# Associer un pont Ethernet et netfilter

## Version française du *Ethernet Bridge + netfilter Howto*


### Nils Radtke

<Nils.Radtke_@_Think-Future.de>

**François Romieu –** Traduction française

**Guillaume Lelarge –** Relecture de la version française



0.2.fr.1.0

2003–03–09


**Historique des versions**

| | | |
|---|---|---|
| Version 0.2.fr.1.0 | 2003–03–09 | Revu par : FR |
| Première traduction française | | |
| Version 0.2 | 2002–10–08 | Revu par : NR |
| Ajout de la partie d'exemple de configuration et d'indications dans la configuration du routage et le test du routage respectivement. | | |
| Version 0.1 | 2002–09–19 | Revu par : NR |
| Mise à jour du lien vers ebtables dans la section des « Thèmes voisins ». Ajout d'une remarque ayant trait aux messages de déverminage des faux positifs de br_nf | | |


La mise en place d'un pont Ethernet permet d'ajouter une entité d'audit ou de régulation à un réseau de façon transparente. Une telle installation n'impose aucun changement à la topologie du réseau d'accueil. Elle s'effectue en connectant le pont Ethernet entre le réseau à analyser et l'élément responsable du routage (l'équipement connecté à l'Internet).


***Table des matières***

# 1. Introduction

La version originale de ce guide pratique est disponible dans d'autres formats. Pour le téléchargement, nous vous recommandons cette archive tar. La version originale de ce document est publié par le Projet de documentation Linux (LDP).

La dernière version française de ce document est disponible sur le site du projet de traduction traduc.org.

Pour ceux qui sont à la recherche d'une traduction, il existe aussi une version version allemande !

# 2. Introduction

Les ponts Ethernet joignent de façon transparente plusieurs segments Ethernet.

Un pont Ethernet distribue les trames qui se présentent à un port aux autres ports. Il l'effectue de façon intelligente : une fois qu'il sait grâce à partir de quel port joindre une interface d'adresse MAC donnée, la trame ne sera émise que sur le port correspondant sans polluer les autres segments.

Des interfaces Ethernet peuvent s'ajouter à une interface existante et devenir des ports (logiques) de l'interface du pont.

L'emploi d'un dispositif de type netfilter au dessus d'un pont rend le système capable d'effectuer du filtrage. On obtient ainsi un filtrage transparent. Aucune adresse IP n'est même nécessaire. Il est bien sûr possible d'en affecter une à l'interface de pontage à des fins de maintenance (via ssh :o) ).

L'intérêt du dispositif est évident. La transparence épargne à l'administrateur la charge de reprise de la topologie réseau. Les utilisateurs ne remarquent pas l'existence du pont mais les connexions sont bloquées. Enfin, la transition ne perturbe pas le fonctionnement opérationnel (qu'on se figure un réseau où la perte de connectivité réseau coûte cher !).

L'autre cas courant concerne les personnes connectées à l'Internet au moyen d'un routeur dédié. Les fournisseurs d'accès ne partagent guère les privilèges d'administration sur les équipements loués et le client ne peut donc pas modifier la configuration. Le client a cependant un réseau dont il ne veut pas reprendre toute la configuration. Il n'est effectivement pas obligé de le faire s'il a recours à un pont.

# 3. Logiciels requis

La configuration logicielle suivante est nécessaire sur l'hôte de pontage conformément à notre terrain de test.

## 3.1. Noyau Linux

La prise en charge du pontage Ethernet est disponible en standard à partir du noyau *2.4.18*. Aucun ajout n'est requis.

