RFC 6296 : IPv6-to-IPv6 Network Prefix Translation

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 30 juin 2011

Date de publication du RFC : Juin 2011

https://www.bortzmeyer.org/6296.html

Avec ce RFC, encore expérimental, IPv6 gagne la possibilité de faire de la **traduction d'adresses**, c'est-à-dire d'avoir dans un réseau local des adresses internes, qui seront dynamiquement traduites en adresses externes par le routeur de sortie, qui pourra également faire l'opération inverse (de l'extérieur vers l'intérieur).

Cette traduction est souvent nommée NAT ("*Network Address Translation*") mais ce terme n'est pas correct : la traduction habituelle en IPv4, celle que tout le monde doit supporter pour sa connexion Internet à la maison et pour son accès 3G, NAT44, ne traduit pas uniquement l'adresse. Pour faire face à la pénurie d'adresses IPv4, elle met en correspondance plusieurs adresses internes (typiquement du RFC 1918 [1]), avec une adresse externe, en utilisant les numéros de port pour démultiplexer les paquets entrants. Cette technique devrait donc s'appeler NAPT ("*Network Address and Port Translation*") et pas NAT. Elle a des tas d'inconvénients, décrits dans le RFC 6269.

En IPv6 au contraire, sujet de notre RFC, il n'y a pas de pénurie d'adresses et pas de raison de faire du NAPT. Le protocole de traduction proposé dans ce RFC 6296 mériterait donc, lui, de s'appeler NAT (ou peut-être NAT66), mais, comme ce terme est désormais très galvaudé, ses auteurs ont plutôt choisi **NPT** pour "*Network Prefix Translation*". NPTv6 permet une correspondance univoque (1 :1) entre les adresses internes et les adresses externes, évitant ainsi le partage d'adresses dont le RFC 6269 décrit les défauts. Pour diverses raisons (cf. RFC 2993 et section 5 de notre RFC), l'IETF n'encourage pas les systèmes de traduction mais, si un utilisateur tient à en faire, ce RFC lui fournit un protocole bien étudié. Il n'évite pas tous les problèmes de la traduction (RFC 4864, RFC 5902) mais les limite.

N'utilisant pas les ports, NPTv6 fonctionne donc avec tous les protocoles de transport, contrairement au NAT44 traditionnel (cf. section 6). Il est **sans état** (chaque paquet est traduit indépendemment des autres) donc le routeur qui fait la traduction n'a pas de problèmes comme le remplissage de la table (fréquent avec NAT44). Ainsi, un réseau peut être servi par deux traducteurs (par exemple pour la redondance) sans qu'ils aient à partager un état. Mais ce caractère sans état a aussi d'autres conséquences :

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc1918.txt

— NPT est juste une technique de traduction, il ne fournit pas de filtrage, ce n'est pas un pare-feu (les routeurs NAT44 mélangent les deux fonctions de traducteur et de pare-feu avec état), si on veut un pare-feu, il faut donc le faire en plus (cf. RFC 6092),
— il permet le maintien du principe de la connexion de bout en bout, y compris pour les connexions « entrantes », mais, comme toute technique de traduction, NPT a des problèmes avec les applications qui incluent les adresses IP dans les données,
— même problème avec les services qui incluent l'adresse IP dans le calcul d'un MAC comme l'option d'authentification de TCP (RFC 5925).

Voilà pour un résumé très rapide des propriétés de NPT v6.

Mais pour quelles raisons peut-on avoir envie de déployer un système de traduction ? La section 1.1 de notre RFC propose de les résumer en un principe : indépendance vis-à-vis de l'adresse. Ce principe comprend plusieurs propriétés :

— Les adresses internes au réseau local étant distinctes de celles du réseau extérieur, aucune nécessité de renuméroter si on change de FAI (RFC 5887).
— Le site peut choisir son opérateur, sa connexion, ses opérateurs multiples, même ("*multi-homing*" sans BGP),
— Pas besoin d'obtenir des adresses PI pour avoir des adresses à soi,
— Et pour le FAI, pas besoin de BCP 38 <https://www.bortzmeyer.org/bcp38.html> (RFC 2827), les adresses annoncées étant traduites en ses adresses.

« Adresses internes + traduction » est donc potentiellement un « PI du pauvre ». En IPv4, beaucoup de sites utilisaient le NAT pour atteindre cette indépendance, même lorsqu'elles pouvaient obtenir suffisamment d'adresses IP. Le RFC 4864 décrit d'ailleurs un concept proche nommé « autonomie du réseau ».

NPTv6 est donc un mécanisme qui permet de réaliser cette indépendance du réseau. Meilleur que le NAT44 (pas de manipulation des ports, fonctionne donc avec tous les protocoles de transport, comme SCTP, permet les connexions entrantes, purement « algorithmique » donc sans état), il peut donc être un ajout intéressant à la boîte à outils de l'administrateur de réseaux IPv6.

