[DNS] CNAME à l'apex d'une zone, pourquoi et comment ?

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 23 septembre 2018

http://www.bortzmeyer.org/cname-apex.html

Un problème courant que rencontrent les techniciens débutant en DNS est « je voudrais mettre un enregistrement CNAME - un alias - à l'apex de ma zone DNS mais l'ordinateur ne veut pas ». Pourquoi est-ce refusé ? Comment l'autoriser sans casser tout l'Internet ? La discussion dure depuis de nombreuses années et, je vous révèle tout de suite la conclusion de cet article, n'est pas près de se terminer.

Commençons par une description concrète du problème. Alice, technicienne DNS, a été informée par le webmestre de sa boîte (nommée Michu SA et ayant le nom de domaine `michu.example`) que le serveur Web de ladite boîte est désormais hébergé par un CDN nommé Example et qu'il est accessible par le nom `michu-sa.example-cdn.net`. Et le webmestre voudrait que les visiteurs puissent juste taper `https://michu.example/`. Alice connait assez le DNS pour savoir qu'il y a des alias (un type d'enregistrement nommé CNAME) et elle met donc dans le fichier de zone :

```
michu.example. IN CNAME michu-sa.example-cdn.net.
```

Ça devrait contenter tout le monde, pense-t-elle. Mais, si elle connait assez le DNS pour savoir que le type d'enregistrement CNAME existe, elle ne le connait pas assez pour avoir lu le RFC 1034 [1], section 3.6.2 : "*If a CNAME RR is present at a node [a node in the domain name tree, so a domain name], no other data should be present*". Et c'est le drame, au chargement de la zone, NSD dit "*error : /etc/nsd/michu.example :19 : CNAME and other data at the same name*". Si Alice avait utilisé BIND, elle aurait eu une erreur similaire : "*dns_master_load : michu.example :19 : michu.example : CNAME and other data*". L'enregistrement CNAME rentre en conflit avec les enregistrements qu'on trouve à l'apex d'une zone, comme le SOA et les NS. La solution simple ne marche donc pas.

```
www.michu.example. IN CNAME michu-sa.example-cdn.net.
```

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc1034.txt`

aurait marché mais le service communication voudrait un URL sans `www` devant.

Au passage, pourquoi est-ce que c'est interdit de mettre un alias et d'autres données au même nom ? La principale raison est qu'un enregistrement CNAME est censé créer un alias, un synonyme. Si on pouvait écrire :

```
michu.example. IN AAAA 2001:db8::ff
michu.example. IN CNAME michu-sa.example-cdn.net.
```

Et qu'un client DNS demande l'enregistrement de type AAAA pour `michu.example`, que faudrait-il lui répondre ? `2001:db8::ff`, ou bien l'adresse IPv6 de `michu-sa.example-cdn.net` ? Avec l'interdiction de coexistence du CNAME et d'autres données, le problème est réglé.

Bon, se dit Alice, ça ne marche pas, mais on peut essayer autrement : cherchons l'adresse IP correspondant au nom `michu-sa.example-cdn.net` et mettons-là dans le fichier de zone :

```
% dig +short AAAA michu-sa.example-cdn.net
2001:db8:1000:21::21:4
```

```
michu.example. IN AAAA 2001:db8:1000:21::21:4
```

Cette fois, ça marche, mais il y a plusieurs inconvénients :

— Le principal est que le CDN peut changer l'adresse IP à tout moment, sans prévenir. Au minimum, il faudra donc retester l'adresse, et changer la zone. À la main, aucune chance que ce soit fait assez souvent (il faut respecter le TTL du CDN). Il faudra donc l'automatiser.

— Des CDN renvoient une adresse IP différente selon l'adresse IP source du client DNS, afin de diriger vers un serveur « proche ». L'astuce d'Alice casse ce service.

