RFC 7227 : Guidelines for Creating New DHCPv6 Options

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 11 mai 2014

Date de publication du RFC : Mai 2014

https://www.bortzmeyer.org/7227.html

Le protocole DHCPv6, normalisé dans le RFC 8415 [1], permet de configurer des clients IPv6 depuis un serveur central, qui maintient un état (adresses IP attribuées, par exemple). Des tas d'options <https://www.iana.org/assignments/dhcpv6-parameters/dhcpv6-parameters.xml> aux requêtes et réponses DHCPv6 permettent de fournir toutes sortes d'informations utiles. La liste de ces options n'est pas figée, on peut en définir de nouvelles (sections 21 et 24 du RFC 8415). Ce nouveau RFC guide les concepteurs d'options (et leurs collègues à l'IETF qui examineront les nouvelles demandes d'options) : que faire lorsqu'on crée une nouvelle option DHCPv6 et surtout que faut-il ne pas faire. Il s'adresse donc surtout à un public de normalisateurs et de programmeurs.

Pour les programmeurs, la question (section 2 du RFC) est « qu'est-ce que la nouvelle option va nécessiter comme travail ? Un ajout trivial aux clients et serveurs DHCP ou au contraire de vrais changements ? » Dans le second cas, les chances de déploiement de la nouvelle option sont minces... (Aussi bien en DHCPv4 pour IPv4 qu'en DHCPv6, des tas d'options ont été normalisées mais jamais réellement déployées dans la nature, ou parfois au bout de plusieurs années seulement.) En effet, la nouvelle option ne nécessite pas que des changements dans le message tel qu'il circule sur le réseau : il faut changer toute la chaîne de traitement, par exemple l'interface de configuration du serveur. Si ce dernier est configuré par un fichier de configuration, il faudra modifier l'analyseur pour reconnaître la définition d'une valeur dans cette option. S'il est configuré par un GUI, il faudra ajouter menus et boîtes de dialogues. Ensuite, il faudra que les administrateurs réseaux utilisent cette option, ce qui plaide en faveur d'options simples, ne nécessitant pas de lire des tonnes de documentation pour être configurée.

Donc, l'option qui a le plus de chances d'être adoptée, est celle qui ne nécessite que peu ou pas de changement dans les fichiers de configuration, celle qui ne nécessite pas trop de changement (voire pas

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc8415.txt

du tout, en utilisant les capacités de traitement d'options inconnues) dans le code source, celle qui est proche d'options existantes, et, évidemment, celle qui est utile dans une grande gamme de cas.

La section 3 du RFC rappelle dans quel cas il est intéressant d'utiliser DHCPv6. Contrairement à IPv4 où il n'y a guère le choix (pas de mécanisme d'auto-configuration général), IPv6 a deux méthodes de configuration automatique, SLAAC (RFC 4862) et DHCPv6 (RFC 8415). Tout (« "*Any knob, dial, slider, or checkbox on the client system*" », dit le RFC) peut être configuré par DHCP, de l'adresse IP du client, au nom de domaine qu'il utilisera par défaut, en passant par la température à laquelle il doit automatiquement s'arrêter (ce dernier exemple n'est pas normalisé). Comme DHCP est une configuration centralisée, sous le contrôle de l'administrateur réseaux du réseau local où la machine est actuellement attachée, le client n'a pas forcément envie de tout laisser configurer. DHCP est utile pour les paramètres qui dépendent étroitement du réseau actuel (l'adresse IP étant l'exemple typique) mais pas pour les paramètres où le client veut garder le contrôle.

Enfin, la section 3 rappelle que le client a le dernier mot : il peut toujours ignorer les options reçues en DHCP ou les remplacer par les siennes. Par contre, cela peut l'empêcher, notamment dans les réseaux d'entreprise, d'accéder à certains services. Par exemple, si le client DHCP ignore le résolveur DNS indiqué, il pourra rater certains noms qui n'existent que dans une vue DNS locale.

Bon, assez de préliminaires : la section 4 liste les principes généraux qui président à la définition de nouvelles options. Principe n° 1 : la réutilisation. La nouvelle option doit réutiliser autant que possible ce qui existe déjà. Ainsi, si une option prend comme valeur (prenons un exemple irréaliste) un ISBN, il est peu probable que le code pour analyser des ISBN dans le fichier de configuration existe. En revanche, si elle prend comme valeur une adresse IP, le code existe certainement car des tas d'options ont déjà besoin d'une adresse IP.

