Réseau pour machines virtuelles

GERARD Cyril SZYMCZAK Jerome SALECKI Simon
DUSART Clément HERBAUT Djezon

16 mars 2018

Université de Lille

Table des matières

Chapitre 1

Cadre du projet

1.1 Intitulé du sujet

Il s'agit de construire une architecture réseau IPv4 permettant à des machines virtuelles de communiquer, de manière transparente, entre elles, avec la machine physique qui les héberge ainsi qu'avec les machines du segment réseau sur lequel est intégrée la machine physique.

L'objectif est de pouvoir effectuer des TP d'administration système utilisant plusieurs machines (en mode *serveur* ou *poste de travail*) sur le même réseau sans trop de difficultés pour l'étape de construction des machines, mais en conservant un minimum de flexibilité (accès extérieur, manipulation des interfaces des machines de TP).

1.2 Contraintes

L'architecture doit être déployable sur une machine physique Debian via l'installation d'un paquet au format Debian.
Elle doit permettre d'accéder à tous les ports des machines virtuelles sans être obligé de mettre en place des redirections de ports.
Les seuls accès `root` qu'elle doit nécessiter sont, au moment de son installation et de l'installation, des outils de virtualisation.
Un utilisateur standard doit être capable de créer et démarrer des machines virtuelles sans droit d'administration.

Le réseau doit permettre *à minima* de connecter des machines virtuelles gérées par VirtualBox.
l'objectif étant de permettre au final de pouvoir y connecter des machines virtuelles gérées par VMware Workstation Player, QEMU/KVM ainsi que des containers Linux LXC.

À défaut d'une automatisation complète par la fourniture de commandes de création de machines virtuelles adéquates, des explications simples, permettant à un utilisateur de créer et connecter sa machine virtuelle à ce réseau, doivent être fournies sous la forme de documentation adaptée à chaque cas géré.

Chapitre 2

Organisation

2.1 Règles de gestion du projet

- L'intégralité du travail est gérée dans un dépôt GIT.
  - un dépôt particulier fait **référence** pour l'avancement du projet : celui accessible par le serveur GitLab de l'IUT ;
  - dans la mesure du possible, chaque participant du projet y pousse ses modifications *au moins* une fois par jour (en fin de journée) ;
- Un compte-rendu d'activité hebdomadaire est déposé, via un *commit* particulier, chaque fin de semaine sur le dépôt.
  - il détaille les tâches effectuées et leur responsable ;
  - il liste les éventuels points bloquants ;
  - il liste sommairement les tâches planifiées pour la semaine suivante ;
  - il précise la date et le lieu de la prochaine rencontre avec les tuteurs.
- Des conventions de nommage et de codage sont respectées.

2.2 Outils utilisés pour la gestion du projet

- GIT
- Pour la rédaction des documentations et présentations
  - Markdown
    - GitHub Flavored Markdown
    - Mastering Markdown
  - Mdoc
    - `https://manpages.bsd.lv/mdoc.html`
    - man(7), mdoc(7), groff(7)

Chapitre 3

Etat de l'art

3.1 Recherche sur les différents types d'hyperviseurs

3.1.1 Qu'est-ce que la virtualisation ?

Par définition, la virtualisation consiste en la création d'une version virtuelle (par opposition à réelle) d'un ou de plusieurs éléments, tel qu'un système d'exploitation, un serveur, un dispositif de stockage ou des ressources réseau sur un même système physique.

3.1.2 But d'une virtualisation

Le principal objectif est économique. Il y a encore quelques années, on séparait chaque service sur une machine physique distincte et l'utilisation moyenne du matériel était de l'ordre de 10%. Aujourd'hui, l'augmentation du nombre de coeurs au sein d'un même processeur et l'augmentation des capacités mémoires nous permettent de faire cohabiter au sein d'un même matériel plusieurs services.

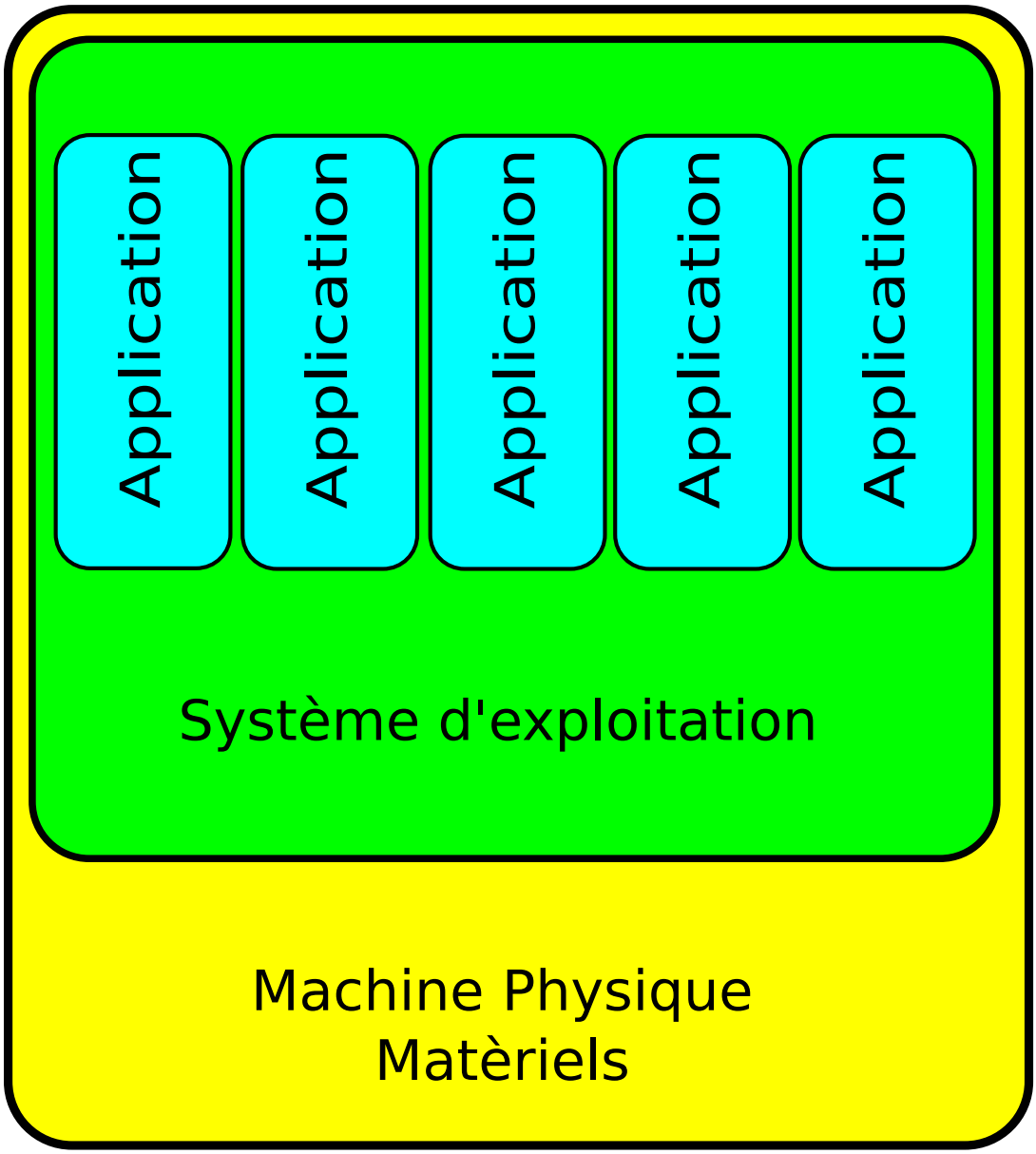
Un autre objectif est celui de la facilité d'administration. En effet, le processus d'installation d'un système est une opération lourde, gourmande en temps, et présente un risque de petites variations de configurations. Ainsi, la virtualisation permet de déplacer un serveur virtuel d'un hôte à un autre de manière très aisée, y compris sur des environnements matériels très hétérogènes, puisque les couches matérielles dans les serveurs virtuels sont le plus souvent génériques.

