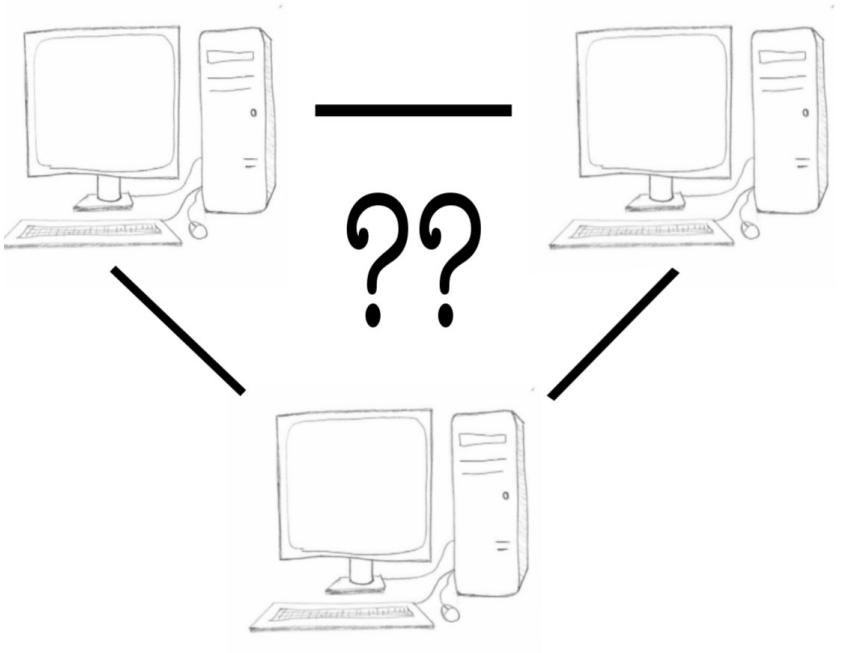
PROGRAMMATION RÉSEAU

Arnaud Sangnier
sangnier@liafa.univ-paris-diderot.fr

Broadcast & Multicast



La diffusion

- Jusqu'à présent : communication point à point
- Il peut être utile ou nécessaire de vouloir atteindre plusieurs destinataires
  - C'est ce qu'on appelle la **diffusion**
- Quel mode de communication pour la diffusion ?
  - Établir une connexion point à point fiable est déjà coûteux et difficile
  - La diffusion se fera donc uniquement en **communication par paquet**
  - La diffusion est donc *'non fiable'*
  - On ne sait pas si un paquet émis est reçu

Communication par diffusion

Types de diffusion

- Deux façon de faire de la diffusion :
  - **Diffusion intégrale** (**broadcast**)
    - La diffusion s'effectue en direction de toutes les machines d'un réseau donné
    - Diffusion d'un message à tout le monde
    - Parallèle possible : sirène d'incendie
  - **Multi-diffusion** (**multicast**)
    - La diffusion s'effectue en direction d'un groupe de machines qui se sont **abonnées**
    - Seuls les abonnés reçoivent donc les messages
    - Parallèle possible : télévision, radio

Quelles adresses pour la diffusion ?

- Une adresse IP est classiquement divisée en deux parties :
  - Les bits les plus à gauche caractérisent les réseau
  - Les bits les plus à droite caractérisent les machines dans le réseau
- Les adresses IPv4 sont divisés en 5 classes
- **Les adresses de classe A** :
  - Elles vont de 0.0.0.0 à 127.255.255.255
  - Le bit le plus à gauche de ces adresses est 0
  - 8 premiers bits utilisés pour l'adresse du réseau
  - 24 bits suivants pour les membres du réseau
- **Les adresses de classe B** :
  - Elles vont de 128.0.0.0 à 191.255.255.255
  - Les deux bits les plus à gauche sont 10
  - 16 premiers bits utilisés pour l'adresse du réseau
  - 16 bits suivants pour les membres du réseau

Quelles adresses pour la diffusion ?

