RFC 7706 : Decreasing Access Time to Root Servers by Running One on Loopback

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 26 novembre 2015

Date de publication du RFC : Novembre 2015

https://www.bortzmeyer.org/7706.html

Toute résolution DNS commence par la racine (de l'arbre des noms de domaine). Bien sûr, la mémorisation (la mise en cache) des réponses fait qu'on n'a pas besoin tout le temps de contacter un serveur racine. Mais c'est quand même fréquent et les performances de la racine sont donc cruciales. L'idée documentée dans ce RFC est donc d'avoir en local un serveur esclave de la racine, copiant celle-ci et permettant donc de répondre localement aux requêtes. Notez que ce RFC a depuis été remplacé par le RFC 8806 [1].

Le problème est particulièrement important pour les noms qui n'existent **pas**. Si les TLD existants comme `.com` ou `.fr` vont vite se retrouver dans la mémoire (le cache) du résolveur DNS, les fautes de frappe ou autres cas où un TLD n'existe pas vont nécessiter la plupart du temps un aller-retour jusqu'au serveur racine le plus proche. Les réponses négatives seront également mémorisées mais 1) il y a davantage de noms non existants que de noms existants 2) le TTL est plus court (actuellement deux fois plus court). Ces noms non existants représentent ainsi la majorité du trafic de la racine.

Bien qu'il existe aujourd'hui des centaines de sites dans le monde où se trouve une instance d'un serveur racine, ce nombre reste faible par rapport au nombre total de réseaux connectés à l'Internet. Dans certains endroits de la planète, le serveur racine le plus proche est assez lointain. Voici les RTT en millisecondes avec les serveurs racine observés depuis un réseau tunisien (notez les deux serveurs qui répondent bien plus vite que les autres, car ils ont une instance à Tunis) :

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc8806.txt`

```
% check-soa -4 -i .
a.root-servers.net.
198.41.0.4: OK: 2015112501 (54 ms)
b.root-servers.net.
192.228.79.201: OK: 2015112501 (236 ms)
c.root-servers.net.
192.33.4.12: OK: 2015112501 (62 ms)
d.root-servers.net.
199.7.91.13: OK: 2015112501 (23 ms)
e.root-servers.net.
192.203.230.10: OK: 2015112501 (18 ms)
f.root-servers.net.
192.5.5.241: OK: 2015112501 (69 ms)
g.root-servers.net.
192.112.36.4: OK: 2015112501 (62 ms)
h.root-servers.net.
128.63.2.53: OK: 2015112501 (153 ms)
i.root-servers.net.
192.36.148.17: OK: 2015112501 (67 ms)
j.root-servers.net.
192.58.128.30: OK: 2015112501 (55 ms)
k.root-servers.net.
193.0.14.129: OK: 2015112501 (72 ms)
l.root-servers.net.
199.7.83.42: ERROR: Timeout
m.root-servers.net.
202.12.27.33: OK: 2015112501 (79 ms)
```

Ces délais peuvent sembler courts mais ils ne forment qu'une partie du travail de résolution, il est donc légitime de vouloir les réduire encore.

En outre, ces requêtes à la racine peuvent être observées, que ce soit par les opérateurs de serveurs racine, ou par des tiers sur le projet, ce qui n'est pas forcément souhaitable, question vie privée (cf. RFC 7626).

Donc, l'idée de base de ce RFC est de :
— Mettre un serveur esclave de la racine sur sa machine, écoutant sur l'adresse locale ("*loopback*"),
— Configurer le résolveur pour interroger d'abord ce serveur.
Cette idée est **documentée** dans ce RFC mais n'est **pas encouragée** (c'est un très vieux débat, dont j'avais déjà parlé <https://www.bortzmeyer.org/slaving-the-root.html>). En effet, cela ajoute un composant à la résolution (le serveur local faisant autorité pour la racine), composant peu ou pas géré et qui peut défaillir, entrainant ainsi des problèmes graves et difficiles à déboguer. Mais pourquoi documenter une idée qui n'est pas une bonne idée ? Parce que des gens le font déjà et qu'il vaut mieux documenter cette pratique, et en limiter les plus mauvais effets. C'est pour cela, par exemple, que notre RFC demande que le serveur local n'écoute que sur les adresses locales, pour limiter les conséquences d'une éventuelle défaillance à une seule machine.