Bref, que faire ? Plusieurs sociétés ont développé une solution non-standard ne marchant que chez eux <https://support.cloudflare.com/hc/en-us/articles/200169056-CNAME-Flattening-RFC-co
L'IETF, dont c'est le rôle de développer des solutions standards, s'est penchée à de nombreuses reprises sur la question, et d'innombrables brouillons ont été produits, sans qu'un consensus ne se dégage. Déjà, il n'y a pas d'accord clair sur le cahier des charges. (Le scénario d'usage présenté plus haut n'est qu'un des scénarios possibles.) Je ne vais pas présenter tous ces brouillons, juste dégager les pistes de solution. Mais, comme vous le verrez, aucune n'est parfaite. Une des contraintes fortes est qu'on sait bien que, quelle que soit la décision prise par l'IETF, le nouveau logiciel, mettant en œuvre la décision, ne sera pas déployé immédiatement : il faudra des années, voire davantage, pour qu'une part significative des clients et serveurs DNS soient à jour. Il faudra donc, non seulement modifier les règles du DNS, mais également spécifier ce qu'il faudra faire pendant la très longue période de transition. (Et, s'il vous plait, pas de yakafokon du genre « les gens n'ont qu'à mettre à jour leur logiciel plus souvent ». Cela n'arrivera pas.)

Voici maintenant les diverses pistes envisagées. Avant de dire « ah, celle-ci a l'air cool », prudence. Rappelez-vous qu'il existe de nombreux cas (minimisation de la requête - RFC 7816 - ou pas, résolveur chaud - ayant déjà des informations dans sa mémoire - ou pas), et que la période de transition va être très longue, période pendant laquelle il faut que cela marche pour les anciens et pour les modernes.

Première idée, décider qu'il n'y a pas de problème. On laisse le DNS comme il est et le problème doit être entièrement traité côté avitaillement (le type qui édite le fichier de zone, ou bien le logiciel qui le produit). C'est l'idée de base du hack utilisé par Alice plus haut. Cela peut se faire avec un script shell du genre :

```
#!/bin/sh

zonefile=$1

while :
do
    target=$(awk '/ALIAS/ {print $3}'  $zonefile)
    ttl=$(dig A $target | awk "/^$target.*IN\s+A/ {print \$2}" | head -n 1)
    ip=$(dig A $target | awk "/^$target.*IN\s+A/ {print \$5}" | head -n 1)
    sed "s/IN.*ALIAS.*/  IN $ttl   A  $ip/" $zonefile > ${zonefile}-patched
    echo "Patched with IP address $ip"
    sleep $ttl
done
```

(En production, on voudra probablement quelque chose de plus propre, notamment en gestion d'erreurs, et de traitement des réponses multiples.)

Comme indiqué plus haut, cela marche, mais cela ne permet pas de tirer profit des caractéristiques du CDN, par exemple de la réponse différente selon le client. On peut voir cette variation de la réponse, en demandant à cent sondes RIPE Atlas <https://atlas.ripe.net/> :

```
% blaeu-resolve --requested 100 --type A www.elysee.fr
[208.178.167.254 4.26.226.126 4.27.28.126 8.27.4.254] : 1 occurrences
[8.254.28.126 8.254.45.254 8.27.151.253 8.27.226.126] : 1 occurrences
[205.128.73.126 206.33.35.125 209.84.20.126 8.27.243.253] : 1 occurrences
[207.123.56.252 4.26.228.254 4.27.28.126 8.26.223.254] : 1 occurrences
[205.128.90.126 209.84.9.126 8.254.214.254 8.254.94.254] : 1 occurrences
[8.254.164.126 8.254.209.126 8.254.210.126 8.27.9.254] : 1 occurrences
[4.23.62.126 4.26.227.126 8.254.173.254 8.254.95.126] : 1 occurrences
[207.123.39.254 8.254.196.126 8.254.219.254 8.27.5.254] : 2 occurrences
[120.28.42.254 36.66.10.126] : 1 occurrences
...
```

Deuxième idée, relâcher la contrainte donnée dans le RFC 1034 et autoriser le CNAME à l'apex (note au passage : beaucoup de gens sont nuls en arbrologie et appellent incorrectement l'apex la racine, ce qui a un tout autre sens dans le DNS). L'expérience a été tentée par Ond[Caractère Unicode non montré [2]]ej Sur[Caractère Unicode non montré] au hackathon de l'IETF <https://www.ietf.org/how/runningcode/hackathons/102-hackathon/> à Montréal en juillet 2018 (les résultats complets de l'équipe DNS sont dans cette présentation <https://github.com/IETF-Hackathon/ietf102-project-presentations/blob/master/dns_hackathon-presentation.pdf>). Une fois le serveur faisant autorité modifié pour autoriser le CNAME à l'apex, on l'interroge via un résolveur. En gros, dans la plupart des cas, le CNAME masque les autres enregistrements (ce qui est logique puisqu'il est censé être seul). Ainsi, si on a :

```
michu.example.     IN  CNAME   foobar.example.com.
                   IN  MX  10    gimmedata.gafa.example.
```

2. Car trop difficile à faire afficher par LaTeX

Un client DNS qui demande le MX de `michu.example` pour lui envoyer du courrier récupérera le MX de `foobar.example.com`, le vrai MX étant masqué.

