RFC 8324 : DNS Privacy, Authorization, Special Uses, Encoding, Characters, Matching, and Root Structure: Time for Another Look?

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 28 février 2018

Date de publication du RFC : Février 2018

https://www.bortzmeyer.org/8324.html

Le DNS est une infrastructure essentielle de l'Internet. S'il est en panne, rien ne marche (sauf si vous faites partie de la minorité qui fait uniquement des `ping -n` et des `traceroute -n`). S'il est lent, tout rame. Comme le DNS, heureusement, marche très bien, et s'est montré efficace, fiable et rapide, il souffre aujourd'hui de la malédiction des techniques à succès : on essaie de charger la barque, de lui faire faire plein de choses pour lesquelles il n'était pas prévu. D'un côté, c'est un signe de succès. De l'autre, c'est parfois fragilisant. Dans ce RFC individuel (qui exprime juste le point de vue d'un individu, et n'est pas du tout une norme IETF), John Klensin, qui ne participe plus activement au développement du DNS depuis des années, revient sur certaines de ces choses qu'on essaie de faire faire au DNS et se demande si ce n'est pas trop, et à partir de quel point il faudrait arrêter de « perfectionner » le DNS et plutôt passer à « autre chose » (« quand le seul outil qu'on a est un marteau, tous les problèmes ressemblent à des clous… »). Une bonne lecture pour celleszetceux qui ne veulent pas seulement faire marcher le DNS mais aussi se demander « pourquoi c'est comme ça ? » et « est-ce que ça pourrait être différent ? »

Améliorer le système petit à petit ou bien le remplacer complètement ? C'est une question que se posent régulièrement les ingénieurs, à propos d'un logiciel, d'un langage de programmation, d'un protocole réseau. À l'extrême, il y a l'ultra-conservateur qui ne voit que des inconvénients aux solutions radicales, à l'autre il y a l'ultra-optimiste qui en a marre des rustines et qui voudrait jeter le vieux système, pour le remplacer par un système forcément meilleur, car plus récent. Entre les deux, beaucoup d'informaticiens hésitent. L'ultra-conservateur oublie que les rustines successives ont transformé l'ancien système en un monstre ingérable et difficile à maintenir, l'ultra-optimiste croit naïvement qu'un système nouveau, rationnellement conçu (par lui…) sera à coup sûr plus efficace et moins bogué. Mais les deux camps, et tout celleszetceux qui sont entre les deux peuvent tirer profit de ce RFC, pour approfondir leur réflexion.

Klensin note d'abord que le DNS est vieux. La première réflexion à ce sujet était le RFC 799[1] en 1981, et le premier RFC décrivant le DNS est le RFC 882, en novembre 1983. (Paul Mockapetris a raconté le développement du DNS dans « "*Development of the Domain Name System*" <ftp://ftp.isi.edu/isi-pubs/rs-88-219.pdf> », j'ai fait un résumé des articles d'histoire du DNS <https://www.bortzmeyer.org/bind-dns-history.html>.) Le DNS remplaçait l'ancien système fondé sur un fichier centralisé de noms de machines (RFC 810, RFC 952, et peut-être aussi RFC 953). Tout n'est pas écrit : le DNS n'a pas aujourd'hui une spécification unique et à jour, aux RFC de base (RFC 1034 et RFC 1035, il faut ajouter des dizaines de RFC qui complètent ou modifient ces deux-ci, ainsi que pas mal de culture orale. (Un exemple de cette difficulté était que, pendant le développement du RFC 7816, son auteur s'est aperçu que personne ne se souvenait pourquoi les résolveurs envoyaient le FQDN complet dans les requêtes.) Plusieurs techniques ont même été supprimées comme les requêtes inverses (RFC 3425) ou comme les types d'enregistrement WKS, MD, MF et MG.

D'autres auraient dû être supprimées, car inutilisables en pratique, comme les classes (que prétendait utiliser le projet Net4D <https://www.bortzmeyer.org/net4d.html>), et qui ont fait l'objet de l'Internet-Draft « "*The DNS Is Not Classy : DNS Classes Considered Useless*" <https://datatracker.ietf.org/doc/draft-sullivan-dns-class-useless/> », malheureusement jamais adopté dans une IETF parfois paralysée par la règle du consensus.

