ProxyARP Subnetting HOWTO

Bob Edwards, <Robert.Edwards@anu.edu.au> Août 1997

Ce HOWTO explique l'utilisation d'un sous-réseau avec mandataire (*proxy*) ARP (protocole de résolution d'adresse), pour rendre visible un petit réseau de machines comme si ces machines étaient reliées directement au réseau principal. (traduction : Michel Billaud, <billaud@labri.u-bordeaux.fr>).

Contents

1 Introduction

Ce HOWTO explique l'utilisation d'un sous-réseau avec mandataire (*proxy*) ARP (protocole de résolution d'adresse), pour rendre visible un petit réseau de machines (nous l'appellerons *réseau 0* dans la suite) comme si ces machines étaient reliées directement au réseau principal (*réseau 1*).

Ceci n'a de sens que si les machines sont reliées par Ethernet ou autres dispositifs de type *ether* (autrement dit ça ne convient pas pour SLIP, PPP, CSLIP, etc.)

2 Remerciements

Ni ce document, ni mon implémentation du mandatement ARP, n'auraient été possibles sans l'aide :
- d'Andrew Tridgell, qui a implémenté sous Linux les options de sous-réseau pour *arp*, et qui m'a aidé personnellement à le faire marcher ;
- du mini-HOWTO *Proxy-ARP* écrit par Al LongYear ;
- du mini-HOWTO *Multiple-Ethernet* de Don Becker ;
- du code source et de la page de manuel de la commande `arp(8)`, de Fred N. van Kempen et Bernd Eckenfels.

3 Pourquoi utiliser un sous-réseau avec mandataire ARP ?

Les applications d'un sous-réseau avec mandataire ARP sont assez spécifiques.

Dans mon cas, j'avais une carte Ethernet sans-fil ISA 8 bits. Je voulais utiliser cette carte pour raccorder un certain nombre de machines. Après avoir écrit un pilote (*driver*) pour cette carte ISA (c'est le sujet d'un autre document), je pouvais l'utiliser dans une machine Linux. À partir de là, il était seulement nécessaire d'ajouter une seconde interface Ethernet à la machine Linux, et d'utiliser alors un mécanisme quelconque pour interconnecter les deux réseaux.

Dans la suite, j'appellerai *réseau 0* le réseau local Ethernet relié à la machine Linux par une carte Ethernet (clone NE-2000) sur *eth0*. Le *réseau 1* est le réseau principal connecté par la carte Ethernet sans-fil sur *eth1*. La *machine A* est la machine Linux avec ses deux interfaces. La *machine B* est une station quelconque du réseau 0, et la *machine C* est sur le réseau 1.

Normalement, pour réaliser l'interconnexion, on pourrait :

- utiliser le logiciel IP-Bridge (voir le *Bridge mini-HOWTO*) pour réaliser un pont entre les deux interfaces réseau. Malheureusement, la carte Ethernet sans-fil ne peut pas être mise en mode "promiscuous" (autrement dit elle ne peut pas voir tous les paquets circulant sur le réseau 1). C'est principalement à cause de la faible bande passante de la carte Ethernet sans-fil (2 Mbits/sec), ce qui implique que nous ne voulons pas supporter le trafic qui n'est pas destiné à une autre machine sans-fil - dans notre cas la machine A -, ou les diffusions générales (*broadcasts*). De plus, un pont charge assez lourdement le processeur.
- ou bien utiliser des sous-réseaux et un routeur pour transmettre les paquets entre les réseaux (voir le *IP-Subnetworking mini-HOWTO*). C'est une solution dépendante du protocole, bénéficiant du fait que le noyau Linux sait gérer les paquets IP (Internet Protocol), mais demandant du logiciel supplémentaire pour router d'autres protocoles (tels qu'AppleTalk). De plus, ceci nécessite d'allouer un nouveau numéro de sous-réseau IP, ce qui n'est pas toujours possible.

