RFC 7556 : Multiple Provisioning Domain Architecture

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 16 juin 2015

Date de publication du RFC : Juin 2015

https://www.bortzmeyer.org/7556.html

Ce RFC décrit une architecture nouvelle permettant de mieux gérer les machines ayant plusieurs connexions à l'Internet. On ne parle pas ici de gros routeurs ou serveurs connectés à plusieurs réseaux, non, un simple "*smartphone*" qui a à la fois le WiFi et la 4G est un exemple d'une machine qui a plusieurs interfaces. Et cela pose des sérieux problèmes avec l'Internet d'aujourd'hui, car, parfois, chaque interface va nécessiter des réglages différents. Notre nouvelle architecture aborde ce problème en définissant le **PvD** ("*ProVisioning Domain*" ou domaine d'avitaillement), un ensemble cohérent de réglages, qui sera attaché à chaque interface.

Pourquoi est-ce un problème d'avoir plusieurs interfaces actives ? Car certains réglages ne sont valables que pour une seule interface, alors que TCP/IP considérait au début que toutes les interfaces se valaient et pouvaient être utilisées. Par exemple, l'adresse IP obtenue sur une interface ne doit pas être utilisée comme adresse IP source sur une autre interface, de peur de se faire mordre par le RFC 2827 [1]. Le résolveur DNS obtenu sur une interface ne doit pas être utilisé pour une autre car il refusera probablement de répondre (RFC 5358). Et, même si les résolveurs DNS acceptaient tous les clients, il arrive qu'ils renvoient des données qui ne sont pas les mêmes pour tout le monde (par exemple pour un serveur qui n'est accessible que depuis un certain réseau). Le RFC 6418 décrit plus en détail le problème des interfaces multiples, et le RFC 6419 proposait déjà quelques solutions, chacune spécifique à une plate-forme donnée. Pour résumer (à l'extrême) le RFC 6418, les risques sont :

— Incohérence entre les informations obtenues depuis les différentes interfaces (comme dans l'exemple DNS ci-dessus),
— Risque de mélanger les informations obtenues sur différentes interfaces (tenter de résoudre le nom d'un relais avec le serveur DNS d'un autre opérateur...),
— Usage non conforme à la politique d'un des réseaux (par exemple, téléchargement d'un jeu via le VPN de l'entreprise...)

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc2827.txt

Donc, pour résoudre ce genre de problème, on introduit le concept de PvD. Mais c'est quoi, un PvD (section 2) ? Un "*ProVisioning Domain*" est simplement un ensemble cohérent d'informations de configuration d'un réseau, pour une interface donnée. Cela comprend par exemple l'adresse IP à utiliser (ou le préfixe IP), les adresses IP des résolveurs DNS, le routeur par défaut, le "*proxy*" Web à utiliser, etc. Une machine "*PvD-aware*" sait gérer cette information (notamment en l'association à une interface et en gardant en mémoire que, par exemple, ce résolveur DNS est fait pour cette interface). Une application "*PvD-aware*" a du code pour gérer la coexistence de différents PvD.

Aujourd'hui, les PvD sont **implicites**. La machine connectée reçoit de l'information de configuration sur chaque interface, via des processus divers (DHCP, mais pas seulement), et cette information n'est pas groupée, sous l'étiquette d'un PvD donné. C'est à la machine "*PvD-aware*" de gérer une table de PvD en les nommant et en les associant à chaque interface. Par défaut, avec les PvD implicites, une interface = un PvD.

Mais il y aura plus tard des PvD **explicites**. Cette distinction entre PvD implicites et explicites est essentielle. L'idée est de développer via l'IETF des mécanismes (comme celui du RFC 8801) pour communiquer à la machine des PvD explicites, regroupant sous un nom donné un jeu d'informations de configuration cohérentes. Par exemple, on peut imaginer une option DHCP qui indique le nom du PvD correspondant aux options de configuration contenues dans cette réponse DHCP (ce n'est qu'une hypothèse : le RFC liste d'autres possibilités, cf. section 3). Avec les PvD explicites, il n'y aura plus forcément une interface = un PvD. On pourra avoir plusieurs interfaces utilisant le même PvD (si elles sont gérées par la même organisation).