D'autres évolutions ont eu lieu : le DNS original ne proposait aucun mécanisme d'options, tous les clients et tous les serveurs avaient exactement les mêmes capacités. Cela a changé avec l'introduction d'EDNS, dans le RFC 2671 en 1999.

Beaucoup d'articles ont été écrits sur les systèmes de nommage. (Le RFC recommande l'article de V. Cerf, « "*Desirable Properties of Internet Identifiers*" <http://ieeexplore.ieee.org/abstract/document/8114616/> », ou bien le livre « "*Signposts in Cyberspace : The Domain Name System and Internet Navigation*" <https://www.nap.edu/catalog/11258/signposts-in-cyberspace-the-domain-name
Je me permets de rajouter mes articles, « Inventer un meilleur système de nommage : pas si facile <https://www.bortzmeyer.org/no-free-lunch.html> », « Un DNS en pair-à-pair ? <https://www.bortzmeyer.org/dns-p2p.html> » et « Mon premier nom Namecoin enregistré <https://www.bortzmeyer.org/namecoin.html> ».)

Pourquoi est-ce que les gens ne sont pas contents du DNS actuel et veulent le changer (section 4 du RFC, la plus longue du RFC) ? Il y a des tas de raisons. Certaines, dit l'auteur, peuvent mener à des évolutions raisonnables du DNS actuel. Certaines nécessiteraient un protocole complètement nouveau, incompatible avec le DNS. D'autres enfin seraient irréalistes, quel que soit le système utilisé. La section 4 les passe en revue (rappelez-vous que ce RFC est une initiative individuelle, pas une opinion consensuelle à l'IETF).

Premier problème, les requêtes « multi-types ». À l'heure actuelle, une requête DNS est essentiellement composée d'un **nom** (QNAME, "*Query Name*") et d'un **type** (QTYPE, "*Query Type*", par exemple AAAA pour une adresse IP, TLSA pour une clé publique, etc). Or, on aurait parfois besoin de plusieurs types. L'exemple classique est celui d'une machine double-pile (IPv6 et le vieil IPv4), qui ne sait pas quelle version d'IP est acceptée en face et qui demande donc l'adresse IPv6 et l'adresse IPv4 du pair. Il n'y a actuellement pas de solution pour ce problème, il faut faire deux requêtes DNS. (Et, non, ANY ne résout pas ce problème, notamment en raison de l'interaction avec les caches : que doit faire un cache qui ne connait qu'une seule des deux adresses ?)

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc799.txt

Deuxième problème, la sensibilité à la casse. La norme originale prévoyait des requêtes insensibles à la casse (RFC 1034, section 3.1), ce qui semble logique aux utilisateurs de l'alphabet latin. Mais c'est plutôt une source d'ennuis pour les autres écritures (et c'est une des raisons pour lesquelles accepter l'Unicode dans les noms de domaine nécessite des méthodes particulières <https://www.bortzmeyer.org/pourquoi-idn-et-pas-un-dns-unicode.html>). Avec ASCII, l'insensibilité à la casse est facile (juste un bit à changer pour passer de majuscule en minuscule et réciproquement) mais ce n'est pas le cas pour le reste d'Unicode. En outre, il n'est pas toujours évident de connaitre la correspondance majuscule-minuscule (cf. les débats entre germanophones sur la majuscule de [Caractère Unicode non montré [2]]). Actuellement, les noms de domaine en ASCII sont insensibles à la casse et ceux dans le reste du jeu de caractères Unicode sont forcément en minuscules (cf. RFC 5890), libre à l'application de mettre ses propres règles d'insensibilité à la casse si elle veut, lorsque l'utilisateur utilise un nom en majuscules comme RÉUSSIR-EN.FR. On referait le DNS en partant de zéro, peut-être adopterait-on UTF-8 comme encodage obligatoire, avec normalisation NFC dans les serveurs de noms, mais c'est trop tard pour le faire.

