

Présentation de netkit

Juin 2009 - Keepin.eu présente :

The poor man's system to experiment
computer networking

<http://wiki.netkit.org/>

- 1 Qu'est-ce que netkit ?
- 2 Terminologie
- 3 Installation de netkit
- 4 Préparer un lab (au sens netkit du terme)
- 5 Étude de cas (routage statique)
- 6 more, more... with netkit
- 7 Contributions

netkit en quelques lignes

- Émule des réseaux d'ordinateurs
- Un outil de maquettage et de simulation
- Pour comprendre le fonctionnement des protocoles
- Sans avoir à investir dans des équipements
- Logiciels open source (licence GPL)
- Utilise des logiciels libres (testé sur debian)
- Basé sur UML (User Mode Linux)

Concept général

- Plusieurs machines virtuelles sont créées sur le même hôte
- Les machines virtuelles sont reliées à des domaines de collisions virtuels et elles peuvent communiquer entre-elles
- Chaque machine virtuelle peut jouer le rôle de PC, routeur ou switch

Fonctionnalités réseau générales

- Couche physique Ethernet
- Liaison de données 802.1d, 802.1q, ppp
- Couche réseau, arp, rarp, icmp, IPv4, IPv6
- Couche transport TCP, UDP
- Couche application DNS, SMTP, FTP, HTTP, SSH, Squid...

Fonctionnalités de routage

- MPLS
- BGP (quagga)
- OSPF (quagga)
- RIP (quagga)
- Répartition de charge, multicast

Fonctionnalités de sécurité

- Transport IPsec
- IKE (openswan, racoon)
- IDS snort
- freeradius

Manipulation de paquet

- Encapsulation GRE, MPLS
- Analyse de trame (ettercap, ssldump, tcpdump...)
- Filtrage de paquets (netfilter)
- Modification de paquets (dsniff, hping, senip, tcprelay)

- 1 Qu'est-ce que netkit ?
- 2 Terminologie**
- 3 Installation de netkit
- 4 Préparer un lab (au sens netkit du terme)
- 5 Étude de cas (routage statique)
- 6 more, more... with netkit
- 7 Contributions

Émulation d'un réseau d'ordinateur

- Fonctionne sur des machines virtuelles (VM)
- Chaque nœud du réseau émulé est une VM
- Les VMs sont basées sur User Mode Linux¹ (UML)
- Plusieurs VMs peuvent être exécutées à un instant 't'
- Les VMs peuvent communiquer entre-elles

1. <http://user-mode-linux.sourceforge.net/>

Les machines virtuelles - 1

Chaque VM dispose de :

- une ou deux consoles d'administration
- un espace mémoire indépendant
- un système de fichier (un seul fichier) pour sauvegarder les modifications
- une ou plusieurs interfaces réseaux

Qu'est-ce que netkit ?

Terminologie

Installation de netkit

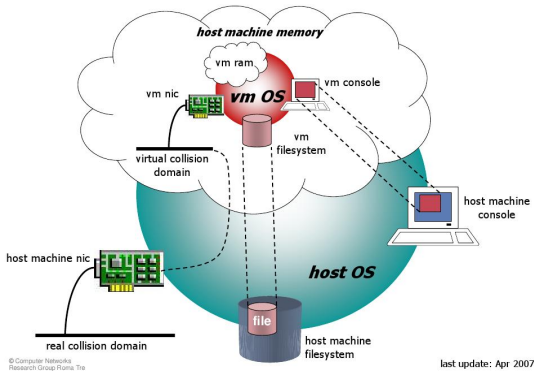
Préparer un lab (au sens netkit du terme)

Étude de cas (routage statique)

more, more... with netkit

Contributions

Émulation d'une VM dans UML



© Computer Networks
Research Group Roma Tre

last update: Apr 2007

FIGURE: Source netkit.org

keepin¹⁴

Les machines virtuelles - 2

- elles peuvent être auto-configurées (fichier de déclaration et scripts)
- configurables manuellement
 - pour les paramètres réseaux
 - pour les applications
- les modifications peuvent être sauvegardées

Les domaines de collision virtuels

Les nœuds sont raccordés sur des domaines de collision.

Un domaine de collision virtuel peut :

- être connecté à plusieurs interfaces
- chaque interface peut être connectée à un ou plusieurs domaines de collision

Qu'est-ce que netkit ?

Terminologie

Installation de netkit

Préparer un lab (au sens netkit du terme)

Étude de cas (routage statique)

more, more... with netkit

Contributions

Émulation d'un réseau d'ordinateurs avec UML

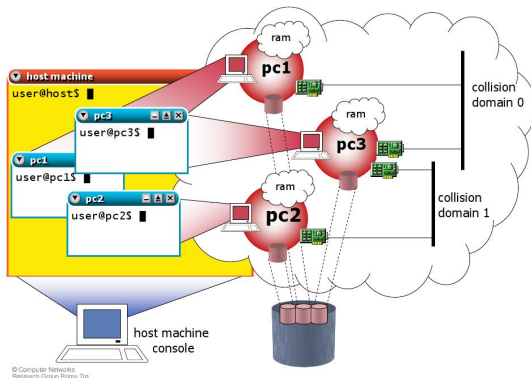


FIGURE: Source netkit.org

- 1 Qu'est-ce que netkit ?
- 2 Terminologie
- 3 Installation de netkit**
- 4 Préparer un lab (au sens netkit du terme)
- 5 Étude de cas (routage statique)
- 6 more, more... with netkit
- 7 Contributions

Installation de netkit

```
$ cd && mkdir netkit && cd netkit
```

- Téléchargement : <http://wiki.netkit.org/>
 - netkit-X.Y.tar.bz2
 - netkit-filesystem-FX.Y.tar.bz2 (>100MB)
 - netkit-kernel-KX.Y.tar.bz2

- Installer dans le même répertoire

```
$ for f in `ls`; do \  
>     tar xjf $f ;\  
> done
```

Variables environnement - exemple

```
export NETKIT_HOME=/la/ou/est/netkit
export PATH=$PATH:$NETKIT_HOME/bin
export MANPATH=$MANPATH:$NETKIT_HOME/man
```

Vérification de l'environnement

```
$ cd $NETKIT_HOME # (~/.netkit par exemple)
$ ./check_configuration.sh
```

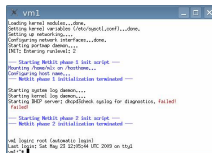
- Si tout est bien configuré vous devez avoir un message :

```
[ READY ] Congratulations!
Your Netkit setup is now complete!"
      " Enjoy Netkit!"
```

Les variables devront être chargées avant l'utilisation de netkit.
Mettre cela dans ~/.bashrc par exemple.

