# Autoconfiguration IPv6

**Philippe Latu**
philippe.latu(at)inetdoc.net

***https://www.inetdoc.net***

Résumé

À l'heure où les objets connectés à Internet se multiplient, il est probable que le service d'autoconfiguration IPv6 sans état soit enfin utilisé à grande échelle. Ce document propose un exemple de mise en œuvre de l'autoconfiguration entre un routeur et les hôtes de deux réseaux locaux distincts. Le but est d'utiliser au maximum le principe des échanges d'informations réseau stateless. En plus de l'attribution des des paramètres réseau usuels d'un hôte client, on ajoute le service multicast Domain Name System (mDNS) qui suit aussi le même principe et permet de contacter un hôte réseau par le nom qu'il annonce.

## Table des matières

1. Copyright et Licence ..... 1
2. Préambule : Stateful versus Stateless ..... 1
3. Présentation de la maquette ..... 4
4. Autoconfiguration côté routeur ..... 6
5. Autoconfiguration côté client ..... 8
6. Resolver DNS ..... 10
7. Multicast DNS ..... 11
8. Pour conclure ..... 13
Glossaire ..... 14

## 1. Copyright et Licence

```
Copyright (c) 2000,2024 Philippe Latu.
Permission is granted to copy, distribute and/or modify this document under the
terms of the GNU Free Documentation License, Version 1.3 or any later version
published by the Free Software Foundation; with no Invariant Sections, no
Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included
in the section entitled "GNU Free Documentation License".
```

```
Copyright (c) 2000,2024 Philippe Latu.
Permission est accordée de copier, distribuer et/ou modifier ce document selon
les termes de la Licence de Documentation Libre GNU (GNU Free Documentation
License), version 1.3 ou toute version ultérieure publiée par la Free Software
Foundation ; sans Sections Invariables ; sans Texte de Première de Couverture,
et sans Texte de Quatrième de Couverture. Une copie de la présente Licence est
incluse dans la section intitulée « Licence de Documentation Libre GNU ».
```

### Méta-information

Cet article est écrit avec DocBook XML sur un système Debian GNU/Linux. Il est disponible en version imprimable au format PDF : `autoconf-ipv6.pdf`.

## 2. Préambule : *Stateful* versus *Stateless*

Les deux termes stateful et stateless font partie du jargon réseau depuis de nombreuses années. Mais que viennent-ils donc faire ici ? Ils sont en fait si lourds de sens qu'ils méritent quelques éclaircissements quant à leur utilisation dans les communications réseau.

Stateful

> En première approximation, on peut assimiler le terme Stateful à un mode de communication avec un mécanisme de suivi d'état. Ce mécanisme est implanté sur un équipement qui assure la correspondance entre les flux sortants et entrants qui le traversent. La mémorisation des

adresses présentes dans les en-têtes des différents protocoles permet de construire des tables de correspondances des flux réseaux. De très nombreux services, historiquement liés au protocole IPv4, sont basés sur ces mécanismes de suivi d'état : attribution des paramètres réseaux d'un hôte (DHCP), traduction d'adresse (NAT), pare-feux, balance de charge, redirecteurs, service mandataire (proxy), etc.

Le défaut de ce mode de communication le plus souvent évoqué est relatif à la nécessité d'un point d'enregistrement unique pour tous les flux. On parle alors du fameux Single Point Of Failure ou SPOF. En effet, si l'équipement Stateful unique est défaillant ou si il subit un déni de service, tous les flux réseaux seront bloqués. Pour pallier à ce défaut, les fonctions des services Stateful ont été étendues : les pare-feux peuvent partager leurs tables de suivi d'état, les serveurs DHCP peuvent partager l'état des baux délivrés aux postes clients, etc.

Le propos de ce document est de remettre en question l'enregistrement de l'état des flux réseau. Dans un Internet qui contient plus d'objets connectés que de dispositifs manipulés directement par des humains, il devient illusoire de chercher à identifier, référencer et enregistrer les paramètres de toutes les interfaces de tous les hôtes raccordés au réseau. On peut voir cette mise en cause du mode de communication Stateful comme un retour aux origines de l'Internet ; un réseau à commutation de paquets dans lequel tout hôte est susceptible de communiquer avec n'importe quel autre hôte.

Stateless

Si les mécanismes de suivi d'état au niveau des couches transport, réseau et liaison sont remis en question, il sera toujours nécessaire de s'authentifier auprès d'un service pour identifier la source et la destination d'un flux d'informations. Dans les démonstrations ou les publicités sur les objets connectés, les capteurs transmettent leurs informations directement sur les réseaux sociaux. Avec ces exemples, l'identification et/ou l'authentification se fait au niveau application et non au niveau liaison avec l'enregistrement de l'adresse MAC de l'interface réseau dans un serveur DHCP.

Avec l'abandon de l'enregistrement des paramètres réseau des couches basses liées à la transmission de l'information, on autorise une plus grande dynamique dans les communications réseau. Le nombre des hôtes présents dans les domaines de diffusion peut varier librement et on déplace les fonctions d'enregistrement (identification, authentification, etc.) vers la couche application liée au traitement de l'information.

On peut légitimement se demander pourquoi l'autoconfiguration n'a pas connu plus de succès dans les années précédentes. En effet, le document RFC4862 IPv6 Stateless Address Autoconfiguration date de 2007. Quelques arguments peuvent être avancés sans tenir compte de la lenteur dans l'adoption du protocole IPv6.

