RFC 7233 : Hypertext Transfer Protocol (HTTP/1.1): Range Requests

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 15 juin 2014

Date de publication du RFC : Juin 2014

https://www.bortzmeyer.org/7233.html

Parmi les fonctions du protocole HTTP 1.1 <https://www.bortzmeyer.org/http-11-reecrit.html> qui ont eu droit à un RFC spécifique, les **intervalles**. C'est la possibilité de récupérer en HTTP, non pas la totalité d'une ressource mais une partie, un intervalle ("*range*") de celle-ci. Ce RFC les décrivait mais a été ensuite supplanté par le RFC 9110 [1].

Cette fonction est utile, par exemple, lorsqu'un transfert est interrompu en cours de route et qu'on voudrait pouvoir reprendre le téléchargement au point où il s'était arrêté, et non pas bêtement depuis le début. C'est possible avec le protocole rsync, comment faire avec HTTP ? Le client lit la partie déjà téléchargée, note sa taille, et demande au serveur le reste, en utilisant les intervalles. D'autres usages sont possibles comme de ne récupérer qu'une partie d'un document quand on n'a pas la possibilité de le traiter en entier. Attention, toutefois, ces intervalles sont optionnels pour les serveurs HTTP et un serveur qui ne les gère pas va vous envoyer toutes les données, même si vous n'aviez demandé qu'une partie.

Voici un exemple d'utilisation de ces intervalles avec le client HTTP curl et son option `-r` (`--range` en version longue). Ici, on récupère la fin du fichier, à partir du 20 001 ème octet (on compte en partant de zéro), et jusqu'à la fin :

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

```
% curl  -v --range "20000-" http://www.bortzmeyer.org/images/moi.jpg > moi-partiel.jpg

> GET /images/moi.jpg HTTP/1.1
> Range: bytes=20000-

< HTTP/1.1 206 Partial Content
< Accept-Ranges: bytes
< Content-Length: 34907
< Content-Range: bytes 20000-54906/54907
...
```

Vous avez vu les en-têtes utilisés pour décrire les intervalles. Notez qu'avec curl, on peut aussi mettre directement les en-têtes (par exemple `--header "Range: bytes=20000-"` au lieu de `--range "20000-"`) mais c'est déconseillé car cela ne corrige pas les erreurs de syntaxe.

Maintenant, comment cela fonctionne t-il sur le câble, lorsqu'on examine le trafic HTTP ? D'abord, un mot sur les unités utilisées. L'en-tête `Range:` peut indiquer l'intervalle en octets, comme plus haut, ou dans d'autres unités (pour l'instant, aucune n'a été normalisée, lorsque ce sera le cas, elles seront ajoutées dans le registre IANA <https://www.iana.org/assignments/http-parameters/http-parameters.xhtml#range-units>). L'avantage des octets est qu'ils conviennent à tous les types de ressources, les représentations des ressources étant transférées sous forme d'un flot d'octets. L'inconvénient est qu'ils ne tiennent pas compte de la syntaxe sous-jacente de la ressource. Par exemple, pour un fichier CSV, il serait sans doute plus pratique d'indiquer l'intervalle en nombre de lignes, pour permettre de récupérer un fichier non corrompu, mais cela supposerait un serveur HTTP qui connaisse la syntaxe CSV. Autre exemple d'une limite des octets : pour un fichier texte seul dont le jeu de caractères est Unicode, le nombre de caractères serait certainement une unité plus pertinente que le nombre d'octets (avec des octets, on risque de récupérer un fichier où certains caractères ne soient pas complets). Mais les serveurs HTTP ne savent pas forcément analyser les encodages d'Unicode, comme UTF-8.

Les intervalles peuvent être ouverts ou fermés. Un exemple d'intervalle fermé est `bytes=500-999` qui spécifie le deuxième groupe de 500 octets de la ressource, de l'octet 500 (inclus) à l'octet 999 (inclus aussi). Un intervalle ouvert est `bytes=-999` qui indique les 999 derniers octets du fichier (notez que le 999 n'a donc pas du tout la même signification selon qu'il y a un chiffre avant le tiret ou pas). On peut indiquer plusieurs intervalles discontinus, par exemple `bytes=500-600,601-999`. Ces intervalles doivent être dans l'ordre croissant et n'ont pas le droit de se recouvrir. Attention, un client facétieux ou malveillant peut utiliser des chiffres très grands pour un intervalle ou bien des intervalles qui se recoupent, ce qui, dans au moins un cas avec Apache, a mené à une faille de sécurité <http://httpd.apache.org/security/CVE-2011-3192.txt> (voir aussi la section 6.1 sur ce problème de sécurité).

Le serveur peut indiquer quelles unités il accepte pour les intervalles avec l'en-tête `Accept-Ranges:`, par exemple :

```
Accept-Ranges: bytes
```

Le client, pour indiquer qu'il ne demande qu'une partie de la représentation d'une ressource, utilise, on l'a vu dans l'exemple plus haut, l'en-tête `Range:`. Combiné avec la requête `GET`, cet en-tête indique que le serveur, s'il est d'accord, s'il gère l'unité indiquée, et si la syntaxe est correcte, peut n'envoyer que la partie demandée. Ces en-têtes spécifiques aux requêtes avec intervalle sont stockés

dans le registre IANA des en-têtes <https://www.iana.org/assignments/message-headers/message-header-index.html>.

Un cas un peu plus compliqué est celui où le client a récupéré une partie d'une ressource, la ressource a été modifiée sur le serveur, le client veut récupérer le reste. Les intervalles demandées par le client ne signifient alors plus rien. Un mécanisme peu connu existe pour traiter ce cas : l'en-tête `If-Range:`, suivi d'un validateur (cf. RFC 7232) dit au serveur « envoie-moi seulement l'intervalle, s'il n'y a pas eu de changement depuis ce validateur, ou bien tout le fichier sinon ».

Quelles sont les réponses spécifiques aux requêtes avec intervalles ? La section 4 en donne la liste (elles ne sont que deux, dans les autres cas, le code de retour est un classique, comme par exemple 200 pour un transfert complet réussi, et sont dans le registre IANA <https://www.iana.org/assignments/http-status-codes>). D'abord, il y a 206 qui signifie un succès (classe = 2) mais qui prévient que, comme demandé, seule une partie du contenu a été envoyé. Le serveur doit inclure un en-tête `Content-Range:` dans la réponse, indiquant ce qui a été réellement envoyé (regardez l'exemple curl au début). Si plusieurs intervalles étaient spécifiés dans la requête, le résultat est transporté dans une structure MIME de type `multipart/byteranges` décrite dans l'annexe A. Chaque partie de la réponse aura son propre `Content-Range:` indiquant où elle se situe dans la ressource originale.

L'autre code de retour possible est 416 qui indique une erreur (classe = 4) car l'intervalle indiqué n'est pas admissible. Par exemple, il y a recouvrement entre les intervalles, ou bien l'octet de départ est situé après la fin du fichier. Attention, rappelez-vous que les intervalles sont optionnels, un serveur peut ignorer l'en-tête `Range:` et, dans ce cas, répondre systématiquement avec un 200 en envoyant tout le contenu de la ressource.

Les changements depuis le RFC 2616 ? L'annexe B en fait la liste mais elle est courte, notre nouveau RFC est surtout une reformulation, avec peu de changements : insistance sur la liberté du serveur à honorer ou pas les requêtes avec intervalles, précisions sur les validateurs acceptables, création du registre des unités - même s'il ne compte pour l'instant qu'une seule unité, `bytes`...