Dans tous les cas, le RFC recommande que la configuration décrite ici ne soit **pas** celle par défaut : l'utilisateur doit l'activer explicitement (et en supporter les conséquences).

Pas découragé ? Vous voulez encore le faire ? Alors, les détails pratiques. D'abord (section 2 du RFC), les pré-requis. DNSSEC est indispensable (pour éviter de se faire refiler un faux fichier de zone par de faux serveurs racine). Ensuite (section 3), vous mettez un serveur faisant autorité (par exemple NSD ou Knot) qui écoute sur une des adresses locales (en `127.0.0.0/8`, IPv6 est moins pratique car il ne fournit paradoxalement qu'une seule adresse locale à la machine) et qui est esclave des serveurs racine. À noter que votre serveur, n'étant pas connu des serveurs racine, ne recevra pas les notifications (RFC

1996) et sera donc parfois un peu en retard sur la vraie racine (ce qui n'est pas très grave, elle bouge peu).

Il est important de lister plusieurs serveurs maîtres dans sa configuration. En effet, si la mise à jour de la racine dans votre serveur esclave échoue, ce sera catastrophique (signatures DNSSEC expirées, etc) et cette configuration locale, contrairement à la « vraie » racine, n'a aucune redondance. (Une autre raison pour laquelle ce n'est pas une idée géniale.) Quels serveurs maîtres indiquer ? Certains serveurs racine permettent le transfert de zone (RFC 5936) mais ce n'est jamais officiel, ils peuvent cesser à tout moment (l'annexe A du RFC donne une liste et discute de ce choix). Une raison de plus de se méfier.

Il est donc important d'avoir un mécanisme de supervision, pour être prévenu si quelque chose échoue. On peut par exemple interroger le numéro de série dans l'enregistrement SOA de la racine et vérifier qu'il change.

Ensuite, une fois ce serveur faisant autorité configuré, il ne reste qu'à indiquer à un résolveur (comme Unbound) de l'utiliser (section 4 du RFC).

Voici un exemple, pris dans l'annexe B du RFC et testé. J'ai choisi l'exemple avec NSD et Unbound mais il y en a d'autres dans le RFC. D'abord, la configuration de NSD (notez la longue liste de maîtres, pour maximiser les chances que l'un d'eux fonctionne ; notez aussi l'adresse "*loopback*" choisie, `127.12.12.12`) :

```
# RFC 7706
server:
       ip-address: 127.12.12.12
zone:
       name: "."
       request-xfr: 192.228.79.201 NOKEY # b.root-servers.net
       request-xfr: 192.33.4.12 NOKEY    # c.root-servers.net
       request-xfr: 192.5.5.241 NOKEY    # f.root-servers.net
       request-xfr: 192.112.36.4 NOKEY   # g.root-servers.net
       request-xfr: 193.0.14.129 NOKEY   # k.root-servers.net
       request-xfr: 192.0.47.132 NOKEY   # xfr.cjr.dns.icann.org
       request-xfr: 192.0.32.132 NOKEY   # xfr.lax.dns.icann.org
       request-xfr: 2001:500:84::b NOKEY # b.root-servers.net
       request-xfr: 2001:500:2f::f NOKEY # f.root-servers.net
       request-xfr: 2001:7fd::1 NOKEY    # k.root-servers.net
       request-xfr: 2620:0:2830:202::132 NOKEY  # xfr.cjr.dns.icann.org
       request-xfr: 2620:0:2d0:202::132 NOKEY  # xfr.lax.dns.icann.org
```

Le démarrage de NSD (notez qu'il faut patienter un peu la première fois, le temps que le premier transfert de zone se passe) :

```
Nov 25 19:50:43 machine-locale nsd[2154]: [2015-11-25 19:50:43.791] nsd[2175]: notice: nsd started (NSD 4.1.2), p
Nov 25 19:50:56 machine-locale nsd[2154]: zone . serial 0 is updated to 2015112501.
Nov 25 19:50:56 machine-locale nsd[2154]: [2015-11-25 19:50:56.166] nsd[2154]: info: zone . serial 0 is updated t
```

