# RFC 8900 : IP Fragmentation Considered Fragile

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 12 septembre 2020

Date de publication du RFC : Septembre 2020

https://www.bortzmeyer.org/8900.html

---

Un concept important d'IP est la fragmentation, le découpage d'un paquet trop gros en plusieurs fragments, chacun acheminé dans un datagramme différent. Excellente idée sur le papier, la fragmentation est handicapée, dans l'Internet actuel, par les nombreuses erreurs de configuration de diverses "*middleboxes*". Ce nouveau RFC constate la triste réalité : en pratique, la fragmentation marche mal, et les machines qui émettent les paquets devraient essayer de faire en sorte qu'elle ne soit pas utilisée.

Rappelons d'abord ce qu'est la fragmentation, dans IP. Tout lien entre deux machines a une MTU, la taille maximale des datagrammes qui peuvent passer. C'est par exemple 1 500 octets pour l'Ethernet classique. Si le paquet IP est plus grand, il faudra le fragmenter, c'est-à-dire le découper en plusieurs fragments, chacun de taille inférieure à la MTU. En IPv4, n'importe quel routeur sur le trajet peut fragmenter un paquet (sauf si le bit DF - "*Don't Fragment*" - est mis à un dans l'en-tête IP), en IPv6, seule la machine émettrice peut fragmenter (tout ce passe comme si DF était systématiquement présent).

Ces fragments seront ensuite réassemblés en un paquet à la destination. Chacun étant transporté dans un datagramme IP différent, ils auront pu arriver dans le désordre, certains fragments ont même pu être perdus, le réassemblage est donc une opération non-triviale. Historiquement, certaines bogues dans le code de réassemblage ont même pu mener à des failles de sécurité.

Une légende urbaine s'est constituée petit à petit, racontant que les fragments, en eux-mêmes, posaient un problème de sécurité. C'est faux, mais ce genre de légendes a la vie dure, et a mené un certain nombre d'administrateurs de pare-feux à bloquer les fragments. Le RFC 7872[1] faisait déjà le constat que les fragments, souvent, n'arrivaient pas à destination. Outre ce RFC, on peut citer une étude très ancienne, qui montre que le problème ne date pas d'aujourd'hui, ≪ ""*Fragmentation Considered Harmful*" <http://www.hpl.hp.com/techreports/Compaq-DEC/WRL-87-3.pdf> (SIGCOMM

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc7872.txt

'87 "*Workshop on Frontiers in Computer Communications Technology*") ou, pour le cas spécifique du DNS, la plus récente « "*IPv6, Large UDP Packets and the DNS*" <http://www.potaroo.net/ispcol/2017-08/xtn-hdrs.html> ».

La section 2 de notre RFC explique en grand détail la fragmentation IP. Notez par exemple qu'IPv4 impose (RFC 791, section 3.2) une MTU minimale de 68 octets mais, en pratique, on a toujours au moins 576 octets (ou, sinon, une autre fragmentation/réassemblage, dans la couche 2). IPv6, lui, impose (RFC 8200), 1 280 octets. Il a même été suggéré <https://www.bortzmeyer.org/fragmentation-ip-1280.html> de ne jamais envoyer de paquets plus grands, pour éviter la fragmentation. Un Ethernet typique offre une MTU de 1 500 octets, mais elle peut être réduite par la suite en cas d'utilisation de tunnels.

