# Programmation Concurrentielle

Deuxième Année Cycle d’ingénieur IRISI

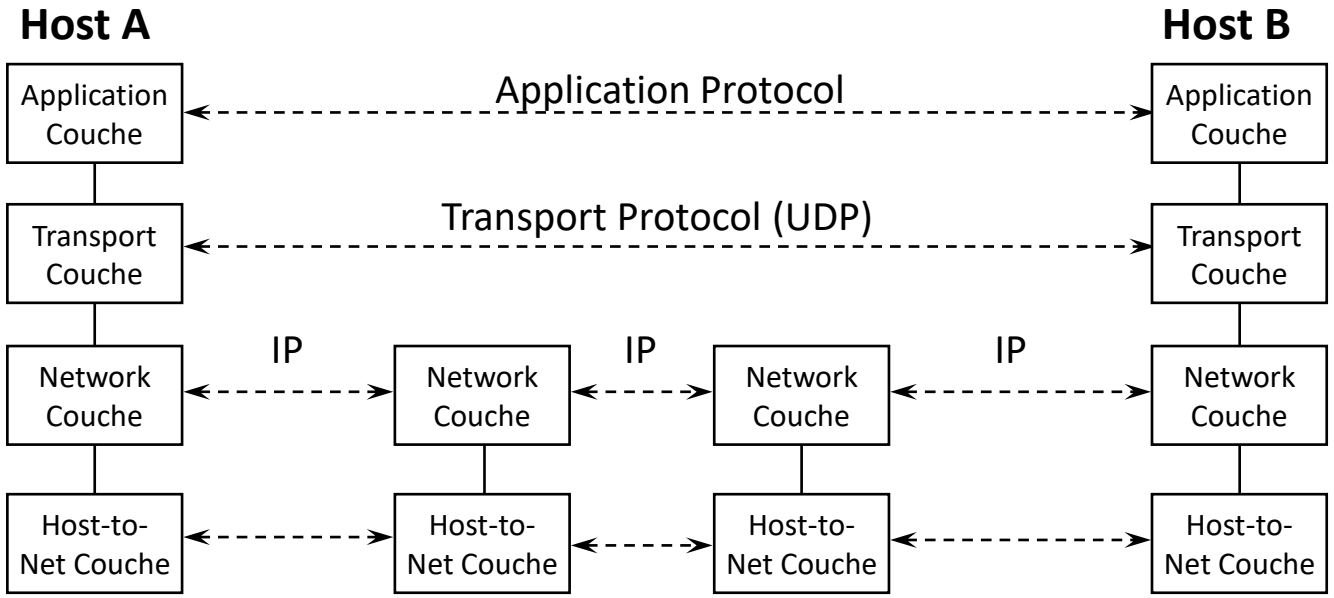

كليـــــة العلـــوم
والتقنيات - مراكش
FACULTE DES SCIENCES
ET TECHNIQUES - MARRAKECH

