Groupe de travail Réseau
Request for Comments : 3678
Catégorie : Information
Traduction Claude Broère de L'Isle

D. Thaler, Microsoft
B. Fenner, AT&T Research
B. Quinn, Stardust.com
janvier 2004

Extensions d'interface de prise pour filtres de source de diffusion groupée

Résumé
La version 3 du protocole de gestion de groupe Internet (IGMPv3, *Internet Group Management Protocol version 3*) pour IPv4 et la version 2 de la découverte d'écoutant de diffusion groupée (MLDv2, *Multicast Listener Discovery v2*) pour IPv6 ajoutent pour les applications la capacité d'exprimer des filtres de source sur l'adhésion aux groupes de diffusion groupée, ce qui permet aux applications receveuses de déterminer l'ensemble d'envoyeurs (sources) à partir desquels accepter le trafic en diffusion groupée. Cette capacité simplifie aussi la prise en charge de types d'applications de diffusion groupée de un à beaucoup.

Le présent document spécifie de nouvelles options et fonctions des prises pour gérer les filtres de source pour les adhésions aux groupes de diffusion groupée IP. Il définit aussi les structures de prises pour fournir des arguments d'entrée et de sortie à ces nouvelles interfaces de programme d'application (API, *application program interface*). Ces extensions sont conçues pour donner accès aux caractéristiques de filtrage de source, tout en introduisant un minimum de changements au système et en assurant une complète compatibilité avec les applications de diffusion groupée existantes.

Table des matières

1. Introduction

Le standard de facto d'interface de programme d'application (API, *application program interface*) pour les applications TCP/IP est l'interface de "prises". Bien que cette API ait été développée pour Unix au début des années 1980, elle a aussi été

mise en œuvre sur une grande variété de systèmes non Unix. Les applications TCP/IP écrites en utilisant l'API de prise ont joui dans le passé d'un haut degré de portabilité et on aimerait la même portabilité pour les applications qui emploient les filtres de source de diffusion groupée. Il est nécessaire d'apporter des changements à l'API de prises pour la prise en charge d'un tel filtrage et le présent mémoire décrit ces changements.

Le présent document spécifie de nouvelles options de prise et de nouvelles fonctions pour gérer les filtres de source pour l'adhésion aux groupes de diffusion groupée IP. Il définit aussi les structures de prise pour fournir les arguments d'entrée et de sortie à ces nouvelles API. Ces extensions sont conçues pour fournir l'accès aux dispositifs de filtrage de source requis par ces applications, tout en introduisant un minimum de changement dans le système et en assurant une complète compatibilité avec les applications existantes de diffusion groupée.

De plus, la [RFC3493] définit des extensions d'interface de prise pour IPv6, incluant des fonctions indépendantes du protocole pour la plupart des opérations.

Cependant, bien qu'elle définisse des fonctions de jonction et d'abandon pour IPv6, elle ne fournit pas des versions indépendantes du protocole pour ces opérations. De telles fonctions seront décrites dans le présent document.

Le lecteur devrait noter que le présent document est seulement pour information, et que la spécification officielle de la norme de l'API de prise est la norme [IEEE 1003.1-2001].

2. Considérations de conception

Il y a un certain nombre de considérations importantes pour la conception des changements à cette API bien conservée :

- Les changements à l'API devraient assurer la compatibilité aussi bien de source que binaire pour les programmes écrits sur l'API d'origine. C'est-à-dire que le binaire des programmes existants devrait continuer de fonctionner sur un système qui prend en charge la nouvelle API. De plus, les applications existantes qui sont recompilées et fonctionnent sur un système qui prend en charge la nouvelle API devraient continuer de fonctionner. Dit simplement, les changements d'API pour les receveurs de diffusion groupée qui spécifient des filtres de source ne devraient pas casser les programmes existants.

- Les changements à l'API devraient être aussi minimes que possible afin de simplifier la tâche de conversion des applications existantes de receveur de diffusion groupée pour utiliser les filtres de source.

