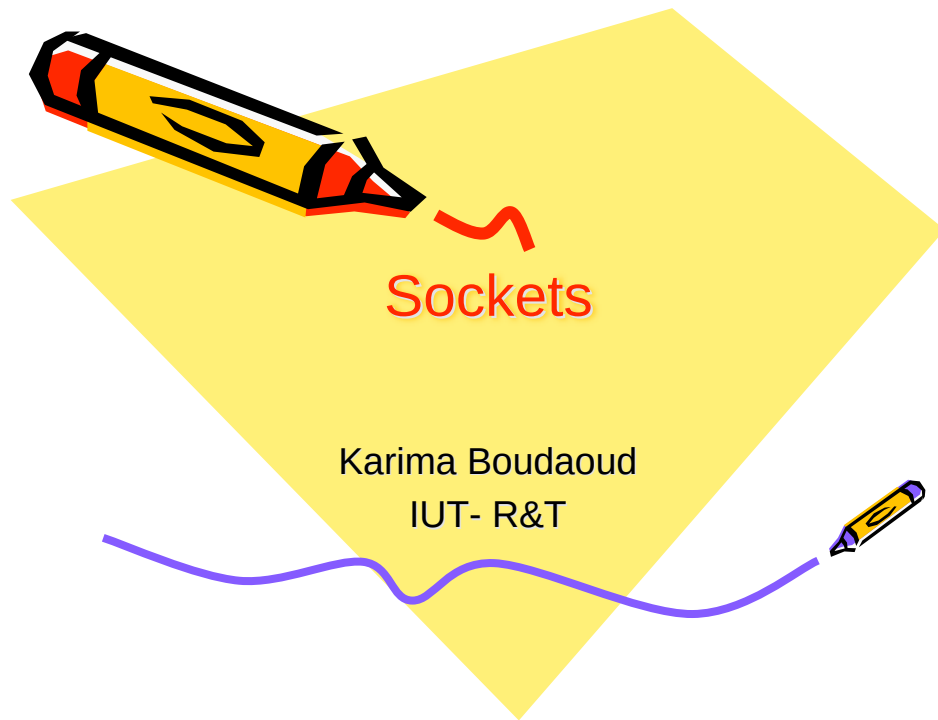




Rappels sur les sockets

○ Qu'est ce qu'un socket?

- Point d'entrée entre 2 appli. du réseau
- Permet l'échange de donnée entre elles à l'aide des mécanismes d'E/S (**java.io**)

○ Différents types de sockets

• Stream Sockets (TCP)

- ✓ établir une communication en mode connecté
- ✓ si connexion interrompue : applications informées

• Datagram Sockets (UDP)

- ✓ établir une communication en mode non connecté
- ✓ données envoyées sous forme de paquets indépendants de toute connexion. Plus rapide, moins fiable que TCP

Karima Boudaoud

IUT GTR-Sophia Antipolis

2

Des exemples d'applications

○ Applications TCP les plus connues

- ✓ FTP, SMTP, TELNET, etc...

○ Applications UDP les plus connues

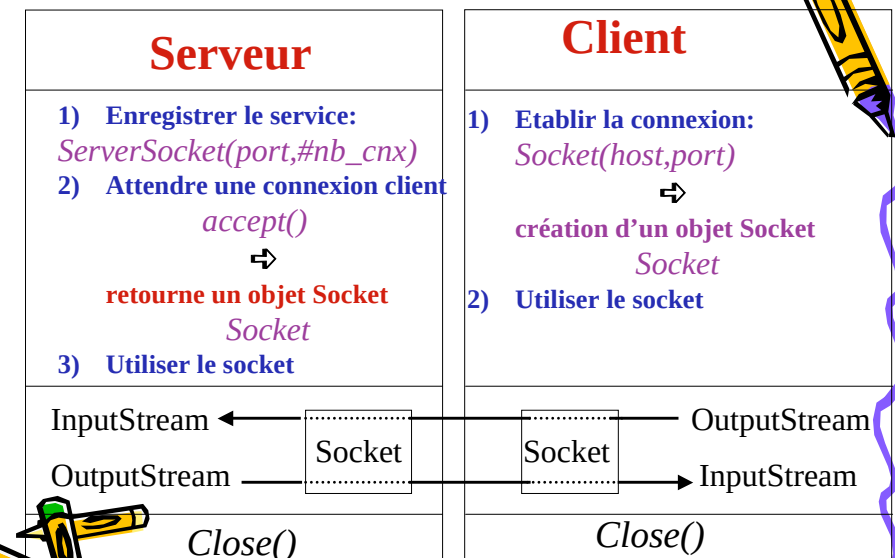
- ✓ Simple Network Management Protocol (SNMP)
- ✓ Trivial File Transfer Protocol (TFTP): version datagramme de FTPD (pour le boot par le réseau)

