Solution transitoire de mise à jour dynamique des adresses IPv4 et IPv6 dans le DNS

Nicolas CUISSARD
Université Paris 1 Panthéon–Sorbonne
Direction du Système d'Information
90, rue de Tolbiac
75 013 Paris

Résumé

La mise à jour dynamique du DNS par DHCP et la « double pile » IPv4/IPv6 sont deux mécanismes largement déployés mais leur utilisation combinée reste pour le moment problématique.

L'Université Paris 1 a adopté IPv6 dès 2005 avec une gestion manuelle du DNS et s'est posée la question de la mise à jour des adresses IPv4 et IPv6 dans le DNS lors de la mise en place de réseaux en DHCP dynamique. Cette présentation exposera les limites actuelles d'un fonctionnement simultané de deux serveurs (DHCPv4 et DHCPv6) causées par des identifiants non-compatibles au niveau des clients et les problèmes engendrés par un fonctionnement sans détection des conflits de mises à jour (parfois présenté comme une solution).

Pour répondre au besoin, l'architecture finalement retenue a été d'utiliser uniquement le protocole DHCPv4 et d'automatiser par script la mise à jour d'adresses EUI-64. Cette méthode est une amélioration du script « ddns-ipv6 » utilisé à l'Université de Nouvelle-Galles du Sud, elle présente l'avantage de pouvoir fonctionner avec tout type de matériel (ordinateurs, imprimantes, copieurs, etc.) et nécessite peu de configuration au niveau des hôtes et de l'infrastructure.

Après une introduction sur la gestion des adresses IPv6 dans le DNS à l'Université Paris 1 et un rappel théorique sur la détection des conflits de mises à jour, cette présentation examinera plusieurs architectures sous forme d'exemples. Les différentes pistes pour contourner le problème d'identification des clients seront évoquées et un retour d'expérience sur la migration des hôtes et la configuration du serveur DHCP sera fait.

Mots clefs

IPv6, DDNS, DHCPv6, EUI-64, double-pile, DUID, conflits de mises à jour, isc-dhcp, RFC 4361.

1 Introduction

1.1 La gestion manuelle du DNS avec « `majdns` »

L'Université Paris 1 est présente sur 30 sites géographiques, son parc informatique est composé d'environ 4 000 postes clients. Suite à la mise en place d'IPv6, elle a développé l'outil « `majdns` » afin de simplifier la gestion de son DNS [1]. Le fonctionnement était manuel et pour les postes de travail, les adresses EUI-64 [2] étaient référencées. Tout renommage, changement de réseau ou changement d'adresse MAC des clients nécessitait une mise à jour manuelle de l'adresse IPv6 dans le DNS. Le DHCP était utilisé mais en mode « statique » où chaque adresse MAC était manuellement associée à une adresse IPv4.

1.2 Nouvelle architecture avec DNS dynamique

En 2010 a débuté un projet de modernisation de la gestion de parc et des réseaux LAN comprenant la mise en place de réseaux privés utilisant la traduction d'adresses, un domaine Active Directory et la virtualisation des postes de travail pour les salles d'enseignement. La mise à jour dynamique du DNS (DDNS) [3] s'est alors imposée comme une solution à la lourdeur de la gestion des zones avec « `majdns` ». L'architecture cible à mettre en place était classique : la configuration

IP des clients est envoyée par un serveur DHCP et celui-ci met à jour une zone DNS dynamique [4]. La gestion du DNS est alors automatisée et se retrouve décentralisée sur les postes clients.

Un autre type d'architecture, qui ne sera pas abordée dans cet article, est possible où ce n'est plus le DHCP qui fait le DDNS mais les clients directement. Puisqu'ils ont connaissance de leur configuration réseau, ils peuvent mettre à jour leur adresse IPv6 temporaire ou obtenue par DHCPv6. Cependant, il est nécessaire de pouvoir authentifier et restreindre les modifications des clients à leurs enregistrements DNS. Le moyen le plus souple pour y parvenir est d'utiliser une infrastructure Kerberos, c'est de cette manière que fonctionnent Active Directory et FreeIPA [5].

Comme historiquement les adresses IPv6 étaient référencées dans le DNS et que certains mécanismes s'appuyaient sur cette information, il nous a paru important de continuer à mettre à jour ces adresses sur la nouvelle architecture. Notre choix s'est initialement porté sur le protocole DHCPv6 pour la distribution d'adresses IPv6 aux hôtes [6] et la mise à jour dynamique dans le DNS [7] avec en parallèle le protocole DHCPv4 pour la partie IPv4.

