Initiation à IPv6 en pratique : IPv6, radvd et DHCPv6

Nous avons pu, lors de la dernière approche, relier une machine directement au réseau IPv6 et nous avons entamé une première approche dans l'adressage IPv6, plus particulièrement sur la partie réseau des adresses. Nous allons cette fois-ci aborder la configuration de la partie hôte.

Nous allons, dans une première partie, revenir sur l'adressage IPv6 afin d'en préciser certains aspects. Cela nous permettra de voir comment le protocole permet l'auto-configuration des hôtes IPv6 à partir de leurs adresses MAC. Nous verrons tout d'abord comment l'auto-configuration est utilisée en mode dit stateless, ou mode sans état, et ensuite, comment mettre en place une configuration en mode dit stateful, ou mode avec état, en utilisant DHCPv6.

Comme notre souci premier est de favoriser la pratique, nous continuerons à faire évoluer notre laboratoire afin de mettre en place un démon **radvd** d'auto-configuration des hôtes sur un réseau IPv6 et nous terminerons sur la mise en place d'un serveur DHCPv6 afin de voir ce que ce type de serveur peut apporter dans un réseau autoconfiguré.

1. L'adressage des hôtes sur IPv6

En IPv4, vous aviez quatre digits de huit bits chacun. L'adresse IP de chaque poste d'un réseau local était liée à une adresse MAC par l'intermédiaire du protocole ARP. Comme nous l'avons déjà vu, ce protocole a disparu et nous allons voir que l'adresse MAC est toujours utilisée, la suite ci-après.

Il s'agit ici de se concentrer sur la partie hôte de l'adressage, ou autrement dit, les 64 bits de poids faible des adresses. Il existe plusieurs types d'adresses IPv6, comme les adresses IPv4 dites compatibles IPv6 et notées :

ou en forme abrégée :

Les deniers 32 bits peuvent être notés en hexadécimal, mais généralement ils sont laissés avec la notation traditionnelle. Ce type d'adresse conçu au début pour la transition IPv4/IPv6 devient obsolète.

Il existe également les adresses IPv4 mappées quand un client IPv4 souhaite communiquer un service serveur double pile IPv4/IPv6 tout en restant compatible avec les adresses IPv6 si ce dernier n'offre qu'une socket Ipv6, comme le montre l'extrait ci-dessous :

tcp6 :::80 :::* LISTEN

Ici, nous n'avons qu'une seule socket tcp6 en écoute, or les clients IPv4 doivent également pouvoir passer leurs requêtes. Ce sont des adresses IPv4 mappées qui sont utilisées. Ces adresses sont calculées et notées :

Cet aspect sera revu plus en détail quand nous aborderons le chapitre de l'interopérabilité des applications client/serveur dans le jeu de la double pile de protocoles et plus particulièrement dans le fonctionnement des sockets des services serveurs.

Ces types d'adresses sont utilisés dans des cas particuliers. Ce qui nous intéresse surtout ici, c'est la configuration des hôtes IPv6. Conservons à l'esprit qu'il y a deux parties :

2. L'identifiant d'interface

Cette adresse peut être créée dynamiquement tout en étant unique dans un lien donné. Nous allons regarder essentiellement la création des identifiants d'interface dans le cas où ceux-ci sont codés à partir de l'adresse MAC de l'interface, sur 48 bits. L'identifiant MAC IEEE 802.2 doit être modifié pour être adapté à un format 64 bits ou EUI-64. Afin de constituer une adresse sur 64 bits à partir d'une adresse MAC, on insère 2 octets de valeur **0xFF** et **0xFE** au milieu des 48 bits **[EUI64]**. Une adresse MAC :

devient

Jusque-là, ce n'est pas très compliqué, mais (mal)heureusement, ce n'est pas tout. Les bits (6) **u** et (7) **g** sont significatifs.

```
0      0 0      1 1      2

|0      7 8      5 6      3|

+----+----+----+----+----+----+

|cccc|ccug|cccc|cccc|cccc|cccc|

+----+----+----+----+----+----+
```

L'état du bit **u** (universel) est mis à 1 afin d'indiquer si l'identifiant d'interface est universel (adresses IEEE MAC sur 48 bits, par exemple) ou unique. D'autres interfaces logiques, comme celles créées pour les tunnels (liaison point à point), n'ont pas d'adresses MAC, mais on peut en utiliser une générée aléatoirement ou manuellement pour créer l'identifiant de cette interface. En procédant ainsi, cette adresse peut ne pas être unique, au sens universel. Dans ce cas, cet identifiant est caractérisé comme local. Cela est réalisé en positionnant le bit **u** à 0 afin de permettre, assez simplement et sans risques, la création d'identifiants d'interfaces. Il n'empêche qu'un identifiant d'interface, local ou universel, doit être unique sur un lien.