Premier test

- Exportez les variables environnements
- Mettez vous dans un répertoire de travail
- `vstart vm1` // lance votre première VM *vm1*
- `vlist` // pour voir les VM lancées
- `vhalt vm1` // pour arrêter la VM *vm1*



```
vm1
Loading kernel modules...done.
Setting kernel variables: /etc/sgen1.conf1...done.
Setting up network...done.
Configuring network interfaces...done.
Starting portmap daemon...
DNTC: Entering runstate: 2
--- Starting NetKit phase 1: Init script ---
Mounting /home/hic on /home...
Configuring host name...
--- NetKit phase 1 initialization terminated ---
Starting system log daemon...
Starting kernel log daemon...
Starting DHCP server: dhcpdcheck: quiet for diagnosis, failed!
failed!
--- Starting NetKit phase 2: Init script ---
--- NetKit phase 2 initialization terminated ---
vml login root (ndowd@ic login)
Last login: Sat May 22 22:05:04 UTC 2009 on ttyd
vml>
```

FIGURE: La VM *vm1*

Les commandes netkit

- netkit fournit deux groupes de commandes
 - les vcommandes, préfixées par 'v'
 - les lcommandes, préfixées par 'l'
- les vcommandes servent pour manipuler une seule VM
- les lcommandes servent à manipuler des ensembles complexes de machines virtuelles en réseau (*lab*).
- si vous avez une seule VM, utilisez les vcommandes
- si vous créez un laboratoire (*lab*), utilisez les lcommandes

Les vcommandes

- utilisées pour manipuler des VMs isolées
 - vstart, démarrer une VM
 - vlist, afficher les VMs actives
 - vconfig, affecter une interface à la volée à une machine virtuelle.(vconfig -eth0=dc1 pc1 - affecte eth0 à la VM pc1, dans le domaine de collision dc1)
 - vhalt, arrête une VM
 - vcrash, déclenche un crash virtuel
 - vclean, nettoie les processus et configurations

Les lcommandes

- Utilisées sur les scénarios complexes de plusieurs VMs (*lab*) dans la terminologie de netkit.
- Un laboratoire est une maquette comprenant plusieurs VMs
 - lstart, lancer un laboratoire netkit
 - lhalt, arrêter le laboratoire
 - lcrash, déclencher un crash de toutes les VM
 - lclean, supprimer les fichiers temporaires
 - linfo, information sur le laboratoire
 - ltest, vérification du bon fonctionnement du laboratoire

Les commandes en général

Consulter les pages du manuel

très complètes, claires et détaillées

man vstart, man vconfig, man lclean ...

Échange de données entre VMs

Deux scénarios possibles :

- échange de fichiers via l'hôte
- accès à internet (téléchargement)

Échange de données entre VMs via l'hôte 2

- Chaque VM à un répertoire /hosthome
- /hosthome est mappé à la racine du compte utilisateur qui a lancé la VM

Exemple :

- si le compte qui a lancé la VM est *foobar*
- le /hosthome de la vm correspondra à
- /home/foobar sur l'hôte

Échange de données avec internet

- utile pour ajouter des paquets
- netkit peut émuler un tunnel (interface tap)

Exemple :

- `vstart vm1 -eth0=tap,10.0.0.1,10.0.0.2`
- lance une VM vm1 avec une interface eth0
- la VM aura une adresse 10.0.0.2
- l'interface tap sur l'hôte aura l'adresse 10.0.0.1
- il faut activer l'ipmasquerade sur l'hôte
- `sudo iptables -t nat -A POSTROUTING -j MASQUERADE`
- voir `man vstart`

Activer une interface tap

Vous devrez avoir le mot de passe root

```
sudo iptables -t nat -A POSTROUTING -j MASQUERADE
```

```
vstart vmname --ethi=tap,adresse_hôte,adresse_VM
```

```
vstart vm1 --eth0=tap,10.0.0.1,10.0.0.2
```

les adresses doivent être sur le même [sous]réseau

Interface tap sous netkit

- Avoir le mot de passe root (voir man vstart)
- Activer l'ipmasquerade sur l'hôte
- Configurer /etc/resolv.conf (cf

<http://www.opendns.com/>)

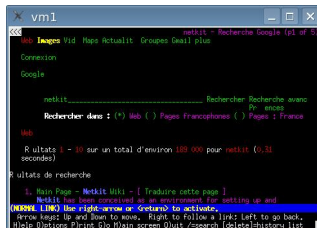


FIGURE: La VM vm1 avec lynx

L'interface tap sur l'hôte et la VM en action

```
nk_tap_mlx Link encap:Ethernet HWaddr 00:ff:12:b5:2f:0a  
      inet adr:10.0.0.1 Bcast:10.255.255.255 Masque:255.0.0.0
```

```
Starting system log daemon....  
Starting kernel log daemon....  
Starting DHCP server: dhcpd3check syslog for diagnostics. failed!  
failed!
```

```
--- Starting Netkit phase 2 init script ---  
--- Netkit phase 2 initialization terminated ---
```

```
vm login: root (automatic login)  
Last login: Thu May 21 14:41:48 UTC 2009 on tty1  
vm:~# ping -c 5 www.keepin.eu  
PING keepin.eu (86.64.63.109) 56(84) bytes of data:  
64 bytes from ns2.beaupeyrat.com (86.64.63.109): icmp_seq=1 ttl=55 time=69.3 ms  
64 bytes from ns2.beaupeyrat.com (86.64.63.109): icmp_seq=2 ttl=55 time=72.6 ms  
64 bytes from ns2.beaupeyrat.com (86.64.63.109): icmp_seq=3 ttl=55 time=70.5 ms  
64 bytes from ns2.beaupeyrat.com (86.64.63.109): icmp_seq=4 ttl=55 time=73.5 ms  
64 bytes from ns2.beaupeyrat.com (86.64.63.109): icmp_seq=5 ttl=55 time=70.7 ms
```

```
--- keepin.eu ping statistics ---  
5 packets transmitted, 5 received, 0% packet loss, time 4046ms  
rtt min/avg/max/mdev = 69.317/71.344/73.509/1.530 ms  
vm:~#
```

- 1 Qu'est-ce que netkit ?
- 2 Terminologie
- 3 Installation de netkit
- 4 Préparer un lab (au sens netkit du terme)**
- 5 Étude de cas (routage statique)
- 6 more, more... with netkit
- 7 Contributions

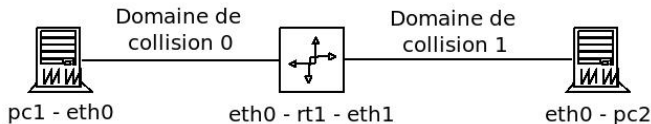
Création d'un lab

Un '*lab*' est un ensemble de machines pré-configurées et définissant un réseau.