Dans mon cas, il n'était pas possible d'obtenir un nouveau numéro de sous-réseau, alors je voulais une solution qui permette aux machines du réseau 0 d'apparaître comme si elles étaient sur le réseau 1. C'est à cela que sert le mandatement ARP. D'autres solutions sont utilisées pour connecter d'autres protocoles (non-IP), comme *netatalk* pour le routage AppleTalk.

4 Comment marche le mandatement ARP d'un sous-réseau ?

En fait, le mandatement ARP sert uniquement à faire passer les paquets du réseau 1 vers le réseau 0. Pour faire passer les paquets dans l'autre sens, on emploie le routage IP normal.

Dans mon cas, le réseau 1 possède un masque de sous-réseau à 8 bits `255.255.255.0`. Pour le réseau 0, j'ai choisi un masque à 4 bits (`255.255.255.240`), qui permet d'avoir 14 noeuds IP sur le réseau 0 (2^4 = 16, moins deux pour l'adresse de réseau remplie de zéros et l'adresse de diffusion remplie de uns). Remarquez que toute taille de masque de sous-réseau convient, jusqu'à la taille - non comprise - du masque de l'autre réseau (dans notre cas : 2, 3, 4, 5, 6 ou 7 bits - pour un seul bit utilisez le mandatement ARP normal !).

Les numéros IP (au total 16) du réseau 0 apparaissent comme un sous-ensemble du réseau 1. Remarquez qu'il est très important, dans ce cas, de ne pas donner aux machines qui sont connectées directement au réseau 1 un numéro pris dans cet intervalle. Dans mon cas, j'ai "réservé" les numéros IP du réseau 1 qui se terminent par 64 à 79 pour le réseau 0. Les numéros IP qui se terminent par 64 et 79 ne peuvent pas être attribués à des machines : 79 est l'adresse de diffusion pour le réseau 0.

La machine A a deux numéros IP, l'un dans la plage d'adresses du réseau 0 pour sa vraie carte Ethernet (*eth0*), l'autre dans la plage du réseau 1 (mais en dehors de la plage du réseau 0) pour la carte Ethernet sans-fil (*eth1*).

Supposons que la machine C (du réseau 1) veuille envoyer un paquet à la machine B (du réseau 0). Comme le numéro IP de la machine B laisse croire à la machine C que B est sur le même réseau physique, la machine C va utiliser le protocole de résolution d'adresse ARP pour envoyer un message de diffusion sur le réseau 1, demandant à la machine qui a le numéro IP de B de répondre avec son adresse matérielle (adresse Ethernet ou MAC). La machine B ne verra pas cette requête, puisqu'en réalité elle n'est pas sur le réseau 1, mais la machine A, qui est sur les deux réseaux, la verra.

La première chose magique se produit maintenant, lorsque le code *arp* du noyau de la machine Linux (configurée en mandataire ARP avec sous-réseau) détermine que la requête est arrivée sur l'interface du réseau 1 (*eth1*), et que le numéro IP à résoudre est dans l'intervalle du réseau 0. La machine A envoie alors sa propre adresse matérielle (adresse Ethernet) à la machine C dans un paquet de réponse ARP.

La machine C met alors à jour son cache ARP en y ajoutant une entrée pour la machine B, mais avec l'adresse matérielle (Ethernet) de la machine A (la carte Ethernet sans-fil). La machine C peut alors envoyer le paquet pour B à cette adresse matérielle (Ethernet), et la machine A le reçoit.

La machine A remarque que l'adresse de destination IP n'est pas la sienne, mais celle de B. Le code de routage du noyau Linux de la machine A essaie alors de faire suivre ce paquet vers la machine B en cherchant dans ses tables de routage pour savoir quelle interface contient le numéro de réseau de B. Quoi qu'il en soit, le numéro IP de B est valide aussi bien pour le réseau 0 (*eth0*) que pour le réseau 1 (*eth1*).

