RFC 7915 : IP/ICMP Translation Algorithm

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 22 juillet 2016

Date de publication du RFC : Juin 2016

https://www.bortzmeyer.org/7915.html

Parmi les nombreuses techniques de coexistence d'IPv4 et IPv6, l'ex-groupe de travail Behave <http://tools.ietf.org/wg/behave/> développait un mécanisme de traduction permettant à une machine IPv6 de communiquer avec une machine v4 et réciproquement. Une des pièces de ce mécanisme (désormé transféré au groupe de travail v6ops <http://tools.ietf.org/wg/v6ops/>) est l'algorithme de traduction, présenté dans ce RFC 7915[1]. (Il remplace l'ancien RFC 6145, avec assez peu de changements.)

Ce nouveau mécanisme de traduction entre IPv4 et IPv6 s'inscrit dans le cadre décrit dans le RFC 6144. Il vise à remplacer « NAT-PT » (RFC 2765 et RFC 2766), officiellement retiré, après n'avoir eu aucun succès.

Plus spécifiquement, notre RFC 7915 est la version **sans état** du traducteur, c'est-à-dire que le traducteur peut traiter chaque paquet indépendamment des autres. Le traducteur n'a pas de table des flots en cours. S'il redémarre et perd toute mémoire, pas de problème, il peut immédiatement reprendre son travail. Cette technique permet notamment le **passage à l'échelle** : un traducteur d'adresses peut gérer un très grand nombre de clients sans épuiser sa mémoire. Et on peut mettre plusieurs traducteurs en parallèle, sans coordination entre eux : les paquets d'un même flot peuvent passer par des traducteurs différents. Mais cette technique a aussi quelques inconvénients comme le fait que les fragments ICMP ne seront pas traduits. Il existe donc également une version **avec état** du nouveau mécanisme, normalisée dans le RFC 6146. La section 1.3 discute plus en détail ces deux possibilités, avec ou sans état.

Donc, le NAT64 (son nom non officiel) vise à traduire des adresses IP entre deux réseaux, l'un v4 et l'autre v6. Comme notre RFC 7915 ne traite que le cas sans état, les adresses doivent être converties de

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc7915.txt

manière purement algorithmique, sans référence à leur histoire (cf. RFC 6052, et la section 6 de ce RFC, pour ces algorithmes).

Dit comme ça, cela a l'air simple mais la traduction entre deux protocoles différents porte pas mal de pièges. Ainsi, les en-têtes de paquet n'ont pas la même taille en v4 (20 octets ou plus) et en v6 (40 octets, fixe), ce qui fait que des problèmes de MTU peuvent se poser (section 1.4). Le traducteur doit se comporter comme un routeur, donc fragmenter les paquets trop gros (en IPv4) ou retourner un message ICMP « "*Packet too big*" ».

La section 4 du RFC décrit en détail les opérations que doit faire le traducteur depuis IPv4 vers IPv6 (rappelez-vous que la traduction des adresses v4 en v6 est essentiellement dans le RFC 6052, section 2). Il y a des détails que ne connaissent pas les routeurs NAT44, qui se contentent de changer adresses et ports. En effet, ici, il faut traduire tout l'en-tête. Je ne vais pas résumer complètement cette section, juste pointer quelques pièges possibles. Ainsi, les en-têtes n'étant pas les mêmes en v4 et en v6, notre RFC doit spécifier quoi en faire. Certains cas sont évidents (comme le champ Longueur), d'autres moins. Ainsi, le champ TOS de IPv4 doit être copié verbatim dans le champ "*Traffic Class*" de IPv6. Mais le traducteur doit aussi offrir une option pour ignorer le TOS et mettre systématiquement zéro comme "*Traffic Class*". Le champ TTL de IPv4 doit être copié dans "*Hop limit*" mais **après** avoir été décrémenté (rappelez-vous que le traducteur est un routeur).

La vraie difficulté n'est pas tellement dans la traduction d'IP que dans celle d'ICMP. En effet, le paquet ICMP contient le début du paquet IP qui a motivé son envoi, et ce début de paquet doit être traduit, également, sans quoi le destinataire du paquet ICMP n'y comprendra rien (par exemple, sans traduction ICMP, il recevrait en IPv6 un paquet ICMP dont le contenu est un paquet IPv4...). Notre RFC détaille donc les traductions à faire pour tous les modèles de paquets ICMP.

Le traducteur doit aussi veiller au champ Type d'ICMP, dont les valeurs sont différentes entre IPv4 et IPv6 (par exemple, "*Echo Reply*" est 0 en v4 et 129 en v6). Certains types n'ont pas d'équivalent (comme les types "*Timestamp*" ou "*Address Mask*" d'ICMPv4, absents d'ICMPv6) et le paquet ICMP doit donc être abandonné.

