École Polytechnique de Montréal

Département de génie informatique et génie logiciel

INF3405

Réseaux informatiques

Automne 2018

Projet en réseaux informatiques

Gestionnaire de fichier

Soumis par :

Son-Thang Pham (1856338)

Gabriel Côté-Jones (1771119)

Soumis à :

Dion-Paquin Émilie

Section (04)

26 octobre 2018

Introduction

Nous avons comme objectif pour ce premier laboratoire de développer notre propre application client-serveur permettant principalement de déposer un fichier sur un serveur de stockage ou bien de pouvoir récupérer un fichier. En nous assurant que le fonctionnement de notre application client-serveur et les requis fonctionnels sont nos priorités principales, nous aurons à créer deux produits, un serveur et un client qui pourront effectuer des échanges entre eux en utilisant les sockets. Ils permettront de gérer des flux entrant et sortant afin d'assurer une communication entre les deux produits à l'aide du protocole TCP/IP. Nous utiliserons aussi les notions de threads pour pouvoir accepter plusieurs clients dans ce même serveur. Nous vous décrirons premièrement la solution implémentée, nous poursuivrons avec les difficultés rencontrées et nous finirons avec notre critique du laboratoire avant de conclure.

Présentation

Client-Serveur

Le principe de notre client-serveur est le suivant :

1. Le client entre une commande et l'envoie au serveur par le socket.
2. Le serveur vérifie la commande et envoie un message de confirmation.
3. Dépendamment de la confirmation, le client et le serveur lancent l'action nécessaire ou affichent un message d'erreur.

Par souci de simplicité d'implémentation, les vérifications des commandes de téléversement sont faites du côté client avant d'envoyer quoi que ce soit au serveur.

Contrôle de l'adresse IP et du port

Nous utilisons la librairie Java Swing pour entrer les informations adresse IP et port au démarrage. Cela permet au serveur et à l'utilisateur d'entrer les informations nécessaires pour communiquer avec le serveur: l'adresse IP et le port. Pour vérifier la validité l'adresse IP, nous utilisons une formule regex. Pour le port, nous faisons simplement vérification booléenne pour déterminer si l'entrée est bien entre 5000 et 5050. L'interface pourra établir une connexion seulement lorsque les deux champs désirés sont valides, sinon elle retourne un message d'erreur.

Affichage des demandes traitées

Lorsque la connexion s'établit, le serveur attend les demandes en « String » provenant du client. Nous pouvons avoir des commandes avec une ou deux parties. Les commandes possibles sont cd (argument), ls, mkdir (argument), upload (argument), download (argument) et exit. L'affichage dans le serveur est alors possible en récupérant l'adresse IP, le port, la date et la commande. Nous pouvons récupérer l'adresse IP à l'aide de la fonction dans le socket: « getInetAddress().getHostAddress() ». Nous récupérerons le port avec la fonction « getPort() » de la même manière. Pour la date, nous utilisons « LocalDateTime.now() » pour récupérer la date d'aujourd'hui et nous la convertissons dans le format désiré.

Ensuite, la commande est séparée en deux lorsqu'il est composé de deux parties afin de pouvoir récupérer l'argument qui sera nécessaire pour effectuer un processus de navigation ou de création.

Navigation, présentation et création à partir du client

Lorsque la connexion s'établit, le client commence dans le répertoire « currentPath » qui est « ./root » au départ dans le serveur. Ce « currentPath » nous permet de déterminer l'endroit où nous nous situons dans le serveur. À partir de ce chemin, nous pouvons naviguer, y créer des répertoires, téléverser ou télécharger des fichiers et afficher les éléments contenus dedans.

Pour naviguer dans celui-ci, il faut que l'utilisateur entre la commande appropriée composée de deux parties : cd « argument ». La partie « argument » correspond au répertoire désiré. Le serveur effectuera une mise à jour du « currentPath » en ajoutant « / » et l'argument si ce chemin existe dans le « root ». Dans le cas d'un argument valant « .. », il y aura mise à jour du de la variable en coupant la dernière partie suivant le dernier « / » seulement s'il n'est pas situé au « root ».