# Les sockets java

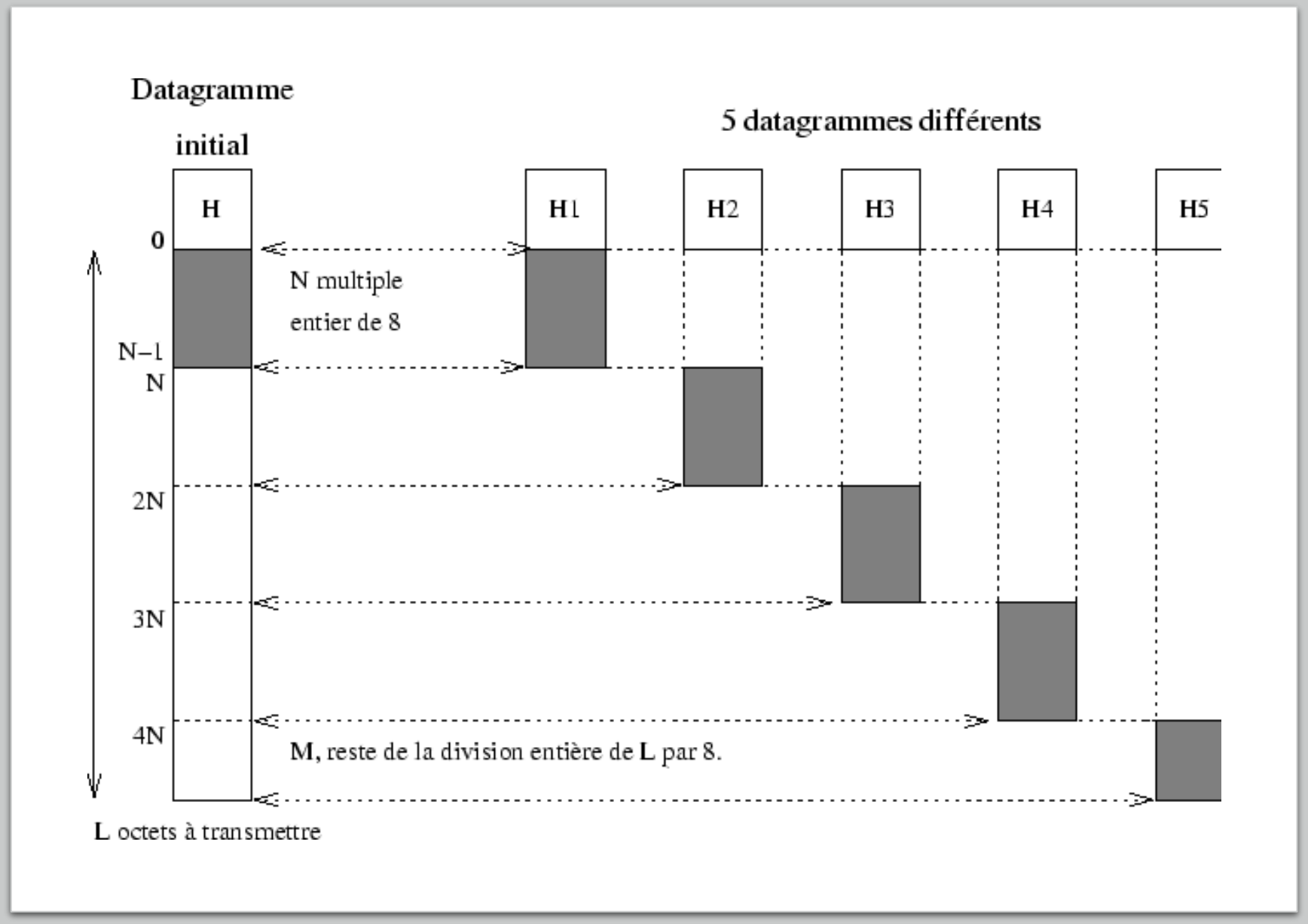
## Mode Non Connecté

# Communication Orientée Packet

- Protocole de communication UDP
  - Transmet des paquets indépendants
  - Sans aucune garantie d'arrivée, ni d'ordonnancement