Modèle périmétrique

Pendant très longtemps, la bonne pratique voulait que l'on cherche à découper les réseaux en périmètres fonctionnels. À l'intérieur d'un périmètre on est sensé avoir une connaissance exhaustive des flux et des hôtes en présence. Dans cette optique, les services Stateful ont été naturellement mis en avant. Ils sont le point de passage obligé qui garantit que seules les fonctions assignées au périmètre sont présentes.

Même si ce modèle a encore de beaux jours devant lui, il est sûr qu'il va falloir évoluer dans le mode de conception des architectures. Côté Parc, le nombre et la nature des hôtes croît sans cesse. Avec cette croissance, l'idée même de la liste exhaustive des hôtes autorisés s'envole. Même en connectant les services DNS et DHCP statiques à l'aide d'applications spécifiques, il devient difficile de suivre la dynamique du nombre d'hôtes. Côté infrastructure, la virtualisation apporte aussi une dynamique inconnue jusqu'alors. Les outils de gestion intégrée des hyperviseurs permettent l'ajout et la suppression d'instances de serveurs virtuels très rapidement. Là encore, les outils classiques d'enregistrement manuel des entrées ne suivent pas la cadence.

Face à ce manque de réactivité du réseau, la bonne pratique de conception a évolué et on parle de plus en plus souvent de «VLANs d'étage» montrant ainsi que l'étendue du domaine de diffusion prévaut sur le découpage fonctionnel. Cependant, l'enregistrement des paramètres d'interface réseau perdure même s'il est sérieusement bousculé par les modes du style Bring Your Own Device ou BYOD. Dans de nombreux contextes professionnels, le contrôleur de domaine constitue encore

le point d'enregistrement ultime, alors que dans le même temps, de plus en plus de flux transitent vers le Cloud. Ce fameux Cloud est aujourd'hui le principal «opposant» au modèle périmétrique.

Autoconfiguration incomplète

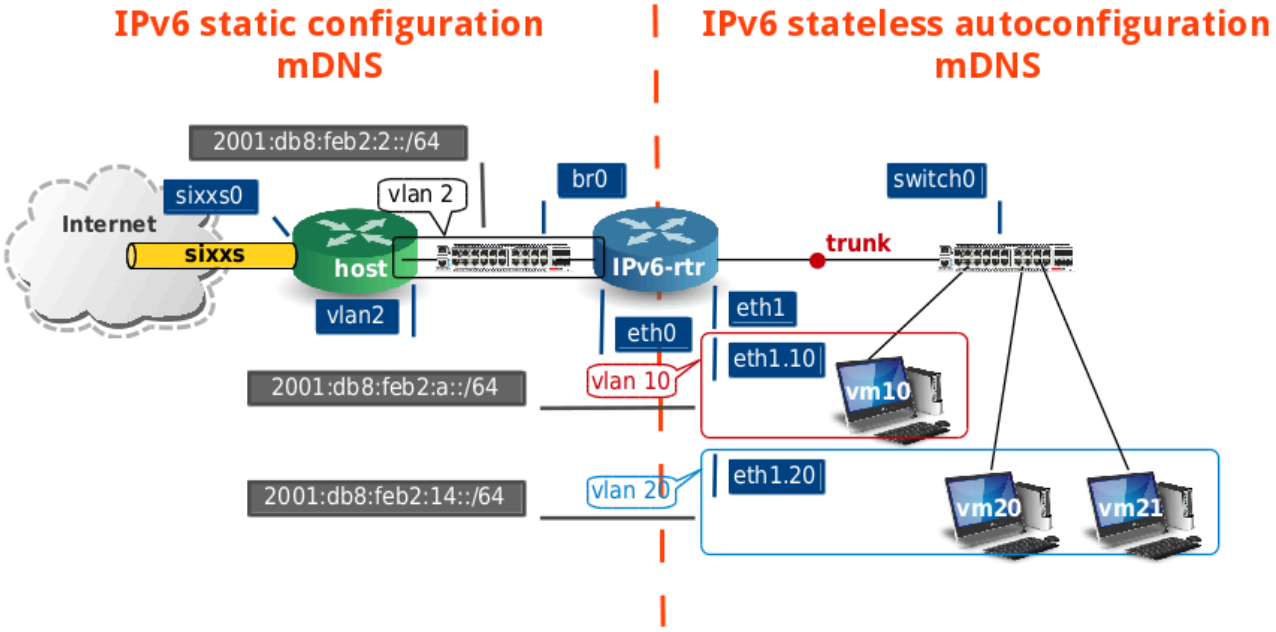
L'autoconfiguration, telle qu'elle est présentée dans le document RFC4862 ne couvre pas la même liste d'attributs qu'un service DHCP basique. L'absence de configuration du resolver DNS a cruellement fait défaut dans l'adoption de cette solution. En effet, tout dispositif raccordé à un réseau a nécessairement besoin d'utiliser des noms pour identifier les hôtes à contacter. L'autoconfiguration strictement limitée à la couche réseau s'est avérée bloquante dans les usages. Le fait qu'il faille ajouter un programme supplémentaire dans la couche application est un frein important dans la mesure où il génère un surcoût d'administration.

Un nouveau document, RFC6106 IPv6 Router Advertisement Options for DNS Configuration, est venu compléter la liste des attributs «propulsés» par le service d'autoconfiguration avec la désignation du ou des serveurs DNS. Il date de 2010 et son adoption n'a pas été aussi rapide qu'escompté. À titre d'exemple, il semble que la fonction ne soit disponible que dans la version XE du système IOS de Cisco™ au moment de la rédaction de ces lignes.

