**Babel - RFC**

**Simon Ninon – Navid Emad – Alain Nguyen – Davy Tran – Pierre Guego**

**Dernière mise à jour : 4 novembre 2014**

**Préambule**

Le Babel est un programme de communication vocal (VOIP).

Ce document explique comment les clients communiquent avec le serveur et comment les clients communiquent entre eux.

**Communication Client - Client**

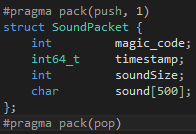
1. ***Préambule***

La communication “client - client” est utilisée dans le cadre de l’envoi de paquet de sons.

La communication se fait en **protocole UDP** afin que la communication audio soit plus performante.

1. ***SoundPacket***

Les paquets audio sont formatés grâce à une structure “*SoundPacket*”:



|  |  |  |
| --- | --- | --- |
| **Type** | **Nom** | **Utilité** |
| int | magic\_code | Permet de vérifier la validité du paquet. Le magic code doit valoir: **0x150407CA** |
| int64\_t | timestamp | Date de l’envoie du paquet de son. Si la date est antérieure au paquet précédemment reçu, le client peut ne pas traiter le paquet afin de gagner en performance. |
| int | soundSize | taille du paquet de son dans le buffer “sound” |
| char[500] | sound | Buffer de son. Ce buffer contient un champ de bytes qui correspond à du son encodé par le codec “opus” |

Il est important de noter dans l’exemple la présence du “*#pragma pack*”. Cette directive permet de forcer le compilateur à ne pas faire de padding afin de ne pas avoir de soucis lors de la récupération et de l’envoie des paquets.

1. ***Comportement***

L’émetteur d’un paquet ne se soucie pas d’un éventuel problème de communication et de réseau et n’intègre pas de gestion de timeout.

Tout paquet ne comportant pas le bon magic code ou ayant un timestamp inférieur au précédent paquet reçu ne doit pas être traité.

Le son transmis doit obligatoirement avoir été encodé au préalable.

**Communication Client - Serveur**

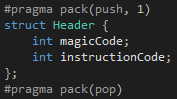
1. ***Préambule***

La communication client - serveur est au cœur du Babel. C’est le serveur qui gère les comptes utilisateurs, qui centralise les informations et transmets des informations entre les contacts.

La communication client - serveur se fait par le **protocole TCP** afin d’assurer une bonne transmission des commandes.

1. ***Formatage des commandes***

Toutes les commandes, qu’elles soient envoyées par un client ou par un serveur, possèdent le même header.



|  |  |  |
| --- | --- | --- |
| **Type** | **Nom** | **Utilité** |
| int | magic\_code | Permet de vérifier la validité du paquet. Le magic code doit valoir: **0x150407CA** |
| int | instructionCode | Permet d’identifier la commande envoyée. |

Même si les paquets possèdent tous le même header, ils possèdent en revanche un “*body*” différent: chaque commande possède sa propre structure, adaptée à ses besoins.

C’est grâce à l’instruction code que le client ou le serveur sera en mesure de déterminer ce qu’il va recevoir comme body.

1. ***Formatage des commandes envoyées par le serveur***
   1. **add**

Utilisée pour notifier un client qu’un autre client souhaite l’ajouter en contact.



|  |  |  |
| --- | --- | --- |
| **Type** | **Nom** | **Utilité** |
| char[256] | accountName | Nom de compte de la personne qui envoie la demande |

* 1. **call**

Utilisée pour notifier un client qu’un de ses contacts souhaite l’appeler.



|  |  |  |
| --- | --- | --- |
| **Type** | **Nom** | **Utilité** |
| char[256] | accountName | Nom de compte de la personne qui envoie la demande |

* 1. **close\_call**

Utilisée pour notifier un client que l’appel en cours est terminée.



|  |  |  |
| --- | --- | --- |
| **Type** | **Nom** | **Utilité** |
| char[256] | accountName | Nom de compte de la personne avec qui le client est en communication |

* 1. **del**

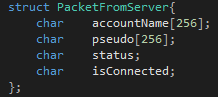
Utilisée pour notifier un client qu’un autre client l’a supprimé de sa liste de contact.



|  |  |  |
| --- | --- | --- |
| **Type** | **Nom** | **Utilité** |
| char[256] | accountName | Nom de compte de la personne qui a supprimé le client de sa liste de contact |

