Initiation au routage, 1ère partie

Publié par :

Philippe Latu

philippe.latu(at)linux-france.org

http://www.linux-france.org/prj/inetdoc/

Historique des versions		
$Revision: 1365 $	$Date: 2009-01-03 11:03:16 +0100 (sam 03 jan 2009) $	$Author: latu $
Année universitaire 2006-2007		

Résumé

Ce document est le premier article d'une série rédigée par Pacôme Massol sur l'utilisation d'un système GNU/Linux comme routeur. Le logiciel présenté : Zebra ainsi que son successeur Quagga possèdent de nombreux atouts pour faciliter le développement d'équipements d'interconnexion réseau entièrement basés sur du logiciel libre.Ce document a été publié dans le numéro 42 de Linux Magazine en Septembre 2002. La version publiée ici pour le projet inetdoc.LINUX ne contient que quelques différences mineures sur la configuration du logiciel.

Table des matières

1. Copyright et Licence

1.1. Méta-information

Cet article est écrit avec *DocBook*[2] XML sur un système *Debian GNU/Linux*[3]. Il est disponible en version imprimable aux formats PDF et Postscript : `zebra.statique.pdf`[4]| `zebra.statique.ps.gz`[5].

Toutes les commandes utilisées dans ce document ne sont pas spécifiques à une version particulière des systèmes UNIX ou GNU/Linux. Comme la distribution *Debian GNU/Linux* est utilisée pour l'ensemble des supports du projet *inetdoc.LINUX*, voici une liste des paquets contenant les commandes nécessaires :

- net-tools - The NET-3 networking toolkit
- quagga - Unofficial successor of the Zebra BGP/OSPF/RIP routing daemon
- quagga-doc - info files for quagga
- zebra & zebra-doc - anciens paquets non mis à jour depuis 2003.

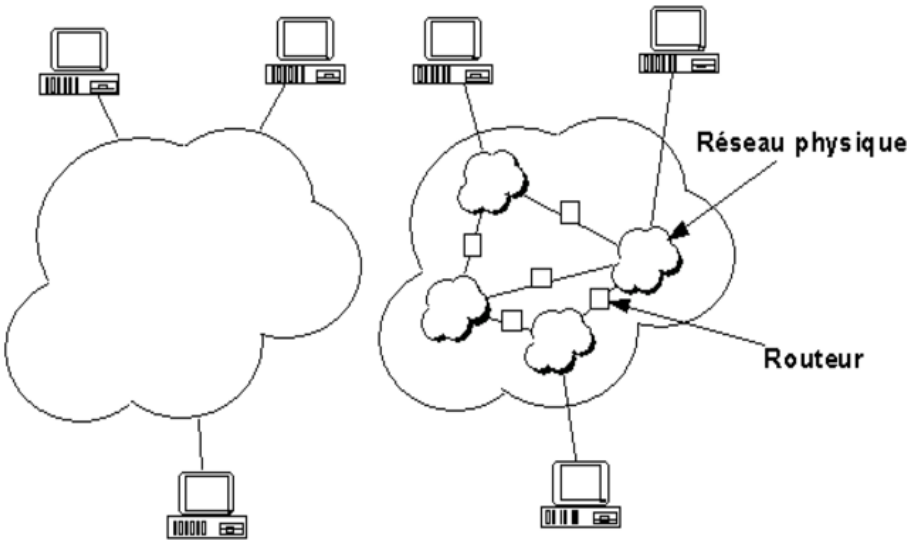
Les copies d'écran présentées ici correspondent à la publication initiale. Depuis, les versions de Zebra puis de Quagga ont évolué et l'affichage des informations de routage a été modifié. Cependant, ces modifications ne devraient pas gêner les lecteurs.

