RFC 8415 : Dynamic Host Configuration Protocol for IPv6 (DHCPv6)

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 3 mai 2019

Date de publication du RFC : Novembre 2018

https://www.bortzmeyer.org/8415.html

IPv6 dispose de trois mécanismes principaux pour l'allocation d'une adresse IP à une machine. L'allocation statique, « à la main », le système d'« autoconfiguration » SLAAC du RFC 4862[1] et DHCP. DHCP pour IPv6 avait été normalisé pour la première fois dans le RFC 3315, que notre RFC met à jour. Le protocole n'a guère changé mais le texte du RFC a été sérieusement revu (DHCP est un protocole compliqué).

DHCP permet à une machine (qui n'est pas forcément un ordinateur) d'obtenir une adresse IP (ainsi que plusieurs autres informations de configuration) à partir d'un serveur DHCP du réseau local. C'est donc une configuration « avec état », du moins dans son mode d'utilisation le plus connu. (Notre RFC documente également un mode sans état.) DHCP nécessite un serveur, par opposition à l'autoconfiguration du RFC 4862 qui ne dépend pas d'un serveur (cette autoconfiguration sans état peut être utilisée à la place de, ou bien en plus de DHCP). Deux utilisations typiques de DHCP sont le SoHo où le routeur ADSL est également serveur DHCP pour les trois PC connectés et le réseau local d'entreprise où deux ou trois machines Unix distribuent adresses IP et informations de configuration à des centaines de machines.

Le principe de base de DHCP (IPv4 ou IPv6) est simple : la nouvelle machine annonce à la cantonade qu'elle cherche une adresse IP, le serveur lui répond, l'adresse est allouée pour une certaine durée, le **bail**, la machine cliente devra renouveler le bail de temps en temps.

L'administrateur d'un réseau IPv6 se pose souvent la question « DHCP ou SLAAC » ? Notez que les deux peuvent coexister, ne serait-ce que parce que certaines possibilités n'existent que pour un seul des

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc4862.txt

deux protocoles. Ainsi, DHCP seul ne peut indiquer l'adresse du routeur par défaut. Pour le reste, c'est une question de goût.

Le DHCP spécifié par notre RFC ne fonctionne que pour IPv6, les RFC 2131 et RFC 2132 traitant d'IPv4. Les deux protocoles restent donc complètement séparés, le RFC 4477 donnant quelques idées sur leur coexistence. Il a parfois été question de produire une description unique de DHCPv4 et DHCPv6, ajoutant ensuite les spécificités de chacun, mais le projet n'a pas eu de suite (section 1.2 de ce RFC), les deux protocoles étant trop différents. De même, l'idée de permettre l'envoi d'informations spécifiques à IPv4 (par exemple l'adresse IPv4, proposée dans le RFC 3315, section 1) au-dessus de DHCPv6 a été abandonnée (mais voyez quand même le RFC 7341).

DHCP fonctionne par diffusion restreinte. Un **client** DHCP, c'est-à-dire une machine qui veut obtenir une adresse, diffuse (DHCP fonctionne au-dessus d'UDP, RFC 768, le port source est 546, le port de destination, où le serveur écoute, est 547) sa demande à l'adresse "*multicast*" locale au lien `ff02::1:2`. Le serveur se reconnait et lui répond. S'il n'y a pas de réponse, c'est, comme dans le DNS, c'est au client de réémettre (section 15). L'adresse IP source du client est également une adresse locale au lien.

(Notez qu'une autre adresse de diffusion restreinte est réservée, `ff05::1:3`; elle inclut également tous les serveurs DHCP mais, contrairement à la précédente, elle exclut les relais, qui transmettent les requêtes DHCP d'un réseau local à un autre.)

Le serveur choisit sur quels critères il alloue les adresses IP. Il peut les distribuer de manière statique (une même machine a toujours la même adresse IP) ou bien les prendre dans un "*pool*" d'adresses et chaque client aura donc une adresse « dynamique ». Le fichier de configuration du serveur DHCP ISC ci-dessous montre un mélange des deux approches.

Il faut bien noter (et notre RFC le fait dans sa section 22) que DHCP n'offre aucune sécurité. Comme il est conçu pour servir des machines non configurées, sur lesquelles on ne souhaite pas intervenir, authentifier la communication est difficile. Un serveur DHCP pirate, ou, tout simplement, un serveur DHCP accidentellement activé, peuvent donc être très gênants. Les approches de sécurité suggérées dans le RFC 3315 se sont avérées peu pratiques et plusieurs ont été retirées dans notre nouveau RFC.