C'est alors qu'un autre fait magique se produit : comme le masque de sous-réseau du réseau 0 a plus de bits à 1 (il est plus spécifique) que celui du réseau 1, le code de routage du noyau Linux va associer le numéro IP de B à l'interface du réseau 0, et ne va pas chercher à voir si il correspond à l'interface du réseau 1 (par laquelle le paquet est arrivé).

Maintenant la machine A doit trouver la “vraie” adresse matérielle (Ethernet) de la machine B (en supposant qu'elle ne l'a pas déjà dans le cache ARP). La machine A utilise une requête ARP, mais cette fois-ci le code `arp` du noyau Linux voit que la requête ne vient pas de l'interface du réseau 1 (*eth1*), et donc ne renvoie pas l'adresse du mandataire ARP. À la place, il envoie la requête ARP sur l'interface du réseau 0 (*eth0*), où la machine B le verra et répondra en donnant sa propre adresse matérielle (Ethernet). La machine A peut alors envoyer le paquet (qui venait de C) vers la machine B.

La machine B reçoit le paquet de C (qui est passé par A) et veut alors envoyer une réponse. Cette fois, B remarque que C est sur un sous-réseau différent (le masque de sous-réseau 255.255.255.240 exclut toutes les machines qui ne sont pas dans la plage d'adresses IP du réseau 0). La machine B est configurée avec une route par défaut vers l'adresse IP de A sur le réseau 0, et envoie le paquet à la machine A. Cette fois-ci, le code de routage du noyau Linux de A trouve que l'adresse IP de la destination (machine C) est sur le réseau 1, et envoie le paquet à la machine C par l'interface Ethernet *eth1*.

Des choses du même genre (mais moins compliquées) se produisent pour les paquets émis (ou reçus) par la machine A en direction (ou provenant) d'autres machines sur l'un ou l'autre des deux réseaux.

De la même façon, il est évident que si une autre machine D du réseau 0 envoie une requête ARP concernant B sur le réseau 0, la machine A recevra cette requête sur son interface du réseau 0 (*eth0*) et s'abstiendra d'y répondre, puisqu'elle n'est configurée comme mandataire que sur son interface du réseau 1 (*eth1*).

Remarquez aussi que les machines B, C (et D) n'ont de spécial à faire, du point de vue IP. Dans mon cas, il y a un mélange de SUN, de MAC et de PC sous Windows 95 sur le réseau 0, qui se connectent toutes au reste du monde à travers la machine Linux A.

Pour finir, notez qu'une fois que les adresses matérielles (Ethernet) ont été trouvées par chacune des machines A, B, C (et D), elles sont placées dans leur cache ARP, et que les paquets suivants sont tranférés sans surcoût dû à l'ARP. Normalement, les caches ARP suppriment les informations au bout de 5 minutes d'inactivité.

5 Installation du mandataire ARP de sous-réseau

J'ai installé le mandataire ARP du sous-réseau sur un noyau Linux version 2.0.30, mais il parait que le code fonctionne avec une version 1.2.x.

La première chose à noter est que le code ARP est en deux parties : une partie dans le noyau, qui envoie et reçoit les requêtes et les réponses ARP et met à jour le cache ARP, etc. ; l'autre partie est constituée de la commande `arp(8)` qui permet au super-utilisateur de mettre à jour manuellement le cache ARP, et à tout le monde de le consulter.

Le premier problème que j'ai eu était que la commande `arp(8)` de ma distribution Slackware 3.1 était antique (datée de 1994 !) et ne communiquait pas correctement du tout avec le code ARP du noyau ("`arp -a`" donnait un résultat étrange).

La commande `arp(8)` de "`net-tools-1.33a`", qui est disponible sur un grand nombre de sites, en particulier (d'après le README qui lui est joint) `ftp.linux.org.uk :/pub/linux/Networking/PROGRAMS/NetTools/`, fonctionne correctement, et contient de nouvelles pages de manuel `arp(8)` qui expliquent les choses bien mieux que les anciennes.

