Ajouter un en-tête Destination aux paquets IPv6

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 11 mars 2012

http://www.bortzmeyer.org/destination-options-ipv6.html

Le protocole IPv6 permet de définir des options dans le paquet IPv6, options qui vont s'insérer entre l'en-tête IPv6 et les données. Comment les définir depuis son programme ? On trouve très peu d'informations à ce sujet sur le Web. Alors, voici comment je fais, en C.

Lorsqu'on voulait mettre des valeurs quelconques dans un paquet IPv4, la technique la plus utilisée était celle des « prises brutes <http://www.bortzmeyer.org/raw-sockets.html> » ("*raw sockets*") où on peut définir chaque bit à sa convenance. Elle est mal normalisée et dangereuse (il est facile de produire des paquets invalides). IPv6 préfère donc une autre approche, normalisée dans le RFC 3542 [1], où des primitives de plus haut niveau sont fournies pour changer des choses dans le paquet. Mais le moins qu'on puisse dire est que le RFC n'est pas spécialement convivial (pas un seul exemple) et qu'on trouve peu de tutoriels en ligne... J'avais besoin d'insérer dans les paquets une option « Destination » (RFC 8200, section 4.6) et je n'ai rien trouvé nulle part. La deuxième édition du livre de Stevens <http://www.bortzmeyer.org/unix-network-programming.html> n'est guère prolixe et un des rares exemples est même carrément faux.

Bref, en repartant du RFC 3542, sections 8 et 9, voici comment insérer une option Destination dans un paquet IPv6 sortant : il existe deux méthodes, une par paquet, à base de `msghdr` qu'on passe à `sendmsg`, et une globale, affectant tous les paquets envoyés sur la prise, à base de `inet6_opt_*`. C'est cette dernière que j'utilise. Le principe est de constituer l'en-tête Destination à l'aide des routines `inet6_opt_*`, puis d'utiliser `setsockopt` pour l'attacher à la prise. L'en-tête sera alors ajouté à chaque paquet sortant. D'abord, quelques constantes dont on aura besoin :

```
#define MIN_EXTLEN 8
#define EXT_TYPE 11
```

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc3542.txt`

Ici, `MIN_EXTLEN` est la longueur minimale d'un en-tête Destination, en octets (attention en lisant les paquets en binaire : le champ `Hdr Ext Len` de l'en-tête est en unités de 8 octets et, en prime, il part de zéro ; l'interface du RFC 3542 nous dispense heureusement de ces particularités). Quant aux options contenues dans l'en-tête Destination, on n'en mettra qu'une, de type 11, actuellement non affecté <https://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xml#ipv6-parameters- (Notez que les deux bits de poids le plus fort encodent le comportement attendu du récepteur, s'il ne connait pas l'option. Pour 11, ces deux bits sont à zéro, indiquant au récepteur qu'il doit ignorer cette option, s'il ne la comprend pas.)

On crée ensuite la prise comme d'habitude. Puis on va fabriquer l'en-tête Destination. D'abord :

```
char           *ext_buffer;
ext_buffer = malloc(MIN_EXTLEN);
ext_offset = inet6_opt_init(ext_buffer, MIN_EXTLEN);
ext_buffer[0] = SOL_UDP;
```

Ici, `ext_buffer` va contenir l'option Destination. Pour l'initialiser, on indique juste la taille (en multiples de 8, obligatoirement). Rappelez-vous que la taille indiquée ici n'est pas dans les mêmes unités, ni avec le même point de départ, que ce qui apparaîtra dans le paquet sur le câble. Lisez donc bien le RFC 3542, qui normalise l'API et pas le RFC 2460 qui normalise le protocole. À ce stade, `ext_buffer` contient {0x11, 0x00}, où le 0x11 désigne UDP, l'en-tête qui suivra notre en-tête Destination. À chaque étape, `ext_offset` indiquera la taille de `ext_buffer` en octets

On ajoute ensuite notre option `EXT_TYPE` :

```
char           *ext_option;
uint8_t         value;
ext_option = malloc(MIN_EXTLEN);
ext_offset =
         inet6_opt_append(ext_buffer, MIN_EXTLEN, ext_offset, EXT_TYPE,
                          sizeof(value), 1,
  &ext_option);
```

On a indiqué la taille de l'option (`sizeof(value)`, notez que `inet6_opt_append` ajoutera tout seul la taille pour indiquer le type et la longueur, les options étant encodées en TLV) et l'alignement désiré (1, c'est-à-dire pas de contraintes). `ext_option` nous donnera accès au contenu de l'option. `ext_buffer` vaut {0x11, 0x00, 0x0b, 0x01, 0xd0} où 0x0b est le type de notre option, 0x01 sa longueur et 0xd0 une valeur quelconque puisqu'on n'a encore rien indiqué (elle va donc dépendre de votre implémentation, 0xd0 est ce que j'obtiens sur ma machine).