2. Avant-propos

Comment, au travers d'un réseau étendu comme Internet, un ordinateur arrive-t-il à communiquer avec un autre situé à des kilomètres et dont il ne connaît à peu près rien si ce n'est son adresse IP ? Les réseaux de nombreux opérateurs publics et privés assurent une connexion physique entre les deux appareils. Mais ce n'est pas tout. Cela fonctionne parce que le protocole IP est routable. Les objectifs de cette série d'articles sont de vous présenter les principes du routage ainsi que la configuration des routeurs dans un réseau.

D'un point de vue utilisateur, nous considérons Internet comme sur la figure A : un immense et unique réseau. En réalité, Internet est composé d'un ensemble de réseaux reliés via des appareils particuliers : les routeurs (figure B).

[2] http://www.docbook.org
[3] http://www.debian.org
[4] http://www.linux-france.org/prj/inetdoc/telechargement/zebra.statique.pdf
[5] http://www.linux-france.org/prj/inetdoc/telechargement/zebra.statique.ps.gz



D'après D. COMER

Internet un réseau de routeurs[6]

Le protocole réseau de l'Internet : IP est capable de choisir un chemin (une route) suivant lequel les paquets de données seront relayés de proche en proche jusqu'au destinataire. À chaque relais sur la route correspond un routeur. L'ordinateur émetteur du paquet de données doit trouver le premier relais. Ensuite, chaque routeur est chargé de trouver le suivant. Enfin, le dernier routeur remet le paquet sur le réseau du destinataire. Le routage IP fonctionne de façon totalement décentralisée au niveau des appareils qui constituent le réseau. Aucun n'a une vision globale de la route que prendront les paquets de données.

3. Les principes du routage

Avant d'aborder la partie pratique, je vais vous présenter quelques explications théoriques qui me semblent un préalable indispensable à une compréhension précise du routage. Je vais essayer de ne pas être trop long. Le routage IP repose sur quatre principes :

3.1. Une adresse IP est structurée

Chaque interface réseau d'un appareil possède une adresse IP unique dans tout le réseau global. Cette adresse est structurée en deux parties :

- la première partie (ou préfixe) donne le numéro du réseau,
- la seconde partie (ou suffixe) donne le numéro de l'interface dans ce réseau.

Un masque est associé à cette adresse et permet au logiciel IP de déterminer le préfixe réseau d'une adresse en effectuant un ET logique avec le masque. Prenons l'exemple d'une interface `eth0` avec une adresse IP `192.168.2.254` et un masque réseau `255.255.255.0` :

```
   192.168.  2.1
ET 255.255.255.0
  ---------------
   192.168.  2.0 => préfixe réseau de l'adresse
```

Si vous ne vous sentez pas à l'aise avec ces notions, inutile d'aller plus loin : je vous renvoie vers les précédents numéros de Linux magazine qui ont déjà traité ce point.

[6] http://www.linux-france.org/prj/inetdoc/guides/zebra.statique/images/Internet.png

3.2. Les paquets de données comportent l'adresse IP de l'émetteur et du destinataire

Lors de l'émission, le protocole découpe les données en petits paquets (souvent appelés datagrammes IP). Ces paquets ont tous la même structure :

Datagramme IP[7]

C'est l'en-tête qui contient, entre autre, les adresses de l'émetteur et du destinataire. Un appareil chargé du routage analysera l'adresse du destinataire afin d'aiguiller le paquet vers le prochain routeur menant à sa destination.

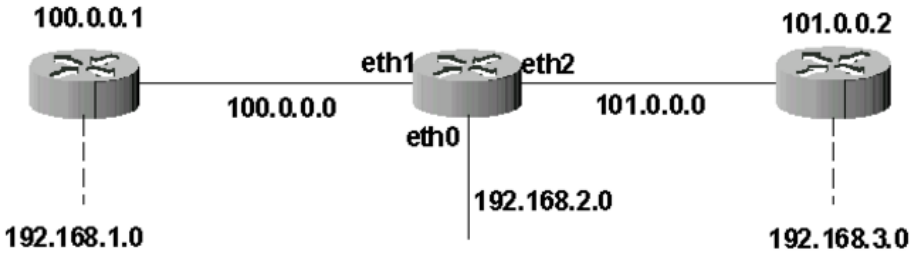
3.3. Chaque appareil possède une table de routage gérée par le logiciel IP