Concrètement, une adresse MAC **00:00:96:52:d2:e3** transformée en identifiant d'interface **0000:96ff:fe52:d2e3** est valide pour une interface logique, mais devient **0200:96ff:fe52:d2e3**, le bit **u** étant passé à 1 pour une interface physique. Attention, dans les systèmes de virtualisation comme Netkit, où les adresses MAC ne correspondent pas à des interfaces physiques réelles, ce bit est tout simplement inversé, comme nous le verrons un peu plus loin.

Le bit **g** (groupe) vaut 1 si l'adresse appartient à un groupe (multicast), sinon il est mis à 0.

Bien, nous pouvons continuer. Techniquement, cela permet de penser que chaque nœud, sur un réseau IPv6, peut s'auto-configurer, tout au moins configurer sa partie hôte, à partir du moment où il connaît son préfixe réseau. En aparté, cet aspect d'auto-configuration sur adresse MAC des nœuds IPv6 a soulevé la question de la confidentialité et du respect de la vie privée. En effet, l'utilisation de l'adresse MAC d'une machine, pour peu qu'elle soit utilisée sur le lieu de travail, à la maison, sur les salons [...] peut laisser penser qu'il est assez facile de suivre quelqu'un. Surtout si ce procédé est étendu aux autres équipements. Nous en reparlons plus bas. Mais, du coup, d'autres procédés d'auto-configuration ont été envisagés afin de permettre de ne pas être pisté dans son quotidien (travail depuis l'ordinateur de la maison, loisir depuis l'ordinateur de travail, ...) **[RFC 4941]**.

3. Séquence récréation

Puisque nous sommes tous arrivés jusqu'ici, nous avons bien mérité un petit espace/temps (c'est quantique !) de détente. Nous allons jouer avec l'adressage de cette partie hôte des adresses IPv6. Ceux qui savent que 10 et 10 font 10, à gauche, les autres à droite.

3.1. Le groupe de gauche

Voici comment expérimenter le bit **u** et les adresses MAC sous **netkit**. Mettez-vous où vous voulez :

```
P:~# ifconfig | grep -Ei "link|inet6"
eth0      Link encap:Ethernet HWaddr 16:21:a0:b3:0a:49
          inet6 addr: fe80::1421:a0ff:feb3:a49/64 Scope:Link
```

On obtient bien une auto-configuration, cohérente avec ce qui a été dit plus haut. Le bit **u** est ici tout simplement inversé et transforme le 16 de l'adresse MAC en 14. Maintenant, regardez bien ce qui se passe, ici :

```
P:~# ifconfig eth0 hw ether 00:00:00:00:00:01
```

```
P:~# ifconfig | grep -Ei "link|inet6"
eth0      Link encap:Ethernet HWaddr 00:00:00:00:00:01
          inet6 addr: fe80::200:ff:fe00:1/64 Scope:Link
```

on a une adresse avec le bit **u** positionné à 1 et là :

```
P:~# ifconfig eth0 hw ether 02:00:00:00:00:01
P:~# ifconfig | grep -Ei "link|inet6"
eth0      Link encap:Ethernet HWaddr 02:00:00:00:00:01
          inet6 addr: fe80::ff:fe00:1/64 Scope:Link
```

le bit **u** est positionné à 0. On constate qu'il est assez simple de créer des adresses IPv6 avec un identifiant d'interface à état local. Vous pouvez récupérer la base OUI **[OUI]** (Organizationally Unique Identifier) si vous voulez savoir à quel organisme a été attribué tel préfixe MAC. Pour le groupe de droite, on se retrouve à la fin de l'article. Maintenant que nous savons comment se compose un identifiant d'interface en mode auto-configuration, voyons ce qui caractérise les modes dits stateless et stateful.

4. Configuration et auto-configuration

Les interfaces peuvent être configurées (disons manuellement pour l'instant) ou s'auto-configurer. Voyons de quoi il retourne.

4.1. Le mode de configuration stateless

Dans le cas de l'auto-configuration sans état (stateless) **[RFC2462]**, où seul le préfixe est donné, l'équipement aura la charge de générer le suffixe de l'adresse. Dès qu'une interface est activée (démarrage de la machine, par exemple), une adresse de type lien local est automatiquement générée à partir de l'adresse MAC de l'interface. Par exemple, une carte réseau d'adresse MAC **00:0D:61:22:34:76** aura l'adresse IPv6 **fe80::20d:61ff:fe22:3476**.

Le préfixe est **fe80::**, l'adresse est donc de type lien local. Le suffixe est généré à partir de l'adresse MAC de la machine. C'est ce type d'adresse qui est généré sur votre machine si vous avez le module IPv6, comme c'est le cas sous Linux en version 2.6.

Le démon **radvd** (Router ADVertisement Daemon) permet d'auto-configurer toutes les interfaces du réseau avec un autre préfixe de réseau (par exemple, celui qui vous aura été fourni par votre FAI, celui que vous aurez choisi pour vos tests ou encore votre préfixe ULA) en utilisant un routeur IPv6 sur lequel **radvd** est installé. Cela doit permettre de configurer les adresses des nœuds de type lien global (Scope Global).