En parlant d'IDN, d'ailleurs, ce sujet a été à l'origine de nombreuses discussions, incluant pas mal de malentendus (pour lesquels, à mon humble avis, l'auteur de RFC a une sérieuse responsabilité). Unicode a une particularité que n'a pas ASCII : le même caractère peut être représenté de plusieurs façons. L'exemple classique est le É qui peut être représenté par un point de code, U+00C9 ("*LATIN CAPITAL LETTER E WITH ACUTE*"), ou par deux, U+0045 ("*LATIN CAPITAL LETTER E*") et U+0301 ("*COMBINING ACUTE ACCENT*"). Je parle bien de la représentation en points de code, pas de celle en bits sur le réseau, qui est une autre affaire ; notez que la plupart des gens qui s'expriment à propos d'Unicode sur les forums ne connaissent pas Unicode. La normalisation Unicode vise justement à n'avoir qu'une forme (celle à un point de code si on utilise NFC) mais elle ne traite pas tous les cas gênants. Par exemple, dans certains cas, la fonction de changement de casse dépend de la langue (que le DNS ne connait évidemment pas). Le cas le plus célèbre est celui du i sans point U+0131, qui a une règle spécifique en turc. Il ne sert à rien de râler contre les langues humaines (elles sont comme ça, point), ou contre Unicode (dont la complexité ne fait que refléter celles des langues humaines et de leurs écritures). Le point important est qu'on n'arrivera pas à faire en sorte que le DNS se comporte comme M. Toutlemonde s'y attend, sauf si on se limite à un M. Toutlemonde étatsunien (et encore).

Les IDN ont souvent été accusés, y compris dans ce RFC, de permettre, ou en tout cas de faciliter, le hameçonnage par la confusion possible entre deux caractères visuellement proches. En fait, le problème n'est pas spécifique aux IDN (regardez `google.com` et `goog1e.com`) et les études montrent que les utilisateurs ne vérifient pas les noms <https://www.bortzmeyer.org/idn-et-phishing.html>, de toute façon. Bref, il s'agit de simple propagande de la part de ceux qui n'ont jamais vraiment accepté Unicode.

Les IDN nous amènent à un problème proche, celui des synonymes. Les noms de domaine `color.example` et `colour.example` sont différents alors que, pour tout anglophone, "*color*" <https://en.wiktionary.org/wiki/color#Alternative_forms> et "*colour*" <https://en.wiktionary.org/wiki/color#Usage_notes> sont « équivalents ». J'ai mis le mot « équivalent » entre guillemets car sa définition même est floue. Est-ce que « Saint-Martin » est équivalent à « St-Martin » ? Et est-ce que « Dupont » est équivalent à « Dupond » ? Sans même aller chercher des exemples comme l'équivalence entre sinogrammes simplifiés et sinogrammes traditionnels, on voit que l'équivalence est un concept difficile à cerner. Souvent, M. Michu s'agace « je tape `st-quentin.fr`, pourquoi est-ce que ça n'est pas la même chose que `saint-quentin-en-yvelines.fr` ? » Fondamentalement, la réponse est que le DNS ne gère pas les requêtes approximatives, et qu'il n'est pas évident que tout le monde soit d'accord sur l'équivalence de deux noms. Les humains se débrouillent avec des requêtes floues car ils ont un **contexte**. Si on est dans les Yvelines, je sais que « St-Quentin » est celui-ci alors que, si on est dans l'Aisne, mon interlocuteur parle probablement de celui-là. Mais le DNS n'a pas ce contexte.

2. Car trop difficile à faire afficher par LaTeX

Plusieurs RFC ont été écrit à ce sujet, RFC 3743, RFC 4290, RFC 6927 ou RFC 7940, sans résultats convaincants. Le DNS a bien sûr des mécanismes permettant de dire que deux noms sont équivalents, comme les alias (enregistrements CNAME) ou comme les DNAME du RFC 6672. Mais :

— Aucun d'entre eux n'a exactement la sémantique que les utilisateurs voudraient (d'où des propositions régulières d'un nouveau mécanisme comme les BNAME <https://datatracker.ietf.org/doc/draft-ietf-dnsext-aliasing-requirements>),
— Et ils supposent que quelqu'un ait configuré les correspondances, ce qui suscitera des conflits, et ne satisfera jamais tout le monde.