3.1.3 Il faut differencier :

- **L'hyperviseur :**
- Hyperviseur Type 1 : (Ex : VMware Vsphere, Oracle VM, Microsoft Hyper-V Server) : C'est un logiciel qui s'insère entre le matériel et les différents systèmes d'exploitation virtualisés assurant ainsi directement la communication avec ce dernier.
- Hyperviseur Type 2 : (Ex : VMware Workstation, Oracle VirtualBox) : C'est un logiciel qui s'insère entre le système d'exploitation hôte et les différents systèmes d'exploitation virtualisés. c'est le système d'exploitation hôte qui assure la communication avec le matériel .
- **L'isolateur** : (Ex : chroot, LXC, Docker) : C'est un logiciel permettant de créer un environnement utilisateur cloisonné au sein d'un système d'exploitation. Cet environnement peut alors exécuter des programmes sans que leur exécution ne perturbe le système d'exploitation de la machine en cas de dysfonctionnement. Ces environnements sont appelés des contextes ou bien des zones d'exécution.
- **L'émulateur** : (Ex : QEMU) : C'est un logiciel qui consiste à simuler l'exécution d'un programme en interprétant chacune des instructions destinées au micro-processeur. Il est possible d'émuler ainsi n'importe quel processeur et l'environnement complet.

