RFC 7567 : IETF Recommendations Regarding Active Queue Management

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 12 juillet 2015

Date de publication du RFC : Juillet 2015

https://www.bortzmeyer.org/7567.html

L'AQM ("*Active Queue Management*") désigne l'ensemble des techniques utilisées par un routeur pour gérer les files d'attente de paquets attendant de partir. La technique triviale est de faire passer les paquets en FIFO et de refuser de nouveaux paquets quand la file d'attente est pleine. Mais il existe plein d'autres techniques possibles, qui forment l'AQM. Ce nouveau RFC résume les recommandations de l'IETF sur l'AQM. Notamment, il n'y a plus de méthode privilégiée, comme l'était RED précédemment (dans l'ancien RFC 2309 [1]).

Petit rappel sur IP d'abord (section 1 du RFC) : IP (RFC 791 ou RFC 2460) est un protocole sans état (chaque paquet est routé indépendamment des autres) et donc sans connexion. Cela rend les routeurs bien plus efficaces (ils n'ont pas besoin d'avoir de mémoire), cela permet de changer de route facilement, cela permet de survivre à la perte ou au redémarrage d'un routeur. L'extrême robustesse de l'Internet le montre tous les jours. En revanche, une des faiblesses de ce mécanisme sans connexion est sa sensibilité aux fortes charges : si un routeur congestionné commence à perdre des paquets car ses liens de sortie sont saturés, les machines vont réémettre, envoyant davantage de paquets, aggravant la situation et ainsi de suite jusqu'à la fusion du réseau (le fameux "*Internet meltdown*", cf. RFC 896 et RFC 970). Ce phénomène a été plusieurs fois observés dans les années 1980 sur le cœur de l'Internet. Il est depuis confiné aux marges de l'Internet, par l'applications de meilleures techniques dans les routeurs, mais il menace en permanence de réapparaitre.

Le RFC 2309, publié en 1998, est le dernier qui fasse le point sur le problème de la congestion et ses conséquences. Sa principale recommandation concrète était de déployer massivement une technique de gestion des files d'attente, RED (« "*Internet routers should implement some active queue management [...] The*

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc2309.txt

default mechanism for managing queue lengths to meet these goals in FIFO queues is Random Early Detection (RED). Unless a developer has reasons to provide another equivalent mechanism, we recommend that RED be used." ») Mais, depuis 1998, l'Internet a évolué, notamment par la demande plus forte de limitation de la latence <https://www.bortzmeyer.org/latence.html>. La recommandation du RFC 2309 ne semble plus aussi opportune. (Et RED s'est avéré difficile à configurer en pratique.)

Tout le travail n'est pas à faire du côté des routeurs. Le problème de la congestion et le risque de fusion sont tellement sérieux, surtout depuis les grandes pannes des années 1980, que beaucoup d'efforts ont été dépensés pour les traiter. C'est ainsi que Van Jacobson a développé les règles (« "*Congestion Avoidance and Control*" <http://ee.lbl.gov/papers/congavoid.pdf> » dans le SIGCOMM "*Symposium proceedings on Communications architectures and protocols*" en 1988) que doit suivre TCP pour éviter que ce dernier n'aggrave la congestion : en cas de perte de paquets, TCP ne doit pas réémettre bêtement comme un porc mais au contraire se calmer, arrêter d'envoyer des paquets à la vitesse maximale, pour donner une chance aux files d'attente des routeurs de se vider. Ces règles sont aujourd'hui documentées dans le RFC 5681. Elles s'appliquent aux machines terminales de l'Internet, alors que notre RFC 7567 concerne les routeurs, qui ont aussi leur rôle à jouer. Les machines terminales ne peuvent pas tout, d'autant plus que toutes ne sont pas bien élevées : il y a des programmes mal écrits ou malveillants qui sont connectés au réseau, cf. la section 5. Des travaux sont en cours pour gérer le cas de machines qui ne réagissent pas à la congestion et continuent à émettre au maximum (cf. RFC 6789).

Mais le risque d'écroulement du réseau sous l'effet de la congestion n'est pas le seul. Il faut aussi penser à la latence <https://www.bortzmeyer.org/latence.html>. Si on augmente la taille des files d'attente dans les routeurs, pour diminuer le risque d'avoir à jeter des paquets, on augmente aussi la latence, car vider ces files prendra du temps. (L'augmentation excessive de la taille des files d'attentes est connue sous le nom de "*bufferbloat*", voir aussi la section 2.3, et le RFC 8033.)