Même écrire un cahier des charges des « variantes <https://datatracker.ietf.org/doc/draft-ietf-dnse > » n'a jamais été possible. (C'est également un sujet sur lequel j'avais écrit un article <https://www.bortzmeyer.org/domaines-synchrones.html>.)

Passons maintenant aux questions de protection de la vie privée. L'auteur du RFC note que la question suscite davantage de préoccupations aujourd'hui mais ne rappelle pas que ces préoccupations ne sont pas irrationnnelles, elles viennent en grande partie de la révélation de programmes de surveillance massive comme MoreCowBell <https://www.bortzmeyer.org/morecowbell.html>. Et il « oublie » d'ailleurs de citer le RFC 7626, qui décrit en détail le problème de la vie privée lors de l'utilisation du DNS.

J'ai parlé plus haut du problème des classes dans le DNS, ce paramètre supplémentaire des enregistrements DNS (un enregistrement est identifié par trois choses, le nom, la classe et le type). L'idée au début (RFC 1034, section 3.6) était de gérer depuis le DNS plusieurs protocoles très différents (IP, bien sûr, mais aussi CHAOS et d'autres futurs), à l'époque où le débat faisait rage entre partisans d'un réseau à protocole unique (le futur Internet) et ceux et celles qui préféraient un "*catenet*", fondé sur l'interconnexion de réseaux techniquement différents. Mais, aujourd'hui, la seule classe qui sert réellement est IN (Internet) et, en pratique, il y a peu de chances que les autres soient jamais utilisées. Il a parfois été suggéré d'utiliser les classes pour partitionner l'espace de noms (une classe IN pour l'ICANN et créer une classe UN afin de la donner à l'UIT pour qu'elle puisse jouer à la gouvernance ?) mais le fait que les classes aient été très mal normalisées <https://www.ietf.org/mail-archive/web/ietf/current/msg103486.html> laisse peu d'espoir. (Est-ce que `IN example`, `CH example` et `UN example` sont la même zone ? Ont-ils les mêmes serveurs de noms ? Cela n'a jamais été précisé.)

Une particularité du DNS qui déroute souvent les nouveaux administrateurs système est le fait que les données ne soient que faiblement synchronisées : à un moment donné, il est parfaitement normal que plusieurs valeurs coexistent dans l'Internet. Cela est dû à plusieurs choix, notamment :

— Celui d'avoir plusieurs serveurs faisant autorité pour une zone, et sans mécanisme assurant leur synchronisation forte. Lorsque le serveur maître (ou primaire) est mis à jour, il sert immédiatement les nouvelles données, sans attendre que les esclaves (ou secondaires) se mettent à jour.
— Et le choix d'utiliser intensivement les **caches**, la mémoire des résolveurs. Si un serveur faisant autorité sert un enregistrement avec un TTL de 7 200 secondes (deux heures) et que, cinq minutes après qu'un résolveur ait récupéré cet enregistrement, le serveur faisant autorité modifie l'enregistrement, les clients du résolveur verront encore l'ancienne valeur pendant 7 200 - 300 = 6 900 secondes.

Cela a donné naissance à la légende de la propagation du DNS <https://www.bortzmeyer.org/dns-propagation.html> et aux chiffres fantaisistes qui accompagnent cette légende comme « il faut 24 h pour que le DNS se propage ».

Ces choix ont assuré le succès du DNS, en lui permettant de passer à l'échelle, vers un Internet bien plus grand que prévu à l'origine. Un modèle à synchronisation forte aurait été plus compliqué, plus fragile et moins performant.

Mais tout choix en ingéniérie a des bonnes conséquences **et** des mauvaises : la synchronisation faible empêche d'utiliser le DNS pour des données changeant souvent. Des perfectionnements ont eu lieu

(comme la notification non sollicitée du RFC 1996, qui permet aux serveurs secondaires d'être au courant rapidement d'un changement, mais qui ne marche que dans le cas où on connait tous les secondaires) mais n'ont pas fondamentalement changé le tableau. Bien sûr, les serveurs faisant autorité qui désireraient une réjuvénation <https://www.bortzmeyer.org/dns-propagation.html> plus rapide peuvent toujours abaisser le TTL mais, en dessous d'une certaine valeur (typiquement 30 à 60 minutes), les TTL trop bas sont parfois ignorés <https://www.bortzmeyer.org/forcer-ttl.html>.