3.1.4 Schéma de fonctionnement :

Système d'exploitation

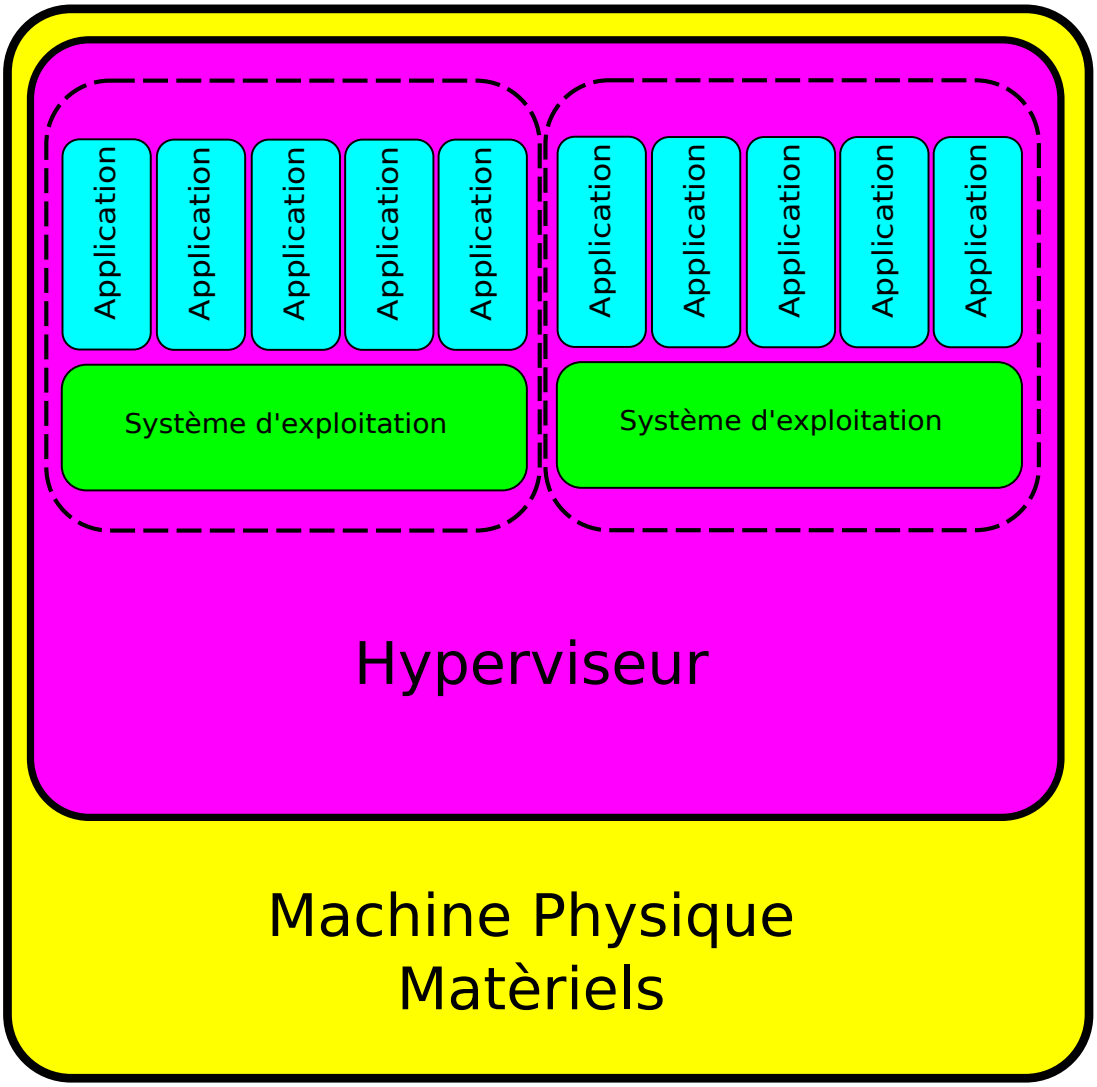


Machine physique
Traditionnelle

{

width=20% }

Hyperviseur Type 1

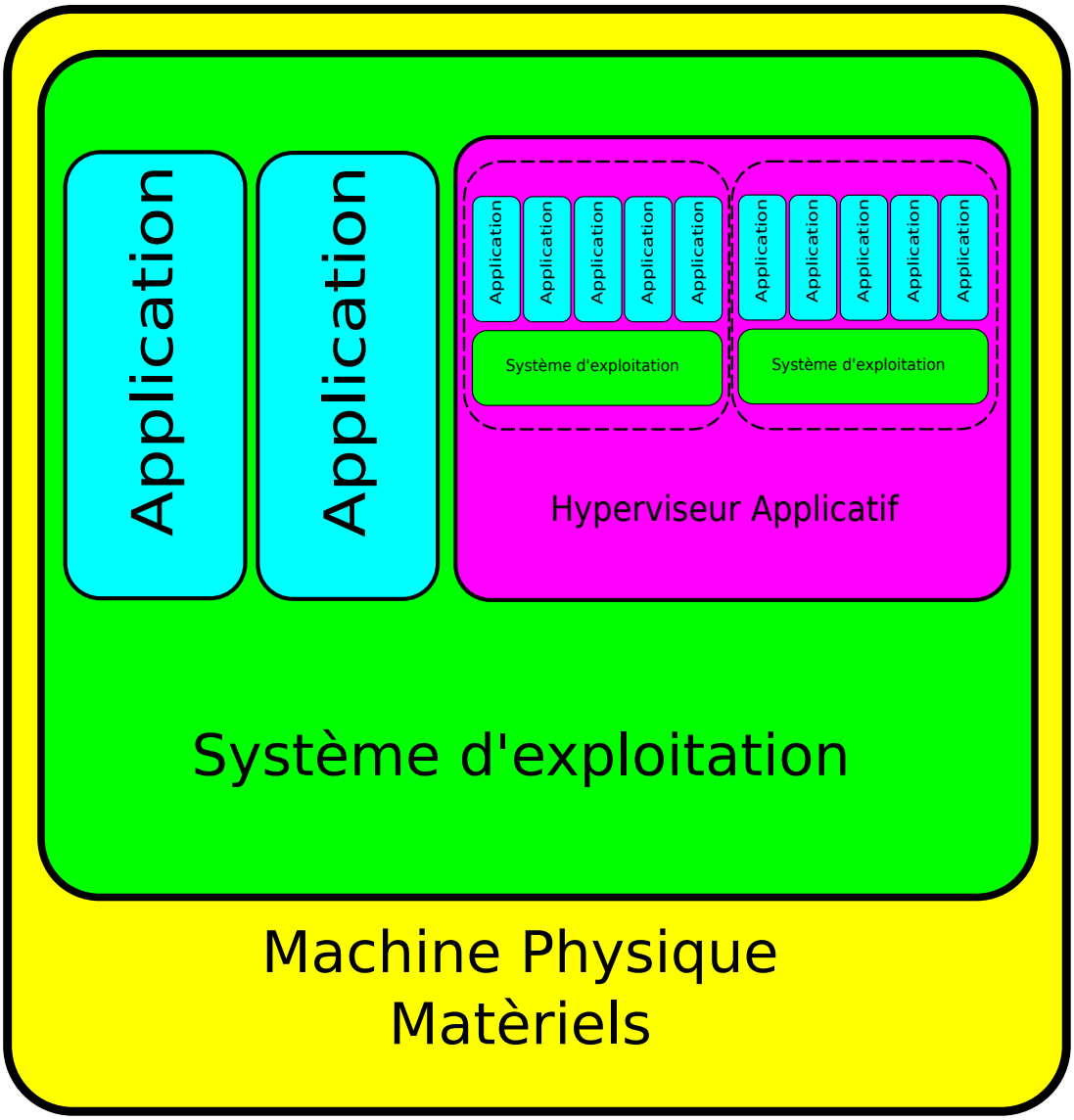


Machine physique
Hyperviseur Type 1

{

width=30% }

Hyperviseur Type 2

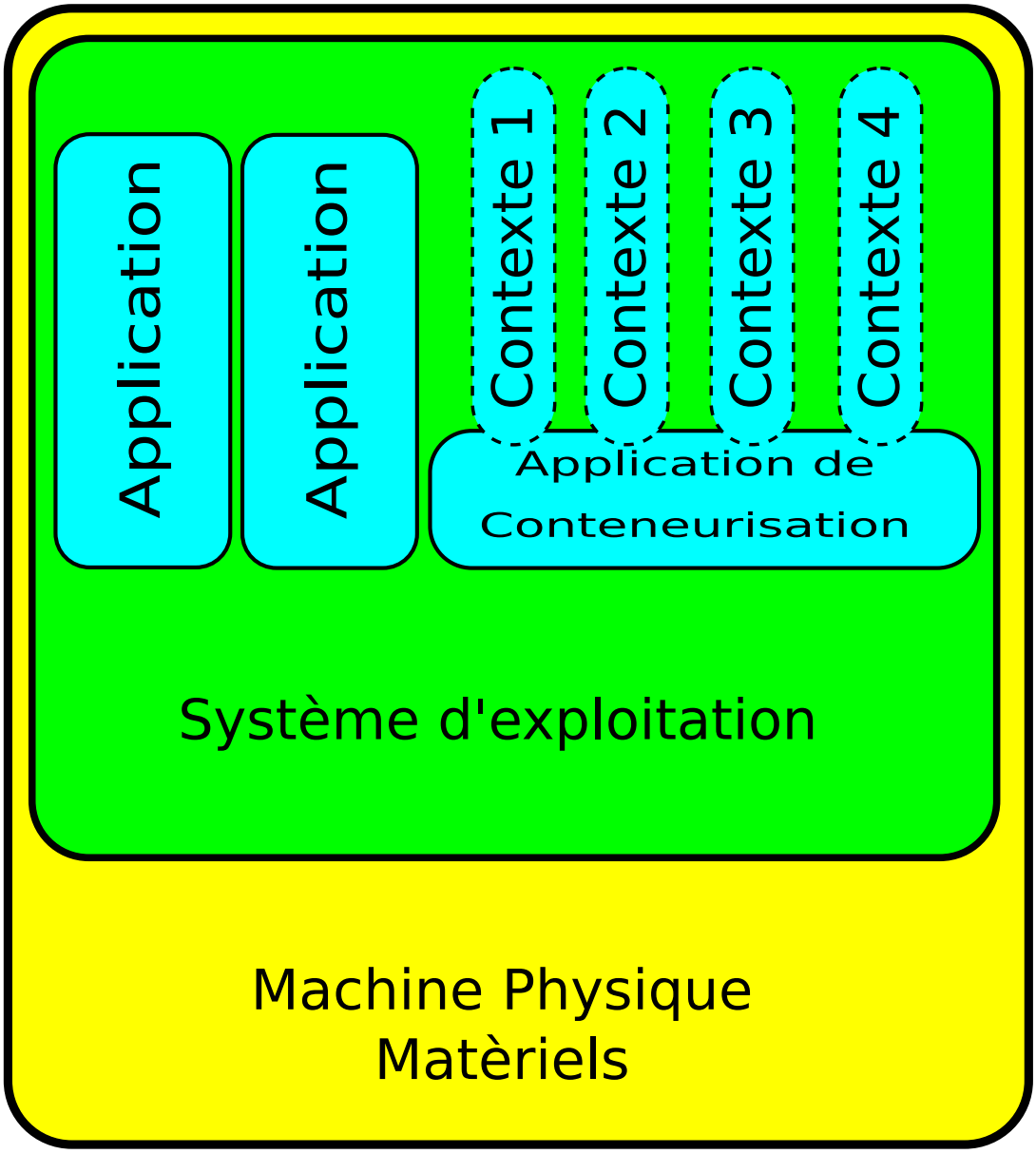


Machine physique
Hyperviseur Type 2

{

width=30% }

Isolateur



Machine physique
Isolateur

{
width=20% }

3.1.5 VirtualBox

Virtualbox est une application de virtualisation, c'est un hyperviseur "type 2".
Virtualbox n'exige pas une architecture processeur complexe, il n'a pas besoin des jeux d'instructions tels que Intel VT-x ou AMD-V, contrairement à beaucoup d'autres solutions de virtualisation.

Virtualbox fonctionne de manière identique sur toutes les plateformes hôtes, il utilise les mêmes formats de fichiers et d'images. Ceci permet d'exécuter des machines virtuelles créées sur un hôte possédant un système d'exploitation différent.
Vous pouvez ainsi créer une machine virtuelle sur Windows et l'utiliser sous Linux. De cette façon, vous pouvez lancer des logiciels écrits pour un système d'exploitation dans un autre. Virtualbox offre une grande souplesse d'usage, on peut geler, copier,

sauvegarder et créer des instantanés.

Il peut-être exécuté soit en mode graphique ou ligne de commandes « VboxManage ».
