RFC 9616 : Delay-based Metric Extension for the Babel Routing Protocol

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 14 septembre 2024

Date de publication du RFC : Septembre 2024

https://www.bortzmeyer.org/9616.html

Voici un RFC sur le protocole de routage Babel. Il normalise une extension au protocole pour tenir compte du RTT, le temps d'aller-retour sur un segment du réseau. C'est l'occasion de revenir sur ce critère de routage, souvent cité mais rarement utilisé (et pour de bonnes raisons).

On lit parfois des affirmations rapides comme « les routeurs Internet choisissent la route la plus rapide ». Outre que « rapide » n'a pas de définition précise (parle-t-on de la latence <https://www.bortzmeyer.org/latence.html> ? De la capacité <https://www.bortzmeyer.org/capacite.html> ?), la phrase est fausse : le routage se fait sur des critères nettement moins dynamiques. Dans le cas de BGP, ce sont en bonne partie des critères de business. Das le cas d'un IGP, des critères de performance sont pris en compte, mais en se limitant à des critères assez statiques. En effet, si on prend en compte naïvement une variable très dynamique, qui change souvent, comme l'est le RTT, on risque d'avoir un routage très instable : un lien est peu utilisé, ses performances sont bonnes, les routeurs vont y envoyer tout le trafic, ses performances vont chuter, les routeurs vont tout envoyer ailleurs, les performances vont remonter, les routeurs vont encore changer d'avis, etc. Tenir compte des performances immédiates du lien est donc une fausse bonne idée ou, plus précisément, ne doit pas être fait naïvement.

C'est justement ce que fait ce RFC qui explique comment utiliser intelligemment un critère très mouvant, le RTT (la latence, mais mesurée en aller-retour), pour le choix des routes dans Babel. Babel est un protocole de routage de type IGP, normalisé dans le RFC 8966 [1], surtout prévu pour des réseaux sans administration centrale.

Si par exemple (section 1 du RFC), une machine A à Paris a deux moyens de joindre une machine D également à Paris, une des routes passant par un routeur B à Paris et une autre par un routeur C à Tokyo,

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc8966.txt

il est assez évident qu'il vaut mieux rester à Paris. Mais ce n'est pas forcément ce que feront les mises en œuvre de Babel, qui se disent simplement que les deux routes ont le même nombre de segments (deux). Bien sûr, attribuer manuellement des préférences aux segments résoudrait le problème (un lien vers le Japon est plus « cher ») mais l'un des buts de Babel est de ne pas requérir de configuration manuelle par un administrateur réseaux.

Utiliser le RTT, qui est relativement facile à évaluer (mais continuez la lecture : il y a des pièges) semble une solution évidente. Mais, comme dit plus haut, cela peut mener à de violentes instabilités du routage. Il faut donc évaluer le RTT, puis le traiter avant de l'utiliser, pour éviter des oscillations du routage. Et attention, la latence n'est pas toujours le meilleur critère de choix, d'autres, comme le coût monétaire, peuvent être pris en compte, par exemple pour privilégier un lien WiFi sur un lien 5G. Babel n'impose pas de critère unique de choix des routes.

Cette extension ne nécessite pas que les différents routeurs aient une horloge synchronisée (ce qui serait difficile à faire dans un réseau sans administration), juste que les horloges ne dérivent pas trop vite l'une par rapport à l'autre. D'autre part, comme l'algorithme essaie de limiter les oscillations de routes, il ne s'ajustera pas instantanément aux changements, et ne sera donc pas optimal dans tous les cas.