Une table de routage est une liste contenant essentiellement trois types d'informations : des adresses réseau avec le masque réseau associé et le moyen de les atteindre. Soit le réseau est directement connecté à l'appareil, dans ce cas le moyen de l'atteindre est le nom de l'interface, soit, il s'agit de l'adresse du prochain routeur situé sur la route vers ce réseau. Par exemple, considérons sur un appareil quelconque, sa table de routage :

Tableau 1. Table de routage

Réseau	Masque	Moyen de l'atteindre
192.168.2.0	255.255.255.0	eth0
100.0.0.0	255.0.0.0	eth1
101.0.0.0	255.0.0.0	eth2
192.168.1.0	255.255.255.0	100.0.0.1
192.168.3.0	255.255.255.0	101.0.0.2

Cette table est riche d'enseignements. On apprend très précisément que l'appareil possède trois interfaces réseau (eth0, eth1, eth2) ainsi que les adresses IP des réseaux qui sont directement reliés à ces interfaces. On connaît les adresses IP de deux routeurs. On sait qu'il existe deux réseaux 192.168.1.0 et 192.168.3.0 et qu'ils sont respectivement derrière les routeurs 100.0.0.1 et 101.0.0.2. En revanche, il est impossible d'affirmer que ces deux réseaux sont directement reliés à ces routeurs. Pour résumer, on peut dresser le schéma suivant :



Topologie d'après une table de routage[8]

Quelques observations complémentaires :

- Étant donné que l'appareil observé possède trois interfaces, c'est très probablement un routeur. Cependant, notez que tout appareil fonctionnant sous TCP/IP possède une table de routage (qu'il soit routeur ou non).

[7] http://www.linux-france.org/prj/inetdoc/guides/zebra.statique/images/datagr.png
[8] http://www.linux-france.org/prj/inetdoc/guides/zebra.statique/images/topo1.png

- Pour que le routage fonctionne, il est impératif que toutes les interfaces réseau possédant le même préfixe réseau soient reliées au même réseau physique.

3.4. Tous les appareils sous IP exécutent le même algorithme

Lors de l'émission d'un paquet de données, le logiciel IP recherche une correspondance dans la table en appliquant le masque réseau de chaque ligne avec l'adresse IP de destination du paquet. Notez qu'il parcourt la table dans l'ordre décroissant des masques afin de garantir la correspondance la plus précise entre l'adresse dans la table et l'adresse de destination (*best match*).

Au total, seules quatre possibilités sont imaginables :

- Ce préfixe correspond à celui d'un réseau directement connecté ; il y a remise directe du paquet sur le réseau et le routage est terminé.
- Ce préfixe correspond à celui d'un réseau accessible via un routeur on récupère l'adresse physique de ce routeur et on lui transmet le paquet. Notez que l'adresse IP de l'émetteur reste inchangée.
- Ce préfixe n'a pas de correspondance dans la table mais il existe un routeur par défaut dans la table ; on transmet au routeur par défaut.
- Si aucun des trois cas précédents n'est rempli, on déclare une erreur de routage.

4. Place à la pratique

Voilà, vous savez tout sur les aspects théoriques. Une mise en pratique est maintenant indispensable. Je vous propose de travailler avec le routeur logiciel GNU *Zebra*[9]. C'est, à mon sens, un excellent logiciel, pour les raisons suivantes :