Il est possible d'installer les suppléments invités « pack d'extension » de Virtualbox afin d'accroître les performances et la communication avec la machine hôte (dossier partage). Virtualbox offre un bon support matériel cela inclut les contrôleurs de disques (IDE, SCSI, SATA, le support USB 2.0 3.0). Attention cette extension est sous licence (GPL2, CDDL et VPUEL pour Virtualbox Personal use and Evaluation License).

Virtualbox est libre d'utilisation pour sa partie principale mais les extensions, quant à elles, sont disponibles uniquement pour un usage privé. Il est possible d'organiser ces machines virtuelles en créant des groupes en sachant qu'une machine virtuelle peut appartenir à plusieurs groupes. Cela permet entre autres de commander toutes les machines (démarrer, arrêter, sauvegarder, ...). Le format d'enregistrement est le VDI , il peut avoir une forme fixe ou dynamique.

3.1.6 VMware Player

VMware Player est un hyperviseur de type 2, c'est l'outil gratuit mais propriétaire de VMware qui permet de découvrir la virtualisation.
Il permet la création d'une ou plusieurs machines virtuelles au sein d'un même système d'exploitation (Windows ou Linux), celles-ci pouvant être reliées au réseau local avec une adresse IP différente, tout en étant sur la même machine physique (machine existant réellement).
Il est possible de faire fonctionner plusieurs machines virtuelles en même temps, la limite correspondant aux performances de l'ordinateur hôte. Pour des fonctions plus poussées (snapshot par exemple), vous devrez passer à VMware Workstation qui lui est payant.

