TP N°7 : Programmation Réseau avec les sockets UDP

Objectifs du TP :

Le but de ce TP est de montrer la mise en œuvre du protocole de transport UDP (mode non connecté) au moyen des sockets en Java.

Principes généraux

Le mode de communication utilisé par le protocole UDP est le mode non connecté. Dans ce mode de fonctionnement, il n'est pas nécessaire de préparer la communication. Chaque datagramme est transféré indépendamment des autres. Ceci implique que chaque datagramme devra rechercher une route vers le destinataire. De ce fait, il n'y a pas de garantie, ni sur l'arrivée des paquets à destination, ni sur l'ordre d'arrivée de ces derniers. La seule garantie offerte par ce mode de communication est le fait que les données reçues sont sans erreur.

Voici un schéma d'émission et de réception d'un datagramme UDP en Java :

Emission

1. Création d'un socket UDP. Pour cela, utilisez un constructeur de la classe J*ava.net.DatagramSocket*.
2. Fabrication d'un paquet avec les données à envoyer ;
3. Envoyer le paquet à travers le socket.
4. Recommencer l´étape de fabrication si nécessaire.

Réception

1. Création d'un socket UDP sur un port spécifique (le port de réception des données). Pour cela, utilisez un constructeur de la classe *Java.net.DatagramSocket*;
2. Préparation d'un paquet coquille pour la réception des données ;
3. Attendre la réception d'un paquet ;
4. Récupérer les informations contenues dans le paquet si nécessaire (adresse et port de l´émetteur, données).
5. Utilisez les données en fonction de vos besoins.

Exercice 1 :

Copier puis tester sur même machine et deux machines distants le code des parties client et serveur.

Coté client :

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

public class ClientUDP {

    public static void main(String argv[]) {
        int port = 0;
        String host = "";
        Scanner keyb = new Scanner(System.in);

        try {
            // on récupère les paramètres : nom de la machine serveur et
            // numéro de port
            System.out.println("Adress du serveur : ");
            host = keyb.next();
            System.out.println("port d'écoute du serveur : ");
            port = keyb.nextInt();
            InetAddress adr;
            DatagramPacket packet;
            DatagramSocket socket;

            // adr contient l'@IP de la partie serveur
            adr = InetAddress.getByName(host);

            // données à envoyer : chaîne de caractères
            byte[] data = (new String("Hello Word")).getBytes();

            // création du paquet avec les données et en précisant l'adresse
            // du serveur (@IP et port sur lequel il écoute)
            packet = new DatagramPacket(data, data.length, adr, port);

            // création d'une socket, sans la lier à un port particulier
            socket = new DatagramSocket();

            // envoi du paquet via la socket
            socket.send(packet);

            // création d'un tableau vide pour la réception
            byte[] reponse = new byte[15];
            packet.setData(reponse);
            packet.setLength(reponse.length);

            // attente paquet envoyé sur la socket du client
            socket.receive(packet);

            // récupération et affichage de la donnée contenue dans le paquet
            String chaine = new String(packet.getData(), 0,
                        packet.getLength());
            System.out.println(" reçu du serveur : " + chaine);
        } catch (Exception e) {
            System.err.println("Erreur : " + e);
        }
    }
}
```

Coté serveur :

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

public class ServeurUDP {

    public static void main(String argv[]) {

        int port = 0;
        Scanner keyb = new Scanner(System.in);
        try {

            // on récupère le paramètre : port d'écoute
            System.out.println("port d'écoute : ");
            port = keyb.nextInt();

            DatagramPacket packet;

            // création d'une socket liée au port précisé en paramètre
            DatagramSocket socket = new DatagramSocket(port);

            // tableau de 15 octets qui contiendra les données reçues
            byte[] data = new byte[15];

            // création d'un paquet en utilisant le tableau d'octets
            packet = new DatagramPacket(data, data.length);

            // attente de la réception d'un paquet. Le paquet reçu est placé
            // dans packet et ses données dans data.
            socket.receive(packet);

            // récupération et affichage des données (une chaîne de caractères)
            String chaine = new String(packet.getData(), 0,
                    packet.getLength());
            System.out.println(" recu : " + chaine);

            System.out.println(" ca vient de : " + packet.getAddress() + ":" +
                  packet.getPort());

            // on met une nouvelle donnée dans le paquet
            // (qui contient donc le couple @IP/port de la socket coté client)
            byte[] reponse = (new String("bien recu")).getBytes();
            packet.setData(reponse);
            packet.setLength(reponse.length);

            // on envoie le paquet au client
            socket.send(packet);
        } catch (Exception e) {
            System.err.println("Erreur : " + e);
        }
    }
}
```

Exercice 2 :

On souhaite échanger entre le client/serveur les objets d'une classe voiture via les sockets UDP. La classe voiture

Pour cela le client Crée un objet voiture et l'envoi au serveur pour lui fixer une quantité de carburant avec la méthode *setCarburant().*

```
import java.io.*;

class voiture implements Serializable {

    private int carburant;
    private String model;
    private String type;
    private static int capacite = 300;

    voiture(String _type, String _model) {
        type = _type;
        model = _model;
        carburant = 0;
    }

    public void setCarburant(int c) {
        int maxi = capacite - carburant;
        if (c < maxi) {
            carburant += c;
            System.out.println("Le remplissage a été effectué sans problème.");
        } else {
            carburant = capacite;
            System.out.println((c - maxi) + " litre(s) de carburant ont
débordé.");
        }
    }
    public int getCarburant() {
        return carburant;
    }
    public int getCapacite() {
        return capacite;
    }
}
```

1. Ecrire la partie client de l'application et la partie serveur.
2. Tester cette application sur la même machine.
3. Tester cette application sur deux machines reliées par réseaux.

Exercice 3 :

Créez localement un serveur UDP qui écoutera sur le port 1250 et qui pour chaque datagramme reçu et quelque soit leur contenu, retournera à l'émetteur un datagramme contenant la date et l'heure courante.