SYR – Réseau Licence 3 Info - MIAGE

Programmation socket

Adlen Ksentini

istic Informatique Électronique

UNIVERSITÉ DE RENNES 1

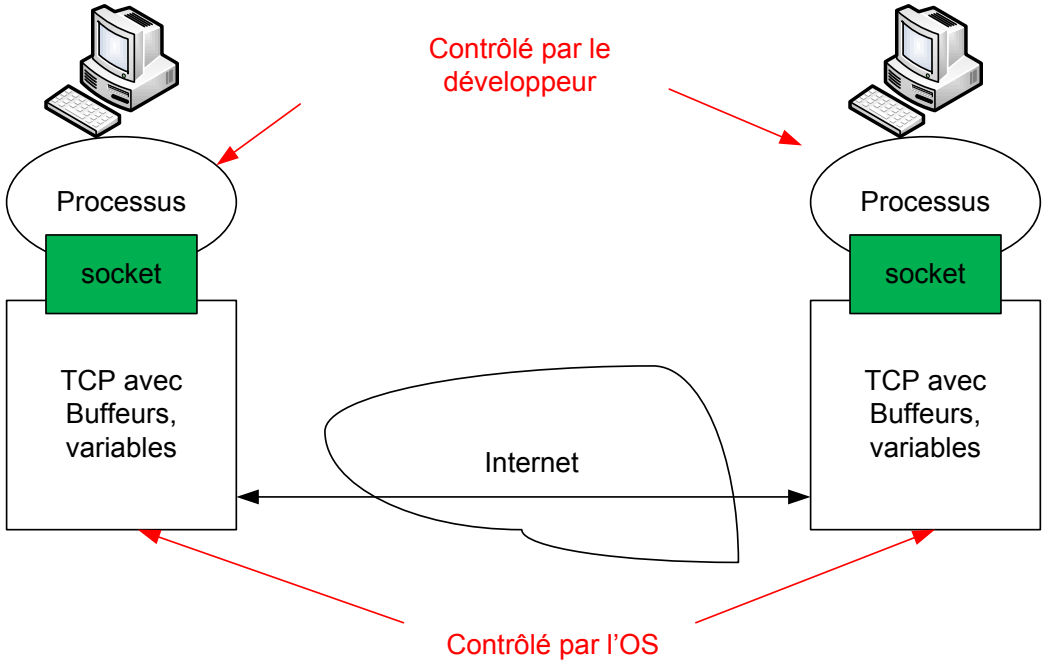
Programmation socket

- But : apprendre à concevoir une application Client/Serveur qui communique par le biais des sockets
- API socket
- Introduit par le système Unix BSD 4.1 en 1981
- Explicitement crée, utilisée et libérée par l'application
- Client/Serveur architecture
- Deux types de service proposés
  - Non-fiable (datagramme)
  - Fiable (flux ou stream)

Socket :
- Une prise entre la couche application et la couche transport
- Une porte permettant aux applications de recevoir/ transmettre sur un réseau

Programmation socket : TCP

- Service TCP : transport fiable d'Octets d'un processus vers un autre



Programmation socket : TCP

Le client doit contacter le serveur

- Le processus du serveur doit être en écoute
- Le serveur crée une socket pour attendre les demandes de connexion des clients

Le client contacte le serveur en :

- Créant une socket de type client TCP
- Spécifiant l' @IP et le numéro du port sur lequel le processus serveur est en écoute
- Quand le client crée la socket : le client TCP établie la connexion avec le serveur

- Quand un serveur est contacté par un client, le serveur TCP crée une nouvelle socket associée avec le client (communique avec le client)
  - Permet au serveur de gérer plusieurs clients à la fois
  - Le port source permet de différencier les connexions avec le client

