RFC 7059 : A Comparison of IPv6 over IPv4 Tunnel Mechanisms

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 27 novembre 2013

Date de publication du RFC : Novembre 2013

https://www.bortzmeyer.org/7059.html

Il existe d'innombrables techniques <https://www.bortzmeyer.org/transition-ipv6-guilde.html> pour faire coexister IPv4 et IPv6 sur l'Internet. Tellement qu'on s'y perd facilement. Ce nouveau RFC se concentre sur une catégorie particulière, les tunnels « IPv6 sur IPv4 » et fait la liste de tous les mécanismes de cette catégorie (des plus répandus aux plus exotiques), avec leurs forces et leurs faiblesses.

Ces tunnels sont dictés par la nécessité. La bonne méthode pour se connecter en IPv6 est clairement d'utiliser une **connexion native**. Mais on n'a pas toujours le choix. Aujourd'hui, depuis de nombreuses années, et sans doute encore pour un certain temps, il existe de nombreuses îles IPv6, séparées les unes des autres par des réseaux purement IPv4. Par exemple, vous avez loué une machine virtuelle chez un fournisseur qui est resté à l'ancien protocole (comme Numergy) mais vous voulez accéder à l'Internet IPv6. Ou bien vous avez déployé IPv6 sur votre campus mais votre opérateur réseau n'est toujours pas capable de fournir de l'IPv6 ce qui vous désespère. Dans ces deux cas, et dans plusieurs autres, vous serez sans doute obligé d'utiliser un tunnel. Un tunnel fonctionne en encapsulant les paquets d'un protocole dans ceux d'un autre protocole. Ainsi, pour transporter de l'IPv6 sur l'IPv4, le routeur d'entrée de tunnel met le paquet IPv6 à l'intérieur d'un paquet IPv4, celui-ci voyage ensuite par les mécanismes IPv4 habituels, sur un réseau qui ne connait qu'IPv4 et, à l'arrivée sur le routeur de sortie de tunnel, le paquet IPv6 est décapsulé (extrait du paquet IPv4) puis continue son chemin dans le réseau IPv6.

Ce principe est le même pour toutes les techniques de tunnels. Mais les nombreuses techniques existent diffèrent par bien d'autres points, ce qui sème souvent la confusion chez les administrateurs réseau. D'autant plus que ces techniques ne se valent pas : certaines posent des gros problèmes de sécurité ou de fiabilité.

Ce RFC fait le tour de ces techniques. Attention : il ne couvre que le cas « IPv6 tunnelé dans IPv4 ». Il existe plein d'autres techniques de tunnels, pour faire des VPN par exemple. De même, ce RFC 7059 [1] ne parle pas de DS-Lite, qui n'est pas une technologie de IPv6 sur IPv4 mais, au contraire, un moyen de transporter l'IPv4 sur des réseaux purement IPv6.

La section 3 est le gros morceau du RFC, contenant la liste de tous les mécanismes de tunnels étudiés (je ne les ai pas tous repris dans cet article). La plupart des tunnels font une encapsulation directe : pas d'intermédiaire entre l'en-tête IPv4 et le paquet IPv6. L'en-tête IPv4 a un champ « Protocole » qui contient la valeur 41, identifiant IPv6 (cf. section 5.1). L'adresse IPv6 des extrémités du tunnel est parfois automatiquement créée en fonction de l'adresse IPv4 (tunnels automatiques), pour trouver facilement l'extrémité du tunnel (ce point est détaillé en section 5.4). Au contraire, dans les tunnels manuels, il a fallu configurer explicitement les paramètres du tunnel (notamment l'adresse IPv4 de sortie). Un cas intermédiaire est celui où le tunnel est manuel mais la configuration se fait via un protocole auxiliaire de gestion du tunnel, qui dispense l'utilisateur de cette tâche.

D'autres tunnels ne font pas une encapsulation directe : ils mettent l'IPv6 dans de l'UDP qui est ensuite transporté sur IPv4. Cela permet la traversée des NAT et résout le problème de l'ossification de l'Internet v4, où seuls UDP et TCP arrivent à passer, les autres protocoles de couche 4 (comme le 41) étant de facto interdits en beaucoup d'endroits.