Ce démon doit être installé sur un des postes de votre réseau. Nous aurons l'occasion de le configurer et de l'exploiter lors d'un atelier, mais schématiquement cela ressemble à un scénario comme celui-ci : **radvd** est utilisé sur les systèmes GNU/Linux pour communiquer des informations aux clients quand ils doivent être auto-configurés. Le démon envoie régulièrement sur un réseau local Ethernet des messages classés RA - Router Advertisement. Il est également habilité à répondre à des requêtes de type RS - Router Solicitation. Ce processus fait partie du protocole ND (Neighbor Discovery), ou protocole de découverte des voisins **[RFC2461]**. Vous trouverez un exemple un peu plus loin.

Le client utilise son adresse de type lien local en **fe80::**. Ce processus est utilisé en cas d'absence de serveur DHCP ou en cas de configuration non personnalisée (configuration manuelle).

4.2. Le mode de configuration stateful

L'auto-configuration avec état sera utilisée avec DHCPv6 afin de voir comment attribuer des options au client comme les adresses des serveurs de noms, adresses des serveurs SIP, réserver des adresses IP particulières, ... Passons à l'application.

5. Les mains dans le cambouis : radvd avec Netkit

Note

Cher lecteur, si tu te demandes à quel laboratoire les auteurs font référence, sache que tu es passé dans une warpzone temporelle et qu'on t'a volé un mois de ta vie ! Cours vite (enfin, tourne...) à la page anciens numéros pour commander le numéro précédent. En auteur bienveillant, le lecteur trouvera également l'ensemble de la configuration du laboratoire sur le site des auteurs **[FL]**.

Nous avons déjà trois machines dans notre lab : R4, R et C461. Cette partie de l'application va porter uniquement sur le segment ip46. Afin de mettre en place les protocoles, nous allons procéder de la façon suivante :

- ajout des machines C462, S (le futur serveur de noms) et DH6 (le futur serveur DHCPv6) ;

- configuration en IP statique de S et DH6, qui prendront respectivement les adresses **2001:db8:46::1** et **2001:db8:46::2** ;

- configuration de **radvd** sur R ;

- vérification du fonctionnement de **radvd** et de l'auto-configuration des clients C461 et C462. À la fin, toutes les machines devront pouvoir se joindre sur leurs adresses IPv6 de type Global Scope ;

- modification de la configuration des clients avec le service DHCPv6.

5.1. Adaptation de la maquette

Le fichier **lab.conf** devient :

machines="R R4 C461 C462 S DH6"

R[0]=tap,10.0.0.1,10.0.0.2

On a ajouté les trois machines. Nous avons également gonflé la mémoire à DH6, qui en aura besoin. Vous pouvez lancer le lab et configurer les interfaces des machines R, S et DH6.

R:~# /sbin/ifconfig eth3 up

R:~# /sbin/ifconfig eth3 inet6 add 2001:db8:46:0::ff/64

DH6:~# /sbin/ifconfig eth0 up

DH6:~# /sbin/ifconfig eth0 inet6 add 2001:db8:46:0::2/64

S:~# /sbin/ifconfig eth0 up

S:~# /sbin/ifconfig eth0 inet6 add 2001:db8:46:0::1/64

- Bien sûr, tout cela est automatisable lors du lancement du lab. Référez-vous à la documentation de Netkit pour cela.

- Ces trois machines doivent pouvoir se joindre avec la commande **ping6** et la machine R doit pouvoir accéder à Internet afin de pouvoir installer des paquets. Si ce n'est pas le cas, il faut configurer R. Cela se résume à définir un serveur de noms dans le fichier **resolv.conf** de R. Celui de votre machine hôte doit fonctionner. La commande **aptitude update** sur R doit fonctionner.

- Les clients C461 et C462 ne doivent pas avoir leur interface eth0 activée.

5.2. RADVD

Le paquet n'étant pas installé sur R, nous allons commencer par cela.

Attention, si vous n'êtes pas sur un système Debian, adaptez...

C'est terminé. **radvd** ne démarrera pas car il n'a pas de fichier de configuration, mais nous allons régler cela rapidement.

5.2.1. Mise en place de l'auto-configuration sur un segment

Un routeur d'avertissement peut servir pour plusieurs segments et distribuer des préfixes différents. Ici, nous allons configurer l'interface pour servir le segment ip46. Cela se passe dans le fichier **/etc/radvd.conf** de R.