3.1.7 KVM/QEMU

KVM (Kernel-based Virtual Machine) est un hyperviseur libre de type I pour Linux qui permet la virtualisation simultanée de différents systèmes d'exploitation sur la même machine hôte. KVM est intégré dans le noyau Linux depuis la version 2.6.20 et est une instance de QEMU[1].

C'est un système optimisé pour la virtualisation de serveurs. Pour virtualiser des systèmes de type desktop, on peut lui préférer virtualbox. KVM semble en effet plus performant en consommation de processeurs mais plus lent pour l'émulation de périphérique graphique. L'utilisation d'un bureau virtualisé dans VirtualBox pourra donc laisser une meilleure impression à l'utilisateur. Vous pouvez tout de même préférer KVM pour sa meilleure compatibilité avec des systèmes d'exploitation anciens ou peu populaires.

3.1.8 Proxmox

Proxmox (Proxmox Virtual Environment) est un logiciel libre de virtualisation, plus précisément un hyperviseur de machines virtuelles.
Proxmox permet donc de monter facilement un serveur de virtualisation dont l'administration se fera via une interface web.
Proxmox VE installe les outils complets du système d'exploitation et de gestion en 3 à 5 minutes (dépend du matériel utilisé).
C'est une solution de virtualisation "bare metal"[3].

3.2 Recherche sur les solutions existantes

3.2.1 LXC

LXC (contraction de l'anglais de Linux Containers) est un système de virtualisation utilisant l'isolation au niveau système d'exploitation comme méthode de cloisonnement.
Son rôle est de créer un environnement aussi proche que possible d'une installation Linux standard, mais sans avoir besoin d'un noyau séparé. Les conteneurs LXC sont souvent considérés comme un compromis entre le mode "chroot[2]" et une machine virtuelle. LXC est donc un ensemble de processus qui nous permettent d'isoler des éléments du reste du système.
Il aura également accès à sa propre interface réseau, sa table de routage. Mais la différence notable, contrairement à Xen et KVM c'est l'absence d'un deuxième noyau. LXC va utiliser le même noyau que la machine hôte (Dom0). Les avantages de cette solution sont un gain de performances en l'absence d'hyperviseur et de noyau intermédiaire. L'autre avantage est la faible occupation de la ressource mémoire.

Les conteneurs LXC ne fournissent pas une isolation complète, ce qui est un inconvénient. C'est dû au fait que le noyau est partagé entre le Dom0 et les conteneurs. L'autre inconvénient est une mise en place plus complexe qu'une installation sur machine virtuelle.
Après la mise en place de quelques prérequis nécessaires au bon fonctionnement, il s'agira de mettre en fonctionnement notre

configuration réseau.
Ainsi, chaque conteneur aura une interface réseau virtuelle et la connexion au vrai réseau passera par un pont. Il existe deux manières de se connecter à l'interface virtuelle, soit branchée sur l'interface physique de la machine hôte (directement sur le réseau), soit branchée sur une autre interface virtuelle de l'hôte (pourra router le trafic). Les deux solutions passent par le biais du paquet bridge-utils dont dépend LXC. C'est la seconde solution que nous avons retenue.
Dans un premier temps, il faut configurer le fichier lxc-net afin qu'il crée le switch. Puis, dans un second temps, on crée l'interface virtuelle tap0 et enfin un tunnel entre tap0 ; le switch est créé par lxc-net. L'hôte fera donc office de passerelle pour que nos machines virtuelles puissent communiquer avec l'extérieur.

3.2.2 vmnet de VMware

Suite à l'installation de VMWare Player, deux cartes réseau virtuelles sont ajoutées à l'ordinateur hôte : VMnet1 et VMnet8.
Lors de la configuration d'une interface réseau, VMWare Player propose 3 types de connections :

- Bridged (pont) : La machine virtuelle est connectée au réseau physique via la carte physique de la machine hôte. L'adressage de la carte peut se faire manuellement ou via le DHCP fournissant le réseau physique.

- NAT (Network Address Translation) : La machine virtuelle est connectée à un réseau virtuel. La machine hôte est connectée à ce même réseau virtuel via la carte réseau VMnet8. Un routeur virtuel assure la communication etre le réseau virtuel et le réseau physique. Un DHCP virtuel permet l'attribution d'adresses aux machines virtuelles présentes sur ce réseau.

- Host-only : La machine virtuelle est connectée à un réseau virtuel. La machine hôte est connectée à ce même réseau virtuel par l'intermédiaire de la carte réseau VMnet1. Un DHCP virtuel permet l'attribution d'adresses aux machines virtuelles présentes sur ce réseau mais l'absence de routeur permet l'isolation de ce réseau par rapport au réseau physique.

3.2.3 OpenVSwitch

Open VSwitch est un commutateur logiciel multicouches sous licence open source Apache 2.
Il est bien adapté pour fonctionner comme un commutateur virtuel, qui fournit des fonctionnalités avancées de commutation, mais également de QoS, d'agrégation de liens, de Vlan, de collecte de données.
Il a été conçu pour prendre en charge la distribution sur plusieurs serveurs physiques. Open vswitch prend en charge plusieurs technologies de virtualisation basées sur Linux, notamment Xen / Xen Server, KVM et Virtualbox. La majeure partie du code est écrite en C, indépendant de la plate-forme, et peut-être facilement transférée vers d'autres environnements.
OpenvSwitch est un commutateur virtuel compatible avec les chipsets des switchs modernes commutateur administrable avec le protocole Open flow.
Il supporte le VLAN 802.1 Q, isolation et filtre de traffics, d'agrégation de lien, lac, Channel boding, gestion des flux et QoS Bande passante.
Il est conçu pour prendre en charge la distribution sur plusieurs serveurs physiques similaires au vswitch distribué de Vmware ou au Nexus 1000V de Cisco.

```
VMware a officialisé l'abandon prochain de son API VDS, qui permettait l'intégration de commutateurs virtu
source :"http://www.lemagit.fr"
```

3.2.4 TUN/TAP

TUN/TAP est une fonction de réception et de transmission de paquets entre le noyau et les programmes de l'espace utilisateur. Cette fonction peut être vue comme une simple interface point à point ou Ethernet qui, au lieu de recevoir les paquets d'un média physique, les reçoit du programme de l'espace utilisateur. De même, cette interface au lieu d'envoyer les paquets vers un média physique, les transmet au programme de l'espace utilisateur.
Dans notre contexte, le programme de l'espace mémoire utilisateur est l'instance virtuelle de système d'exploitation. L'interface réseau TUN/TAP devient un canal de communication réseau entre le système hôte et un système virtualisé.