Commençons par les tunnels manuels, les plus anciens (ils étaient déjà dans le RFC 1933 en 1996). Leur norme actuelle est le RFC 4213. On les nomme aussi tunnels statiques ou bien 6in4. Le principe est de désigner explicitement, sur chaque point d'entrée, quel est le point de sortie du tunnel. Pour des exemples de configuration de tels tunnels, voir mes articles « Connecter un serveur dédié à IPv6 avec un tunnel manuel <https://www.bortzmeyer.org/ipv6-he.html> » et, plus compliqué « Un tunnel IPv6-in-v4 sur un tunnel GRE... <https://www.bortzmeyer.org/tunnel-over-tunnel.html> ». Cette configuration manuelle rend cette solution « Michu-"*hostile*" » mais elle a des avantages : le réseau est prévisible (on sait exactement où les paquets vont passer) et facile à déboguer. À noter que la configuration peut être simplifiée par l'utilisation d'un courtier <https://www.bortzmeyer.org/tunnel-broker.html> ("*broker*"). Les performances vont dépendre du choix de l'autre extrémité du tunnel (dans mon exemple au Cameroun <https://www.bortzmeyer.org/tunnel-over-tunnel.html>, elle était à Londres, nous n'avions rien trouvé de plus proche). Autrefois, il était courant que le tunnel s'étende sur deux continents différents, allongeant sérieusement le RTT. Ces mauvais choix (tunnel trop long) ont souvent donné une mauvaise réputation aux tunnels. À tort : à titre personnel, je trouve qu'un tunnel manuel est une solution simple, fiable et efficace pour se connecter en IPv6 si on n'a pas de fournisseur IPv6 sous la main. Le seul piège est qu'il faut bien choisir son fournisseur de tunnel.

On peut aussi utiliser GRE (RFC 2784), qui est très répandu dans les routeurs (mais pas dans les machines terminales typiques). C'est un protocole d'encapsulation très généraliste (IPv4 sur IPv4, IPv6 sur IPv4, etc).

GRE est ultra-simple, avec son RFC de moins de neuf pages. Trop dans certains cas, alors on peut lui préférer SEAL (dont le RFC n'a pas encore été publié) qui prévoit quelques services supplémentaires dont un protocole de contrôle permettant aux deux extrémités du tunnel de dialoguer. Un autre exemple de « GRE++ » est AYIYA (pas encore de RFC non plus). Notez que SEAL, contrairement à GRE, n'a pas encore connu beaucoup d'utilisations.

Comme la nécessité d'une configuration manuelle refroidit beaucoup de gens et peut sembler un frein au déploiement d'IPv6, il existe des solutions de tunnels automatiques. Par exemple, le RFC 2893

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc7059.txt

décrivait une solution (supprimée depuis) où les adresses IPv6 étaient des adresses « compatibles IPv4 » (par exemple `::192.0.2.1`, alias `::c000:201`, équivalent IPv6 de `192.0.2.1`). Le gros inconvénient de cette solution est qu'elle ne marchait qu'entre machines ayant cette technologie, et pas avec l'Internet IPv6. Elle n'a donc plus de rôle aujourd'hui.

Au contraire, 6to4 (RFC 3056) est très répandu (on le trouve dans plusieurs routeurs CPE). Il fonctionne automatiquement, en mettant l'adresse IPv4 du tunnel dans une adresse IPv6 préfixée par `2002::/16`, et suivie de l'adresse IPv4. 6to4 dépend de relais (en général gérés bénévolement) capables de servir de point d'entrée et de sortie du tunnel. Grâce à l'"*anycast*" (RFC 3068) dont 6to4 avait été un des premiers utilisateurs, plusieurs relais sont accessibles pour un préfixe donné. Ils ont tous l'adresse IPv4 publique `192.88.99.1` (`2002:c058:6301::` en IPv6). La route vers `2002::/16` est annoncée vers l'Internet IPv6 par tous les relais et le plus « proche » est sélectionné, répartissant ainsi automatiquement le travail. Sans configuration manuelle, 6to4 est bien adapté au petit réseau qui veut se connecter rapidement. Malheureusement, 6to4 est très imprévisible : les relais sont variés dans leur sérieux et la qualité de leur connexion, et on ne sait pas lequel on va utiliser. Le routage est en général asymétrique (on utilise un relais différent à l'aller et au retour) ce qui rend le débogage des problèmes de connectivité difficile. Le RFC 6343 liste les problèmes de 6to4 et ne recommande pas son usage. Le RFC 7526 est allé plus loin en abandonnant officiellement 6to4.