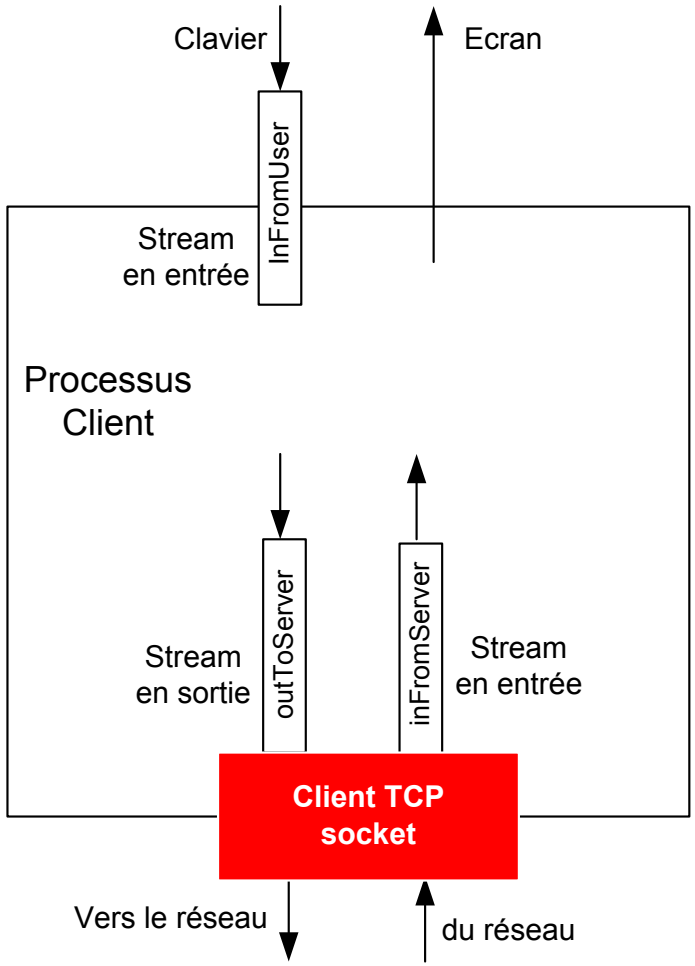
Les flux ou stream

- Un stream est une séquence de caractères qui provient ou qui est à destination d'un processus
- Un input stream est attaché à une entrée (*input*) pour le processus
  - Ex. clavier ou socket
- Un output stream est attaché à une sortie (*output)*
  - Ex. écran ou socket