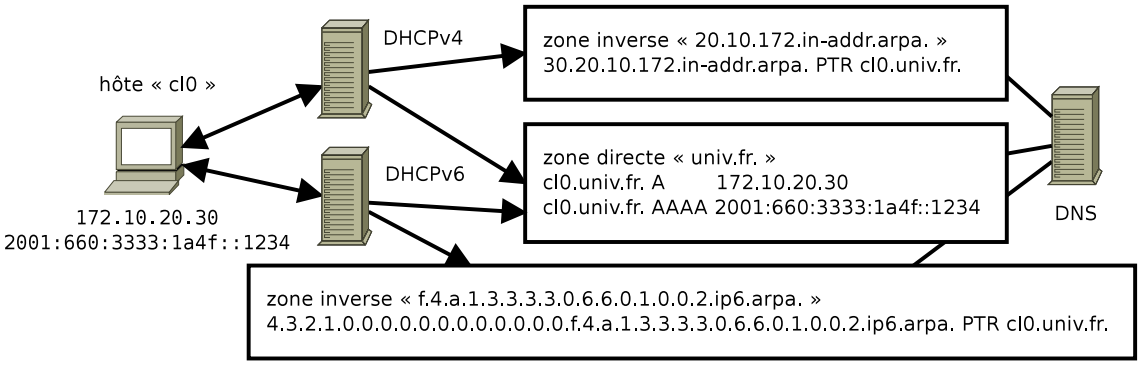

Figure 1 - Architecture initiale avec DHCPv4 et DHCPv6 fonctionnant simultanément

La figure 1 montre un exemple où un hôte « `cl0` » s'adresserait aux serveurs DHCP et obtiendrait en retour deux adresses : une adresse IPv4 `172.10.20.30` et une adresse IPv6 `2001:660:3333:1a4f::1234`. Les DHCP mettraient ensuite à jour la zone directe avec les enregistrements `A` et `AAAA` ainsi que les enregistrements `PTR` sur les deux zones inverses.

2 Rappel sur la détection des conflits de mises à jour

Afin d'éviter des situations de conflits où plusieurs hôtes porteraient le même nom (ou le nom d'un enregistrement ayant été inséré par un autre moyen), il faut pouvoir « sécuriser » les enregistrements dans les zones dynamiques du DNS [8].

Lors des différentes transactions entre le client et le serveur DHCP, le client fournit au serveur un identifiant. Une signature est calculée sur cet identifiant par le serveur DHCP et elle est stockée dans la zone directe en étant associé à l'enregistrement de l'hôte. La RFC 4701 propose d'utiliser des enregistrements de type `DHCID` et une signature SHA-256 [9], ISC-DHCP ne suit pas la RFC et utilise des enregistrements `TXT` et une signature MD5 [10].

Le but est de pouvoir identifier le client qui a mis à jour le DNS en associant la signature de son identifiant à son nom d'hôte et de lui donner la possibilité de modification ou de suppression de manière exclusive. L'enregistrement DNS est alors « verrouillé » tant qu'il n'y a pas eu libération ou expiration du bail DHCP. Le fonctionnement est similaire entre DHCPv4 et DHCPv6 [4] [7].

Avec ISC-DHCP, lorsque la détection des conflits est activée, les conditions à réunir pour faire la mise à jour sont les suivantes :

- il n'existe aucun enregistrement `A` ou `AAAA` correspondant au nom d'hôte qu'on souhaite mettre à jour,

ou

- il existe un enregistrement `A` ou `AAAA` ainsi qu'un enregistrement `TXT` associé au nom qu'on souhaite mettre à jour et il correspond à la signature calculée sur l'identifiant du client.

Dans le cas où il existe un enregistrement `A` ou `AAAA` et que l'enregistrement `TXT` est absent ou diffère de la valeur calculée sur l'identifiant alors le serveur DHCP refusera de faire la mise à jour.

3 Utilisation simultanée de DHCPv4 et DHCPv6 pour le DDNS

3.1 Problématique

L'Université Paris 1 utilise principalement des distributions Linux pour son infrastructure système (messagerie, annuaire, authentification, *etc.*), c'est donc ISC-DHCP et Bind qui ont été utilisés pour le DDNS. Le DHCPv6 est simple à faire fonctionner puisqu'il s'agit d'un service `dhcpd` exécuté avec l'option « `-6` ». La configuration de DDNS est similaire à DHCPv4.

Sur les hôtes c'est « Dibbler » qui a été installé pour Linux et Windows XP. Un client DHCPv6 est déjà intégré à Windows Seven donc aucune installation n'a été nécessaire pour cet OS.

