Comment vivre avec un réseau IPv6 pur ?

Matthieu Herrb
CNRS LAAS
7, avenue du Colonel Roche, 31077 Toulouse Cedex 4
Université de Toulouse : UPS, INSA, INP, ISAE, UT1, UTM, LAAS.

Résumé

Lorsque l'on évoque le déploiement d'IPv6, ce sont principalement les mécanismes de transition, c'est à dire la phase de cohabitation avec IPv4, sous forme de double-pile qui sont étudiés. Or, face à l'épuisement annoncé des adresses IPv4, il apparaît intéressant d'évaluer la faisabilité d'un déploiement utilisant uniquement IPv6 en interne en limitant au maximum le recours aux adresses IPv4, qui deviennent alors redondantes.

La première partie de cet article traite du déploiement en réseau local, en particulier des problèmes concrets qui peuvent se poser à un administrateur système ou réseau, pour différents types de postes de travail, s'il souhaite les connecter sans adresse IPv4.

Ensuite l'accès à l'Internet IPv4 de postes en IPv6 seul est abordé. Les mécanismes basés sur le RFC 3142 puis sur NAT64 sont présentés avec des possibilités de déploiement réel.

Les difficultés et les problèmes soulevés par cette étude peuvent servir à éclairer les choix de type d'adressage et des modes de déploiement lors de l'installation de nouveaux réseaux.

Mots clefs

IPv6, DHCPv6, NAT64, DNS64, mécanismes de transition

1 Introduction

La double pile IPv4/IPv6 est souvent présentée comme la voie naturelle de migration vers IPv6. S'il est vrai que cette approche permet une transition en douceur, l'objectif final reste la disparition d'IPv4.

Conserver durant une transition trop longue la connectivité IPv4 pour l'ensemble des nœuds d'un réseau maintient de fait IPv6 dans un statut de protocole optionnel. Même lorsque la politique du résolveur DNS favorise les adresses IPv6, la commutation vers IPv4 en cas de défaillance constitue un refuge confortable qui n'encourage pas au déploiement de services et d'infrastructures IPv6 opérationnels et performants.

Ensuite l'existence de la double pile entraîne une augmentation de la complexité de la gestion du parc de machines. Il est nécessaire de maintenir des tables DNS, un inventaire ou encore des politiques de sécurité pour les deux familles d'adresses. Par ailleurs les différences de comportement des différents systèmes à double-pile rendent le diagnostic de problèmes et l'assistance aux utilisateurs dans ce contexte nettement plus difficiles.

En admettant l'axiome selon lequel la transition vers IPv6 est inéluctable en raison de la pénurie d'adresses IPv4 et d'autres limitations du protocole IPv4, il apparaît préférable que la double pile ne soit qu'une phase transitoire destinée à durer le moins longtemps possible. C'est pourquoi des expérimentations d'exploitation de postes clients en IPv6 seul sont utiles, afin de préparer au plus vite l'abandon total d'IPv4 [1].

Cet article présente des tests réalisés sur des postes de travail dans un réseau local n'utilisant que des adresses IPv6, et ayant recours pour l'accès aux sites Internet en IPv4 à deux mécanismes de transition parmi tous ceux qui ont été proposés [2][3] : *faithd* (RFC 3124) puis NAT64 (RFC 6052, 6144 à 6146).

Dans le cas où IPv6 n'arriverait pas à s'imposer et où la traduction d'adresses IPv4 resterait comme solution satisfaisante sur le long terme face à la pénurie d'adresses (en particulier parce que ces solutions permettent plus aisément un contrôle centralisé de l'utilisation d'internet) toute cette étude deviendrait sans objet.

2 Description de la configuration expérimentale.

Pour l'ensemble des tests décrits dans cet article, la configuration expérimentale utilisée repose sur le commutateur-routeur du cœur de réseau du laboratoire et sur un VLAN dédié au réseau IPv6 seul. Ce VLAN est diffusé sur l'ensemble des commutateurs d'extrémités sur lesquels l'affectation des ports aux VLAN se fait en fonction de l'adresse MAC, via le protocole propriétaire VMPS. Cette configuration permet de tester un poste client en environnement IPv6 pur en modifiant simplement l'affectation de son adresse MAC vers le VLAN utilisé pour l'expérimentation.

