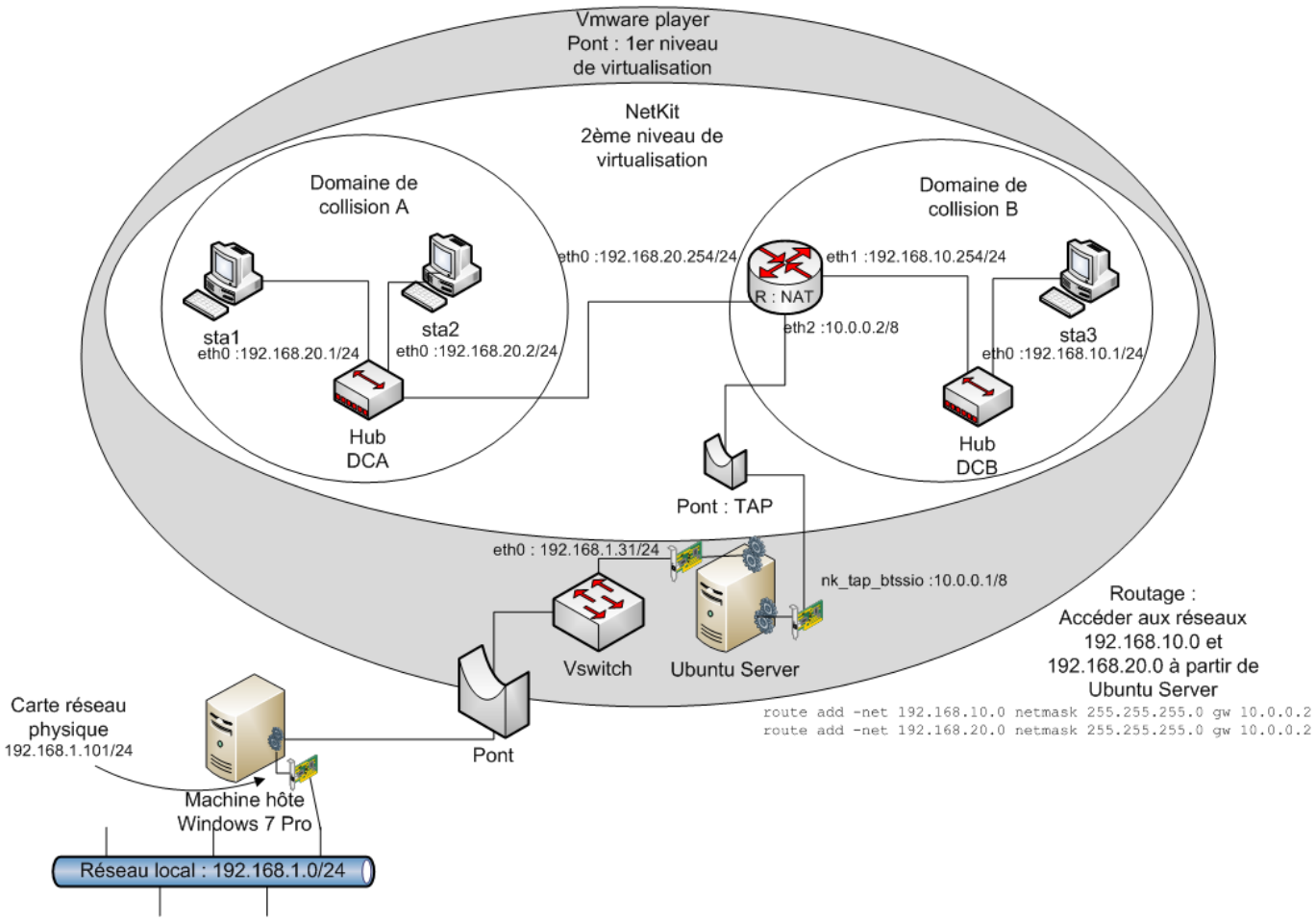
Découverte de Netkit

Présentation de l'infrastructure réseau à réaliser

Voici ce que vous allez réaliser à la fin de ce tutoriel de découverte de Netkit :



Le réseau local utilisé est le réseau physique que vous avez au lycée. Pour ce tutoriel, ce réseau est identifié avec l'adresse 192.168.1.0/24. Il faut **adapter cette adresse de réseau** à l'adresse réelle du réseau local du lycée. L'adresse IP 192.168.1.101 correspond à l'**hôte Windows** qui héberge RINN et l'adresse IP 192.168.1.31 est celle de la VM RINN basée sur Ubuntu Server sous VMware Player/Workstation. Ces adresses ont été attribuées par le serveur DHCP du réseau. Ces adresses seront différentes pour votre environnement. Toutes les autres adresses IP, c'est à dire celle des VMs Netkit seront définies dans le cadre de cet atelier.

Utiliser Netkit

Netkit fournit deux groupes de commandes :

- les **vcommandes**, préfixées par '**v**', qui permettent de manipuler une seule VM.
- les **lcommandes**, préfixées par '**l**' servent à manipuler des ensembles complexes de machines virtuelles en réseau. Dans le langage de Netkit, il s'agit des 'Lab' (laboratoires).

Si vous souhaitez travailler avec une seule VM, utilisez les **vcommandes**. Sinon, pour travailler avec plusieurs VMs, il est préférable et bien plus pratique de créer un laboratoire (Lab) et d'utilisez alors les **lcommandes**.