3.2.5 Notes :

1. *QEMU est un logiciel libre de machine virtuelle, pouvant émuler un processeur et, plus généralement, une architecture différente si besoin. Il permet d'exécuter un ou plusieurs systèmes d'exploitation via les hyperviseurs KVM et Xen, ou seulement des binaires, dans l'environnement d'un système d'exploitation déjà installé sur la machine.*
2. *Cette commande permet d'isoler l'exécution d'un programme.*
3. *Bare metal (metal nu) signifie que vous commencez à partir d'un serveur vide et qu'il n'y a donc nul besoin d'installer un système d'exploitation auparavant.*

Chapitre 4

Mise en œuvre du projet

4.1 Solution retenue

Pour choisir notre solution, nous avions deux solutions réalisables en respectant les contraintes fixées dans le cahier des charges, soit passer par le biais d'un script existant(lxc-net), soit créer notre propre script permettant la création d'un switch virtuel.Les deux solutions retenues passent par le biais du paquet bridge-utils.

C'est la première solution que nous avons retenue, car elle comprenait plusieurs avantages tels que ne pas dépendre de l'architecture en place sur la machine, un léger coût en consommation de mémoire et une utilisation assez simple. Pour permettre une connexion par pont il nous a fallu créer une interface TAP.

4.2 Création du switch virtuel

La création du switch virtuel se fait grâce au script lxc-net compris dans LXC.
Nous devons tout d'abord copier l'entête du script et la mettre dans le fichier /etc/default/lxc-net
Puis changer les variables utiles(nom du switch, l'ip de l'interface, la plage d'IP servies par le DHCP,...).
Et nous pouvons ainsi lancer le script :

```
systemctl restart lxc-net
```

4.3 Création et configuration de l'interface tap

Nous allons créer une interface tap en ligne de commande (dans notre paquet, elle se fera grâce à un script)
Création d'une interface en mode TAP.

```
ip tuntap add mode tap tap0
```

Connecter cette interface au switch créé par lxc-net

```
ip link set dev tap0 master lxcbr0
```

Activer l'interface TAP créée

```
ip link set tap0 up
```

4.4 Implémentation d'un paquet Debian

Le paquet nécessaire pour faire ses propre paquets est dpkg. Le programme dpkg-deb qui est contenu dans le paquet dpkg est le programme qui construit un fichier .deb.

4.4.1 L'arborescence d'un paquet Debian

Afin de permettre à dpkg de faire un paquet, nous devons respecter une aborescence particulière. En effet, pour la création d'un paquet, l'arborescence à créer est simple (selon les paquets). Voici l'arborescence à créer.

- myscript/
  - DEBIAN/
    - control* (fichier décrivant les informations relatives à notre paquet)
    - preinst (script exécuté après l'installation du paquet)
    - postinst (script exécuté après l'installation du paquet)
    - prerm (script exécuté après la désinstallation du paquet)
    - postrm (script exécuté après la désinstallation du paquet)
    - md5sums (permet la vérification de l'intégralité des données récupérées)
  - usr/
    - bin/
      - myscript (notre script, exposé ci-dessus)
    - share/doc/
      - README (informations relatives à l'utilisation de myscript)
      - copyright
      - changelog (changements apportés par rapport à la dernière version)
      - changelog.Debian (idem, mais seulement pour le paquet Debian)

* *C'est un fichier principal de contrôle qui contient un certain nombre de champs. chaque champ commence par une étiquette suivie de ':' et du contenu du champ.*

4.5 Description sur le fonctionnement du paquet

4.5.1 Installation et désinstallation de notre paquet

Installation du paquet

```
# dpkg -i tarr-steps.deb
# apt install -f
```

Lors de l'installation de notre script, postinst va s'éxécuter en créant un fichier de configuration de lxc-net, il va aussi démarrer une interface tap.

Désinstallation du paquet

```
# apt remove tarr-steps
```

Lors de la désinstallation, notre script prerm va arrêter l'interface tap0 pour permettre ensuite de désinstaller notre paquet et ses dépendances.
Attention : Lors de la désinstallation de notre paquet, les dépendances ne seront pas désintallées, il faudra le faire à la main

4.5.2 Fonctionnement de notre script

Nous étions tout d'abord partis sur un système de menu pour permettre à tout le monde d'utiliser avec facilité notre script.
Mais en discutant avec les principaux intéressés (admin de l'IUT), nous avons changé le script pour répondre ainsi au mieux à leurs attentes.
Voici les options qui peuvent être utilisées sur notre script.

```
-h,  --help          affiche ce message d'aide
-ip, --ip            change l'adresse ip du switch
```

```
-l,  --liste          liste les informations liées au switch
-c,  --check          vérifie que l'interface tap a bien été créée
-st, --start          start sur le script lxc-net
-sp, --stop           stop sur le script lxc-net
-r,  --reload         reload sur le script lxc-net
```

4.6 Difficultés rencontrées

L'un des premiers obstacles fut la maîtrise d'un dépôt GIT, mais à force d'utilisation et de conflits, nous avons réussi à faire ce qu'il fallait et ainsi apprendre à l'utiliser correctement. Nous avons aussi fait une autre découverte, celle des fichiers Markdown, ça n'a pas été simple au début pour chacun d'entre nous. Ce que nous avons remarqué sur celui-ci c'est le manque visuel, nous avons toujours été habitué à faire de la mise en page.

Lors de la mise en place de notre projet, nous avions des difficultés à débattre d'une solution viable et simple d'utilisation. Les différentes solutions existantes étaient complexes et plus contraignantes.

