RFC 6077 : Open Research Issues in Internet Congestion Control

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 3 février 2011

Date de publication du RFC : Février 2011

https://www.bortzmeyer.org/6077.html

La congestion est la plaie des réseaux informatiques depuis leurs débuts. Quelle que soit la capacité de ceux-ci, les utilisations emplissent les tuyaux jusqu'à ce que ceux-ci débordent. Il y a donc depuis quarante ans, mais surtout depuis le travail de Van Jacobson en 1988 (voir son ancien mais toujours bon « "*Congestion Avoidance and Control*" <ftp://ftp.ee.lbl.gov/papers/congavoid.ps.Z> », "*Proceeding of ACM SIGCOMM'88 Symposium*", août 1988 ; attention, c'est assez matheux, réservé aux gens qui comprennent les fonctions de Liapounov), d'innombrables études et recherches sur la congestion. (Voir par exemple l'article du Wikipédia anglophone sur TCP.) Pourtant, le problème est loin d'être épuisé, comme le montre ce RFC du groupe ICCRG <http://tools.ietf.org/group/irtf/trac/wiki/ICCRG> de l'IRTF consacré aux problèmes non résolus en matière de congestion. Couvrant tous les aspects de ce problème, il est également une passionnante lecture sur l'état actuel de l'Internet et le bouillonnement d'activité qu'il suscite toujours chez les chercheurs.

Qu'est-ce exactement que la congestion ? La section 1 la définit comme l'état d'un réseau dont les ressources sont tellement utilisées que des phénomènes visibles par les utilisateurs et objectivement mesurables (comme le retard d'acheminement des paquets, voire leur perte) apparaissent. Sur un réseau où les ressources sont partagées mais pas réservées (cas de l'Internet), la congestion se traduit par une diminution de la qualité de service. (Voir Keshav, S., « "*What is congestion and what is congestion control*" <http://trac.tools.ietf.org/group/irtf/trac/raw-attachment/wiki/Agenda/keshav.pdf> », "*Presentation at IRTF ICCRG Workshop, PFLDNet 2007*", Los Angeles, février 2007.)

Le contrôle de congestion, lui, est l'ensemble des algorithmes qui permettent de partager les ressources en minimisant la congestion. Il existe sous deux formes, "*primal*" et "*dual*" (je n'ai pas cherché à traduire ces termes rares, apparemment issus de la recherche opérationnelle). La première fait référence à la capacité des émetteurs à adapter leur rythme d'envoi s'ils reçoivent des indications comme quoi la congestion a lieu. Sur l'Internet, c'est typiquement le rôle de TCP. La seconde forme fait référence au travail des équipements intermédiaires, les routeurs, qui détectent la congestion et réagissent, par exemple via la RED.

Aujourd'hui, des dizaines de RFC ont « congestion » dans leur titre ; le RFC 5783 [1] fournit un survol de cette littérature et le RFC 5681 synthétise tout ce qu'il faut savoir sur la congestion dans TCP. Dans les mises en œuvre de TCP/IP aujourd'hui, d'innombrables lignes de code sont là pour réagir à la congestion. Malheureusement, un bon nombre des algorithmes et méthodes « historiques » pour gérer la congestion sur l'Internet rencontrent leurs limites avec les évolution des réseaux. Résultat, plusieurs systèmes d'exploitation sont livrés avec des algorithmes non-standard (comme, dans certains cas, Linux, avec l'algorithme Cubic ; l'algorithme en cours peut être affiché avec `sysctl net/ipv4/tcp_congestion_control` et les disponibles avec `sysctl net/ipv4/tcp_available_congestion_control`). (On peut voir aussi les projets de FreeBSD <http://freebsdfoundation.blogspot.com/2010/12/five-new-tcp-congestion-control.html> qui incluent également Cubic et d'autres.) À noter que tous les nouveaux algorithmes ne sont pas forcément un progrès. Francis Dupont fait ainsi remarquer, suite à cet article, que des algorithmes comme « TCP Vegas », excellent lors des tests, ne réussissaient pas aussi bien sur le vrai réseau.

Autre évolution, l'utilisation de solutions qui ne sont pas uniquement de bout en bout mais font participer les routeurs, comme ECN (RFC 3168).