Utilisation de machines autonomes

Les v-commandes

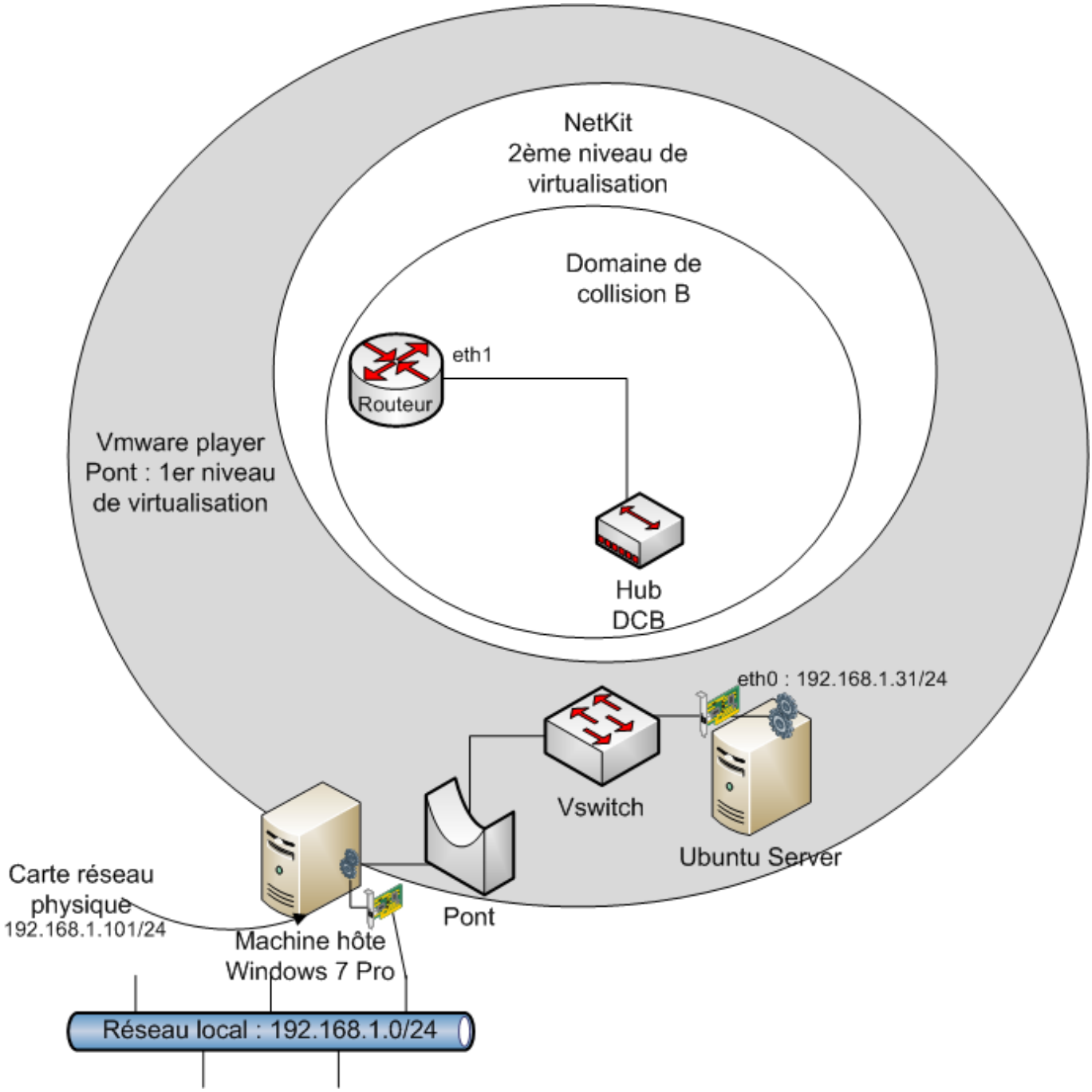
Commande	Action
vstart	→ pour démarrer une machine
vhalt	→ arrêter une machine
vlist	→ donner la liste des VMs actives
vconfig	→ configurer à la volée une VM comme par exemple affecter une interface à la volée.
vcrash	→ arrêter brutalement une VM (crash virtuel)
vclean	→ nettoyer les processus, configurations et fichiers temporaires créés

Ces commandes sont à utiliser avec des options qui sont décrites dans les pages de manuel.

```
btssio@ubuntunetkit:~$ man vstart
btssio@ubuntunetkit:~$ man vconfig
…
```

Exemple : créer un routeur

Réseau à obtenir :

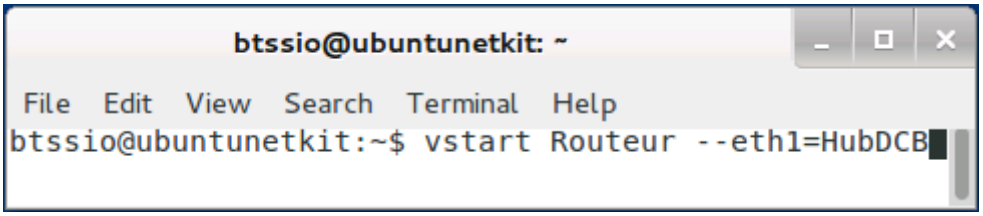


Le routeur va être créé en tant que machine virtuelle, avec une carte réseau eth0 connectée à un domaine de collision appelé **HubDCB**. Prenez la bonne habitude de donner des noms significatifs aux VMs comme aux domaines de collision. Un domaine de collision correspond à un concentrateur.

Pour l'instant, votre VM Routeur est créée avec une carte réseau mais sa configuration IP n'est pas encore établie. Cette VM est isolée des autres réseaux.

Note : votre VM est créée pour être un routeur. De ce fait, ce routeur est représenté dans le schéma avec le symbole du routeur plutôt qu'avec le symbole d'un ordinateur.

Dans le terminal, tapez la commande suivante :



Voici la VM Netkit créée :

Testez quelques commandes :

A partir de votre VM Routeur, tapez les commandes qui suivent :

```
Routeur:~#  ifconfig
```

La carte réseau eth0 n'apparait pas alors qu'elle bien été précisé au lancement de la VM.

```
Routeur:~# ifconfig
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:2 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:100 (100.0 B)  TX bytes:100 (100.0 B)
```

Seule la carte de loopback (lo) est affichée

Pour la visualiser, utilisez le paramètre -a :

```
Routeur:~#  ifconfig –a
```

La carte eth1 existe bien mais sans adresse IP :

```
Routeur:~# ifconfig -a
eth1      Link encap:Ethernet  HWaddr c6:d9:6a:3d:0a:c3
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:5

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:2 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:100 (100.0 B)  TX bytes:100 (100.0 B)

teql0     Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
-00
          NOARP  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

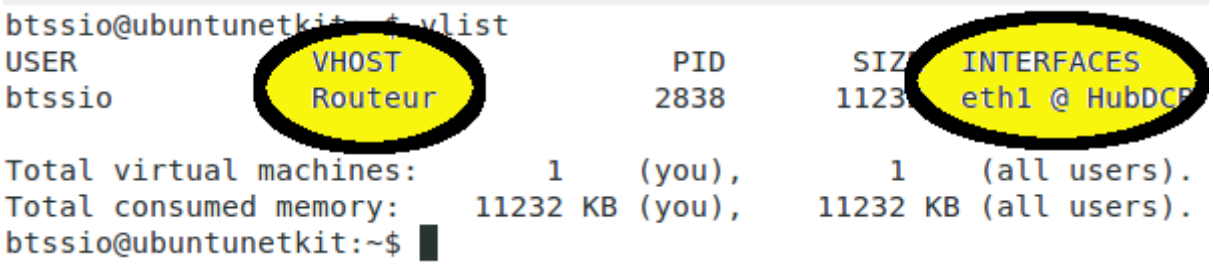
Configuration IP statique de l'interface eth1 de la VM routeur :