Les fonctions du document RFC6106 sont utilisées dans la maquette présentée dans les sections suivantes. Elles font clairement apparaître la nécessité d'une configuration aux deux extrémités en communication. Les mêmes échanges de paquets RA (Router Advertisements) sont utilisés pour passer les paramètres mais une application spécifique doit appliquer la configuration du resolver DNS.

## 3. Présentation de la maquette

Dans le but d'illustrer l'autoconfiguration IPv6, on doit chercher à reproduire un contexte réaliste. Ici, la maquette propose un routeur IPv6 qui dessert deux domaines de diffusion (VLANs) dans lesquels on trouve des postes clients qui reçoivent les annonces.




Accès Internet

L'accès au réseau public se fait via un tunnel fourni par le service SixXS ( IPv6 Deployment & Tunnel Broker). Ce service gratuit a pour but de promouvoir l'utilisation du protocole IPv6. Il constitue une opportunité fantastique pour apprendre et surtout pratiquer sur l'interconnexion de réseaux IPv6 avant de passer à l'exploitation réelle.

Dans l'exemple traité ici, les 32 premiers bits de tous les préfixes IPv6 ont été corrigés avec le préfixe `2001:db8` dédié à la documentation. En réalité, on utilise des réseaux appartenant à un préfixe `/48` gracieusement fourni par SixXS. Ainsi, les possibilités d'interconnexion et de configuration sont quasi infinies.

Le tunnel d'accès à l'Internet débouche sur un système hôte à partir duquel on virtualise une instance de routeur Debian GNU/Linux et des postes clients qui utilisent le même système. Les fonctions de commutation des trames Ethernet et de gestion des VLANs sont assurées par Open vSwitch. Bref, un classique du site inetdoc !

Routeur `IPv6-rtr`

Cette instance de machine virtuelle assure le routage statique entre le VLAN de raccordement au système hôte et les deux autres VLANs sur lesquels les hôtes utilisent l'autoconfiguration IPv6.

Bien sûr, la fonction de routage IPv6 doit être active sur ce système. Dans ce but un fichier `ipv6.conf` spécifique a été placé dans le répertoire `/etc/sysctl.d/`. Dans ce fichier, on trouve en premier l'activation de la fonction de routage du noyau puis la désactivation de l'autoconfiguration pour toutes les interfaces du système. Le routeur de la maquette présentée doit uniquement publier les informations d'autoconfiguration.

```
etu@IPv6-rtr:~$ cat /etc/sysctl.d/ipv6.conf
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~é
net.ipv6.conf.default.forwarding = 1
#
net.ipv6.conf.default.autoconf = 0
net.ipv6.conf.default.accept_ra = 0
net.ipv6.conf.default.accept_ra_defrtr = 0
net.ipv6.conf.default.accept_ra_rtr_pref = 0
net.ipv6.conf.default.accept_ra_pinfo = 0
net.ipv6.conf.default.accept_source_route = 0
net.ipv6.conf.default.accept_redirects = 0
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~é
net.ipv6.conf.all.forwarding = 1
#
net.ipv6.conf.all.autoconf = 0
net.ipv6.conf.all.accept_ra = 0
net.ipv6.conf.all.accept_ra_defrtr = 0
net.ipv6.conf.all.accept_ra_rtr_pref = 0
net.ipv6.conf.all.accept_ra_pinfo = 0
net.ipv6.conf.all.accept_source_route = 0
net.ipv6.conf.all.accept_redirects = 0
```

Voici une copie de la table de routage qui fait apparaître les 3 réseaux représentés sur le schéma.

```
etu@IPv6-rtr:~$ ip -6 route ls
2001:db8:feb2:2::/64 dev eth0  proto kernel  metric 256
2001:db8:feb2:a::/64 dev eth1.10  proto kernel  metric 256
2001:db8:feb2:14::/64 dev eth1.20  proto kernel  metric 256
fe80::/64 dev eth0  proto kernel  metric 256
fe80::/64 dev eth1  proto kernel  metric 256
fe80::/64 dev eth1.10  proto kernel  metric 256
fe80::/64 dev eth1.20  proto kernel  metric 256
default via 2001:db8:feb2:2::1 dev eth0  metric 1024
```

Le fichier de configuration principal des interfaces réseau se présente de la façon suivante. Les adresses des interfaces sont statiques et définies dans ce fichier.

```
etu@IPv6-rtr:~$ cat /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet6 static
        address 2001:db8:feb2:2::2/64
        gateway 2001:db8:feb2:2::1
        dns-nameservers 2001:db8:feb2:2::1 2001:4860:4860::8888❶

auto eth1
iface eth1 inet6 manual
        up echo 0 > /proc/sys/net/ipv6/conf/eth1/forwarding❷

auto eth1.10
iface eth1.10 inet6 static
        hwaddress de:ad:00:be:ef:0a❸
        address 2001:db8:feb2:a::1/64
        vlan-raw-device eth1

auto eth1.20
iface eth1.20 inet6 static
        hwaddress de:ad:00:be:ef:14❹
        address 2001:db8:feb2:14::1/64
        vlan-raw-device eth1
```

❶ La configuration du resolver DNS est directement fournie avec la configuration des interfaces. Les informations fournies avec la directive `dns-nameservers` sont exploitées par le paquet resolvconf. Le rôle de ce paquet, spécifique à la distribution Debian GNU/Linux, est d'arbitrer

les accès au fichier `/etc/resolv.conf`. Dans le cas présent, la seule source est le fichier `/etc/network/interfaces` alors que dans le cas d'un hôte mobile avec ou sans double pile IPv4 & IPv6, plusieurs programmes sont susceptibles de modifier la configuration du resolver. Le recours au paquet resolvconf est notamment nécessaire dans le cas des postes clients.