Ce VLAN expérimental est connecté au reste du réseau du laboratoire par une passerelle qui est réalisée à l'aide d'une machine virtuelle disposant d'une interface réseau dans le VLAN dédié à l'expérience et d'interfaces dans les autres VLAN du laboratoire. Les services de routage, de configuration du réseau, de serveur DNS proxy sont assurés par cette passerelle.

L'utilisation d'une machine virtuelle permet une grande souplesse durant les tests, facilitant l'expérimentation des systèmes différents ou des configurations différentes avec possibilités de revenir en arrière.

Le commutateur/routeur central (de marque Cisco) ne peut pas être utilisé directement, essentiellement parce qu'il ne peut assurer que le routage d'IPv6 et ne propose aucun des autres services nécessaires pour ces expérimentations. De plus, réaliser de nombreux changements de configuration sur un équipement central, critique pour l'activité de l'ensemble du laboratoire comporte des risques qu'il a été jugé préférable d'éviter.

3 Déploiement des postes clients

Le déploiement des postes consiste à définir un plan d'adressage et à mettre en place les mécanismes qui vont permettre de configurer les postes clients : adresse IP, routage et mécanisme de résolution des noms (DNS). En IPv4, ce déploiement est fait soit manuellement soit par le protocole DHCP.

IPv6 propose une méthode de déploiement originale, maintenant bien connue : l'auto-configuration de l'adresse IP à partir de son adresse MAC, couplée au protocole de découverte de la topologie du réseau et de l'adresse de la passerelle [4].

Cette méthode est implémentée sur tous les équipements IPv6, qu'ils soient routeurs ou postes clients et sa mise en œuvre est très facile. Elle présente cependant plusieurs inconvénients dans sa version initiale :

- rien n'est prévu pour la configuration du service DNS. Il reste donc nécessaire de renseigner manuellement les adresses des serveurs utilisables. Ce qui peut être problématique, en particulier dans un environnement nomade. Le RFC 6106 (voir ci-dessous) propose une extension au mécanisme de découverte de la passerelle pour également découvrir les serveurs de nommage.
- Le fait que l'adresse v6 d'un poste dérive de l'adresse MAC laisse apparaître des informations sur le poste client (constructeur de la carte réseau). Diverses variantes du mécanisme d'auto-configuration permettent de créer des adresses IPv6 temporaires qui protègent davantage la vie privée de leur utilisateur.

3.1 Configuration des serveurs DNS sur les postes clients

Le RFC 5006, remplacé et complété par le RFC 6106 a proposé une nouvelle option du protocole de diffusion des adresses de routeurs (*router advertisement*) pour diffuser également les adresses des serveurs de noms du réseau. Cette option est appelée RDNSS : *Recursive DNS* Server. Elle peut être présente dans les annonces de routeur. De plus le RFC prévoit explicitement la cohabitation de l'option RDNSS avec la possibilité pour un client d'utiliser DHCPv6 pour obtenir cette information.

Le principal problème soulevé par cette extension des paquets RA concerne la façon dont les postes clients traitent l'information ainsi reçue. En effet la liste des serveurs DNS est traditionnellement gérée dans le fichier */etc/resolv.conf* ou son équivalent dans la base des registres. Tant que le client DHCP (un processus en espace utilisateur) était le seul processus susceptible de mettre à jour cette information, il était relativement simple de garantir la cohérence des données dans ce fichier.

Dans le cas d'une option RDNSS arrivant dans un paquet RA, cette information arrive au niveau du noyau qui ne peut directement mettre à jour le fichier *resolv.conf*. Il faut donc un processus supplémentaire qui attend d'être informé de l'arrivée d'un RDNS par le noyau pour effectuer la mise à jour, en se synchronisant avec un éventuel client DHCP (v4 ou v6).

Le déploiement d'un réseau IPv6 pur, en choisissant l'une ou l'autre des méthodes de configuration des serveurs DNS pour un poste donné permet de garantir que *resolv.conf* ne contiendra bien que des serveurs accessibles en v6.