```
Routeur:~#  ifconfig eth1 192.168.10.254 netmask 255.255.255.0
```

Le fichier de la VM Routeur : Lors de la création d'une VM, un disque virtuel est créé qui contiendra toutes les modifications que vous effectuerez sur cette machine. De cette manière, lors du prochain lancement, vous retrouverez la configuration précédemment établie. Cependant ces disques virtuels sont volumineux (environ 10 Go) pour pouvoir être facilement sauvegardés. Deux autres moyens doivent peuvent être utilisés pour cela : l'écriture d'un script pour les v-commandes ou bien la création d'un 'lab'.

Pour visualiser la liste des VMs créées ainsi que leurs interfaces, tapez la commande suivante dans le Terminal :

```
btssio@ubuntunetkit:~$  vlist
```



Pour arrêter une VM Netkit :

```
btssio@ubuntunetkit:~$   vhalt –r Routeur
```

Le paramètre –r permet de supprimer le disque de la VM Netkit.

Le domaine de collision 'tap'

Un accès internet est parfois indispensable à partir des VMs pour ajouter par exemple des paquets manquants ou pour réaliser des tests. L'accès au réseau physique n'est possible qu'à partir de l'interface physique de l'hôte qui héberge les VMs. Un domaine de collision particulier '**tap**' est créé à cet effet.

Remarque : La fonction **TUN/TAP** du noyau Linux est la solution retenue afin de faire communiquer des machines virtuelles avec un hôte. **TUN** et **TAP** sont des pilotes réseaux qui simulent des interfaces réseaux virtuelles complètement émulées sous forme logicielle. **TAP** simule une interface Ethernet et fonctionne comme une interface de niveau 2 (OSI). **TUN** simule un TUNnel et fonctionne au niveau 3 (réseau – IP) du modèle OSI. TAP crée un pont entre les 2 interfaces, **TUN** assure du routage. Ces interfaces sont manipulées à l'aide de la commande tunctl du paquet **uml-utilities**.

Création de la VM Routeur avec le domaine de collision '**tap**' :

Format général de la commande :

```
btssio@ubuntunetkit:~$   vstart vm --ethN=tap,TAP-ADDRESS,GUEST-ADDRESS
```

où **TAP-ADDRESS** est l'adresse côté hôte et **GUEST-ADDRESS**, l'adresse côté VM. Ces adresses doivent être sur le même réseau ou sous- réseau IP. Cela donne pour le Routeur :

```
btssio@ubuntunetkit:~$   vstart Routeur -—eth1=HubDCB -
—eth2=tap,10.0.0.1,10.0.0.2
```

Remarque : La création d'une interface TAP nécessite des droits administrateur. Vous aurez à fournir le mot de passe du compte btssio pour exécuter la commande vstart en mode superutilisateur.

```
btssio@ubuntunetkit: ~
File  Edit  View  Search  Terminal  Help
btssio@ubuntunetkit:~$ vstart Routeur --eth0=HubDCB --eth2=tap,10.0.0.1,10.0.0.2


============= Starting virtual machine "Routeur" =============
   Kernel:     /home/btssio/netkit/kernel/netkit-kernel
   Modules:    /home/btssio/netkit/kernel/modules
   Memory:     32 MB
   Model fs:   /home/btssio/netkit/fs/netkit-fs
   Filesystem: /home/btssio/Routeur.disk
   Interfaces: eth0 @ HubDCB    (/home/btssio/.netkit/hubs/vhub_btssio_HubDCB.cn
ct)
               eth2 @ tap       (/home/btssio/.netkit/hubs/vhub_btssio_tap.cnct)
   Hostfs at:  /home/btssio

Running ==> /home/btssio/netkit/bin/uml_switch -hub -unix /home/btssio/.netkit/h
ubs/vhub_btssio_HubDCB.cnct </dev/null 2>&1
******** Starting Internet connected virtual hub ********
   10.0.0.1 (host side) - 10.0.0.2 (guest side)
********     (root privileges are required)      ********
Running ==> /home/btssio/netkit/bin/manage_tuntap start btssio 10.0.0.1 10.0.0.2
 /home/btssio/.netkit/hubs/vhub_btssio_tap.cnct
btssio's password:
```

Voici la nouvelle carte réseau sur votre serveur Ubuntu :

