Cahier des charges

KostiTeam

24 novembre 2016

Table des matières

1 Problématique

La virtualisation est le fait de créer une version virtuelle d'une entité physique, ces versions virtuelles sont alors appelées Machines virtuelles ou VM (Virtual Machine). Nous pouvons prendre comme exemple Virtual Box, qui est un outil de virtualisation permettant d'émuler un système d'exploitation (linux ou autre). Les différentes ressources de la machine hôte sont alors partagées et allouées dynamiquement aux différentes machines virtuelles par des logiciels appelés hyperviseur.

La virtualisation peut être utile dans diverses situations. En effet, virtualiser permet de tester un environnement afin de trouver ses meilleurs caractéristiques avant de le créer. De plus, la virtualisation peut être utilisée à des fins pédagogiques. Dans le cadre d'études informatiques, il peut être pratique de simuler des réseaux dans lesquels des machines dialoguent. Marionnet est un exemple de logiciel de simulation de réseaux.

1.1 Marionnet

Marionnet est un simulateur de réseau. Voici un exemple d'utilisation : On souhaite simuler un réseau constitué de 2 sous-réseaux. Chaque sous-réseau contient 3 machines. On relie donc chaque groupe de 3 machines avec un hub, puis les deux sous-réseaux avec une passerelle (gateway).

On peut ensuite configurer les tables de routage, les adresses ipv4, ipv6 etc. pour simuler des échanges entre les machines.

Cependant, après avoir utiliser un tant soit peu Marionnet, il s'avère que plusieurs défauts gênent son utilisation :

— crashs assez fréquents ;
— impossible de changer les configurations des machines en marche ;
— processus d'arrêt des machines trop fastidieux ;
— interface peu ergonomique ;
— logiciel trop gourmand en ressources.

Face à ces problèmes, nous avons donc décidé de créer un nouveau simulateur de réseaux.

2 Solution

Pour pallier les problèmes exposés précedemment, il nous semble pertinent de ne pas virtualiser les machines utilisées (VM) mais uniquement les environnements (VE).

Les VM ont un défaut pour nous : elles simulent tout le materiel d'une machine (processeur, RAM...). Nous prefererons donc utiliser des VE (Virtual Environement). Les VE permettent de ne simuler que le systeme d'exploitation, et de partager le même noyeau que la machine hôte, et donc, de répartir les ressources entre machine hôte et les differents environnements virtuels. Cette solution répond donc à nos besoins : un gain notable de performance, notamment lorsque le nombre de machines est important.

Pour virtualiser les environnements, nous avons choisi d'utiliser la technologie de LXC.

3 LXC

3.1 Définition

LXC est un outil de virtualisation permettant de créer des environnements virtuels, différents systemes d'exploitations sont mis a disposition (Ubuntu, Debian...). Ces Environnements sont appelées containers.

Le partage des ressources est assuré par l'outil Cgroups du noyau, qui permet de limiter, compter et isoler l'utilisation des ressources.

Chaque environnement virtuel est isolée, de la même manière que l'isolement d'un programme avec "*chroot*" : chaque environnement est créé de manière à ce qu'ils naient pas accès au système d'exploitation de la machine hôte. En revanche, la machine hôte, elle, a accès aux machines virtuelles. Cette isolation entre les machines virtuelles et la machine hôte, permet de garantir une certaine sécurité.

3.2 Pourquoi LXC ?

LXC possède sa propre API, notamment en C/C++, ce qui nous permet de l'intégrer) notre projet efficacement. Il est plus léger que des technologies comme Docker (qui hérite de LXC) et convient à notre utilisation.

3.3 Fonctionnement de LXC

3.3.1 Les containers, leur fonctionnement

Les environnements virtuels, ou, containers (conteneurs en francais) doivent, en premier temps, être créés à l'aide de la commande "*lxc-create ...*" (voir notices pour plus de précisions sur les commandes). De nombreux systèmes d'exploitation seront alors proposés, (nous avons choisi de prendre Debian, Jessie, i386). Par défaut, les machines créées ne sont pas configurées : elles n'ont pas d'interfaces, elles n'ont pas de compilateur, et les utilitaires préinstallés sont très rudimentaire (pas de ping, ifconfig, tcpdump...).

Chaque container possède un fichier de configuration situé à l'emplacement suivant : "*/var/lib/lxc/<nom du container>/config*". Il est possible de configurer de nombreux aspect du container dans ce fichier (par exemple : modifier les variables d'environnement, ou changer le nom d'hôte ("*hostname*") du container. Voir "*man lxc*" pour en savoir plus). Ce fichier va notamment permettre de paramétrer la configuration réseau du container à son démarrage (c'est ce qui va nous intéresser en priorité avec ce fichier).

Les containers se lancent avec la commande "*lxc-start ...*" ; il est préférable de les lancer en démons (en arrière-plan), puis d'y "*attacher*" un terminal avec "*lxc-attach*", afin d'être connecté en super utilisateur (ou root) car, premièrement, par défaut, aucun profil d'utilisateur n'est créé sur le container, et, de plus, cela permet d'avoir l'entier contrôle du container afin de, par exemple, modifier son adresse ipv4.

3.3.2 Bridges et fonctionnement réseau

3.3.2.1 fichier de configuration

Comme expliqué plus haut, les paramètres réseaux du container peuvent être modifiés via le fichier config. Dans ce fichier, chaque "*block*" permet de définir une interface. Un block commence toujours par la définition du type de réseau, nous allons donc nous intéresser tout d'abord au type de réseau ("*lxc.network.type*"). Plusieurs types de réseaux sont disponibles, mais, nous allons choisir le type veth, qui signifie Virtual Ethernet, ce type permet d'établir un lien entre l'interface virtuelle du container et un pont, préalablement créé sur la machine host (nous verrons cette liaison plus en détail par la suite).

