Réseau IP sans configuration. "Bonjour" : avancée réelle ou illusion ?

Laurent Ghys

Réseau IP sans configuration

« Bonjour » : avancée réelle ou illusion ?

Laurent Ghys
Institut de Recherche et Coordination Acoustique/Musique
Laurent.Ghys@ircam.fr

Résumé

De nombreuses propositions sont faites par divers groupes de travail pour faire fonctionner de petits réseaux IP démunis de serveurs DHCP et DNS, sans aucune configuration. La problématique est celle de l'attribution des adresses, de leur traduction en noms utilisables, de l'annonce et la découverte des services.

Parmi ces propositions, nous présentons celle d'Apple, qui souhaitant trouver un successeur au protocole Appletalk, propose sous le nom Bonjour [1] *une série de protocoles IP permettant l'utilisation d'ordinateurs et de périphériques communiquant sans la moindre configuration réseau préalable.*

Quel peut-être le regard de l'administrateur systèmes et réseaux sur ces technologies d'autoconfiguration ? Doit-il les ignorer, les rejeter ou au contraire les intégrer ?

Mots clefs

Zeronconf, adresse de lien-local, multicast DNS, mDNS, LLMNR, découverte de services, SLP, DNS-SD.

1 Introduction

Nous autres, administrateurs systèmes, sommes par définition occupés à gérer nos serveurs et nos réseaux. Mais il nous arrive de plus en plus souvent de devoir tempérer les ardeurs d'un gentil utilisateur ayant « juste un tout petit dernier problème » avec son réseau personnel domestique connecté à Internet par la toute nouvelle « borne Wifi qui fait aussi routeur et même du NAT » récemment acquise, problème qu'il aimerait bien nous soumettre ...

Car jusqu'à ces derniers temps, la mise en service d'un réseau de machines sous IP, même de taille on ne peut plus modeste, comme par exemple deux simples ordinateurs portables reliés par une câble croisé (où plutôt de nos jours par une liaison sans fil) nécessitait au minimum une configuration manuelle avec des adresses privées de type RFC 1918 avant toute utilisation.

De plus, en l'absence de serveur DNS (ou du bon vieux fichier `/etc/hosts`) et donc d'un système de nommage, on utilisait très souvent dans les applications directement ces adresses configurées manuellement.

Il s'en suit une discrimination qui dépend de plus plus de la débrouillardise des utilisateurs. On arrive parfois à des pratiques qui vont aux extrêmes, de l'abandon pur et simple de l'idée de la mise en réseau pour cause de découragement total, et de l'utilisation d'une clef USB (qui a supplanté la disquette) pour transférer des fichiers, ou au contraire à la mise en oeuvre par des amateurs très éclairés (souvent grâce aux logiciels libres) de véritables réseaux personnels n'ayant rien à envier à ceux de nos laboratoires.

Ce problème est appelé au sein de l'IETF le problème du « bureau du dentiste isolé ». Le dentiste étant riche, il peut s'acheter plusieurs ordinateurs, mais dans le cas général il est rare qu'il sache mettre en oeuvre un serveur DHCP. [2]

Il existe une autre catégorie d'utilisateurs, souvent insouciants, ceux du système Mac OS d'Apple, qui n'ont généralement pas remarqué que le constructeur à la Pomme, lors du passage à OS X, soucieux de ne pas perdre sa fidèle clientèle particulièrement attachée au confort d'utilisation d'Appletalk, l'avait remplacé progressivement à chaque version du système par du « tout IP » en y incorporant toutes les fonctionnalités nécessaires pour garder la simplicité de l'autoconfiguration et du partage de services.

2 Les réseaux sans configuration

Qu'entend t'on exactement par « réseau sans configuration » ?

Avant que les imprimantes ou autres équipements connectables en réseau ne soient administrables par des serveurs web intégrés (dont il faut quand même aller chercher l'adresse IP par défaut sur le CD-ROM d'installation) on se retrouvait parfois devant un tableau de contrôle réduit ressemblant à celui de la Figure 1.

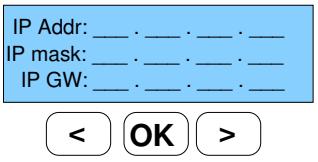

Figure 1 – *Un panneau de contrôle manuel*

[1] Apple a abandonné l'ancien nom *RendezVous* pour des raisons de conflit de dépôt de marque.
[2] Je remercie chaleureusement Stéphane Bortzmeyer qui m'a donné cette information.

A partir de ce panneau il fallait entrer l'ensemble des adresses et masques IP nécessaires à leur mise en réseau. Il s'agissait bien à n'en pas douter de *configuration manuelle*.

Si on utilise maintenant un serveur DHCP pour distribuer des adresses statiques ou dynamiques il faut bien qu'une personne ait auparavant configuré ce serveur pour le rendre opérationnel. De plus si ce serveur distribue des adresses statiques, c'est qu'une personne les aura nécessairement enregistrées (ainsi que les adresses Ethernet) dans la base de données utilisée par ce serveur. Dans ces deux cas il y a bien *configuration*.

Imaginons maintenant le cas d'un équipement compatible IP qui n'aurait aucun panneau de contrôle, connecté à un réseau sans serveur DHCP ni DNS. Notons au passage que les futurs équipements IPv6 seront dans leur très grande majorité de ce type. Le fait de pouvoir utiliser en réseau ce type d'équipement ressort du monde de l'autoconfiguration.

Ce monde existe déjà : une personne fortunée qui s'achète actuellement quelques Macintosh, une borne Airport et une imprimante avec un port Ethernet aura l'heureuse surprise de constater que « dès sa sortie du carton » tout ce matériel fonctionne en réseau *sous IP et non sous Appletalk*, sans qu'elle n'ait eu besoin de configurer quoi que ce soit !