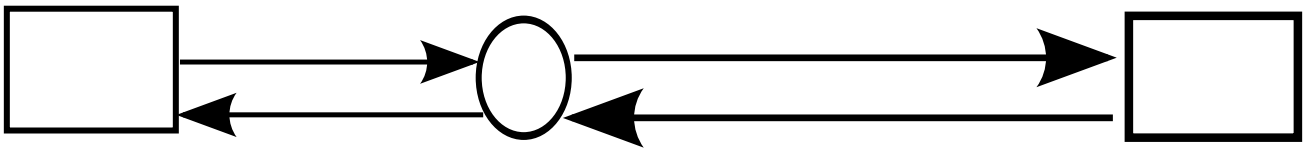
Pour être tout à fait complet, il faut aussi rappeler ses inconvénients (RFC 2993) :

— Comme il modifie l'en-tête IP, il est incompatible avec les techniques de sécurisation de cet en-tête comme l'AH d'IPsec (RFC 4302),
— Comme toute traduction d'adresses, il ne marche pas avec les protocoles qui transmettent des adresses IP dans les données comme SIP ; ceux-ci auront besoin d'un ALG,
— La configuration du DNS va être pénible car il faudra prévoir une vue externe et une interne ("*split DNS*").

Bref, NPTv6 n'est pas une solution parfaite et c'est pour cela que ce RFC reste au statut Expérimental.

Si on décide de poursuivre cette voie, il faut choisir les adresses à utiliser en interne. Les ULA du RFC 4193 semblent la solution idéale, maximisant l'indépendance.

Voici pour les grands principes. Quelques détails pratiques maintenant. Prenons l'exemple simple de la section 2.1 : un réseau interne, utilisant un préfixe ULA, `fd01:203:405:/48`, est connecté à l'Internet via un unique FAI, et un routeur qui fait du NPTv6. Le préfixe PA alloué par le FAI et routable publiquement est `2001:0db8:1:/48` (notez que NPT n'impose pas que les deux préfixes aient la même longueur, cf. section 3.7). Lorsqu'un paquet « sort » (va du réseau local vers l'Internet), l'adresse IP source est remplacée par une adresse IP prise dans le préfixe du FAI. Les autres champs de l'en-tête IP ne sont pas modifiés. En sens inverse, lorsqu'un paquet « entre » (vient de l'Internet vers le réseau local), c'est l'adresse de destination qui est touchée, pour être remplacée par une adresse du réseau local.



La partie identifiant la machine sur le réseau local (80 bits dans cet exemple) n'est en général pas conservée telle quelle. Ainsi, si `fd01:203:405:1::1234` écrit à `2a00:1450:8007::13`, Gmail, le destinataire, verra le paquet venant de `2001:0db8:1:d550::1234`

En effet, la traduction est neutre pour la somme de contrôle. Si on suit l'algorithme standard du RFC 1071, on veut obtenir la même somme de contrôle avant et après traduction. Les protocoles qui utilisent une somme calculée sur une partie de l'en-tête IP (c'est le cas de TCP) ne seront donc pas affectés. Astuce amusante, pour atteindre cet objectif de neutralité, 16 bits de l'adresse sont réservés pour pouvoir y faire les modifications qui annuleront les autres changements faits dans l'adresse. L'algorithme exact de calcul est dans les sections 3.1 et suivantes. Une mise en œuvre en C figure en annexe B. À noter qu'elle ne prétend pas être optimale, faisant par exemple les calculs de somme de contrôle avec du code portable et pas de l'assembleur adapté à une machine particulière.

On peut avoir des cas plus compliqués que ce simple réseau, par exemple du "*multi-homing*", où le réseau local est connecté à deux FAI (section 2.4). Chacun de ces deux FAI alloue un préfixe PA différent. Les liens avec les deux FAI passent par deux routeurs NPT différents, tous les deux configurés avec le même préfixe interne mais différents préfixes externes. Une machine du réseau local ne pourra pas savoir sous quelle adresse elle apparaitra à l'extérieur, sauf à utiliser une technique comme STUN (RFC 8489). La décision de sortir par un FAI ou l'autre peut être prise comme on veut. Par contre, par rapport à du vrai "*multi-homing*", avec adresses PI et BGP, un changement de FAI de sortie entraîne un changement de l'adresse IP vue par l'extérieur et coupe donc toutes les sessions en cours.

Continuons avec les considérations de déploiement et de configuration (section 4). Le plus probable est que NPTv6 sera mis en œuvre dans les routeurs, comme dans l'exemple ci-dessus, et, pour les particuliers et petites organisations dans le CPE. Les obligations du RFC 7084 s'appliquent donc à l'engin qui fait la traduction.