- Les applications devraient être capables de détecter quand les nouvelles API de filtre de source ne sont pas disponibles (par exemple, des appels échouent avec l'erreur ENOTSUPP) et de réagir en douceur (par exemple, en revenant à l'ancienne API sans filtre de source ou en affichant un message d'erreur significatif pour l'utilisateur).

- Le manque de sécurité de type dans une API est une mauvaise chose qui devrait être évitée lorsque possible.

Il existe plusieurs mises en œuvre qui utilisent ioctl() pour une portion de la fonctionnalité décrite ici, et pour des raisons historiques, l'API ioctl est donnée à la Section 8. L'API préférée inclut cependant de nouvelles fonctions. Les raisons de l'ajout de nouvelles fonctions sont :

- Les nouvelles fonctions fournissent la sécurité de type, à la différence de ioctl, getsockopt, et setsockopt.
- Une nouvelle fonction peut être écrite comme enveloppe par dessus un appel ioctl, getsockopt, ou setsockopt, si nécessaire. Donc, elle donne plus de liberté quant à la façon dont la fonctionnalité est mise en œuvre dans un système d'exploitation. Par exemple, une nouvelle fonction pourrait être mise en œuvre comme fonction en ligne dans un fichier inclus, ou une fonction exportée à partir d'une bibliothèque en mode utilisateur qui utilise en interne un mécanisme pour échanger des informations avec le noyau, ou être mise en œuvre directement dans le noyau.
- Au moins une opération définie ici doit être capable à la fois de passer des informations à la pile TCP/IP, et de restituer des informations à partir d'elle. Dans certaines mises en œuvre, cela est problématique même sans changer getsockopt ou en utilisant ioctl. Utiliser les nouvelles fonctions évite d'avoir besoin de changer de telles mises en œuvre.

2.1 Ce qui doit être ajouté

Les API de diffusion groupée IP existantes permettent à une application receveuse de spécifier le groupe d'adresses (de destination) et (facultativement) l'interface locale. Ces API existantes n'ont pas besoin de changer (et ne le peuvent pas, pour garder la compatibilité binaire). Donc, ce qui est nécessaire est de nouvelles API de filtre de source qui fournissent la même fonctionnalité et permettent aussi aux applications receveuses de diffusion groupée de :

- spécifier zéro, une ou plusieurs adresses (de source) en envoi individuel dans un filtre de source,
- déterminer si le filtre de source décrit une liste de sources inclusive ou exclusive.

Le nouveau concept d'API doit permettre cette fonctionnalité pour IPv4 et IPv6.

2.2 Types de données

Les types de données des éléments de structure donnés dans le présent mémoire sont destinés à être des exemples, et non des exigences absolues. Chaque fois que possible, on utilise les types de données provenant de POSIX 1003.1g [IEEE-1003] : uintN_t signifie un entier non signé de exactement N bits (par exemple, uint32_t).

2.3 En-têtes

Lorsque des prototypes et structures de fonctions sont montrés, on montre les en-têtes qui doivent être inclus pour faire que cet élément soit défini.

2.4 Structures

Lorsque des structures sont décrites, les membres montrés sont ceux qui doivent apparaître dans une mise en œuvre. Des membres non standard supplémentaires peuvent aussi être définis par une mise en œuvre. Pour une précaution supplémentaire, des membres non standard pourraient être vérifiés par des macros d'essai de caractéristique *(Feature Test Macros)* comme décrit dans [IEEE-1003]. (De telles macros ne sont pas définies par la présente RFC.)

L'ordre indiqué pour les membres d'une structure est celui qui est recommandé, étant données les considérations d'alignement des membres sur plusieurs octets, mais une mise en œuvre peut ordonner les membres différemment.

3. Vue d'ensemble des API

Il y a un certain nombre d'API différentes, décrites dans ce document, qui sont appropriées pour différents types d'applications et versions IP. Avant de fournir des descriptions détaillées, la présente section donne une "taxonomie" avec une brève description de chaque taxon.