Outre l'adresse IP, DHCP peut indiquer des options comme les adresses des serveurs DNS à utiliser (RFC 3646).

Notre version IPv6 de DHCP est assez différente de la version IPv4 (et le RFC est plus de trois fois plus long). Par exemple, l'échange « normal » entre client et serveur prend quatre paquets IP (section 5) et non pas deux. (Il y a aussi un échange simplifié à deux paquets, cf. section 5.1.) L'encodage des messages est très différent, et il y a des différences internes comme l'IA ("*Identity Association*") de la section 12. Il y a aussi des différences visibles à l'utilisateur comme le concept de DUID ("*DHCP Unique IDentifier*"), section 11, qui remplace les anciens "*client identifier*" et "*server identifier*" de DHCP v4. Les différences sont telles que le RFC précise que leur intégration avec DHCP pour IPv4 n'est pas envisagée.

À l'heure actuelle, il existe plusieurs mises en œuvre de DHCPv6, Dibbler <`http://klub.com.pl/dhcpv6/`> (client et serveur, mais qui n'est plus maintenu), celle de l'Institut Leibniz à Dresde <`http://dhcpy6d.ifw-dresden.de/`> (serveur seulement), celles de l'ISC, l'ancien DHCP <`https://www.isc.org/downloads/dhcp/`> et le nouveau, nommé Kea <`https://www.isc.org/kea/`> (dans les deux cas, serveur seulement) et dhcpcd <`https://roy.marples.name/projects/dhcpcd`> (client seulement). Pour celles et ceux qui utilisent une Freebox comme serveur DHCP, il semble qu'elle ait DHCPv6 depuis 2018 <`https://www.zdnet.fr/actualites/la-freebox-se-dote-de-nouvelles-fo`

htm> (je n'ai pas testé). Il parait que la Livebox le fait également. Je n'ai pas non plus essayé pour la Turris Omnia <https://www.bortzmeyer.org/turris.html> mais cela devrait marcher puisqu'elle utilise le serveur odhcpd <https://git.openwrt.org/project/odhcpd.git>, qui sait faire du DHCPv6. Et il y a bien sûr des implémentations non-libres dans des équipements comme les Cisco. Notez que ces mises en œuvre de DHCPv6 n'ont pas forcément déjà intégré les modifications de notre RFC 8415.

Voici un exemple d'utilisation de Dibbler, qui nous affiche les quatre messages (Solicit - Advertise - Request - Reply) :

```
% sudo dibbler-client run
...
2019.05.02 11:35:15 Client Notice    Unable to open DUID file (client-duid), generating new DUID.
2019.05.02 11:35:15 Client Notice    DUID creation: Generating 14-bytes long link-local+time (duid-llt) DUID.
2019.05.02 11:35:15 Client Info      My DUID is 00:01:00:01:24:5d:76:53:00:1e:8c:76:29:b6.
...
2019.05.02 12:12:38 Client Info      Creating SOLICIT message with 1 IA(s), no TA and 0 PD(s) on eth1/2 interfac
2019.05.02 12:12:38 Client Info      Received ADVERTISE on eth1/2,trans-id=0x614722, 4 opts: 1 2 3 23
2019.05.02 12:12:39 Client Info      Processing msg (SOLICIT,transID=0x614722,opts: 1 3 8 6)
2019.05.02 12:12:39 Client Info      Creating REQUEST. Backup server list contains 1 server(s).
2019.05.02 12:12:39 Client Info      Received REPLY on eth1/2,trans-id=0x634881, 4 opts: 1 2 3 23
2019.05.02 12:12:39 Client Notice    Address fde8:9fa9:1aba:0:fafa::2/128 added to eth1/2 interface.
2019.05.02 12:12:39 Client Notice    Setting up DNS server 2001:db8:2::dead:beef on interface eth1/2.
```

Le serveur en face était un Kea ainsi configuré :

```
"subnet6": [
        {
           "interface": "eth0",
           "subnet": "fde8:9fa9:1aba:0::/64",
           "pools": [ { "pool": "fde8:9fa9:1aba:0:fafa::/80" } ],
...
```

Pour que Kea puisse écouter sur le port DHCP, il a aussi fallu :

```
%  sudo setcap 'cap_net_bind_service=+ep' /usr/sbin/kea-dhcp6
```

