Babel : protocole binaire de communication

1. Communication client - serveur

La communication entre un client et le serveur (utilisant le protocole **TCP**) se fait selon un modèle précis : chaque **requête** d'un client contient d'abord **un header**. Ce header contiendra lui même deux choses :

- d'abord **un magic number**, ce dernier nous permettra de nous assurer que le la donnée reçue provient bien de l'un de nos programme et pas d'un programme extérieur.
- **un numéro de commande**, ce dernier servira à savoir comment se servir des informations qui suivent, comment les traiter. Le numéro de la commande sera l'un de ceux-ci :
    - 0 : commande "register"
    - 1 : commande "login"
    - 2 : commande "get contacts"
    - 3 : commande "add contact"
    - 4 : commande "start call"
    - 5 : commande "stop call"
    - 6: commande "disconnect"
    - 7 : commande inconnue (utile côté serveur)

Le header est ensuite suivi d'**informations utiles** au traitement de la demande.

Concernant la **réponse** du serveur, celle-ci se fera selon le même modèle : le **header** sera toujours présent, celui-ci sera suivit, à la différence d'une requête du client, d'un **code de réponse** correspondant à l'un des codes suivants :

- 0 : ok
- 1 : non connecté
- 2 : déjà connecté
- 3 : mauvais contact
- 4 : mauvaise combinaison nom d'utilisateur - mot de passe
- 5 : nom d'utilisateur trop court (minimum 3 caractères)
- 6 : mot de passe trop court (minimum 3 caractères)
- 7 : nom d'utilisateur déjà utilisé
- 8 : l'utilisateur appelé est déconnecté
- 9 : l'utilisateur appelé est déjà en appel
- 10 : aucun appel en cours
- 11 : adresse ip non-valide
- 12 : autre

Le code de réponse sera suivit des **données demandée par le client**.

Commande “register” :

- **Description** : commande utilisée lorsqu’un utilisateur souhaite se créer un compte.
- **Body de la requête** :
  - Nom d’utilisateur
  - Mot de passe
- **Codes de réponse possibles** :
  - 0
  - 5
  - 6
  - 7
  - 12
- **Body de la réponse** : vide

Commande “login” :

- **Description** : commande utilisée lorsqu’un utilisateur souhaite se connecter sur la plateforme
- **Body de la requête** :
  - Nom d’utilisateur
  - Mot de passe
  - Ip de l’utilisateur
- **Codes de réponse possibles** :
  - 0
  - 2
  - 4
  - 11
  - 12
- **Body de la réponse** : vide

Commande “get contacts” :

- **Description** : commande utilisée pour récupérer les contacts d’un utilisateur connecté
- **Body de la requête** : vide
- **Codes de réponse possibles** :
  - 0
  - 1
  - 12
- **Body de la réponse** :
  - la liste de contacts de l’utilisateur

Commande “add contact” :

- **Description** : commande utilisée pour ajouter un contact
- **Body de la requête** :

    - Nom d’utilisateur du contact à ajouter

- **Codes de réponse possibles** :
    - 0
    - 1
    - 3
    - 12
- **Body de la réponse** :
    - la nouvelle liste de contacts de l’utilisateur

Commande “start call” :

- **Description** : commande utilisée pour commencer un appel
- **Body de la requête** :
    - la liste des utilisateurs à appeler
- **Codes de réponse possibles** :
    - 0
    - 1
    - 8
    - 9
    - 12
- **Body de la réponse** :
    - la liste des utilisateurs appelés avec leurs IPs, cette liste est également envoyée à tous les utilisateurs appelés

Commande “stop call” :

- **Description** : commande utilisée pour stopper un appel, cette commande enverra à tous les utilisateurs dans l’appel que l’appel est terminé.
- **Body de la requête** : vide
- **Codes de réponse possibles** :
    - 0
    - 1
    - 10
- **Body de la réponse** : vide

Commande “disconnect” :

- **Description** : commande utilisée lorsqu’un utilisateur se déconnecte
- **Body de la requête** : vide
- **Codes de réponse possibles** :
    - 0
    - 1
- **Body de la réponse** : vide

2. Communication client - client

Lors d'un appel, les données audio sont transmises de client à client par **UDP**. Un paquet envoyé à un client contient **quatre choses** :

- **Un magic number,** ce qui permet de vérifier que le paquet appartient bien à notre programme
- **Un timestamp**, permettant de ne pas accepter les paquets plus vieux que le dernier paquet reçu (cela provoquerait un problème de son)
- **la taille de l'échantillon de son**, ce qui permet de vérifier que ce dernier soit complet
- **l'échantillon de son**