RFC 9292 : Binary Representation of HTTP Messages

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 5 mai 2025

Date de publication du RFC : Août 2022

https://www.bortzmeyer.org/9292.html

Un message HTTP est traditionnellement représenté comme du texte mais ce RFC spécifie une alternative, une représentation binaire d'un message.

À quoi ça sert ? L'un des buts est de faciliter la transmission de messages HTTP (requêtes ou réponses, cf. RFC 9110 [1]) par d'autres protocoles que HTTP. C'est ainsi que l'"*oblivious HTTP*" du RFC 9458 utilise ce format binaire. D'autre part, un format binaire a des chances d'être plus compact. (Le RFC dit aussi que cela permet l'application du chiffrement intègre, mais ce serait le cas aussi avec du texte, je trouve.) Et l'analyse d'un tel format est plus simple, avec moins de pièges que celle du format texte, qui mènent parfois à des failles de sécurité (voir par exemple la section 11 du RFC 9112). Comme vous le savez, les versions 2 et 3 de HTTP encodent déjà l'en-tête en binaire et leurs spécifications (RFC 9113 et RFC 9114) ont servi de base à ce nouveau RFC. Le nouveau format a le type `message/bhttp` (le format traditionnel étant `message/http`).

On peut encoder le message HTTP en binaire suivant deux modes. Le premier préfixe tous les éléments par leur longueur, ce qui est plus efficace. Le deuxième ne le fait pas, ce qui permet de commencer à encoder sans connaitre la longueur finale.

Notez bien que c'est la sémantique du message qui est encodée, pas sa représentation texte. Concrètement, cela veut dire qu'un message HTTP représenté en texte, traduit en binaire, puis re-traduit en texte ne sera **pas** identique au bit près.

Bon, mais c'est quoi un message HTTP ? La section 6 du RFC 9110 nous l'apprend. S'il peut être encodé de plusieurs manières, la vision abstraite du message est toujours la même : le message est une

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc9110.txt`

requête ou une réponse, s'il est une requête, il y a une méthode (GET, PUT, etc) et la ressource demandée, s'il est une réponse, un code de retour (comme le fameux 404), puis dans les deux cas, un en-tête (avec les champs comme `Host:` ou `Content-Type:`), un corps (qui, comme l'en-tête, peut être vide), un pied (en général absent), et parfois du remplissage. Dans notre encodage en binaire, le fait que le message soit une requête ou une réponse sera indiqué par le "*framing indicator*", qui vaut 0 dans le premier cas et 1 dans le second, pour le mode à longueur connue, et 2 ou 3 pour le mode à longueur pas connue à l'avance (section 3.3).

Le reste de ce que vous devez savoir sur ce format est dans les sections 3.4 (pour les requêtes) et 3.5 (pour les messages). Mais on va voir cela avec des exemples (la section 5 du RFC en contient plusieurs) et du code qui tourne.

Puisque ce format binaire est, à l'heure actuelle, surtout utilisé par "*oblivious HTTP*", on va se tourner vers les mises en œuvre de cette technique. Prenons ohttp <`https://github.com/martinthomson/ohttp`>. Il est écrit en Rust. Les instructions de compilation marchent bien, je les résume ici (c'était pour une machine Arch) :

```
export workspace=$HOME/tmp
sudo apk add mercurial gyp ca-certificates coreutils curl git make mercurial clang llvm lld ninja-build
cd $workspace
git clone https://github.com/martinthomson/ohttp.git
git clone https://github.com/nss-dev/nss ./nss
hg clone https://hg.mozilla.org/projects/nspr ./nspr
export NSS_DIR=$workspace/nss
export LD_LIBRARY_PATH=$workspace/dist/Debug/lib
cd ohttp
cargo build
cargo test
```

Et commençons par encoder une simple requête (je vous rappelle que la section 5 du RFC, et le répertoire `examples/` de ohttp contiennent d'autres exemples, y compris plus complexes) :

```
% cat request-1.txt
GET /hello.txt HTTP/1.1
User-Agent: curl/7.16.3 libcurl/7.16.3 OpenSSL/0.9.7l zlib/1.2.3
Host: www.framasoft.fr

% ./ohttp/target/debug/bhttp-convert < ./request-1.txt > request-1.bin
```

