Mon fichier a-t-il été modifié pendant son voyage ?

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 7 juin 2020

https://www.bortzmeyer.org/somme-controle-corruption.html

Une très intéressante discussion à l'IETF a lieu en ce moment sur la modification des données lorsqu'elles circulent sur l'Internet : les mécanismes de protection contre ces modifications suffisent-ils ? Ou bien est-ce que nous recevons de temps en temps des données modifiées, sans nous en rendre compte ?

Le problème a été posé par Craig Partridge dans un message récent <https://mailarchive.ietf.org/arch/msg/ietf/UlX-EcQvxErWPUTsrf0lUyRZYMQ>. Pour le comprendre, il faut partir du fait que, lorsque des données circulent sur l'Internet, les bits n'arrivent pas toujours comme ils sont partis. Contrairement à ce qu'essaie de faire croire le discours commercial sur « le virtuel » ou « le nuage », l'Internet repose sur le monde physique, et sur la physique. Un bit, ce n'est qu'une information stockée dans un dispositif physique. Un rayon cosmique passe, un composant électronique défaille et paf, un zéro devient un ou le contraire. Notez aussi que le problème n'est pas uniquement sur le réseau : les équipements intermédiaires (les routeurs) et terminaux n'ont pas des mémoires parfaites (on n'utilise pas toujours ECC). Dans un monde physique, la perfection n'existe pas : il y aura toujours de la corruption de données. (J'ai déjà parlé de ce problème dans mon article sur le "*bitsquatting*" <https://www.bortzmeyer.org/bitsquatting.html>.)

Le problème s'aggrave évidemment avec la taille des données et il est donc plus aigu aujourd'hui. Si vous transférez un film en haute définition (mettons un fichier de 60 Go), et qu'un bit a une probabilité de $10\hat{\}$-12 (une chance sur mille milliards, les réseaux réels sont souvent bien moins fiables) d'être modifié, vous avez une chance - ou plutôt une malchance - sur deux que le fichier arrive corrompu.

Est-ce grave ? Dans le cas de ce film, il est probable que la modidication d'un seul bit ne soit pas détectable par le spectateur. De même que le changement d'un bit dans la phrase précédant celle-ci vous a sans doute échappé (merci à Brian Carpenter pour l'idée), d'autant plus qu'elle affectait un mot assez long pour qu'il reste compréhensible. Mais les couches basses de la communication ne savent pas ce qui est vital ou pas. Elles doivent donc essayer d'assurer une fiabilité, sinon totale (ce serait utopique), du moins raisonnable en pratique.

C'est pour cela que dans la plupart des couches, il y a des sommes de contrôle qui résument les données en une somme : si elle ne correspond pas entre le départ et l'arrivée, les données sont jetées et

on recommence. Ethernet a une telle somme de contrôle, et, dans le domaine de l'IETF, TCP (RFC 793 [1], section 3.1) et UDP (RFC 768) l'ont aussi. C'est le même algorithme, la « somme de contrôle de l'Internet », décrit dans le RFC 1071. (IPv4 a aussi une somme de contrôle, supprimée en IPv6 car redondante entre celles de la couche 2 et de la couche 4.) Si un paquet TCP arrive et que la somme de contrôle ne correspond pas, le récepteur n'envoie pas d'accusé de réception, l'émetteur note que le paquet est perdu (ce qui n'est pas faux) et réémet.

Est-ce que ces sommes de contrôle suffisent ? Non. Une somme de contrôle ne peut pas détecter **tous** les problèmes, notamment parce que Shannon ne veut pas : la somme de contrôle étant plus petite que les données, plusieurs paquets ont la même somme et certains changements dans les bits d'un paquet ne modifieront donc pas la somme de contrôle. (Notez que beaucoup de calculs sur la probabilité de corruption considèrent que les changements des bits sont indépendants, ce qui n'est pas forcément le cas. En général, quand un bit est changé, ses voisins ont une probabilité plus forte de l'être également, parce qu'ils sont dans la même barrette mémoire, ou bien parce qu'ils ont été victimes du même rayon cosmique.)

Bon, donc, l'application peut recevoir des données corrompues bien qu'on apprenne en cours d'informatique que TCP fournit un « canal fiable ». Que va-t-elle faire alors ? Parfois rien, elle accepte béatement les données corrompues. Dans le cas du film cité plus haut, ce n'est pas forcément idiot. Un léger changement d'une image ne se verra pas. Dans d'autres cas, le problème est plus sérieux et certaines applications calculent leur propre somme de contrôle une fois le transfert de données terminé, parfois avec des algorithmes meilleurs que la « somme de contrôle de l'Internet », comme SHA-2. (Les algorithmes de condensation comme SHA-2 fournissent en plus d'autres propriétés de sécurité, mais ils peuvent être utilisés comme de simples sommes de contrôle.) Le problème, dans ce cas, est d'efficacité : si on détecte un problème, il faut tout recommencer et, pour des données de grande taille, cela fait beaucoup. C'est pour cela que certaines applications, comme BitTorrent, ont leur propre découpage des données, comme le fait TCP, pour éviter de devoir retransmettre la totalité d'un fichier. (On peut faire la même chose avec HTTP et ses "*range requests*", cf. RFC 7233 mais toutes les applications ne gèrent pas cela.)