Déclarer dans le fichier les directives suivantes et enregistrer

AdvSendAdvert on; # Le routeur envoie les avertissements

prefix 2001:db8:46::/64 {

AdvRouterAddr on; # par défaut à off

Voir les options dans man radvd.conf

On ne met rien d'autre pour l'instant

La configuration est réduite au strict minimum pour nos besoins. Les pages de manuels de **radvd** et **radvd.conf** contiennent toutes les informations nécessaires. Nous pouvons activer **radvd** et nous allons en profiter pour lancer un analyseur de trames :

R:~# echo 1 > /proc/sys/net/ipv6/conf/all/forwarding

R:~# /etc/init.d/radvd start

Sans attendra trop longtemps, on commence à voir des trames émises :

20:45:34.303512 IP6 fe80::b8da:b9ff:fefd:8ec6 > ff02::1: ICMP6, router advertisement, length 56

Le moment est venu de voir comment se comporte un client. Nous regarderons, par exemple, C461. Il suffit d'activer l'interface **eth0** :

C461:~# ip link set eth0 up

Immédiatement, le dialogue ICMP6 est capturé sur R. Nous voyons deux requêtes passer. Un **RS** router solicitation et un **RA** router advertisement.

IP6 fe80::10de:89ff:fe19:21dd > ff02::2: ICMP6, router solicitation, length 16

IP6 fe80::3047:cbff:fe52:16e2 > ff02::1: ICMP6, router advertisement, length 56

L'adresse **ff02::2** correspond à **ip6-allrouters** et le client a bien pris une

auto-configuration sur le préfixe **2001:db8:46:** en utilisant son adresse MAC :

C461:~# ifconfig | grep inet6

inet6 addr: 2001:db8:46:0:10de:89ff:fe19:21dd/64 Scope:Global

inet6 addr: fe80::10de:89ff:fe19:21dd/64 Scope:Link

inet6 addr: ::1/128 Scope:Host

Toutes les machines du segment ip46 sont désormais joignables sur leurs adresses **2001:db8:46::/64**. Allons voir maintenant le client C461 :

C461:~# ip -6 route list dev eth0

2001:db8:46::/64 proto kernel (...)

fe80::/64 metric 256 (...)

default via fe80::3047:cbff:fe52:16e2 (...)

C'est la dernière qui nous intéresse. La passerelle par défaut est l'adresse unicast du routeur **radvd**.

5.2.2. Auto-configuration sur plusieurs segments

Il suffit d'ajouter un paragraphe dans le fichier **radvd.conf** pour les autres segments. Par exemple :

AdvSendAdvert on; # Le routeur envoie les avertissements

prefix 2001:db8:60::/64 {

AdvRouterAddr on; # par défaut à off

Voir les options dans man radvd.conf

On ne met rien d'autre pour l'instant

5.3. Conclusion sur radvd

Ce qui suit relève plus, pour le moment, de la réflexion car il n'y a pas d'éléments de réponse concrets apportés. Seule l'expérience et les usages pourront éclairer sur les meilleures façons de procéder ou les conduites à tenir, d'autant qu'il est difficile de généraliser. Un petit réseau de PME ne pose pas du tout les mêmes problèmes qu'un réseau de plusieurs milliers ou dizaines de milliers de machines. Le mode de configuration stateless des clients que nous venons de voir sera à rapprocher de la configuration avec un serveur DHCPv6 que nous allons découvrir. Tel que nous l'avons utilisé, **radvd** convient très bien pour l'annonce des routeurs, du préfixe et pour l'auto-configuration sans état des clients, mais comment faire si l'on souhaite passer d'autres paramètres aux clients, comme l'adresse des serveurs de noms, le nom du réseau, ... Par ailleurs, **radvd**, s'il permet une configuration simplifiée des clients, ne va pas sans poser quelques petits problèmes.

Le premier problème concerne l'anonymat, comme nous l'avons déjà cité. Si l'adresse est construite à partir de l'adresse MAC, il n'y a rien de plus simple que de suivre à la trace une personne, l'adresse étant liée au poste que cette personne utilise. Ce risque à été soulevé assez tôt lors de la création du protocole pour qu'un autre mode de configuration, dit mode stateful, soit proposé. Il n'y a pas de réponse binaire que l'on puisse apporter, pensez simplement aux implications pour vos collègues et le respect de la vie privée imposé par la loi.

Le second problème concerne la configuration des serveurs de noms. Nous aborderons le service DNS, mais personne n'est sans savoir que chaque adresse doit être déclarée. Comment faire si les adresses des hôtes sont construites à partir des adresses MAC ? Répertorier les adresses MAC à travers la gestion de parc et faire vivre cette base intelligemment a déjà donné son heure de gloire aux laboratoires qui produisent de l'aspirine. Les enregistrer serait une solution, mais peu simple à tenir dans le temps. À chaque remplacement d'interface, il faut penser à modifier les DNS sans commettre d'erreurs. Par ailleurs, si on considère que la durée de vie d'un

poste est de 4 à 5 ans, cela revient (à la louche à tartiflette, nous en convenons) à considérer que 20 à 25 % d'un parc est renouvelé chaque année. Sur un parc de plusieurs milliers de postes, la tenue à jour des DNS devient potentiellement une corvée (un débat politique peut naître pour savoir si IPv6 sera créateur d'emplois ;)). Reste la mise en place d'un DNS dynamique. Cela est effectivement envisageable s'il fonctionne avec DHCPv6, mais là encore, au moins pendant un temps, il y aura sûrement des solutions à développer car, comme nous le verrons, le passage IPv4/IPv6 ne se fera pas du jour au lendemain et une cohabitation des deux piles de protocoles (double stack) sera quasi obligatoire pendant longtemps, ce qui ne simplifiera pas la problématique de la tenue des services DNS et des applications.

