RFC 9199 : Considerations for Large Authoritative DNS Servers Operators

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 22 mars 2022

Date de publication du RFC : Mars 2022

https://www.bortzmeyer.org/9199.html

On sait que le DNS est un des services d'infrastructure les plus essentiels de l'Internet. Il est, par exemple, crucial que les serveurs DNS faisant autorité <https://www.bortzmeyer.org/serveur-dns-faisant-au
html> soient correctement gérés, et robustes face aux problèmes qui peuvent survenir. Ce RFC donne un certain nombre de conseils (juste des conseils, il ne s'agit pas d'une norme) aux opérateurs de ces serveurs.

Il n'est pas obligatoire de suivre ces conseils mais il faut noter qu'ils sont tous fondés sur des articles publiés dans des revues techniques avec examen par les pairs. Donc, c'est quand même sérieux, même si ce RFC n'a pas le statut de norme. Le RFC vise surtout les « gros » serveurs faisant autorité, par exemple les serveurs de TLD importants. Ces serveurs sont typiquement "*anycastés*" (RFC 1546 [1] et RFC 4786). Si le vôtre est "*unicast*" et ne sert qu'à trois visiteurs de temps en temps, vous n'êtes pas forcément concerné·e.

Au passage, cette distinction entre serveurs faisant autorité <https://www.bortzmeyer.org/
serveur-dns-faisant-autorite.html> et résolveurs <https://www.bortzmeyer.org/resolveur-dns.
html> est cruciale dès qu'on parle du DNS. La section 2 du RFC commence d'ailleurs par un rappel de cette distinction. D'autre part, cette même section insiste sur l'importance du DNS pour le vécu de l'utilisatrice. Par exemple la latence <https://www.bortzmeyer.org/latence.html> perçue par celle-ci dépend en bonne partie du DNS. Voici un exemple :

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc1546.txt

```
% curl-timing https://www.nic.af/
DNS: 0,920891 s
TCP connect: 1,124164 s
TLS connect: 1,455578 s
Start transfer: 1,598288 s
Total: 1,735741 s
```

On voit que la résolution DNS a pris plus de la moitié du temps total. (La commande `curl-timing` est définie par `alias curl-timing='curl --silent --output /dev/null --write-out "DNS: %{time_namelookup} s\nTCP connect: %{time_connect} s\nTLS connect: %{time_appconnect} s\nStart transfer: %{time_starttransfer} s\nTotal: %{time_total} s\n"'`.) Sur ce sujet de la latence, voir aussi l'article « *"The Internet at the Speed of Light"* <http://speedierweb.web.engr.illinois.edu/cspeed/papers/hotnets14.pdf> ». La latence dépend en grande partie de la distance géographique, et la réduire impose donc une présence mondiale.

Et ces performances, et cette robustesse sont menacées, entre autres, par les nombreuses attaques par déni de service qui visent le DNS, comme celle décrite dans l'article « *"Anycast vs. DDoS : Evaluating the November 2015 Root DNS Event"* <https://www.isi.edu/~johnh/PAPERS/Moura16b.pdf> ».

La section 3 est le cœur du RFC, contenant les recommandations, numérotées de C1 à C6. Commençons par C1, le conseil de n'avoir que des serveurs *"anycast"*. L'*"anycast"* est indispensable pour répartir très largement les serveurs, assurant la performance et la robustesse de la résolution DNS. Bien sûr, on peut se contenter d'avoir plusieurs enregistrements NS mais cela ne suffit pas : diverses raisons font qu'on ne peut pas avoir des centaines d'enregistrements dans un ensemble NS, alors que certains nuages *"anycast"* atteignent cette taille. Donc, avoir plusieurs enregistrements NS est nécessaire mais cela ne dispense pas de l'*"anycast"*.

C'est d'autant plus vrai que, parmi les enregistrements NS, la sélection de celui utilisé est faite par le résolveur. Les résolveurs corrects mesurent la latence et choisissent le serveur faisant autorité le plus rapide. Mais il y a d'autres résolveurs qui n'utilisent pas de bons algorithmes de sélection (cf. « *"Recursives in the Wild : Engineering Authoritative DNS Servers"* <https://www.isi.edu/%7ejohnh/PAPERS/Mueller17b.pdf> »). Avec l'*"anycast"*, la sélection de l'instance utilisée est faite par le réseau, via BGP, et peut donc être meilleure. Du fait de ces résolveurs sous-performants, un opérateur a donc intérêt à ce que tous les serveurs faisant autorité soient *"anycastés"*. S'il y a un « maillon faible », *"unicast"* et mal connecté, il sera quand même consulté par certains résolveurs, et dégradera donc la latence dans certains cas. Par exemple, le TLD `.nl` est passé en tout *"anycast"* en 2018.