Un autre point où les demandes de beaucoup d'utilisateurs rentrent en friction avec les concepts du DNS est celui des noms privés, des noms qui n'existeraient qu'à l'intérieur d'une organisation particulière, et qui ne nécessiteraient pas d'enregistrement auprès d'un tiers. La bonne méthode pour avoir des noms privés est d'utiliser un sous-domaine d'un domaine qu'on a enregistré (aujourd'hui, tout le monde peut avoir son domaine assez facilement, voir gratuitement <https://nic.eu.org/>), et de le déléguer à des serveurs de noms qui ne sont accessibles qu'en interne. Si on est l'association Example et qu'on est titulaire de `example.org`, on crée `priv.example.org` et on y met ensuite les noms « privés » (je mets privé entre guillemets car, en pratique, comme le montrent les statistiques des serveurs de noms publics, de tels noms fuitent souvent à l'extérieur, par exemple quand un ordinateur portable passe du réseau interne à celui d'un FAI public).

Il faut noter que beaucoup d'organisations, au lieu d'utiliser la bonne méthode citée ci-dessus, repèrent un TLD actuellement inutilisé (`.home`, `.lan`, `.private`...) et s'en servent. C'est une très mauvaise idée, car, un jour, ces TLD seront peut-être délégués, avec les risques de confusion que cela entrainera (cf. le cas de .box <https://github.com/felixfischer/node-red-contrib-fritzbox-presence/issues/2> et celui de .dev <https://ma.ttias.be/chrome-force-dev-domains-https-via-preloaded-hsts/>).

Les administrateurs système demandent souvent « mais quel est le TLD réservé pour les usages internes » et sont surpris d'apprendre qu'il n'en existe pas. C'est en partie pour de bonnes raisons (imaginez deux entreprises utilisant ce TLD et fusionnant...Ou simplement s'interconnectant via un VPN...Un problème qu'on voit souvent avec le RFC 1918.) Mais c'est aussi en partie parce que les tentatives d'en créer un se sont toujours enlisées dans les sables de la bureaucratie (personne n'a envie de passer dix ans de sa vie professionnelle à faire du lobbying auprès de l'ICANN pour réserver un tel TLD). La dernière tentative était celle de `.internal` mais elle n'a pas marché.

Il y a bien un registre des noms de domaines (pas uniquement des TLD) « à usage spécial <https://www.iana.org/assignments/special-use-domain-names/special-use-domain-names.xml> », créé par le RFC 6761. Il a malheureusement été gelé par l'IESG <https://ietf.org/blog/onion/> et fait l'objet de contestations (RFC 8244). Aucun des noms qu'il contient ne convient vraiment au besoin de ceux qui voudraient des noms de domaine internes (à part `.test` qui devrait logiquement être utilisé pour les bancs de test, de développement, etc). Le RFC note qu'un des principaux problèmes d'un tel registre est qu'il est impossible de garder à jour tous les résolveurs de la planète quand ce registre est modifié. On ne peut donc pas garantir qu'un nouveau TLD réservé sera bien traité de manière spéciale par tous les résolveurs.

Une caractéristique du DNS qui a suscité beaucoup de débats, pas toujours bien informés et pas toujours honnêtes, est l'existence de la racine du DNS, et des serveurs qui la servent. Lors de la mise au point du DNS, la question s'était déjà posée, certains faisant remarquer que cette racine allait focaliser les problèmes, aussi bien techniques que politiques. L'expérience a montré qu'en fait la racine marchait bien, mais cela n'a pas évité les polémiques. Le RFC note que le sujet est très chaud : qui doit gérer un serveur racine ? Où faut-il les placer physiquement ? Si l'"*anycast*" a largement résolu la seconde question (RFC 7094), la première reste ouverte. Le RFC n'en parle pas mais, si la liste des onze (ou douze, ça dépend comment on compte) organisations qui gèrent un serveur racine n'a pas évolué depuis vingt

ans, ce n'est pas pour des raisons techniques <https://yeti-dns.org/yeti/blog/2018/01/12/How-to-scale-the-root.html>, ni parce qu'aucune organisation n'est capable de faire mieux que les gérants actuels, mais tout simplement parce qu'il n'existe aucun processus pour supprimer ou ajouter un serveur racine. Comme pour les membres permanents du Conseil de Sécurité de l'ONU, on en reste au statu quo, aussi inacceptable soit-il, simplement parce qu'on ne sait pas faire autrement.

