RFC 6940 : REsource LOcation And Discovery (RELOAD) Base Protocol

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 17 janvier 2014

Date de publication du RFC : Janvier 2014

https://www.bortzmeyer.org/6940.html

Le groupe de travail IETF p2psip <http://tools.ietf.org/wg/p2psip> travaille à permettre la communication (par exemple téléphonique) entre utilisateurs de SIP **sans** passer par un fournisseur SIP, mais en se débrouillant en pair-à-pair. Pour cela, le groupe a choisi une approche modulaire et a d'abord développé un protocole de communication pair-à-pair, qui va servir de socle aux futures utilisations. Ce protocole se nomme **RELOAD** (pour "*REsource LOcation And Discovery*"), et fait l'objet de ce très long RFC, le premier du groupe p2psip.

L'enjeu est d'importance : le système fermé Skype a obtenu beaucoup d'utilisateurs en s'appuyant partiellement sur des techniques pair-à-pair. Le SIP traditionnel, où deux correspondants ne communiquent que via leurs fournisseurs SIP respectifs <https://www.bortzmeyer.org/sip-free-direct.html> devait être amélioré en exploitant ces capacités pair-à-pair. Échanger des paquets RTP directement est facile. Se trouver et initier la communication l'est moins. Pour le faire en pair-à-pair, il n'existait pas de protocole ouvert et normalisé. On notera d'ailleurs que RELOAD est le premier protocole pair-à-pair (au sens moderne parce que, objectivement, IP a toujours été pair-à-pair) normalisé par l'IETF.

RELOAD est suffisemment général pour pouvoir être utilisé dans le futur par d'autres systèmes que SIP. Qu'est-ce qu'il fournit comme service que les autres protocoles existants ne fournissaient pas ?

— Une gestion de la sécurité, avec un serveur central d'inscription qui laisse entrer (ou pas) les pairs. Si vous pensez qu'un tel serveur n'est pas pair-à-pair, vous avez raison. Si vous pensez qu'on peut s'en passer, vous avez tort <https://www.bortzmeyer.org/pairapair-securite.html>.
— La possibilité d'être utilisé par des applications différentes (par exemple SIP et XMPP),
— Des fonctions de passage à travers les NAT (un point pénible pour SIP et ses équivalents),

— La possibilité d'utiliser plusieurs algorithmes pour la gestion de l'"*overlay*" (le réseau virtuel entre les pairs). Le pair-à-pair est en effet un domaine encore relativement récent et il serait imprudent de croire qu'on connait **le** bon algorithme d'"*overlay*" et qu'on peut ignorer tous les autres. RELOAD permet des réseaux structurés (comme les DHT) et des non structurés. À des fins d'interopérabilité, RELOAD impose la possibilité d'utiliser la classique DHT Chord (ainsi, deux mises en œuvre de RELOAD sont sûres d'avoir au moins un algorithme en commun) mais permet aux "*overlays*" d'en utiliser d'autres. La section 3.5 détaille cette particularité de RELOAD. Les systèmes possibles sont stockés dans un registre IANA <https://www.iana.org/assignments/reload/reload.xml#overlay-alg>.
— Une distinction entre pairs et clients, qui permet de profiter de RELOAD sans y participer comme pair. Le même protocole est utilisé (section 3.2 pour les détails). L'annexe B du RFC explique plus en détail pourquoi on autorise les « simples » clients, notamment parce que certaines machines n'ont pas assez de ressources pour assurer les tâches (routage et stockage) d'un pair (machines sur batterie, par exemple).

Si vous voulez vous cultiver sur Chord, l'article original est « "*Chord : A Scalable Peer-to-peer Lookup Protocol for Internet Applications*" <http://pdos.csail.mit.edu/papers/ton:chord/paper-ton.pdf> » (IEEE/ACM Transactions on Networking Volume 11, Issue 1, 17-32, Feb 2003, 2001) et Wikipédia en anglais a un bon article.

Une instance RELOAD est donc un "*overlay*" (un réseau virtuel utilisant un algorithme donné) où chaque nœud (pair ou simple client) a un identificateur, le "*Node-ID*". Cet "*overlay*" forme un graphe où tout nœud n'a pas forcément directement accès à tout autre nœud, et les pairs doivent donc se préparer à router les messages (les simples clients en sont dispensés).