Il y a deux catégories d'API de filtre de source, dont chacune est conçue pour permettre aux applications receveuses de diffusion groupée de désigner la ou les adresses d'envoi individuel du ou des envoyeurs ainsi que le groupe de diffusion groupée (adresse de destination) à recevoir :

- Basique (fondé sur le delta) : certaines applications désirent la simplicité d'une API fondée sur les différences dans laquelle chaque appel de fonction spécifie une seule adresse de source qui devrait être ajoutée ou retirée du filtre existant pour une certaine adresse de groupe de diffusion groupée sur laquelle écouter. De telles applications tombent normalement dans l'une des deux catégories suivantes :
  - Diffusion groupée toutes sources : par défaut, toutes les sources sont acceptées. Des sources individuelles peuvent être désactivées et reprises en tant que de besoin selon le moment. On appelle aussi ce mode "exclure", car le filtre de source contient une liste des sources exclues.
  - Diffusion groupée spécifique de source : seules les sources d'une certaine liste sont admises. La liste peut changer avec le temps. C'est aussi appelé le mode "inclure", car le filtre de sources contient une liste des sources incluses. Cette API serait utilisée, par exemple, par des applications à une "seule source" comme une diffusion audio/vidéo. Elle serait aussi utilisée pour des sessions logiques multi-sources où chaque source alloue de façon indépendante son adresse de groupe de diffusion groupée spécifique de source.
- Avancé (état plein) : cette API permet à une application de définir un filtre de source complet comportant zéro, une ou plusieurs adresses de source, et de remplacer le filtre précédent par un nouveau filtre. Les applications qui exigent la capacité de changer de mode de filtre sans quitter un groupe doivent utiliser une API à état plein (c'est-à-dire, changer la sémantique du filtre de source de inclusif en exclusif, ou vice versa). Les applications qui utilisent une grande liste de sources pour une certaine adresse de groupe devraient aussi utiliser l'API à état plein, car les changements de filtre peuvent être faits de façon atomique en une seule opération.

Les types d'API ci-dessus existent dans des variantes spécifiques de IPv4 ainsi qu'avec des variantes indépendantes du protocole. On peut se demander pourquoi les API indépendantes du protocole ne peuvent pas s'accommoder des applications IPv4 et IPv6. Comme toute application IPv4 exige de toutes façons des modifications pour utiliser des filtres de source de diffusion groupée, cela pourrait sembler une bonne opportunité pour créer un code source compatible IPv6.

Les principales raisons pour étendre une API spécifique d'IPv4 sont :

- de minimiser les changements nécessaires du code source d'application existante de diffusion groupée IPv4 pour ajouter la prise en charge de filtre de source,
- d'éviter de surcharger les API pour s'accommoder des différences entre les adresses d'interface IPv4 (par exemple, dans la

structure ip_mreq) et les indices d'interface.

4. API de filtre de source de diffusion groupée IPv4

La version 3 du protocole Internet de gestion de groupe (IGMPv3) [RFC3376] et la version 2 du protocole de découverte d'écoutant de diffusion groupée (MLDv2) [RFC3810] donnent la capacité de communiquer les informations de filtre de source au routeur et donc d'éviter de tirer des données de sources non désirées sur la liaison locale. Cependant, les filtres de source peuvent être mis en œuvre par le système d'exploitation sans considération de la prise en charge de IGMPv3 ou MLDv2 par les routeurs, de sorte que lorsque l'API de filtre de source est disponible, les applications peuvent toujours tirer parti de leur utilisation.

4.1 API de base (fondée sur le delta) pour IPv4

La réception de paquets de diffusion groupée est contrôlée par l'option setsockopt() décrite ci-dessous. Une erreur de EOPNOTSUPP est retournée si ces options sont utilisés avec getsockopt().

