**Réseau FA - TP UDP - Compte Rendu**

**I - Sockets UDP en Java**

**Le protocole UDP**

Le protocole **UDP** (User Datagram Protocol) est un protocole s'exécutant en amont du protocole Internet (IP) et développé pour les applications qui ne nécessitent pas de contrôle de flux au niveau de la couche transport.

L’UDP fournit un **adressage** sous la forme de **ports**.

Les messages, appelés **datagrammes**, sont envoyés à d'autres **hôtes** sur un réseau IP sans qu'il soit nécessaire d'établir au préalable des canaux de transmission spéciaux. Il suffit d'ouvrir un **socket UDP** pour communiquer. Il écoute les messages entrants et envoie les messages sortants sur demande.

**Objectifs du TP**

L’objectif de ce TP est d’implémenter un scanner de ports UDP, tel que le scanner permettra, pour une plage de ports, de retourner ouvert/fermé pour chacun de ces ports.

Ensuite, nous développerons un modèle d’une communication client/serveur UDP, en local et à distance.

Pour cela, nous avons :

* La classe Serveur : Le serveur reçoit des requêtes et instancie, via une communication, une réponse.
* La classe Client : Le client est celui qui envoie une première requête au serveur, puis reçoit une réponse du serveur.
* La classe Communication : Cette classe permet au serveur de recevoir des requêtes de différents clients, et de leur répondre individuellement.

## 

## 

## **Les Sockets UDP en Java**

## **La classe DatagramSocket**

Cette classe permet de créer des **sockets UDP** qui envoient et de recoivent des **datagrammes UDP**.

Avant toute communication en mode UDP il est nécessaire de créer un **socket** aussi bien du **côté client** (classe Client) que du **côté serveur** (classe Serveur). Pour cela Java propose le constructeur :

**public DatagramSocket (int port, InetAddress addr) throws SocketException**

Qui crée un objet de type **socket**, l'attache au **port UDP** passé en paramètre et à une **adresse spécifique** de la machine.

Cependant si **aucun adressage** n’est connu, l’argument InetAddress peut être enlevé.

**public DatagramSocket (int port) throws SocketException**

De même, il n’est pas obligatoire de passer un argument int port dans le cas où le **port d’attachement est inconnu.**

**public DatagramSocket () throws SocketException**

Ensuite, une fois le **socket** créé, il est possible d'envoyer et de recevoir des **datagrammes** via ce **socket**, au moyen des méthodes suivantes :

**public void send(DatagramPacket data) throws IOException**

Permet d'envoyer les **données contenues** dans la variable **data** vers la **machine** et le **port** dont les **valeurs** ont été **spécifiées**.

**public synchronized void receive(DatagramPacket data) throws IOException**

Permet de recevoir un **datagramme** qui sera **stocké dans data**. Après appel, data contient les **données reçues**, leur **taille**, l'**adresse de l'envoyeur** ainsi que son **port d'attachement**.

Si le **message est trop long** pour être stocké, celui-ci est tronqué, et le reste est **perdu**. Il faut donc obligatoirement connaître leur taille à l’avance.

Il est aussi possible de spécifier un **délai d'attente maximum** en **réception** avec la variable **timeout.**

**public synchronized void setSoTimeout(int timeout) throws SocketException**

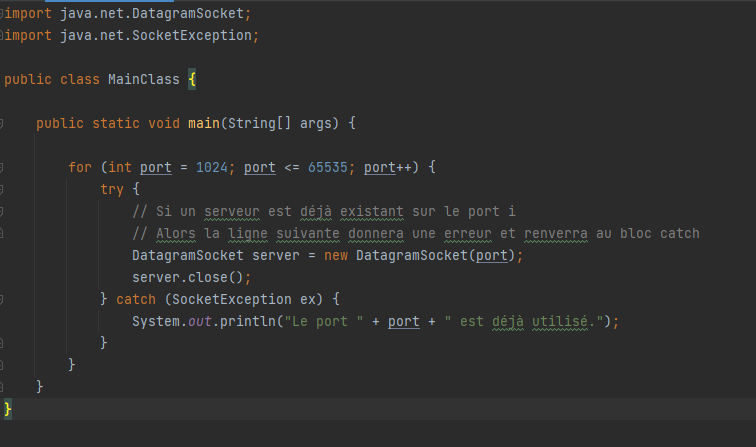
**public void close ()**

Ferme le **socket** et libère les ressources qui lui sont associées. Le **socket** est ensuite inutilisable.

