RFC 8054 : Network News Transfer Protocol (NNTP) Extension for Compression

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 26 janvier 2017

Date de publication du RFC : Janvier 2017

http://www.bortzmeyer.org/8054.html

Ce nouveau RFC définit un mécanisme standard de compression des "*news*" échangées en NNTP, sur Usenet.

NNTP, normalisé dans le RFC 3977[1] est un protocole gourmand en débit. Comprimer les données transmises est donc très souhaitable. C'est aussi un protocole **très** ancien, ce qui se voit dans certaines références du RFC, comme l'allusion à la compression PPP du RFC 1962 ou bien à la compression par modem comme V42bis :-)

Mais, malgré ce besoin de compression, il n'y avait pas encore de solution standard en NNTP. Un certain nombre de mécanismes non-standards avaient été déployés avec des noms comme `XZVER`, `XZHDR`, `XFEATURE COMPRESS`, ou `MODE COMPRESS`. Outre l'absence de normalisation, ils souffraient de ne comprimer que les réponses du serveur de "*news*".

Compte-tenu du déploiement de plus en plus fréquent de TLS, pour assurer la confidentialité des échanges, il avait été envisagé à une époque de compter sur le mécanisme de compression de TLS (RFC 4642, remis en cause par le RFC 8143). Celui-ci présente malheureusement des dangers, qui fait que son usage est déconseillé dans beaucoup de cas (section 3.3 du RFC 7525, section 2.6 du RFC 7457, et RFC 8143). En outre, la solution de ce RFC bénéficie de davantage de souplesse : elle peut par exemple n'être activée qu'une fois l'authentification faite, pour éviter les attaques comme CRIME (voir aussi les sections 2.2.2 et 7 de notre RFC, pour tous les détails de sécurité).

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc3977.txt`

Pour assurer l'interopérabilité maximale, un seul algorithme de compression est défini, et il est, logiquement, obligatoire. Cela garantit qu'un client et un serveur NNTP auront toujours cet algorithme en commun. Il s'agit de Deflate, normalisé dans le RFC 1951.

(Un petit point qui n'a rien à voir avec NNTP et la compression : comme le demandait l'"*Internet-Draft*" qui a donné naissance à notre RFC, j'ai mis un accent à la première lettre du nom d'un des auteurs, ce qui n'est pas possible dans le RFC original, cela ne le sera que lorsque le RFC 7997 sera mis en œuvre.)

Maintenant, les détails techniques (section 2 du RFC). Le serveur doit annoncer sa capacité à comprimer en réponse à la commande `CAPABILITIES`. Par exemple (C = client, S = serveur) :

```
[C] CAPABILITIES
[S] 101 Capability list:
[S] VERSION 2
[S] READER
[S] IHAVE
[S] COMPRESS DEFLATE SHRINK
[S] LIST ACTIVE NEWSGROUPS
[S] .
```

L'annonce de la capacité est suivie de la liste des algorithmes gérés. On trouve évidemment l'algorithme obligatoire `DEFLATE` mais aussi un algorithme non-standard (imaginaire, ce n'est qu'un exemple) `SHRINK`.

Le client peut alors utiliser la commande `COMPRESS`, suivie du nom d'un algorithme (cette commande a été ajoutée au registre IANA des commandes NNTP <https://www.iana.org/assignments/nntp-parameters/nntp-parameters.xml#nntp-parameters-1>). Voici un exemple où le serveur accepte la compression :

```
[C] COMPRESS DEFLATE
[S] 206 Compression active
(À partir de là, le trafic est comprimé)
```

Attention à ne pas confondre la réponse du serveur à une demande de ses capacités, et la commande envoyée par le client (dans les deux cas, ce sera une ligne `COMPRESS DEFLATE`).