Les structures suivantes sont utilisées par les deux API Diffusion groupée toutes sources et Diffusion groupée spécifique de source :

```
#include <netinet/in.h>

struct ip_mreq {
  struct in_addr imr_multiaddr;          /* adresse IP de groupe */
  struct in_addr imr_interface;          /* adresse IP d'interface */
};

struct ip_mreq_source {
  struct in_addr imr_multiaddr;          /* adresse IP de groupe */
  struct in_addr imr_sourceaddr;         /* adresse IP de source */
  struct in_addr imr_interface;          /* adresse IP d'interface */
};
```

4.1.1 API de diffusion groupe toutes sources IPv4

Les options de prise suivantes sont définies dans <netinet/in.h> pour les applications de la catégorie Diffusion groupée toutes sources :

Option de prise	**type d'argument**
IP_ADD_MEMBERSHIP	struct ip_mreq
IP_BLOCK_SOURCE	struct ip_mreq_source
IP_UNBLOCK_SOURCE	struct ip_mreq_source
IP_DROP_MEMBERSHIP	struct ip_mreq

IP_ADD_MEMBERSHIP et IP_DROP_MEMBERSHIP sont déjà mises en œuvre sur la plupart des systèmes d'exploitation, et sont utilisées pour se joindre à, et quitter, un groupe toutes sources.

IP_BLOCK_SOURCE peut être utilisée pour bloquer des données d'une certaine source sur une certain groupe (par exemple, si l'utilisateur "rend muette" cette source) et IP_UNBLOCK_SOURCE peut être utilisée pour défaire cela (par exemple, si l'usager "rend la parole" à la source).

4.1.2 API de diffusion groupée spécifique de source IPv4

Les options de prise suivantes sont disponibles pour les applications de la catégorie Spécifique de source.

Option de prise	**Type d'argument**
IP_ADD_SOURCE_MEMBERSHIP	struct ip_mreq_source
IP_DROP_SOURCE_MEMBERSHIP	struct ip_mreq_source
IP_DROP_MEMBERSHIP	struct ip_mreq

IP_ADD_SOURCE_MEMBERSHIP et IP_DROP_SOURCE_MEMBERSHIP sont utilisées pour se joindre à, et quitter, un

groupe spécifique de source.

IP_DROP_MEMBERSHIP est pris en charge, comme facilité, pour abandonner toutes les sources qui ont été jointes pour un groupe et interface particuliers. Les opérations sont les mêmes que si la prise avait été close.

4.1.3 Codes d'erreur

Lorsque l'option serait légale sur le groupe, mais qu'une adresse est invalide (par exemple, lorsque on essaye de bloquer une source qui est déjà bloquée par la prise, ou quand on essaye d'abandonner un groupe non joint) l'erreur générée est EADDRNOTAVAIL.

Lorsque l'option elle-même n'est pas légale pour le groupe (c'est-à-dire, lorsque on essaye une option spécifique de source sur un groupe après avoir fait IP_ADD_MEMBERSHIP, ou lorsque on essaye une option Toutes sources sans faire IP_ADD_MEMBERSHIP) l'erreur générée est EINVAL.

Lorsque une de ces options est utilisée avec getsockopt(), l'erreur générée est EOPNOTSUPP.

Finalement, si la mise en œuvre impose une limite au nombre maximum de sources dans un filtre de source, ENOBUFS est généré lorsque une opération excèderait le maximum.

4.2 API évoluée (à état plein) pour IPv4

Il existe plusieurs mises en œuvre qui utilisent ioctl() pour cette API, et pour des raisons historiques, l'API ioctl() est documentée à la Section 8. L'API préférée utilise les nouvelles fonctions décrites ci-dessous.

4.2.1 Filtre Set Source

```
#include <netinet/in.h>

int setipv4sourcefilter(int s, struct in_addr interface,
                        struct in_addr group, uint32_t fmode,
                        uint32_t numsrc, struct in_addr *slist);
```

En cas de succès, la valeur 0 est retournée, et en cas d'échec, la valeur -1 est retournée et errno est réglé en conséquence.
L'argument s identifie la prise.
L'argument interface contient l'adresse IP locale de l'interface.
L'argument group contient l'adresse IP de diffusion groupée du groupe.
L'argument fmode identifie le mode de filtre. La valeur de ce champ doit être MCAST_INCLUDE ou MCAST_EXCLUDE, qui sont aussi définis dans <netinet/in.h>.
L'argument numsrc contient le nombre d'adresses de source dans le dispositif slist.
L'argument slist pointe sur un dispositif d'adresses IP de sources à inclure ou exclure selon le mode de filtre.

