RFC 6537 : Host Identity Protocol Distributed Hash Table Interface

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 1 mars 2012

Date de publication du RFC : Février 2012

https://www.bortzmeyer.org/6537.html

Dans tout protocole réseau séparant le localisateur et l'identificateur <https://www.bortzmeyer.org/separation-identificateur-localisateur.html>, il y a une grosse question : comment relier les deux ? Comment, lorsqu'on ne connait que l'identificateur d'une machine, trouver son localisateur afin de lui envoyer un paquet ? Le protocole HIP <https://www.bortzmeyer.org/hip-resume.html> a plusieurs solutions possibles et ce RFC de l'IRTF en propose une nouvelle : faire appel à une DHT.

Dans HIP, l'identificateur est la partie publique d'une clé cryptographique, et le localisateur une adresse IP. Notre RFC 6537 [1] propose de stocker la correspondance entre les deux dans une DHT. En outre, un second service est proposé, celui permettant de trouver l'identificateur à partir d'un nom de domaine. En fait, ce RFC ne décrit pas tellement la DHT, mais simplement l'interface entre la machine HIP et un service de résolution (qui pourrait aussi bien être mis en œuvre autrement ; le titre du RFC est donc trompeur).

Pourquoi, d'ailleurs, une DHT ? Comme l'expose la section 1, une des bonnes raisons est que l'espace de nommage des identificateurs est plat : il n'y a pas de hiérarchie permettant des résolutions efficaces comme avec le DNS. Les DHT permettent justement de résoudre des noms situés dans un espace plat aussi efficacement que les protocoles reposant sur des noms hiérarchiques.

Petit détail au passage, le protocole de ce RFC n'utilise pas directement les identificateurs (HI pour "*Host Identifier*") mais leur condensat, le HIT ("*Host Identifier Tag*"). Il y a donc deux services, nom-¿HIT et HIT-¿adresses.

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc6537.txt

Le DNS est très optimisé depuis longtemps (et il existe un autre travail en cours pour résoudre les identificateurs HIP en localisateurs avec le DNS) mais explorer de nouvelles techniques est toujours intéressant. Il s'agit donc d'un travail plutôt expérimental, où plein de questions n'ont pas encore de réponse.

Une expérience concrète a été faite en utilisant la DHT Bamboo <http://bamboo-dht.org/>, écrite en Java, et qui formait la base du service OpenDHT <https://www.bortzmeyer.org/opendht-debut.html> (service qui tournait sur PlanetLab et a été fermé en 2009). Comme toute DHT, ou d'ailleurs comme toute table (les dictionnaires de Python, les "*hashes*" de Perl, les `java.util.Dictionary` de Java, etc), OpenDHT fournissait trois primitives, `put`, `get` et `remove`. La première permet de stocker une valeur, sous un certain index (on dit souvent « clé » mais je l'ai évité pour ne pas risquer une confusion avec les clés cryptographiques), la seconde de récupérer une valeur dont on connait l'index, et la troisième (`rm` dans la documentation) de supprimer une valeur. Dans OpenDHT, ces trois primitives étaient appelées via le protocole XML-RPC.

La section 2 décrit précisement cette interface XML-RPC. Elle avait été choisie parce qu'OpenDHT fournissait un service simple et gratuit accessible aux chercheurs. Les index pouvaient avoir jusqu'à 20 octets, les valeurs jusqu'à 1024. Toutes les entrées étaient temporaires, OpenDHT les supprimant au plus une semaine après leur insertion (un problème classique des DHT et du pair-à-pair en général est sa vulnérabilité aux méchants, ici à des machines insérant des quantités énormes de données pour faire une DoS).

Outre les paramètres cités plus haut (l'index et la valeur pour `put`, l'index pour `get` et `rm`), OpenDHT imposait quelques autres paramètres comme le TTL du précédent paragraphe, comme un identifiant de l'application, ou comme un mot de passe permettant de fournir un peu de sécurité <https://www.bortzmeyer.org/authenticite-dans-dht.html>.

Une fois qu'on a compris l'interface XML-RPC d'OpenDHT et ses paramètres, comment l'utilise-t-on pour HIP ? C'est l'objet de la section 3 et des suivantes. D'abord, on définit en section 3 une structure de données, le **HDRR** ("*HIP DHT Resource Record*"). Son format binaire est le même que celui d'un paquet HIP (RFC 5201, section 5, le HDRR a le numéro 20 dans le registre IANA <https://www.iana.org/assignments/hip-parameters/hip-parameters.xml#hip-parameters-1>), c'est-à-dire essentiellement une suite de paramètres encodés en TLV. Il n'a donc rien à voir avec le "*Resource Record*" DNS du RFC 5205.

