RFC 8483 : Yeti DNS Testbed

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 17 décembre 2018

Date de publication du RFC : Octobre 2018

https://www.bortzmeyer.org/8483.html

Ce RFC décrit une expérience, celle qui, de mai 2015 à décembre 2018, a consisté à faire tourner une racine DNS alternative nommée Yeti <https://yeti-dns.org/>. Contrairement aux racines alternatives commerciales qui ne sont typiquement que des escroqueries visant à vendre à des gogos des TLD reconnus par personne, Yeti était une expérience technique ; il s'agissait de tester un certain nombre de techniques qu'on ne pouvait pas se permettre de tester sur la « vraie » racine.

Parmi ces techniques, l'utilisation d'un grand nombre de serveurs racine (pour casser la légende comme quoi l'actuelle limite à 13 serveurs aurait une justification technique), n'utiliser qu'IPv6, jouer avec des paramètres DNSSEC différents, etc. J'ai participé à ce projet, à la fois comme gérant de deux des serveurs racine, et comme utilisateur de la racine Yeti, reconfigurant des résolveurs DNS pour utiliser Yeti. Les deux serveurs racines dahu1.yeti.eu.org et dahu2.yeti.eu.org appartenaient au groupe Dahu, formé par l'AFNIC (cf. cet article sur le site de l'AFNIC <https://www.afnic.fr/fr/ressources/blog/le-projet-yeti-d-experimentation-d-une-racine-dns-2.html>), Gandi et eu.org <https://nic.eu.org/>. (Le nom vient d'un animal aussi mythique que le yéti.)

Outre l'aspect technique, un autre intéret de Yeti était qu'il s'agissait d'un projet international. Réellement international, pas seulement des états-uniens et des européens de divers pays ! Yeti est d'inspiration chinoise, la direction du projet était faite en Chine, aux États-Unis et au Japon, et parmi les équipes les plus impliquées dans le projet, il y avait des Russes, des Indiens, des Français, des Chiliens…Le projet, on l'a dit, était surtout technique (même si certains participants pouvaient avoir des arrière-pensées) et la zone racine servie par Yeti était donc exactement la même que celle de l'IANA, aux noms des serveurs et aux signatures DNSSEC près. Un utilisateur ordinaire de Yeti ne voyait donc aucune différence. Le projet étant de nature expérimentale, les utilisateurs étaient tous des volontaires, conscients des risques possibles (il y a eu deux ou trois cafouillages).

L'annexe E du RFC est consacrée aux controverses sur le principe même du projet Yeti. Le projet a toujours été discuté en public, et présenté à de nombreuses réunions. Mais il y a toujours des râleurs, affirmant par exemple que ce projet était une racine alternative (ce qui n'est pas faux mais attendez,

lisez jusqu'au bout) et qu'il violait donc le RFC 2826[1]. Outre que ce RFC 2826 est très contestable, il faut noter qu'il ne s'applique pas à Yeti ; il concerne uniquement les racines alternatives servant un contenu différent de celui de la racine « officielle » alors que Yeti a toujours été prévu et annoncé comme servant **exactement** la même racine (comme le faisait ORSN <https://www.bortzmeyer.org/orsn-vraiment-fini.html>). Rien à voir donc avec ces racines alternatives <https://www.bortzmeyer.org/racines-alternatives.html> qui vous vendent des TLD bidons, que personne ne pourra utiliser. Comme le disait Paul Vixie, Yeti pratique le "*Responsible Alternate Rootism*" <https://yeti-dns.org/resource/workshop/paul-2015-10-Yokohama.pdf>. Notez quand même que certains participants à Yeti (notamment en Chine et en Inde) avaient des objectifs qui n'étaient pas purement techniques (s'insérant dans le problème de la gouvernance Internet, et plus spécialement celle de la racine).