Si la mise en œuvre impose une limite au nombre maximum de sources dans un filtre de sources, ENOBUFS est généré lorsque l'opération excèderait le maximum.

4.2.2 Filtre Get Source

```
#include <netinet/in.h>

int getipv4sourcefilter(int s, struct in_addr interface,
                        struct in_addr group, uint32_t *fmode,
                        uint32_t *numsrc, struct in_addr *slist);
```

En cas de succès, la valeur 0 est retournée, et en cas d'échec, la valeur -1 est retournée et errno est réglé en conséquence.
L'argument s identifie la prise.
L'argument interface contient l'adresse IP locale de l'interface.
L'argument group contient l'adresse IP de diffusion groupée du groupe.
L'argument fmode pointe sur un entier qui va contenir le mode de filtre lors d'un retour réussi. La valeur de ce champ sera MCAST_INCLUDE ou MCAST_EXCLUDE, qui est aussi définie dans <netinet/in.h>.
En entrée, l'argument numsrc contient le nombre d'adresses de source qui vont tenir dans le dispositif slist. En sortie, l'argument numsrc va contenir le nombre total de sources dans le filtre.

L'argument slist pointe sur la mémoire tampon dans laquelle une matrice d'adresses IP de sources incluses ou exclues (selon le mode du filtre) sera écrite. Si numsrc était 0 en entrée, un pointeur NULL peut être fourni.

Si l'application ne connait pas déjà la taille de la liste des sources, elle peut faire une hypothèse raisonnable (par exemple, 0) et si après coup, numsrc contient une plus grande valeur, l'opération peut être répétée avec une mémoire tampon assez grande. C'est-à-dire qu'au retour, numsrc est toujours mis à jour pour être le nombre total de sources dans le filtre, alors que slist va contenir autant d'adresses de source qu'il y tient, jusqu'au minimum de la taille de dispositif passé comme valeur originelle de numsrc et du nombre total de sources dans le filtre.

5. API de filtre de source de diffusion groupée indépendantes du protocole

Des fonctions indépendantes du protocole sont fournies pour se joindre et quitter les opérations afin qu'une application puisse passer une structure sockaddr_storage obtenue d'appels tels que getaddrinfo() [1] comme le groupe auquel se joindre. Par exemple, une application peut résoudre un nom DNS (par exemple, NTP.MCAST.NET) en adresse de diffusion groupée qui peut être IPv4 ou IPv6, et peut facilement se joindre et quitter le groupe.

5.1 API de base (fondée sur le delta)

La réception de paquets en diffusion groupée est contrôlée par les options setsockopt() résumées ci-dessous. Une erreur de EOPNOTSUPP est retournée si ces options sont utilisées avec getsockopt().

Les structures suivantes sont utilisés par les deux API Toute source de diffusion groupée et Diffusion groupée spécifique de source : #include <netinet/in.h>

```
struct group_req {
  uint32_t             gr_interface;          /* indice d'interface */
  struct sockaddr_storage gr_group;           /* adresse de groupe */
};

struct group_source_req {
  uint32_t             gsr_interface;         /* indice d'interface */
  struct sockaddr_storage gsr_group;          /* adresse de groupe */
  struct sockaddr_storage gsr_source;         /* adresse de source */
};
```

La structure sockaddr_storage est définie dans la [RFC3493] comme étant assez grande pour contenir les informations d'adresse IPv4 aussi bien qu'IPv6.

Les règles pour générer les erreurs sont les mêmes que celles données au paragraphe 4.1.3.