Et voici un exemple où le serveur refuse, par exemple parce que la compression a déjà été activée :

```
[C] COMPRESS DEFLATE
[S] 502 Command unavailable
```

Si on utilise TLS, ce qui est évidemment recommandé pour des raisons de confidentialité et d'authentification, l'envoyeur doit d'abord comprimer, puis (si SASL est activé) appliquer SASL (RFC 4422), puis seulement à la fin chiffrer avec TLS. À la réception, c'est bien sûr le contraire, on déchiffre le TLS, on analyse SASL, puis on décomprime.

Voici un exemple d'un dialogue plus détaillé, avec TLS et compression :

```
[C] CAPABILITIES
[S] 101 Capability list:
[S] VERSION 2
[S] READER
[S] STARTTLS
[S] AUTHINFO
[S] COMPRESS DEFLATE
[S] LIST ACTIVE NEWSGROUPS
[S] .
[C] STARTTLS
[S] 382 Continue with TLS negotiation
(Négociation TLS)
(Désormais, tout est chiffré)
[C] CAPABILITIES
[S] 101 Capability list:
[S] VERSION 2
[S] READER
[S] AUTHINFO USER
[S] COMPRESS DEFLATE
[S] LIST ACTIVE NEWSGROUPS
[S] .
[C] AUTHINFO USER michu
[S] 381 Enter passphrase
[C] AUTHINFO PASS monsieur
[S] 281 Authentication accepted
[C] CAPABILITIES
[S] 101 Capability list:
[S] VERSION 2
[S] READER
[S] POST
[S] COMPRESS DEFLATE
[S] LIST ACTIVE NEWSGROUPS
[S] .
[C] COMPRESS DEFLATE
[S] 206 Compression active
(Désormais, toutes les données envoyées sont comprimées, puis chiffrées)
[C] CAPABILITIES
[S] 101 Capability list:
[S] VERSION 2
[S] READER
[S] POST
[S] LIST ACTIVE NEWSGROUPS
[S] .
```

Et voici deux exemples où le serveur refuse la compression. D'abord parce qu'il ne peut pas (manque de mémoire, par exemple) :

```
[C] COMPRESS DEFLATE
[S] 403 Unable to activate compression
```

Et ici parce que le client essaie d'utiliser un algorithme que le serveur ne connait pas :

```
[C] COMPRESS SHRINK
[S] 503 Compression algorithm not supported
```

La liste des algorithmes standards (pour l'instant réduite à un seul) est dans un registre IANA <https://www.iana.org/assignments/nntp-compression-algorithms/nntp-compression-algorithms.xml>.

NNTP est un protocole dont les spécificités posent des problèmes amusants lorsqu'on veut comprimer son trafic (section 3 du RFC). Les messages sont très divers, ce qui peut être contrariant pour une compression fondée sur un dictionnaire. Les réponses à certaines commandes (`DATE`, `GROUP`, `NEXT`, et le `CHECK` du RFC 4644) sont peu comprimables. Par contre, les réponses à `LIST`, `LISTGROUP` ou `NEWNEWS` sont facilement réduites à 25 à 40 % de la taille originale avec zlib.

En outre, les "*news*" envoyées sont dans des formats différents. Un article sera parfois du texte seul, relativement court (et souvent uniquement en ASCII) et se comprimera bien. Les textes plus longs sont souvent envoyés sous un format déjà comprimé et, là, le compresseur NNTP va s'essouffler pour rien. Mais il y a aussi souvent des données binaires (images, par exemple), encodées en Base64 ou uuencode. On peut souvent les réduire à 75 % de l'original. (Deflate marche bien sur des données en 8 bits mais l'encodage lui dissimule la nature 8-bitesque de ces données.) Si les données sont encodées en yEnc, elles seront moins compressibles.

Il y a apparemment au moins un logiciel serveur (INN) et un client (flnews <`http://micha.freeshell.org/flnews/`>) qui gèrent cette compression.

Merci à Julien Élie pour sa relecture attentive (et pour avoir trouvé au moins une grosse faute.)