Il y a deux sortes d'options : les simples qui se contentent de transporter des données du serveur vers le client et qui peuvent donc être gérées par du code générique. Et les compliquées, qui nécessitent un traitement spécifique par le serveur (comme l'option du RFC 4704), avant d'envoyer les données. Inutile de dire que les premières seront plus... simples à déployer.

La réutilisation est surtout importante dans les formats. Si l'option n'utilise que des éléments déjà connus (comme les adresses IP citées plus haut), elles seront bien plus simples à ajouter aux serveurs et aux clients, le code étant déjà là. Bien qu'il puisse être nécessaire de définir des nouveaux formats, notre RFC recommande fortement de construire plutôt à partir des briques existantes. La section 5 détaille ce point. Car, pour le concepteur d'une nouvelle option, fouiller tous les RFC définissant une option DHCPv6, pour voir ce qui serait réutilisable, serait très long. Notre RFC propose donc une liste d'éléments (de briques) courants. Si on a besoin de structures un peu compliquées, on pourra toujours les construire à partir de ces briques.

Premier élément commun et courant, l'adresse IPv6. Des tas d'options en incluent une (liste de serveurs SIP - RFC 3319, résolveurs DNS - RFC 3646, serveur NTP - RFC 5908, etc). Il suffit donc, lorsqu'on définit une option qui distribue (entre autres) des adresses IPv6 de reprendre la définition (l'encodage est résumé dans la section 5.1)... et le code marchera tout seul.

Si on veut un préfixe IPv6 (adresse + longueur du préfixe), cela existe également déjà (quoique ce soit bien plus rare, cela ne se trouve pour l'instant pas encore dans des RFC).

Plus court qu'une adresse IPv6, le booléen. Il existe dans au moins une option (le "*rapid commit*" du RFC 8415, section 22.14). Plutôt que de le représenter sous forme d'un octet (un bit pour la valeur et sept pour le remplissage), le RFC recommande plutôt que la seule présence ou absence de l'option indique la

valeur. On définit une valeur par défaut et, si l'option est présente, sa valeur sera l'opposé de la valeur par défaut. Cela permet d'encoder uniquement le code et la longueur (forcément zéro mais DHCPv6 impose la présence de ce champ Longueur), sans aucun contenu. Le nom de l'option doit refléter l'effet qu'aura sa présence. Par exemple, s'il existe un booléen `FOO` et que sa valeur par défaut est Vrai, l'option devra s'appeler `DISABLE_FOO` puisque sa présence mettra `FOO` à Faux.

Plus grands que les booléens, les entiers, comme le "*refresh time*" du RFC 4242. Si on veut transmettre un entier de 32 bits, on peut réutiliser ce format.

Plus grand encore, et plus compliqué, un URI. Par exemple, le RFC 5970 fournit une option qui permet d'indiquer les coordonnées du code à charger au démarrage d'une machine. Ces coordonnées sont un URI, par exemple `ftp://[2001:db8:23a::1]/bootfiles/pi.exe`. Sur le câble, cette option est encodée exactement comme du texte libre (exemple suivant) mais le fait de la marquer comme contenant un URI permet au serveur DHCP de tester sa syntaxe avant de charger cette option.

Ah, le texte, justement. Le RFC recommande que toutes les options utilisant du texte libre imposent à celui-ci d'être en Unicode, avec les restrictions du RFC 5198, et encodé en UTF-8. (Rappelez-vous que du texte en pur ASCII remplit ces conditions.) Le RFC 4833, qui définit une option en texte libre pour indiquer le fuseau horaire, sans le dire clairement, se limite au seul ASCII et, donc, si on créait une option prenant du texte libre en Unicode, il faudrait probablement développer du code spécifique, personne ne l'ayant encore fait.

Certaines options prennent comme valeurs des listes et pas de simples scalaires. C'est le cas d'options comme la liste de serveurs SIP déjà vue plus haut.

À part la réutilisation des formats, que conseille notre RFC ? La section 7 recommande d'éviter les alias, ces options qui donnent accès au même paramètre mais via des formats différents. Par exemple, si on veut indiquer un serveur réseau, avoir deux options, une prenant en paramètre une adresse IP et l'autre un nom de domaine, est fortement déconseillé. Cela augmente la quantité de code et laisse dans l'ambiguité la question de savoir quelle option gagne si les deux sont présentes. À la place, il faut une seule option, le type de paramètre dépendant des propriétés que l'on veut (ici, la simplicité pour l'adresse IP, la liaison retardée pour le nom de domaine).