La racine du DNS est quelque chose d'absolument critique pour le bon fonctionnement de l'Internet. Quasiment toutes les activités sur l'Internet démarrent par une ou plusieurs requêtes DNS. S'il n'y a plus de résolution DNS, c'est à peu près comme s'il n'y avait plus d'Internet (même si quelques services pair-à-pair, comme Bitcoin, peuvent encore fonctionner). Du fait de la nature arborescente du DNS, si la racine a un problème, le service est sérieusement dégradé (mais pas arrêté, notamment en raison des mémoires - les « caches » - des résolveurs). On ne peut donc pas jouer avec la racine, par exemple en essayant des idées trop nouvelles et peu testées. Cela n'a pas empêché la racine de changer beaucoup : il y a eu par exemple le déploiement massif de l'"*anycast*", qui semblait inimaginable il y a dix-sept ans, le déploiement de DNSSEC (avec le récent changement de clé, qui s'est bien passé), ou celui d'IPv6, plus ancien. Le fonctionnement de la racine était traditionnellement peu ou pas documenté mais il y a quand même eu quelques documents utiles, comme le RFC 7720, la première description de l'anycast <http://ftp.isc.org/isc/pubs/tn/isc-tn-2003-1.txt>, ou les documents du RSSAC <https://www.icann.org/groups/rssac>, comme RSSAC 001 <https://www.icann.org/en/system/files/files/rssac-001-root-service-expectations-04dec15-en.pdf>. Celle ou celui qui veut se renseigner sur la racine a donc des choses à lire.

Mais le point important est que la racine est un système en production, avec lequel on ne peut pas expérimenter à loisir. D'où l'idée, portée notamment par BII <http://www.biigroup.com/>, mais aussi par TISF et WIDE, d'une racine alternative n'ayant pas les contraintes de la « vraie » racine. Yeti (section 1 du RFC) n'est pas un projet habituel, avec création d'un consortium, longues réunions sur les statuts, et majorité du temps passé en recherches de financement. C'est un projet léger, indépendant d'organismes comme l'ICANN, géré par des volontaires, sans structure formelle et sans budget central, dans la meilleure tradition des grands projets Internet. À son maximum, Yeti a eu 25 serveurs racine, gérés par 16 organisations différentes.

Au passage, puisqu'on parle d'un projet international, il faut noter que ce RFC a été sérieusement ralenti par des problèmes de langue. Eh oui, tout le monde n'est pas anglophone et devoir rédiger un RFC en anglais handicape sérieusement, par exemple, les Chinois.

Parmi les idées testées sur Yeti (section 3 du RFC) :
— Une racine alternative qui marche (avec supervision sérieuse ; beaucoup de services se présentant comme « racine alternative » ont régulièrement la moitié de leurs serveurs racine en panne),
— Diverses méthodes pour signer et distribuer la zone racine,
— Schémas de nommage pour les serveurs racine (dans la racine IANA, tous les serveurs sont sous root-servers.net),
— Signer les zones des serveurs racine (actcuellement, root-servers.net n'est pas signé),
— Utiliser exclusivement IPv6,

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc2826.txt

— Effectuer des remplacements de clé (notamment en comptant sur le RFC 5011),
— Sans compter les autres idées décrites dans cette section 3 du RFC mais qui n'ont pas pu être testées.

La section 4 du RFC décrit l'infrastructure de Yeti. Elle a évidemment changé plusieurs fois, ce qui est normal pour un service voué aux expérimentations. Yeti utilise l'architecture classique du DNS. Les serveurs racine sont remplacés par ceux de Yeti, les autres serveurs faisant autorité (ceux de `.fr` ou `.org`, par exemple) ne sont pas touchés. Les résolveurs doivent évidemment être reconfigurés pour utiliser Yeti (d'une manière qui est documentée sur le site Web du projet <`https://yeti-dns.org/join.html`>). Au démarrage, un résolveur ne connait en effet que la liste des noms et adresses IP des serveurs de la racine. La racine « officielle » est configurée par défaut et doit ici être remplacée (annexe A du RFC). Il faut aussi changer la clé de la racine (la *"root trust anchor"*) puisque Yeti signe avec sa propre clé.

Voici la configuration de mon résolveur à la maison, avec Knot <`https://www.knot-resolver.cz/`> sur une Turris Omnia :

```
 config resolver 'common'
option keyfile '/etc/kresd/yeti-root.keys'
option prefered_resolver 'kresd'

config resolver 'kresd'
option rundir '/tmp/kresd'
option log_stderr '1'
option log_stdout '1'
option forks '1'
option include_config '/etc/kresd/custom.conf'
```

