RFC 8312 : CUBIC for Fast Long-Distance Networks

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 8 février 2018

Date de publication du RFC : Février 2018

https://www.bortzmeyer.org/8312.html

Longtemps après sa mise au point et son déploiement massif sur Linux, voici la description officielle de l'algorithme CUBIC, un algorithme de contrôle de la congestion dans TCP. (Cette description a depuis été remplacée par celle du RFC 9438 [1].)

CUBIC doit son nom au fait que la fonction de calcul de la fenêtre d'envoi des données est une fonction cubique (elle a un terme du troisième degré) et non pas linéaire. CUBIC est l'algorithme utilisé par défaut dans Linux depuis pas mal d'années :

```
% sudo sysctl net.ipv4.tcp_congestion_control
net.ipv4.tcp_congestion_control = cubic
```

CUBIC est plus énergique lorsqu'il s'agit d'agrandir la fenêtre d'envoi de données, lorsque le réseau a une grande capacité <https://www.bortzmeyer.org/capacite.html> mais un RTT important. Dans le cas de ces réseaux « éléphants » (terme issu de la prononciation en anglais de LFN, "*Long and Fat Network*", voir RFC 7323, section 1.1), le RTT élevé fait que l'émetteur TCP met du temps à recevoir les accusés de réception, et donc à comprendre que tout va bien et qu'il peut envoyer davantage de données, pour remplir le long tuyau. CUBIC permet d'accélérer cette phase.

Notez que CUBIC ne contrôle que l'émetteur, le récepteur est inchangé. Cela facilite le déploiement : un émetteur CUBIC peut parfaitement communiquer avec un récepteur traditionnel.

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc9438.txt

Avant de lire la suite du RFC, il est recommandé de (re)lire le RFC 5681, la bible sur le contrôle de congestion TCP, et notamment sur cette notion de fenêtre d'envoi (ou fenêtre de congestion).

TCP (RFC 793) a évidemment une mission difficile. L'intérêt de l'émetteur est d'envoyer le plus de données le plus vite possible. Mais à condition qu'elles arrivent sinon, s'il y a de la congestion, les données seront perdues et il faudra recommencer (ré-émettre). Et on n'est pas tout seul sur le réseau : il faut aussi tenir compte des autres, et chercher à partager équitablement l'Internet. L'algorithme doit donc être énergique (chercher à utiliser les ressources au maximum) mais pas bourrin (il ne faut pas dépasser le maximum), tout en étant juste (on n'est pas dans la "*startup nation*", il ne faut pas écraser les autres, mais partager avec eux).

Le problème des éléphants, des réseaux à fort BDP, est connu depuis longtemps (article de T. Kelly, « "*Scalable TCP : Improving Performance in HighSpeed Wide Area Networks*" <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.107.5330&rep=rep1&type=pdf> », et RFC 3649.) Dans ces réseaux, TCP tend à être trop prudent, à ouvrir sa fenêtre (les données qu'on peut envoyer tout de suite) trop lentement. Cette prudence l'honore, mais peut mener à des réseaux qui ne sont pas utilisés à fond. L'article de Ha, S., Kim, Y., Le, L., Rhee, I., et L. Xu, « "*A Step toward Realistic Performance Evaluation of High-Speed TCP Variants*" <https://pdfs.semanticscholar.org/56bd/dd44e0d73d2494d9a579c7dabf754560d2c6.pdf> » expose ce problème. Il touche toutes les variantes de TCP depuis le TCP Reno décrit dans le RFC 5681 : le New Reno des RFC 6582 et RFC 6675, et même des protocoles non-TCP mais ayant le même algorithme, comme UDP (TFRC, RFC 5348) ou SCTP (RFC 9260).

CUBIC a été originellement documenté dans l'article de S. Ha, Injong Rhee, et Lisong Xu, « "*CUBIC : A New TCP-Friendly High-Speed TCP Variant*" <http://www4.ncsu.edu/~rhee/export/bitcp/cubic-paper.pdf> », en 2008. Sur Linux, il a remplacé BIC pour les réseaux à haut BDP.

La section 3 du RFC rappelle les principes de conception de CUBIC, notamment :

— Utilisation de la partie concave (la fenêtre s'agrandit rapidement au début puis plus lentement ensuite) **et** de la partie convexe de la fonction, et pas seulement la partie convexe (on ouvre la fenêtre calmement puis on accélère), comme l'ont tenté la plupart des propositions alternatives. Si vous avez du mal avec les termes concave et convexe, la figure 2 de cet article de comparaison de CUBIC et BIC <https://research.csc.ncsu.edu/netsrv/?q=content/bic-and-cubic> illustre bien, graphiquement, ces concepts. La courbe est d'abord concave, puis convexe.
— Comportement identique à celui de ses prédécesseurs pour les liaisons à faible RTT (ou faible BDP). Les algorithmes TCP traditionnels n'ont en effet pas de problème dans ce secteur (cf. section 4.2, et Floyd, S., Handley, M., et J. Padhye, « "*A Comparison of Equation-Based and AIMD Congestion Control*" <https://www.icir.org/tfrc/aimd.pdf> »). « Si ce n'est pas cassé, ne le réparez pas. » CUBIC ne se différencie donc des autres algorithmes que pour les réseaux à RTT élevé, ce que rappelle le titre de notre RFC.
— Juste partage de la capacité entre des flots ayant des RTT différents.
— CUBIC mène à un agrandissement plus rapide de la fenêtre d'envoi, mais également à une réduction moins rapide lorsqu'il détecte de la congestion (paramètre « beta_cubic », le facteur de réduction de la fenêtre, voir aussi le RFC 3649.)

La section 4 du RFC spécifie précisement l'algorithme, après beaucoup de discussion avec les développeurs du noyau Linux (puisque le code a été écrit avant le RFC). Cette section est à lire si vous voulez comprendre tous les détails. Notez l'importance du point d'inflexion entre la partie concave et la partie convexe de la courbe qui décrit la fonction de changement de taille de la fenêtre. Ce point d'inflexion est mis à la valeur de la fenêtre d'envoi juste avant la dernière fois où TCP avait détecté de la congestion.

Notez que Linux met en outre en œuvre l'algorithme HyStart, décrit dans « "*Taming the Elephants : New TCP Slow Start*" <https://pdfs.semanticscholar.org/7f9a/d9212ccb9ab9b5614bef93347a4b0526

pdf> ». Hystart mesure le RTT entre émetteur et récepteur pour détecter (par l'augmentation du RTT) un début de congestion avant que des pertes de paquets se produisent. (Merci à Lucas Nussbaum pour l'information.)

La section 5 analyse le comportement CUBIC selon les critères de comparaison des algorithmes de contrôle de la congestion décrits dans le RFC 5033.

Pour finir, voici une intéressante comparaison des algorithmes de contrôle de congestion <http://tetcos.com/downloads/TCP_Congestion_Control_Comparison.pdf>.