< WALL-E2>

Protocole de communication

Version 2.0

Historique des révisions

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Auteur** |
| 2017-02-08 | 1.0 | Première version terminée | Abdellatif Amrani et Samir El Kaoukabi |
| 2017-02-09 | 1.1 | Modification de quelques exemples de messages échangés entre le client et le serveur | Abdellatif Amrani et Samir El Kaoukabi |
| 2017-02-10 | 1.2 | Vérification complète du document et correction des erreurs d’orthographe | Abdellatif Amrani et Samir El Kaoukabi |
| 2017-04-10 | 2.0 | Mise à jour reflétant les derniers changements apportés au protocole de communication. | Abdellatif Amrani et Samir El Kaoukabi |

Table des matières

1. Introduction 4

2. Communication client-serveur 4

3. Description des paquets 5

3.1. Gestion des connexions 6

3.2. Les configurations 6

3.3. Le clavardage 7

3.4. Le mode édition 9

3.5. Le mode simulation 11

Protocole de communication

# Introduction

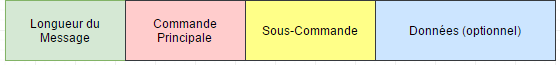
La première partie de ce document sera consacrée à la description générale du modèle de communication entre le serveur et les clients. La deuxième partie fournira des détails sur le contenu des différents paquets échangés au sein de ce protocole durant les principales fonctionnalités de notre logiciel : la gestion des connexions, les configurations, le clavardage, le mode édition et le mode simulation.

# Communication client-serveur

Après avoir analysé le problème et après l’expérimentation de plusieurs solutions sous forme de mini prototypes rapides, nous avons décidé d’utiliser des sockets TCP asynchrones, vu la fiabilité garantie du protocole TCP en comparaison avec d’autres protocoles comme UDP. En effet, TCP inclut des mécanismes permettant de réassembler les paquets correspondant aux données envoyées dans le bon ordre grâce aux numéros de séquence. Il s’assure aussi de retransmettre tout paquet perdu ou endommagé grâce à ce même mécanisme, ce qui est très important pour notre produit. Nous avons choisi d’utiliser des sockets asynchrones vu qu’ils permettent des opérations non bloquantes, ce qui permettra la connexion de plusieurs clients au serveur sans problème, et ce, en utilisant un répartiteur (dispatcher) conçu soigneusement. Le serveur servira de relai. Les calculs se feront donc au niveau des clients et chaque changement relatif à un client sera relayé par le serveur aux autres clients connectés qui se serviront du message échangé pour mettre à jour leurs états en conséquence. Les échanges seront principalement des chaines de caractères structurées en respectant un format précis qui sera détaillé dans les parties suivantes. À noter qu’il y aura une structure de messages des clients vers le serveur et une autre structure similaire pour les échanges partant du serveur vers les clients. Le serveur détiendra tous les détails relatifs aux clients (profils des utilisateurs, détails des cartes…) et il sera bien entendu capable de distinguer les clients lourds et le client léger.

# Description des paquets

En général, les messages échangés seront sous la forme suivante:



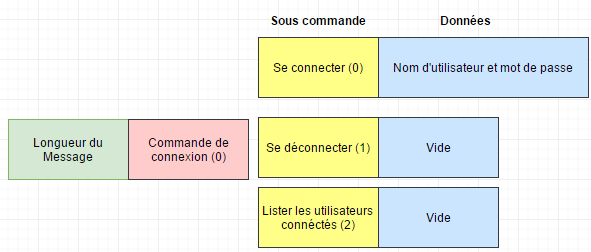
Ce formatage des messages constitue un petit risque lorsqu'un composant du message lui-même contient le séparateur (; dans notre cas). Nous avons réussi à contourner ce problème en optant pour un encodage/décodage qui élimine tous les risques.

Les codes principaux étant : **0** pour la **gestion des connexions**, **1** pour les commandes de **configuration**, **2** pour le **clavardage**, **3** pour les opérations de **l’édition en ligne** et **4** pour **la simulation en ligne**. Tous les paquets utilisés par notre logiciel seront détaillés dans ce qui suit.