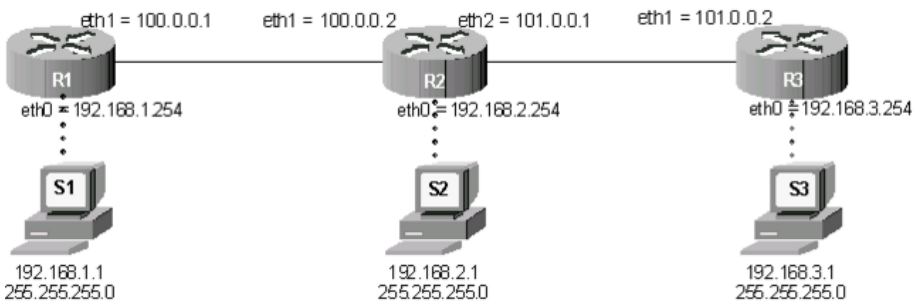
- c'est un logiciel libre sous licence GNU,
- il propose une interface de configuration interactive accessible via telnet,
- il fonctionne selon une philosophie et un langage de configuration proche de routeurs répandus dans les entreprises (ce qui permet d'avoir accès à une bonne bibliographie),
- il supporte les principaux protocoles de routage (nous développerons ce point dans la deuxième partie de cet article),
- il fonctionne avec IPv6.

Les manipulations présentées dans la version originale du document ont été réalisées avec *Zebra* 0.91a sous une Redhat 7.2. Ce logiciel est prévu pour fonctionner sous Linux (noyau 2.0.37 et suivants) et BSD. Une version pour Hurd était prévue.

Les mêmes manipulations peuvent être effectuées avec *Quagga*[10] ; le successeur de *Zebra*. Seule la localisation des fichiers de configuration change.

Tous les éléments de syntaxe restent conformes à la syntaxe de l'`IOS` du constructeur Cisco Systems™. Au fil des ans, cette syntaxe est devenue le standard de fait de la configuration des équipements d'interconnexion.

Je vous propose de travailler avec la maquette suivante :



Topologie de travail[11]

[9] http://www.zebra.org/
[10] http://www.quagga.net/
[11] http://www.linux-france.org/prj/inetdoc/guides/zebra.statique/images/topo2.png

Ce réseau est composé de 3 routeurs (R1, R2 et R3) et de trois stations (S1, S2, S3). Je suppose que toutes les interfaces réseau sont actives et correctement configurées. Les masques réseau à utiliser sont les masques par défaut de la classe d'adresse (255.0.0.0 pour les adresses commençant par 100 et 101 et 255.255.255.0 pour les adresses commençant par 192).

4.1. Mise en place des routeurs

Zebra ou *Quagga* doivent être installés sur chaque ordinateur qui fera office de routeur. Vous pouvez vous procurer des paquets d'installation ou bien compiler le logiciel à partir des fichiers sources. Dans ce cas, l'installation se fait par les commandes habituelles :

```
./configure ; make ; make install
```

Cette phase produit plusieurs exécutables (un par protocole de routage) mais nous n'utiliserons pour l'instant que **zebra**. En principe, les exécutables ont été copiés dans `/usr/local/bin` et les fichiers de configuration dans `/usr/local/etc` (ils portent le même nom que l'exécutable avec l'extension .conf). Suivant la méthode d'installation, ils pourront être situés ailleurs, ce n'est pas un problème. Pour une aide plus détaillée, reportez-vous sur mon site, vous y trouverez une traduction française du manuel.

Tableau 2. Table de localisation des éléments logiciels

Type d'installation	**Fichiers binaires**	**Fichiers de configuration**	**(Script|Commande) d'initialisation**
À partir des sources	`/usr/local/bin/`	`/usr/local/etc/`	**zebra -d**
Paquetage Debian zebra	`/usr/lib/zebra/`	`/etc/zebra/`	**/etc/init.d/zebra (start|stop|restart)**
Paquetage Debian quagga	`/usr/lib/quagga/`	`/etc/quagga/`	**/etc/init.d/quagga (start|stop|restart)**

Avant de charger le démon de routage, dans le répertoire des fichiers de configuration de chaque routeur créez un fichier `zebra.conf`. Insérez les deux lignes suivantes :

```
hostname Rx(Zebra)
password foo
```

- Remplacez le x de `Rx(Zebra)` par le numéro du routeur
- A la place de `foo`, indiquez le mot de passe que vous souhaitez saisir lorsque vous vous connecterez au routeur via telnet.

