RFC 6950 : Architectural Considerations on Application Features in the DNS

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 10 octobre 2013

Date de publication du RFC : Octobre 2013

https://www.bortzmeyer.org/6950.html

Le DNS est un des plus grands succès de l'Internet : une base de données répartie, fiable, sur laquelle on peut compter, et qui est très souple dans ses usages, lui permettant d'être utilisée pour beaucoup de choses (seuls les journalistes écrivent encore que « le DNS sert à trouver une adresse IP à partir d'un nom de domaine » : la réalité est bien plus riche). Mais, justement, la disponibilité et l'efficacité du DNS font qu'il est utilisé par beaucoup d'applications dont les auteurs ne sont pas forcément conscients des forces et faiblesses du DNS. Ne faudrait-il pas les appeler à réfléchir deux secondes sur ces forces et ces faiblesses ?

Le DNS permet de récupérer des données (quelconques : ce ne sont pas forcément des adresses IP) à partir d'un nom de domaine. Il est même utilisé par des applications dont les identifiants ne sont pas des noms de domaine : une petite transformation pour convertir l'identifiant en nom de domaine et hop (ONS, ENUM, etc). Le premier RFC à avoir décrit l'usage du DNS pour autre chose qu'une correspondance nom-¿adresse est le RFC 974 [1] en 1986. C'est dire que c'est ancien et cela explique mes critiques sévères des ignorants qui continuent à considérer le DNS comme un simple moyen de trouver des adresses. Depuis ce RFC 974, qui introduisait l'enregistrement MX, l'idée a suivi son chemin. Un mécanisme plus général, le SRV, a été introduit par le RFC 2052 (remplacé depuis par le RFC 2782). Et, pour faire des mécanismes de délégation aussi compliqués qu'on le souhaite, il y a le NAPTR créé par le RFC 2168, généralisé ensuite par le RFC 3401. Enfin, il y a des applications qui mettent simplement leurs données dans le très général et pas structuré enregistrement TXT, ce que fait par exemple DKIM (RFC 6376).

Mais ce succès, comme tous les succès, a aussi son revers : les applications demandent de plus en plus au DNS alors que son vaste déploiement est largement dû à sa focalisation sur un petit nombre de fonctions qu'il réalise très bien. La mentalité fréquente aujourd'hui de « on met tout dans le DNS » (illustrée

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc974.txt

par un gadget fameux <http://www.cafepress.com/nxdomain/8592477>) est parfois en conflit avec les principes du DNS. La confidentialité est un bon exemple : elle était délibérement exclue du cahier des charges du DNS (et à juste titre : lisez le RFC 3414 si vous voulez un exemple de la complexité que la confidentalité apporte à un protocole requête-réponse). Vouloir l'ajouter aujourd'hui risque fort de faire perdre au DNS ses propriétés intéressantes (il n'est pas cité par ce RFC, mais c'est l'une des raisons de l'échec de DNScurve <https://www.bortzmeyer.org/dnscurve.html>, qui essayait de faire trop de choses). Bien des applications qui veulent utiliser le DNS ont réclamé une certaine dose de confidentialité, alors que le DNS doit une partie de son efficacité au fait que toutes les données sont publiques. Ainsi, un cache n'a pas à s'inquiéter si les données qu'il mémorise doivent être servies à tous ou pas.

Ce nouveau RFC de l'IAB vise donc à aider les concepteurs d'applications en exposant clairement ce que le DNS fait **bien**, ce qu'il **peut** faire, et ce qu'il ne sait **pas** faire. C'est un excellent document pour qui veut comprendre en détail le DNS et, malgré sa longueur, il mérite d'être lu attentivement. Il fournit les informations nécessaires pour que ledit concepteur puisse répondre intelligemment à la question « Dois-je mettre cette information dans le DNS ou bien ailleurs ? » Il complète le RFC 5507, qui restait plutôt sur les questions de syntaxe, alors que ce nouveau RFC 6950 adopte une vue plus générale, plus tournée vers l'architecture du système. Pour le résumer en deux mots : le DNS doit son succès à ce qu'il n'essaie pas de résoudre tous les problèmes mais un ensemble bien précis de problèmes, pour lesquels il est une bonne solution. Comme il existe d'autres protocoles que le DNS, une application ou un service qui envisage d'utiliser le DNS dit sérieusement étudier s'il est vraiment la solution la plus adaptée (parfois oui mais parfois non).