Est-ce que ce problème arrive ? Combien de fois les applications acceptent des fichiers corrompus ? Et, pour les applications qui testent une somme de contrôle applicative, est-ce que la réémission est fréquente ? En fait, on ne le sait pas tellement et c'était le but originel de l'article de Craig Partridge : annoncer un projet de recherche visant à récolter des données sur ce phénomène.

Arrivé à ce stade, mes lectrices et mes lecteurs, qui sont toutes et tous de grands connaisseurs de l'Internet, pensent depuis longtemps « mais, aujourd'hui, toutes les communications sont chiffrées et, dans ce cas, c'est le mécanisme de cryptographie qui détecte la corruption ; au fur et à mesure que le chiffrement s'étend, ce problème de corruption devrait disparaitre ». D'abord, c'est plus compliqué que cela. Certains algorithmes de chiffrement ne détectent pas la corruption et, quand elle a lieu, produisent des données en clair qui sont elle-même corrompues. Le chiffrement intègre a justement pour but de résoudre ce problème, en détectant la corruption avant de perdre son temps à déchiffrer. Il est donc logique que ce chiffrement intègre soit de plus en plus répandu (il est même obligatoire en TLS 1.3, cf. RFC 8446). D'autre part, les protocoles qui utilisent le cryptographie ont souvent leur propre mécanisme de détection de la corruption, au-dessus du protocole de transport mais en dessous de celui d' application. Par contre, ces mécanismes ne sont pas forcément bien intégrés au transport. Ainsi, TLS peut détecter des erreurs que la somme de contrôle de TCP n'a pas vues mais n'a aucun moyen de demander la retransmission, il ne peut qu'avorter la communication (ou bien prévenir l'application). SSH a la même limite (ce qui est logique, tous les deux supposent une couche Transport fiable). Des études semblent indiquer

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc793.txt`

que les gros transferts de fichiers sur TLS ou SSH échouent souvent pour cette raison, et doivent être relancés au niveau applicatif. QUIC <https://www.bortzmeyer.org/quic.html> devrait améliorer les choses sur ce point. Bref, le chiffrement aide (et on peut donc espérer qu'il continuera à se répandre) mais n'est pas une solution magique.

Bon, et quelles sont les solutions envisagées ? La faiblesse de la somme de contrôle de l'Internet, qui échoue à détecter un certain nombre d'erreurs, est connue depuis longtemps, et l'augmentation de la quantité de données transférées ne fait qu'aggraver les choses. Une première solution serait de la remplacer par une somme de contrôle plus forte (mettons SHA-256 pour donner une idée, mais MD5 suffirait, il s'agit de se protéger contre des accidents, pas des attaques) mais aussi plus lente à calculer (mais la vitesse des processeurs a augmenté plus vite que celle des réseaux). Un problème amusant est « comment déployer cette nouvelle somme de contrôle » dans l'Internet ossifié et pourri de "*middleboxes*" que nous avons aujourd'hui. En théorie, le protocole de transport est de bout en bout et les équipements intermédiaires ne devraient pas poser de problèmes, ils passent juste les paquets sans les comprendre. En pratique, ce n'est pas le cas. Le RFC 1146 fournit un mécanisme de changement de la somme de contrôle standard mais il est certain qu'il n'est pas massivement déployé et, si on essayait aujourd'hui, on peut être sûr que la majorité des "*middleboxes*", intrusives et programmées avec les pieds, réagirait mal. Et UDP ? C'est encore pire, il n'a pas de mécanisme standard pour changer l'algorithme de somme de contrôle (un travail à ce sujet est en cours à l'IETF <https://datatracker.ietf.org/doc/draft-ietf-tsvwg-udp-options/>).

Bref, un remplacement de TCP par QUIC <https://www.bortzmeyer.org/quic.html> serait peut-être la meilleure solution…

Quelques lectures sur ce sujet, rassemblées par Craig Partridge :

— L'article classique, de 2000 : « "*When the CRC and TCP checksum disagree*" <http://conferences.sigcomm.org/sigcomm/2000/conf/paper/sigcomm2000-9-1.pdf> ».
— Un article inquiétant de 2018 montrant l'importance de la corruption de données dans le monde réel, « "*Cross-geography scientific data transferring trends and behavior*" <https://www.mcs.anl.gov/~zcliu/file/hpdc2018.pdf> ».
— Un autre article, de 2017 (écrit dans le contexte de l'ICN, cf. RFC 7476) montre également beaucoup trop de problèmes, « "*Request Aggregation, Caching, and Forwarding Strategies for Improving Large Climate Data Distribution with NDN : A Case Study*" <https://conferences.sigcomm.org/acm-icn/2017/proceedings/icn17-3.pdf> ».
— Et une dernière étude, « "*Transferring a petabyte in a day*" <https://www.mcs.anl.gov/~kettimut/publications/fgcs18-1pb.pdf> ».