Une fois muni d'une commande `arp(8)` décente, il ne me restait plus qu'à modifier le seul fichier `/etc/rc.d/rc.inet1` (pour la Slackware - c'est probablement différent pour d'autres distributions). Tout d'abord, il nous faut changer l'adresse de diffusion, le numéro de réseau, et le masque de *eth0* :

```
NETMASK=255.255.255.240 # pour la partie hôte sur 4 bits
NETWORK=x.y.z.64        # notre nouveau réseau
                        #     (remplacez x.y.z par votre réseau)
BROADCAST=x.y.z.79      # pour moi.
```

Il faut ensuite ajouter une ligne pour configurer la seconde interface Ethernet (après les chargements de modules qui sont éventuellement nécessaires pour lancer le pilote) :

`/sbin/ifconfig eth1` *nom_sur_le_réseau_1* `broadcast` *x.y.z.255* `netmask 255.255.255.0`

Puis nous ajoutons une route pour la nouvelle interface :

`/sbin/route add -net` *x.y.z.0* `netmask 255.255.255.0`

Et vous aurez sans doute besoin de changer la passerelle par défaut pour utiliser celle du réseau 1.

Arrivés à ce point, nous pouvons ajouter la ligne pour le mandatement ARP :

```
/sbin/arp -i eth1 -Ds ${NETWORK} eth1 netmask ${NETMASK} pub
```

Ceci demande à ARP d'ajouter au cache une entrée statique ("`s`") pour le réseau `${NETWORK}`. Le "`-D`" dit d'utiliser la même adresse matérielle que l'interface *eth1* (la seconde interface), ce qui nous évite d'avoir à chercher l'adresse matérielle de *eth1* et de la coder directement dans la commande.

L'option `netmask` indique qu'il s'agit d'une entrée ARP concernant un *sous-réseau*, qui est constitué des numéros IP tels que

numéro_ip `& ${NETMASK} == ${NETWORK} & ${NETMASK}`

L'option `pub` demande de *publier* cette entrée ARP, c'est-à-dire qu'il s'agit d'*mandatement*, et qu'il faudra répondre au nom de ces numéros IP. L'option "`-i eth1`" précise qu'il ne faudra répondre qu'aux requêtes ARP arrivant par l'interface *eth1*.

Normalement, à ce point, si la machine est redémarrée, tous les machines du réseau 0 sembleront être sur le réseau 1. Vous pouvez vérifier que l'entrée de mandatement ARP de sous-réseau a été prise en compte correctement sur la machine A. Sur ma machine (j'ai changé les noms pour protéger les innocents) c'est :

```
#/sbin/arp -an
Address                 HWtype  HWaddress           Flags Mask            Iface
x.y.z.1                 ether   00:00:0C:13:6F:17   C     *               eth1
x.y.z.65                ether   00:40:05:49:77:01   C     *               eth0
x.y.z.67                ether   08:00:20:0B:79:47   C     *               eth0
x.y.z.5                 ether   00:00:3B:80:18:E5   C     *               eth1
x.y.z.64                ether   00:40:96:20:CD:D2   CMP   255.255.255.240 eth1
```

Vous pouvez aussi regarder le fichier `/proc/net/arp`, par exemple avec `cat(1)`.

La dernière ligne est l'entrée de mandatement pour le sous-réseau. Les indicateurs `CMP` révèlent qu'il s'agit d'une donnée statique (entrée Manuellement) qui doit être Publiée. Elle ne servira qu'à répondre qu'aux requêtes ARP qui arrivent par *eth1* et pour lesquelles le numéro IP, une fois masqué, correspond au numéro de réseau (également masqué). Remarquez que la commande `arp(8)` a trouvé automatiquement l'adresse matérielle de *eth1* et l'a employée comme adresse à utiliser (à cause de l'option "`-Ds`").

