RFC 6052 : IPv6 Addressing of IPv4/IPv6 Translators

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 1 novembre 2010

Date de publication du RFC : Octobre 2010

https://www.bortzmeyer.org/6052.html

Il existe plusieurs mécanismes pour assurer la co-existence entre les protocoles IPv4 et IPv6. Ce RFC ne normalise pas un nouveau mécanisme mais spécifie les formats d'adresse à utiliser pour les traducteurs IPv4¡-¿IPv6, les logiciels qui vont convertir d'un format dans l'autre. Il est donc le socle des résultats de la nouvelle version du groupe de travail IETF Behave <http://tools.ietf.org/wg/behave> qui, depuis l'échec de NAT-PT, essaie de normaliser une technique de traduction d'adresses permettant d'assurer la coexistence dans certains cas.

Le RFC du groupe Behave qui décrit le cadre général de cette traduction ("*Framework for IPv4/IPv6 Translation*") est le RFC 6144[1]. Empruntons-lui quelques exemples. Supposons par exemple un réseau purement IPv6 (hypothèse réaliste à l'heure où il ne reste plus que quelques mois d'adresses v4 <https://www.bortzmeyer.org/epuisement-adresses-ipv4.html>) et qui veut parler aux serveurs Web existants qui n'ont que v4. Un des mécanismes possibles est la traduction d'adresses. Supposons que le site Web convoité soit en 198.51.100.129. La machine purement v6 interroge un serveur DNS 64 (protocole pas encore normalisé) qui lui répond « Va voir en 64:ff9b::c633:6481 » (c633:6481 == 198.51.100.129). Elle émet alors un paquet IPv6 vers cette adresse. Sur le trajet, un traducteur d'adresses va prendre ce paquet, le traduire en v4 algorithmiquement et, lorsque la réponse reviendra, il la traduira en v6, l'envoyant à la machine cliente. Voici un exemple où une machine purement IPv6 tente de contacter Twitter, service purement IPv4 :

```
% telnet twitter.com 80
Trying 64:ff9b::80f2:f0f4...
Connected to twitter.com.
Escape character is '^]'.
```

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc6144.txt

Bien sûr, cette présentation est très sommaire mais c'est parce que le travail de spécification n'est pas encore terminé. La première étape de cette spécification est de décider des adresses utilisées et c'est ce que fait notre RFC. (Il pourrait d'ailleurs, dans le futur, être utilisé par des techniques autre que la traduction, comme l'encapsulation, cf. section 1.1.)

Un peu de vocabulaire, d'abord (section 1.3) :

— Un traducteur d'adresse est tout dispositif qui va tripoter des adresses, qu'il ait accès aux paquets (dans ce cas, on peut l'appeler « traducteur v4/v6 » ou « traducteur IP ») ou pas (cas d'un serveur DNS 64).
— "*IPv4-embedded IPv6 address*", une adresse IPv6 qui contient une adresse IPv4 (comme dans l'exemple `64:ff9b::c633:6481` plus haut).
— "*IPv4-converted IPv6 address*", un cas particulier des précédentes,
— "*IPv4-translatable IPv6 address*", un autre cas particulier des "*IPv4-embedded IPv6 address*", où l'adresse IPv6 est globalement unique et peut donc être routée dans le grand Internet.

Les traductions peuvent être **sans état** ("*stateless*") ou bien **avec état** ("*stateful*"). Dans le premier cas, le traducteur n'a aucune mémoire. Chaque paquet est traité isolément, comme si le traducteur venait juste de démarrer. Cela permet donc de traiter une bien plus grande quantité de paquets et assure les meilleures performances et, surtout, le passage à l'échelle. Dans le second cas, celui de la traduction avec état, la traduction dépend des paquets précédents, par exemple parce que le traducteur se souvient que les paquets venant de tel port sont destinés à telle adresse. Nécessitant une table des correspondances en mémoire, la traduction avec état passe moins à l'échelle. Mais, dans certains cas, elle est la seule réaliste, puisqu'on ne peut pas stocker toutes les informations dans une seule adresse, surtout si elle est IPv4.

