routeur ***Hub numéro 2***, on identifie quatre routes de ce type : deux pour les réseaux d'hébergement et deux pour les réseaux de transit. Ces quatre entrées correspondent aux routeurs ***Spoke*** raccordés au ***Hub numéro 1***.

```
sh ipv6 route
```

```
Codes: K - kernel route, C - connected, L - local, S - static,
       R - RIPng, O - OSPFv3, I - IS-IS, B - BGP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,
       f - OpenFabric, t - Table-Direct,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup
       t - trapped, o - offload failure

K>* ::/0 [0/1024] via fe80:78::1, enp0s1.120 onlink, 20:53:20
O   2001:678:3fc:78::/64 [110/20] via fe80::baad:caff:fefe:5, enp0s1.470, weight 1, 14:56:06
K>* 2001:678:3fc:78::/64 [0/512] is directly connected, enp0s1.120, 20:53:20
L>* 2001:678:3fc:78:baad:caff:fefe:8/128 is directly connected, enp0s1.120, 20:53:20
O>* fda0:7a62:1::/64 [110/20] via fe80::baad:caff:fefe:5, enp0s1.470, weight 1, 14:56:06
O>* fda0:7a62:2::/64 [110/20] via fe80::baad:caff:fefe:5, enp0s1.470, weight 1, 14:56:06
K>* fda0:7a62:3::/64 [0/1024] is directly connected, ppp0, 14:51:49
K>* fda0:7a62:4::/64 [0/1024] is directly connected, ppp1, 14:51:21
C * fe80::/64 is directly connected, enp0s1.477, 20:53:20
C * fe80::/64 is directly connected, enp0s1.476, 20:53:20
C * fe80::/64 is directly connected, enp0s1.475, 20:53:20
C * fe80::/64 is directly connected, enp0s1.470, 20:53:20
C * fe80::/64 is directly connected, enp0s1.120, 20:53:20
C * fe80::/64 is directly connected, enp0s1.478, 20:53:20
C>* fe80::/64 is directly connected, enp0s1, 20:53:20
C>* fe80::4407:6511:26c1:58d1/128 is directly connected, ppp1, 14:51:21
C>* fe80::551a:aa5f:6325:8f85/128 is directly connected, ppp0, 14:51:49
C>* fe80:1d6::/64 is directly connected, enp0s1.470, 20:53:20
C>* fe80:1db::/64 is directly connected, enp0s1.475, 20:53:20
C>* fe80:1dd::/64 is directly connected, enp0s1.477, 20:53:20
```

Comme on l'a déjà vu, toutes les entrées marquées `C>*` correspondent aux interfaces actives ou connectées. On peut afficher le résumé de l'état des interfaces du routeur pour voir la correspondance.

```
 sh int brief
```

```
Interface       Status  VRF             Addresses
---------       ------  ---             ---------
enp0s1          up      default         fe80::baad:caff:fefe:8/64
enp0s1.120      up      default         172.20.120.3/23
                                        + 2001:678:3fc:78:baad:caff:fefe:8/64
enp0s1.470      up      default         172.20.70.2/29
                                        + fe80:1d6::2/64
enp0s1.475      up      default         + fe80:1db::1/64
enp0s1.476      up      default         fe80::baad:caff:fefe:8/64
enp0s1.477      up      default         + fe80:1dd::1/64
enp0s1.478      up      default         fe80::baad:caff:fefe:8/64
lo              up      default
ppp0            up      default         10.47.6.1/32
                                        fe80::1cd0:13ee:bdd1:22bb/128
ppp1            up      default         10.47.8.1/32
                                        fe80::7993:fc0f:f5c8:d0c4/128
```

6.3. Pour conclure...

Maintenant que l'on a validé l'ensemble des échanges entre les conteneurs situés aux extrémités de la topologie d'interconnexion de réseaux, on peut passer à la conclusion.

Ce dernier document de la série des travaux pratiques fournit une synthèse détaillée de tous les modes d'interconnexion de réseaux locaux (LAN) et étendus (WAN) en utilisant des protocoles comme PPPoE et OSPF. Il couvre les principes de configuration des routeurs d'une architecture ***Hub & Spoke***, l'installation et la configuration de la solution FRRouting avec la mise en place du routage dynamique OSPF pour IPv4 et IPv6.

On peut considérer que nous sommes arrivés à la limite de l'étude des interconnexions de réseaux sans automatisation. Pour aller plus loin, l'automatisation ainsi que l'intégration et le déploiement continu permettraient de déployer et gérer ces configurations à grande échelle, réduisant ainsi les erreurs manuelles et améliorant l'efficacité opérationnelle des réseaux d'entreprise.