Le problème de la gestion de la racine n'est pas uniquement celui de la gestion des serveurs racine. Le contenu de la zone racine est tout aussi discuté. Si les serveurs racine sont les imprimeurs du DNS, le gérant de la zone racine en est l'éditeur. Par exemple, combien faut-il de TLD ? Si quelqu'un veut créer `.pizza`, faut-il le permettre ? Et `.xxx` ? Et `.vin`, que le gouvernement français avait vigoureusement combattu <https://www.numerama.com/magazine/29772-vin-wine-noms-de-domaine-menaces-axel.html> ? Ou encore `.home`, déjà largement utilisé informellement dans beaucoup de réseaux locaux, mais pour lequel il y avait trois candidatures à l'ICANN (rejetées peu de temps avant la publication du RFC). Ces questions, qui se prêtent bien aux jeux politiciens, occupent actuellement un bon bout des réunions ICANN.

La base technique à ces discussions est qu'il n'y a qu'une seule racine (RFC 2826). Son contrôle va donc forcément susciter des conflits. Un autre système de nommage que le DNS, si on le concevait de nos jours, pourrait éviter le problème en évitant ces points de contrôle. Les techniques à base de chaînes de blocs comme Namecoin sont évidemment des candidates possibles. Outre les problèmes pratiques (avec Namecoin, quand on perd sa clé privée, on perd son domaine <https://www.bortzmeyer.org/maman-j-ai-perdu-mon-namecoin.html>), la question de fond est « quelle gouvernance souhaite-t-on ? »

La question de la sémantique dans les noms de domaines est également délicate. L'auteur affirme que les noms de domaines sont (ou en tout cas devraient être) de purs identificateurs techniques, sans sémantique. Cela permet de justifier les limites des noms (RFC 1034, section 3.5) : s'ils sont de purs identificateurs techniques, il n'est pas nécessaire de permettre les IDN, par exemple. On peut se contenter des lettres ASCII, des chiffres et du tiret, la règle dite LDH, qui vient du RFC 952. Cette règle « "*Letters-Digits-Hyphen*" » a été une première fois remise en cause vers 1986 lorsque 3Com a voulu son nom de domaine `3com.com` (à l'époque, un nom devait commencer par une lettre, ce qui a été changé par la norme actuelle, RFC 1123). Mais cela laisse d'autres marques sans nom de domaine adapté, par exemple C&A ne peut pas avoir `c&a.fr`. Sans parler des cas de ceux et celles qui n'utilisent pas l'alphabet latin.

L'argument de Klensin est que ce n'est pas grave : on demande juste aux noms de domaine d'être des identificateurs uniques et non ambigus. Qu'ils ne soient pas très « conviviaux » n'est pas un problème. Inutile de dire que ce point de vue personnel ne fait pas l'unanimité.

Bien sûr, il y a aussi un aspect technique. Si on voulait, dit l'auteur ironiquement, permettre l'utilisation de la langue naturelle dans les noms de domaine, il faudrait aussi supprimer la limite de 63 caractères par composant (255 caractères pour le nom complet). Il est certain qu'il est difficile d'avoir des identificateurs qui soient à la fois utiles pour les programmes (simples, non ambigus) et pour les humains.

Le DNS n'est pas figé, et a évolué depuis ses débuts. Notamment, beaucoup de nouveaux types (RRTYPE, pour "*Resource Record Type*") ont été créés avec le temps (cf. RFC 6895). Ce sont, par exemple :

— NAPTR (RFC 3403), système que je trouve fort compliqué et qui n'a pas eu de succès,
— URI (RFC 7553) qui permet de stocker des URI dans le DNS (par exemple <https://www.bortzmeyer.org/dns-code-postal-lonlat.html>, `dig +short URI 78100.cp.bortzmeyer.fr` va vous donner un URI OpenStreetMap correspondant au code postal 78100),

— SRV (RFC 2782), qui généralise le vieux MX en permettant une indirection depuis le nom de domaine vers un nom de serveur (hélas, HTTP est le seul protocole Internet qui, stupidement, ne l'utilise pas).