2.1 Les ancêtres

Des réseaux sans configuration existent depuis très longtemps : le bon vieux *Appletalk* avec son fameux *Sélecteur* de Mac OS 9, l'*IPX* de Novell, ou le couple *NetBIOS/SMB* de Microsoft et son non moins célèbre *Voisinage réseau* fournissent depuis longtemps l'allocation automatique des adresses, des fonctions de nommage et de découverte de services décentralisés qui rendent possible la communication locale permettant par exemple le partage de fichiers et d'imprimantes.

2.2 Le groupe de travail IETF Zeroconf

Sous l'impulsion de Stuart Cheshire [3] qui réfléchissait au remplacement d'Appletalk et avait eu l'idée d'un NBP/IP (*AppleTalk Name Binding Protocol*) en *multicast*, un groupe de travail de l'IETF a été créé : *Zero Configuration Networking* alias *Zeroconf* [1].

Dans un article de la revue *IEEE Internet Computing*, « Autoconfiguration for IP Networking : Enabling Local Communication » [2] Erik Guttman, qui a dirigé ce groupe de travail, décrit les concepts des réseaux sans configuration. Il énumère les quatre points requis pour un réseau *Zéroconf* :
– autoconfiguration des adresses,
– résolution adresses <-> nom,
– découverte des services,
– allocation d'adresses *multicast*.
Nous ne traiterons pas ce dernier point qui par exemple a donné lieu à la proposition du protocole ZMAAP car il dépasse largement le cadre de notre sujet.

3 Adresses IPv4 de lien-local

Si la notion d'adresse IPv6 de lien-local est bien connue des fidèles lecteurs de Gisèle Cizault [3] et est parfaitement définie par le RFC 2373, pour IPv4 cette notion est bien plus nouvelle. Pourtant la parution du RFC 3927 en mars 2005 n'a fait qu'entériner l'autoconfiguration d'adresses IPv4 de lien-local qui était déjà incluse dans Mac OS depuis la version 8.5 et dans Windows depuis Windows 98. [4]

Nous allons voir que le concept d'adresse IPv4 de lien-local est cependant assez différent de celui d'IPv6.

L'autoconfiguration a été prévue dès la conception du protocole IPv6. Mais l'établissement automatique de l'adresse de lien lien-local n'est qu'une toute petite étape dans l'autoconfiguration de tout le protocole - y compris les adresses routables [5] - grâce aux différents mécanismes de découverte : découverte de voisins, découverte de routeurs, découverte des préfixes, détection d'adresses dupliquées et de divers paramètres comme le MTU.

L'adresse IPv6 de lien-local est donc prévue pour cohabiter en permanence avec les adresses routables.

En IPv4, le RFC 3927 - *dès son résumé* - indique que les adresses de lien-local et les adresses routables ne sont pas prévues pour être configurées simultanément sur une même interface.

Elles ne servent en effet que pour les petits réseaux *ad-hoc* ou lorsqu'un serveur DHCP n'est pas accessible. Lorsque qu'une adresse routable (privée ou publique) peut être obtenue par DHCP, alors l'adresse de lien-local est abandonnée.

3.1 Allocation des adresses de lien-local

En IPv6 l'adresse de lien-local est construite à partir du préfixe `FE80::/64` et de l'identifiant EUI-64 lui-même calculé pour Ethernet à partir de l'adresse MAC (voir [3]).

En IPv4, l'adresse de lien-local est sélectionnée au moyen d'un générateur pseudo-aléatoire dans la plage d'adresses de `169.254.1.0` à `169.254.254.255` incluses. [6] Pour que les machines utilisent, si possible, toujours la même adresse, il est conseillé de commencer la séquence pseudo-aléatoire, par exemple avec une valeur dérivée de l'adresse MAC, et de sauvegarder - quand c'est possible - la dernière adresse IPv4 de lien-local utilisée lors de l'arrêt de l'équipement.

Le RFC indique que les adresses de la plage `169.254/16` : **ne sont pas routables** et que **certaines précautions** doivent

[3] Qui était étudiant à Stanford à cet époque.
[4] On ne devrait donc plus utiliser différentes expressions comme « autoip » ou comme « APIPA » mais plutôt « adresses RFC 3927 ».
[5] Pour le terme anglais *global* nous avons préféré le terme français *routable* à celui de *mondiale* à surtout à celui du faux ami *globale*.
[6] La plage d'adresses 169.254.0.0/16 avait été resérvée dans le RFC 3330.

être respectées :
– elles ne doivent **pas** être enregistrées dans un **serveur DHCP**, **ni configurées à la main**
– elles ne doivent **pas** être résolues par le **DNS**, que ce soit en direct ou en *reverse*
– les clients ne doivent pas faire de requêtes DNS classique pour des noms appartenant au domaine : `254.169.in-addr.arpa.`
– elles ne doivent **jamais** être envoyées à un routeur pour être **relayées**.

3.2 Vérification de l'unicité de l'adresse

Tout comme pour IPv6, l'unicité d'une adresse IPv4 sur lien-local doit-être vérifiée avant son utilisation effective.

Le mécanisme est assez différent de celui d'IPv6 où ce sont des messages ICMPv6 de *sollicitation d'un voisin* qui sont envoyés à l'adresse *multicast sollicité*. Rappelons aussi qu'en IPv6, une seule valeur d'adresse de lien-local est prévue par interface : s'il y a conflit lors de l'autoconfiguration, ce qui est très peu probable compte-tenu de l'emploi de l'adresse EUI-64 sur un lien local, alors une intervention humaine est nécessaire.

