tetaneutral.net

Rapport écrit du TER de Master 2

Maquettage de réseau virtualisé avec interconnexion de niveaux 2 et 3

Auteurs :
Vincent Passama
Othmane Elmouden

Responsables :
M. Philippe LATU
M. Laurent GUERBY

20 Mars 2013

Sommaire

Remerciements

Nous souhaitons remercier M. Laurent GUERBY, président de l'association tetaneutral.net, pour ses conseils et son amabilité qui nous ont aidé et permis de réaliser notre sujet de TER dans de bonnes conditions.

Nous remerions également M. Philippe LATU, enseignant à l'IUP STRI et tuteur de notre TER pour son aide précieuse et ses conseils qui nous ont guidé et orienté dans le déroulement de ce TER.

Enfin nous tenons à remercier notre directeur M. André AOUN ainsi que l'ensemble du personnel enseignant et administratif STRI.

Résumé

Ce TER s'est déroulé en partenariat avec l'association tetaneutral.net durant 2 mois. Il s'est ponctué de réunions hebdomadaires avec l'enseignant tuteur et le responsable de l'association.

L'association tetaneutral.net fondée le 3 Janvier 2011 exerce les fonctions de fournisseur d'accès à Internet, d'hébergeur internet et d'opérateur sans but lucratif. Le but étant de garantir la neutralité du réseau et promouvoir la compréhension de l'Internet à un large public.

Dans le but d'étendre et de modifier son réseau de fournisseur, comprenant des routeurs BGP regroupés dans un AS attribué à l'association et adjacent à d'autres AS de l'Internet, tetaneutral.net souhaite bénéficier d'une solution de maquettage virtuel. Cette solution comprendrait une mise en place et une gestion d'un réseau de machines virtuelles auxquelles ont aurait attribué des fonctionnalités de routage, notamment la gestion des protocoles BGP et OSPF.

Lors de ce TER nous avons donc créée un parc de machines virtuelles en utilisant l'outil KVM sur un hôte hébergeant le système d'exploitation Debian. Avec ce parc de machines, nous nous sommes documentés et avons utilisé un ensemble d'outils tels que BIRD et Open vSwitch afin de mettre en place diverses topologies de test pour terminer sur celle souhaitée.

Finalement nous avons atteint une partie des objectifs souhaités et nous avons pu en apprendre plus sur la virtualisation et l'échange de routes sur Internet grâce au protocole BGP. Grâce à nos acquis en système linux et nos à connaissances en protocoles réseaux nous avons pu mettre en application le savoir faire enseigné au sein de la formation STRI.

Introduction

L'association tetaneutral.net

L'association tetaneutral.net fondée le 3 Janvier 2011 exerce les fonctions de fournisseur d'accès à Internet, d'hébergeur internet et d'opérateur sans but lucratif. Le but étant de garantir la neutralité du réseau et promouvoir la compréhension de l'Internet à un large public.

Cette assiociation propose aux membres certains services payants afin de couvrir les frais d'infrastructure. L'adhésion se fait à un tarif libre annuel (cotisation normale de 20 euros).

tetaneutral.net propose en outre l'hébergement de machines physiques et virtuelles, l'accès à Internet et de la certification OpenVPN. Pour ce faire, l'association possède un réseau filaire en fibre optique ainsi qu'un réseau radio déployé dans Toulouse.

Description du TER

Problématique

Afin de modifier leur réseau de façon fiable, tetaneutral souhaite posséder un outil de maquettage manipulant des machines virtuelles. Pour ce faire, le réseau de machines virtuelles doit reprendre la topologie physique mise en place et les protocoles BGP et OSPF doivent être déployés de façon logique au sein des routeurs virtuels.

Moyens

Dans le but de répondre à ce sujet de TER nous avons utilisé des outils recommandés par notre tuteur et le responsable de l'association. Pour la virtualisation il s'agit d'utiliser KVM, Qemu et libvirt. En ce qui concerne le déploiement des interfaces virtuelles, Open vSwitch a été préconisé. Enfin pour la gestion des routeurs virtuels, nous avons travaillé sur BIRD et Quagga. Nous verrons en détail lors des parties consacrées à ces outils quels sont leur fonctionnement et leur utilité.

Organisation du travail

ChiliProject

Afin d'organiser un suivi et un planning, l'association a mis à notre disposition un outil de travail collaboratif nommé ChiliProject. Cet outil permet notamment de déployer un wiki regroupant liens, images, explications et aperçus de code sur une interface WEB bien pensée et claire. La prise en main est aisée et le rendu très lisible.

Via une fonction de suivi, des membres inscrits à l'interface peuvent recevoir des notifications de modification sur n'importe quel sujet du site. Ainsi nos tuteurs nous ont conseillé et suivi tout le long du projet, en ajoutant des conseils et des questions au sujet.

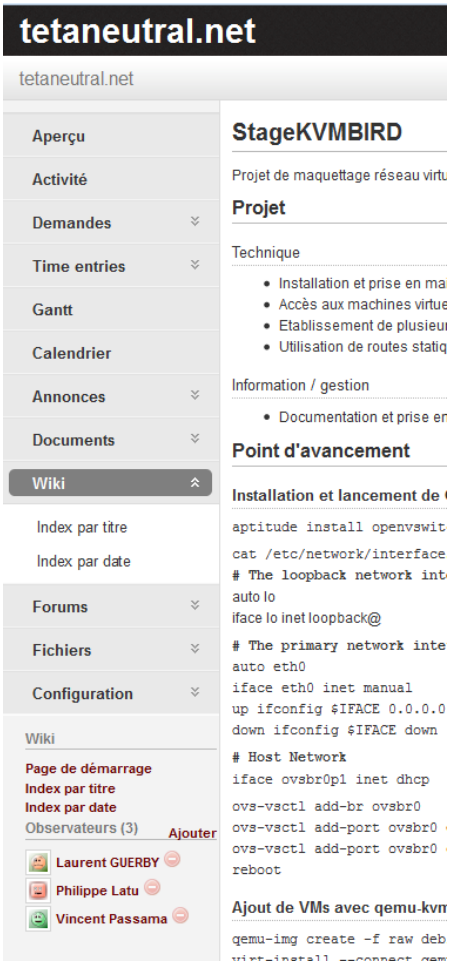

FIGURE 1 – Aperçu de ChiliProject

Réunions hebdomadaires

Lors du déroulement de ce TER, des réunions hebdomadaires ont été organisées selon les disponibilités du responsable de l'association. Ainsi nous avons pu bénéficier d'aide précieuse et d'échanges avec nos tuteurs afin de garder un objectif clair en tête et d'avancer sur la bonne voie.

Planning

Nous n'avons pas édité de planning sur le ChiliProject mais voici un aperçu du déroulement des recherches et de la mise en place de la maquette au cours du temps.

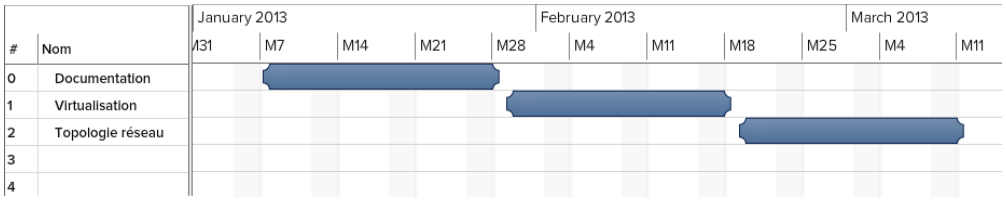

FIGURE 2 – Planning des tâches

Nous voyons qu'il y a eu 3 grandes étapes dans le déroulement du TER qui nous ont permis d'atteindre nos objectifs. La documentation était primordiale avant de prendre en main la plupart des outils de virtualisation et de mettre en place les topologies réseaux.

1 La virtualisation

1.1 Définition et avantages

La virtualisation est un moyen de faire fonctionner plusieurs systèmes d'exploitation sur une seule machine hôte (ou plusieurs). De ce fait celà permet de mutualiser les ressources physiques d'une machine et d'héberger plusieurs serveurs par exemple en utilisant au maximum les capacités de l'hôte. La virtualisation permet dans notre cas de simuler un réseau physique existant sur une seule machine et ce de façon aisée.