Chaque lien a sa MTU mais ce qui est important, pour le paquet, c'est la **MTU du chemin** ("*Path MTU*"), c'est-à-dire la plus petite des MTU rencontrées sur le chemin entre le départ et l'arrivée. C'est cette MTU du chemin qui déterminera si on fragmentera ou pas. (Attention, le routage étant dynamique, cette MTU du chemin peut changer dans le temps.) Il est donc intéressant, pour la machine émettrice d'un paquet, de déterminer cette MTU du chemin. Cela peut se faire en envoyant des paquets avec le bit DF ("*Don't Fragment*", qui est implicite en IPv6) et en regardant les paquets ICMP renvoyés par les routeurs, indiquant que la MTU est trop faible pour ce paquet (et, depuis le RFC 1191, le message ICMP indique la MTU du lien suivant, ce qui évite de la deviner par essais/erreurs successifs.) Cette procédure est décrite dans les RFC 1191 et RFC 8201. Mais (et c'est un gros mais), cela suppose que les erreurs ICMP « "*Packet Too Big*" » arrivent bien à l'émetteur et, hélas, beaucoup de pare-feux configurés par des ignorants bloquent ces messages ICMP. D'autre part, les messages ICMP ne sont pas authentifiés (RFC 5927) et un attaquant peut générer de fausses erreurs ICMP pour faire croire à une diminution de la MTU du chemin, affectant indirectement les performances. (Une MTU plus faible implique des paquets plus petits, donc davantage de paquets.)

Quand une machine fragmente un paquet (en IPv4, cette machine peut être l'émetteur ou un routeur intermédiaire, en IPv6, c'est forcément l'émetteur), elle crée plusieurs fragments, dont seul le premier porte les informations des couches supérieures, comme le fait qu'on utilise UDP ou TCP, ou bien le numéro de port. La description détaillée figure dans le RFC 791 pour IPv4 et dans le RFC 8200 (notamment la section 4.5) pour IPv6.

Voici un exemple où la fragmentation a eu lieu, vu par tcpdump (vous pouvez récupérer le pcap complet en ). La machine `2605:4500:2:245b::bad:dcaf` a fait une requête DNS à `2607:5300:201:3100::2f69`, qui est un des serveurs faisant autorité pour le TLD `.md`. La réponse a dû être fragmentée en deux :

```
% tcpdump -e -n -r dns-frag-md.pcap
16:53:07.968917 length 105: 2605:4500:2:245b::bad:dcaf.44104 > 2607:5300:201:3100::2f69.53: 65002+ [1au] ANY
16:53:07.994555 length 1510: 2607:5300:201:3100::2f69 > 2605:4500:2:245b::bad:dcaf: frag (0|1448) 53 > 44104
16:53:07.994585 length 321: 2607:5300:201:3100::2f69 > 2605:4500:2:245b::bad:dcaf: frag (1448|259)
```

Le premier paquet est la requête, le second est le premier fragment de la réponse (qui va des octets 0 à 1447), le troisième paquet est le second fragment de cette même réponse (octets 1448 à la fin). Regardez les longueur des paquets IP, et le fait que seul le premier fragment, qui porte l'en-tête UDP, a pu être interprété comme étant du DNS.

Notez que certaines versions de traceroute ont une option qui permet d'afficher la MTU du lien (par exemple `traceroute -n --mtu IP-ADDRESS`.)

Les couches supérieures (par exemple UDP ou TCP) peuvent ignorer ces questions et juste envoyer leurs paquets, comptant qu'IP fera son travail, ou bien elles peuvent tenir compte de la MTU du chemin, par exemple pour optimiser le débit, en n'envoyant que des paquets assez petits pour ne pas être fragmentés. Cela veut dire qu'elles ont accès au mécanisme de découverte de la MTU du chemin, ou bien qu'elles font leur propre découverte, via la procédure PLPMTUD ("*Packetization Layer Path MTU Discovery*") du RFC 4821 (qui a l'avantage de ne pas dépendre de la bonne réception des paquets ICMP).

Bon, maintenant qu'on a vu la framentation, voyons les difficultés qui surviennent dans l'Internet d'aujourd'hui (section 3 du RFC). D'abord, chez les pare-feux et, d'une manière générale, tous les équipements intermédiaires qui prennent des décisions en fonction de critères au-dessus de la couche 3, par exemple un répartiteur de charge, ou un routeur qui enverrait les paquets à destination du port 443 vers un autre chemin que le reste des paquets. Ces décisions dépendent d'informations qui ne sont que dans le premier fragment d'un datagramme fragmenté. Décider du sort des fragments suivants n'est pas évident, surtout si on veut le faire sans maintenir d'état. Un pare-feu sans état peut toujours essayer d'accepter tous les fragments ultérieurs (ce qui pourrait potentiellement autoriser certaines attaques) ou bien tous les bloquer (ce qui arrêterait du trafic légitime fragmenté). Et les pare-feux avec état ne sont pas une solution idéale, puisque stocker et maintenir cet état est un gros travail, qui plante souvent, notamment en cas d'attaque par déni de service.