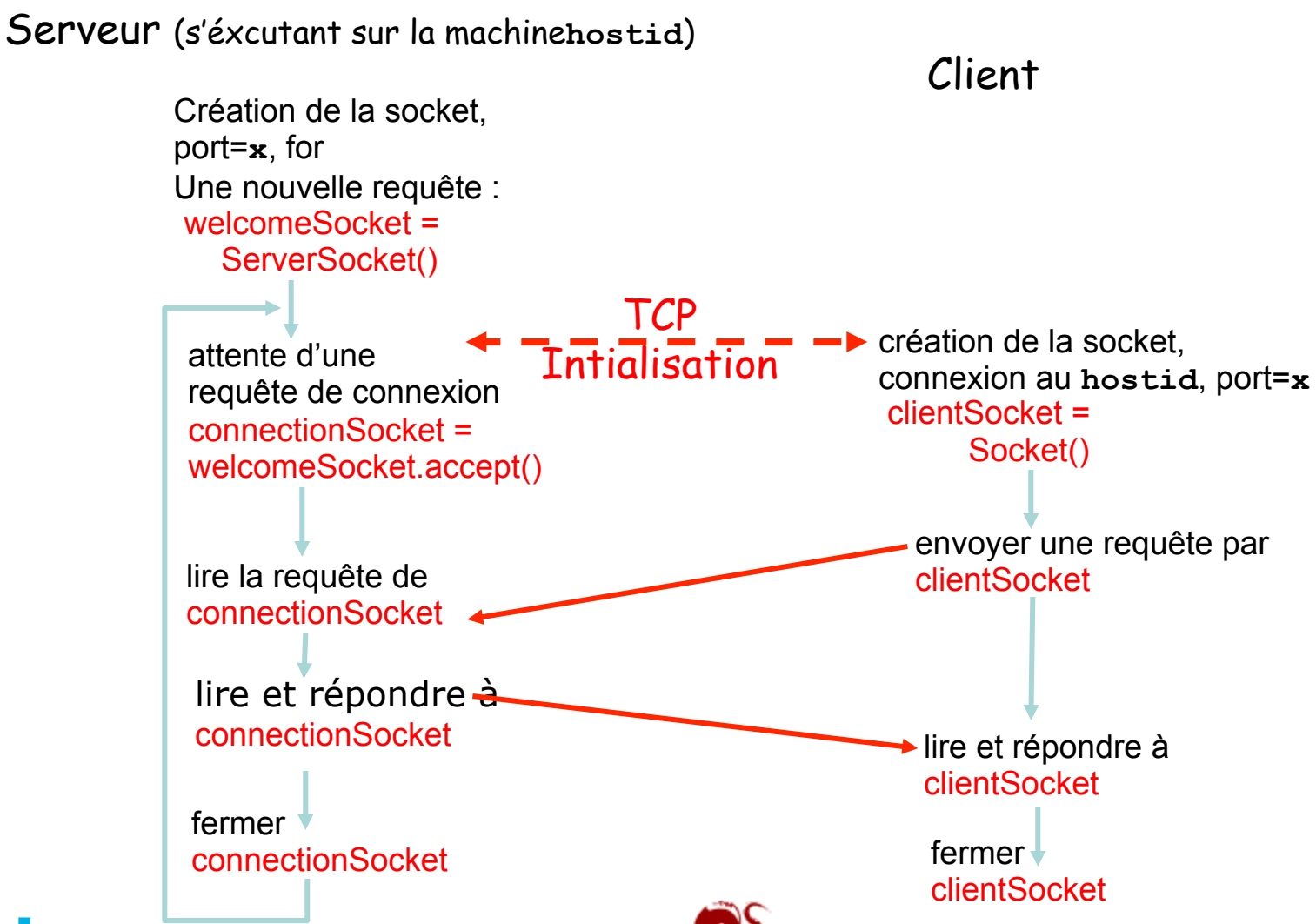
Programmation socket : TCP

Exemple d'une appli. Client/ Serveur

1) Le client lit à partir d'une entrée standard (`inFromUser` stream), envoie au serveur via la socket (`outToServer` stream)
2) Le serveur lit une ligne à partir de la socket
3) Le serveur convertit la ligne en Majuscule, renvoie le résultat au client
4) Le client lit, imprime la ligne modifiée à partir de la socket (`inFromServer` stream)



Interaction Client/Serveur en mode connecté



Exemple : Client Java (TCP)

```
import java.io.*;
import java.net.*;
class TCPClient {

    public static void main(String argv[]) throws Exception
    {
        String sentence;
        String modifiedSentence;

        BufferedReader inFromUser =
          new BufferedReader(new InputStreamReader(System.in));

        Socket clientSocket = new Socket("hostname", 6789);

        DataOutputStream outToServer =
          new DataOutputStream(clientSocket.getOutputStream());
```

créer un input stream

créer Un client socket, connexion au serveur

créer un output stream attacher au socket

Exemple : Client Java (TCP) suite

créer un intput stream attacher au socket

```
BufferedReader inFromServer =
  new BufferedReader(new
  InputStreamReader(clientSocket.getInputStream()));

sentence = inFromUser.readLine();
```

envoie d'une ligne au serveur

```
outToServer.writeBytes(sentence + '\n');
```

lecture d'une ligne du serveur

```
modifiedSentence = inFromServer.readLine();

System.out.println("FROM SERVER: " + modifiedSentence);

clientSocket.close();

    }
}
```

