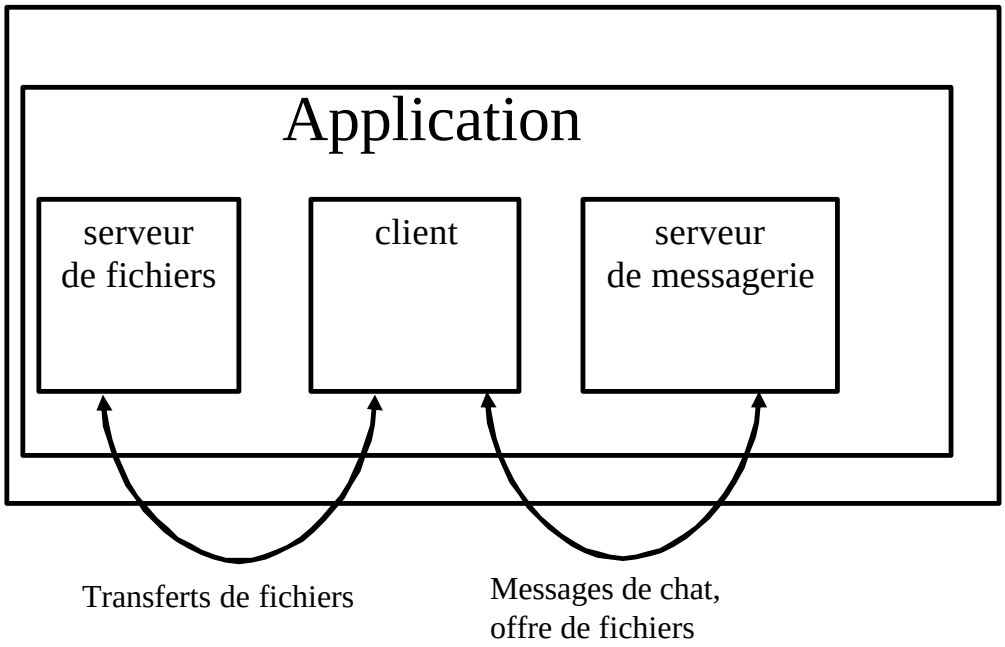
Communications Unix : TP2

Préambule :

L'objectif de ce TP (sur les deux prochaines séances) est de mettre en place une application de peer-to-peer décentralisée, à l'aide du multicast. Cette application intégrera également un outil de transfert de fichiers, et un module de chat/recherche de fichiers.



Le Client:

Le client propose les fonctionnalités suivantes à travers un menu textuel :

1. Emettre un message au groupe de discussion
2. Récupérer un fichier
3. Quitter
4. (Offrir un fichier aux membres du groupe (en définissant un mot de passe))

Le serveur de fichiers:

Basé sur le TP1, ce serveur offre en TCP les services suivants:

1. Servir (uploader) un fichier partagé aux utilisateurs qui connaissent le mot de passe
2. (Enregistrer un fichier associé à un mot de passe comme partagé avec les autres membres du groupe)

Le serveur de messagerie:

Ce serveur offre en UDP les services suivants:

1. Réception de messages adressés au groupe de discussion, et leur affichage en continu
2. (Réception d'une requête correspondant à un nouveau fichier partagé)

Yann.Thierry-Mieg@lip6.fr

Partie I: Le chat (6 points)

Serveur de Messagerie : [Dans une série de nouveaux fichiers ChatServer.c…]

- En vous inspirant des sources du support du cours, écrivez une fonction `newChatServer(IP,port)` qui attends indéfiniment des messages sur une adresse de multicast `IP` sur le port `port` sur l'interface eth0 (utilisez gethostname et gethostbyname pour obtenir l'adresse de eth0) et affiche les messages reçus sur stdout avec l'adresse de l'émetteur i.e: 132.227.64.88 :> coucou !

Client: [Dans une série de nouveaux fichiers ChatClient.c]

- Reprenez les sources du support pour écrire un client multicast qui boucle indéfiniment sur un prompt "Votre message ?" et émet le message vers une adresse de multicast fixée
- **(Fixez IP et port par des constantes " en dur)**

A ce stade notre programme de chat est capable d'envoyer et de recevoir des messages sur un groupe de multicast.

