IX - La multidiffusion

La multidiffusion

- **unicast** : communication 1 à 1 — **TCP, UDP**
  i.e. communication entre deux points (entités)
- **broadcast** : communication 1 à tous — **UDP**
  i.e. une entité communique avec toutes les entités du sous-réseau
- **multicast** : communication 1 à un groupe d'entités — **UDP**
  i.e. une entité communique avec toutes les entités d'un même groupe du sous-réseau $\Rightarrow$ nécessité d'organiser la gestion du groupe
- **anycast** : communication 1 à n'importe qui — **UDP**
  i.e. une entité communique avec n'importe quelle entité du sous-réseau.
  Utile lorsque lorsqu'il y a plusieurs serveurs sur le sous-réseau proposant le même service et qu'une entité souhaite faire une requête et recevoir une réponse de n'importe lequel de ces serveurs

La multidiffusion

adresses IP

Les adresses IP allouées pour le multicast sont :

- en IPv4 : adresses entre `224.0.0.0` et `239.255.255.255`
  - Attention : certaines adresses sont réservées, et donc inutilisables
  - En pratique, évitez les adresses commençant par 224, 232, 233 et 239
- en IPv6 :
  - adresses `FF00::/8`

    `FF00::` → `FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF`
  - adresses **multicast local au lien** : `FF12::/16`

    `FF12::` → `FF12:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF`

    Ce sont les adresses que l'on peut utiliser (non allouées par l'IANA [1]) pour faire du multicast sur le réseau local

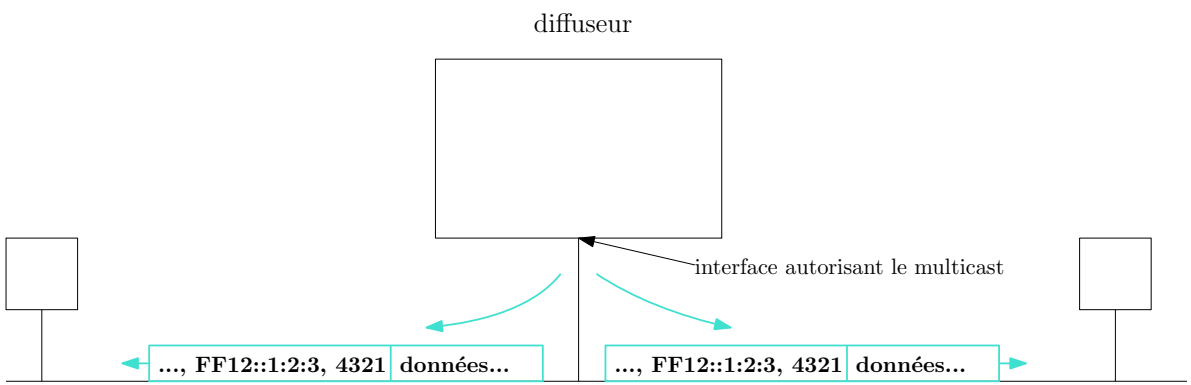
1. Internet Assigned Numbers Authority

La multidiffusion

diffuseur

Les étapes pour préparer une application faisant de la diffusion multicast sont

1. déclarer une socket utilisant le protocole UDP,
2. initialiser l'adresse multicast du groupe (IP + port),
3. récupérer l'index d'une interface locale autorisant le multicast
4. initialiser l'interface locale, par laquelle partiront les paquets multicast



La multidiffusion

diffuseur : socket et adresse

On commence par déclarer une socket utilisant le protocole UDP

```
sock = socket(AF_INET6, SOCK_DGRAM, 0);
```

Puis, on initialise l'adresse multicast du groupe (IP + port)

```
struct sockaddr_in6 grsock;
memset(&grsock, 0, sizeof(grsock));
grsock.sin6_family = AF_INET6;
inet_pton(AF_INET6, "ff12::1:2:3", &grsock.sin6_addr);
grsock.sin6_port = htons(4321);
```

C'est à cette adresse que les entités souhaitant recevoir les messages multicast du groupe devront s'abonner.