Certains types de NAT ont également des problèmes avec la fragmentation. Ainsi, les techniques A+P (RFC 6346) et CGN (RFC 6888) nécessitent toutes les deux que les fragments soient réassemblés en un seul paquet, avant de traduire.

Qui dit fragmentation dit réassemblage à un moment. C'est une opération délicate, et plusieurs programmeurs se sont déjà plantés en réassemblant sans prendre de précautions. Mais il y a aussi un problème de performance. Et il y a les limites d'IPv4. L'identificateur d'un fragment ne fait que 16 bits et cela peut mener rapidement à des réutilisations de cet identificateur, et donc à des réassemblages incorrects (les sommes de contrôle de TCP et UDP ne sont pas toujours suffisantes pour détecter ces erreurs, cf. RFC 4693). IPv6, heureusement, n'a pas ce problème, l'identificateur de fragment faisant 32 bits.

On l'a dit plus haut, la fragmentation, et surtout le réassemblage, ont une longue histoire de failles de sécurité liées à une lecture trop rapide du RFC par le programmeur qui a écrit le code de réassemblage. Ainsi, les fragments recouvrants sont un grand classique, décrits dans les RFC 1858, RFC 3128 et RFC 5722. Normalement, le récepteur doit être paranoïaque, et ne pas faire une confiance aveugle aux décalages ("*offset*") indiqués dans les paquets entrants, mais tous les programmeurs ne sont pas prudents. Il y a aussi le risque d'épuisement des ressources, puisque le récepteur doit garder en mémoire (le réassemblage implique le maintien d'un état) les fragments pas encore réassemblés. Un attaquant peut donc épuiser la mémoire en envoyant des fragments d'un datagramme qui ne sera jamais complet. Et il y a des identificateurs de fragment non-aléatoires, qui permettent d'autres attaques, documentées dans le RFC 7739 ou dans des articles comme « "*Fragmentation Considered Poisonous*" <https://arxiv.org/abs/1205.4011> de Herzberg et Shulman (cf. aussi mon résumé <https://www.bortzmeyer.org/dns-attaques-shulman.html>). Enfin, la fragmentation peut aider à échapper au regard des IDS (cf. « "*Insertion, Evasion and Denial of Service : Eluding Network Intrusion Detection*" <http://www.aciri.org/vern/Ptacek-Newsham-Evasion-98.ps> »).

Un autre problème très fréquent avec la fragmentation est causé par la configuration erronée de pare-feux. Souvent, des administrateurs réseau incompétents bloquent les messages ICMP "*Packet Too Big*", nécessaires pour la découverte de la MTU du chemin. C'est de leur part une grosse erreur (expliquée dans le RFC 4890) mais cela arrive trop souvent. Résultat, si la MTU du chemin est inférieure à la MTU du premier lien, la machine émettrice envoie des paquets trop gros, et ne sait pas que ces paquets n'ont pas pu passer. On a donc créé un trou noir (les paquets disparaissent sans laisser de trace).

Ce bloquage injustifié des messages ICMP peut également être dû à des causes plus subtiles. Par exemple, si un pare-feu laisse sortir tous les paquets mais, en entrée, n'autorise que les paquets dont l'adresse IP source a été utilisée comme destination récemment, alors, les erreurs ICMP, émises par des routeurs intermédiaires et ayant donc une adresse IP source jamais vue, seront jetées. Notre RFC note que cette bogue dans les pare-feux est apparemment assez fréquente dans les "*boxes*".

