

Pourquoi avoir développé IDNA au lieu d'utiliser directement l'Unicode dans le DNS ?

Stéphane Bortzmeyer

<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 29 Juin 2008

<http://www.bortzmeyer.org/pourquoi-idn-et-pas-un-dns-unicode.html>

On voit souvent des questions du genre « Pourquoi la norme actuelle sur les noms de domaine internationalisés, IDNA (RFC 5890¹), utilise t-elle un encodage en ASCII (le Punycode du RFC 3492), plutôt que d'utiliser directement Unicode dans le DNS ? ». Cet article tente de répondre à cette question.

Actuellement, la norme pour les noms de domaine internationaux, IDNA (pour "*Internationalized Domain Names in Applications*"), prévoit que le nom de domaine en Unicode, par exemple [Caractère Unicode non montré²][Caractère Unicode non montré][Caractère Unicode non montré][Caractère Unicode non montré][Caractère Unicode non montré].gr, va être traduit dans un sous-ensemble d'ASCII, ici en xn--sxaajkh11633b.gr (le TLD grec n'est pas encore traduit). C'est ce nom en ASCII qui sera mis dans les fichiers de zone et qui circulera entre serveurs DNS. Pourquoi avoir introduit ce système ? Pourquoi ne pas avoir permis directement l'Unicode dans le DNS ?

On lit parfois que c'est parce que le DNS ne permettrait qu'ASCII. C'est totalement faux. Celui qui écrit ça montre qu'il ne connaît pas le sujet. Le DNS a en effet toujours permis n'importe quel contenu, bien au delà d'ASCII. Le RFC 2181, dans sa section 11, a bien enfoncé le clou en rappelant que tout caractère est légal dans le DNS.

Le premier problème à l'utilisation d'Unicode ne vient pas du DNS mais des applications qui gèrent des noms de machines et pour lesquelles la règle (RFC 1123, section 2.1) est en effet bien plus restrictive. Cette règle est connue sous le nom de LDH (pour "*Letters-Digits-Hyphen*", cette règle limite en effet les noms de machine aux lettres, chiffres et au tiret).

1. Pour voir le RFC de numéro NNN, <http://www.ietf.org/rfc/rfcNNN.txt>, par exemple <http://www.ietf.org/rfc/rfc5890.txt>

2. Car trop difficile à faire afficher par L^AT_EX

Comme on enregistre en général un nom de domaine pour y mettre des noms de machine, la plupart des registres ont intégré cette règle et ne permettent que des noms de domaines LDH. Par exemple, les règles d'enregistrement dans ".fr" <<http://www.afnic.fr/obtenir/chartes>> disent « Sont admis à titre de noms de domaine les termes alphanumériques constitués de lettres de l'alphabet français [sic] A à Z et de chiffres de 0 à 9 et du tiret - ». Mais c'est une politique des registres, pas une limitation du DNS.

Changer cette règle nécessiterait de modifier beaucoup de logiciels, tous ceux qui manipulent des noms de machine.

Mais il existe un second problème, plus subtil. Le DNS ne permet pas de recherches floues. Soit le nom correspond parfaitement à un nom dans la base, soit le serveur renvoie le code NXDOMAIN ("*No Such Domain*"). Comme c'est un peu dur pour les utilisateurs, une **canonicalisation** (ou **normalisation**) est prévue, actuellement uniquement l'insensibilité à la casse. Cette canonicalisation rend le DNS un peu plus agréable à l'utilisateur, qui serait certainement dérouté s'il fallait taper exactement le bon caractère, et que CHOCOLAT.fr soit différent de chocolat.fr.

Elle suffit pour ASCII. Pour Unicode, elle ne serait pas réaliste car Unicode offre d'autres possibilités. Ainsi, « thé » peut s'écrire en Unicode avec trois caractères, U+0074, U+0068, et U+00E9 (le dernier identifiant le « e avec accent aigu ») ou bien avec quatre, U+0074, U+0068, U+0065, et U+0301, le dernier étant l'accent aigu combinant. Pour la plupart des lecteurs, ces deux chaînes de caractères sont « identiques ». Pour un logiciel, elles ne le sont pas. Seule une canonicalisation permet une comparaison « raisonnable » de ces deux chaînes. Ainsi, NFC, un des algorithmes standard de canonicalisation d'Unicode, ne laisserait que la première forme, celle à trois caractères.

Il faudrait donc, si on déploie un jour un hypothétique « DNSv2 » (100 % Unicode), mettre NFC (ou un de ses équivalents) dans tous les serveurs de noms... Cet algorithme est compliqué, nécessite des tables (alors que, en ASCII, on peut passer des majuscules ou minuscules sans tables, uniquement en ajoutant ou retirant 32), a des points obscurs (notamment si le caractère Unicode n'est pas affecté)... Peu d'auteurs de serveurs de noms envisagent de l'inclure dans leur code.

Cette seconde raison est la principale. Même si on choisissait une approche « table rase » et qu'on fasse DNSv2 en partant de zéro, en ignorant l'existant, il faudrait résoudre ce problème de canonicalisation (voir le RFC 5198 pour une façon de le traiter).