RFC 7352 : Sieve Email Filtering: Detecting Duplicate Deliveries

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 16 septembre 2014

Date de publication du RFC : Septembre 2014

https://www.bortzmeyer.org/7352.html

Voici un nouveau test pour le langage de filtrage du courrier Sieve : `duplicate` permet de tester si un message reçu est un double d'un message déjà arrivé (en général en se fiant au `Message-Id:`) et, par exemple, d'ignorer le doublon.

Tout le monde a déjà rencontré ce problème : on est inscrit à une liste de diffusion et quelqu'un écrit à la liste en vous mettant en copie. Résultat, on reçoit deux fois le même message. La consommation de ressources réseau et système est négligeable, mais c'est gênant pour le lecteur, qui se retrouve avec deux messages à traiter au lieu d'un. Il serait bien plus pratique de détecter automatiquement le duplicata et de le résorber. Même chose si on est abonné à deux listes de diffusion (encore que, dans ce cas, des informations spécifiques à la liste et ajoutées dans les messages, comme le `Archived-At:` du RFC 5064 [1], peuvent être utiles).

Beaucoup de gens traitent automatiquement leur courrier, avec des techniques comme Sieve ou procmail et la solution décrite dans ce RFC concerne Sieve (qui est normalisé dans le RFC 5228).

La section 3 de notre RFC décrit le nouveau test `duplicate`. Il s'utilise ainsi :

```
require ["duplicate", "fileinto", "mailbox"];

if duplicate {
     fileinto :create "Trash/Duplicate";
}
```

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc5064.txt`

Ici, si le message est un doublon d'un message existant, il est mis dans le dossier `Trash/Duplicate`. Mais comment le test `duplicate` sait-il que le message est un doublon ? Il fonctionne en déterminant, pour chaque message, un identificateur unique. Lors du test, on regarde si l'identificateur est présent dans une base de données et, si ce n'est pas le cas, on l'y stocke. Contrairement à bien des extensions à Sieve, celle-ci va donc nécessiter un mécanisme de mémoire, permettant de garder trace des identificateurs de messages déjà vus.

Notez bien que la base est mise à jour lors du test `duplicate`, pas à la réception de chaque message (le test `duplicate` peut ne pas être exécuté dans tous les cas, par exemple s'il dépend d'un autre test). De même, l'ajout de l'identificateur unique dans la base ne doit être fait que si le script Sieve se termine normalement, pour éviter de considérer les messages suivants comme des doublons si le premier n'a pas pu être stocké correctement (disque plein ou autre erreur). Dans certains cas (livraison en parallèle), cela mènera à la distribution de deux copies mais le RFC considère qu'il faut éviter à tout prix la perte accidentelle d'un message donc qu'il vaut mieux quelques doublons en trop plutôt que le classement erroné d'un message comme étant un doublon.

Mais comment est déterminé l'identificateur unique indispensable au classement ? Il existe un en-tête prévu pour cela, `Message-ID:`, dont la section 3.6.4 du RFC 5322 dit bien qu'il doit être unique (il est en général formé d'une concaténation d'un grand nombre tiré au hasard et du nom de domaine du serveur, par exemple `5400B19D.70904@hackersrepublic.org`). Mais, en pratique, on voit parfois des `Message-ID:` dupliqués et il faut donc prévoir des solutions de secours.

C'est ce que contient la section 3.1 de notre RFC, avec les arguments `:header` et `:uniqueid`. Par défaut, l'identificateur unique utilisé par le test `duplicate` est le `Message-ID:`. Si on utilise le paramètre `:header`, c'est le contenu de cet en-tête qui est utilisé comme identificateur unique. Par exemple :

```
if duplicate :header "X-Event-ID" {
     discard;
   }
```

va jeter tous les messages dont l'en-tête (non standard) `X-Event-ID:` a une valeur déjà vue (cas d'un système de supervision envoyant ses informations par courrier).

Si on utilise le paramètre `:uniqueid`, c'est la valeur indiquée par le paramètre qui est l'identificateur unique. Cela n'a d'intérêt, je crois, qu'en combinaison avec les variables Sieve du RFC 5229 (l'exemple suivant utilise aussi les statuts IMAP du RFC 5232) :

```
require ["duplicate", "variables", "imap4flags"]
if header :matches "subject" "ALERT: *" {
     if duplicate :uniqueid "${1}" {
       setflag "\\seen";
     }
```

Ici, la valeur du paramètre `:uniqueid` vaut la variable `${1}`, c'est-à-dire le contenu du sujet du message, après l'étiquette « ALERT : ».

