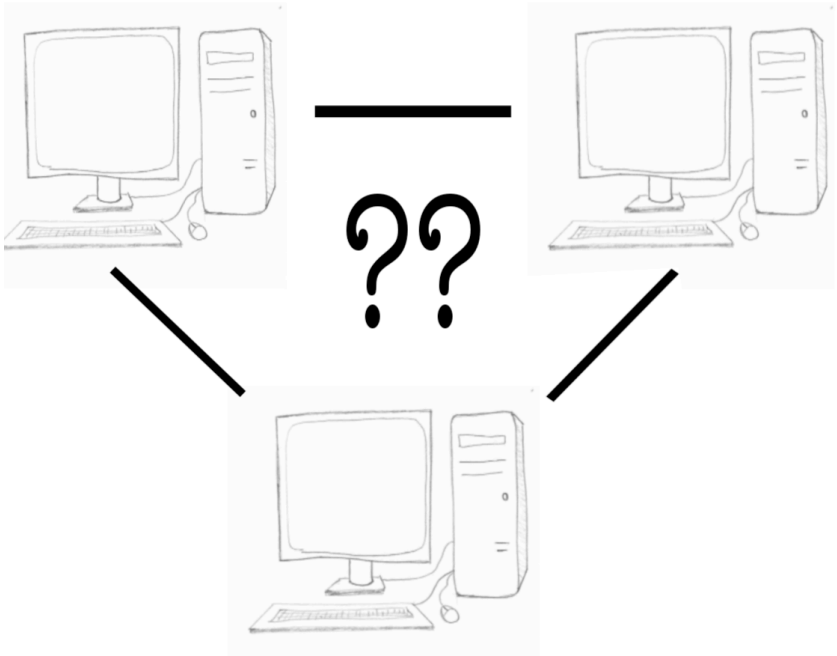
PROGRAMMATION RÉSEAU

Arnaud Sangnier
sangnier@irif.fr

Multicast



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++){
                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 RecoitMulticast {
    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 d'exécuter 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);
  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;
}
```