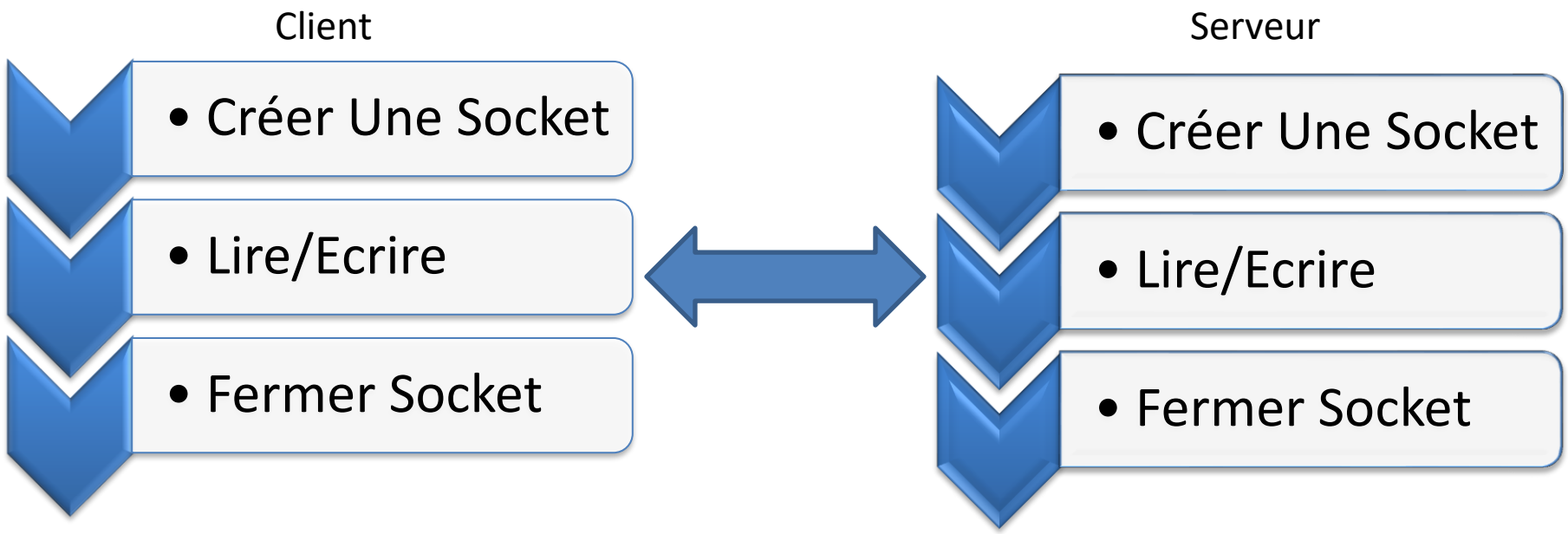


# Fragmentation

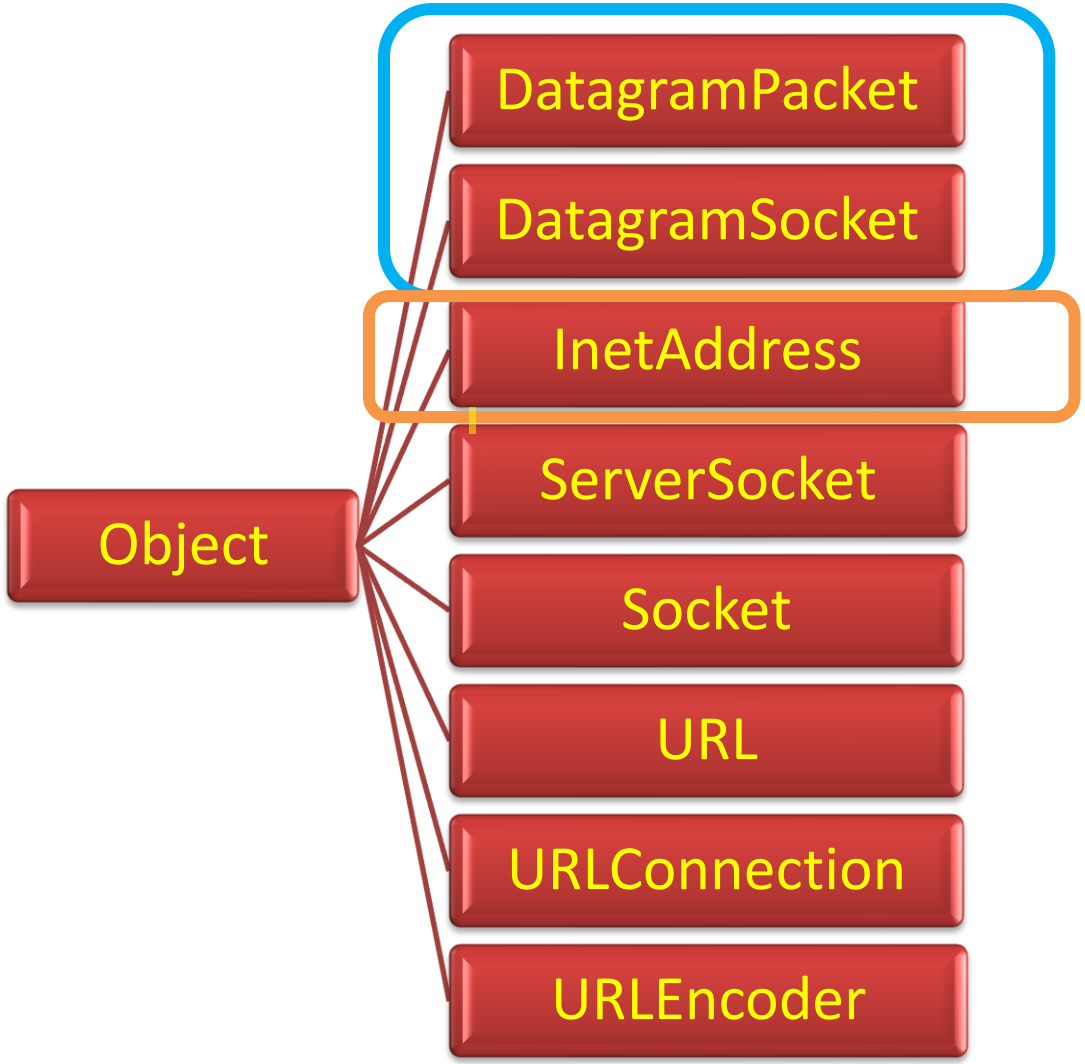


# Communication Orientée Packet

- Protocole de communication UDP



# API java.net


DatagramPacket
DatagramSocket
InetAddress
ServerSocket
Object
Socket
URL
URLConnection
URLEncoder

# Les sockets java

## Classe DatagramPacket

# Classe DatagramPacket

- **Constructeurs** DatagramPacket destiné à l'envoi de données

```
public DatagramPacket(byte[] tampon, int longueur, InetAddress ia, int port)
public DatagramPacket(byte[]  tampon, int offset, int  longueur, InetAddress ia, int port)
```