Commençons par l'évaluation du RTT. Il ne suffit pas de soustraire le temps de départ d'une question au temps d'arrivée d'une réponse, entre autre parce que la génération de la réponse peut prendre du temps (et puis un routeur Babel n'est pas obligé de répondre à tous les messages Hello). Et rappelez-vous que cette extension n'impose pas une synchronisation des horloges. L'algorithme est donc un peu plus compliqué, c'est celui créé par Mills et utilisé entre autres pour NTP. Un routeur nommé Alice qui envoie un message Hello (RFC 8966, section 3.4.1) ajoute à ces messages le temps de départ, appelé t1. Le routeur nommé Bob le reçoit à un temps t1'. Comme les horloges ne sont pas forcément synchronisées, on ne peut pas comparer directement t1 et t1'. Au lieu de cela, Bob, quand il enverra un message Babel IHU ("*I Heard yoU*") indiquera à la fois t1 et t1' dans ce message, ainsi que le temps t2' d'émission. Alice, recevant ce message, n'aura qu'à calculer (t2 - t1) - (t2' - t1') et aura ainsi le RTT, même si les horloges sont différentes (tant qu'elles ne dérivent pas trop dans l'intervalle entre t1 et t2). Bon, j'ai simplifié, mais vous avez tous les détails dans le RFC, section 3.2.

Notez que cet algorithme impose aux deux routeurs de garder des informations supplémentaires sur leurs voisins, dans la table documentée dans la section 3.2.4 du RFC 8966. Celle-ci devra stocker les estampilles temporelles reçues.

Ces estampilles étant stockées sur 32 bits et ayant une résolution en microsecondes, elles reviendront à zéro toutes les 71 minutes. Le routeur doit donc prendre garde à ignorer les estampilles situées dans son futur (par exemple s'il a redémarré et perdu la notion du temps) ou trop éloignées. Sur un système POSIX, le routeur peut donc utiliser `clock_gettime(CLOCK_MONOTONIC)`.

Bon, désormais, nous avons le RTT. Qu'en faire ? Comme indiqué plus haut, il ne faut pas l'utiliser tel quel comme critère de sélection des routes, sous peine d'osccilations importantes des tables de routage. Il y a plusieurs opérations à faire (section 4). D'abord, il faut lisser le RTT, éliminant ainsi les cas extrêmes. Ainsi, on va utiliser la formule RTT [Caractère Unicode non montré [2]] [Caractère Unicode non montré] RTT + (1 - [Caractère Unicode non montré]) RTTn, où RTTn désigne la mesure qu'on vient juste de faire, et où la constante [Caractère Unicode non montré] a une valeur recommandée de 0,836. (Les détails figurent dans l'article « "*A delay-based routing metric*" <http://arxiv.org/abs/1403.3488> ».)

2. Car trop difficile à faire afficher par LaTeX

Ensuite, pour nourrir l'algorithme de choix des routes, il faut convertir ce RTT en un coût. La fonction est simple : de 0 à une valeur rtt-min, le coût est constant, il augmente ensuite linéairement vers une valeur maximale, atteinte pour un RTT égal à rtt-max, et constante ensuite. Les valeurs par défaut recommandées sont de 10 ms pour rtt-min et 150 ms pour rtt-max. En d'autres termes, sur l'Internet, tout RTT inférieur à 10 ms est bon, tout RTT supérieur à 150 ms est à éviter autant que possible.

Enfin, car il reste encore des variations, la section 4 sur les traitements du RTT se termine en demandant l'application d'un mécanisme d'hystérésis, comme celui du RFC 8966, annexe A.3.

Le format des paquets, maintenant. Rappelons que Babel encode les données en TLV et permet des sous-TLV (la valeur d'un TLV est elle-même un TLV). Un routeur doit ignorer les sous-TLV qu'il ne comprend pas, ce qui permet d'ajouter des nouvelles informations sans risque (section 5).

Donc, notre RFC normalise le sous-TLV "*Timestamp*" (type 3 <https://www.iana.org/assignments/babel/babel.xml#sub-tlv-types>), qui permet de stocker les estampilles temporelles, et peut apparaitre sous les sous-TLV Hello et IHU. Dans le premier cas, il stockera juste une valeur (t1 ou t2' dans l'exemple plus haut), dans le second, deux valeurs (t1 et t1' dans l'exemple plus haut).

La mise en œuvre « officielle » de Babel a, dans sa version de développement <https://github.com/jech/babeld>, la capacité de déterminer ces RTT (option `enable-timestamps` dans la configuration).