# Sockets UDP

## TP Réseau - Java

**Adrien COMTE 11206512**

**04/05/2015**

### Table des matières

# 1. Introduction

Le projet se divise en 3 packages :
Le package udp qui contient la partie commune au client et serveur.
Le package udp.client qui contient la partie scénaristique d'un client standard.
Le package udp.serveur qui contient la définition plus spécifique du serveur et des accès concurrentiels au serveur.
Le projet est divisé ainsi pour que lors de la distribution des packages, la personne puisse choisir si elle veut télécharger uniquement le client du programme, le serveur, ou l'intégralité.

# 2. Scénario choisi et fonctionnalités développées

## Scénario

Un client se connecte sur un serveur distant en utilisant le port par défaut connu de ce serveur (8080).
Le serveur accepte la demande de connexion du client, et lui accorde un port spécifique de connexion. Ce port doit se situer dans une plage définie à l'avance (10000-11000) pour limiter le nombre de connexion simultanée au serveur et pour s'assurer des paramètres de pare-feu adaptés.
Une fois la connexion établie sur le nouveau port ouvert par le serveur, le client peut envoyer au serveur toutes les chaines de caractères qu'il souhaite jusqu'à une limite de 512 caractères, et ce, autant de fois que le client le souhaite.
Le serveur répond systématiquement à ces chaines en les renvoyant à l'identique.
Lorsque le client a envoyé au serveur toutes les chaines de caractères voulu, il peut terminer la connexion en envoyant la chaine "stop" au serveur.
Le serveur ferme alors le port d'écoute.
Durant tous les échanges entre un client et le serveur, le serveur continue d'écouter son port standard, et ouvre autant de nouveaux ports que de nouvelles demandes de connexion.

## Fonctionnalités

- Scanner de ports libres
    - Cette méthode permet de trouver les ports libres sur une plage choisie. Elle parcourt tous les ports de la plage choisie, tente d'ouvrir un nouveau DatagramSocket sur chaque port. Si le DatagramSocket s'est ouvert avec succès, le numéro de port est conservé dans un tableau et le port est refermé. Si une erreur est renvoyée, le port est donc déjà occupé.
- Accès concurrentiels au serveur
    - Le serveur est capable d'écouter en permanence son port standard pour initialiser des nouvelles connexions aux nouveaux clients, tout en écoutant sur chacun de ses ports déjà ouverts. La réception peut se faire sur n'importe quel port écouté dans n'importe quel ordre, sans interférence entre les différents clients. Un client ne recevra pas ce qui ne lui est pas destiné si il contacte le serveur en même temps qu'un autre client.
- Initiation de nouvelles connexions sécurisée
    - Le serveur ouvre un nouveau port au client et lui répond seulement si celui-ci envoie la chaine de caractère "hello serveur RX302" lors de sa demande de connexion. Si une autre chaine est envoyée, le serveur ne répond pas, et une erreur s'affiche sur le serveur pour signaler la tentative de connexion avortée.
- Fin de connexion
    - Lorsque le serveur reçoit la chaine de caractères "stop" sur un de ses ports d'écoute, ce port est immédiatement fermé et libéré pour une autre connexion.

## 3. Package udp

Ce package ne contient qu'une seule classe, qui est commune au client et au serveur et qui contient l'ensemble des attributs et méthodes de gestion de la carte réseau. Les clients et serveurs héritent de cette unique classe.

### ObjetConnecte.java

Cette classe est abstraite. C'est la base du client et du serveur.

#### Attributs :

- public static final int MAX = 512;
  - Cet attribut correspond à la taille maximale allouée au buffer lors de la réception ou de l'envoie de donnée par la carte réseau.
- public static final int PORT_HOST_STANDARD = 8080;
  - Cet attribut correspond au port standard du serveur à contacter, ou du port à ouvrir par le serveur pour réceptionner correctement les requêtes d'initialisation de connexion.
- protected DatagramSocket ds;
  - Cet attribut correspond au DatagramSocket pour les envois et les réceptions de paquets. Il est initialisé sur le port 8080 pour les serveurs, sur un port aléatoire pout les clients, et sur un port compris dans une plage spécifique pour les réponses au client.

#### Méthodes

- public DatagramPacket receptionner (byte[]);
  - Cette méthode permet au client ou au serveur d'attendre une communication sur son port local. Elle renvoie le contenu de la communication dans le buffer passé en paramètre.
- public void envoyer(String, InetAddress, int);
  - Cette méthode envoie la chaine de caractère à l'adresse passée en paramètre, sur le port passé en paramètre, à partir du port local.
- public ArrayList scanPortLibre(int, int);
  - Cette méthode renvoie un tableau des ports libres sur la plage choisie. Elle permet au serveur d'ouvrir un port libre pour répondre au client, dans une plage acceptable pour un pare-feu par exemple.