RFC 9915 : Dynamic Host Configuration Protocol for IPv6 (DHCPv6)

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 11 février 2026

Date de publication du RFC : Janvier 2026

https://www.bortzmeyer.org/9915.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 était normalisé dans le RFC 8415, que notre RFC met à jour. Le protocole n'a guère changé, le principal changement est la suppression de certaines fonctions peu utilisées.

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

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

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.

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 Kea 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.

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. (Mais 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, comme Kea <https://www.isc.org/kea/> (serveur seulement) et dhcpcd <https://roy.marples.name/projects/dhcpcd> (client seulement). (Notez qu'une liste complète figurait dans le brouillon du RFC <https://www.ietf.org/archive/id/draft-ietf-dhc-rfc8415bis-12.html#name-implementation-status>.) 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-fonctions-3986333.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 (ceci dit, je ne vois pas comment l'activer depuis les menus de Luci). 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 9915.

Il existe aussi des programmes qui ne sont plus maintenus comme Dibbler <https://github.com/tomaszmrugalski/dibbler> (client et serveur), l'ancien programme <https://www.isc.org/downloads/dhcp/> de l'ISC (le nouveau est Kea), etc.

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

```
% sudo dibbler-client run
...
2026.02.11 08:28:04 Client Notice    DUID creation: Generating 14-bytes long link-local+time (duid-llt) DUID.
2026.02.11 08:28:04 Client Notice    DUID creation: generated using wlan0/4 interface.
2026.02.11 08:28:04 Client Info      My DUID is 00:01:00:01:31:1e:fa:14:f6:fc:69:10:65:09.
...
2026.02.11 08:28:04 Client Info      Creating SOLICIT message with 1 IA(s), no TA and 0 PD(s) on eth1/3 interfac
2026.02.11 08:28:04 Client Debug     Sending SOLICIT(opts:1 3 39 8 6 ) on eth1/3 to multicast.
2026.02.11 08:28:04 Client Info      Received ADVERTISE on eth1/3,trans-id=0x5ded1b, 5 opts: 1 2 3 23 39
2026.02.11 08:28:05 Client Info      Processing msg (SOLICIT,transID=0x5ded1b,opts: 1 3 39 8 6)
2026.02.11 08:28:05 Client Info      Creating REQUEST. Backup server list contains 1 server(s).
2026.02.11 08:28:05 Client Debug     Advertise from Server ID=00:01:00:01:31:19:db:68:38:f7:cd:ce:22:c6, prefere
2026.02.11 08:28:05 Client Debug     Sending REQUEST(opts:1 3 39 6 2 8 ) on eth1/3 to multicast.
2026.02.11 08:28:05 Client Info      Received REPLY on eth1/3,trans-id=0x6eb008, 5 opts: 1 2 3 23 39
2026.02.11 08:28:05 Client Notice    Address fc00:cafe:1234:4321:5678::2/128 added to eth1/3 interface.
2026.02.11 08:28:05 Client Debug     RENEW(IA_NA) will be sent (T1) after 1000, REBIND (T2) after 2000 seconds.
2026.02.11 08:28:08 Client Notice    FQDN: About to perform DNS Update: DNS server=2001:db8:2::dead:beef, IP=fc0
...
```

Le serveur en face était un Kea ainsi configuré :

```
"subnet6": [
        {
           "interface": "eth0",  // Ce n'est pas bien documenté mais
    // cette option est cruciale, autrement le client reçoit
    // des « Server could not select subnet for this client ».
           "subnet": "fc00:cafe:1234:4321::/64",
           "pools": [ { "pool": "fc00:cafe:1234:4321:5678::/80" } ],
...
```

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

```
09:28:04.624687 IP6 fe80::606d:ad11:58ca:6cab.546 > ff02::1:2.547: dhcp6 solicit
09:28:04.639479 IP6 fe80::6ee4:5672:da95:1018.547 > fe80::606d:ad11:58ca:6cab.546: dhcp6 advertise
09:28:05.652900 IP6 fe80::606d:ad11:58ca:6cab.546 > ff02::1:2.547: dhcp6 request
09:28:05.667843 IP6 fe80::6ee4:5672:da95:1018.547 > fe80::606d:ad11:58ca:6cab.546: dhcp6 reply
```

La requête est émise depuis une adresse **lien-local** (ici `fe80::606d:ad11:58ca:6cab`) pas depuis une adresse « tout zéro » comme en IPv4 (section 17 du RFC). On voit bien les 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). Dans l'échange à quatre messages, 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`). Avec l'option `-vvv`, tcpdump est plus bavard et montre qu'il analyse bien DHCPv6 :

```
09:28:04.624687 IP6 (flowlabel 0x51d28, hlim 1, next-header UDP (17) payload length: 99) fe80::606d:ad11:58c
09:28:04.639479 IP6 (flowlabel 0xf6846, hlim 64, next-header UDP (17) payload length: 140) fe80::6ee4:5672:c
09:28:05.652900 IP6 (flowlabel 0x51d28, hlim 1, next-header UDP (17) payload length: 117) fe80::606d:ad11:58
09:28:05.667843 IP6 (flowlabel 0xf6846, hlim 64, next-header UDP (17) payload length: 140) fe80::6ee4:5672:c
```

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

Notez que certains clients DHCP dépendent de la présence d'un routeur qui envoie des RA (RFC 4861) avec le bit M - "*Managed*" - à 1 (RFC 4861, section 4.2). En l'absence de ces annonces, le client se contente de demander des informations diverses au serveur DHCP, mais pas d'adresse IP.

Et si vous voulez compiler dhcpcd vous-même, c'est simple :

```
wget https://github.com/NetworkConfiguration/dhcpcd/releases/download/v10.3.0/dhcpcd-10.3.0.tar.xz
dhcpcd-10.3.0.tar.xz
tar xvf dhcpcd-10.3.0.tar
dhcpcd-10.3.0
./configure
make
sudo make install
```

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`.

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).

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). On peut utiliser l'adresse Ethernet. En combinaison avec des commutateurs qui filtrent sur l'adresse MAC, cela peut améliorer la sécurité.

Puisqu'on peut aussi attribuer des adresses statiquement à une machine, en la reconnaissant, par exemple, à son adresse MAC ou à son DUID, voici comment on peut configurer Kea pour donner une adresse IP fixe au client d'un certain DUID :

```
"reservations": [
  {
    "duid": "00:01:00:01:31:1e:fa:14:f6:fc:69:10:65:09",
    "ip-addresses": [ "fc00:cafe:1234:4321:b0f:1111::1" ]
  }
```

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.

Maintenant, 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 l'ancien RFC 8415. Rien d'essentiel n'a été changé. On notera :

— Suppression de certains mécanismes optionnels (comme ils étaient de toute façon optionnels, cela n'affecte pas l'interopérabilité) qui étaient complexes et peu mis en œuvre : les adresses temporaires (IA_TA), il faut désormais relâcher explicitement celles qu'on veut temporaires, la possibilité de faire de l'"*unicast*", et donc, logiquement, l'option qui forçait le "*multicast*".
— Correction de plusieurs erreurs signalées <https://www.rfc-editor.org/errata_search.php?rfc=8415&rec_status=15&presentation=table> (certaines n'ont pas été corrigées puisqu'elles s'appliquaient à des options supprimées).
— Texte plus précis sur les ports UDP utilisés.
— Progression du RFC au statut de norme Internet complète (alors que le RFC 8415 était officiellement une proposition de norme).