(Sinon, c'est le "*Permission denied*".) Si vous voulez, le pcap de l'échange est disponible (en ligne sur https://www.bortzmeyer.org/files/dhcpv6.pcap) (capture faite avec tcpdump -w /tmp/dhcpv6.pcap udp and port 546 or port 547). tcpdump voit le trafic ainsi :

```
12:17:44.531859 IP6 fe80::21e:8cff:fe76:29b6.546 > ff02::1:2.547: dhcp6 solicit
12:17:44.555202 IP6 fe80::d40b:5ff:fee8:a36b.547 > fe80::21e:8cff:fe76:29b6.546: dhcp6 advertise
12:17:45.559247 IP6 fe80::21e:8cff:fe76:29b6.546 > ff02::1:2.547: dhcp6 request
12:17:45.567875 IP6 fe80::d40b:5ff:fee8:a36b.547 > fe80::21e:8cff:fe76:29b6.546: dhcp6 reply
```

On voit bien les quatre messages (Solicit - Advertise - Request - Reply). Avec l'option -vvv, tcpdump est plus bavard et montre qu'il analyse bien DHCPv6 :

```
12:17:44.531859 IP6 (flowlabel 0x90d05, hlim 1, next-header UDP (17) payload length: 58) fe80::21e:8cff:fe76
12:17:44.555202 IP6 (flowlabel 0xa6d2a, hlim 64, next-header UDP (17) payload length: 128) fe80::d40b:5ff:fe
12:17:45.559247 IP6 (flowlabel 0x90d05, hlim 1, next-header UDP (17) payload length: 104) fe80::21e:8cff:fe7
12:17:45.567875 IP6 (flowlabel 0xa6d2a, hlim 64, next-header UDP (17) payload length: 128) fe80::d40b:5ff:fe
```

Mais si vous préférez tshark, l'analyse de cet échange est également disponible (en ligne sur `https://www.bortzmeyer.org/files/dhcpv6.txt`).

Quant au fichier de configuration du traditionnel serveur ISC (celui avant Kea), il ressemble beaucoup à ce qu'il est en v4 :

```
subnet6 2001:db8:dead:babe::/64 {
     range6      2001:db8:dead:babe::100 2001:db8:dead:babe::FFF;
     # On peut aussi utiliser préfixe/longueur au lieu d'indiquer les
     # adresses de début et de fin de la plage
}
```

On doit lancer le serveur avec l'option -6 (le même démon ne peut pas servir le v4 et le v6 en même temps, les deux protocoles étant trop différents) :

```
# dhcpd -6 -d -f
Internet Systems Consortium DHCP Server 4.0.0
...
Listening on Socket/eth0/2001:db8:dead:babe::/64
Sending on   Socket/eth0/2001:db8:dead:babe::/64

[Puis arrive une requête]

Solicit message from fe80::219:b9ff:fee4:25f9 port 546, transaction ID 0x4BB14F00
Picking pool address 2001:db8:dead:babe::fbb
Sending Advertise to fe80::219:b9ff:fee4:25f9 port 546
Request message from fe80::219:b9ff:fee4:25f9 port 546, transaction ID 0x46B10900
Sending Reply to fe80::219:b9ff:fee4:25f9 port 546
```

(Le concept de "*transaction ID*" est décrit sections 8 et 16.1.) La requête est émise depuis une adresse **lien-local** (ici `fe80::219:b9ff:fee4:25f9`) pas depuis une adresse « tout zéro » comme en IPv4 (section 17 du RFC). Vu avec tcpdump, la requête est :

```
15:07:43.455918 IP6 fe80::219:b9ff:fee4:25f9.546 > ff02::1:2.547: dhcp6 solicit
15:07:43.456098 IP6 fe80::219:b9ff:fee4:2987.547 > fe80::219:b9ff:fee4:25f9.546: dhcp6 advertise
15:07:44.512946 IP6 fe80::219:b9ff:fee4:25f9.546 > ff02::1:2.547: dhcp6 request
15:07:44.513233 IP6 fe80::219:b9ff:fee4:2987.547 > fe80::219:b9ff:fee4:25f9.546: dhcp6 reply
```

On note que l'échange a été celui à quatre messages (`Solicit - Advertise - Request - Reply`), décrit section 5.2 (et la liste des types possibles est en section 7.3). Le serveur n'a pas répondu directement avec un `Reply`, parce que le client n'a pas inclus l'option `Rapid Commit` (section 21.14). Cette option n'est pas actuellement gérée par le client DHCP utilisé (l'option `dhcp6.rapid-commit` existe mais la documentation précise qu'elle est ignorée). Dans l'échange à quatre message,s, le client demande à tous (`Solicit`), un(s) serveur(s) DHCP répond(ent) (`Advertise`), le client envoie alors sa requête au serveur choisi (`Request`), le serveur donne (ou pas) son accord (`Reply`).

