Allan Mercou
Léon Souffes

1. Rappel de analyse du sujet

Le but de ce TP est de faire communiquer plusieurs clients avec un serveur via RPC. Ainsi, nous avons développé un tchat permettant à plusieurs clients de communiquer avec un serveur. Nous abordons dans ce TP le RPC, géré par les librairies jportmap, jrpcgen et oncrpc. Le tout est géré par des packets UDP. C'est à dire que nous ne sommes pas connectés.

2. Choix techniques effectués

Pour ce faire, nous avons tout d'abord créé un fichier en langage RPCL :

```
typedef string Chaine<255>;
typedef Chaine MessageList<255>;

program CHAT_PROG {
        version CHAT_VERSION {
        int connect(Chaine nickname)=0;
        void sendMessage(Chaine message)=1;
        MessageList resyncMessages(void)=2;
        }=1;
} = 0x22222220;
```

Ainsi, nous définissons une chaine, qui a pour type primaire "string". Ensuite, nous avons un MessageList, qui est un type qui va contenir plusieurs chaines, notre liste de messages, ici, donc.

Nous avons 3 méthodes :

- connect, qui prend en paramètres un pseudo. Elle sert juste à introduire un nouvel utilisateur dans le tchat. On ne se connecte pas réellement puisque le RPC ne sauvegarde pas l'adresse ou le pseudo de l'utilisateur et que c'est du UDP.
- sendMessage, qui prend en paramètre une chaine, permet d'envoyer un message au serveur. Dès lors, le serveur enregistre le message et le renverra aux clients demandant la synchronisation des messages.
- resyncMessages ne prenant pas de paramètres, permet au client de demander au serveur de lui retourner l'entièreté des messages. Ainsi, les clients demandent en boucle les messages et peuvent ainsi se synchroniser entre eux et instantanément recevoir les messages, puisque le RPC tourne en UDP, non connecté.

Du côté du client, il fonctionne de la façon suivante :

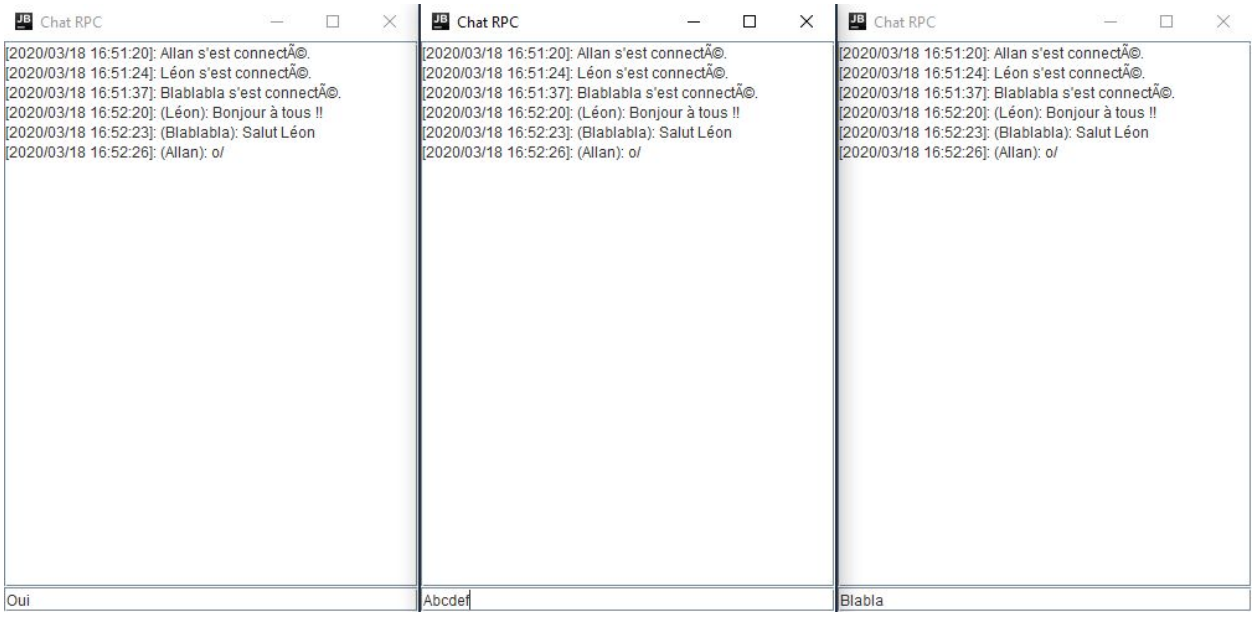
- On se connecte
- On envoie son pseudo

- On envoie des messages

Et en parrallèle, on demande au serveur de nous renvoyer tous les messages.

3. Résultats et tests

Ainsi, comme le montre l'image, nous avons ici trois clients qui peuvent communiquer entre eux:



4. Conclusion

Il n'y avait pas grand chose de difficile puisque tout le côté bas niveau est géré par les librairies. Cependant, il est très largement possible d'améliorer le programme : on pourrait ajouter des canaux spécifiques, des commandes, des pseudonymes uniques… Mais on se rapprocherait rapidement du protocole IRC. Il serait aussi probablement préférable de communiquer en TCP pour gérer la connexion et déconnexion des utilisateurs.