RFC 9743 : Specifying New Congestion Control Algorithms

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 14 mars 2025

Date de publication du RFC : Mars 2025

https://www.bortzmeyer.org/9743.html

La lutte contre la congestion du réseau est une tâche permanente des algorithmes utilisés dans l'Internet. Si les émetteurs de données envoyaient des octets aussi vite qu'ils le pouvaient, sans jamais réfléchir, l'Internet s'écroulerait vite. Des protocoles comme TCP ou QUIC contiennent donc des algorithmes de contrôle de la congestion. Il y en a plusieurs, certains peuvent être néfastes, et ce RFC, qui remplace le RFC 5033 [1], explique comment de nouveaux algorithmes doivent être évalués avant d'être déployés en vrai. Bref, la prudence est nécessaire.

Le RFC 2914 expose les principes généraux du contrôle de congestion du réseau. Notre RFC 9743 a un autre but : créer un cadre d'évaluation des algorithmes de contrôle de la congestion. Car c'est très difficile de concevoir et de spécifier un tel algorithme. Songez qu'il doit fonctionner sur des types de liens très différents, des réseaux à énorme capacité <https://www.bortzmeyer.org/capacite.html> sur les courtes distance d'un centre de données (RFC 8257) à des liaisons radio à grande distance et avec un fort taux de perte de paquets. Il doit fonctionner en Wifi, sur la fibre, sur des liens surchargés, bien exploiter des liens à forte capacité <https://www.bortzmeyer.org/capacite.html>, supporter des liens à forte latence <https://www.bortzmeyer.org/latence.html>, etc. Et l'efficacité de ces algorithmes dépend aussi de l'application qui va les utiliser, la vidéoconférence ne produisant pas les mêmes effets que le transfert de fichiers, le visionnage de vidéos haute définition, les connexions interactives type SSH, l'accès à des sites Web, etc).

La plupart des protocoles de transport normalisés par l'IETF ont du contrôle de congestion. Quand le RFC 5033 a été publié, cela concernait surtout TCP (RFC 9293), avec des "*outsiders*" comme DCCP (RFC 4340) ou SCTP (RFC 9260). Mais il y a aussi désormais QUIC (RFC 9000) et RMCAT (RFC 8836).

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc5033.txt

Parmi les algorithmes de contrôle de la congestion développés pour l'Internet, on peut citer CUBIC RFC 9438, Reno RFC 5681, BBR (pas encore de RFC mais il y a un brouillon, `draft-ietf-ccwg-bbr`), et autres.

Le RFC 5033 ne dit pas clairement si on a « le droit » de déployer un nouvel algorithme de contrôle de la congestion sans l'avoir normalisé dans un RFC. En tout cas, en pratique, comme le montre le déploiement de CUBIC et BBR, on n'attend pas le RFC (l'Internet est sans permission), on publie le code, sous une licence libre dans le cas ci-dessus (CUBIC a été surtout porté par sa mise en œuvre dans Linux) et on y va. Pourtant, avoir une spécification écrite et disponible, revue par l'IETF, serait certainement bénéficiaire pour tout le monde, permettant un examen large du nouvel algorithme et de certaines de ses conséquences imprévues.

(Le RFC cite un très bizarre argument comme quoi, dans certaines entreprises, les développeurs ont le droit de lire les RFC mais pas le code source publié sous une licence libre. Je ne suis pas juriste mais cela ressemble à un disjonctage sérieux de la part du juriste qui a prétendu cela.)

Notre RFC 9743 est donc un guide pour les gens qui vont faire cette très souhaitable évaluation des algorithmes de lutte contre la congestion, dans le prolongement du RFC 2914.

Donc, place au guide, en section 3, le centre de ce RFC. Premier objectif, qu'il existe plusieurs mises en œuvre de l'algorithme, et sous une licence libre. Ce n'est pas une obligation (surtout si le RFC décrivant l'algorithme a le statut « Expérimental ») mais c'est souhaité.

En parlant d'algorithmes expérimentaux, le RFC rappelle que la spécification d'un tel algorithme doit indiquer quels résultats on attend de l'expérience, et ce qu'il faudrait pour progresser. La section 12 du RFC 6928 et la section 4 du RFC 4614 sont de bons exemples en ce sens.

Un algorithme expérimental ne doit pas être activé par défaut et doit être désactivable s'il a des comportements désagréables. Mais il faut quand même le tester sinon il ne progressera jamais et ce test peut, au début, être fait dans un simulateur (RFC 8869) mais il doit, à un moment, être fait en vrai grandeur, dans l'Internet réel. Le RFC ajoute donc que tout déploiement ne doit être fait qu'accompagné de mesures qui permettront d'obtenir des informations.

Comme résultat de l'évaluation, tout RFC décrivant un algorithme de contrôle de congestion doit indiquer explicitement, dès le début, s'il est sûr et peut être déployé massivement sans précautions particulières. (Un exemple d'expérimentation figure dans le RFC 3649.)

