RFC 6936 : Applicability Statement for the use of IPv6 UDP Datagrams with Zero Checksums

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 1 mai 2013

Date de publication du RFC : Avril 2013

https://www.bortzmeyer.org/6936.html

Le RFC 2460 [1] qui normalise IPv6 indique dans sa section 8.1 que les paquets UDP sur IPv6 **doivent** avoir une somme de contrôle (elle était facultative en IPv4) calculée selon le RFC 1071. Dans quels cas est-elle réellement nécessaire ? Peut-on s'en passer parfois ? Ce nouveau RFC discute des conditions qui peuvent permettre de mettre zéro dans le champ "*Checksum*", indiquant qu'on n'a pas utilisé de somme de contrôle. Pour résumer : si on veut se passer de somme de contrôle, cela doit se faire application par application, il faut être sûr qu'on supporte les paquets corrompus ou délivrés à tort, et il faut tester le chemin emprunté pour être sûr que les « nouveaux » paquets passent bien.

Ce RFC discute des solutions à un problème de performance d'UDP et des conditions dans lesquels on peut ne plus mettre de somme de contrôle. Il sert donc d'arrière-plan au RFC normatif, le RFC 6935 qui définit les nouvelles règles pour UDP sur IPv6 (somme de contrôle facultative) et fait donc un choix ferme parmi les propositions discutées ici. Notez toutefois qu'il y a pas mal de recouvrement et de redites entre les deux RFC.

UDP est très utilisé, et pour beaucoup d'applications (le RFC 8085 donne des conseils à ce sujet). Un des usages fréquents est pour construire des tunnels, afin de pouvoir passer à travers les millions de "*middleboxes*" stupides qui infestent l'Internet, et qui bloquent souvent tous les protocoles autres que UDP et TCP. Or, un routeur d'entrée ou de sortie de tunnel peut avoir à encapsuler ou décapsuler des millions de paquets par seconde et la génération ou la vérification des sommes de contrôle de chaque paquet peut avoir un effet mesurable sur ses performances (sections 1.3.2 et 1.3.3). D'autant plus qu'il n'est pas toujours possible d'utiliser le matériel spécialisé du routeur pour cela car il ne donne souvent

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc2460.txt

accès qu'aux N premiers octets du paquet (avec N typiquement ¡ 128), alors que le calcul de la somme de contrôle nécessite d'accéder à tout le paquet.

En UDP/IPv4, la solution est de couper la somme de contrôle sur le paquet externe (le protocole interne, qui n'est pas forcément UDP, reste protégé par sa propre somme de contrôle), ce qui est légal en IPv4. Le tunnel devient alors un lien virtuel non fiable, pouvant corrompre des paquets (comme c'est le cas d'un certain nombre de liens physiques). C'est par exemple la solution recommandée pour un gros consommateur de tunnels UDP, LISP. Il est donc tentant de l'utiliser également en UDP/IPv6.

Cela peut poser des problèmes avec certaines "*middleboxes*" qui vérifient la somme de contrôle UDP et n'accepteront pas d'y trouver zéro. Décréter que la somme de contrôle UDP en IPv6 devient facultative ne signifie donc pas que ces paquets optimisés pourront voyager dès demain sur tout l'Internet : un certain nombre de pare-feux les bloqueront. Cela peut sembler un problème rédhibitoire (on utilisait UDP justement pour passer facilement à travers les "*middleboxes*" puis on réforme les règles d'UDP, créant des paquets qui sembleront illégaux à beaucoup de ces "*middleboxes*") mais il faut noter que la plupart des ces "*middleboxes*" ne font pas encore d'IPv6 et que donc les premiers déploiements ont des chances de se faire avec des "*middleboxes*" conformes à la nouvelle règle.

À part l'UDP existant, et l'UDP avec somme facultative du RFC 6935, quelles sont les autres technologies disponibles pour des tunnels (section 2 du RFC) ? Il y a UDP Lite (RFC 3828), avec une somme de contrôle qui ne couvre que le début du paquet (et est donc rapide à calculer), et qui est certes mis en œuvre dans le noyau Linux mais qui est peu reconnu par les "*middleboxes*". Il utilise un identifiant (champ "*Next Header*" qui identifie en général le protocole de transport) différent de celui d'UDP (136 contre le 17 d'UDP) et est donc souvent bloqué par des pare-feux trop zélés. Il y a les techniques de tunnels génériques comme IP-in-IP et GRE. Elles n'ont pas de somme de contrôle du tout et sont donc rapides. Ce sont les solutions idéales pour les tunnels mais elles aussi utilisent un champ "*Next Header*" qui peut ne pas être connu de la "*middlebox*" (94 pour IP-in-IP et 47 pour GRE) et ils passent certainement moins souvent qu'UDP (d'où le choix d'UDP pour LISP). Notons aussi que la section 6 déconseille les tunnels récursifs (un tunnel dans un tunnel dans un tunnel...) qui augmentent les risques d'erreur pour le paquet le plus interne (et ajoutent une grande complexité à la fragmentation).