Partie II: Le transfert de fichiers (8 points)

Nous allons reprendre le TP1 pour permettre le transfert de fichiers:

Client: [Dans une copie `FileClient.c` de votre code client du TP1]

- Ecrivez une fonction qui prenne en argument une IP, un numéro de port, un mot de passe et un nom de fichier, `getFileFromServer(IP,port,pwd,filename)` et qui essaie de se connecter au serveur à l'adresse `IP:port` en TCP, s'authentifie avec le mot de passe `pwd` et demande le fichier `file` avant de terminer la connexion. (Cette fonction pourrait rendre un code d'erreur si une des étapes échoue).
- Modifiez votre fonction pour écrire le fichier reçu en local. Utilisez :

`open (),write(),close()`

- **Eliminez le main**

Serveur: [Dans une copie `FileServer.c` de votre code serveur du TP1] (2 points)

- Ecrivez une fonction qui prenne en argument un numéro de port et un mot de passe, `newFileServer(port,pwd)` et qui démarre un serveur de fichiers multithreadé développé au TP2 sur le port `port` sur l'interface eth0, avec le mot de passe `pwd` comme authentification et attends indéfiniment des requêtes de clients.
- **Eliminez le main**

Partie III: Intégration (5 points)

Client: [Dans le code de ChatClient]

- Ajoutez un fork qui instancie un nouveau processus qui appelle : `newFileServer(port,pwd)`
- Ecrivez un menu qui offre les options 1 à 3 du préambule (3 points)

1. Prompte pour le message et l'envoie
2. Demande un fichier au serveur. Pour simplifier essayez une forme comme : (2 points)

Yann.Thierry-Mieg@lip6.fr

"IP","port","pwd","file" que vous reconnaitrez à l'aide scanf("%s,%d,%s,%s ",…)
Utilisez getFileFromServer développé plus haut, appelé par un nouveau processus (fork).

3. Quitte l'application

A ce stade notre application permet le partage de fichiers : il suffit d'envoyer un message au groupe comme : >>> venez prendre sur : 127.12.34.56,1664,monpass,monImage.jpg et qu'un autre membre du groupe copie/colle cette adresse dans son menu (2) pour échanger des fichiers.

Extensions à envisager : (2 points par extension utilisant un nouveau mécanisme de communication inter-processus)

- Proposer un fichier par une requête particulière (qui remplit IP,port,pass)
- Différencier une trame proposant un fichier d'un message de chat, et stocker les fichiers « vus »
- Mettre en place une communication qui permette au serveur de chat d'informer le client des fichiers rencontrés. On pourra se baser sur un fichier (attention utiliser « flock » pour empêcher les accès concurrents)
- Protéger chaque fichier avec un mot de passe différent.
- Mettre en place une communication qui permette au client de piloter le serveur de fichiers, (à travers une interface sur lo:127.0.0.1 par exemple) indiquer quels sont les fichiers partagés, interroger les statistiques du serveur…
- …

Notation et relevé des TP

Le barême indiqué est purement indicatif, et se rapporte à une note sur 20. La notation prendra également en compte la propreté du code, les commentaires, la structuration en modules etc. (1 point)

Les TP seront à rendre avant le Lundi 10 Novembre minuit, par email à l'adresse Yann.Thierry-Mieg@lip6.fr, sous la forme d'un mail portant en copie votre binôme, et ayant pour Objet : [TPUNIX] Groupe xx Binome1 Binome2, et une pièce jointe au format tgz ou zip nommée Binome1_Binome2.tgz contenant uniquement les sources et éventuellement des traces d'exécution et/ou des commentaires textuels (PAS d'exécutables binaires svp).

Respectez ces consignes pour faciliter la notation et le filtrage dans ma messagerie, merci.

Je vous enverrai une réponse à la réception, si vous tardez à la recevoir renvoyez moi un mail.

Ne copiez pas vos fichiers sur ceux d'un autre groupe, la commande cp n'étant pas au programme de ce cours.

N'hésitez pas à me contacter si vous avez besoin d'éclaircissements.