Pour disposer de netfilter et pouvoir se servir d'iptables, il faut toutefois appliquer un supplément de code. Le nécessaire se trouve dans la page sourceforge du pontage Ethernet.

```
root@bridge:~> cd /usr/src/
root@bridge:~> wget -c http://bridge.sourceforge.net/devel/bridge-nf/bridge-nf-0.0.7-against-2.4.18.diff
root@bridge:~> cd /usr/src/linux/
root@bridge:~> patch -p1 -i ../bridge-nf/bridge-nf-0.0.7-against-2.4.18.diff
```

Une fois le noyau standard rectifié, on active les options de configuration adéquates du noyau. On peut se reporter au document suivant pour la mise au point d'un noyau personnel : CD−Net−Install−HOWTO, boîte à outils. Oui, c'est encore en allemand. Je corrigerai ça à l'occasion. Pour l'instant, dans :

```
        Code maturity level options
```

on active :

```
        [*] Prompt for development and/or incomplete code/drivers
```

et dans :

```
        Loadable module support
```

```
        [*] Enable loadable module support
        [*]   Set version information on all module symbols
        [*]   Kernel module loader
```

Jusqu'ici, tout va bien. À présent, dans :

```
        Networking options
```

on active :

```
        [*] Network packet filtering (replaces ipchains)
        [*]   Network packet filtering debugging
```

De même, dans :

```
          IP: Netfilter Configuration  --->
```

on choisit tout ce qui est souhaité . Enfin, on active :

```
        [M] 802.1d Ethernet Bridging
```

et [1] :

```
        [*]   netfilter (firewalling) support
```

Il ne reste plus qu'à exécuter un cycle :

```
root@bridge:~> make dep clean bzImage modules modules_install
```

C'est tout. On n'oubliera pas d'éditer le fichier `/etc/lilo.conf` en conséquence avant de taper:

```
root@bridge:~> lilo -t
root@bridge:~> lilo
root@bridge:~> reboot
```

> Pourquoi ne pas identifier le noyau comme destiné au pontage ? On édite le Makefile de plus haut niveau dans les sources du noyau et on modifie la ligne qui comprend *EXTRAVERSION =*. On peut la positionner à *bridge* par exemple. Une fois l'étape *modules_install* effectuée, les modules se trouveront dans le répertoire `/lib/modules/2.4.18bridge`.

## 3.2. L'utilitaire brctl

Une fois le noyau capable de jouer les ponts Ethernet et de supporter netfilter, on prépare l'utilitaire brctl. brctl est l'outil de configuration pour le pontage. On télécharge les sources du paquetage puis on le décompresse et on se positionne dans le répertoire créé.

```
root@bridge:~> wget -c http://bridge.sourceforge.net/bridge-utils/bridge-utils-0.9.5.tar.gz
root@bridge:~> tar xvzf bridge-utils-0.9.5.tar.gz
root@bridge:~> cd bridge-utils-0.9.5
```

Il est temps de lire le fichier `README` ainsi que ceux qui se trouvent dans le répertoire `doc/`. On peut alors lancer une commande make. L'exécutable brctl/brctl qui en résulte est à copier dans le répertoire `/sbin/`.

```
root@bridge:~> make
root@bridge:~> cp -vi brctl/brctl /sbin/
```

On peut à présent passer à la section d'installation.

# 4. Mise en service de Linux

## 4.1. Mise en service du pont

Linux doit être mis au courant de l'existence du pont. On commence donc par réclamer une interface de pontage Ethernet virtuelle (à exécuter sur la machine *bridge*, voir la configuration de test) :

```
root@bridge:~> brctl addbr br0
```

Le protocole d'établissement d'arbre (Spanning Tree) n'est pas nécessaire. On suppose qu'il n'y a qu'un seul routeur. Une boucle est donc peu probable. La fonctionnalité correspondante peut donc être désactivée. Le bavardage réseau diminue alors.

```
root@bridge:~> brctl stp br0 off
```

Après cette phase préparatoire, on lance enfin quelques commandes intéressantes. On ajoute les interfaces Ethernet physiques en les attachant à l'interface de pontage virtuelle *br0* qui vient d'être créée :

```
root@bridge:~> brctl addif br0 eth0
root@bridge:~> brctl addif br0 eth1
```

À présent les interfaces Ethernet sont chacune devenues une extrémité du pont. Certes, elles étaient et elles sont toujours là (on peut les voir :o) ) mais comme elles appartiennent au pont, elles n'ont plus besoin de leur adresse IP. On leur retire donc celle−ci :

```
root@bridge:~> ifconfig eth0 down
root@bridge:~> ifconfig eth1 down
root@bridge:~> ifconfig eth0 0.0.0.0 up
root@bridge:~> ifconfig eth1 0.0.0.0 up
```

Parfait, on dispose donc à présent d'une station sans adresse IP. Si ce n'était pas déjà le cas, il est temps de passer sur une console locale à la machine pour la configurer. Une console série est la bienvenue.

