GoBox

# Projet L3-MCS

CANO Oliver

CHAPELET Adrien

Ce compte rendu a pour but d’aider à la prise en main de notre projet, mais aussi d’éclairer sur la logique de notre application et son fonctionnement. Vous trouverez également les problèmes que nous avons rencontrés, et les perspectives d’évolution d’un projet tel que GoBox.

# Guide d’utilisation

Il est possible d’importer le projet à partir de <https://github.com/adrien3d/gobox>

Ou si go est installé sur votre machine en utilisant la commande :

go get github.com/adrien3d/gobox

NB : Ne pas oublier de laisser un dossier « gobox » dans le même dossier que l’exécutable (côté client ET côté serveur).

## Serveur

Compiler ou utiliser l’exécutable « server\_linux » sur une machine Linux/Mac. Puis lancer simplement l’exécutable.

## Client

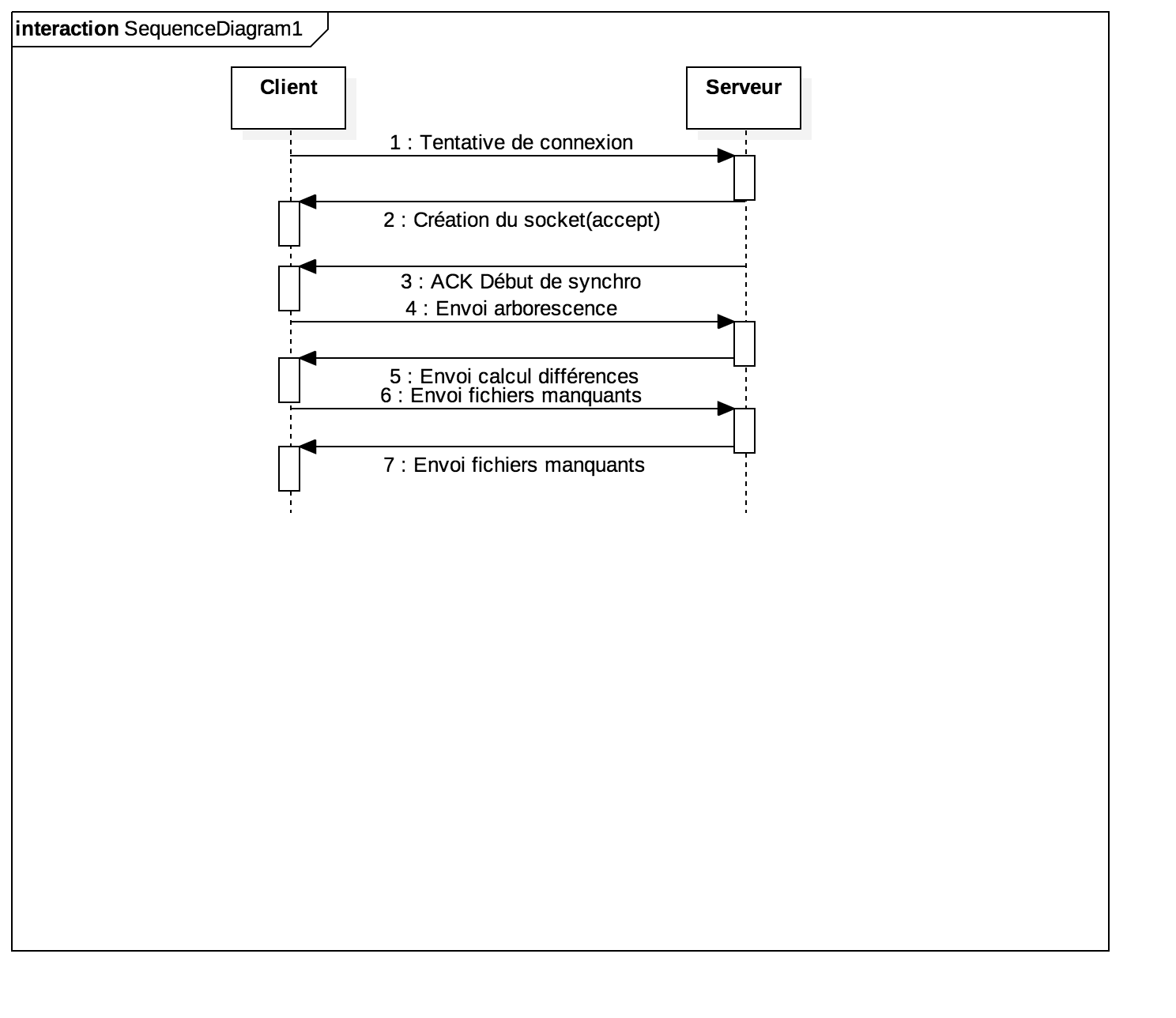
Le mieux est de recompiler le client sur une machine Linux/Mac, sinon le client présent dans le dossier « client\_linux » se connecte sur un serveur local ( 127.0.0.1 ).

