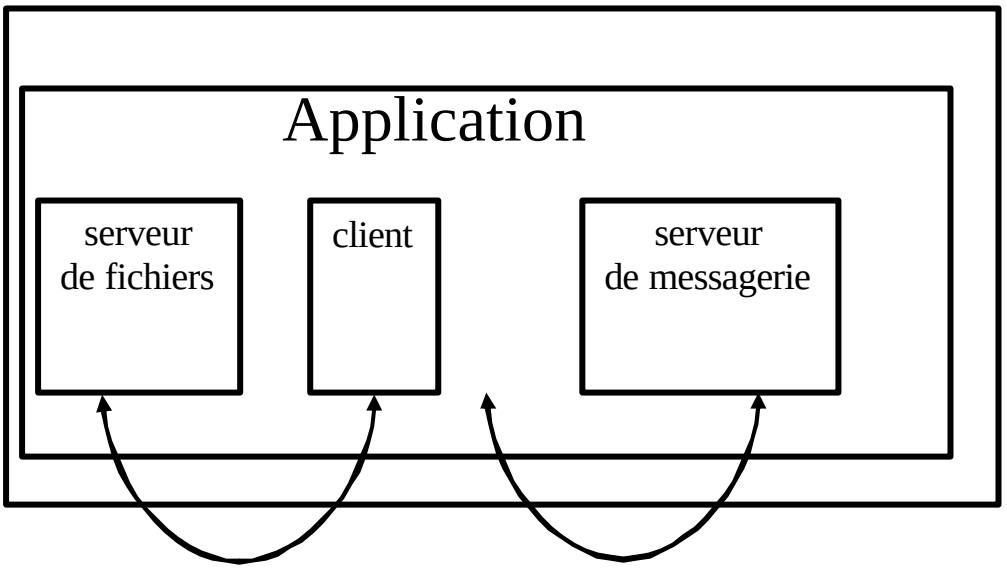
# Python et Unix : TP3

## *Préambule :*

L'objectif de ce TP 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 TP2, 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)

## *Partie I: Le chat (10 points)*

**Serveur de Messagerie :** [Dans un nouveau script ChatServer.py] *(5 points)*

- En vous inspirant des sources du support du cours 2, écrivez une fonction `newChatServer(IP,port)` qui attends indéfiniment des messages sur une adresse de multicast `IP` sur le port `port` et affiche les messages reçus sur stdout avec l'adresse de l'émetteur i.e: 132.227.64.88 :> coucou !
- **Eliminez toutes les portions qui exécutent du code quand on importe le module (que la def de cette fonction)**

**Client**: [Dans un nouveau script ChatClient.py] *(5 points)*

- 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 *(3 points)*
- Ajoutez un appel à *(2 points)*
  `thread.start_new_thread(ChatServer.newChatServer(IP,PORT))`
- **(Fixez IP et port par des constantes " en dur)**

A ce stade notre script client est capable d'envoyer et de recevoir simultanément des messages sur un groupe de multicast.

## *Partie II: Le transfert de fichiers (6 points)*

Nous allons reprendre les scripts du TP2 pour les rendre plus modulaires:

**Client**: [Dans une copie `FileClient`.py de votre script client du TP2] (4 points)

- 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,file))` 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 essayez try: et except: vus au TP2 pour récupérer les erreurs).*(3 points)*
- Modifiez votre script pour écrire le fichier reçu en local. Utilisez : *(1 point)*

```
f = open (path_to_fich,'w')
f.write(string)
f.close()
```

- **Eliminez toutes les portions qui exécutent du code quand on importe le module (que des def de fonctions)**

**Serveur**: [Dans une copie `FileServer`.py de votre script serveur du TP2] (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 toutes les portions qui exécutent du code quand on importe le module (que des def de fonctions)**

## *Partie III: Intégration (4 points)*

**Client**: [Dans le script ChatClient.py] *(4 points)*

- Ajoutez un appel à *(1 point)*
  `thread.start_new_thread(FileServer.newFileServer(port,pwd) (IP,PORT))`
- Ecrivez un menu qui offre les options 1 à 3 du préambule (3 points)

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

```
(IP,port),(pwd,fname) = input('  ("IP",port),("pwd","file")? ')
FileClient.getFileFromServer((IP,port),(pwd,fname))
```

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 ("monIP",monPort),("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:*

- Proposer un fichier par une requête particulière (qui remplit IP,port,pass)
- Créer un dictionnaire des fichiers proposés par les autres peer (forme d'indexation du contenu du réseau P2P)
- Protéger chaque fichier avec un mot de passe différent, être capable d'indiquer au serveur de fichiers quels sont les fichiers partagés (interface sur lo:127.0.0.1 par exemple)
- …