Communication avec UDP - *package java.net*

Le but de ce TP est de réaliser deux programmes permettant le transfert d'un fichier. Un programme devra lire un fichier depuis le disque dur et l'envoyer en utilisant le protocole UDP. Le deuxième programme devra recevoir le fichier par le réseau et l'enregistrer sur le disque dur [1].

Pour les expérimentations, on utilisera des fichiers de tailles différentes, allant de 10Ko à 1 Mo, de préférence d'un format binaire comme une image ou un document pdf.

Les classes nécessaires

Dans les programmes émetteur et récepteur, il faut utiliser les classes :

- `DatagramSocket` : qui représente le point d'accès des programmes avec le réseau. L'émetteur doit en créer un sans paramètre, alors que le récepteur doit en creer un et indiquer sur quel port (entre 1024 et 65535) il recevra les données.
- `DatagramPacket` : qui représente un datagramme respectant le protocole UDP. L'émetteur doit en créer un et le remplir des données provenant du fichier avec la méthode `setData(byte[])` puis l'envoyer par le `DatagramSocket` vers le programme récepteur. Pour cela, il faut que le programme émetteur donne au `DatagramPacket` l'adresse IP de la machine sur laquelle le programme récepteur s'exécute ainsi que le numéro de port de réception (méthodes `setAddress(InetAddress)` et `setPort(int)`). L'adresse IP est obtenue par l'appel de la méthode statique :
  `InetAddress.getByName(''nom_machine'');`
  où "nom_machine" peut être "localhost" pour le cas où les deux programmes sont sur la même machine.
- `FileInputStream` : pour lire le fichier à transmettre, avec la métode `read(byte[])`.
- `FileOutputStream` : pour créer et écrire le fichier reçu avec la méthode `write(byte[])`.

Description du fonctionnement

Le `byte[]` lu depuis le fichier devra être mis dans le `DatagramPacket` avec sa méthode `setData(byte[])`. Le datagramme sera envoyé par la méthode `send(DatagramPacket)` du `DatagramSocket`. Le `byte[]` contenant les données ne devrait pas excéder la taille maximale d'un datagramme UDP qui est théoriquement de 65536 octets. Le système généralement se garde quelques octets et donc il faut trouver la taille maximale d'un `byte[]` à envoyer dans un `DatagramPacket`. C'est au moment de l'émission (méthode `send(DatagramPacket)`) qu'un exception java signale un paquet trop grand.

Le datagramme sera reçu dans le programme du récepteur par la méthode `receive(DatagramPacket)` et le `byte[]` contenu sera récupéré par la méthode `getData()` du `DatagramPacket` reçu. Ce `byte[]` pourra enfin être écrit dans le fichier reçu.

Le programme émetteur peut ajuster la taille du `byte[]` à envoyer car il connait la taille du fichier. En revanche, le récepteur doit prévoir la taille maximale car il ne peut pas savoir à

1. Comme il est possible de faire ce TP sur une seule machine, il faut faire attention à soit écrire les deux programmes dans des répertoires différents soit ne pas donner le même nom au fichier envoyé et reçu

l'avance la taille du fichier. Un effet secondaire est que le récepteur créera un fichier de la taille maximale, même si le fichier reçu est plus petit.

Attention, la grande majorité des méthodes utilisées peuvent générer des exceptions java, il est possible de toutes les prendre en compte par exemple en le signalant de la manière suivante :

```
public static void main(String[] args) throws Exception {
```

Ainsi, tout message d'erreur s'affichera à l'écran.

Tests et mesures

Dans le programme émetteur récupérer le temps du système avec la méthode statique :
`System.nanoTime()`
avant et après l'émission du datagramme. Faire 10 fois chaque mesure pour des fichiers allant de 10 Ko à 60 Ko par tranche de 10 Ko. Comme il y a une limitation dans la taille des datagrammes, on se limitera à des fichiers dont la taille est inférieure ou égale à celle des datagrammes. Tracer la courbe obtenue avec les moyennes de chaque série de mesures. Vérifier que le fichier reçu est bien le même que le fichier émis.