Travail pratique #2 - IFT-2245

Kevin Belisle et Gabriel Lemyre

30 mars 2018

Le but de ce travail était d'implémenter l'algorithme du banquier sous le langage C en utilisant une architecture à plusieurs fils clients et plusieurs fils serveurs.

Tout d'abord, afin d'effectuer la connexion, nous avons utilisé la référence offerte sur le site du cours par les assistants afin d'effectuer nos connexions. Nous séparons nos connexion en trois catégories : la connexion d'initialisation, les connexions des fils et la connexion de terminaison. En premier lieu, la connexion d'initialiser sert à effectuer les requêtes BEG et PRO servant à initialiser le serveur pour les requêtes des fils allant suivre. Les fils générés ensuite servent à initialiser les ressources puis à effectuer les requêtes. Nous étions ensuite confrontés au problème de terminaison. Nous étions ici pris avec un problème au niveau de (pthread join) qui ne fonctionnait pas. Afin de rejoindre tous les fils lancés avec celui étant principal, nous avons dû retirer (pthread detach) du code qui nous était fourni sans quoi le programme se terminait automatiquement.

Pour le multi-fil, l'architecture était presque totalement implémentée dans le code fourni. Il ne nous restait simplement qu'à effectuer les connexions afin de bénéficier de plusieurs fils. Il nous fallait toutefois commencer à penser à l'architecture que nous devions développer avec plusieurs fils. C'est ici que nous avons commencer à avoir des problèmes majeurs qui se sont malheureusement éternisés. Parla faute d'un problème probablement dû à la version utilisée Cygwin-64. Nous avons perdu des dizaines d'heures de travail à tenter de comprendre plus en détails comment effectuer certains éléments qui ne semblaient pas fonctionner, mais qui en réalité, fonctionnaient à merveille sous UBUNTU. Nous n'arrivions pas à établir la cause des problèmes. Certaines fois, le code ne fonctionnait pas du tout et d'autres à merveille, parfois partiellement. Nous pensions qu'il s'agissait d'un problème de mauvaise compréhension de notre part alors qu'il s'agissait d'un problème externe au code. Cela nous a fait perdre plus d'une semaine à tenter de tout régler et à comprendre individuellement chaque élément sans succès.

À ce qui à trait aux conditions de course, nous avons séparer les différentes variables globales utilisées sous le serveur par des mutex. Ainsi, ces différentes valeurs seront toujours considérées individuellement sans problème de lectures multiples.

Nous avons aussi optés de créer une structure et un enum pour le protocole de communication afin d'uniformiser les requêtes les plus possible pour faciliter la maintenance dans le cas de la réutilisation de ce code lors d'un prochain travail. L'utilisation d'un enum en particulier nous permet d'avoir moins de bouts de code à modifier afin dans le cas d'un changement de protocole et la structure nous permet de donner et recevoir des informations à la fonction de transmission uniformément en nécessitant une moins grande quantité de fonctions et variables différentes. Puisque le protocole est partagé par une entête utilisée comme interface, minimiser les différences était un objectif en soi.

Pour les requêtes REQ, nous bloquons l'accès au resources dès que l'analyse et validation de la requête est terminée. Par la suite, nous bloquons l'accès au resources pendant que le banquier s'exécute pour éviter une réponse incohérente de l'algorithme du banquier. Pour les requête INI et CLO, nous bloquons les resources pour s'assurer que le banquier s'exécute sur le même nombre de clients. Une fois que la validation et nous nous sommes assurés que toutes les resources du client ont été libérées, nous enlèvons le client du thread serveur et nous fermons la connexion. Au moment du BEG, on assigne le nombre de resources. Ce n'est qu'au moment du PRO que nous l'instancions les tableaux associé aux resources maximum et aux resources alloués du serveur. Finalement pour la requête END, nous vérifions que tous les clients sont déconnectés et que toutes les resources sont retournées au serveur, si c'est le cas, nous fermons le server et arrête d'accepter des connexions.