RFC 6419 : Current Practices for Multiple Interface Hosts

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 22 novembre 2011

Date de publication du RFC : Novembre 2011

https://www.bortzmeyer.org/6419.html

Il fut une époque où la machine terminale connectée à l'Internet n'avait typiquement qu'une seule interface réseau. Seuls les routeurs avaient plusieurs interfaces et donc des décisions difficiles à prendre comme « quelle adresse source choisir pour le paquet que je génère ? » Aujourd'hui, les machines les plus ordinaires ont souvent, à leur tour, plusieurs interfaces réseau. Mon "*smartphone*" a une interface WiFi et une 3G. Mon ordinateur portable a une interface filaire, une 3G, une Wifi et une Bluetooth. Qui dit plusieurs interfaces dit décisions à prendre. Quel serveur DNS utiliser ? Celui appris via l'interface filaire ou bien celui appris en Wifi ? Par quelle interface envoyer un paquet qui sort de la machine ? La question est d'autant plus difficile que les différentes interfaces fournissent en général des services très différents, en terme de qualité et de prix. Le groupe de travail MIF <http://tools.ietf.org/wg/mif> de l'IETF travaille à normaliser le comportement des machines sur ce point. Son premier RFC (un autre, le RFC 6418 [1], concerne l'exposé du problème) commence par examiner l'état de l'art : que font aujourd'hui les divers systèmes d'exploitation confrontés à cette question ?

Le RFC ne prétend pas couvrir tous les systèmes. Il se contente de ceux que les membres du groupe de travail connaissaient, ceux sur lesquels ils avaient de la documentation et des informations : Nokia S60, Windows Mobile, Blackberry, Android, Windows, et des systèmes à base de Linux.

Comment ces systèmes gèrent cette multiplicité des interfaces réseaux, se demande la section 2. Une première approche est de centraliser la gestion des connexions : ce n'est pas l'application qui choisit l'interface à utiliser, elle passe par un gestionnaire de connexions (en lui transmettant parfois quelques souhaits) qui choisit pour elle. Le gestionnaire peut choisir en fonction de la table de routages (si l'application veut écrire à 172.23.1.35 et qu'une seule interface a une route vers cette adresse, le choix est vite fait) mais elle ne suffit pas toujours (que faire si deux interfaces sont connectées à un réseau privé

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc6418.txt

`172.23.0.0/16`?). Le gestionnaire doit aussi tenir compte d'autres facteurs. Le coût est souvent le plus important (prix élevés de la 3G, surtout en itinérance, par rapport au Wifi ou au filaire, par exemple).

Une autre solution est que l'application choisisse elle-même, en fonction des informations qu'elle reçoit du système. Certaines applications ont des idées bien précises sur la connexion qu'elles veulent utiliser. Par exemple, sur Android, la plupart des clients SIP refusent par défaut d'utiliser les connexions 3G, les opérateurs y interdisant en général le trafix voix (en violation de la neutralité du réseau <`https://www.bortzmeyer.org/neutralite.html`>). Les deux méthodes (gestionnaire de connexions centralisé et choix par l'application) ne sont pas complètement exclusives. Comme indiqué, on peut aussi avoir un gestionnaire centralisé, mais qui permet aux applications d'indiquer des préférences (« essaie d'éviter la 3G »).

Certains choix sont par contre typiquement réglés par une troisième approche : le faire entièrement dans le système, sans demander à l'utilisateur ou aux applications (section 2.3). Le système rassemble des informations de sources diverses (DHCP, RA ("*Router Advertisment*"), configuration manuelle par l'administrateur système) et fait ensuite des choix qui sont globaux au système, ou bien spécifiques à chaque interface. Il n'y a aucune règle commune entre les différents systèmes d'exploitation sur ce plan, et aucun ne semble fournir une solution parfaite.

Par exemple, la configuration DNS voit certains systèmes permettre un jeu de résolveurs DNS différent par interface, avec des règles indiquant quel jeu est choisi. D'autres ont au contraire un seul jeu de résolveurs.