La commande « ls » permet au serveur d'envoyer la liste d'items en tant que strings situés dans le « currentPath » au client à l'aide d'une boucle « for ». Pour déterminer s'il s'agit d'un fichier, nous utilisons la fonction « toFile().isFile() » disponible pour un « Path » .

La commande « mkdir (argument) » permet de créer un répertoire nommé « argument » dans le « currentPath » à l'aide la fonction « createDirectory() » qui prend comme argument le chemin incluant l'argument pour le créer dans le serveur.

Téléversement et téléchargement d'un fichier

Le téléversement se fait en créant un « stream » de lecture sur fichier du côté client et un « stream » d'écriture du côté serveur. Le client attend la confirmation de départ du serveur puis envoie à ce dernier la taille du fichier à téléverser et attend à nouveau une confirmation. Une fois confirmé, le client envoie le fichier octet par octet par le socket. Lorsqu'il a terminé, il attend à nouveau une confirmation du serveur. Les confirmations sont des constantes « String ». S'il y a une erreur dans le procédé, la constante ABORT est envoyée. Et le procédé est annulé pour le client et pour le serveur.

Le téléchargement est tout simplement l'inverse. C'est-à-dire, le client prend le rôle qu'avait le serveur lors du téléversement et vice-versa.

Threads

Les threads sont utilisés comme demandé dans l'énoncé du TP du côté serveur (plusieurs clients pour un serveur). Ils sont également utilisés du côté client. Tout dépendamment de la commande entrée, un thread approprié pour la réaction du client sera lancé puis fermé (une fois sa tâche accomplit). Par exemple le thread « download ».

Déconnexion

Du côté serveur, le thread du client roule tant que le booléen « isThreadRunning » est à « true ». Il suffit de le changer pour mettre fin au thread. Le socket approprié est également fermé.

Difficultés rencontrées

Les deux produits que nous avons créés ont une dépendance bidirectionnelle puisqu'ils ont une relation client-serveur et doivent absolument se communiquer entre eux. L'idée de séparer le travail en deux était une de nos options : une personne s'occupe du client et l'autre du serveur. La nature de ce programme fait en sorte que diviser le travail en deux était un peu plus dure que prévu en raison de l'interdépendance que nos produits possèdent. En effet, le serveur pouvait être fait en partie, mais sans le code du client avancé, il n'y a pas vraiment de moyen de vérifier si le code du serveur roule tel que prévu ou bien si nous avons laissé glisser des erreurs dans le serveur en observant les résultats du client. La même idée pourrait s'appliquer de l'autre sens dans cette relation. Il était alors important de bien communiquer en équipe comme le fait un client-serveur efficace et de travailler dans les deux produits en même temps le plus souvent possible.

Critiques et Améliorations

Le temps offert pour compléter ce laboratoire est sans aucun doute très généreux, mais elle l'est peut-être un peu trop. En effet, nous avons l'impression que ce travail aurait très bien pu être fait en moins de temps que celui offert. Évidemment, cela pourrait diverger d'une équipe à l'autre. Le fait de construire au complet les deux applications est une idée intéressante et nous offre beaucoup de liberté dans la manière de procédé. En revanche, en faisait le travail avec autant de liberté, il y aura sans aucun plusieurs solutions envisageables. Avec autant de chemins possibles, il aurait peut-être été utile de nous guider en nous offrant une piste plus précise à suivre ou bien en nous référant à plus de documentations.

Conclusion

Ce laboratoire nous a aidés à comprendre la manière qu'un serveur de stockage en ligne comme Dropbox ou Google Drive fonctionne en créant justement une version semblable, mais moins complexe tout en gardant l'essentiel du mécanisme de fonctionnement. Nous avons pu utiliser directement les notions apprises dans le cours telle que la programmation à l'aide du protocole TCP/IP. La notion des threads est un des sujets avec lequel nous étions un peu moins familiers. Nous avons dû rechercher par nous même les informations nécessaires pour mieux comprendre la théorie derrière celle-ci et de bien l'implémenter. Ce laboratoire a été une excellente introduction au cours en nous présentant certains principes qui régissent le fonctionnement général des réseaux informatiques et nous a permis d'avoir une meilleure compréhension de ceux-ci.