

Fonctionnement du client du « tunnel broker » sur Unix

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 31 Octobre 2007. Dernière mise à jour le 25
Novembre 2009

<http://www.bortzmeyer.org/tunnel-broker.html>

Il existe d'innombrables solutions pour connecter un îlot de machines IPv6 au reste de l'Internet IPv6, même lorsque les fournisseurs sur le trajet sont purement IPv4. Cet article décrit le fonctionnement concret d'une de ces solutions, le "*tunnel broker*".

Le principe du "*tunnel broker*" est le suivant : le routeur IPv6 du réseau local négocie avec un serveur de tunnels qui lui indique le préfixe délégué et d'autres caractéristiques du tunnel. Ledit routeur va ensuite encapsuler ses paquets IPv6 dans de l'IPv4 à destination dudit serveur de tunnels, qui sert également de routeur.

Le serveur n'existe apparemment pas en logiciel libre mais il y a plusieurs clients comme tspc ou gw6. Ici, le routeur IPv6 du réseau local est une Debian et se sert du logiciel tspc (<http://packages.debian.org/stable/tspc>). Avant de configurer le client, il faut trouver un serveur, il en existe un gratuit à Freenet6 (<http://www.freenet6.net/>) (expliqué plus loin) mais, comme le réseau étudié ici s'inscrivait dans un projet auquel Renater participe, on utilise d'abord le tunnel broker de Renater (<http://www.renater.fr/spip.php?article328>). Il faut d'abord obtenir un compte auprès de Renater (si on veut un « préfixe de site », un /48, les /64 sont, eux, sans formalité), puis installer tspc (`aptitudeinstalltspc`).

La configuration se fait dans le fichier `/etc/tspc.conf`. Voyons les points saillants :

```
# authentication method:
# Valeur par défaut...
auth_method=any

# IPv4 address of the client for its tunnel endpoint:
# Valeur par défaut...
client_v4=auto

# user identification:
# Le compte obtenu auprès du serveur de tunnels
userid=bortzmeyer
```

```
# password:
# Non, je ne le montre pas :-)

# Name of the script:
# Valeur par défaut...
template=setup

# 'server' is the tunnel broker identifier
# C'est le serveur indiqué par Renater, réservé à leurs clients ou
# partenaires. Notez le numéro de port, un autre serveur de tunnels
# existe sur cette machine avec un port différent (selon qu'on a un
# compte ou bien qu'on bénéficie du service gratuit, lisez bien la
# documentation de votre fournisseur).
server=tunnel-broker.renater.fr:3654

# retry_delay=time
# Valeur par défaut. Testé, ce service fonctionne bien, même lorsque
# la liaison IPv4 sous-jacente est fragile.
retry_delay=30

# Tunnel encapsulation mode:
# Valeur par défaut...
tunnel_mode=v6anyv4

# Tunnel Interface name:
# Valeur par défaut...
if_tunnel_v6v4=sit1
if_tunnel_v6udpv4=tun

# proxy_client indicates that this client acts as a TSP proxy for
# Valeur par défaut...
proxy_client=no

# Keepalive for v6udpv4 tunnels:
# Valeur par défaut...
keepalive=yes
keepalive_interval=30

# Logging facility uses syslog on Unix platforms
# Ne semble pas fonctionner, tout est apparemment enregistré dans /var/log/tspc.log
syslog_facility=DAEMON
syslog_level=ERR

# Router configuration
# On est routeur pour tout le réseau local, donc :
host_type=router

# prefixlen specifies the required prefix length for the TSP client
# Pour un routeur, c'est la valeur normale (cf. RFC 3177)
prefixlen=48

# if_prefix is the name of the OS interface that will be configured
# with the first /64 of the received prefix from the broker and the
# router advertisement daemon is started to advertise that prefix
# on the if_prefix interface.
#if_prefix=eth0
if_prefix=dummy0
# Une mention particulière ici. Contrairement à ce qu'on pourrait
# croire, cette mention est obligatoire. Comme nous préférons
# configurer à la main un certain nombre de paramètres, nous indiquons
# quand même une interface, mais "bidon", pour éviter que tspc ne
# modifie les réglages de notre vraie interface, eth0. L'interface
# "bidon" se crée sur Linux avec "modprobe dummy; ifconfig dummy0 up".

# For reverse DNS delegation of the prefix, define the following:
# Pas utilisé.
#dns_server=
```

Une fois le client lancé (c'est fait automatiquement au démarrage, sinon `/etc/init.d/tspcstart`), on peut voir les interfaces, `dummy0` a été configuré par `tspc`, à partir des informations données par le serveur de tunnels, `eth0` a été configurée à la main (via le fichier standard Debian, `/etc/network/interfaces`):

```
dummy0    Lien encap:Ethernet HWaddr 32:A5:E1:71:37:A3
          adr inet6: 2001:660:f108::1/64 Scope:Global
          adr inet6: fe80::30a5:eff:fe71:37a3/64 Scope:Lien
          UP BROADCAST RUNNING NOARP MTU:1500 Metric:1
          ...

eth0      Lien encap:Ethernet HWaddr 00:0C:6E:6E:E5:D4
          inet adr:82.151.64.129 Bcast:82.151.64.255 Masque:255.255.255.128
          adr inet6: 2001:660:f108::1/64 Scope:Global
          adr inet6: fe80::20c:6eff:fe6e:e5d4/64 Scope:Lien
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          ...

sit1      Lien encap:IPv6-dans-IPv4
          adr inet6: fe80::5297:4081/64 Scope:Lien
          adr inet6: 2001:660:f001::d/128 Scope:Global
          UP POINTOPOINT RUNNING NOARP MTU:1280 Metric:1
          RX packets:1548 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1748 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:0
          RX bytes:520111 (507.9 KiB) TX bytes:312956 (305.6 KiB)
```

(L'adresse IPv6 est envoyée par le serveur de tunnels et, avec le service `Freenet6`, peut changer de temps en temps.) Et le routage IPv6 se passe bien, aussi bien pour le routeur, que pour les autres machines du réseau local (testé avec des adresses IPv6 statiques mais on aurait pu utiliser RA, "*Router Advertisement*").