La section 3 examine ensuite les choses qui peuvent aller mal avec la solution retenue par le RFC 6935. Si on ne met pas de somme de contrôle dans le paquet UDP (ou, plus exactement, qu'on met zéro, indiquant qu'on n'a pas calculé de somme et que le récepteur ne doit pas vérifier), et qu'un bit du paquet est modifié, que peut-il se passer d'horrible ? Pour IPv4, la somme de contrôle UDP ne servait qu'à protéger les numéros de port et les données, des informations comme l'adresse IP de destination étaient protégées par la somme de contrôle de la couche 3. Celle-ci a disparu en IPv6 et la somme de contrôle UDP doit donc tout faire. Si elle est erronée, on n'est même pas sûr que le paquet nous était bien destiné (l'adresse IP de destination a pu être modifiée).

Comme l'expliquent le RFC 3819 ou des études comme « "*When the CRC and TCP Checksum Disagree*" <http://conferences.sigcomm.org/sigcomm/2000/conf/paper/sigcomm2000-9-1.ps.gz> », la corruption arrive réellement dans l'Internet. Des mesures ont indiqué jusqu'à 1 paquet corrompu sur 10 000 (cela dépend fortement du chemin suivi : le taux de corruption est quasi-nul sur certains chemins). Le RFC note (section 3.2) qu'on ne sait pas exactement où la corruption survient : cela peut être dans la RAM du routeur, sur le câble, dans les récepteurs et les émetteurs optiques ou électriques, etc.

Si c'est l'adresse IP de destination qui est modifiée, le paquet peut n'arriver nulle part, ou bien arriver à une machine qui ne l'attend pas. Si aucun programme n'écoute sur le port en question, le paquet est jeté (et un message d'erreur envoyé à la source, qui sera bien surprise). Si un programme écoute, il va recevoir un paquet invalide. Même chose si c'est le port de destination seul qui a été corrompu. Si l'application n'a pas de mécanisme (comme des numéros de séquence) pour détecter le problème,

le paquet invalide pourra être accepté. Donc, une règle importante : une application qui est prête à débrayer les sommes de contrôle UDP doit être prête à réagir proprement si elle reçoit un datagramme envoyé à tort. C'est une des raisons pour lesquelles le RFC exige que la suppression de la somme de contrôle UDP ne se fasse pas globalement, pour toute la machine ou pour toute l'interface réseau, mais uniquement sur requête explicite d'une application majeure et consentante (section 3.5)

Si c'est l'adresse IP source qui est corrompue, le paquet arrivera sans doute à la bonne destination (mais pas à coup sûr, par exemple en cas de filtrage RFC 2827). Si l'application garde en mémoire ses correspondants, le paquet de ce correspondant inconnu sera jeté. Si elle ne le fait pas, si une réponse est envoyée... elle le sera au mauvais correspondant.

Et si c'est l'information de fragmentation qui est corrompue ? Cela peut empêcher le réassemblage (menant à la perte du paquet) mais aussi, ce qui est bien pire, conduire à des réassemblages erronés (par exemple, si le champ ID est modifié, attribuant le fragment à un autre datagramme). Comme le note le RFC 5722, il faut multiplier les tests lors du réassemblage.

Nous avons vu plus haut qu'un certain nombre de boîtiers intermédiaires jetteraient probablement sans autre forme de procès ces « nouveaux » paquets UDP avec une somme de contrôle nulle (ils étaient illégaux avant la sortie du RFC 6935). Il est donc essentiel, pour une application qui veut utiliser cette nouveauté, de tester le chemin et de vérifier que rien sur le trajet ne jette les paquets sans somme de contrôle. Il faut aussi penser à répéter ce test de temps en temps : le chemin peut changer (par exemple parce que BGP l'a décidé) et passer tout à coup par des boîtiers différents. (Et il faut faire le test dans les deux sens, ne serait-ce que parce que le routage n'est pas forcément symétrique.)

Dans tous les cas, il faut se rappeler que la somme de contrôle ne fournit de protection que contre la corruption accidentelle. Ce n'est pas un mécanisme d'authentification, et elle ne protège pas du tout contre un attaquant (qui peut modifier le paquet sans changer la somme de contrôle).

Les règles d'applicabilité sont formalisées dans les sections 4 (pour les implémentations) et 5 (pour l'usage de ces implémentations). Les programmes qui mettent en œuvre UDP sur IPv6 :

— Doivent, par défaut, calculer une somme de contrôle et la mettre dans les paquets,
— Doivent permettre de débrayer cette somme de contrôle pour certains ports (ceux utilisés par l'application volontaire),
— Même dans ce cas, il faut fournir un mécanisme pour permettre à l'application d'envoyer certains datagrammes avec une somme de contrôle (pour les paquets "*keep-alive*", par exemple),
— Doivent, par défaut, rejeter les datagrammes entrants ayant une somme de contrôle nulle (et le journaliser quelque part),
— Doivent fournir un moyen de supprimer ce contrôle des paquets UDP entrants, pour certains ports,
— Vérifier les paquets ICMP entrants qui se réfèrent à un paquet UDP ayant une somme de contrôle nulle, par exemple que les adresses dans le paquet cité dans le message ICMP soient correctes.

