RFC 7541 : HPACK - Header Compression for HTTP/2

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 16 mai 2015

Date de publication du RFC : Mai 2015

https://www.bortzmeyer.org/7541.html

Une des nouveautés de la version 2 de HTTP, désormais normalisée dans le RFC 7540[1], est de représenter les en-têtes des requêtes et des réponses, non plus sous une forme texte, mais sous une forme binaire, avec compression. Ce RFC normalise le mécanisme de compression, nommé **HPACK**.

Fini, donc, de taper des en-têtes HTTP à la main dans une session telnet, ou bien d'écrire un serveur HTTP qui envoie simplement ses réponses avec un `fprintf`. Désormais, tout se fait en binaire et sera comprimé, pour gagner quelques octets. L'encodage peu efficace des en-têtes de HTTP version 1 faisait perdre de l'espace, donc du temps, d'autant plus qu'ils sont répétés à chaque requête. Il fallait donc comprimer. Mais un algorithme de compression général, comme le DEFLATE (RFC 1951) qu'utilisait le premier prototype de HTTP 2, le protocole SPDY, avait des conséquences néfastes en terme de sécurité, lorsqu'il était combiné au chiffrement. D'où l'idée d'un algorithme spécifique à HTTP et n'ayant pas ces défauts, l'actuel HPACK. HPACK se veut simple (le RFC fait quand même 58 pages, celui de DEFLATE n'en faisait que 17), et inflexible (pas d'options à négocier).

HPACK est prévu (section 1.1 du RFC) pour comprimer des en-têtes HTTP, c'est-à-dire une suite **ordonnée** de doublets {nom, valeur}. La duplication des noms est autorisée.

La spécification de HPACK n'impose pas un algorithme spécifique pour le compresseur, elle décrit juste ce à quoi doivent ressembler les données comprimées, pour que le décompresseur arrive à les reconstituer.

Les données comprimées vont contenir des index, qui référencent une table, et des valeurs littérales. HPACK utilise deux tables, qui permettent d'associer un index à un en-tête. La première table est

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc7540.txt`

statique : définie dans ce RFC, dans l'annexe A, elle comporte les en-têtes les plus courants. Ainsi, `Accept-encoding: gzip,deflate` (RFC 7231, section 5.3.4), présent chez presque tous les navigateurs Web, sera représenté par l'index 16. Cette table statique a surtout des entrées qui sont des pseudo-en-têtes, c'est-à-dire des valeurs dans les requêtes ou les réponses qui n'étaient pas définies comme des en-têtes en HTTP 1 mais qui sont assimilés à des en-têtes en HTTP 2 (on les reconnait à leur nom qui commence par deux-points). Ainsi, le classique `GET` (`:method GET` pour HTTP 2) sera l'index 2 et `/index.html` (`:path /index.html` en HTTP 2) sera l'index 5.

La seconde table est dynamique : vide au début, elle se remplit au fur et à mesure de la compression ou de la décompression. Ses index commencent là où la table statique se termine (à l'index 61).

Un en-tête pourra donc être représenté par un index (cas du `Accept-encoding: gzip,deflate` plus haut) ou bien par une combinaison d'un nom d'en-tête et d'une valeur. Le nom d'en-tête pourra être un index (`Content-type:`, index 31, est un exemple dans la table statique) ou bien indiqué littéralement.

Quelles données peut-on mettre dans un flot de données HPACK ? La section 5 expose comment sont représentés entiers et chaînes de caractères. Rappelez-vous que le but est de gagner de la place et les entiers peuvent donc avoir une représentation un peu baroque, un entier pouvant commencer au beau milieu d'un octet (des exemples figurent en annexe C.1). Les chaînes de caractères peuvent être représentées telles quelles mais aussi être comprimées avec Huffman (le code Huffman utilisé est dans l'annexe B).

La section 6 décrit la représentation complète en binaire. Bon, et la table dynamique, elle est gérée comment ? Sa taille peut changer dynamiquement mais la taille maximale est déterminée par le paramètre `SETTINGS_HEADER_TABLE_SIZE`. Certains éléments dans les données ne changent pas la table mais d'autres ajoutent une entrée à la table dynamique. Selon les bits placés au début d'un élément (section 6), le décodeur sait si la donnée est indexée ou littérale et, si elle est littérale, s'il faut l'ajouter à la table dynamique. De nombreux exemples en annexe C permettent de se faire une meilleure idée de l'encodage.

Enfin, la section 7 se penche sur quelques problèmes de sécurité. Contrairement à DEFLATE (RFC 1951) qui permettait de deviner un préfixe du terme comprimé (ce qui, combiné avec le fait que TLS ne masque pas la longueur des données, seulement leur contenu, menait à l'attaque CRIME), HPACK oblige l'attaquant actif à tester un terme entier. Une attaque par force brute pour essayer de deviner le contenu en se basant sur la longueur observée reste donc possible, bien que nettement plus coûteuse qu'avec DEFLATE. Pour rendre plus difficile cette attaque, on peut avoir intérêt à ne pas comprimer les en-têtes les plus sensibles (comme les "*cookies*"). D'autres méthodes dépendent de l'application : par exemple, un navigateur Web ne devrait pas permettre l'utilisation de la même connexion HTTP 2 (donc du même contexte de compression/décompression) par du code issu de deux origines (RFC 6454) différentes.

Attention enfin à la consommation mémoire : compression et décompression peuvent en nécessiter beaucoup. La limite à la taille de la table dynamique est là pour empêcher les excès.

Toutes les mises en œuvre de HTTP 2 ont également HPACK, et en voici une liste `<https://github.com/http2/http2-spec/wiki/Implementations>`.