Ce fichier permet aussi d'établir des adresses ipv4 ("*lxc.network.ipv4*"), ipv6 ("*lxc.network.ipv6*"), et mac ("*lxc.network.hwaddr*"), ainsi que leurs Broadcast. Cela permet de mettre en place le réseau virtuel avant de lancer les machines ; bien que ces paramètres puissent être modifiés une fois la machine lancée

à l'aide de l'outil "*ifconfig*". Attention, les tables de routage ne peuvent pas être configurées d'avance, il faut les configurer une fois le VE lancé.

Le dernier paramètre que nous verrons pour ce fichier est le lien ("*lxc.network.link*") ; ce paramètre va permettre d'indiquer à quel bridge nous voulons connecter notre interface.

3.3.2.2 Bridges

Les bridges (ou pont en francais) sont des équipements réseaux qui permettent de relier deux (ou plus) interfaces de manière complètement transparente : en observant les paquets qui transitent, le pont peut connaitre les adresses mac des interfaces, et ainsi, rediriger les paquets. Les ponts peuvent par exemple, être utilisés pour rediriger une connexion Ethernet : une machine se connecte en Ethernet, une seconde machine se connecte à la première, elles établissent un pont entre deux de leurs interfaces (une interface de la première machine, et une interface de la deuxième machine), et, ainsi, la deuxième machine peut avoir accès à internet.

Lorsqu'un VE se lance, une interface se créé sur la machine hôte pour chaque interface présente sur le VE. Ces interfaces créées ont un noms qui commencent toujours par "*VETH*" suivit de quatre caractères. Ces interfaces sur la machine hôte representent les interfaces de l'environement virtuels, et vons nous permettre des reliers les environnements entre eux ; il est possible de voir la correspondance entre les interfaces d'un container et les interfaces de la machine hôte grace a la commande "*lxc-info ...*".

Il est donc necessaire de relier ces interfaces sur la machine hôte a un même bridge pour que les containers puissent communiquer !

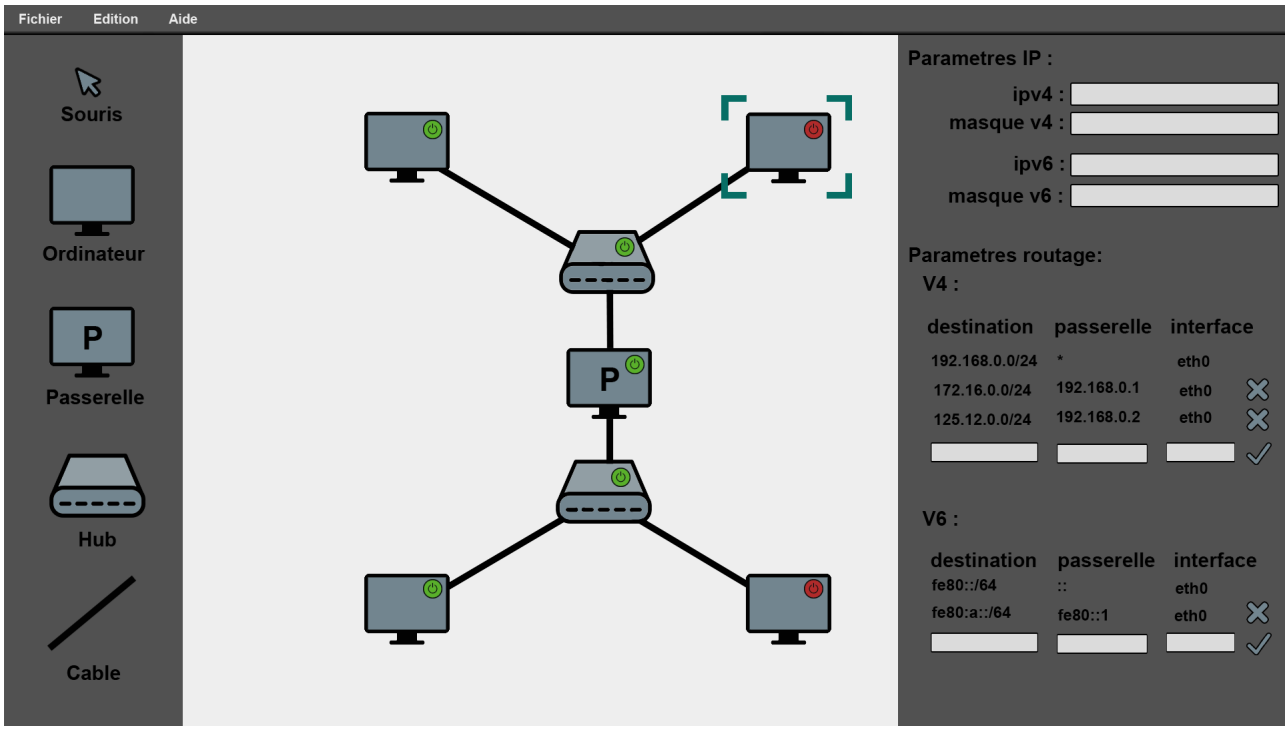
4 Langage et autres outils utilisés

Les principaux langages utilisés seront :
— Le C++, car c'est un langage orienté objet compatible avec l'API C de LXC ;
— Le langage shell pour utiliser des fonctions telles que la gestion des bridges (brctl).
Pour créer l'interface graphique, nous utiliserons l'API Qt. En effet, cette dernière nous semble être de qualité, et QtCreator rend la compilation plus pratique.