La section 8 brode davantage sur ce thème du choix entre nom de domaine et adresse IP. S'il est fortement conseillé de n'avoir qu'une option, laquelle ? Le RFC évite de faire une recommandation générale, car cela dépend du paramètre. Par exemple, le résolveur DNS doit évidemment être indiqué par une adresse IP. Les paramètres qui sont liés à la topologie du réseau (les paramètres « couche 3 » comme le serveur de mobilité) sont sans doute mieux représentés par des adresses IP. Ceux qui sont plus proches des applications (les paramètres « couche 7 » comme un serveur SIP) ont plutôt intérêt à utiliser des noms de domaines, voire des URI.

Il y a d'autres points à prendre en compte. Par exemple, les applications réagissent en général mieux aux problèmes DNS temporaires qu'aux problèmes réseau. Un nom de domaine sera donc intéressant si on veut éviter qu'une panne plante l'application. Le problème de résolution DNS est attendu et, en pratique, souvent bien géré, ce qui n'est pas le cas si, par exemple, un client DHCP reçoit une adresse IP qui, en pratique, ne marche pas : ce cas n'est pas prévu et le client ne réessaiera pas automatiquement.

L'indirection supplémentaire fournie par le nom de domaine est, la plupart du temps, un avantage. Un exemple où, au contraire, c'est un inconvénient, est celui où on veut fournir un service qui dépend de la localisation. Le DNS est bâti sur le principe qu'une question entrainera la même réponse partout. Il existe des services DNS géo-localisés mais qui, en pratique, ne sont pas très satisfaisants puisqu'ils

violent ce principe. Au contraire, le serveur DHCP sait forcément où est connecté le client (sinon, il ne pourrait pas choisir des paramètres comme l'adresse IP) et est donc bien placé pour renvoyer une adresse IP qui dépend de la localisation.

Autre question soulevée par ce choix entre adresse IP et nom de domaine : le temps qu'il faudra pour qu'un changement dans la configuration soit pris en compte. Si le serveur DHCP renvoie une adresse IP, et décide ensuite de la changer, il ne peut pas garantir que les clients la recevront immédiatement. Certes, il existe une option `reconfigure` dans DHCPv6 (RFC 8415, sections 16.11, 18.2.11 et 18.3.11), permettant au serveur des notifications non sollicitées, mais elle n'est pas obligatoire et il semble que bien des clients ne la gèrent pas. En revanche, si le serveur avait renvoyé un nom de domaine, et si l'application pense à le redemander souvent (un gros Si...), la nouvelle valeur sera effective au bout de N secondes où N est le TTL choisi par le serveur DNS. Le choix dépend donc de pas mal de paramètres (par exemple, est-ce que l'administrateur du serveur DHCP contrôle aussi le serveur DNS et peut choisir le TTL à volonté ?) que le concepteur de l'option ne peut hélas pas connaître à l'avance.

Autre argument de prudence avant d'utiliser des noms de domaine : il crée une dépendance envers le DNS. Il faut donc s'assurer que le serveur envoie bien la liste des résolveurs DNS, et que ceux-ci fonctionnent correctement.

Une caractéristique de DHCPv6 est le fait que les options peuvent être incluses dans d'autres options. Cela permet de faire des options spécifiques à certains cas, du genre une option Y qui n'a de sens que si on a aussi accepté l'option X. On encapsule alors Y dans X. Il y a donc des options "*top-level*" et des options encapsulées. Il y a aussi des sous-options, qui n'ont pas de numéro d'option propre mais sont numérotées par rapport au numéro de l'option qui les contient. Les numéros d'options dans DHCPv6 sont codés sur 16 bits (8 bits pour DHCPv4) et il y a donc abondance de numéros disponibles. Le RFC conseille donc de ne pas chercher à les économiser et recommande les options encapsulées plutôt que les sous-options. Plus rigolo, l'encapsulation peut être à plus de deux niveaux : c'est le cas de la délégation de préfixe du RFC 8415, où le préfixe peut contenir une option d'exclusion qui elle-même contient une option préfixe, indiquant le préfixe exclu.

DHCP (v4 ou v6) est un mécanisme à état : le serveur DHCP se souvient de ce qu'il a fait et en tient compte pour les réponses suivantes. L'exemple le plus classique concerne l'allocation d'adresses IP : le serveur doit se souvenir de quelles adresses ont été allouées, pour ne pas allouer deux fois la même. Toutefois, dès qu'il faut maintenir un état supplémentaire, en général pour donner des réponses différentes par client, on augmente la complexité du serveur. Le RFC recommande donc d'éviter de définir de nouvelles options imposant le maintien d'un état. On peut souvent obtenir le même effet en laissant le client générer une valeur unique, à partir de son adresse IP et d'informations générales.