5.1.1 API de diffusion groupée toutes sources

Option de prise	Type d'argument
MCAST_JOIN_GROUP	struct group_req
MCAST_BLOCK_SOURCE	struct group_source_req
MCAST_UNBLOCK_SOURCE	struct group_source_req
MCAST_LEAVE_GROUP	struct group_req

MCAST_JOIN_GROUP et MCAST_LEAVE_GROUP sont utilisés pour se joindre à, et quitter, un groupe de diffusion à la cantonade.

MCAST_BLOCK_SOURCE peut être utilisé pour bloquer les données provenant d'une certaine source à un certain groupe (par exemple, si l'usager "rend muette" cette source) et MCAST_UNBLOCK_SOURCE peut être utilisé pour défaire cela (par exemple, si alors l'usager "rend la parole" à la source).

5.1.2 API de diffusion groupée spécifique de source

Option de prise	Type d'argument
MCAST_JOIN_SOURCE_GROUP	struct group_source_req
MCAST_LEAVE_SOURCE_GROUP	struct group_source_req

MCAST_LEAVE_GROUP struct group_req

MCAST_JOIN_SOURCE_GROUP et MCAST_LEAVE_SOURCE_GROUP sont utilisés pour se joindre à, et quitter, un groupe spécifique de source.

MCAST_LEAVE_GROUP est pris en charge, comme une facilité, pour abandonner toutes les sources auxquelles se sont joints un groupe et interface particuliers. Les opérations sont les mêmes que si la prise avait été fermée.

5.2 API évoluée (à état plein)

Il peut exister des mises en œuvre qui utilisent ioctl() pour cette API, et pour des raisons historiques, l'API ioctl() est documentée à la Section 8. L'API préférée utilise les nouvelles fonctions décrites ci-dessous.

5.2.1 Filtre Set Source

```
#include <netinet/in.h>

int setsourcefilter(int s, uint32_t interface,
                    struct sockaddr *group, socklen_t grouplen,
                    uint32_t fmode, uint_t numsrc,
                    struct sockaddr_storage *slist);
```

En cas de succès, la valeur 0 est retournée, et en cas d'échec, la valeur -1 est retournée et une errno est établie en conséquence.

L'argument s identifie la prise.

L'argument interface contient l'indice d'interface de l'interface.

L'argument group pointe soit sur une structure sockaddr_in (pour IPv4) soit sur une structure sockaddr_in6 (pour IPv6) qui contient l'adresse IP de diffusion groupée du groupe.

L'argument grouplen donne la longueur de la structure sockaddr_in ou sockaddr_in6.

L'argument fmode identifie le mode de filtre. La valeur de ce champ doit être soit MCAST_INCLUDE, soit MCAST_EXCLUDE, qui sont aussi définies dans <netinet/in.h>.

L'argument numsrc contient le nombre d'adresses de source dans le dispositif slist.

L'argument slist pointe sur une matrice d'adresses IP de sources ç inclure ou exclure selon le mode de filtre.

Si la mise en œuvre impose une limite au nombre maximum de sources dans un filtre de sources, ENOBUFS est généré lorsque l'opération excèderait le maximum.

5.2.2 Filtre Get Source

```
#include <netinet/in.h>

int getsourcefilter(int s, uint32_t interface,
                    struct sockaddr *group, socklen_t grouplen,
                    uint32_t fmode, uint_t *numsrc,
                    struct sockaddr_storage *slist);
```

En cas de succès, la valeur 0 est retournée, et en cas d'échec, la valeur -1 est retournée et une errno est établie en conséquence.

L'argument s identifie la prise.

L'argument interface contient l'indice d'interface de l'interface.

L'argument group pointe soit sur une structure sockaddr_in (pour IPv4) soit sur une structure sockaddr_in6 (pour IPv6) qui contient l'adresse IP de diffusion groupée du groupe.

L'argument fmode pointe sur un entier qui va contenir le mode de filtre si le retour est réussi. La valeur de ce champ sera soit

MCAST_INCLUDE, soit MCAST_EXCLUDE, qui seront aussi définies dans <netinet/in.h>.

