Résolution de noms : les programmeurs vont-ils avoir de meilleures interfaces ?

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 12 mars 2013

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

La vie du programmeur d'applications réseau n'est pas facile. Il a souvent besoin de **résoudre** des noms en adresses IP. Il existe pour cela des API standards depuis longtemps. Mais elles ne donnent pas toute l'information nécessaire, loin de là. Aura-t-on un jour de meilleures API ?

Le mécanisme normal en C pour résoudre un nom en adresse est `getaddrinfo()`. (Si vous rencontrez un programmeur qui utilise encore `gethostbyname()`, euthanasiez-le : cette fonction, qui était limitée à IPv4, a été remplacée il y a longtemps.) Dans d'autres langages de programmation, des fonctions similaires existent, parfois avec le même nom (comme `getaddrinfo()` en Python), parfois sous un autre nom (comme `LookupHost` en Go). Notez que, dans ces langages de plus haut niveau, on n'a la plupart du temps **pas besoin** de faire ces résolutions de noms explicitement. On appele des fonctions de connexion qui prennent un nom en paramètre (comme `Dial` en Go).

Ces fonctions ont des limitations communes. D'abord, elles n'indiquent pas pendant combien de temps l'information est valable. Or, les adresses IP changent et une application qui fonctionne longtemps (un navigateur Web, par exemple), si elle voulait garder en mémoire le résultat de `getaddrinfo()` ne saurait pas pour combien de temps cela serait possible (si le nom a été trouvé dans le DNS, `getaddrinfo()` ne permet pas de récupérer le TTL). Résultat, l'application continue parfois à utiliser une adresse qui n'est plus valable, phénomène connu sous le nom de "*pinning*" (cf. RFC 5887 [1], par exemple).

Autre limite de `getaddrinfo()` et de ses équivalents, il n'indique pas si la résolution a été sécurisée ou pas. `getaddrinfo()` peut utiliser plusieurs sources pour faire la résolution : le DNS mais aussi `/etc/hosts` (ou équivalent, pour les systèmes non-Unix), LDAP, etc (la liste des sources est configurable, par exemple sur Ubuntu, c'est dans `/etc/nsswitch.conf`.) Certaines de ces sources peuvent être considérées comme sûres (`/etc/hosts`, LDAP avec TLS, DNS avec DNSSEC), d'autres pas (DNS sans DNSSEC, car il est vulnérable aux attaques par empoisonnement de cache). Mais l'application qui

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc5887.txt`

utilise `getaddrinfo()` ne le sait pas ! L'information (pourtant essentielle) sur l'authenticité du résultat n'est pas renvoyée au programme.

Ce n'est pas tout : une autre limite de ces fonctions de résolution de noms est qu'elles ne permettent que de trouver une adresse IP en échange d'un nom. On a souvent besoin de plus d'informations, que l'on trouve dans le DNS. Par exemple, la plupart des protocoles réseau (mais, hélas, pas HTTP) permettent de stocker dans le DNS le nom du serveur qui assure un service pour un domaine donné (ce qui sépare le nom du domaine de celui du serveur alors qu'en HTTP, le serveur du domaine `example.com` est forcément la machine `example.com`), avec d'autres indications comme le numéro de port à utiliser. Cela se fait via les enregistrements SRV décrits dans le RFC 2782. Ils sont utilisés, par exemple, dans XMPP. Cela veut dire qu'un logiciel de messagerie instantanée ne peut pas se contenter de `getaddrinfo()`, il doit faire des requêtes DNS plus générales. Même cas pour OpenSSH lorsqu'il cherche des clés publiques dans le DNS (RFC 4255) ou pour un client Web qui veut trouver les enregistrements DANE (RFC 6698).

Et c'est là que les ennuis commencent : il n'existe pas d'API standard pour cela. (Voir deux questions sur Stack Overflow <https://www.bortzmeyer.org/stack-overflow.html>, « *"Code to do a direct DNS lookup"* <http://stackoverflow.com/questions/907528/code-to-do-a-direct-dns-lookup « *"Querying full DNS record"* <http://stackoverflow.com/questions/907528/code-to-do-a-direct-dn
On ne manque pas de bibliothèques mais aucune n'est standard, obligeant le programmeur à en choisir une et à demander à ceux qui compilent son programme d'installer ladite bibliothèque. En C, on a :

- ldns <http://nlnetlabs.nl/projects/ldns/> (celle que je préfère, bien documentée <http://www.nlnetlabs.nl/projects/ldns/doc/tutorial1_mx.html> et facile d'usage),
- la plus ancienne et plus traditionnelle libbind <http://www.isc.org/software/libbind> (qui faisait autrefois partie de BIND et s'appelait libresolv, mais peut maintenant s'installer seule), avec les fonctions anciennes et connues comme `res_query()`, qui font que certains la considèrent comme la solution standard, malgré ses difficultés d'utilisation,
- libunbound <http://www.unbound.net/documentation/libunbound.html>, tirée du résolveur du même nom.

Et c'est pareil pour tous les langages de programmation par exemple, en Python, il a l'excellente DNSpython <https://www.bortzmeyer.org/dnspython.html> mais aussi plusieurs autres bibliothèques. Même chose pour Go qui a godns <https://github.com/miekg/dns> mais pas du tout de paquetage DNS standard. Et ces trois langages ne sont que des exemples, ce n'est pas mieux pour les autres langages (signalez-moi les langages de programmation ayant une bibliothèque standard permettant de récupérer des enregistrements SSHFP et de vérifier si cela a été obtenu de manière sécurisée).

Bref, avoir une API standard ne serait pas du luxe. Pour C, un nouvel effort est en cours, et fait l'objet d'une campagne de promotion à la réunion IETF d'Orlando : getdns <http://www.vpnc.org/getdns-api/>. getdns a tous les services souhaités ici, est conçu pour C (si c'est un succès, son adaptation à d'autres langages viendra peut-être par la suite), n'a pas encore de mise en œuvre et a la particularité que les structures de données, parfois complexes, du DNS, sont modélisées dans un langage très proche de JSON.

Programmeurs d'applications réseau utilisant C, c'est le moment de lire la proposition d'API, et de faire des commentaires !