Une observation à partir de l'étude du déploiement de tous les nouveaux types d'enregistrement est que ça se passe mal : pare-feux débiles qui bloquent les types qu'ils ne connaissent pas, interfaces de gestion du contenu des zones qui ne sont jamais mises à jour (bien des hébergeurs DNS ne permettent pas d'éditer URI ou TLSA, voir simplement SRV), bibliothèques qui ne permettent pas de manipuler ces types...Cela a entrainé bien des concepteurs de protocole à utiliser le type « fourre-tout » TXT. Le RFC 5507 explique ses avantages et (nombreux) inconvénients. (Le RFC 6686 raconte comment le type générique TXT a vaincu le type spécifique SPF.)

Aujourd'hui, tout le monde et son chien a un nom de domaine. Des noms se créent en quantité industrielle, ce qui est facilité par l'automatisation des procédures, et le choix de certains registres de faire des promotions commerciales. Il n'est pas exagéré de dire que, surtout dans les nouveaux TLD ICANN, la majorité des noms sont créés à des fins malveillantes. Il est donc important de pouvoir évaluer la réputation d'un nom : si `mail.enlargeyourzob.xyz` veut m'envoyer du courrier, puis-je utiliser la réputation de ce domaine (ce qu'il a fait précédemment) pour décider de rejeter le message ou pas ? Et si un utilisateur clique sur `http://www.bitcoinspaschers.town/`, le navigateur Web doit-il l'avertir que ce domaine a mauvaise réputation ? Le RFC, souvent nostalgique, rappelle que le modèle original du DNS, formalisé dans le RFC 1591, était que chaque administrateur de zone était compétent, responsable et honnête. Aujourd'hui, chacune de ces qualités est rare et leur combinaison est encore plus rare. L'auteur du RFC regrette que les registres ne soient pas davantage comptables du contenu des zones qu'ils gèrent, ce qui est un point de vue personnel et très contestable : pour un TLD qui est un service public, ce serait une violation du principe de neutralité.

Bref, en pratique, il est clair aujourd'hui qu'on trouve de tout dans le DNS. Il serait donc souhaitable qu'on puisse trier le bon grain de l'ivraie mais cela présuppose qu'on connaisse les frontières administratives. Elles ne coïncident pas forcément avec les frontières techniques (`.fr` et `gouv.fr` sont actuellement dans la même zone alors que le premier est sous la responsabilité de l'AFNIC et le second sous celle du gouvernement français). Rien dans le DNS ne les indique (le point dans un nom de domaine indique une frontière de domaine, pas forcément une frontière de zone, encore moins une frontière de responsabilité). Beaucoup de légendes circulent à ce sujet, par exemple beaucoup de gens croient à tort que tout ce qui se trouve avant les deux derniers composants d'un nom est sous la même autorité que le nom de deuxième niveau (cf. mon article sur l'analyse d'un nom <`https://www.bortzmeyer.org/parties-nom-domaine.html`>). Il n'y a pas actuellement de mécanisme standard et sérieux pour déterminer les frontières de responsabilité dans un nom de domaine. Plusieurs efforts avaient été tentés à l'IETF mais ont toujours échoué <`https://www.bortzmeyer.org/fin-dbound.html`>. La moins mauvaise solution, aujourd'hui, est la "*Public Suffix List*".

Beaucoup plus technique, parmi les problèmes du DNS, est celui de la taille des paquets. Car la taille compte. Il y a très très longtemps, la taille d'une réponse DNS était limitée à 512 octets. Cette limite a été supprimée en 1999 avec le RFC 2671 (c'est d'ailleurs une excellente question pour un entretien d'embauche lorsque le candidat a mis « DNS » dans la liste de ses compétences : « quelle est la taille maximale d'une réponse DNS ? »). En théorie, les réponses peuvent désormais être plus grandes (la plupart des serveurs sont configurés pour 4 096 octets) mais on se heurte à une autre limite : la MTU de 1 500 octets tend à devenir une valeur sacrée, et les réponses plus grandes que cette taille ont du mal à passer, par exemple parce qu'un pare-feu idiot bloque les fragments IP, ou parce qu'un pare-feu tout aussi crétin bloque l'ICMP, empêchant les messages "*Packet Too Big*" de passer (cf. RFC 7872).

