RFC 5482 : TCP User Timeout Option

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 17 mars 2009

Date de publication du RFC : Mars 2009

http://www.bortzmeyer.org/5482.html

Le RFC 793 [1], qui normalise TCP, prévoit un paramètre local "*user timeout*" qui permet de régler la « patience » de TCP. S'il n'arrive pas à échanger des données avec son partenaire avant l'expiration de ce délai, TCP renonce et coupe la connexion. Notre RFC étend l'utilité de ce paramètre, en permettant de le communiquer au partenaire, pour qu'il soit prévenu de la patience ou de l'impatience de celui avec qui il communique. On augmente ainsi les chances des connexion TCP de survivre aux longues coupures ou, au contraire, de détecter plus rapidement la coupure du réseau.

Ce paramètre "*user timeout*" peut être spécifié par l'application qui utilise TCP (ceci-dit, je ne trouve pas l'option de `setsockopt` qui permettrait de le faire sur tout Unix ; d'après le RFC, Solaris a une telle option). Il vaut cinq minutes par défaut (section 3.8 du RFC 793). Mais il n'était jamais connu de l'autre machine avec laquelle on est connecté en TCP. C'est ce que change notre RFC qui permet de communiquer ce paramètre.

Cette communication se fait via le mécanisme d'**option** de TCP, décrit dans la section 3.1 du RFC 793. La nouvelle option porte le nom d'UTO ("*User Timeout Option*") et est strictement indicative : le TCP qui la reçoit peut parfaitement l'ignorer. S'il l'accepte, il va ajuster son propre "*user timeout*" à la valeur indiquée. Ainsi, TCP pourra, par exemple, survivre à de longues périodes de déconnexion, sans que le partenaire ne raccroche.

La section 1 fournit d'autres cas où cette option UTO est utile : pour des mobiles utilisant une technique qui conserve l'identificateur comme HIP (RFC 4423) ou Mobile-IP (RFC 3344), les périodes pendant lesquelles le mobile n'a pas de connexion risquent d'être longues. Le fait de garder l'identificateur (adresse IP ou "*Host Identity*") constant permet en théorie de garder les sessions TCP lorsqu'on se

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc793.txt`

déconnecte et se reconnecte mais, avant l'option UTO, cela ne servait à rien puisque TCP coupait la connexion qui lui semblait en panne.

De même, l'application peut dans certains cas connaître la durée d'une déconnexion et utiliser UTO pour prévenir son partenaire (le RFC cite l'exemple amusant d'une sonde spatiale qui sait, grâce aux lois de la mécanique céleste, qu'elle ne sera pas en vue de sa base pendant une longue période).

La section 3 définit rigoureusement la nouvelle option. Quatre paramètres définissent le comportement du TCP local : `USER_TIMEOUT`, qui existe déjà (mais n'est pas réglable sur tous les systèmes d'exploitation, cf. sections 3.1 et 5), `ADV_UTO`, la valeur de l'UTO communiquée au pair (typiquement identique à `USER_TIMEOUT`), `ENABLED` qui indique si le pair doit être notifié par la nouvelle option UTO et `CHANGEABLE` qui indique si `USER_TIMEOUT` doit être ajusté en fonction des notifications du pair (ou bien s'il doit être ignoré, ce qui est permis).

La section 3.1 contient une discussion détaillée des « bonnes » valeurs pour `USER_TIMEOUT` (notons que ce ne sera pas forcément le même pour les deux pâirs). Le RFC 1122 donnait quelques recommandations à ce sujet (par exemple, les valeurs inférieures à 100 s sont découragées). La 3.2 discute de la fiabilité de la transmisson d'UTO (inexistante, comme toutes les options TCP ; il ne faut donc pas tenir pour acquis qu'elle a été reçue par le pair). Et 3.3 décrit le format concret de l'option : numéro 28 (enregistré dans le registre IANA <https://www.iana.org/assignments/tcp-parameters>), un bit qui indique l'unité utilisée (minutes ou secondes) et 15 bits qui contiennent la valeur elle-même.

La section 4 passe aux problèmes pratiques que peut rencontrer le déploiement de l'option. Le RFC rappelle en section 3 que l'espace des options dans TCP est limité (40 octets) et que l'ajout d'UTO peut nécessiter de renoncer à certaines options. Il faut aussi dire que beaucoup de pare-feux bogués rejettent tout paquet TCP ayant des options (cf. "*Probing the viability of TCP extensions*" <https://www.imperialviolet.org/binary/ecntest.pdf>) et que celles-ci sont donc difficiles à utiliser en pratique. En théorie, une implémentation de TCP qui ne connait pas cette option devrait l'ignorer (RFC 1122) et le RFC décrète que, bien qu'il y aie quelques violations de cette règle (cf. "*Measuring interactions between transport protocols and middleboxes*" <http://www.imconf.net/imc-2004/papers/p336-medina.pdf>), elles ne méritent pas qu'on renonce à une option utile (section 4.1). Notez aussi qu'un pare-feu à état a typiquement ses propres délais de garde et peut donc couper une connexion sans tenir compte de l'UTO annoncée (les pare-feux futurs tiendront peut-être compte de cette option).

J'ai déjà mentionné la difficulté qu'il y a à changer la valeur de `USER_TIMEOUT`. La section 5 revient sur cette question de l'API mais ne fournit pas de proposition de normalisation de l'API des prises.

Bien sûr, l'UTO introduit des problèmes de sécurité nouveaux, discutés en section 6. Par exemple, un méchant pourrait spécifier de très longs délais de garde, puis s'en aller pendant que sa victime serait forcée de garder une connexion ouverte pendant de longues minutes, malgré l'absence de trafic. Notre RFC propose des mécanismes pour limiter les risques comme n'accepter les UTO qu'après authentification IP (par exemple par IPsec), ou bien seulement après que l'application aie indiqué à TCP qu'elle avait réussi une authentification à son niveau, par exemple par TLS. Sans authentification, TCP pourrait aussi mettre des limites au délai de garde total d'un pair (s'il a ouvert plusieurs connexions) et au délai de garde total (pour limiter les risques dûs aux dDoS).

Je ne trouve pas encore de trace de mises en œuvre de TCP qui incluent cette option UTO.