La deux adresses IPv6 données en paramètres correspondent au système hôte sur lequel le logiciel bind9 est exécuté et à un serveur DNS public de Google™.

❷ Le protocole IPv6 introduit une condition spécifique dans la configuration du routage inter-VLAN. Avec l'attribution automatique d'une adresse IPv6 de type lien local (préfixe `fe80::/10`), l'interface principale d'un trunk peut participer au processus de routage. La directive du fichier de configuration empêche le routage des trames sans étiquette de VLAN sur la base de l'adresse de lien local de l'interface `eth1`.

❸❹ De façon à distinguer les adresses de passerelles par défaut des deux VLANs, on fixe manuellement les adresses MAC des deux sous-interfaces. On peut très bien se passer de cette «manœuvre» sachant que la portée ou la visibilité d'une adresse MAC est limitée à un domaine de diffusion unique. On perd cependant en lisibilité puisque, sans ce changement d'adresse MAC, plusieurs VLANs utilisent la même adresse IPv6 de lien local comme passerelle par défaut. Ici, on utilise un jeu de mot hexadécimal très connu (dead beef) suivi du numéro de VLAN pour distinguer les interfaces du routeur qui servent de passerelle par défaut aux hôtes raccordés aux VLANs.

Une fois la partie routage statique en place, on peut passer à la présentation de l'autoconfiguration côté routeur puis côté client.

## 4. Autoconfiguration côté routeur

Le routeur est responsable de la publication des paramètres de configuration des hôtes clients raccordés aux deux VLANs de la maquette. Il existe plusieurs logiciels pour assurer cette fonction. Au moment de la rédaction de ces lignes, seul le paquet radvd supporte la publication des paramètres DNS définis dans le document RFC6106 IPv6 Router Advertisement Options for DNS Configuration. On a vu dans la Section 2, « Préambule : Stateful versus Stateless » que pour que l'autoconfiguration IPv6 soit une solution satisfaisante, il faut qu'elle couvre la même liste d'attributs de base qu'un service DHCP classique.

La configuration du paquet radvd se résume au fichier `/etc/radvd.conf` dont voici une copie. On identifie facilement les deux sections `interface` qui correspondent aux deux VLANs desservis par le routeur ainsi que les préfixes IPv6 associés.

```
etu@IPv6-rtr:~$ cat /etc/radvd.conf
interface eth1.10
{
   AdvSendAdvert on; ❶
   prefix 2001:db8:feb2:a::/64 ❷
   { };

   RDNSS 2001:db8:feb2:2::1 ❸
   { };

   RDNSS 2001:4860:4860::8888
   { };
};

interface eth1.20
{
   AdvSendAdvert on;
   prefix 2001:db8:feb2:14::/64
   { };

   RDNSS 2001:db8:feb2:2::1
   { };

   RDNSS 2001:4860:4860::8888
   { };
};
```

❶ L'option `AdvSendAdvert` doit être positionnée à la valeur `on` pour que les annonces soient publiées.

❷ La directive `prefix`, comme son nom l'indique, désigne le réseau IPv6 annoncé dans le domaine de diffusion. L'espace vide entre les accolades indique que l'on s'appuie sur les valeurs par défaut de la configuration de radvd.

❸ Les directives `RDNSS` désignent les adresses des serveurs DNS à annoncer aux postes clients. L'acronyme RDNSS correspond à Recursive DNS server. Ce sont ces attributs qui ont été introduits dans le document RFC6106.

Le document RFC6106 spécifie un autre attribut : DNSSL ou DNS Search List. Cet attribut sert à annoncer la liste des suffixes DNS à utiliser par défaut pour les requêtes utilisant des noms d'hôtes incomplets. Malheureusement, au moment de la rédaction de ces lignes l'attribut DNSSL n'est pas supporté côté client sur la distribution Debian GNU/Linux. Voir le rapport de bug rdnssd does not support DNSSL.

## 5. Autoconfiguration côté client

Le poste client doit être configuré pour exploiter correctement les annonces émises par le routeur dans un domaine de diffusion donné. On débute avec la configuration de base de l'interface réseau du client et on complète la liste des outils pour que le resolver DNS fonctionne correctement.

De façon classique, le fichier `/etc/network/interfaces` de l'hôte client se présente comme suit.

```
etu@vm21:~$ cat /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet dhcp
iface eth0 inet6 auto
```

On voit ici que l'interface Ethernet `eth0` a été configurée pour obtenir sa configuration «automatiquement». Le résultat de l'autoconfiguration IPv6 apparaît dans la liste des adresses et la table de routage du poste client.