Armé de ces définitions, quels sont les formats d'adresses "*IPv4-embedded*" possibles ? La section 2 les liste. D'abord, un préfixe spécial a été réservé à l'IANA <`https://www.iana.org/assignments/ipv6-address-space/ipv6-address-space.xhtml#note6`> (cf. section 6) pour **certaines** de ces adresses : `64:ff9b::/96` (section 2.1). Lorsque vous verrez passer une adresse IPv6 commençant par ce préfixe, vous pouvez être sûr de voir une adresse IPv4 embarquée.

Ensuite, toutes les adresses IPv6 "*IPv4-embedded*", qu'elles utilisent ce préfixe ou pas, suivent le même format (section 2.2) : un préfixe (`64:ff9b::/96` ou bien un préfixe tiré de l'espace d'adressage de l'opérateur), l'adresse IPv4, un suffixe (parfois nul). Pour respecter la section 2.5.1 du RFC 4291, un octet nul est parfois ajouté au milieu de l'adresse IPv4. Six longueurs de préfixe sont acceptées, 32, 40, 48, 56, 64, ou 96 (cette dernière est celle qui est typiquement utilisée avec le préfixe `64:ff9b::/96`). Les sections 3.3 et 3.4 expliquent quelle longueur choisir selon les cas.

Si on utilise un préfixe de longueur 48, comme l'octet qui va des bits 64 à 71 inclus doit être nul, l'adresse IPv4 est en deux parties. On a donc d'abord le préfixe IPv6, les deux premiers octets de l'adresse IPv4, un octet nul, le reste de l'adresse IPv4 et un suffixe de cinq octets. Avec un préfixe de longueur 96, tout est plus simple, le prefixe occupe les douze premiers octets, l'adresse IPv4 prend le reste (et le suffixe est absent). Le tableau 1 dans la section 2.2 résume les autres possibilités.

Une fois ce format défini, la section 2.3 décrit le (très simple) algorithme qui permet de passer d'une adresse IPv4 à une adresse IPv6 et réciproquement.

Petite question de présentation <`https://www.bortzmeyer.org/representation-texte.html`> : comment afficher les adresses IPv6 "*IPv4-embedded*" ? Bien sûr en obéissant au RFC 4291 (section 2.2) et au RFC 5952. Comme le RFC 4291 permet de représenter les quatre derniers octets (et uniquement ceux-ci) sous la forme d'une adresse IPv4, l'adresse `64:ff9b::c633:6481`, citée plus haut, aurait aussi pu s'écrire `64:ff9b::198.51.100.129`, ce qui est certainement plus lisible.

Ensuite, comment utiliser ces adresses sur un vrai réseau ? La section 3 couvre les questions de déploiement. D'abord, le préfixe `64:ff9b::/96` ne doit être utilisé qu'avec des adresses IPv4 globalement uniques (donc, par exemple, pas avec celles du RFC 1918). Moins évident, il ne doit pas non plus être utilisé pour construire des adresses IPv6 "*IPv4-translatable*", celles-ci devant être globalement uniques. Si tout le monde est libre de se servir de ce préfixe `64:ff9b::/96`, on ne peut pas compter qu'on recevra les paquets qui lui sont adressés depuis un point quelconque de l'Internet. Son usage normal est limité à l'intérieur d'un réseau (section 3.1), autrement, il faut se servir de préfixes spécifiques d'un opérateur.

Pendant qu'on parle de ce préfixe, il faut noter que les opérateurs qui s'en servent peuvent l'annoncer dans leurs tables de routage et qu'il apparaîtra donc sur l'Internet, par exemple si un opérateur décide de fournir un service de traduction d'adresses à un autre. Mais une telle annonce devrait normalement être contrôlée (section 3.2) et ne pas être transmise à toute la DFZ. Et, normalement, ceux qui annoncent ce préfixe en BGP doivent logiquement fournir un service de traduction ensuite (ou bien ils créent un trou noir).