Après quelques tests nous avons constaté des problèmes de conflits lors des tentatives de mise à jour par le DHCPv6 : l'adresse IPv6 n'était pas enregistrée sur la zone directe et inverse. La raison est que les clients et serveurs DHCPv4 et DHCPv6 fonctionnent de manière autonome et utilisent des formats d'identifiants distincts. Dans ces conditions, le mécanisme de verrouillage exclut mutuellement les mises jour du serveur DHCPv4 ou DHCPv6 même si elles proviennent d'un même hôte.

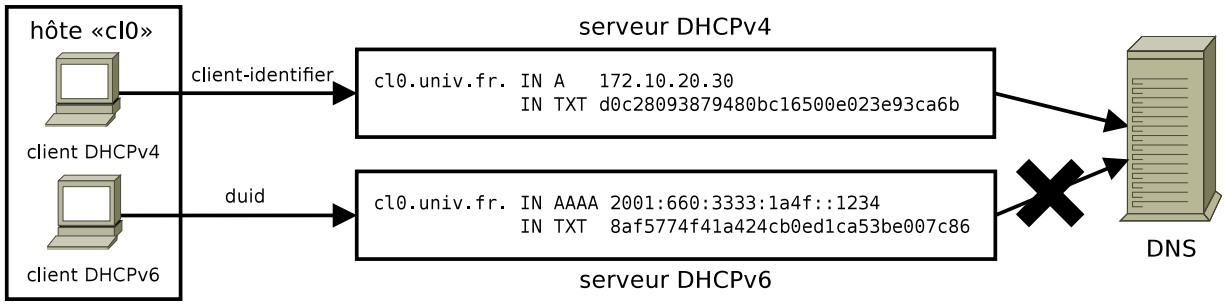

Figure 2 - Conflit de mise à jour entre le DHCPv4 et le DHCPv6.

Le problème est illustré figure 2. L'hôte « `cl0` », qui est en fait composé de deux clients DHCP complètement indépendants, va faire une requête DHCPv4 en fournissant son nom d'hôte et son *client-identifier* [11], en retour il reçoit une adresse IPv4 ainsi que sa configuration (passerelle par défaut, masque de sous-réseau, listes des serveurs DNS, *etc.*). Le serveur DHCPv4 va calculer une signature sur le *client-identifier* et essayer de l'insérer dans le DNS sous la forme d'un enregistrement `TXT` en même temps que les enregistrements `A` et `PTR` [4].

Un processus similaire se déroule en parallèle en IPv6 : le client DHCPv6 demande une adresse et communique son nom et un DUID comme identifiant [6]. Le serveur DHCPv6 calcule une signature sur le DUID et obtient une valeur différente de la signature obtenue par le DHCPv4 : les identifiants sont différents donc les signatures le sont également. Il va également essayer d'insérer dans le DNS un enregistrement `TXT` contenant la signature du DUID en même temps que l'enregistrement `AAAA` et `PTR` [7].

Si l'on suppose que les zones DNS sont vierges et que la détection des conflits est activée sur les deux serveurs, le premier DHCP qui recevra une requête (le DHCPv4 dans l'exemple), pourra sans problème mettre à jour le DNS. Le second DHCP à recevoir les informations va tenter d'effectuer la même opération mais va refuser de faire la mise à jour puisqu'il va trouver un enregistrement `TXT` inconnu associé au même nom d'hôte. Les mises à jour dynamiques d'une même zone DNS (ici « `univ.fr` ») sont donc incompatibles entre DHCPv4 et DHCPv6 à cause d'identifiants différents utilisés par les clients.

Notons que l'utilisation d'une zone spécifique pour les enregistrements IPv6 (par exemple « `ipv6.univ.fr` ») éviterait les collisions avec les enregistrements du DHCPv4. Cependant, il ne serait pas possible de garantir qu'un enregistrement `A` sur « `univ.fr` » et `AAAA` sur « `ipv6.univ.fr` » appartiendraient au même hôte [1].

1. Exemple : un hôte « `cl0.univ.fr` » démarre avec juste un client DHCPv4. Un second hôte, homonyme, démarre avec un client DHCPv4 et DHCPv6. Le résultat est que « `cl0.univ.fr` » pointera vers l'adresse IPv4 du premier hôte et « `cl0.ipv6.univ.fr` » pointera vers l'adresse IPv6 du second.