La multidiffusion

diffuseur : interface

Il faut maintenant choisir une interface locale par laquelle partiront les paquets multicast.

On interroge la configuration réseau avec la commande ip a et on choisit une interface permettant le multicast (local au lien)

```
lulu$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 100
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0@if12: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group defau
    link/ether 4a:49:43:49:79:bf brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.168.70.236/24 brd 192.168.70.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fdc7:9dd5:2c66:be86:4849:43ff:fe49:79bf/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::4849:43ff:fe49:79bf/64 scope link
       valid_lft forever preferred_lft forever
```

Sur **lulu**, **eth0** est la seule interface autorisant le **MULTICAST**.

La multidiffusion

diffuseur : interface

Il faut récupérer l'index de l'interface.

Pour cela, on fait appel à la fonction

```
unsigned int if_nametoindex(const char *ifname);
```

qui prend en paramètre un nom d'interface et retourne l'index s'il existe, 0 sinon.

```
int ifindex = if_nametoindex ("eth0");
```

Vous disposez également de la fonction inverse

```
char *if_indextoname(unsigned int ifindex, char *ifname);
```

La multidiffusion

diffuseur : interface

On peut alors initialiser l'interface locale, par laquelle partiront les paquets multicast, par l'appel à

```
if(setsockopt(sock, IPPROTO_IPV6, IPV6_MULTICAST_IF, &ifindex, sizeof(ifindex)))
    perror("erreur initialisation de l'interface locale");
```

qui permet

- de modfier l'option `IPV6_MULTICAST_IF` de la socket `sock`,
- avec la valeur de `ifindex` dont l'adresse est passée en paramètre,
- au niveau `IPPROTO_IPV6`.

Si on passe `ifindex` avec la valeur 0, alors c'est l'interface par défaut autorisant le multicast qui est attribuée.

La multidiffusion

diffuseur : envoi

On peut maintenant envoyer des messages multicast aux abonnés du groupe

```
int s = sendto(sock, buf, buflen, 0, (struct sockaddr*)&grsock, sizeof(grsock));
if (s < 0)
  perror("erreur send\n");
```

où **buf** contient **buflen** octets de données.

La multidiffusion

abonné

Les étapes pour créer une application s'abonnant à une adresse multicast sont

1. déclarer une socket utilisant le protocole UDP,
2. initialiser l'**adresse de réception** des paquets multicast,
3. lier cette adresse à la socket,
4. récupérer l'index d'une interface locale autorisant le multicast
5. initialiser l'**adresse IP d'abonnement** et l'interface locale sur laquelle seront reçus les paquets,
6. s'abonner au groupe.

La multidiffusion

abonné : socket et adresse de réception

On commence par déclarer une socket utilisant le protocole UDP.

Puis, on initialise l'**adresse de réception**

```
struct sockaddr_in6 grsock;
memset(&grsock, 0, sizeof(grsock));
grsock.sin6_family = AF_INET6;
grsock.sin6_addr = in6addr_any;
grsock.sin6_port = htons(4321);
```

et on lie cette adresse à la socket **sock** afin que cette dernière puisse écouter sur le port **4321** et permettre la réception de paquets à destination de n'importe quelle adresse IPv6 locale

```
if(bind(sock, (struct sockaddr*)&grsock, sizeof(grsock))) {
  perror("erreur bind");
  close(sock);
}
```

La multidiffusion

abonné : socket et adresse

Il reste à abonner l'entité au groupe multicast afin qu'elle reçoive exclusivement les paquets mutlidiffusés sur l'**adresse IP d'abonnement**.

a. initialiser la structure

```
struct ipv6_mreq {
  struct in6_addr ipv6mr_multiaddr;
  unsigned int    ipv6mr_interface;
};
```

avec

- `ipv6mr_multiaddr` l'adresse IPv6 du groupe
- `ipv6mr_interface` l'index d'une interface locale autorisant le multicast ou 0 si on veut l'interface multicast par défaut.

b. modfier l'option de la socket `sock`

- `IPV6_JOIN_GROUP`,
- avec la valeur de `group` dont l'adresse est passée en paramètre,
- au niveau `IPPROTO_IPV6`.