Exemple : Serveur Java (TCP)

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

class TCPServer {

  public static void main(String argv[]) throws Exception
   {
     String clientSentence;
     String capitalizedSentence;

     ServerSocket welcomeSocket = new ServerSocket(6789);

     while(true) {

       Socket connectionSocket = welcomeSocket.accept();

      BufferedReader inFromClient =
        new BufferedReader(new
        InputStreamReader(connectionSocket.getInputStream()));
```

créer La welcom socket port 6789

écouter sur La welcom socket

créer un input stream, l'attacher à la socket

Exemple : Serveur Java (TCP) (suite)

créer un Output stream, l'attacher à la socket

```
DataOutputStream  outToClient =
  new DataOutputStream(connectionSocket.getOutputStream());
```

lire une ligne de la socket

```
clientSentence = inFromClient.readLine();

capitalizedSentence = clientSentence.toUpperCase() + '\n';
```

écrire une ligne à la socket

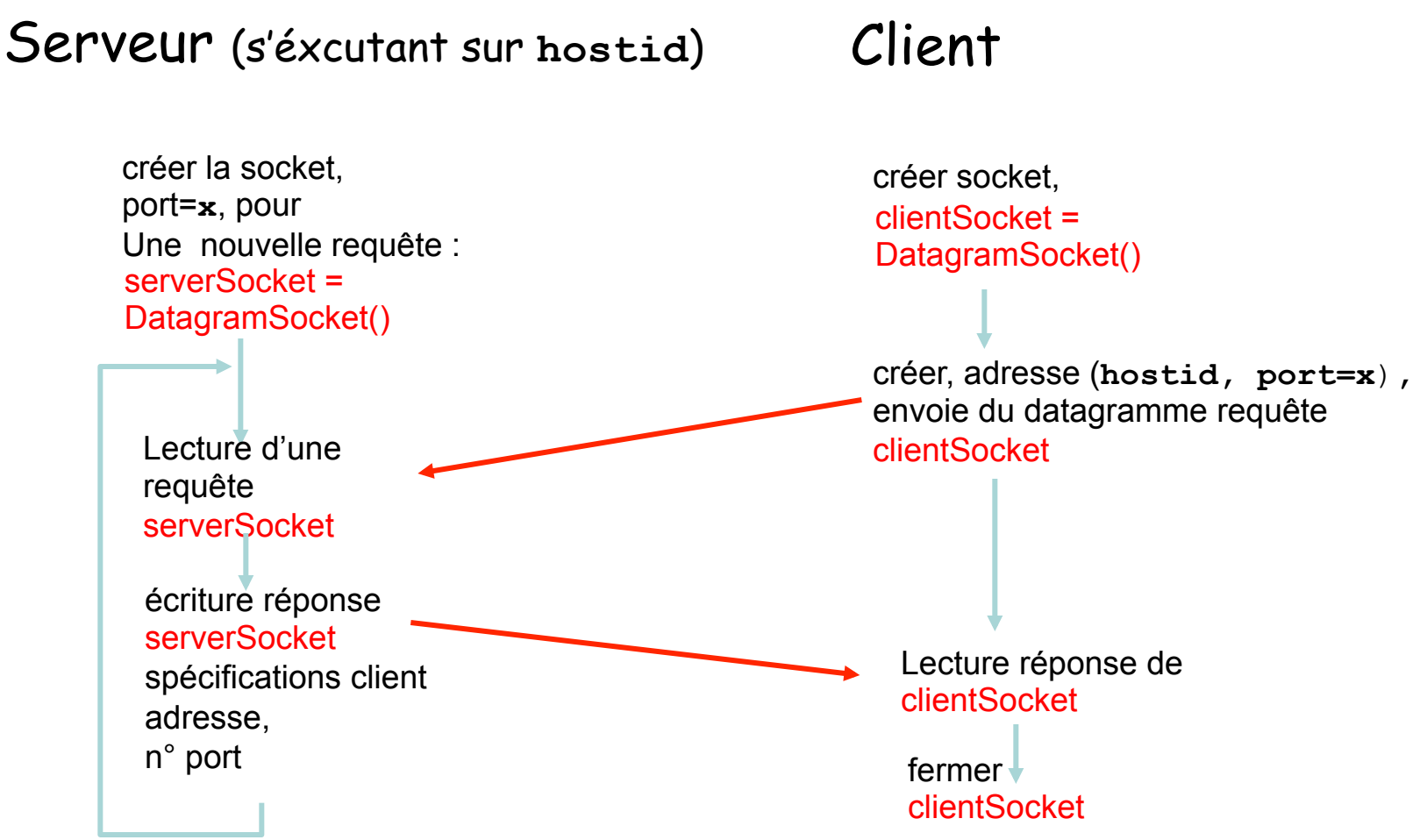
```
outToClient.writeBytes(capitalizedSentence);
        }
    }
}
```

