NSD, un autre serveur de noms pour servir ses zones

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 8 mai 2011

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

Pour la résilience de l'Internet, sa capacité à survivre aux pannes et aux attaques, il est essentiel qu'il existe un certain pluralisme ; que, pour la plupart des fonctions importantes, il existe plusieurs logiciels. Ainsi, si tous les routeurs de l'Internet étaient des Cisco, une bogue dans le code d'IOS, comme celle de l'attribut BGP 99 <https://www.bortzmeyer.org/bgp-attribut-99.html>, pourrait arrêter tout l'Internet. Si tous les serveurs DNS étaient des BIND, une bogue comme celle permettant d'arrêter un serveur par une mise à jour dynamique <https://www.bortzmeyer.org/bind-dos-update.html>, et le DNS s'arrête. Il faut donc qu'il y ait des alternatives, non pas qu'elles aient forcément moins de bogues, mais simplement parce qu'elles ne seront pas publiées au même moment.

Cela semble du bon sens mais, en pratique, on constate que la plupart des techniciens se simplifient la vie en n'utilisant qu'un seul logiciel, en général le plus répandu. C'est d'ailleurs un bon test des compétences d'un administrateur systèmes : demandez-lui pourquoi il utilise tel logiciel. S'il répond par une formule passe-partout comme « parce que c'est le standard de l'industrie », c'est probablement qu'il n'a même pas envisagé de décider lui-même, et qu'il n'a sans doute pas les compétences nécessaires pour évaluer les différents logiciels existants.

Ainsi, dans le contexte du DNS, la plupart des administrateurs système Unix qui installent un serveur DNS mettent BIND sans réflechir, non pas qu'ils aient étudié plusieurs serveurs et choisi celui-ci (BIND a des tas de qualités, notamment le record de la richesse fonctionnelle), mais simplement parce que l'idée du choix ne traverse même pas leur cerveau de certifié.

Pourquoi envisager NSD comme alternative ? C'est un logiciel pour serveur **faisant autorité**, c'est-à-dire le serveur qui distribue directement le contenu des zones (par opposition aux **résolveurs** qui relaient les requêtes du client final vers les serveurs faisant autorité). Il est utilisé par plusieurs gros TLD comme `.FR` ou `.DE`, ainsi que par la racine (à chaque fois, sur une partie seulement des serveurs, pour les raisons indiquées plus haut). NSD a fait le choix de ne pas en faire trop, afin de rester simple et rapide (tous les tests <https://www.bortzmeyer.org/performances-serveur-dns.html> indiquent des performances deux à trois fois meilleures que celles de BIND). Le code source est donc logiquement plus court :

```
% sloccount nsd-3.2.5
...
Total Physical Source Lines of Code (SLOC)                = 25,542

% sloccount bind-9.7.3
...
Total Physical Source Lines of Code (SLOC)                = 317,497
```

Bien sûr que la comparaison est injuste, puisque BIND fait beaucoup plus de choses. Mais c'est justement l'un de mes arguments, plus le code est long (et BIND 9 ne permet pas d'exclure du code à la compilation) et plus il y sans doute de bogues et de failles de sécurité dedans.

Comment configure t-on NSD ? D'abord, un fichier de configuration, `nsd.conf` :

```
server:
     ip-address: 2001:db8:1::53
     # Toutes les valeurs peuvent avoir une configuration par défaut

...

# Une zone DNS, "bortzmeyer.42" :
zone:
        name: "bortzmeyer.42"
        zonefile: "primary/bortzmeyer.42"
        notify: 2001:db8:43c::1 NOKEY
        provide-xfr: 2001:db8:43c::1 NOKEY
        outgoing-interface: 2001:db8:1::53
```

Ici, on indique à NSD qu'il est serveur **maître** pour `bortzmeyer.42` et qu'il est donc la source authentique des données. Celles-ci seront trouvées dans le fichier `primary/bortzmeyer.42`, qui suit la syntaxe standard du RFC 1035[1], section 5 (la même syntaxe que BIND). Ce serveur a un **esclave**, `2001:db8:43c::1` et on notifiera cet esclave en cas de mise à jour de la zone (RFC 1996), tout en l'autorisant à transférer la zone depuis chez nous (RFC 5936).

Pour atteindre ses performances élevées, NSD calcule beaucoup de choses à l'avance (ce qui augmente sa consommation mémoire : on n'a rien sans rien). Même si les versions actuelles de NSD sont moins radicales que la version 1 (qui stockait en mêmoire les paquets correspondant à n'importe quelle question possible, il n'y avait plus qu'à en choisir une et à faire un `write()`), NSD repose beaucoup sur ces pré-calculs. Il est donc nécessaire de compiler la zone d'abord :

```
% sudo nsdc rebuild
zonec: reading zone "bortzmeyer.42".
zonec: processed 8 RRs in "bortzmeyer.42".
zonec: done with 0 errors.
```

Le résultat est stocké sur disque (l'endroit dépend des options, disons `/var/lib/nsd/nsd.db`) et il faut dire à NSD de le recharger :

```
% sudo nsdc reload
```

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc1035.txt`

Et voilà, NSD commence à servir la zone.

Si le serveur NSD est un esclave, on doit indiquer quel est le maître avec la directive `request-xfr` :

```
request-xfr: AXFR 2001:db8:1::53 NOKEY
allow-notify: 2001:db8:1::53  NOKEY
```

Il est prudent de forcer l'adresse IP utilisée pour les requêtes sortantes, pour que les demandes de transfert viennent bien de l'adresse qu'attend le maître :

```
# À mettre pour chaque zone:
outgoing-interface: 2001:db8:43c::1
```

Une fois le transfert de la zone fait (NSD enregistrera dans le journal quelque chose comme `nsd[27250]: info: Zone bortzmeyer.42 serial 0 is updated to 2011050800`), NSD modifiera sa base des réponses automatiquement. Notez que NSD, dans sa version 3, peut utiliser les transferts incrémentaux (IXFR, RFC 1995) lorsqu'il est esclave, mais ne peut pas les générer lorsqu'il est maître. Dans ce cas, il faut transférer la zone entière.

On le voit, il est fréquent qu'il y ait des répétitions dans la configuration de NSD. Par exemple, si on est esclave et qu'on sert dix zones ayant toutes le même maître, on va faire beaucoup de copier/coller. Fidèle au principe de ne pas multiplier le code, NSD ne fournit pas de mécanisme de macros. Le mieux est d'utiliser un système externe comme cpp.

Si on compile NSD soi-même, un certain nombre d'options passées à `./configure` permettent de retirer complètement du code qu'on n'utilise pas, ce qui est probablement une bonne chose pour la sécurité. Faites un `./configure --help` pour voir ces options.

La version 4 de NSD est actuellement en cours de développement. Vous pouvez en savoir plus avec l'exposé fait au RIPE 62 <http://ripe62.ripe.net/presentations/146-NSD4-RIPE62-03.pdf>. Sinon, il existe un bon article en français sur NSD <http://blog.philpep.org/post/nsd-pour-remplacer-

Et, en guise de conclusion, une devise pour l'administrateur systèmes, « Nous vous rappelons qu'il existe d'autres possibilités. »