Ces paramètres sont surtout des HI ("*Host Identifier*", identificateurs), des HIT (rappelez-vous que c'est un condensat cryptographique du HI) et des localisateurs (les adresses IP, v4 ou v6). On peut aussi y trouver un certificat, au format du RFC 6253 ou une signature HIP. Ces deux derniers paramètres pourraient, dans une DHT spécialisée, être utilisée pour vérifier que la machine qui enregistre un localisateur pour un identificateur donné est bien autorisée à le faire (cd. RFC 7343, c'est une des forces de HIP, que d'avoir des clés cryptographique comme identificateur, cela permet de signer ses paquets et d'éviter ainsi l'usurpation). Toutefois, cela n'est pas mis en œuvre dans Bamboo, DHT généraliste qui ne sert pas qu'à HIP. Le problème de la DHT spécialisée en HIP est de toute façon plus complexe que cela : par exemple, les vérifications cryptographiques nécessaires faciliteraient une éventuelle attaque par déni de service contre la DHT (HIP évite ce problème par un protocole d'établissement de connexion conçu à cet effet, cf. RFC 5201, section 4.1.1).

Puis, la section 4 décrit le protocole de requête. Il fournit deux services :
— Étant donné un nom, trouver le HIT. Le nom est typiquement un FQDN. Notez que le RFC 5205 fournit une autre solution à ce problème, en utilisant le DNS.
— Étant donné un HIT, trouver le ou les localisateurs.

Il n'y a pas de requête directe du nom vers les localisateurs. La première correspondance, nom-¿HIT, est supposée relativement stable (une machine HIP n'a guère de raisons de changer d'identificateur). La seconde correspondance est typiquement variable, voire très variable en cas de mobilité (RFC 5206).

Le reste, ce sont des détails techniques. Par exemple, l'index utilisé pour la requête à la DHT dans le premier cas n'est pas réellement le nom mais un SHA-1 du nom car les index OpenDHT sont limités à vingt octets. Autre point qu'il a fallu régler, les HIT sont typiquement représentés comme des adresses IPv6 ORCHID (RFC 7343), et donc tous les HIT commencent par les mêmes 28 bits (la longueur du préfixe ORCHID). Pour une DHT, qui utilise un hachage des index pour trouver le nœud responsable du stockage des données correspondantes, cela risquerait de causer une répartition très inégale, certains pairs de la DHT étant plus chargés que d'autres. Lors de la deuxième requête (HIT-¿localisateurs), on n'utilise donc comme index que les 100 derniers bits du HIT.

Une fois ces détails de syntaxe réglés, la section 5 décrit la chorégraphie des requêtes. D'abord, pour trouver un identificateur (ou bien son HIT) à partir d'un nom de domaine, faut-il utiliser le DNS, comme le prévoit le RFC 5205 ou bien le service DHT décrit dans ce RFC 6537 ? La réponse dépend des cas. Tout le monde n'a pas forcément le contrôle des serveurs de noms du domaine souhaité et, même si c'est le cas, tous les serveurs de noms ne permettent pas forcément d'ajouter des enregistrements DNS de type HIP dans leur domaine (testez avec l'interface de votre hébergeur DNS pour voir...) Donc, la solution DHT est intéressante au moins dans ces cas.

On l'a vu, les localisateurs changent, c'est normal. Quand faut-il faire la requête HI-¿localisateur ? Si on la fait très à l'avance, les localisateurs seront peut-être usagés lorsqu'on s'en servira. Si on la fait au moment de la connexion, on peut ralentir celle-ci de manière intolérable. Le RFC suggère donc de faire les requêtes à l'avance, et, si la première tentative de connexion échoue, de refaire une requête de résolution avant de tenter la deuxième connexion.

Les informations enregistrées dans OpenDHT ont une durée de vie (le paramètre `ttl` de la requête, mais la DHT peut aussi supprimer les données avant l'expiration de celui-ci, par exemple s'il faut faire de la place). Une machine HIP doit donc penser à mettre à jour régulièrement ses entrées dans la DHT, pour s'assurer qu'il y a toujours quelque chose. Et, naturellement, si l'information change, la machine doit prévenir la DHT dès que possible.

