RFC 8201 : Path MTU Discovery for IP version 6

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 16 juillet 2017

Date de publication du RFC : Juillet 2017

http://www.bortzmeyer.org/8201.html

Ce RFC est l'adaptation à IPv6 du protocole décrit dans le RFC 1191[1], protocole qui permet de découvrir la MTU du chemin entre deux machines reliées par Internet. Il remplace le RFC 1981.

Avec IPv4, déterminer la MTU maximale du chemin est très utile pour optimiser les performances. Mais elle devient presque indispensable en IPv6, où les routeurs n'ont pas le droit de fragmenter les paquets trop gros (toute fragmentation doit être faite dans la machine de départ). PMTU fait donc quasi-obligatoirement partie d'IPv6. Une alternative est de n'envoyer que des paquets suffisamment petits pour passer partout <http://www.bortzmeyer.org/fragmentation-ip-1280.html>. C'est moins efficace (on ne tirera pas profit des MTU plus importantes), mais ça peut être intéressant pour des mises en œuvre légères d'IPv6, par exemple pour des objets connectés. (La section 4 du RFC précise en effet que mettre en œuvre cette découverte de la MTU du chemin n'est **pas** obligatoire.)

L'algorithme (section 3 de notre RFC) est le même qu'en v4, envoyer des paquets, et voir si on reçoit des paquets ICMP "*Packet too big*" (RFC 4443, section 3.2) qui contiennent en général la MTU maximale du lien suivant. (Pas de bit DF - "*Don't fragment*" - en IPv6 puisque la fragmentation n'est possible qu'à la source. L'annexe A détaille les différences avec l'algorithme IPv4 décrit dans le RFC 1191.)

Voici par exemple un de ces paquets ICMP, vu par tcpdump (déclenché par une commande `ping -s 1500 ...`):

```
18:18:44.813927 IP6 2a02:8400:0:3::a > 2605:4500:2:245b::bad:dcaf: ICMP6, packet too big, mtu 1450, length 1240
```

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc1191.txt

Il ne faut pas s'arrêter au premier paquet "*Packet too big*" reçu, car il peut y avoir des liens réseau avec une MTU encore plus basse plus loin. En outre, la MTU du chemin change avec le temps et il faut donc rester vigilant, on peut recevoir des "*Packet too big*" si le trafic passe désormais par un lien de MTU plus faible que la précédente MTU du chemin. Plus délicat, il faut aussi de temps en temps réessayer avec des paquets plus gros, pour voir si la MTU du chemin n'aurait pas augmenté. Comme cela va faire perdre des paquets, s'il n'y a pas eu d'augmentation, il ne faut pas le faire trop souvent. (Dix minutes, dit la section 5.)

La section 4 du RFC précise les exigences qu'il normalise. (Je donne quelques exemples de leur mise en œuvre dans le noyau Linux, dans `net/ipv6/route.c`.) Par exemple, le nœud IPv6 **doit** vérifier le contenu des paquets ICMP "*Packet Too Big*" qu'il reçoit (le "*Message Body*", cf. RFC 4443, où le routeur émetteur est censé placer des données du paquet original), pour éviter de croire des faux messages ICMP envoyés par un attaquant (rien n'authentifie un paquet ICMP, à part son contenu).

Nouveauté de ce RFC, par rapport à son prédecesseur, un nœud IPv6 **doit** ignorer les messages "*Packet Too Big*" indiquant une MTU inférieur aux 1 280 octets minimum d'IPv6 (le RFC 8021 explique pourquoi). C'est le but de l'instruction `mtu = max_t(u32, mtu, IPV6_MIN_MTU);` dans le code du noyau Linux, qui ne garde pas la MTU indiquée si elle est inférieur au minimum.

Et la machine IPv6 ne doit jamais **augmenter** la MTU du chemin suite à la réception d'un "*Packet Too Big*" indiquant une MTU supérieure à l'actuelle. Un tel message ne peut être qu'une erreur ou une attaque. Voir les commentaires au début de la fonction `rt6_mtu_change_route` dans le code Linux.