5 Architecture du programme

5.1 Vues

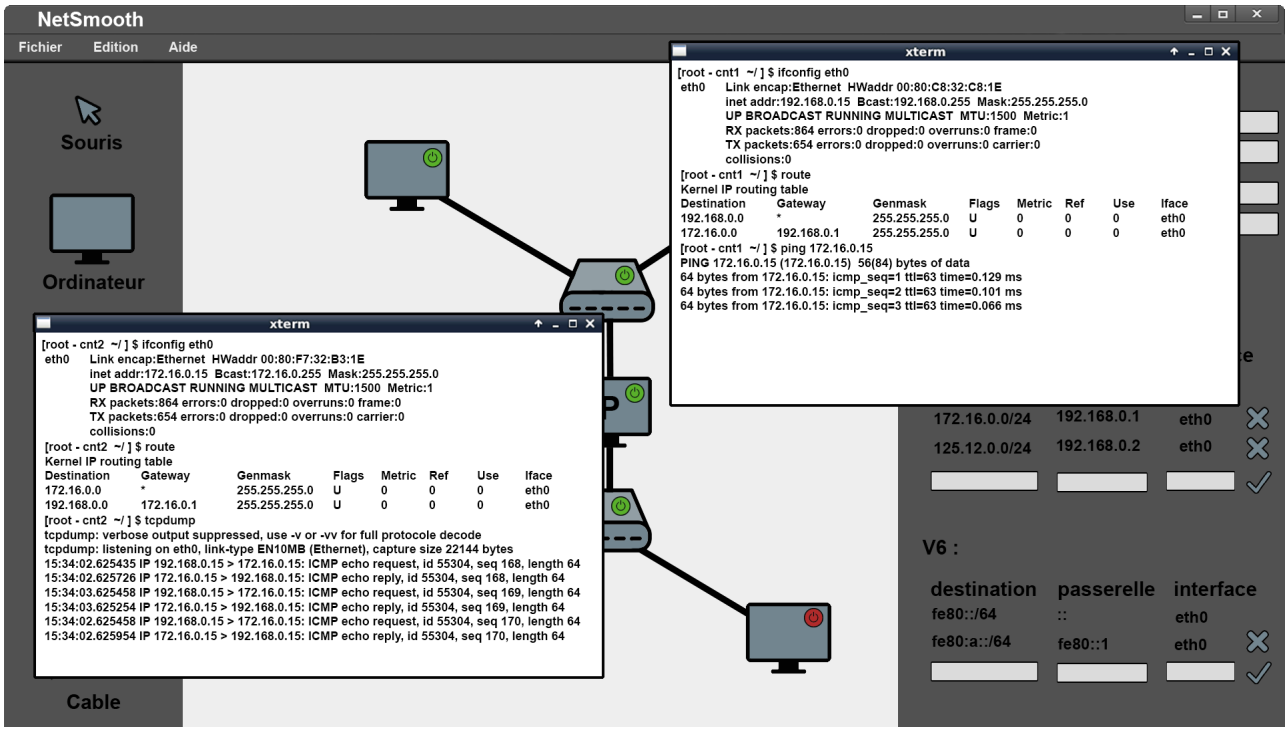
Le logiciel a une seule interface :



Elle est divisible en 4 parties :

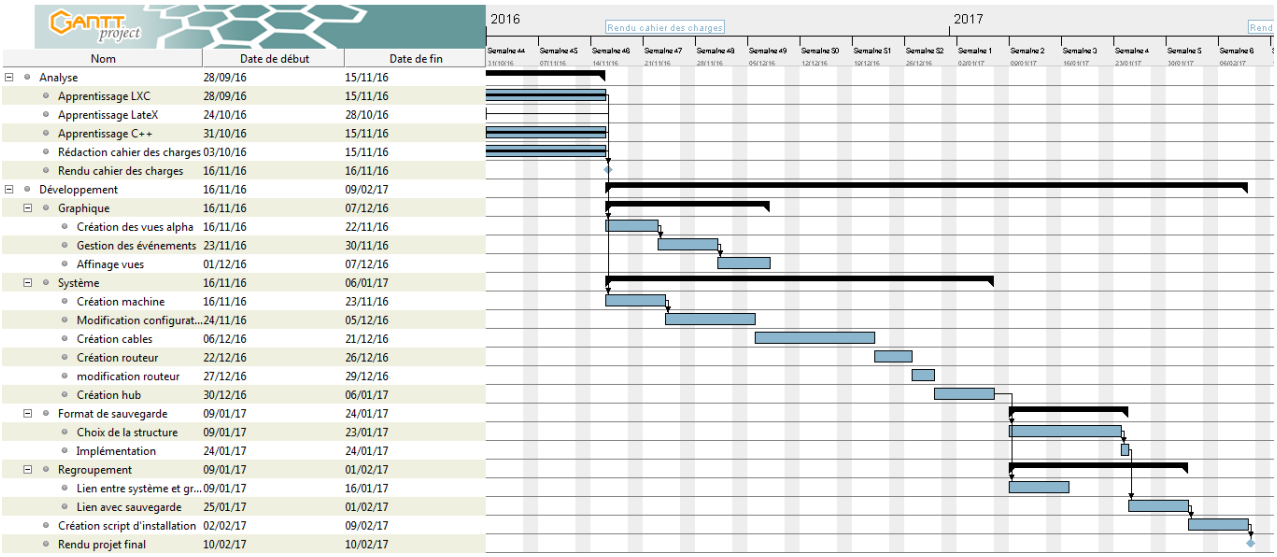
- La barre supérieure (Fichier, Éditions, Aide...),
- la barre de sélection (à gauche) où on peut choisir quel entité ajouter,
- l'écran principal (au centre),
- la barre d'information (à droite) où sont affichées les configurations des entités sélectionnées.

