TP 2: Socket UDP/IPv4

Le but de ce TP est de voir la programmation de socket UDP tant côté client que côté serveur. Le CR est à rendre sur AMETICE avant la prochaine séance.

Client UDP

Un serveur netcat

Afin de pouvoir tester notre client, nous allons commencer par mettre en place un serveur simple en UDP avec la commande netcat

```
$ nc -u -l 1234
```

Client UDP Echo en Java

Le but de cette partie est d'écrire en Java un client UDP qui lit une ligne au clavier, l'envoie ensuite au serveur.

Une connexion UDP est faite avec un `DatagramSocket` sur lequel on peut faire `send(packet)` ou `receive(packet)`, où `packet` est un `DatagramPacket` (https://docs.oracle.com/javase/8/docs/api/java/net/DatagramPacket.html).

Pour le client, il est important de noter que l'adresse de destination est simplement précisée sur `packet` avec la méthode `setSocketAddress` (ou le constructeur).

On utilisera l'adresse `localhost` et le port `1234` pour créer un objet `InetAddress` permettant de définir l'adresse de destination. La méthode `send(packet)` sera ensuite utilisée pour envoyer les données (sous forme de `byte[]`).

1. pourquoi le client ne peut-il utiliser le port 1234 (comme port source) pendant votre TP?
2. pourquoi ne définit-on pas explicitement, en général, le port du client ?

Bien tester avec le serveur `netcat`.

Client Avancé

On complétera le client par

- une boucle permettant de lire au clavier jusqu'à fermeture de l'entrée standard (par `Ctrl+D`)
- une utilisation des paramètres de la fonction `main` pour pouvoir entrer les paramètres de serveur et port par la syntaxe suivante

```
java EchoUDPClient serveur port
```

- en l'absence de ces paramètres, un message d'erreur et d'usage sera affiché.

1. Quelle est la différence entre terminer le client par `Ctrl-C` ou `Ctrl-D`?

Serveur UDP

Fonctionnalité Echo

Le but du serveur est de faire un simple écho des lignes envoyées, précédé d'un `>`.

La méthode `receive(packet)` sera utilisée pour réceptionner le paquet. Pour décoder la chaine de caractère il faudra utiliser `new String(packet.getData())`.

Le renvoi de la nouvelle chaîne se fera, comme pour le client, avec `send(response)` où `response` est un `DatagramPacket`.

Test Fonctionnels 1

Vérifier que votre serveur fonctionne avec un client UDP `netcat`

```
$ nc -u localhost 1234
```

Tester avec le client

Utiliser votre client pour tester votre serveur.

Que se passe-t-il si vous utilisez deux clients en même temps.

Autres Clients Echo

python ou C

Ecrire un deuxième client dans un autre langage de programmation, comme C ou python.

Test

Tester ce client avec le serveur java. En quoi ce test illustre-t-il le paradigme client-serveur.

Tests d'intégration

Automatiser les tests pour toutes les combinaisons possibles de clients et serveurs.

Modifié le: vendredi 7 janvier 2022, 17:48