RFC 6556 : Testing Eyeball Happiness

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 7 avril 2012

Date de publication du RFC : Avril 2012

https://www.bortzmeyer.org/6556.html

Le problème du bonheur des globes oculaires <https://www.bortzmeyer.org/globes-oculaires-heureux.html> (*"eyeball happiness"*) fait l'objet de recherches actives en ce moment. Il s'agit de s'assurer que, lorsqu'un service Internet (par exemple un site Web) est accessible à la fois en IPv4 et en IPv6, l'utilisateur (le propriétaire des globes oculaires) sera heureux (il verra le site très vite), que sa connectivité IPv6 fonctionne correctement ou pas. L'incertitude à ce sujet est la principale cause du très faible nombre de sites Web qui ont un enregistrement AAAA (adresse IPv6) dans le DNS aujourd'hui.

Ce RFC ne propose pas de solution : issu de travaux menés notamment au sein du groupe de travail benchmarks <http://tools.ietf.org/wg/bmwg> de l'IETF, il propose simplement de **mesurer** ce bonheur des globes oculaires, de manière à ce qu'on puisse comparer les solutions. On pourra alors retenir les meilleures, de manière à faire sauter ce qui est aujourd'hui un sérieux obstacle au déploiement d'IPv6.

La section 1 du RFC résume le problème de base : si on a une connectivité où plusieurs préfixes IP sont possibles (un IPv4 et un IPv6, ou bien plusieurs IPv6), et que l'application choisit comme source le « mauvais » (celui qui a la moins bonne connectivité, voire pas de connectivité du tout), quelles seront les conséquences pratiques ? L'établissement de la session en sera-t-il ralenti ? Comment TCP va-t-il réagir (RFC 5461 [1]) ?

L'algorithme naïf pour l'application est de récupérer une liste d'adresses possibles pour la destination, via `getaddrinfo()`, et d'essayer chaque adresse successivement. Si certaines *"timeoutent"*, le temps total d'établissement de la connexion (attendre l'expiration du délai de garde, puis essayer l'adresse suivante) va largement dépasser les capacités de l'utilisateur même le plus patient (cela peut

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc5461.txt

dépasser la minute, la valeur exacte dépend de nombreux paramètres). Les globes oculaires seront malheureux.

C'est à cause de cela que, alors que plus de 40 % des zones dans `.fr` ont une adresse IPv6 pour leur service DNS (mesures faites en janvier 2012), seules moins d'1 % en ont une pour leur service Web. C'est parce que le DNS a été prévu pour résister aux pannes dès le début (le résolveur se souvient de quelles adresses marchent, pour une zone) et que l'ajout d'un AAAA cassé n'a pas de conséquences graves. Au contraire, le Web n'a pas de mécanisme natif pour assurer la haute disponibilité et un AAAA cassé se paie très cher en temps d'attente pour les clients. Beaucoup de gérants de sites Web préfèrent donc ne pas courir le risque, même s'il ne frapperait qu'une petite minorité de leurs clients.

L'idée de ce RFC, présentée en section 2, est de quantifier ce problème. Le but est d'aboutir à des chiffres indiquant, pour un système donné (nommé Alice), le temps qu'il met à ouvrir une session avec un service (nommé Bob) dont certaines adresses sont incorrectes. Plusieurs cas d'incorrection peuvent se présenter :

— Un routeur jette les paquets mais transmet un message ICMP "*administratively prohibited*" ou bien "*destination unreachable*",
— Un routeur jette silencieusement les paquets (cible `DROP` dans iptables par exemple),
— Un programme intermédiaire génère un `RST` ("*reset*") TCP,
— Un problème de MTU fait que les paquets au delà d'une certaine taille ne sont pas transmis. Normalement, la découverte de la MTU du chemin (RFC 1191, RFC 1981) fait que ce problème ne doit jamais se produire. En pratique, des configurations incorrectes (blocage aveugle de tout ICMP) font qu'il arrive quand même.

Le test doit être fait pour tous ces cas car rien ne dit que l'application réagira de la même façon à tous ces problèmes. Le plus important est sans doute celui de la perte silencieuse des paquets, car c'est le pire : si l'application réagit bien même dans ce cas, elle s'en tirera sans doute dans les autres. Si on n'en teste qu'un, cela doit donc être celui-ci. (C'est ce que dit le RFC. En pratique, le cas des problèmes de MTU est souvent aussi grave, et il ne se détecte qu'**après** l'ouverture de la session, lorsqu'on commence à envoyer de gros paquets.)

La section 2.2 explique la procédure, notamment le fait qu'on intègre le temps de la requête DNS préalable (ce que je trouve contestable, mais les raisons ne sont pas expliquées). Les métriques elles-mêmes sont en section 2.3. On mesure le temps entre la première requête DNS et la fin de l'établissement de la connexion TCP, qui est le moment où l'application peut commencer à travailler. Cela se nomme le "*Session Setup Interval*". Si on garde les temps minimum (meilleur cas, probablement celui où la première tentative réussie) et maximum (pire cas, lorsqu'il aura fallu essayer toutes les adresses de la cible), on les nomme "*Minimum Session Setup Interval*" et "*Maximum Session Setup Interval*". Le reste du RFC est composé de détails concrets pour qui voudrait effectuer cette mesure. Telle qu'elle est définie (avec des notions comme « l'arrivée du dernier but »), elle n'est pas triviale, voire pas possible à mettre en œuvre sur une machine Unix normale.

Il existe aussi une métrique « qualitative », où on décrit le trafic observé ("*Attempt Pattern*"). Cela peut être simple (« Un paquet TCP `SYN` est envoyé successivement à toutes les adresses, à des intervalles de X milli-secondes ») mais aussi très complexe. Mon article sur les algorithmes `<https://www.bortzmeyer.org/globes-oculaires-heureux.html>` en donne quelques exemples.