Quels sont les défis d'aujourd'hui ? La section 2 examine les défis généraux. L'un des premiers est l'hétérogénéité de l'Internet (section 2.1). Des liens radio laissant passer quelques misérables kb/s aux fibres optiques Gb/s, l'éventail des capacités est immense. Même chose pour la latence, entre les réseaux locaux où elle est inférieure à la milli-seconde et les liaisons satellites où elle approche la seconde. Rien n'indique que de tels écarts se résorberont. On imagine le travail de TCP, qui doit fonctionner dans un environnement aussi varié. À l'époque de Van Jacobson, latence et capacité des réseaux faisait qu'une connexion TCP n'avait en transit, à un moment donné, que quelques douzaines de paquets. Aujourd'hui, cela pourrait être beaucoup plus. Il n'y a pas actuellement de consensus à l'IETF pour choisir les nouveaux algorithmes, d'autant plus que leurs interactions (entre un pair TCP qui utiliserait un ancien algorithme et un pair qui utiliserait un nouveau) sont mal connues. Si le RFC 5033 définit des critères sur le choix d'un algorithme, ceux-ci ne répondent pas à toutes les questions, comme celle de savoir si on accepte de réduire les performances des anciennes implémentations (ce que font certaines nouvelles méthodes).

Second grand problème, la stabilité (section 2.2). On souhaiterait que, à situation inchangée, les algorithmes de contrôle de la congestion **convergent** vers un état stable, au lieu de faire du ping-pong entre deux états (on ouvre les vannes, la congestion reprend, on diminue le débit, le trafic passe très en dessous du seuil, alors on rouvre les vannes, etc). Il existe des théories mathématiques à ce sujet pour des algorithmes en boucle fermée, comme TCP, mais qui ne servent pas dans le cas où il y a interaction entre plusieurs boucles de contrôle, ce qui est le cas de l'Internet, où le ralentissement du débit d'une connexion TCP peut pousser les autres à augmenter leur débit. On souhaite également une stabilité locale : si un système est perturbé, après la fin de la perturbation, il revient rapidement à l'équilibre.

Les modélisations issues de la théorie de l'automatique permettent d'éliminer les algorithmes les plus mauvais (s'ils ne marchent pas dans les cas simples, ce n'est pas la peine de se demander s'ils fonctionneront sur l'Internet). Mais elles sont insuffisantes pour les cas réels, l'approche la plus courante aujourd'hui est donc la simulation, ce qui ne garantit pas que ça fonctionnera en vrai. Est-ce que TCP est stable dans des conditions réelles ? Il n'y a aujourd'hui aucune certitude scientifique à ce sujet.

Ce ne sont pas les heuristiques qui manquent, lorsqu'on conçoit un nouvel algorithme de gestion de la congestion, comme par exemple le principe de la conservation des paquets (faire en sorte que

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc5783.txt`

l'émetteur envoie autant de paquets par seconde que le récepteur en consomme). Mais elles peuvent être trop exigeantes (c'est le cas de la conservation de paquets, dont il a été démontré qu'elle était un principe suffisant, mais pas nécessaire) ou tout simplement non démontrées.

Un exemple d'un problème réel est que TCP démarre lentement : son débit s'accroit avec le temps au fur et à mesure que l'envoyeur se rassure qu'il n'y a pas congestion. Mais beaucoup de flots de données, à commencer par les courtes connexions HTTP qui récupèrent une seule page HTML, sont de durée tellement faible que TCP n'atteint jamais son état d'équilibre et de débit maximum. Cela peut pousser certains à « améliorer » l'algorithme dans leur intérêt (c'est le cas de Google <http://blog.benstrong.com/2010/11/google-and-microsoft-cheat-on-slow.html>, voir leur article <http://www.google.com/research/pubs/pub36640.html>). Égoïsme ou bien usage normal de la liberté ? En tout cas, cela a poussé à des recherches sur un TCP résistant aux égoïstes comme dans l'article « "*TCP congestion control with a misbehaving receiver*" <http://www.cs.washington.edu/homes/tom/pubs/CCR99.pdf> ».