Bref, on ne peut plus trop compter sur la fragmentation, et les serveurs limitent parfois leur réponse en UDP (TCP n'a pas de problème) à moins de 1 500 octets pour cela.

Une section entière du RFC, la 5, est consacrée au problème de la requête inverse, c'est-à-dire comment trouver un nom de domaine en connaissant le contenu d'un enregistrement lié à ce nom. La norme historique prévoyait une requête spécifique pour cela, IQUERY (RFC 1035, sections 4.1.1 et 6.4). Mais elle n'a jamais réellement marché (essentiellement parce qu'elle suppose qu'on connaisse déjà le serveur faisant autorité...pour un nom de domaine qu'on ne connait pas encore) et a été retirée dans le RFC 3425. Pour permettre quand même des « requêtes inverses » pour le cas le plus demandé, la résolution d'une adresse IP en un nom de domaine, un truc spécifique a été développé, l'« arbre inverse » `in-addr.arpa` (puis plus tard `ip6.arpa` pour IPv6). Ce truc n'utilise pas les IQUERY mais des requêtes normales, avec le type PTR. Ainsi, l'option `-x` de dig permet à la fois de fabriquer le nom en `in-addr.arpa` ou `ip6.arpa` et de faire une requête de type PTR pour lui :

```
% dig -x 2001:678:c::1
...
;; ANSWER SECTION:
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.c.0.0.0.8.7.6.0.1.0.0.2.ip6.arpa. 172796 IN PTR d.nic.fr.
```

Cette technique pose des problèmes dans le cas de préfixes IPv4 ne tenant pas sur une frontière d'octets (cas traité par le RFC 2317). Globalement, c'est toujours resté un truc, parfois utile, souvent surestimé, mais ne fournissant jamais un service général. Une tentative avait été faite à l'IETF pour décrire ce truc et son utilisation, mais elle n'a jamais abouti (`draft-ietf-dnsop-reverse-mapping-considerations`), la question étant trop sensible (même écrire « des gens trouvent que les enregistrements PTR sont utiles, d'autres personnes ne sont pas d'accord », même une phrase comme cela ne rencontrait pas de consensus).

Enfin, la section 7 du RFC est consacrée à un problème difficile : le DNS ne permet pas de recherche floue. Il faut connaitre le nom exact pour avoir les données. Beaucoup d'utilisateurs voudraient quelque chose qui ressemble davantage à un moteur de recherche. Ils n'ont pas d'idée très précise sur comment ça devrait fonctionner mais ils voudraient pouvoir taper « st quentin » et que ça arrive sur `www.saint-quentin-en-yvelines.fr`. Le fait qu'il existe plusieurs villes nommées Saint-Quentin ne les arrête pas ; ils voudraient que ça marche « tout seul ». Le DNS a un objectif très différent : fournir une réponse exacte à une question non ambigüe.

Peut-être aurait-on pu développer un service de recherche floue au dessus du DNS. Il y a eu quelques réflexions à ce sujet (comme le projet IRNSS <`https://datatracker.ietf.org/wg/irnss/about/`> ou comme le RFC 2345) mais ce n'est jamais allé très loin. En pratique, les moteurs de recherche jouent ce rôle, sans que l'utilisateur comprenne la différence <`https://www.bortzmeyer.org/identificateur-vs-m`
`html`>. Peu d'entre eux savent qu'un nom de domaine, ce n'est pas comme un terme tapé dans un moteur de recherche. On voit aussi bien des gens taper un nom de domaine dans la boîte de saisie du terme de recherche, que des gens utiliser le moteur de recherche comme fournisseurs d'identificateurs (« pour voir notre site Web, tapez "trucmachin" dans Google »). Le dernier clou dans le cercueil de la compréhension de la différence entre identificateur et moteur de recherche a été planté quand les navigateurs ont commencé à fusionner barre d'adresses et boite de saisie de la recherche.