Attention : L’application ne fonctionne pas s’il y a un proxy entre le client et le serveur, comme celui de l'IG2I (ou Centrale Lille).

# Scenarii

Le serveur se lance sur la machine occupant le rôle de serveur (typiquement connectée à internet). Le serveur va alors être en attente d’une demande de synchronisation.

Le client lance une demande de synchronisation, puis lorsque le serveur est prêt l’échange se déroule de manière asynchrone comme le montre le diagramme de séquence ci-dessous :



Dans l’ordre :

Client : Tentative de connexion avec le serveur

Serveur : Création du socket

Serveur : Envoi d'un acknoledgment "Début de synchronisation"

Client : Envoi de l'arborescence

Serveur : Envoi du calcul des différences

Client : Envoi des fichiers manquants sur le serveur

Serveur : Envoi des fichiers manquants sur le client

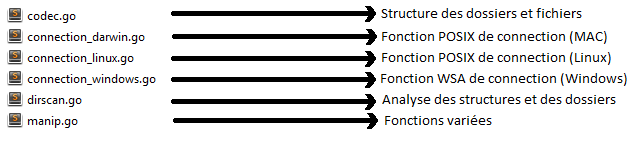
Fin de la synchronisation

# Organisation du code

Nous avons répartis le code en différents packages :

* Package « client » : le main du client.
* Package « server » : le main du serveur.
* Package « util » : toutes les fonctions utilisées sur serveur et client.

Ce dernier package est le plus fourni, il contient lui-même différents fichiers sources organisés de la manière suivante :



# Compatibilité

Le serveur et le client fonctionnent sur toutes les versions Linux et Mac. Malgré nos efforts d’utiliser l’API de socket Windows (WSA), nous n’avons pas réussi à résoudre un souci de réception de donnée sur le client Windows.

**Cependant la création de socket et l’envoi de paquets fonctionnent parfaitement sous Windows (voir connection\_windows.go).**

# Conclusion

Réaliser GoBox nous a permis de réaliser à quel point la conception est importante avant la réalisation d’un middleware. D’autant plus que notre application sans la partie réseau demandait déjà une réflexion importante sur l’architecture du logiciel.

C’est pourquoi si c’était à refaire, nous le referions plus modulable ; une base de données permettrait une gestion multi-utilisateur ainsi qu’une optimisation des échanges de fichier.

Pour ce qui est du format protocolaire, le schéma actuel est assez correct car il évite une surcharge inutile du serveur.

A noter tout de même notre déception de ne pas avoir réussi à utiliser la « Windows Socket API » (WSA) malgré notre nombre d’heure acharné à tenter de l’utiliser. Cette API née en 1992 est très similaire à POSIX (1988). A quelques bugs résolus prêt notre application aurait pu compiler sous Windows pour le client ET le serveur.