### Liste des adresses de l'hôte vm21

```
etu@vm21:~$ ip addr ls
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether ba:ad:00:ca:fe:15 brd ff:ff:ff:ff:ff:ff
    inet6 2001:db8:feb2:14:b8ad:ff:feca:fe15/64 scope global dynamic
       valid_lft 85946sec preferred_lft 13946sec
    inet6 fe80::b8ad:ff:feca:fe15/64 scope link
       valid_lft forever preferred_lft forever
```

### Table de routage de l'hôte vm21

```
etu@vm21:~$ ip -6 route ls
2001:db8:feb2:14::/64 dev eth0  proto kernel  metric 256  expires 86141sec
fe80::/64 dev eth0  proto kernel  metric 256
default via fe80::dcad:ff:febe:ef14 dev eth0  proto ra  metric 1024  expires 1541sec
```

Les deux copies d'écran ci-dessus permettent de donner quelques explications sur la composition des adresses IPv6 avec l'autoconfiguration SLAAC. Si on se réfère au schéma de la maquette, on voit que le système `vm21` est un hôte du VLAN 20 dont le préfixe réseau est `2001:db8:feb2:14::/64` ; préfixe qui apparaît en première ligne de la table de routage.

Voyons comment les adresses de l'interface `eth0` de l'hôte `vm21` ont été composées. Une fois de plus, c'est l'adresse MAC qui constitue la clé de composition des autres adresses. Dans cet exemple, l'adresse `ba:ad:00:ca:fe:15` a été définie dans le script de lancement du système virtuel (Voir Virtualisation système et enseignement).

Il s'agit d'une adresse de type EUI48 comprenant 6 octets à partir de laquelle le système compose une adresse de type EUI64 avec 8 octets en insérant l'empreinte `ff:fe` au milieu. Ce n'est pas tout, le bit U/L de l'octet situé le plus à gauche passe de 1 à 0. Ainsi l'adresse MAC de départ `ba:ad:00:ca:fe:15` devient `b8:ad:ff:fe:ca:fe:15`. Pour plus de détail voir la section Types d'adresses MAC de l'article Routage Inter-VLAN.

À la suite de ce traitement sur l'adresse MAC, deux adresses IPv6 sont construites.

`fe80::b8ad:ff:feca:fe15/64`

> Cette adresse de type lien local utilise le préfixe `fe80::/10`. L'adresse MAC de type EUI64 est placée à droite (bits de poids faible) du préfixe de façon à composer une adresse sur 128 bits. Comme la

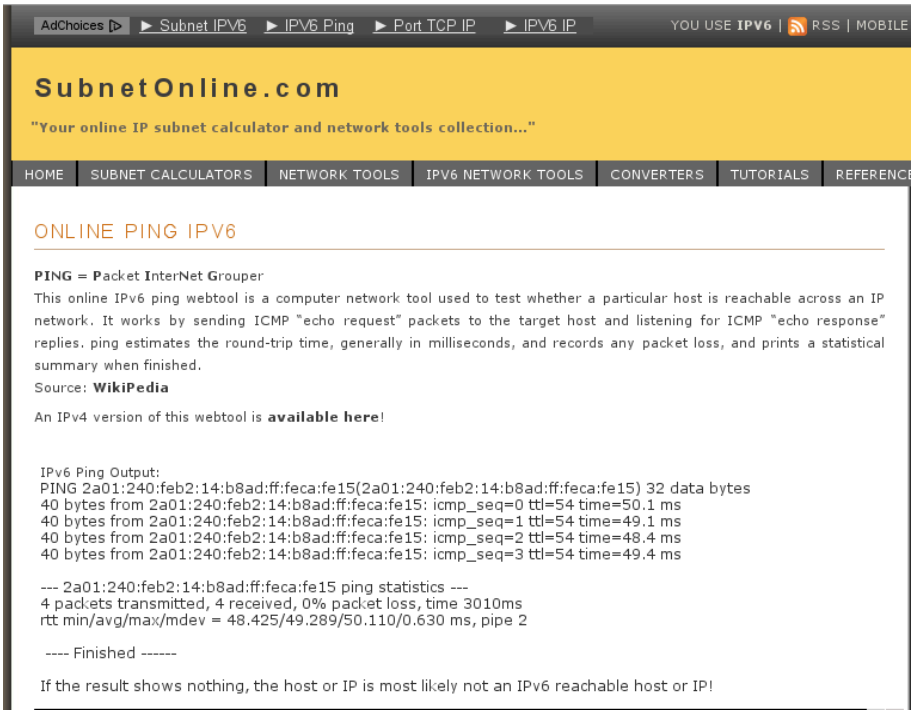
mention scope link de la copie d'écran l'indique, la portée de cette adresse est limitée au VLAN, ou encore au domaine de diffusion. Ces adresses sont très utiles pour joindre les hôtes voisins appartenant au même réseau. Par exemple, il est possible de contacter l'hôte `vm20` de la façon suivante.

```
etu@vm21:~$ ping6 -c 2 fe80::b8ad:ff:feca:fe14%eth0
PING fe80::b8ad:ff:feca:fe14%eth0(fe80::b8ad:ff:feca:fe14) 56 data bytes
64 bytes from fe80::b8ad:ff:feca:fe14: icmp_seq=1 ttl=64 time=1.65 ms
64 bytes from fe80::b8ad:ff:feca:fe14: icmp_seq=2 ttl=64 time=0.624 ms

--- fe80::b8ad:ff:feca:fe14%eth0 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.624/1.141/1.658/0.517 ms
```

`2001:db8:feb2:14:b8ad:ff:feca:fe15/64`

Cette adresse de type global utilise le préfixe `2001:db8:feb2:14::/64` annoncé dans les paquets RAs émis par le routeur. C'est le résultat du travail effectué par le paquet radvd présenté dans la Section 4, « Autoconfiguration côté routeur ». Comme dans le cas de l'adresse de type lien local, l'adresse MAC au format EUI64 est utilisée comme suffixe pour composer l'adresse IPv6 complète sur 128 bits. À la différence de l'adresse précédente, la mention global dynamic indique que cette adresse à une portée globale et qu'elle est utilisable depuis n'importe quel autre réseau. À titre d'illustration, voici une copie d'écran qui montre le résultat d'un ping IPv6 depuis le service en ligne subnetonline.com.




