RFC 6817 : Low Extra Delay Background Transport (LEDBAT)

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 20 décembre 2012

Date de publication du RFC : Décembre 2012

https://www.bortzmeyer.org/6817.html

Alors que tant d'efforts de recherche ont été dépensés pour faire des réseaux informatiques et des protocoles qui permettent d'aller **plus vite**, d'attendre moins longtemps avant de voir la page d'accueil de TF1, le groupe de travail LEDBAT <http://tools.ietf.org/wg/ledbat> ("*Low Extra Delay Background Transport* ") de l'IETF travaillait à un tout autre projet : un protocole de transport de données qui aille **moins vite**, de façon à être un bon citoyen du réseau, à n'utiliser celui-ci que lorsqu'il est vide et qu'on peut donc faire passer ses bits sans gêner personnne. Ce RFC décrit l'**algorithme** LEDBAT, un algorithme « développement durable ».

LEDBAT n'est donc pas un protocole complet, mais un algorithme de contrôle de la **fenêtre de congestion**, ce mécanisme par lequel les protocoles de transport évitent de saturer le réseau. Le plus connu et le plus utilisé de ces mécanismes est celui de TCP (RFC 5681 [1]) et ses objectifs sont d'utiliser le réseau à fond et d'assurer une relative égalité entre les flots de données qui se concurrencent sur ce réseau. LEDBAT, au contraire, vise avant tout à **céder** la place aux autre flots, non-LEDBAT.

Mais pourquoi diable voudrait-on être si généreux ? Cela peut être parce qu'on estime les autres flots plus importants : si je télécharge Plus belle la vie pendant que je passe un coup de téléphone via SIP, je souhaite que le téléchargement ne prenne pas de capacité si SIP en a besoin (c'est la différence entre applications d'« arrière-plan » comme le transfert de gros fichiers et d'« avant-plan » comme un coup de téléphone ou une session SSH interactive). Ou bien cela peut être pour profiter de réductions offertes par le réseau : après tout, un routeur ou une fibre optique ne coûtent pas plus cher à l'usage, que les octets circulent ou pas (contrairement à un autoroute ou une voie ferrée). Il serait donc logique que les transports « charognards », comme LEDBAT, qui n'utilisent la capacité réseau que lorsque personne

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc5681.txt

n'en veut, reçoivent une récompense financière, par exemple une réduction des prix (parlez-en à votre FAI).

Pour les détails sur les motivations de LEDBAT et les raisons pour lesquelles des technqiues comme le "*shaping*" ne conviennent pas, voir le premier RFC du groupe LEDBAT, le RFC 6297. Ici, je vais me focaliser sur l'algorithme spécifié par LEDBAT et qui répond au cahier des charges : céder la place le plus vite possible.

Le principe de cet algorithme est simple : utiliser les variations du temps de voyage des paquets pour détecter l'approche de la congestion et refermer alors la fenêtre de transmission. TCP utilise essentiellement le taux de pertes de paquets comme indicateur (ou les marques ECN du RFC 3168). Les routeurs ayant des tampons d'entrée-sortie, lorsque la ligne de sortie est saturée, les paquets commencent à s'entasser dans ce tampon. Lorsqu'il est plein, le routeur jette des paquets (et TCP va alors réagir). On voit que l'augmentation du temps de voyage (dû au séjour dans le tampon) **précède** la perte de paquets. En réagissant dès cette augmentation, LEDBAT atteint son objectif de céder la place à TCP. (À noter qu'il existe des variantes de TCP qui utilisent également le temps de voyage comme indicateur de l'approche de la congestion, par exemple TCP Vegas, documenté dans « "*TCP Vegas : New techniques for congestion detection and avoidance*" `<http://www.cs.umd.edu/class/spring2010/cmsc711/vegas.pdf>` » de Brakmo, L., O'Malley, S., et L. Peterson, mais voir le RFC 6297 pour un tour d'horizon général.)