Toujours côté mauvaise configuration, le RFC cite aussi le problème des routeurs qui jettent les paquets ayant options ou extensions qu'ils ne connaissent pas, ce qui peut inclure les fragments. L'analyse du RFC 7872, ou celle dans l'article de Huston « "*IPv6, Large UDP Packets and the DNS*" <http://www.potaroo.net/ispcol/2017-08/xtn-hdrs.html> », montre bien que ce problème est fréquent, trop fréquent. Ainsi, même si la découverte de la MTU du chemin se passe bien, les fragments n'arriveront pas à destination. Pourquoi cette mauvaise configuration ? C'est évidemment difficile à dire, cela peut aller de logiciels bogués jusqu'à un choix délibéré d'un administrateur réseau ignorant qui a vaguement entendu une légende urbaine du genre « les fragments sont un risque de sécurité ».

Dans les cas précédents, la perte du message ICMP "*Packet Too Big*" était clairement de la faute d'un humain, l'administrateur du pare-feu. Mais il peut y avoir des obstacles plus fondamentaux au bon fonctionnement de la découverte de la MTU du chemin. Par exemple, si un serveur DNS "*anycasté*" envoie un paquet trop gros, et qu'un routeur intermédiaire envoie le message "*Packet Too Big*", ledit message ira vers l'adresse "*anycast*" du serveur, et atterrira peut-être vers une autre instance du serveur DNS, si le routeur qui a signalé le problème n'est pas dans le même bassin d'attraction que le client original. Le message ICMP ne sera donc pas reçu par l'instance qui aurait eu besoin de l'information. Le problème est d'autant plus génant que le DNS est le plus gros utilisateur d'UDP, et est donc particulièrement sensible aux problèmes de fragmentation (TCP gère mieux ces problèmes, avec la négociation de MSS).

Une variante du problème "*anycast*" survient lorsque le routage est unidirectionnel. Si l'émetteur n'est pas joignable, il ne recevra pas les messages ICMP. (Le cas est cité par le RFC mais me semble peu convaincant ; il y a peu de protocoles où l'émetteur peut se passer de recevoir des réponses. Et beaucoup de routeurs jettent les paquets pour lesquels ils n'ont pas de voie de retour, cf. RFC 3704.)

Maintenant qu'on a présenté en détail les problèmes liés à la fragmentation IP dans l'Internet actuel, quelles sont les approches possibles pour traiter ce problème ? La section 4 du RFC les présente. D'abord, les solutions situées dans la couche Transport. Comme indiqué plus haut, TCP peut éviter la fragmentation en découpant les données en segments dont chacun a une taille inférieure à la MTU du chemin (paramètre MSS, "*Maximum Segment Size*"). Cela suppose que la MSS soit réglée à une telle valeur (cf. mon article <https://www.bortzmeyer.org/mtu-et-mss-sont-dans-un-reseau.html>). Cela peut être manuel (si on sait qu'on va toujours passer par un tunnel avec faible MTU, on peut toujours configurer sa machine pour réduire la MSS), cela peut utiliser la procédure classique de découverte de la MTU du chemin ou bien cela peut utiliser la découverte de MTU sans ICMP du RFC 4821. D'autres protocoles que TCP peuvent fonctionner ainsi, comme DCCP (RFC 4340) ou SCTP (RFC 9260). À noter qu'UDP, lui, n'a pas de tel mécanisme, même si des travaux sont en cours pour cela.

Pour TCP, la méthode manuelle se nomme "*TCP clamping*" et peut se faire, par exemple avec Netfilter en mettant sur le routeur :

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

La méthode avec ICMP, on l'a vu, est fragile car les messages ICMP peuvent être bloqués. La méthode sans ICMP consiste pour TCP, en cas de détection de perte de paquets, à envoyer des paquets plus petits pour essayer de s'ajuster à la MTU du chemin. (Bien sûr, des paquets peuvent se perdre pour d'autres raisons que la MTU trop basse, et la mise en œuvre de cette technique est donc délicate.)

