POLYTECHNIQUE MONTRÉAL

École Polytechnique de Montréal

Département Génie Informatique et Génie Logiciel

INF3405 – Réseaux Informatiques

TP1 : Projet en réseaux informatiques
Gestionnaire de fichier

1. Informations générales

Session	Hiver 2019
Public cible	Étudiants de 1er cycle
Date et Lieu de réalisation	À partir du 18 janvier au Laboratoire L-4708
Taille de l'équipe	2 étudiants
Pondération	15 %
Date de remise du projet	**Remise pour tous les groupes le 28 février (23h55 au plus tard)**
Directives particulières	✓ Tout rapport sera pénalisé de **5** points s'il est soumis par une équipe dont la taille est différente de deux (02) étudiants sans l'approbation préalable du chargé de laboratoire. ✓ Soumission par ***moodle*** uniquement (http://moodle.polymtl.ca). ✓ **Soumission d'une archive compressée (.zip / .rar) contenant le rapport (en format PDF), les projets contenant le client et le serveur et les exécutables.** ✓ Toute soumission de l'archive en retard est pénalisée à raison de 5 points par heure de retard.
Chargé de laboratoire	Groupe 01/02 - Émilie Dion-Paquin (emilie.dion-paquin@polymtl.ca)
Auteurs :	Mehdi Kadi, Bilal Itani, Émilie Dion-Paquin

2. Connaissances requises

- Sockets
- Threads
- Programmation **Java ou C/C++**

3. Objectifs du laboratoire

L'objectif de ce laboratoire est de familiariser l'étudiant :

- aux échanges Client/Serveur en utilisant les sockets;
- au développement « d'applications réseau » en utilisant les threads.

Ce travail pratique consiste, par la même occasion, à évaluer deux des 12 qualités de l'ingénieur définies par le BCAPG (Bureau canadien d'agrément des programmes de génie). Le Bureau d'agrément a pour

mandat d'attester que les futurs ingénieurs ont atteint ces 12 qualités à un niveau acceptable. Les deux qualités en question sont :

Qualité 4 (Conception) : capacité de concevoir des solutions à des problèmes d'ingénierie complexes et évolutifs et de concevoir des systèmes, des composants ou des processus qui répondent aux besoins spécifiés, tout en tenant compte des risques pour la santé et la sécurité publiques, des aspects législatifs et réglementaires, ainsi que des incidences économiques, environnementales, culturelle et sociales.

Qualité 7 (Communication) : habileté à communiquer efficacement des concepts d'ingénierie complexes, au sein de la profession et au public en général, notamment lire, rédiger, parler et écouter, comprendre et rédiger de façon efficace des rapports et de la documentation pour la conception, ainsi qu'à énoncer des directives claires et y donner suite.

4. Description

Mise en contexte

Vous en avez marre de ne plus avoir d'espace de stockage sur votre compte Dropbox ou Google Drive. À chaque fois que vous essayez de mettre un fichier sur le nuage, les géants de l'infonuagique essaient de vous soutirer de l'argent de vos petites poches pour vous vendre encore plus de stockage. La vie d'étudiant étant rude, et sachant que vous n'avez pas d'argent à donner à ces compagnies, vous choisissez de vous révolter. Étant étudiant à Polytechnique Montréal et surtout, expert en réseau informatique avant même d'avoir fini le cours, vous choisissez de développer votre propre application client-serveur permettant de stocker n'importe quel type de fichier sur un serveur de stockage. Sachant que vous n'êtes pas le seul étudiant dans cette situation, vous décidez de faire profiter vos ami(e)s de votre idée de génie. De plus, votre grand-mère, vous encourage dans votre quête et vous fait don de son bon vieux Pentium 3 que vous utilisez comme serveur de stockage.

Pour l'instant, vous prévoyez qu'une simple **interface console**, car vous voulez assurer uniquement le fonctionnement de votre application client-serveur et une interface graphique n'est pas une de vos priorités.