Une exception est faite par notre RFC sur les « environnements contrôlés » (section 4). Il s'agit de réseaux qui sont sous le contrôle d'une seule organisation, qui maitrise tout ce qui s'y passe, sans conséquences pour des tiers (RFC 8799), et peuvent donc utiliser des algorithmes de contrôle de la congestion qu'on ne voudrait pas voir déployés dans l'Internet public. Un exemple typique d'un tel « environnement contrôlé » est un centre de données privé, avec souvent des commutateurs qui signalent aux machines terminales l'état du réseau, leur permettant d'ajuster leur débit (sur les centres de données, voir aussi la section 7.11).

Une fois passé ce cas particulier, allons voir la section 5, qui est le cœur de ce RFC ; elle liste les critères d'évaluation d'un algorithme de contrôle de la congestion. D'abord, dans le cas relativement simple où plusieurs flux de données sont en compétition sur un même lien mais que tous utilisent l'algorithme évalué. Est-ce que chaque flux a bien sa juste part (qui est une part égale aux autres, lorsque les flux veulent transmettre le plus possible) ? En outre, est-ce que l'algorithme évite l'écroulement qui

se produit lorsque, en raison de pertes de paquets, les émetteurs ré-émettent massivement, aggravant le problème ? Pour cela, il faut regarder si l'algorithme ralentit en cas de perte de paquets, voire arrête d'émettre lorsque le taux de pertes devient trop important (cf. RFC 3714). Lire aussi les RFC 2914 et RFC 8961, qui posent les principes de cette réaction à la perte de paquets. Ce critère d'évaluation n'impose pas que le mécanisme d'évitement de la congestion soit identique à celui de TCP.

Une façon d'éviter de perdre les paquets est de les mettre dans une file d'attente, en attendant qu'ils puissent être transmis. Avec l'augmentation de la taille des mémoires, on pourrait penser que cette solution est de plus en plus bénéfique. Mais elle mène à de longues files d'attente, où les paquets séjourneront de plus en plus longtemps, au détriment de la latence <https://www.bortzmeyer.org/latence.html>, un phénomène connu sous le nom de "*bufferbloat*". Le RFC demande donc qu'on évalue les mécanismes de contrôle de la congestion en regardant comment ils évitent ces longues files d'attente (RFC 8961 et RFC 8085, notez que Reno et CUBIC ne font rien pour éviter le "*bufferbloat*").

Il est bien sûr crucial d'éviter les pertes de paquets ; même si les protocoles comme TCP savent les rattraper (en demandant une réémission des paquets manquants), les performances en souffrent, et c'est du gaspillage que d'envoyer des paquets qui seront perdus en route. Un algorithme comme la première version de BBR (dans `draft-cardwell-iccrg-bbr-congestion-control-00`) avait ce défaut <https://ieeexplore.ieee.org/document/8117540> de mener à des pertes de paquets. Mais cette exigence laisse quand même pas mal de marge de manœuvre aux protocoles de contrôle de la congestion.

Autre exigence importante, l'équité (ou la justice). On la décrit souvent en disant que tous les flux doivent avoir le même débit. Mais c'est une vision trop simple : certains flux n'ont pas grand'chose à envoyer et il est alors normal que leur débit soit plus faible. L'équité se mesure entre des flux qui sont similaires.

En parlant de flux, il faut noter que, comme les algorithmes de contrôle de congestion n'atteignent leur état stable qu'au bout d'un certain temps, les flux courts, qui peuvent représenter une bonne partie du trafic (songez à une connexion TCP créée pour une seule requête HTTP d'une page de petite taille) n'auront jamais leur débit théorique. Les auteurs d'un protocole de contrôle de la congestion doivent donc étudier ce qui se passe lors de flux courts, qui n'atteignent pas le débit asymptotique.

La question de l'équité ne concerne pas que les flux utilisant tous le même algorithme de contrôle de la congestion. Il faut aussi penser au cas où plusieurs flux se partagent la capacité du réseau, tout en utilisant des algorithmes différents (section 5.2). Le principe, lorsqu'on envisage un nouveau protocole, est que les flux qui l'utilisent ne doivent pas obtenir une part de la capacité plus grande que les flux utilisant les autres protocoles (RFC 5681, RFC 9002 et RFC 9438). Les algorithmes du nouveau protocole doivent être étudiés en s'assurant qu'ils ne vont pas dégrader la situation, et ne pas supposer que le protocole sera le seul, il devra au contraire coexister avec les autres protocoles. (Dans un réseau décentralisé comme l'Internet, on ne peut pas espérer que tout le monde utilise le même protocole de contrôle de la congestion.)

Pour compliquer la chose, il y a aussi des algorithmes qui visent à satisfaire les besoins spécifiques des applications « temps réel ». Leur débit attendu est faible mais ils sont par contre très exigeants sur la latence maximale (cf. RFC 8836). La question est étudiée dans les RFC 8868 et RFC 8867 ainsi que, pour le cas spécifique de RTP, dans le RFC 9392. Si on prétend déployer un nouveau protocole de contrôle de la congestion, il doit coexister harmonieusement avec ces protocoles temps réel. La tâche est d'autant plus difficile, note notre RFC, que ces protocoles sont souvent peu ou mal documentés.