3.2 L'évolution des protocoles

Ce problème de conflits entre DHCPv4 et DHCPv6 a été soulevé dès 2006 [8]. Le cœur du problème étant l'incompatibilité des identifiants, la RFC 4361 suggère une harmonisation en proposant que les clients DHCPv4 et DHCPv6 utilisent un DUID unique comme identifiant [12]. Il serait alors possible de mettre à jour les enregistrements `A` et `AAAA` sans risque de conflit et les serveurs pourraient fonctionner de manière indépendante. Actuellement, la publication de cette RFC n'a donné lieu à aucune implémentation commerciale mais la version 4.3 d'ISC-DHCP pourrait suivre ces recommandations [13].

Une autre solution récemment explorée est l'ajout d'une nouvelle option au protocole DHCPv6 pour permettre la communication de l'adresse matérielle de l'hôte entre le relais et le serveur DHCPv6 [14] . Le serveur DHCPv6 se servira de l'adresse matérielle pour identifier le client par une méthode qui reste à déterminer [2]. L'inconvénient par rapport à la RFC 4361 est qu'il ne serait plus possible d'utiliser des serveurs indépendants.

3.3 Le fonctionnement sans détection des conflits de mises à jour

Pour contourner le problème de conflits, certains tutoriaux [15] suggèrent de désactiver la détection des conflits sur le serveur DHCPv6. Ce mode de fonctionnement est à proscrire puisqu'il empêche un fonctionnement correct de DDNS.

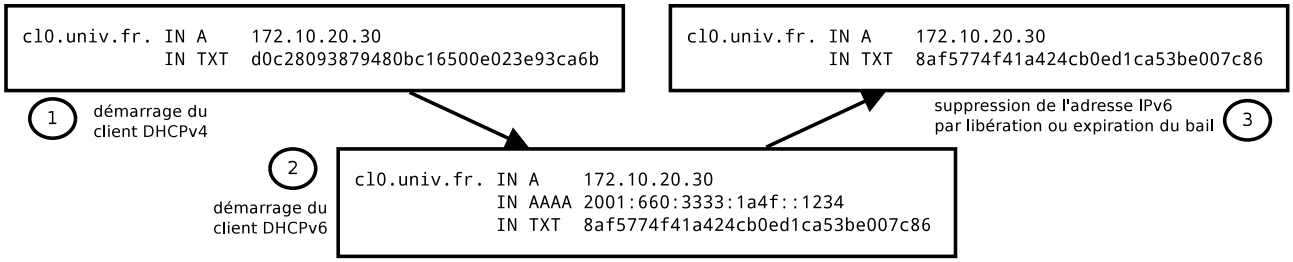

Figure 3 - Évolution de la zone directe avec un serveur DHCPv6 configuré sans détection des conflits.

Prenons l'exemple d'un serveur DHCPv4 avec la détection des conflits activée et d'un serveur DHCPv6 configuré sans détection, les deux sont paramétrés pour faire des mises à jour dynamiques sur la zone « `univ.fr` ». Un hôte nommé « `cl0` », disposant d'un client DHCPv4 et d'un client DHCPv6, est mis sous tension. Si le client DHCPv4 démarre en premier le serveur DHCPv4 pourra ajouter sans problème son adresse IPv4 (`172.10.20.30`) ainsi qu'un enregistrement `TXT` calculé à partir de son *client-identifier* (figure 3.1).

Au démarrage du client DHCPv6, le serveur DHCPv6 va rajouter l'enregistrement `AAAA` avec l'adresse IPv6 qu'il a distribuée (`2001:660:3333:1a4f::1234`) mais il va également *remplacer* l'enregistrement `TXT` par une nouvelle valeur calculée sur le DUID. Comme la détection des conflits est désactivée, il ignore le `TXT` déjà présent censé « protéger » les enregistrements de l'hôte « `cl0` » (figure 3.2).

En apparence, le but est atteint : dans la zone directe, le client a réussi à mettre à jour son adresse IPv4 et IPv6 mais il n'y a aucune garantie sur la provenance de l'enregistrement `AAAA`. De plus, le serveur DHCPv4 ne peut plus modifier l'enregistrement `A` puisque l'enregistrement `TXT` n'a plus de rapport avec le *client-identifier* du client DHCPv4.

Lorsque le client DHCPv6 va libérer son adresse ou lorsque que le bail de l'adresse IPv6 va expirer, le serveur DHCPv6 va supprimer l'enregistrement `AAAA` mais il va laisser l'enregistrement `TXT` (figure 3.3). Ce comportement est dû à la logique de la mise à jour dynamique : le `TXT` n'est pas supprimé car il reste un enregistrement `A` associé au nom d'hôte [8]. Le résultat final est un couple d'enregistrements `A` et `TXT` incohérents ne pouvant ni être supprimés par le DHCPv4 à cause de la détection des conflits, ni par le DHCPv6 à cause de l'algorithme de DDNS.