De la même façon il est probablement prudent de vérifier que la table de routage a été remplie correctement. Voici la mienne (ici aussi, les noms ont été changés pour protéger les innocents) :

```
#/bin/netstat -rn
Kernel routing table
Destination     Gateway         Genmask         Flags Metric Ref Use    Iface
x.y.z.64        0.0.0.0         255.255.255.240 U     0      0        71 eth0
x.y.z.0         0.0.0.0         255.255.255.0   U     0      0       389 eth1
127.0.0.0       0.0.0.0         255.0.0.0       U     0      0         7 lo
0.0.0.0         x.y.z.1         0.0.0.0         UG    1      0       573 eth1
```

Vous pouvez aussi regarder le contenu du fichier `/proc/net/route` (par exemple avec `cat(1)`).

Remarquez que la première entrée concerne un sous-ensemble de la seconde, mais la table de routage les classe dans l'ordre des masques, et donc l'entrée *eth0* sera testée avant celle de *eth1*.

6 Autres alternatives au mandatement ARP de sous-réseau

Dans la même situation il y a d'autres possibilités que le mandatement ARP de sous-réseau et celles que j'ai déjà mentionnées (utilisation d'un pont et routage direct) :

- "*Masquerading IP*" (voir le *IP-Masquerade mini-HOWTO*), dans lequel le réseau 0 est "cachée" du reste du monde derrière la machine A. Quand les machines du réseau 0 tentent de se connecter à l'extérieur à travers la machine A, celle-ci modifie l'adresse d'origine et le numéro de port des paquets pour qu'ils aient l'air d'avoir été envoyés par elle-même, plutôt que par la machine cachée du réseau 0. C'est une solution élégante, bien qu'elle empêche toute machine du réseau 1 d'établir une connexion vers une machine du réseau 0, puisque les machines du réseau 0 n'existent pas en dehors de celui-ci. Ceci améliore efficacement la sécurité des machines du réseau 0, mais cela signifie aussi que les serveurs du réseau 1 ne peuvent pas vérifier l'identité des clients du réseau 0 en se basant sur leur numéros IP (les serveurs NFS, par exemple, utilisent les noms IP pour contrôler l'accès aux systèmes de fichiers).
- Une autre possibilité serait un *tunnel IP sur IP*, ce qui n'est pas supporté par toutes les plateformes (comme les Macs et les machines sous Windows), et je n'ai donc pas opté pour cette solution.
- Utiliser le mandatement ARP sans sous-réseau. C'est tout à fait possible, cela signifie simplement qu'il faut créer une entrée individuelle pour chaque machine du réseau 0, au lieu d'une seule entrée pour toutes les machines (présentes et futures) du réseau 0.
- Il se peut que l'*aliasing IP* puisse être utilisé ici (NdT : francheement ça m'étonnerait), mais je n'ai pas du tout exploré cette voie.

7 Autres applications du mandatement ARP de sous-réseau

Je ne connais qu'une autre application du mandatement ARP de sous-réseau, ici à l'Australian National University (ANU). C'est celle pour laquelle Andrew Tridgell a écrit, à l'origine, les extensions du mandatement ARP pour les sous-réseaux. Quoiqu'il en soit, Andrew m'informe qu'il y a, de fait, plusieurs autres sites dans le monde qui l'utilisent également (je n'ai aucun détail).

Á l'ANU, l'autre application concerne un laboratoire d'enseignement qui sert à apprendre aux étudiants comment configurer des machines pour utiliser TCP/IP, y compris pour configurer la passerelle. Le réseau utilisé est un réseau de classe C, et Andrew avait besoin de le découper en sous-réseaux pour des raisons de sécurité, de contrôle du trafic et la raison pédagogique mentionnée plus haut. Il l'a fait en utilisant le mandatement ARP, et a alors décidé qu'une seule entrée dans le cache ARP pour tout le sous-réseau serait plus rapide et plus propre qu'une pour chaque machine du sous-réseau. Et voilà. Mandatement ARP de sous-réseau !

Les corrections et les suggestions sont les bienvenues !

8 Copyright