Voici un exemple avec un ping entre deux machines capturé avec tcpdump :



6 Etapes prévues

Voici un diagramme de Gantt qui décrit les étapes et l'avancement actuel du projet :



7 Notice d'utilisation de LXC

Ce guide est destinée aux personnes voulant débuter avec LXC. Les bugs rencontrés sont précisés en bas de page.

7.1 Bien débuter

Toutes les commandes exécutées sont effectuées en root (super-utilisateur).

7.1.1 Installer LXC

7.1.1.1 Arch-linux *#pacman -S lxc arch-install-scripts*

7.1.1.2 Debian *#apt-get install lxc*

7.1.2 Créer un container

#lxc-create -t download -n name : créer un container de nom name en proposant la liste des images d'OS possibles à télécharger.

#lxc-create -t download -n name -d debian -r jessie -a i386 : créer un container de nom *name* en téléchargeant une image de distribution *debian*, de release *jessie* et d'architecture *i386* (32 bits). [1]

#lxc-ls : obtenir la liste des containers créés –fancy pour plus de détails.

7.1.3 Démarrer un container

#lxc-start -n name -d : démarrer le container de nom *name* en daemon *(-d)*.
Par défaut, aucun compte utilisateur n'est créé. Il faut donc se connecter en root.

#lxc-attach -n name : se connecter en root sur le container *name*.
Une fois un compte utilisateur créé, et le container lancé, il est possible de se connecter a une session en particulier.

#lxc-console -n name -t 0 : ouvrir un écran de login sur le terminal tty0 du container *name*. [2]
Ou, si le conatiner est stoppé, il est possible de lancer le container sur un écran de login, en enlevant l'option *-d* a la commande *lxc-start*

7.1.4 Configurer un container

7.1.4.1 Fichier de configuration Le fichier de configuration d'un container *name* est /var/lib/lxc/*name*/config.
Voici un exemple de configuration de passerelle :
RTFM : lxc.container.conf

/var/lib/lxc/passerelle/config

# Distribution configuration
lxc.include = /usr/share/lxc/config/debian.common.conf

1. Créer un container d'architecture 64 bits sur un host 32 bits **ne fonctionne pas**.
2. **BUG :** sur certaines distributions, -t différent de 0 ne fonctionne pas

```
lxc.arch = x86_64

# Container specific configuration
lxc.rootfs = /var/lib/lxc/passerelle/rootfs
lxc.rootfs.backend = dir
lxc.utsname = passerelle

# Network configuration
lxc.network.type = veth
lxc.network.name = eth0
lxc.network.link = lxcbr0
lxc.network.flags = up
lxc.network.hwaddr = 00 :16 :3e :5b :0e :8f
lxc.network.ipv4 = 172.16.1.1
lxc.network.ipv6 = fec00 :0 :0 :2 : :1

lxc.network.type = veth
lxc.network.name = eth1
lxc.network.link = lxcbr0
lxc.network.flags = up
lxc.network.hwaddr = 00 :16 :3e :5b :0e :8f
lxc.network.ipv4 = 192.168.1.1
lxc.network.ipv6 = fc00 :0 :0 :1 : :1
```

Ici, le container possède deux interfaces (eth0 et eth1) avec chacune leurs adresses ipv4 (*lxc.network.ipv4*), ipv6 (*lxc.network.ipv6*) et MAC (*lxc.network.hwaddr*).
lxc.network.flags indique quelle action effectuer (up active l'interface).
lxc.network.type indique quelle type de virtualisation de réseau utiliser (RTFM).
lxc.network.link indique l'interface à utiliser pour le vrai trafic, cette notion sera vu plus en détail par la suite.

7.1.4.2 ifconfig, ip

Commandes ip/ifconfig pour ipv4,ipv6

7.1.5 Relier les containers avec les ponts (ou bridges)

Pour relier les containers entre eux, ou même au host, LXC utilise les bridge (pont en francais). Lors du lancement d'un container, l'activation de chacune de ses interfaces virtuelles va créer une interface sur la machine host. Relier ces interfaces à un même pont permet de relier les containers "physiquement".

7.1.5.1 La méthode manuelle Il est possible de relier les containers aux bridges apres les avoir lancer :
#brctl addbr br0 : crée un bridge de nom *br0*
#ifconfig br0 up : active l'interface *br0*
#brctl addif br0 VETH12345 : relie le bridge *br0* à l'interface physique *VETH12345*[3]

3. Voir l'autre doc pour plus de details sur le nom de l'interface

7.1.5.2 La méthode automatique Il est aussi possible de relier les containers par le fichier de configuration. [4] Attention, si vous esssayez de lancer un container, qui dépend d'un bridge non créé, il y aura une erreur.
#lxc.network.link permet de préciser le pont auquel relier l'interface créée.