```
nk_tap_btssio Link encap:Ethernet  HWaddr 42:3d:b0:25:f6:23
          inet addr:10.0.0.1  Bcast:10.255.255.255  Mask:255.0.0.0
```

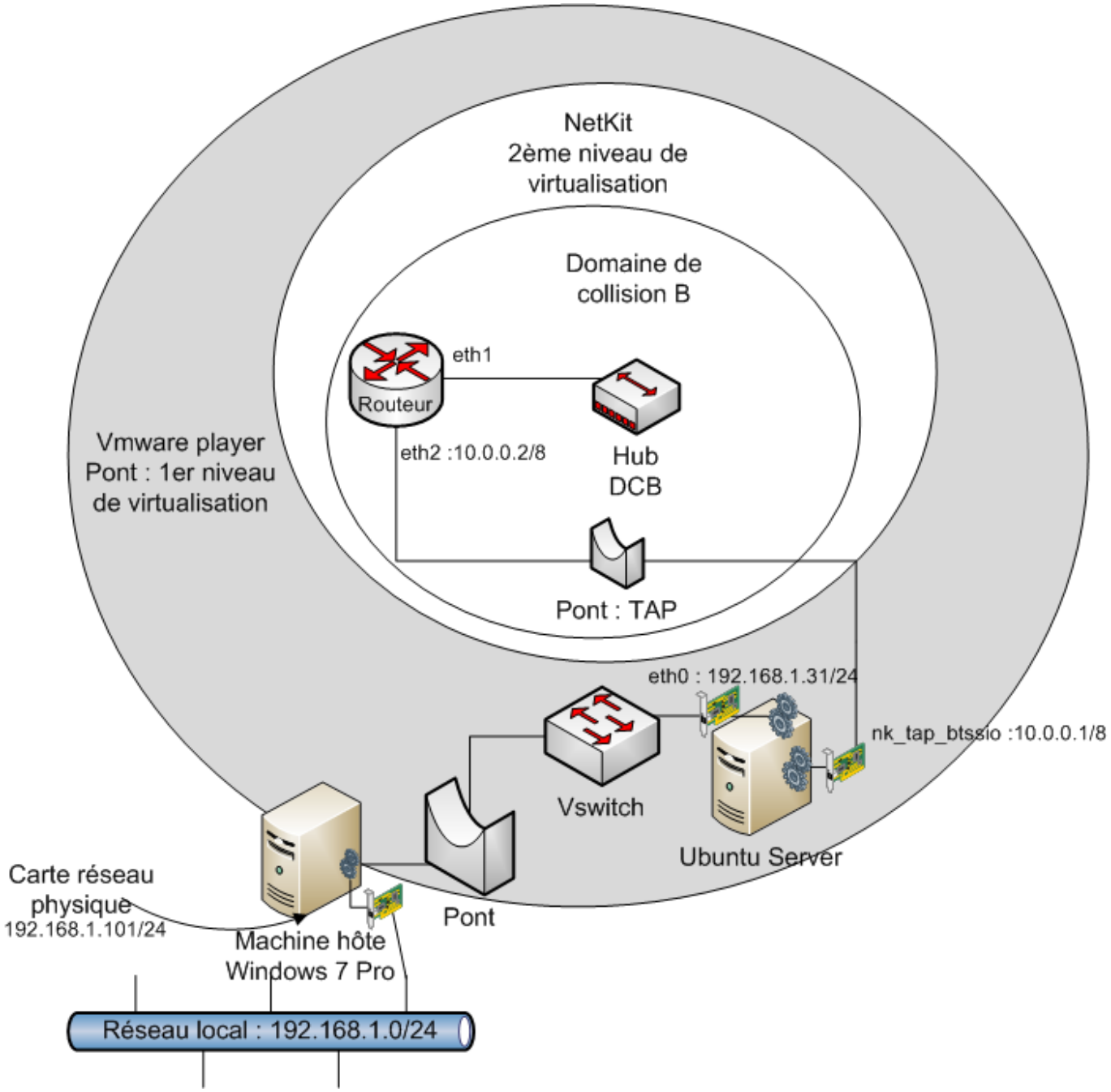
Et la nouvelle carte réseau sur la VM Netkit Routeur :

```
Routeur:~# ifconfig -a
eth0      Link encap:Ethernet  HWaddr da:8a:e6:78:49:1f
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:5

eth2      Link encap:Ethernet  HWaddr 1e:da:dd:9d:f9:f8
          inet addr:10.0.0.2  Bcast:10.255.255.255  Mask:255.0.0.0
          inet6 addr: fe80::1cda:ddff:fe9d:f9f8/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:21 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1949 (1.9 KiB)  TX bytes:468 (468.0 B)
          Interrupt:5

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
```

Voici le réseau obtenu :



Votre VM Netkit Routeur peut communiquer avec Ubuntu Server :

Routeur

```
Routeur:~# ping -c 4 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=2.69 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.278 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.274 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=1.06 ms

--- 10.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3037ms
rtt min/avg/max/mdev = 0.274/1.078/2.694/0.987 ms
Routeur:~#
```

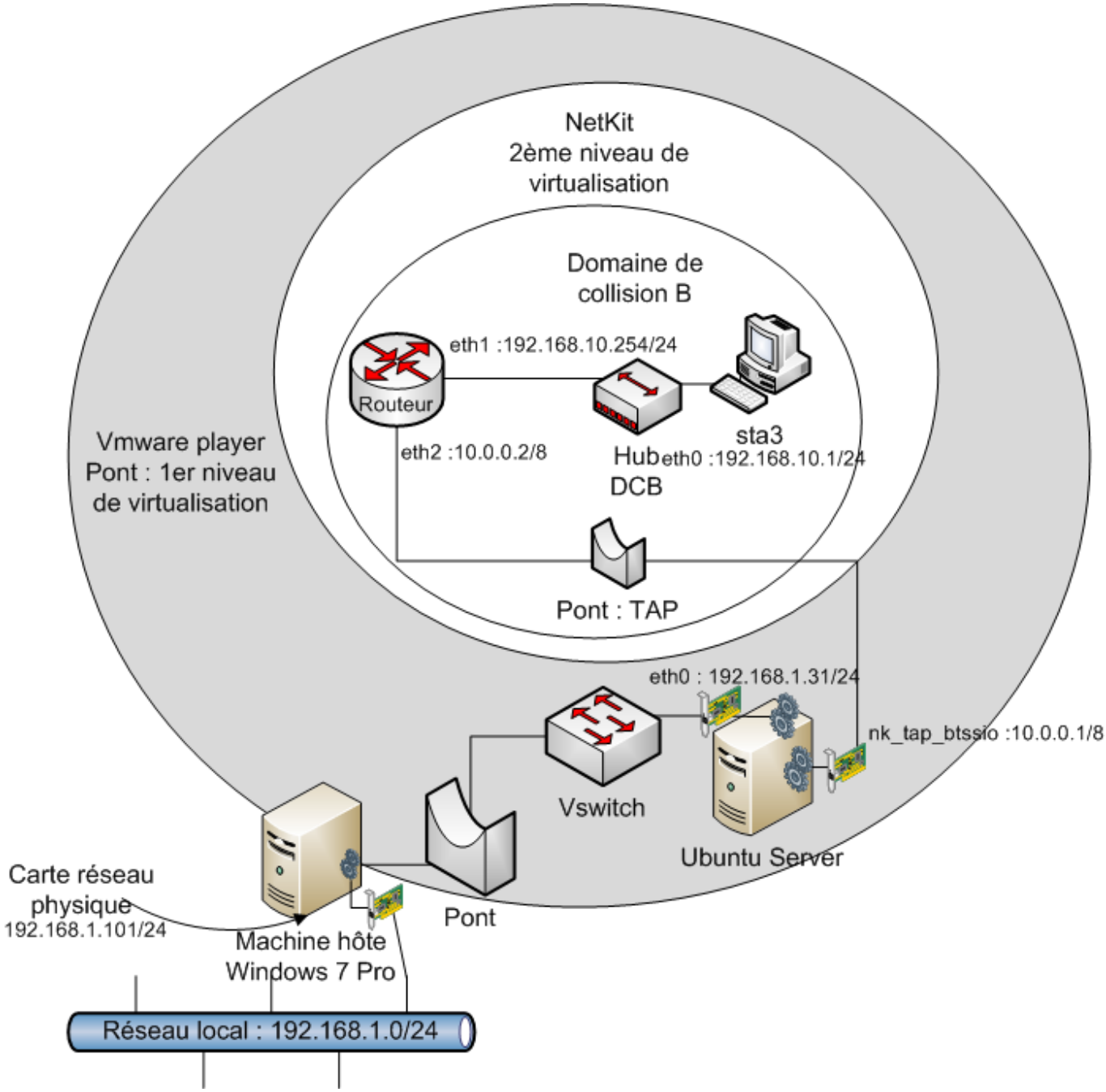
Et avec Internet via la carte eth0 de la VM Ubuntu Server (adresse IP 192.168.1.30) puis via la carte réseau physique de l'hôte (Windows 7 adresse IP 192.168.1.101).

Routeur