Avec un tel paramétrage, les zones DNS sont rapidement « polluées » par des enregistrements `TXT` qui verrouillent improprement des enregistrements `A`. Au fil des libérations d'adresses et des expirations de baux DHCP, ces « mauvais » enregistrements vont s'accumuler et provoquer de plus en plus d'incohérences entre le contenu des zones et la réalité des hôtes connectés sur le réseau.

2. L'utilisation d'un serveur DHCP unifié qui gèrerait les requêtes DHCPv4 et DHCPv6 et qui stockerait les informations dans une seule base de données semble la solution la plus probable.

3.4 Conclusion sur la cohabitation DHCPv4/DHCPv6

La mise à jour d'une zone DNS dynamique par un serveur DHCPv4 et un serveur DHCPv6 de manière concurrente nécessite une modification au niveau des clients DHCPv4 (RFC 4361) ou des relais et serveurs DHCPv6 (RFC 6939).

Actuellement aucune solution n'est disponible pour ce type d'architecture mais le support de DHCPv6 étant de plus en plus standard dans les systèmes d'exploitation, on peut imaginer qu'une plus grande utilisation du protocole fera ressortir ce problème d'incompatibilité et accélérera l'adoption d'une RFC ou même l'abandon du fonctionnement en mode double-pile.

4 Mise à jour automatisée d'adresses EUI-64

4.1 Principe de base

La solution du DHCPv6 pour le DDNS des adresses IPv6 n'étant pas satisfaisante, l'architecture a été modifiée pour ne conserver qu'un serveur DHCPv4 exécutant un script de mise à jour des adresses EUI-64. Cette méthode a été proposée par l'UNSW (Université de Nouvelle-Galles du Sud) [16] et s'appuie sur la capacité d'un serveur DHCP à exécuter un script lors de l'affectation, la libération ou l'expiration d'un bail grâce à un composant interne à ISC-DHCP nommé « `dhcp-eval` » [17]. Si on lui communique le préfixe IPv6, un serveur DHCPv4 dispose de suffisamment d'informations pour calculer l'adresse EUI-64 de l'hôte et la stocker (ou la retirer) du DNS sur les zones directes et inverses. Le script doit aussi « valider » l'existence de l'adresse en utilisant un ping IPv6 avant la mise à jour car il n'y a aucune garantie que le protocole IPv6 soit activé sur l'hôte et qu'il soit correctement configuré pour utiliser une adresse EUI-64.

Les prérequis pour les hôtes sont donc les suivants :

- ils doivent utiliser une adresse IPv6 EUI-64,
- ils doivent répondre au ping IPv6.

Cette configuration est simple à mettre en œuvre pour les systèmes d'exploitation Windows, Linux et MacOS et elle est souvent standard pour les imprimantes et les copieurs reliés au réseau.

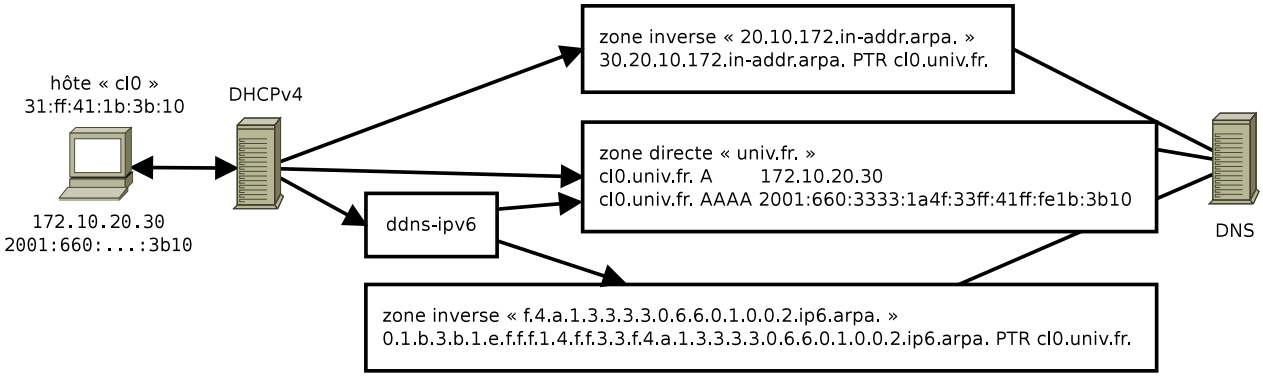

Figure 4 - Deuxième architecture avec un seul DHCPv4 et un script de mise à jour « `ddns-ipv6` »

La figure 4 illustre le mécanisme. L'adresse IPv6 de l'hôte « `cl0` » n'est plus obtenue par DHCPv6 mais par le mécanisme d'auto-configuration standard d'IPv6 [3]. Le serveur DHCPv4 distribue une adresse IPv4 (`172.10.20.30`) et met à jour dynamiquement la zone directe et la zone inverse avec les enregistrements `A` et `PTR` de manière classique.