Exemple :
lxc.network.type = veth
lxc.network.name = eth1
lxc.network.link = lxcbr0
lxc.network.flags = up
lxc.network.hwaddr = 00 :16 :3e :5b :0e :8f
lxc.network.ipv4 = 192.168.1.1
lxc.network.ipv6 = fc00 :0 :0 :1 : :1

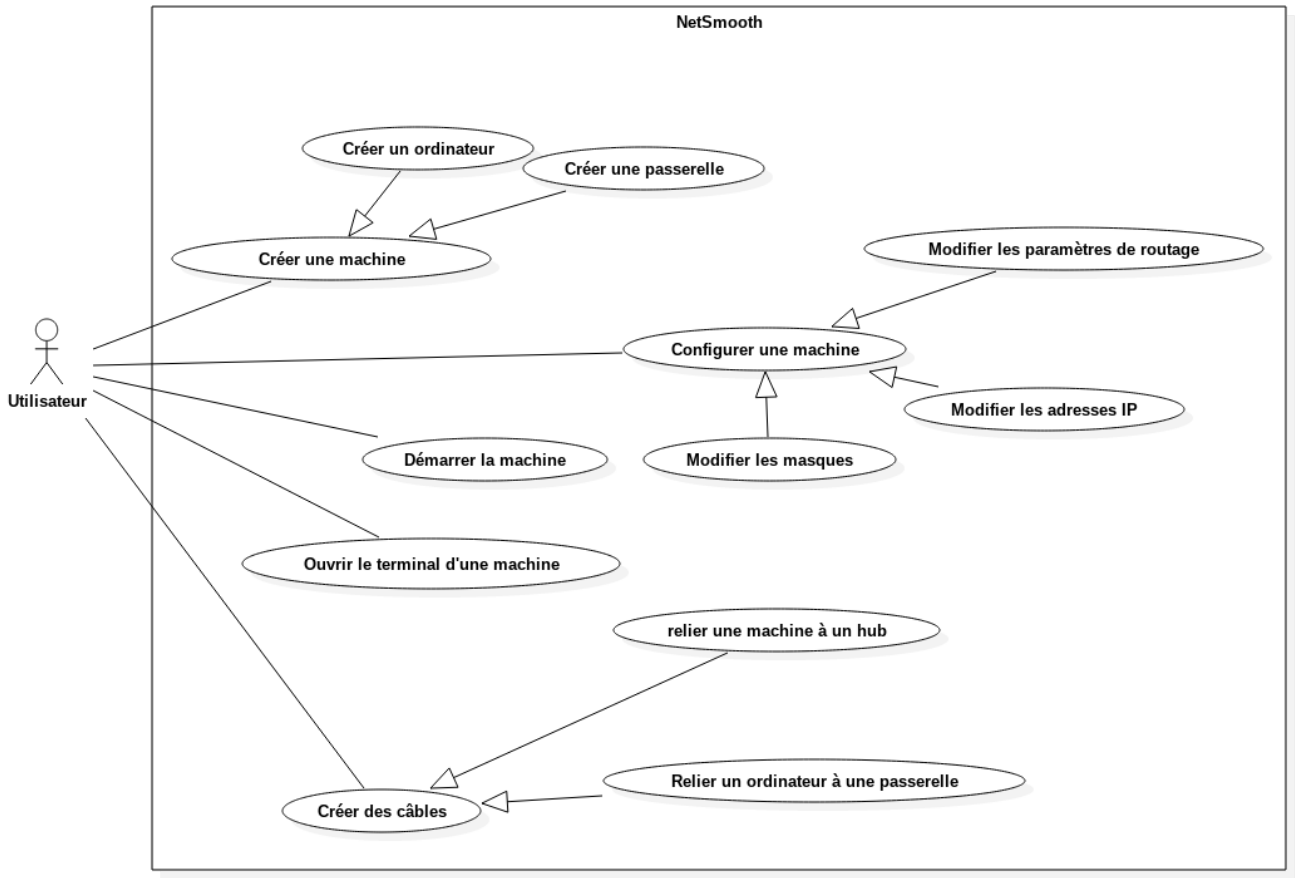
L'interface eth1 sera reliée (indirectement) au pont *lxcbr0*.

4. **Attention :** le pont ne peut pas être **créé** par le fichier de configuration. Il faut le créer manuellement.

