Aurait-il fallu faire IPv6 « compatible » avec IPv4 ?

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 7 juin 2018

https://www.bortzmeyer.org/ipv6-compatibilite.html

Le déploiement du protocole IPv6 continue, mais à un rythme très réduit par rapport à ce qui était prévu et espéré. À cette vitesse, on aura encore de l'IPv4 dans 20 ou 30 ans. Face à cette lenteur exaspérante, on entend souvent des Monsieur Jesaistout affirmer que c'est la conséquence d'une erreur fondamentale au début de la conception du protocole IPv6 : il aurait fallu le faire « compatible avec IPv4 ». Qu'est-ce que cela veut dire, et est-ce que cela aurait été possible ?

On trouve par exemple ce regret d'une absence de compatibilité dans le livre de Milton Mueller, « "*Will the Internet Fragment ? : Sovereignty, Globalization and Cyberspace*" ». Cet auteur, quoique non-technicien, fait en général très attention à être rigoureux sur les questions techniques et à ne pas écrire de bêtises. Pourtant, là, il en a fait une belle (« "*in a fateful blunder, [...] the IETF did not make it backwards compatible with the old one*" »). Pour comprendre en quoi, voyons d'abord en quoi IPv6 est incompatible avec IPv4. Cela concerne notamment les routeurs et les applications. En effet, le format de l'en-tête des paquets IPv6 est très différent de celui de l'en-tête des paquets IPv4. Un routeur ne connaissant qu'IPv4 ne peut rien faire d'un paquet IPv6, il ne peut même pas l'analyser. IPv6 imposait donc une mise à jour de tous les routeurs (aujourd'hui largement faite, même sur le bas de gamme). Et les applications ? Normalement, une bonne partie des applications n'a pas besoin de connaitre les détails de la couche réseau. Après tout, c'est un des buts du modèle en couches que d'isoler les applications des particularités du réseau. Mais il y a des exceptions (application serveur ayant des ACL et devant donc manipuler des adresses IP, par exemple), et, surtout, beaucoup d'applications ne sont pas écrites avec une API de haut niveau <https://www.bortzmeyer.org/network-high-level-programming.html> : le programmeur ou la programmeuse a utilisé, par exemple, l'API socket, qui expose des tas de détails inutiles, comme la taille des adresses IP, liant ainsi l'application à un protocole réseau particulier. IPv6 impose donc de mettre à jour pas mal d'applications, ce qui est fait depuis longtemps pour les grands logiciels libres connus (Apache, Unbound, Postfix, etc) mais pas forcément pour les petits logiciels locaux développés par l'ESN du coin.

Aurait-on pu s'en tirer en concevant IPv6 différemment ? En gros, non. Pour voir pourquoi, il faut repartir du cahier des charges d'IPv6 : le principal problème était l'épuisement des adresses IPv4 <https://www.bortzmeyer.org/epuisement-adresses-ipv4.html>. Il fallait donc des adresses plus longues (elles font 128 bits pour IPv6 contre 32 pour IPv4). Même si cela avait été le seul changement dans le format de l'en-tête des paquets, cela aurait suffit à le rendre incompatible, et donc à obliger

à changer les routeurs, ainsi que les applications dépendant d'IPv4. Regretter que l'IETF ait changé d'autres aspects de l'en-tête, qu'on aurait pu laisser tranquilles, n'a pas de sens : rien que le changement de taille des adresses invalide tout le code IPv4. Cela ne serait pas le cas si les en-têtes des paquets IP étaient encodés en TLV, ou bien dans un autre format avec des champs de taille variable. Mais, pour des raisons de performance (un routeur peut avoir à traiter des centaines de millions de paquets par seconde), les paquets IP ont un encodage en binaire, avec des champs de taille fixe. Toute modification de la taille d'un de ces champs nécessite donc de changer tout le code de traitement des paquets, tous les ASIC des routeurs.

Même en l'absence de ce problème d'encodage « sur le câble », il n'est pas sûr que tous les programmes existants supporteraient le changement de taille des adresses. Combien d'applications anciennes tiennent pour acquis que les adresses IP ont une taille de seulement 32 bits et, si elles sont écrites en C, les mettent dans des `int` (entier qui fait en général 32 bits) ?

Néanmoins, malgré ces faits connus depuis longtemps, on croise relativement souvent des affirmations du genre « l'IETF aurait juste dû rajouter des bits aux adresses mais sans changer le format ». Comme on l'a vu, tout changement de taille des adresses change le format. Et, si on ne change pas la taille des adresses, pourquoi utiliser un nouveau protocole ? À moins que, malgré Shannon, on ne croit à la possibilité de mettre 128 bits dans 32 bits ?