Le DHCP exécute ensuite automatiquement le script « `ddns-ipv6` », ce dernier calcule l'adresse EUI-64 correspondant à l'hôte [4] et après avoir fait toutes les vérifications nécessaires (validation de l'adresse par ICMPv6, détection d'une adresse IPv6 déjà existante dans le DNS, *etc.*) il met à jour les enregistrements `AAAA` et `PTR` avec l'adresse IPv6.

À la libération de l'adresse ou l'expiration du bail, le script est ré-exécuté et supprime les enregistrements IPv6 `AAAA` et `PTR`.

3. Le système d'exploitation de l'hôte ou le matériel aura dû être préalablement paramétré pour utiliser une adresse EUI-64.
4. Dans cet exemple, le préfixe IPv6 est `2001:660:3333:1a4f::/64` et l'adresse MAC de l'hôte est `31:ff:41:1b:3b:10` donc l'adresse EUI-64 obtenue est `2001:660:3333:1a4f:33ff:41ff:fe1b:3b10`.

4.2 Améliorations

Le script, tel qu'il est fourni par l'UNSW, a deux limitations majeures :

- les enregistrements IPv6 `AAAA` et `PTR` ne sont pas supprimés si le bail expire,
- il n'y a pas de vérification des conflits de mise à jour.

Le problème de suppression lors de l'expiration du bail peut facilement être contourné en stockant dans des variables l'adresse IPv4 associée au bail ainsi que l'adresse matérielle [5]. Le problème de vérification des conflits est plus long à résoudre car il faut reproduire le mécanisme intégré au code source du serveur ISC-DHCP.

Il est important de noter que `dhcp-eval` exécute le script *systématiquement* à l'affectation, la libération et l'expiration d'un bail, quelque soit le résultat de la tentative de mise à jour du DNS par le serveur DHCP. Si un hôte se présente avec un nom déjà existant dans la zone DNS, le DDNS échouera, mais le script sera quand même exécuté, il est donc très important de faire des vérifications préalables pour ne pas rajouter des adresses EUI-64 sur des enregistrements homonymes.

4.3 Le script « `ddns-ipv6` »

Le script de mise à jour doit être installé localement sur le serveur DHCP. Le script initial de l'UNSW a été progressivement amélioré, le langage bash a été conservé et l'implémentation de la détection des conflits a été réalisée avec des requêtes DNS en utilisant l'outil `dig`. Les calculs d'adresses EUI-64 et des noms de zones inverses sont faits avec `ipv6calc`. Enfin, les mises à jour du DNS sont réalisées avec `nsupdate`.

D'autres approches auraient été possibles comme par exemple calculer la signature du *client-identifier* et comparer la valeur des enregistrements `TXT` pour reproduire exactement l'algorithme de mise à jour dynamique de DHCP ou utiliser le langage Perl avec des librairies telles que `Net::DNS` au lieu de conserver bash et des exécutables externes.

4.4 Configuration du DHCP

Au niveau du serveur DHCP, la configuration minimale d'un *subnet* consiste à déclarer le préfixe IPv6 dans une variable et rajouter un bloc « `on commit { }` » et « `on release or expiry { }` ». Ces blocs d'instructions sont génériques et seront plutôt insérés sous la forme d'une instruction « `include` » pour ne pas surcharger les fichiers de configuration.

Les variables `cache-ipv4` et `cache-mac` ne sont déclarées qu'à l'affectation de l'adresse IP et sont ensuite stockées et disponibles dans la base de données des baux du serveur DHCP. La commande `execute` permet d'appeler un script avec des paramètres. Ici ce sont le préfixe IPv6, le nom d'hôte, l'adresse MAC et l'adresse IPv4 qui sont passés en arguments. Dans le bloc `on release or expiry` on rajoute le paramètre « `-d` » pour indiquer au script qu'il s'agit d'une suppression.

L'adresse IPv4 de l'hôte n'a, à priori, aucune utilité pour un script qui manipule des adresses IPv6 mais elle permet de faire une détection de conflits basique. Le script interroge le DNS pour chercher l'adresse IPv4 associé au nom d'hôte et il compare l'adresse reçue avec celle qui a été passée en paramètre. Si elles sont identiques il peut continuer les tests, sinon cela signifie qu'un autre hôte, homonyme, s'est déjà enregistré et qu'on se trouve dans une situation de conflit.