```
Routeur:~# ping -c 4 192.168.1.101
PING 192.168.1.101 (192.168.1.101) 56(84) bytes of data.
64 bytes from 192.168.1.101: icmp_seq=1 ttl=127 time=1.14 ms
64 bytes from 192.168.1.101: icmp_seq=2 ttl=127 time=0.614 ms
64 bytes from 192.168.1.101: icmp_seq=3 ttl=127 time=0.678 ms
64 bytes from 192.168.1.101: icmp_seq=4 ttl=127 time=0.645 ms

--- 192.168.1.101 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3039ms
rtt min/avg/max/mdev = 0.614/0.770/1.143/0.216 ms
Routeur:~#
```

Exercice 1 Arrêter la VM Netkit Routeur et créez le réseau suivant, qui correspond au précédent en y ajoutant la VM Netkit sta3. Utilisez les v-commandes et les commandes linux nécessaires. Attention à bien configurer les adresses IP : De l'interface **eth1** du **Routeur** (192.168.10.254) ; De l'interface **eth0** de **sta3** (192.168.10.1) ; De configurer une adresse de passerelle pour **sta3** (192.168.10.254).



L'ipmasquerade

Si vous avez bien réalisé cet exercice, sta3 doit pouvoir communiquer avec Routeur mais pas avec Ubuntu Server et n'ont pas accès au réseau local et à Internet.

Visualisons les routes de tous ces hôtes :

Pour sta3 et à partir de cet hôte :

```
sta3:~#   route -n
```

Une ligne indiquant un '**UG**' confirme qu'il y a bien une route UP (U) et par défaut (G). Cette route par défaut est bien l'adresse IP du routeur **Routeur**.

```
sta3:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.10.0    0.0.0.0         255.255.255.0   U     0      0        0 eth0
0.0.0.0         192.168.10.254  0.0.0.0         UG    0      0        0 eth0
sta3:~#
```

Pour Routeur et à partir de cet hôte :

```
Routeur:~#   route –n
```

```
Routeur:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.10.0    0.0.0.0         255.255.255.0   U     0      0        0 eth1
10.0.0.0        0.0.0.0         255.0.0.0       U     0      0        0 eth2
0.0.0.0         10.0.0.1        0.0.0.0         UG    0      0        0 eth2
Routeur:~#
```

Pour Ubuntu Server et à partir de cet hôte :

```
btssio@ubuntunetkit:~$   sudo route -n
```

Remarque : pour rester en permanence en superutilsateur utiliser la commande :

```
btssio@ubuntunetkit:~$   sudo -s
```

Votre VM Ubuntu Server ne connait pas le réseau 192.168.10.0 (celui de **sta3**).

```
btssio@ubuntunetkit:~$ sudo route -n
[sudo] password for btssio:
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.1.254   0.0.0.0         UG    100    0        0 eth0
10.0.0.0        0.0.0.0         255.0.0.0       U     0      0        0 nk_tap_b
tssio
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
btssio@ubuntunetkit:~$
```

Les paquets envoyés par **Routeur** depuis le réseau 192.168.10.0 arrive bien à Ubuntu Server par son interface **nk_tap_btssio**. Cependant la VM Ubuntu Server ne connaissant pas où se trouve le réseau 192.168.10.0, va chercher à envoyer les paquets vers sa passerelle par défaut 192.168.1.254 qui est la box Internet. Celle-ci bien sûr ignore également ce réseau 192.168.10.0. Cela explique pourquoi les VMs Netkit ne peuvent pas communiquer sur le réseau local et sur Internet. Pour cela, il va falloir activer **l'ipmasquerade** sur la VM Netkit Routeur. Le principe est de « **masquer** » les VM Netkit pour le réseau local et d'utiliser l'adresse IP de Routeur pour toute communication sur le réseau local. Comme vous l'avez compris il s'agit de configurer un routage NAT sur la VM Netkit **Routeur** pour les VMs Netkit.

La commande à utiliser sur R est la suivante :

```
Routeur:~#   iptables -t nat -A POSTROUTING -o eth2 -j MASQUERADE
```

La règle utilise la table de correspondance de paquets de NAT (-t nat) et spécifie la chaîne intégrée

POSTROUTING pour NAT (-A POSTROUTING) sur le périphérique réseau externe du pare-feu (-o eth2 de Routeur). POSTROUTING permet aux paquets d'être modifiés lorsqu'ils quittent le périphérique externe du pare-feu. La cible -j MASQUERADE est spécifiée pour masquer l'adresse IP privée d'un noeud avec l'adresse IP externe du pare-feu / de la passerelle. Cela doit permettre aux machines situées derrière le routeur d'accéder à la machine hôte et à internet.

Maintenant la VM Netkit sta3 accède à Internet. Pour le vérifier :