Où est-ce que LEDBAT va être mis en œuvre ? Cela peut être dans un protocole de transport, par exemple comme une extension de TCP, ou bien dans l'application. LEDBAT est un algorithme, pas un protocole précis. Il peut être utilisé dans plusieurs protocoles, du moment que ceux-ci permettent l'estampillage temporel des paquets, pour que les deux machines qui communiquent puissent mesurer le temps de voyage (section 4.1).

La section 2 décrit l'algorithme exact. LEDBAT a une fenêtre de congestion, notée `cwnd` qui indique combien d'octets l'émetteur peut envoyer avant un nouvel accusé de réception. L'émetteur met dans chaque paquet le moment où il a envoyé ce paquet. Le récepteur regarde l'heure, en déduit le temps de voyage (aller simple, puisque l'encombrement n'est pas forcément le même dans les deux sens, une mesure aller-retour ne servirait pas à grand'chose) et retransmet cette indication à l'émetteur. Lorsque celui-ci voit le temps de voyage augmenter (signe que les tampons des routeurs se remplissent), il diminue la fenêtre de congestion. L'émetteur notamment utilise deux paramètres, `TARGET` et `GAIN`. `TARGET` est l'augmentation du temps de voyage en dessous de laquelle LEDBAT ne fait rien. `GAIN` (qui vaut entre 0 et 1) indique le facteur d'échelle entre l'augmentation du temps de voyage et la réduction de la fenêtre de congestion. Plus il est élevé, plus LEDBAT réagit rapidement et vigoureusement. Un `GAIN` de 1 est équivalent à TCP Reno. À noter qu'on pourrait avoir deux valeurs de `GAIN`, une pour augmenter la fenêtre de congestion et une pour la diminuer. En mettant un `GAIN` plus grand pour la diminution de la fenêtre, on garantit que LEDBAT cédera très vite la place dès le plus petit signal d'un ralentissement. (Pour les amateurs de pseudo-code, une description de l'algorithme avec cette technique figure dans le RFC.)

Bien sûr, le temps de voyage varie pour tout un tas de raisons et il n'est pas forcément souhaitable de refermer la fenêtre de congestion à chaque dérapage. LEDBAT filtre donc les "*outliers*" à l'aide d'une fonction `FILTER()` qui fait partie des paramètres de l'algorithme (elle n'est pas imposée par le RFC, on peut tester plusieurs mécanismes de filtrage des données.) Un filtre sommaire est `NULL` (aucun filtrage, on accepte toutes les mesures). Un autre plus sophistiqué est `EWMA` ("*Exponentially-Weighted Moving Average*"). Un bon filtre résiste au bruit inévitable, mais reste sensible aux signaux indiquant une vraie congestion.

Le temps de voyage se décompose en temps de transmission sur le câble (qui dépend de la vitesse du médium, par exemple 100 Mb/s pour du Fast Ethernet), temps de propagation (lié à la vitesse de la

lumière), temps d'attente dans les tampons et temps de traitement à la destination. Tous ces temps sont constants ou presque, à l'exception du temps d'attente dans les tampons. Ce sont donc ses variations qui déterminent l'essentiel des variations du temps de voyage. Pour estimer ce temps d'attente dans les tampons, LEDBAT calcule un **temps de base** (section 3.1.1) qui est le temps de voyage minimum observé. Il permet de connaître la valeur minimale en dessous de laquelle le temps de voyage ne pourra pas descendre. Les calculs de fenêtre de congestion se font à partir de (temps de voyage - temps de base), une grandeur qui part donc de zéro (les méthodes du RFC 6298 peuvent être utiles ici). L'algorithme de LEDBAT est linéaire : la fenêtre est réduite ou agrandie proportionnellement à cette grandeur, le facteur de proportionnalité étant le paramètre `GAIN`. Autre intérêt du concept de temps de base : en cas de changement de la route pendant la session (par exemple, panne d'un lien et re-routage par un chemin plus long), la mesure du temps de base pendant les N dernières secondes permettra de voir que le trajet a changé et que les calculs doivent utiliser le nouveau temps de base. (Le choix de N est un compromis : trop petit et le temps de base va varier souvent, trop grand et il retardera le moment où on détecte un changement de chemin.)