Autre solution, plutôt que d'impliquer la couche Transport, faire appel à la couche Application. Le RFC 8085 conseille aux applications qui font de l'UDP d'essayer d'éviter la fragmentation. Par exemple, si vous gérez un serveur DNS avec NSD, cela peut se faire en mettant dans le fichier de configuration :

```
ipv4-edns-size: 1432
ipv6-edns-size: 1432
```

Vous pouvez voir le résultat sur, par exemple, un des serveurs faisant autorité pour `.bostik` :

```
% dig +ignore @d.nic.fr DNSKEY bostik

; <<>> DiG 9.11.5-P4-5.1-Debian <<>> +ignore @d.nic.fr DNSKEY bostik
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12029
;; flags: qr aa tc rd; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 1432
; COOKIE: ef8669d08e2d9b26bb1a8ab85e21830a8931f9ab23403def (good)
;; QUESTION SECTION:
;bostik. IN DNSKEY

;; Query time: 2 msec
;; SERVER: 2001:678:c::1#53(2001:678:c::1)
;; WHEN: Fri Jan 17 10:48:58 CET 2020
;; MSG SIZE  rcvd: 63
```

Vous voyez que le serveur n'envoie pas de réponses de taille supérieure à 1 432 octets (la "*OPT PSEUDOSECTION*"). Au moment du test, la réponse faisait 1 461 octets, d'où le "*flag*" `tc` ("*TRuncated*"). Normalement, un client DNS, voyant que la réponse a été tronquée, réessaie en TCP (j'ai mis l'option `+ignore` à dig pour empêcher cela et illustrer le fonctionnement du DNS.)

En parlant du DNS, la section 5 du RFC liste des applications pour lesquelles la fragmentation joue un rôle important :

— Le DNS utilise encore largement UDP (malgré le RFC 7766) et, notamment avec DNSSEC, les réponses peuvent nettement excéder la MTU typique. C'est en général le premier protocole qui souffre, quand la fragmentation ne marche pas, d'autant plus que la grande majorité des échanges sur l'Internet commence par des requêtes DNS. Le client DNS peut limiter la taille des données envoyées, via EDNS (RFC 6891) et le serveur peut également avoir sa propre limite. Le serveur qui n'a pas la place de tout mettre dans la réponse peut, dans certains cas, limiter les données envoyées (en omettant les adresses de colle, par exemple) et si ça ne suffit pas, indiquer que la réponse est tronquée, ce qui doit mener le client à réessayer avec TCP (que tout le monde devrait gérer mais certains clients et certains serveurs, ou plutôt leurs réseaux, ont des problèmes avec TCP, comme illustré dans « "*Measuring ATR*" `<http://www.potaroo.net/ispcol/2018-04/atr.html>` »). L'importance de la fragmentation pour le DNS fait que le "*DNS Flag Day*" `<https://dnsflagday.net/2020/>` de 2020 est consacré à ce sujet.

— Le protocole de routage OSPF utilise également UDP mais le problème est moins grave, car c'est un protocole interne, pas utilisé sur l'Internet public, l'administrateur réseau peut donc en général s'assurer que la fragmentation se passera bien. Et, de toute façon, la plupart des mises en œuvre d'OSPF limitent la taille de leurs messages pour être sûr de rester en dessous de la MTU.
— L'encapsulation des paquets IP (pas vraiment la couche Application mais quand même un usage répandu) peut également créer des problèmes avec la fragmentation (cf. RFC 4459.) Cela concerne des protocoles comme IP-in-IP (RFC 2003), GRE (RFC 2784 et RFC 8086), et d'autres. Le RFC 7588 décrit une stratégie générale pour ces protocoles.
— Le RFC note que certains protocoles, pour des raisons de performance, assument tout à fait d'envoyer de très grands paquets et donc de compter sur une fragmentation qui marche. Ce sont notamment des protocoles qui sont conçus pour des milieux très particuliers, comme LTP (RFC 5326) qui doit fonctionner en présence d'énormes latences <https://www.bortzmeyer.org/latence.html>.