```
sta3:~#   ping –c 208.67.220.220
```

Remarque : 208.67.222.222 et 208.67.220.220 sont des serveurs de nom publics (open DNS). Vous pouvez aussi utilisez ceux de Google : 8.8.8.8. ou 8.8.4.4.

Par contre, il n'est pas possible d'accéder aux VMs Netkit situées après la VM Netkit Routeur depuis Ubuntu Server. Le réseau 192.168.10.0 est inaccessible. Nous verrons plus loin comment remédier à cette situation. Mais vous avez peut-être une petite idée ?

La résolution de nom DNS

La résolution de nom est généralement réalisée par les DNS. Pour, cela il faut indiquer l'adresse de serveurs de nom (DNS) dans le fichier **/etc/resolv.conf** de la VM Netkit **sta3**.

Editez ce fichier :

```
sta3:~#   nano /etc/resolv.conf
```

Et rajoutez les lignes suivantes :

```
nameserver 208.67.222.222
nameserver 208.67.220.220
```

Testez la résolution de nom et utilisez le navigateur en mode de texte **lynx** accéder au site ww.ac-limoges.fr.

```
sta3:~#   ping www.ac-limoges.fr
sta3:~#   lynx www.ac-limoges.fr
```

Utilisation des labs

Quand vous avez à créer et gérer une seule VM Netkit, vous utilisez les v-commandes. Pour des infrastructures utilisant plusieurs VMs Netkit, il est préférable de créer un laboratoire ou **lab** et de manipuler ce lab avec les **l-commandes**. Ces labs permettent de concevoir mais aussi de conserver une architecture réseau complexe ou que l'on souhaite pouvoir réutiliser et cela d'autant plus facilement qu'un lab se traduit par une arborescence de dossiers contenant des fichiers de configuration. Un lab occupe très peu de place et est facile à sauvegarder et à échanger. Vous

trouverez sur Internet des ressources et des exemples de labs.

Voici des liens vers quelques sites intéressants :

http://wiki.netkit.org/index.php/Main_Page Sur ce site, vous trouverez les dernières versions de Netkit, VisualNetkit (un complément permettant de gérer graphiquement des labs) ainsi que des simulations qui peuvent être très copplexes.

http://kartoch.msi.unilim.fr/wiki/ C'est le site de Julien Iguchi-Cartigny, professeur associé à XLIM, Université de Limoges. Il est utilisateur de Netkit et propose des TP pour ses étudiants.

En cherchant sur le Net, vous verrez que Netkit est utilisé par de nombreuses Universités de part le monde.

Un premier lab

Un Lab 'minimaliste' consiste en une arborescence comprenant :

- un **répertoire** par machine virtuelle qui sera lancée, dans lequel on peut stocker des fichiers. Pour l'instant, laissez ce répertoire vide.
- un fichier de configuration (**lab.conf**) qui décrit les machines virtuelles qui seront lancées, leurs interfaces et les domaines de collision.
- un fichier **VMx.startup** et un fichier **VMx.shutdown** qui indiquent les actions à réaliser lors du lancement ou de l'arrêt de la VMx (VMx correspond au nom de la VM netkit). Cela permet de configurées automatiquement la VMx lors du lancement du lab à l'aide de scripts shell. Les scripts doivent être exécutables.

Votre premier lab :

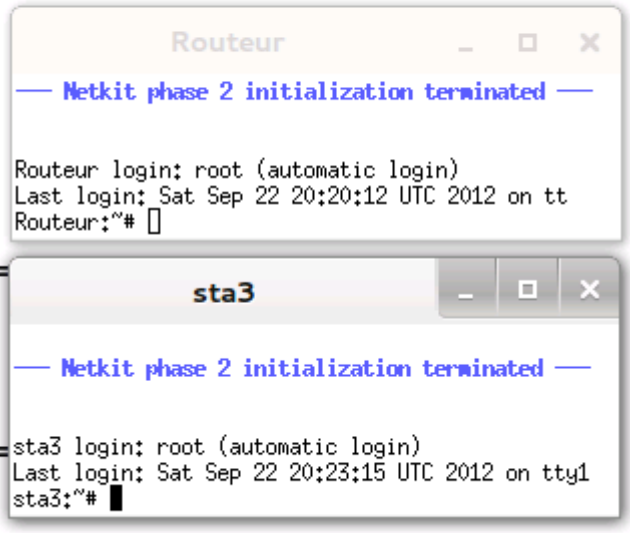
- dans le dossier personnel de btssio (menu **Raccourcis** / **Dossier personnel**), créez le sous-dossier **lab_1** ;
- dans ce sous-dossier, créez un sous dossier pour chaque machine virtuelle **Routeur** et **sta3** ;
- créez un fichier **lab.conf** vide ;
- lancez le **Terminal** et positionnez-vous dans le dossier **lab_1** ;
- tapez la commande **lstart** pour lancer le **lab** et créer les deux **VMs**.

```
btssio@ubuntunetkit:~$ cd lab_1
btssio@ubuntunetkit:~/lab_1$ lstart

======================== Starting lab ============================
Lab directory: /home/btssio/lab_1
Version:       <unknown>
Author:        <unknown>
Email:         <unknown>
Web:           <unknown>
Description:
<unknown>
==================================
Starting "Routeur"...
Starting "sta3"...

The lab has been started.
==================================

btssio@ubuntunetkit:~/lab_1$
```



Les l-commandes

Commande	Action
lstart	→ pour lancer un lab. Il faut être dans le répertoire où est le fichier lab.conf.
lhalt	→ arrêter un lab
lcrash	→ arrêter brutalement le lab
lclean	→ nettoyer les fichier temporaires créés et les disques virtuels.
linfo→ information sur le laboratoire.	
ltest	→ vérification du bon fonctionnement du laboratoire.

Ce premier **lab** était simpliste car on peut aller plus loin dans la **configuration** du lab.

Le fichier lab.conf

Le fichier lab.conf contient :

- la description du lab
- les paramètres des VMs
- la topologie du réseau

Chaque machine est une déclaration **machine[arg]=valeur**

- **machine** est le nom de la VM
- si arg (lire i) est un nombre, alors valeur est le nom du domaine de collision sur lequel l'interface ethi est attaché.
- si arg est une chaîne, alors c'est le nom d'une option de vstart et valeur est un argument de cette option.

Exemple :

```
Routeur[1]= HubDCB
# L'interface eth1 de la VM Routeur est attachee au domaine
# de collision  HubDCB
sta3[0]= HubDCB
Sta3[mem]=256
# L'interface eth0 de la VM sta3 est attachee au domaine
# de collision  HubDCB
# La VM sta3 dispose de 256 MB de mémoire virtuelle.
# La liste des paramètres est dans man vstart
```

Le fichier **lab.conf** peu également contenir des paramètres optionnels informatifs qui sont affichés dans la console de la VM au démarrage du lab.