Karima Boudaoud

IUT GTR-Sophia Antipolis

3

Le modèle client-serveur Java



Karima Boudaoud

IUT GTR-Sophia Antipolis

4

Principe de fonctionnement (1)

○ Serveur: enregistrer le service

- le serveur enregistre son service sous un numéro de port, indiquant le nombre de clients qu'il accepte de faire buffériser à un instant T (`new ServerSocket(...)`)

○ Serveur : attente de connexion

- il se met en attente d'une connexion (méthode `accept()` de son instance de `ServerSocket`)



Karima Boudaoud

IUT GTR-Sophia Antipolis

5

Principe de fonctionnement (2)

○ Client : établir la connexion

- le client peut alors établir une connexion en demandant la création d'un socket (`new Socket()`) à destination du serveur pour le port sur lequel le service a été enregistré.

○ Serveur

- le serveur sort de son `accept()` et récupère un `Socket` de communication avec le client

○ Les deux : utilisation du socket

- le client et le serveur peuvent alors utiliser des `InputStream` et `OutputStream` pour échanger les données



Karima Boudaoud

IUT GTR-Sophia Antipolis

6

Un serveur TCP/IP (1)

- il utilise la classe `java.net.ServerSocket` pour accepter des connexions de clients
- quand un client se connecte à un port sur lequel un `ServerSocket` écoute, `ServerSocket` crée une nouvelle instance de la classe `Socket` pour supporter les communications côté serveur :

```
int port = ...;  
ServerSocket server = new ServerSocket(port);  
Socket connection = server.accept();
```



Karima Boudaoud

IUT GTR-Sophia Antipolis

7

Un serveur TCP/IP (2)

- les constructeurs et la plupart des méthodes peuvent générer une `IOException`
- la méthode `accept()` est dite bloquante ce qui implique un type de programmation particulier : boucle infinie qui se termine seulement si une erreur grave se produit



Karima Boudaoud

IUT GTR-Sophia Antipolis

8

java.net.ServerSocket

```
final int PORT = ...;

try {
    ServerSocket serveur = new ServerSocket(PORT,5);
    while (true) {
        Socket socket = serveur.accept();
    }
} catch (IOException e){
    ....
}
```



Karima Boudaoud

IUT GTR-Sophia Antipolis

9

Un client TCP/IP

- le client se connecte au serveur en créant une instance de la classe **java.net.Socket** : connexion synchrone
`String host = ...;`
`int port = ...;`
`Socket connection = new Socket (host,port);`
- le socket permet de supporter les communications côté client
- la méthode `close()` ferme (détruit) le socket
- les constructeurs et la plupart des méthodes peuvent générer une **IOException**
- le serveur doit être démarré avant le client. Dans le cas contraire, si le client se connecte à un serveur inexistant, une exception sera levée après un *time-out*



Karima Boudaoud

IUT GTR-Sophia Antipolis

10

java.net.Socket

```
final String HOST = "... ";
final int PORT = ...;

try {
    Socket socket = new Socket (HOST,PORT);
}
finally {
    try {socket.close();} catch (IOException e){}
}
```