De ce fait nous verrons que la virtualisation avec KVM fonctionne à l'aide d'un hyperviseur, une couche logicielle permettant l'exécution de systèmes d'exploitation invités. Dans notre cas, il s'agit de QEMU-KVM, un fork de QEMU.

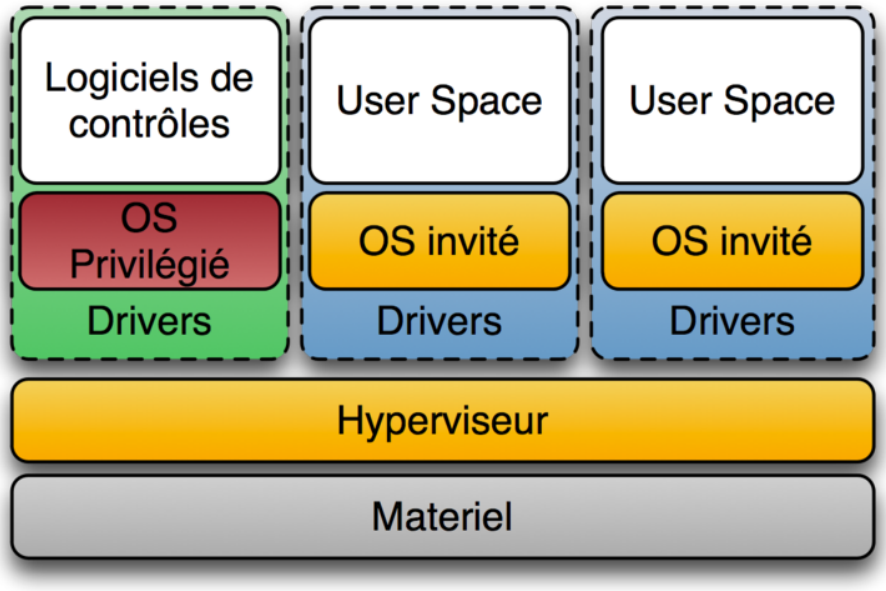

FIGURE 3 – Hyperviseur de type 1 – http://commons.wikimedia.org/wiki/

Afin de positionner tous les outils utilisés lors de la création d'une machine virtuelle, voici un schéma regroupant ceux-ci en couches :

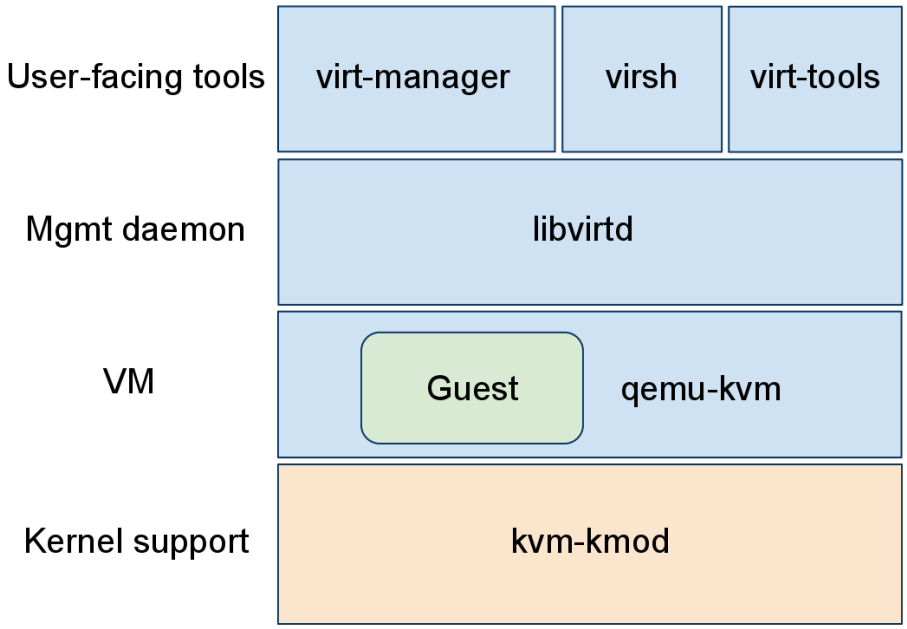

FIGURE 4 – Architecture de KVM et libvirt – http://www.ibm.com/

1.2 Outils utilisés

1.2.1 Kernel-Based Virtual Machine

KVM est solution de virtualisation libre pour Linux dérivée de QEMU. Cette solution intègre un module noyau et un module attaché aux architectures Intel VT ou AMD-V. KVM est directement intégré dans le noyau depuis la version 2.6.20 de 2007. Cet hyperviseur utilise notamment le module KQEMU pour accélérer l'émulation du matériel via virtio (paravirtualisation).

QEMU pour Quick EMUlator est quant à lui un émulateur libre de hardware qui permet la virtualisation des machines et l'exécution d'instances de systèmes d'exploitation.

1.2.2 libvirt

libvirt est une API, démon et outil de supervision libre pour des plateformes de virtualisation. Il procure notamment la possibilité d'utiliser un shell comme virsh ou un

superviseur graphique comme virt-manager afin de gérer ses machines virtuelles et de définir leur configuration matérielle. Il s'interface avec KVM pour procurer les outils nécessaires d'administration des VM.

1.3 Création de machines virtuelles

1.3.1 Installation des paquets

Pour tester tout d'abord si la virtualisation est possible :

```
egrep '^flags .*(vmx|svm)' /proc/cpuinfo >/dev/null && echo OK || echo KO
```

Afin d'installer et de configurer les machines virtuelles, voici les commandes d'usage :

```
aptitude install qemu-kvm libvirt-bin virtinst virt-viewer virt-manager
```

Afin de permettre à un utilisateur de manipuler les machines virtuelles, il suffit de le rajouter dans le groupe de libvirt et kvm :

```
addgroup user libvirt
addgroup user kvm
```

Il faut désormais relogger l'utilisateur et il pourra manipuler comme bon lui semble ses machines virtuelles.

1.3.2 Les images disque

Afin d'installer les machines, il faut tout d'abord créer une image disque permettant leur installation. Pour ce faire, divers format de fichiers sont utilisables :
- **RAW** : Format binaire brut. Les images à ce format utilisent seulement l'espace nécessaire par rapport à la taille allouée.
- **QCOW2** : Format dit "copy-on-write" intégrant des fonctionnalités plus évoluées comme la compression, le chiffrement et les snapshots.

Pour créer une image disque, voici un exemple de commandes :

```
qemu-img create -f qcow2 image.qcow2 2G
qemu-img create -f raw image.raw 2G
```

L'extension est définie par l'option "-f" et la taille est définie en fin de commande. Une fois l'image créée, il faut installer une machine virtuelle en l'utilisant.

1.3.3 Création d'une machine virtuelle

Afin de créer une machine virtuelle, on utilise cette commande :

```
virt-install --name=VM --hvm --noautoconsole --ram 1024 --file=/path/to/image.qcow2 --vnc
--os-type=linux --cdrom /path/to/debian-wheezy-DI-rc1-amd64-CD-1.iso --nonetworks
```

Nous remarquons ici plusieurs choses :

- Il faut utiliser des chemins absolus pour définir les emplacements des disques et images.
- On peut définir le type d'OS et la quantité maximale de mémoire vive attribuée.
- On peut ajouter un port VNC pour se connecter à la machine virtuelle.
- On peut nommer sa machine pour la reconnaître facilement dans l'ensemble du parc de machines virtuelles.
- Ici, on ne définit pa d'interface réseau car nous passerons via Open vSwitch (expliqué plus loin).

Une fois l'installation effectuée, nous pouvons visualiser la machine virtuelle via différentes méthodes :

- Via la console
- Via VNC
- Via virt-viewer
- Via virt-manager

Une fois créée, cette machine virtuelle possède un fichier de configuration en XML associé et créé automatiquement dans :

```
/etc/libvirt/qemu/image.xml
```

Dans ce fichier on peut éditer le matériel émulé et notamment les interfaces réseaux (ce qui nous intéressera plus tard). En annexe vous trouverez un exemple de fichier.