L'échange à deux messages (`Solicit- Reply`) est, lui, spécifié dans la section 5.1. Il s'utilise si le client n'a pas besoin d'une adresse IP, juste d'autres informations de configuration comme l'adresse du serveur NTP, comme décrit dans le RFC 4075. Même si le client demande une adresse IP, il est possible d'utiliser l'échange à deux messages, via la procédure rapide avec l'option `Rapid Commit`.

Actuellement, l'attribution d'adresses statiques à une machine, en la reconnaissant, par exemple, à son adresse MAC est plus délicate avec le serveur de l'ISC (merci à Shane Kerr pour son aide). Il faut trouver le "*client identifier*" (section 21.2 du RFC, deux méthodes possibles pour le trouver sont expliquées plus loin) et le mettre dans `dhcpd.conf` :

```
host lilith {
  host-identifier option dhcp6.client-id 0:1:0:1:47:96:21:f7:0:19:b9:e4:25:f9;
  fixed-address6 2001:db8:dead:babe::2;
}
```

et cette adresse IP fixe est donnée au client.

Pour trouver le "*client identifier*", une méthode spécifique au client DHCP de l'ISC est de regarder dans le fichier des baux du client (typiquement `/var/db/dhclient6.leases`) :

```
...
  option dhcp6.client-id 0:1:0:1:47:96:21:f7:0:19:b9:e4:25:f9;
```

Il suffit alors de le copier-coller.

Une autre méthode, plus complexe, mais qui marche avec tous les clients DHCP est de lancer tcpdump en mode bavard :

```
# tcpdump -n -vvv ip6 and udp and port 547
12:24:15.084006 IP6 (hlim 64, next-header UDP (17) payload length: 60) fe80::219:b9ff:fee4:25f9.546 > ff02::1:2.
```

Tout client ou serveur DHCP v6 a un DUID ("*DHCP Unique Identifier*", décrit en section 11). Le DUID est opaque et ne devrait pas être analysé par la machine qui le reçoit. La seule opération admise est de tester si deux DUID sont égaux (indiquant qu'en face, c'est la même machine). Il existe plusieurs façons de générer un DUID (dans l'exemple plus haut, Dibbler avait choisi la méthode `duid-llt`, adresse locale et heure) et de nouvelles pourront apparaitre dans le futur. Par exempe, un DUID peut être fabriqué à partir d'un UUID (RFC 6355).

Le "*client identifier*" de l'exemple avec le serveur de l'ISC ci-dessus, le DUID, a été fabriqué, comme pour Dibbler, en concaténant le type de DUID (ici, 1, "*Link-layer Address Plus Time*", section 11.2 du RFC), le type de matériel (1 pour Ethernet), le temps (ici 1201021431, notons que ce client DHCP violait le RFC en comptant les secondes à partir de 1970 et pas de 2000) et l'adresse MAC, ce qui redonne le même résultat au prix de quelques calculs avec `bc`.

Mais l'utilisation exclusive du DUID, au détriment de l'adresse MAC, n'est pas une obligation du RFC (le RFC, section 11, dit juste « "*DHCP servers use DUIDs to identify clients for the selection of configuration parameters*" », ce qui n'interdit pas d'autres méthodes), juste un choix des développeurs de l'ISC. Le serveur de Dresde, dhcpy6d `<http://dhcpy6d.ifw-dresden.de/>`, permet d'utiliser les adresses

MAC, comme on le fait traditionnellement en IPv4. En combinaison avec des commutateurs qui filtrent sur l'adresse MAC, cela peut améliorer la sécurité.

La section 6 de notre RFC décrit les différentes façons d'utiliser DHCPv6. On peut se servir de DHCPv6 en mode sans état (section 6.1), lorsqu'on veut juste des informations de configuration, ou avec état (section 6.2, qui décrit la façon historique d'utiliser DHCP), lorsqu'on veut réserver une ressource (typiquement l'adresse IP) et qu'il faut alors que le serveur enregistre (et pas juste dans sa mémoire, car il peut redémarrer) ce qui a été réservé. On peut aussi faire quelque chose qui n'a pas d'équivalent en IPv4, se faire déléguer un préfixe d'adresses IP entier (section 6.3). Un client DHCP qui reçoit un préfixe, mettons, /60, peut ensuite redéléguer des bouts, par exemple ici des /64. (Le RFC 7084 est une utile lecture au sujet des routeurs installés chez M. Toutlemonde.)