Maintenant, vous pouvez lancer le démon de routage avec la commande ou le script indiqué dans la table ci-dessus afin que le logiciel de routage s'exécute en tâche de fond.

Configuration manuelle sans paquet

Exemple du routeur R1 ; respectez bien l'ordre des commandes :

```
R1 # zebra -d
```

Configuration avec le paquet Quagga :

Le fichier `/etc/quagga/daemons` spécifie la liste des démons à utiliser :

```
# This file tells the quagga package which daemons to start.
#
# Entries are in the format: <daemon>=(yes|no|priority)
#   0, "no"  = disabled
#   1, "yes" = highest priority
#   2 .. 10  = lower priorities
# Read /usr/share/doc/quagga/README.Debian for details.
#
# Sample configurations for these daemons can be found in
# /usr/share/doc/quagga/examples/.
```

```
#
# ATTENTION:
#
# When activation a daemon at the first time, a config file, even if it is
# empty, has to be present *and* be owned by the user and group "quagga", else
# the daemon will not be started by /etc/init.d/quagga. The permissions should
# be u=rw,g=r,o=.
# When using "vtysh" such a config file is also needed. It should be owned by
# group "quaggavty" and set to ug=rw,o= though.
#
zebra=yes
bgpd=no
ospfd=no
ospf6d=no
ripd=no
ripngd=no
isisd=no
```

La commande suivante sert à relancer les démons après avoir édité le fichier ci-dessus :

```
R1 # /etc/init.d/quagga start
```

4.2. Configuration des stations

Plaçons-nous dans le shell de S1 et observons la configuration des interfaces :

```
S1 # ifconfig
eth0      Lien encap:Ethernet  HWaddr 00:50:56:40:40:98
          inet adr:192.168.1.1  Bcast:192.168.1.255  Masque:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:89 errors:0 dropped:0 overruns:0 frame:0
          TX packets:58 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:100
          RX bytes:6771 (6.6 Kb) TX bytes:3357 (3.2 Kb)
          Interruption:10 Adresse de base:0x1080

lo        Lien encap:Boucle locale
          inet adr:127.0.0.1 Masque:255.0.0.0
          UP LOOPBACK RUNNING MTU:16436 Metric:1
          RX packets:77 errors:0 dropped:0 overruns:0 frame:0
          TX packets:77 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:0
          RX bytes:4758 (4.6 Kb) TX bytes:4758 (4.6 Kb)
```

Cet appareil dispose d'une interface Ethernet active nommée `eth0` ainsi que de l'interface de bouclage logiciel `lo`. Toute machine fonctionnant avec IP possède cette interface.

Je vous ai dit tout à l'heure que tout appareil fonctionnant sous IP disposait d'une table de routage. Listons le contenu de cette table sur S1 :

```
S1 # route -n
Table de routage IP du noyau
Destination     Passerelle      Genmask         Indic Metric Ref    Use Iface
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
```

A partir des adresses IP des interfaces de l'ordinateur, le logiciel IP en a déduit cette table de routage élémentaire. Pour lui, toutes les machines qui disposent d'une adresse commençant par 192.168.1.0 sont forcément sur le réseau physique connecté à l'interface `eth0`.

Conclusion, si je tente un **ping** vers une adresse de ce réseau (et si une machine possède cette adresse), j'obtiens une réponse. Essayons entre S1 et R1 qui font partie du même réseau :

```
S1 # ping -c2 192.168.1.254
```

```
PING 192.168.1.254 (192.168.1.254): 56 data bytes
64 bytes from 192.168.1.254: icmp_seq=0 ttl=64 time=2.308 ms
64 bytes from 192.168.1.254: icmp_seq=1 ttl=64 time=2.308 ms

--- 192.168.1.254 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 2.308/2.308/2.308 ms
```

Je cherche à contacter un appareil dont l'adresse commence par 192.168.1.0. Cette adresse figure dans la table routage, donc tout va bien.

