RFC 6760 : Requirements for a Protocol to Replace AppleTalk NBP

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 20 février 2013

Date de publication du RFC : Février 2013

https://www.bortzmeyer.org/6760.html

Il y a bien longtemps, à l'époque où même DHCP n'existait pas, et où les seules machines connectées à l'Internet étaient de gros Vax configurés à la main, Apple avait mis au point un ensemble de protocoles concurrents de TCP/IP, la famille AppleTalk. Durant les années suivantes, ces protocoles privés ont été petit à petit remplacé par les normes ouvertes et AppleTalk a quasiment disparu. Mais certains des protocoles de la famille n'ont pas été réellement remplacés et c'est dommage. Parmi eux, NBP, le "*Name Binding Protocol*", qui n'a pas encore d'équivalent standard dans le monde TCP/IP. Que faudrait-il pour le remplacer ? Ce RFC décrit les fonctions souhaitées.

AppleTalk comportait tout un ensemble de protocoles, même si l'insistance des marketroïdes d'Apple à appeler « AppleTalk » le mécanisme physique de connexion LocalTalk a sérieusement brouillé les pistes. Ces protocoles AppleTalk avaient en commun de pouvoir fonctionner sans configuration, dans la configuration dite « bureau du dentiste ». Le dentiste est supposé riche (il a plusieurs ordinateurs) mais n'a pas de compétence en informatique. Il faut donc que tout marche tout seul et NBP jouait un rôle essentiel dans cette auto-configuration.

Cette partie du projet de remplacement d'AppleTalk par IP, concernant l'auto-configuration, a souvent été désignée du nom de « Zeroconf ». Pour développer un équivalent à NBP, il faut le connaître et c'est un des rôles de ce RFC. Un autre rôle est de spécifier plus rigoureusement le remplaçant de NBP. Faire au moins aussi bien était un des buts mais pas le seul. Le protocole dont les propriétés sont décrites dans ce RFC 6760 [1] étend NBP sur plusieurs aspects comme l'internationalisation. AppleTalk utilisait un jeu de caractères privé, Apple Extended ASCII, alors que son successeur se sert évidemment d'Unicode.

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc6760.txt

Un autre changement, moins visible, est que NBP, conçu uniquement pour des LAN était très bavard et que son successeur doit être plus économe des ressources réseau.

Ce RFC 6760 désigne comme successeur de NBP les protocoles "*Multicast DNS*" (RFC 6762, notons que le nom est très mauvais puisque ce protocole n'est pas le DNS) et "*DNS-based service discovery*" (RFC 6763). Ce sont deux protocoles conçus entièrement par Apple, sans participation de l'IETF, et qu'Apple a essayé pendant des années de faire adopter par l'IETF. Le RFC 6760 n'a donc pas uniquempent un rôle technique mais sert aussi de plaidoyer "*pro domo*" d'Apple. Ce RFC 6760 sert donc aussi d'arrière-plan et de justification aux RFC 6761, RFC 6762 et RFC 6763.

Revenons à l'auto-configuration (section 2). Traditionnellement, tous les protocoles TCP/IP avaient été conçus pour des configurations manuelles (par exemple dans le fichier `/etc/network/interfaces` sur une Debian). Cette configuration peut être centralisée, grâce à DHCP (RFC 2131), qui évite de configurer chaque machine. AppleTalk permettait au contraire une auto-configuration des ordinateurs du dentiste, sans qu'aucun administrateur réseaux ne soit intervenu. Bref, AppleTalk était pair-à-pair même si Apple, entreprise costard-cravate, n'utilise guère ce terme, trop connoté ≪ pirate ≫.

IP dispose désormais de mécanismes d'auto-configuration, comme les adresses lien-local (RFC 4862 et RFC 3927). Il reste donc à auto-configurer le mécanisme de résolution de noms en adresses.

La section 3 donne la liste des exigences pour le nouveau protocole. Elle découle des fonctions qu'assurait NBP. La plus connue et la plus spectaculaire est le choix d'un service réseau via le "*Chooser*" (Sélecteur) du Macintosh mais la section 3.1 nous rappelle que ce n'est qu'une extension : le premier service de NBP n'est pas le butinage, c'est la traduction de nom en adresse. À chaque fois que le dentiste veut imprimer, NBP est appelé pour traduire le nom de l'imprimante sélectionnée en une adresse, vers laquelle on va ensuite établir une session, avec le protocole d'impression.

