RFC 6297 : A Survey of Lower-than-Best-Effort Transport Protocols

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 29 juin 2011

Date de publication du RFC : Juin 2011

https://www.bortzmeyer.org/6297.html

Aujourd'hui, de nombreuses applications se battent pour accéder à une ressource critique, la **capacité** du réseau. Tout le monde veut télécharger plus vite et les applications tentent d'obtenir le meilleur débit possible pour recevoir le dernier épisode de Dr House ou bien la page de ses amis sur Facebook. Pourtant, il existe une catégorie d'applications plus philosophes, qui cherchent à être le moins dérangeantes possibles et à ne transférer des octets que lorsqu'absolument personne d'autre n'a besoin du réseau. Ce RFC examine les techniques utilisées par ces applications tranquilles.

Il existe bien des cas où il n'y a pas d'urgence à transférer des données. Des mises à jour quotidiennes de gros fichiers ou bases de données non critiques, le transport des nouvelles Usenet (cf. RFC 5537 [1]), du téléchargement qu'on laisse tourner tout le temps, par exemple. Ces transferts, même s'ils n'ont pas besoin d'être « temps-réel » rentrent actuellement en compétition avec les transferts pressés. Si on télécharge tranquillement, sans être pressé, un gros film, et qu'on a un besoin urgent d'un fichier pendant ce temps, le fichier n'aura, pendant son transfert, que la moitié de la capacité, l'autre étant prise par le film. Il serait donc intéressant d'avoir des applications gentilles, modestes, qui cèdent systématiquement la place aux applications normales, dès que le réseau est encombré. L'IETF a un groupe de travail pour ce sujet du trafic « d'arrière-plan », dit aussi « moins que au mieux » (le trafic normal dans l'Internet étant acheminé « au mieux » - "*best effort*"). Ce groupe se nomme LEDBAT <http://tools.ietf.org/wg/ledbat> ("*Low Extra Delay Background Transport*"), et voici son premier RFC, qui étudie les techniques existantes pour atteindre cet objectif.

On pourrait même imaginer des tarifs intéressants ou des conditions d'accès au réseau plus favorables pour ces applications. Après tout, l'essentiel des coûts d'un réseau sont fixes : une fois la fibre posée et allumée, une fois les routeurs et commutateurs installés et configurés, le coût d'exploitation est

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc5537.txt

le même que des bits passent ou pas. Ce qui coûte cher, en exploitation, c'est la **congestion**, lorsque la capacité du réseau est trop faible, que le service commence à se dégrader, et que les clients réclament une mise à jour de l'infrastructure, qui ne sera pas gratuite. Par contre, si le réseau est inutilisé à certains moments, y faire passer nos transferts de données gentils ne coûte rien. Dans tous les cas, tarifs intéressants ou pas, avoir des applications modestes aideraient certains services gourmands (comme le transfert de fichiers en pair-à-pair) à se faire accepter : on n'hésitera plus à laisser tourner BitTorrent en permanence si on est sûr qu'il cédera toujours aux applications habituelles.

La référence pour le trafic normal est TCP, le protocole de transport qu'utilisent quasiment tous les transferts de données et qui fait au mieux pour utiliser toute la capacité du réseau, ou pour la partager équitablement si elle est insuffisante (RFC 5681). Notre RFC vise donc le trafic « moins que TCP » tel que, s'il y a compétition entre trafic normal et notre trafic d'arrière-plan, le trafic normal ait la part du lion.

Le trafic d'arrière-plan est nommé dans ce RFC **LBE** pour "*less than best-effort*". Un système LBE doit donc réagir à la congestion plus vite que TCP, pour abaisser son débit. Il y a quatre catégories de systèmes LBE :