Bien sûr, on utilise `:header` ou `:uniqueid` mais jamais les deux en même temps.

Le test `duplicate` met tous les identificateurs uniques dans une seule base. Si on souhaite avoir plusieurs bases, on utilise le paramètre `:handle` (section 3.2) :

```
if duplicate :header "X-Event-ID" :handle "notifier" {
     discard;
   }
if duplicate :header "X-Ticket-ID" :handle "support" {
     # Utilisation d'une base différente: un X-Ticket-ID peut être
     # dans la base notifier mais pas dans la base support.
}
```

Pour mettre en œuvre le test `duplicate`, une façon simple de ne pas se compliquer la vie avec plusieurs fichiers est de garder dans un fichier unique un condensat de la concaténation de `:handle` avec l'identificateur unique.

On souhaite parfois que la mémorisation d'un identificateur unique ne soit que provisoire : si un message arrive un mois après, même s'il a un identificateur déjà vu, c'est probablement un nouveau message qui, par erreur ou par bogue, a un identificateur déjà existant. Il est peu probable que le même message, transmis par deux listes de diffusion différentes, traine autant... Le RFC conseille de ne mémoriser l'identificateur que pendant une semaine et fournit un paramètre `:seconds` (section 3.3) pour contrôler cette durée :

```
if not duplicate :seconds 1800 {
    notify :message "[SIEVE] New interesting message"
       "xmpp:user@im.example.com";
```

La durée est en secondes : ici, on indique une expiration au bout de seulement une demi-heure, peut-être parce qu'on ne veut absolument pas perdre de messages et qu'on craint que les identificateurs ne soient réutilisés trop fréquemment. (L'extension pour une notification par XMPP est dans le RFC 5437.)

La durée indiquée par le paramètre `:seconds` se compte à partir du premier message vu. Les messages ultérieurs portant le même identificateur ne remettent pas le compteur à zéro. Si on utilise le paramètre `:last`, par contre, la durée est comptée à partir du dernier message vu : tant que des messages arrivent avec cet identificateur, il n'y a pas d'expiration.

Comme toutes les extensions de Sieve, `duplicate` doit être annoncée au début du script Sieve (RFC 5228, section 6) :

```
require ["duplicate"]
```

Et la sécurité (section 6) ? Pour éviter de se faire DoSer, il faut imposer une limite à la taille de la base (en supprimant les identificateurs les plus anciens). Autrement, un flot important de messages avec chacun un identificateur unique pourrait faire exploser le disque dur.

Un autre problème de sécurité est le risque de faux positif : typiquement, on jette les messages dupliqués. Or des identificateurs censés être uniques, comme le `Message-ID:` ne le sont pas toujours. On risque donc, avec le test `duplicate`, de jeter à tort des messages. Il est donc prudent de ne pas détruire les messages dupliqués (action `discard` de Sieve) mais plutôt de les stocker dans une boîte spéciale (avec `fileinto`).

Il peut y avoir aussi de légères conséquences pour la vie privée : un utilisateur qui a détruit un message sur le serveur sera peut-être surpris que la base des identificateurs uniques continue à stocker le `Message-ID:` de ce message, qui peut être assez révélateur. Ceci dit, ce n'est pas pire que les actuels journaux (genre `/var/log/mail.log`, qui contient aussi ce genre d'informations).

Il y a apparemment deux mises en œuvre de Sieve qui gèrent cette extension.

Sinon, en dehors du monde Sieve, voici la solution traditionnelle avec procmail pour obtenir le même résultat :

```
:0:/var/tmp/.duplicate.lock
* ? formail -D 8192 $HOME/.duplicate.procmail-cache
$HOME/.mail-duplicates
```

Qui se lit : si le `Message-ID:` est déjà dans le fichier `$HOME/.duplicate.procmail-cache` (qui garde les 8 192 dernières entrées), on ne le distribue pas dans la boîte normale, mais dans `$HOME/.mail-duplicates`