Karima Boudaoud

IUT GTR-Sophia Antipolis

11

Flux de données (1)

- une fois la connexion réalisée, il faut obtenir les **streams** d'E/S (**java.io**) auprès de l'instance de la classe **Socket** en cours
- Flux entrant**
 - ✓ obtention d'un **stream** simple: définit les op. de base
`InputStream in = socket.getInputStream();`
 - ✓ création d'un **stream** convertissant les bytes reçus en char
`InputStreamReader reader = new InputStreamReader(in);`
 - ✓ création d'un **stream** de lecture avec tampon: pour lire ligne par ligne dans un **stream** de caractères
`BufferedReader istream = new BufferedReader(reader);`
 - ✓ lecture d'une chaîne de caractères
`String line = istream.readLine();`



Karima Boudaoud

IUT GTR-Sophia Antipolis

12

Flux de données (2)

● Flux sortant

- ✓ obtention du flot de données sortantes : bytes
`OutputStream out = socket.getOutputStream();`
- ✓ création d'un **stream** convertissant les bytes en chaînes de caractères
`PrintWriter ostream = new PrintWriter(out);`
- ✓ envoi d'une ligne de caractères
`ostream.println(str);`
- ✓ envoi effectif sur le réseau des bytes (important)
`ostream.flush();`



Karima Boudaoud

IUT GTR-Sophia Antipolis

13

Flux de données (3)

```
try {
    Socket socket = new Socket(HOST,PORT);
    //Lecture du flux d'entrée en provenance du serveur
    InputStreamReader reader = new
        InputStreamReader(socket.getInputStream());
    BufferedReader istream = new BufferedReader(reader);
    String line = istream.readLine();
    //Echo la ligne lue vers le serveur
    PrintWriter ostream = new PrintWriter(socket.getOutputStream());
    ostream.println(line);
    ostream.flush();
} catch (IOException e) {...}
finally {try{socket.close();}catch (IOException e){}}
```



Karima Boudaoud

IUT GTR-Sophia Antipolis

14

Un serveur TCP/IP multiclents (1)

- le serveur précédent accepte plusieurs connexions simultanées, mais ne traite qu'un client à la fois, les autres sont mis en attente
- pour y remédier, utiliser les **threads** java (**java.lang.Thread**)

```
try{
    ServerSocket serveur = new ServerSocket(PORT);
    while (true) {
        //accepter une connexion
        Socket socket = serveur.accept();
        // créer un thread : pour échanger les données avec le client
        Connexion c = new Connexion(socket);
        Thread processus_connexion = new Thread(c);
        processus_connexion.start();
    }
} catch (IOException e) {...}
```



Karima Boudaoud

IUT GTR-Sophia Antipolis

15

Un serveur TCP/IP multiclents (2)

```
class Connexion implements Runnable
{
    .....
    public Connexion (Socket s) {
        this.s = s;
        try{
            in=new BufferedReader(
                new
                InputStreamReader(s.getInputStream()));

            out = new PrintWriter(s.getOutputStream());
        } catch (IOException e) {...}
    }
}
```



Karima Boudaoud

IUT GTR-Sophia Antipolis

16

Un serveur TCP/IP multiclients (3)

```
public void run() {
    try{
        while (true) {
            String ligne = in.readLine();
            if (ligne == null) break; //fin de connexion côté client
            output.println(ligne); out.flush();
        }
    } catch (IOException e) {...}
    finally {try{s.close();}catch (IOException e){}}
```