**public int getLocalPort ()**

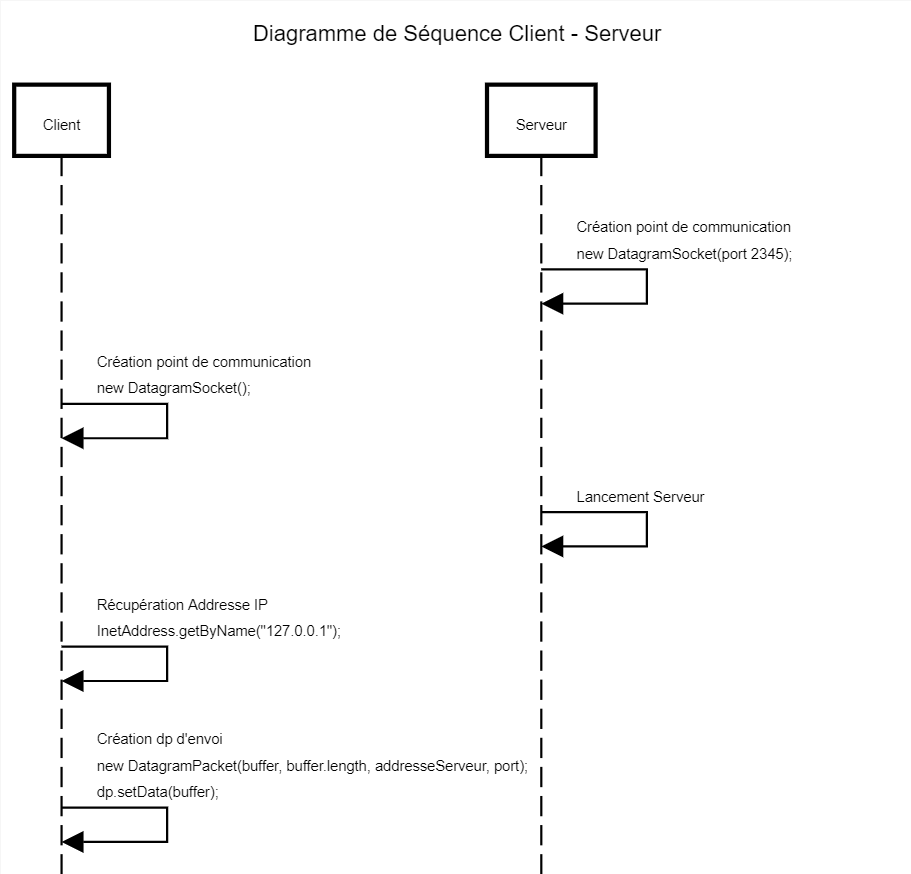
Retourne le **port d'attachement** du **socket.**

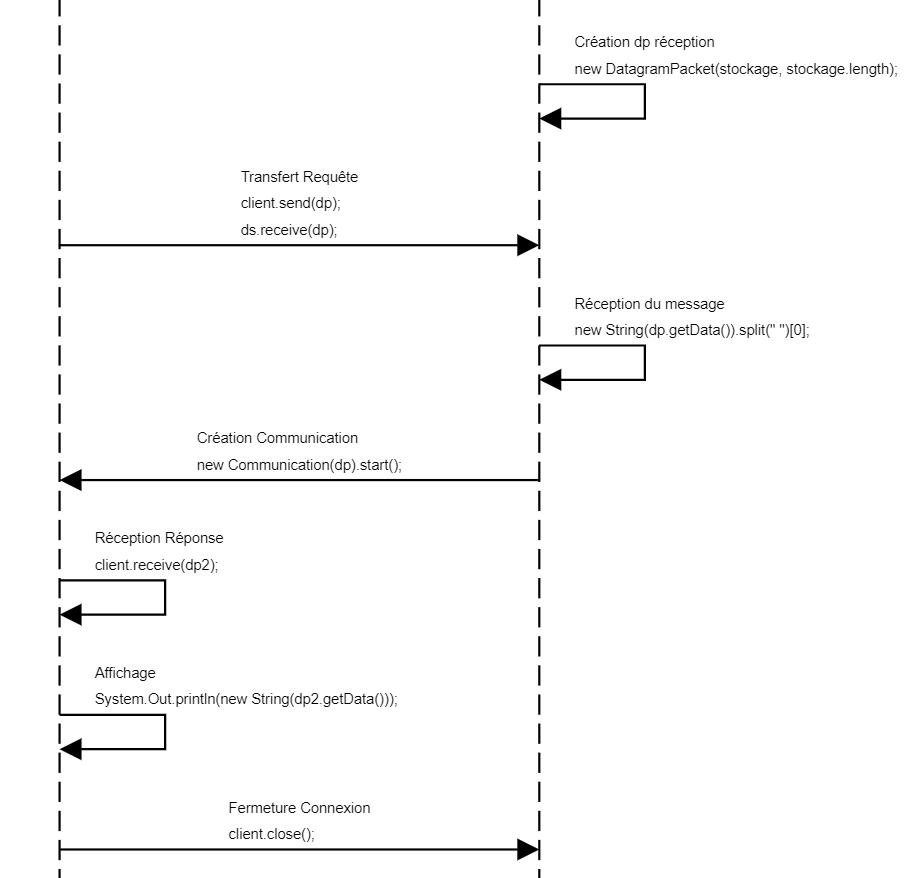
**II - Un scanner de ports UDP**

****

Pour chaque port i, si il existe un serveur sur ce port, le scanner va renvoyer le numéro du port i et nous informer qu’il y a un serveur qui l’utilise. Ainsi, on sait quel port est ouvert, lequel est fermé (s' il existe un serveur sur un port, alors il est ouvert). Lors de l’implémentation de notre programme, nous avons également utilisé l’exception levée lors de l’appel de l’ouverture d’un DatagramSocket sur un port occupé.

**III - Diagramme de Séquence Client - Serveur**

****

****

**IV - Développement Client-Serveur**

Veuillez retrouver notre en pièce jointe