- **Les adresses de classe C** :
  - Elles vont de 192.0.0.0 à 223.255.255.255
  - Les trois bits les plus à gauche de ces adresses sont 110
  - 24 premiers bits utilisés pour l'adresse du réseau
  - 8 bits suivants pour les membres du réseau
- **Les adresses de classe D** :
  - Elles vont de 224.0.0.0 à 239.255.255.255
  - Les quatre bits les plus à gauche sont 1110
  - **Ce sont les adresses utilisées pour la multi-diffusion**
  - **ATTENTION :** Certaines adresses sont réservées, et donc inutilisables
  - En pratique, évitez les adresses commençant par 224, 232, 233 et 239

Le broadcast

- La diffusion intégrale s'effectue en envoyant un paquet sur la « dernière » adresse possible du réseau
- L'alias pour désigner l'adresse de diffusion intégrale pour n'importe quelle réseau est donc l'adresse :
  - **255.255.255.255**
- En envoyant un message là, on envoie en théorie à toutes les machines connectés via Internet ....
- En fait, cet envoi est limité au réseau local

  -> *Lorsque l'on fait du broadcast, il n'y a pas de routage*

Sortir du réseau local

- En fait Si
- Il faut pour cela connaître l'adresse de broadcast du réseau local

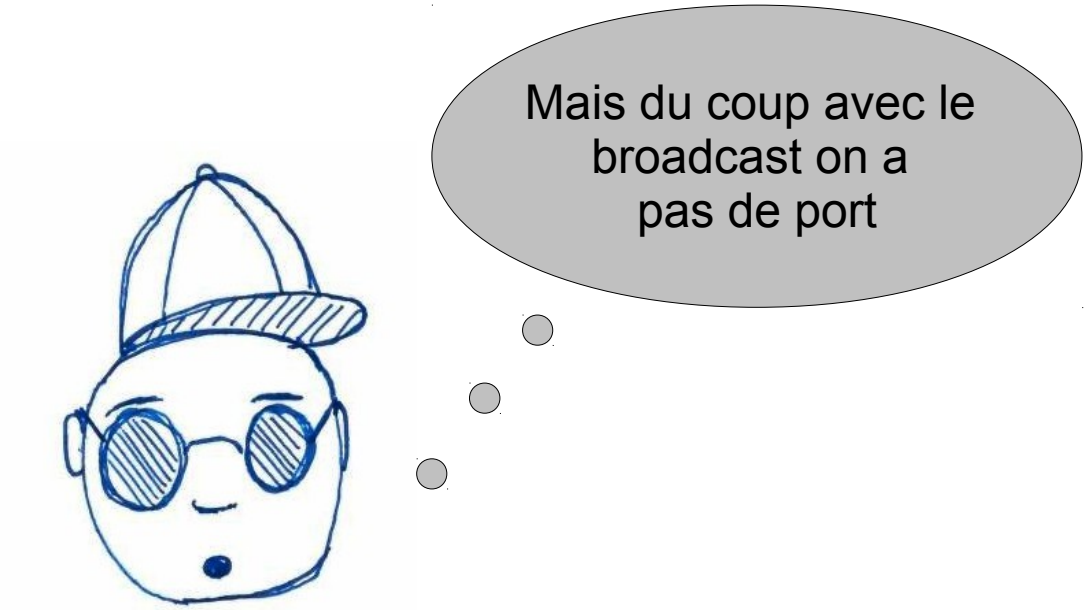
Déterminer l'adresse de broadcast

- Comme on l'a dit, chaque adresse IP vient avec
  - une partie réseau (un certain nombre de bits à gauche)
  - le reste sert pour les machines dans le réseau
- En fait, chaque réseau local vient avec un mask
- Ce mask précise quelles sont les bits des adresses du réseau qui correspondent à l'adresse réseau(indiqué par / après l'adresse)
  - Par exemple
    - Pour l'adresse **127.50.24.0/24**
    - Les 24 premiers bits correspondent à l'adresse du réseau
    - **125.50.24** correspond à l'adresse du réseau