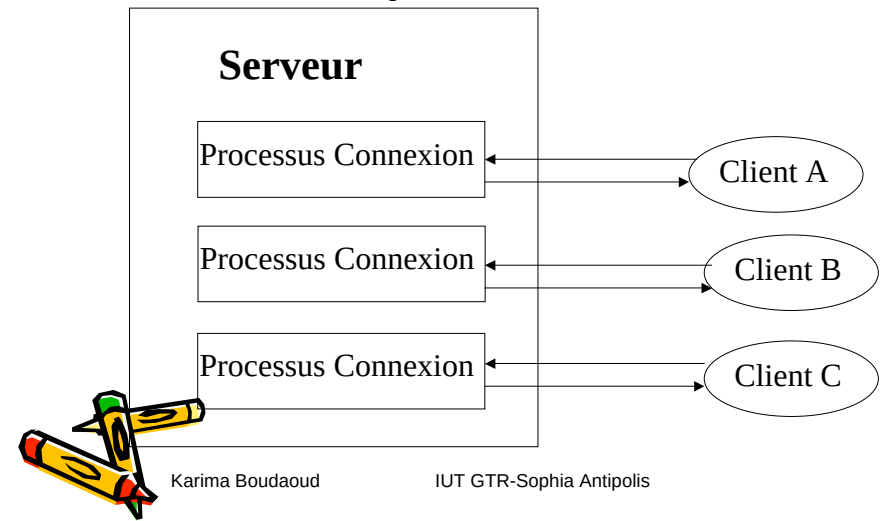
Karima Boudaoud

IUT GTR-Sophia Antipolis

17

Un serveur TCP/IP multiclients (4)

- le serveur utilise une classe **Connexion** implémentant l'interface **Runnable** (*thread*) pour gérer les échanges de données en tâche de fond. C'est ce *thread* qui réalise le service demandé



Karima Boudaoud

IUT GTR-Sophia Antipolis

18

Utilisation des sockets Datagrams (1)

○ Quelles classes utiliser ?

- il faut utiliser les classes **DatagramPacket** et **DatagramSocket**
- Ces objets sont initialisés différemment selon qu'ils sont utilisés pour *envoyer* ou *recevoir* des paquets



Karima Boudaoud

IUT GTR-Sophia Antipolis

19

Utilisation des sockets Datagrams (2)

○ Envoi d'un Datagram

- créer un **DatagramPacket** en spécifiant :
 - les données à envoyer
 - leur longueur
 - la machine réceptrice et le port
- utiliser la méthode **send(DatagramPacket)** de **DatagramSocket**
 - pas d'arguments pour le constructeur car toutes les informations se trouvent dans le paquet envoyé



Karima Boudaoud

IUT GTR-Sophia Antipolis

20

Envoi d'un Datagram

//Machine destinataire

```
InetAddress address = InetAddress.getByName("rainbow.essi.fr");
static final int PORT = 4562;
//Création du message à envoyer
String s = new String("Message à envoyer");
int longueur = s.length();
byte[] message = new byte[longueur];
s.getBytes(0, longueur, message, 0);
//Initialisation du paquet avec toutes les informations
DatagramPacket paquet = new DatagramPacket(message, longueur,
address, PORT);
//Création du socket et envoi du paquet
DatagramSocket socket = new DatagramSocket();
socket.send(paquet);...
```



Karima Boudaoud

IUT GTR-Sophia Antipolis

21

Réception d'un Datagram (1)

○ Réception d'un Datagram

1. créer un `DatagramSocket` qui écoute sur le port de la machine du destinataire
2. créer un `DatagramPacket` pour recevoir les paquets envoyés par le serveur
 - dimensionner le buffer assez grand
3. utiliser la méthode `receive()` de `DatagramPacket`
 - cette méthode est bloquante



Karima Boudaoud

IUT GTR-Sophia Antipolis

22

Réception d'un Datagram (2)

```
//Définir un buffer de réception
byte[] buffer = new byte[1024];
//On associe un paquet à un buffer vide pour la réception
DatagramPacket paquet = new
    DatagramPacket(buffer, buffer.length());
//On crée un socket pour écouter sur le port
DatagramSocket socket = new DatagramSocket(PORT);
while (true) {
    //attente de réception
    socket.receive(paquet);
    //affichage du paquet reçu
    String s = new String(buffer, 0, 0, paquet.getLength());
    System.out.println("Paquet reçu : + s);
}
```