```
LAB_DESCRIPTION="Premier exemple de reseau "
LAB_VERSION=1.0
LAB_AUTHOR="Techer"
LAB_EMAIL= charles.techer@ac-limoges.fr
LAB_WEB=http://www.ac-limoges.fr
```

Il est possible de préciser la liste des machines à démarrer.

```
machines=''Routeur, sta3''
```

Le domaine de collision 'tap' est créé de cette manière :

```
Routeur[2]=tap,10.0.0.1,10.0.0.2
```

Les sous-répertoires du lab

Le dossier **/hosthome** des VMs correspond à votre **$HOME** sur l'hôte. Cela permet d'avoir un espace disque partagé entre les VMs et l'hôte.

Le dossier **/hostlab** des VMs correspond au répertoire où est installé le lab sur l'hôte. Cela permet également d'avoir un espace disque partagé entre les VMs et l'hôte.

Toute arborescence mise sur le lab, dans le répertoire d'une VM est recopié dans la VM en partant de la racine lors de son lancement. Par exemple, l'arborescence **/etc/resolv.conf** mise dans le répertoire **sta3** du lab **lab-1**, sera recopiée dans la VM **sta3** à l'endroit **/etc/resolv.conf**. Cette copie de fichiers prérenseignés, permet de configurer les VMs lors de leur lancement. Cela complète l'utilisation des fichiers **VMx.startup** et un fichier **VMx.shutdown**. Les fichiers **shared.startup** et **shared.shutdown** concernent toutes les VMs du lab et sont exécutés avant **vmx.startup** et **vmx.shutdown**.

Récapitulons :

- Netkit lance une VM pour chaque sous répertoire existant à moins que la directive 'machines' existe dans lab.conf ;
- Le contenu du répertoire d'une VM est copié dans l'arborescence de la VM

correspondante ;
- Les fichiers shared.startup et shared.shutdown sont exécutés ;
- le fichier VMx.startup est exécuté au lancement de la VM ;
- Un système de gestion de fichier vmx.disk est créé pour chaque VM vmx lors du premier lancement de la VM ;
- le fichier VMx.shutdown est exécuté à l'arrêt de la VM.

Remarques :

Le fichier vmx.disk permet de garder des configurations. Lors d'un prochain redémarrage, il n'est pas nécessaire de refaire les actions de configuration. Il peut être sauvegardé mais est très volumineux.

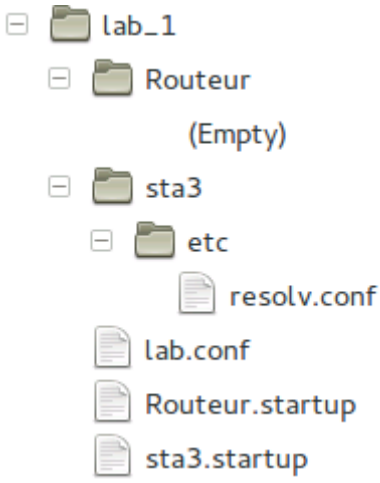
Si vous avez réalisé une modification dans un fichier sur vos VMs, cele-ci est enregistré dans le fichier vmx.disk. Si vous ajoutez ensuite un script qui modifie également ce fichier, la modification peut ne pas être prise en compte. Il faudra supprimer les fichiers .disk des VMs et relancer le lab pour que l'exécution du script soit effective.

Tout le dispositif (création des interfaces, routage, masquage d'adresse...) peut être mis en place par les scripts Netkit qui sont exécutés au lancement du lab.

Le lab complet

Voici le lab complet pour le réseau constitué avec les VMs Netkit **Routeur** et **sta3**.

L'arborescence des dossiers :



Le fichier **$HOME/lab_1/lab.conf**

```
LAB_DESCRIPTION="Premier exemple de réseau "
LAB_VERSION=1.0
LAB_AUTHOR="Techer"
```

```
LAB_EMAIL= accueil-poitiers@cned.fr
LAB_WEB=http://www.cned.fr

Routeur[1]= HubDCB
# L interface eth1 de la VM Routeur est attachee au domaine
# de collision  HubDCB
Routeur[2]=tap,10.0.0.1,10.0.0.2
# interface tap entre la VM Routeur
# et l'hôte Ubuntu Server

sta3[0]= HubDCB
# L interface eth0 de la VM sta3 est attachee au domaine
# de collision  HubDCB
```

Le fichier **$HOME/lab_1/sta3/etc/resolv.conf**

```
nameserver 208.67.222.222
nameserver 208.67.220.220
```

Le fichier **$HOME/lab_1/Routeur.startup**

```
# configuration de la carte eth1
ifconfig eth1 192.168.10.254 netmask 255.255.255.0 up

# Activer l ipmasquerade
iptables -t nat -A POSTROUTING -o eth2 -j MASQUERADE
```

Le fichier **$HOME/lab_1/sta3.startup**

```
ifconfig eth0 192.168.10.1 netmask 255.255.255.0 up
route add default gw 192.168.10.254 dev eth0
```

Communiquer avec les VMs Netkit

Vos VMs Netkit peuvent communiquer sur le réseau mais sont masqués et donc inaccessibles. Dans les ateliers suivants, vous allez mettre en place des infrastructures réseaux qui utilisent à la fois vos VMs Netkit mais également vos VMs VMware Workstation. Toutes ces VMs doivent pouvoir communiquer entre elles.

Netkit permet la création d'une interface '**tap**' entre une VM Netkit et l'hôte qui est ici Ubuntu Server. Comme vous l'avez constaté, les réseaux qui sont ensuite créés avec Netkit sont inconnus d'Ubuntu Server. En visualisant la table de routage d'Ubuntu server vous avez constaté qu'aucune route n'est indiquée pour pouvoir communiquer avec les réseaux créés sous Netkit. Seule peuvent communiquer avec l'hôte Ubuntu Sever les VMs avec lesquelles une interface '**tap**' a été crée.

Vous allez donc rajouter une **route statique** à l'hôte Ubuntu Server afin de lui indiquer comment accéder aux réseaux créés après la VM Netkit Routeur.

Voici la commande à exécuter dans le Terminal de la VM d'Ubuntu Server :