En IPv4, la vérification d'unicité d'adresse de lien-local fonctionne à partir de trois requêtes *ARP probes* qui sont diffusées sur le lien local, c'est-à-dire des requêtes ARP avec l'adresse source IP positionnée à zéro, afin de ne pas polluer les caches avec des informations qui ne sont pas encore valides. Si l'adresse IP est déjà utilisée (ou demandée) par une autre machine, alors on tente une autre valeur d'adresse fournie par le générateur pseudo-aléatoire.

Lorsque la machine a trouvé une adresse libre, c'est-à-dire si elle n'a pas vu passer de réponse pendant les deux secondes suivant sa dernière requête ARP, alors elle diffuse en *broadcast* deux *annonces ARP*, c'est-à-dire la même chose que les *ARP probes* mais avec cette fois l'adresse source IP contenant l'adresse sélectionnée.

3.3 Détection d'adresse dupliquée

Le mécanisme de détection d'adresse dupliquée (DAD) n'est pas uniquement utilisé par les machines pendant la phase initiale d'allocation d'adresse de lien-local [7]

Pendant toute la période où une machine utilise une adresse de lien-local sur une interface, elle doit surveiller tous les paquets ARP qu'elle voit passer, qu'ils contiennent des requêtes **ou** des réponses, car elles ont la particularité, quand elles concernent des adresses de lien-local, d'être toujours envoyées en *broadcast*.

Si cette machine y trouve une requête concernant une adresse IP qu'elle utilise provenant d'une autre machine alors il y a conflit. Elle ne peut l'ignorer et a alors le choix entre deux solutions :

– elle peut choisir de changer immédiatement d'adresse de lien-local,
– si elle souhaite garder l'adresse, par exemple parce qu'elle a des connexions TCP en cours, si elle n'a pas vu passer de conflits ARP depuis plus de 10 secondes, elle peut alors défendre son adresse en envoyant simplement une annonce ARP en *broadcast*. Si au contraire ce n'est pas la première annonce de conflit qu'elle voit passer depuis les dix dernières secondes, alors elle doit immédiatement cesser d'utiliser cette adresse.

Il est à noter que bien qu'étant en principe très peu probables, ces conflits peuvent se produire, par exemple lors de la fusion de deux réseaux. Il n'est donc pas possible d'empêcher totalement la coupure de connexions TCP.

L'utilisation de ce mécanisme d'allocation d'adresses apporte de nouveaux types d'attaques comme par exemple le vol d'adresse. Le RFC laisse le soin aux implémenteurs de faire le nécessaire. [8]

3.4 DHCP et autoconfiguration

On trouve en annexe du RFC 3927 les descriptions des implémentations de ce protocole pour Mac OS et Windows, où sont indiquées les interactions entre DHCP et l'autoconfiguration. Dans toutes ces versions, des requêtes DHCP sont faites en premier et se poursuivent pendant l'autoconfiguration.

Si à un quelconque moment, on obtient une adresse par DHCP alors on utilise cette adresse et on abandonne le processus d'autoconfiguration sur le lien local.

Si on utilise déjà une adresse de lien-local, des requêtes DHCP sont effectuées périodiquement toutes les 5 minutes ; si elles sont fructueuses, on passe en adresse routable et on cesse d'utiliser l'adresse de lien-local.

4 DNS multicast sur le lien local

Une fois que nous avons des adresses IP opérationnelles, l'étape suivante est de travailler avec des noms. Car on oublie souvent que sans DNS, vraiment plus rien ne fonctionne ! La plupart des outils comme la messagerie électronique ou les navigateurs sont complètement bloqués par l'absence de résolution de noms. Nous n'avons pas droit au serveur DNS ; c'est le *multicast* qui va être notre sauveur.

Les choses se compliquent un peu, car il y a deux propositions en cours qui s'affrontent :
– LLMNR (*Linklocal Multicast Name Resolution*), soutenu par Microsoft (Bernard Aboba)
– mDNS (*Multicast DNS*), soutenu par Apple (Stuart Cheshire).

Ces deux propositions de protocoles ne sont pas du tout dans le même état d'avancement, bien que du point de vue IETF, ils en sont tous les deux au stade de *Draft*, à ceci près que

[7] Il ne s'agit que du lien-local et non pas du cas général IPv4, pour lequel Stuart Cheshire fait une autre proposition de DAD dans un *Draft*.
[8] Rappelons que les deux premières implémentations étaient Mac OS 8 et Windows 98 ...

celui de LLMNR est proposé par le groupe de travail *dnsext* alors que mDNS est une proposition individuelle.

Nous verrons par la suite qu'il existe plusieurs implémentations de mDNS dont *Bonjour* qui est disponible pour OS X, Windows et pour les systèmes POSIX. La plupart des modèles d'imprimantes Ethernet récentes supportent mDNS.

Nous ne connaissons pas d'implémentation de LLMNR.

4.1 Différences entre LLMNR et mDNS

Nous supposerons que le lecteur est familier avec le DNS (RFC 1034, 1035).

Dans l'article de Wikipedia en anglais consacré à zeroconf, il semble qu'une erreur (volontaire ou pas) soit commise par les auteurs quand ils affirment que « Les différences entre mDNS et LLMNR sont mineures, et concernent par exemple la manière dont les TTLs (*time to live*) des paquets IP sont envoyés.'»

Si en effet on se contente de réduire leurs différences uniquement au tableau comparatif ci-dessus, c'est certainement que l'on n'a pas étudié ces deux protocoles !

	mDNS	**LLMNR**
IPv4 multicast	224.0.0.251	224.0.0.252
IPv6 multicast	ff02::fb	ff02::1:3
port	5353	5355
TTL - Hop Limit	255	1

Car en dehors du choix de TTL (au sens nombre de sauts), ce tableau ne fait que montrer la *conséquence* du fait que ces deux protocoles étant incompatibles entre-eux, on leur a alloué des adresses et des ports différents. Rien de plus.