On remarque que l'adresse IPv6 testée utilise le préfixe SixXS et non le préfixe de documentation `2001:db8::/32`.

Côté table de routage, la seule particularité vient du fait que la passerelle par défaut est désignée par son adresse de type lien local et non par son adresse globale statique : `fe80::dcad:ff:febe:ef14`. C'est bien cette adresse de type lien local qui est annoncée par le service d'autoconfiguration dont la portée est justement limitée au domaine de diffusion. C'est à ce niveau qu'intervient le mécanisme de sélection d'adresse qui fait que les deux types d'adresses peuvent coexister. Ainsi, la table des voisins sur le réseau local peut donner les résultats suivants.

```
etu@vm21:~$ ip -6 neighbor ls
fe80::dcad:ff:febe:ef14 dev eth0 lladdr de:ad:00:be:ef:14 router STALE
2001:db8:feb2:14::1 dev eth0 lladdr de:ad:00:be:ef:14 router DELAY
fe80::b8ad:ff:feca:fe14 dev eth0 lladdr ba:ad:00:ca:fe:14 STALE
2001:db8:feb2:14:b8ad:ff:feca:fe14 dev eth0 lladdr ba:ad:00:ca:fe:14 STALE
```

Les deux premières entrées de la table ci-dessus correspondent bien au même hôte voisin : le routeur `IPv6-rtr`.

## 6. *Resolver* DNS

Même si cette section fait formellement partie de l'autoconfiguration du poste client, on lui réserve un traitement particulier. Comme indiqué dans la Section 2, « Préambule : Stateful versus Stateless », l'autoconfiguration IPv6, telle qu'elle est présentée dans la section précédente, se limite strictement à la couche réseau de la modélisation. Relativement à un service DHCP classique, il manque la désignation du serveur DNS. On se retrouve dans une situation assez paradoxale. Un service d'autoconfiguration limité à la couche réseau convient très bien aux équipements d'interconnexion réseau. Or, on peut dire que jusqu'à présent, ces équipements ont des besoins limités dans ce domaine. À l'opposé, les hôtes raccordés aux réseaux ont un grand besoin du service d'autoconfiguration et ne peuvent pas se passer de la configuration du resolver DNS. En effet, la moindre communication réseau utilise les noms de domaines sur un hôte et il est impossible d'initier cette communication si les programmes de la couche application ne disposent pas d'un serveur DNS à contacter.

Dans la Section 4, « Autoconfiguration côté routeur », le paquet radvd a été configuré pour annoncer deux adresses de serveur DNS. Il faut maintenant que ces annonces soient exploitées côté client. Cette opération implique l'utilisation de deux paquets.

resolvconf

Le paquet resolvconf n'est pas lié à priori à l'autoconfiguration IPv6. Son rôle est d'arbitrer les accès au fichier `/etc/resolv.conf`. Les entrées placées dans ce fichier sont utilisées par la bibliothèque standard dès qu'il est question d'une opération sur un nom d'hôte. On peut citer l'exemple de l'appel à `getaddrinfo()` présenté dans le document Initiation au développement C sur les sockets IPv4 & IPv6.

Historiquement, de nombreuses applications de configuration réseau avaient la possibilité d'écrire dans le fichier `/etc/resolv.conf` sans tenir compte de la «concurrence». Le script resolvconf est appelé dès qu'une interface réseau est activée ou désactivée. Il se charge alors d'éditer le contenu du fichier de configuration si besoin.

Dans le cas présent, resolvconf est là pour implanter les entrées de serveur DNS annoncées dans les paquets RAs.

rdnssd

Le démon `rdnssd` scrute les annonces RDNSS et les traduit dans son propre fichier `resolv.conf`. Ce fichier est ensuite traité par le script resolvconf de façon à fournir au système un fichier `/etc/resolv.conf`.

Dans le cas de la maquette, seul le protocole IPv6 est utilisé. Dans un cas de fonctionnement en double pile IPv4 et IPv6, le script resolvconf veille à conserver les entrées des deux protocoles pour respecter les choix des programmes de la couche application.

Voici le résultat produit lorsque les deux paquets sont installés sur le système.

```
etu@vm21:~$ cat /etc/resolv.conf
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 2001:db8:feb2:2::1
nameserver 2001:4860:4860::8888
```

On retrouve bien les deux adresses IPv6 définies dans le fichier de configuration du démon `radvd`.

On teste simplement l'utilisation de la bibliothèque standard à l'aide de la commande host du paquet bind9-host.

```
etu@vm21:~$ host www.nic.fr
www.nic.fr is an alias for web01.nic.fr.
web01.nic.fr has address 192.134.5.5
web01.nic.fr has IPv6 address 2001:67c:2218:30::5
```

## 7. Multicast DNS

Une fois que l'autoconfiguration IPv6 (SLAAC) est en place, les hôtes de la maquette sont capables de contacter n'importe quel autre hôte IPv6 sur l'Internet à partir de son nom enregistré dans le service DNS. Qu'en est-il des hôtes voisins qui n'ont pas été enregistrés ? Est-il même nécessaire de les enregistrer ?

