**Server Project (Chat)**

# Connexion des clients au serveur en UDP ou TCP

On utilise le protocole de transmission TCP car on veut que les paquets soient surs d’arriver à leur destination.

# Capacité d’envoyer et recevoir des messages (chat) (280 caractères par message)

Utiliser netcat (cf TD5)

# Transférer des fichiers (jpeg ou autres) (client à client, client à serveur)

# Multi-threaded plus de clients (50) par server

# Ligne de commande sur le serveur (#Kill Tim pour déconnecter le client Tim)

## Fonctions du serveur :

### #Help (list command)

### #Exit (server shutdown)

### #Kill

### #ListU (list of users in a server)

### #ListF (list of files in a server)

### # Private (private chat with another user)

### #Alert

## Fonctions du client :

### #Help (list command)

### #Exit (client exit)

### #ListU (list of users in a server)

### #ListF (list of files in a server)

### #Ring (notification if the user is logged in)

### #TrfU (Upload file transfer to a server)

### #TrfD (transfer Download file to a server)

### # Private (private chat with another user)

### #Public (back to the public)

# Le serveur nécessite un username et un mot de passe pour se connecter. (SQLite pour traiter les password) la database doit être sur le serveur

Utilisation de argparse pour récupérer en paramètre de lancement de programme le username et password de l’utilisateur

# Log log (login, @IP, date, etc.) on the server

Simple formulaire de login avec récupération de données fournies et check avec la DB

# Notre fonctionnalité originale

Faire des chat rooms

**Bonus**

# Le serveur tourne sur Linux et Windows

On est obligé vu qu’on travaille avec Linux et Windows

# Transmission sécurisée avec un chiffrement de bout-en-bout

Valot reprend son programme de chiffrement d’Australie