Pour résoudre ces problèmes sérieux de 6to4, certains FAI (comme Free en France) ont déployé 6rd (RFC 5969). 6rd leur permet de déployer IPv6 pour leurs clients, en ne changeant qu'une partie du réseau, sans qu'il soit nécessaire qu'il fonctionne intégralement en IPv6. 6rd ressemble beaucoup à 6to4 mais n'utilise pas le préfixe commun `2002::/16`, mais un préfixe spécifique au FAI (ce qui veut dire que, dans le journal d'un serveur, on ne repère pas les clients 6rd, contrairement aux clients 6to4). Ce préfixe doit être envoyé au client, par exemple en DHCP. À noter que, comme les clients 6rd d'un même FAI partagent en général un préfixe IPv4 commun, il n'est pas nécessaire d'encoder tous les 32 bits de l'adresse IPv4 dans l'adresse IPv6, ce qui libère quelques bits (section 4 du RFC 5969). Si, contrairement à 6to4, 6rd ne peut pas être déployé par l'utilisateur seul, il a par contre l'avantage d'être bien plus prévisible et facile à déboguer. La responsabilité de la connectivité est bien plus claire, elle est entièrement chez le FAI, sans avoir besoin d'impliquer des relais extérieurs.

Comme 6to4, 6rd est sans état et les routeurs relais peuvent donc utiliser l'"*anycast*".

6to4 et 6rd utilisent l'encapsulation directe, où le paquet IPv6 est mis directement dans IPv4, ce dernier l'indiquant par le numéro de protocole 41. L'un des inconvénients que cela présente est que cela empêche la traversée des NAT. Un autre protocole de tunnel, Teredo (RFC 4380), résout le problème en ajoutant UDP. On a donc IPv6-dans-UDP-dans-IPv4. Cela permet aussi d'avoir plusieurs clients derrière le même routeur NAT. Teredo étant activé par défaut dans certaines versions de Windows, son usage est répandu. Teredo inclut le port UDP, avec les adresses IPv4 du tunnel, dans l'adresse IPv6, qui est préfixée par `2001:0::/32`.

Du point de vue de la fiabilité et des performances, Teredo est pire que 6to4, comme l'illustre l'article « "*Testing Teredo*" <http://www.potaroo.net/ispcol/2011-04/teredo.html> ».

Une solution de tunnel bien plus exotique et rare est LISP (RFC 6830). LISP n'a pas été spécialement conçu pour mettre de l'IPv6 dans l'IPv4, il est une solution générale de séparation de l'identificateur et du localisateur <https://www.bortzmeyer.org/separation-identificateur-localisateur.html>. Les identificateurs sont nommés EID dans LISP et les localisateurs RLOC. Tous les deux ont la forme d'une adresse IP. On peut avoir un EID IPv6 et un RLOC IPv4, réalisant ainsi un tunnel IPv6-sur-IPv4. Donc, LISP permet de faire nos tunnels mais c'est un protocole riche et complexe et l'utiliser uniquement pour cela semble exagéré.

Parmi les autres tunnels possibles, c'est dans ce RFC que j'ai appris l'existence de 6bed4 <http://devel.0cpm.org/6bed4/>. Son originalité est de fournir un mécanisme pour débrayer automatiquement le tunnel si le correspondant est joignable en IPv6 natif, par exemple s'il est sur le même réseau local. Cela lui permet d'atteindre des performances plus proches de celles de l'IPv4. Comme Teredo, 6bed4 met dans ses adresses IPv6 les adresses IPv4 et les numéros de ports UDP des routeurs du tunnel.

Les mécanismes de tunnel utilisent souvent des mécanismes auxiliaires, qui ne sont pas des tunnels mais qui aident à leur établissement et à leur gestion. La section 4 fait le tour des principaux. On y trouve par exemple le TSP du RFC 5572, qui permet de configurer automatiquement un tunnel, évitant l'étape « lecture de la doc' et tentative de la recopier ». Ce mécanisme est par exemple utilisé par Freenet6 <http://www.gogo6.com/freenet6> et des exemples figurent dans mon article sur les serveurs de tunnel <https://www.bortzmeyer.org/tunnel-broker.html>.

