RFC 8961 : Requirements for Time-Based Loss Detection

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 24 novembre 2020

Date de publication du RFC : Novembre 2020

https://www.bortzmeyer.org/8961.html

Vous savez certainement que l'Internet fait circuler les données sous forme de paquets indépendants, et qu'un paquet peut toujours être perdu, par exemple parce qu'un rayon cosmique agressif est passé à ce moment-là ou, moins spectaculaire, parce que les files d'un routeur étaient pleines et qu'il a dû se résigner à laisser tomber le paquet. Des protocoles existent donc pour gérer ces pertes de paquets, ce qui implique de les détecter. Et comment sait-on qu'un paquet a été perdu ? C'est plus complexe que ça n'en a l'air, et ce RFC tente d'établir un cadre générique pour la détection de pertes.

Ne tournons pas autour du pot : la seule façon fiable de savoir si un paquet a été perdu, c'est d'attendre qu'il arrive (ce que le RFC nomme, dans son titre, "*time-based loss detection*") et, s'il n'arrive pas, de le déclarer perdu. Plus précisément, pour l'émetteur (car le récepteur ne sait pas forcément qu'on lui a envoyé un paquet), on émet un paquet et on attend une confirmation qu'il a été reçu (avec TCP, cette confirmation sera le `ACK`, avec le DNS sur UDP, ce sera la réponse DNS). Si la confirmation n'a pas été reçue, s'il y a "*timeout*", c'est qu'un paquet (la demande, ou bien l'accusé de réception) n'est pas arrivé. Mais attendre combien de temps ? Si on attend peu de temps (mettons 100 millisecondes), on risque de considérer le paquet comme perdu, alors que le voyage était simplement un peu long (en 100 ms, vous ne pouvez même pas faire un aller-retour entre la France et les Philippines, ne serait-ce qu'à cause de la limite de la vitesse de la lumière). Et si on attend longtemps (mettons 5 secondes), on ne pourra pas réagir rapidement aux pertes, et la latence <https://www.bortzmeyer.org/latence.html> perçue par l'utilisateur sera insupportable (la sensation de « vitesse » pour l'utilisateur dépend davantage de la latence que de la capacité <https://www.bortzmeyer.org/capacite.html>). Il faut donc faire un compromis entre réactivité et justesse.

Il existe d'autres méthodes que l'attente pour détecter des pertes, par exemple TCP et SCTP utilisent aussi les accusés de réception sélectifs (RFC 2018 [1], RFC 9260 dans sa section 3.3.4 et RFC 6675) mais

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc2018.txt`

aucune de ces autres méthodes ne détecte toutes les pertes, et la détection par absence de réponse reste donc indispensable.

L'Internet, vous le savez, est un ensemble compliqué de réseaux, reliés par des câbles très variés (et quelques liens radio), avec des débits bien différents et changeant d'un moment à l'autre. Tout chemin d'une machine à l'autre va avoir un certain nombre de propriétés (qui varient dans le temps) telles que la latence ou la capacité. Et le taux de perte de paquets, qui nous intéresse ici. (Voir aussi le RFC 7680.)

Notre RFC suppose que la perte de paquets est une indication de congestion (RFC 5681). Ce n'est pas vrai à 100 %, surtout sur les liens radio, où des paquets peuvent être détruits par des perturbations électro-magnétiques sans qu'il y ait congestion, mais c'est quand même proche de la vérité.

Et, au fait, pourquoi détecter la perte de paquets ? Pourquoi ne pas tout simplement ignorer le problème ? Deux raisons :

— Pour pouvoir envoyer des données de manière fiable, il faut détecter les paquets manquants, afin de pouvoir demander leur retransmission. C'est ce que fait TCP, par exemple, sans quoi on ne pourrait pas transmettre un fichier en étant sûr qu'il arrive complet.
— Puisque la perte de paquets signale en général qu'il y a congestion, détecter cette perte permet de ralentir l'envoi de données et donc de lutter contre la congestion.

Résultat, beaucoup de protocoles ont un mécanisme de détection de pertes : TCP (RFC 6298), bien sûr, mais aussi SCTP (RFC 9260), SIP (RFC 3261), etc.

Le RFC cite souvent l'article de Allman, M. et Paxson V, « "*On Estimating End-to-End Network Path Properties*" <https://ntrs.nasa.gov/api/citations/20000004338/downloads/20000004338.pdf> » donc vous avez le droit d'interrompre votre lecture ici pour lire cet article avant de continuer.

Reprenons, avec la section 2 du RFC, qui explique les buts et non-buts de ce RFC :

— Ce RFC ne change aucun protocole existant, les RFC restent les mêmes, vous n'avez pas à réapprendre TCP,
— Ce RFC vise surtout les RFC futurs, qui auraient intérêt à se conformer aux principes énoncés ici (c'est par exemple le cas si vous développez un nouveau protocole au-dessus d'UDP et que vous devez donc mettre en œuvre la détection de pertes),
— Ce RFC n'impose pas des règles absolues, il donne des principes qui marchent dans la plupart des cas, c'est tout.