Beaucoup plus chaude est la question de la **justice**, surtout dans le contexte du débat sur la neutralité de l'Internet. C'est en effet une chose d'assurer le bien de l'Internet, c'en est une autre que de vérifier que cela ne se fait pas aux dépens de certains utilisateurs, sacrifiés au bien commun. Intuitivement, la justice, dans un réseau, c'est que, en cas de sacrifices, tout le monde doit prendre sa part de manière équitable (section 2.3). On touche là clairement à des questions politiques, suscitant des articles vengeurs comme le très intéressant article de Briscoe, « "*Flow Rate Fairness : Dismantling a Religion*" <http://ccr.sigcomm.org/online/?q=node/172> » ("*ACM SIGCOMM Computer Communication Review, Vol.37, No.2, pp.63-74*", avril 2007). Briscoe argumente essentiellement sur le fait que la justice entre les flots TCP n'a aucun intérêt, il faudrait chercher la justice entre les personnes, ou bien entre les organisations. Mais, de toute façon, la question de la justice est connue pour être impossible à définir de façon cohérente dans le parallélisme (je cite Francis Dupont) : elle est sujette à un problème de la famille du paradoxe de Condorcet sur le vote équitable, c'est-à-dire que les critères qui établissent l'équitabilité ne sont pas compatibles.

Des RFC ont également traité ce cas comme le RFC 3714. Dans l'approche traditionnelle, pour le cas ultra-simple de deux utilisateurs regardant tous les deux YouTube, la justice était considérée comme l'égalité de leurs débits. Certains idéologues du marché ont par exemple proposé de définir la justice comme « ce que les utilisateurs sont prêts à payer pour éviter la congestion ». Dans cette approche, les deux débits pourraient être très différents. Mais intégrer l'aspect financier ouvre bien d'autres boîtes remplies de questions difficiles comme le mécanisme par lequel on va comptabiliser la contribution à la congestion.

Même si on ne prend pas en compte l'argent dans l'équation, la définition de ce qui est « juste » n'est pas triviale. Pas exemple, l'égalité doit-elle être celle des bits/s ou bien celle des paquets/s ? Faut-il la mesurer sur le long terme de façon à ce qu'une application sur une machine puisse profiter à fond du réseau parce que cette machine a été raisonnable dans les heures précédentes ? Le RFC note justement que ce n'est pas forcément une question de recherche, c'est un débat, et aucun article scientifique ne va le trancher. Le RFC note également que des mécanismes trop complexes, au nom de la justice, pourraient sérieusement handicaper l'Internet (car ralentissant trop les équipements, cf. RFC 5290).

Place maintenant à l'étude détaillée des défis qui attendent l'IETF dans ce domaine. Ils font l'objet de la section 3. Le premier est celui de la répartition du travail entre le réseau (les équipements intermédiaires comme les routeurs) et machines situées aux extrémités (section 3.1). Traditionnellement, l'Internet fonctionne en limitant sérieusement les fonctions des routeurs et en concentrant le travail difficile aux extrémités, où les machines ont davantage de ressources (section 3.1.1), peuvent maintenir un état complexe, et surtout sont sous le contrôle des utilisateurs (mettre trop de fonctions dans les routeurs peut sérieusement menacer la neutralité du réseau).

En outre, si de nouveaux mécanismes de gestion de la congestion apparaissent, et qu'ils sont situés dans les couches basses (comme la couche 3), ils mettront du temps à être déployés, d'autant plus que les concepteurs de routeurs attendront sans doute que ces mécanismes soient stables avant de les câbler en dur dans leurs circuits. Mais, d'abord, que peut faire un routeur pour aider à gérer la congestion ? Il peut le faire par une meilleure gestion de ses files d'attente, ou bien il peut le faire en **signalant** aux extrémités les problèmes de congestion qu'il voit. La première approche est déjà utilisée mais pas de manière uniforme dans tout l'Internet. Une de ses limites est qu'on ne sait pas trop quels sont les paramètres idéaux, par exemple pour les méthodes RED ou AVQ ("*Adaptive Virtual Queue*") et qu'on ne dispose pas de moyen de les déterminer automatiquement.

