RFC 6864 : Updated Specification of the IPv4 ID Field

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 9 février 2013

Date de publication du RFC : Février 2013

https://www.bortzmeyer.org/6864.html

Le champ ID ("*internet identification field*") de l'en-tête des paquets IPv4 sert à plusieurs usages mais le plus connu est pour identifier les fragments d'un même datagramme original, afin de réassembler ce datagramme. La norme officielle de ce champ pose des contraintes qui ne peuvent tout simplement pas être respectées avec les réseaux modernes. Donc, toutes les mises en œuvre d'IP violaient la norme, avant que ce RFC 6864 [1] ne les ramènent dans la légalité en changeant la norme, par le relâchement de ces contraintes. Par ailleurs, cet excellent RFC, très bien écrit, est une lecture très recommandée pour celui ou celle qui veut comprendre les détails de la fragmentation dans IPv4.

La principale contrainte était que le champ ID devait être **unique** pour tout tuple {adresse source, adresse destination, protocole de transport}, pendant une période qui est la durée de vie attendue des paquets. Avec les capacités des réseaux modernes, ce champ, limité à 16 bits, ne pouvait tout simplement pas rester unique aussi longtemps. Notre RFC change donc cette contrainte pour la limiter aux paquets effectivement fragmentés (une minorité, en général). Notez qu'IPv6 n'a pas ce problème, à la fois parce que son champ ID est plus grand (RFC 2460, section 4.5) et parce que cette contrainte limitée avait été définie dès le début.

Si vous voulez les détails du calcul, avec ses 16 bits, le champ ID peut stocker 65 536 valeurs. La durée de vie des paquets dans le réseau est la MDL ("*maximum datagram lifetime*", typiquement deux minutes, cf. RFC 791 et RFC 1122 qui ne donnent pas directement de valeur précise). Avec une MTU typique de 1 500 octets, en supposant que les datagrammes IP ont la taille de cette MTU (moins 20 octets pris par l'en-tête) cela ferait (1 480 * 8) * (65 536 / (2 * 60)) = 6,465 Mb/s au maximum ! Avec une MTU irréaliste (le maximum permis par IP, 65 535 octets), on n'irait encore qu'à 286 Mb/s. (Si vous aimez la précision, notez qu'on pourrait, pour avoir ces 286 Mb/s, envoyer des datagrammes de 65 535 octets sur un réseau dont la MTU n'est que de 1 500 octets. Mais, alors, tous les paquets seraient

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc6864.txt

fragmentés et le coût de réassemblage et les problèmes liés feraient que l'opération serait sans doute une mauvaise idée.) Le calcul complet figure dans le RFC 4963. Notez que tous ces RFC donnent des valeurs de plusieurs minutes pour le MDL, valeurs qui me semblent complètement irréalistes. Je n'ai jamais vu un datagramme se promener dans l'Internet pendant deux minutes entières avant d'arriver ! (Voir la discussion en section 3.2.) Ceci dit, même avec une MDL de 3 secondes, on n'arriverait encore qu'à 255 Mb/s (pour la MTU typique), ce qui n'est pas suffisant. Des interfaces Ethernet à 1 Gb/s sont aujourd'hui banales.

Les réseaux modernes vont évidemment bien plus vite que cela et il est donc clair que les mises en œuvre d'IP violent cette contraine d'unicité. Typiquement, elles réutilisent l'`ID` bien avant l'expiration du délai maximum (section 5.1), ou bien elles utilisent un `ID` constant dès que le paquet n'est pas fragmenté (ROHC en tient compte pour comprimer ce champ, cf. section 5.4).

Avant de revenir sur les changements de ce RFC, retournons au fonctionnement de la fragmentation (section 3). Celle-ci survient lorsqu'un datagramme est plus grand que la MTU du prochain lien à traverser. Le datagramme est alors fragmenté, chaque fragment est un paquet IP routé indépendemment, mais tous les fragments d'un même datagramme ont la même valeur du champ `ID`. Vu par tcpdump, voici une réponse DNS de 3,5 ko fragmentée, les trois fragments du datagramme original ayant l'`ID` 60977 (indiqué juste après le TTL) :

```
16:32:35.631159 IP (tos 0x0, ttl 64, id 60977, offset 0, flags [+], proto UDP (17), length 1500)
    217.70.190.232.53 > 88.189.152.187.44168: 41314*- q: ANY? sources.org. 23/0/1 sources.org. [1d] SOA ns3.
16:32:35.631169 IP (tos 0x0, ttl 64, id 60977, offset 1480, flags [+], proto UDP (17), length 1500)
    217.70.190.232 > 88.189.152.187: udp
16:32:35.631173 IP (tos 0x0, ttl 64, id 60977, offset 2960, flags [none], proto UDP (17), length 419)
    217.70.190.232 > 88.189.152.187: udp
```