Plus ennuyeux est le fait que cela dépend : dans certains cas, un autre enregistrement que celui du CNAME est récupéré. On ne peut rien reprocher aux logiciels qui font cela : ils sont conformes aux RFC actuels. Cette variabilité rend difficile de simplement autoriser le CNAME à l'apex.

Troisième possibilité : on peut aussi décider que le résolveur fera l'essentiel du boulot. On crée un nouveau type d'enregistrement, mettons SEEALSO, qui peut coexister avec les types existants :

```
michu.example.   IN SEEALSO  michu-sa.example-cdn.net.
michu.example.   IN MX 10 gimmedata.gafa.example.
```

Le serveur faisant autorité n'aurait rien de particulier à faire, il renvoie juste le SEEALSO au résolveur (en terminologie DNS, on dit « il n'y a pas de traitement additionnel »). Le résolveur va alors, recevant le SEEALSO, le suivre. Le principal problème de cette approche est qu'initialement, peu de résolveurs auront le nouveau code, ce qui ne motivera pas les gens qui gèrent les zones à ajouter ce SEEALSO.

Quatrième idée, toujours avec un nouveau type d'enregistrement (et donc avec les problèmes de déploiement que cela pose dans un Internet non centralisé), généralement appelé ANAME ou ALIAS. L'idée est qu'on mettra dans la zone :

```
michu.example.   IN ANAME  michu-sa.example-cdn.net.
michu.example.   IN MX 10 gimmedata.gafa.example.
```

Le ANAME, contrairement au CNAME, a le droit de coexister avec d'autres enregistrements, ici un MX. L'idée est que le serveur faisant autorité, chargeant la zone, va résoudre la cible (la partie droite du ANAME) et répondre avec l'adresse IP de la cible lorsqu'on l'interrogera. Le résolveur ne voit donc pas le CDN, si l'adresse au CDN est `2001:db8::ff`, le résolveur recevra :

```
michu.example.   3600 IN AAAA 2001:db8::ff
```

Le serveur faisant autorité, ayant chargé la zone, sera responsable de changer cette valeur lorsque le TTL expirera.

On voit que cette solution nécessite que le serveur faisant autorité soit également résolveur. C'est considéré comme une mauvaise pratique, car cela complique sérieusement le déboguage : on ne sait plus d'où viennent les données, et celles du résolveur peuvent potentiellement masquer celles qui font autorité. D'ailleurs, les meilleurs logiciels serveur faisant autorité, comme NSD, n'ont pas du tout de code pour faire de la résolution (cette simplicité améliore grandement la sécurité).

D'autre part, introduire un nouveau type de données DNS n'est jamais évident, cela nécessite de modifier les serveurs faisant autorité (les résolveurs, eux, n'ont pas besoin d'être modifiés, pour ce ANAME), mais également les logiciels d'avitaillement (interfaces web chez l'hébergeur DNS permettant de gérer sa zone).

`http://www.bortzmeyer.org/cname-apex.html`

Cette idée de ANAME pose également des problèmes à DNSSEC. Comme le serveur faisant autorité va devoir modifier le contenu de la zone fréquemment (les TTL des CDN sont souvent courts), il faudra qu'il signe les enregistrements créés, ce qui obligera à avoir la clé privée disponible (on préfère parfois la garder hors-ligne, pour des raisons de sécurité, ce qui est opérationnellement faisable si on ne modifie pas la zone trop souvent). Et cela nécessitera un serveur qui peut signer dynamiquement, ou bien des bricolages particuliers.