3.2 Implémentation de RDNSS sur les serveurs

Sur les systèmes BSD, c'est le démon *rtadvd* qui assure l'implémentation du protocole ND. Il existe plusieurs correctifs expérimentaux ajoutant la possibilité de diffuser les paquets avec l'option RDNSS.

Sur Linux c'est le démon *radvd*[1] qui est chargé de la diffusion des RA. Il supporte les options RDNSS depuis la version 1.0 en 2006, et a été mis à jour pour supporter pleinement le RFC 6106 dans sa version 1.7, datée de janvier 2011.

Une alternative compatible à la fois avec les systèmes BSD et Linux est le démon *radns*[2].

Enfin les routeurs Cisco récents supportent à la fois DHCPv6 et RDNSS avec la version 15 de leur système IOS.

4 Comportement d'un poste client en v6 seul

Une fois le mécanisme de déploiement choisi et configuré au niveau d'une passerelle réseau, il reste à installer des postes de travail et des équipements connexes dans le réseau IPv6 ainsi créé.

4.1 Postes de travail

L'outil « *Network Manager* » utilisé par les distributions Linux récentes pour configurer les interfaces réseau considère par défaut (sur Ubuntu 10.04 et Fedora 14 au moins) que la configuration d'une interface a échoué si celle-ci n'a pas réussi à obtenir une adresse v4. Heureusement, une option de configuration permet d'ignorer l'absence d'adresse IPv4, mais elle doit être activée manuellement.

Une fois cette option sélectionnée, un poste de travail Linux peut s'insérer sans autre difficulté sur le réseau IPv6 pur. *Network Manager* permet en outre de choisir entre la configuration par DHCPv6 ou l'auto-configuration mais propose également un mode « tout automatique » qui tente une requête DHCPv6 si aucun RA n'est reçu.

Sur les systèmes BSD, le mécanisme traditionnel de configuration des interfaces réseau permet de gérer l'auto-configuration IPv6. Aucune erreur n'est signalée si IPv4 n'est pas configuré. Par contre, l'intégration de clients DHCPv6 et de RFC 6106 est plus délicate et nécessite l'installation de logiciels supplémentaires et une modification des scripts de démarrage.

Mac OS X supporte à la fois l'auto-configuration via RDNSS ou DHCPv6 à partir de la version 10.6.8 (mise à jour de *Snow Leopard*) ou 10.7 *(Lion*).

Windows XP n'est pas capable d'utiliser un serveur DNS accessible uniquement en IPv6, il est donc impossible de déployer ce système sur un réseau IPv6 seul. Les versions suivantes, Vista et Windows 7, supportent quant à elles les deux modes de configuration et fonctionnent sans adresse IPv4.

Parmi les applications qui ont été utilisées durant l'expérience (les outils standards de bureautique, de messagerie et de navigation Internet sur Mac OS X, Windows 7 et Linux Ubuntu 11.04 et Fedora 15), la plupart supportent correctement la connexion en IPv6 seul. En particulier, l'essentiel du travail de rédaction de cet article, incluant les figures et les recherches de références sur le Web a été réalisé dans cet environnement. Les deux exceptions notables parmi les logiciels les plus courant sont le gestionnaire de base de données *MySQL* et le logiciel de téléphonie *Skype*. Quelques logiciels spécifiques un peu anciens (des développements internes du laboratoire notamment...) n'ont pas fonctionné correctement, soit en raison de l'absence totale de support, soit en raison d'erreurs de codage du support d'IPv6 qui les rendent partiellement inutilisables.

D'autres études ([1] et [5]) ont également exploré le fonctionnement d'applications en IPv6 seul et présentent des résultats complémentaires.

4.2 Problèmes

Dans l'expérience présentée ici, il n'existe que très peu d'équipements annexes capables de fonctionner dans un réseau sans adresses IPv4. Commutateurs, points d'accès wifi, imprimantes réseau, automates programmables,... ne sont souvent adressables qu'en IPv4.

---

[1] http://www.litech.org/radvd/

[2] http://hack.org/mc/hacks/radns/

L'administration des équipements IPv4 seuls peut se faire en diffusant à ceux-ci et seulement à ceux-ci des adresses IPv4 privées (RFC 1918) et en utilisant le mécanisme NAT64 présenté ci-dessous pour y accéder.