La sélection de l'adresse source est plus « standard ». L'application fournit l'adresse de destination et, lorsqu'elle n'a pas choisi une adresse source explicité (option `-b` de OpenSSH, par exemple), le système en sélectionne automatiquement une. Pour IPv6, il existe même une norme pour cela, le RFC 6724 (et son prédécesseur, le RFC 3484, qui est encore la référence). Un grand nombre de systèmes d'exploitation ont adapté cette norme à IPv4 et l'utilisent donc pour toutes les familles d'adresses. En revanche, tous ne fournissent pas forcément un mécanisme permettant à l'administrateur système d'influencer cette sélection (le `/etc/gai.conf` sur Linux). Autre point où les systèmes diffèrent, certains choisissent l'interface de sortie avant l'adresse source, d'autres après.

La section 3 est ensuite l'exposé des pratiques des différents systèmes, à l'heure actuelle (une version future pourra changer leur comportement). Ainsi, le système S60 (qui tourne sur Symbian) introduit le concept d'IAP ("*Internet Access Point*") ; un IAP rassemble toutes les informations sur une interface (physique ou virtuelle, dans le cas des tunnels). La machine a en général plusieurs IAP et l'application choisit une IAP au moment où elle a besoin du réseau. Ce choix est motivé par le fait que toutes les IAP ne fournissent pas forcément le service attendu par l'application. Par exemple, l'envoi d'un MMS nécessite l'usage d'un IAP qui donne accès à la passerelle MMS de l'opérateur, qui n'est en général pas joignable depuis l'Internet public. De même, une application qui accède aux serveurs privés d'une entreprise va choisir l'IAP qui correspond au VPN vers l'entreprise.

Ce concept d'IAP permet d'éviter bien des questions. Ainsi, le serveur DNS est indiqué dans l'IAP et, une fois celle-ci choisie, il n'y a pas à se demander quel serveur utiliser. (Reprenez l'exemple du VPN : cela garantira que l'application "*corporate*" utilisera un serveur DNS sûr et pas celui du "*hotspot*" où le téléphone est actuellement connecté.) De même, il existe une route par défaut par IAP et ce n'est donc pas un problème si deux interfaces de la même machine ont des plages d'adresses RFC 1918 qui se recouvrent (ou même si les deux interfaces ont la même adresse IP).

Comment l'application choisit-elle l'IAP ? Cela peut être fixé en dur dans le programme (cas de l'application MMS, en général), choisi par l'utilisateur au moment de la connexion, ou bien déterminé automatiquement par le système. La documentation de ce système figure en ligne <`http://www.forum.`

nokia.com/info/sw.nokia.com/id/190358c8-7cb1-4be3-9321-f9d6788ecae5/S60_Platform_IP_Bearer_Management_v1_0_en.pdf.html>.

Pour Windows Mobile et Windows Phone 7, tout dépend du "*Connection Manager*". Celui-ci gère les connexions pour le compte des applications, en tenant compte de critères comme le coût ou comme la capacité du lien. Les applications peuvent l'influencer en demandant des caractéristiques particulières (par exemple une capacité minimale). Windows met en œuvre le RFC 3484 et l'administrateur système peut configurer des règles qui remplacent celles par défaut. Ce système est documenté en ligne <http://msdn.microsoft.com/en-us/library/aa457829.aspx>.

Le Blackberry laisse les applications choisir la connexion qu'elles veulent, même s'il fournit un relais pour celles qui le désirent. Cela dépend toutefois de la configuration du réseau (l'utilisation obligatoire du "*Enterprise Server*" peut être choisie, par exemple). Les réglages comme le DNS sont par interface et non pas globaux au système. Plus de détails sont fournis dans la documentation <http://na.blackberry.com/eng/deliverables/5827/Wireless_gateways_447132_11.jsp>.