Le hasard a voulu que pendant la rédaction du présent papier (août 2005), l'IESG a reçu la demande *Last Call* du groupe de travail *dnsext* pour faire entrer LLMNR [4] dans la chemin des normes. Ceci a provoqué une vive discussion lancée par Stuart Cheshire. Il faut dire que LLMNR a quand même le triste privilège d'atteindre au moment de cette discussion les 43 versions de *Draft* !

Si on a la patience de consulter les deux *Drafts*, on s'aperçoit assez rapidement que ces deux protocoles, qui paraissent semblables sont en réalité assez différents car ils ont été créés avec des objectifs distincts :

- mDNS a été clairement conçu pour remplacer Appletalk (voir [5]) et par conséquent pour permettre efficacement la découverte de services sur le lien local,
- LLMNR est conçu pour pallier uniquement l'absence momentanée de serveur DNS sur un réseau. [9]

Avant de quitter LLMNR, parce qu'il n'y a pas encore d'implémentation (encore moins de version libre dont nous pourrions examiner les sources), et parce que Bonjour utilise mDNS, nous indiquerons cependant très rapidement les principales caractéristiques de LLMNR qui montrent que malgré quelques différences dues à l'emploi du *multicast* pour les requêtes, il reste très proche du DNS :

- le protocole permet de résoudre des noms dans toutes les zones du DNS,
- les requêtes sont envoyées en *multicast* et les réponses sont toujours en *unicast*,
- une seule requête est envoyée à la fois,
- le champ RCODE est utilisé dans les réponses qui peuvent donc être vides ou négatives,
- l'identificateur de requête (champ ID) est recopié dans les réponses,
- c'est dans les requêtes qu'on trouve un indicateur de conflit de noms,
- si la réponse est trop longue pour tenir dans un seul paquet UDP, l'indicateur TC permet de demander à l'envoyeur de recommencer sa requête en TCP,
- il n'y a pas de requêtes en *unicast* UDP,
- tous les TTL des RR sont positionnés à la même valeur qui est de 30 secondes par défaut.

Le fameux point concernant la différence des TTL de 1 pour LLMNR et 255 pour mDNS est certes crucial sur le plan de la sécurité, mais c'est un paramètre isolé dans chaque proposition dont la modification ne remettrait pas en cause le reste des spécifications. Pour mDNS la valeur 255 n'est d'ailleurs là que pour assurer la compatibilité avec les anciens clients.

4.2 mDNS

Rappelons que mDNS [6] est une des briques permettant de construire le remplaçant en IP du protocole Appletalk.

De nombreuses et riches idées sont proposées dans ce protocole, certaines étant assez exotiques comme par exemple ce que l'auteur appelle *Sleep Proxy Service*, un *proxy* qui conserve la liste des services qu'offre une machine pendant qu'elle est en sommeil (pour cause d'économie d'énergie par exemple) et sait la réveiller lorsque qu'une autre machine souhaite accéder à l'un de ces services.

L'auteur du *Draft* indique que tout ce qui pouvait être utilisé dans le DNS classique a été conservé. Les différences sont rendues nécessaires à la fois à cause de l'utilisation du *multicast*, mais aussi par le fait de vouloir créer des communautés de machines qui coopèrent plutôt que de vouloir définir une hiérarchie de délégations.

Il fait le résumé des différences de mDNS par rapport au DNS :

- utilise le *multicast* pour les requêtes **et** les réponses,
- utilise le port 5353 à la place du port 53,
- concerne une partie précise de l'espace de noms du DNS, uniquement le domaine racine `.local.`,
- utilise seulement UTF-8 pour coder les noms dans des RR,
- définit une limite précise pour la longueur des noms des domaines (255),
- fixe la taille maximum des paquets UDP au MTU du lien-local,

[9] et certainement aussi pour contrer mDNS, car on peut se poser la question de savoir pourquoi il est n'est pas encore implémenté.

– permet plus d'une question dans un paquet de requête,
– utilise la section Réponses dans une requête pour indiquer les réponses déjà connues,
– utilise le bit TC dans une requête pour indiquer qu'on va y joindre des réponses déjà connues,
– ignore le champ ID des requêtes,
– ne nécessite pas la copie des requêtes dans les réponses,
– utilise des réponses gratuites pour les annonces de nouveaux enregistrements aux autres machines,
– définit un bit de demande de réponse *unicast* dans les *rrclass* des requêtes,
– définit un bit de réinitialisation du cache dans les *rrclass* des réponses,
– utilise un TTL DNS de 0 pour indiquer qu'un enregistrement a été supprimé,
– surveille les requêtes pour la Suppression des Requêtes Dupliquées,
– surveille les réponses pour la Suppression des Réponses Dupliquées,
– ... détecte les conflits à la volée ...
– ... et intègre un système de cohérence de caches.

On voit que cette liste simplifiée est déjà très longue. Certains points nécessitent de longues explications, comme par exemple le fait que mDNS, bien qu'il permet techniquement de résoudre tout nom du DNS, recommande fortement pour des raisons évidentes de sécurité de ne servir que le suffixe `.local.` qui n'a de signification que sur le lien local. [10]

C'est d'ailleurs un des points qui anime copieusement la discussion de l'IETF actuelle citée plus haut. Il se trouve qu'une grande proportion des requêtes pour de mauvais TLD qui parviennent aux serveurs racines du DNS concernent le domaine `.local.`. La polémique porte sur l'origine de ces requêtes. Entre mDNS qui théoriquement « confine » ce domaine dans le lien local et LLMNR qui pour tout nom de domaine fait toujours une tentative de résolution DNS avant de faire sa résolution locale, les accusations pleuvent.