et `custom.conf` contient la liste des serveurs racine :

```
hints.root({
['bii.dns-lab.net.'] = '240c:f:1:22::6',
['yeti-ns.tisf.net .'] = '2001:4f8:3:1006::1:4',
['yeti-ns.wide.ad.jp.'] = '2001:200:1d9::35',
['yeti-ns.as59715.net.'] = '2a02:cdc5:9715:0:185:5:203:53',
['dahu1.yeti.eu.org.'] = '2001:4b98:dc2:45:216:3eff:fe4b:8c5b',
['ns-yeti.bondis.org.'] = '2a02:2810:0:405::250',
['yeti-ns.ix.ru .'] = '2001:6d0:6d06::53',
['yeti.bofh.priv.at.'] = '2a01:4f8:161:6106:1::10',
['yeti.ipv6.ernet.in.'] = '2001:e30:1c1e:1::333',
['yeti-dns01.dnsworkshop.org.'] = '2001:1608:10:167:32e::53',
['yeti-ns.conit.co.'] = '2604:6600:2000:11::4854:a010',
['dahu2.yeti.eu.org.'] = '2001:67c:217c:6::2',
['yeti.aquaray.com.'] = '2a02:ec0:200::1',
['yeti-ns.switch.ch.'] = '2001:620:0:ff::29',
['yeti-ns.lab.nic.cl.'] = '2001:1398:1:21::8001',
['yeti-ns1.dns-lab.net.'] = '2001:da8:a3:a027::6',
['yeti-ns2.dns-lab.net.'] = '2001:da8:268:4200::6',
['yeti-ns3.dns-lab.net.'] = '2400:a980:30ff::6',
['ca978112ca1bbdcafac231b39a23dc.yeti-dns.net.'] = '2c0f:f530::6',
['yeti-ns.datev.net.'] = '2a00:e50:f15c:1000::1:53',
['3f79bb7b435b05321651daefd374cd.yeti-dns.net.'] = '2401:c900:1401:3b:c::6',
['xn--r2bi1c.xn--h2bv6c0a.xn--h2brj9c.'] = '2001:e30:1c1e:10::333',
['yeti1.ipv6.ernet.in.'] = '2001:e30:187d::333',
['yeti-dns02.dnsworkshop.org.'] = '2001:19f0:0:1133::53',
['yeti.mind-dns.nl.'] = '2a02:990:100:b01::53:0'
})
```

Au bureau, avec Unbound, cela donnait :

```
server:
    auto-trust-anchor-file: "/var/lib/unbound/yeti.key"
    root-hints: "yeti-hints"
```

Et `yeti-hints` est disponible en annexe A du RFC (attention, comme le note la section 7 du RFC, à utiliser une source fiable, et à le récupérer de manière sécurisée).

Comme Yeti s'est engagé à ne pas modifier le contenu de la zone racine (liste des TLD, et serveurs de noms de ceux-ci), et comme Yeti visait à gérer la racine de manière moins concentrée, avec trois organisations (BII, TISF et WIDE) signant et distribuant la racine, le mécanisme adopté a été :

— Les trois organisations copient la zone racine « normale »,
— En retirent les clés et les signatures DNSSEC et la liste des serveurs de la racine,
— Modifient le SOA,
— Ajoutent la liste des serveurs Yeti, les clés Yeti, et signent la zone,
— Les serveurs racine copient automatiquement cette nouvelle zone signée depuis un des trois DM ("*Distribution Master*", section 4.1 du RFC).

Voici le SOA Yeti :

```
% dig SOA .
...
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 45919
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1
...
;; ANSWER SECTION:
. 86400 IN SOA www.yeti-dns.org. bii.yeti-dns.org. (
2018121600 ; serial
1800       ; refresh (30 minutes)
900        ; retry (15 minutes)
604800     ; expire (1 week)
86400      ; minimum (1 day)
)
. 86400 IN RRSIG SOA 8 0 86400 (
20181223050259 20181216050259 46038 .
BNoxqfGq5+rBEdY4rdp8W6ckNK/GAOtBWQ3P36YFq5N+
...
;; Query time: 44 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sun Dec 16 16:01:27 CET 2018
;; MSG SIZE  rcvd: 369
```

