Lier une prise à IPv6 seulement ou bien aux deux familles, v4 et v6 ?

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 11 juin 2010. Dernière mise à jour le 3 mars 2013

https://www.bortzmeyer.org/bindv6only.html

Parmi les nombreuses questions que soulève le déploiement d'IPv6, il y en a une qui ne semble pas progresser : lorsqu'un programmeur branche une prise, celle-ci doit-elle écouter en IPv6 ou bien dans les deux protocoles, c'est-à-dire également en IPv4 ? La question resurgit régulièrement, fait l'objet de polémiques, mais il ne semble pas y avoir encore de consensus.

Pour comprendre le problème, voyons le cas d'un serveur simple, écrit en C dont le programmeur n'a pas très envie de créer deux prises (une pour v4 et une pour v6) et d'utiliser `select()`. Ce n'est pas qu'il est paresseux, c'est simplement qu'il ne voit pas pourquoi son serveur, qui travaille en couche 7, devrait se préoccuper du protocole de couche 3. S'il écrit simplement (le code complet figure en (en ligne sur `https://www.bortzmeyer.org/files/server-v4v6-bind.c`)) :

```
me->sin6_family = AF_INET6;
me->sin6_addr = in6addr_any;
me->sin6_port = htons(PORT);
bind(fd, me, struct_sockaddr_size);
```

Le serveur va t-il écouter en IPv6 seulement ? Eh bien, cela dépend. Sur Linux, avec le réglage par défaut le plus courant, non, ce code fera un serveur v6 **et** v4. Il pourra alors recevoir des connexions aussi bien de clients v4 que de clients v6, et avec une seule prise. le serveur (en ligne sur `https://www.bortzmeyer.org/files/server-v4v6-bind.c`) affichera :

```
One connection accepted from ::ffff:192.168.2.25
One connection accepted from 2a01:e35:8bd9:8bb0:21c:23ff:fe00:6b7f
```

la première connexion provenant d'une adresse IPv4 (à la syntaxe définie dans la section 2.5.5.2 du RFC 4291 [1]) et la seconde d'une v6.

À quoi est-ce dû ? À une option `sysctl` très pratique, `/proc/sys/net/ipv6/bindv6only`. Si le contenu de ce fichier vaut 0, le cas le plus fréquent, une prise de la famille `AF_INET6` écoute sur les deux protocoles. Ce mécanisme est celui recommandé dans la section 3.7 du RFC 3493 ou bien par l'OpenGroup <http://www.opengroup.org/onlinepubs/9699919799/functions/V2_chap02.html#tag_15_10_20_02>. On peut aussi lire la section 4 du RFC 4038. Ce mécanisme est très simple pour le programmeur et lui permet de traiter le vieux protocole IPv4 comme un simple cas particulier.

Par contre, si ce fichier contient 1 (qui se lit « "*bindv6only*" est VRAI donc je n'écoute qu'en v6 »), le serveur ne servira que les clients IPv6 (nettement moins nombreux, à l'heure actuelle). Il est donc préférable pour le programmeur Linux que cette option soit à 0. Le problème est que ce n'est pas garanti. L'administrateur système a pu la mettre à 1 et l'application n'a pas de moyen de changer cette valeur (il faudrait être root).

Pire, certains systèmes d'exploitation utilisant un noyau Linux ont pu décider de changer la valeur par défaut et la grande majorité des machines utilisant ces systèmes vont alors avoir `/proc/sys/net/ipv6/bindv6` à 1. C'est le cas de Debian. Vous pouvez consulter le texte de la décision originelle <http://lists.debian.org/debian-devel/2009/10/msg00541.html> en octobre 2009 (pas du tout convaincant, selon moi) ou bien les opinions opposées <http://lists.debian.org/debian-devel/2010/04/msg00099.html> ou encore un des nombreux rapports de bogue critiquant cette décision et demandant un retour à la valeur initiale de 0 (bogues #576633 <http://bugs.debian.org/576633>, #560056 <http://bugs.debian.org/560056>, #560238 <http://bugs.debian.org/560238>, etc). Cette décision a en effet cassé pas mal de programmes comme le JRE d'Oracle (voir le rapport de bogue <http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=6342561>) .

Même si Debian changeait d'avis et revenait à la valeur originale de `/proc/sys/net/ipv6/bindv6only=0`, il reste les autres systèmes comme NetBSD ou FreeBSD où `net.inet6.ip6.v6only` a toujours été à un par défaut, et sur lesquels les prises v6 n'écoutent par défaut qu'en v6. Que le programmeur pense que `bindv6only` devrait être à 0 ou à 1 ne change pas grand'chose : s'il veut que son code soit portable, il doit marcher dans les deux cas (voir en ce sens une opinion sur Debian <http://lists.debian.org/debian-devel/2010/04/msg00120.html>, « "*I think that any application that breaks if the default value is 0 or 1 is broken*" »).

Y a t-il une autre solution ? Oui, le même RFC 3493, dans sa section 5.3, propose une nouvelle option `setsockopt()`, `IPV6_V6ONLY` que l'**application** et non plus l'**administrateur système** peut utiliser pour contrôler si les prises v6 acceptent aussi le v4 :

```
*v6only = false;
setsockopt(fd, IPPROTO_IPV6, IPV6_V6ONLY,
                (*v6only ? &true_opt : &false_opt), sizeof(int));
```

Ainsi, une application qui tient à n'avoir qu'une seule prise, pour des raisons de simplicité, et à traiter les deux familles, v4 et v6, de la même manière (ce qui est logique pour la grande majorité des applications, la couche réseau n'étant pas leur souci), peut utiliser cette option et tout marche. Cela oblige simplement le programmeur à un peu plus de travail.

