TP 3: Socket TCP/IPv4

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

Client TCP

Un serveur netcat

Afin de pouvoir tester notre client, nous allons commencer par mettre en place un serveur simple en TCP avec la commande netcat (ou ncat ou nc)

```
$ nc -l 1234
```

Client TCP Echo en Java

Se positionner dans un autre terminal

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

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 EchoTCPCLient serveur port

On pourra réutiliser le code correspondant au client UDP (TP 2).

Serveur TCP

Fonctionnalité Echo

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

Test Fonctionnels 1

Vérifier que votre serveur fonctionne avec un client `netcat` (voir TP1). Captez la communication avec tcpdump/Wireshark. Voir cette page pour l'utilisation des filtres pour montrer seulement les paquets IP concernés, par exemple avec une contrainte comme «tcp.port==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.

Expliquez.

Autres Clients Echo

python

Écrire un deuxième client dans un autre langage de programmation, comme python.

Test

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

Compatibilité UDP et TCP

Tester vos clients TCP avec le serveur UDP. Tester vos clients UDP avec le serveur TCP.