6. Expérimentation avec le protocole DHCPv6 et Netkit

Qu'en est-il du protocole DHCP, largement répandu sur les réseaux ? En a-t-on besoin, puisque les adresses IPv6 peuvent s'auto-configurer à partir d'un préfixe et de leur propre adresse MAC ? Les avis sont partagés mais nous pensons que oui, pour plusieurs raisons :

- Le service DHCP va toujours permettre de distribuer des informations aux clients (serveur de noms, nom du domaine, adresse d'un serveur SIP, ...).

- Il peut prendre en charge l'inscription des clients auprès des serveurs de noms.

Regardons comment mettre en place un service DHCP IPv6 et étudions deux scénarios de configuration des clients. Dans le premier scénario, les clients seront dits en mode stateless, c'est-à-dire que leur configuration sera réalisée :

- à partir d'un préfixe distribué par un routeur d'annonce de préfixe pour la partie réseau ;

- à partir de l'adresse MAC de l'interface pour la partie client.

Le serveur DHCP distribuera les adresses des DNS.

Dans le second scénario, la configuration des clients sera réalisée en mode stateful ; dans ce cas, le serveur DHCP distribuera les adresses à partir d'une plage, comme cela se passe avec IPv4.

La version du service DHCP que nous connaissons bien pour IPv4 est la version 3. L'Internet Systems Consortium **[ISC]** développe la version 4, qui est compatible avec IPv6, nous utiliserons celle-ci.

Avant de commencer, nous allons récupérer et compiler les programmes car ils ne sont pas dans les dépôts. Il faut télécharger sur le site de l'ISC les sources de DHCP **[DHCP]**. Pour nous, cela donne :

```
cd ~/bin && mkdir dhcp4 && cd dhcp4

wget http://ftp.isc.org/isc/dhcp/dhcp-4.2.0-P2.tar.gz

tar xzf dhcp-4.2.0-P2.tar.gz
```

Une fois la compilation terminée, on met de côté les binaires **client/dhclient**, **server/dhcpd**, **relay/dhcrelay**, ce dernier servant à installer un proxy DHCP et le script **client/scripts/linux** qui est utilisé par **dhclient**. Nous copions tout cela dans le répertoire où est installé le lab (là où est votre fichier **lab.conf**), les binaires seront accessibles avec la commande **/hostlab/dhcpd**, **/hostlab/dhclient**... Maintenant que nous avons tout, passons à la pratique.

Nous utiliserons le même lab Netkit que nous avons créé lors de la mise en place du démon **radvd**, bien que deux machines suffisent en théorie, mais le lab, facilement extensible, vous permettra d'expérimenter également l'agent relais **dhcrelay**, qui fonctionne exactement sur le même principe que dans sa version v4.

Vous pouvez lancer le lab utilisé pour **radvd**.

6.1. Premier scénario - configuration en mode stateless des hôtes

La première chose à faire est de configurer et lancer **radvd** sur R, comme nous l'avons vu plus haut.

6.1.1. Configuration du routeur d'avertissement

Déclarez un serveur de noms dans le **resolv.conf** de R et installez **radvd**. Le démon ne se lance pas car il n'y a pas encore de configuration, mais ça va venir. Éditez le fichier **/etc/radvd.conf** pour déclarer le préfixe et lui demander de l'annoncer aux clients :

```
AdvSendAdvert on;

prefix 2001:db8:46::/64 {

AdvRouterAddr on;
```

et lancez le service : **/etc/init.d/radvd start**. Les logs, situés dans **/var/syslog**, doivent bien indiquer :

```
Dec 19 22:32:31 R radvd[688]: version 1.6 started
```

À ce stade, les clients C461 et C462 doivent déjà pouvoir obtenir des adresses IPv6 valides. Il reste à configurer le serveur DHCP pour qu'il distribue d'autres paramètres aux clients. Cela va se passer sur la machine DH6.

6.1.2. Configuration du serveur DHCPv6

Il nous faut créer un fichier de configuration pour **dhcpd** :

```
DH6:~# vi /etc/dhcp4/dhcpd.conf

option dhcp6.domain-search "formation-libre.fr";

option dhcp6.name-servers 2001:db8:46::1; 2001:db8:46::2;

# On laisse une plage vide

subnet6 2001:db8:46::/64 { }
```

Là, par contre, il va falloir faire bien attention. On active un serveur DHCP pour IPv6, et non pas pour IPv4. Qu'indiquent les options de la ligne de commandes ?

