VIII - La multidiffusion

La multidiffusion

- **unicast** : communication 1 à 1 — **TCP, UDP**
  communication entre deux points (entités)
- **broadcast** : communication 1 à tous — **UDP**
  une entité communique avec toutes les entités du sous-réseau
- **multicast** : communication 1 à un groupe d'entités — **UDP**
  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**
  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

- **unicast** : communication 1 à 1 — TCP, UDP
  communication entre deux points (entités) — IPv4, IPV6
- **broadcast** : communication 1 à tous — UDP
  une entité communique avec toutes les entités du sous-réseau — IPv4
- **multicast** : communication 1 à un groupe d'entités — UDP
  une entité communique avec toutes les entités d'un même groupe du sous-réseau — IPv4, IPV6
  $\Rightarrow$ nécessité d'organiser la gestion du groupe
- **anycast** : communication 1 à n'importe qui — UDP
  une entité communique avec n'importe quelle entité du sous-réseau — IPV6

communication sur UDP $\Rightarrow$ `communication par paquets`, ce qui signifie

1 `sendto` $\leftrightarrow$ 1 paquet envoyé $\leftrightarrow$ 0 ou (1 paquet reçu $\leftrightarrow$ 1 `recv`)

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

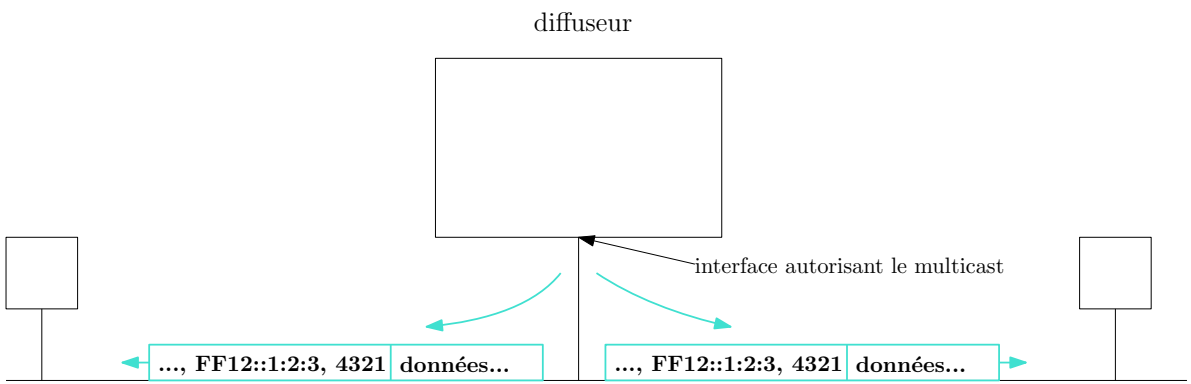
1. Internet Assigned Numbers Authority

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 gradr;
memset(&gradr, 0, sizeof(gradr));
gradr.sin6_family = AF_INET6;
inet_pton(AF_INET6, "ff12::1:2:3", &gradr.sin6_addr);
gradr.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 :

```
gadr.sin6_scope_id = ifindex; //ou 0 pour l'interface multicast par défaut
```

ou bien

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

qui permet

- de modifier 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*)&gradr, sizeof(gradr));
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. Pour la multidiffusion au niveau du lien local, il faut l'index de l'interface reliée à ce lien local.
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 adr;
memset(&adr, 0, sizeof(adr));
adr.sin6_family = AF_INET6;
adr.sin6_addr = in6addr_any;
adr.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*)&adr, sizeof(adr))) {
  perror("erreur bind");
  close(sock);
}
```

La multidiffusion

abonné : socket et adresse

Il reste à abonner l'entité au groupe multicast afin qu'elle reçoive 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. Pour la multidiffusion au niveau du lien local, il faut l'index de l'interface reliée à ce lien local.

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

- `IPV6_JOIN_GROUP`,
- avec le groupe multicast dont la référence 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);
group.ipv6mr_interface = if_nametoindex("eth0");
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

diffuseur/abonnés

diffuseur	abonnés
socket UDP : `socket()`	socket UDP : `socket()`
	`adresse de réception` : port d'écoute
	lier la socket et l'adresse de réception : `bind()`
adresse d'abonnement : IP + port + interface : `struct sockaddr_in6`	**adresse d'abonnement** : `struct ipv6_mreq`
	inscription : `setsockopt` et `IPV6_JOIN_GROUP`
multidiffusion : `sendto()` $\longrightarrow$	réception : `read()` ou `recv()` ou `recvfrom()`

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 gradr;
memset(&gradr, 0, sizeof(gradr));
gradr.sin_family = AF_INET;
inet_pton(AF_INET, "225.1.2.3", &gradr.sin_addr);
gradr.sin_port = htons(4321);


/* Initialisation de l'interface */
int ifindex = if_nametoindex("eth0");


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

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 ? diffuseur

En IPv4, si on veut choisir l'interface multicast *par défaut*, on peut juste activer l'option de socket `SO_BROADCAST` au niveau `SOL_SOCKET`.

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

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

/* Initialisation de l'interface par défaut */
int ok = 1;
if(setsockopt(sock, SOL_SOCKET, SO_BROADCAST, &ok, sizeof(ok))){
  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 adr;
memset(&adr, 0, sizeof(adr));
adr.sin_family = AF_INET;
adr.sin_addr.s_addr = htonl(INADDR_ANY);
adr.sin_port = htons(4321);

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

/* s'abonner au groupe multicast */
struct ip_mreqn group;
memset(&group, 0, sizeof(group));
inet_pton(AF_INET, "225.1.2.3", &group.imr_multiaddr);
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, sur une machine, 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;
}
```