(Notez que l'adresse du responsable de la zone indique le DM qui a été utilisé par ce résolveur particulier. Un autre résolveur pourrait montrer un autre SOA, si le DM était différent.) Comme les serveurs racine « officiels » n'envoient pas de message `NOTIFY` (RFC 1996) aux serveurs Yeti, la seule solution est d'interroger régulièrement ces serveurs officiels. (Cela fait que Yeti sera toujours un peu en retard sur la racine « officielle », cf. section 5.2.2.) Plusieurs de ces serveurs acceptent le transfert de zone (RFC 5936), par exemple `k.root-servers.net` :

```
% dig @k.root-servers.net AXFR . > /tmp/root.zone

% head -n 25 /tmp/root.zone

; <<>> DiG 9.11.3-1ubuntu1.3-Ubuntu <<>> @k.root-servers.net AXFR .
; (2 servers found)
;; global options: +cmd
. 86400 IN SOA a.root-servers.net. nstld.verisign-grs.com. (
2018121600 ; serial
1800       ; refresh (30 minutes)
900        ; retry (15 minutes)
604800     ; expire (1 week)
86400      ; minimum (1 day)
)
. 172800 IN DNSKEY 256 3 8 (
AwEAAdp440E6Mz7c+Vl4sPd0lTv2Qnc85dTW64j0RDD7
...
```

De son côté, l'ICANN gère deux machines qui acceptent le transfert de zone, `xfr.cjr.dns.icann.org` et `xfr.lax.dns.icann.org`. On peut enfin récupérer cette zone par FTP.

Pour l'étape de signature de la zone, Yeti a testé plusieurs façons de répartir le travail entre les trois DM (*"Distribution Masters"*) :

— Clés (KSK et ZSK) partagées entre tous les DM. Cela nécessitait donc de transmettre les clés privées.
— KSK unique mais une ZSK différente par DM (chacune étant signée par la KSK). Chacun garde alors la clé privée de sa ZSK. (Voir la section 5.2.3 pour quelques conséquences pratiques de cette configuration.)

Dans les deux cas, Yeti supprime la totalité des signatures de la racine « officielle » avant d'apposer la sienne. Il a été suggéré <`http://yeti-dns.org/yeti/blog/2018/05/01/Experiment-plan-for-PINZ.html`> (mais pas testé) d'essayer d'en conserver une partie, pour faciliter la vérification du fait que Yeti n'avait pas ajouté ou retiré de TLD.

La configuration chez les DM et leur usage de git (les risques de sécurité que cela pose sont discutés en section 7) pour se synchroniser quand c'est nécessaire est documentée ici <`https://github.com/BII-Lab/Yeti-Project/blob/master/doc/Yeti-DM-Sync.md`>.

Les serveurs racine de Yeti n'ont plus ensuite qu'à récupérer la zone depuis un des DM ; chaque serveur racine peut utiliser n'importe quel DM et en changer, de façon à éviter de dépendre du bon fonctionnement d'un DM particulier. Voici par exemple la configuration d'un serveur NSD :

```
server:
ip-address: 2001:4b98:dc2:45:216:3eff:fe4b:8c5b
      nsid: "ascii_dahu1.yeti.eu.org"
        # RFC 8201
ipv6-edns-size: 1460

zone:
     name: "."
     outgoing-interface: 2001:4b98:dc2:45:216:3eff:fe4b:8c5b
     # We use AXFR (not the default, IXFR) because of http://open.nlnetlabs.nl/pipermail/nsd-users/2016-February
     # BII
     request-xfr: AXFR 240c:f:1:22::7 NOKEY
     allow-notify: 240c:f:1:22::7 NOKEY
     # TISF
     request-xfr: AXFR 2001:4f8:3:1006::1:5 NOKEY
     allow-notify: 2001:4f8:3:1006::1:5 NOKEY
     # WIDE
     request-xfr: AXFR 2001:200:1d9::53 NOKEY
     allow-notify: 2001:200:1d9::53 NOKEY
```

Notez que la configuration réseau était un peu plus complexe, la machine ayant deux interfaces, une de service, pour les requêtes DNS, et une d'aministration, pour se connecter via ssh. Il fallait s'assurer que les messages DNS partent bien par la bonne interface réseau, donc faire du routage selon l'adresse IP source. Le fichier de configuration Linux pour cela est .

Contrairement aux serveurs de la racine « officielle », qui sont tous sous le domaine `root-servers.net`, ceux de Yeti, ont des noms variés. Le suffixe identique permet, grâce à la compression des noms (RFC 1035, section 4.1.4 et RSSAC 023 <`https://www.icann.org/en/system/files/files/rssac-023-04nov16-pdf`>) de gagner quelques octets sur la taille des messages DNS. Yeti cherchant au contraire à tester la faisabilité de messages DNS plus grands, cette optimisation n'était pas utile.