On note que les numéros de port, ainsi que l'analyse du contenu du paquet, ne sont possibles que pour le premier fragment (le seul qui contient les en-têtes UDP et DNS). D'autres paquets de la même source vers la même destination, auront un `ID` différent :

```
16:42:17.601777 IP (tos 0x0, ttl 64, id 60988, offset 0, flags [+], proto UDP (17), length 1500)
    217.70.190.232.53 > 88.189.152.187.54895: 15361*- q: ANY? sources.org. 23/0/1 sources.org. [1d] SOA ns3.
16:42:17.601787 IP (tos 0x0, ttl 64, id 60988, offset 1480, flags [+], proto UDP (17), length 1500)
    217.70.190.232 > 88.189.152.187: udp
16:42:17.601792 IP (tos 0x0, ttl 64, id 60988, offset 2960, flags [none], proto UDP (17), length 419)
    217.70.190.232 > 88.189.152.187: udp
```

En IPv4, la fragmentation peut être faite par la source des datagrammes, ou bien par un routeur intermédiaire, lorsqu'il détecte que le lien suivant a une MTU trop faible. L'émetteur ne sait donc pas forcément si son paquet sera fragmenté. En IPv6, par contre, seule la source peut fragmenter et elle sait donc si le paquet sera fragmenté ou pas.

Pour représenter la fragmentation, IPv4 a quatre champs, `ID`, déjà cité, le bit DF ("*Don't fragment*", qui interdit aux routeurs de fragmenter), la distance depuis le début du datagramme ("*fragment offset*", 0, 1480 puis 2960 dans les exemples plus haut) et le bit MF ("*More fragments*", s'il vaut zéro, cela signifie que ce fragment est le dernier dans le datagramme original, il est noté par un signe + dans la sortie de tcpdump). En IPv6, ce n'est plus dans l'en-tête mais dans un en-tête d'extension ("*fragment header*"), qui ne comporte que trois champs : le bit DF a disparu car, en IPv6, il est toujours à 1 (un routeur intermédiaire n'a pas le droit de fragmenter). Cet en-tête de fragmentation n'est présent que si le paquet est effectivement fragmenté. Contrairement à IPv4, IPv6 ne gaspille pas d'identificateur (`ID`) pour les

paquets non fragmentés. IPv4 doit mettre un identificateur dans tous les paquets, même non fragmentés, car il ne peut pas être sûr qu'un routeur plus loin ne va pas décider de fragmenter.

En outre, en IPv6, le champ `ID` fait 32 bits et permet donc de stocker bien plus de valeurs. Bref, notre RFC ne couvre que le cas d'IPv4, IPv6 n'a pas de problèmes. Même si tous les paquets sont fragmentés, le champ `ID` plus grand permettrait à IPv6 de pomper 18,8 Tb/s (en supposant des datagrammes de la taille maximale permise par IP). Si les paquets ne sont pas fragmentés, il n'y a plus de limite.

Donc, que faire pour IPv4, afin de réconcilier la norme et la pratique ? La section 4 expose les nouvelles règles pour le champ `ID`. L'idée de base est que tous les paquets ne sont pas fragmentés. En fait, avec la PMTUD (RFC 1191), la plupart ne le sont pas. Donc, l'obligation d'unicité du champ `ID` n'est maintenue **que** pour les paquets fragmentés ou susceptibles de l'être. La grande majorité des paquets TCP, aujourd'hui, grâce à la PMTUD, ne seront pas fragmentés (ils ont le bit DF, "*Don't fragment*") et n'ont donc pas d'obligation d'un identificateur unique.