Virt-manager quant à lui permet également d'éditer les configurations et de superviser les machines virtuelles mais cette fois-ci en utilisant une interface graphique.

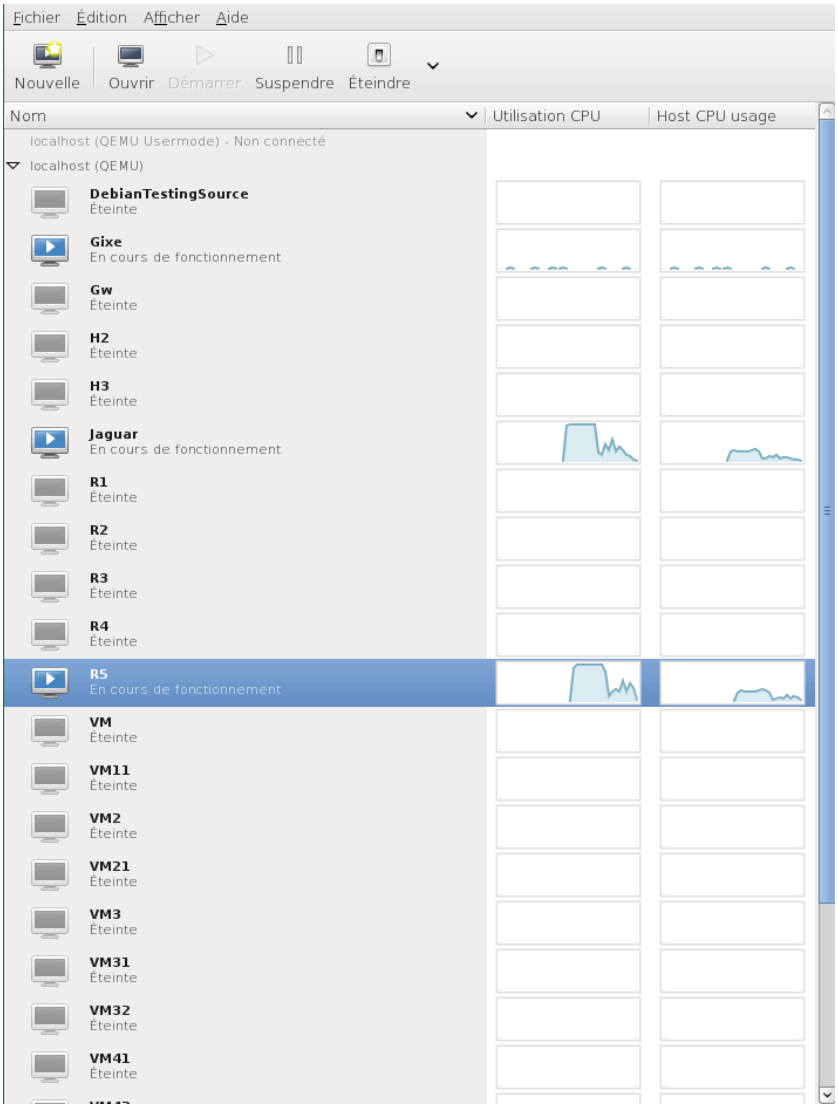

FIGURE 5 – Supervision des machines virtuelles via virt-manager

Virsh est un équivalent en mode console et prospose une multitude de fonctionnalités en bénéficiant de l'auto-complétion.

```
virsh # list
 Id    Name                           State
----------------------------------------------------
 1     Gixe                           running
 2     Jaguar                         running
 3     R5                             running

virsh # help
Grouped commands:

 Domain Management (help keyword 'domain'):
    attach-device                  attach device from an XML file
    attach-disk                    attach disk device
    attach-interface               attach network interface
    autostart                      autostart a domain
    blkdeviotune                   Set or query a block device I/O tuning parame
ters.
    blkiotune                      Get or set blkio parameters
    blockcopy                      Start a block copy operation.
    blockjob                       Manage active block operations
    blockpull                      Populate a disk from its backing image.
    blockresize                    Resize block device of domain.
    change-media                   Change media of CD or floppy drive
```

FIGURE 6 – Supervision des machines virtuelles via virsh

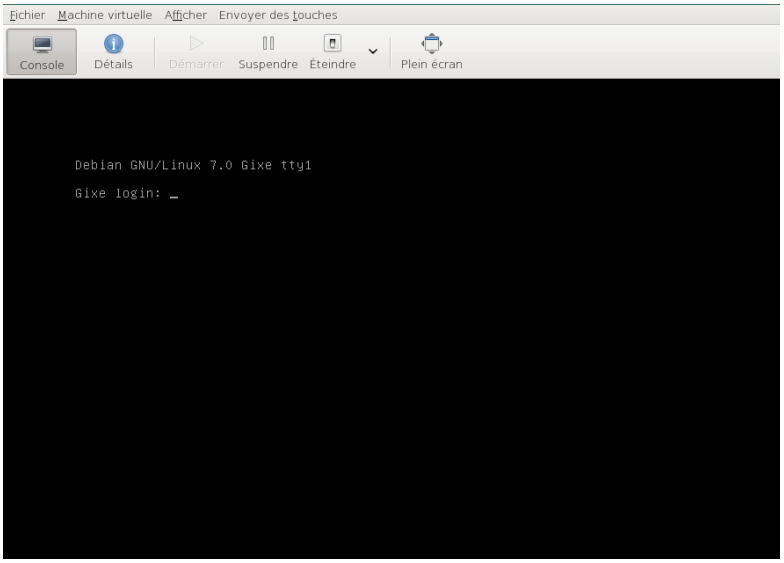

FIGURE 7 – Visualisation d'une machine virtuelle via virt-manager

1.3.4 Clônage des machines virtuelles

Une fois qu'une machine à été installée avec un système d'exploitation mis à jour, on peut préférer le clônage de cette machine afin de bénéficier d'un gain de temps non négligeable. A savoir que les possibilités sont immenses en matière de partage et clônage des machines virtuelles. Nous n'avons pas pu pousser très loin l'administration de plusieurs machines, comme reporter la mise à jour d'une machine sur les autres, utiliser les images à taille différentielle (une fonctionnalité de qcow2 qui permet de limiter l'espace disque utilisé en ne rajoutant que la différence de taille avec l'image source), etc...

Nous avons donc utilisé le clônage simple et efficace d'une machine installée et configurée proprement :

```
virt-clone --original image --name new_image
--file ./debianX.raw
```

Ainsi le parc de machines a pu être mis en place plus rapidement une fois ces étapes franchies.

1.3.5 Isolement du parc de VM par utilisateur

Afin de bien isoler les actions de chaque utilisateur par rapport à la création et la suppression des machines virtuelles, on peut choisir avec libvirt d'utiliser des URI pour se connecter à des instances d'hyperviseur différentes :

- **QEMU System** : Instance globale du système (commune à root et aux utilisateurs privilégiés).
- **QEMU Session** : Instance locale à l'utilisateur.

Pour ce faire on peut choisir pour chaque action utilisant une commande en relation avec libvirt d'utiliser différentes URI pour spécifier l'instance souhaitée :

```
virsh -c qemu:///system
virsh -c qemu:///session (URI par défaut)
```

On peut voir sur l'image qui suit que deux utilisateurs ayant accès à l'instance globale ont créé chacun une VM différente dans leur instance locale qui ne peut être modifiée que par eux-mêmes.

