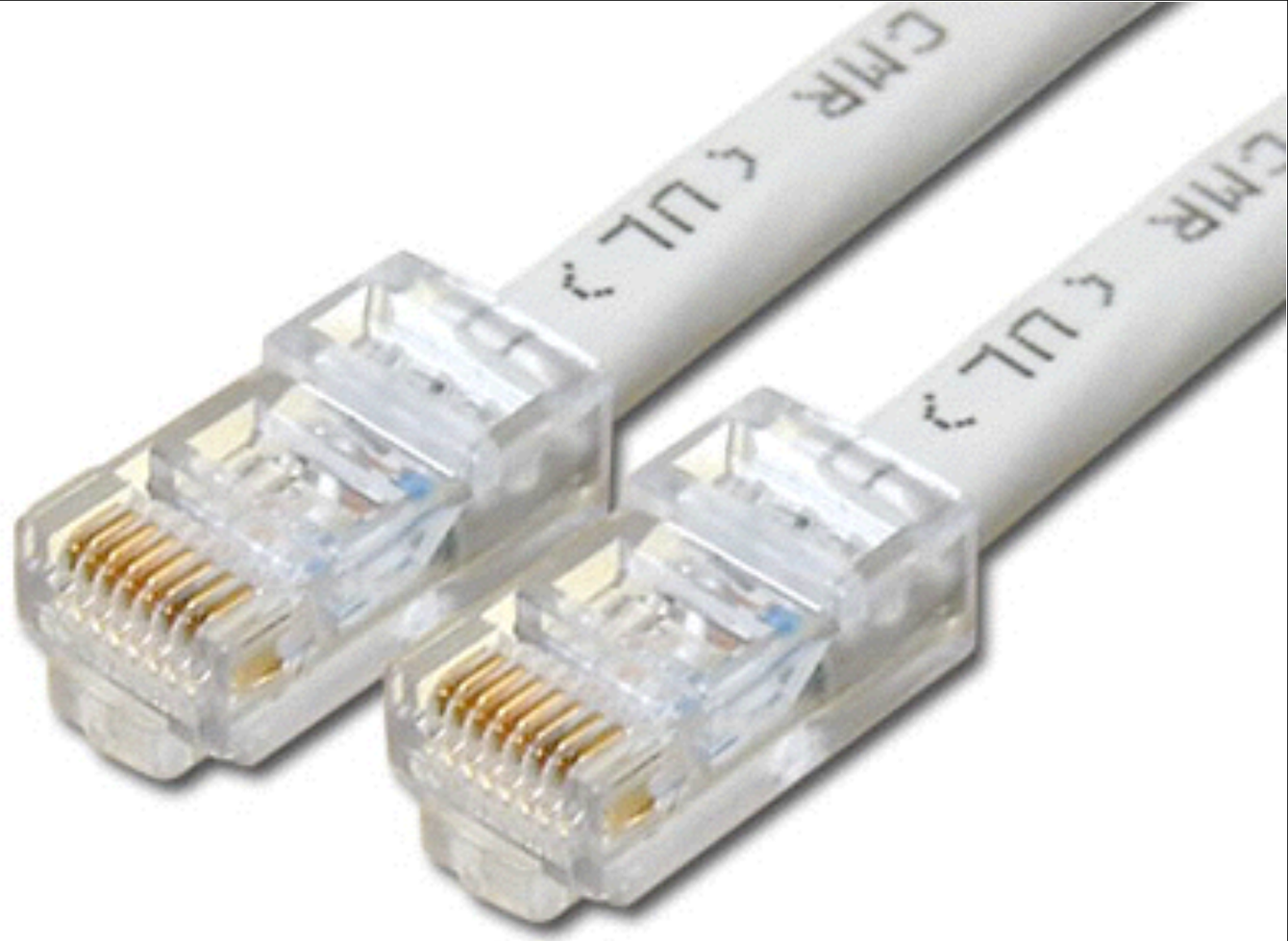
Programmation Réseau

# API Java UDP



Jean-Baptiste.Yunes@univ-paris-diderot.fr

UFR Informatique

2013-2014

université PARIS DIDEROT PARIS 7

# La transmission de paquets en Java

- pré-requis : le package `java.io`
- toujours dans `java.net` on trouve les classes :
  - `DatagramSocket`
  - `DatagramPacket`