Cela, c'était pour l'implémentation d'UDP et d'IPv6 (qui, dans un Unix, est dans le noyau). Maintenant, pour les applications qui l'utilisent, la section 5 dit qu'elles :

— Doivent se souvenir qu'un paquet UDP/IPv6 sans somme de contrôle n'est pas équivalent à un paquet UDP/IPv4 sans somme de contrôle : IPv4 a toujours une somme de contrôle en couche 3, que n'a pas IPv6. UDP/IPv6 sans somme de contrôle est donc plus risqué et doit n'être utilisé qu'à bon escient.
— Doivent n'activer la suppression de la somme de contrôle que pour un petit nombre de ports, ceux qu'elles utilisent,
— Doivent avoir un mécanisme de contrôle d'intégrité au dessus d'UDP, idéalement, ce mécanisme devrait détecter la corruption le plus tôt possible, par exemple pour éviter d'injecter des paquets corrompus dans le flot transporté par un tunnel,

— Doivent tester le chemin (est-ce que les paquets sans somme de contrôle arrivent bien ?) et fonctionner même si le chemin change et que les paquets passent soudain par une nouvelle "*middlebox*", ayant un comportement différent,
— Doivent utiliser des paquets avec somme de contrôle pour les contrôles, les tests, etc, pour faire la différence entre une rupture complète du lien, et un problème ne touchant que les paquets sans somme de contrôle.

Voilà, les règles d'applicabilités sont posées. Mais le RFC compte aussi une annexe intéressante, l'annexe A, qui est consacrée aux propositions alternatives. La solution choisie par le RFC 6935 a été d'autoriser les sommes de contrôle nulles. Mais on aurait pu faire différemment. Par exemple, le problème de performance et d'accès à la totalité du paquet aurait pu être traité par les calculs incrémentaux de somme de contôle du RFC 1624 (mais ils ne marchent pas quand il y a fragmentation IP). On aurait pu, comme cité, utiliser un protocole comme UDP Lite, ou UDPTT <http://tools.ietf.org/id/draft-fairhurst-6man-tsvwg-udptt>. Il avait aussi été proposé une mesure unilatérale, qu'un receveur ignore la somme de contrôle, quoiqu'ait mis l'émetteur (c'était plus simple mais il était alors impossible de détecter la corruption, pour tous les paquets). D'autres avaient suggéré une nouvelle option dans un en-tête "*Destination Options*", ce qui revenait, à mon avis, à réintroduire une somme de contrôle en couche 3, ce qu'IPv6 avait délibérement abandonné. De toute façon, beaucoup de "*middleboxes*" auraient jeté les paquets avec cet en-tête (qui est rare, en pratique). Ces différentes propositions sont examinées en fonction de plusieurs critères : traversabilité des "*middleboxes*", performances, déployabilité et résistance à la corruption de paquets.

Pour la traversée des "*middleboxes*", l'UDP actuel ou bien un UDP avec un calcul incrémental des sommes de contrôle, est ce qui marche le mieux. Aucun changement à faire. L'UDP sans somme de contrôle du RFC 6935 aura sans doute quelques soucis (actuellement, de tels paquets sont illégaux), ainsi que les paquets avec l'en-tête "*Destination Options*" (pourtant légaux). Et les tunnels génériques seront sans doute bloqués par défaut par la plupart des pare-feux, ainsi qu'UDP Lite.

C'est à peu près la même chose pour la déployabilité : l'UDP actuel existe déjà et n'a pas à être déployé. Les sommes de contrôle facultatives du RFC 6935 nécessiteront au moins un changement dans les extrémités (ne pas calculer la somme, ne pas la vérifier) mais aussi dans certaines "*middleboxes*" qui se mêlent de vérifier la somme de contrôle, alors qu'on ne leur a rien demandé.

Pour les performances, l'UDP actuel est certainement le pire, puisqu'il faut calculer la somme de contrôle sur la totalité du paquet. C'est bien cela qui a motivé tout ce travail sur la réforme des sommes de contrôle UDP. UDP Lite, qui ne vérifie qu'une partie du paquet, et l'UDP réformé du RFC 6935 ont des bien meilleures performances (c'est aussi le cas des tunnels génériques).

Bref, il n'y a pas de solution parfaite, qui résolve tous les problèmes. Si on veut avant tout passer partout, l'UDP actuel est la meilleure solution. Si on veut une solution propre architecturalement, IP-in-IP est parfait. Si on se préoccupe des performances, l'UDP avec somme de contrôle facultative du RFC 6935 est une bonne approche et reçoit un avis favorable dans ce RFC.