Une des conséquences est que la réponse initiale à un résolveur (RFC 8109) est assez grande :

```
% dig @bii.dns-lab.net. NS .
...
;; SERVER: 240c:f:1:22::6#53(240c:f:1:22::6)
;; MSG SIZE  rcvd: 1591
```

On voit qu'elle dépasse la MTU d'Ethernet. Certains serveurs, pour réduire la taille de cette réponse, n'indiquent pas la totalité des adresses IP des serveurs racine (la colle) dans la réponse. (BIND, avec `minimum-responses: yes` n'envoie même aucune adresse IP, forçant le résolveur à effectuer des requêtes pour les adresses IP des serveurs). Cela peut augmenter la latence avant les premières résolutions réussies, et diminuer la robustesse (si les serveurs dont l'adresse est envoyée sont justement ceux en panne). Mais cela n'empêche pas le DNS de fonctionner et Yeti, après discussion, a décidé de ne pas chercher à uniformiser les réponses des serveurs racine.

Au moment de la publication du RFC, Yeti avait 25 serveurs racine gérés dans 16 pays différents (section 4.6 du RFC), ici vus par check-soa <`https://github.com/bortzmeyer/check-soa`> (rappelez-vous qu'ils n'ont que des adresses IPv6) :

```
% check-soa -i .
3f79bb7b435b05321651daefd374cd.yeti-dns.net.
2401:c900:1401:3b:c::6: OK: 2018121400 (334 ms)
bii.dns-lab.net.
240c:f:1:22::6: OK: 2018121400 (239 ms)
ca978112ca1bbdcafac231b39a23dc.yeti-dns.net.
2c0f:f530::6: OK: 2018121400 (170 ms)
dahu1.yeti.eu.org.
2001:4b98:dc2:45:216:3eff:fe4b:8c5b: OK: 2018121400 (18 ms)
dahu2.yeti.eu.org.
2001:67c:217c:6::2: OK: 2018121400 (3 ms)
ns-yeti.bondis.org.
2a02:2810:0:405::250: OK: 2018121400 (24 ms)
xn--r2bi1c.xn--h2bv6c0a.xn--h2brj9c.
2001:e30:1c1e:10::333: OK: 2018121400 (188 ms)
yeti-ns.as59715.net.
2a02:cdc5:9715:0:185:5:203:53: OK: 2018121400 (43 ms)
yeti-ns.datev.net.
2a00:e50:f155:e::1:53: OK: 2018121400 (19 ms)
yeti-ns.ix.ru.
2001:6d0:6d06::53: OK: 2018121400 (54 ms)
yeti-ns.lab.nic.cl.
2001:1398:1:21::8001: OK: 2018121400 (228 ms)
yeti-ns.switch.ch.
2001:620:0:ff::29: OK: 2018121400 (16 ms)
yeti-ns.tisf.net.
```

```
2001:4f8:3:1006::1:4: OK: 2018121400 (175 ms)
yeti-ns.wide.ad.jp.
2001:200:1d9::35: OK: 2018121400 (258 ms)
yeti-ns1.dns-lab.net.
2400:a980:60ff:7::2: OK: 2018121400 (258 ms)
yeti-ns2.dns-lab.net.
2001:da8:268:4200::6: OK: 2018121400 (261 ms)
yeti-ns3.dns-lab.net.
2400:a980:30ff::6: OK: 2018121400 (268 ms)
yeti.aquaray.com.
2a02:ec0:200::1: OK: 2018121400 (4 ms)
yeti.bofh.priv.at.
2a01:4f8:161:6106:1::10: OK: 2018121400 (31 ms)
yeti.ipv6.ernet.in.
2001:e30:1c1e:1::333: OK: 2018121400 (182 ms)
yeti.jhcloos.net.
2001:19f0:5401:1c3::53: OK: 2018121400 (108 ms)
yeti.mind-dns.nl.
2a02:990:100:b01::53:0: OK: 2018121400 (33 ms)
```