Déterminer l'adresse de broadcast (2)

- Pour obtenir l'adresse de broadcast du réseau
  - On prend les bits correspondant à l'adresse du réseau
  - On rajoute des bits à 1 pour avoir l'adresse à 1
- Pour le réseau **127.50.24.0/24**
  - L'adresse de broadcast est donc : **127.50.24.255**
- Pour le réseau **127.50.24.0/23**
  - L'adresse de broadcast est donc : **127.50.25.255**
- En pratique, nous prendrons comme adresse de broadcast :
  - **255.255.255.255**
  - C'est l'adresse de broadcast du réseau **0.0.0.0**
  - **0.0.0.0** est le réseau sur lequel on est connecté

Où écoute-t-on ?



- Si, il faut aussi dire sur quel port on fait broadcast
- Si on broadcast sur 255.255.255.255
- Toutes les machines écoutant sur le port précisé, recevrons le message

Broadcast en Java

- En fait, le broadcast en Java se passe comme la communication point à point UDP
- La différence c'est que quand on envoie un paquet, on l'envoie sur l'adresse **255.255.255.255**
- Avant on envoyait sur l'adresse de la machine qui était connectée
- Pour la réception, on ne change rien

Rappel : Envoi de paquets

- Pour envoyer des paquets, on n'a pas besoin d'attacher la socket à un port
- On met l'adresse et le port du destinataire dans le paquet

```
String s="MESSAGE "+i+" \n";
byte[]data = s.getBytes();
InetSocketAddress ia=new InetSocketAddress("localhost",5555);
DatagramPacket paquet=new DatagramPacket(data,data.length,ia);
```

- Ou encore :

```
String s="MESSAGE "+i+" \n";
byte[]data = s.getBytes();
DatagramPacket paquet=new DatagramPacket(data,data.length,
                                InetAddress.getByName("localhost"),5555);
```

- **Ici on ne mettra plus l'adresse de la machine, mais 255.255.255.255**
- **ATTENTION :** sur une même machine on ne pourra pas écouter sur le même port

Exemple envoi broadcast

```
import java.io.*;
import java.net.*;

public class EnvoiBroadcast {
    public  static void main(String[] args){
        try{
            DatagramSocket dso=new DatagramSocket();
            byte[]data;
            for(int i=0;i <= 10; i++){
                Thread.sleep(1000);
                String s="MESSAGE "+i+" \n";
                data=s.getBytes();
                InetSocketAddress ia=new
                     InetSocketAddress("255.255.255.255",8888);
                DatagramPacket paquet=new
                     DatagramPacket(data,data.length,ia);
                dso.send(paquet);
            }
        } catch(Exception e){
            e.printStackTrace();
        }
    }
}
```

Exemple réception

```
import java.io.*;
import java.net.*;

public class ReceiveBroadcast {
    public  static void main(String[] args){
        try{
            DatagramSocket dso=new DatagramSocket(8888);
            byte[]data=new byte[100];
            DatagramPacket paquet=new DatagramPacket(data,data.length);
            while(true){
                dso.receive(paquet);
                String st=new
                    String(paquet.getData(),0,paquet.getLength());
                System.out.println("J'ai reçu :"+st);
            }
        } catch(Exception e){
            e.printStackTrace();
       }
    }
}
```

Broadcast en C

- En C, pour l'adresse d'envoi on utilise la macro **INADDR_BROADCAST**
- Il faut préciser que la socket doit pouvoir envoyer en diffusion intégrale
  - On utilise la fonction setsockopt
- En pratique :

```
int ok=1;
int r=setsockopt(sock,SOL_SOCKET,SO_BROADCAST,&ok,sizeof(ok));
```

où sock est la socket UDP sur laquelle on souhaite envoyer

- setsockopt renvoie 0 si tout se passe bien
- L'option **SO_BROADCAST** autorise la socket à émettre en broadcast
- Si on ne fait pas cela, cela ne marche pas

