RFC 7838 : HTTP Alternative Services

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 15 avril 2016

Date de publication du RFC : Avril 2016

https://www.bortzmeyer.org/7838.html

Une nouveauté dans le monde HTTP : ce RFC décrit un mécanisme permettant d'indiquer des endroits alternatifs où trouver le contenu demandé. Ce n'est **pas** une classique redirection : l'URI reste le même, on découple simplement l'identificateur (l'URI) et le localisateur (l'endroit où chercher et le protocole à utiliser).

Ce concept est assez nouveau dans le monde HTTP (section 1 du RFC). En effet, le RFC 7230 [1] tend à mélanger l'identificateur d'une ressource (`http://www.example.com/foo/bar`) et les moyens d'y accéder (la même ressource peut aussi être en `https://www.example.com/foo/bar` voire sur un miroir à un URL très différent). Le but du mécanisme de ce nouveau RFC 7838 est de séparer plus clairement **identificateur** et **localisateur**.

Quelques exemples pratiques sont donnés dans cette section 1 :

— Le serveur d'origine peut vouloir rediriger vers un serveur de secours si lui-même est sous forte charge, ou bien s'il sait qu'il existe un serveur plus proche de ce client particulier.
— Le serveur d'origine peut rediriger le client vers une connexion sécurisée (avec HTTPS, cf. RFC 8164) ou vers un protocole plus moderne (HTTP 2, cf. RFC 7540).
— Le serveur d'origine peut renvoyer certains clients à des serveurs spécialisés, par exemple pour le cas de clients ne gérant pas certaines options comme SNI (RFC 6066, section 3).

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc7230.txt`

Actuellement, ce genre de choses est fait par des redirections HTTP explicites (codes 301, 302, 307 ou 308, voir le RFC 7231, section 6.4). L'URL vers lequel on est redirigé est montré par le navigateur (ce qui n'est clairement pas une bonne idée dans le cas de redirections temporaires comme 307). Autre façon de faire, un relais, qui se connecte au serveur effectif, sans le dire du tout au client HTTP. Les « services alternatifs » de ce RFC 7838 sont situés entre les deux façons : annoncés au client HTTP mais pas mémorisés et pas montrés à l'utilisateur (l'URI ne **change pas**). Le contexte de sécurité n'est pas modifié (si l'URI commençait par le plan `https://`, le service alternatif devra donc présenter un certificat valide), le champ `Host:` de la requête HTTP contiendra toujours le nom original (notez qu'il existe désormais un champ `Alt-Used:` pour indiquer le nom alternatif, cf. section 5). Normalement, l'utilisateur ne verra donc pas qu'on utilise un service alternatif (ceci dit, comme son logiciel client, lui, est au courant, des options de débogage pourront permettre de l'afficher, contrairement à ce qui se passe avec un relais).

La solution est en deux parties : des concepts abstraits (section 2) puis des mécanismes concrets pour les différents protocoles, dans les sections 3 et 4. Commençons par le cadre abstrait. Un « service alternatif » existe lorsqu'un serveur d'origine (concept lui-même défini dans le RFC 6454, section 4) sait que ses ressources sont accessibles avec une combinaison {protocole, serveur, port} différente. Un tel service alternatif fait autorité (au sens du RFC 7230, section 9.1), ce n'est pas un simple miroir, ni un cache. Notez que la granularité est pour une origine complète (la combinaison {protocole, serveur, port}) : on ne peut pas faire un service alternatif qui couvrirait uniquement le chemin `/foo` de l'URI.

Un exemple de service alternatif ? Mettons que l'origine soit {http, www.example.com, 80} et que les mêmes ressources soient accessibles en HTTP 2 à {h2, new.example.com, 81}, alors on peut dire que ce {h2, new.example.com, 81} est un service alternatif de l'origine.

Notez que j'ai appelé « protocole » le premier membre du tuple. Ce n'est **pas** le plan ("*scheme*") dans l'URI (plan que pas mal de gens appellent à tort protocole). En effet, cette valeur est plus spécifique que le plan, elle peut inclure des détails techniques supplémentaires (par exemple, lorsque le plan est `http://`, on peut y accéder avec les protocoles `http` ou `h2`, ce second désignant HTTP 2.) Le protocole, pour les services alternatifs, est un nom ALPN (RFC 7301 et le registre IANA ALPN <`https://www.iana.org/assignments/tls-extensiontype-values/tls-extensiontype-values.xml#alpn-protocol-ic`

Le tuple {protocole, serveur, port} peut être accompagné d'une durée de vie, indiquant combien de temps le client HTTP peut garder en mémoire l'information sur le service alternatif.