* 1. **show**

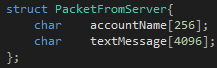
Utilisée pour donner des informations à un client concernant un utilisateur.



|  |  |  |
| --- | --- | --- |
| **Type** | **Nom** | **Utilité** |
| char[256] | accountName | Nom de compte de la personne dont on transmet les informations |
| char[256] | pseudo | Pseudo de la personne dont on transmet les informations |
| char | status | Statuts de la personne dont on transmet les informations. Ce statuts correspond à un enum définit plus loin dans cette documentation |
| char | isConnected | Indique si la personne est actuellement connectée. Cet attribut correspond à un booléen et contiendra 0 ou 1 (1 signifiant que l’utilisateur est connecté) |

* 1. **send**

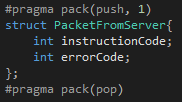
Utilisée pour transmettre un message d’un utilisateur à un autre utilisateur.



|  |  |  |
| --- | --- | --- |
| **Type** | **Nom** | **Utilité** |
| char[256] | accountName | Nom de compte de la personne qui envoie le message |
| char[4096] | textMessage | message |

* 1. **err**

Utilisée pour notifier un utilisateur que la dernière action effectuée s’est bien déroulée ou non.

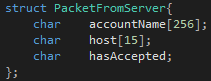


|  |  |  |
| --- | --- | --- |
| **Type** | **Nom** | **Utilité** |
| int | instructionCode | numéro de l’instruction concernée. Cet attribut correspond à un enum définit plus loin dans cette documentation |
| int | errorCode | Code d’erreur. Cet attribut correspond à un enum définit plus loin dans cette documentation |

* 1. **accept\_call**

Cette commande permet 2 choses:

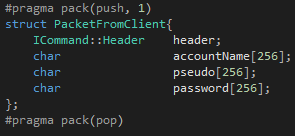
* notifier à un utilisateur que le contact qu’il a essayé d’appeler à accepter, ou non, la demande d’appel
* Informer les 2 contacts qui souhaitent communiquer à quelle IP ils doivent d’adresser.



|  |  |  |
| --- | --- | --- |
| **Type** | **Nom** | **Utilité** |
| char[256] | accountName | Nom de compte du contact avec qui le client va communiquer |
| char[15] | host | Adresse IP du contact avec qui le client va communiquer |
| char | hasAccepted | Indique si la personne qui a reçu la demande d’appel à accepter ou non la demande |

1. ***Formatage des commandes envoyées par le client***
   1. **reg**

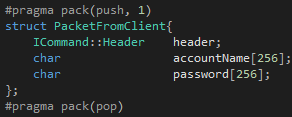
Utilisée pour créer un compte.



|  |  |  |
| --- | --- | --- |
| **Type** | **Nom** | **Utilité** |
| char[256] | accountName | Nom de compte |
| char[256] | pseudo | pseudo |
| char[256] | password | mot de passe |

* 1. **log**

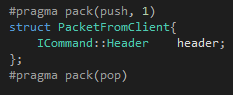
Utilisée pour s’authentifier et se connecter.



|  |  |  |
| --- | --- | --- |
| **Type** | **Nom** | **Utilité** |
| char[256] | accountName | Nom de compte |
| char[256] | password | mot de passe |

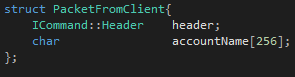
* 1. **list**

Utilisée pour demander la liste de contact. Cette commande ne prend pas de paramètres.



* 1. **show**

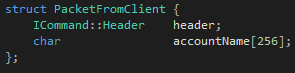
Utilisée pour demander des informations sur un contact.



|  |  |  |
| --- | --- | --- |
| **Type** | **Nom** | **Utilité** |
| char[256] | accountName | Nom de compte de la personne dont on souhaite obtenir des informations |

* 1. **add**

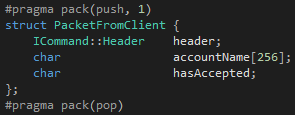
Permet d’envoyer une demande d’ajout de contact.



|  |  |  |
| --- | --- | --- |
| **Type** | **Nom** | **Utilité** |
| char[256] | accountName | Nom de compte de la personne que l’on souhaite ajouter |

* 1. **accept\_add**

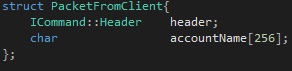
Permet de répondre à une demande d’ajout de contact.