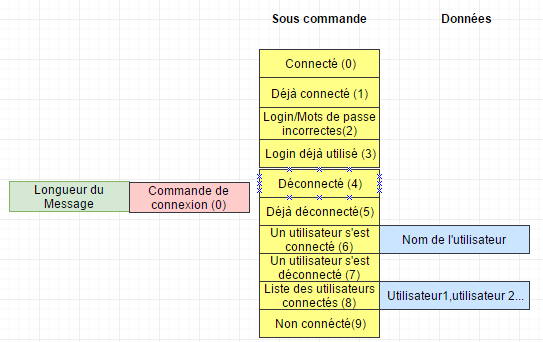
# 3.1. Gestion des connexions

Voici les différents paquets utilisés pour la gestion des connexions :

Client vers serveur :



Serveur vers client :

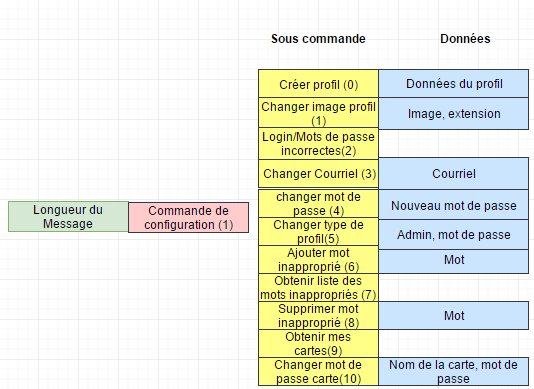


# 

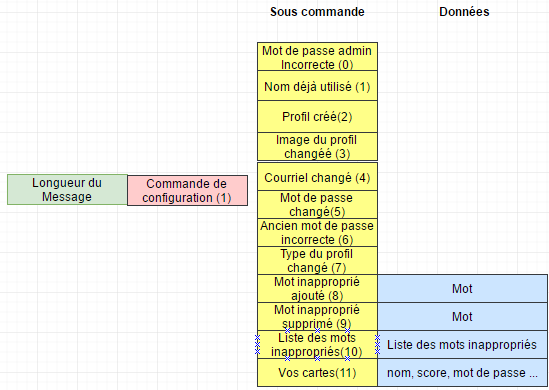
# 3.3. Les configurations

Voici les différents paquets utilisés pour les configurations.

Client-Serveur :



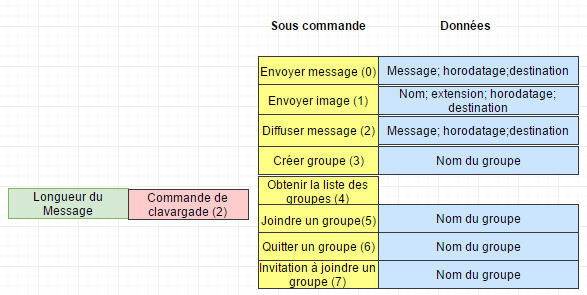
Serveur-client :



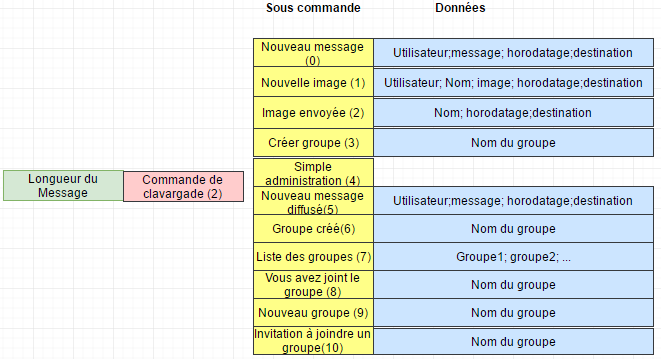
# 3.3. Le clavardage

Voici les différents paquets utilisés pour le clavardage.

Client-Serveur :



Serveur vers client :