> Option : On affecte une adresse IP à l'interface logique et on l'active :
>
> ```
> root@bridge:~> ifconfig br0 10.0.3.129 up
> ```

C'est tout. Il est conseillé de s'attarder sur une remarque importante !

---

## 4.2. Configuration du routage

Dans le cas de la configuration d'un passerelle, on active la transmission de paquets du noyau Linux :

```
root@bridge:~> echo "1" > /proc/sys/net/ipv4/ip_forward
```

La machine dispose déjà d'une adresse IP mais n'a aucune route par défaut. On corrige ce manque  :

```
root@bridge:~> route add default gw 10.0.3.129
```

La connectivité réseau devrait être normale, depuis, vers et au travers de la passerelle.

---

# 5. Test du pontage Ethernet

## 5.1. Configuration de test

On part de la situation suivante ou d'un schéma analogue :

```
                                                           /\
          Ethernet            Ethernet             ATM    /  -/\
+---------+       +---------+          +---------+     /-/      !
| Station |-------|  Pont   |----------| Routeur |-----| Inter-  \
+---------+       +---------+          +---------+     \  net  ---|
         ^        ^         ^          ^                \     /
         |        |         |          |                 \---/
        eth0     eth0      eth1       if0                   ^
         |        |         |          |                   |
      10.0.3.2   rien/10.0.3.1      195.137.15.7   le reste du monde
                  \         /
                   \       /
   ^                \-br0-/
   |                                      ^             ^
   |                   ^                  |             |
```

```
   |                    |                  |             |
 perso                perso             étranger      agressif
```

Les possibilités d'administration sont limitées aux équipements marqués *perso*. Le routeur, et l'Internet, sont inaccessibles.

Si on veut contrôler le trafic sur le brin Ethernet, on ne peut qu'ajouter un pare−feu ou intégrer un pont.

La méthode habituelle a pour revers le changement de route par défaut sur chaque machine du réseau interne. C'est extrêmement pénible et personne n'a envie de devoir changer 5 routes par défaut sur 5 hôtes plus d'une fois. En outre, ça consomme du temps, on peut se tromper et la sécurité n'est pas améliorée.

La seconde approche est plus systématique, consomme moins de temps et réduit les risques d'erreur. Elle est plus sûre en ce sens qu'il n'est pas nécessaire de faire apparaître une adresse IP supplémentaire. Pas d'IP, pas de danger. Enfin, il s'agit de la théorie en supposant que les piles sont sûres (ce qui a intérêt à être vérifié). L'emploi d'un pont est transparent, pas de changements d'IP ou d'adresses MAC, c'est là son attrait.

Chacun choisira sa méthode mais seule la plus amusante est examinée ici.

## 5.2. Ping le, Max !

On configure l'interface eth0 comme d'habitude. Les interfaces du pont sont configurées conformément à la section d'installation.

La commande ci−dessous est importante pour activer la transmission de paquets.

```
root@bridge:~> echo "1" > /proc/sys/net/ipv4/ip_forward
```

On configure éventuellement une route par défaut :

```
root@bridge:~> route add default gw 10.0.3.129
```

On met en place les règles de filtrage sur *bridge* :

```
root@bridge:~> iptables −P FORWARD DROP
root@bridge:~> iptables −F FORWARD
root@bridge:~> iptables −I FORWARD −j ACCEPT
root@bridge:~> iptables −I FORWARD −j LOG
root@bridge:~> iptables −I FORWARD −j DROP
root@bridge:~> iptables −A FORWARD −j DROP
root@bridge:~> iptables −x −v −−line−numbers −L FORWARD
```

