RFC 1144 : Compressing TCP/IP headers for low-speed serial links

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 26 septembre 2010

Date de publication du RFC : Février 1990

https://www.bortzmeyer.org/1144.html

Ce RFC n'est sans doute plus utilisé nulle part aujourd'hui mais, pendant des années, presque toutes les connexions à l'Internet depuis la maison (et, souvent, depuis le bureau), passaient par un lent modem, ce qui justifiait des efforts important pour comprimer les paquets, notamment (et c'était le but de de ce RFC), les en-têtes IP et TCP en profitant des diverses redondances qu'ils contiennent. La « compression Van Jacobson » (on voyait parfois dans les documentations « *"VJ compression"* »), documentée dans ce RFC, était un outil indispensable si on voulait pouvoir utiliser son modem de manière satisfaisante.

Contrairement à beaucoup de RFC de la grande époque, il est long (49 pages) car il y a beaucoup de détails techniques à spécifier (et il y a du code C à la fin). Comme plusieurs RFC de ce temps, il existe également en version PostScript, en <http://www.rfc-editor.org/rfc/rfc1144.ps>. Cette version est plus jolie et est donc recommandée pour la lecture de ce document difficile. Elle a en outre l'avantage de permettre l'inclusion de graphiques, comme la figure 8 qui indique le débit réel en fonction de la MTU.

La section 1 du RFC rappelle le contexte... de l'époque (1990). Des ordinateurs chez les particuliers (enfin, chez les plus riches), un Internet qui, certes, ne figurait pas sur l'écran radar des décideurs et des journalistes (surtout en France où l'Internet était mal vu et où il n'y en avait que pour le Minitel qui, contrairement aux techniques comme SLIP (RFC 1055 [1]) utilisées pour l'Internet sur modems, avait un débit asymétrique), des modems allant de 300 b/s (modems acoustiques <http://fr.wikipedia.org/wiki/Fichier:AJ_311_Acoustic_modem.JPG>) à 19 200 b/s (le record - non normalisé - de la norme V.32). Il existait avant ce RFC d'autres techniques de compression d'en-têtes (RFC 914) mais celle-ci était plus performante. À noter que le protocole « Van Jacobson » est spécifique à TCP, car il

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc1055.txt

dépend du format d'en-tête de TCP, et de plusieurs fonctions de TCP, comme la réémission des paquets perdus, qui a permis de mettre aucune fonction de correction d'erreur. (Le RFC 2508 a proposé une méthode pour les autres protocoles de transport.)

Quel était le problème lorsqu'on se connectait à l'Internet via un modem, en utilisant des protocoles comme SLIP ? (PPP sera normalisé quelques mois plus tard, dans le RFC 1172, mais mettra très longtemps à être déployé.) Le problème, expliqué dans la section 2 du RFC, est que des applications interactives (telnet, à l'époque) et des applications transférant des données en bloc (NNTP, SMTP et FTP à l'époque) coexistent sur le câble et que les premières, qui exigent un bon temps de réponse, souffrent de la taille excessive des en-têtes (excessive par rapport au contenu « utile »). Si on se fixe un objectif (raisonnable pour une application interactive) de 100 à 200 ms de délai, on voit qu'un paquet TCP complet, avec ses en-têtes, est trop gros (sur le format des en-têtes, cf. RFC 791, section 3.1, pour IP et RFC 793, section 3.1, pour TCP). Le paquet fait 41 octets pour un seul caractère transmis (et autant pour l'écho effectué par le shell distant), ce qui nécessiterait 4 kb/s pour être transmis en moins de 200 ms. Même pour un modem de ce débit, si un transfert de données est en cours en même temps, les paquets de telnet peuvent avoir besoin d'attendre que la ligne soit libérée. Cela interdit de faire des paquets de données trop gros (ils bloqueraient la ligne pendant plus de 200 ms) et donc la taille relative des en-têtes est aussi un problème pour eux. Le cahier du charge du RFC était donc de permettre de faire du telnet sur des modems à 300 b/s. Un humain tapant environ cinq caractères par seconde impose au maximum cinq octets d'en-tête, ce qui permettrait de réduire la taille des paquets de transfert de données et donc de résoudre deux problèmes d'un coup.