Essayons maintenant de contacter l'interface `eth1` de R1 :

```
S1 # ping 100.0.0.1
connect: Network is unreachable
```

La sanction est immédiate. En clair, votre système d'exploitation favori vous répond : «désolé, mais je n'ai absolument aucune idée de la façon dont je pourrais bien atteindre le réseau 100.0.0.0».

Pour résoudre ce problème, il faut que je renseigne ma table de routage et que j'indique comment atteindre le réseau 100.0.0.0. Sur le schéma, c'est très clair. Pour aller sur ce réseau, il faut passer par R1. Comme S1 ne peut joindre pour l'instant que les appareils dont l'adresse commence par 192.168.1.0, j'indiquerai comme moyen d'atteindre le réseau, l'adresse 192.168.1.254 qui est l'adresse IP de l'interface du routeur qui se situe sur le réseau de S1. Pour réaliser cette configuration, tapons la commande :

```
S1 # route add -net 100.0.0.0 netmask 255.0.0.0 gw 192.168.1.254
```

Félicitations ! Vous venez de saisir votre première commande de configuration de routage. Elle signifie que le réseau 100.0.0.0/8 (masque réseau sur 8 bits) est situé derrière le routeur (gw = gateway) d'adresse 192.168.1.254. Vous pouvez le tester, cette configuration fonctionne pour le réseau 100.0.0.0 mais si l'on généralise, il faudrait saisir pour tous les réseaux que l'on cherche à contacter, une commande identique ! Observez bien le schéma. R1 est le seul routeur directement accessible par S1. Quel que soit le réseau que S1 cherche à contacter, il ne peut être que derrière R1. Par conséquent, il existe une commande qui permet d'indiquer une route par défaut :

```
S1 # route add default gw 192.168.1.254
```

Listons le contenu de la table de routage :

```
S1 # route -n
Table de routage IP du noyau
Destination     Passerelle      Genmask         Indic Metric Ref    Use Iface
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
100.0.0.0       192.168.1.254   255.0.0.0       U     0      0        0 eth0
default         192.168.1.254   0.0.0.0         UG    0      0        0 eth0
```

La ligne commençant par 100.0.0.0 est devenue inutile. Supprimons la :

```
S1 # route del -net 100.0.0.0 netmask 255.0.0.0 gw 192.168.1.254
```

La plupart du temps, il n'existe qu'un seul routeur pour sortir d'un réseau d'extrémité. On configure alors sur chaque station l'adresse IP de ce routeur par défaut. Le logiciel IP crée une entrée dans sa table de routage identique à celle que nous venons d'observer. Vous devrez donc, sur chaque station de notre réseau définir l'adresse de son routeur par défaut, soit en tapant une commande route, soit en modifiant les fichiers de configuration des cartes réseau et en redémarrant.

Revenons sur S1 et testons notre configuration. Contactons à nouveau l'interface 100.0.0.1 du routeur R1 :

```
S1 # ping -c2 100.0.0.1
PING 100.0.0.1 (100.0.0.1): 56 data bytes
64 bytes from 100.0.0.1: icmp_seq=0 ttl=64 time=3.746 ms
64 bytes from 100.0.0.1: icmp_seq=1 ttl=64 time=1.812 ms

--- 100.0.0.1 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
```

```
round-trip min/avg/max = 1.812/2.779/3.746 ms
```

Parfait ça marche ! Je devrais donc pouvoir également contacter R2 puisqu'il est lui aussi dans le réseau 100.0.0.0 :

```
S1 # ping 100.0.0.2
PING 100.0.0.2 (100.0.0.2) from 192.168.1.1 : 56(84) bytes of data.

(il ne se passe rien, donc CTRL-C)

--- 100.0.0.2 ping statistics ---
10 packets transmitted, 0 packets received, 100% packet loss
```

Eh bien, ce n'est pas brillant. Certes, S1 n'indique plus de message d'erreur mais les paquets transmis ne sont jamais retournés. Vous vous doutez qu'il existe une solution. Profitons-en, c'est l'occasion de vous donner quelques éléments pour repérer un problème de routage.