Un inconvénient des serveurs de tunnel se présente lorsque le client change d'adresse IPv4 (cas d'une adresse dynamique dans certains abonnements). Avant, il fallait arrêter le tunnel et en créer un nouveau. Le protocole "*SixXS Heartbeat*" permet d'éviter cela : le client envoie régulièrement des paquets au serveur de tunnel, qui peut ainsi apprendre un changement d'adresses et se reconfigurer. Le serveur de SixXS <http://www.sixxs.net/> fait cela, et les clients typiques aussi. À noter qu'AYIYA inclut cette fonction de « battement de cœur ».

Enfin, après TSP, un autre protocole de négociation de paramètres et de création de tunnel est TIC, également utilisé à SixXS <http://www.sixxs.net/tools/tic/>. Il a notamment été mis en œuvre dans un petit routeur CPE très populaire en Allemagne et aux Pays-Bas, le Fritz !Box AVM.

La section 5 de notre RFC discute les aspects communs à tous (ou en tout cas à une bonne partie) de ces mécanismes de tunnel. Par exemple, les routeurs NAT (plus exactement NAPT car ils changent le port, pas seulement l'adresse IP, et doivent donc connaître le protocole de couche 4 utilisé) et les pare-feux sont une cause fréquente de problème pour les tunnels, comme ils gênent d'ailleurs bien d'autres services. Ainsi, le protocole de « transport » 41 (encapsulation directe d'IPv6 dans IPv4) est souvent bloqué, ce qui a mené à l'utilisation d'UDP (par exemple par Teredo), pour contourner ce blocage. Puisqu'il n'a pas de port, le protocole 41 ne peut pas passer à travers un routeur NAPT. Il pourrait passer à travers un routeur NAT (rappelez-vous que la plupart des équipements NAT sont en fait du NAPT) dans certaines conditions. Mais, si l'adresse IPv6 est dérivée de l'IPv4, la traduction d'adresses va certainement casser le tunnel. C'est le cas de 6to4 et 6rd (6rd fonctionne en général car il ne traverse pas le routeur NAPT : il démarre sur ce routeur, qui est le point d'entrée du tunnel).

Par contre, GRE et les tunnels manuels peuvent fonctionner à travers un NAT. Il y a parfois des surprises et il peut être préférable d'utiliser un mécanisme prévu dès le début pour traverser le NAT, comme Teredo, AYIYA, ou 6bed4.

Et puis, bien sûr, une plaie récurrente de tous les tunnels est la question de la MTU (section 5.3). En raison de l'encapsulation, **tout** mécanisme de tunnel diminue la MTU effective de quelques octets. Normalement, la fragmentation et la découverte de la MTU du chemin devraient gérer cela et permettre au trafic de passer à travers le tunnel. En pratique, le nombre de pare-feux mal configurés qui bloquent les paquets ICMP nécessaires à la découverte de la MTU (message ICMP « "*Packet Too Big*" ») est tel que les problèmes sont fréquents. Si l'extrémité du tunnel est sur la machine terminale, celle-ci peut encore réussir à communiquer avec TCP, la MSS de ce dernier s'ajustera. Sinon, on aura des problèmes à première vue mystérieux comme le fait qu'un ping ordinaire passe mais pas un ping avec une taille différente <https://www.bortzmeyer.org/ping-taille-compte.html>. Ou bien on verra les connexions TCP s'établir, le client HTTP envoyer sa requête mais la réponse, plus grande et ne

tenant pas dans la MTU, n'arrivera jamais. Ces problèmes liés à la MTU sont une des plaies de l'Internet <https://www.bortzmeyer.org/mtu-et-mss-sont-dans-un-reseau.html> et l'utilisation des tunnels les rend encore plus fréquents.

