Université Ain Temouchent BELHADJ Bouchaib
Institut des Sciences
Département de Mathématiques et Informatique

TP N°4

Manipulation de plusieurs types avec UDP

Soit les deux classes java suivantes :

```
import java.io.*;
import java.net.*;
public class ClientUDPint{
public static void main(String[] args) throws Exception {
DatagramSocket socket = new DatagramSocket();                    // ligne 5
InetAddress serveur = InetAddress.getByName("localhost");
ByteArrayOutputStream a = new ByteArrayOutputStream();
DataOutputStream b = new DataOutputStream(a);
// Ecrire : 10 true bonjour 1.2 dans le outputstream
b.writeInt(10);
b.writeBoolean(true);
b.writeUTF("bonjour");
b.writeDouble(1.2);
byte[] buffer = a.toByteArray();
DatagramPacket packet = new DatagramPacket(buffer,buffer.length,serveur ,2000);
socket.send(packet);
System.out.println("Client a envoyé 10 true bonjour 1.2 au serveur");
}
}
```

```
import java.io.*;
import java.net.*;
public class ServeurUDPint {
public static void main(String[] args) throws Exception {
DatagramSocket socket = new DatagramSocket(2000);
DatagramPacket packet = new DatagramPacket(new byte[1024] , 1024);
socket.receive(packet);
byte[] data = packet.getData();
ByteArrayInputStream a = new ByteArrayInputStream(data);
DataInputStream b = new DataInputStream(a);
System.out.println("Entier recu : "+b.readInt());
System.out.println("Boolean recu : "+b.readBoolean());
System.out.println("String recu : "+b.readUTF());
System.out.println("Double recu : "+b.readDouble());
}
}
```

Université Ain Temouchent BELHADJ Bouchaib
Institut des Sciences
Département de Mathématiques et Informatique

Master1 RID Systèmes distribués

TP N°4

- Lancez l'exécution des deux classes précédentes.
- Quel est l'intérêt de DataOutputStream et DataInputStream pour UDP ?
- Examinez la documentation de DatagramSocket dans le lien suivant https://docs.oracle.com/javase/7/docs/api/java/net/DatagramSocket.html et affichez la taille de la zone tampon (buffer) utilisée pour recevoir et pour envoyer les données.
- Dans le DatagramPacket du client, modifiez le numéro de port de 2000 à 3000 et lancez l'exécution des deux classes précédentes. Il n'y a pas d'erreurs. Pourquoi ?
- Reprendre le DatagramPacket du client avec la valeur 2000 pour le numéro de port et ajoutez dans la classe ClientUDPint.java, les deux instructions suivantes après la ligne 5 :

  ```
  System.out.println("PORT LOCAL "+socket.getLocalPort());
  System.out.println("PORT DISTANT "+socket.getPort());
  ```

- Faites plusieurs exécutions de cette classe. Comment se fait le choix du port local par le SE ?

  Le port distant affiche toujours -1.
- En gardant les deux affichages précédents. Ajoutez l'instruction suivante après la ligne 5 (devant les deux affichages). socket.connect(new InetSocketAddress ("localhost",2000)); et lancez l'exécution des deux classes précédentes. L'exécution se déroule normalement.
- Modifiez l'instruction précédente par : socket.connect(new InetSocketAddress ("localhost",1000));

  Faites une nouvelle exécution. Le code précise le numéro de port 1000 pour connect et 2000 pour DatagramPacket.
- Quelle est finalement la différence entre connect de TCP et connect de UDP ? Voir si c'est bloquante ou pas. UDP est un protocole déconnecté et malgré cela des exceptions sont levées dans le cas de connect (des erreurs).
- Déclarez maintenant un tableau d'entier (ex. int [ ] tab = {1, 6, 8, 9, 13, 10};) coté client. Ce dernier doit l'envoyer par UDP au serveur. Le serveur affichera par la suite son contenu (les entiers avec des sauts de lignes) → pensez à utiliser ObjectOutputStream : tab est un objet qu'on peut le convertir en octets pour l'envoyer avec UDP.
- Modifiez par la suite votre code afin que le serveur puisse répondre par le sous tableau contenant uniquement les entiers pairs ({6, 8, 10}).