Exemple envoi broadcast

```
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==0){
    struct addrinfo *first_info;
    struct addrinfo hints;
    memset(&hints, 0, sizeof(struct addrinfo));
    hints.ai_family = AF_INET;
    hints.ai_socktype=SOCK_DGRAM;
    r=getaddrinfo("255.255.255.255","8888",NULL,&first_info);
    if(r==0){
      if(first_info!=NULL){
        struct sockaddr *saddr=first_info->ai_addr;
        char tampon[100];
        int i=0;
        for(i=0;i<=10;i++){
          strcpy(tampon,"MESSAGE ");
          char entier[3];
          sprintf(entier,"%d",i);
          strcat(tampon,entier);
          sendto(sock,tampon,strlen(tampon),0,saddr,(socklen_t)sizeof(struct
sockaddr_in));
        }
      }
    }
  }
  return 0;
}
```

Exemple réception

```
int main() {
  int sock=socket(PF_INET,SOCK_DGRAM,0);
  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==0){
    char tampon[100];
    while(1){
      int rec=recv(sock,tampon,100,0);
      tampon[rec]='\0';
      printf("Message recu : %s\n",tampon);
    }
  }
  return 0;
}
```

Pour le multicast

- Pour le multicast, il faut choisir une adresse de multi-diffusion
  - Adresses de **classe D** comprises entre **224.0.0.0** à **239.255.255.255**
  - **ATTENTION** elles ne sont pas toutes disponibles
- Là aussi, il faut choisir un port
- Du côte de l'émetteur
  - on envoie des paquets UDP sur l'adresse et le port choisi
- Du côté des récepteurs
  - Il faut s'abonner à l'adresse de multi-diffusion
  - Tous les abonnés qui écoutent reçoivent les paquets envoyés

Le multicast en Java

- Du côté du récepteur
  - Au lieu de prendre une **DatagramSocket**
  - On prendra une classe **MulticastSocket** qui hérite de **DatagramSocket**
  - Constructeur
    - **MulticastSocket(int port)**
    - le port sera le port de multi-diffusion
  - On pourra alors utiliser la méthode
    - **void joinGroup(InetAddress mcastaddr)**
    - elle permet de rejoindre le groupe correspondant à l'adresse de multi-diffusion donné en argument
  - On peut aussi quitter un groupe
    - **void leaveGroup(InetAddress mcastaddr)**
- Pour l'émetteur, comme en UDP

Exemple envoi multicast

```
import java.io.*;
import java.net.*;

public class EnvoiMulticast {
    public  static void main(String[] args){
        try{
            DatagramSocket dso=new DatagramSocket();
            byte[]data;
            for(int i=0;i <= 10; i++){
                //Thread.sleep(1000);
                String s="MESSAGE "+i+" \n";
                data=s.getBytes();
                InetSocketAddress ia=new InetSocketAddress("225.1.2.4",9999);
                DatagramPacket paquet=new
                        DatagramPacket(data,data.length,ia);
                dso.send(paquet);
            }
        } catch(Exception e){
            e.printStackTrace();
        }
    }
}
```

Exemple réception multicast

```
import java.io.*;
import java.net.*;

public class ReceiveMulticast {
    public  static void main(String[] args){
        try{
            MulticastSocket mso=new MulticastSocket(9999);
            mso.joinGroup(InetAddress.getByName("225.1.2.4"));
            byte[]data=new byte[100];
            DatagramPacket paquet=new DatagramPacket(data,data.length);
            while(true){
                mso.receive(paquet);
                String st=new String(paquet.getData(),0,paquet.getLength());
                System.out.println("J'ai reçu :"+st);
            }
        } catch(Exception e){
            e.printStackTrace();
        }
    }
}
```

Il peut être nécessaire de compiler avec l'option -Djava.net.preferIPv4Stack=true.