R:~# /hostlab/dhcpd --help

Usage: dhcpd [-p <UDP port #>] [-f] [-d] [-q] [-t|-T]

[-4|-6] [-cf config-file] [-lf lease-file]

[-tf trace-output-file]

[-play trace-input-file]

[-pf pid-file] [-s server] [ifo [...ifN]]

Voilà le problème. Les options **-4** et **-6** montrent qu'il n'est possible d'activer le service que pour IPv4 ou exclusivement IPv6. Il faut donc choisir. On peut donc en déduire que si nous avions des clients IPv4 et IPv6, il nous faudrait deux serveurs ou, tout au moins, deux instances du démon utilisant des ports différents. Ce qui nous intéresse ici, ce sont les options **-6**, **-d** pour mettre le serveur en mode debug, **-cf** pour passer en paramètre notre fichier de configuration, et enfin, **-lf** pour le fichier de journalisation.

DH6:~# touch /etc/dhcp4/dhcpd6.leases

DH6:~# /hostlab/dhcpd -6 -d -cf /etc/dhcp4/dhcpd.conf -lf /etc/dhcp4/dhcpd6.leases

Internet Systems Consortium DHCP Server 4.2.0-P2

Copyright 2004-2010 Internet Systems Consortium.

For info, please visit https://www.isc.org/software/dhcp/

Wrote 0 leases to leases file.

Listening on Socket/5/eth0/2001:db8:46::/64

Sending on Socket/5/eth0/2001:db8:46::/64

Le mode debug va nous permettre de visualiser tout se qui se passe entre le client et le serveur. Maintenant, procédons au test sur le client C461 après avoir vérifié qu'il a déjà une configuration réalisée à partir de **radvd**...

6.1.3. Application sur le client

Deux notes avant de démarrer en phase de test :

- Pensez à vider les fichiers **dhcpd6.leases** s'il en existe.

- Vérifiez avec **ps** que sur le client, il n'y ait pas déjà de **dhclient** actif quand vous lancez la commande.

Voyons les options de la commande **dhclient** :

C461:~# /hostlab/dhclient --help

Usage: dhclient [-4|-6] [-SNTP1dvrx] [-nw] [-p] [-D LL|LLT] [-s server]

[-cf config-file] [-lf lease-file][-pf pid-file] [-e VAR=val]

[-sf script-file] [interface]

Ici, nous allons utiliser les options **-6** et **-lf**, que nous connaissons déjà, mais également :

- **-S** afin d'indiquer que le client ne souhaite pas d'adresse IP ;

- **-sf** pour passer en paramètre le script que pourra utiliser **dhclient**, pour nous **linux**.

On vide le fichier resolv.conf de P1

C461:~# > /etc/resolv.conf

C461:~# touch dhcpd6.leases

C461:~# /hostlab/dhclient -6 -S -lf ./dhcpd6.leases eth0 -sf /hostlab/linux

Et on voit le dialogue sur DH6 :

Information-request message from fe80::64ad:f9ff:fef3:d5ec (..)

Sending Reply to fe80::64ad:f9ff:fef3:d5ec port 546

Le client C461 a bien récupéré les informations concernant le domaine et les DNS, il suffit pour s'en assurer de consulter le fichier **resolv.conf** de C461 :

C461:~# cat /etc/resolv.conf

search formation-libre.fr.

nameserver 2001:db8:46::1

nameserver 2001:db8:46::2

Si tout fonctionne, il n'y a plus qu'à organiser tout cela un peu mieux en mettant les fichiers de configuration et les binaires dans l'arborescence **/usr/local**, par exemple, et coder le script pour **init.d**. Passons au deuxième scénario pour une configuration en mode stateful.

6.2. Deuxième scénario - configuration en mode stateful des hôtes

La configuration IP des hôtes ne sera pas réalisée par **radvd**, mais par le serveur DHCPv6. Le plus simple est d'arrêter le serveur **radvd** sur R ou de modifier la configuration de la façon suivante, afin que le préfixe ne soit plus utilisé pour l'auto-configuration. Relancez le serveur.

```
AdvSendAdvert on;

prefix 2001:1:1:1::/64 {

# Mettre cette ligne, le préfixe

# ne sera pas utilisé pour l'autoconfiguration
```

AdvAutonomous off;

6.3. Modification du serveur DHCP

Il faut allouer une plage d'adresses. Le fichier de configuration devient :

option dhcp6.domain-search "formation-libre.fr";

option dhcp6.name-servers 2001:db8:46::1, 2001:db8:46::2;

On laisse une plage vide

subnet6 2001:db8:46::/64 {

range6 2001:db8:46::10 2001:db8:46::20;

et on a plus qu'à relancer le serveur.

