Programmation réseau 9

Juliusz Chroboczek

8 avril 2020

Lors d'une communication téléphonique, les données émises sont reçues une seule fois, par un seul destinataire : on parle de communication 1-1 ou communication *unicast*. Lors de la diffusion d'un programme de télévision, par contre, les données sont émises une seule fois mais reçues plusieurs fois, par plusieurs récepteurs : on parle de communication 1-*n*, de *diffusion*, de *multidiffusion* ou encore de communcation *broadcast*.

1 Paradigmes de communication 1-*n*

En commutation de paquets, il existe trois paradigmes principaux de communication 1-*n* :

- le *broadcast*, où un paquet est émis à destination de toutes les interfaces du lien local, et les hôtes qui ne sont pas intéressés ignorent le paquet reçu ;
- le *multicast*, où un paquet est émis à destination d'un ensemble d'interfaces, appelé un *groupe multicast* ; c'est le destinataire qui décide s'il appartient au groupe (on dit qu'il est *abonné* au groupe), et l'émetteur n'en connait pas forcément tous les membres ;
- le *multi-unicast*, où un paquet est émis à destination d'un ensemble d'adresses unicast spécifié par l'émetteur ; une émission *multi-unicast* est donc équivalente à plusieurs émissions unicast, mais peut être plus efficace.

IPv4 supporte la communication *broadcast* depuis son inception, et, de ce fait, les vieux protocoles utilisent souvent le *broadcast* IPv4. Cependant, l'envoi d'un paquet *broadcast* réveille tous les nœuds du lien local, qui doivent analyser le paquet reçu pour décider s'il les intéresse : de ce fait, il consomme rapidement la batterie des nœuds mobiles (par exemple les téléphones portables). Le *multicast* n'a pas ce problème : une interface ne reçoit un paquet *multicast* que si elle s'est abonnée à un groupe qui, a priori, l'intéresse ; en 1989, IPv4 a été étendu pour supporter le *multicast* (RFC 1112), et les nouveaux protocoles utilisent le *multicast* plutôt que le *broadcast*.

En IPv6, la situation est plus simple : seul le *multicast* est supporté. Il existe un groupe *multicast* auquel tous les nœuds IPv6 sont automatiquement abonnés, qui peut être utilisé par les applications qui auraient besoin d'effectuer un *broadcast*.

Le *multi-unicast* n'est supporté ni par IPv4 ni par IPv6. Je ne le mentionne donc plus dans la suite de ce cours.

2 Applications de la communication 1-*n*

Les techniques de communication 1-*n* sont utiles pour deux types d'applications : les applications de diffusion, où le même flot de données est utile à plusieurs récepteurs simultanément (pensez par exemple à une partie de *football streamée* en temps réel), et les applications de découverte, où le destinataire ne connaît pas encore les adresses IP des interfaces auxquelles il s'adresse.

Streaming multicast La plupart des applications de *streaming* vidéo utilisent de la communication *unicast*, et cela pour deux raisons : tout d'abord, les utilisateurs aiment bien mettre en pause ou avancer les vidéos, et, ensuite, le *multicast* n'est pas déployé à l'échelle globale (voir paragraphe 3 ci-dessous). Il existe une exception importante à cela : la télévision sur IP.

Le type d'offre de connexion Internet grand public le plus courant est le *triple play*, où une seule liaison physique (ADSL ou fibre optique) sert à l'accès à l'Internet, à la téléphonie et à la télévision. Les trois types de traffic sont démultiplexés par un routeur sur site (*CPE (Customer Premises Equipment)*, souvent appelé *box* en France). Le fournisseur de *triple play* n'a qu'une seule infrastructure pour les trois types de traffic : le traffic Internet, le traffic de téléphonie et la télévision sont transmis sur le même réseau IP.

Il n'est naturellement pas raisonnable d'envoyer un flux *unicast* pour chaque chaîne de télévision à chaque abonné *triple play*. Pour éviter cela, les fournisseurs déploient une infrastructure *multicast*, et envoient un flux *multicast* par chaîne de télévision ; la légère pause que vous constatez parfois lorsque vous changez de chaîne est due au délai nécessaire pour s'abonner au bon groupe.