L'instance RELOAD offre deux services, la transmission de messages et le stockage de données (données de petite taille, a priori, celles nécessaires à la communication). Chaque donnée a un identificateur, le "*Resource-ID*", qui utilise le même espace de nommage que les "*Node-ID*". Il y a ensuite une relation entre les "*Node-ID*" et les "*Resource-ID*" par exemple, qu'un nœud est responsable de toutes les données dont le "*Resource-ID*" est compris entre son "*Node-ID*" et le "*Node-ID*" précédent (si vous connaissez les DHT, ce genre de concept vous est familier. Sinon, cela peut être le moment de s'arrêter pour lire des choses sur les DHT.) La relation exacte dépend de l'algorithme d'"*overlay*" utilisée. Celle donnée en exemple est valable pour Chord.

Pour accomplir ses tâches, RELOAD a plusieurs composants. De haut en bas (si cela ressemble au modèle en couches, c'est exprès, sauf que ces couches sont entassées sur l'"*overlay*", pas sur un réseau physique) :

— L'application, par exemple SIP (la liste des applications est dans un registre IANA <https://www.iana.org/assignments/reload/reload.xml#app-id-rules>).
— Un composant de transport des messages, qui assure des fonctions comme la fiabilité de bout en bout. Ce composant permet donc d'envoyer un message de n'importe quel pair de l'"*overlay*" à n'importe quel autre.
— Un composant de stockage des données.
— Un greffon de gestion de la topologie. RELOAD utilise le terme de "*topology plugin*", pour mettre en avant le fait qu'il est modifiable : on peut remplacer le greffon par défaut, qui utilise Chord, par un autre.
— Un composant de gestion des liens entre pairs, qui s'occupera notamment du difficile problème de la traversée des NAT, grâce à ICE (RFC 8445 [1] et section 6.5.1.3).

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc8445.txt

— Et plusieurs composants de transport sur les liens, entre deux pairs. Un exemple est un composant utilisant TLS (RFC 5246) sur TCP. Un autre composant au même niveau pourrait être DTLS (RFC 6347) sur UDP. Contrairement au composant de transport des messages, le premier cité, qui avait une responsabilité de bout en bout, ces composants de transport sur un lien ne sont responsables que d'envoyer des octets d'un pair à un pair **voisin** (voisin dans l'"*overlay*", pas du point de vue IP...)

La sécurité est traditionnellement **le** point faible des DHT (et du pair-à-pair en général). (Voir l'excellent exposé d'Eric Rescorla <http://www.ietf.org/proceedings/65/slides/plenaryt-2.pdf>.) Le principe de base de la sécurité de RELOAD est que chaque pair aura un certificat délivré par une instance centrale (qui ne sera responsable que de l'inscription des pairs : elle ne participera ni au routage, ni à l'envoi des messages, ni au stockage,). Chaque message, et chaque objet stocké, sera signé avec ce certificat et toutes les communications de bas niveau chiffrées (par exemple avec TLS). Une machine qui veut rejoindre l'"*overlay*" doit se connecter en HTTPS au serveur central, qui lui attribuera un "*Node-ID*" (dans le champ `subjectAltName` du certificat RFC 5280). Le serveur central demandera probablement une authentification (par exemple nom et mot de passe). Ce serveur et les vérifications qu'il fait sont le cœur de la sécurité de RELOAD. La machine joindra ensuite le réseau virtuel (les détails sur la façon dont elle le fait dépendent de l'algorithme utilisé pour le maintien de l'"*overlay*").

Ne vous inquiétez pas forcément de ces obligations de sécurité : elles ne sont valables que pour un "*overlay*" installé sur le grand Internet public. Si vous avez un environnement restreint et sécurisé (disons un groupe de machines formant un réseau ad hoc local), vous avez le droit de vous en dispenser et de vous contenter de certificats auto-signés par les pairs, où le "*Node-ID*" est simplement un condensat de la clé publique. Une telle façon de fonctionner laisse le réseau très vulnérable aux attaques Sybil mais peut convenir si vous savez que vous êtes « entre vous ». Autre possibilité de simplification de la sécurité : RELOAD a un mode où les pairs s'authentifient, non pas par un certificat, mais par un secret partagé (par exemple RFC 4279).