DH6:/etc/dhcp4# /hostlab/dhcpd -6 -d -cf /etc/dhcp4/dhcpd.conf -lf /etc/dhcp4/dhcpd6.leases

6.3.1. Application sur le client

Avant de commencer, une petite modification s'impose à cause du script **linux** utilisé par **dhclient**. Le script utilise **/sbin/ip**, or la commande **ip** est dans **/usr/sbin/ip**, donc la meilleure solution est encore de créer un petit lien symbolique :

C461:~# ln -s /usr/sbin/ip /sbin/ip

et là, on est tranquille.

Pour le client, la ligne de commandes ne contient plus l'option **-S**, mais il convient de faire un peu de ménage :

- vérifier qu'il n'y a pas de client **dhclient** actif ;

- vider le fichier **dhcpd6.leases** ;

- démonter et remonter l'interface.

Cela donne :

C461:~# ifconfig eth0 down && ifconfig eth0 up

C461:~# /hostlab/dhclient -6 -lf ./dhcpd6.leases eth0 -sf /hostlab/linux

Dans la console du serveur, on voit bien l'adresse retenue et distribuée :

Picking pool address 2001:db8:46::20

Et sur le client, l'adresse affectée :

P1:~# ifconfig eth0 | grep inet6

inet6 addr: 2001:db8:46::20/64 Scope:Global

inet6 addr: fe80::10de:89ff:fe19:21dd/64 Scope:Link

6.4. Et la réservation d'adresse IP sur adresse MAC (!pomme) ?

Cela fonctionne-t-il comme pour IPv4 ? Oui, à quelques petites différences près (pourquoi faire simple quand on peut faire compliqué ?). La réservation d'adresse sur DHCPv6 fonctionne extérieurement de façon un peu similaire, avec ce que nous connaissons sous IPv4, mais cela n'est pas réalisé à partir de l'adresse MAC du client mais de son DUID (DHCP Unique Identifier). Chaque serveur et chaque client disposent d'un DUID. Il y a 3 types de DUID, qui sont décrits dans la **[RFC3315]** :

- **duid-llt** composée de l'adresse du lien et un timestamp ;

- **duid-en** composée à partir de l'adresse MAC ;

- **duid-ll** composée sur l'adresse du lien.

Les identifiants sont générés lors de la première activation du serveur ou du client et lorsqu'on remplace une interface physique. Ils n'ont, en dehors de ces cas-là, pas vocation à être modifiés. Pour les obtenir, le plus simple est de les prendre dans le fichier de journalisation (les fichiers leases), soit

sur le client pour le client, soit sur le serveur... pour le serveur.

Voyons pour le client C461 comment on pourrait mettre cela en place.

C461:~# cat dhcpd6.leases | grep option

  option dhcp6.client-id 0:1:0:1:14:ae:72:28:12:de:89:19:21:dd;

  option dhcp6.server-id 0:1:0:1:14:ae:6e:dc:32:4f:35:d3:2a:49;

On adapte la configuration du serveur DHCP pour réserver une adresse MAC au client :

option dhcp6.domain-search "formation-libre.fr";

option dhcp6.name-servers 2001:1:1:1::3, 2001:1:1:1::4;

subnet6 2001:1:1:1::/64 {

range6 2001:1:1:1::10 2001:1:1:1::20;

    # Attention à ne pas vous tromper dans le DUID du client

    host-identifier option dhcp6.client-id 0:1:0:1:14:ae:72:28:12:de:89:19:21:dd;

    fixed-address6 2001:db8:46::0ffe;

Avant de relancer le serveur et le client, nettoyez les fichiers **dhcpd6.leases** mais sans enlever les lignes suivantes :

server-duid "\000\001\(...)";

default-duid "\000\001\(...)";

Relancez le serveur :

DH6:/etc/dhcp4# /hostlab/dhcpd -6 -d -cf /etc/dhcp4/dhcpd.conf -lf /etc/dhcp4/dhcpd6.leases

puis le client :

C461:~# /hostlab/dhclient -6 -lf ./dhcpd6.leases eth0 -sf /hostlab/linux

La machine obtient l'adresse qui lui est réservée et la configuration de son **resolv.conf** :

P1:~# ifconfig eth0 | grep inet6

eth0 Link encap:Ethernet HWaddr 12:de:89:19:21:dd

inet6 addr: fe80::10de:89ff:fe19:21dd/64 Scope:Link

inet6 addr: 2001:db8:46::ffe/64 Scope:Global

6.5. Conclusion sur DHCPv6

Nous avons réalisé une bonne entrée dans le service. Il resterait à voir aussi le fonctionnement du service proxy DHCP, mais il ne pose pas de difficulté car il se rapproche de celui d'IPv4. Ce qui reste à voir, surtout, ce sont les options propres à DHCP sur Ipv6, qui sont décrites dans les pages de manuel, mais il serait également intéressant d'expérimenter une cohabitation du service sur les deux piles de protocoles en relation avec un DDNS.