Deuxième point important de NBP (section 3.2), il nomme des services pas des machines. Supposons que le même ordinateur porte un service de fichiers et un service d'impression, le dentiste Michu se moque de savoir si les deux services sont sur la même machine ou pas. Seul l'administrateur réseaux se préoccupe des machines, l'utilisateur pense aux services. Même si on est administrateur réseaux, d'ailleurs, on ne communique pas avec des machines mais avec le logiciel qui tourne sur ces machines. Comme le dit le RFC, ≪ on ne pingue pas une machine, on pingue le logiciel sur cette machine qui met en œuvre le RFC 792 ≫. Conséquence, les machines n'ont pas de nom dans AppleTalk, seuls les services en ont.

Le RFC rappelle d'ailleurs que le DNS a déjà cette possibilité de nommer des services et pas des machines. On ne sait pas si `www.example.org` et `ldap.example.org` sont sur la même machine et, d'ailleurs, on s'en fiche (sauf si on est l'administrateur réseaux en train de déboguer). Toutefois, cette approche a une limite, c'est que le DNS ne fournit en échange du nom qu'une adresse IP. Pour se connecter au service, il faut encore un numéro de port. Parfois, il est bien connu, fixé en dur dans le protocole (80 pour HTTP). Si on veut deux services HTTP distincts (par exemple `www.example.org` et `www.makemoney.example`), c'est à l'application de gérer le démultiplexage (dans le cas de HTTP, avec le champ `Host:` de la requête). Celle-ci double alors un travail que le noyau sait déjà faire. Et même ce démultiplexage par l'application ne résout pas tout : par exemple il est difficile de faire tourner deux serveurs HTTP différents (mettons Apache et nginx) sur la même machine. Parfois, le port est indiqué explicitement par l'utilisateur, par exemple `http://www.example.org:8042/`, ce qui n'est pas pratique et annule une bonne partie de l'intérêt d'avoir un système d'annuaire. Et puis le nombre de ports est limité (65536) et le fait d'avoir des ports fixes n'aide pas à la gestion de cette ressource limitée.

NBP, au contraire (section 3.3) fournissait systématiquement adresse **et** port en échange d'un nom (le port était nommé "*socket number*"). Son remplaçant doit donc garder la même propriété.

Mes lecteurs qui connaissent le DNS, à ce stade, sont déjà devenus tout rouges en criant « Mais il suffit d'utiliser le RFC 2782 ! » Mais ces enregistrements SRV ne résolvent pas tout : ils prennent en entrée un nom de domaine alors que, pour un service d'impression, on veut en plus utiliser d'autres éléments comme le nom de la file d'attente lpr. (Je suis personnellement sceptique par rapport à cet argument : si le serveur d'impression a deux files, `library` et `secretariat`, pourquoi ne pas avoir deux noms, `library.printer.example.org` et `secretariat.printer.example.org`, chacun avec son SRV ?)

NBP permettait également de chercher un service par **type** et par **zone**. La syntaxe complète était `Nom:Type@Zone` où les valeurs pouvaient être remplacées par des signes égal pour indiquer qu'on recherchait tout (voir section 3.10). Le `Type` indiquait le service recherché (par exemple, pour des raisons historiques, le type `LaserWriter` désignait tout service parlant PostScript au dessus du protocole PAP, même si ce service n'était pas une imprimante laser). Quant à la `Zone`, elle fournissait un mécanisme permettant de répartir les grands réseaux AppleTalk en plusieurs zones dans lesquelles la recherche se faisait indépendemment. Rappelez-vous que NBP travaillait par inondation dans tous les réseaux connectés, et ne passait donc pas à l'échelle. Dès qu'on avait un ensemble de réseaux plus grand que le bureau du dentiste, par exemple à l'échelle d'un campus, il fallait le découper en zones (le mécanisme de découverte des zones distantes fonctionnait également par inondation : AppleTalk était conçu avec une vision étroite des réseaux, limités à une seule organisation, et n'avait pas tiré les leçons du développement de l'Internet). Bref, le remplaçant de NBP doit fournir également un service de partitionnement, qu'on puisse avoir plusieurs serveurs de fichiers nommés "*File server*". Il doit également permettre d'informer l'utilisateur de quelle est la zone courante et quelle est la liste des zones accessibles (section 3.11).

Et la syntaxe des noms ? La section 3.5 note que des noms qui sont tapés sur la ligne de commande doivent être courts et sans fioritures (pas d'espace). Mais des noms choisis dans une liste ont plutôt intérêt à être longs et précis. Donc, `lp-fred` convient bien à l'imprimante dans le bureau de Frédérique si on travaille en ligne de commande, mais « Imprimante couleur de Frédérique » est préférable pour choisir la bonne imprimante dans une liste de possibilités. Le remplaçant de NBP doit donc autoriser peu ou prou tous les caractères Unicode, de préférence encodés en UTF-8. La section 3.5 rappelle également que des caractères comme . ou : doivent être autorisés dans les noms.