Cela ne veut pas dire qu'IPv4 et IPv6 doivent être incapables de se parler, comme « des navires qui se croisent dans la nuit ». On peut penser qu'une solution de traduction d'adresses permettrait au moins certains échanges. Mais attention à ne pas copier simplement le NAT d'IPv4 : IPv4 utilise les ports de TCP et UDP pour identifier une session particulière et savoir où envoyer les paquets. Il n'y a que 16 bits pour stocker les ports, et cela ne suffirait donc pas pour permettre de représenter toutes les adresses IPv6 dans des adresses IPv4 (il manquerait encore 80 bits à trouver…) Il y a bien des solutions avec traduction d'adresses, comme NAT64 (RFC 6146 [1]) mais elles ne peuvent s'utiliser que dans des cas limités (pour NAT64, entre un client purement IPv6 et un serveur purement IPv4), et entrainent des dépendances supplémentaires (pour NAT64, la nécessité de disposer d'un résolveur DNS spécial, cf. RFC 6147). Bref, enfonçons le clou : il n'existe pas et il ne peut pas exister de mécanisme permettant une compatibilité complète entre un protocole qui utilise des adresses de 32 bits et un protocole qui utilise des adresses de 128 bits. Il y a des solutions partielles (la plus simple, qu'on oublie souvent, est d'avoir un relais applicatif), mais pas de solution complète.

Bien sûr, c'est en supposant qu'on veut garder la compatibilité avec les anciennes machines et logiciels. Si on repartait de zéro <https://www.bortzmeyer.org/repartir-de-zero.html>, on pourrait faire un protocole de couche 3 aux adresses de taille variable, mais ce ne serait plus IP, et un tel protocole serait encore plus difficile et coûteux à déployer qu'une nouvelle version d'IP, comme IPv6.

Est-ce simplement moi qui ne vois pas de solution, ou bien est-ce vraiment un problème de fond ? Jusqu'à présent, plein de gens ont râlé « il aurait fallu faire un IPv6 compatible avec IPv4 » mais je n'ai encore vu **aucune** proposition **détaillée** indiquant comment faire cela. Il y a plein d'idées « dos de l'enveloppe », de ces idées griffonnées en deux minutes mais qui n'iront jamais plus loin. Écrire un tweet, c'est une chose. Spécifier, même partiellement, un protocole, c'est autre chose. On voit par exemple quelqu'un qui sort de son domaine de compétence (la cryptographie) écrire « "*they designed the IPv6 address space as an alternative to the IPv4 address space, rather than an extension to the IPv4 address space*" <http://cr.yp.to/djbdns/ipv6mess.html> »). Mais il n'est pas allé plus loin. Au moins, l'auteur du ridicule projet baptisé IPv10 <https://datatracker.ietf.org/doc/draft-omar-ipv10/> avait fait l'effort

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc6146.txt

de détailler un peu sa proposition (le même auteur avait commis un projet de connecter les satellites par des fibres optiques <https://datatracker.ietf.org/doc/draft-omar-si/>). C'est d'ailleurs le fait que sa proposition soit relativement détaillée qui permet de voir qu'elle ne tient pas la route : le format des paquets (la seule chose qu'il spécifie de manière un peu précise) étant différent, son déploiement serait au moins aussi lent que celui d'IPv6. Le cryptographe cité plus haut, lui, ne s'est même pas donné cette peine.

Si vous n'êtes pas convaincus par mon raisonnement, je vous propose d'essayer de spécifier un « IPv4 bis » qui serait compatible avec IPv4 tout en offrant davantage d'adresses. Vous pouvez écrire cette spécification sous forme d'un "*Internet-Draft*", d'un article scientifique, ou d'un article sur votre blog. Mettre en œuvre n'est pas obligatoire. Envoyez-moi l'URL ensuite, je suis curieux de voir les résultats. (Rappel : les vagues propositions, trop vagues pour être réfutées, ne sont pas acceptées, il faut une vraie spécification, qu'on puisse soumettre à une analyse technique.) Quelques exemples :

— IPv10, déjà cité,
— EnIP <https://datatracker.ietf.org/doc/draft-chimiak-enhanced-ipv4/> (notez son optimisme sur les pare-feux qui, tout à coup, respecteraient ce qu'ils ne connaissent pas).
— Le délirant IPv4+ <https://www.ripe.net/ripe/mail/archives/members-discuss/2020-April/003676.html> d'Elad Cohen, bien démonté par Johann <https://www.mail-archive.com/frnog@frnog.org/msg60034.html>.