Comme avec IPv4, l'algorithme PMTU marche mal en pratique car beaucoup de sites filtrent stupidement tous les paquets ICMP et la machine qui tente de faire de la "*Path MTU discovery*" n'aura jamais de réponse. IPv6, comme IPv4, devra donc plutôt utiliser la technique du RFC 4821.

La section 5 du RFC rassemble diverses questions de mise en œuvre de cette découverte de la MTU du chemin. D'abord, il y a l'interaction entre IP (couche 3) et les protocoles au dessus, comme TCP mais aussi des protocoles utilisant UDP, comme le DNS. Ces protocoles de transport ont une MMS_S ("*maximum send transport-message size*", voir le RFC 1122) qui est à l'origine la MTU moins la taille des en-têtes. Les protocoles de transport peuvent diminuer cette MMS_S pour ne pas avoir à faire de fragmentation.

Pour le DNS, par exemple, on peut diminuer l'usage de la découverte de la MTU du chemin en diminuant la taille EDNS. Par exemple, avec le serveur nsd, on mettrait :

```
ipv6-edns-size: 1460
```

Et, avec 1 460 octets possibles, on serait alors raisonnablement sûr de ne pas avoir de fragmentation donc, même si la découverte de la MTU du chemin marche mal, tout ira bien.

Une fois la découverte de la MTU du chemin faite, où faut-il garder l'information, afin d'éviter de recommencer cette découverte à chaque fois ? Idéalement, cette information est valable pour un chemin donné : si le chemin change (ou, a fortiori, si la destination change), il faudra découvrir à nouveau. Mais la machine finale ne connait pas en général tout le chemin suivi (on ne fait pas un traceroute à chaque connexion). Le RFC suggère donc (section 5.2) de stocker une MTU du chemin par couple {destination, interface}. Si on n'a qu'une seule interface réseau, ce qui est le cas de la plupart des machines terminales, on stocke la MTU du chemin par destination et on maintient donc un état pour chaque destination (même quand on ne lui parle qu'en UDP, qui n'a normalement pas d'état). Les deux exemples suivants concernent une destination où la MTU du chemin est de 1 450 octets. Pour afficher cette information sur Linux pour une adresse de destination :

```
% ip -6  route get 2a02:8428:46c:5801::4
2a02:8428:46c:5801::4 from :: via 2001:67c:1348:7::1 dev eth0 src 2001:67c:1348:7::86:133 metric 0
    cache  expires 366sec mtu 1450 pref medium
```

S'il n'y a pas de PMTU indiquée, c'est que le cache a expiré, réessayez (en UDP, car TCP se fie à la MSS). Notez que, tant qu'on n'a pas reçu un seul "*Packet Too Big*", il n'y a pas de MTU affichée. Donc, pour la plupart des destinations (celles joignables avec comme PMTU la MTU d'Ethernet), vous ne verrez rien. Sur FreeBSD, cet affichage peut se faire avec :

```
% sysctl -o net.inet.tcp.hostcache.list | fgrep 2a02:8428:46c:5801::4
2a02:8428:46c:5801::4  1450        0     23ms     34ms         0  4388        0 0   23    4 3600
```

Pour d'autres systèmes, je vous recommande cet article <https://njetwork.wordpress.com/2014/01/30/show-ipv6-destination-cache/>.

Notez qu'il existe des cas compliqués (ECMP) où cette approche de stockage de la PMTU par destination peut être insuffisante. Ce n'est donc qu'une suggestion, chaque implémentation d'IPv6 peut choisir sa façon de mémoriser les MTU des chemins.

Si on veut simplifier la mise en œuvre, et la consommation mémoire, on peut ne mémoriser qu'une seule MTU du chemin, la MTU minimale de tous les chemins testés. Ce n'est pas optimal, mais ça marchera.

Si on stocke la MTU du chemin, comme l'Internet peut changer, il faut se préparer à ce que l'information stockée devienne obsolète. Si elle est trop grande, on recevra des "*Packet too big*" (il ne faut donc pas s'attendre à n'en recevoir que pendant la découverte initiale). Si elle est trop petite, on va envoyer des paquets plus petits que ce qu'on pourrait se permettre. Le RFC suggère donc (section 5.3) de ne garder l'information sur la MTU du chemin que pendant une dizaine de minutes.