Découverte Lors d'un envoi *broadcast* ou *multicast*, l'émetteur n'a pas besoin de connaître l'ensemble des destinataires d'un paquet : il envoie le paquet, et c'est les destinataires qui décident s'ils sont intéressés. De ce fait, il est possible d'envoyer un paquet avant de connaître l'ensemble des destinataires ou leurs adresses IP.

Un bon exemple est le protocole DHCPv4. Lorsqu'un hôte se connecte à un nouveau lien (par exemple parce qu'il vient de *booter*, ou parce qu'il vient de s'associer à un lien *WiFi*), il envoie un paquet demandant qu'on lui attribue une adresse IPv4 aux serveurs DHCPv4 du lien. Bien sûr, le nouveau hôte ne connaît pas encore les adresses IPv4 de ces derniers ; il ne peut donc pas envoyer une requête *unicast*. Il envoie donc un paquet *broadcast* à destination du port UDP 67, sur lequel les serveurs DHCPv4 attendent des paquets. Comme le client DHCPv4 n'a pas encore d'adresse IPv4, la réponse ne peut pas se faire en *unicast* — nous verrons les sordides détails l'année prochaine.

En IPv6, DHCPv4 a été remplacé par les protocoles *RA* et *DHCPv6*, qui utilisent un groupe *multicast* auquel sont abonnés les serveurs, ce qui évite de réveiller tous les nœuds du lien local à chaque requête. Comme le client RA ou DHCPv6 possède déjà une adresse locale au lien (comme décrit dans un cours précédent), la réponse peut se faire en unicast [1].

Un autre exemple est le protocole *mDNS*, initialement développé par Apple sous le nom *Zeroconf* et ensuite présenté au public sous le nom *Bonjour*, qui sert par exemple à découvrir les imprimantes présentes sur le lien local. Un hôte désirant découvrir les imprimantes envoie une requête *mDNS* au groupe *multicast* affecté à *mDNS* ; si tout se passe bien, les imprimantes répondent, ce

1. Pour des raisons que nous verrons l'année prochaine, la réponse RA se fait généralement en multicast.

qui permet au hôte de découvrir leurs adresses IP [2]. (On pourrait argumenter que mDNS n'est pas très bien conçu : il y a un seul groupe pour tous les nœuds implémentant mDNS, ce qui fait qu'une requête les réveille tous, même ceux qui ne sont pas, par exemple, des imprimantes.) Le protocole mDNS utilise du *multicast* aussi bien en IPv4 qu'en IPv6.

3 Déploiement

Le *broadcast* est une technologie qui est locale au lien, et ne dépend donc pas d'une infrastructure globale : de ce fait, il peut être utilisé partout, même dans les réseaux locaux qui ne sont pas connectés à l'Internet. Il en est de même du *multicast* local au lien.

La situation est différente pour le *multicast* global : pour qu'un émetteur *multicast* puisse atteindre des nœuds qui ne sont pas sur le même lien que lui, il faut que les routeurs intermédiaires implémentent un protocole de routage *multicast*. Or, les fournisseurs n'ont aucun avantage commercial à déployer le *multicast*, et de ce fait le *multicast* ne fonctionne généralement pas au-delà du lien local.

IPv6 prévoit un cas intermédiaire, le *multicast local au site*, un type de multicast qui traverse les routeurs à l'intérieur d'un site [3] mais pas les routeurs sur l'Internet Global. À part dans le cas des réseaux de télévision sur IP, déjà mentionnés ci-dessus, je n'ai jamais vu de déploiement du multicast local au site.

Dans la suite de ce cours, je mentionne rapidement le *broadcast* IPv4, du fait de son importance historique, mais je me focalise principalement sur le *multicast* local au lien IPv6, qui est utile, relativement efficace, et facile à utiliser.

4 Programmation

Le protocole TCP maintient de l'état, et cet état est spécifique à une paire de pairs (deux nœuds) ; de ce fait, il n'est pas possible d'utiliser TCP pour de la communication 1-*n*. UDP, par contre, ne maintient aucun état, et se moque de savoir si un datagramme est destiné à un seul nœud ou plusieurs ; UDP est donc parfaitement adapté à la communication *broadcast* et *multicast*. (Je ne connais pas de protocoles de couche transport dédiés à la communication 1-*n*.)