Deux solutions :

- au travers d'un script qui appelle '*vstart*' pour chaque machine virtuelle (non recommandé).
- au travers d'un lab qui utilise les 'l-commandes' '*lstart*'

Exemple de scénario



- rt1 est un routeur
- rt1 : eth0 fait partie du domaine de collision 0
- rt1 : eth1 fait partie du domaine de collision 1
- pc1 est dans le domaine de collision 0
- pc2 est dans le domaine de collision 1

Option avec vstart - utilisation d'un script simple

Les commandes sont réalisées lors de l'initialisation des VMs

```
vstart rt1 --eth0=0 --eth1=1 --exec=unscript.sh
vstart pc1 --eth0=0 --exec=unscript.sh
vstart pc2 --eth0=1 --exec=unscript.sh
if [ `id -u` == "0" ]; then
    case "$HOSTNAME" in
        rt1)
            ifconfig eth0 10.0.0.1 up
            ifconfig eth1 11.0.0.1 up;;
        pc1)
            ifconfig eth0 10.0.0.2 up;;
        pc2)
            ifconfig eth0 11.0.0.2 up;;
    esac
fi
```

L'option `-exec`

- permet de faire exécuter un script ou un programme au démarrage de la VM
- le script ou le programme est exécuté à l'intérieur de la VM
- le script doit être exécutable et dans un chemin accessible à partir de la VM

Exemple :

- `vstart pc1 -eth0=1 -exec="/bin/lis" -verbose`
- `vstart pc1 -eth0=1 -exec="/hosthome/unscript.sh"`

Option avec lstart

Un '*lab*' au sens netkit du terme est une arborescence contenant :

- un fichier '*lab.conf*' qui décrit la topologie du réseau
- un répertoire qui contient la configuration de chaque machine
- 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*

Option avec lstart

Les fichiers optionnels :

- '*lab.dep*', décrit éventuellement un ordre de lancement des VMs
- un répertoire *_test* contenant des scripts afin de tester le lab

Le fichier lab.conf

- Le fichier décrit :
 - 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 attachée
 - si `arg` est une chaîne, alors c'est le nom d'une option de `vstart` et `valeur` est un argument de cette option.

Le fichier lab.conf - exemple de déclaration

```
pc1[0]=1
# L'interface eth0 de la VM pc1 est attachée au domaine
# de collision 1
pc2[1]=2
pc2[mem]=256
# L'interface eth1 de la VM pc2 est attachée au domaine
# de collision 2
# La VM pc2 dispose de 256 MB de mémoire virtuelle.
# La liste des paramètres est dans man vstart
```

Le fichier lab.conf - paramètres optionnels

```
LAB_DESCRIPTION="Exemple sur le routage statique"  
LAB_VERSION=1.0  
LAB_AUTHOR="keepin"  
LAB_EMAIL=contact@keepin.eu  
LAB_WEB=http://www.keepin.eu
```

Les paramètres informatifs sont affichés dans la console de la VM au démarrage du lab.

Le fichier lab.conf - paramètres optionnels

```
machines='pc1, pc2, rtr...'
```

Donne la liste des machines à démarrer.

Par défaut, il y a une VM pour chaque sous-répertoire existant dans le répertoire où est lancé le lab.

Les sous-répertoires du lab

- 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 / de la VM
- par exemple *vm/foo/file* est copié dans */foo/file* de la VM
- un SGF *vmx.disk* est créé pour chaque VM *vmx* lors du premier lancement de la VM
- il permet de garder des configurations et peut être sauvegardé

les scripts `vm.startup` et `vm.shutdown`

- indiquent les commandes à réaliser au démarrage et à l'arrêt d'une VM pour la VM `vm`
- ils sont exécutés à l'intérieur de la VM
- `shared.startup` et `shared.shutdown` concernent toutes les VMs du lab
- `shared.startup` et `shared.shutdown` sont exécutés avant `vmx.startup` et `vmx.shutdown`

Exemple de fichier vm.startup

Sert par exemple à configurer le réseau ou lancer un service

```
ifconfig eth0 1.1.1.1 up  
/etc/init.d/sshd start
```

le fichier lab.dep

- les VMs démarrent les unes après les autres
- elles peuvent démarrer en mode parallèle (lstart -p)
- on peut imposer un ordre de lancement : pc1 et pc2 et après rtr
- on crée un fichier lab.dep (syntaxe similaire à un Makefile)
- un fichier lab.dep active par défaut le mode de lancement parallèle

Exemple de fichier lab.dep

```
#Exemple lab.deb
# la VM rtr n'est activée qu'une fois
# pc1 et pc2 lancés.
rtr:    pc1  pc2
```

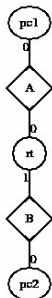
Les lcommandes

- *lcommande -d lab_directory [machine...]*
- sinon, se mettre dans le répertoire et entrer la lcommande
- *lstart*, lancer le lab
- *lhalt*, arrêter les VMs
- *lcrash*, stopper brutalement les VMs
- on peut passer en paramètre les machines concernées

Supprimer les fichiers temporaires

- Des fichiers temporaires sont créés dans le répertoire du lab et dans le répertoire courant
- Iclean permet de supprimer les fichiers temporaires
- Iclean supprime aussi les fichiers '.disk'
- Ne pas utiliser Iclean si vous voulez réutiliser les fichiers '.disk'

La commande linfo



- linfo imprime les informations concernant un lab
- linfo -m out.eps, génère un graphique de la topologie
- nécessite graphviz <http://graphviz.org>

- 1 Qu'est-ce que netkit ?
- 2 Terminologie
- 3 Installation de netkit
- 4 Préparer un lab (au sens netkit du terme)
- 5 Étude de cas (routage statique)**
- 6 more, more... with netkit
- 7 Contributions

Scénario

On souhaite créer :

- deux domaines de collision : 'A' et 'B'
- un routeur rt, eth0 sur 'A' et eth1 'B'
- un pc, pc1 sur 'A'
- un pc, pc2 sur 'B'

création de l'environnement

- # On crée l'environnement de travail pour le lab
- `mkdir firstLAB && cd firstLAB`
- # On crée les répertoires pour les VM (nœuds)
- `mkdir -p pc1 pc2 rt`

Le fichier lab.conf 1/2

```
# On a 2 domaines de collisions, A et B
# rt est le routeur
LAB_DESCRIPTION="Exemple sur le routage statique"
LAB_VERSION=1.0
LAB_AUTHOR="keepin"
LAB_EMAIL=contact@keepin.eu
LAB_WEB=http://www.keepin.eu
```

Le fichier lab.conf 2/2

```
# interface eth0 de rt sur A
rt[0]="A"
# interface eth1 de rt sur B
rt[1]="B"
# interface eth0 de pc2 sur B
pc2[0]="B"
# interface eth0 de pc1 sur A
pc1[0]="A"
```

Le fichier pc1.startup

- On va s'en servir pour configurer la VM

```
ifconfig eth0 10.0.0.1 netmask 255.0.0.0 \  
    broadcast 10.255.255.255 up  
route add default gw 10.0.0.254 dev eth0
```