Un autre point qui montre que mDNS diffère du DNS classique est qu'il supporte trois modes de requêtes :
– les requêtes simples, (celles du DNS traditionnel), une requête attend une réponse,
– les requêtes simples avec accumulation des réponses multiples venant de plusieurs serveurs mDNS répartis,
– les requêtes continues utilisées par les logiciels qui intègrent un explorateur ou un sélecteur réseau (*IP browser*).

L'étude approfondie du *Draft* de mDNS montre que c'est un système très sophistiqué. Il profite pleinement de la richesse du *multicast* quand il est utilisé à la fois pour les requêtes et les réponses pour créer un système de nommage local, avec un système de cohérence de caches très bien adapté pour des petits réseaux de machines coopérant d'égal à égal.

Ce sont les TTL des RR (qui ont des valeurs très courtes) ainsi que les informations contenues dans les requêtes (contenant les réponses connues) et les réponses multi-diffusées qui sont astucieusement utilisés pour assurer que toutes les machines disposent du plus grand nombre d'informations à jour, tout en minimisant le trafic réseau.

Si un jour à son tour il est candidat au chemin des normes, il est certain qu'il suscitera d'énormes débats sur la sécurité, par exemple pour son TTL IP de 255. Mais il soulève d'autres problèmes. Comme le signale C. Huitema, il casse la séparation entre espace de nommage DNS et topologie réseau, ce que certains refusent catégoriquement et d'autres au contraire considèrent comme une nécessité pour les réseaux isolés.

5 Découverte des services

Les choses s'aggravent encore quand on aborde la découverte des services IP, car il y a une multitude de propositions. Certaines de ces propositions proviennent de groupes très puissants comme le Forum UPnP [7], certaines comme SLP sont bien des normes IETF, mais n'ont pas eu le succès escompté, d'autres comme DNS-SD ne sont pas encore des normes IETF.

5.1 SLP, SSDP, DNS-SD

Si on s'en tient à celles qui ont donné lieu à des documents de l'IETF, il y a trois concurrents :

– SLP (*Service Location Protocol*),
– SSDP (*Simple Service Discovery Protocol*), proposé par le Forum UPnP, utilisé dans Windows,
– DNS-SD (*DNS-Based Service Discovery*), proposé par Apple et utilisé dans Bonjour.

Du point de vue technique, un point sépare nettement les deux protocoles SLP et SSDP par rapport à DNS-SD : ils utilisent tous les deux des URL pour transporter les annonces de service, alors que DNS-SD n'est que du « pur DNS » qui utilise des enregistrements SRV et TXT.

La norme de découverte de services « la plus ancienne et la plus officielle de l'IETF » est sans aucun doute SLP v2 (RFC 2608, du groupe de travail *srvloc*) qui est connu entre autres par son implémentation libre OpenSLP et a donné lieu à de nombreux RFC complémentaires. Mais il semble qu'il soit un peu délaissé par les grands acteurs de l'informatique dont Microsoft.

Il faut dire que SLP est un protocole complexe, totalement indépendant du DNS. Il utilise directement du *multicast* avec sa propre adresse 224.0.1.22 et le port 427. Il intègre trois sortes d'Agents : *User Agent* pour les requêtes, *Service Agent* pour les annonces de services, et *Directory Agent* pour les enregistrements de services (pour les plus grands réseaux). Il emprunte à LDAP son langage de requêtes, comprend 11 types de messages dont certains sont obligatoires et d'autres optionnels. La lecture des nombreux RFC concernant SLP nous interdit de le qualifier de « *Simple Protocol* » ou de « *Lightweight Protocol* ».

[10] Dans le *Draft* il est indiqué que cela concerne donc bien un protocole de l'IETF et non une règle d'usage de l'ICANN.

Apple avait intégré un agent SLP dans Mac OS X version 10.1, bien qu'il soit toujours présent, la politique actuelle du constructeur est clairement de promouvoir Bonjour.

Le protocole SSDP en est resté au stade de *Draft* et n'est jamais rentré dans le chemin des normes. Tout comme SLP, SSDP est un protocole indépendant. Il utilise aussi une adresse *multicast* particulière, qui est fonction de la portée (*scope*) des requêtes. Enfin nous signalerons que les USN (*Unique Service Names*) de SSDP ne sont pas sous la gouverne de l'ICANN mais sous la responsabilité du Forum UPnP.

5.2 DNS Service Discovery

Au contraire de SSDP ou de SLP, DNS-SD (*DNS Service Discovery*) [8] est un protocole poids plume ! Comme son nom l'indique, c'est un protocole de découverte de services entièrement basé sur le DNS, ce qui fait sa simplicité. Il peut s'appuyer sur la version classique du DNS ou sur mDNS sans aucune obligation d'utiliser ce dernier.

Il n'utilise donc pas de port ni d'adresse IP en propre. Ce n'est pas un nouveau protocole circulant sur le réseau, aucune attribution IANA n'est demandée pour ce protocole.

5.3 Rappel sur les DNS SRV (RFC 2782)

DNS-SD emprunte les concepts du RFC 2782 (DNS SRV) en les étendant pour pouvoir permettre la découverte d'instances de services, aussi allons nous rappeler rapidement les enregistrements de type SRV car leur usage n'est pas encore très répandu dans nos DNS. [11]

Le RFC 2782 [12] a créé le nouveau type d'enregistrement SRV pour la recherche de serveur(s) pour un service.

L'idée sous-jacente est que les clients demandent au DNS un service (ou un protocole) et qu'il leur retourne la liste des machines du domaine qui le proposent. Par exemple pour le service http, il va renvoyer les noms d'un ou de plusieurs serveurs assurant indifféremment ce service.