La seconde façon dont les routeurs peuvent aider, la signalisation explicite, est utilisée dans ECN (RFC 3168), Quick-Start (RFC 4782), XCP ("*eXplicit Control Protocol*", `draft-falk-xcp-spec`) et dans d'autres protocoles. Contrairement à TCP, XCP sépare le contrôle de la justice (s'assurer qu'aucun flot de données ne prend toute la capacité) et le contrôle de la congestion. Mais XCP reste très expérimental et produit parfois de drôles de résultats (« "*An Example of Instability in XCP*" `<http://netlab.caltech.edu/maxnet/XCP_instability.pdf>` »). Dans tous les cas, ces mécanismes, qui donnent un plus grand rôle aux équipements intermédiaires, doivent être évalués dans le contexte du fonctionnement de bout en bout, qui est l'un des principaux piliers de l'Internet (RFC 1958), qui a assuré son succès et sa survie. La congestion est un problème intéressant précisément parce qu'elle ne peut pas se régler uniquement de bout en bout (puisque c'est un phénomène spécifique au réseau) mais il faut bien réfléchir avant de mettre dans le réseau des fonctions qui pourraient interférer avec les politiques d'utilisation des machines terminales.

Ainsi, un protocole comme XCP ne nécessite pas d'état dans le réseau (et semble donc respecter le principe de bout en bout) mais c'est uniquement à condition que les machines situées aux extrémités jouent le jeu, c'est-à-dire ne mentent pas sur les débits mesurés. Si on veut faire marcher XCP dans un environnement... non-coopératif (avec des tricheurs), les routeurs doivent alors garder trace des différents flots de données, ce qui met en péril le principe de bout en bout.

Tous ces problèmes (de comptabiliser l'usage que les différents flots de données font du réseau, et en résistant à la triche) existent depuis qu'on parle de QoS sur l'Internet, c'est-à-dire au moins depuis IntServ (RFC 2208). Le problème reste ouvert.

Une des raisons pour lesquels ils ne sont pas résolus est qu'il n'existe aucune technique (section 3.1.2) qui ne repose pas sur un état maintenu dans le routeur, par flot (donc, posant de grands problèmes de passage à l'échelle).

Autre problème pour l'usage des équipements réseaux à des fins de lutter contre la congestion, l'accès à l'information (section 3.1.3). Pour être efficace, le routeur qui veut gérer la congestion doit connaître des choses comme la capacité des liens. Attendez, me direz-vous, un routeur sait forcément que son interface `ge-3/0/2` est à 1 Gb/s ! Mais c'est plus complexe que cela : par exemple en cas d'interface vers un réseau partagé (Wifi) ou en cas de tunnels (RFC 2784 ou RFC 4301 ou bien d'autres techniques). Dans ces circonstances, un routeur IP ne connaît pas forcément la capacité des liens auxquels il est connecté.

Idéalement, si le routeur connaît la capacité du lien et le débit effectif, il sait si on approche de la congestion ou pas, via une simple division. Nous avons vu que la capacité n'était pas toujours connue mais le débit ne l'est pas non plus forcément. S'il est très haché, avec de brusques pics, déterminer **un** débit, pour le diviser par la capacité, n'est pas trivial. Bref, le routeur manque souvent d'informations pour décider. Faut-il créer de nouveaux mécanismes pour l'informer ? Ne risquent-ils pas de tourner très vite à l'usine à gaz ? (Un routeur IP a déjà **beaucoup** de choses à faire.)

Deuxième défi auquel est confronté TCP/IP : différencier la perte d'un paquet due à la congestion de l'abandon de ce paquet en raison de sa corruption (section 3.2). Actuellement, un routeur abandonne un paquet lorsque la congestion empêche sa transmission sur l'interface suivante. L'émetteur va typiquement interpréter toute perte de paquets comme un signe de congestion et diminuer le débit (TCP le fait automatiquement). Mais la perte du paquet peut être due à une autre cause, par exemple une modification accidentelle d'un ou plusieurs bits, qui rendra invalide la somme de contrôle, amenant la carte réseau à ignorer le paquet anormal. De telles modifications sont rares sur un lien filaire (il faut un gros rayon cosmique) mais relativement fréquentes sur les liaisons radio. TCP diminuerait alors son débit à tort. Le problème est bien connu (voir Krishnan, Sterbenz, Eddy, Partridge & Allman, « "*Explicit Transport Error Notification (ETEN) for Error-Prone Wireless and Satellite Networks*" <http://www.icir.org/mallman/papers/eten-exec-summary.pdf> », "*Computer Networks*", 2004, ou bien le RFC 6069). La solution n'est pas évidente, d'autant plus que les deux phénomènes peuvent être liés (une congestion qui se traduit par des corruptions) et qu'il n'est pas évident qu'il faille continuer à envoyer au même débit si on perd la moitié des paquets suite à la corruption (voir un exemple de discussion <http://www.ietf.org/mail-archive/web/dccp/current/mail6.html>).