Le fichier pc2.startup

- idem.

```
ifconfig eth0 11.0.0.1 netmask 255.0.0.0 \  
        broadcast 10.255.255.255 up  
route add default gw 11.0.0.254 dev eth0
```


Le fichier rt.startup

- Même opération

```
ifconfig eth0 10.0.0.254 netmask 255.0.0.0 \  
        broadcast 10.255.255.255 up  
ifconfig eth1 11.0.0.254 netmask 255.0.0.0 \  
        broadcast 11.255.255.255 up
```

Qu'est-ce que netkit ?
Terminologie
Installation de netkit
Préparer un lab (au sens netkit du terme)
Étude de cas (routage statique)
more, more... with netkit
Contributions

Le lab en fonction

```
Session Édition Affichage Signets Configuration Aide
Terminal

X pc1
pc1 login: root (automatic login)
Last login: Fri May 22 12:55:11 UTC 2009 on tty1
pc1~# ifconfig
eth0: Link encap:Ethernet HWaddr 6a:59:96:37:0c:07
       inet addr:10.0.0.1 Bcast:10.0.0.255 Mask:255.0.0.0
       inet6 addr: fe80::d5b:109:1f33::c0/64 Scope:link
       UP BROADCAST RUNNING SLIPNET MTU:1500 NetScout
       RX packets:0 errors:0 dropped:0 overruns:0 Frame:0
       TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 transmit:1000
       RX bytes:284 (304.0 B) TX bytes:488 (488.0 B)
       Interrupt:5

lo: Link encap:Local Loopback
       inet addr:127.0.0.1 Mask:255.0.0.0
       inet6 addr: ::1::1 Scope:host
       UP LOOPBACK RUNNING MTU:65536 NetScout
       RX packets:0 errors:0 dropped:0 overruns:0 Frame:0
       TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 transmit:0
       RX bytes:200 (200.0 B) TX bytes:100 (100.0 B)

pc1~#

X pc2
pc2 login: root (automatic login)
Last login: Fri May 22 12:59:34 UTC 2009 on tty1
pc2~# ping -c 3 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=63 time=0.459 ms
 64 bytes from 10.0.0.1: icmp_seq=2 ttl=63 time=0.420 ms
 64 bytes from 10.0.0.1: icmp_seq=3 ttl=63 time=0.420 ms

--- 10.0.0.1 ping statistics ---
 3 packets transmitted, 3 received, 0% packet loss, time 2018ms
rtt min/avg/max/mdev = 0.439/0.435/0.430/0.197 ms
pc2~#

X rt
Lab directory (net): /media/sdb7/home/hiv/download/netkit/NetTESTS/firstLab
Version: 1.0
Author: keepin
Email: contact@keepin.eu
Web: http://www.keepin.eu
Description:
Compte sur le routage statique

--- Netkit phase 2 initialization terminated ---

rt login: root (automatic login)
Last login: Fri May 22 12:59:56 UTC 2009 on tty1
rt~# route -n
NetScout IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
10.0.0.0 0.0.0.0 255.0.0.0 U 0 0 0 eth0
11.0.0.0 0.0.0.0 255.0.0.0 U 0 0 0 eth1
rt~#
```

On distingue bien les 3 VMs.

Un automate de test

Comment tester le lab ?

- utiliser la commande `ltest`
- les résultats sont dans `_ltest/results/vmx.default`
- on peut créer des scripts `vmx.test`
- le résultat sera dans `_ltest/results/vmx.user`

Exemple de script pc1.test

```
#!/bin/sh  
# On teste le bon fonctionnement du routage  
# entre pc1 et pc2  
ping -c3 11.0.0.2
```

Extrait du résultat de pc1.default

```
[ROUTING TABLE]
0.0.0.0          10.0.0.254  [...]
10.0.0.0         0.0.0.0    [...]

[LISTENING PORTS]
tcp      0      0 0.0.0.0:111  [...]
udp      0      0 0.0.0.0:111  [...]

[PROCESSES]
  0 /bin/bash /etc/init.d/netkit-test-phase
  0 /bin/bash /etc/init.d/netkit-test-phase
  0 /bin/bash /etc/rc2.d/S99netkit-phase2 start
[... ]
  0 sleep 1
  0 sort
  1 /sbin/portmap
```

Exemple de résultat pc1.user

```
PING 11.0.0.1 (11.0.0.1) 56(84) bytes of data.  
64 bytes from 11.0.0.1: icmp\seq=1 ttl=63 time=22.0 ms  
64 bytes from 11.0.0.1: icmp\seq=2 ttl=63 time=0.586 ms  
64 bytes from 11.0.0.1: icmp\seq=3 ttl=63 time=0.691 ms  
  
--- 11.0.0.1 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2051ms  
rtt min/avg/max/mdev = 0.586/7.773/22.043/10.090 ms
```

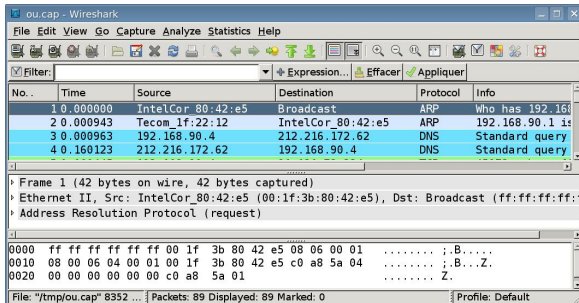
- 1 Qu'est-ce que netkit ?
- 2 Terminologie
- 3 Installation de netkit
- 4 Préparer un lab (au sens netkit du terme)
- 5 Étude de cas (routage statique)
- 6 more, more... with netkit**
- 7 Contributions

Les sauvegardes

- Les modifications apportées aux VMs sont dans des fichiers *vmx.disk*
- Copier ces fichiers sur l'unité de sauvegarde
- Les fichiers font plus d'1GO
- Pour les 'labs', sauvegarder l'arborescence du 'lab'

Capture et analyse de trame

- Utiliser tcpdump ou tshark et l'option -w pour stocker le dump dans un fichier *.cap*
- Ouvrir sur la machine hôte le fichier *.cap* avec wireshark



Capture et analyse de trame graphique dans un pipe

- mkfifo ne semble pas fonctionner entre une VM et l'hôte
- sur l'hôte lancer wireshark pour récupérer la capture :
- `wireshark -k -i<(tail -f VMi.out)&`
- se mettre dans `/hosthome` ou `/hostlab` de chaque VM pour laquelle on veut faire une capture
- lancer la capture `tcpdump -U -n -i eth0 -w VMi.out`

Autre option : utiliser le patch de Julien C.