J'ai déjà parlé plus haut du fait que les clients DHCP ne vont pas magiquement chercher des nouveaux paramètres dès que le serveur DHCP a été reconfiguré. Le mécanisme `reconfigure` ne va pas forcément marcher avec tous les clients et l'administrateur du serveur DHCP doit donc être prêt à voir des clients continuer à utiliser des vieux paramètres. Cela peut être important aussi pour le concepteur d'options, au cas où il espérerait des options dont les valeurs changent souvent (section 11).

Un autre cas rigolo est celui où il y a plusieurs serveurs DHCP sur le même réseau. Cela n'est pas forcément le résultat d'une erreur ou d'un piratage. Par exemple, si un réseau à la maison est connecté par deux FAI, chacun apportant sa "*box*", deux serveurs DHCP répondront aux requêtes des clients. Le protocole DHCP ne fournit pas de mécanisme pour gérer cela, juste quelques conseils dans le RFC 8415, conseils qui ne semblent pas suivis par les implémentations actuelles. En pratique, bien des clients n'écoutent que le premier serveur qui leur a répondu (section 12 de notre RFC). Ils n'essaient pas de fusionner intelligemment les différentes sources. Par exemple, il serait logique d'utiliser le résolveur DNS du FAI 1 lorsqu'on va utiliser les adresses IP, et la ligne, de ce FAI, et le résolveur du FAI 2 dans

l'autre cas. Mais ce n'est pas ce qui se passe et, aujourd'hui, le cas de la machine à plusieurs connexions réseau est toujours assez mal géré. C'est un problème général, qui n'est pas soluble dans le cadre d'une option DHCP particulière (voir les RFC du groupe de travail MIF </search?pattern=mif>).

En DHCP, la taille compte. C'est ce qu'explique la section 15, qui note que les paquets DHCPv6 ont une taille maximale de 64 ko. Contrairement à DHCPv4, il n'y a pas de trucs utilisés en couche 2, DHCPv6 est purement une application tournant sur UDP sur IPv6, avec des adresses locales au lien. Bon, c'est la théorie. En pratique, si on bourre un paquet d'options de grande taille, peut-on vraiment aller jusqu'à 64 ko ? Le RFC 8415 est plus prudent et dit « "*The server must be aware of the recommendations on packet sizes and the use of fragmentation as discussed in Section 5 of .*" ». Il ne semble pas qu'aucune mise en œuvre de DHCPv6 ait câblé en dur des limites de taille inférieures aux 64 ko légaux. Donc, ça devrait marcher mais, dans la réalité, personne n'a encore testé des réponses DHCP supérieures aux 1 280 octets qui sont le miminum qu'IPv6 doit accepter ou aux 1 500 octets qui sont la MTU d'Ethernet. Notre RFC suggère que, si on veut vraiment envoyer des grandes quantités de données, une option DHCP n'est pas la meilleure solution et qu'il vaut mieux envoyer au client un URL indiquant où chercher ces données.

Pas encore épuisé par tous les points qu'il faut garder en tête en définissant une nouvelle option DHCP ? C'est que le RFC n'est pas terminé. Une option peut-elle apparaître plusieurs fois dans une réponse DHCP (section 16) ? Par défaut, non, mais c'est possible si et seulement si c'est indiqué explicitement dans la spécification de l'option. Attention, l'ordre d'apparition n'est pas significatif (section 17). Un client peut mettre les options dans une structure de données qui ne préserve pas l'ordre (un dictionnaire, par exemple).

Autre piège lorsqu'il y a plusieurs options : il faut spécifier la sémantique des données ainsi récupérées. Ainsi, l'option `AFTR` du RFC 6334 (utilisée pour DS-Lite) est un singleton, une option à apparition unique. Il avait été envisagé de permettre plusieurs occurrences de l'option, permettant d'indiquer plusieurs AFTR (l'extrémité du tunnel DS-Lite auquel on se connecte, cf. RFC 6333). Mais comment le client devait-il choisir parmi eux ? Une répartition de charge entre les AFTR était-elle possible ? Et une bascule automatique d'un AFTR à l'autre en cas de panne ? Spécifier tout cela aurait été trop compliqué et l'option est donc restée un singleton.

Les messages DHCPv6 peuvent être relayés afin d'atteindre des réseaux où il n'y a pas de serveur DHCP. Contrairement à DHCPv4, les options spécifiques aux relais sont des options ordinaires, prises dans le même espace de numérotation que les autres (section 18 de notre RFC).