Par ailleurs, un autre aspect du déploiement des postes clients, l'installation du système est également pratiquement impossible à faire via le réseau dans un environnement IPv6 pur : les BIOS des postes de travail ne savent pas réaliser un démarrage réseau en IPv6. En attendant l'apparition de BIOS capables de faire de l'IPv6, il est là aussi possible d'utiliser un adressage IPv4 privé uniquement durant la phase d'installation du système.

L'existence de logiciels ne supportant pas encore IPv6 doit également être prise en compte au moment de décider de se débarrasser d'IPv4. Dans de nombreux cas, en particulier pour des logiciels libres, il existe des correctifs ou des versions de développement qui corrigent ces problèmes, mais leur utilisation dans un réseau de production peut être difficile.

Enfin, le protocole IPv6 repose sur l'utilisation du multicast ethernet pour un certain nombre de fonctions. En plus des difficultés qui peuvent être causées en wifi par des points d'accès qui ne relaient pas les trames multicast vers le réseau radio dans le but d'économiser la bande passante, il existe également des commutateurs filaires à bas coût qui ne relaient pas les trames multicast. Ces équipements doivent être éliminés d'un réseau IPv6 afin d'obtenir un service utilisable.

5 Connexion vers l'Internet IPv4

Afin d'atteindre des services réseaux accessibles uniquement en IPv4, il faut disposer d'une passerelle qui va d'une manière ou d'une autre relayer le trafic IPv6 vers les sites IPv4. Dans le futur, des mécanismes de transition prévus pour être déployés par les fournisseurs d'accès internet tel que DS-Lite (*Dual-Stack lite* – RFC6333 [6]) pourraient assurer la traduction directement au niveau du réseau de l'opérateur, qui ne distribuerait alors plus du tout d'adresse IPv4. Mais actuellement, la situation la plus courante (sur le réseau Renater et chez des fournisseurs d'accès qui proposent une connectivité IPv6 comme Free ou Nerim) est que l'on dispose d'un petit nombre d'adresses IPv4 (parfois une seule) et d'un préfixe IPv6 /48 ou /64. La passerelle est alors installée sur le site qui souhaite déployer le réseau IPv6 seul.

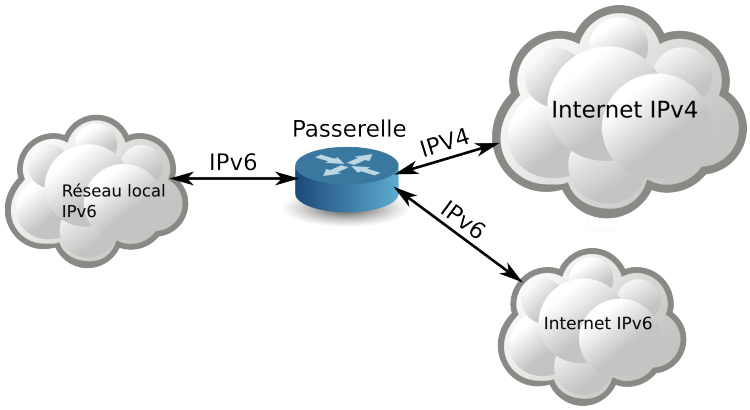

Figure 1: Principe général

Dans cet article, il n'est volontairement pas du tout question des mécanismes permettant à une machine sur un site ne disposant que d'adresses IPv4 de se connecter à un service disponible seulement en IPv6. Dans le contexte des réseaux de l'enseignement supérieur et de la recherche, il est facile d'obtenir un accès IPv6 natif qui résout définitivement cette question. Et dans le cas où un accès IPv6 natif n'est pas envisageable, la littérature sur cette problématique ainsi que les outils permettant d'accéder à IPv6 via divers types de tunnels sont plus qu'abondants, par exemple en utilisant un tunnel GRE [7].

5.1 RFC 3142

Le RFC 3142 définit une manière de traduire les adresses IPv6 en adresses IPv4 dans un relais afin de faciliter la transition. Le projet KAME a conçu ce protocole, rédigé le RFC et l'a implémenté dans le démon *faithd* [8]. Celui-ci est disponible en standard sur les systèmes BSD qui ont utilisé l'implémentation IPv6 de KAME.