4.1 *Broadcast* IPv4

Le *broadcast* IPv4 est inefficace — un paquet *broadcast* est reçu par toutes les interfaces du lien (même celles qui ne parlent pas IPv4), ce qui a notamment pour effet de réveiller les nœuds mobiles qui étaient en veille. De plus, il est difficile à utiliser sur les nœuds multihomés. Pour ces raisons, je ne recommande son utilisation que lorsqu'il s'agit d'interopérer avec des protocoles existants ; les nouveaux protocoles devraient utiliser le *multicast* IPv6 local au lien.

2. Si vous êtes sur un lien partagé avec vos camarades, tapez « `avahi-browse -a` » et moquez-vous des gens qui utilisent un Mac.

3. Personne n'est bien sûr de ce qu'est un site. L'UFR d'Informatique ? Le bâtiment Sophie-Germain ? Le campus ? L'Université ? Le réseau *Renater* ?

Pour émettre des paquets *broadcast*, il suffit de les envoyer à l'adresse `255.255.255.255`. Les paquets avec une destination *broadcast* sont émis sur une interface qui est choisie par le système. Pour spécifier explicitement l'interface sortante, on peut utiliser l'option de *socket* `SO_BINDTODEVICE`, déjà mentionnée dans le cours sur la communication locale au lien.

Pour recevoir les paquets *broadcast*, il n'y a rien de spécial à faire, il suffit d'écouter sur une *socket* liée au bon port. Il n'est pas possible de distinguer un paquet *broadcast* d'un paquet *unicast*, sauf en consultant l'adresse de destination, ce qui sera décrit au cours suivant ; attention cependant au *broadcast dirigé*, une technique obsolète de *broadcast* qui utilise une adresse destination différente de `255.255.255.255`. En pratique, il vaut mieux concevoir les protocoles *broadcast* pour que la sémantique d'un paquet ne dépende pas de son adresse destination.

4.2 *Multicast* IPv6

Je considère le *multicast* IPv4 comme obsolète, et je n'en parle donc pas.

4.2.1 Adressage

Les adresses de groupe *multicast* IPv6 sont dans le préfixe `ff00::/8`. Le premier octet vaut `0xff` (255) pour indiquer une adresse *multicast*. Le deuxième octet consiste de deux champs de 4 bits chacun. Le champ d'ordre haut contient des *flags* ; le seul qui nous intéresse est le *flag* d'ordre bas, qui vaut 0 pour une adresse officiellement attribuée [4], et 1 pour une adresse choisie localement (par exemple en la tirant au hasard). Le dernier demi-octet indique la portée de l'adresse : 2 pour une adresse locale au lien, 5 pour une adresse locale au site [5], et `0xE` (14) pour une adresse globale.

Par exemple, l'adresse

```
ff02::1:6
```

est une adresse *multicast* locale au lien affectée globalement. Par contre, l'adresse

```
ff15::229f:da04:bf76:b56b
```

est une adresse *multicast* locale au site affectée localement (j'ai tiré les 64 bits d'ordre bas au hasard).

Adresses *multicast* importantes Lors du lancement, chaque nœud IPv6 abonne chacune de ses interfaces au groupe local au lien `ff02::1` (« `ip6-allnodes` »), qui contient donc tous les nœuds IPv6 du lien ; ce groupe remplace donc l'adresse *broadcast* en IPv6. Il est notamment utile pour découvrir tous les nœuds d'un lien, par exemple quand on cherche à accéder à un nœud qui n'a pas encore été numéroté ou dont on a oublié l'adresse IP :

```
ping ff02::1%eth0
```

Un routeur (mais pas un hôte) abonne en outre toutes ses interfaces aux groupes `ff02::2` (tous les routeurs du lien) et `ff05::2` (tous les routeurs du site).

4. Ce qui est complètement idiot — il n'y a aucune raison de coordonner l'affectation des adresses *multicast* dans un espace de 120 bits, mais les gens adorent la bureaucracie. Voyez `https://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.xhtml`.