|  |  |  |
| --- | --- | --- |
| **Type** | **Nom** | **Utilité** |
| char[256] | accountName | Nom de compte de la personne qui souhaite nous ajouter en contact |
| char | hasAccepted | Réponse à la demande |

* 1. **call**

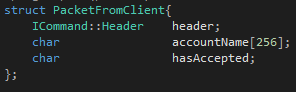
Utilisée pour envoyer une demande d’appel à un contact.



|  |  |  |
| --- | --- | --- |
| **Type** | **Nom** | **Utilité** |
| char[256] | accountName | Nom de compte de la personne que l’on souhaite appeler |

* 1. **accept\_call**

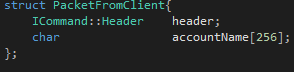
Utilisée pour répondre à une demande d’appel.



|  |  |  |
| --- | --- | --- |
| **Type** | **Nom** | **Utilité** |
| char[256] | accountName | Nom de compte de la personne qui envoie la demande |
| char | hasAccepted | Réponse à la demande |

* 1. **close\_call**

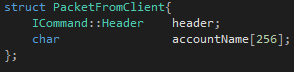
Utilisée pour mettre fin à un appel en cours.



|  |  |  |
| --- | --- | --- |
| **Type** | **Nom** | **Utilité** |
| char[256] | accountName | Nom de compte de la personne avec qui l’on est en communication |

* 1. **del**

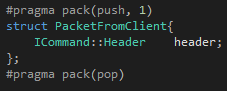
Utilisée pour supprimer un contact de sa liste de contact.



|  |  |  |
| --- | --- | --- |
| **Type** | **Nom** | **Utilité** |
| char[256] | accountName | Nom de compte de la personne que l’on souhaite supprimer de sa liste de contact |

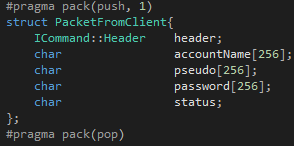
* 1. **exit**

Utilisée pour se déconnecter. Cette commande ne prend pas de paramètres.



* 1. **update**

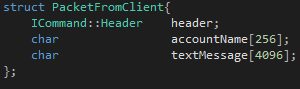
Utilisée pour mettre à jour les informations du client actuellement connecté.



|  |  |  |
| --- | --- | --- |
| **Type** | **Nom** | **Utilité** |
| char[256] | accountName | Nom de compte de la personne qui souhaite changer ses informations. Le nom de compte ne peut pas être modifié |
| char[256] | pseudo | nouveau pseudo |
| char[256] | password | nouveau mot de passe |
| char | status | nouveau status. Le status correspond à un enum définit plus bas dans cette documentation |

* 1. **send**

Cette commande est utilisée pour envoyer un message à un contact.



|  |  |  |
| --- | --- | --- |
| **Type** | **Nom** | **Utilité** |
| char[256] | accountName | Nom de compte de la personne à qui on souhaite envoyer un message |
| char[4096] | textMessage | Message à envoyer |

1. ***Cas d’utilisations et réponses attendues***
   1. Inscription

Client 1 envoie la commande “reg” au serveur.

Le serveur renvoie la commande “err” au Client 1.

* 1. Connexion

Client 1 envoie la commande “log” au serveur.

Le serveur renvoie la commande “err” au Client 1.

En cas de succès, le serveur envoie la commande “show” à tous les contacts de Client 1.

En cas de succès, le serveur renvoie une boucle de “show” à Client 1 pour lui lister ses contacts.

* 1. Ajout d’un contact

Client 1 envoie la commande “add” au serveur.

Le serveur renvoie “err” au Client 1.

En cas de succès, le serveur envoyé la commande “accept\_add” au Client 2.

* 1. Accepter une invitation de contact

Client 1 envoie la commande “accept\_add” au serveur.

Le serveur renvoie “err” au Client 1.

En cas de succès, le serveur envoie la commande “show” aux deux clients.

* 1. Supprimer un contact

Client 1 envoie la commande “del” au serveur.

Le serveur renvoie la commande “err” au client 1.

En cas de succès, le serveur envoie la commande “del” au 2 clients.

* 1. Appeler

Client 1 envoie la commande “call” au serveur.

Le serveur renvoie la commande “err” au client 1.