Voici donc les grands principes de RELOAD. Je ne vais pas ensuite détailler les 186 pages du RFC, juste quelques points que je trouve intéressants. Le mécanisme de routage est présenté en section 3.3. Par défaut, RELOAD utilise du **récursif symétrique** (section 6.2). La requête enregistre les nœuds par lesquels elle est passée (dans la "*Via List*") et la réponse suivra le même chemin en sens inverse. Si la topologie a changé entre-temps, la réponse sera perdue et ce sera aux couches supérieures de réémettre. (Il existe une alternative à ce stockage du chemin dans le message lui-même : si les pairs intermédiaires peuvent stocker des données, ils peuvent garder la mémoire des routes suivies, au lieu d'utiliser la "*Via List*".)

L'inscription initiale dans l'"*overlay*" est un problème intéressant, car on voudrait que cela soit simple pour l'utilisateur, par exemple qu'il n'ait pas à rentrer 50 paramètres dans son client SIP. L'idée est que le nouveau pair n'a à connaître que le nom de l'"*overlay*" (un FQDN) et peut-être des informations d'authentification. Une requête SRV dans le DNS à partir du nom du réseau (précédé de `_reload-config._tcp`) donne le nom du serveur de configuration, qu'on contacte en HTTPS en lui demandant la ressource `.well-known/reload-config` (RFC 8615 pour `.well-known`). Ce serveur envoie alors tous les détails de configuration de l'"*overlay*", en XML, un exemple figure en section 11 et à la fin de cet article. (Une machine peut aussi avoir cette configuration dans un fichier local.) C'est alors que le futur pair connait le nom du serveur d'inscription et peut le contacter pour apprendre son "*Node-ID*".

Le stockage pair-à-pair dans RELOAD (section 4.1) permet de stocker différents types ("*kind*") d'objets. Un type peut être défini comme stockant un scalaire, un tableau ou un dictionnaire (section 7.2 pour les détails). Tous les objets stockés sont signés par le déposant, qui pourra ainsi vérifier que ses pairs ne se sont pas moqués de lui. (Voir mon article « Assurer l'authenticité des données stockée dans une DHT <https://www.bortzmeyer.org/authenticite-dans-dht.html>».) À noter qu'un

pair ne peut pas stocker des objets sous des noms arbitraires. Typiquement (cela dépend de l'application), il ne peut stocker que sous son propre nom (`alice@example.com` pour du SIP), ce qui fait qu'il ne pourra pas noyer tous les nœuds de la DHT sous des objets à stocker.

Une des choses les plus nécessaires pour communiquer en pair-à-pair est la **découverte**. Trouver quelqu'un ou quelque chose dans l'immensitude de l'Internet est difficile (alors que c'est trivial en centralisé, d'où le succès de services comme Facebook). Il existe actuellement plusieurs *"Internet-Drafts"* décrivant de futurs services de découverte qui tourneront au dessus de RELOAD.

Le format des messages RELOAD est décrit en section 6.3 : format binaire, avec les premiers en-têtes, qui sont obligatoires, à une position fixe (pour une analyse plus rapide) et les suivants, qui sont optionnels, encodés en TLV (pour la souplesse et l'extensibilité). Chaque message est auto-suffisant, un peu comme un datagramme. Sa première partie, le *"Forwarding Header"*, est la seule qui ait besoin d'être examinée par les routeurs intermédiaires. Les deux autres, le *"Message Content"* et le *"Security Block"* (les signatures), ne concernent que la machine de destination.

Notre RFC utilise une syntaxe analogue à celle du C pour décrire le format des messages (un peu comme le fait le RFC 5246). Principal point à noter pour lire cette syntaxe (la section 6.3.1 fournit tous les détails) : les termes entre chevrons sont des valeurs de taille variable. Par exemple `data<0..15>` désigne de zéro à quinze octets de données.