4.3. Configuration des routeurs

Puisque le routage est une chaîne, il faut suivre les paquets dans chaque maillon afin de trouver l'origine du problème. Nous savons que l'interface eth1 du routeur R1 reçoit les paquets puisqu'elle nous les retourne. Donc, le problème vient de R2. Positionnons-nous dans le shell de R2, la commande **tcpdump** va nous aider à observer ce qu'il se passe sur son interface eth1 :

```
R2 # tcpdump -nt -i eth1
tcpdump: listening on eth1
192.168.1.1 > 100.0.0.2: icmp: echo request (DF)
192.168.1.1 > 100.0.0.2: icmp: echo request (DF)
```

Oui, elle reçoit les paquets... mais elle n'en retourne aucun. Vous l'avez compris : comme pour S1 tout à l'heure, R2 n'a aucune idée de l'endroit où se trouve le réseau de l'émetteur des paquets (192.168.1.0) puisque celui-ci n'est pas directement connecté. Il faut donc configurer sa table de routage. Pour ce faire, nous allons cette fois travailler avec Zebra. Zebra possède une interface telnet sur le port 2601. Dans le shell de R2, tapez :

```
R2 # telnet localhost 2601
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

Hello, this is zebra (version 0.91a).
Copyright 1996-2001 Kunihiro Ishiguro.

User Access Verification
Password:
```

Tapez le mot de passe que vous avez saisi dans le fichier `zebra.conf`, vous arrivez dans le mode de visualisation de la configuration du routeur. Ensuite, passez en mode de configuration (mode privilégié appelé mode enable dans le logiciel) :

```
R2(Zebra)> enable
```

Pour vous repérer dans Zebra, observez bien le prompt, il vous indique dans quel mode vous vous trouvez (par exemple, le # indique que vous êtes en mode privilégié). Ensuite, dans l'interpréteur de commande, vous pouvez saisir à tout moment un ? Pour obtenir la liste contextuelle des commandes. Enfin, lorsque vous appuyez sur la touche tabulation, Zebra complète la commande en cours de saisie.

Observons la table de routage gérée par Zebra :

```
R2(Zebra)# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - ISIS, B - BGP, > - selected route, * - FIB route

C>* 100.0.0.0/8 is directly connected, eth1
C>* 101.0.0.0/8 is directly connected, eth2
```

```
C>* 127.0.0.0/8 is directly connected, lo
C>* 192.168.2.0/24 is directly connected, eth0
```

Nous ne voyons aucune trace du réseau 192.168.1.0. C'est pour cela que R2 ne peut retourner les paquets ICMP à S1. Bien sûr, Zebra permet d'ajouter une route. Passons en mode «terminal de configuration» :

```
R2(Zebra)# configure terminal
```

Puis ajoutons la route :

```
R2(Zebra)(config)# ip route 192.168.1.0/24 100.0.0.1
```

Revenons au mode enable et listons à nouveau la table :

```
R2(Zebra)(config)# end
R2(Zebra)# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - ISIS, B - BGP, > - selected route, * - FIB route

C>* 100.0.0.0/8 is directly connected, eth1
C>* 101.0.0.0/8 is directly connected, eth2
C>* 127.0.0.0/8 is directly connected, lo
S>* 192.168.1.0/24 [1/0] via 100.0.0.1, eth1
C>* 192.168.2.0/24 is directly connected, eth0
```

Une route statique (notée S) est apparue (vous apprendrez dans le prochain article comment configurer une route dynamique ainsi que la signification des nombres entre crochets [1/0]).