Maintenant, C2, le conseil de travailler son routage. Une fois qu'on a décidé de mettre de l'*"anycast"* partout, il reste à optimiser. BGP ne garantit pas du tout, contrairement à ce qu'on lit parfois, que la route la plus rapide sera choisie. BGP fait ce que les administrateurs réseau lui demandent de faire. On voit régulièrement des traceroute qui montrent qu'on ne va pas à l'instance *"anycast"* la plus proche. Certains opérateurs DNS compensent en ajoutant davantage d'instances mais l'article « *"Anycast Latency : How Many Sites Are Enough ?"* <https://www.isi.edu/%7ejohnh/PAPERS/Schmidt17a.pdf> » montre, en testant des serveurs de la racine depuis des sondes RIPE Atlas <https://atlas.ripe.net>, que le nombre d'instances compte moins que le soin apporté au routage. (On ne parle ici que de performances ; pour la résistance aux attaques par déni de service, le nombre d'instances reste essentiel.) Ici, le temps de réponse des serveurs faisant autorité pour `.fr`, depuis une machine en France (il y a une instance proche pour chaque serveur) :

```
% check-soa -i fr
d.nic.fr.
2001:678:c::1: OK: 2230506452 (4 ms)
```

```
194.0.9.1: OK: 2230506452 (0 ms)
e.ext.nic.fr.
193.176.144.22: OK: 2230506434 (15 ms)
2a00:d78:0:102:193:176:144:22: OK: 2230506434 (15 ms)
f.ext.nic.fr.
194.146.106.46: OK: 2230506434 (6 ms)
2001:67c:1010:11::53: OK: 2230506452 (11 ms)
g.ext.nic.fr.
2001:678:4c::1: OK: 2230506434 (2 ms)
194.0.36.1: OK: 2230506434 (2 ms)
```

Le conseil C3 porte sur un concept très important lorsqu'on fait de l'"*anycast*", celui de bassin d'attraction. Ce terme désigne l'ensemble des réseaux qui vont envoyer leurs paquets vers une instance donnée. Si toutes les instances ont à peu près les mêmes capacités de traitement des requêtes, on souhaite en général placer les instances et configurer BGP de manière à ce que toutes les instances reçoivent à peu près la même quantité de requêtes. Mais en pratique c'est rarement le cas : le réseau est une chose compliquée. Il existe des méthodes pour améliorer les choses (cf. l'article « "*Broad and Load-Aware Anycast Mapping with Verfploeter*" <https://www.isi.edu/%7ejohnh/PAPERS/Vries17b.pdf> », qui décrit l'outil Verfploeter <https://github.com/Woutifier/verfploeter>) mais ce n'est jamais parfait. Au moins, un outil comme Verfploeter permet de procéder plus scientifiquement qu'au pifomètre, et d'avoir une idée de ce que produiront des changements de configuration BGP pour faire du "*traffic engineering*" (par exemple, allonger le chemin d'AS dans les annonces, pour diminuer le trafic d'une instance).

Le conseil C4, quant à lui, porte sur les attaques par déni de service (comme celle décrite dans « "*Anycast vs. DDoS : Evaluating the November 2015 Root DNS Event*" <https://www.isi.edu/~johnh/PAPERS/Moura16b.pdf> ») et sur le comportement des serveurs soumis à un stress intense. Que peut faire l'opérateur lorsque Mirai ou un de ses semblables attaque ?

— Essayer de diminuer le trafic entrant, soit en faisant du "*traffic engineering*" BGP pour envoyer le trafic vers une autre instance, si certaines ont davantage de capacité, ou bien ne sont pas attaquées, voire, si on n'a pas de réserves, en arrêtant complètement d'annoncer les routes (ce qui arrête le service - ce qui était le but de l'attaquant - mais cela peut être nécessaire s'il y a d'autres services au même endroit et qu'on veut les épargner),
— Ou bien encaisser en silence, sachant qu'on perdra des requêtes et qu'on ne pourra pas répondre à tout. Si l'attaque ne frappait que certaines instances, cela peut permettre d'épargner les instances non attaquées.