On peut avoir des tas de variantes sur cette idée. C'est d'ailleurs une des raisons pour laquelle le débat est compliqué. Chaque idée a des sous-idées. Par exemple, puisque dans ce cas, le serveur faisant autorité est également résolveur, on pourrait renvoyer le ANAME au client DNS, avec l'adresse IP du CDN, pour gagner du temps. La réponse serait :

```
;; ANSWER SECTION:
michu.example.   3600 IN ANAME michu-sa.example-cdn.net.

;; ADDITIONAL SECTION:
michu-sa.example-cdn.net. 600 IN AAAA 2001:db8::ff
```

Plus de problèmes avec DNSSEC cette fois, puisqu'on n'importerait plus de données extérieures dans la zone `michu.example`. Le principal problème de cette variante est que l'optimisation serait probablement inutile : un résolveur DNS raisonnablement paranoïaque, craignant une attaque par empoisonnement, ignorerait la section additionnelle, puisque le serveur interrogé ne fait autorité que pour `michu.example`, pas pour `example-cdn.net`. On serait donc ramené à la troisième idée (le résolveur fait tout).

Cinquième idée, résoudre le problème côté client, comme dans la troisième, mais en modifiant les applications et non plus les résolveurs. Après tout, le principal scénario d'usage est pour HTTP. Ce sont les gens du Web qui se plaignent de ne pas pouvoir mettre des CNAME à l'apex. Les gens de SMTP ou de XMPP ne se plaignent pas, car ils ont un système d'indirection. On indique dans le DNS le nom du serveur pour un domaine donné (enregistrement MX pour SMTP et les plus généraux enregistrements SRV pour XMPP). HTTP est hélas le seul protocole normalisé qui a « oublié » de faire une indirection entre domaine et serveur. C'est une de ses plus graves fautes de conception. (Un protocole comme SSH est un cas à part puisque son but est de se connecter à une machine spécifique.) Donc, la meilleure solution, du point de vue de l'architecture de l'Internet, est de modifier HTTP pour que les clients HTTP utilisent SRV. La zone serait alors :

```
michu.example.   IN SRV 0 1 80  michu-sa.example-cdn.net.
; Autres types
michu.example.   IN MX 10 gimmedata.gafa.example.
```

SRV, normalisé dans le RFC 2782, a plein d'autres possibilités très pratiques, comme d'indiquer plusieurs serveurs, avec des poids différents (fournissant ainsi un système non centralisé de répartition de charge) et des priorités différentes (les serveurs de faible priorité n'étant sollicité qu'en cas de panne des autres).

Tout cela pose évidemment un réel problème de déploiement puisqu'il faudrait modifier tous les clients HTTP (rappelez-vous qu'il n'y a pas que les navigateurs !) Tant qu'il n'y aura pas de déploiement significatif, les titulaires de noms de domaine devront avoir le SRV **et** les enregistrements classiques. D'autre part, HTTP ayant évolué sans les enregistrements SRV, il y a quelques points d'accrochage. Par exemple, SRV permet d'indiquer le port et cela peut rentrer en conflit avec le concept d'origine du Web,

---

`http://www.bortzmeyer.org/cname-apex.html`

qui est essentiel pour sa sécurité (`http://example.com/` et `http://example.com:3000/` sont des origines différentes, cf. RFC 6454). Il y a aussi quelques pièges liés au SRV (voir mes notes à ce sujet <https://github.com/bortzmeyer/ietf-http-srv> et, cette discussion au sujet des SRV dans Mastodon <https://github.com/tootsuite/mastodon/issues/1931>.)

Et je n'ai pas cité toutes les idées, comme la possibilité d'utiliser Alt-Svc (RFC 7838).

Conclusion ? Le problème est bien sûr réel et se pose à beaucoup d'acteurs de l'Internet. Mais il n'y a pas de solution idéale. Il faudra, soit continuer comme actuellement si l'IETF n'arrive pas à un accord, soit adopter une solution qui, de toute façon, créera ses propres problèmes. Personnellement, je pense que la solution la plus propre serait de modifier HTTP, pour utiliser une indirection, comme tous les autres protocoles. Si cela n'est pas possible, il vaut encore mieux ne rien faire : le dromadaire est assez chargé comme cela <https://blog.powerdns.com/2018/03/22/the-dns-camel-or-the-rise-in-dns->.