Au départ de cette expérimentation, c'est ce mécanisme qui s'est naturellement imposé pour les premiers tests. Lorsque *faithd* reçoit des paquets IPv6 à destination d'une plage d'adresses réservées pour cet usage, il se connecte à l'adresse IPv4 obtenue à partir des quatre derniers octets de l'adresse IPv6 d'origine et transmet les données du paquet IPv6 initialement reçu. Réciproquement, les paquets IPv4 reçus sur le port géré par le démon sont ré-émis vers l'adresse IPv6 d'origine.

Afin de permettre effectivement à une machine qui n'a que des adresses IPv6 d'établir une connexion vers une machine IPv4, il faut en plus avoir un service de noms qui va convertir l'adresse IPv4 en adresse IPv6. En marge du projet KAME, le serveur *totd* a été développé pour cela.

Exemple :

Supposons que le préfixe `2001:db8:4819:6:ffff::/96` soit réservé pour faithd. Une machine IPv6 souhaite établir une connexion HTTP vers `www.example.com` qui n'a que l'adresse IPv4 `192.9.0.12`. Le serveur de nom *totd*, agissant comme serveur cache DNS va fournir comme réponse à la requête de type AAAA pour `www.example.com` l'adresse IPv6 suivante construite à partir du préfixe réservé à faithd : `2001:db8:4819:6:ffff::c009:c` (figure 2).

Les quatre derniers octets de l'adresse IPv6 sont les quatre octets de l'adresse IPv4.

Lorsque le poste client IPv6 ouvre une connexion vers cette adresse, ce flux est dirigé vers le démon *faithd* sur la passerelle disposant de la double-pile qui le traduit vers l'adresse IPv4 destination obtenue en ne conservant que les quatre octets de poids faible de l'adresse IP en question, et le flux de retour (envoyé vers l'adresse IPv4 de la passerelle) sera traduit dans l'autre sens et transmis au poste client.

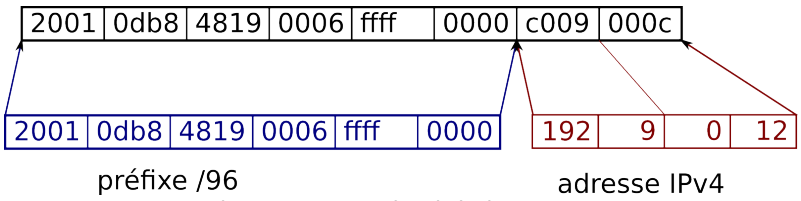

Figure 2: Construction de l'adresse IPv6

En pratique, *faithd* crée une interface réseau virtuelle `faith0` sur la passerelle et la table de routage dirige le trafic à destination du préfixe réservé vers cette interface. Le démon écoute cette interface virtuelle pour recevoir les paquets qu'il doit traduire. Pour chaque service géré, celui-ci écoute sur le port correspondant à son adresse IPv4 et réalise la traduction inverse pour le trafic dans le sens entrant.

Cette architecture est relativement simple à mettre en place sur un routeur OpenBSD, qui intègre le démon *faithd* et le support dans le noyau. Le serveur *totd* est disponible via le mécanisme des ports.

Avec cette configuration, une attention particulière doit être portée au contrôle d'accès vers l'interface virtuelle `faith0`. Le routeur *faithd* ne doit pas accepter de trafic à destination du préfixe réservé en provenance d'hôtes en dehors du réseau IPv6 pour lequel il fournit le service de traduction. Dans le cas contraire, ce service pourrait être utilisé pour contourner la politique de filtrage.

Le démon *faithd* dispose d'une fonctionnalité de filtrage permettant de limiter les adresses IPv4 accessibles via la traduction, que l'on configure dans le fichier `/etc/faithd.conf`.

En plus des risques de sécurité mentionnés plus haut, l'approche du RFC 3142 comporte d'autres inconvénients. En particulier le recours à un démon en mode utilisateur pour chaque service à traduire consomme des ressources et limite à la fois les performances et la généralité de la solution. Enfin, *faithd* est limité au protocole TCP.

C'est pourquoi l'approche proposée dans ce RFC est maintenant progressivement abandonnée. En particulier le projet OpenBSD envisage de retirer le support de *faithd* lors de ses prochaines versions.