En entrée, l'argument numsrc contient le nombre d'adresses de source qui vont tenir dans le dispositif slist. En sortie, l'argument numsrc va contenir le nombre total de sources dans le filtre.

L'argument slist pointe sur la mémoire tampon dans laquelle va être écrite une matrice d'adresses IP des sources incluses ou exclues (selon le mode de filtre). Si numsrc est 0 en entrée, un pointeur NUL peut être fourni.

Si l'application ne connaît pas par avance la taille de la liste de sources, elle peut faire une estimation raisonnable (par exemple, 0) et si une fois achevé, numsrc contient une valeur plus grande, l'opération peut être répétée avec une mémoire tampon suffisante. C'est à dire qu'en retour, numsrc est toujours mis à jour pour être le nombre total de sources dans le filtre, tandis que slist va contenir autant d'adresses de source qu'il en tient, jusqu'au minimum de la taille de la matrice passée dans la valeur originale de numsrc et le nombre total de sources dans le filtre.

6. Considérations pour la sécurité

Bien que le filtrage de source puisse aider à combattre les attaques de déni de service, il n'est pas par lui seul une solution complète, car il ne fournit pas de protection contre l'espionnage de l'adresse de source qui est une source autorisée. Les protocoles d'acheminement de diffusion groupée qui utilisent la transmission de chemin inverse fondée sur l'adresse de source fournissent bien cependant une protection naturelle contre l'espionnage de l'adresse de source, car si un routeur reçoit un paquet sur une interface autre que celle qui est tournée vers la source "réelle", il va éliminer le paquet. Cependant, cela ne fournit aucune garantie de protection.

7. Remerciements

Le présent document a été mis à jour sur la base des retours des groupes de travail IDMR et MAGMA de l'IETF, et du groupe Austin. Wilbert de Graaf a aussi fourni de nombreux commentaires utiles.

8. Utilisation de ioctl() pour les opérations à états pleins

L'API définie ici est historique, mais elle est documentée ici pour information car elle est mise en œuvre par plusieurs plates-formes. Les nouvelles fonctions définies plus haut dans ce document devraient maintenant être utilisées à sa place.

Restituer le filtre de source pour un groupe donné ne peut être fait avec getsockopt() sur certaines plates-formes existantes, car le groupe et l'interface doivent être parcourus afin de restituer le filtre correct, et getsockopt ne prend en charge qu'une mémoire tampon de sortie. Cela peut, cependant, être fait avec un ioctl(), et donc, pour la symétrie, les établissements et restitutions sont tous deux faits avec un ioctl.

8.1 Options IPv4

Ce qui suit est défini dans <sys/sockio.h>:

- ioctl() SIOCGIPMSFILTER : pour restituer la liste des adresses de source que comporte le filtre de source ainsi que le mode de filtre actuel.

- ioctl() SIOCSIPMSFILTER : pour établir ou modifier le contenu du filtre de source (par exemple, une liste d'adresses de source en envoi individuel) ou du mode (exclure ou inclure).

Option Ioctl	Type d'argument
SIOCGIPMSFILTER	struct ip_msfilter
SIOCSIPMSFILTER	struct ip_msfilter

```
struct ip_msfilter {
  struct in_addr imsf_multiaddr;        /* adresse IP de groupe de diffusion groupée */
  struct in_addr imsf_interface;        /* adresse IP locale d'interface */
  uint32_t       imsf_fmode;            /* mode filtre */
  uint32_t       imsf_numsrc;           /* nombre de sources dans src_list */
  struct in_addr imsf_slist[1];         /* début de la liste de sources */
```

```
};

#define IP_MSFILTER_SIZE(numsrc) \
  (sizeof(struct ip_msfilter) - sizeof(struct in_addr) \
  + (numsrc) * sizeof(struct in_addr))
```

Le mode imsf_fmode est un entier de 32 bits qui identifie le mode de filtre. La valeur de ce champ doit être soit MCAST_INCLUDE, soit MCAST_EXCLUDE, qui sont aussi définis dans <netinet/in.h>.

