COMPTE RENDU
L3 – SR SERVEUR FTP
FROLOV – DO

1. Tâches implémentés/organisation de code

Pendant le temps donné, nous avons fini: **Étape I**, **Étape II**, **Étape III**, **Étape IV** (question 7 uniquement).

Organisation du code écrit:

Le répertoire principal contient :

— **README.md** détaillé expliquant tout : exécution, modules, protocole, etc. (écrit en anglais)

— **Makefile** et le répertoire */scripts*, contenant des scripts pour le makefile (*voir p.b. sur makefile et lancement*).

— **Répertoires**:

/Files – fichiers utilisés dans les tests qui sont copiés sur les serveurs

/Clients – repertoire côté client

/Clusters – contient des répertoires pour chacun des 4 esclaves avec ces ports (*voir p.b. pour l'explication de la structure*)

/Master – repertoire pour le maître, contient initialement un fichier json indiquant les esclaves avec leurs ports et leurs noms.

— **Fichiers source**:

– module ***csapp.c/.h*** – fourni

– module ***utils.c/.h*** – fonctions utilitaires

– module ***cJSON.c/.h*** – parser de fichiers JSON, utilisé dans les scripts pour makefile pour copier le contenu dans chaque cluster à partir du répertoire ***/Files***.

– **ftpmaster.c** – programme du serveur maître

– **ftpserver.c** – programme pour chaque cluster à exécuter, possède des fonctions de protocole (***cd, ftp***), définit des handlers.

– **ftpclient.c** – programme client, récupère la commande de l'utilisateur, communique avec le serveur via un protocole (*voir p.b. pour l'explication du protocole*).

2. Structure du système

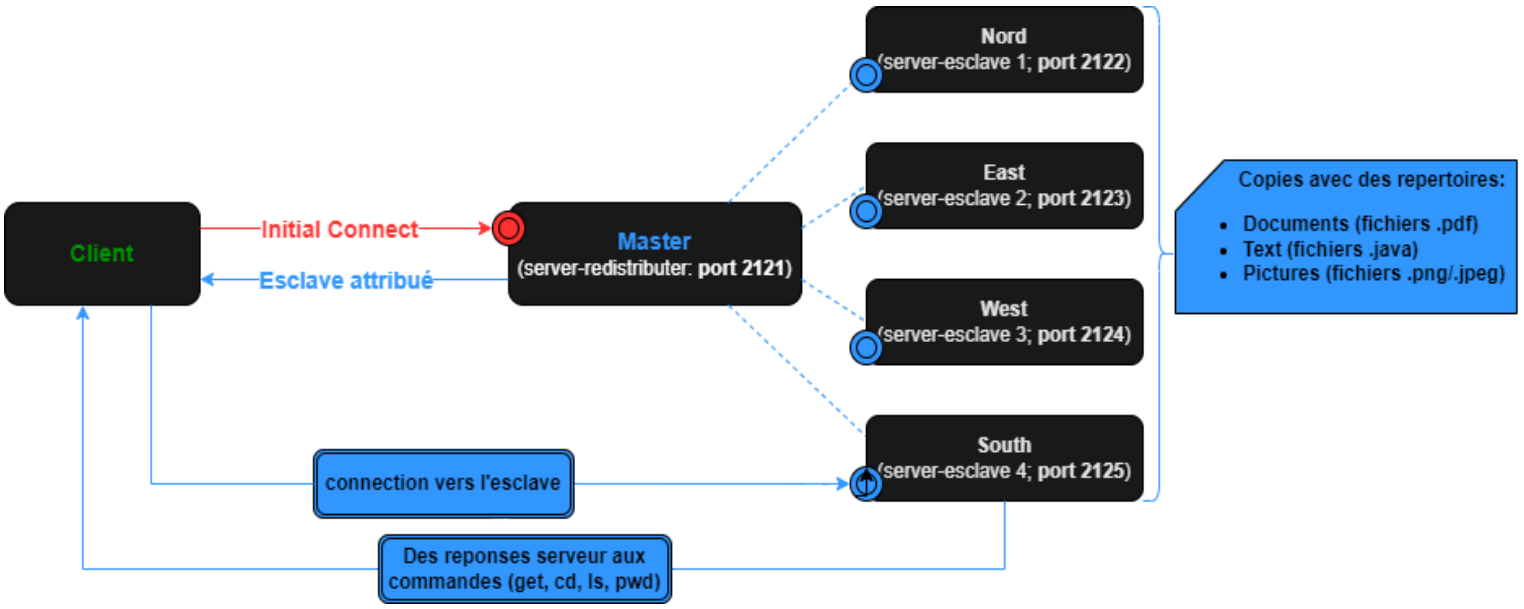

Schéma de relation client - server

Explication:

Le client se connecte d'abord au port **2121** (port maître) afin de pouvoir être distribué au cluster. Maître fait un "*turngate*", essayant de redistribuer les clients (s'ils sont multiples) vers différentes instances du serveur. Une fois que le client a attribué un serveur, la connexion au maître est fermée et le client se connecte directement à un cluster où le client peut désormais effectuer des requêtes de fichiers (***get***), parcourir des répertoires (***cd, pwd***), etc.

Principalement il y a trois parties:

— **ftpmaster:**

Serveur-maître sert uniquement pour assigner aux clients les serveurs-esclaves (clusters).

— **ftpserver:**

Chaque cluster exécute le **ftpserver** qui est le programme serveur principal. Il crée d'abord un pool (**NB_PROC** = 2, donc 2 forks). Après cela, il boucle sur les requêtes du client, lit la requête, gère les erreurs et essaie d'exécuter des commandes.

Protocol pour la requête ***get:***

– **line 1:** Message "**Successful request**"

– **line 2:** Taille fichier en octets

– **line 3:** Contenu (envoyé par paquets de **MAXLINE** = 8192)

– **line 4:** Message “**End**”

Protocol pour les commandes (***pwd, cd, ls*):**

– **line 1:** Message “**Successful command**”

– **line 2:** Sortie de la commande

— ftpclient:

Côté client, comme indiqué dans le schéma, se connecte d'abord au port maître, obtient le port du cluster du maître et s'y connecte. Après une connexion réussie, il envoie la requête de l'utilisateur au serveur et analyse la réponse en lisant la première ligne, de la manière suivante:

1. Réponse vide = crash serveur, termine la connexion.
2. Message “**Goodbye**!” = requête ***bye***, termine la connexion.
3. Message “**Successful command**” = protocole des commandes.
4. Message “**Successful request**” = protocole ***get***. Ainsi, le client lit la taille puis les octets des donnés. On time le transfert et vérifions le nombre d'octets transférés pour regarder l'intégrité.

3. Make et lancement

Pour mieux organiser le travail en séparant les fichiers, nous avons écrit des scripts shell pour utiliser dans le makefile (répertoire ***/scripts***).

Donc, pour lancer notre programme vous pouvez:

— Fait en premier: **make** (ça crée des exécutables et copies des clusters)

— Pour lancer le server en arrière plan: **make start_server**

— Dans autre terminal pour lancer le client: **make client**

— Dans le terminal ou vous avez le server pour terminer: **make end_server**

— Pour nettoyer tous: **make clean**

Exemple d'exécution:

```
maxim@frolovasus:~/S6/SR/FTP/FTP$ make client
Lauching a client!
scripts/open_client.sh
Connected to FTP Master Server
Redirected to Cluster East
ftp> ls
Documents Pictures Text ftpserver log.txt
ftp> cd Documents
Changed to directory: /Documents
ftp> ls
cours1.pdf cours2.pdf cours3.pdf cours4.pdf
ftp> get cours1.pdf
Successful request
Transfer successfully completed.
41474 bytes received in 0.000279 seconds (148652.33 KBs/s).
ftp> bye
Goodbye!
```

4. Tests

Nous avons écrit quelques petits tests pour vérifier la navigation et les téléchargements depuis le serveur. Pour lancer le test ayant un serveur déjà tournant, fait:

make client < tests/file.test

Fichiers ***.test*** disponibles :

– ***commands.test*** – se déplace sur le serveur en utilisant ***'cd'***, imprime les informations en utilisant ***'ls'*** et ***'pwd'***

– ***documents.test*** – Télécharge tous les fichiers (***.pdf***) depuis le répertoire ***/Documents***.

– ***pictures.test*** – télécharge toutes les images du répertoire ***/Pictures***.

– ***text.test*** – télécharge tous les fichiers texte (.java) à partir du répertoire ***/Text***

– ***incorrect_requests.test*** – fait quelques requêtes incorrectes à gérer par le serveur