Cela implique entre autres que le traducteur NPT soit capable de faire des virages en épingle à cheveux (renvoyer vers le réseau local un paquet qui était à destination du réseau local, cf. RFC 4787), afin que des machines du réseau local puissent se parler en utilisant leurs adresses publiques. Comme NPT ne tripote pas les ports, la plupart des autres exigences des RFC du groupe BEHAVE `<https://www.bortzmeyer.org/behave-wg.html>` ne s'appliquent pas à lui.

Et pour les applications, quelles sont les conséquences (section 5)? Plusieurs des problèmes classiques de la traduction, qui avaient déjà été décrits dans le RFC 2993 sont toujours présents. Les applications ne verront pas les mêmes adresses, selon qu'elles sont situées d'un côté ou de l'autre du traducteur. Par exemple, si un ordinateur portable se déplace de part et d'autre du traducteur, il verra ses connexions s'interrompre, son adresse IP ayant changé. Mais les problèmes les plus fréquents et les

plus sérieux seront pour les protocoles applicatifs qui utilisent des **références**, c'est-à-dire qui indiquent dans les données à quelle adresse IP envoyer les paquets. Les cas les plus connus sont FTP et SIP. Si un client SIP envoie à un autre son adresse interne, pour recevoir un flux RTP, cette adresse ne sera pas traduite, et le flux ne joindra pas la destination. Que peut-on faire pour limiter les dégâts ?

— Comme avec le NAT44, les applications peuvent utiliser STUN pour apprendre leur adresse,

— Dans le futur, un mécanisme de référence général, résistant aux traductions, sera peut-être développé.

Et la sécurité du NPTv6 ? La section 7 résume les points importants. Le principal est que, contrairement au NAT classique, avec état, NPT n'empêche pas les connexions entrantes. C'est une bonne chose mais cela peut dérouter certains administrateurs réseaux qui croyaient que la traduction les protégeait de l'extérieur. Ce n'est pas le cas avec NPTv6 et si on veut une telle protection, il faut installer un pare-feu explicite (alors que le NAT traditionnel mélange les deux rôles de traduction et de protection), comme décrit dans le RFC 6092.

Curieusement, cette idée de traduction d'adresses sans état, selon une cardinalité 1 :1, est très ancienne. Si notre RFC 6296 semble être le premier RFC à la décrire, elle avait été à la base du projet GSE, auquel l'annexe A rend hommage. GSE, décrit dans le "*draft*" draft-ietf-ipngwg-gseaddr <http://tools.ietf.org/id/draft-ietf-ipngwg-gseaddr>, était un des projets qui avaient été élaborés pour le successeur d'IPv4. Contrairement au projet retenu (le futur IPv6) qui conservait les concepts de base de son prédécesseur, GSE repensait nettement le problème de l'adressage dans l'Internet. Celui-ci était divisé en deux, les réseaux de transit et les réseaux de bordure. Dans GSE, seuls les réseaux de transit avaient des adresses publiques et annoncées dans la table de routage globale. Les réseaux de bordure avaient des adresses stables et uniques mais qui n'étaient pas routées. Les adresses étaient traduites lors du passage du réseau de bordure au réseau de transit et réciproquement. Le but était surtout de limiter la croissance de la table de routage (problème qu'IPv6 avait finalement décidé de ne pas traiter). En décembre 2010, sur l'Internet, il y a 36 000 systèmes autonomes dont 15 000 n'annoncent qu'un seul préfixe (et sont donc probablement des réseaux de bordure). Seuls 5 000 systèmes autonomes apparaissent dans des chemins (en dehors de l'origine) et sont donc des réseaux de transit.

Et les implémentations publiquement disponibles ? (Merci à François Romieu et Samuel Thibault pour leurs informations.) Il y a map66 <http://map66.sourceforge.net/> et aussi nfnat66 (NAT66 pour Netfilter) <http://nfnat66.sourceforge.net/>, qui utilise netfilter et est présenté en français dans l'exposé de Guy Leduc <http://freefr.dl.sourceforge.net/project/nfnat66/NAT66_slides.pdf> et plus en détail dans la thèse de Terry Moës, « "*IPv6 address translation in a Linux kernel*" <http://freefr.dl.sourceforge.net/project/nfnat66/MOES%20Terry%20-%20Master%20Thesis%20-%20NAT66.pdf> ». Ne m'en demandez pas plus, je n'ai pas testé. (À noter qu'il en existe aussi pour des techniques de traduction IPv6¡-¿IPv6 différentes, par exemple avec état.)

Mais c'est une autre mise en œuvre qui a été intégrée dans le noyau Linux en août 2012 ("*commit*" 8a91bb0c304b0853f8c59b1b48c7822c52362cba) et qui a donc été livrée avec la sortie de la version 3.7 en décembre 2012. Regardez les fichiers en net/ipv6/netfilter comme ip6table_nat.c.