Mais les plus gros problèmes rencontrés lors de notre projet furent liés au réseau. Sur les ordinateurs des salles TP, nous n'étions pas root et donc aucune possibilité de modifier les interfaces, pour régler ce soucie, nous avions eu accès à un ordinateur. Pour utiliser cette machine, nous n'avions aucun accès physique, ce qui nécessite l'intervention des administrateurs systèmes lorsque nous perdions la connexion à celle-ci.

Chapitre 5

Procédure de test

5.1 Installation de notre paquet

- 1.Installation du paquet
- Le paquet a bien été installé mais il manque les dépendances : OK
  `# dpkg -i tarr-steps`
- Les dépendances se sont bien installé : OK
  `# apt-get install -f`
- Vérification que le préinst s'éxécute
  `$ cat /etc/default/lxc-net`
  `$ ip a`
- 2.L'éxécution de notre script
- affichage de l'aide à l'éxécution du script : OK
  `$ tarr-steps -h`
- Configuration de l'dresse du switch et affichage de ces changements : OK
  `$ tarr-steps -ip 192.168.194.1/24`
  `$ tarr-steps -l`
- Relance du script lxc-net pour mettre à jour le switch virtuel : OK
  `$ tarr-steps -r`
  `$ ip a`

5.2 Test de communication entre machines virtuelles

Tout les tests qui suivent ont été réalisés avec VirtualBox, en testant auparavant avec KVM nous obtenions les mêmes résultats

Après installation du paquet "tarr-steps" :

- Verification des paramétres dans le fichier LXC-NET :
  `$ cat /etc/default/lxc-net`
  ou
  `$ tarr-steps -l`
- Verification de la configuration du réseau :
  `$ ip a`
  `2: enp0s25:  172.18.50.4/22`
  `3: vswitch0:  192.168.194.1/24`

Nous constatons la corélation entre le fichier lxc-net et l'attribution des adresse ip des interfaces réseaux

Lancement de VirtualBox. Création de 2 machines virtuelles "TestVM1" et "TestVM2".

Sur chaque machine virtuelle, dans Configuration -> Réseau -> Carte1 :
Mode d'accés réseau : Accès par pont
Nom : iut0

Démarrage des machines virtuelles.

Sur TestVM1 :

```
# ip a
# cat /etc/resolv.conf
# ip route
```

Attribution de l'adresse ip 192.168.194.243 avec comme resolveur DNS 192.168.194.1 et une route par defaut 192.168.194.1 et la route pour le réseau 192.168.19.0/24 accessible via 192.168.194.243

Sur TestVM2 :

```
# ip a
# cat /etc/resolv.conf
# ip route
```

Attribution de l'adresse ip 192.168.194.206 avec comme resolveur DNS 192.168.194.1 et une route par defaut 192.168.194.1 et la route pour le réseau 192.168.19.0/24 accessible via 192.168.194.206

A partir de la machine TestVM1

```
$ping 192.168.194.206
$ping 192.168.194.1
$ping 172.18.50.4
ssh root@192.168.194.206
```

A partir de la machine TestVM2

```
$ping 192.168.194.243
$ping 192.168.194.1
$ping 172.18.50.4
ssh root@192.168.194.243
```

Tous les pings repondent et la connexions ssh entres machines est possibles.

5.3 Désintallation de notre paquet

- Le paquet a bien été désintaller : OK
  ```
  # apt-get remove tarr-steps
  # dpkg -l tarr-steps
  ```
- Vérification que le postrm s'éxécute : OK
  ```
  # cat /etc/default/lxc-net
  $ ip a
  ```

Remarque :
Lors de la désintallation, les dépendances ne sont pas désintaller.
On pourrait les désinaller avec un script 'posrrm' qui ferai un apt-get autoremove.

Chapitre 6

Conclusion

Au terme de ce deuxième projet, nous avons été confrontés à une problématique plus compliquée.
Nous avons dû appréhender la complexité des réseaux virtuels et lire une documentation technique pas toujours évidente.
Nous avons été confrontés à des contraintes techniques lié à l'environnement universitaire (proxy, port réseau « tagged », liaison ssh sur machine distante).
En effet la principale problématique a été les pertes de connexion réseau sans possibilités de redémarrer de manière autonome.
Néanmoins nous avons pu démontrer notre capacité d'adaptation et faire face aux difficultés rencontrées.

Nous avons su mettre en adéquation nos différentes expériences et de nouvelles connaissances.
Avec l'usage de GIT et de MarkDown(langage de balisage), nous partagions la même convention de nommage afin de mieux communiquer entre nous.

L'expérience a été enrichissante autant sur le plan humain que technique. Ce projet nous a permis de mettre en pratique et développer différents aspects vus en cours. Malgré certaines erreurs que nous avons pu commettre, nous avons su apprendre de celles-ci et
ainsi mieux les appréhender.

Ce projet nous a confortés dans notre choix de carrière. Il nous a permis de développer des qualités telles que la réflexion et d'autonomie afin de nous intégrer au mieux dans le monde du travail.