Au fait, comment un serveur sait-il quelles sont les options acceptées et gérées par le client ? Il existe une option `ORO` ("*Option Request Option*"), normalisé dans le RFC 8415, pour que le client indique ses capacités, à la fois ce qu'il acceptera, et ce qu'il utilisera réellement. Elle n'est pas obligatoire et ne représente donc pas un vrai mécanisme de négociation des options.

Et les innombrables techniques de transition <https://www.bortzmeyer.org/transition-ipv6-guilde.html> d'IPv4 vers IPv6 ? Elles nécessitent sûrement des options DHCPv6, non (section 20) ? En fait, non. Notre RFC recommande de les transporter via DHCPv4 puisqu'elles deviendront inutiles lorsque IPv4 disparaitra (ah, l'optimisme...) et elles ne doivent donc pas encombrer l'espace des options DHCPv6.

Arrivés à ce point, les auteurs de nouvelles options DHCPv6 doivent se demander s'ils arriveront un jour à compléter le futur RFC décrivant leur option chérie. La section 21 est là pour les aider en fournissant une liste de points à couvrir, avec un exemple de texte pour chacun. Le RFC recommande trois sous-sections, une pour les clients DHCP, une pour les serveurs et une pour les relais (pour ces derniers, la plupart du temps, il n'y aura rien à faire, un relais doit relayer aveuglément les options qu'il ne connait pas). Par exemple, pour les clients, le RFC suggère de commencer par « "*Clients MAY request*

option foo, as defined in [RFC3315], sections 17.1.1, 18.1.1, 18.1.3, 18.1.4, 18.1.5 and 22.7. As a convenience to the reader, we mention here that the client includes requested option codes in Option Request Option." ≫... Plus qu'à copier/coller en mettant le vrai nom de l'option à la place.

Autre question bureaucratique, le RFC décrivant la nouvelle option doit-il indiquer dans ses métadonnées qu'il met à jour les RFC existants comme le RFC 8415 ? Non, répond la section 22. Il y a aujourd'hui environ 80 options définies et si toutes mettaient à jour le RFC 8415, l'implémenteur de DHCPv6 en aurait, des documents à lire ! Donc, la plupart des RFC définissant des options ne changeront pas les RFC existants. Par contre, s'il est nécessaire de remplacer une partie de la norme existante, alors, oui, on doit indiquer cette mise à jour. C'est le cas du RFC 6644 qui est bien décrit ainsi dans l'index des RFC ≪ "*6644 Rebind Capability in DHCPv6 Reconfigure Messages. D. Evans, R. Droms, S. Jiang. July 2012. (Format : TXT=19176 bytes) (Updates RFC3315) (Status : PROPOSED STANDARD)*" ≫.

Un peu de sécurité et de vie privée pour terminer. La section 23 rappelle aux auteurs d'options l'importance de la sécurité. DHCP est bien connu pour son absence totale d'authentification, qui fait qu'un serveur ≪ pirate ≫ peut facilement se glisser à la place d'un vrai et servir n'importe quoi. Il existe pourtant en théorie un mécanisme d'authentification (section 21 du RFC 3315) mais soyons francs : presque personne ne l'a programmé et quasiment personne ne l'a déployé. Bref, les concepteurs d'options doivent être conscients que les options seront transmises en clair et, pire, qu'un méchant pourra envoyer des fausses valeurs.

Autre problème de sécurité, la longueur des options. Si elle est de taille fixe, le client qui la lit devra vérifier que la longueur réelle est bien la longueur théorique (autrement, il risque un débordement de tampon). Si elle est de taille variable (un URI, par exemple), le lecteur de l'option doit se méfier, l'option peut être très longue... Des valeurs ayant la taille normale peuvent néanmoins donner de drôles de résultats. Par exemple, une option contenant une adresse IP peut indiquer, si le serveur est malveillant ou bien s'il a été doublé par un serveur pirate, une adresse de diffusion, pour pousser le client à arroser tout le réseau...

À noter que le RFC ne rappelle pas une spécificité de DHCPv6 : il est courant qu'un réseau IPv6 n'ait pas de serveur DHCP légitime du tout, l'administrateur réseaux préférant la configuration SLAAC. Dans ce cas, un serveur pirate aurait la partie très facile, n'ayant pas à faire la course avec le serveur légitime.

Enfin, il reste la question de la vie privée (section 24). Les messages DHCP sont transmis quasiment toujours en clair et il ne faut donc pas compter sur leur confidentialité. En théorie, on pourrait utiliser IPsec pour les chiffrer mais personne ne le fait.