<http://kartoch.msi.unilim.fr>

Qu'est-ce que netkit ?
Terminologie
Installation de netkit
Préparer un lab (au sens netkit du terme)
Étude de cas (routage statique)
more, more... with netkit
Contributions

Capture graphique simultanée sur 2 VMs

The screenshot displays a Netkit virtual machine environment. The main window is titled "/dev/td/63: Capturing - Wireshark <2>". It shows a packet capture interface with a table of captured packets. The table has columns for No., Time, Source, Destination, Protocol, and Info. The first few packets are:

No.	Time	Source	Destination	Protocol	Info
217	148.447791	10.0.0.1	10.0.0.2	SSHv2	Encrypted request pack
218	148.447890	10.0.0.2	10.0.0.1	TCP	ssh > 42262 [ACK] Seq=
219	148.936297	10.0.0.1	10.0.0.2	SSHv2	Encrypted request pack
220	148.936577	10.0.0.2	10.0.0.1	TCP	ssh > 42262 [ACK] Seq=
221	153.230562	4e:c1:77:72:78:94	ae:c9:f0:97:34:65	ARP	Who has 10.0.0.1? Tel
222	153.230589	ae:c9:f0:97:34:65	4e:c1:77:72:78:94	ARP	10.0.0.1 is at ae:c9:f

Below the table, there is a section for "Frame 219 (130 bytes on wire, 96 bytes captured)". It details the Ethernet II, Internet Protocol, Transmission Control Protocol, and SSH Protocol fields.

At the bottom, there is a terminal window titled "mlx@mr:/home/mlx/download/netkit/myLABS - Terminal 0.4 - Konsole". It shows a session where the user attempts to connect to 10.0.0.2 via telnet and ssh. The telnet connection is refused, and the ssh connection is also refused, with a warning about the RSA key fingerprint.

On the right side of the terminal window, there is a small window titled "X b" showing a command prompt with the following commands and output:

```

$ ssh root@10.0.0.2
Last login: Mon Jan 1 12:52:12 2009
root@10.0.0.2:~#

```

Live CD/DVD/USB

- Vous pouvez utiliser un DVD Live conçu spécialement pour l'étude des réseaux. *Netkit4TIC*
- Il y a également un CD installable sur disque ou clé USB
- Voir : <http://wiki.netkit.org>

switch ou hub ?

- sur netkit les interfaces sont attachées à des domaines de collision
- c'est plus simple pour les tests et captures de trames
- est-il possible de commuter ?
- est-il possible de tester des maquettes 802.1q ou 802.1d ?

Oui, voyons comment.

La commutation 1

- dans le répertoire bin de netkit il faut modifier le fichier *script_utils* qui est en mode hub par défaut

```
# il faut modifier :  
$ grep -En '\-hub' script_utils-hub  
266 (...) -hub -unix $1 </dev/null 2>&1"  
300 (...) -hub -unix $1 </dev/null 2>&1"  
# par  
        -unix $1 </dev/null 2>&1"  
        -unix $1 </dev/null 2>&1"
```

La commutation 2

```
$ cd netkit /bin
$ cp script_utils script_utils-hub
$ mv script_utils script_utils-switch
$ sed -i -e s/'-hub'//g script_utils-switch
# il suffira désormais d'utiliser
ln -s script_utils-hub script_utils
# ou alors
ln -s script_utils-switch script_utils
```

La commutation 3 - pour tester

```
# Mettez vous dans le mode hub par exemple
# Lancez 3 vm
for i in a, b, c ; do vstart $i --eth0=1; done
# Configurez les VM a b c sur un même réseau IP
# Lancez tcpdump sur a et b par exemple
# Générez du trafic et analysez le résultat
# Passez dans le mode switch et refaites le test
```


Le mode hub

```

X a
a:~$ tcpdump -i eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
15:10:52.391381 arp who-has 10.0.0.1 tell 10.0.0.3
15:10:52.391919 arp reply 10.0.0.1 is-at a2c394c0379a:65
15:10:52.391473 IP 10.0.0.3 > 10.0.0.1: ICMP echo request, id 4354, seq 1, length 64
15:10:52.391513 IP 10.0.0.1 > 10.0.0.3: ICMP echo reply, id 4354, seq 1, length 64
15:10:53.409824 IP 10.0.0.3 > 10.0.0.1: ICMP echo request, id 4354, seq 2, length 64
15:10:53.409852 IP 10.0.0.1 > 10.0.0.3: ICMP echo reply, id 4354, seq 2, length 64
15:10:53.409899 IP 10.0.0.3 > 10.0.0.1: ICMP echo request, id 4354, seq 3, length 64
15:10:54.409899 IP 10.0.0.3 > 10.0.0.1: ICMP echo request, id 4354, seq 3, length 64
15:10:54.409947 IP 10.0.0.1 > 10.0.0.3: ICMP echo reply, id 4354, seq 3, length 64
15:10:57.394417 arp who-has 10.0.0.3 tell 10.0.0.1
15:10:57.394936 arp reply 10.0.0.3 is-at a2c394c0379a:34
^

X b
b:~$ tcpdump -i eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
15:10:52.134370 arp who-has 10.0.0.1 tell 10.0.0.3
15:10:52.134397 arp reply 10.0.0.1 is-at a2c394c0379a:65
15:10:52.134666 IP 10.0.0.3 > 10.0.0.1: ICMP echo request, id 4354, seq 1, length 64
15:10:52.134685 IP 10.0.0.1 > 10.0.0.3: ICMP echo reply, id 4354, seq 1, length 64
15:10:53.152684 IP 10.0.0.3 > 10.0.0.1: ICMP echo request, id 4354, seq 2, length 64
15:10:53.152833 IP 10.0.0.1 > 10.0.0.3: ICMP echo reply, id 4354, seq 2, length 64
15:10:54.152941 IP 10.0.0.3 > 10.0.0.1: ICMP echo request, id 4354, seq 3, length 64
15:10:54.153056 IP 10.0.0.1 > 10.0.0.3: ICMP echo reply, id 4354, seq 3, length 64
15:10:57.127359 arp who-has 10.0.0.3 tell 10.0.0.1
15:10:57.127439 arp reply 10.0.0.3 is-at a2c394c0379a:34
^

X c
c:~$ ifconfig eth0 10.0.0.3
c:~$ ping -c 3 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=0.93 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.255 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=1.07 ms

--- 10.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 202ms
rtt min/avg/max/mdev = 0.255/3.420/6.930/3.510 ms
c:~$

```

Les paquets inondent tous les nœuds.