Et Google Android ? Utilisant un noyau Linux, il partage certaines caractéristiques avec les autres systèmes Linux. Par défaut, il se comporte selon le modèle « machine terminale ouverte » ("*weak host model*", cf. RFC 1122, section 3.3.4.2), mais peut aussi utiliser le modèle « machine strictement terminale » ("*strong host model*"). Dans le premier cas, les différentes interfaces réseaux ne sont pas strictement séparées, un paquet reçu sur une interface est accepté même si l'adresse IP de destination n'est pas celle de cette interface. Dans le second, la machine est stricte et refuse les paquets entrants si l'adresse IP de destination ne correspond pas à l'interface d'entrée du paquet.

La configuration DNS est globale (attention, Android ne la note pas dans `/etc/resolv.conf`, dont le contenu est ignoré). Cela veut dire que, dès qu'une réponse DHCP est reçue sur une interface, la configuration DNS de cette interface remplace la précédente.

Depuis Android 2.2, le RFC 3484 est géré. (Notez que, à l'heure actuelle, Android ne peut pas faire d'IPv6 sur l'interface 3G.)

La documentation figure dans le paquetage `android.net`. Les applications peuvent utiliser la classe `ConnectivityManager` si elles veulent influencer la sélection de l'interface. Pour cela, le système met à leur disposition la liste des interfaces et leurs caractéristiques. (Pour étudier un programme qui l'utilise, vous pouvez par exemple regarder les occurrences de `ConnectivityManager` dans le source Java de CSIPsimple <http://code.google.com/p/csipsimple/source/checkout>.)

Le RFC se penche aussi sur un système moins connu, Arena, qui utilise également Linux et fournit un gestionnaire de connexions pour choisir les interfaces, avec indication des préférences par les applications. Il sera décrit dans un futur RFC.

Et sur les systèmes d'exploitation utilisés sur le bureau ? Windows est présenté mais sera décrit plus en détail dans un autre RFC. Les systèmes utilisant Linux (comme Ubuntu ou Fedora) partagent plusieurs caractéristiques :

— Les informations comme l'adresse IP utilisée sont stockées par interface et sont obtenues par DHCP (ou RA).
— Un certain nombre d'informations, notamment la route par défaut, les serveurs DNS ou les serveurs NTP sont globales au système. Bien que les clients DHCP utilisés, comme celui de l'ISC <http://www.isc.org/software/dhcp>, permettent une configuration différente par interface, des paramètres comme la liste des résolveurs DNS, quelle que soit l'interface d'où ils viennent, effacent et remplacent globalement les paramètres précédemment obtenus.

— La dernière interface configurée est donc forcément l'interface principale.
Les systèmes à base de BSD ont en gros les mêmes caractéristiques. Dans les deux cas, Linux ou BSD, certaines options permettent de passer outre les informations obtenues par DHCP. Par exemple, des "*option modifiers*" dans le client DHCP OpenBSD offrent la possibilité d'indiquer des valeurs à utiliser avant, ou bien à la place de, celles fournies par le serveur DHCP. Même chose pour des options comme `-R` du client DHCP Phystech <http://www.phystech.com/download/dhcpcd.html> (qui indique de ne pas modifier `resolv.conf`) ou comme `nodns` et `nogateway` dans les options de Pump, pour ignorer les résolveurs DNS et la route par défaut de certaines interfaces. D'autres systèmes disposent d'options similaires.

Enfin, même si le RFC ne le mentionne que très rapidement, il existe des systèmes de plus haut niveau (comme Network Manager, très pratique pour les machines qui se déplacent fréquemment) dont la tâche est d'orchestrer tout cela, de surveiller la disponibilité des interfaces, de lancer le client DHCP lorsque c'est nécessaire, d'accepter ou de passer outre ses choix, etc.

Voilà en gros la situation actuelle. Prochains efforts du groupe MIF <http://tools.ietf.org/wg/mif> : l'améliorer... Le RFC 6418 donne le cahier des charges de cet effort. Le RFC 6731 normalise les nouvelles règles de sélection pour les serveurs DNS.