La section 2 de notre RFC présente un certain nombre d'usages du DNS par des applications. Il commence évidemment par le routage du courrier, avec les enregistrements MX. Le MX était la première utilisation du DNS pour faire autre chose que de la simple traduction de nom en adresse IP. En partie droite d'un MX, on trouve le nom du ou des serveurs qui gèrent la messagerie pour le domaine en partie gauche. Bien sûr, une **convention de nommage** (du genre « le serveur de messagerie de `example.org` se nomme `mail.example.org` ») aurait pu jouer un rôle similaire. Mais les MX sont plus souples (le serveur d'un domaine n'est pas obligé d'avoir un nom dans le domaine, cf. RFC 4367) et offrent des possibilités supplémentaires (plusieurs serveurs, avec des priorités différentes). Mais le MX est spécifique au courrier électronique. Les enregistrements SRV, créés par le RFC 2052 (aujourd'hui RFC 2782), ont étendu le principe à tous les protocoles qui voulaient en profiter (avec des choses en plus comme l'indication du numéro de port ou bien comme la répartition de charge). Tous les protocoles créés depuis utilisent ces enregistrements, à la triste exception de HTTP qui, stupidement (l'avis est le mien, il n'est pas dans le RFC), ne fournit pas de mécanisme pour trouver le serveur d'un domaine (obligeant à utiliser des conventions de nommage comme `www.example.com` ou, pire, à mettre une adresse IP sur le domaine lui-même, et empêchant d'utiliser le DNS pour la répartition de charge et la résilience, ce qu'auraient permis les SRV).

Autre service fourni aux applications par le DNS, les enregistrements NAPTR. L'idée au début était de pouvoir trouver n'importe quel identificateur, même distinct d'un nom de domaine, dans le DNS (RFC 2915). Les NAPTR permettent de spécifier des transformations complexes depuis une famille d'identificateurs, vers les noms de domaine. L'idée est ancienne. Sa première manifestation avait été le domaine `in-addr` dans le RFC 883 (c'était un TLD à l'époque, il est devenu `in-addr.arpa` dans le RFC 973). Son but était de permettre de traduire des adresses IP en noms, en convertissant d'abord l'adresse IP en un nom de domaine se terminant en `in-addr`. Ce genre de transformation textuelle sur un identificateur pour en faire un nom de domaine a ensuite été reprise par `tpc.int` (RFC 1530, un mécanisme pour router des appels téléphoniques via l'Internet, qui allait déboucher sur ENUM). Ces mécanismes ne changent pas le protocole DNS mais ils changent la manière dont on se sert du DNS, et apportent de nouveaux utilisateurs, ayant de nouvelles attentes.

Une des demandes de ces « nouvelles » applications est de stocker des données quelconques dans le DNS. On peut mettre ce qu'on veut dans un URI (et même directement des données, cf. RFC 2397). Donc,

comme un NAPTR mène à un URI, il peut nous mener à n'importe quoi. Mais, avant même cette astuce, le DNS pouvait déjà stocker n'importe quoi. Il existe même un type d'enregistrement, TXT, spécialement prévu pour des données non structurées. La souplesse du TXT et son absence de contraintes ont attiré plein de gens, malgré les conseils de prudence du RFC 5507. Il y a même eu une proposition de structurer le contenu des TXT (RFC 1464).

Bon, mais qui y a-t-il de mal à ce que les gens se servent du DNS pour y mettre des informations ? En combinant les différentes techniques vues ci-dessus, on pourrait faire tenir n'importe quel protocole requête/réponse dans le DNS. Mais ce n'est pas complètement exact. La section 3 décrit les défis auxquels son succès confronte le DNS. Certes, le DNS est une base de données répartie, fiable, et rapide. Mais lorsqu'on dit « base de données », certains utilisateurs voudraient que le DNS fournisse des services classiques des bases de données, comme la confidentialité, le contrôle d'accès, l'introspection (utiliser le protocole d'accès pour découvrir le schéma) et la possibilité de requêtes structurées complexes, un peu comme ce que fournit SQL. Mais le DNS n'a pas été conçu pour cela et ce RFC argumente que, plutôt que de forcer le DNS à fournir ces services, on utiliserait mieux son temps à développer d'autres protocoles ou tout simplement à utiliser d'autres protocoles existants. Le succès du DNS et ses qualités viennent justement de ce qu'il n'essayait pas de tout faire.

Les exemples donnés par le RFC sont tous empruntés au monde ENUM (RFC 6116) mais les questions soulevées ne sont pas spécifiques à ENUM.