La DHT étant globale, les adresses IP privées (RFC 1918) ne doivent pas être publiées dans la DHT. Si une machine HIP est coincée derrière un routeur NAT, elle doit, si elle veut être contactée, enregistrer l'adresse du serveur HIP de rendez-vous (RFC 5204).

OpenDHT présentait quelques particularités qui pouvaient surprendre la machine HIP tentant de stocker ses identificateurs et localisateurs. Par exemple, un `put` ne remplace pas la donnée précédente, il s'y ajoute. Et rien ne garantit que l'ordre dans lequel seront renvoyés les HDRR sera celui dans lequel ils ont été insérés. Il faut donc identifier le plus récent (le HDRR a un paramètre `Update ID`, dont je n'avais pas encore parlé, juste pour cela). Le mieux serait de penser à faire le ménage et virer explicitement ses vieilles informations.

Notez d'ailleurs que, le nom n'étant pas authentifié, on peut voir plusieurs machines s'enregistrer sous le même nom. En attendant un système d'authentification du nom (certificat X.509 portant ce nom en sujet ?), il est recommandé de tester et de choisir un autre nom, si celui qu'on voulait est déjà pris (ce qui ne devrait pas arriver, avec les noms de domaine, mais la DHT ne vérifie pas que celui qui enregistre `www.google.com` a le droit).

Autre problème pratique, le choix de la passerelle XML-RPC vers OpenDHT. Il en existait plusieurs, d'une fiabilité douteuse (OpenDHT était un projet de recherche, qui ne visait pas à fournir un service fiable à 100 %) et le client doit donc être prêt à détecter une panne et à passer sur une autre passerelle (OpenDHT fournissait un service "*anycast*", `opendht.nyuld.net`, qui facilitait cette tâche.)

À propos de qualité de service, le RFC note aussi qu'OpenDHT ne garantissait pas de temps de réponse, et que les opérations `put` et `get` peuvent être lentes, ce à quoi les clients de la DHT doivent se préparer.

Pour approfondir davantage les questions de sécurité (point faible classique de toutes les DHT), la section 7 fait un examen complet des problèmes que pose cette solution. Concernant le **contenu** de l'information, l'attaque peut être menée par un client de la DHT qui enregistre délibérement des informations fausses, ou bien par un pair participant à la DHT qui modifie les informations avant de les rendre. En l'absence d'un mécanisme tiers d'authentification des données (la DHT de CoDoNS <`http://www.cs.cornell.edu/People/egs/beehive/codons.php`> utilisait DNSSEC pour cela), il n'y a pas grand'chose à faire. On l'a dit, c'est un problème récurrent avec toutes les DHT. Le RFC dit que les paranoïaques n'ont pas d'autres solutions que d'échanger les HIT à la main (ensuite, même si la DHT renvoie de mauvais localisateurs, le protocole HIP permettra de vérifier qu'on ne se connecte pas à une machine pirate). Dans le futur, une solution serait peut-être d'utiliser des certificats numériques.

Je l'ai dit, le protocole HIP garantit au moins, par la signature des paquets échangés lors de la connexion, qu'on se connecte bien à la machine possédant l'identificateur voulu. Pour un attaquant, enregistrer dans la DHT de fausses correspondances HIT-¿localisateur ne permet donc pas de détourner vers une machine pirate. Pour ce stade, la seule possibilité de l'attaquant est d'enregistrer plein de correspondances bidons, pour faire une attaque par déni de service.

La seule solution à long terme à cette attaque serait que la DHT vérifie les données avant de les enregistrer, et donc connaisse HIP. Elle pourrait tester que la signature incluse dans la requête corresponde à la clé publique (l'identificateur). Cette vérification laisserait la DHT vulnérable à d'autres attaques par déni de service, où l'attaquant ferait beaucoup de requêtes et imposerait donc beaucoup de vérifications. Pour parer cela, on peut imaginer une DHT qui impose que ses clients la contactent en HIP, réutilisant ainsi les protections anti-DoS de HIP. Et cela permettrait de s'assurer que le client n'enregistre des localisateurs que pour lui.

À noter enfin que OpenLookup <`http://code.google.com/p/openlookup/`> est un exemple d'un service fournissant la même interface qu'OpenDHT même si, derrière, ce n'est pas une DHT.