La dernière ligne produit l'affichage suivant :

```
Chain FORWARD (policy DROP 0 packets, 0 bytes)
num      pkts      bytes target    prot opt in     out     source   destination
1           0        0 DROP        all  --  any    any     anywhere anywhere
2           0        0 LOG         all  --  any    any     anywhere anywhere      LOG level warning
3           0        0 ACCEPT      all  --  any    any     anywhere anywhere
4           0        0 DROP        all  --  any    any     anywhere anywhere
```

La cible *LOG* trace tous les paquets via *syslogd*. Une telle configuration devrait se limiter à la phase de test puisqu'elle ouvre la voie à un épuisement prématuré de la capacité de stockage de la machine en cas d'attaque de type déni de service.

On teste les règles de filtrage en pingant l'adresse IP (195.137.15.7) du routeur sur la machine *babasse* :

```
root@box:~> ping -c 3 195.137.15.7
PING router.provider.net (195.137.15.7) from 10.0.3.2 : 56(84) bytes of data.
--- router.provider.net ping statistics ---
3 packets transmitted, 0 received, 100% loss, time 2020ms
^C
root@box:~>
```

La règle par défaut rejette (DROP) le trafic. Pas de réponse ni de traçage des trames. Cette configuration netfilter est destinée à jeter toutes les trames à moins que la règle 1 qui précède la règle LOG ne soit supprimée :

```
root@bridge:~> iptables -D FORWARD 1
root@bridge:~> iptables -x -v --line-numbers -L FORWARD
```

Les règles sont à présent :

```
Chain FORWARD (policy DROP 0 packets, 0 bytes)
num      pkts      bytes target   prot opt in     out     source   destination
2           0        0 LOG        all  --  any    any     anywhere anywhere      LOG level warning
3           0        0 ACCEPT     all  --  any    any     anywhere anywhere
4           0        0 DROP       all  --  any    any     anywhere anywhere
```

Tous les paquets devraient passer. On le confirme avec un ping sur l'hôte *babasse* :

```
root@box:~> ping -c 3 195.137.15.7
PING router.provider.net (195.137.15.7) from 10.0.3.2 : 56(84) bytes of data.
64 bytes from router.provider.net (195.137.15.7): icmp_seq=1 ttl=255 time=0.103 ms
64 bytes from router.provider.net (195.137.15.7): icmp_seq=2 ttl=255 time=0.082 ms
64 bytes from router.provider.net (195.137.15.7): icmp_seq=3 ttl=255 time=0.083 ms

--- router.provider.net ping statistics ---
3 packets transmitted, 3 received, 0% loss, time 2002ms
rtt min/avg/max/mdev = 0.082/0.089/0.103/0.012 ms
root@box:~>
```

Parfait, le routeur est vivant et opérationnel (bien sûr, il l'a été toute la journée).

> Une fois l'interface du pont activée, il faut compter dans les trente secondes pour que le pont soit complètement opérationnel. La phase d'apprentissage du pont est d'à peu près trente secondes. Pendant ce temps, le pont analyse les adresses MAC au contact de chaque port. L'auteur du code, Lennert, précise que ce point est susceptible d'amélioration un de ces jours. Pendant la période d'apprentissage, aucun paquet n'est transmis et aucun ping n'obtiendra de réponse. Il vaut mieux ne pas l'oublier.

## 5.3. Exemple de configuration

Cette partie est destinée à donner au lecteur quelques indications sur l'allure que doit avoir son système après avoir suivi les indications du HOWTO.