D'abord, les critères de recherche : dans le DNS, la clé d'accès est le triplet {nom de domaine, classe, type}, par exemple (en syntaxe dig) `www.afnic.fr. IN AAAA`, et la correspondance doit être exacte (pas de recherche floue). Or, certains souhaiteraient pouvoir exprimer des requêtes plus riches. Des tentatives ont été faites pour mettre des critères supplémentaires dans le nom de domaine lui-même (par exemple, pour ENUM, `tg011.0.0.4.5.4.3.4.1.7.5.1.e164.arpa` pour ajouter au nom normal, `0.0.4.5.4.3.4.1.7.5.1.e164.arpa`, qui est dérivé du numéro de téléphone, le "*trunk group*" `tg011`) mais cela devient vite pénible lorsque le nombre de critères augmente.

Autre solution envisagée, EDNS (RFC 6891), en passant les critères supplémentaires comme options EDNS. Mais cela ne marche pas comme certains le prévoient car EDNS est de saut en saut et pas de bout en bout : un serveur DNS relais ne va pas transmettre les options EDNS.

Enfin, les requêtes complexes posent un problème pour les caches, si fréquents avec le DNS. Les caches actuels considèrent que la réponse ne varie qu'avec le triplet {nom de domaine, classe, type} et pourrait donc garder en cache à tort des réponses faites à des requêtes plus complexes.

Autre demande fréquente des nouveaux utilisateurs du DNS, avoir des réponses « à la tête du client », dépendant de l'émetteur de la question. Il est par exemple courant aujourd'hui de servir des réponses différentes selon l'adresse IP de la source (option `view` de BIND). Tant que cela ne sert qu'à présenter des versions adaptés d'un contenu (un portail Web différent selon le pays d'origine de la requête), sans problème de sécurité, ce n'est pas trop grave : une erreur n'aura pas de conséquences trop ennuyeuses. Pour des cas plus sensibles, cela peut être gênant. D'autant plus que l'adresse IP de la source n'est pas celle du vrai client, mais celle du résolveur qu'il utilise. Dans certains cas (Google Public DNS <`https://www.bortzmeyer.org/google-dns.html`>), la distance entre les deux peut être énorme. Une option EDNS a été proposée pour que le résolveur puisse indiquer au serveur faisant autorité la vraie adresse IP du client mais elle n'est pas encore adoptée (entre autres, elle pose des problèmes si le client a une adresse IP privée, genre RFC 1918).

Le DNS a d'autres limites lorsqu'on veut l'utiliser comme base de données générique, par exemple la taille des noms de domaine (limitée à 63 caractères par composant) et la taille des réponses. Quelle limite

de taille ? L'ancienne limite de 512 octets n'est normalement plus qu'un souvenir (mais il existe encore des pare-feux bogués ou mal gérés qui imposent cette limite) mais il y a deux autres seuils derrière, la MTU (si la réponse est plus grosse, on risque de la fragmentation) et les 4 096 octets qui sont, en pratique, la limite de la plupart des serveurs. Si la réponse est un URI, notez que le RFC 2397 sur les URI `data:` indique que ceux-ci doivent être « courts » mais il ne définit pas cet adjectif. Le RFC note que, dans le contexte d'ENUM, stocker des sonneries de téléphone rigolotes sous forme de fichiers MP3 dans un URI `data:` n'est probablement pas raisonnable.

En outre, le DNS reposant sur UDP, qui ne garantit pas l'adresse IP source, des données de grande taille augmentent les risques d'attaque avec amplification (RFC 4732, section 3). Dans ces attaques, le méchant émet une requête DNS (de petite taille, donc) en usurpant l'adresse IP source de sa victime. Le serveur va alors répondre à celle qu'il croit être l'émetteur et la réponse est souvent bien plus grande que la question (le RFC cite l'exemple d'un certificat stocké dans le DNS, comme dans le RFC 4398). L'attaquant pourra donc obtenir ainsi un trafic plus important que ce qu'il peut lui-même générer. C'est d'ailleurs une des raisons pour lesquels les serveurs de `.com`, par exemple, limitent leurs réponses à 1 460 octets. Bref, malgré EDNS, on ne peut pas espérer faire passer dans le DNS des données de taille quelconque. Le DNS est prévu pour des informations courtes.