En cas des succès, le serveur envoie la commande “accept\_call” au client 2.

* 1. Accepter un appel

Client 1 envoie la commande “accept\_call” au serveur.

Le serveur renvoie la commande “err” au client 1.

En cas de succès, 2 possibilités sont possibles:

* client 1 a accepté: le serveur envoie la commande “accept\_call” aux deux clients
* client 1 a refusé: le serveur envoie la commande “accept\_call” à client 2.
  1. Raccrocher

Client 1 envoie la commande “close\_call” au serveur.

Le serveur renvoie la commande “err” à client 1.

En cas de succès, le serveur envoie la commande “close\_call” aux 2 clients.

* 1. Envoyer un message

Client 1 envoie la commande “send” au serveur.

Le serveur renvoie la commande “err” à client 1.

En cas de succès, le serveur envoie la commande “send” à client 2.

* 1. Se déconnecter

Client 1 envoie la commande “exit” au serveur.

Le serveur renvoie la commande “err” à client 1.

En cas de succès, le serveur envoie la commande “show” à tous les contacts de client 1.

* 1. Modifier ses informations

Client 1 envoie la commande “update” au serveur.

Le serveur renvoie la commande “err” à client 1.

En cas de succès, le serveur envoie la commande “show” à tous les contacts de client 1.

En cas de succès, le serveur envoie la commande “show” à client 1.

**Annexes**

1. ***Instructions codes***

|  |  |
| --- | --- |
| **Code** | **Instruction** |
| 0x01 | reg |
| 0x02 | log |
| 0x03 | list |
| 0x04 | show |
| 0x05 | add |
| 0x06 | accept\_add |
| 0x07 | del |
| 0x08 | exit |
| 0x09 | update |
| 0x0A | send |
| 0x0B | call |
| 0x0C | accept\_call |
| 0x0D | close\_call |
| 0x0E | err |

1. ***Errors codes***

|  |  |
| --- | --- |
| **Code** | **Erreur** |
| 0x00 | Everything is OK MODAFUCKER |
| 0x01 | The impossible happened |
| 0x02 | FAIL\_INIT\_SOCKET |
| 0x03 | FAIL\_INIT\_AUDIO |
| 0x04 | WRONG\_PACKET\_STRUCT |
| 0x05 | REGISTER\_ACC\_ALREADY\_USED |
| 0x06 | LOGIN\_WRONG\_PASSWORD |
| 0x07 | UNKNOWN\_ACCOUNT |
| 0x08 | ACTIONS\_TO\_OFFLINE\_ACCOUNT |
| 0x09 | CANNOT\_DO\_CALL\_MULTIPLE |
| 0x0A | CANNOT\_ACCEPT\_CALL\_FROM\_NOT\_A\_CALLER |
| 0x0B | CANNOT\_CLOSE\_CALL\_WHEN\_YOU\_ARENT\_CALLING |
| 0x0C | LOGIN\_ON\_ALREADY\_LOGGED\_ACCOUNT |
| 0x0D | CANNOT\_ADD\_YOURSELF |
| 0x0E | CANNOT\_CALL\_YOURSELF |
| 0x0F | YOU\_ARE\_NOT\_LOGGED |
| 0x10 | ALREADY\_CONNECTED |
| 0x11 | ALREADY\_IN\_YOUR\_CONTACT\_LIST |
| 0x12 | NOT\_IN\_YOUR\_CONTACT\_LIST |
| 0x13 | BUSY\_CONTACT\_CANNOT\_REPLY |
| 0x14 | NOT\_IN\_COMMUNICATION\_WITH\_HIM |
| 0x15 | CANNOT\_LOAD\_DATA |
| 0x16 | CANNOT\_SAVE\_DATA |
| 0x17 | YOU\_CANNOT\_UPDATE\_OTHER\_ACCOUNT |
| 0x18 | INVALID\_STATUS\_ID |

1. ***Statuts***

|  |  |
| --- | --- |
| **Code** | **Statuts** |
| 0x00 | Connecté |
| 0x01 | Déconnecté |
| 0x02 | Occupé |
| 0x03 | Absent |
| 0x04 | Kipour |
| 0x05 | Grasse mat’ 10h |
| 0x06 | Ramadan |
| 0x07 | Sport |
| 0x08 | Petit coin |
| 0x09 | YOLO |
| 0x0A | Parti bouder |