Notez que l'un d'eux a un nom IDN, `[Caractère Unicode non montré [2]][Caractère Unicode non montré][Caractère Unicode non montré].[Caractère Unicode non montré][Caractère Unicode non montré][Caractère Unicode non montré][Caractère Unicode non montré].[Caractère Unicode non montré][Caractère Unicode non montré][Caractère Unicode non montré]` (affiché par `check-soa` comme `xn--r2bi1c.xn--h2bv6c0a.xn--h2brj9c`). 18 des serveurs sont des VPS, le reste étant des machines physiques. 15 utilisent le noyau Linux, 4 FreeBSD, 1 NetBSD et 1 (oui, oui) tourne sur Windows. Question logiciel, 16 utilisent BIND, 4 NSD, 2 Knot, 1 Bundy (l'ex-BIND 10), 1 PowerDNS et 1 Microsoft DNS.

Pour tester que la racine Yeti fonctionnait vraiment, il ne suffisait évidemment pas de faire quelques dig, check-soa et tests avec les sondes RIPE Atlas <https://atlas.ripe.net>. Il fallait un trafic plus réaliste. Certains résolveurs (dont les miens, à la maison et au bureau) ont été configurés pour utiliser la racine Yeti et fournissaient donc un trafic réel, quoique faible. En raison des caches des résolveurs, le trafic réel ne représentait que quelques dizaines de requêtes par seconde. Il était difficile d'augmenter ce nombre, Yeti étant une racine expérimentale, où des choses risquées étaient tentées, on ne pouvait pas utiliser des résolveurs de production. Il a donc fallu aussi injecter du trafic artificiel.

Tout le trafic atteignant les serveurs racines Yeti était capturé (c'est une autre raison pour laquelle on ne pouvait pas utiliser les résolveurs de production ; Yeti voyait toutes leurs requêtes à la racine) et étudié. Pour la capture, des outils comme dnscap <https://www.dns-oarc.net/tools/dnscap> ou pcapdump <https://packages.debian.org/sid/pcaputils> (avec un petit patch <https://bugs.debian.org/545985>) étaient utilisés pour produire des pcap, ensuite copiés vers BII avec `rsync`.

La section 5 du RFC décrit les problèmes opérationnels qu'a connu Yeti. Si vous voulez tous les détails, vous pouvez regarder les archives de la liste de diffusion du projet <http://lists.yeti-dns.org/pipermail/discuss>, et le blog du projet <https://yeti-dns.org/blog.html>. D'abord, ce qui concerne IPv6. Comme d'habitude, des ennuis sont survenus avec la fragmentation. En raison du nombre de serveurs racine, et de l'absence de schéma de nommage permettant la compression, les réponses Yeti sont souvent assez grandes pour devoir être fragmentées (1 754 octets avec

2. Car trop difficile à faire afficher par LaTeX

toutes les adresses des serveurs racine, et 1 975 avec le mode « une ZSK par DM »). Cela ne serait pas un problème (la fragmentation des datagrammes étant spécifiée dans IPv4 et IPv6 depuis le début) si tout le monde configurait son réseau correctement. Hélas, beaucoup d'incompétents et de maladroits ont configuré leurs systèmes pour bloquer les fragments IP, ou pour bloquer les messages ICMP nécessaires à la découverte de la MTU du chemin (RFC 8201). Ce triste état des choses a été décrit dans le RFC 7872, dans `draft-taylor-v6ops-fragdrop`, et dans « "*Dealing with IPv6 fragmentation in the DNS*" <https://blog.apnic.net/2017/08/22/dealing-ipv6-fragmentation-dns> ». Il a même été proposé de ne jamais envoyer de datagrammes de taille supérieure à 1 280 octets <https://www.bortzmeyer.org/fragmentation-ip-1280.html>.

En pratique, le meilleur contournement de ce problème est de réduire la taille maximale des réponses EDNS. Par exemple, dans NSD :

```
ipv6-edns-size: 1460
```

Les réponses resteront à moins de 1 460 octets et ne seront donc en général pas fragmentées.

Les transferts de zone depuis les DM ont levé quelques problèmes. Les zones sont légèrement différentes d'un DM à l'autre (SOA et surtout signatures). Les transferts de zone incrémentaux (IXFR, RFC 1995), ne peuvent donc pas être utilisés : si un serveur racine interroge un DM, puis un autre, les résultats seront incompatibles. Ce cas, très spécifique à Yeti, n'est pas pris en compte par les logiciels. Les serveurs doivent donc utiliser le transfert complet (AXFR) uniquement (d'où le `AXFR` dans la configuration du serveur racine NSD vue plus haut). Ce n'est pas très grave, vu la petite taille de la zone racine.