Le protocole utilisé entre client et serveur de tunnels a fait l'objet d'une normalisation dans le RFC 5572¹. Le principe est simple : le client envoie ses demandes en XML et le serveur lui renvoie dans le même format les paramètres choisis (il n'y a pas de négociation, c'est à prendre ou à laisser). Voici un exemple de sessions :

```
# /usr/sbin/tspc -vvvvv
Connecting to server with tcp
Using TSP protocol version 2.0.0
Establishing connection with tunnel broker...
Getting capabilities from server
Connection established
Authenticating bortzmeyer
Using authentication mechanism DIGEST-MD5
Authentication success
Asking for a tunnel
sent: Content-length: 252
<tunnel action="create" type="v6v4" proxy="no">
  <client>
    <address type="ipv4">192.134.7.249</address>
  <keepalive interval="30"><address type="ipv6">:::</address></keepalive <router>
    <prefix length="48"/>
  </router>
</client>
</tunnel>
...
```

¹Pour voir le RFC de numéro NNN, <http://www.ietf.org/rfc/rfcNNN.txt>, par exemple <http://www.ietf.org/rfc/rfc5572.txt>

Et si cela ne marche pas ? Le client affiche le code d'erreur numérique à trois chiffres et il n'y a plus qu'à regarder dans le document cité plus haut ce qu'ils signifient. Par exemple, 303 est "*Unsupported tunnel type*" et se produit lorsqu'on demande à un "*tunnel broker*" un service qu'il ne fournit pas.

Pour Freenet6, on va utiliser une machine Gentoo hébergée aux États-Unis (le routeur de la société mère de Freenet6, Hexago, est au Canada). Sur une Gentoo, il n'y a pas de paquetage tspanc tout fait, il faut compiler le client. On obtient le source en s'inscrivant sur <http://www.go6.net/4105/freenet.asp>. Attention, c'est gratuit mais il faut remplir un formulaire où les cases « Je veux recevoir du spam » sont cochées par défaut. Ne pas oublier de les décocher ! Une fois obtenu le source, l'installation (tel que documentée dans le fichier joint, HEX_DC_0005_Gateway6_Client_Guide.pdf) est simple :

```
% cd tspanc-advanced
% make all target=linux
root# make install installdir=/usr/local/gw6c target=linux
```

On doit alors configurer le fichier `gw6c.conf`. Les points saillants sont :

```
# Obtenus en s'inscrivant
userid=bortzmeyer
passwd=SECRET
...
server=broker.freenet6.net
...
auth_method=any
...
# Il est possible qu'on puisse obtenir un /48 mais je n'ai pas essayé
prefixlen=64
...
# La délégation DNS ne fonctionne pas pour moi. Peut-être n'est-elle
# accessible que si on a au moins un /48 ou bien que si on est un
# client payant ?
dns_server=ns1.bortzmeyer.org:ns2.bortzmeyer.org
```

Les tunnels sont souvent plus lents que le chemin direct (d'avantage de routeurs, des passerelles entre v4 et v6 surchargées), moins fiables, sans compter les problèmes de MTU. On va donc demander à privilégier l'accès IPv4, ce qui se fait à l'aide du fichier `/etc/gai.conf` :

```
# For testing purposes, always use IPv6 for AFNIC
precedence 2001:660:3003::/48 200
# Otherwise, always prefer IPv4
precedence ::ffff:0:0/96 100
```

Ces règles disent que les adresses IPv6 doivent être privilégiées pour l'accès à l'AFNIC (préférence 200) mais, dans tous les autres cas, on préfère IPv4 (préférence inférieure, 100). Testons avec telnet :

```
% telnet www.afnic.fr
Trying 2001:660:3003::4:20...
Trying 192.134.4.20...
...
% telnet www.kame.net
Trying 203.178.141.194...
Trying 2001:200:0:8002:203:47ff:fea5:3085...
```

Pour ces deux machines double-pile (v4 et v6), on obtient bien l'ordre attendu.

Pour s'assurer que le démon tpsc soit bien lancé au démarrage, on crée un fichier `/etc/init.d/freenet6`:

```
#!/sbin/runscript
opts="start stop check"

depend() {
    use net
}

start() {
    ebegin "Starting IPv6 connection through Freenet6"
    (
        modprobe ipv6
        /usr/local/gw6c/bin/gw6c -f /usr/local/gw6c/etc/gw6c.conf
    )
    eend $?
}

stop() {
    ebegin "Stopping IPv6 connection"
    start-stop-daemon --stop --quiet --exec /usr/local/gw6c/bin/gw6c
    eend $?
}

check() {
    ebegin "Performing Consistency Check"
    killall -IOT gw6c &>/dev/null
    eend $?
}
```

et on l'installe avec :

```
# rc-update add freenet6 default
```

Merci à Simon Muyal pour ses explications et à Mohsen Souissi pour l'aide.