Le mode switch

```
X a
a2~* tcpdump -n -i eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
15:14:19.141990 arp who-has 10.0.0.1 tell 10.0.0.3
15:14:19.142336 arp reply 10.0.0.1 is-at a2:c5:f0:37:54:e5
15:14:19.142625 IP 10.0.0.3 > 10.0.0.1: ICMP echo request, id 4096, seq 1, length 64
15:14:19.142672 IP 10.0.0.1 > 10.0.0.3: ICMP echo reply, id 4096, seq 1, length 64
15:14:20.161128 IP 10.0.0.3 > 10.0.0.1: ICMP echo request, id 4096, seq 2, length 64
15:14:20.161169 IP 10.0.0.1 > 10.0.0.3: ICMP echo reply, id 4096, seq 2, length 64
15:14:21.161076 IP 10.0.0.3 > 10.0.0.1: ICMP echo request, id 4096, seq 3, length 64
15:14:21.161113 IP 10.0.0.1 > 10.0.0.3: ICMP echo reply, id 4096, seq 3, length 64
15:14:24.123712 arp who-has 10.0.0.5 tell 10.0.0.1
15:14:24.124106 arp reply 10.0.0.5 is-at a2:33:4c:c3:7e:54
[]

X b
b2~* tcpdump -n -i eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
15:14:19.001062 arp who-has 10.0.0.1 tell 10.0.0.3
[]

X c
-- Netkit phase 1 initialization terminated --
Starting system log daemon,...
Starting kernel log daemon,...
Starting SNMP server: dhcpdCheck syslog for diagnostics, failed!
failed!
-- Starting Netkit phase 2 init script --
-- Netkit phase 2 initialization terminated --

c login: root (automatic login)
Last login: Fri May 29 15:15:31 UTC 2009 on ttyd
c~* ifconfig eth0 10.0.0.3
c~* ping -c 3 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=5.16 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.339 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.300 ms

-- 10.0.0.1 ping statistics --
3 packets transmitted: 3 received, 0% packet loss, time 2023ms
rtt min/avg/max/mdev = 0.300/1.932/5.162/2.284 ms
c~* []
```

On ne voit passer qu'un paquet arp sur la machine b.

netkit, VLAN et qualité de service

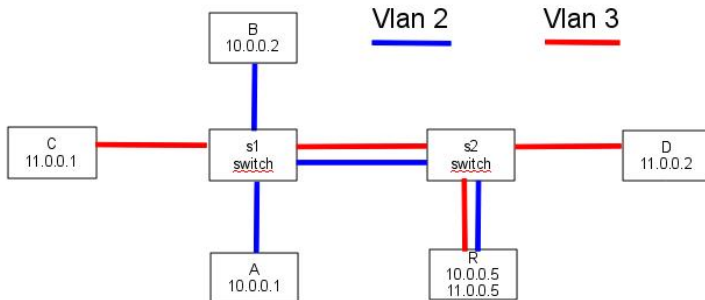
C'est possible en utilisant ce que nous propose GNU/Linux :

- le module 8021q
- iproute2
- le paquet bridge-utils

Netkit, la commutation, scénario de trunk

- On a deux clients, a et b sur le VLAN 2
- On a deux clients, c et d, sur le VLAN 3
- Les VLANs sont répartis sur deux switches
- On a un serveur qui pourrait servir de routeur si besoin pour faire du routage inter-VLAN
- On va utiliser :
 - le module 8021q pour créer les VLANs
 - un pont pour simuler les VLANs et le trunk

Topologie



Le fichier lab.conf

```
a[0]="A"
a[mem]=64
```

```
b[0]="B"
b[mem]=64
```

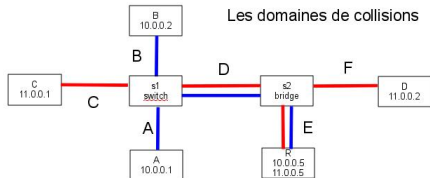
```
c[0]="C"
c[mem]=64
```

```
d[0]="F"
d[mem]=64
```

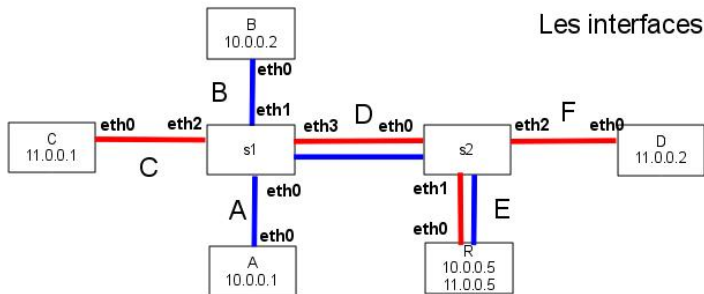
```
r[0]="E"
r[mem]=64
```

```
s1[0]="A"
s1[1]="B"
s1[2]="C"
s1[3]="D"
s1[mem]=64
```

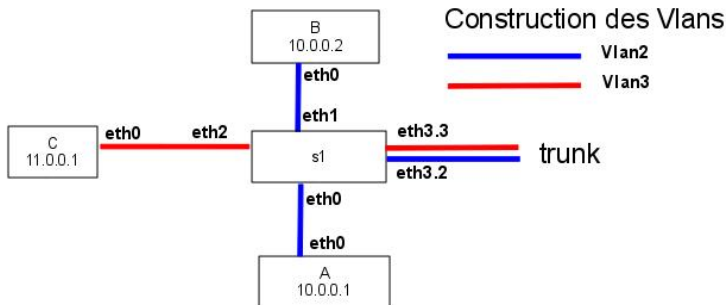
```
s2[0]="D"
s2[1]="E"
s2[2]="F"
s2[mem]=64
```



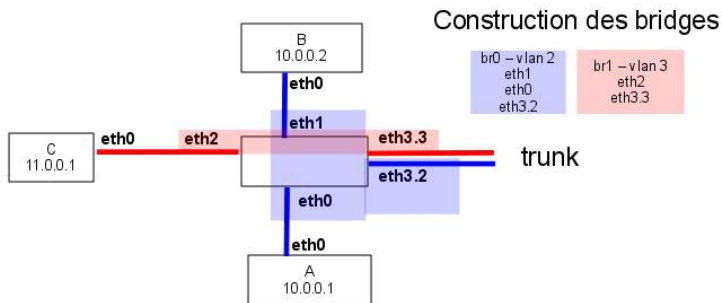
Les interfaces réseaux



Construction des Vlan (avec vconfig)



Construction des bridges (avec brctl)



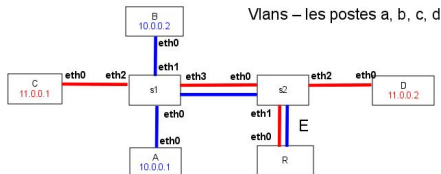
Les postes a, b, c, d

```
# a.startup
ifconfig eth0 up
ifconfig eth0 10.0.0.1
```

```
# b.startup
ifconfig eth0 up
ifconfig eth0 10.0.0.2
```

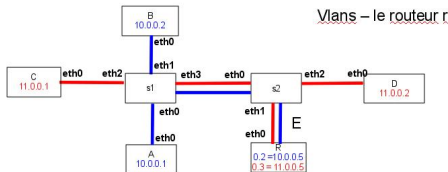
```
# c.startup
ifconfig eth0 up
ifconfig eth0 11.0.0.1
```

```
# d.startup
ifconfig eth0 up
ifconfig eth0 11.0.0.2
```