Fin de la boucle while,
Attendre une nouvelle connexion d'un client

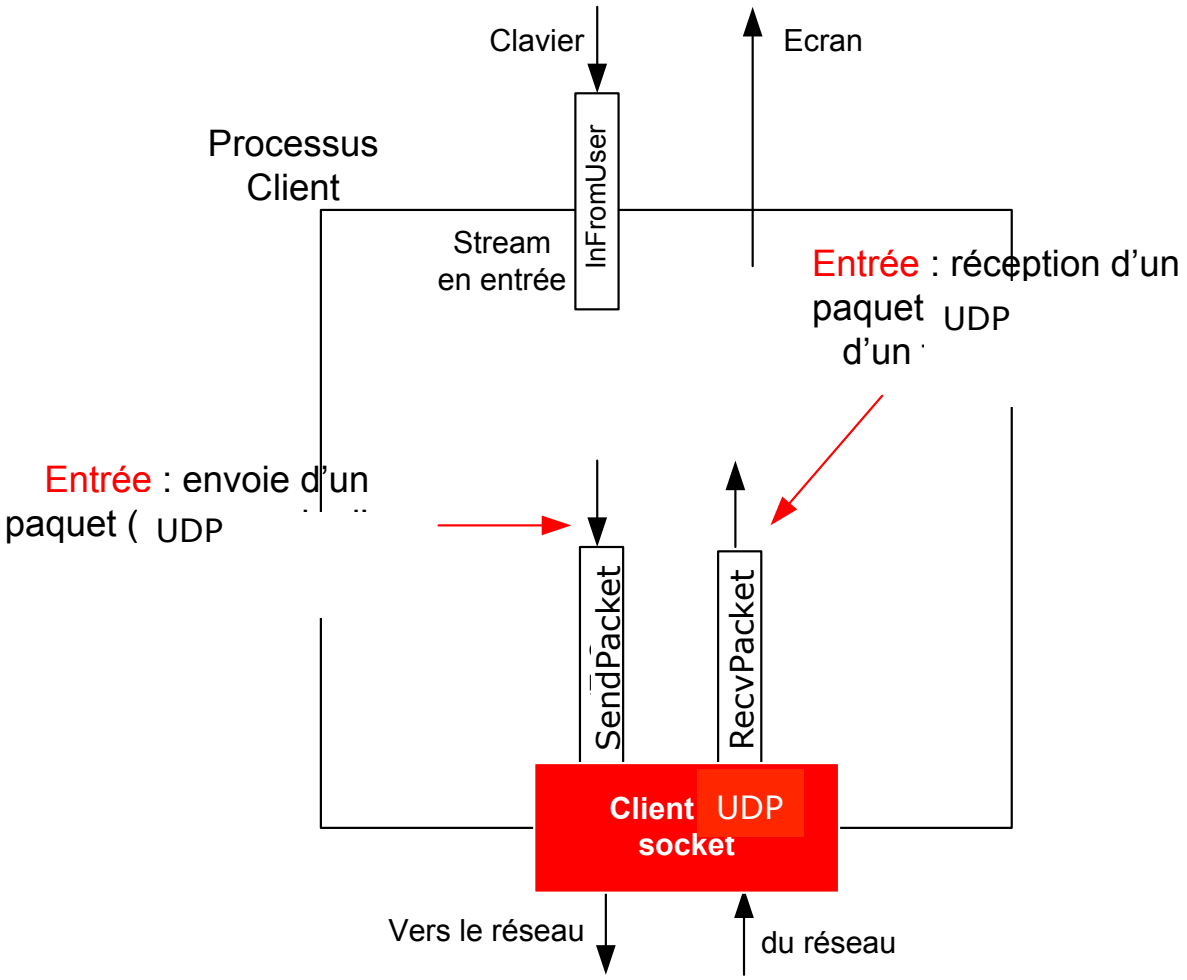
Programmation socket : UDP

- UDP : pas de connexion entre le client et le serveur
  - Pas d'établissement de connexion
  - A chaque émission d'un paquet, l'émetteur spécifie l'@IP et le numéro de port
  - Le serveur doit extraire l'@IP et le port de l'émetteur du paquet reçu
- UDP les données peuvent être reçues dans le désordre

Interaction Client/Serveur en mode non connecté



Exemple : Client Java (UDP)



Exemple : Client Java (UDP)

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

class UDPClient {
   public static void main(String args[]) throws Exception
   {

      BufferedReader inFromUser =
        new BufferedReader(new InputStreamReader(System.in));

      DatagramSocket clientSocket = new DatagramSocket();

      InetAddress IPAddress = InetAddress.getByName("hostname");

      byte[] sendData = new byte[1024];
      byte[] receiveData = new byte[1024];

      String sentence = inFromUser.readLine();

      sendData = sentence.getBytes();
```