Enfin, le traducteur doit aussi prendre en charge les en-têtes de la couche 4 car la traduction des adresses peut ne pas être neutre pour la somme de contrôle (section 4.5) et il peut donc être nécessaire de recalculer cette dernière.

Et en sens inverse ? La section 5 décrit la traduction d'IPv6 vers IPv4. On retrouve TOS et "*Traffic Class*", les problèmes de MTU et de fragmentation, et la traduction des messages ICMP.

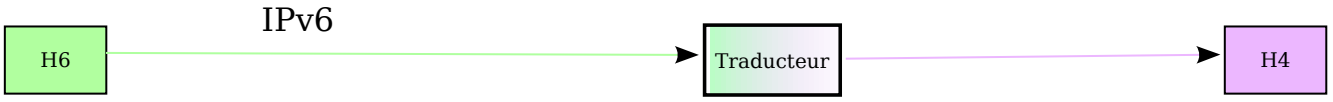
La section 6 de notre RFC, une nouveauté depuis le RFC 6145, revient sur les algorithmes de correspondance entre les adresses IPv4 et IPv6. Les algorithmes obligatoires sont décrits dans le RFC 6052, mais un traducteur peut en ajouter d'autres.

Les problèmes de MTU et de fragmentation sont des cauchemars habituels sur l'Internet, notamment en raison d'équipements intermédiaires (typiquement des pare-feux) programmés avec les pieds par des anonymes, ou bien configurés n'importe comment, et qui bloquent les paquets ICMP, voire les modifient, en ignorant complètement le RFC 4890. Les traducteurs v4¡-¿v6 auront donc ce problème, comme tant d'autres techniques utiles. La section 7 doit donc se pencher sur le traitement des paquets ICMP "*Packet too big*" qui signalent normalement que la MTU est plus réduite que ne le croit l'émetteur et qu'il faut fragmenter. Comme ces paquets sont parfois interceptés, voire modifiés, par des machines gérées par des incompétents, le traducteur doit donc parfois faire des efforts spécifiques. Si les paquets sont interceptés,

la détection de la MTU du chemin n'est plus possible (RFC 2923) et il ne restera plus que la solution du RFC 4821 (faire faire le travail par les couches supérieures).

Rien n'étant parfait en ce bas monde, les traducteurs NAT64 vont aussi introduire de nouvelles questions de sécurité. La section 8 tente de prévoir lesquelles mais reste optimiste en considérant que la plupart des problèmes existent déjà dans IPv4 ou dans IPv6. Ainsi, comme avec le NAT traditionnel, l'authentification des paquets par IPsec (RFC 4302) ne marchera pas.

Pour les gens qui aiment bien les exposés concrets, avec des 0 et des 1, l'annexe A explique en grand détail le processus de traduction avec un réseau simple d'exemple, connectant le réseau traditionnel `198.51.100.0/24` et le réseau nouveau `2001:db8::/32`, via un traducteur. Deux machines, H4 et H6, Alice et Bob du NAT64, vont tenter de communiquer. Le traducteur utilise le préfixe `2001:db8:100::/40` pour représenter les adresses IPv4 et `192.0.2.0/24` pour représenter les adresses IPv6. Ainsi, H6, qui a l'adresse `2001:db8:1c0:2:21::` est représenté en v4 par `192.0.2.33`. H4 a l'adresse `198.51.100.2`. Une fois le routage correctement configuré pour passer par le traducteur, sui-



vez le RFC pour voir le trajet des paquets et les opérations qu'ils subissent, dans le cas où c'est H6 qui initie la connexion, et dans le cas inverse.

Au moins deux mises en œuvre existent déjà, faite par Ecdysis/Viagénie <`http://ecdysis.viagenie.ca/`> (avec état) et Tayga <`http://www.litech.org/tayga/`> (sans état, mais nettement plus simple, d'après ses utilisateurs). Pour compenser l'absence de traduction avec état chez Tayga, on peut, sur Linux, le coupler avec SNAT/MASQUERADE (merci à Guillaume Leclanché pour sa suggestion).

Quels sont les changements entre ce mécanisme et celui du RFC 6145 ? Rien de dramatique. Ils sont résumés en section 2. Certains traitent les bogues détectées dans l'ancien RFC <`http://www.rfc-editor.org/errata_search.php?rfc=6145`>. D'autres changements tiennent compte du conseil actuel qui est de ne plus compter sur les « fragments atomiques » du RFC 6946 (ils ont même été franchement abandonnés avec le RFC 8021.)

Pour réfléchir à la grande question « avec état ou sans état », un article « pro-état » : « "*Stateless NAT64 is useless*" <`http://blog.ioshints.info/2011/05/stateless-nat64-is-useless.html`> ».