Le protocole lui-même figure en section 3. Il n'est pas évident de comprimer les en-têtes IP et TCP. Contrairement aux protocoles OSI, chaque champ a une utilité et ne peut pas être simplement omis. En revanche, certains de ces champs sont constants au cours d'une même connexion TCP et certains évoluent de manière prédictible. Si les deux parties de la connexion gardent en mémoire l'état de celle-ci, il leur suffit de transmettre avec chaque paquet l'identificateur de la connexion et les champs qui ont changé de façon imprévue. Tout le reste peut être reconstitué localement. C'est sur cette observation que repose l'algorithme Van Jacobson. Après, il ne reste plus qu'à identifier un par un, pour chaque champ, s'il est constant, s'il change de manière prévisible, ou bien s'il peut être déduit de l'information données par d'autres couches (par exemple, le champ "*Total Length*" de l'en-tête IP est souvent doublé par une fonction de la couche 2 qui indique la même information). Comme exemple de champ qui change de manière prédictible, on peut citer les numéros de séquence TCP, qui s'incrémentent avec le nombre d'octets envoyés. À noter que toutes les informations vont du compresseur vers le décompresseur : il n'y a pas de rétroaction dans le protocole VJ.

Les détails pratiques figurent en section 3.2. Par exemple, 3.2.2 indique le format du paquet comprimé, 3.2.3 les actions du compresseur et 3.2.4 le mécanisme que doit suivre le décompresseur. Ce dernier doit appliquer aveuglément son algorithme, il ne peut pas demander confirmation ou éclaircissements au compresseur. Il existe quatre types de paquets :

— `TYPE_ERROR` : la couche 2 indique que le paquet a été mal reçu. Le décompresseur le jette et compte sur le TCP à l'autre bout pour réémettre.
— `TYPE_IP` : un paquet IP non-TCP, non modifié. L'algorithme VJ ne les comprime pas et le décompresseur n'a donc rien à faire.
— `UNCOMPRESSED_TCP` (identifié par le champ « Protocole » de IP, normalement à 6 - pour TCP - et qui est ici l'index d'une connexion) : la connexion TCP en question vient juste de commencer ou bien un problème s'est produit, nécessitant une remise à zéro de l'état du compresseur. En recevant ce paquet, le décompresseur doit mettre à jour son état, enregistrer l'index de la connexion et les valeurs des en-têtes.
— Autrement, le paquet est de type `COMPRESSED_TCP` et le décompresseur a alors le plus de travail à faire, puisqu'il doit reconstituer le paquet TCP complet (en regardant les valeurs qui avait été stockées pour cet index) qu'il passera au TCP local.

Tout cela, c'est très joli, si tout se passe bien. Mais s'il y a une erreur ? C'est fréquent avec les modems... La section 4 couvre ce problème. D'abord, il faut détecter l'erreur. Le principe de la compression étant de supprimer la redondance, la détection d'erreur, typiquement basée sur une information redondante, devient difficile. N'importe quel bruit aléatoire sur la ligne peut être « décompressé » en un paquet TCP/IP valide. La somme de contrôle, trop courte, n'est pas une protection suffisante : sur une ligne à 9 600 b/s, un parasite de 400 μs (ce qui est très court) va corrompre 16 bits, ce que la somme de contrôle va probablement rater. Le protocole VJ dépend donc essentiellement de la détection d'erreur de la couche 2. Un autre cas où un protocole fondé sur une rétroaction du décompresseur vers le compresseur ne marcherait pas est le cas où le paquet serait totalement détruit : le décompresseur ne peut alors même pas savoir qu'il y a eu un problème. Dans ce cas, c'est TCP qui détectera l'erreur, en ne recevant pas d'accusé de réception.