# 

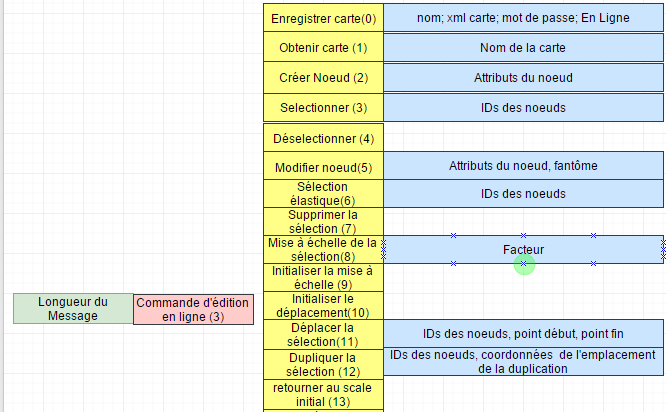
# 3.4. Le mode édition

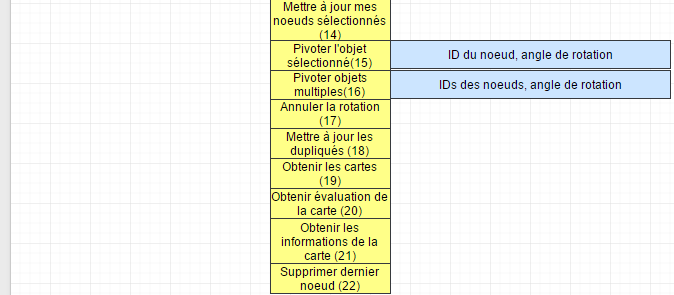
Le mode d’édition en ligne permet à plusieurs clients connectés d’éditer simultanément une carte. Le serveur détient en tout temps les détails les plus à jour relatifs à toutes les cartes et se charge de répercuter tout changement sur une carte à tous les clients en temps réel. Voici un exemple illustrant l’édition en ligne d’une carte :

Lorsqu’un client lourd effectue un changement sur un objet dans une carte (déplacement par exemple), au fur et à mesure que les nouvelles coordonnées de l’objet en question changent, des messages indiquant ces changements sont envoyés au serveur. Ce dernier met à jour les informations de la carte concernée. Ainsi, tout autre utilisateur qui édite la même carte est en mesure de voir les changements effectués par les autres utilisateurs en temps presque réel par rapport à l’œil humain.

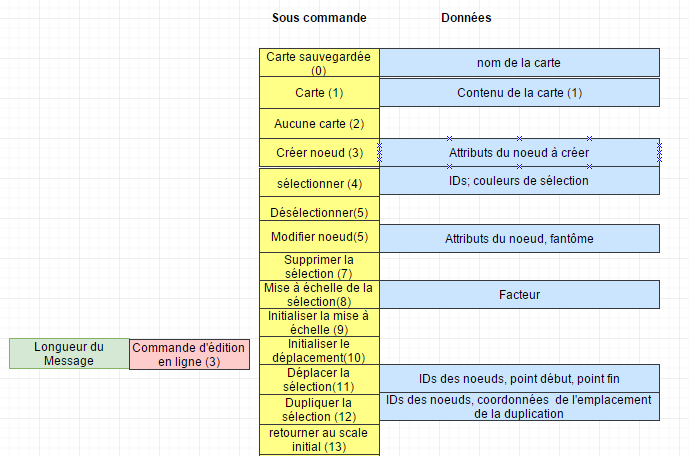
Voici les différents paquets utilisés pour l’édition en ligne.

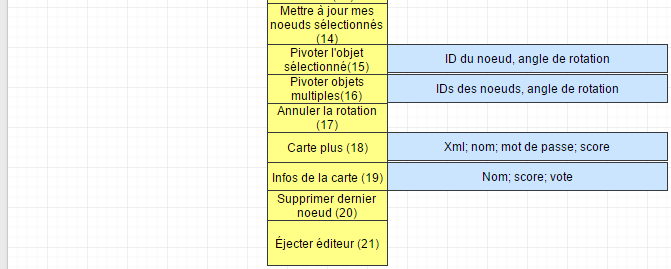
Client-Serveur :





Serveur vers client :





# 

# 3.5. Le mode simulation

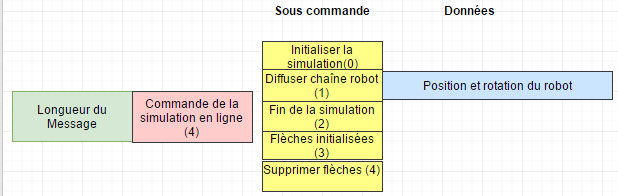
# 

Pour la simulation en ligne, nous avons mis à jour l’approche utilisée, et donc les paquets échangés en utilisant cette fonctionnalité ont aussi changé.

Lorsqu’un éditeur (pas un spectateur) d’une certaine carte lance la simulation, tous les éditeurs de cette carte vont participer automatiquement à cette simulation. Le serveur se charge d’attribuer des positions différentes pour les flèches de départ afin que les robots partent de différentes positions sur la table. Chaque utilisateur contrôle son robot et envoie périodiquement au serveur la position et la rotation de son robot, pourvue ce dernier puisse la diffuser aux autres utilisateurs.

Voici les différents paquets utilisés pour la simulation en ligne.

Client vers Serveur :



Serveur vers client :