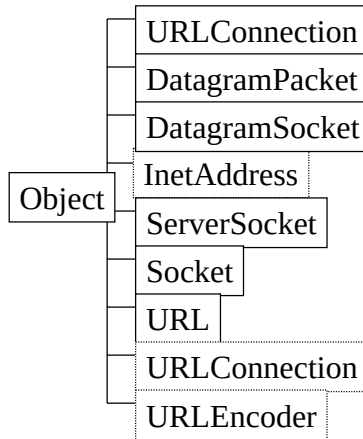
Karima Boudaoud

IUT GTR-Sophia Antipolis

23



java.net.*



Karima Boudaoud

IUT GTR-Sophia Antipolis

25

java.net.Socket

- cette classe implémente une socket TCP coté client

```
String serveur = "www.inria.fr";
int port = 80;
```

```
Socket socket = new Socket(serveur, port);
```

```
PrintStream ps = new PrintStream(socket.getOutputStream());
ps.println("GET /index.html");
```

```
DataInputStream dis = new
    DataInputStream(s.getInputStream());
```

```
String line;
while((line = dis.readLine()) != null)
    System.out.println(line);
```

Karima Boudaoud

IUT GTR-Sophia Antipolis

26

java.net.ServerSocket

- cette classe implémente une socket TCP coté serveur.

```
int port_d_ecoute = 1234;
ServerSocket serveur = new ServerSocket(port_d_ecoute);
```

```
while(true)
{
    Socket socket_de_travail = serveur.accept();
    new ClasseQuiFaitLeTraitement(socket_travail);
}
```

Karima Boudaoud

IUT GTR-Sophia Antipolis

27

java.net.DatagramSocket (1)

- cette classe implémente une socket UDP

// Client

```
Byte[] data = "un message".getBytes();
```

```
InetAddress addr = InetAddress.getByName("falconet.inria.fr");
```

```
DatagramPacket packet = new DatagramPacket(data, data.length, addr,
    1234);
```

```
DatagramSocket ds = new DatagramSocket();
```

```
ds.send(packet);
ds.close();
```

Karima Boudaoud

IUT GTR-Sophia Antipolis

28

java.net.DatagramSocket (2)

// Serveur

```
DatagramSocket ds = new DatagramSocket(1234);

while(true)
{
    DatagramPacket packet = new DatagramPacket(new
    byte[1024], 1024);
    ds.receive(packet);
    System.out.println("Message: " + packet.getData());
}
```



Karima Boudaoud

IUT GTR-Sophia Antipolis

29

java.net.MulticastSocket (1)

- cette classe implémente une socket multicast (UDP)

// Client

```
Byte[] data = "un message".getBytes();

InetAddress addr = InetAddress.getByName("falconet.inria.fr");

DatagramPacket packet = new DatagramPacket(data,
    data.length, addr, 1234);

MulticastSocket s = new MulticastSocket();
s.send(packet,(byte)1);
s.close();
```



Karima Boudaoud

IUT GTR-Sophia Antipolis

30

java.net.MulticastSocket (2)

// Serveur

```
MulticastSocket s = new MulticastSocket(1234);
System.out.println("I listen on port " + s.getLocalPort());

s.joinGroup(InetAddress.getByName("falconet.inria.fr"));

DatagramPacket packet = new DatagramPacket(new byte[1024],
    1024);
s.receive(packet);
System.out.println("from: " + packet.getAddress());
System.out.println("Message: " + packet.getData());

s.leaveGroup(InetAddress.getByName("falconet.inria.fr"));
s.close();
```



Karima Boudaoud

IUT GTR-Sophia Antipolis

31

java.net.URL

```
URL url = new
    URL("http://falconet.inria.fr/index.html");
```

```
DataInputStream dis = new
    DataInputStream(url.openStream());
```

```
String line;
while ((line = dis.readLine()) != null)
    System.out.println(line);
```



Karima Boudaoud

IUT GTR-Sophia Antipolis

32