- la classe `DatagramSocket` représente un point de réception ou un point d'envoi
  - pour la réception d'un paquet, il est nécessaire d'attacher la socket à un port donné sur une adresse (ou toutes) de la machine
- soit en construisant une socket attachée
- soit en construisant une socket puis en l'attachant

- les principaux constructeurs sont :
  - `DatagramSocket(int port);`
  - `DatagramSocket(SocketAddress sa);`
  - `DatagramSocket(int port,`
    `InetAdress adresse);`

- `DatagramSocket(int port);`
  - permet de construire une socket pour réception de paquets attachée au `port` donné sur toutes les adresses de la machine
- les autres constructeurs permettent de spécifier l'adresse de l'attachement en particulier

- l'attachement d'une socket non déjà attachée s'effectue par un appel à

```
void bind(SocketAddress adresse);
```

- où la classe `SocketAddress` est abstraite; mais possède une sous-classe concrète `InetSocketAddress` :

- `InetSocketAddress(int port);`
- `InetSocketAddress(String nom, int port)`

- on notera que si l'on souhaite réaliser des résolutions de nom, la classe `InetAddress` possède différentes méthodes statiques utilisables à cette fin :

  - `InetAddress getByName(String nom);`
  - `InetAddress []getAllByName(String nom);`
  - …

- la réception (bloquante) d'un paquet s'effectue par appel à la méthode

```
void receive(DatagramPacket paquet);
```

- où la classe `DatagramPacket` représente un message envoyé

- la classe `DatagramPacket` possède (entre autres) le constructeur

```
DatagramPacket(byte [],int longueur);
```

- et les méthodes

```
byte [] getData();

int getLength();

int getPort();

InetAddress getAddress();
```

- pour l'envoi d'un paquet, il n'est pas nécessaire d'attacher la socket qui peut être simplement construite à l'aide de

```
DatagramSocket();
```

- l'envoi s'effectue par appel à

```
void send(DatagramPacket paquet);
```

- où le paquet doit désigner la destination du message...

- dans ce cas le `DatagramPacket` peut être construit par emploi de

```
DatagramPacket(byte []data,
               int longueur,
               SocketAddress destination);
```

```
DatagramPacket(byte []data,
               int longueur,
               InetAddress addresse,
               int port);
```

- Attention, on notera que si l’envoi réussit, cela ne constitue en AUCUN cas une preuve de bonne réception…
  - dans le mode paquet, les paquet peuvent être perdus…
    - pas de receveur ou receveur non atteignable
    - trop-plein d’envoi
  - ou peuvent se doubler…

```java
import java.net.*;

public class Serveur {
  public static void main(String []args) {
    try {
      DatagramSocket s = new DatagramSocket(5678);
      byte []data = new byte[100];
      DatagramPacket paquet = new DatagramPacket(data,data.length);
      while (true) {
        s.receive(paquet);
        String st = new String(paquet.getData(),0,paquet.getLength());
        System.out.print("J'ai reçu ["+st+"]");
        System.out.println(" depuis la machine "+
              paquet.getAddress().getCanonicalHostName()+
              paquet.getPort());
      }
    } catch(Exception ex) {
      ex.printStackTrace();
      System.exit(1);
    }
  }
}
```

```
import java.net.*;

public class Client {
  public static void main(String []args) {
    try {
      DatagramSocket s = new DatagramSocket();
      byte []data = args[0].getBytes();
      InetSocketAddress sa = new InetSocketAddress("localhost",5678);
      DatagramPacket paquet = new DatagramPacket(data,data.length,sa);
      s.send(paquet);
    } catch(Exception ex) {
      ex.printStackTrace();
      System.exit(1);
    }
  }
}
```