  - Conseil : utiliser `getBytes()` de la classe String pour transformer des chaînes de caractères en tableaux d'octets
  - Exemple d'utilisation :

```
String message = "Mon Message UDP !";
byte[] tampon = message.getBytes();
InetAddress adresse = InetAddress.getByName("Host Destination");
Int port = 2000; // Port destination
DatagramPacket datagramme = new DatagramPacket(tampon, tampon.length, adresse, port);
```

# Classe DatagramPacket

- **Constructeurs** DatagramPacket destiné à la réception de données

```
public DatagramPacket(byte tampon[], int longueur)
public DatagramPacket(byte[]  tampon, int offset, int  longueur)
```

  - Réception des données dans tampon
    - Longueur maximale : 65 507 octets
    - Java remplit les champs de `DatagramPacket` : adresse IP de la machine distante et le numéro de port concerné
  - Exemple d'utilisation :

```
byte[] tampon = new byte[1024];
DatagramPacket datagramme = new DatagramPacket(tampon, tampon.length);
```

# **Classe** DatagramPacket

- Information
  - `public` **`int getPort()`**
    - Numéro de port de provenance si datagramme reçu
    - Numéro de port de destination si datagramme créé localement
  - `public` **`InetAddress getAddress()`**
    - Adresse du hôte distant si datagramme reçu
    - Adresse du hôte local si datagramme créé localement

# **Classe** DatagramPacket

- Information (suite)
  - public **byte[] getData()**
    - Tableau d'octets du datagramme
    - Caractères ASCII :

```
String s=new String(datapacket.getData());
```

    - Caractères non ASCII :

```
ByteArrayInputStream b=new ByteArrayInputStream(dp.getData());
```

  - public **int getLength()**
    - Taille en octets du datagramme

# Les sockets java

## Classe DatagramSocket

# Classe DatagramSocket

- **Constructeurs**
  - DatagramSocket utilisé par le client
    - public **DatagramSocket()** throws SocketException
    - Ports de destination et de source sont spécifiés dans le datagramme (objet DatagramPacket)
  - DatagramSocket utilisé par le serveur
    - public **DatagramSocket(int port)** throws SocketException
  - Remarque : les ports TCP et leurs équivalents UDP ne sont pas liés i.e. un même numéro de port peut être associé à la fois à une socket TCP et une socket UDP sans provoquer aucun problème de conflit
  - DatagramSocket utilisé par le serveur multi-adresse
    - public **DatagramSocket(int port, InetAdress ia)** throws SocketException

# **Classe** DatagramSocket

- Emission/Réception de datagrammes
  - public void **send(DatagramPacket dp)** throws IOException
  - public void **receive(DatagramPacket dp)** throws IOException
- Information
  - public int **getLocalPort()**
  - Numéro de port (anonyme et assigné par le système) sur lequel la socket courante écoute
- Fermeture
  - public void **close()**
- Option : **SO_TIMEOUT**
  - public synchronized void setSoTimeout(int timeout) throws SocketException



# Les sockets java

## UDP…Reconstitution de l’exemple

# Exemple : Envoi ASCII

- Client – exemple émission

```
1. sc = new DatagramSocket()
2. DatagramPacket out = new DatagramPacket(buf,buf.length, IPDest,PortDest)
3. sc.send(out)
```

```
public void envoiMsgASCII(String host,int port, String msg) {
byte[] tampon = msg.getBytes();
InetAddress adresse;
try {
dataSockClient = new DatagramSocket();
adresse = InetAddress.getByName(host);
DatagramPacket packetOut = new DatagramPacket(tampon, tampon.length, adresse, port);
dataSockClient.send(packetOut);
} catch (IOException e) {
e.printStackTrace();
}
}
```



# Exemple : Envoi ASCII

## Serveur – exemple réception

```
1.ss = new DatagramSocket(port)
2.DatagramPacket in = new DatagramPacket(buf, buf.length)
3.ss.receive(in)
3.analyse(in)
```

```
public void receptASCII() {
byte[] tampon = new byte[tailleMax];
try {
dataSockDest = new DatagramSocket(port);
DatagramPacket packetIn = new DatagramPacket(tampon, tampon.length);
dataSockDest.receive(packetIn);
System.out.println("Un packet vient d'etre recu de : " + packetIn.getSocketAddress());
System.out.println("Le Message recu est : "+new String(packetIn.getData()));
} catch (IOException e) {
e.printStackTrace();
}
}
```

# Exercices

- Reprendre l’exemple
- Réaliser l’envoi d’une petite image encapsulée dans une seul paquet
- Réaliser l’envoi d’une image de taille importante à envoyer par plusieurs paquets