Autre limite du DNS lorsqu'on essaie de s'en servir comme d'une base de données générique, la non-correspondance des frontières administratives avec celles du DNS : les frontières des composants d'un nom de domaine ne sont pas forcément celles qu'on voudrait. Par exemple, pour la téléphonie, les anciennes numérotations étaient très hiérarchiques (et correspondaient donc bien au DNS) alors que, depuis la portabilité des numéros de téléphone, ce n'est plus le cas et il n'est donc pas évident de savoir où déléguer les noms ENUM. Ce n'est pas juste un problème esthétique : le bon fonctionnement du DNS dépend des délégations qui sont « cachées » (gardées en mémoire) dans les résolveurs, et qui épargnent à la racine la grande majorité des requêtes. Avec la portabilité des numéros téléphoniques, il faut interroger la racine ENUM pour tout numéro (puisqu'on ne peut pas savoir à l'avance quel opérateur téléphonique le gère). C'est d'autant plus ennuyeux pour la racine ENUM que les correspondances entre un numéro et un opérateur changent et qu'on souhaite souvent une portabilité rapide (de l'ordre de quinze minutes), peu compatible avec une racine simple et efficace.

Si vous pensez que ce n'est pas si grave, que `.com` est un espace plat avec de nombreux noms et des changements rapides, ce qui démontre que le DNS peut s'en tirer, pensez que dans les seuls États-Unis, il y a trois cents millions de numéros de téléphone attribués, trois fois la taille de `.com`.

Le problème n'est évidemment pas spécifique à ENUM : si on créait un mécanisme de portabilité pour les adresses IP, les domaines comme `in-addr.arpa` auraient les mêmes problèmes.

La section 4 est entièrement consacrée à un problème particulier qui a fait couler beaucoup d'encre, le désir d'avoir des réponses DNS « à la tête du client ». Officiellement, le DNS présent une vue unique à tous les utilisateurs (ce point est développé dans le RFC 2826) qui affirme que les noms doivent être uniques et donc donner un résultat unique. Mais il existe une forte demande pour avoir des noms qui ne fonctionnent que dans un espace privé (à l'intérieur d'une entreprise, par exemple), afin de limiter l'accès à certaines ressources. Il existe plusieurs solutions techniques pour avoir des réponses différentes en local et en public mais aucune n'est parfaitement satisfaisante. Par exemple, si on utilise des noms locaux avec un TLD bidon comme `.prive` ou `.local`, ces noms « fuiront » un jour ou l'autre, seront vus à l'extérieur, ce qui générera de la confusion (section 3.3 du RFC 5507).

Bien, après toutes ces critiques et toutes ces limites du DNS, quels conseils pratiques donner à ceux et celles qui voudraient quand même utiliser le DNS ? La section 5 est composée d'excellents conseils pour les développeurs d'applications ou de services utilisant le DNS. En gros, le DNS sera sans doute une bonne solution si le service qui veut l'utiliser a toutes ces propriétés :

— Les données à publier peuvent être distribuées (et mémorisées dans les caches) en utilisant les mécanismes standards du DNS,
— Les données sont accessibles par des clés qu'on peut convertir en noms de domaine sans violer leur sémantique,
— Les clés en question (les noms de domaine) doivent être utilisées dans leur totalité (pas de recherche sur un nom partiel),
— Les réponses ne dépendent pas du demandeur,
— Une fois ses informations obtenues, l'application utilisera le DNS pour trouver le pair à contacter.

Si une seule de ces propriétés manque, le DNS n'est peut-être pas la bonne solution pour le problème. À l'inverse, de bons signaux d'alarme, indiquant qu'on utilise le DNS à tort, sont :

— On a besoin de rajouter des données indiquant les frontières administratives (ce que les applications n'ont normalement pas besoin de connaître),
— Les identificateurs utilisés ont une sémantique ou un modèle radicalement différent de celui du DNS (par exemple, ils sont dans un espace plat, alors que le DNS doit son efficacité à son modèle hiérarchique),
— On veut distribuer des données à certaines personnes et pas à d'autres,
— Les réponses sont de grande taille, poussant les limites du DNS trop loin.

Bon, mais si on n'utilise pas le DNS, alors quoi ? Le RFC suggère que, dans beaucoup de cas de protocole requête/réponse, HTTP convient parfaitement (par exemple, il a de l'authentification, il peut transporter des données de taille quelconque, il est largement déployé...)

Pendant l'élaboration de ce RFC, des commentaires intéressants ont été enregistrés dans le système de suivi des tâches <http://trac.tools.ietf.org/group/iab/trac/query?status=assigned&status=closed&status=new&status=reopened&component=draft-iab-dns-applications>.