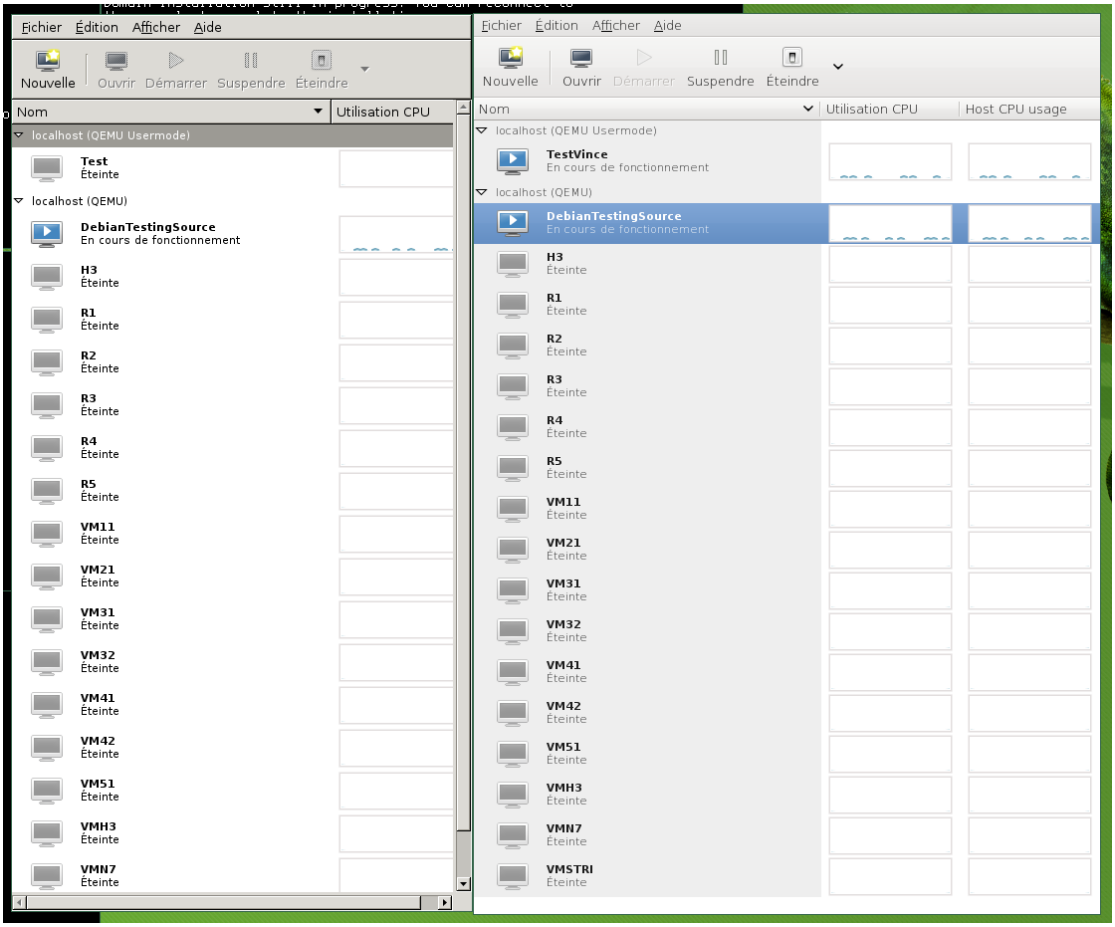

FIGURE 8 – Les instances de l'hyperviseur QEMU

2 Le pont virtuel

2.1 Définition et utilité

Afin de rendre les interfaces de chaque machine virtuelle visibles depuis l'extérieur, il faut utiliser une solution de bridge entre l'interface physique et l'interface virtuelle. Par défaut libvirt apporte une interface mais lors de ce TER nous nous sommes orientés vers une autre solution sur les conseils de notre tuteur enseignant.

2.2 Choix d'Open vSwitch

Ce choix a été proposé par notre enseignant tuteur pour diverses raisons :

- Il repousse les limites d'un mode bridge standard en intégrant une virtualisation complète d'un switch et l'apport des fonctionnalités qui en découlent.
- Il n'est pas connu par l'association tetaneutral.net et c'est un bon moyen de tester ses possibilités.
- Il pourrait (si souhaité) permettre la mise en place de VLAN entre les interfaces de chaque machine virtuelle.

Comme vous l'aurez deviné suite à ces raisons, Open vSwitch permet de créer un switch virtuel possédant les fonctionnalités d'un switch physique. Il possède une couche de compatibilité avec la solution de bridge intégrée à Linux sous le nom de "Brcompat".

2.3 Mise en place

2.3.1 Installation

Liste des commandes d'installation et configuration d'Open vSwitch :

```
aptitude install openvswitch-controller openvswitch-brcompat openvswitch-switch

cat /etc/default/openvswitch-switch

# BRCOMPAT: If 'yes' and the openvswitch-brcompat package is installed, then
# Linux bridge compatibility will be enabled.
```

```
BRCOMPAT=yes

ovs-vsctl add-br ovsbr0
ovs-vsctl add-port ovsbr0 eth0
ovs-vsctl add-port ovsbr0 ovsbr0p1 -- set interface ovsbr0p1 type=internal

service openvswitch-controller restart
service openvswitch-switch restart

cat /etc/network/interfaces

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet manual
up ifconfig $IFACE 0.0.0.0 up
down ifconfig $IFACE down

# Host Network
auto ovsbr0p1
iface ovsbr0p1 inet dhcp
```

Ici nous avons installé Open vSwitch, activé le mode de compatibilité et ajouté un switch virtuel avec un pont sur l'interface physique de la machine hôte.

2.3.2 Interfaces virtuelles

Afin de relier simplement les interfaces virtuelles au switch virtuel, nous avons rajouté des interfaces dans les fichiers de configuration des machines virtuelles :

```
virsh edit image

#Ajout de ces lignes dans le XML
<interface type='bridge'>
    <source bridge='ovsbr0'/>
    <virtualport type='openvswitch'/>
    <model type='virtio'/>
</interface>

# Sauvegarde puis vérification dans le fichier
cat /etc/libvirt/qemu/image.xml

<interface type='bridge'>
      <mac address='52:54:00:fc:4a:c6'/>
      <source bridge='ovsbr0'/>
      <virtualport type='openvswitch'>
        <parameters interfaceid='a6e93881-d7f7-5c2e-5e8e-67c52d163e48'/>
      </virtualport>
      <model type='virtio'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x04' function='0x0'/>
    </interface>
```

Nous voyons que pour chaque interface ajoutée, une adresse MAC est attribuée automatiquement et les paramètres de virtualisation sont complétés par libvirt. Ainsi nous avons ajouté autant d'interface que nécessaire pour chaque machine virtuelle.

En définitive, si nous le souhaitons, une machine virtuelle peut sortir du réseau de machines virtuelles pour communiquer via l'interface physique vers le monde extérieur et ainsi accéder à Internet.

Nous n'avons pas testé l'ajout de VLAN car tetaneutral.net n'en avait en définitive pas besoin dans un avenir proche bien qu'il soit intéressant de pouvoir le faire.

3 Les protocoles réseaux

3.1 Définitions

3.1.1 Border Gateway Protocol

BGP est un protocole d'échange de routes entre Autonomous Systems sur le réseau Internet. C'est un protocole à vecteur de chemin car il fonde les décisions de routage sur les chemins parcourus, les attributs de préfixes et diverses règles définies par chaque AS.

Il permet notamment la diffusion de grands volumes de données et des possibilités étendues sur les choix de routes. Il utilise également l'agrégation de routes afin de limiter la taille des tables de routage.

Un Autonomous System, abrégé en AS, ou Système Autonome, est un ensemble de réseaux informatiques IP intégrés à Internet et dont la politique de routage interne (routes à choisir en priorité, filtrage des annonces) est cohérente. Un AS est généralement sous le contrôle d'une entité/organisation unique, typiquement un fournisseur d'accès à Internet.

Au sein d'un AS, le protocole de routage est qualifié d'interne et entre deux AS, le routage est externe. Chaque AS est identifié par un numéro de 32 bits (depuis 2007) appelé ASN pour Autonomous System Number. Ce numéro est affecté par les Registres Internet Régionaux (RIR).

Les connexions entre deux voisins BGP (neighbors ou peers) sont configurées explicitement entre deux routeurs. Ils communiquent alors entre eux via une session TCP sur le port 179 initiée par l'un des deux routeurs. BGP est le seul protocole de routage à utiliser TCP comme protocole de transport.

Il existe deux versions de BGP : Interior BGP (iBGP) et Exterior BGP (eBGP). iBGP est utilisé à l'intérieur d'un Autonomous System alors que eBGP est utilisé entre deux AS.

Ces définitions sont largement inspirées deWikipédia.