La section 2 décrit en détail le problème de la gestion des files d'attente dans un routeur de l'Internet. La méthode traditionnelle, la plus simple, est connue sous le nom de "*tail drop*" : la queue a une taille finie, lorsqu'elle est pleine, on arrête d'accepter des paquets, point. Cela veut dire que les premiers paquets jetés sont les derniers arrivés. Cette méthode est simple à programmer mais elle a plusieurs inconvénients :

— "*Tail drop*" ne signale la congestion aux machines terminales (en jetant les paquets) que lorsque la file d'attente est pleine. Si la file reste en permanence à 50-80 % pleine, aucun paquet ne sera jeté, les machines terminales continueront à émettre au même rythme, alors que le fait que la file soit plus qu'à moité pleine aura un effet négatif sur la latence.
— Cette technique ne garantit pas l'égalité des différents flots de communication : dans certains cas, un seul flot peut monopoliser la queue.
— Elle a aussi la propriété qu'elle tend à synchroniser les différentes machines qui partagent le même goulet d'étranglement (Floyd, S. et V. Jacobsen, « "*The Synchronization of Periodic Routing Messages*" <http://ee.lbl.gov/papers/sync_94.pdf> »).

On pourrait réduire certains de ces problèmes (notamment le premier) en réduisant la taille de la file d'attente. Mais celle-ci est indispensable dans le cas d'augmentation brusque de trafic : le rythme d'arrivée des paquets n'est pas constant, il y a des moments où tout le monde parle en même temps (Leland, W., Taqqu, M., Willinger, W., et D. Wilson, « "*On the Self-Similar Nature of Ethernet Traffic (Extended Version)*" <http://eeweb.poly.edu/el933/papers/Willinger.pdf> », "*IEEE/ACM Transactions on Networking*"), et les files d'attente sont là pour lisser ces soubresauts. Idéalement, on voudrait une queue vide en temps normal, qui ne se remplisse que pendant les pics de trafic. L'algorithme "*tail drop*" a l'inconvénient de créer souvent des queues pleines en permanence, au détriment de la latence.

Mais il y a d'autres algorithmes. Le "*random drop on full*" jette, lorsque la file est pleine, non pas le dernier paquet mais un paquet au hasard. Cela évite la monopolisation de la file par un seul flot de données mais cela ne résout pas le problème des files toujours pleines. Même chose pour le "*head drop*" qui consiste à jeter le premier paquet et non pas le dernier. La bonne solution est donc de ne **pas** attendre

que la file soit pleine pour agir. C'est cette idée qui est à la base de l'AQM. En jetant des paquets avant que la file ne soit remplie (ou bien en les marquant avec ECN, cf. RFC 3168), on va dire indirectement aux machines terminales de ralentir, et on évitera ainsi les files complètement pleines, sauf en cas de brusque pic de trafic.

AQM est donc un concept **proactif**. Il permet de :

— Jeter moins de paquets,
— Réduire l'occupation des files d'attente, et donc la latence, pour le plus grand plaisir des applications interactives,
— Éviter la monopolisation par un seul flot,
— Réduire la synchronisation (par l'usage de choix aléatoires).

Tout cela est très bien si les applications utilisent toutes TCP, protocole habitué à réagir à la congestion (en diminuant son débit), et qui protège l'Internet contre les abus. Tant qu'il n'y a que TCP, avec des algorithmes conformes aux recommandations du RFC 5681, tout le monde partage le réseau en bonne intelligence et sans catastrophe, et chacun obtient une part égale de la capacité <https://www.bortzmeyer.org/capacite.html>.

Mais il n'y a pas que TCP dans l'Internet, certains applications utilisent, par exemple, UDP (conseil personnel au passage : le programmeur débutant qui ne connait pas bien les problèmes de congestion devrait n'utiliser que TCP ou un équivalent comme SCTP, afin d'éviter d'écrouler le réseau). On peut classer les flots de données non-TCP en trois catégories :