La longueur de structure sur laquelle on pointe doit être d'au moins IP_MSFILTER_SIZE(0) octets, et le paramètre imsf_numsrc devrait être réglé de telle sorte que IP_MSFILTER_SIZE(imsf_numsrc) indique la longueur de la mémoire tampon.

Si la mise en œuvre impose une limite au nombre maximum de sources dans un filtre de sources, ENOBUFS est généré lorsque une opération set dépasserait le maximum.

Le résultat d'une opération get (SIOCGIPMSFILTER) sera que les champs imsf_multiaddr et imsf_interface resteront inchangés, tandis que imsf_fmode, imsf_numsrc, et autant d'adresses de source qu'il en tient seront entrés dans la mémoire tampon de l'application.

Si l'application ne connaît pas par avance la taille de la liste des sources, elle peut faire une estimation raisonnable (par exemple, 0) et si après exécution, le champ imsf_numsrc contient une plus grande valeur, l'opération peut être répétée avec une mémoire tampon de taille suffisante. C'est à dire qu'au retour de SIOCGIPMSFILTER, imsf_numsrc est toujours mis à jour pour être le nombre total de sources dans le filtre, tandis que imsf_slist va contenir autant d'adresses de source qu'il en tient, jusqu'au minimum de la taille du dispositif donné dans la valeur d'origine de imsf_numsrc et le nombre total de sources dans le filtre.

8.2 Options indépendantes du protocole

Ce qui suit est défini dans <sys/sockio.h> :

- ioctl() SIOCGMSFILTER : pour restituer la liste des adresses de source que comprend le filtre de source avec le mode de filtre actuel.

- ioctl() SIOCSMSFILTER : pour établir ou modifier le contenu du filtre de source (par exemple, la liste des adresse de source en envoi individuel) ou du mode (exclure ou inclure).

Option Ioctl	**Type d'argument**
SIOCGMSFILTER	struct group_filter
SIOCSMSFILTER	struct group_filter

```
struct group_filter {
  uint32_t            gf_interface;       /* indice d'interface*/
  struct sockaddr_storage gf_group;       /* adresse de diffusion groupée */
  uint32_t            gf_fmode;           /* mode de filtre */
  uint32_t            gf_numsrc;          /* nombre de sources */
  struct sockaddr_storage gf_slist[1];    /* adresse de source */
};

#define GROUP_FILTER_SIZE(numsrc) \
  (sizeof(struct group_filter) - sizeof(struct sockaddr_storage) \
  + (numsrc) * sizeof(struct sockaddr_storage))
```

Le champ imf_numsrc est utilisé de la même façon que décrit ci-dessus pour imsf_numsrc.

9. Références

[IEEE-1003] Norme IEEE 1003.1-2001 "Standard for Information Technology -- Portable Operating System Interface (POSIX). Open Group Technical Standard: Base Specifications", 6 décembre 2001. ISO/CEI 9945:2002. http://www.opengroup.org/austin

[RFC3376] B. Cain et autres, "Protocole Internet de gestion de groupe, IGMP version 3", octobre 2002.

[RFC3493] R. Gilligan et autres, "Extensions d'interface de prise de base pour IPv6", février 2003. *(Information)*

[RFC3810] R. Vida, L. Costa, éditeurs, "Découverte d'écouteur de diffusion groupée version 2 (MLDv2) pour IPv6", juin 2004.

10. Adresse des auteurs

Dave Thaler
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399
téléphone : +1 425 703 8835
mél : dthaler@microsoft.com

Bill Fenner
75 Willow Road
Menlo Park, CA 94025
téléphone : +1 650 867 6073
mél : fenner@research.att.com

Bob Quinn
IP Multicast Initiative (IPMI)
Stardust.com
1901 S. Bascom Ave. #333
Campbell, CA 95008
téléphone : +1 408 879 8080
mél : rcq@ipmulticast.com

11. Déclaration de droits de reproduction

Remerciement
Le financement de la fonction d'édition des RFC est actuellement fourni par la Internet Society.