Ce RFC introduit le concept de « datagramme atomique ». Un datagramme atomique est un datagramme qui n'est pas fragmenté (MF = 0 et distance au début du datagramme = 0) et qui ne le sera pas (DF = 1). En pseudo-code ressemblant à du C, `(DF==1)&&(MF==0)&&(frag_offset==0)`. Les datagrammes non-atomiques sont tous les autres (ils sont déjà fragmentés ou bien, n'ayant pas le bit DF à un, ils le seront peut-être plus tard). Aujourd'hui, dans le trafic TCP habituel, quasiment tous les datagrammes sont atomiques. La nouvelle règle de non-unicité ne s'applique qu'aux datagrammes non-atomiques. Pour les datagrammes atomiques, IP peut mettre le champ `ID` à n'importe quelle valeur (section 4.2). Pour les autres, il faut continuer à garder `ID` unique pendant la MDL (section 4.3). Cela a des conséquences sur les sources qui émettent beaucoup de datagrammes non-atomiques, typiquement les gros serveurs DNS (cf. RFC 6891, des réponses DNS plus grosses que la MTU sont courantes et les serveurs ne font pas de PMTUD en UDP). Ces sources, pour réussir à respecter l'unicité des `ID` peuvent se trouver dans l'obligation de s'auto-limiter et donc de ne pas envoyer trop de paquets (section 5.2).

À noter qu'il y a d'autres usages possibles pour le champ `ID` que la fragmentation et le réassemblage. Il a été envisagé d'utiliser ce champ pour détecter des datagrammes dupliqués (chose qui est possible dans un routeur congestionné, cf. RFC 1122, section 3.2.1.5), ce qui diminuerait le travail de protocoles de transport comme TCP qui, actuellement, doivent gérer ce problème. Il a aussi été proposé de se servir d'`ID` pour des activités de diagnostic. Aucun de ces projets n'est allé très loin et, de toute façon, ils ne marcheraient pas en IPv6 qui n'a de champ `ID` que pour les datagrammes effectivement fragmentés.

Comme la nouvelle règle ne marche que pour la fragmentation, les autres usages du champ `ID` sont désormais interdits : plus question d'essayer d'utiliser ce champ pour autre chose, puisqu'il n'est officiellement plus unique. Ce champ doit donc être ignoré si le datagramme est atomique.

Le reste ne change pas. Par exemple, le bit DF garde exactement la même signification (« ne fragmente surtout pas ce paquet »).

En pratique, qu'est-ce que cela va changer (section 5) ? Rappelez-vous qu'en pratique, la nouvelle règle est depuis longtemps la règle de fait et que l'Internet ne s'est pas écroulé. Donc, notre RFC estime qu'il ne faut pas s'attendre à des problèmes, sinon, ceux-ci auraient été signalés il y a des années.

Une autre solution au problème du champ `ID` trop court a été mentionnée plus haut : réutiliser les valeurs bien avant l'expiration du MDL, d'autant plus qu'on a vu que celui-ci est typiquement très grand. L'inconvénient de cette approche est que, si des paquets fragmentés traînent et arrivent ensuite à peu près en même temps que des fragments d'un autre datagramme, utilisant le même identificateur, le

réassemblage va produire des paquets qui ne correspondent pas à l'original, en fusionnant les deux paquets (RFC 4963). Les sommes de contrôle vont peut-être détecter le problème, mais le RFC recommande d'utiliser des contraintes d'intégrité dans les applications, pour être sûr. Si on a de telles contraintes bien fortes, on peut réutiliser les identificateurs plus vite.

Autre problème potentiel, avec les "*middleboxes*" qui réécrivent les paquets, par exemple les routeurs NAT. Elles ne doivent pas bêtement copier le champ `ID` mais en générer un correctement. Comme le routeur NAT typique réécrit l'adresse source de N machines en un nombre bien plus petit d'adresses (une seule pour le petit routeur NAT à la maison, un peu plus pour un CGN), le risque de casser l'unicité est élevé : si deux machines sur le réseau local communiquent avec la même destination externe et émettent des datagrammes avec le même `ID`, une réécriture bête de l'adresse IP source cassera l'unicité. Les CGN ont davantage d'adresses externes mais aussi bien davantage de trafic et le risque est donc encore plus élevé avec eux.

On a vu que ce RFC changeait l'ancienne norme (impossible à respecter) pour l'aligner avec la pratique. La section 6 donne la liste exacte des RFC modifiés : RFC 791, la norme IPv4 originale, et RFC 1122 notamment.

La fragmentation a des tas de conséquences sur la sécurité, d'où la section 7 qui examine les résultats de la nouvelle règle. Par exemple, elle rendra plus difficile le comptage des machines situées derrière un routeur NAT, même si ce n'était pas son objectif (cf. S. Bellovin, "*A Technique for Counting NATted Hosts*" `<http://www.cs.columbia.edu/~smb/papers/fnat.pdf>`).