(Attention, le fichier texte doit avoir des fins de ligne CR-LF, comme le prescrit la norme HTTP et il doit y avoir une ligne vide à la fin. Si vous éditez avec emacs sur une machine Unix, `C-x RET f` puis `dos`). Regardons le fichier binaire produit (on aurait aussi pu utiliser `od`, ou bien emacs avec l'excellent mode `hexl-mode`) :

```
% xxd request-1.bin
00000000: 0003 4745 5405 6874 7470 7300 0a2f 6865  ..GET.https../he
00000010: 6c6c 6f2e 7478 7440 560a 7573 6572 2d61  llo.txt@V.user-a
00000020: 6765 6e74 3463 7572 6c2f 372e 3136 2e33  gent4curl/7.16.3
00000030: 206c 6962 6375 726c 2f37 2e31 362e 3320   libcurl/7.16.3
00000040: 4f70 656e 5353 4c2f 302e 392e 376c 207a  OpenSSL/0.9.7l z
00000050: 6c69 622f 312e 322e 3304 686f 7374 1077  lib/1.2.3.host.w
00000060: 7777 2e66 7261 6d61 736f 6674 2e66 7200  ww.framasoft.fr.
00000070: 00                                       .
```

Le premier octet est le "*framing indicator*", encodé selon la section 16 du RFC 9000 (dans le source de ohttp, regardez `bhttp/src/rw.rs`). Il vaut zéro, indiquant une requête de longueur connue. Le deuxième octet, de valeur trois, indique la longueur de la méthode HTTP, puis suivent les octets de cette méthode (G E T).

Encodons ensuite une réponse :

```
% cat response-1.txt
HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
ETag: "34aa387-d-1568eb00"
Content-Length: 51
Content-Type: text/plain

Hello World! My content includes a trailing CRLF.

% ./ohttp/target/debug/bhttp-convert < ./response-1.txt > response-1.bin

% xxd response-1.bin
00000000: 0140 c840 a104 6461 7465 1d4d 6f6e 2c20  .@.@..date.Mon,
00000010: 3237 204a 756c 2032 3030 3920 3132 3a32  27 Jul 2009 12:2
00000020: 383a 3533 2047 4d54 0673 6572 7665 7206  8:53 GMT.server.
00000030: 4170 6163 6865 0d6c 6173 742d 6d6f 6469  Apache.last-modi
00000040: 6669 6564 1d57 6564 2c20 3232 204a 756c  fied.Wed, 22 Jul
00000050: 2032 3030 3920 3139 3a31 353a 3536 2047   2009 19:15:56 G
00000060: 4d54 0465 7461 6714 2233 3461 6133 3837  MT.etag."34aa387
00000070: 2d64 2d31 3536 3865 6230 3022 0e63 6f6e  -d-1568eb00".con
00000080: 7465 6e74 2d6c 656e 6774 6802 3531 0c63  tent-length.51.c
00000090: 6f6e 7465 6e74 2d74 7970 650a 7465 7874  ontent-type.text
000000a0: 2f70 6c61 696e 3348 656c 6c6f 2057 6f72  /plain3Hello Wor
000000b0: 6c64 2120 4d79 2063 6f6e 7465 6e74 2069  ld! My content i
000000c0: 6e63 6c75 6465 7320 6120 7472 6169 6c69  ncludes a traili
000000d0: 6e67 2043 524c 462e 0d0a 00              ng CRLF....
```

Ici, le "*framing indicator*" vaut un, indiquant une réponse de longueur connue. Vous avez vu le C8 en troisième position ? C'est le code de retour 200.

Le format binaire de HTTP a été enregistré <`https://www.iana.org/assignments/media-types/media-types.xml#message`> sous le nom `message/bhttp`.

Ah, et il existe au moins une autre mise en œuvre de ce RFC, en Go, dans ohttp-go <`https://github.com/chris-wood/ohttp-go`>.