7. Récréation, suite...

7.1. Le groupe de droite

C'est un peu moins court, nous en convenons, mais joueur quand même. Nous avons vu que l'adressage basé sur l'adresse MAC pouvait, pour certains, poser question. D'autres solutions ont été proposées, comme envisager une adresse hôte basée sur un protocole NTP... Pouvons-nous expérimenter cela ? Oui, certes, il faut juste un peu reprendre le lab sur **radvd** et surtout la partie client. Relancez le lab, nous allons modifier le comportement des clients. Sur la machine C461, il faut lui demander de construire une adresse plus temporaire. Cela est réalisé de la façon suivante :

C461:~# echo 2 > /proc/sys/net/ipv6/conf/eth0/use_tempaddr

et ensuite on réactive l'interface :

C461:~# ifconfig eth0 down

eth0 Link encap:Ethernet HWaddr 02:00:00:00:00:01

inet6 addr: 2001:db8:46::ff:fe00:1/64 Scope:Global

inet6 addr: fe80::ff:fe00:1/64 Scope:Link

inet6 addr: 2001:db8:46:0:a49f:1703:3937:e0b5/64 Scope:Global

On constate que l'adresse initiale (Scope Global) est conservée, mais que l'hôte en obtient une seconde qui n'est pas liée à l'adresse MAC :

inet6 addr: 2001:db8:46:0:a49f:1703:3937:e0b5/64 Scope:Global

C'est ça qui est intéressant. La valeur 2 mise dans **use_tempaddr** indique que c'est cette adresse que le système doit privilégier, ce que nous pouvons vérifier. Un **ping6** vers l'adresse IP de C462 va confirmer si on lance préalablement un analyseur de trames :

C462:~# tcpdump -n -i eth0

2001:db8:46:0:a49f:1703:3937:e0b5 > ff02::1:ff00:1: ICMP6, neighbor

solicitation, who has 2001:db8:46:0:200:ff:fe00:1, length 32

2001:db8:46:0:200:ff:fe00:1 > 2001:db8:46:0:a49f:1703:3937:e0b5: ICMP6,

neighbor advertisement, tgt is 2001:db8:46:0:200:ff:fe00:1, length 32

2001:db8:46:0:a49f:1703:3937:e0b5 > 2001:db8:46:0:200:ff:fe00:1: ICMP6,

echo request, seq 1, length 64

IP6 2001:db8:46:0:200:ff:fe00:1 > 2001:db8:46:0:a49f:1703:3937:e0b5:

ICMP6, echo reply, seq 1, length 64

C'est bien l'adresse temporaire qui a été utilisée par C461.

Conclusion sur la séquence

Nous avons vu dans cette deuxième partie la configuration des clients avec ou sans état. Gardez le lab, car nous l'utiliserons la prochaine fois pour la configuration des serveurs de noms.

À l'heure de conclure cet article, l'un des deux auteurs est ballotté par le train qui l'emmène dans le centre de la France. Alors qu'il goûte son heure de retard qui lui permet une relecture amusée, il fait de l'IPv6 sur papier. Les événements actuels nous amènent d'ailleurs à changer nos mots de conclusion. Nous avons une pensée émue pour le pays du soleil levant, le Japon, où, comme vous le savez tous, une tragédie humaine est en train de se dérouler. Nous leur souhaitons le meilleur pour l'avenir et espérons que la reconstruction se fera au plus tôt pour l'ensemble du pays. Lors de notre prochaine rencontre, nous vous proposerons une découverte de la résolution de nom sur IPv6. D'ici là, bon routage !

Références

Pour toutes les RFC : RFC Editor - **http://www.rfc-editor.org**

[RFC2460] Internet Protocol, Version 6 (IPv6)- Specification

[RFC2461] Neighbor Discovery for IP Version 6 (IPv6)

[RFC2462] IPv6 Stateless Address Autoconfiguration

[RFC2463] Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6)

[RFC2464] Transmission of IPv6 Packets over Ethernet Networks

[RFC3306] Unicast-Prefix-based IPv6 Multicast Addresses

[RFC3315] Dynamic Host Configuration Protocol for IPv6 (DHCPv6)

[RFC3587] IPv6 Global Unicast Address Format

[RFC4191] Default Router Preferences and More-Specific Routes

[RFC4193] Unique Local IPv6 Unicast Addresse

[RFC4291] IP Version 6 Addressing Architecture

[RFC4941] Privacy Extensions for Stateless Address Autoconfiguration in IPv6

EUI64 - **http://standards.ieee.org/regauth/oui/tutorials/EUI64.html**

ISC - **http://www.isc.org/**

DHCP - **http://www.isc.org/software/dhcp**

OUI - **http://standards.ieee.org/regauth/oui/oui.txt**

FL - **http://www.formation-libre.fr/articles/**