Régler les problèmes de MTU et de MSS

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 27 juin 2007. Dernière mise à jour le 3 septembre 2007

https://www.bortzmeyer.org/mtu-et-mss-sont-dans-un-reseau.html

---

Les liens Internet dont la MTU est plus faible que les traditionnels 1500 octets rencontrent souvent de nombreux problèmes, qui frappent particulièrement les transferts TCP. Quelles sont les solutions pratiques ?

Le lien « standard » sur Internet a une MTU de 1500 octets, héritée d'Ethernet. Si, de bout en bout, tous les liens ont cette MTU, la machine émettrice peut fabriquer des paquets de 1500 octets et ils arriveront intacts. Mais il y a souvent sur le trajet un lien dont la MTU est plus faible, par exemple un tunnel d'IPv6 dans IPv4 ou bien un tunnel GRE ou encore une connexion PPPoE sur de l'ADSL. On observe alors souvent un phénomène désagréable : les paquets de petite taille, tels que ceux fabriqués par ping ou traceroute passent mais les gros paquets, par exemple les transferts de fichier avec HTTP bloquent mystérieusement. Le débogage est donc difficile. Une bonne technique est de tester la connectivité, non pas avec `ping AUTRE-MACHINE` tout court mais avec `ping -s 1480 AUTRE-MACHINE` pour forcer l'usage de paquets de 1480 octets. Si le premier ping fonctionne mais pas le second, vous avez un problème de MTU.

Si tous les ping fonctionnent, ainsi que des essais avec UDP, mais que TCP n'y arrive pas, cela peut être parce que les implémentations d'IP ne mettent souvent le bit DF ("*Don't Fragment*", un bit dans l'en-tête IP qui indique aux routeurs de ne pas fragmenter le paquet) que pour TCP. Sur Linux, mettre l'option `net.ipv4.ip_no_pmtu_disc` du noyau à 1 pourrait permettre de tester cette hypothèse (je n'ai pas essayé).

En théorie, ces problèmes de MTU ne devraient jamais arriver. En IPv4, le routeur qui connecte au lien « trop étroit » devrait fragmenter les paquets, en IPv6, la découverte de la MTU du chemin, décrite dans le RFC 1981 [1], devrait empêcher la machine émettrice d'envoyer des paquets trop gros.

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc1981.txt`

(Cette découverte de la MTU du chemin peut aussi se faire en IPv4, cf. le RFC 1191, pour éviter la chute de performance provoquée par la fragmentation.)

Mais, en pratique, cela marche mal (le RFC 2923 a été le premier à expliquer ce problème, le RFC 4459 le détaillant ensuite et le RFC 4821 finissant par proposer d'abandonner l'ancien mécanisme et d'en utiliser un tout nouvau). En effet, la découverte de la MTU du chemin dépend de la bonne réception des messages ICMP "*Packet too big*". Ceux-ci sont malheureusement souvent filtrés par des coupe-feux mal configurés par des administrateurs ignorants qui croient améliorer la sécurité en bloquant tout ICMP. On constate alors qu'on peut transférer des fichiers avec certains sites et pas d'autres, selon la configuration des coupe-feux situés sur le chemin. (Un excellent article très complet sur la question est "*A Tale of Two Protocols : IPv4, IPv6, MTUs and Fragmentation*" <http://www.potaroo.net/ispcol/2009-01/mtu6.html>.)

Il existe alors plusieurs solutions. On peut se passer de tunnel, on peut tricher un petit peu pour remonter la MTU de PPPoE (le RFC 4638 explique comment), on peut attendre le déploiement de la nouvelle technique de découverte de MTU du chemin, normalisée dans le RFC 4821 mais il existe aussi des solutions plus raisonnables.

Une classique est est d'abaisser manuellement la MTU sur toutes les machines du réseau local (sur Unix, cela se fait avec la commande `ifconfig`). C'est pénible car il ne faut pas oublier de machine et cela abaisse la MTU (donc les performances) même pour les transferts purement locaux.