Bon, quelques autres trucs à garder en tête. D'abord, il faut évidemment que tout nouveau protocole soit déployable de manière incrémentale. Pas question d'un "*flag day*" où tout l'Internet changerait de

mécanisme de contrôle de la congestion du jour au lendemain (section 5.3.2). Si le protocole est conçu pour des environnements très spécifiques (comme l'intérieur d'un centre de données privé), il peut se permettre d'être plus radical dans ses choix mais le RFC insiste sur l'importance, dans ce cas, de décrire les mesures par lesquelles on va s'assurer que le protocole n'est réellement utilisé que dans ces environnements. Un exemple de discussion sur le déploiement incrémental figure dans les sections 10.3 et 10.4 du RFC 4782.

La section 6 de notre RFC rappelle que le protocole doit être évalué dans des environnements différents qu'on peut rencontrer sur l'Internet public. Ainsi, les liaisons filaires se caractérisent par une très faible quantité de perte de paquets, sauf dans les routeurs, lorsque leur file d'attente est pleine. Et leurs caractéristiques sont stables dans le temps. En revanche, les liens sans fil ont des caractéristiques variables (par exemple en fonction d'évènements météorologiques) et des pertes de paquets en route, pas seulement dans les routeurs. Le RFC 3819 et le brouillon `draft-irtf-tmrg-tools` (dans sa section 16) discutent ce problème, que l'algorithme de contrôle de congestion doit prendre en compte.

Tout cela est bien compliqué, vous allez me dire ? Mais ce n'est pas fini, il reste les « cas spéciaux » de la section 7. D'abord, si le routeur fait de l'AQM, c'est-à-dire jette des paquets avant que la file d'attente ne soit pleine. Des exemples de tels algorithmes sont Flow Queue CoDel (RFC 8290), PIE ("*Proportional Integral Controller Enhanced*", RFC 8033) ou L4S ("*Low Latency, Low Loss, and Scalable Throughput*", RFC 9332). L'évaluation doit donc tenir compte de ces algorithmes.

Certains réseaux disposent de disjoncteurs ("*breakers*", RFC 8084), qui surveillent le trafic et coupent ou réduisent lorsque la congestion apparait. Là encore, tout nouvel algorithme de contrôle de la congestion qu'on utilise doit gérer le cas où de tels disjoncteurs sont présents.

Autre cas qui complique les choses, les réseaux à latence variable. Avec un réseau simple, la latence dans les câbles est fixe et la latence vue par les flux ne varie qu'avec l'occupation des files d'attentes du routeur. Mais un chemin à travers l'Internet n'est pas simple, et la latence peut varier dans le temps, par exemple parce que les routes changent et qu'on passe par d'autres réseaux. (Voir par exemple les conséquences d'une coupure de câble <`https://labs.ripe.net/author/emileaben/does-the-internet-`>.) Bref, le protocole de contrôle de la congestion ne doit pas supposer une latence constante. En parlant de latence, il y a aussi des liens qui ont une latence bien plus élevée que ce que qu'attendent certains protocoles simples. C'est par exemple le cas des liens satellite (RFC 2488 et RFC 3649). Et en parlant de variations, l'Internet voit souvent des brusques changements, par exemple un brusque afflux de paquets, et notre protocole devra gérer cela (section 9.2 du RFC 4782). J'avais dit que l'Internet était compliqué !

Et pour ajouter à cette complication, il y a des réseaux qui changent l'ordre des paquets. Oui, vous savez cela. Tout le monde sait qu'IP ne garantit pas l'ordre d'arrivée des paquets. Tout le monde le sait mais notre RFC juge utile de le rappeler car c'est un des défis que devra relever un protocole de contrôle de la congestion : bien fonctionner même en cas de sérieux réordonnancement des paquets, comme discuté dans le RFC 4653.

L'algorithme utilisé pour gérer la congestion ne doit pas non plus être trop subtil, car il faudra qu'il puisse être exécuté par des machines limitées (en processeur, en mémoire, etc, cf. RFC 7228).

Le cas où un flux emprunte plusieurs chemins ("*Multipath TCP*", RFC 8684, mais lisez aussi le RFC 6356) est délicat. Par exemple, on peut avoir un seul des chemins qui est congestionné (le protocole de gestion des chemins peut alors décider d'abandonner ce chemin) mais aussi plusieurs chemins qui partagent le même goulet d'étranglement.

Et, bien sûr, l'Internet est une jungle. On ne peut pas compter que toutes les machines seront gérées par des gentilles licornes arc-en-ciel bienveillantes. Tout protocole de contrôle de la congestion déployé dans l'Internet public doit donc tenir compte des méchants. (Il y a aussi, bien plus nombreux, les programmes bogués et les administrateurs système incompétents. Mais, si un protocole tient le coup en présence d'attaques, a fortiori, il résistera aux programmes non conformes.) Une lecture intéressante est le RFC 4782, dans ses sections 9.4 à 9.6.

L'annexe A résume les changements depuis le RFC 5033. Il y a notamment :
— L'ajout de QUIC,
— un texte plus précis sur le "*bufferbloat*",
— le cas des machines contraintes (pour l'IoT),
— des discussions sur l'AQM, les flux courts, le temps réel…