Le *"Forwarding Header"* contient les *"Via Lists"* vues plus haut et les destinations (la plupart du temps, il n'y en a qu'une seule mais RELOAD permet d'indiquer une liste de destinations à parcourir successivement : pour comparer avec le datagramme IP, disons que la liste des destinations est une fusion de l'adresse IP de destination et de l'option *"Source route"*). Cela donne :

```
struct {
         ...
         uint32             overlay; /* Un condensat du nom */
         ...
         uint8              ttl;
         ...
         uint32             length;
         ...
         uint16             via_list_length; /* Pour le routage
 symétrique */
         uint16             destination_list_length;
         ...
         Destination        via_list[via_list_length];
         Destination        destination_list
                              [destination_list_length];
         ...
       } ForwardingHeader;
```

Pour cet en-tête, qui doit être lu très vite par les routeurs intermédiaires, RELOAD utilise l'encodage {longueur, tableau}. Les autres termes de longueur variable utiliseront un encodage moins rapide à analyser mais plus propre. Le type `Destination` est, en résumant, un *"Node ID"* ou *"Resource ID"* (rappelez-vous que ces ID ont la même forme - dépendante du type d'*"overlay"* - et sont tirés du même espace).

La deuxième partie du message, *"Message Content"*, est largement opaque :

```
struct {
     uint16                message_code;
     opaque                message_body<0..2^32-1>;
     MessageExtension      extensions<0..2^32-1>;
   } MessageContents;
```

C'est donc surtout une suite d'octets, 2{}32 au maximum. Le code indique la sémantique du message (demande de stockage de données, « ping », annonce d'un nouveau pair, etc) et la liste des codes est dans un registre IANA <https://www.iana.org/assignments/reload/reload.xml#message-codes>.

La troisième et dernière partie du message, le "*Security Block*", est une liste de certificats X.509 et une signature :

```
struct {
      GenericCertificate certificates<0..2^16-1>;
      Signature          signature;
   } SecurityBlock;
```

Tous les messages doivent être signés et les signatures vérifiées à la réception.

Les responsabilités du greffon de gestion de la topologie figurent dans la section 6.4. Elles sont décrites sous forme de messages abstraits, à incarner pour chaque algorithme d'"*overlay*". Un exemple, le message `Join` lorsqu'un nouveau pair arrive :

```
struct {
            NodeId                joining_peer_id;
            opaque                overlay_specific_data<0..2^16-1>;
         } JoinReq;
```

Les protocoles de communication entre deux nœuds devront fournir des mécanimes d'authentification du "*Node ID*", d'intégrité et de confidentialité, par exemple en s'appuyant sur TLS (section 6.6.1). Le RFC suggère d'explorer la possibilité d'utiliser HIP (RFC 7401 et section 6.6.1.1) qui a toutes les propriétés voulues. Ils doivent aussi respecter les principes du RFC 8085, i.e. ne pas noyer les destinataires sous les retransmissions.

Le stockage des données fait l'objet de la section 7. On stocke des objets qui ont cette forme :

```
 struct {
          uint32                length;
          uint64                storage_time;
          uint32                lifetime;
          StoredDataValue       value;
          Signature             signature;
       } StoredData;
```

On voit qu'ils sont signés, et ont une durée de vie maximale.

On l'a dit, RELOAD utilise par défaut la DHT Chord et toutes les mises en œuvre de RELOAD doivent connaître Chord, pour assurer l'interopérabilité. Le mécanisme utilisé par RELOAD est légèrement différent du Chord original et la section 10 détaille ces différences. Les deux principales :

— Chord ne permettait qu'un seul prédécesseur et un seul successeur par nœud, alors que plusieurs sont possibles avec Chord-RELOAD,
— Le routage dans Chord était itératif (le nœud de départ devait contacter directement celui d'arrivée, les autres pairs ne servaient qu'à indiquer où le trouver), il est devenu récursif (les autres pairs transmettent le message, difficile de faire autrement avec les NAT). L'annexe A du RFC discute en détail les raisons de ce choix.

Les grands principes de Chord sont gardés : nœuds organisés en un anneau (le dernier nœud dans l'ordre des "*Node ID*" est donc le précédesseur du premier), utilisation de SHA-1 (tronqué aux 128 premiers bits dans Chord-RELOAD) pour trouver le "*Resource ID*" à partir des données, table de routage composée, non seulement des voisins immédiats, mais aussi de pointeurs (les "*fingers*") vers un certain nombre de non-voisins (pour des raisons de performance, car cela évite de parcourir la moitié de l'anneau, mais aussi de résilience, pour survivre à la perte des voisins), etc