IX - La diffusion intégrale (Broadcast)

Diffusion intégrale

Broadcast

- communication sur le protocole `UDP`
- $\Rightarrow$ `communication par paquets`, ce qui signifie

  1 `sendto` $\leftrightarrow$ 1 paquet envoyé $\leftrightarrow$ 0 ou (1 paquet reçu $\leftrightarrow$ 1 `recv`)
- la diffusion s'effectue en direction de toutes les machines d'un réseau donné
- possible uniquement en `IPv4`
- en IPv6, on peut utiliser l'adresse spéciale de multidiffusion `ff02::1` qui signifie *tous les noeuds du lien local*.

  multidiffusion sur l'adresse IPv6 `ff02::1` et port `P` $\leftrightarrow$ récepteurs UDP écoutant sur le port `P`

Diffusion intégrale

adresse

La diffusion intégrale s'effectue en envoyant un paquet sur la « dernière » adresse possible du réseau

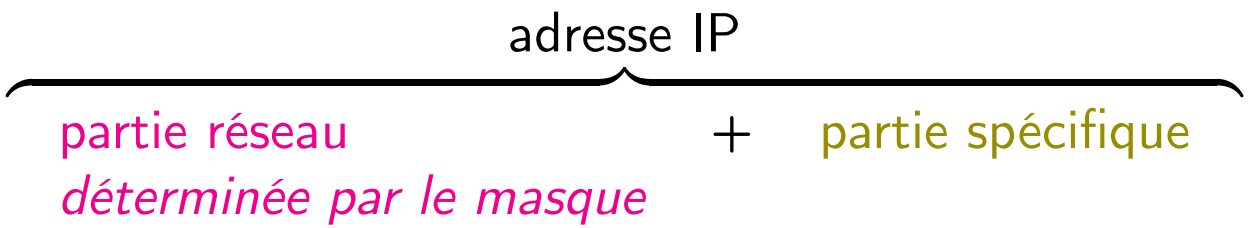
- l'adresse de broadcast du réseau `0.0.0.0` est donc `255.255.255.255`
- `0.0.0.0` est l'adresse générique du réseau local
- diffusion intégrale sur l'adresse `255.255.255.255` = diffusion intégrale sur le réseau local
  → *pas de routage* des paquets
  On parle de **broadcast limité**.

Comment diffuser intégralement en dehors du réseau local ?

Diffusion intégrale

adresse

On a besoin de connaître l'adresse du réseau sur lequel on veut diffuser.




```
lulu:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 ...
2: eth0@if12: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    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
```

L'adresse IPV4 de lulu est 192.168.70.236 et le masque est de 24

i.e. 24 bits (3 octets) de poids forts = adresse du réseau de lulu

$\Rightarrow$ adresse de broadcast de ce réseau : 192.168.70.255

On parle de **broadcast dirigé**.

Remarque : `brd` indique l'adresse de broadcast dans `ip a`

Diffusion intégrale

envoi

Pour envoyer des messages en diffusion intégrale, les étapes sont

1. créer une socket UDP IPv4
2. activer l'option de diffusion intégrale de la socket

   → option `SO_BROADCAST` au niveau `SOL_SOCKET`
3. initialiser l'adresse de diffusion : IP + port
4. envoyer les messages à l'adresse de diffusion

   → `sendto`

Diffusion intégrale

envoi

```
int main() {
  int sock=socket(PF_INET,SOCK_DGRAM,0);

  int ok=1;
  int r=setsockopt(sock,SOL_SOCKET,SO_BROADCAST,&ok,sizeof(ok));
  if(r == -1) exit(1);

  struct sockaddr_in adrdiff;
  memset(&adrdiff, 0, sizeof(adrdiff));
  adrdiff.sin_family = AF_INET;
  adrdiff.sin_port = htons(8888);
  r = inet_pton(AF_INET, "255.255.255.255", &adrdiff.sin_addr);
  if( r <= 0) exit(1);

  char buf[100];
  sprintf(buf,"bonjour la terre\n");
  r = sendto(sock, buf, strlen(buf), 0, (struct sockaddr *) &adrdiff, (socklen_t)sizeof(struct sockaddr_in));
  if(r < 0) exit(1);

  return 0;
}
```

Diffusion intégrale

réception

Pour recevoir des messages diffusés, les étapes sont

1. créer une socket UDP IPv4
2. initialiser l'adresse de réception avec le port d'écoute
3. lier la socket à l'adresse de réception
4. recevoir les messages

C'est un récepteur UDP standard.

Diffusion intégrale

réception

```
int main() {
  int sock=socket(PF_INET,SOCK_DGRAM,0);

  struct sockaddr_in address_sock;
  address_sock.sin_family=AF_INET;
  address_sock.sin_port=htons(8888);
  address_sock.sin_addr.s_addr=htonl(INADDR_ANY);

  int r = bind(sock,(struct sockaddr *)&address_sock,sizeof(struct sockaddr_in));
  if(r){
    perror("bind");
    exit(1);
  }

  struct sockaddr_in emet;
  socklen_t taille=sizeof(emet);
  char buf[100], adr[100];
  while(1){
    int rec=recvfrom(sock,buf,100,0,(struct sockaddr *)&emet,&taille);
    buf[rec]='\0';
    printf("Message recu : %s\n",buf);
    printf("Port de l'emetteur: %d\n",ntohs(emet.sin_port));
    inet_ntop(AF_INET, &(emet.sin_addr), adr, sizeof(adr));
    printf("Adresse de l'emetteur: %s\n\n", adr);
  }

  return 0;
}
```