Le RFC demande enfin (section 5.5) que les mises en œuvre d'IPv6 permettent à l'ingénieur système de configurer la découverte de la MTU du chemin : la débrayer sur certaines interfaces, fixer à la main la MTU d'un chemin, etc.

Il ne faut pas oublier les problèmes de sécurité (section 6 du RFC) : les "*Packet Too Big*" ne sont pas authentifiés et un attaquant peut donc en générer pour transmettre une fausse information. Si la MTU indiquée est plus grande que la réalité, les paquets seront jetés. Si elle est plus petite, on aura des sous-performances.

Autre problème de sécurité, le risque que certains administrateurs réseau incompétents ne filtrent tous les messages ICMP, donc également les "*Packet Too Big*" (c'est stupide, mais cela arrive). Dans ce cas, la découverte de la MTU du chemin ne fonctionnera pas (cf. RFC 4890 sur les recommandations concernant le filtrage ICMP). L'émetteur devra alors se rabattre sur des techniques comme celle du RFC 4821.

L'annexe B de notre RFC résume les changements faits depuis le RFC 1981. Rien de radical, les principaux portent sur la sécurité (ce sont aussi ceux qui ont engendré le plus de discussions à l'IETF) :

— Bien expliquer les problèmes créés par le blocage des messages ICMP,
— Formulations plus strictes sur la validation des messages ICMP,
— Et surtout, suppression des « fragments atomiques » (RFC 8021).

Le texte a également subi une actualisation générale, les références (pré-)historiques à BSD 4.2 ou bien à TP4 ont été supprimées.

Enfin, pour faire joli, un exemple de traceroute avec recherche de la MTU du chemin activée (option `--mtu`). Notez la réception d'un "*Packet Too Big*" à l'étape 12, annonçant une MTU de 1 450 octets :

```
% sudo traceroute -6 --mtu --udp --port=53 eu.org
traceroute to eu.org (2a02:8428:46c:5801::4), 30 hops max, 65000 byte packets
 ...
 3  vl387-te2-6-paris1-rtr-021.noc.renater.fr (2001:660:300c:1002:0:131:0:2200)  2.605 ms  2.654 ms  2.507 r
 4  2001:660:7903:6000:1::4 (2001:660:7903:6000:1::4)  5.002 ms  4.484 ms  5.494 ms
 5  neuf-telecom.sfinx.tm.fr (2001:7f8:4e:2::178)  4.193 ms  3.682 ms  3.763 ms
 6  neuf-telecom.sfinx.tm.fr (2001:7f8:4e:2::178)  14.264 ms  11.927 ms  11.509 ms
 7  2a02-8400-0000-0003-0000-0000-0000-1006.rev.sfr.net (2a02:8400:0:3::1006)  12.760 ms  10.446 ms  11.902
 8  2a02-8400-0000-0003-0000-0000-0000-1c2e.rev.sfr.net (2a02:8400:0:3::1c2e)  10.524 ms  11.477 ms  11.896
 9  2a02-8400-0000-0003-0000-0000-0000-1c2e.rev.sfr.net (2a02:8400:0:3::1c2e)  13.061 ms  11.589 ms  11.915
10  2a02-8400-0000-0003-0000-0000-0000-117e.rev.sfr.net (2a02:8400:0:3::117e)  10.446 ms  10.420 ms  10.361
11  2a02-8400-0000-0003-0000-0000-0000-000a.rev.sfr.net (2a02:8400:0:3::a)  10.460 ms  10.453 ms  10.517 ms
12  2a02-8428-046c-5801-0000-0000-0000-00ff.rev.sfr.net (2a02:8428:46c:5801::ff)  12.495 ms F=1450  12.102 r
13  2a02-8428-046c-5801-0000-0000-0000-0004.rev.sfr.net (2a02:8428:46c:5801::4)  12.397 ms  12.217 ms  12.50
```

La question de la fragmentation en IPv6 a suscité d'innombrables articles. Vous pouvez commencer par celui de Geoff Huston <https://blog.apnic.net/2016/05/19/fragmenting-ipv6/>.

Merci à Pierre Beyssac pour avoir fourni le serveur de test configuré comme il fallait, et à Laurent Frigault pour la solution sur FreeBSD.