Voici le format de l'enregistrement SRV tel qu'il figurera dans un fichier de zone :

_Serv . _Proto SRV *Prio Poids Port Cible*

Serv	le nom du service au sens Assigned Numbers RFC 1700
Proto	le nom du protocole actuellement soit udp ou tcp
Prio	la priorité (comme pour les MX) priorité à la plus basse valeur (0-65365)
Poids	le poids relatif, pour une même priorité (0-65365)
Port	le port du service sur la machine cible
Cible	le nom de domaine de la machine cible.

Le poids relatif est un nouveau champ dans les enregistrements DNS qui permet de donner une probabilité d'accès entre les différents serveurs. Si par exemple deux machines de même priorité ont des poids de 1 et 3, alors la répartition sera de 25% et 75% respectivement.

Le nom de la machine Cible ne doit pas être un alias DNS (CNAME). La machine doit avoir au moins un enregistrement d'adresse. Si la cible vaut '.' alors cela signifie que par décision ce service n'est pas supporté dans ce domaine.

6 Le protocole DNS-SD

Du point de vue technique, l'idée principale de DNS-SD est d'ajouter un niveau d'indirection. Au lieu de faire une requête sur les enregistrements de type SRV sur le couple <*service*>.<*domaine*>, le client fait une requête sur les enregistrements de type PTR.

Le résultat de cette requête est une liste de zéro ou plusieurs enregistrements PTR qui contiennent des Noms d'Instance de Service sous la forme :

<*Instance*> . <*Service*> . <*Domaine*>

- La partie <*Instance*> est un simple label DNS qui contient du texte encodé en UTF-8 suivant le RFC 3629. Il n'y a aucune restriction sur les caractères utilisés. La longueur des Instances est limitée à 63 octets.
- La partie <*Service*> est identique à celle des enregistrements SRV du RFC 2782.
- La partie <*Domaine*> est un nom de domaine du DNS.

Les noms d'Instances sont codés en UTF-8 car ils ne sont pas faits pour être manipulés à la main. Ils sont en général présentés à l'utilisateur dans des menus, dans lesquels il fait des sélections.

Il n'y a pas de représentation interne des données, à un seul détail près : comme le séparateur de champs est un point et que celui-ci est autorisé dans les noms d'instances, il est remplacé par '\.' (et le '\' par '\\') avant l'appel du DNS.

On remarque qu'un choix fondamental a été fait en choisissant l'ordre :

<*Instance*> . <*Service*> . <*Domaine*>

et non pas :

<*Service*> . <*Instance*> . <*Domaine*>

L'auteur donne trois raisons pour ce choix :
- l'utilisateur connaît le service qu'il cherche, il en cherche les instances, pas le contraire,
- la compression des paquets de réponses multiples est plus efficace puisque la partie <Service>.<Domaine> est commune,
- cela rend possible la délégation de sous-domaines de services. On peut par exemple confier le sous domaine « _tcp » ou « _ipp._tcp » à un autre serveur DNS.

[11] par contre Active Directory les utilise copieusement dans ses mises à jours dynamiques ...
[12] qui a remplacé le RFC 2052, Expérimental

Une fois que le client a un Nom d'Instance de Service et qu'il veut accéder à ce service, il fait une nouvelle requête DNS en demandant cette fois l'enregistrement SRV de ce nom.

Le résultat de cette requête est un (ou plusieurs) enregistrement(s) SRV contenant le port et la machine cible où l'on peut trouver ce service. Dans le cas exceptionnel où il y a plusieurs enregistrements SRV retournés, le client doit interpréter les priorités et les poids. En général il y a un seul enregistrement SRV dont la priorité et le poids sont nuls.

6.1 Exemples pratiques

Afin qu'il perçoive pleinement ce que permet réellement DNS-SD, nous encourageons fortement le lecteur à tester lui-même les exemples de la section 16 du *Draft*. Il pourra vérifier par la même occasion que DNS-SD ne dépend pas de mDNS et fonctionne parfaitement sur le DNS actuel.

Il lui suffit pour cela de se trouver sur une machine connectée à Internet et de disposer de la commande *nslookup*. Si votre machine est sous Unix c'est certainement le cas.

Comme pour notre part nous préférons fortement la commande *dig*, voici ces quelques exemples réalisés avec *dig* :

Par exemple, la commande :

```
dig _ftp._tcp.dns-sd.org. PTR
```

retournera la liste des services FTP qui sont annoncés sur *dns-sd.org*. Dans cette liste figure entre autres la réponse suivante [13] :

```
_ftp._tcp.dns-sd.org. 60 IN PTR \
Apple\032QuickTime\032Files.\
_ftp._tcp.dns-sd.org.
```

On remarquera au passage la présentation faite par la commande *dig* qui a remplacé les espaces par '\032'.

Et si l'on veut connaître comment on fait pour accéder à ces « Fichiers QuickTime » il suffira cette fois de faire la requête suivante :

```
dig 'Apple QuickTime Files.\
_ftp._tcp.dns-sd.org.' ANY
```

Dont la réponse, réellement obtenue sur Internet lors de la rédaction du présent document, est limpide :

```
Apple\032QuickTime\032Files.\
_ftp._tcp.dns-sd.org. \
60 IN SRV 0 0 21 ftp.apple.com.

Apple\032QuickTime\032Files.\
_ftp._tcp.dns-sd.org. \
60 IN TXT "path=/quicktime"
```

c'est-à-dire sur le serveur *ftp.apple.com* sur le port 21 dans le dossier */quicktime*.

6.2 Format des enregistrements TXT

Le protocole spécifie aussi des règles de format pour les enregistrements de type TXT qui vont permettre de passer des paramètres supplémentaires.

Les enregistrements TXT utilisés par DNS-SD respectent évidemment le RFC 1035. Ils servent à transporter des paires nom/valeur qui sont encodées sous la forme « nom=valeur » puis concaténées. S'il n'y a pas de signe '=' les valeurs sont booléennes.

