École Polytechnique de Montréal

INF3405
Réseaux informatiques

Hiver 2020 laboratoire 1

Soumis par
Roman Zhornytskiy (1899786) et
Hakim Payman (1938609)

Soumis à Esther Guerrier

14 février 2020

Introduction

Ayant reçu de belles appréciations de notre entourage sur un algorithme (trouvé sur Internet) pouvant appliquer un filtre de Sobel sur des photos, nous décidons de le rendre plus accessible en créant un client pouvant se connecter à notre serveur où le filtre Sobel y est hébergé pour pouvoir être utilisé. Ainsi, l'objectif du laboratoire est de nous familiariser aux échanges Client/Serveur à travers des sockets et au développement d'une « application réseau » employant des threads. Plus particulièrement, nous aurons à concevoir une application client-serveur permettant d'appliquer un filtre de Sobel sur des images envoyées depuis le client vers le serveur. De plus, le serveur devra pouvoir renvoyer l'image traitée au client afin que ce dernier puisse la sauvegarder sur son disque dur.

Présentation

L'application comporte 4 fichiers : Sobel.java (fournit avec l'énoncé), Generals.java, Client.java et Server.java.

Premièrement, Generals.java contient les déclarations des requêtes du client et des réponses du serveur disponible dans l'application.

Deuxièmement, Client.java contient la définition de la classe Client représentant le client se connectant au serveur. La classe Client permet à l'utilisateur de sélectionner une image, de l'envoyer au serveur puis de la sauvegarder dans le disque dur une fois le traitement du serveur terminé. Ainsi, elle a comme première responsabilité d'établir la connexion avec le serveur (*createSocket*). Puis, une fois la connexion établie, cette dernière est chargée d'acheminer les informations nécessaires à l'identification (*askUserCredentials*) de l'utilisateur et les requêtes de celui-ci (*processUserRequests*) vers le serveur par le biais d'une interface console. Le processus d'envoi d'une image, une fois le fichier image choisi et chargé dans le programme, se fait en deux temps : il débute par l'envoi de la taille de l'image vers le serveur puis se poursuit avec l'envoi de l'image vers le serveur sous la forme d'un tableau d'octets. Cela est effectué par les méthodes *prepareImage* et *sendImage*. La réception d'une image est effectuée par la méthode *receiveImage* où la taille de l'image à recevoir est d'abord lu puis l'image en provenance du serveur est lu au complet par la méthode *DataInputStream.readFully*. Cette méthode permet de lire l'image reçu au complet, c'est-à-dire, avec la taille de l'image lu.

Troisièmement, Server.java contient la définition de la classe Server. Cette classe a pour but d'initialiser le serveur et de gérer les requêtes de connexion des clients. La gestion des demandes de connexions des clients est faite en démarrant un thread pour chaque nouveau client souhaitant se connecter. C'est aussi cette classe qui s'occupe de lire la base de données local pour stocker les utilisateurs et leur mot de passe dans une map afin que le programme puisse facilement identifier les utilisateurs.

Finalement, la classe ClientHandler, définit dans la classe Server, contient tout le code servant à identifier le client et répondre aux requêtes du client. L'identification du client est faite par la méthode *setupUser* dans laquelle le nom utilisateur et le mot de passe sont utilisé pour valider l'identité du client avec les informations stockées sur tous les autres utilisateurs dans une base de données local. Dans le cas où le nom d'utilisateur est absent de la base de données, un nouvel utilisateur est alors ajouté avec le mot de passe fournit par l'utilisateur. Une fois l'identité de l'utilisateur validée, le client peut alors envoyer des requêtes au serveur qui seront gérées à l'intérieur d'une boucle dans la méthode *run* du thread associé à un client. L'envoi et la réception d'image suit le même processus que celui décrit dans la classe Client avec comme seule différence l'emploi du filtre de Sobel immédiatement après la réception de l'image.

Difficultés rencontrées

Une première difficulté rencontrée, bien que mineure, était la création de l'interface console qu'il nous fallût créer pour le Client. En effet, nous étions un peu laissés à nous même pour la concevoir sans avoir d'informations explicites sur comment les options devraient être présentées au client et comment ce dernier devrait pouvoir les choisir. Nous avons finalement opté pour une interface simple présentant les options sous forme de liste numérotée où l'utilisateur peut choisir une option en saisissant le chiffre qui lui est associé.

La dernière et la plus grande difficulté que nous avons rencontrée était le processus d'envoi et de réception d'image. En effet, au début du laboratoire, nous ne connaissions pas les outils que Java fournit pour gérer cette sorte d'opération et nous avions de la difficulté à déterminer quelle structure de données employer. Finalement, beaucoup de recherche dans la documentation et sur internet nous ont permis de trouver une façon de procéder qui répondait parfaitement aux requis du laboratoire, c'est-à-dire, l'emploi de *DataInputStream.readFully* et l'utilisation de la taille de l'image pour assurer une lecture complète de l'image. L'image lu était

stockée dans un tableau d'octets qui pouvait ensuite être reconstruit en un fichier image pour être traité.

Critiques et Améliorations

Pour les itérations futures de ce laboratoire, il serait bien d'avoir un exemple explicite de l'interface console voulue pour présenter les options à l'utilisateur afin de pouvoir mieux nous guider dans la conception de cette application.

Ensuite, il serait souhaitable d'avoir des pistes de solutions en ce qui a trait à la démarche à prendre pour le transfert de fichier du client vers le serveur et vice versa. En effet, cette partie du travail mérite d'être montré de la bonne manière au lieu de l'apprendre « sur le tas ».

Pour améliorer ce laboratoire, il serait intéressant d'explorer d'autres domaines d'applications des applications client/serveur autre que le transfert de fichier. Par exemple, il serait intéressant d'implémenter une application client/serveur permettant de faire communiquer plusieurs clients entre eux dans le cadre d'un simple petit jeu multijoueur (comme tic-tac-toe ou pong) ou d'un petit groupe de clavardage.

Conclusion

En conclusion, ce laboratoire nous a permis de comprendre comment construire une application client/serveur et comment effectuer des transferts de fichier du client vers le serveur et vice versa à l'aide des sockets. Ce travail nous a fortement enrichi au niveau de nos connaissances en réseautique et en manipulation de fichiers et nous servira de pilier de base pour démarrer nos prochains projets en lien avec la réseautique.