Cela impliquera un système (non encore défini) de nommage des PvD. Le « PvD ID » devra être unique (puisque le but est de différencier les informations de configuration de réseaux différents), par exemple en le tirant au hasard dans un espace de grande taille. Le PvD ID ne sera donc pas forcément lisible par un humain ; si on veut une telle lisibilité, il faudra développer une solution, par exemple de métadonnées associées au PvD.

Un PvD peut être commun à plusieurs familles IP (par exemple IPv4 et IPv6) et comporter donc plusieurs adresses. C'est utile si certaines options ont un sens pour plusieurs familles (c'est le cas de la politique de sélection de l'adresse du RFC 7078). Pour les PvD implicites, utiliser un seul PvD pour toutes les familles présentes sur l'interface est la politique recommandée.

La section 3 examine quelques protocoles existants pour voir comment ils pourraient être modifiés pour transporter de l'information explicite sur les PvD. Aujourd'hui, la configuration des machines terminales est faite en général par DHCP (RFC 8415) ou RA (RFC 3971). Notons qu'elles ne sont pas sécurisées par défaut. Des solutions existent pour authentifier la machine qui émet la réponse DHCP ou RA mais authentification n'est pas autorisation : elle ne nous dit pas si cette machine est autorisée à annoncer explicitement tel PvD. Pour cela, il faudrait un moyen de signer les PvD.

Évidemment, si on ajoute des mécanismes de PvD à ces protocoles, cela devra être fait d'une manière compatible avec les machines existantes, qui ne sont pas "*PvD-aware*". Pendant la phase de coexistence, il faudra peut-être dupliquer de l'information dans les réponses DHCP ou RA, pour satisfaire tout le monde.

La section 4 du RFC présente trois études de cas concrètes. La première est celle d'une machine mobile, un "*smartphone*", par exemple. Il a une interface avec un réseau mobile, mettons 3G, et une interface WiFi. S'il est dehors, loin de tout "*hotspot*", il n'aura qu'une interface active et donc un seul PvD : pas de difficultés. S'il s'approche d'un point d'accès WiFi et s'y connecte, le dilemme commence : quelle interface utiliser ? Android, par exemple, ne peut en utiliser qu'une à la fois et va donc cesser

d'utiliser la 3G. Cela simplifie le modèle de connexion pour les développeurs mais n'est pas satisfaisant pour l'utilisateur. Il serait plus satisfaisant de pouvoir utiliser les deux, mais en respectant à chaque fois le PvD (qui peut être différent) de chaque interface.

Autre étude de cas, un VPN, qui peut être considéré comme une interface supplémentaire, quoique virtuelle, et un réseau à la maison, connecté à la fois à un FAI généraliste classique et à un service de VoD, avec une interface séparée qu'on n'utilise que pour la vidéo, et qui est donc décrite par un PvD différent.

La section 5 de notre RFC étudie plus en détail certains points spécifiques qui doivent être traités par un nœud "*PvD-aware*". Par exemple, la résolution de noms, déjà citée plus haut, qui dépend fortement du PvD : un résolveur DNS n'est typiquement utilisable que par certains clients, et pas par le monde entier. En outre, les résultats d'une résolution de noms peuvent varier selon le résolveur interrogé (cas d'un service spécifique à un FAI, par exemple le MMS). La machine doit donc savoir quel PvD utiliser pour chaque nom, avec des règles du genre « pour tous les noms se terminant en `ma-boîte.example`, se servir du résolveur DNS du PvD du VPN de ma boîte ». Si plusieurs PvD correspondent à des FAI généralistes, pouvant servir pour tout accès à un serveur public, la machine "*PvD-aware*" peut faire les résolutions de noms en parallèle, et prendre le plus rapide, selon un approche identique à celle des globes oculaires heureux du RFC 6555.