Lors des essais de remplacement de la KSK (on sait que, depuis la parution de ce RFC, la KSK de la racine « officielle » a été successivement remplacée le 11 octobre 2018 <https://www.icann.org/news/announcement-2018-10-15-en>) quelques problèmes sont survenus. Par exemple, la documentation de BIND n'indiquait pas, lorsque le résolveur utilise l'option `managed-keys`, que celle-ci doit être configurée dans toutes les vues. (Au passage, j'ai toujours trouvé que les vues sont un système compliqué et menant à des erreurs déroutantes.)

La capture du trafic DNS avec les serveurs racine Yeti a entrainé d'autres problèmes (section 5.4 du RFC). Il existe plusieurs façons d'enregistrer le trafic d'un serveur de noms, de la plus courante (tcpdump avec l'option `-w`) à la plus précise (dnstap <http://dnstap.info/>). dnstap étant encore peu répandu sur les serveurs de noms, Yeti a utilisé une capture « brute » des paquets, dans des fichiers pcap qu'il fallait ensuite analyser. L'un des problèmes avec les fichiers pcap est qu'une connexion TCP, même d'une seule requête, va se retrouver sur plusieurs paquets, pas forcément consécutifs. Il faudra donc réassembler ces connexions TCP, par exemple avec un outil développé pour Yeti, PcapParser <https://github.com/RunxiaWan/PcapParser> (décrit plus longuement dans l'annexe D de notre RFC).

Les serveurs racine changent de temps en temps. Dans la racine « officielle », les changements des noms sont très rares. En effet, pour des raisons politiques, on ne peut pas modifier la liste des organisations qui gèrent un serveur racine. Vouloir ajouter ou retirer une organisation déclencherait une crise du genre « pourquoi lui ? ». L'ICANN est donc paralysée sur ce point. Mais les serveurs changent parfois d'adresse IP. C'est rare, mais ça arrive. Si les résolveurs ne changent pas leur configuration, ils auront une liste incorrecte. Un exemple de la lenteur avec laquelle se diffusent les changements d'adresses IP des serveurs racine est le cas de `j.root-servers.net` qui, **treize ans** après son changement d'adresse IP, continue à recevoir du trafic à l'ancienne adresse <https://indico.dns-oarc.net/event/24/session/10/contribution/10/material/slides/0.pdf>. Ceci dit, ce n'est pas très grave en

pratique, car, à l'initialisation du résolveur (RFC 8109), le résolveur reçoit du serveur racine consulté une liste à jour. Tant que la liste qui est dans la configuration du résolveur ne dévie pas trop de la vraie liste, il n'y a pas de problème, le résolveur finira par obtenir une liste correcte.

Mais Yeti est différent : les changements sont beaucoup plus fréquents et, avec eux, le risque que la liste connue par les résolveurs dévie trop. D'où la création d'un outil spécial, hintUpdate <https://github.com/BII-Lab/Hintfile-Auto-Update> (personnellement, je ne l'ai jamais utilisé, je modifie la configuration du résolveur, c'est tout). Un point intéressant d'hintUpdate est qu'il dépend de DNSSEC pour vérifier les informations reçues. Cela marche avec Yeti, où les noms des serveurs racine sont (théoriquement) signés, mais cela ne marcherait pas avec la racine officielle, root-servers.net n'étant pas signé.

Dernier problème, et rigolo, celui-ci, la compression inutile. En utilisant le logiciel Knot pour un serveur racine, nous nous sommes aperçus qu'il comprimait même le nom de la zone racine, faisant passer sa taille de un à deux octets. Une compression négative donc, légale mais inutile. À noter que cela plantait la bibliothèque Go DNS <https://miek.nl/2014/august/16/go-dns-package/>. Depuis, cette bibliothèque a été rendue plus robuste, et Knot a corrigé cette optimisation ratée.

La conclusion du RFC, en section 6, rappelle l'importance de disposer de bancs de test, puisqu'on ne peut pas faire courir de risques à la racine de production. La conclusion s'achève en proposant de chercher des moyens de rendre le DNS moins dépendant de la racine actuelle. Et, vu la mode actuelle, le mot de chaîne de blocs est même prononcé…