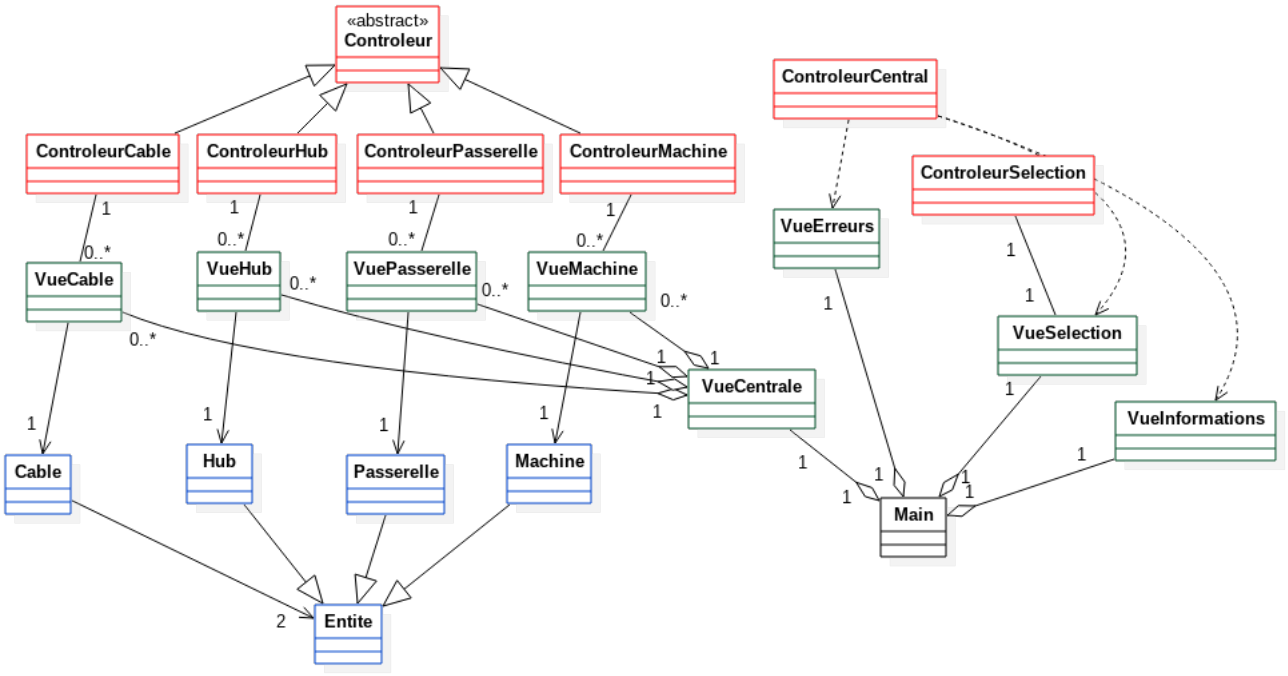
8 Annexe

8.1 Diagrammes

8.1.1 Diagramme de cas d'utilisation :



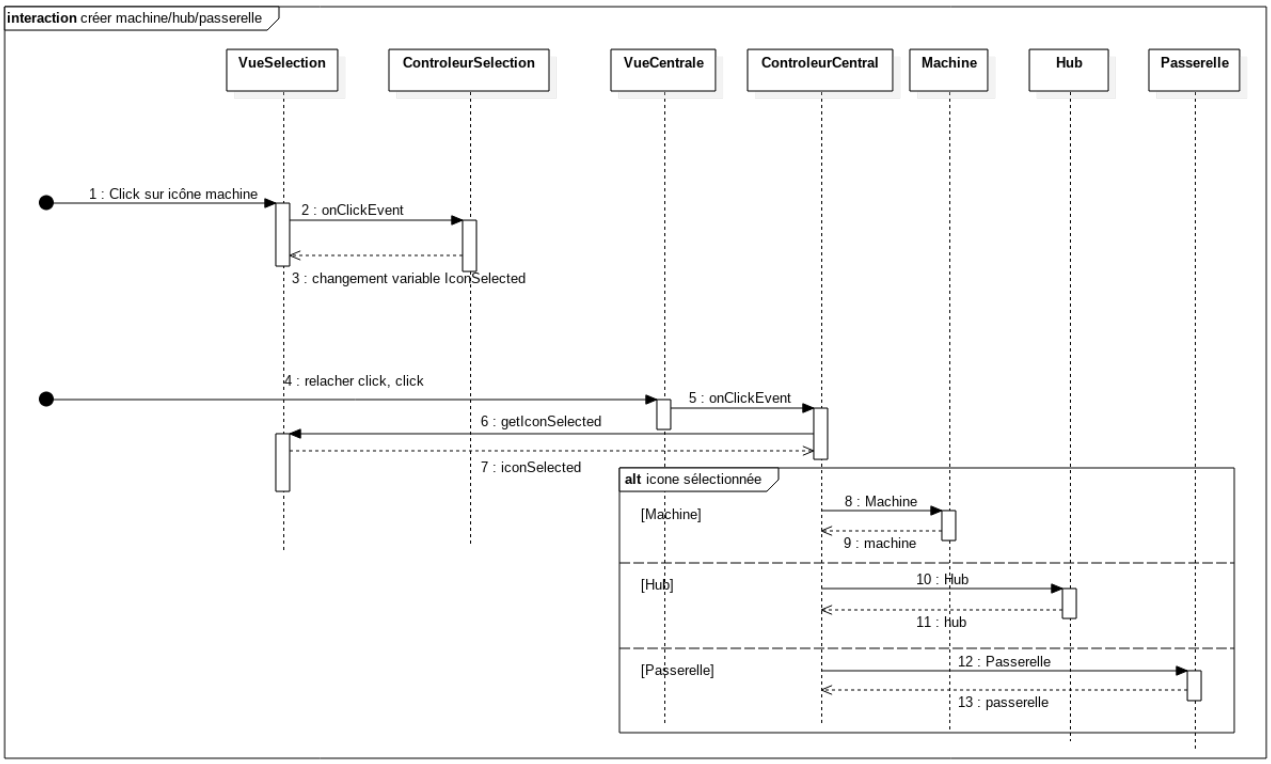
8.1.2 Diagramme de classe d'analyse



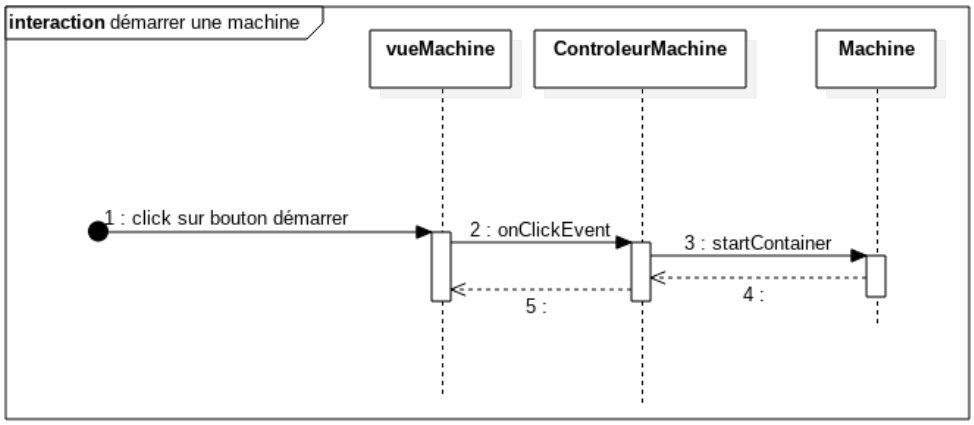
Les classes en rouge représentent les contrôleurs, les classe en vert les vues, puis les classes en bleu les modèles.