créer uninput stream

créer Le client socket

Traduire hostname en adresse IP avec DNS

Exemple : Client Java (UDP) (suite)

Créer le datagramme avec les données Envoyer, **@IP, port**

Envoie du datagramme au serveur

Lecture du datagramme du serveur

```
DatagramPacket sendPacket =
   new DatagramPacket(sendData, sendData.length, IPAddress, 9876);

clientSocket.send(sendPacket);

DatagramPacket receivePacket =
   new DatagramPacket(receiveData, receiveData.length);

clientSocket.receive(receivePacket);

String modifiedSentence =
   new String(receivePacket.getData());

System.out.println("FROM SERVER:" + modifiedSentence);
clientSocket.close();
}

}
```

Exemple : serveur Java (UDP)

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

class UDPServer {
  public static void main(String args[]) throws Exception
   {

    DatagramSocket serverSocket = new DatagramSocket(9876);

    byte[ ] receiveData = new byte[1024];
    byte[ ] sendData  = new byte[1024];

    while(true)
     {

      DatagramPacket receivePacket =
        new DatagramPacket(receiveData, receiveData.length);

      serverSocket.receive(receivePacket);
```

créer Un datagramme socket écouter le port 9876

créer un éspace pour les datagrammes reçus

Réception d'un datagramme

Exemple : serveur Java (UDP) (suite)

```
String sentence = new String(receivePacket.getData());

InetAddress IPAddress = receivePacket.getAddress();

int port = receivePacket.getPort();

String capitalizedSentence = sentence.toUpperCase();

sendData = capitalizedSentence.getBytes();

DatagramPacket sendPacket =
   new DatagramPacket(sendData, sendData.length, IPAddress,
                      port);

serverSocket.send(sendPacket);
    }
  }
}
```

Récupérer @ IP , port # De l'émetteur

créer le datagramme à envoyer au client

écriture du datagramme à la socket

Fin de la boucle while, attente d'un nouveau datagramme