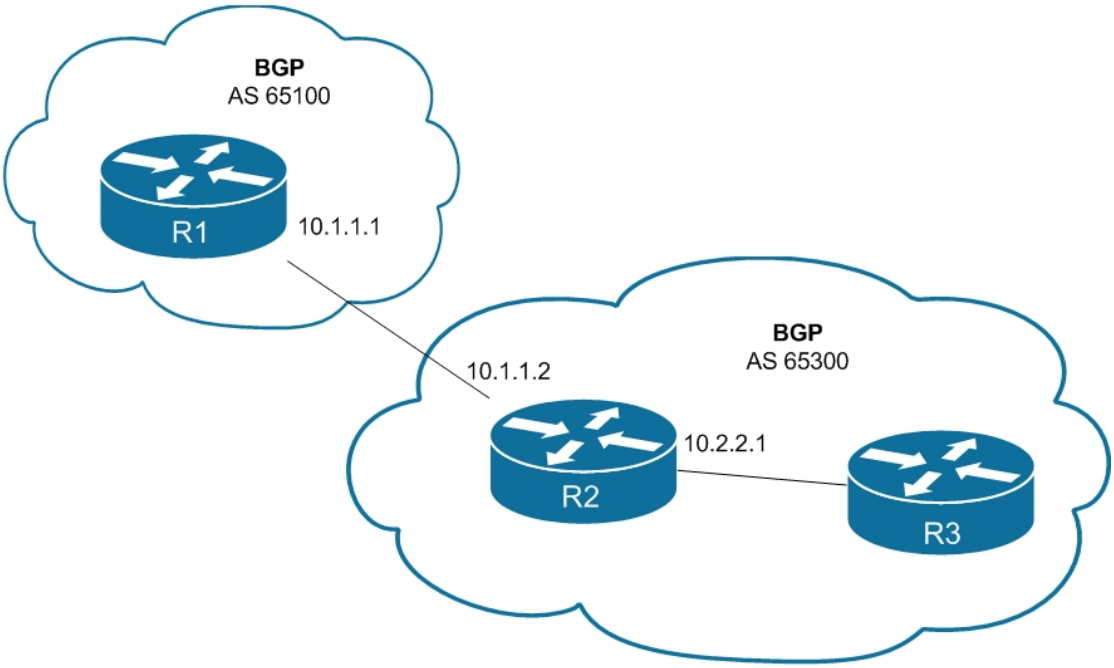

FIGURE 9 – 2 AS adjacents – http://www.ccnpguide.com/ccnp-route-642-902-bgp/

3.1.2 Open Shorted Path First

OSPF quant à lui est un protocole de routage interne dynamique basé sur l'algorithme de Dijkstra publié en 1959. Il a été développé par l'IETF à partir de 1987 pour remplacer RIP et a atteint la version 2 en 1997. La version 3 se base sur IPv6.

Edsger Dijkstra était un mathématicien et informaticien renommé Néerlandais. Il est mort le 6 Août 2002. Son algorithme issue de la théorie des graphes sert à résoudre le problème du plus court chemin dans un graphe.

Dans OSPF, chaque routeur établit des relations d'adjacence avec ses voisins immédiats en envoyant des messages hello à intervalle régulier. Chaque routeur communique ensuite la liste des réseaux auxquels il est connecté par des messages Link-state advertisements (LSA) propagés de proche en proche à tous les routeurs du réseau. L'ensemble des LSA forme une base de données de l'état des liens Link-State Database (LSDB) pour chaque aire, qui est identique pour tous les routeurs participants dans cette aire. Chaque routeur utilise ensuite l'algorithme de Dijkstra, Shortest Path First (SPF) pour déterminer la route la plus courte vers chacun des réseaux connus dans la LSDB.

Le bon fonctionnement d'OSPF requiert donc une complète cohérence dans le calcul SPF, il n'est donc par exemple pas possible de filtrer des routes ou de les résumer à l'intérieur d'une aire.

En cas de changement de topologie, de nouveaux LSA sont propagés de proche en proche, et l'algorithme SPF est exécuté à nouveau sur chaque routeur.

Afin d'éviter de propager la totalité de la base de données des liens et de limiter l'impact négatif du bagottement ou flapping (alternance rapide dans la disponibilité d'un lien), on segmente l'ensemble des routeurs en groupes connexes appelés aires, à la frontière desquels on peut procéder à des résumés. Chaque aire est distinguée par un nombre entier positif ou nul variant de 0 à 4 294 967 295, ce nombre est parfois exprimé en notation décimale pointée, de la même manière qu'une adresse IP. Chaque sous-réseau appartient à une seule aire.

Il existe toujours une aire dorsale (backbone area), area 0 ou encore area 0.0.0.0 à laquelle toutes les autres aires sont connectées.

Les aires sont logiquement contiguës. Si les routeurs qui constituent une aire ne sont pas physiquement contigus, alors des liens virtuels sont configurés entre les routeurs qui ont en commun une aire de transit. Ces liens virtuels appartiennent à l'aire 0. Le protocole les traite comme des liens point-à-point non numérotés.

Ces définitions sont largement inspirées de Wikipédia.

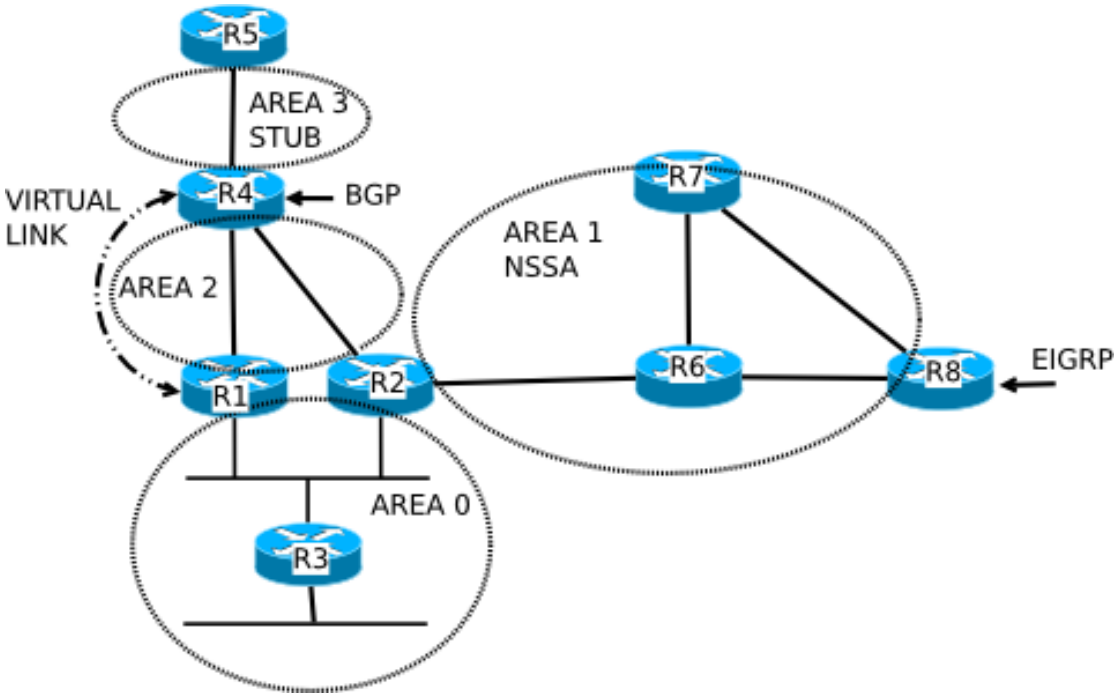

FIGURE 10 – Aires d'un réseau routé via OSPF – http://commons.wikimedia.org

3.2 BIRD

3.2.1 Présentation

Afin de mettre en place la topologie réseau utilisant les protocoles définis plus haut, nous avons utilisé un outil de routage (un démon exactement) nommé BIRD.