Il y a donc deux problèmes ouverts, détecter la congestion et y réagir. Pour la détection, on pourrait penser compter sur la somme de contrôle dans l'en-tête IP (à condition que la couche 2 n'ait pas déjà jeté le paquet corrompu, ce que fait Ethernet). UDP-lite (RFC 3828) et DCCP (RFC 4340) le font (sans indiquer quelle doit être la réaction à ces corruptions). Une étude a montré que cela marchait bien au-dessus de GPRS et mal au dessus du WiFi. De toute façon, toute détection de corruption dans la couche 3 poserait des problèmes de coopération entre couches, difficiles à assurer pour IP, qui est conçu pour fonctionner sur toutes les couches 2 possibles.

Quant à la réaction à la corruption, il n'existe aucun protocole standard pour cela, bien que plusieurs propositions aient été faites dans la littérature scientifique (le TCP dit « Westwood »).

Un autre « point noir » de la gestion de la congestion est la prise en compte de la taille du paquet (section 3.3). Aujourd'hui, TCP l'ignore complètement, en partie parce que les très grands paquets (¿ 1500 octets, la MTU d'Ethernet) sont extrêmement rares sur l'Internet. La question de savoir si TCP a tort ou raison d'ignorer ce facteur est toujours un sujet de recherche. DCCP, par contre, peut tenir compte de la taille des paquets (cf. RFC 5622). Les routeurs typiques, en cas de congestion, ne tiennent pas compte de la taille d'un paquet avant de le jeter, alors que certaines études semblent indiquer que cela pourrait être une bonne idée. Beaucoup de questions restent ouvertes sur ce sujet : le facteur limitatif sur le réseau, dans les prochaines années, sera-t-il le nombre de bits ou le nombre de paquets (sur les réseaux lents, c'est clairement le nombre de bits qui peut déclencher la congestion ; mais au fur et à mesure que les réseaux deviennent plus rapides, c'est la capacité des équipements à traiter un grand nombre de paquets qui devient problématique). En cas de congestion, faut-il diminuer le nombre de bits (en réduisant le débit d'émission, ce que fait TCP) ou le nombre de paquets (en tentant des paquets plus gros) ? Actuellement, l'émetteur ne sait pas pourquoi il y a congestion, ce qui rend difficile le changement de son comportement. Prendre en compte la distinction entre débit en bits et débit en paquets compliquerait l'Internet mais y aurait-il un bénéfice ? Si la taille maximale des paquets sur le réseau, actuellement limitée en pratique à 1500 octets dans la plupart des cas, devait augmenter, la question deviendrait brûlante. (Ce point est couvert avec plus de détails dans le RFC 7141.)

La grande majorité des études sur le comportement de protocoles comme TCP supposait un flot relativement régulier sur le long terme, permettant ainsi aux algorithmes d'atteindre leur équilibre. Mais, dans la réalité, les choses ne se passent pas comme cela : un fichier HTML typique se charge avec seulement quelques paquets, et TCP sera donc, pendant toute la durée du flot, dans l'état « démarrage ». Comment améliorer ce cas fréquent (section 3.4) ? Au départ, le protocole de transport ne connaît rien des caractéristiques de la liaison et ne peut donc pas optimiser son débit. L'algorithme standard de TCP est tout de prudence : conçu par Van Jacobson et connu sous le nom significatif de "*slow start*", il consiste à être très modéré tant qu'on ne connaît pas les détails de la liaison (RFC 2581 et RFC 5681). Résultat, le

débit moyen est souvent trop bas, le réseau restant inoccupé. Ce problème est reconnu depuis longtemps (RFC 3742) mais sa solution souffre du manque de bases théoriques à cette difficile question (le "*slow start*" a été conçu de manière empirique).