Exemple de configuration d'un *subnet* pour un serveur ISC-DHCP :

```
subnet 172.10.20.0 netmask 255.255.255.0 {
  option routers 172.10.20.1;
  range 172.10.20.2 172.10.20.254;

  set prefix = "2001:660:3333:1a4f";

  on commit {
    if (not static) {
      set cache-ipv4 = binary-to-ascii(10, 8, ".", leased-address);
```

5. L'expiration du bail ne donne pas lieu à un échange entre le client et le serveur DHCPv4 et ces deux informations ne sont pas accessibles par ISC-DHCP de manière standard.

```
      set cache-mac = binary-to-ascii(16, 8, ":", substring(hardware, 1, 6));
      set ddns-hostname = option host-name;

      execute ("/usr/local/bin/ddns-ipv6", prefix,
               ddns-hostname, cache-mac, cache-ipv4);
    }
  }
  on release or expiry {
    execute ("/usr/local/bin/ddns-ipv6", prefix,
             "-d", ddns-hostname, cache-mac, cache-ipv4);
  }
}
```

5 Retour d'expérience

5.1 Migration des hôtes vers la nouvelle infrastructure

Il n'y a pas eu de problème lié à la mise à jour dynamique des adresses EUI-64 lorsque des postes clients ont été migrés de réseaux en DHCP « statique » vers la nouvelle infrastructure en DHCP dynamique. Si les postes étaient bien configurés sur l'ancienne infrastructure, leur adresse IPv6 était correctement mise à jour sur la nouvelle infrastructure.

Comme les adresses IPv6 EUI-64 étaient largement utilisées sur nos réseaux depuis 2005, la bascule a pu se faire sans modification des hôtes. Cela n'aurait pas été le cas avec l'utilisation de DHCPv6 car il aurait été nécessaire d'installer des clients spécifiques sous certains systèmes d'exploitation et modifier la configuration des routeurs pour activer le protocole et le relayage.

5.2 Problèmes liés au DHCPv4 et correction avec le script

Les problèmes rencontrés étaient indépendant de l'utilisation du script « `ddns-ipv6` » et concernaient le fonctionnement du DDNS en DHCPv4. Les tests initiaux ont été faits avec Debian Squeeze et la version 4.1.x d'ISC-DHCP comportait certains dysfonctionnements, notamment :

- enregistrement `PTR` parfois non mis à jour (ajout ou suppression),
- enregistrement `A` supprimé mais enregistrement `TXT` associé subsistant (ou inversement).

L'utilisation de la version 4.2.x a résolu une partie des problèmes mais, pour s'assurer de la bonne mise à jour des zones, nous avons modifié le script « `ddns-ipv6` » pour qu'il effectue des vérifications supplémentaires afin d'insérer ou supprimer des enregistrements IPv4 `A`, `TXT` ou `PTR` qui auraient été mal gérés par le binaire `dhcpd`.

Le script a maintenant une double utilité : la mise à jour dynamique des adresses EUI-64 et la fiabilisation de la mise à jour des adresses IPv4.

5.3 Le fonctionnement en mode *failover*

ISC-DHCP peut fonctionner en mode *failover* où deux serveurs DHCP sont actifs en même temps. Ils communiquent via un protocole spécifique et s'échangent des informations sur les baux en cours d'utilisation. En cas de panne d'un serveur, la gestion de ses baux peut être reprise par le serveur qui reste opérationnel.

L'utilisation du script « `ddns-ipv6` » ne pose pas de problème dans ce mode de fonctionnement mais il faut s'assurer que la même version du script est bien présente sur les deux serveurs DHCP.

6 Conclusion

La cohabitation d'un serveur DHCPv4 et d'un serveur DHCPv6 est un scénario à envisager pendant la phase de transition vers du « tout-IPv6 ». Malheureusement, à l'heure actuelle, cette solution ne permet pas la mise à jour dynamique du DNS de manière satisfaisante.

Le protocole DHCPv6 s'enrichit régulièrement de nouvelles options mais il ne semble pas encore mature pour une utilisation généralisée. Son amélioration est en cours : le groupe de travail en charge du DHCP à l'IETF étudie par exemple la possibilité d'utiliser DHCPv6 pour mettre à jour dynamiquement dans le DNS les adresses IPv6 autoconfigurées [18].

L'installation de clients de type *DynDNS*, l'analyse du trafic réseau ou l'utilisation de fonctionnalités avancées de certains matériels pourrait également répondre au besoin mais nécessiterait une configuration spécifique des hôtes ou du matériel.