Quel préfixe choisir pour ses services de traduction ? La section 3.3 donne quelques conseils. Par exemple, la « meilleure » longueur est sans doute 32, car, une fois l'adresse IPv4 ajoutée, la totalité de l'information se trouve dans les bits « de routage » (les 64 premiers). Pour le routage, c'est donc la meilleure solution mais obtenir de son RIR un /32 entier uniquement pour le service de traduction d'adresses peut être dificile. Des préfixes plus longs sont donc plus raisonnables. Le RFC suggère par exemple qu'un opérateur qui a un /32 en tout consacre un /40 au service de traduction (moins de 0,5 % de son espace d'adressage). S'il s'agit d'un simple client et pas d'un opérateur, et qu'il a un /48, un /56 conviendra pour le service de traduction. Mais la valeur exacte dépend aussi du scénario de coexistence considéré.

En cas de traduction avec état (section 3.4), les choix peuvent être différents et le préfixe `64:ff9b::/96` est souvent la solution la plus intéressante.

Un des intérêts de ce RFC est la section 4, qui explique les raisons des choix faits. Ainsi, le suffixe, pour les formats où il y en avait un, aurait pu être utilisé pour stocker de l'information sur les ports <https://www.bortzmeyer.org/loguer-adresse-et-port.html> (section 4.1). Cela aurait augmenté les chances de pouvoir faire une traduction sans état. Cette option n'a pas été retenue dans l'immédiat mais pourrait être ajoutée plus tard. En attendant, le suffixe a obligatoirement tous ses bits à zéro ce qui, pour le format avec préfixe /32, mène à un identificateur d'interface (les 64 derniers bits d'une adresse IPv6) entièrement nul. C'est normalement interdit (section 2.6.1 du RFC 4291) mais notre RFC explique que c'est acceptable dans le contexte limité de la traduction d'adresse.

Le choix de la valeur `64:ff9b::/96` pour le préfixe de référence est expliqué en section 4.2. D'abord, le préfixe existant pour les adresses IPv4, `::ffff:0:0/96` (section 2.5.5.2 du RFC 4291), aurait pu être réutilisé (c'était prévu dans la section 2.1 du RFC 2765). Cela aurait simplifié l'effort de normalisation. Le problème est que, justement, ce préfixe est déjà connu. Plusieurs mises en œuvre d'IPv6 le traitent à part (par exemple, Windows génère systématiquement des paquets IPv4 lorsqu'on lui présente ces adresses IPv6).

Il fallait donc allouer un nouveau préfixe. Un /32 aurait permis d'avoir tous les bits significatifs dans les 64 premiers bits mais n'aurait pas permis d'utiliser la représentation texte avec une adresse IPv4 (qui n'est possible qu'à la fin d'une adresse v6). Finalement, un préfixe /96 a été choisi. Sa valeur `64:ff9b::/96` n'a pas été tirée au hasard : elle est neutre pour le calcul de la somme de contrôle. La somme de 64 et de ff9b est ffff, ce qui est équivalent à zéro en complément à un. (Depuis, un autre préfixe l'a rejoint, le `64:ff9b:1::/48` du RFC 8215.)

Ces systèmes de traduction d'adresse posent-ils des problèmes de sécurité ? Oui, dit la section 5 qui en expose certains. Par exemple, un paquet peut mentir sur son adresse IP source, qu'il y ait traduction ou pas. En fait, note la section 5.1, un traducteur IP est comme un routeur, il a les mêmes vulnérabilités et il peut mettre en œuvre les mêmes protections. Ainsi, lorsqu'il reçoit un paquet IPv6 incluant une adresse IPv4, il peut, avant traduction, s'assurer que la machine était autorisée à utiliser cette adresse IPv4 (si le paquet vient de l'intérieur d'un réseau, c'est relativement facile). D'autres tests sont possibles comme les vérifications du chemin de retour (cf. RFC 3704).

D'autre part, lorsqu'un pare-feu teste les adresses IPv4 (section 5.2), on pourrait imaginer un attaquant qui envoie des paquets IPv6 avec adresse v4 embarquée. Le paquet v6 ne serait pas arrêté par les filtres v4 mais, après traduction, le paquet v4 pourrait faire des dégâts. Il est donc crucial que le pare-feu vérifie les adresses v4 embarquées comme si elles étaient natives.