Il supporte ces fonctionnalités :

- Support d'IPv4 et d'IPv6
- Protocoles BGP, RIP, OSPF et routage statique
- Interface en ligne de commande (birdc)
- Configuration aisée par déclarations de protocoles
- Support sur Linux, FreeBSD, NetBSD, OpenBSD

3.2.2 Installation et configuration

```
aptitude install bird bird6
vim /etc/bird.conf

/*
 *      This is an example configuration file.
 */

# Yes, even shell-like comments work...

# Configure logging
#log syslog { debug, trace, info, remote, warning, error, auth, fatal, bug };
#log stderr all;
#log "tmp" all;

# Override router ID
#router id 198.51.100.1;

# You can define your own symbols...
#define xyzzy = (120+10);
#define '1a-a1' = (30+40);

### ETC... ###

service bird stop|start
```

4 Topologies mises en place

Afin de s'approprier la configuration du démon de routage BIRD, nous avons choisi de mettre en place plusieurs topologies réseau. Les schémas qui seront présentés ont tous été dessinés avec l'outil de création de diagrammes libre nommé Dia.

4.1 Topologie simple avec routes statiques

Dans cette topologie, nous avons déclaré 3 AS adjacents reliés par des routes statiques.

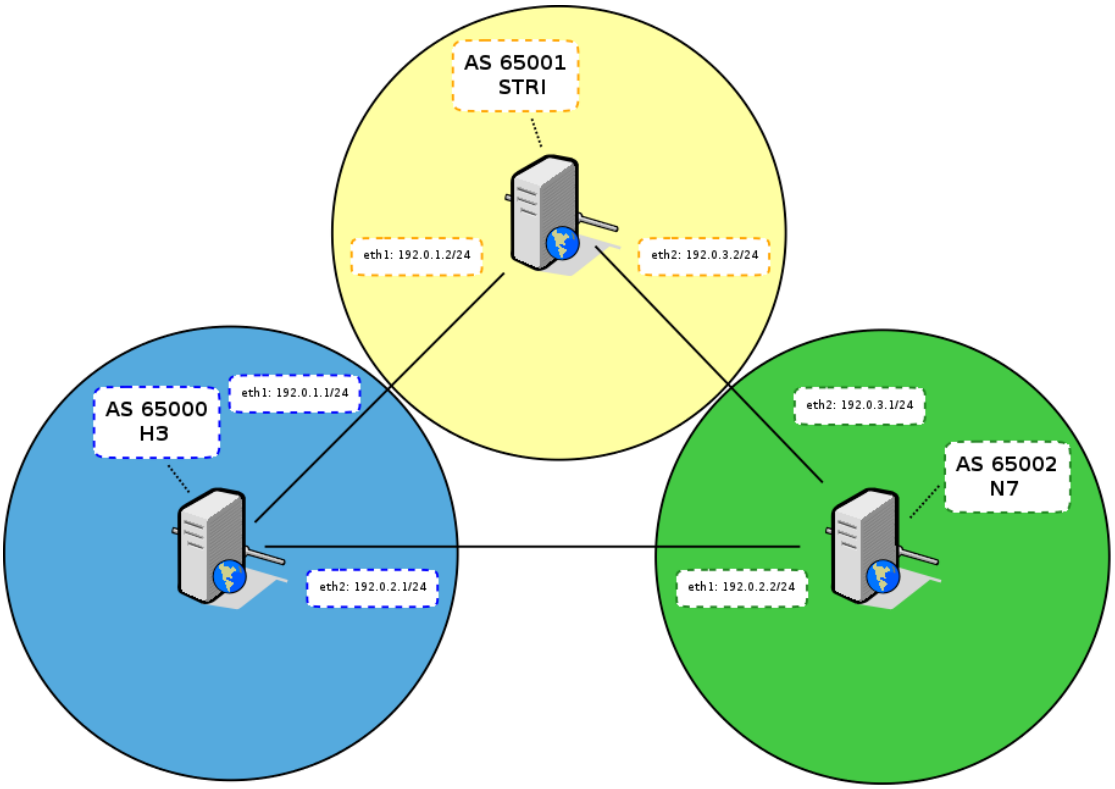

FIGURE 11 – 3 AS adjacents en statique

Pour ce faire voyons de plus près la configuration du routeur H3 :

```
log syslog all;

protocol kernel {
    persist;            # Don't remove routes on bird shutdown
    scan time 20;       # Scan kernel routing table every 20 seconds
    export all;         # Default is export none
}

# This pseudo-protocol watches all interface up/down events.
protocol device {
    scan time 10;       # Scan interfaces every 10 seconds
}

protocol static {
    import all;
    route 192.0.1.0/24 via 192.0.1.1;
    route 192.0.2.0/24 via 192.0.2.1;
}

protocol bgp h31 {
    description "BGP H3 65001";
    local as 65000;
    neighbor 192.0.1.2 as 65001;
    import all;
    export all;
}

protocol bgp h32 {
    description "BGP H3 65002";
    local as 65000;
    neighbor 192.0.2.2 as 65002;
    import all;
    export all;
}
```

Nous voyons ici la déclaration de 2 sessions BGP entre AS adjacents avec les IP et ASN associés. Les AS exportent leurs routes statiques déclarées dans le protocole statique grâce à l'option export all du protocole BGP. Ici nous aurions pu utiliser des filtres afin de choisir précisément quel type de routes sont diffusées ou acceptées depuis d'autres AS.

La partie kernel permet de récupérer les routes des interfaces directement connectées et accessibles dans l'espace noyau.

La partie device permet de rafraîchir l'état des interfaces et de définir des options sur celles-ci.

4.2 Topologie avec routage dynamique

Dans cette topologie, nous avons rajouté des sous-réseaux au sein des différents AS en assumant une politique de routage interne par le protocole OSPF. Ainsi nous avons pu tester la bonne diffusion des routes dans l'ensemble du réseau et les bons choix de routage exercés par les routeurs internes aux AS.