5.2 NAT64

En reprenant le principe de traduction utilisé dans le RFC3142, le groupe de travail « *behave* » de l'IETF a proposé une approche similaire basée sur les mécanismes de traduction d'adresse (NAT) généralisés afin de réaliser en plus une traduction vers une famille d'adresses différente. Ce mécanisme est connu sous le nom de NAT64 (défini dans les RFC 6144 à 6146). En intégrant la ré-écriture des ports source, NAT64 permet de traiter l'ensemble du trafic IP (TCP, UDP et ICMP) de manière transparente et se prête à une implémentation directe dans le noyau.

Exactement comme pour le RFC3142, NAT64 a besoin d'un résolveur spécialisé qui traduit les adresses IPv4 en fonction du préfixe v6 utilisé par la traduction. Le mécanisme de génération de l'adresse IPv6 à partir de l'adresse IPv4 est décrit dans le RFC 6052. Et le mécanisme de résolution qui était implémenté dans *totd* est maintenant connu sous le nom de DNS64 (et décrit dans le RFC 6147) et, en plus de totd, qui reste tout à fait utilisable avec NAT64, d'autres implémentations sont apparues pour deux logiciels assurant la fonction de serveur DNS récursif ou cache : *bind* (à partir de la version 9.8.0 [9]) et sous forme d'une modification pour *unbound*[3].

Le cheminement d'une connexion utilisant DNS64 et NAT64 vers un site IPv4 est résumé sur la figure 3.

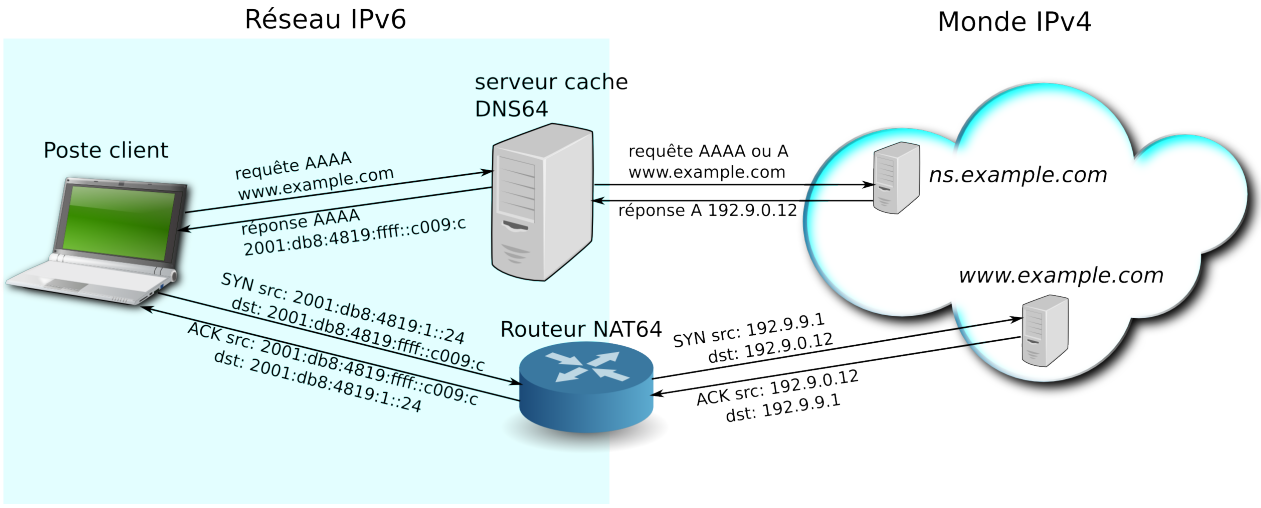

Figure 3: Principe de fonctionnement de NAT64/DNS64

Il existe plusieurs implémentations libres de NAT64 pour différents systèmes d'exploitation. En particulier le groupe canadien Viagenie propose des implémentations pour Linux et BSD[4]. Sous Linux il s'agit d'un module pour le noyau qui utilise *netfilter* pour la réécriture des paquets.