La multidiffusion

abonné : socket et adresse

```
struct ipv6_mreq group;
inet_pton (AF_INET6, "ff12::1:2:3", &group.ipv6mr_multiaddr.s6_addr);
group.ipv6mr_interface = ifindex;

if(setsockopt(sock, IPPROTO_IPV6, IPV6_JOIN_GROUP, &group, sizeof(group))<0){
  perror("erreur abonnement groupe");
  close(sock);
}
```

Si on veut pouvoir **recevoir** les messages du groupe **sur plusieurs interfaces**, il faut s'abonner pour chaque interface.

On peut alors recevoir les messages multicast adressés au groupe

```
if (read(sock, buf, BUF_SIZE) < 0)
  perror("erreur read");
```

On peut également utiliser `recvfrom` si l'on veut récupérer l'adresse de l'expéditeur

La multidiffusion

et en IPv4 ?

Pour passer de IPv6 à IPv4, en dehors de la modification des structures et constantes déjà rencontrées, il y a une différence lors de l'initialisation de l'interface locale.

On utilise la structure

```
struct ip_mreqn {
  struct in_addr imr_multiaddr;
  struct in_addr imr_address;
  int            imr_ifindex;
};
```

avec

- `imr_multiaddr` : adresse multicast du groupe
- `imr_address` : adresse IPv4 de l'interface locale (si `INADDR_ANY` alors une interface est choisie par le système)
- `imr_ifindex` : index de l'interface locale (si `0` alors une interface est choisie par le système)

La multidiffusion

et en IPv4 ? diffuseur

```
if((sock = socket(AF_INET, SOCK_DGRAM, 0)) < 0) return 1;

/* Initialisation de l'adresse d'abonnement */
struct sockaddr_in grsock;
memset(&grsock, 0, sizeof(grsock));
grsock.sin_family = AF_INET;
inet_pton(AF_INET, "225.1.2.3", &grsock.sin_addr);
grsock.sin_port = htons(4321);

struct ip_mreqn group;
memset(&group, 0, sizeof(group));
group.imr_multiaddr.s_addr = htonl(INADDR_ANY);
group.imr_ifindex = if_nametoindex ("eth0");

if(setsockopt(sock, IPPROTO_IP, IP_MULTICAST_IF, &group, sizeof(group))){
  perror("erreur initialisation de l'interface locale");
  return 1;
}
```

La multidiffusion

et en IPv4 ? abonné

```
if((sock = socket(AF_INET, SOCK_DGRAM, 0)) < 0) return 1;

struct sockaddr_in grsock;
memset(&grsock, 0, sizeof(grsock));
grsock.sin_family = AF_INET;
grsock.sin_addr.s_addr = htonl(INADDR_ANY);
grsock.sin_port = htons(4321);

if(bind(sock, (struct sockaddr*) &grsock, sizeof(grsock))) {
  close(sock);
  return 1;
}

/* s'abonner au groupe multicast */
struct ip_mreqn group;
memset(&group, 0, sizeof(group));
group.imr_multiaddr.s_addr = inet_addr("225.1.2.3");
group.imr_ifindex = if_nametoindex ("eth0");

if(setsockopt(sock, IPPROTO_IP, IP_ADD_MEMBERSHIP, &group, sizeof(group)) < 0) {
  perror("echec de abonnement groupe");
  close(sock);
  return 1;
}
```

La multidiffusion

complément

Si on souhaite avoir plusieurs instances d'une application écoutant sur le port multicast et recevant chacune les différents paquets, il faut activer l'option de socket SO_REUSEADDR :

```
int ok = 1;
if(setsockopt(sock, SOL_SOCKET, SO_REUSEADDR, &ok, sizeof(ok)) < 0) {
  perror("echec de SO_REUSEADDR");
  close(sock);
  return 1;
}
```