Nous arrivons maintenant à la section 4 du RFC, qui liste les exigences auxquelles doivent obéir les mécanismes de détection de pertes. (En pratique, les mécanismes existants collent déjà à ces exigences mais elles n'avaient pas été formalisées.) Un petit retour sur la notion de latence, d'abord. On veut savoir combien de temps attendre avant de déclarer un paquet perdu. Cette durée se nomme RTO ("*Retransmission TimeOut*"). Les latences dans l'Internet étant extrêmement variables, il serait intéressant de faire dépendre le RTO de la latence. Mais quelle latence ? Le temps d'aller-retour entre deux machines est le temps qu'il faut à un paquet IP pour aller de la machine A à la machine B plus le temps qu'il faut à un paquet IP pour aller de B à A. On ne peut pas mesurer directement ce temps, car le temps de traitement dans la machine B n'est pas forcément connu. Dans le cas de l'ICMP Echo utilisé par ping, ce temps est considéré comme négligeable, ce qui est assez correct si l'amer <https://www.bortzmeyer.org/amer-mire.html> est une machine Unix dont le noyau traite l'ICMP Echo. Cela l'est moins si l'amer est un routeur qui traite les réponses ICMP à sa plus basse priorité. Et cela l'est encore moins si un client DNS essaie d'estimer la latence vers un résolveur <https://www.bortzmeyer.org/resolveur-dns.html>. Si le résolveur avait la réponse dans sa mémoire, le temps de traitement est faible. S'il devait demander aux serveurs faisant autorité <https://www.bortzmeyer.org/serveur-dns-faisa
html>, ce temps peut être bien supérieur à la latence. Le RFC distingue donc RTT ("*Round-Trip Time*"), la vraie latence, et FT ("*Feedback Time*") qui est ce qu'affichent ping, dig et autres outils. La machine qui veut savoir combien de temps attendre une réaction de la machine en face, avant de déclarer qu'un paquet est perdu, a tout intérêt à avoir une idée du FT.

Certaines des exigences du RFC sont quantitatives. Ainsi, tant qu'on n'a pas mesuré le FT (au début de la session), le RFC requiert que le RTO soit d'au moins une seconde, pour ne pas surcharger le réseau avec des réémissions, et parce que la perte étant interprétée comme un signe de congestion, un RTO trop faible amènerait à réduire la quantité de données qu'on peut envoyer, diminuant ainsi la capacité effective. Sans compter le problème de l'ambiguïté. Si on a réémis un paquet, et qu'une réponse revient, était-elle pour le paquet initial ou pour la réémission ? Dans le doute, il ne faut pas utiliser le temps mesuré pour changer son estimation du RTO. Et c'est une des raisons pour lesquelles il faut prendre son temps pour les premières mesures. Ah, au fait, pourquoi une seconde et pas 0,75 ou 1,25 ? Cela vient d'une analyse quantitative des RTT typiques de l'Internet, exposée dans l'annexe A du RFC 6298.

Après plusieurs mesures, on connait mieux le FT et on peut abaisser le RTO. Il n'y a pas de durée minimale, donc on peut s'approcher de zéro tant qu'on veut.

Le RFC dit aussi que le RTO ("*Retransmission TimeOut*", le délai d'attente) doit se mesurer à partir de **plusieurs** observations du FT, pour éviter une mesure faussée par un cas extrême. Et il faut refaire ces observations souvent car le réseau change ; les vieilles mesures n'ont pas d'intérêt. C'est ce que fait TCP avec son "*smoothed RTT*", utilisant la moyenne mobile exponentielle (RFC 6298).

L'Internet étant ce qu'il est, il est recommandé de s'assurer qu'un méchant ne puisse pas facilement fausser cette mesure en injectant des faux paquets. Pour TCP, cette assurance est donnée par le caractère imprévisible du numéro de séquence initial, si l'attaquant n'est pas sur le chemin (RFC 5961). Si on veut une protection contre un attaquant situé sur le chemin, il faut de la cryptographie.

Le RFC recommande de refaire l'évaluation du RTO au moins une fois par RTT et, de préference, aussi souvent que des données sont échangées. TCP le fait une fois par RTT et, si on utilise le RFC 7323, à chaque accusé de réception.

Le RFC rappelle qu'en l'absence d'indication du contraire, une perte de paquets doit être considérée comme un indicateur de congestion, et qu'il faut donc ralentir (RFC 5681, pour le cas de TCP).

Et, dernière exigence, en cas de perte de paquets, le RTO doit croitre exponentiellement, pour s'ajuster rapidement à la charge du réseau. On pourra réduire le RTO lorsqu'on aura la preuve que les paquets passent et qu'on reçoit des accusés de réception et/ou des réponses. Dans tous les cas, le RFC limite le RTO à 60 secondes, ce qui est déjà énorme (je n'ai jamais vu une réponse revenir après une durée aussi longue).

Enfin, la section 5 discute des exigences posées et de leurs limites. La tension entre le désir de réactivité (un RTO faible) et celui de mesures correctes (un RTO plus important, pour être raisonnablement sûr de ne pas conclure à tort qu'il y a eu une perte) est une tension fondamentale : on n'aura jamais de solution parfaite, juste des compromis. Il existe des techniques qui permettent de mieux détecter les pertes (l'algorithme Eifel - RFC 3522, le F-RTO du RFC 5682, DSACK - RFC 2883 et RFC 3708…) mais elles ne sont pas complètes et l'attente bornée par le RTO reste donc nécessaire en dernier recours.

Le noyau Linux a mis en œuvre plusieurs techniques non normalisées (par exemple des modifications du RFC 6298) sans que cela ne crée apparemment de problèmes. D'une manière générale, les algorithmes de détection de pertes de TCP, SCTP ou QUIC sont largement compatibles avec les exigences de ce RFC.