5. Même si personne ne sait exactement ce qu'est un site.

4.2.2 Émission de paquets *multicast*

Pour envoyer un paquet *multicast*, il n'y a rien de particulier à faire — il suffit de stocker une adresse *multicast* dans le champ `sin6_addr` de la destination lors d'un appel à `sendto` ou `sendmsg`. Si l'adresse destination est locale au lien (de la forme `ff`*x*`2:` …), il faut prendre soin de stocker un indice d'interface dans le champ `sin6_scope_id` :

```
int ifindex;
unsigned char maddr[16];

rc = inet_pton(AF_INET6, "ff12::2c9f:eb92:5977:6a91", maddr);
if(rc < 1)
    abort();

ifindex = if_nametoindex("eth0");
if(ifindex <= 0)
    ...

struct sockaddr sin6;
memset(&sin6, 0, sizeof(sin6));
sin6.sin6_family = AF_INET6;
memcpy(&sin6.sin6_addr, maddr, 16);
sin6.sin6_port = htons(port);
sin6.sin6_scope_id = ifindex;
rc = sendto(..., &sin6, sizeof(sin6));
if(rc < 0)
    ...
```

Remarquez qu'il n'est pas nécessaire d'être abonné à un groupe pour y envoyer des paquets.

Il existe plusieurs options qui permettent de paramétrer l'émission. La plus importante est l'option `IPV6_MULTICAST_LOOP`, qui spécifie si les paquets émis sont reçus par l'émetteur si celui-ci est lui-même abonné au groupe ; il est souvent utile de la mettre à 0 pour éviter de recevoir ses propres paquets.

L'option `IPV6_MULTICAST_HOPS` permet de limiter la portée des paquets émis. Son paramètre spécifie le nombre de routeurs qu'un paquet va traverser avant d'être éliminé ; en particulier, s'il vaut 1, le communication est restreinte au lien local. Cette option n'est utile que pour un groupe de portée strictement supérieure à 2, cependant il est habituel de la fixer à 1 même dans ce cas.

4.2.3 Réception de paquets *multicast*

Pour recevoir un paquet destiné à un groupe *multicast*, il faut non seulement écouter sur une *socket* liée au bon port, mais aussi être abonné. On s'abonne une interface à un groupe à l'aide de l'option `IPV6_JOIN_GROUP`, qui prend en paramètre une structure `ipv6_mreq` :

```
struct ipv6_mreq {
```

```
        struct in6_addr ipv6mr_multiaddr;
        unsigned        ipv6mr_interface;
};
```

Le champ `ipv6mr_multiaddr` contient l'adresse du groupe *multicast* auquel on veut s'abonner. Le champ `ipv6mr_interface` contient l'indice de l'interface qu'on veut abonner (il est possible d'abonner plusieurs interfaces au même groupe).

```
struct ipv6_mreq mreq;
unsigned char maddr[16];
int ifindex;

rc = inet_pton(AF_INET6, "ff12::2c9f:eb92:5977:6a91", maddr);
if(rc < 1)
    abort();

ifindex = if_nametoindex("eth0");
if(ifindex <= 0)
    ...

memset(&mreq, 0, sizeof(mreq));
memcpy(&mreq.ipv6mr_multiaddr, maddr, 16);
mreq.ipv6mr_interface = ifindex;
rc = setsockopt(s, IPPROTO_IPV6, IPV6_JOIN_GROUP,
                &mreq, sizeof(mreq));
if(rc < 0) ...
```

La réception des paquets *multicast* se fait normalement, à l'aide de `recvfrom` ou `recvmsg`; dans le cas d'un groupe local au lien, l'indice de l'interface entrante est stockée dans le champ `sin6_scope_id` de la `struct sockaddr_in6` retournée par l'appel (comme dans le cas d'un paquet local au lien *unicast*).

Il n'est pas possible de distinguer un paquet *multicast* d'un paquet *unicast* autrement qu'en discriminant sur l'adresse de destination (ce que nous verrons au cours suivant). Il est donc recommandé de concevoir les protocoles de façon à ce que la sémantique d'un paquet ne dépende pas de son adresse de destination.