Une fois l'erreur détectée par le décompresseur, comment la rattraper (section 4.2) ? Le compresseur a la responsabilité d'envoyer un paquet `UNCOMPRESSED_TCP` pour signaler au décompresseur qu'il doit se resynchroniser. Comment le compresseur sait-il, lui, qu'un paquet est mal arrivé, ou pas arrivé du tout ? Il observe simplement les paquets TCP : si le numéro de séquence n'augmente pas (à part quelques cas spéciaux que le compresseur doit reconnaitre), c'est qu'il y a retransmission, et que des paquets ont été perdus. Le compresseur sait alors qu'il doit envoyer un paquet non comprimé. Le recevant, le décompresseur saura qu'il y a eu un problème et commencera un contexte neuf.

Le protocole VJ permet-il de configurer certains de ses paramètres ? Comme l'explique la section 5, il n'y en a pas beaucoup, car il n'y a pas de moyen de les transmettre entre les deux parties. Les deux paramètres possibles sont « compression activée ou pas » et nombre de contextes (de flux TCP simultanés) à conserver. Le premier paramètre (activer ou pas la compression VJ) ne se négocie pas, il doit donc être configuré de manière identique des deux côtés. À l'époque, c'était une source d'ennuis fréquents, qui se terminait souvent par « on va essayer avec et sans VJ et on verra bien lequel marche ». Le second paramètre est fixé à 16, avec juste une exception pour le changer si le protocole de couche 2,5 fournit un moyen pour cela (SLIP n'en fournit pas).

Quels sont les résultats obtenus effectivement par la compression VJ ? La section 5.3, très détaillée, fait la comparaison entre la compression VJ et une compression générique (Lempel-Ziv) appliquée à tout le paquet. Pour du trafic interactif (telnet), VJ l'emporte d'un facteur 3. Pour du trafic de données (FTP), c'est le contraire. La section 6 revient sur les performances du compresseur et mesure le temps pris sur les machines typiques de l'époque (Sun 3/60 ou DECstation 3100). La plupart des machines récentes (pour l'époque !) arrivaient à comprimer et à décomprimer à 64 kb/s (ce qui permettait même d'utiliser la compression VJ pour le RNIS).

Pour les programmeurs en C, l'annexe A fournit une mise en œuvre complète de l'algorithme, documentée très en détail (avec même un petit réglement de compte contre les machines qui osaient être petit-boutiennes comme le Vax). Elle était distribuée à l'époque dans CSLIP (mise en œuvre de SLIP avec compression). Une copie complète du code se trouve en <ftp://ftp.ee.lbl.gov/cslip.tar.Z>, à l'endroit exact indiqué par le RFC il y a vingt ans ! Bel existence de persistence d'URL <http://www.w3.org/Provider/Style/URI> ! (Par contre, l'adresse IP, indiquée dans le RFC, a changé, bon exemple du fait que les noms de domaine fournissent de la permanence et pas de la convivialité <https://www.bortzmeyer.org/pourquoi-le-dns.html>.)

Avec PPP, la compression VJ peut être négociée en toute sécurité (cf. RFC 1332, section 4). En revanche, SLIP n'ayant pas de mécanisme de négociation. L'annexe B suggère donc une méthode pour détecter automatiquement, à l'une des extrémités d'une liaison SLIP, si l'autre extrémité est en train de comprimer avec l'algorithme VJ. Cela permet, par exemple, d'avoir un « serveur » SLIP qui s'adapte automatiquement à des clients différents.

Aujourd'hui, l'algorithme VJ n'est plus utilisé. Des techniques plus perfectionnées l'ont remplacé, notamment ROHC (RFC 5795). Mais, pendant de nombreuses années, la plupart des connexions à la maison, ou dans des sites éloignées qui n'avaient pas d'autre liaison que le vieux réseau téléphonique, passaient par cet algorithme.

Et merci à Ludovic Rousseau, Pierre-Yves Lochou et Laurent Chemla pour m'avoir aidé lors de mes débuts avec la compression VJ !