8.2 Diagrammes de séquence

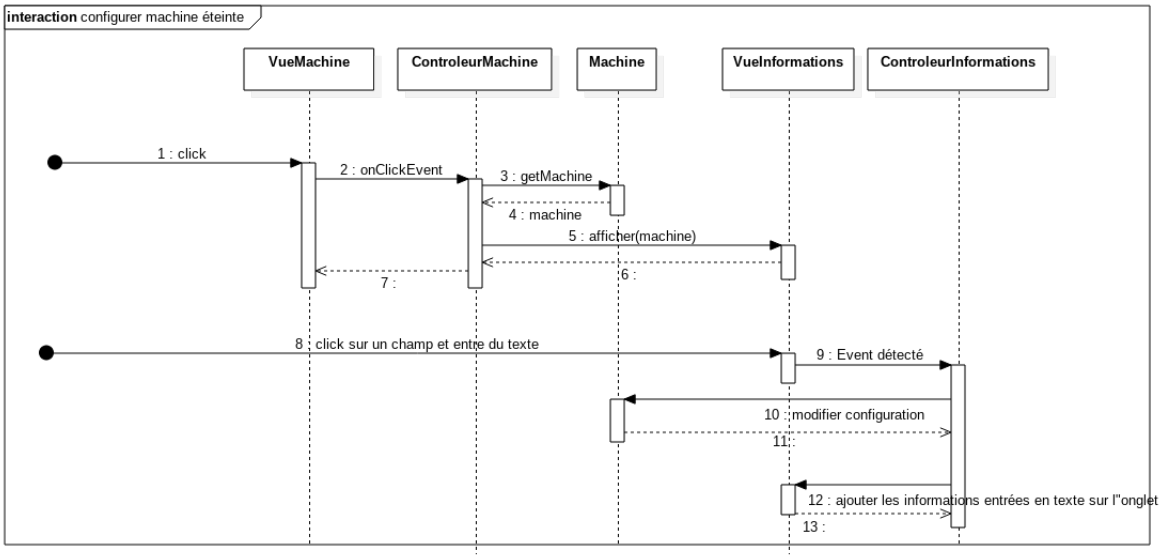
Créer une machine, un hub ou une passerelle - première version



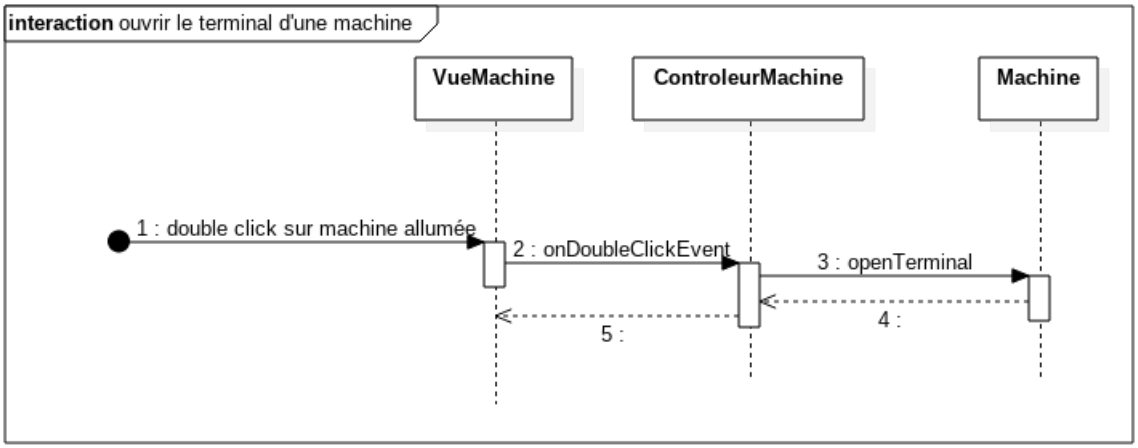
Démarrer une machine ou une passerelle



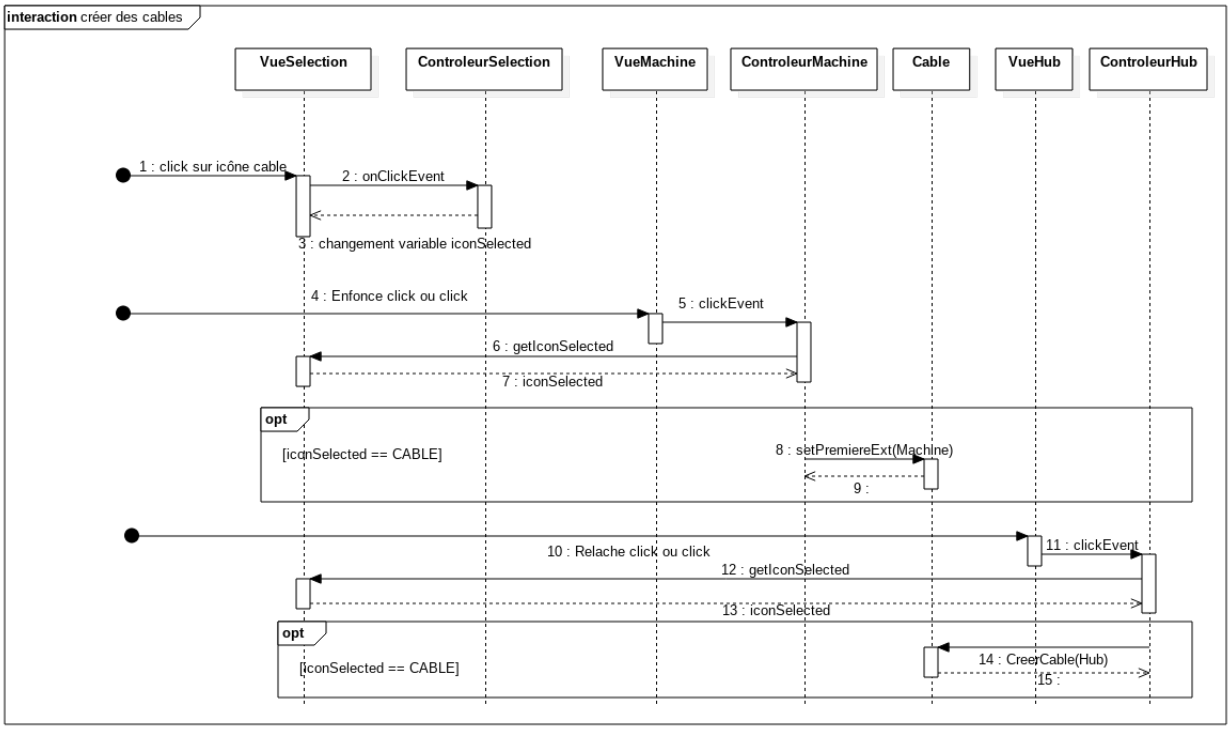
Configurer une machine/passerelle éteinte