En attendant une évolution des clients ou des serveurs et une stabilisation de DHCPv6, la méthode introduite par l'UNSW et améliorée par l'Université Paris 1 permet d'assurer le service de mise à jour dynamique du DNS sans changement important au niveaux des hôtes et de l'infrastructure système et réseau. Le fait d'utiliser DHCPv4 et les adresses IPv6 EUI-64, deux mécanismes très standards, permet un fonctionnement quasi-universel du DDNS IPv6, fonctionnant aussi bien avec des imprimantes réseau qu'avec des systèmes d'exploitation qui ne supportent pas DHCPv6 nativement.

Bibliographie

[1] Benoît Branciard, David Chopard-Lallier, et Yvonne Girard. Déploiement d'IPv6 à l'Université Paris 1 – Panthéon-Sorbonne. Dans *Actes du congrès JRES2005*, Marseille, dec 2005. http://2005.jres.org/paper/61.pdf.

[2] Robert M. Hinden et Stephen E. Deering. RFC 4291 - IP version 6 addressing architecture. fev 2006. http://tools.ietf.org/html/rfc4291.

[3] Jim Bound et Paul Vixie. RFC 2136 - Dynamic updates in the Domain Name System (DNS UPDATE). avril 1997. http://tools.ietf.org/html/rfc2136.

[4] Mark Stapp et Bernie Volz. RFC 4702 - The Dynamic Host Configuration Protocol (DHCP) client Fully Qualified Domain Name (FQDN) option. oct 2006. http://tools.ietf.org/html/rfc4702.

[5] Fedora Documentation Project. Chapter 10. identity : Managing DNS. http://docs.fedoraproject.org/en-US/Fedora/18/html/FreeIPA_Guide/Working_with_DNS.html.

[6] Jim Bound, Bernie Volz, Ted Lemon, Charles E. Perkins, et Mike Carney. RFC 3315 - Dynamic Host Configuration Protocol for IPv6 (DHCPv6). juil 2003. http://tools.ietf.org/html/rfc3315.

[7] Bernie Volz. RFC 4704 - The Dynamic Host Configuration Protocol for IPv6 (DHCPv6) client Fully Qualified Domain Name (FQDN) option. oct 2006. http://tools.ietf.org/html/rfc4704.

[8] Mark Stapp et Bernie Volz. RFC 4703 - Resolution of Fully Qualified Domain Name (FQDN) conflicts among Dynamic Host Configuration Protocol (DHCP) clients. oct 2006. http://tools.ietf.org/html/rfc4703.

[9] Mark Stapp, Ted Lemon, et Andreas Gustafsson. RFC 4701 - A DNS Resource Record (RR) for encoding Dynamic Host Configuration Protocol (DHCP) information (DHCID RR). oct 2006. http://tools.ietf.org/html/rfc4701.

[10] Ted Lemon. dhcpd.conf(5) - linux man page. http://linux.die.net/man/5/dhcpd.conf.

[11] Ralph Droms. RFC 2131 - Dynamic Host Configuration Protocol. mars 1997. http://tools.ietf.org/html/rfc2131.

[12] Ted Lemon et Bill Sommerfeld. RFC 4361 - Node-specific client identifiers for Dynamic Host Configuration Protocol version four (DHCPv4). fev 2006. http://tools.ietf.org/html/rfc4361.

[13] Laura Hendriksen, Vicky Risk, et Eddy Winstead. ISC and the DNS company product roadmap. oct 2013. https://deepthought.isc.org/getAttach/86/AA-00974/webinar-ISC+Product+RoadmapOct2013.pdf.

[14] Gaurav Halwasia, Shwetha Bhandari, et Wojciech Dec. RFC 6939 - Client link-layer address option in DHCPv6. mai 2013. http://tools.ietf.org/html/rfc6939.

[15] Mike MacLeod. IPv6 part 8 : Configuring DNS and DHCPv6 on an IPv6 network. aout 2011. http://www.mmacleod.ca/blog/2011/08/ipv6-part-8-configuring-dns-and-dhcpv6-on-an-ipv6-network/.

[16] Peter Chubb. Dynamic DNS for IPv6 in a mixed-stack environment. jan 2012. http://www.gelato.unsw.edu.au/IA64wiki/IPv6DDNS.

[17] Ted Lemon. dhcp-eval(5) - linux man page. http://linux.die.net/man/5/dhcp-eval.

[18] Sheng Jiang, Gang Chen, et Suresh Krishnan. Registering self-generated IPv6 addresses in DNS using DHCPv6. août 2013. http://tools.ietf.org/html/draft-ietf-dhc-addr-registration.