\appendix

Chapitre 7

Annexe

7.1 Documents techniques

7.1.1 Le script de notre paquet Debian

```
#!/bin/bash
# Permet de configurer de l'interface créer par lxc-net.
#
# usage: tarr-steps [OPTIONS] [VALEUR]
#
# OPTIONS
#
#     -h,  --help          affiche ce message d'aide
#     -ip, --ip            change l'adresse ip du switch
#     -l,  --liste         liste les informations liées au switch
#     -c,  --check         vérifie que l'interface tap a bien été créer
#     -st, --start         start sur le script lxc-net
#     -sp, --stop          stop sur le script lxc-net
#     -r,  --reload        reload sur le script lxc-net
#
# EXEMPLES
#
# Changer l'adresse du switch virtuel
#
#     tarr-steps -ip 192.168.194.1/24
#
# Vérifier l'état du fichier de configuration
#
#     tarr-steps -l
#
# Vérifier que l'interface tap0 est prête à l'utilisation
#
#     tarr-steps -c
#

IP="LXC_ADDR="
newIP=$(ipcalc $2 | grep "Address" | cut -d\  -f 4)

NETMASK="LXC_NETMASK="
newMask=$(ipcalc $2 | grep "Netmask" | cut -d\  -f 4)

NETWORK="LXC_NETWORK="
newNetwork=$(ipcalc $2 | grep "Network" | cut -d\  -f 4 | cut -d/ -f1)

PLAGE="LXC_DHCP_RANGE="
plageIPmin=$(ipcalc $2 | grep "HostMi" | cut -d\  -f 4)
plageIPmax=$(ipcalc $2 | grep "HostMa" | cut -d\  -f 4)
```

```
NBIP="LXC_DHCP_MAX="
newNbIP=$(ipcalc $2 | grep "Hosts" | cut -d\  -f 2)

veriftap=$(ip a | grep iut0 | egrep -c "vswitch0|master")

doc(){ sed -n '2,/^$/ { s/^ *#// ; s/^ //g ; t ok ; d ; :ok ; p }' <$0 ; }
change (){ sed -i "s@$1\".*\"@$1\"$2\"@g" "/etc/default/lxc-net" ; }
check(){
if(test $veriftap -eq 1)
then
    echo "Votre interface iut0 a bien été créée"
else
    echo "Votre interface fait des siennes"
fi
}
changertout(){
    change $IP $newIP
    change $NETMASK $newMask
    change $NETWORK $newNetwork
    change $PLAGE "$plageIPmin,$plageIPmax"
    change $NBIP $newNbIP
}
while [ ! -z "$1" ] ;
do
    case "$1" in
        "-h"|"--help") doc && exit ;;
        "-ip"|"--ip") changertout && exit ;;
        "-l"|"--list") cat /etc/default/lxc-net && exit ;;
        "-c"|"--check") check && exit ;;
        "-st"|"--start") systemctl start lxc-net && exit ;;
        "-sp"|"--stop") systemctl stop lxc-net && exit ;;
        "-r"|"--reload") systemctl stop lxc-net && systemctl start lxc-net && exit ;;
        *) break;;
    esac
    shift
done
```

7.1.2 Comparaison sur les différentes solutions existantes

LXC

Description :
LXC est conteneur Linux (ensemble de processus qui sont isolés du reste du système).
Dans LXC nous avons lxc.network qui est très utile pour pour avoir accès à internet.
Les conteneurs doivent se connecter à une interface bridge sur l'hôte. Celle-ci peut avoir été créée par le paquetage (lxc-net), on pourra donc la créer manuellement.

Dépendances : liblxc1, python3-lxc, libapparmor1, libc6, libcap2, libgnutls30, libseccomp2, libselinux1, python3 :any, lsb-base

Complexité : 3

Avantages :

- N'est pas dépendant de l'architecture
- Est léger en consommation mémoire
- Simple à utiliser

Inconvénients :

- Assez dur à comprendre comment le script fonctionne

Comment ça fonctionne :

1. copier le début du code de /usr/lib/x86_64-linux-gnu/lxc/lxc-net dans /etc/default/lxc-net

2. Relancer le service lxc-net

3. Créer une interface tap0

4. Créer un tunnel entre tap0 et le switch créé par lxc-net

5. Mettre la machine virtuelle en accès par pont sur l'interface 'tap0'

Fonction TUN/TAP

Description :
Un dispositif TUN/TAP peut être vu comme une interface réseau qui communique avec un programme utilisateur (dispositif logiciel) au lieu d'une vraie carte matérielle (TUN pour mimer un périphérique point à point, TAP pour mimer un périphérique Ethernet).

Dépendances : bridge-utils, uml-utilities

Complexité : 2

Avantage :

– Très simple à mettre en place

Inconvénient :

– Modification des interfaces existantes

Comment ça fonctionne :

1. Installation des paquets

2. création d'une interface bridge

3. Création d'un tunnel entre le bridge et tap0

4. Mettre la machine en accès par pont sur l'interface 'tap0'

vmnet de VMware

Description :

– VMnet0 pour relier les VMs au réseau physique direct (Bridged mode)

– VMnet1 isole totalement les cartes qui lui sont reliées du reste du monde, mais pas entre elles (Host Only mode)

– VMnet8 relie les VMs au réseau physique en passant par de la translation d'adresses (NAT mode)

Dépendances : aucune

Complexité : 0

Avantage :

– Facile à installer

Inconvénient :

– On doit installer un hyperviseur complet

Comment ça fonctionne :

– Installation de VMplayer

– Mettre la machine virtuelle en accès par pont sur l'interface 'vmnet8'

7.2 Sources

– Recherche virtualisateurs

  - http://www.lemagit.fr/definition/Virtualisation

  - http://infrastructure.smile.eu/Tout-savoir-sur/Virtualisation-et-cloud/Les-principes-de-la-virtualisat
- VirtualBox
  - https://www.virtualbox.org/download/testcase/manual/UserManual_fr_FR.pdf
- VMware
  - http://www.electro-info.ovh/index.php?id=60

  - https://fr.wikipedia.org/wiki/VMware

  - http://www.tuto-it.fr/PresentationProduitVMware.php
- QEMU/KVM
  - https://virt.kernelnewbies.org/KVM_Multiuser_Usage?highlight=(kvm)|(usb)

  - https://doc.ubuntu-fr.org/kvm

  - http://debian-facile.org/doc:systeme:kvm
- Proxmox
  - https://www.proxmox.com/en/

  - https://fr.wikipedia.org/wiki/Proxmox_VE

  - http://linuxfr.org/news/proxmox-la-virtualisation-facile
- LXC
  - http://www.linuxembedded.fr/2013/07/configuration-reseau-de-lxc/

  - https://wiki.debian.org/fr/LXC

  - https://wiki.debian.org/fr/LXC/SimpleBridge
- TUN/TAP
  - http://debian-facile.org/doc:reseau:interfaces:tapbridge

  - https://www.inetdoc.net/guides/vm/vm.network.tun-tap.html
- vmnet
  - http://g.urroz.online.fr/doc/ch03s02.html