Résolution de noms Internet dans un nouveau langage de programmation

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 4 mai 2011

https://www.bortzmeyer.org/resolution-noms-langage.html

Une question philosophique aujourd'hui : si on conçoit un nouveau langage de programmation, et qu'on dote celui-ci d'une belle bibliothèque standard, une des fonctions de base sera certainement la résolution de noms de domaine en adresses IP. Comment implémenter cette résolution ? Coder tout soi-même dans son nouveau langage ou bien s'interfacer avec la bibliothèque existante du système d'exploitation, typiquement en appelant la fonction standard `getaddrinfo()` (RFC 3493 [1]) ?

La question n'est pas purement théorique. Si on regarde la bogue #1166 <http://code.google.com/p/go/issues/detail?id=1166> d'une des implémentations du langage Go <https://www.bortzmeyer.org/expose-go.html>, on voit qu'il a fallu plusieurs mois pour décider puis pour changer. Pourquoi était-ce un problème ?

Parce que résoudre un nom en adresse IP peut utiliser plusieurs mécanismes, et que le choix entre ces mécanismes peut dépendre de politiques locales. Ainsi, l'ancienne implémentation dans Go utilisait uniquement le fichier `/etc/hosts` et le DNS. Mais c'est erroné : la résolution de noms peut se faire via bien d'autres moyens, comme LDAP ou NIS. Et de nouveaux moyens peuvent être ajoutés dans le système, forçant à ajouter encore du code à la bibliothèque.

Pire, la sélection entre ces différents moyens et, pour chaque moyen, entre ses différentes options, peut dépendre de réglages locaux que la bibliothèque standard devrait respecter. Ainsi, sur Unix, on peut choisir le mécanisme de résolution via le fichier `/etc/nsswitch.conf`. Par exemple :

```
hosts: dns ldap files
```

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc3493.txt`

cherchera à résoudre les noms d'abord par le DNS puis LDAP puis enfin via un fichier statique (`/etc/hosts`). Même en se limitant au DNS, d'autres réglages sont possibles. Ainsi de la priorité entre IPv4 et IPv6 qui se règle en `/etc/gai.conf` (RFC 6724). Une implémentation de la résolution de noms doit donc lire et analyser ce fichier pour servir des adresses IPv4 ou IPv6 selon ce qu'a décidé l'administrateur système. Le malheureux mainteneur de la bibliothèque standard du nouveau langage a donc beaucoup de travail avant de reproduire le comportement exact de `getaddrinfo()`. Et, s'il ne le fait pas, un programme écrit en Go n'aura pas le même comportement qu'un programme écrit en C (par exemple, l'un utilisera en priorité les adresses IPv4 et l'autre les adresses IPv6).

L'autre solution est donc d'appeler la fonction standard `getaddrinfo()`, typiquement en utilisant les conventions d'interface pour C, que quasiment tous les langages de programmation savent utiliser (appeler les bibliothèques écrites en C est indispensable). Cela présente des inconvénients : C ne garantit pas certaines propriétés que des langages de plus haut niveau considèrent comme indispensables. Par exemple, dans un langage à gestion de la mémoire, comme Go, on ne peut plus garantir que toute la mémoire sera bien récupérée. Dans un langage parallèle (toujours comme Go), on ne peut pas toujours être sûr que le sous-programme C est réentrant.

Malgré ces inconvénients, la majorité des biobliothèques standard qui font de la résolution de noms semble avoir choisi la même voie : appeler le sous-programme C `getaddrinfo()` pour être sûr d'avoir exactement la même sémantique que lui.