Tayga est une implémentation de NAT64 en mode utilisateur[5], qui se base sur les interfaces virtuelles TUN pour détourner les paquets à traduire vers un démon. On se retrouve alors dans une situation similaire à celle décrite plus haut avec *faithd*, la seule différence étant que NAT64 permet de traduire tous les protocoles IP.

5.2.1 NAT64 sur OpenBSD

Dans le cas d'OpenBSD, Viagenie a proposé un ensemble de modifications qui étendent le filtre de paquets PF pour réaliser NAT64. La configuration est intégrée à la configuration de PF par une extension de la syntaxe des règles de ré-écriture qui permet de décrire la traduction des adresses entre familles.

Un premier test a été réalisé à l'aide d'un live-CD d'OpenBSD 4.5 diffusé par Viagenie. Dans la version 4.6 d'OpenBSD, PF a été très largement ré-écrit et, la syntaxe et le code du mécanisme de traduction d'adresse ont profondément changé. Une version

[3] http://unbound.net/

[4] http://ecdysis.viagenie.ca/

[5] http://www.litech.org/tayga/

officielle de NAT64 pour OpenBSD, basée sur le code initial de Viagenie est en cours de développement et de test. Celle-ci a été utilisée lors des tests présentés ici.

Afin d'utiliser un résolveur DNS plus général que *totd*, c'est le serveur DNS cache *unbound*, modifié pour supporter DNS64 qui a été utilisé. Au moment où cet article est écrit, le projet unbound n'a pas intégré officiellement cette contribution, mais il semblerait que ce soit imminent.

La configuration de PF pour réaliser NAT64 se résume à une ligne dans le fichier `pf.conf` : si `int_if` désigne l'interface réseau interne (IPv6 seulement) et `ext_if` désigne l'interface externe disposant de la double pile on écrira :

```
pass in on $int_if inet6 to 2001:db8:4819:ffff::/96 af-to inet from ($ext_if)
```

Le trafic IPv6 entrant dans la passerelle à destination d'une adresse du préfixe réservé à la traduction d'adresses est réécrit (« `af-to` ») en utilisant comme adresse IPv4 source celle de l'interface externe (« `$(ext_if)` »). La traduction inverse pour le trafic de retour se fait automatiquement via l'état créé dans la table d'états de PF par le premier paquet sortant.

La configuration du serveur cache DNS *unbound* se fait également en une seule ligne supplémentaire, décrivant le préfixe IPv6 à utiliser pour répondre aux requêtes de type AAAA lorsqu'il n'existe pas de tel enregistrement (parce que le site en question n'est connecté qu'en IPv4).

```
dns64 prefix 2001:db8:4819:ffff::/96
```

Dans la maquette de test, il suffit donc de spécifier comme serveur DNS l'adresse IPv6 du serveur *unbound* configuré pour DNS64 et de diffuser l'adresse de la passerelle NAT64 comme routeur par défaut pour permettre à ces clients de contacter n'importe quel service IPv4.

Du point de vue de la sécurité, l'intégration avec PF permet de créer facilement une politique de filtrage directement au niveau de la passerelle de traduction d'adresses.

Mais il faut noter que, contrairement au RFC 3142, une passerelle NAT64 basée sur PF ne crée pas de risque nouveau sur un réseau existant. Dans l'expérience décrite ici, la politique de sécurité gérée sur le pare-feu du laboratoire est entièrement suffisante. Les flux IPv6 natifs sont filtrés comme pour n'importe quel autre nœud du réseau, tandis que les flux traduits en IPv4 sont vus (et traités comme tels) par la politique de filtrage IPv4. Le préfixe /96 utilisé par le mécanisme de traduction n'a dans ce cadre aucune raison de sortir du site, il peut donc rester bloqué sur le pare-feu au même titre que les autres préfixes IPv6 non utilisés.

Enfin, parce que PF oblige l'utilisateur à positionner la règle de traduction sur une règle appliquée en entrée sur une interface réseau, il n'est pas possible d'abuser le mécanisme de traduction pour ouvrir à partir d'une machine extérieure un flux vers une adresse IPv4 non autorisée.