Maintenant, on met l'option à une certaine valeur, ici 9 :

```
value = 9:
inet6_opt_set_val(ext_option, 0, &value, sizeof(value)
```

Et c'est fait, `ext_buffer` vaut {0x11, 0x00, 0x0b, 0x01, 0x09}.

Si on voulait mettre plusieurs options dans l'en-tête Destination, on continuerait à appeler `inet6_opt_append` et `inet6_opt_set_val`. Mais je vais m'arrêter ici. (Notez toutefois que `inet6_opt_*`, plus loin, ajoutera automatiquement une option de remplissage.)

On termine le travail :

```
ext_offset = inet6_opt_finish(ext_buffer, MIN_EXTLEN, ext_offset);
```

Cet appel à `inet6_opt_finish` n'est pas juste pour faire joli. N'oubliez pas que l'en-tête Destination doit être composé d'un multiple de 8 octets. Et nous n'en avons que 5 (type, longueur de l'en-tête, type de l'unique option, longueur de l'unique option, valeur de l'unique option, tous à 1 octet). `inet6_opt_finish` va donc être chargé du remplissage ("*padding*"). Après cette fonction, `ext_buffer` vaudra {0x11, 0x00, 0x0b, 0x01, 0x09, 0x01, 0x01, 0x00}. Les trois derniers octets sont ceux de l'option PadN (RFC 2460, section 4.2), également en TLV : un octet de type, un de longueur et un de données.

Voilà, à ce stade, on a un `ext_buffer`, de longueur `ext_offset`, qui contient un bel en-tête Destination, avec toutes les options souhaitées, on n'a plus qu'à l'attacher à la prise :

```
setsockopt(sd, IPPROTO_IPV6, IPV6_DSTOPTS, ext_buffer, ext_offset);
```

Désormais, tout envoi de paquet par la prise `sd` incluera cet en-tête :

```
sendto(sd, message, (size_t) messagesize, 0, result->ai_addr,
               result->ai_addrlen);
```

Sur Linux, il faut être root malheureusement, pour ajouter l'en-tête Destination. (Emmanuel Thierry me fait remarquer que c'est un peu plus compliqué : il faut avoir le privilège `CAP_NET_RAW` - requis pour ouvrir une prise brute, pas nécessairement être root. Donc on peut suggérer plutôt l'utilisation de `setcap 'CAP_NET_RAW+eip' /chemin/vers/le/binaire`. Ou bien démarrer en root, et dès le démarrage du programme ne garder que ce privilège et abandonner le reste.)

On peut voir le résultat avec tcpdump :

```
15:31:41.020647 IP6 (hlim 64, next-header unknown (60) payload length: 26) \
    2a01:e35:8bd9:8bb0:a0a7:ea9c:74e8:d397 > 2001:4b98:dc0:41:216:3eff:fece:1902: \
    DSTOPT (opt_type 0x0b: len=1) (padn) \
    42513 > 42: [udp sum ok] UDP, length 10
```

Le "*next-header unknown*" est l'en-tête Destination, qui a en effet le numéro 60. tcpdump a quand même reconnu cet en-tête `DSTOPT` et vu l'option de type 11 et le PadN. Ensuite, il a bien vu l'UDP vers le port 42. Avec tshark, c'est moins bien :

```
Internet Protocol Version 6
    0110 .... = Version: 6
        [0110 .... = This field makes the filter "ip.version == 6" possible: 6]
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 26
    Next header: IPv6 destination option (0x3c)
    Hop limit: 64
    Source: 2a01:e35:8bd9:8bb0:a0a7:ea9c:74e8:d397 (2a01:e35:8bd9:8bb0:a0a7:ea9c:74e8:d397)
    Destination: 2001:4b98:dc0:41:216:3eff:fece:1902 (2001:4b98:dc0:41:216:3eff:fece:1902)
    Destination Option
        Next header: UDP (0x11)
        Length: 0 (8 bytes)
User Datagram Protocol, Src Port: 42513 (42513), Dst Port: name (42)
    Source port: 42513 (42513)
    Destination port: name (42)
    Length: 18
```

tshark a tout juste trouvé qu'il y avait un en-tête Destination de longueur 8, sans pouvoir l'analyser. Wireshark affiche au moins le contenu de l'option, en hexadécimal.

Si vous aimez lire le C, voici le programme que j'ai utilisé (en ligne sur `http://www.bortzmeyer.org/files/ipv6-dest-options.c`).