Ouvrir terminal d'une machine allumée



Créer un cable



8.3 Scénarios

8.3.1 Premier cas

TITRE : Créer une machine virtuelle.
OBJECTIF : Pouvoir créer une machine pour l'utiliser dans un réseau.
ACTEURS : L'utilisateur.
TYPE : Scénario nominale.
PRECONDITIONS : Il faut avoir lancé le logiciel.
POSTCONDITIONS : La machine est créée.
DESCRIPTIF :

1. L'utilisateur clique sur l'icône de la machine.
2. L'utilisateur déplace l'icône sur la zone qui représente le réseau.
3. L'utilisateur saisit l'adresse IP4 et/ou IP6 de la machine.
4. L'utilisateur saisit le masque 4 et/ou 6 de la machine.
5. L'utilisateur veut définir des paramètres de routages.
6. L'utilisateur saisit l'adresse destination en IP4/IP6.
7. L'utilisateur saisit l'adresse de la passerelle en IP4/IP6
8. L'utilisateur saisi l'interface qu'il veut utiliser.

FLUX ALTERNATIFS : 5. L'utilisateur ne veut pas définir des paramètres de routage.
5a. L'utilisateur laisse les champs vides.
CAS REFERENCES : aucun.

8.3.2 Deuxième cas

TITRE : Créer une passerelle.
OBJECTIF : Pouvoir créer une passerelle et l'utiliser dans le réseau.
ACTEURS : L'utilisateur.
TYPE : Scénario nominale.
PRECONDITIONS : Avoir lancé le logiciel.
POSTCONDITIONS : La passerelle est créée.
DESCRIPTIF :

1. L'utilisateur clique sur l'icône de la passerelle.
2. L'utilisateur déplace l'icône sur la zone qui représente le réseau.
3. L'utilisateur choisi le nombre d'interfaces qu'il veut.
4. L'utilisateur saisi l'adresse IP4 et/ou IP6 de la passerelle pour toutes ses interfaces.
5. L'utilisateur saisi le masque 4 et/ou 6 de la passerelle pour toutes ses interfaces.
6. L'utilisateur veut définir des paramètres de routages.
7. L'utilisateur saisi l'adresse destination en IP4/IP6.
8. L'utilisateur saisi l'adresse de la passerelle en IP4/IP6.
9. L'utilisateur saisi l'interface qu'il veut utiliser.

FLUX ALTERNATIFS :

5. L'utilisateur ne veut pas définir des paramètres de routage.
5a. L'utilisateur laisse les champs vides.

CAS REFERENCES : aucun.

8.3.3 Troisième cas

TITRE : Créer un hub.
OBJECTIF : Pouvoir créer un hub pour l'utiliser dans un réseau.
ACTEURS : L'utilisateur.
TYPE : Scénario nominale.
PRECONDITIONS : Il faut avoir lancé le logiciel.
POSTCONDITIONS : Le hub est créé.
DESCRIPTIF : 1. L'utilisateur clique sur l'icône du hub.
2. L'utilisateur déplace l'icône sur la zone qui représente le réseau.
CAS REFERENCES : aucun.

8.3.4 Quatrième cas

TITRE : Relier une machine à une machine/hub/passerelle.
OBJECTIF : Pouvoir relier une machine avec un autre dispositif pour qu'il puisse se « voir» dans le réseau.
ACTEURS : L'utilisateur.
TYPE : Scénario nominale.
PRECONDITIONS : Il faut avoir lancé le logiciel, créée une machine et un autre dispositif.
POSTCONDITIONS : La machine est reliée avec l'autre dispositif.
DESCRIPTIF :
1. L'utilisateur clique sur l'icône du câble.
2. L'utilisateur déplace l'icône sur la machine à relié.
3. L'utilisateur clique sur le dispositif auquel il veut relier la machine.

CAS REFERENCES : aucun.

8.3.5 Cinquième cas

TITRE : Allumer un dispositif.
OBJECTIF : Pouvoir allumer un dispositif pour qu'il soit visible sur le réseau.
ACTEURS : L'utilisateur.
TYPE : Scénario nominale.
PRECONDITIONS : Il faut avoir lancé le logiciel, et créer un dispositif.
POSTCONDITIONS : Le dispositif est allumé.
DESCRIPTIF :
1. L'utilisateur clique sur l'icône on/off.

CAS REFERENCES : aucun.