Pour revenir au préambule de ce document, l'objectif est d'aller au maximum des possibilités de configuration sans enregistrement d'état. C'est à ce niveau qu'intervient le service décrit dans le (long) document RFC6762 Multicast DNS. Il s'agit de mettre en œuvre un mécanisme de résolution des noms d'hôtes en l'absence d'infrastructure. Pour simplifier à l'extrême, le principe de fonctionnement du service mDNS rassemble les points suivants :

- Le format des messages, ainsi que le format des enregistrements, reste inchangé par rapport au service DNS classique.
- Le numéro de port de la couche transport utilisé est le `5353`.
- Toutes les requêtes et les réponses sont émises à destination de l'adresse de multidiffusion `ff02::fb`. De cette façon, tous les hôtes à l'écoute sont «au courant» des échanges d'enregistrements.
- Pour éviter les conflits avec le service DNS classique, un Top Level Domain (TLD) spécifique a été défini. Il s'agit du `.local`.
- Chaque hôte détient et est responsable de ses propres enregistrements.
- Par défaut, la portée du service est limitée à un seul domaine de diffusion ou VLAN. Par définition, le préfixe IPv6 `ff02::/16` à pour portée le lien local uniquement.

Voici un extrait d'analyse réseau qui illustre chacun des points énoncés ci-dessus.

```
2001:db8:feb2:14:b8ad:ff:feca:fe15 -> ff02::fb     MDNS 94 Standard query 0x0000  AAAA ipv6-rtr.local,
2001:db8:feb2:14::1 -> ff02::fb     MDNS 116 Standard query response 0x0000  AAAA, cache flush 2001:db8
```

- Sur la première ligne, l'hôte `vm21` avec l'adresse IPv6 `2001:db8:feb2:14:b8ad:ff:feca:fe15` cherche à connaître l'adresse IPv6 de l'hôte `ipv6-rtr.local`.
- Sur la seconde ligne, l'hôte concerné, `ipv6-rtr.local` répond directement sur le canal de multidiffusion en donnant son adresse IPv6 : `2001:db8:feb2:14::1`.

Suite à cet échange de question/réponse, les hôtes peuvent communiquer à partir de leurs noms.

Voyons maintenant quelles sont les opérations à effectuer pour bénéficier du service mDNS. Elles sont au nombre de deux.

- Il faut installer les deux paquets avahi-daemon et avahi-utils.

```
etu@vm21:~$ aptitude search ~iavahi
i   avahi-daemon           - Démon Avahi pour mDNS/DNS-SD
i   avahi-utils            - Utilitaires de navigation, publication et découverte pour Avahi
i A libavahi-client3       - bibliothèque client Avahi
i A libavahi-common-data   - Fichiers de données communs d'Avahi
i A libavahi-common3       - bibliothèque commune Avahi
i A libavahi-core7         - Bibliothèque Avahi pour mDNS/DNS-SD pour l'embarqué
```

- Il faut éditer la ligne `hosts` du fichier de configuration du «commutateur» des services de noms, `/etc/nsswitch.conf`, et référencer le service mDNS.

```
etu@vm21:~$ cat /etc/nsswitch.conf
# /etc/nsswitch.conf
#
# Example configuration of GNU Name Service Switch functionality.
# If you have the `glibc-doc-reference' and `info' packages installed, try:
# `info libc "Name Service Switch"' for information about this file.

passwd:         compat
group:          compat
shadow:         compat

hosts:          files mdns_minimal [NOTFOUND=return] dns mdns
networks:       files

protocols:      db files
services:       db files
ethers:         db files
rpc:            db files

netgroup:       nis
```

Pour valider cette configuration, on peut faire un test élémentaire qui désigne l'hôte à contacter par son nom. Avec la commande ping6, on obtient les résultats suivants :

```
etu@vm21:~$ ping6 -c 2 ipv6-rtr.local
PING ipv6-rtr.local(IPv6-rtr.local) 56 data bytes
64 bytes from IPv6-rtr.local: icmp_seq=1 ttl=64 time=0.799 ms
64 bytes from IPv6-rtr.local: icmp_seq=2 ttl=64 time=0.712 ms

--- ipv6-rtr.local ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.712/0.755/0.799/0.051 ms
```

Ce test est effectué entre deux hôtes appartenant au même VLAN. Les hôtes `vm21` et `ipv6-rtr` sont deux placés dans le VLAN numéro `20`.

## Routeur & *reflector* mDNS

La configuration du démon avahi-daemon prévoit que celui-ci puisse jouer le rôle de concentrateur ou reflector. Compte tenu de son rôle particulier, le routeur peut collecter les enregistrements de tous les domaines de diffusion qu'il dessert puis les rendre disponibles à l'ensemble des hôtes.

Pour activer la fonction reflector, il faut éditer le fichier de configuration `/etc/avahi/avahi-daemon.conf` et rechercher la section correspondante. Voici un extrait du fichier avec l'option activée.

```
[reflector]
enable-reflector=yes
reflect-ipv=yes
```