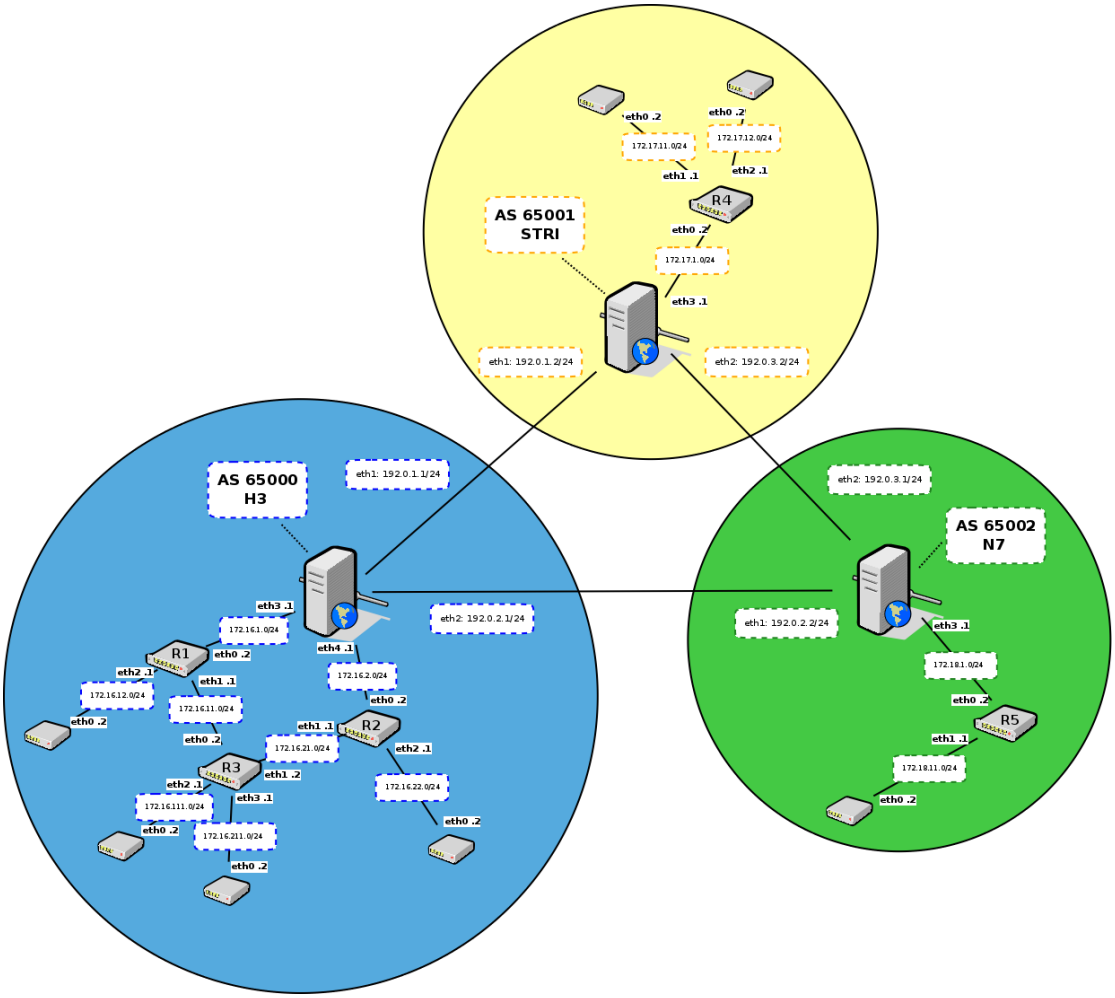

FIGURE 12 – 3 AS adjacents avec OSPF en IBGP

Pour ce faire voyons de plus près la configuration du routeur H3 :

```
log syslog all;

protocol kernel {
    scan time 20;       # Scan kernel routing table every 20 seconds
    export all;         # Default is export none
}

# This pseudo-protocol watches all interface up/down events.
protocol device {
    scan time 10;               # Scan interfaces every 10 seconds
}

protocol ospf H3 {
    import all;
    export all;
    area 0 {
        interface "eth3", "eth4" {
            cost 10;
            type pointopoint;
            hello 5; retransmit 2; wait 10; dead 20;
        };
    };
}

protocol bgp h31 {
    description "BGP H3 65001";
    local as 65000;
    neighbor 192.0.1.2 as 65001;
    import all;
    export all;
}

protocol bgp h32 {
    description "BGP H3 65002";
    local as 65000;
    neighbor 192.0.2.2 as 65002;
    import all;
    export all;
}
```

Nous déclarons les interfaces reliées aux routeurs R1 et R2 du réseau interne comme

étant des liens point à point et utilisant le protocole OSPF pour communiquer les routes découvertes.

Voyons les points intéressants des configurations des routeurs R1 et R3 :

```
# Configuration de R1

protocol ospf R1 {
    import all;
    export all;
    area 0 {
        interface "eth0", "eth1" {
            cost 10;
            type pointopoint;
            hello 5; retransmit 2; wait 10; dead 20;
        };
        interface "eth2" {
            cost 10;
            type broadcast;
            hello 5; retransmit 2; wait 10; dead 20;
        };
    };
}


# Configuration de R3

protocol ospf R3 {
    import all;
    export all;
    area 0 {
        interface "eth0", "eth1" {
            cost 10;
            type pointopoint;
            hello 5; retransmit 2; wait 10; dead 20;
        };
        interface "eth2", "eth3" {
            cost 10;
            type broadcast;
            hello 5; retransmit 2; wait 10; dead 20;
        };
    };
}
```

Noys voyons la différence de déclaration dans le type de liaison sur le protocole OSPF. Quand il s'agit d'une liaison vers un sous-réseau, on déclare le type en broadcast.

4.3 Topologie de Tetaneutral.net

Cette topologie est celle demandée par tetaneutral.net car proche de la réalité. Ils désirent notamment pouvoir basculer leur connexion sur une ligne ADSL entre deux routeurs dans le cas où la liaison par fibre ferait défaut. Le protocol OSPF rempli ce rôle à merveille comme nous allons le voir. Au niveau de la diffusion des routes, elles s'effectuent entre les AS 100 et 200 vers l'AS de tetaneutral.net.

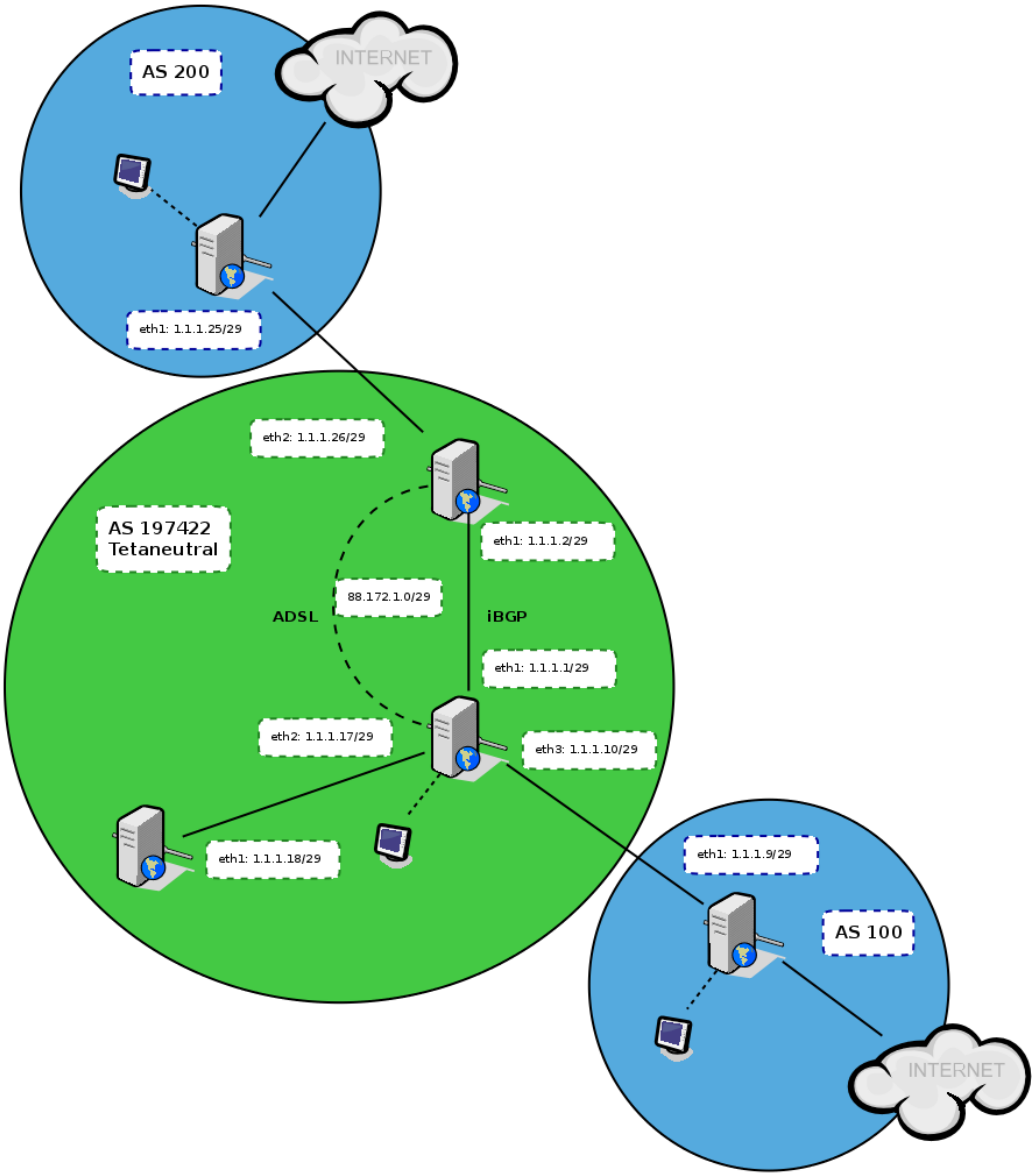

FIGURE 13 – Topologie de réseau virtuel pour tetaneutral.net

Pour illustrer la diffusion des routes depuis les AS externes vers l'AS de tetaneutral.net, voyons de plus près les tables de routage et la configuration :

```
# Jaguar possède une route vers l'Internet

protocol static {
    import all;
    route 10.0.0.0/8 via 10.2.5.8;
}

protocol bgp ebgp_jaguar {
    description "eBGP Jaguar";
    local as 100;
    neighbor 1.1.1.10 as 197422;
    import all;
    export all;
}


# H3 voit cette route

Table de routage IP du noyau
Destination     Passerelle      Genmask         Indic Metric Ref    Use Iface
0.0.0.0         192.168.0.240   0.0.0.0         UG    0      0        0 eth0
1.1.1.0         0.0.0.0         255.255.255.248 U     0      0        0 eth1
1.1.1.8         0.0.0.0         255.255.255.248 U     0      0        0 eth3
1.1.1.16        0.0.0.0         255.255.255.248 U     0      0        0 eth2
1.1.1.24        1.1.1.2         255.255.255.248 UG    0      0        0 eth1
10.0.0.0        1.1.1.9         255.0.0.0       UG    0      0        0 eth3
88.172.1.0      0.0.0.0         255.255.255.248 U     0      0        0 eth5
192.168.0.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0 eth4
```