— Ceux qui agissent uniquement au niveau de l'application, et utilisent donc un TCP normal. C'est plus difficile à faire que ça n'en a l'air, car, si on applique strictement le modèle en couches, les applications n'ont pas connaissance de la congestion. Mais ce sont aussi les plus faciles à déployer, puisqu'on a juste à installer une nouvelle application.
— Ceux qui s'appuient sur un protocole de transport modifié. Il en existe deux catégories, ceux qui se basent sur les délais d'acheminement des paquets (et non sur le taux de pertes, comme le fait TCP), et les autres. Ces protocoles de transport peuvent éventuellement réutiliser une partie de TCP (par exemple son format d'en-tête), facilitant ainsi leur déploiement. (Plusieurs des exemples se nomment « TCP quelque chose » pour cette raison. Wikipédia a une bonne page sur les variantes de TCP.)
— Ceux qui s'appuient sur le réseau. Ce sont évidemment les plus durs à déployer, car ils nécessitent que tous les routeurs du trajet soient modifiés.

Les sections suivantes fournissent divers exemples (je ne les reprends pas tous ici) de chaque catégorie.

Les solutions dans les applications font l'objet de la section 4. A priori, on pourrait croire que les solutions de "*shaping*" comme l'option `--bwlimit` de rsync atteignent notre objectif de « trafic d'arrière-plan », ne rentrant jamais en compétition avec TCP. Mais ce n'est pas le cas. Les solutions de "*shaping*" sont :

— trop modestes lorsque le tuyau est inutilisé. Si je transfère avec `rsync --bwlimit 64`, le débit ne dépassera jamais 64 ko/s.
— trop gourmandes si le tuyau est encombré. rsync essaiera d'avoir ses 64 ko/s dans tous les cas.

Bref, ces options ne sont pas réellement LBE (voir Crovella, M. et P. Barford, « "*The network effects of prefetching*" <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.115.8173&rep=rep1&type=pdf> », "*Proceedings of IEEE INFOCOM 1998*", pour une meilleure description d'une solution de limitation de débit) .

De même, essayer de faire tourner les applications LBE aux heures creuses ne convient pas à toutes les applications (par exemple le transfert de fichiers en pair-à-pair). Autre solution, le BITS <http://msdn.microsoft.com/library/bb968799(VS.85).aspx> de Windows, qui surveille en permanence le débit pour ralentir ces transferts de fichiers en arrière-plan si le débit devient trop important. (Le RFC ne fournit pas d'éléments d'évaluation précis de ce service.)

Comment réaliser une application modeste ? Une première possibilité (section 4.1) est d'agir sur la fenêtre de réception, sur la machine destinataire. Le récepteur diminue la fenêtre TCP, par exemple lorsque le délai d'acheminement des paquets augmente (sur Unix, cela doit être mis en œuvre dans le noyau, je ne crois pas que l'application ait accès à cette fenêtre, mais cela ne nécessite pas de modifier le

protocole TCP, c'est juste un changement unilatéral de comportement). C'est par exemple ce qui est fait en Spring, N., Chesire, M., Berryman, M., Sahasranaman, V., Anderson, T., et B. Bershad, « "*Receiver based management of low bandwidth access links*" <http://www.cs.washington.edu/homes/tom/pubs/low-bw.pdf> ».

Dans les deux catégories de nouveaux protocoles de transport, voyons d'abord ceux fondés sur le délai d'acheminement. Le principe est de mesurer le temps que mettent les paquets à faire le trajet. Si cette durée augmente, c'est sans doute parce que le réseau est encombré et que donc la congestion approche. On diminue donc le débit (TCP, lui, attend bien plus, que les paquets soient effectivement jetés). La section 2 cite comme exemple TCP Vegas, un des premiers à pouvoir partager un lien avec TCP en étant systématiquement moins gourmand que lui, même si, curieusement, ce n'était pas son but initial. TCP Vegas permet au contraire souvent un meilleur débit, lorsqu'il est seul. Cela illustre le fait que ce n'est pas en diminuant le débit (cf. la discussion sur rsync) qu'on atteint l'objectif de modestie.

TCP Vegas est toujours la référence dans ce domaine. Mais, depuis, il existe des améliorations comme TCP Nice (voir Venkataramani, A., Kokku, R., et M. Dahlin, « "*TCP Nice : a mechanism for background transfers*" <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.81.5905&rep=rep1&type=pdf> », qui reprennent le même principe de diminuer le débit lorsque le RTT diminue (sans attendre que les pertes de paquets commencent). À noter que TCP Nice a une mise en œuvre dans Linux (citée dans l'article ci-dessus).