On le voit, la liste des solutions techniques pour tunneler IPv6 dans IPv4 est longue (et encore, je n'ai pas cité dans cet article tous ceux que mentionne le RFC). Comment choisir ? La section 6 du RFC est consacrée à l'évaluation de ces solutions (l'annexe A donne la iste des critères utilisés). D'abord, l'usage qu'ils font des adresses IPv4, celles-ci étant désormais très rares <https://www.bortzmeyer.org/epuisement-adresses-ipv4.html>. Les tunnels manuels, qui dépendent d'une adresse IPv4 fixe et unique, ainsi que 6to4, ne peuvent pas marcher à travers un CGN, lorsque plusieurs clients se partagent une adresse IPv4. Teredo ou AYIYA, au contraire, ont été explicitement conçus pour bien marcher même à travers les pires NAT.

Deuxième critère d'évaluation, la topologie réseau permise. Certains tunnels (par exemple les tunnels manuels) sont point à point, entre deux machines fixes, les routeurs d'entrée et de sortie du tunnel. Cela facilite le débogage car le cheminement du trafic est parfaitement prévisible. D'autres (comme 6to4) sont plutôt un vaste réseau où plusieurs relais peuvent être utilisés pour fournir un lien virtuel qui est NBMA plutôt que point à point. Cela offre plus de souplesse et ne fait pas des deux routeurs d'extrémité du tunnel des SPOF.

En pratique, la section 6.2 du RFC penche nettement vers la première solution, une liaison point à point, qui colle bien au modèle traditionnel suivi par les liens physiques, et qui établit clairement les responsabilités de chaque acteur. Bien que l'autre topologie soit séduisante sur le papier, elle a en pratique entraîné beaucoup de problèmes de performance et de débogage.

Et à propos de SPOF, quelle est la fiabilité de ces techniques de tunnels lors d'une utilisation quotidienne ? L'expérience montre que les tunnels manuels sont plutôt fiables (une fois configuré, il n'y a guère de raison qu'ils arrêtent de marcher) et, surtout, ils sont simples dans leurs problèmes : soit le tunnel marche, soit rien ne passe. Pinguer l'extrémité du tunnel suffit en général à les superviser. D'où le tableau de la section 6.3, qui classe les techniques de tunnel par ordre de fiabilité décroissante, et qui met les tunnels configurés manuellement en haut, et Teredo et 6to4 tout en bas... (LISP est indiqué comme le plus fiable, à cause de ses mécanismes de réparation automatiques mais, contrairement à 6to4 ou aux tunnels, il n'a pas encore beaucoup été testé en vrai.)

Les problèmes de 6to4 ont été aggravés par le fait que certaines mises en œuvre de ce protocole ne testaient même pas que la machine avait une connectivité avec au moins un relais (RFC 6343 et RFC 7526).

Et les performances ? En raison de l'encapsulation, il y a forcément quelques octets perdus par le tunnel. Dans le cas d'une encapsulation directe, la moins coûteuse, cette perte représente 1,3 % d'un paquet de la taille maximale (1 500 octets). À part cette diminution de la charge utile, la performance dépend surtout des routeurs d'entrée et de sortie du tunnel. S'ils traitent les paquets « normaux » dans leur ASIC, mais l'encapsulation et la décapsulation en logiciel, dans leur relativement lent processeur, alors, oui, le tunnel sera lent. Mais ce n'est pas obligatoire. En fait, historiquement, le principal problème de performance des tunnels avait été le fait que les tunnels étaient souvent établis avec des machines relativement lointaines et c'est cet allongement du trajet qui ralentissait le service. Autrement, les tunnels ne sont pas synonymes de lenteur.

Bon, et la sécurité (section 7) ? Les tunnels (pas seulement ceux de IPv6 sur IPv4) sont souvent mauvais sur ce point. Par exemple, si l'entrée du tunnel ne fait rien pour vérifier les adresses IPv6 source des paquets qu'elle encapsule, le tunnel permettra peut-être de contourner les mécanismes anti-usurpation d'adresses (le cas de l'usurpation d'adresses avec 6to4 est couvert dans le RFC 3964). Il est donc important que le routeur d'entrée du tunnel prenne des précautions pour n'accepter que des paquets de ses clients légitimes. Autre faille possible : le tunnel permet d'établir une connectivité qui ne devrait pas « normalement » exister (dans le cas d'IPv6 sur IPv4, c'est son but explicite) et cela peut permettre de contourner les règles de sécurité qui sont en place (RFC 6169).