Le routeur

```
# r.startup
ifconfig eth0 up
vconfig add eth0 2
ifconfig eth0.2 up
vconfig add eth0 3
ifconfig eth0.3 up
ifconfig eth0.2 10.0.0.5
ifconfig eth0.3 11.0.0.5
```



Le switch s1

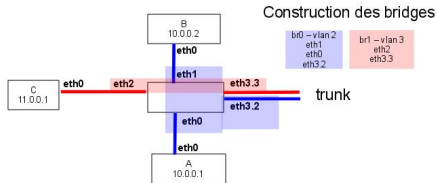
```
# s1.startup
ifconfig eth0 up
ifconfig eth1 up
ifconfig eth2 up
ifconfig eth3 up

vconfig add eth3 2
ifconfig eth3.2 up

vconfig add eth3 3
ifconfig eth3.3 up

brctl addbr br0
brctl addif br0 eth0
brctl addif br0 eth1
brctl addif br0 eth3.2
ifconfig br0 up

brctl addbr br1
brctl addif br1 eth2
brctl addif br1 eth3.3
ifconfig br1 up
```



Le switch s2

```
# s2.startup
ifconfig eth0 up
vconfig add eth0 2
vconfig add eth0 3
ifconfig eth0.2 up
ifconfig eth0.3 up

ifconfig eth1 up
vconfig add eth1 2
vconfig add eth1 3
ifconfig eth1.2 up
ifconfig eth1.3 up

ifconfig eth2 up

brctl addbr br0
brctl addif br0 eth0.2
brctl addif br0 eth1.2
ifconfig br0 up

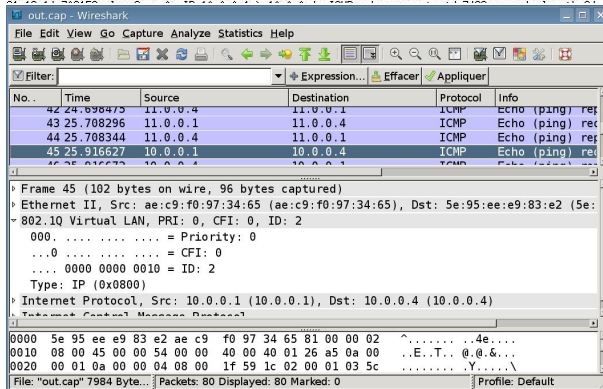
brctl addbr br1
brctl addif br1 eth0.3
brctl addif br1 eth1.3
brctl addif br1 eth2
ifconfig br1 up
```

Activation des VMs



Captures de trames sur r

```
21:16:13.328850 vlan 3, p 0, IP 11.0.0.4 > 11.0.0.1: ICMP echo reply, id 7426, seq 5, length 64
21:16:13.698142 vlan 2, p 0, IP 10.0.0.1 > 10.0.0.4: ICMP echo request, id 7426, seq 3, length 64
21:16:13.698181 vlan 2, p 0, IP 10.0.0.4 > 10.0.0.1: ICMP echo reply, id 7426, seq 3, length 64
21:16:14.338701 vlan 3, p 0, IP 11.0.0.1 > 11.0.0.4: ICMP echo request, id 7426, seq 6, length 64
21:16:14.338742 vlan 3, p 0, IP 11.0.0.4 > 11.0.0.1: ICMP echo reply, id 7426, seq 6, length 64
```



L'agrégat de lien

Le module *bond* permet d'agréger des liens. Avec deux liens à 100 Mbit/s on a en théorie 200 Mbit/s.

ifenslave n'est pas sur netkit, il faudra l'installer.

```
r:~# modeprobe bondig  
r:~# ifconfig bond0 10.0.0.1 up  
r:~# ifenslave bond0 eth2  
r:~# ifenslave bond0 eth3
```


L'agrégat de lien

```
r:~# ifconfig
r:~# ifconfig bond0 10.0.0.1 up
r:~# ifenslave bond0 eth2
r:~# ifenslave bond0 eth3
r:~# ifconfig bond0
bond0      Link encap:Ethernet  HWaddr f6:87:77:5c:49:3c
            inet addr:10.0.0.1  Bcast:10.255.255.255  Mask:255.0.0.0
            inet6 addr: fe80::f487:77ff:fe5c:493c/64 Scope:Link
            UP BROADCAST RUNNING MASTER MULTICAST  MTU:1500  Metric:1
            RX packets:7 errors:0 dropped:0 overruns:0 frame:0
            TX packets:7 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:456 (456.0 B)  TX bytes:554 (554.0 B)
```

L'agrégat de lien

```
r:~# cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.2.5 (March 21, 2008)
```

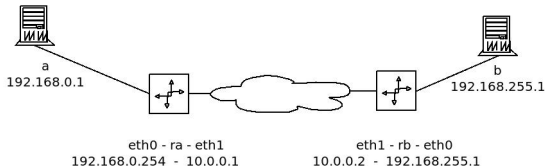
```
Bonding Mode: load balancing (round-robin)
MII Status: up
MII Polling Interval (ms): 0
Up Delay (ms): 0
Down Delay (ms): 0
```

```
Slave Interface: eth2
MII Status: up
Link Failure Count: 0
Permanent HW addr: f6:87:77:5c:49:3c
```

```
Slave Interface: eth3
MII Status: up
Link Failure Count: 0
Permanent HW addr: 16:3c:8c:b1:d5:4e
```

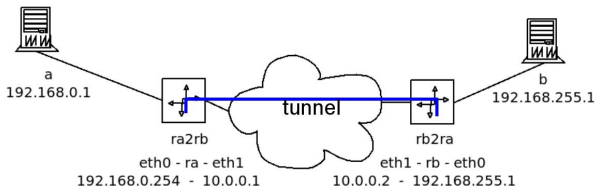
L'encapsulation GRE - maquette

Les interfaces wan sont sur le même réseau ip car il n'y a pas de routeurs sur la maquette entre les réseaux distants, mais ça ne change rien au principe.