Le conseil C4 est donc d'avoir des plans prêts à l'avance, permettant de prendre une décision en cas de crise, et de l'appliquer.

Et les TTL ? Ils ne sont pas oubliés. La mémorisation des réponses, pendant une durée maximale égale au TTL, est un point essentiel du DNS, et un facteur de performance crucial. Même si un serveur faisant autorité répond souvent en moins de 50 ms, la réponse de la mémoire d'un résolveur, qui peut arriver en 1 ms, sera encore préférable. Ce rôle a été largement étudié (voir par exemple « "*Modeling TTL- based Internet Caches*" <http://www.ieee-infocom.org/2003/papers/11_01.PDF> » ou, plus récemment, « "*When the Dike Breaks : Dissecting DNS Defenses During DDoS*" <https://www.isi.edu/~johnh/PAPERS/Moura18b.pdf> »). Une étude des conséquences du choix du TTL sur les performances est la base du conseil C5, l'étude « "*Cache Me If You Can : Effects of DNS Time-to- Live*" <https://www.isi.edu/~hardaker/papers/2019-10-cache-me-ttls.pdf> ». Elle montre que :

— Des TTL plus longs diminuent le temps de réponse (puisque les données sont conservées plus longtemps dans la mémoire des résolveurs), cf. par exemple l'expérience faite sur .uy citée dans l'article ci-dessus,
— Ces TTL plus longs diminuent le trafic DNS, pour la même raison, (logiquement, les gens qui s'inquiètent de l'empreinte environnementale du numérique devraient mettre des TTL longs sur leurs domaines),

— Et ils améliorent la robustesse face aux attaques par déni de service (même si tous les serveurs faisant autorité ont du mal à répondre, les résolveurs pourront continuer à servir les données),
— D'un autre côté, et comme il n'existe pas de solution parfaite, juste des choix et des compromis, des TTL plus longs empêchent de changer rapidement sa configuration ; si le changement était planifié, ce n'est pas grave (on abaisse le TTL avant le changement et on le remonte après), mais si on veut le faire en urgence, c'est plus gênant,
— C'est d'autant plus gênant que certains services d'atténuation des attaques par déni de service fonctionnent en utilisant le DNS pour envoyer le trafic vers l'atténuateur ; des TTL longs limitent cette possibilité (et la planification ne change rien dans ce cas, puisque les attaquants ne se signalent pas forcément à l'avance),
— Même problème avec la répartition de charge utilisant le DNS.

Bref, on ne peut pas faire une recommandation quantitative appliquable à tous les cas. Le conseil C5 est donc plus complexe :

— Pour un domaine ordinaire, des TTL d'au moins une heure, et de préférence d'au moins huit heures sont recommandés,
— Pour les gros registres (par exemple registres de TLD), la plupart des résolveurs utiliseront de toute façon les TTL indiqués dans la zone fille ; pour le bien des autres résolveurs, le RFC recommande un TTL d'au moins une heure,
— Pour les zones « terminales » qui utilisent des répartiteurs de charge ou des services d'atténuation d'attaques qui reposent sur le DNS, on peut se contenter de TTL de 5 minutes, mais le RFC recommande quand même au moins 15 minutes.

Et si les TTL entre la zone parente et la zone fille diffèrent ? La question se pose typiquement pour les enregistrements NS. Par exemple, les enregistrements NS dans la zone racine ont un TTL de 48 heures, alors que dans les zones des TLD, l'ensemble NS a presque toujours un TTL plus faible (une heure pour `.cl`, par exemple). Selon que le résolveur est "*parent-centric*" (une minorité) ou "*child-centric*" (la grande majorité), il utilisera le TTL de la zone parente ou bien celui de la zone fille. Le conseil C6 : on ne peut pas compter que le TTL qu'on a mis soit respecté partout, si l'enregistrement existe aussi dans la zone parente avec des TTL différents. Attention donc lorsqu'on veut changer quelque chose, à intégrer le TTL de la zone parente.

Un bon résumé en anglais de ce RFC a été écrit par un des auteurs <`https://www.sidnlabs.nl/en/news-and-blogs/rfc-9199-considerations-for-large-authoritative-server-operators`>. Il inclut un récit de la création du RFC et du processus suivi.