Autre cas de protocole fondé sur le délai, TCP-LP <http://www.ece.rice.edu/networks/TCP-LP/>, qui utilise le délai en aller simple et pas le RTT comme les deux précédents. C'est normalement un meilleur indicateur (on n'est pas perturbé par le trafic de retour, qui peut n'avoir rien à voir puisque le chemin de retour peut être différent, et/ou moins congestionné) mais il est plus difficile à mesurer (TCP-LP utilise l'option d'estampillage temporel des paquets de TCP, cf. RFC 7323).

Tous ces schémas ont un défaut commun : les études (par exemple McCullagh, G. et D. Leith, « "*Delay-based congestion control : Sampling and correlation issues revisited*" <http://www.hamilton.ie/net/correl_ToN.pdf> ») montrent que le délai d'acheminement n'a qu'un faible rapport avec l'approche de la congestion. Le délai varie souvent pour des raisons qui n'ont rien à voir avec l'encombrement (les tampons devraient être bien plus grands pour mesurer ce délai sans ce bruit ; les liens radio vont varier le délai quel que soit l'encombrement ; etc). En outre, les horloges des machines sont souvent insuffisantes pour mesurer le délai avec une bonne précision, surtout sur un réseau local où les paquets voyagent vite et où l'essentiel du délai est dû au traitement dans les machines, pas au réseau. Enfin, certaines techniques d'optimisation de TCP faussent encore plus cette mesure.

En pratique, ces problèmes peuvent mener à réduire le débit sans raison. Ces « faux positifs » atteignent quand même l'objectif (être moins gourmand que TCP) mais les problèmes de mesure peuvent aussi mener à des faux négatifs, où le protocole de transport ne détecte pas une congestion bien réelle.

Enfin, tout protocole fondé sur le délai d'acheminement donne un avantage aux derniers arrivants : sur un tuyau déjà très encombré, le flot récent voit des délais importants dès le début et ne se rend donc pas compte de la congestion.

Il existe aussi des protocoles de transport LBE (modestes) n'utilisant pas le délai d'acheminement (section 3). Ils jouent sur le rythme d'envoi (qui dépend de la réception des accusés de réception) pour, par exemple, diminuer la fenêtre d'envoi plus vite que TCP. C'est le cas de 4CP <http://academic.research.microsoft.com/Publication/2540229/4cp-competitive-and-considerate-congestion-cont
ou de MulTFRC (une extension du TFRC du RFC 5348).

Enfin, il y a les solutions qui utilisent une assistance du réseau (section 5). Sur le papier, ce sont les plus efficaces (pas besoin de mesures indirectes, comme celle du délai d'acheminement, le routeur sait parfaitement si le réseau est encombré ou pas) mais aussi les plus dures à déployer car elles nécessitent une modification de tous les routeurs.

Un exemple d'une telle technique est NF-TCP (Arumaithurai, M., Fu, X., et K. Ramakrishnan, « "*NF-TCP : A Network Friendly TCP Variant for Background Delay- Insensitive Applications*" <http://www.net.informatik.uni-goettingen.de/publications/1718/NF-TCP-techreport.pdf> ») qui combine ECN (RFC 3168) et RED (RFC 7567) pour diminuer la priorité des flots NF-TCP dans les routeurs dès que la congestion menace.

Armé de la connaissance de toutes ces techniques (on voit que l'imagination des chercheurs et ingénieurs a été intense sur ce sujet), que va faire le groupe de travail LEDBAT <http://tools.ietf.org/wg/ledbat> ? La section 6 explique que LEDBAT a déjà un système, décrit un l'"*Internet-Draft*", fondé sur le délai d'acheminement » (ceux de la section 2). Ce n'est pas à proprement parler un protocole mais un algorithme, qui pourra être utilisé dans les protocoles concrets. Il a depuis été publié dans le RFC 6817.