RFC 8906 : A Common Operational Problem in DNS Servers: Failure To Communicate

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 23 septembre 2020

Date de publication du RFC : Septembre 2020

https://www.bortzmeyer.org/8906.html

Normalement, le protocole DNS est très simple : le client écrit au serveur, posant une question, et le serveur répond. Il peut répondre qu'il ne veut pas ou ne sait pas répondre, mais il le dit **explicitement**. Cela, c'est la théorie. En pratique, beaucoup de serveurs DNS bogués (ou, plus fréquemment, situés derrière une "*middlebox*" boguée) ne répondent pas du tout, laissant le client perplexe se demander s'il doit réeessayer différemment ou pas. Ce nouveau RFC documente le problème et ses conséquences. Il concerne surtout la question des serveurs faisant autorité <https://www.bortzmeyer.org/serveur-dns-faisant-
html>.

(Cette absence de réponse est en général due, non pas aux serveurs DNS eux-même, qui sont la plupart du temps corrects de ce point de vue, mais plutôt à ces "*middleboxes*" codées avec les pieds et mal configurées, que certains "*managers*" s'obstinent à placer devant des serveurs DNS qui marcheraient parfaitement sans cela. Ainsi, on voit des pare-feux inutiles mis parce que « il faut un pare-feu, a dit l'auditeur » et qui bloquent tout ce qu'ils ne comprennent pas. Ainsi, par exemple, le pare-feu laissera passer les requêtes de type A mais pas celles de type NS, menant à une expiration du délai de garde. La section 4 de notre RFC détaille ces erreurs communes des "*middleboxes*".)

Voici un exemple d'un service DNS bogué. Le domaine `mabanque.bnpparibas` est délégué à deux serveurs mal configurés (ou placés derrière une "*middlebox*" mal faite), `sns6.bnpparibas.fr` et `sns5.bnpparibas.net` :

```
% dig @sns6.bnpparibas.fr A mabanque.bnpparibas

; <<>> DiG 9.11.5-P4-5.1-Debian <<>> @sns6.bnpparibas.fr A mabanque.bnpparibas
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 57381
```

```
;; flags: qr aa rd ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;mabanque.bnpparibas. IN A

;; ANSWER SECTION:
mabanque.bnpparibas. 30 IN A 159.50.187.79

;; Query time: 24 msec
;; SERVER: 159.50.105.65#53(159.50.105.65)
;; WHEN: Wed Jun 17 08:21:07 CEST 2020
;; MSG SIZE  rcvd: 53



% dig @sns6.bnpparibas.fr NS mabanque.bnpparibas

; <<>> DiG 9.11.5-P4-5.1-Debian <<>> @sns6.bnpparibas.fr NS mabanque.bnpparibas
; (1 server found)
;; global options: +cmd
;; connection timed out; no servers could be reached
```

Face à un service aussi mal fait, le client DNS est très désarmé. Est-ce que le serveur a ignoré la requête ? Est-ce que le paquet a été perdu (c'est l'Internet, rien n'est garanti) ? Dans le deuxième cas, il faut réessayer, dans le premier, cela ne servirait à rien, qu'à perdre du temps, et à en faire perdre à l'utilisateur. Le client peut aussi supposer que l'absence de réponse est due à telle ou telle nouveauté du protocole DNS qu'il a utilisé, et se dire qu'il faudrait réessayer sans cette nouveauté, voire ne jamais l'utiliser. On voit que ces services grossiers, qui ne répondent pas aux requêtes, imposent un coût conséquent au reste de l'Internet, en délais, en trafic réseau, et en hésitation à déployer les nouvelles normes techniques.

Il n'y a aucune raison valable pour une absence de réponse ? Notre RFC en note une : une attaque par déni de service en cours. Dans ce cas, l'absence de réponse est légitime (et, de toute façon, le serveur peut ne pas avoir le choix). En dehors d'une telle attaque, le serveur **doit** répondre, en utilisant un des codes de retour DNS existants, qui couvrent tous les cas possibles (de `NOERROR`, quand tout va bien, à `REFUSED`, le refus délibéré, en passant par `SERVFAIL`, l'impossibilité de produire une réponse sensée). Ici, un cas où je demande à un serveur de `.fr` des informations sur un `.com`, qu'il n'a évidemment pas, d'où le refus explicite (cf. le champ "*status :*") :

```
 % dig @d.nic.fr A www.microsofthub.com

; <<>> DiG 9.11.5-P4-5.1-Debian <<>> @d.nic.fr A www.microsofthub.com
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: REFUSED, id: 4212
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1432
; COOKIE: 452269774eb7b7c574c779de5ee9b8e6efe0f777274b2b17 (good)
;; QUESTION SECTION:
;www.microsofthub.com. IN A

;; Query time: 4 msec
;; SERVER: 2001:678:c::1#53(2001:678:c::1)
```

```
;; WHEN: Wed Jun 17 08:32:06 CEST 2020
;; MSG SIZE  rcvd: 77
```