Au passage, un mot de sécurité : le client doit évidemment être conscient que, si la communication n'est pas sécurisée (par exemple par TLS), il peut être envoyé vers un faux service alternatif (section 9). Si le service d'origine n'a pas TLS mais que le service alternatif l'a, et présente un certificat au nom du service d'origine, c'est bon. Si aucun des deux n'a TLS, le risque d'avoir été trompé est bien plus élevé. Il n'est donc pas forcément prudent d'utiliser les services alternatifs sans TLS.

Notons enfin que les services alternatifs sont optionnels. Le client n'est pas obligé de les utiliser (d'autant plus que les clients HTTP actuels ne gèrent pas ces services et n'utiliseront donc pas les alternatives, tant qu'ils n'ont pas été mis à jour). Le serveur d'origine doit donc rester prêt à assurer les requêtes.

Maintenant, les mécanismes pour signaler ces services alternatifs. En HTTP 1, en utilise l'en-tête `Alt-Svc:` (section 3, et désormais dans le registre IANA <`https://www.iana.org/assignments/message-headers/message-headers.xml`>). La réponse contient, par exemple :

```
Alt-Svc: h2="new.example.org:80"
```

Cela indique que le client devrait plutôt aller voir avec le protocole `h2` (HTTP 2), sur la machine `new.example.org`. Si on ne change que le protocole et le port, on pourrait avoir :

```
Alt-Svc: h2=":8000"
```

Il peut y avoir plusieurs valeurs, dans l'ordre des préférences décroissantes :

```
Alt-Svc: h2="alt.example.com:8000", h2=":443"
```

Par défaut, le service alternatif peut être mémorisé par le client pendant 24 h. Si on veut changer cette durée, on utilise le paramètre `ma` (*"Maximum Age"*, en secondes) :

```
Alt-Svc: h2=":443"; ma=3600
```

Si le serveur veut qu'on oublie tout ce qu'on a mémorisé sur les services alternatifs, et n'a pas envie d'attendre l'expiration des caches, il envoie la valeur spéciale `clear` :

```
Alt-Svc: clear
```

À part ce paramètre `ma`, et un autre paramètre `persist` (qui indique que, contrairement au comportement par défaut, le client HTTP ne devrait pas vider cette information de son cache si sa configuration réseau change), d'autres paramètres sont possibles dans le futur. Ils devront passer par le procédure « Examen par un expert » du RFC 5226 et seront mis dans le registre IANA <https://www.iana.org/assignments/http-alt-svc-parameters/http-alt-svc-parameters.xml>.

Et si on utilise HTTP 2, où on est en binaire et plus en texte, et où les en-têtes sont représentés d'une façon complètement différente ? Dans ce cas, on se sert de trames du type `ALTSVC` (section 4). C'est une extension au protocole original. La trame `ALTSVC` a le type 0xa (10, dans le registre IANA <https://www.iana.org/assignments/http2-parameters/http2-parameters.xml#frame-type>) et contient un champ Longueur (de l'origine), l'origine, et le service alternatif sous une syntaxe identique à celle de la section précédente.

On l'a vu, le service alternatif fait normalement autorité et doit donc gérer les mêmes ressources. Que se passe-t-il si un serveur maladroit redirige vers une machine qui n'est pas configurée comme service alternatif ? Idéalement, cette machine ne faisant pas autorité doit répondre 421 (RFC 7540, section 9.1.2). Le client doit alors réessayer avec un autre service alternatif, s'il existe, ou bien avec le serveur d'origine. (Aujourd'hui, je crois qu'il est plus fréquent que la machine ne faisant pas autorité renvoie les ressources de sa configuration par défaut.)

Un petit mot d'histoire : ce système des services alternatifs vient d'une solution spécifique à Chrome qui se nommait `Alternate-Protocol:`. Un bon article de synthèse sur ces services alternatifs est celui de l'auteur du RFC <https://www.mnot.net/blog/2016/03/09/alt-svc>.