### 5.3.1. Configuration de l'interface

Résultat de la commande ifconfig :

```
root@bridge:~> ifconfig
br0       Link encap:Ethernet  HWaddr 00:04:75:81:D2:1D
          inet addr:10.0.3.129  Bcast:195.30.198.255  Mask:255.255.255.128
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:826 errors:0 dropped:0 overruns:0 frame:0
          TX packets:737 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:161180 (157.4 Kb)  TX bytes:66708 (65.1 Kb)

eth0      Link encap:Ethernet  HWaddr 00:04:75:81:ED:B7
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:5729 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3115 errors:0 dropped:0 overruns:0 carrier:656
          collisions:0 txqueuelen:100
          RX bytes:1922290 (1.8 Mb)  TX bytes:298837 (291.8 Kb)
          Interrupt:11 Base address:0xe400

eth1      Link encap:Ethernet  HWaddr 00:04:75:81:D2:1D
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:1 frame:0
          TX packets:243 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:342 (342.0 b)  TX bytes:48379 (47.2 Kb)
          Interrupt:7 Base address:0xe800

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:1034 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1034 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:82068 (80.1 Kb)  TX bytes:82068 (80.1 Kb)
```

### 5.3.2. Configuration du routage

Résultat de la commande route :

```
root@bridge:~> route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.3.129      0.0.0.0         255.255.255.128 U     0      0        0 br0
0.0.0.0         10.0.3.129      0.0.0.0         UG    0      0        0 br0
root@bridge:~>
```

### 5.3.3. Configuration d'iptables

On se reportera à la section Ping le, Max!.

## 5.4. Remarque

Il semble y avoir une anomalie dans le code br−nf :

```
From: Bart De Schuymer
Date: Sun, 1 Sep 2002 21:52:46 +0200
To: Nils Radtke
Subject: Re: Ethernet-Brigde-netfilter-HOWTO

Hello Nils,

[...]
Also, network packet filtering debugging is generally a bad idea with the
br-nf patch. It can gives a lot of false warnings (about bugs) in the logs.
[...]
```

NdT: l'auteur du message électronique signale que l'emploi des options de déverminage lorsque br−nf a été appliqué est susceptible de remplir les fichiers d'enregistrement de fausses alertes.

Pour ma part je n'ai jamais eu de fausse alerte dans mes logs. Peut−être que l'anomalie a été corrigée. Contacté sur ce point, Bart a répondu

```
From: Bart De Schuymer
Date: Mon, 2 Sep 2002 18:30:25 +0200
To: Nils Radtke
Subject: Re: Ethernet-Brigde-netfilter-HOWTO

On Monday 02 September 2002 00:39, Nils Radtke wrote:
> Will the revision of the nf-debug code in br-nf be subject of improvement?

I must admit I haven't been running any kernel with netfilter debugging
lately. It sure used to give false positives a few months ago (the bridge
mailing list has posts about that), I've been lacking time to see why and if
it is still the case. It's on my todo list.
[...]
```

NdT: l'auteur reconnaît ne pas avoir essayé la combinaison sus−citée depuis un moment. Il n'a pas eu le temps dernièrement de confirmer le problème ni de l'analyser. Il figure en tout cas dans son pense−bête.

À la date d'écriture de ce document (19/09/2002), je n'ai trouvé aucun message comme quoi l'erreur aurait disparu. Il est donc conseillé de garder un Sil sur la liste de diffusion du pontage Ethernet

# 6. Liens

L'auteur du document peut être contacté en anglais ou en allemand par courrier électronique. Voir la page web de l'auteur.

Merci d'envoyer vos commentaires, remarques, corrections concernant la version française de ce document à commentaires@traduc.org

## 6.1. Pontage Ethernet

- La liste de diffusion du pontage Ethernet.

- Utilitaires, correctifs, et cætera : page web du pontage Ethernet du noyau Linux.
- Le Guide pratique du pontage et de STP.
- Le pare−feu économique, par Shawn Grimes.

## 6.2. Thèmes voisins

- Filtrage au niveau des trames, Ethernet−Bridging−Tables :
    - ebtables chez sourceforge
    - page page de garde sourceforge d'ebtables
    - fonctionnalités d'ebtables
    - exemples pour ebtables : simples, évolués
    - documentation détaillée d'ebtables
    - guide du bricoleur ebtables
- extension IP du pontage Linux IP mode, LVS
- Linux et la haute disponibilité : Linux haute disponibilité
- Serveur virtuel Linux : LVS

## Notes

[1] Remarque : Cette entrée n'est disponible qu'avec un noyau modifié !