Compte-tenu de tout ceci, quelles recommandations concrètes donner ? Cela dépend évidemment du public cible. La section 6 de notre RFC donne des conseils pour les concepteurs et conceptrices de protocoles et pour les différents types de développeurs et développeuses qui vont programmer des parties différentes du système. D'abord, les protocoles, sujet principal pour l'IETF. Compte-tenu des importants problèmes pratiques que pose la fragmentation dans l'Internet actuel, le RFC prend une décision douloureuse : plutôt que de chercher à réparer l'Internet, on jette l'éponge et on ne conçoit plus de protocoles qui dépendent de la fragmentation. De tels protocoles ne seraient raisonnables que dans des environnements fermés et contrôlés. Comme souvent à l'IETF, le choix était difficile car il faut choisir entre les principes (la fragmentation fait partie d'IP, les composants de l'Internet ne doivent pas l'empêcher) et la réalité d'un monde de "*middleboxes*" mal programmées et mal gérées. Comme pour la décision de faire passer beaucoup de nouveaux protocoles sur HTTPS, le choix ici a été de prendre acte de l'ossification de l'Internet, et de s'y résigner.

Pour ne pas dépendre de la fragmentation, les nouveaux protocoles peuvent utiliser une MTU suffisamment petite pour passer partout, ou bien utiliser un système de découverte de la MTU du chemin suffisamment fiable, comme celui du RFC 4821. Pour UDP, le RFC renvoie aux recommandations de la section 3.2 du RFC 8085.

Ensuite, les conseils aux programmeurs et programmeuses. D'abord, dans les systèmes d'exploitation. Le RFC demande que la PLPMTUD (RFC 8899) soit disponible dans les bibliothèques proposées aux applications.

Ensuite, pour ceux et celles qui programment les "*middleboxes*", le RFC rappelle quand même qu'elles doivent respecter les RFC sur IPv4 (RFC 791) et IPv6 (RFC 8200). Pour beaucoup de fonctions assurées par ces boitiers (comme le filtrage), cela implique de réassembler les paquets fragmentés, et donc de maintenir un état. Les systèmes avec état sont plus compliqués et plus chers (il faut davantage de mémoire) ce qui motive parfois à préférer les systèmes sans état. Ceux-là n'ont que deux choix pour gérer la fragmentation : violer les RFC ou bien jeter tous les fragments. Évidemment, aucune de ces deux options n'est acceptable. Le RFC demande qu'au minimum, si on massacre le protocole IP, cela soit documenté. (Ces "*middleboxes*" sont souvent traitées comme des boites noires, installées sans les comprendre et sans pouvoir déboguer les conséquences.)

Enfin, les opérateurs des réseaux doivent s'assurer que la PMTUD fonctionne, donc émettre des "*Packet Too Big*" si le paquet est plus gros que la MTU, et ne pas bloquer les paquets ICMP. Notre RFC permet toutefois de limiter leur rythme (RFC 1812 et RFC 4443). En tout cas, comme le rappelle le RFC 4890, filtrer les paquets ICMP "*Packet Too Big*" est **mal** !

De la même façon, le RFC rappelle aux opérateurs réseau qu'on ne doit **pas** filtrer les fragments. Ils sont utiles et légitimes.

---

Notez que les recommandations de ce RFC peuvent sembler contradictoires : on reconnait que la fragmentation marche mal et qu'il ne faut donc pas compter dessus mais, en même temps, on rappelle qu'il faut se battre pour qu'elle marche mieux. En fait, il n'y a pas de contradiction, juste du réalisme. L'Internet n'ayant pas de Chef Suprême qui coordonne tout, on ne peut pas espérer qu'une recommandation de l'IETF soit déployée immédiatement partout. On se bat donc pour améliorer les choses (ne pas bloquer les fragments, ne pas bloquer les messages ICMP "*Packet Too Big*") tout en étant conscient que ça ne marchera pas à 100 % et que les administrateurs système et réseau doivent en être conscients.