Nous n'entrerons pas dans les détails du protocole et nous contenterons de donner des exemples extraits du *Draft* :

```
Installed PlugIns=JPEG,MPEG2,MPEG4
Papersize=A4
Anon Allowed
```

Nous ajouterons que si les noms de clés ont des restrictions, les valeurs sont des données binaires opaques et qu'elles peuvent contenir par exemple de l'ASCII de l'UTF-8 ou des valeurs numériques directement en binaire. (par exemple une adresse IP occupera 4 octets et non pas une chaîne ASCII de 7 à 15 octets)

6.3 Types de Services DNS-SD

Les noms de Services au sens de DNS-SD peuvent porter à confusion. Bien que certains soient communs, ce ne sont pas les services du fichier */etc/services* autrement dit les Assigned Numbers du RFC 1700.

Un bon exemple qui montre cette confusion est le protocole ftp, bien connu sous Unix pour l'identité entre le nom du protocole et le nom de la première application qui permettait d'accéder à ce service.

Ces noms de protocoles d'applications, notion qui nous semble encore assez floue, ou encore les Types de Service DNS-SD, ont une longueur maximum de 14 caractères et obéissent aux mêmes règles de composition que les noms d'hôtes du DNS ([a-z][0-9][-] etc.).

Le *Draft* propose que ce soit IANA qui prenne en charge l'allocation de ces noms. En attendant la publication du RFC, un système d'enregistrement provisoire pour ces noms a été mis en place sur le site de DNS-SD [9].

6.4 Informations des Services et Domaines

En général les applications n'ont aucune raison de vouloir connaître tous les services qui sont disponibles dans un domaine.

Cependant pour l'utilisation d'outils de gestion réseau, pour la résolution de problèmes, il peut être utile aux administrateurs de trouver la liste des types de services qui sont annoncés sur le réseau. Une méta-requête spéciale a été prévue à cet effet. Une recherche de PTR sur le nom :

_services._dns-sd._udp.<domaine>

[13] pour des raisons de présentation nous avons replié les lignes trop longues, avec le symbole '\' en fin de ligne

retournera une liste de PTR contenant chacun un type de service.

Voici de nouveau un exemple réalisé dans des conditions réelles sous Unix avec la commande *dig* :

```
dig _services._dns-sd._udp.bolo.net PTR
```

La réponse contient plusieurs lignes commençant par :

```
_services._dns-sd._udp.bolo.net. 60 IN
```

et se terminant respectivement par :

```
PTR  _printer._tcp.bolo.net.
PTR  _afpovertcp._tcp.bolo.net.
PTR  _pdl-datastream._tcp.bolo.net.
PTR  _ftp._tcp.bolo.net.
PTR  _ipp._tcp.bolo.net.
PTR  _ssh._tcp.bolo.net.
PTR  _http._tcp.bolo.net.
```

Il suffit ensuite de faire de nouveau des requêtes de type PTR sur ces services comme dans les exemples du paragraphe 6.1 pour obtenir les instances de ces services.

D'autre part, lorsqu'une machine nomade (un ordinateur portable par exemple) se raccorde à un nouveau réseau, elle peut découvrir quels services sont disponibles sur ce réseau sans configuration manuelle. Cette logique peut être appliquée aussi pour la découverte des domaines dans lesquels les services sont enregistrés.

Cinq autres méta-requêtes ont donc également été prévues pour la découverte de tels domaines, elles sont de la forme :

<x>.*_dns-sd._udp.*<*domaine*> avec **<x>** valant :

- **d** liste des domaines recommandés pour la découverte des services,
- **db** le domaine recommandé pour le découverte des services,
- **r** liste des domaines recommandés pour l'enregistrement des services par des mises à jour dynamiques,
- **dr** le domaine recommandé pour l'enregistrement des services,
- **lb** le domaine passé en héritage par le système d'exploitation du client aux applications qui font de la découverte sans préciser le domaine de recherche.

Il faut bien noter que le client ne reçoit ces informations qu'à titre indicatif. Le client ou son utilisateur sont parfaitement libres d'utiliser les domaines qu'ils souhaitent. Il ne s'agit que de proposer au sein des applications des listes de choix de domaines.

7 Bonjour

Bonjour n'est qu'une des implémentations de mDNS et de DNS-SD. Il en existe bien d'autres, comme le montre la liste suivante qui est très loin d'être exhaustive :

- Avahi [10]
- JmDNS [11] en Java,
- Howl [12]
- Liaison [13]
- mdnsd [14] poids plume
- pyZeroConf [15] en Python,

Nous n'avons pas étudié ces implémentations et n'avons aucune préférence ni commentaire sur celles-ci. Certaines intègrent les deux protocoles mDNS et DNS-SD, d'autres seulement mDNS. Certaines sont actives du point de vue du développement, d'autres sont au contraire très calmes.

Dans Bonjour l'implémentation du RFC 3927 est réalisée dans le paquet *bootp* qui est l'un des éléments de Darwin, la partie de Mac OS X disponible sous licence APSL (*Apple Public Source License*). [14]

7.1 mDNSResponder

L'implémentation des protocoles mDNS et DNS-SD de Bonjour est réalisée dans le paquet *mDNSResponder* de Darwin distribué aussi sous licence APSL.

Ce paquet contient les sources pour Mac OS 9, Mac OS X, Microsoft Windows, VxWorks et les systèmes compatibles POSIX comme Linux, Solaris, FreeBSD, etc.

Dans le fichier `README` de ce paquet, on trouve un bref principe de cette implémentation de mDNS. Un programme mDNS normal contient trois composants :

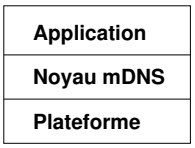

Figure 2 – *Les couches de mDNS*

La couche « Noyau mDNS » est identique pour toutes les applications et les systèmes d'exploitation.