— Flots amicaux avec TCP : ce sont ceux qui utilisent des algorithmes de contrôle de la congestion qui, en pratique, leur donnent un débit proche de celui d'un flot TCP (RFC 5348). Par exemple, ce sont des flots UDP envoyés par des applications dont les auteurs ont lu et appliqué le RFC 8085.
— Flots qui ne réagissent pas à la congestion : ce sont ceux qui continuent à envoyer des paquets sans tenir compte des pertes ou des signalements ECN. Ce sont les « méchants ». Certaines applications de transport de la voix ou de la vidéo sont dans cette catégorie. Jusqu'à présent, la solution anti-congestion à l'IETF avait toujours été d'améliorer les applications. Comme on ne peut pas espérer que 100 % des applications soient bien élevées, il faudra un jour développer des solutions pour gérer ceux qui abusent.
— Flots qui réagissent, mais pas aussi bien que TCP : ils ont un mécanisme de réaction à la congestion mais ce mécanisme est trop peu réactif, et ces flots prennent donc une part disproportionnée du trafic. Cela peut être le résultat d'une maladresse du programmeur (voir mon conseil plus haut : utilisez TCP, sauf si vous êtes vraiment très fort en contrôle de congestion). Mais cela peut être aussi délibéré, pour s'assurer une plus grosse part du gâteau. Par exemple, certaines mises en œuvre de TCP étaient plus agressives que la normale, peut-être pour que le vendeur puisse proclamer que son TCP était « plus rapide ». On est dans une problématique très proche de celle de nombreux problèmes écologiques : si peu de gens trichent, les tricheurs et les égoïstes ont un gros avantage. Mais s'ils entrainent les autres, une spirale de la triche se développe, jusqu'au point où c'est l'enfer pour tout le monde. Pour l'Internet, une telle spirale pourrait aller jusqu'au cas où il n'y aurait plus de réaction à la congestion du tout, chacun poussant ses octets au maximum, sans tenir compte de l'intérêt collectif.

La section 4 de notre RFC résume les recommandations actuelles :

— Il faut faire de l'AQM,
— Ce serait bien de ne pas seulement jeter les paquets mais aussi de pouvoir faire de l'ECN,
— L'administrateur système ou réseaux ne devrait pas avoir à faire de réglages manuels ("*tuning*"), tout devrait s'ajuster automatiquement,
— L'AQM doit répondre à la congestion effective, pas à ce que le routeur croit savoir sur l'application ou sur le protocole de transport (c'est également impératif pour la neutralité de l'Internet),
— Le problème des flots délibérement égoïstes est très important, et nécessite davantage de recherche.

Le reste de la section 4 détaille chacune de ces recommandations.

Par exemple, la seconde vient du fait qu'il n'y avait au début qu'un seul moyen pour un routeur de signaler la congestion : jeter des paquets (ce qui était de toute façon nécessaire : quand il n'y a plus de place, il n'y a plus de place). L'ajout d'ECN, dans le RFC 3168, et les spécifications de son usage (voir par exemple le RFC 6679) ont ajouté un deuxième moyen, qui permet de réagir avant qu'on perde des paquets. (Le RFC note que retarder les paquets, ce que fait la file d'attente, peut aussi être vu comme un signal : l'augmentation du RTT va mener TCP à ajuster son débit.)

La recommandation comme quoi le "*tuning*" (par le biais de paramètres numériques qu'il faudrait ajuster pour obtenir l'effet idéal) ne doit **pas** être obligatoire est discutée en section 4.3. Elle vient de l'expérience ce certaines mises en œuvre d'AQM où le pauvre administrateur du routeur devait absolument fixer certains paramètres, alors qu'il manque de temps et qu'il n'est pas forcément expert en congestion. En environnement de production, il n'est pas réaliste d'espérer que le dit administrateur passe ses nuits à faire du "*tuning*" ; l'algorithme doit avoir des paramètres par défaut raisonnables et, pour le reste, doit s'ajuster tout seul, en tenant compte d'informations comme la capacité des interfaces du routeur, et de variables mesurées en cours de route comme la durée moyenne de séjour des paquets dans la file d'attente ou le pourcentage de paquets jetés. (Un des problèmes de RED est justement la difficulté à s'ajuster automatiquement.)

Cela n'interdit pas au programmeur de fournir des mécanismes permettant le "*tuning*" manuel (ainsi que des outils pour aider l'administrateur à prendre ses décisions) mais ces mécanismes ne doivent pas être indispensables.

Pour la recommandation d'objectivité (décider en fonction de ce qu'on observe, pas de ce qu'on suppose), il est utile de relire le RFC 7141. Par exemple, il ne faut pas tenir compte de la taille des paquets. De même, il ne faut pas supposer que toutes les applications utilisent TCP.

Enfin, sur la recommendation de continuer les recherches, la section 4.7 fournit un programme aux chercheurs qui se demandent sur quel sujet travailler : il reste encore beaucoup de travail.

La section 1.4 de notre RFC décrit les changements depuis le RFC 2309. Les deux plus importants sont :

— La taille d'une file d'attente n'est plus forcément évaluée uniquement en octets. Elle peut l'être en temps passé par le paquet dans la queue.
— L'algorithme RED n'est plus le seul recommandé pour gérer les files d'attente.