Le RFC note qu'il n'y a pas que l'absence de réponse, il y a aussi parfois des réponses incorrectes. Ainsi, certains serveurs (ou, là encore, la "*middlebox*" placée devant eux) copient le bit AD de la requête dans la réponse au lieu de déterminer par eux-mêmes si ce bit - qui signifie "*Authentic Data*" - doit être mis dans la réponse.

Donc, ne pas répondre, c'est mal, et égoïste. Mais quelles sont les conséquences exactes de cette absence de réponse ? Parmi elles :

— Pendant longtemps, les résolveurs DNS, face à une absence de réponse, réessayaient sans EDNS, ce qui résolvait parfois le problème, mais a sérieusement gêné le déploiement d'EDNS.
— Même problème avec les nouvelles utilisations d'EDNS.
— Un comportement fréquent des pare-feux est de bloquer les requêtes demandant des types de données qu'ils ne connaissent pas. La liste connue de ces pare-feux est toujours très réduite par rapport à la liste officielle <https://www.iana.org/assignments/dns-parameters/dns-parameters.xml#dns-parameters-4>, ce qui se traduit par une grande difficulté à déployer de nouveaux types de données DNS. C'est une des raisons derrière l'échec du type SPF (cf. RFC 6686 [1]). Cela encourage l'usage de types de données fourre-tout comme TXT.
— Même problème pour le déploiement de DANE et de son type TLSA.
— Le RFC ne cite pas ce cas, mais ne pas répondre peut aussi dans certains cas faciliter des attaques par empoisonnement d'un résolveur, cf. l'attaque SLIP <https://www.cert.ssi.gouv.fr/avis/CERTA-2013-AVI-506/>.

Quels sont les cas où il est particulièrement fréquent qu'il n'y ait pas de réponse, ou bien une réponse erronée ? La section 3 en décrit un certain nombre (et le RFC vient avec un script qui utilise dig pour tester une grande partie de ces cas). Elle rappelle également les réponses correctes attendues. Par exemple, une requête de type SOA ("*Start Of Authority*") à un serveur faisant autorité <https://www.bortzmeyer.org/serveur-dns-faisant-autorite.html> pour une zone doit renvoyer une réponse (contrairement aux serveurs de BNP Paribas cités plus haut, qui ne répondent pas). Même si le type de données demandé est inconnu du serveur, il doit répondre (probablement NOERROR s'il n'a tout simplement pas de données du type en question).

On voit parfois également des serveurs (ou plutôt des combinaisons serveur / "*middlebox*" boguée située devant le serveur) qui achoppent sur les requêtes utilisant des options ("*flags*") DNS spécifiques. Ici, l'option Z fait que le serveur de la RATP ne répond plus :

```
% dig @193.104.162.15 +noedns +noad +norec +zflag soa ratp.fr
; <<>> DiG 9.11.5-P4-5.1-Debian <<>> @193.104.162.15 +noedns +noad +norec +zflag soa ratp.fr
; (1 server found)
;; global options: +cmd
;; connection timed out; no servers could be reached
```

Alors qu'il marche parfaitement sans cette option :

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc6686.txt

```
  % dig @193.104.162.15 +noedns +noad +norec soa ratp.fr

; <<>> DiG 9.11.5-P4-5.1-Debian <<>> @193.104.162.15 +noedns +noad +norec soa ratp.fr
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 30635
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 2

;; QUESTION SECTION:
;ratp.fr. IN SOA

;; ANSWER SECTION:
ratp.fr. 3600 IN SOA ns0.ratp.fr. hostmaster.ratp.fr. 2020051201 21600 3600 4204800 300
...
;; ADDITIONAL SECTION:
ns0.ratp.fr. 3600 IN A 193.104.162.15
ns1.ratp.fr. 3600 IN A 193.104.162.14

;; Query time: 6 msec
;; SERVER: 193.104.162.15#53(193.104.162.15)
;; WHEN: Mon May 18 17:42:33 CEST 2020
;; MSG SIZE  rcvd: 213
```

Autre exemple, l'option AD ("*Authentic Data*") dans la requête, qui indique que le client espère une validation DNSSEC, déclenche parfois des bogues, comme de ne pas répondre ou bien de répondre en copiant aveuglément le bit AD dans la réponse. La section 6 du RFC détaille un cas similaire, celui des serveurs qui réutilisent une réponse déjà mémorisée, mais pour des options différentes dans la requête.

