BRUIT Gabriel gbruit@etudiant.univ-mlv.fr ou gbruit@hotmail.fr
Neang Thomas thneang@gmail.com

Protocole pour Matou

Sommaire

1. Introduction

Nous présentons dans ce document comment le serveur et les clients échangent leurs informations pour réaliser le service de chat Matou.

2. Fonctionnement générale

Les échanges client/serveur dans ce protocole fonctionnent grâce à la structure générale suivante :

---

int	int	type de la donnée
id	taille des données(octet)	données

L'id à mettre et le type de donnée nécessaire seront détaillé plus bas dans la client et la partie serveur.
Il peut bien sûr y avoir plusieurs type de donnée lors d'un échange.
L'id permet de définir le traitement à adopter que ce soit pour le client ou pour le serveur. On peut ainsi savoir s'il s'agit d'une demande de connexion, de la réception d'un message, etc…
Un ByteBuffer de la bonne taille est alloué pour la réception des données.
Le type est donnée pour représenter le contenu, l'objet n'est pas envoyer dans ce format mai via un ByteBuffer.

Il y a au moins 3 sockets par client, une pour le serveur, une pour la connexion privé à un client et une pour les transferts de fichier.

Nous abordons maintenant les différentes demandes que les clients/serveurs peuvent demander et recevoir.

3. Connexion au serveur

id 0 : Demande de connexion au serveur

int	int	String
0	taille pseudo	pseudo en utf-8

Le client peut demander une connexion au serveur pour rejoindre le chat grâce à l'id 0. Il doit envoyer la taille de son pseudo encodé en UTF-8 puis son pseudo qui permettra au serveur de vérifier sa disponibilité.

id 1 : Acceptation du serveur

int
1

id 2 : Refus du serveur

int
2

Le serveur envoie 1 au client si le pseudo choisie est disponible, il envoie 2 sinon.

4. Connexion privé entre deux clients (c1 veut se connecter à c2)

id 3 : Demande connexion du client c1 au serveur

int	int	String	int	String
3	taille pseudo c1	pseudo c1 en utf-8	taille pseudo c2	pseudo c2 en utf-8

Le client doit envoyer son pseudo et celui de la personne à qui il veut se connecter.
Le pseudo c1 permet à c2 de savoir qui veut se connecter et le pseudo c2 permet au serveur d'envoyer la demande à la bonne personne.

id 4 : Demande de connexion au client c2 par le serveur.

|int | int | string |
|4 | taille pseudo c1 | pseudo c1 |

Le serveur demande ensuite à c2 s'il accepte la connexion de c1
id 5 : Acceptation par c2 de la demande du client c1.

| int | int | String | int | String| int |
|5 | taille pseudo c1 | pseudo c1 | taille addr | addres| port |

Le client c2 accepte la demande et envoie son adresse avec le numéro de son port pour permettre au client c1 de se connecter. On redonne le pseudo de c1 pour permettre au serveur de transmettre la réponse.

id 6 : Refus demande du client c1.

|int| int | String |
| 6 | taille pseudo c1 |pseudo c1|

On refuse la demande de connexion du client c1, même remarque pour le pseudo.

id 7 : Notification d'acceptation au client

|int| int | String | int |
|7 | taille address | address | port |

Si le client c2 accepte la demande de c1, le serveur transmet l'adresse et le port de c2 à c1.

id 8 : Notification de refus au client

|int|
| 8 |

Si le client c2 refuse la demande de c1, le serveur envoie l'entier 8 au client c1.
Attention lors d'une acceptation, il faut éviter le cas suivant :

3 4
client c1 --->serveur --->client c2
7 5
client c1 <---serveur <---client c2

```
            5               7
client c1---->serveur --->client c2 On doit s'arrêter ici, c'est au client de le gérer
              7             5
client c1  <---serveur <---client c2
            5               7
client c1---->serveur --->client c2
```

5. Connexion pour envoie de fichier

La connexion pour envoyer des fichiers dois se faire uniquement après la connexion des deux clients, la demande se fait donc sans passer par le serveur.

id 9 : Demande de connexion pour envoie de fichier

int	int	String	int
9	taille address c1	Address c1	port c1

id 10 : retour connexion pour de envoie fichier

int	int	String	int
10	taille address c2	Address c2	port c2

Nous avons donc le schéma suivant :

```
            3             4
client c1  ->serveur ->client c2
            7             5
client c1  <-serveur <-client c2
            5             7
client c1  ->serveur ->client c2
                  9
client c1  ---------->client c2
                  10
client c1  <----------client c2
```

6. Envoie de fichier

id 11 : Demande envoie de fichier. (Si c1 envoie a c2)

int	int	string			

|11 | taille nom_fichier | Nom fichier|||||

Le client c1 doit demander l'autorisation au client c2 avant d'envoyer un fichier.
Le nom du fichier est donné pour informer c2.

id 12 : Acceptation demande envoie de fichier

|int|
|12|

---

id 13 : Refus demande envoie de fichier

|int |
|13 |

Le client c2 peut accepter ou refuser avec l'id 12 ou 13.

id 14 : Envoie d'un fichier.

|int | int | byte |
|14 | taille_fichier | donnée |

L'envoie du fichier se fait en précisant la taille des données.
Le choix de la taille max est laissé au client.

7. Envoie de message

id 15 : Envoie d'un message.

|int | int | String | int | String |
|15 | taille pseudo |pseudo | taille message | Message|

Chaque client envoie son message avec son propre pseudo pour être identifié sur le chat.
Le message est envoyé au serveur qui doit envoyer le message à tous les autres clients.