Université d'Ottawa
Faculté de génie

École de science informatique
et de génie électrique

uOttawa
L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Electrical Engineering
and Computer Science

Hiver 2023

À remettre le 3 février à 23h 59

Lab1 : Application de bavardage

1. **Objectif** :
   Le but du laboratoire est la conception et la mise en œuvre d'une application clients et serveur de bavardage orientée connexion (TCP/IP). Cette application est réalisée en utilisant la programmation de sockets. L'avantage de l'utilisation du protocole TCP/IP est la fiabilité : la couche transport effectue elle-même le "checksum", la réémission de morceaux de messages perdus, l'élimination des morceaux dupliqués, l'adaptation du débit.

2. **Réalisation**
   L'application des clients et de serveur de bavardage peut être réalisée en C/C++, Java ou Python utilisant la programmation de sockets. La figure 1 montre l'interconnexion entre le serveur et les clients. Lorsqu'un client veut communiquer avec un autre client, il doit d'abord s'enregistrer auprès du serveur. Puis, le client envoie le message à son interlocuteur en passant par le serveur. Si le client en question est en ligne, le serveur transmet le message. Sinon, il envoie un message indiquant « le client n'est pas en ligne ». Lorsqu'un client se déconnecte, le serveur doit fermer le socket.

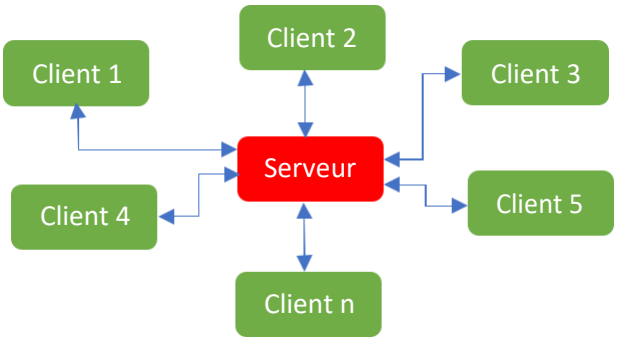

Figure 1 : Interconnexion entre le serveur et les clients

La figure 2 présente la démonstration de l'application client et serveur sur la distribution Linux basée sur Ubuntu. Ceci est une application de départ de votre Lab1. Comme vous pouvez le voir à la figure 2, il y a un programme serveur et trois programmes clients. Pour exécuter le système clients/serveur, il faut d'abord lancer le programme serveur sur le port 4444. Ensuite, on peut lancer les programmes clients. N'oubliez pas de compiler les programmes client et serveur avant d'exécuter. Ici, lorsqu'un client envoie un message, tous les autres clients le reçoivent aussi. Comme le système de bavardage, il faut envoyer le message au client choisi en utilisant par exemple un nom d'utilisateur (mohamed).

```
mohamed@Mohamed:/mnt/c/CEG3585/Lab1/server$ gcc -o server server.c -lpthread
mohamed@Mohamed:/mnt/c/CEG3585/Lab1/server$ ./server 4444
2.0.130.120 connected
New Client connected from port# 33400 and IP 127.0.0.1
2.0.130.122 connected
New Client connected from port# 33402 and IP 127.0.0.1
2.0.130.126 connected
New Client connected from port# 33406 and IP 127.0.0.1
```

```
mohamed@Mohamed:/mnt/c/CEG3585/Lab1/client$ ./client jean 4444
connected to 2.0.17.92, start chatting
alain:Bonjour à toutes et à tous
mohamed:salut
alain:comment ça va?
```

```
mohamed@Mohamed:/mnt/c/CEG3585/Lab1/client$ gcc -o client client.c -lpthread
mohamed@Mohamed:/mnt/c/CEG3585/Lab1/client$ ./client mohamed 4444
connected to 2.0.17.92, start chatting
alain:Bonjour à toutes et à tous
salut
alain:comment ça va?
```

```
mohamed@Mohamed:/mnt/c/CEG3585/Lab1/client$ ./client alain 4444
connected to 2.0.17.92, start chatting
Bonjour à toutes et à tous
mohamed:salut
comment ça va?
```

Figure 2 : Démonstration d'une application clients et serveur

3. Livrables

a. Description brève du but et de la théorie du problème.
b. Explication brève de votre algorithme de solution.
c. Document de conception en utilisant les diagrammes UML.
d. Captures d'écrans de la démonstration de l'application
e. Discussion
f. Conclusion

4. Évaluation

Le fonctionnement de l'application 50% et le rapport 50%.