Le format détaillé des messages est dans la section 8. Le début des messages est toujours le même, un type d'un octet (la liste des types est en section 7.3) suivi d'un identificateur de transaction de trois octets. Le reste est variable, dépendant du type de message.

On a déjà parlé du concept de DUID plus haut, donc sautons la section 11 du RFC, qui parle du DUID, et allons directement à la section 12, qui parle d'IA ("*Identity Association*"). Une IA est composée d'un identifiant numérique, l'IAID ("*IA IDentifier*") et d'un ensemble d'adresses et de préfixes. Le but du concept d'IA est de permettre de gérer collectivement un groupe de ressources (adresses et préfixes). Pour beaucoup de clients, le concept n'est pas nécessaire, on n'a qu'une IA, d'identificateur égal à zéro. Pour les clients plus compliqués, on a plusieurs IA, et les messages DHCP (par exemple d'abandon d'un bail) indiquent l'IA concernée.

Comme pour DHCPv4, une bonne partie des informations est transportée dans des options, décrites dans la section 21. Certaines options sont dans ce RFC, d'autres pourront apparaitre dans des RFC ultérieurs. Toutes les options commencent par deux champs communs, le code identifiant l'option (deux octets), et la longueur de l'option. Ces champs sont suivis par des données, spécifiques à l'option. Ainsi, l'option "*Client Identifier*" a le code 1, et les données sont un DUID (cf. section 11). Autre exemple, l'option "*Vendor Class*" (code 16) permet d'indiquer le fournisseur du logiciel client (notez qu'elle pose des problèmes de sécurité, cf. RFC 7824, et section 23 de notre RFC). Notez qu'il peut y avoir des options dans les options, ainsi, l'adresse IP (code 5) est toujours dans les données d'une option IA (les IA sont décrites en section 12).

Puisqu'on a parlé de sécurité, la section 22 du RFC détaille les questions de sécurité liées à DHCP. Le fond du problème est qu'il y a une profonde incompatibilité entre le désir d'une autoconfiguration simple des clients (le but principal de DHCP) et la sécurité. DHCP n'a pas de chiffrement et tout le monde peut donc écouter les échanges de messages, voire les modifier. Et, de toute façon, le serveur n'est pas authentifié, donc le client ne sait jamais s'il parle au serveur légitime. Il est trivial pour un méchant de configurer un serveur DHCP « pirate » et de répondre à la place du vrai, indiquant par exemple un serveur DNS que le pirate contrôle. Les RFC 7610 et RFC 7513 décrivent des solutions possibles à ce problème.

Des attaques par déni de service sont également possibles, par exemple via un client méchant qui demande des ressources (comme des adresses IP) en quantité. Un serveur prudent peut donc limiter la quantité de ressources accessible à un client.

Nouveauté de ce RFC (elle était quasiment absente du RFC 3315), les questions de vie privée. La section 23 rappelle que DHCP est très indiscret. Le RFC 7824 décrit les risques que DHCP fait courir à la vie privée du client (et le RFC 7844 des solutions possibles).

Les registres IANA ne changent pas par rapport à l'ancien RFC. Les différents paramètres sont en ligne <https://www.iana.org/assignments/dhcpv6-parameters/dhcpv6-parameters.xml>.

L'annexe A de notre RFC décrit les changements depuis le vieil RFC 3315. Ils sont nombreux mais en général pas cruciaux. On notera :

— beaucoup de changements dans le texte, pour clarifier certains points, ou mieux expliquer des questions complexes,
— une section 6 sur les questions d'utilisation de DHCP, toute neuve,
— l'option "*Option Request*" devient obligatoire dans certains messages,
— une nouvelle section sur la vie privée,
— les références à IPsec (jamais déployé) ont été retirées,
— le protocole d'« authentification retardée » (jamais déployé, et probablement pas déployable) a été retiré (cf. section 25),
— la possibilité pour le client d'indiquer la durée d'un bail souhaitée a été retirée,
— plusieurs RFC séparés ont été fusionnés avec ce nouveau RFC, comme la délégation de préfixe, initialement séparée dans le RFC 3633 (l'implémenteur de DHCPv6 aura donc désormais une vue unifiée du protocole, et moins de RFC à lire),
— le mode « sans état », qui était dans le RFC 3736, a été intégré,
— le mécanisme de ralentissement des clients lorsque le serveur ne répond pas (ex-RFC 7083), le relayage des messages de type inconnu (remplaçant donc le RFC 7283), et le fonctionnement des différents modes (au revoir, le RFC 7550) sont également intégrés,
— une bonne douzaine de bogues relevées dans le RFC 3315 et ses RFC proches, ont été corrigées.