Et puisqu'on parle de sélectionner le « meilleur » PvD, il faut rappeler que la connectivité au réseau local, tel qu'elle s'obtient lorsqu'une requête DHCP ou RA a réussi, ne garantit pas un bon accès à tout l'Internet. Le signal local peut être fort, le serveur DHCP très réactif... et la connexion ultérieure échouer. Il est donc prudent de tester les connexions obtenues, par exemple en essayant de se connecter à un amer <`https://www.bortzmeyer.org/amer-mire.html`> bien connu.

On a vu qu'une application pouvait être "*PvD-aware*" si elle sait utiliser l'information sur les PvD pour prendre des décisions. La section 6 décrit les propriétés attendues de l'API qui permettra de développer ces applications. Dans le cas le plus simple, une application "*PvD-aware*" peut se limiter à n'utiliser qu'un seul PvD. C'est le cas d'une application MMS qui dépend d'un opérateur particulier et qui devra donc passer par l'interface correspondante. Un peu plus complexe serait une application qui exprime des exigences et des préférences, du genre « il me faut IPv6 » ou « je fais de la vidéo-conférence, je veux une faible latence <`https://www.bortzmeyer.org/latence.html`> » ou bien « je suis paranoïaque, je ne veux passer que par un PvD cité dans telle liste » ou encore « je ne suis pas pressé, il vaut mieux utiliser la connexion la moins chère ». Le système pourrait alors choisir, en regardant les informations obtenues pour chaque PvD, quel PvD utiliser. Dans le dernier cas (une application qui télécharge beaucoup, comme la mise à jour du système d'exploitation, ou comme une application pair-à-pair d'accès à la culture), un mécanisme de sélection automatique via le PvD simplifierait nettement la tâche de l'utilisateur en le dispensant d'options comme la « n'utiliser qu'avec la WiFi » qu'on trouve souvent sur les applications Android, qui veulent éviter de faire exploser le forfait de téléphonie mobile.

Enfin, les applications les plus avancées pourraient simplement obtenir la liste des PvD et leurs caractéristiques et faire leurs choix elles-mêmes. Ces choix ne seront pas forcément respectés par le système, qui peut avoir ses propres exigences (par exemple forcer la passage par le VPN de l'entreprise pour les connexions aux services de la dite entreprise).

Puisqu'on a parlé de sécurité (« je suis paranoïaque, je ne veux passer que par un PvD cité dans telle liste »), un mot sur les PvD de confiance ou pas (section 7). Par défaut, un PvD n'est pas « de confiance ». Si je me promène dans la rue et que je me connecte à un "*hotspot*" inconnu, le PvD associé ne sera pas considéré comme de confiance. Certains systèmes ou applications ne voudront se connecter à tel service que via un PvD de confiance. Mais attention : supposons qu'on ait une telle

liste (le PvD du VPN de la boîte, le PvD d'un opérateur à qui on fait confiance, le réseau local à la maison...), il ne faut pas pour autant considérer que le PvD est ce qu'il annonce. Si le PvD de l'entreprise est `6a97f31e-34ec-4ef9-ba75-b1e3addd8222` (ce n'est pas la vraie syntaxe des PvD ID, qui n'est pas encore décidée), et que le serveur DHCP du "*hotspot*" annonce ce PvD ID, il ne faut pas pour autant lui faire confiance (voir aussi la section 8). Un PvD de confiance nécessite deux choses : une relation de confiance avec son opérateur, et un moyen de vérifier qu'il s'agit bien du PvD attendu. Ce moyen d'authentification peut passer par une signature du PvD ID, ou bien simplement par le mécanisme d'attachement au réseau : on peut décider que, si on est connecté à un réseau physique stable et sûr, l'annonce du PvD ID sur ce réseau peut être crue. À noter que le RFC fournit un autre exemple, où on fait confiance au réseau d'un opérateur mobile, ce qui ignore le risque des "*IMSI-catchers*".

La section 8 revient d'ailleurs sur les problèmes de sécurité. Par exemple, même si le serveur DHCP ou RA est sûr, un attaquant a pu modifier les données en cours de route. On peut utiliser les techniques (très peu déployées en pratique) de l'option `AUTH` du RFC 3315 (abandonnée depuis, dans le RFC 8415), section 22.11, ou le SEND du RFC 3971.