Ensuite pour illustrer le bon fonctionnement du protocole OSPF en routage interne :

```
#H3 rejoint GW via la liaison fibre optique

Table de routage IP du noyau
Destination     Passerelle      Genmask         Indic Metric Ref    Use Iface
0.0.0.0         192.168.0.240   0.0.0.0         UG    0      0        0 eth0
1.1.1.0         0.0.0.0         255.255.255.248 U     0      0        0 eth1
1.1.1.8         0.0.0.0         255.255.255.248 U     0      0        0 eth3
1.1.1.16        0.0.0.0         255.255.255.248 U     0      0        0 eth2
1.1.1.24        1.1.1.2         255.255.255.248 UG    0      0        0 eth1
10.0.0.0        1.1.1.9         255.0.0.0       UG    0      0        0 eth3
88.172.1.0      0.0.0.0         255.255.255.248 U     0      0        0 eth5
192.168.0.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0 eth4

#Sur H3 on coupe la fibre optique
ifdown eth1

#Ensuite on regarde de nouveau la table de routage
Table de routage IP du noyau
Destination     Passerelle      Genmask         Indic Metric Ref    Use Iface
0.0.0.0         192.168.0.240   0.0.0.0         UG    0      0        0 eth0
1.1.1.0         0.0.0.0         255.255.255.248 U     0      0        0 eth1
1.1.1.8         0.0.0.0         255.255.255.248 U     0      0        0 eth3
1.1.1.16        0.0.0.0         255.255.255.248 U     0      0        0 eth2
1.1.1.24        88.172.1.2      255.255.255.248 UG    0      0        0 eth1
88.172.1.0      0.0.0.0         255.255.255.248 U     0      0        0 eth5
192.168.0.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0 eth4
```

Nous voyons s'effectuer le basculement sur la liaison ADSL qui possède un coût plus élevé de franchissement.

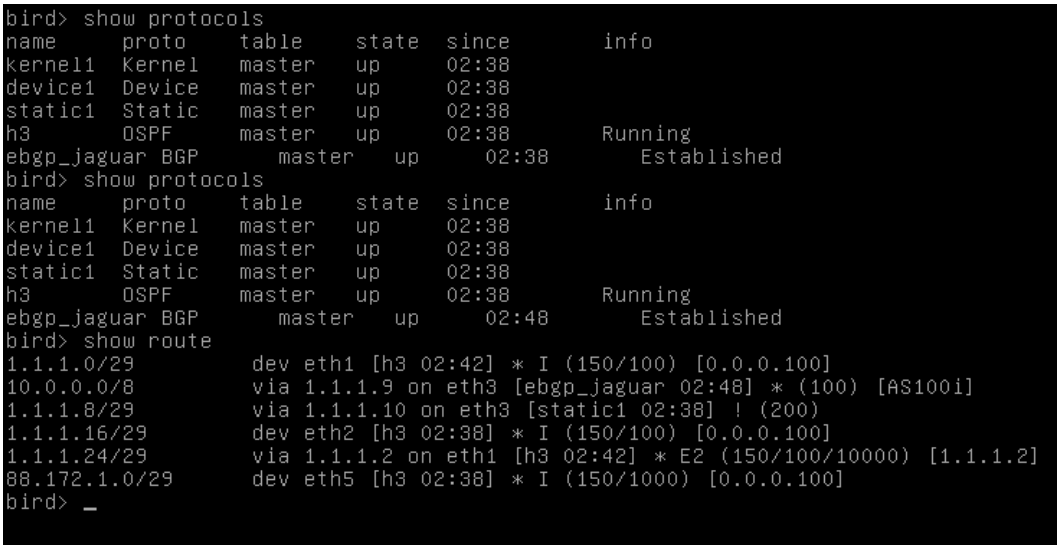

FIGURE 14 – Visualisation des routes sur birdc

Conclusion

Lors de ce TER nous avons réussi à accomplir plusieurs objectifs :

- Mettre en place un parc de machines virtuelles
- Lier les interfaces réseaux virtuelles avec Open vSwitch
- Acquérir un savoir faire sur BIRD
- Mettre en place une topologie fonctionnelle incluant BGP et OSPF

Ainsi nous avons acquis des connaissances supplémentaires dans le monde du réseau, de la virtualisation et des systèmes Linux. Pour ce qui en est de la solution pour tetaneutral, il faudra adapter quelques points sur les outils utilisés afin de mettre la maquette en place et écrire un script d'automatisation pour la gestion du parc de machines.

Il est toutefois possible de continuer sur ce sujet hors cadre scolaire, chose que j'ai déjà commencé à faire en rendant visite au responsable de l'association afin de mettre en place la maquette sur un ordinateur destiné à son hébergement au sein des locaux de tetaneutral.net. Nous pourrons continuer à aider sur ce projet en complétant le wiki sur le site ChiliProject et peut-être participer à d'autres sujets si l'envie se présente.

Nous pouvons conclure que cette expérience à été positive et enrichissante de part le sujet interessant et le cas concret qu'est une demande de l'association.

Références

[1] **tetaneutral.net** – Site de l'association
http://tetaneutral.net/

[2] **ChiliProject** – Outil Collaboratif de l'association
http://chiliproject.tetaneutral.net/

[3] **TER en ligne** – Sujet du TER sur ChiliProject
http://chiliproject.tetaneutral.net/projects/tetaneutral/wiki/StageKVMBIRD

[4] **KVM** – Site officiel de KVM
http://www.linux-kvm.org/page/Main_Page

[5] **QEMU** – Site officiel de QEMU
http://bird.network.cz/

[6] **Open vSwitch** – Site officiel d'Open vSwitch
http://openvswitch.org/

[7] **BIRD** – Site officiel de BIRD
http://bird.network.cz/

[8] **Article virtualisation** – Documentation sur KVM/libvirt
http://www.vogelweith.com/debian_server/14_kvm.php

[9] **Article OVS** – Documentation sur Open vSwitch
http://tech.covoiturage.fr/2012/08/22/open-vswitch-un\-switch-logiciel-pour-des-reseaux-virtuels/

[10] **libvirt** – Site officiel de libvirt
http://libvirt.org/

Annexes

Configuration de H3 dans la topologie du réseau virtuel tetaneutral.net :

```
log syslog all;

router id 0.0.0.100;

protocol kernel {
    scan time 20;       # Scan kernel routing table every 20 seconds
    export all;         # Default is export none
}

# This pseudo-protocol watches all interface up/down events.
protocol device {
    scan time 10;               # Scan interfaces every 10 seconds
}

protocol static {
    route 1.1.1.8/29 via 1.1.1.10;
}

protocol ospf h3 {
    import all;
    export all;
    area 0 {
        interface "eth1", "eth2" {
            cost 100;
            type pointopoint;
            hello 5; retransmit 2; wait 10; dead 20;
        };
        interface "eth5" {
            cost 1000;
            type pointopoint;
            hello 5; retransmit 2; wait 10; dead 20;
        };
    };
}
```

```
protocol bgp ebgp_jaguar {
    description "eBGP Jaguar";
    local as 197422;
    neighbor 1.1.1.9 as 100;
    import all;
    export all;
}
```

FIGURE 15 – Gestion des machines virtuelles

FIGURE 16 – Fichier de configuration XML d'une VM