Avec son mécanisme de réduction de la fenêtre de congestion dès que le temps de voyage augmente, LEDBAT ne pousse normalement jamais jusqu'au point où il y aura des pertes de paquets. Néanmoins, celles-ci peuvent quand même survenir, et LEDBAT doit alors se comporter comme TCP, en refermant la fenêtre (section 3.2.2).

La section 4 du RFC discute ensuite de différents points intéressants de LEDBAT. Par exemple, LEDBAT est efficace pour céder rapidement à TCP lorsque celui-ci transfère de grandes quantités de données. Mais cela laisse le problème des applications qui envoient peu de données mais sont très sensibles à la latence, comme la voix sur IP. Si la conversation téléphonique envoie peu de données, il n'y aura jamais de remplissage des tampons, donc pas d'augmentation du temps d'attente dans ceux-ci donc LEDBAT se croira tranquille et enverra autant de données qu'il veut. Au moment où un des interlocuteurs parlera, ses paquets se trouveront donc peut-être coincés derrière un paquet LEDBAT. La seule protection contre ce problème est le paramètre `TARGET` qui ne doit donc pas être mis trop haut. La norme G.114 de l'UIT suggère 150 ms comme étant le maximum de retard tolérable pour le transport de la voix. LEDBAT doit donc choisir ses paramètres pour réagir avant les 150 ms. Le RFC recommande 100 ms pour le paramètre `TARGET` qui indique l'augmentation de délai de voyage à partir de laquelle LEDBAT réagit. (Ce paramètre `TARGET` détermine le temps maximal d'attente supplémentaire dû à LEDBAT.)

Autre point subtil, la compétition entre flots LEDBAT. On sait que TCP assure la « justice » entre flots TCP : si trois flots sont en compétition, chacun aura un tiers de la capacité. LEDBAT, lui, cède à TCP. Si un flot LEDBAT et un flot TCP sont en compétition, TCP aura toute la capacité. Mais si deux flots LEDBAT parallèles se concurrencent ? L'algorithme de LEDBAT ne garantit **pas** de justice. En général, c'est le flot le plus récent qui va gagner : arrivant tard, dans un réseau déjà bien encombré, il mesure un temps de base très élevé et ne voit donc pas d'augmentation due au temps d'attente dans les tampons, et ne réduit donc pas sa fenêtre de congestion (« "*Rethinking Low Extra Delay Background Transport Protocols*" `<http://arxiv.org/abs/1010.5623>` » de Carofiglio, G., Muscariello, L., Rossi, D., Testa, C., et S. Valenti). Pour corriger cet effet, on ne peut compter que sur le bruit : de temps en temps, les tampons se videront, le temps de voyage diminuera, et le récent arrivé corrigera sa mauvaise estimation du temps de base.

La section 5 du RFC considère les points qui sont encore ouverts à expérimentation. Après tout, un protocole comme LEDBAT est quelque chose de très nouveau dans le zoo de l'Internet. Par exemple, l'effet de changement de routes pendant une session LEDBAT, modifiant le temps de base, n'est pas encore vraiment bien connu. Quant à la valeur des paramètres comme `TARGET` ou `GAIN`, elle aura certainement besoin d'être ajustée à la lumière de l'utilisation réelle. Enfin, le filtre des mesures, qui permet d'éliminer les mesures jugées anormales, et évite donc à LEDBAT d'ajuster brusquement sa fenêtre de congestion pour rien, aura certainement besoin de réglages, lui aussi.