5.2.2 Limitations

Comme toutes les solutions à base de traduction d'adresse, NAT64 ne fonctionne pas correctement avec les protocoles qui transportent les adresses IP des points d'extrémités dans la charge utile des paquets ou qui nécessitent que le client accepte une seconde connexion sur un port négocié dynamiquement.

Il serait nécessaire d'adapter pour NAT64 la notion de *proxy* existant dans le monde NAT44. Le RFC 6144 définit pour cela les *Application Layer Gateways (ALG).* En effet aujourd'hui un site choisissant de traiter le manque d'adresses IPv4 par une solution de type NAT44 (traduction d'une adresse IPv4 vers une autre adresse IPv4) trouvera tous les outils nécessaires pour traiter les protocoles problématiques les plus courants (FTP, H323, SIP, etc) en IPv4 alors que celui qui choisira une solution basée sur le passage à IPv6 restera bloqué tant que les serveurs concernés ne disposent pas eux-mêmes d'une connexion IPv6 ou qu'il n'y a pas de passerelle applicative adaptée pour NAT64.

Par ailleurs, comme chaque fois qu'une passerelle réécrit les en-têtes des paquets l'utilisation du protocole AH avec IPSec est impossible.

Enfin, pour des raisons similaires, DNS64 n'est pas compatible avec la vérification de signatures DNSSEC.

6 Conclusion

Les avancées récentes de la standardisation d'IPv6 ont apporté des solutions à deux difficultés qui compliquaient la vie d'un administrateur système et réseau souhaitant déployer des postes clients IPv6 : les améliorations du processus de configuration automatique des adresses (DHCPv6, RFC 6106, protection de l'adresse MAC) et le couple NAT64/DNS64 pour l'accès au réseau IPv4 seul.

Ces mécanismes permettent maintenant d'exploiter un réseau IPv6 seul fonctionnant de manière satisfaisante. Mais il reste en pratique des obstacles à la transition vers des réseaux 100% IPv6 (« legacy free »). Les principaux obstacles restent le nombre d'équipements plus ou moins enfouis (commutateurs, points d'accès wifi, imprimantes,...) qui ne supportent pas IPv6 et l'absence de support v6 dans certains logiciels. Le cas de ces équipements peut être traité par un réseau IPv4 non routable vers l'extérieur, accessible via le mécanisme NAT64.

Par contre, l'absence de passerelles au niveau application pour le support des protocoles qui ne se comportent pas correctement en présence de traduction d'adresse IPv6 vers IPv4 plaide pour le maintien d'une double pile avec un mécanisme de NAT44. Il serait souhaitable d'apporter rapidement des solutions à cette problématique pour accélérer la transition vers des réseaux sans IPv4.

Bibliographie

[1] J. Arko et A. Keranen, *Experiences from an IPv6-Only Network,* Internet draft, avril 2011, http://tools.ietf.org/html/draft-arkko-ipv6-only-experience-03

[2] *IPv6 transition mechanisms*, Article Wikipedia, http://en.wikipedia.org/wiki/IPv6_transition_mechanisms.

[3] S. Bortzmeyer, *IPv6 en 2011, état, techniques et prévisions ou « comment réussir une transition heureuse »*, présentation à l'association GUILDE, Grenoble, avril 2011. http://www.bortzmeyer.org/files/transition-ipv6-guilde-SHOW.pdf

[4] Gisèle Cizault, *IPv6 Theorie et pratique*, O'Reilly, Paris, 2005. Version en ligne : http://livre.g6.asso.fr/

[5] D. Visser, *Office network transition to IPv6 only*, Blog personnel, septembre 2011, https://confluence.terena.org/display/~visser/Office+network+transition+to+IPv6+only

[6] S. Bortzmeyer, *RFC 6333: Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion,* blog personnel, août 2011, http://www.bortzmeyer.org/6333.html

[7] S. Bortzmeyer, *Un tunnel IPv6-in-v4 sur un tunnel GRE...*, Blog personnel, septembre 2011, http://www.bortzmeyer.org/tunnel-over-tunnel.html

[8] 6Net, *IPv6 Deployment Guide*, Javvin Technologies Inc., décembre 2008, ISBN 978-1602670051.

[9] C. Liu, *DNS and BIND on IPv6*, O'Reilly media Inc, Mai 2011, ISBN 978-1-4493-0519-2.