

Socket UDP

- 1 DatagramPacket
- 2 DatagramSocket
- 3 Broadcast et Multicast

Connexion sûre ou non ?

- Toutes les applications ne nécessitent pas une connexion sûre fournie par TCP
- TCP établit un canal de communication
- Pas le surcoût lié à TCP (ex : diffusion multimédia ...)

UDP

- **UDP** fournit ce mode de communication, où des paquets de données, appelés des **datagrammes**, sont envoyés directement sans établissement au préalable d'un canal de communication
- Le contenu d'un datagramme permet son **acheminement** à travers le réseau (contient l'adresse IP, le port ...)
- Les datagrammes sont des messages envoyés sur le réseau dont la date d'arrivée, le contenu, l'arrivée ne sont **pas garantis**

En java : package **java.net**

- **DatagramPacket**
- **DatagramSocket**

Un programme envoie des DatagramPacket à travers un
DatagramSocket

DatagramPacket

Cette classe représente un datagramme. Les messages sont acheminés grâce aux informations contenues dans ce paquet.

Datagramme à envoyer : constructeur

```
DatagramPacket(byte[] buf, int length,  
               InetAddress address, int port)
```

Construit un datagramme, avec les données buf, de longueur length, à envoyer à l'adresse et au port spécifiés.

Datagramme pour recevoir : constructeur

```
DatagramPacket(byte[] buf, int length)
```

Construit un datagramme pour recevoir des paquets de longueur `length`, qui seront stockés dans `buf`.

- `getAddress()` et `getPort()` retournent respectivement l'adresse et le port du paquet.
- `getData()` retourne les données du paquets

DatagramSocket

Cette classe permet de créer un socket pour envoyer et recevoir des datagrammes

Constructeur

```
DatagramSocket(int numPort);
```

Crée un socket pour datagrammes, et le lie au port spécifié

Réception

```
void receive(DatagramPacket p)
```

Reçoit un paquet et l'enregistre dans le datagramme p. Cette méthode est bloquante. Il est possible de fixer une durée maximale d'attente avec la méthode `setSoTimeout`

Émission

```
void send(DatagramPacket p)
```

Envoie le datagramme p par le socket. Les informations dans le paquet (sa longueur, son ip, le numéro de port) permettent son acheminement

Serveur

```
import java.io.*;import java.net.*;import java.util.*;
public class Serveur {

    public static void main(String [] args) throws IOException {
        DatagramSocket socket = new DatagramSocket(4445);

        byte[] buf = new byte[256];

        // receive request
        DatagramPacket packet = new DatagramPacket(buf, buf.length);
        socket.receive(packet);

        // response
        String dString = new Date().toString();
        buf = dString.getBytes();

        // send the response to the client at "address" and "port"
        InetAddress address = packet.getAddress();
        int port = packet.getPort();
        packet = new DatagramPacket(buf, buf.length, address, port);
        socket.send(packet);
        socket.close();
    }
}
```

Client

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

public class Client {
    public static void main(String[] args) throws IOException {
        // get a datagram socket
        DatagramSocket socket = new DatagramSocket();

        // send request
        byte[] buf = new byte[256];
        InetAddress address = InetAddress.getByName("127.0.0.1");
        DatagramPacket packet = new DatagramPacket(buf, buf.length, address, 4445);
        socket.send(packet);

        // get response
        packet = new DatagramPacket(buf, buf.length);
        socket.receive(packet);

        // display response
        String received = new String(packet.getData(), 0, packet.getLength());
        System.out.println("Date:␣" + received);

        socket.close();
    }
}
```

Broadcast

- Le datagramme est envoyer sur l'adresse de broadcast
- Il est nécessaire d'activer l'option pour autoriser le broadcast sur le socket

```
socket.setBroadcast(true)
```

Multicast

- Il est possible d'envoyer des datagrammes à seulement un groupe de client : multicast
- Les clients doivent rejoindre le groupe, pour cela ils utilisent la classe MulticastSocket

Multicast

```
MulticastSocket socket = new MulticastSocket(4446);  
InetAddress group = InetAddress.getByName("203.0.113.0");  
socket.joinGroup(group);
```

Conclusion

- Permet d'utiliser le protocole UDP
- Communication sans garantie d'arrivée, mais sans le surcoût de TCP
- Deux classes : DatagramPacket et DatagramSocket

- <https://docs.oracle.com/javase/tutorial/networking/datagrams/index.html>