Une fois le démon redémarré, les hôtes des trois domaines de diffusion sont accessibles les uns les autres. Il suffit de faire un test avec l'outil avahi-browse fourni avec le paquet avahi-utils toujours depuis l'hôte `vm21`.

```
etu@vm21:~$ avahi-browse --all
+   eth0 IPv6 vm21 [ba:ad:00:ca:fe:15]          Workstation         local
+   eth0 IPv6 vm20 [ba:ad:00:ca:fe:14]          Workstation         local
+   eth0 IPv6 host [ae:d3:ca:81:db:7e]          Workstation         local
+   eth0 IPv6 IPv6-rtr [de:ad:00:be:ef:14]      Workstation         local
+   eth0 IPv6 vm10 [ba:ad:00:ca:fe:0a]          Workstation         local
+   eth0 IPv6 IPv6-rtr [ba:ad:00:ca:fe:00]      Workstation         local
+   eth0 IPv6 IPv6-rtr [de:ad:00:be:ef:0a]      Workstation         local
+   eth0 IPv6 IPv6-rtr [ba:ad:00:ca:fe:01]      Workstation         local
^CGot SIGINT, quitting
```

Dans la liste ci-dessus, on repère les hôtes des autres VLANs. On peut maintenant les contacter directement par leur nom et le TLD `.local`.

```
etu@vm21:~$ ssh vm10.local
The authenticity of host 'vm10.local (2001:db8:feb2:a:b8ad:ff:feca:fe0a)' can't be established.
RSA key fingerprint is 4e:6e:27:b5:c1:89:94:2c:d6:6c:72:d8:7a:e0:d3:e7.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'vm10.local,2001:db8:feb2:a:b8ad:ff:feca:fe0a' (RSA) to the list of known ho
etu@vm10.local's password:
Linux vm10 3.13-1-686-pae #1 SMP Debian 3.13.10-1 (2014-04-15) i686
No mail.
Last login: Fri May  2 13:51:37 2014 from ipv6-rtr.local
etu@vm10:~$
```

## 8. Pour conclure

Voilà ! On dispose maintenant d'une maquette fonctionnelle avec : autoconfiguration IPv6 SLAAC, resolver DNS et service multicast DNS. Cette maquette est une base qui doit permettre d'évaluer et développer de nouveaux services applicatifs.

Ce document, même s'il est déjà trop long, met l'accent sur le fonctionnement et fait l'impasse sur deux grands sujets au moins. Toutes les annonces RAs s'appuient sur le protocole NDP, décrit dans le document standard RFC4861 Neighbor Discovery for IP version 6 (IPv6), dont on ne parle pas. De la même façon, on ne donne aucune information sur le format et les catégories d'adresses IPv6 utilisées. Ces deux points méritent que l'on s'y attarde.

Une fois de plus dans le domaine des réseaux, il reste encore beaucoup à découvrir et à apprendre !

## Glossaire

| | |
|---|---|
| Dynamic Host Control Protocol (DHCP) | Ce service attribue automatiquement les paramètres de configuration à une interface réseau qui en fait la demande : adresse IP, passerelle par défaut et resolver DNS. L'adresse MAC de l'interface est la clé d'enregistrement des paramètres attribués. En effet, l'adresse MAC d'une interface est la seule «identité» dont l'hôte dispose au moment de la requête de configuration réseau. |
| Domain Name System (DNS) | Le principe de base du service de résolution des noms de domaines est d'associer un nom à une adresse IP. Il repose sur un enregistrement des Resource Records (RRs) qui constituent une base de données. |
| Media Access Control address (MAC) | L'adresse MAC est un identifiant unique affecté à une interface réseau. Elle est utilisée dans les trames du niveau liaison de données de la modélisation pour repérer un hôte dans un réseau local de diffusion. Elle sert aussi d'identifiant dans l'attribution des paramètres de configuration réseau d'une interface. |
| Native Address Translation (NAT) | La traduction d'adresses a été introduite pour prévenir la pénurie des adresses IPv4. Ce mécanisme permet d'associer un groupe d'adresses dites privées à une ou plusieurs adresses publiques. Le passage d'un groupe d'adresses à une seule adresse visible de l'Internet utilise une table d'enregistrement des flux réseau. Il s'agit d'un exemple caractéristique de service Stateful. Si cette technique a été massivement utilisée, elle entraîne une rupture dans le principe des réseaux à commutation de paquets IP qui supposent que n'importe quel hôte soit susceptible de contacter n'importe quel autre hôte. |
| Router Advertisements (RA) | Les annonces de paramètres d'autoconfiguration des interfaces des hôtes raccordés au réseau par les routeurs font partie du protocole décrit dans le document RFC4861 Neighbor Discovery for IP version 6 (IPv6). Dans ces annonces, on trouve le préfixe réseau, la passerelle par défaut ainsi que l'adresse du ou des serveurs DNS. |
| Recursive DNS Server (RDNSS) | Les annonces RDNSS font partie des paramètres d'autoconfiguration IPv6. Elles sont définies dans le document RFC6106 IPv6 Router Advertisement Options for DNS Configuration. Elles contiennent les adresses des serveurs DNS que les hôtes doivent utiliser pour leurs requêtes au service de noms de domaines. |
| IPv6 Stateless Address Autoconfiguration (SLAAC) | L'acronyme SLAAC fait référence à l'autoconfiguration IPv6 décrite dans le document RFC4862 IPv6 Stateless Address Autoconfiguration. |
| Interface trunk | Une interface réseau en mode trunk est un port physique qui véhicule le trafic de plusieurs réseaux locaux distincts appelés VLANs. |
| Virtual Local Area Network (VLAN) | Un VLAN est un domaine de diffusion distribué entre les équipements de commutation de trames Ethernet. Dans le contexte de ce document, les VLANs sont utilisés pour découper 3 réseaux locaux distincts. Parmi ces réseaux, deux sont utilisés pour l'autoconfiguration IPv6. Pour plus d'informations sur le sujet, voir le document Routage Inter-VLAN. |