Une autre solution serait de modifier les applications pour qu'elles réutilisent les connexions TCP existantes. C'est ce que peut faire HTTP, avec les connexions persistentes (RFC 2616, section 8.1).

Les spécificités de l'Internet ajoutent un nouvel élément au tableau (section 3.5). Contrairement à quasiment tous les autres réseaux informatiques qui ont existé, l'Internet est multi-organisations (le RFC dit "*multi-domains*", où un "*domain*" est un système autonome). Ainsi (section 3.5.1), il existe des mécanismes de signalisation de la congestion qui peuvent poser des problèmes dans un tel environnement. C'est le cas de ECN (RFC 3168) où des bits non utilisés jusqu'à présent servent à indiquer que le routeur qui les a mis commence à voir de la congestion. En environnement mono-domaine, tout va bien. Mais dans le cas où le paquet rencontre plusieurs domaines, peut-on encore faire confiance à ces bits, qui ont pu être mis ou retirés à tort, par exemple pour ralentir une victime située dans un autre domaine (sections 8 et 19 du RFC 3168) ? Pour ECN, il y a des solutions techniques possibles (comme le RFC 3540), quoique apparemment peu déployées. Mais, d'une manière générale, toutes les solutions reposant sur la confiance entre domaines sont incertaines (notre RFC cite aussi le cas de TRC, "*TCP rate controller*", un système de surveillance que les flots TCP se comportent bien). L'échange d'informations entre domaines, qui serait certainement utile pour lutter contre la congestion, est frappé de suspicion (section 3.5.2). Mon voisin me dit-il la vérité ? Dois-je lui dire la vérité en sachant que les informations transmises peuvent, entre autres, l'aider à se faire une bonne image de mon réseau, brisant ainsi la confidentialité ? Or, tous les mécanismes de signalisation de la congestion par les équipements réseau sont affectés par ce manque de confiance.

Un problème de sécurité proche est celui des applications malhonnêtes. L'Internet résiste aujourd'hui à la congestion car chaque pair TCP respecte les règles, même quand elles vont contre son propre intérêt. Après tout, un émetteur n'a jamais intérêt à diminuer son débit, si d'autres le feront à sa place. La section 3.7 se penche sur les comportements négatifs que pourraient adopter des applications. Le problème a déjà été étudié et des solutions proposées (RFC 3540). Elles sont efficaces (peut-être trop, dit le RFC, laissant entendre qu'elles pourraient sanctionner des innocents). Mais elles sont très peu déployées. Une des raisons est peut-être que les attaques sont pour l'instant apparemment très peu nombreuses. En fait, note notre RFC, l'étude de la quantité de telles attaques est en soi un sujet de recherche. On ne sait pas vraiment dire pour l'instant s'il y a beaucoup de tricheurs (peut-être l'exemple Google cité plus haut mais il est très récent et pas assez étudié). On soupçonne évidemment que tout logiciel qui annonce qu'il va « optimiser le temps de téléchargement » risque fort d'y arriver en trichant...

TCP n'est pas le seul protocole de transport : certaines applications utilisent un protocole sans contrôle de congestion, comme UDP, et sont donc censées gérer la congestion elles-mêmes. Or, comme indiqué plus haut, ce n'est pas dans leur intérêt et on pourrait voir des abus, comme s'en inquiétait le RFC 3714. Cette question est presque un problème philosophique : une machine connectée à l'Internet est libre, elle peut notamment utiliser le protocole de transport de son choix. Envoyer des octets au maximum de ses possibilités, sans tenir compte de la congestion, n'est-ce pas abuser de cette liberté ? Comme les applications qui utilisent UDP sont souvent les plus gourmandes (vidéo en haute définition, par exemple), le RFC 8085 insistait sur la nécessité qu'elles se comportent gentiment et mettent en œuvre une forme de contrôle de congestion adaptée à leurs bsoins propres.