Un exemple de fichier de configuration d'un réseau virtual ("*overlay*") figure en section 11 :

```
<overlay>
      <configuration instance-name="overlay.example.org" sequence="22"
          expiration="2002-10-10T07:00:00Z" ext:ext-example="stuff" >
          <!-- Le greffon de topologie par défaut : -->
          <topology-plugin>CHORD-RELOAD</topology-plugin>
          <!-- Le certificat qui va signer tout : -->
          <root-cert>
              MIIDJDCCAo2gAwIBAgIBADANBgkqhkiG9w0BAQUFADBwMQswCQYDVQQGEwJVUzET...
          </root-cert>
          <!-- Les serveurs à contacter pour avoir Node-ID et
  certificat : -->
          <enrollment-server>https://example.org</enrollment-server>
          <enrollment-server>https://example.net</enrollment-server>
          <!-- Les pairs à contacter pour rejoindre l'overlay : -->
          <bootstrap-node address="192.0.0.1" port="6084" />
          <bootstrap-node address="192.0.2.2" port="6084" />
          <bootstrap-node address="2001:db8::1" port="6084" />
          <!-- Les paramètres spécifiques à la DHT : -->
          <chord:chord-update-interval>
              400</chord:chord-update-interval>
          <chord:chord-ping-interval>30</chord:chord-ping-interval>
          <chord:chord-reactive>true</chord:chord-reactive>
         <!-- Et bien d'autres choses encor... -->
    </overlay>
```

6084 est le port par défaut pour RELOAD (section 14.2). La grammaire pour ce fichier de configuration est normalisée (en Relax NG) en section 11.1.1.

Pour contacter les serveurs d'inscription ("*enrollment server*"), le protocole à utiliser est en section 11.3. L'IETF a choisi de ne pas réutiliser les protocoles existants (RFC 5272 et RFC 5273) car ils sont trop complexes. À la place, le futur pair fait juste une requête HTTPS `POST` dont le contenu comprend du PKCS#10 (contenant la demande de certificat, cf. RFC 2311). Cette requête doit aussi contenir les informations d'authentification du pair (par exemple un nom et un mot de passe). En échange, le serveur enverra un certificat dont le sujet (le nom) sera un "*Node ID*", choisi au hasard (RFC 4086 ; c'est important car certaines attaquent contre les DHT nécessitent que l'attaquant choisisse son "*Node ID*").

En outre, le champ `subjectAltName` doit contenir le nom que l'utilisateur sera autorisé à présenter à l'application (pour SIP, ce sera un nom de type `machin@truc.example`). Certaines autorisations (par exemple d'accès aux données stockées) dépendront de ce nom.

La faiblesse traditionnelle des DHT (ou du pair-à-pair en général) est la sécurité (RFC 5765). Sans chef pour contrôler les échanges, que faire si un méchant s'insère dans le réseau pair-à-pair pour l'espionner, modifier les données, ou bien empêcher le réseau de fonctionner ? La section 13 revient en détail sur les questions de sécurité. Le fond du problème est qu'on dépend des pairs (pour envoyer des messages à d'autres pairs ou bien pour stocker des données) et qu'on n'a pas de relation bien définie avec ces pairs. Sur un réseau public comme l'Internet, les pairs peuvent être n'importe qui, y compris des méchants. Si la majorité des pairs sont dans ce cas, on est fichu, aucun algorithme astucieux ne peut marcher dans ces conditions. Il faut donc agir sur l'**inscription** ("*enrollment*"). C'est ce que fait RELOAD (section 13.3) en imposant le passage par un serveur d'inscription pour obtenir un certificat, puis en imposant de prouver la possession d'un tel certificat (en signant les messages avec la clé).

RELOAD permet des certificats auto-signés mais, dans ce cas, l'"*overlay*" peut être vite noyé sous les pairs malveillants (ce qu'on nomme une attaque Sybil). La seule sécurité qui reste est le "*Node ID*" (qui est dérivé du certificat et est donc cryptographiquement vérifiable). Bref, une telle sécurité ne convient que dans des environnements fermés, où on est sûr de toutes les machines présentes. Un exemple typique est un réseau ad hoc entre les machines des participants à une réunion (et encore, en Wifi, attention à l'attaquant de l'autre côté du mur...). Dans ce cas, on peut même se passer complètement des certificats en choisissant la solution du secret partagé : on le distribue à tous les participants à la réunion et ça roule (section 13.4).