Le multicast en C

- Il faut travailler au niveau du récepteur
- Tout d'abord il faut abonner à la socket à une adresse de multiplexage
- Pour cela on rempli une structure du type suivant :
  - **struct ip_mreq {**
    struct in_addr imr_multiaddr; /* IP multicast address of group */
    struct in_addr imr_interface; /* local IP address of interface */
    };
- Dans le premier champ, on mettra l'adresse de multi-diffusion
- Dans le deuxième, on laisse le système choisir

```
struct ip_mreq mreq;
mreq.imr_multiaddr.s_addr=inet_addr("239.0.0.1");
mreq.imr_interface.s_addr=htonl(INADDR_ANY);
```

Le multicast en C (2)

- Il faut ensuite faire le lien entre la socket et l'abonnement

```
r=setsockopt(sock,IPPROTO_IP,IP_ADD_MEMBERSHIP,&mreq,sizeof(mreq));
```

- Au niveau du protocole **(IPPROTO_IP**) de la socket **sock** on précise que l'on veut s'abonner au groupe (**IP_ADD_MEMBERSHIP**)
- Pour autoriser plusieurs clients sur une même machine à se joindre au groupe, on doit le dire avant en faisant

```
int ok=1 ;
r=setsockopt(sock,SOL_SOCKET,SO_REUSEPORT,&ok,sizeof(ok));
```

- PARFOIS, il faut mettre SO_REUSEADDR à la place de SO_REUSEPORT (ex salle TP)
- Pour le reste tout se passe, comme un récepteur UDP
  - **bind** de la socket sur une adresse contenant le port etc

Exemple envoi multicast

```
int main() {
  int sock=socket(PF_INET,SOCK_DGRAM,0);
  struct addrinfo *first_info;
  struct addrinfo hints;
  memset(&hints, 0, sizeof(struct addrinfo));
  hints.ai_family = AF_INET;
  hints.ai_socktype=SOCK_DGRAM;
  int r=getaddrinfo("225.1.2.4","9999",NULL,&first_info);
  if(r==0){
    if(first_info!=NULL){
      struct sockaddr *saddr=first_info->ai_addr;
      char tampon[100];
      int i=0;
      for(i=0;i<=10;i++){
        strcpy(tampon,"MESSAGE ");
        char entier[3];
        sprintf(entier,"%d",i);
        strcat(tampon,entier);
        sendto(sock,tampon,strlen(tampon),0,saddr,(socklen_t)sizeof(struct
sockaddr_in));
      }
    }
  }
  return 0;
}
```

Exemple réception multicast

```
int main() {
  int sock=socket(PF_INET,SOCK_DGRAM,0);
  sock=socket(PF_INET,SOCK_DGRAM,0);
  int ok=1;
  int r=setsockopt(sock,SOL_SOCKET,SO_REUSEPORT,&ok,sizeof(ok));
  struct sockaddr_in address_sock;
  address_sock.sin_family=AF_INET;
  address_sock.sin_port=htons(9999);
  address_sock.sin_addr.s_addr=htonl(INADDR_ANY);
  r=bind(sock,(struct sockaddr *)&address_sock,sizeof(struct sockaddr_in));
  struct ip_mreq mreq;
  mreq.imr_multiaddr.s_addr=inet_addr("225.1.2.4");
  mreq.imr_interface.s_addr=htonl(INADDR_ANY);
  r=setsockopt(sock,IPPROTO_IP,IP_ADD_MEMBERSHIP,&mreq,sizeof(mreq));
  char tampon[100];
  while(1){
    int rec=recv(sock,tampon,100,0);
    tampon[rec]='\0';
    printf("Message recu : %s\n",tampon);
  }
  return 0;
}
```