Avec le serveur développé ici (en ligne sur `https://www.bortzmeyer.org/files/server-v4v6-bind.c`), cela se fait avec l'option `-o` (uniquement v6) ou `-b` (les deux) :

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc4291.txt`

```
# Sur une machine NetBSD, où, par défaut, c'est uniquement IPv6 :
% ./server-v4v6-bind

# Sur le client :
% telnet 192.168.2.7 8081
Trying 192.168.2.7...
telnet: Unable to connect to remote host: Connection refused

# Sur la machine NetBSD, avec la bonne option, pour accepter les deux familles :
% ./server-v4v6-bind -b
...
One connection accepted from ::ffff:192.168.2.1

# Le client avait eu :
% telnet 192.168.2.7 8081
Trying 192.168.2.7...
Connected to 192.168.2.7.
Escape character is '^]'.
```

Cela, c'était pour C. Mais pour les autres langages ? Eh bien, c'est souvent le problème. L'option `IPV6_V6ONLY` n'est pas toujours accessible. Un langage comme Go a mis du temps (rapport de bogue #679 <http://code.google.com/p/go/issues/detail?id=679>) mais aujourd'hui les programmes comme le serveur echo NoNewContent <https://framagit.org/bortzmeyer/nonewcontent> gèrent bien IPv4 et IPv6.

En Java, même problème, aucun moyen <http://java.sun.com/j2se/1.4.2/docs/guide/net/socketOpt.html> de positionner l'option `IPV6_V6ONLY`, compter sur l'option `/proc/sys/net/ipv6/bindv6on` du système est la seule solution.

Dans d'autres langages (Perl - cf. bogue Debian #569981 <http://bugs.debian.org/569981>, par exemple), l'option n'est pas connue directement, mais en récupérant la valeur de la constante (dans les en-têtes C, elle vaut 26 sur ma Debian et 27 sur ma NetBSD) et en la précisant à la main on peut quand même utiliser cette option. Un document d'EGEE <https://edms.cern.ch/document/971407> donne davantage de détails.

Pour Python, voici comment faire le programme équivalent à celui en C, dans une version de haut niveau, utilisant la bibliothèque SocketServer <http://docs.python.org/library/socketserver.html> (il faut redéfinir la méthode `server_bind` car, une fois le constructeur `ThreadingTCPServer` appelé, c'est trop tard) :

```
import SocketServer
# Low-level library "socket" required to set IPv6only to false :-(
import socket

PORT = 8081

class AllIPsTCPServer(SocketServer.TCPServer):

    def server_bind(self):
        # Override this method to be sure v6only is false: we want to
        # listen to both IPv4 and IPv6!
        # The socket.IPV6_V6ONLY appeared with Python 2.5.
        self.socket.setsockopt(socket.IPPROTO_IPV6, socket.IPV6_V6ONLY, False)
        SocketServer.TCPServer.server_bind(self)

class MyHandler(SocketServer.StreamRequestHandler):

    def handle(self):
        print "Connection from %s" % self.client_address[0]

AllIPsTCPServer.address_family = socket.AF_INET6
server = AllIPsTCPServer(("", PORT), MyHandler)
server.serve_forever()
```

Bien sûr, c'est assez lourd. Tot l'intérêt d'un langage de haut niveau comme Python et d'une bibliothèque comme SocketServer, qui masque les aspects les plus primitifs des prises, est justement normalement de dispenser de ce genre de manip'. Normalement, tout ceci devrait être fait par SocketServer mais, hélas, ne l'est pas.

On peut aussi en faire une version de bas niveau, utilisant directement la bibliothèque socket <http://docs.python.org/library/socket.html> (mais SocketServer apporte plein d'avantages pour une application réelle, donc je déconseille, ceci dit, il existe une bonne documentation <http://docs.python.org/dev/howto/sockets.html>):

```
import socket

PORT = 8081

s = socket.socket(socket.AF_INET6, socket.SOCK_STREAM)
# Listen on v4 as well as v6
s.setsockopt(socket.IPPROTO_IPV6, socket.IPV6_V6ONLY, False)
s.bind(("", PORT))
s.listen(1)
while True:
    (rs, peer) = s.accept()
    print ("Connection accepted from %s" % peer[0])
```

Merci à Xavier Jeannin, Pierre Beyssac, Vincent Danjean et Étienne Duble pour leurs excellents conseils. Et bien sûr le génial livre de Stevens <https://www.bortzmeyer.org/unix-network-programming.html> m'a beaucoup aidé pour écrire le programme de test (en ligne sur https://www.bortzmeyer.org/files/server-v4v6-bind.c). Autre excellente source d'information, le rapport d'Étienne Dublé du groupe EGEE <https://edms.cern.ch/document/971407> sur les différents langages de programmation et IPv6. Pour un très bon tour d'horizon de tous les aspects de la programmation réseau dans les deux familles, je recommande le chapitre « Programmation » du « Gisèle » <http://livre.g6.asso.fr/index.php/Programmation_d%27applications_bis>.

---

https://www.bortzmeyer.org/bindv6only.html