NBP était conçu pour fonctionner sans aucune configuration et la section 3.6 demande que son successeur en fasse autant, tout en pouvant tirer profit de l'infrastructure existante, s'il en existe une.

Un réseau sans configuration est très sympa, pas d'administrateur réseaux barbu et grognon à supporter, pas de formulaires à remplir, le rêve des utilisateurs : ils sortent le Mac tout neuf de sa boîte, le branchent, le nomment « Mon Macintosh » et c'est parti, sans rien demander à personne. C'est cool, oui, mais ce n'est pas sans risque. Que se passe t-il si Jean-Michel et Frédérique appelent tous les deux leur ordinateur « Mon Macintosh » ? Ou si le fabriquant d'imprimantes Brother livre des imprimantes qui sont toutes pré-configurées avec le nom « Brother Printer » ? La section 3.7 se consacre à la gestion des noms. Le remplaçant de NBP doit donc gérer ces cas. Pour une machine qui a un utilisateur humain juste en face, comme un ordinateur, on peut imaginer un message « Tiens, une autre machine nommée « Mon Mac » vient d'apparaître, voulez-vous changer le nom de la vôtre ? » Pour une machine sans écran et clavier, comme une imprimante, le logiciel doit d'autorité affecter un nouveau nom comme « Brother Printer 2 », l'utilisateur pouvant toujours la renommer ensuite. Attention, cette vérification d'unicité doit se faire en permanence, pas juste au démarrage, car les machines se déplacent et une nouvelle machine risque toujours d'apparaître.

Avec AppleTalk, mais aussi avec des techniques IP comme DHCP ou bien les adresses locales au lien (RFC 4862), une machine n'a pas d'adresse stable. Le logiciel ne doit donc pas, une fois la résolution

faite, stocker des adresses. Il ne doit mémoriser que des noms et refaire une requête de résolution avant chaque connexion (section 3.8).

On l'a vu, NBP permet un mécanisme de joker, où le caractère = remplace n'importe quel nom. Ainsi, la requête NBP `=:LaserWriter@MaZone` signifie « N'importe quelle imprimante compatible dans `MaZone` ». Cela permet d'énumérer toutes les instances d'un service donné et de présenter une liste à l'utilisateur humain (section 3.10) et cette possibilité doit donc être gardée.

Autre demande pour l'interface utilisateur : le protocole doit fournir un moyen de présenter à l'humain une liste à jour en permanence (section 3.15). Pas question d'obliger ce dernier à cliquer sur "*Refresh*" et à attendre dix secondes l'apparition d'une nouvelle liste. Les nouvelles machines sur le réseau doivent donc pouvoir signaler leur présence tout de suite (sans que chaque machine ne doive interroger les autres en permanence, ce qui chargerait le réseau).

Cela commence à faire pas mal d'exigences mais la section 3.9 ajoute une méta-exigence difficile : que le logiciel soit suffisamment simple pour qu'il puisse être mis en œuvre dans des équipements bas de gamme (par exemple une imprimante à jet d'encre bon marché). Voir aussi la section 3.12 sur les économies d'énergie.

Autre méta-exigence, le protocole qui remplacera NBP doit être relativement stable sur le long terme (section 3.13). Pas question de le faire dépendre d'une mode comme par exemple le format de données structuré du moment (XML, JSON, etc sont sans doute visés, en des termes fort critiques pour l'industrie informatique).

Après une telle liste, on peut se demander s'il est possible de concevoir un protocole qui ait toutes ces qualités. La section 4 explore déjà un protocole existant. Question qui fâche : l'IETF n'a-t-elle pas déjà un protocole qui fait tout cela, SLP (RFC 2608) ? Non, répond le RFC : même s'il fournit bien plus de choses que ne faisait NBP en matière de découverte de services, SLP ne fait pas de détection de conflit, et n'a pas de mécanisme efficace de traduction de nom en adresse, la principale fonction de NBP.

Reste la question de la sécurité (section 6). AppleTalk et NBP n'en offraient aucune. C'est clairement un point où le nouveau protocole ne peut pas se contenter de faire « aussi bien » que NBP et où il doit faire mieux.

(Il avait existé un groupe de travail nommé Zeroconf <`http://tools.ietf.org/wg/zeroconf/`> mais qui n'était pas allé très loin, à part le RFC 3927. Le terme est donc surtout du marketing Apple.)