L'encapsulation GRE - tunnel

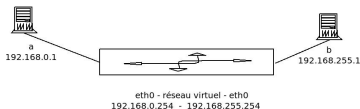
Liaison ppp entre les interfaces logiques ra2rb et rb2ra



Sur ra, les paquets de eth0 sont envoyés/reçus sur ra2rb

Sur rb, les paquets de eth0 sont envoyés/reçus sur rb2ra

L'encapsulation GRE - liaison P2P sur rb



ifconfig et ip link (extraits)

```
#ifconfig rb2ra
rb2ra      Link encap:UNSPEC
           inet addr:192.168.255.254  P-t-P:192.168.255.254
           UP POINTOPOINT RUNNING NOARP  MTU:1476  Metric:1

# ip link show
5: gre0: <NOARP> mtu 1476 qdisc noop
   link/gre 0.0.0.0 brd 0.0.0.0
8: rb2ra@NONE: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 1476
   link/gre 10.0.0.2 peer 10.0.0.1
```

Tunnel GRE - lab.conf

```
# Tunnel GRE
# Les adresses WAN sont sur le même réseau
# car on a pas de routeur intermédiaire
# mais le principe ne change pas

a[0]="A"
a[mem]=64

b[0]="B"
b[mem]=64

# Pour réaliser un tunnel gre entre a/ra et b/rb
ra[0]="A"
ra[1]="W"
r[mem]=64

rb[0]="B"
rb[1]="W"
rb[mem]=64
```

Tunnel GRE - lab.conf

```
# Sur a
ifconfig eth0 up
ifconfig eth0 192.168.0.1
route add default gw 192.168.0.254

# Sur b
ifconfig eth0 up
ifconfig eth0 192.168.255.1
route add default gw 192.168.255.254
```

Tunnel GRE - lab.conf

```
# sur ra
ifconfig eth0 up
ifconfig eth0 192.168.0.254
ifconfig eth1 10.0.0.1 up
modprobe ip_gre
ip tunnel add ra2rb mode gre remote 10.0.0.2 \
    local 10.0.0.1 ttl 255
ip link set ra2rb up
ip addr add 192.168.0.254 dev ra2rb
ip route add 192.168.255.0/24 dev ra2rb
```


Tunnel GRE - lab.conf

```
# Sur rb
ifconfig eth0 up
ifconfig eth0 192.168.255.254
ifconfig eth1 10.0.0.2 up
modprobe ip_gre
ip tunnel add rb2ra mode gre remote 10.0.0.1 \
    local 10.0.0.2 ttl 255
ip link set rb2ra up
ip addr add 192.168.255.254 dev rb2ra
ip route add 192.168.0.0/24 dev rb2ra
```

Tunnel GRE - capture de trame

The screenshot shows the Wireshark interface with a capture of network traffic. The packet list shows several ARP and ICMP Echo (ping) packets. The selected packet (Frame 3) is a GRE-tunneled ICMP Echo request. The packet details pane shows the encapsulation layers: Ethernet II, Internet Protocol (10.0.0.1 to 10.0.0.2), Generic Routing Encapsulation (IP), Internet Protocol (192.168.0.1 to 192.168.255.254), and Internet Control Message Protocol.

No.	Time	Source	Destination	Protocol	Info
1	0.000000	f6:52:a9:14:27:71	ce:4b:6f:f7:1b:4f	ARP	Who has 10.0.0.1? Tel
2	0.000125	ce:4b:6f:f7:1b:4f	f6:52:a9:14:27:71	ARP	10.0.0.1 is at ce:4b:6
3	0.019099	192.168.0.1	192.168.255.254	ICMP	Echo (ping) request
4	0.019142	192.168.255.254	192.168.0.1	ICMP	Echo (ping) reply
5	1.029215	192.168.0.1	192.168.255.254	ICMP	Echo (ping) request
6	1.029267	192.168.255.254	192.168.0.1	ICMP	Echo (ping) reply
7	2.039255	192.168.0.1	192.168.255.254	ICMP	Echo (ping) request
8	2.039307	192.168.255.254	192.168.0.1	ICMP	Echo (ping) reply

Frame 3 (122 bytes on wire, 96 bytes captured)

- Ethernet II, Src: ce:4b:6f:f7:1b:4f (ce:4b:6f:f7:1b:4f), Dst: f6:52:a9:14:27:71 (f6:52:a9:14:27:71)
- Internet Protocol, Src: 10.0.0.1 (10.0.0.1), Dst: 10.0.0.2 (10.0.0.2)
- Generic Routing Encapsulation (IP)
- Internet Protocol, Src: 192.168.0.1 (192.168.0.1), Dst: 192.168.255.254 (192.168.255.254)
- Internet Control Message Protocol

```

0000  f6 52 a9 14 27 71 ce 4b 6f f7 1b 4f 08 00 45 00  .R.. 'q.K o..0..E.
0010  00 6c 00 00 40 00 ff 2f 67 60 0a 00 00 01 0a 00  .l..@../ g.....
0020  00 02 00 00 08 00 45 00 00 54 00 00 40 00 3f 01  .....E..T.@.?.
0030  ba 58 c0 a8 00 01 c0 a8 ff fe 08 00 3e 4b 1d 02  .X.....>K.....
0040  0b 40 3c f1 24 4a 41 34 04 00 08 09 0a 0b 0c 0d  .@<.$JA4 .....
0050  0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d  .....
  
```

File: "out.cap" 1036 Byte... Packets: 10 Displayed: 10 Marked: 0 Profile: Default

- 1 Qu'est-ce que netkit ?
- 2 Terminologie
- 3 Installation de netkit
- 4 Préparer un lab (au sens netkit du terme)
- 5 Étude de cas (routage statique)
- 6 more, more... with netkit
- 7 Contributions**

Installation de visualnetkit

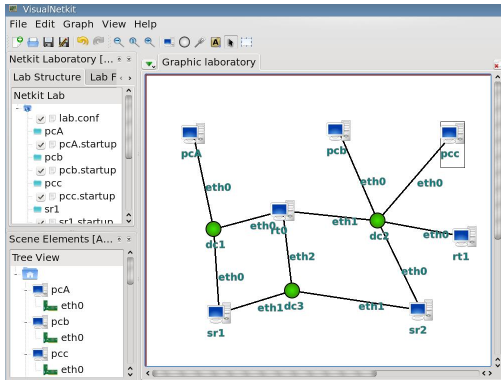
visualnetkit est un environnement graphique pour gérer les 'labs'

- Voir <http://wiki.netkit.org/> (rubrique Contributions)

```
sudo apt-get remove libqt3-headers libqt3-mt-dev qt3-dev-tools
sudo apt-get install libqt4-core libqt4-dev libqt4-qt3support qt4-dev-tools g++
# Télécharger et décompresser l'archive
cd visualnetkit
./build.sh
bin/visualnetkit.sh
```

Qu'est-ce que netkit ?
Terminologie
Installation de netkit
Préparer un lab (au sens netkit du terme)
Étude de cas (routage statique)
more, more... with netkit
Contributions

visualnetkit



more & more...with netkit

- `man netkit.conf, netkit.filesystem...`
- `http://wiki.netkit.org`
- lucky, 'use the source', luck, 'read the source' !
- utiliser netkit

Remerciements

La netkit team :

- Giuseppe Di Battista
- Maurizio Patrignani
- Maurizio Pizzonia
- Massimo Rimondini

du 'Computer Networks Laboratory' de l'université 3 de Rome.

À propos de la présentation

netkit est utilisé

dans le cadre de l'École Ouverte Francophone

<http://eof.eu.org>

comme support de formation par la SARL keepin

<http://keepin.eu>

Historique des modifications

- 20100802 Capture et analyse de trame graphique.