Un ping 100.0.0.2 depuis S1 passe désormais sans problème. Si l'on reprend le schéma du réseau, vous vous doutez que des manipulations similaires sont à réaliser pour le réseau de S3. Il faut donc ajouter une route vers 192.168.3.0/24. Vous connaissez maintenant les commandes :

```
R2(Zebra)# configure terminal
R2(Zebra)(config)# ip route 192.168.3.0/24 101.0.0.2
R2(Zebra)(config)# end
```

Visionnons la configuration en mémoire de Zebra :

```
R2(Zebra)# show running-config

Current configuration:
!
hostname R2(Zebra)
password foo
!
interface lo
!
interface eth0
!
interface eth1
!
interface eth2
!
ip route 192.168.1.0/24 100.0.0.1
ip route 192.168.3.0/24 101.0.0.2
!
line vty
!
end
```

Afin qu'à chaque démarrage de Zebra les routes statiques soient prises en compte, il faut enregistrer cette configuration «mémoire» vers le fichier `zebra.conf` :

```
R2(Zebra)# copy running-config startup-config
```

```
Configuration saved to /etc/zebra/zebra.conf
```

Bien, nous avons presque fini. Quelques routes sont à créer sur R1 et R3 pour que le routage sur notre réseau soit complet.

4.4. Une erreur à éviter

Sur R1, il faut créer une route vers 192.168.3.0. Une erreur fréquente consiste à créer la route suivante :

```
R1(Zebra)(config)# ip route 192.168.3.0/24 101.0.0.2
```

Ce qui signifie : le réseau 192.168.3.0/24 est situé derrière le routeur R3. Bien sûr, cette phrase est juste, mais souvenez-vous de ce que nous disions en introduction : le routage fonctionne de proche en proche. Ainsi, comme nous sommes sur R1, il suffit d'indiquer que le routeur nous permettant d'atteindre le réseau de S3 est R2 et non R3.

4.5. Configuration de R1

Ceci étant dit, voici la configuration de R1 :

```
R1(Zebra)# show running-config

Current configuration:
!
hostname R1(Zebra)
password foo
!
interface lo
!
interface eth0
!
interface eth1
!
ip route 101.0.0.0/8 100.0.0.2
ip route 192.168.2.0/24 100.0.0.2
ip route 192.168.3.0/24 100.0.0.2
!
line vty
!
end
```

4.6. Configuration de R3

Avant de lire ci-dessous, essayez de déterminer la configuration de R3. Elle est très proche de celle de R1.

```
R3(Zebra)# show running-config

Current configuration:
!
hostname R3(Zebra)
password foo
!
interface lo
!
interface eth0
!
interface eth1
!
ip route 100.0.0.0/8 101.0.0.1
ip route 192.168.1.0/24 101.0.0.1
ip route 192.168.2.0/24 101.0.0.1
!
line vty
!
```

```
end
```

Ouf ! Voilà, c'est fini. Vous devez pouvoir réaliser des **ping** de n'importe quelle machine vers n'importe quelle autre.

5. Conclusion

La configuration des routeurs peut vous sembler fastidieuse, voire impossible si le réseau comporte beaucoup de routeurs et que sa topologie évolue fréquemment. Il faudrait sans cesse reconfigurer les routeurs. Heureusement, le monde est bien fait : il existe des protocoles qui permettent aux routeurs de s'échanger les informations de routage dont ils disposent afin que les tables s'adaptent aux évolutions du réseau. Le protocole RIP permet cela et c'est ce que je vous propose d'aborder dans le prochain article.

5.1. Bibliographie

TCP/IP : Architecture, protocoles et applications
Douglas COMER, DUNOD. ISBN: 2-10-005384-1 (09/2001) 830 p.

Le routage dans l'Internet
Christian HUITEMA, EYROLLES. ISBN: 2-212-08902-3 (10/1994) 418 p.

5.2. Liens

- *Zebra*[12]
- *Quagga*[13]
- *Linux Magazine*[14]
- Version originale du document et page personnelle de *Pacôme Massol*[15]

[12] http://www.zebra.org/
[13] http://www.quagga.net/
[14] http://www.linuxmag-france.org/
[15] http://www.pmassol.net/