Les applications ont des exigences souvent très différentes vis-à-vis du réseau. Par exemple, les gros transferts de fichiers peuvent souvent s'exécuter en arrière-plan en profitant de la capacité du réseau quand il est libre, tout en réduisant sérieusement leur débit s'il ne l'est pas (section 3.6). Comment utiliser cette « élasticité » pour mieux gérer la congestion ? L'idée de marquer les paquets en fonction de ces exigences des applications n'est pas neuve (on la trouve déjà dans les bits ToS du RFC 791) mais il

y a aussi des propositions plus récentes comme le RFC 3662. Mais en pratique, ces mécanismes ne sont guère utilisés. Actuellement, un groupe de travail de l'IETF, LEDBAT <http://tools.ietf.org/wg/ledbat>, travaille sur ce sujet (voir leur solution dans le RFC 6817).

Enfin, il reste les autres défis, ceux qui sont jugés relativement peu importants aujourd'hui et n'ont donc pas mérité une section dédiée. Ils sont regroupés dans la section 3.8. On y trouve, par exemple, la question de la mesure du RTT (bien des protocoles dépendent d'une bonne évaluation de ce temps). Aujourd'hui, seul l'émetteur d'un paquet a une bonne idée du RTT, en mesurant le moment où revient l'accusé de réception. Il n'existe malheureusement pas de moyen pour le receveur d'estimer le RTT, ce qui l'empêche d'ajuster son comportement. D'autre part, le RTT brut n'est pas forcément utilisable directement (par exemple parce qu'il varie brutalement dans le temps) et il est utile de se pencher sur des valeurs extrapolées, plus lisses, comme celles du RFC 2988.

Autre défi qui se posera un jour (sections 3.8.2) : le cas des logiciels bogués, qui réagissent mal lorsqu'on essaie une nouvelle technique, même si elle est parfaitement légale : beaucoup de machines sur Internet ont été programmées avec les pieds et plantent (ou même plantent leur voisin) si les données envoyées ne sont pas celles attendues (comme ce fut le cas lors de la fameuse crise de l'attribut BGP 99 <https://www.bortzmeyer.org/bgp-attribut-99.html>). Ainsi, même si on conçoit un nouvel algorithme génial de contrôle de la congestion, il sera peut-être difficile à déployer. C'est ce qui est arrivé à ECN, qui a connu bien des problèmes <http://www.icir.org/tbit/ecn-tbit.html> mais aussi au "*window scaling*" que plusieurs routeurs de bas de gamme rejettaient (c'est arrivé par exemple sur les "*boxes*" d'Alice). Y a-t-il une solution à ce problème ? Le RFC suggère une responsabilité au moins partielle de l'IETF en demandant que, à l'avenir, le comportement par défaut des équipements, face à des options ou des champs inconnus, soit mieux spécifié, pour que le vieux logiciel laisse toujours passer proprement les options récentes. Je pense personnellement que c'est une illusion : dans bien des cas, ces règles existaient et les logiciels bogués les ignorent, car leurs auteurs, dissimulés derrière leur anonymat (comment retrouver les responsables du code des AliceBox ?) ont ignoré complètement ces règles. La section 3.8.6 revient d'ailleurs sur le cas des équipements intermédiaires ("*middleboxes*") qui sont souvent les pires (RFC 2775).

Comme déjà signalé plus haut, un défi général auquel est confronté le contrôle de congestion est celui de la sécurité (section 4). Par exemple (RFC 4948), les DoS sont possibles parce que l'Internet n'impose pas de contrôle de congestion (attention, le RFC parle de dDoS mais c'est à mon avis une erreur : si certains réseaux ont des mécanismes de contrôle de congestion obligatoires, aucun ne prévoit un mécanisme global, qui pourrait arrêter un "*botnet*" composé de milliers de zombies indépendants). De manière plus générale, aujourd'hui, le respect des règles qui empêchent l'Internet de s'écrouler sous le poids de la congestion est facultatif, laissé à la bonne volonté de chacun. Cela peut être vu comme un problème de sécurité. S'il existait des mécanismes de contrôle de la congestion qui résistent aux égoïstes prêts à tricher, ces mécanismes pourraient également être utilisés contre les DoS.

Voilà, désolé d'avoir été si long mais ce RFC est très riche et le champ des problèmes non encore résolus dans ce domaine de la congestion est immense. Étudiants futurs chercheurs, à vous de travailler à le défricher !

Merci à Pierre Beyssac pour sa relecture et ses commentaires.