C'est bon, on a transféré la zone. Testons (notez le bit AA - "*Authoritative Answer*" - dans la réponse) :

```
% dig  @127.12.12.12 SOA .
...
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 61984
;; flags: qr aa rd; QUERY: 1, ANSWER: 2, AUTHORITY: 14, ADDITIONAL: 25
...
;; ANSWER SECTION:
. 86400 IN SOA a.root-servers.net. nstld.verisign-grs.com. (
2015112501 ; serial
...
```

C'est bon.

Maintenant, la configuration d'Unbound (prise dans le RFC) :

```
server:
    # RFC 7706
    do-not-query-localhost: no

# RFC 7706
# Requires a slave auth. running (normally, nsd)
stub-zone:
       name: "."
       stub-prime: no
       stub-addr: 127.12.12.12
```

Et le test :

```
% dig www.cnam.fr
...
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 23562
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 3, ADDITIONAL: 3
...
;; ANSWER SECTION:
www.cnam.fr. 0 IN CNAME sarek.cnam.fr.
sarek.cnam.fr. 86400 IN A 163.173.128.52
```

Ça a marché. Avec tcpdump, on voit le trafic (faible, en raison du cache) vers le serveur racine local :

```
08:37:24.869569 IP (tos 0x0, ttl 64, id 7734, offset 0, flags [none], proto UDP (17), length 76)
    127.0.0.1.10908 > 127.12.12.12.53: 42633% [1au] A? www.tunisair.com.tn. (48)
08:37:24.869671 IP (tos 0x0, ttl 64, id 12657, offset 0, flags [none], proto UDP (17), length 685)
    127.12.12.12.53 > 127.0.0.1.10908: 42633- 0/8/13 (657)
08:38:04.734565 IP (tos 0x0, ttl 64, id 12887, offset 0, flags [none], proto UDP (17), length 71)
    127.0.0.1.45221 > 127.12.12.12.53: 25787% [1au] A? www.edreams.es. (43)
08:38:04.734645 IP (tos 0x0, ttl 64, id 19270, offset 0, flags [none], proto UDP (17), length 749)
    127.12.12.12.53 > 127.0.0.1.45221: 25787- 0/10/14 (721)
08:38:04.734867 IP (tos 0x0, ttl 64, id 12888, offset 0, flags [none], proto UDP (17), length 70)
    127.0.0.1.40575 > 127.12.12.12.53: 61589% [1au] AAAA? ns-ext.nic.cl. (42)
08:38:04.734932 IP (tos 0x0, ttl 64, id 19271, offset 0, flags [none], proto UDP (17), length 622)
    127.12.12.12.53 > 127.0.0.1.40575: 61589- 0/8/11 (594)
08:44:00.366698 IP (tos 0x0, ttl 64, id 27563, offset 0, flags [none], proto UDP (17), length 62)
    127.0.0.1.22009 > 127.12.12.12.53: 30565% [1au] A? po.st. (34)
08:44:00.389126 IP (tos 0x0, ttl 64, id 34039, offset 0, flags [none], proto UDP (17), length 404)
    127.12.12.12.53 > 127.0.0.1.22009: 30565- 0/6/5 (376)
08:44:01.229635 IP (tos 0x0, ttl 64, id 27693, offset 0, flags [none], proto UDP (17), length 69)
    127.0.0.1.65173 > 127.12.12.12.53: 32911% [1au] AAAA? ns3.perf1.eu. (41)
08:44:01.229705 IP (tos 0x0, ttl 64, id 34207, offset 0, flags [none], proto UDP (17), length 560)
    127.12.12.12.53 > 127.0.0.1.65173: 32911- 0/8/10 (532)
```

Pour BIND on peut suivre le ticket #33 <https://gitlab.isc.org/isc-projects/bind9/issues/33>.

À noter qu'il existe un brevet futile (comme tous les brevets...) de Verisign sur cette technique : déclaration #2539 à l'IETF <https://datatracker.ietf.org/ipr/2539/>.