```
btssio@ubuntunetkit:~$   sudo route add –net 192.168.10.0 netmask
255.255.255.0 gw 10.0.0.2
```

La commande est à interpréter de la manière suivante :

Pour atteindre le réseau 192.168.10.0 (celui sur lequel se trouve la VM Netkit sta3), il faut envoyer les paquets 10.0.0.2 qui est l'interface de la VM Routeur.

Visualisez à nouveau la table de routage d'Ubuntu Server :

```
btssio@ubuntunetkit:~/lab_1$ sudo route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.1.254   0.0.0.0         UG    100    0        0 eth0
10.0.0.0        0.0.0.0         255.0.0.0       U     0      0        0 nk_tap_b
tssio
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
192.168.10.0    10.0.0.2        255.255.255.0   UG    0      0        0 nk_tap_b
tssio
btssio@ubuntunetkit:~/lab_1$
```

Une nouvelle route est ajoutée vers 10.0.0.2 en passant par la carte virtuelle **nk_tap_btssio**. Maintenant, à partir de l'hôte Ubuntu Server, vous pouvez communiquer avec le réseau **192.168.10.0** et donc avec **sta3** (adresse IP 192.168.10.1).

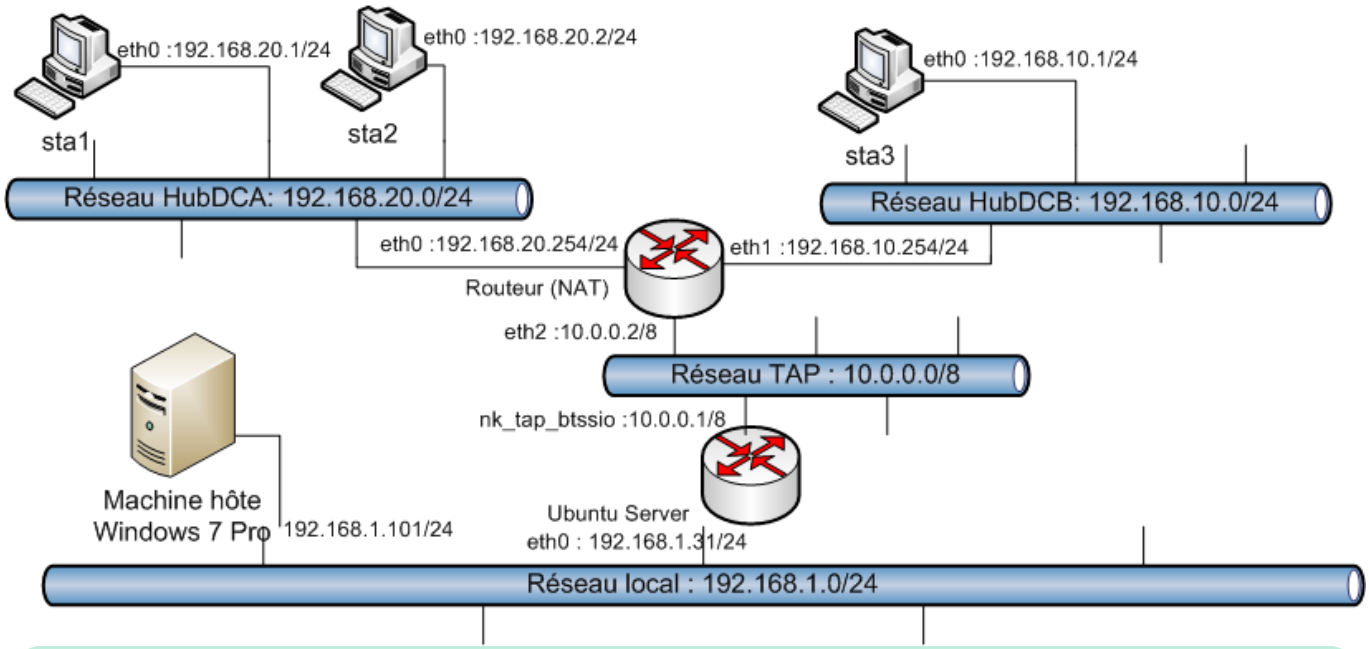
```
btssio@ubuntunetkit:~/lab_1$ ping -c 4 192.168.10.1
PING 192.168.10.1 (192.168.10.1) 56(84) bytes of data.
64 bytes from 192.168.10.1: icmp_req=1 ttl=63 time=1.34 ms
64 bytes from 192.168.10.1: icmp_req=2 ttl=63 time=0.477 ms
64 bytes from 192.168.10.1: icmp_req=3 ttl=63 time=0.450 ms
64 bytes from 192.168.10.1: icmp_req=4 ttl=63 time=0.487 ms

--- 192.168.10.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3000ms
rtt min/avg/max/mdev = 0.450/0.690/1.346/0.378 ms
btssio@ubuntunetkit:~/lab_1$
```

Cette route statique sera supprimée lors de l'arrêt du lab ou du redémarrage de la VM Ubuntu server. La faudra la recréer à ce moment là.

Représentation simplifiée de l'architecture réseau

Voici le schéma simplifié de l'infrastructure réseau présentée en début d'atelier et que vous allez terminer de réaliser.



Exercice 2 :

Complétez le lab pour rajouter le réseau 192.168.20.0/24 contenant les VMs sta1 et sta2 et permettre la communication entre les toutes les VMs Netkit de ce réseau (sta1, sta2) et l'hôte Ubuntu server.

From:
https://siocours.lycees.nouvelle-aquitaine.pro/ - **Les cours du BTS SIO**

Permanent link:
https://siocours.lycees.nouvelle-aquitaine.pro/doku.php/sisr3/r0

Last update: **2014/09/11 23:23**