Projet final

L'évaluation du cours sera faite sur la base d'une présentation de projet.

Le projet est un serveur (et client) de chat à la irc (slack, sans le support media).

Minimum

Fonctionalités

Les fonctionalité demandée pour valider sont:

- Un serveur central auquel tous les participants communiquent (peut avoir une ip fixe et un port fixe).
- Un client permettant d'envoyer et de recevoir des messages du serveur central.
- Le client doit permettre d'écrire des messages tout en permettant de voir les messages envoyés par d'autres clients.

Les clients doivent tous se connecter avec la même commande, et ce quelque soit le nombre d'clients déjà connectés.

Quand un client se connecte ou se déconnecte, les autres clients doivent être prévenus.

Quand un client perd sa connexion ou que le programme est interrompu, il doit être déconnecté du serveur (avec une minute de tolérance).

La communication minimale est celle d'un canal de discussion commun auquel tous les clients ont accès. Chaque message envoyé par un client sur ce canal doit etre reçu par tous les clients connectés à ce canal.

Le serveur doit être capable d'accepter au minimum 20 connections en parallèle et envoyer les messages aux clients en moins d'une seconde.

La contrainte d'ordonancement des messages est la suivante: si un client U envoie le message m1 puis le message m2, les autres clients doivent les recevoir dans cet ordre. Cependant il n'y a pas de priorité entre les clients.

Vous devez pouvoir montrer que des clients sur différentes machines peuvent communiquer via ce serveur (machines virtuelles sur la même machine physique autorisées).

Qualité

- Le code doit être versioné sous git.
- Le code doit être testé (coverage de 50% minimum).
- Le code doit passer le test de memcheck (valgrind).

- Le code doit avoir une intégration continue (je recommande github pour héberger avec travis ou circleCI).
- Le code doit être commenté.
- Le code doit être compilable et lançable sous linux (Ubuntu 16.04 LTS).
- Le code doit avoir des instructions de compilation reproductibles, i.e. une liste de paquets à récupérer avec `apt-get`, une commande de build `make` et lançable en une ligne.
- La qualité du code sera évaluée.

Contraintes

Les bibliothèques externes sont authorisée dans la limite du "bon gout":

- Toute bibliothèque devra être justifiée.
- Les dépendances doivent être explicites et résolue dans la phase d'instalation.
- Elles ne doivent pas faire tout le travail (zmq et rabbit-mq sont interdites par exemple).

Dans le cadre de l'interface du client toutes les bibliothèques sont authorisées, je vous recommande cependant de vous limiter à une interface en ligne de commande (Qt est un enfer que je ne souhaite à personne).

Pour la partie réseau, cpp-netlib ainsi que tout ce qui est dans la bibliotheque standard sont authorisés.

Conseils

Pour vous familiariser avec les notions de réseau je vous conseil de suivre le tutoriel de rabbit-mq (dans le langage de votre choix), en effet ce projet fait parti des tutoriel de cette bibliothèque.

Vous pouvez utiliser netcat ou wget pour debugger la partie réseau.

Le style guide c++ de google vous donnera de bons guide sur ce qu'est du code de 'qualité'.

Pour l'interface du client `ncurses` pourra vous faciliter la tache.

Un bon tutoriel sur les sockets est celui de tutorialpoints.

Fonctionalité supplémentaires

Ces fonctionalité feront augmenter votre note, cependant si le minimum n'est pas atteint, elles ne compteront pas.

- Sauvegarder l'historique de conversation sur le serveur dans un fichier ou une base de donnée (je conseille postgres).
- Vérifier que deux clients n'ont pas le même identifiant.
- Gérer les connexion de manière multithreadée.
- Permettre aux clients de demander l'historique (un moyen de passer des commandes depuis le client est traditionelement d'envoyer un message sous le format `/commande` `arguments`).
- Permettre l'envoi de messages personels (visibles uniquement entre les deux clients concernés).
- Permettre la création de canaux de discussions par les clients (channels sur irc/slacks).

Les évaluations et contraintes sont les mêmes que pour la partie précédente.