La couche « Plateforme » fournit les fonctions qui sont spécifiques à chaque plate-forme pour l'envoi des paquets UDP, du *multicast* etc.

La couche « Application » fait tout ce qu'une application souhaite faire. Elle appelle des procédures fournies par la couche « Noyau mDNS » pour toutes les fonctions dont elle a besoin :

- annonce de service,
- recherche des instances d'un type de service,
- résolution de l'adresse IP et des ports d'une instance de service, lecture des paramètres (TXT) etc.

Sous Mac OS X c'est le démon *mDNSResponder* qui centralise toutes ces requêtes mDNS et DNS-SD ce qui nous donne le schéma suivant :

[14] Cette licence, APSL 2.0, est considérée par certains (FSF) comme libre et par d'autres (Debian) comme non libre.

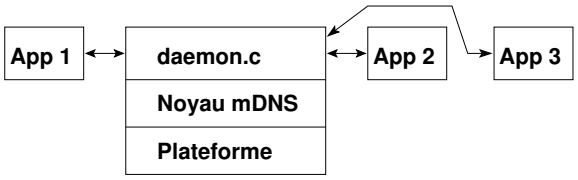

Figure 3 – *Le schéma mDNSResponder*

Ce démon, qui centralise entièrement les communications entre les applications et le réseau, est donc le seul gestionnaire du cache mDNS. Sous Unix, si on lui envoie le signal HINFO, alors il sauve une image complète de son cache dans syslog (dans le fichier */var/log/system.log* sur OS X).

8 Wide Area Bonjour

Lors de la sortie de la version 10.4 de Mac OS X (Tiger), un nouveau pas a été franchi sous le nom de *Wide Area Bonjour*. Nous avons vu que DNS-SD était prévu pour utiliser indifféremment le DNS ou mDNS. La priorité dans Bonjour avait d'abord été mise sur le lien local et donc sur mDNS. *Wide Area Bonjour* étend la découverte de services à des zones plus grandes, par exemple au niveau d'un site, en utilisant le DNS et des mises à jour dynamiques.

Nous sortons du cadre du présent exposé, car nous ne sommes plus dans le domaine de l'autoconfiguration.

L'auteur de Wide Area Bonjour fait un certain nombre de nouvelles propositions autour du DNS sous forme de *Drafts* pour rendre plus efficace la découverte de services via le DNS :
– DNS-LLQ (*DNS Long-Lived Queries*)
– DNS-UL (*Dynamic DNS Update Leases*)
– NAT-PMP (*NAT Port Mapping Protocol*)
Les personnes intéressées pourront se reporter à la page du site Développeur Apple sur Bonjour [16] dans la rubrique spécifications où figurent tous ces protocoles.

9 Conclusion

Nous nous sommes bien gardés au cours de cet article de faire de la publicité pour les *merveilleuses* choses qu'il est possible faire grâce à Bonjour. Apple le fait très bien, ce n'est pas notre propos.

Quant à la question de savoir quelle est l'attitude à avoir vis-à-vis de l'autoconfiguration et de la découverte de services, la réponse dépend fortement du contexte dans lequel nous nous trouvons.

S'il s'agit de nous séparer d'Appletalk sur nos réseaux, Bonjour peut être une des solutions, à la condition préalable d'en avoir bien saisi les mécanismes et évalué toutes les conséquences sur le plan de la sécurité.

Si on accepte sur notre site qu'une personne même de passage ait le droit de connecter un Macintosh, alors il faut bien avoir à l'esprit que les protocoles de Bonjour vont donc circuler sur notre réseau, et qu'il est toujours prudent de les connaître un minimum.

A l'Ircam, nous utilisons lors des tournées et des concerts des mini-réseaux composés de quelques machines isolées du monde qui communiquent en UDP. Jusqu'à ce jour leur configuration IP était entièrement manuelle.

Nous étudions la possibilité d'intégrer Bonjour dans nos applications musicales afin qu'elles puissent détecter automatiquement les adresses IP et les ports utilisés par les applications-soeurs tournant sur les autres machines, en découvrant également les *objets* qu'elles exportent.

Références

[1] Zero Configuration Networking. `http://www.zeroconf.org`.

[2] Erik Guttman. Autoconfiguration for IP networking : Enabling local communication. *IEEE Internet Computing*, 5 :81–86, juin 2001.

[3] Gisèle Cizault. *IPv6, Théorie et pratique, 3e édition*. O'Reilly, Paris, 2002.

[4] L. Esibov Bernard Aboba, D. Thaler. Linklocal Multicast Name Resolution (LLMNR). *Internet Draft (en cours)*, draft-ietf-dnsext-mdns-43.txt, 29 août 2005.

[5] M. Krochmal Stuart Cheshire. Requirements for a Protocol to Replace Appletalk NBP. *Internet Draft (Informational)*, draft-cheshire-dnsext-nbp-04.txt, Juin 2005.

[6] Multicast DNS. `http://www.multicastdns.org`.

[7] Forum UPnP. `http://www.upnp.org`.

[8] DNS Service Discovery. `http://www.dns-sd.org`.

[9] DNS SRV Service Types. `http://www.dns-sd.org/ServiceTypes.html`.

[10] Avahi. `http://www.freedesktop.org/Software/Avahi`.

[11] JmDNS. `http://jmdns.sourceforge.net`.

[12] Howl. `http://www.porchdogsoft.com/products/howl/`.

[13] Liaison. `http://www.acm.uiuc.edu/signet/liaison/`.

[14] mdnsd. `http://www.dotlocal.org/mdnsd/`.

[15] pyzeroconf. `http://sourceforge.net/projects/pyzeroconf`.

[16] Bonjour Apple Developer Connection. `http://developer.apple.com/networking/bonjour`.