Et la sécurité de LEDBAT ? La section 6 rappelle que, même si un attaquant arrive à tromper LEDBAT sur les valeurs du temps de voyage, dans le pire des cas, il ne pourra pas le faire se comporter de manière plus gloutonne que TCP. Par contre, il pourra faire une attaque par déni de service en lui faisant croire que le délai de voyage a augmenté et que LEDBAT devrait ralentir. Pour l'instant, il n'y a pas de mécanisme contre cela.

Le bon fonctionnement de LEDBAT dépend de bonnes mesures. Les fanas de métrologie seront donc ravis de l'annexe A, qui parle des causes d'erreur dans les mesures. Le principal problème est que, pour mesurer les temps de voyage aller-simple dont a besoin LEDBAT, il faut des horloges à peu près synchronisées. Si l'émetteur met dans le paquet une heure de départ à 13 heures, 37 minutes et 56,789 secondes, que le récepteur mesure une arrivée à 13 heures, 37 minutes et 57,123 secondes et que le récepteur a 0,5 secondes de retard sur l'émetteur, il mesurera un temps de voyage de 0,334 secondes (alors qu'il est en fait de 0,834 secondes). Reprenant le vocabulaire de NTP (RFC 5905), on peut dire qu'il y a deux sources de problèmes, l'**écart** des horloges par rapport à une référence et le **décalage** (la variation de l'écart). L'écart entraine une erreur fixe dans la mesure du temps (comme dans notre exemple ci-dessus). Mais LEDBAT n'utilise pas directement le temps mais la différence entre temps actuel et temps de base. Cette erreur s'annule donc et l'écart des horloges par rapport au temps correct n'est donc pas importante pour LEDBAT.

Plus embêtant est le décalage puisque lui ne s'annulera pas. Si l'horloge du récepteur bat plus vite que celle de l'émetteur, il aura toujours l'impression d'un réseau très encombré, puisque ses mesures de temps de voyage seront toujours supérieures au temps de base qu'il avait mesuré avant. Heureusement, alors que des écarts énormes sont souvent vus sur l'Internet (il est fréquent de voir plusieurs minutes, voire plusieurs heures de différence entre une machine et le temps UTC), les décalages sont typiquement bien plus petits. Les mesures citées par le RFC 5905 indiquent des décalages courants de 100 à 200 ppm, soit 6 à 12 ms d'erreur accumulée par minute. Comme une machine LEDBAT limite sa mémoire (paramètre `BASE_HISTORY`, pour lequel le RFC recommande actuellement une valeur de dix minutes), et n'utilise donc que des mesures récentes pour évaluer le temps de base, le problème reste donc limité.

Si malgré tout, le problème se pose, il n'affectera que le cas où le récepteur a une horloge plus rapide, et en déduira donc à tort qu'il doit ralentir. Dans le cas inverse (horloge du récepteur plus lente), l'émetteur aura simplement l'impression que le temps de base augmente.

Pour corriger ce problème de décalage, on peut imaginer d'envoyer les estampilles temporelles dans les deux sens, pour que chaque machine puisse calculer ce que devrait voir l'autre, pour pouvoir détecter qu'un pair a une mauvaise mesure. Ensuite, l'émetteur pourrait corriger ses propres calculs por s'adapter à ce récepteur erroné. Il peut même calculer si le décalage est constant (horloge battant trop vite ou trop lentement, mais à une fréquence constante).

Et question mises en œuvre effectives ? LEDBAT a été testé dans des logiciels comme [Caractère Unicode non montré [2]]Torrent Transport Protocol library <http://github.com/bittorrent/libutp>. Il est l'algorithme utilisé par le microTP de BitTorrent (voir la documentation officielle de μTP <http://www.utorrent.com/help/documentation/utp>). Il est aussi mise en œuvre dans Swift <http://libswift.org/>.

Merci à Mathieu Goessens pour sa suggestion et à André Sintzoff pour ses inlassables corrections.

2. Car trop difficile à faire afficher par LaTeX