Cela peut se faire spécifiquement pour IPv6 si on utilise le RA ("*Router Advertisement*", autoconfiguration IPv6 sans état). Une option permet en effet de fixer la MTU du lien. Plus besoin alors de courir sur toutes les machines. Si on utilise le démon `radvd`, c'est l'option `AdvLinkMTU`. Par exemple :

```
interface eth0
{
...
   AdvLinkMTU 1460;
...
};
```

Une autre approche est de partir du fait que le problème touche surtout TCP, plus souvent impliqué dans les gros transferts et elle est donc spécifique à TCP (le test avec ping échouera donc toujours puisque ping utilise ICMP). Elle consiste à modifier la MSS, la taille maximale des paquets qu'annonce une machine qui fait du TCP à son partenaire. On voit cette annonce avec tcpdump :

```
13:35:32.703105 2001:7a8:7509:0:216:3eff:fe78:b525.65513 > 2001:660:3003:2::4:20.80: \
         S 4227314701:4227314701(0) win 32768 \
         <mss 1400,nop,wscale 0,sackOK,nop,nop,nop,nop,timestamp 0[|tcp]> \
         [flowlabel 0xbbb6b]
```

On voit ici la machine `2001:7a8:7509:0:216:3eff:fe78:b525` qui annonce, dans le paquet d'établissement de connexion TCP (paquet S pour `SYN`) une MSS de 1400 octets.

Pour modifier cette valeur, on peut bien sûr intervenir sur toutes les machines du réseau local mais c'est un travail pénible pour l'administrateur. Mieux vaut modifier de force ce réglage sur le routeur, ce qui se nomme le "*MSS clamping*". Cela peut se faire dans le démon `pppoe` avec l'option -m, par exemple, dans `/etc/ppp/peers/MON-FAI`, on met `pty "pppoe -I eth1 -T 80 -m 1412"`. À

noter que, sur Linux, ce "*clamping*" ne fonctionne apparemment que pour IPv4, obligeant à trouver une autre solution (comme l'annonce d'une MTU par RA) pour IPv6. Le "*clamping*" existe aussi sur d'autres systèmes ; par exemple, sur IOS, c'est l'option `ip tcp adjust-mss` (profitons-en pour signaler que Cisco a une excellente documentation <`http://www.cisco.com/en/US/tech/tk827/tk369/technologies_tech_note09186a0080093f1f.shtml`> sur le sujet).

On peut aussi créer une règle `iptables` ou `ip6tables` avec la cible TCPMSS, mais celle-ci n'est disponible en IPv6 que depuis la version 2.6.21 du noyau Linux, et n'est acceptée que par des versions très récentes d'iptables. Le principe (je n'ai pas testé moi-même, je copie Pascal Hambourg) :

```
iptables -t mangle -A FORWARD -p tcp -m tcp --tcp-flags SYN,RST SYN \
  -j TCPMSS --clamp-mss-to-pmtu
```

On peut aussi remplacer `--clamp-mss-to-pmtu`, qui force ("*to clamp*") la MSS à la valeur de la MTU par `--set-mss XXX` qui permet d'indiquer une valeur explicite.

Notons enfin que le fait de fixer la MTU, quel que soit le moyen utilisé, ne va pas forcément influer directement sur le MSS. Par exemple, avec NetBSD, par défaut, la MSS est la MTU de l'interface réseau non-locale ayant la plus grande MTU, ce qui peut être déroutant si on a une interface virtuelle ayant une MTU de plusieurs dizaines de milliers d'octets. Il peut donc être prudent de mettre les paramètres `sysctl` `net.inet6.tcp6.mss_ifmtu` et `net.inet.tcp.mss_ifmtu` à 1, pour que la MSS soit celle de la MTU de l'interface utilisée.

Un grand merci à Pascal Hambourg pour son aide pour déboguer ces problèmes et les documenter. Et merci aussi à Christophe Wolfhugel pour des détails érudits.