Et les opérations ("*opcodes*") inconnus ? Des opérations comme `NOTIFY` ou `UPDATE` n'existaient pas au début du DNS et d'autres seront encore ajoutées dans le futur. Si le serveur ne connait pas une opération, il doit répondre `NOTIMP` ("*Not Implemented*"). Ici, avec l'opération 1 (`IQUERY`, ancienne mais abandonnée par le RFC 3425) :

```
% dig @d.nic.fr +opcode=1 toto.fr

; <<>> DiG 9.11.5-P4-5.1+deb10u1-Debian <<>> @d.nic.fr +opcode=1 toto.fr
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: IQUERY, status: NOTIMP, id: 20180
;; flags: qr; QUERY: 0, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1432
; COOKIE: 71f9ced2f29076a65d34f3ed5ef2f718bab037034bb82106 (good)
;; Query time: 4 msec
;; SERVER: 2001:678:c::1#53(2001:678:c::1)
;; WHEN: Wed Jun 24 08:47:52 CEST 2020
;; MSG SIZE  rcvd: 51
```

Au contraire, voici un serveur qui répond incorrectement (`REFUSED` au lieu de `NOTIMP`) :

```
% dig  @ns3-205.azure-dns.org. +noedns +noad +opcode=15 +norec microsoft.com

; <<>> DiG 9.11.5-P4-5.1+deb10u1-Debian <<>> @ns3-205.azure-dns.org. +noedns +noad +opcode=15 +norec microsoft.c
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: RESERVED15, status: REFUSED, id: 41518
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;microsoft.com. IN A

;; Query time: 22 msec
;; SERVER: 2a01:111:4000::cd#53(2a01:111:4000::cd)
;; WHEN: Wed Jun 24 09:16:19 CEST 2020
;; MSG SIZE  rcvd: 31
```

Soyons positifs : c'était bien pire il y a encore seulement cinq ou dix ans, malgré des tests techniques obligatoires dans certains registres comme Zonecheck pour `.fr`. Des efforts comme le "*DNS Flag Day*" <https://dnsflagday.net/> ont permis d'arranger sérieusement les choses.

Bien sûr, les serveurs DNS doivent accepter les requêtes venant sur TCP et pas seulement sur UDP. Cela a toujours été le cas, mais le RFC 7766 rend cette exigence encore plus stricte. Il y a fort longtemps, l'outil de test Zonecheck de l'AFNIC testait ce fonctionnement sur TCP, et avait attrapé beaucoup d'erreurs de configuration, suscitant parfois des incompréhensions de la part d'administrateurs système ignorants qui prétendaient que le DNS n'utilisait pas TCP.

Et il y a bien sûr EDNS (RFC 6891). Introduit après la norme originale du DNS, mais quand même ancienne de plus de vingt ans, EDNS est toujours mal compris par certains programmeurs, et bien des "*middleboxes*" déconnent toujours sur des particularités d'EDNS. Déjà, un serveur doit répondre aux requêtes EDNS (même si lui-même ne connait pas EDNS, il doit au moins répondre `FORMERR`). Ensuite, s'il connait EDNS, il doit gérer les numéros de version d'EDNS (attention, la version actuelle est la 0, il n'y a pas encore eu de version 1 mais, pour qu'elle puisse être déployée un jour, il ne faut pas que les serveurs plantent sur les versions supérieures à zéro). EDNS permet d'indiquer des options (la liste complète est dans un registre IANA <https://www.iana.org/assignments/dns-parameters/dns-parameters.xml#dns-parameters-11>) et les options inconnues doivent être ignorées (autrement, on ne pourrait jamais déployer une nouvelle option).

Pour tester si vos serveurs gèrent correctement tous ces cas, le RFC (section 8) vient avec une série de commandes utilisant dig, dont il faudra analyser le résultat manuellement en suivant le RFC. J'en ai fait un script de test, (en ligne sur `https://www.bortzmeyer.org/files/check-dns-respond-rfc8906.sh`). Si vous testez vos serveurs avec ce script, faites-le depuis l'extérieur (pour intégrer dans le test les éventuelles "*middleboxes*", cf. section 4). Un exemple d'exécution du test :

```
%  ./check-dns-respond-rfc8906.sh cyberstructure.fr 2001:4b98:dc0:41:216:3eff:fe27:3d3f >& /tmp/mytest.txt
```

Et il vous reste à lire le fichier des résultats et à comparer avec les résultats attendus.