Fonctionnalités

Vous aurez à faire deux produits, un serveur, et un client pour communiquer avec le serveur. Votre serveur devra être capable de supporter la connexion de **plusieurs clients à la fois**, il faudra utiliser des *threads* au niveau du serveur pour réussir à supporter ces différents clients. Notez que l'application client et l'application serveur ne sont (en théorie) pas exécutées sur la même machine et que, bien évidemment, plusieurs clients peuvent tenter de se connecter au serveur en même temps. Cependant, chaque client connaît d'avance l'adresse IP du serveur ainsi que le numéro de port écouté.

Au démarrage du serveur, celui-ci demande à l'utilisateur d'entrer les informations suivantes : adresse IP du poste sur lequel s'exécute le serveur, port d'écoute (**un port entre 5000 et 5050 uniquement**). Une vérification doit être faite pour s'assurer que l'utilisateur a bien entré des données cohérentes. Par exemple, le serveur doit détecter que l'adresse 7x7.-202.666.888 et le port -9999 sont incohérents. Il devra aussi s'assurer que l'adresse IP entrée est bien sur quatre octets.

Au lancement du client, celui-ci demande à l'utilisateur d'entrer l'**adresse IP du serveur**, le port du serveur (**entre 5000 et 5050**). Le client doit vérifier la validité du format de l'adresse IP et du port entré par l'utilisateur (de la même façon que le fait le serveur) et afficher une erreur en cas d'invalidité. Ces informations sont utilisées pour tenter de se connecter au serveur.

Une fois la connexion établie entre le client et le serveur de stockage, le client doit s'identifier avec un nom d'utilisateur et un mot de passe pour accéder à son propre espace de stockage. Si le nom d'utilisateur n'est pas connu par le serveur, le client aura la possibilité de créer son mot de passe. Dans cette éventualité, l'espace de stockage sera créé avec le nom d'utilisateur. Autrement dit, les espaces de stockage des différents clients seront représentés par des dossiers (au niveau du serveur) que vous pouvez nommer avec le nom d'utilisateur du client. De cette façon, une fois l'authentification faite, le client doit se trouver dans son espace de stockage, sans possibilité d'accéder à un espace de stockage qui n'est pas le sien. Si l'utilisateur entre un mauvais mot de passe, il aura 3 tentatives pour y arriver sinon un message d'erreur doit être affiché au client et l'application doit se terminer tout simplement.

Astuce : Une stratégie simple pour sauvegarder les noms d'utilisateur et mot de passe est d'utiliser un fichier texte au niveau du serveur pour simuler une base de données. Vous pourrez y storer les paires d'informations avec la syntaxe de votre choix pour explorer facilement le contenu et trouver les informations. Cependant, vous êtes libre d'utiliser la méthode de votre choix pour stocker les informations d'identification, l'important est que les clients ne puissent avoir accès à ces informations évidemment.

Une fois l'authentification réussie, le client se trouve maintenant dans son propre espace de stockage et il peut, **à tout moment**, lancer une des cinq commande suivante :

Commande	Description
ls	Commande permettant d'afficher à l'utilisateur tous les dossiers et fichiers dans le répertoire courant de l'utilisateur au niveau du serveur.
upload <Nom du fichier>	Commande permettant le téléversement d'un fichier, se trouvant dans le répertoire local* du client, vers le serveur de stockage
download <Nom du fichier>	Commande permettant le téléchargement d'un fichier, se trouvant dans le répertoire courant de l'utilisateur au niveau du serveur de stockage, vers le répertoire local du client.
delete <Nom du fichier>	Commande permettant de supprimer le fichier de l'espace de stockage.
exit	Commande permettant au client de se déconnecter du serveur de stockage.

** Le répertoire local du client est le répertoire du projet (ou du répertoire contenant l'exécutable).*

Attention : Il n'est pas permis d'utiliser la librairie *Runtime* disponible dans Java (ou équivalent en C/C++) pour simuler le comportement des commandes de console (comme pour *ls*). Le but de ce laboratoire est d'implémenter et synchroniser les envois de trames sur le réseau entre le client et le serveur.

5. Spécifications

Affichage

Le serveur doit afficher dans sa console les informations suivantes quand il reçoit une commande : [Adresse IP client : Port local client - Date et Heure (min, sec)] : <Commande>

Exemple d'affichage au niveau du **serveur** pour les différentes commandes reçues.

```
[132.207.29.107:42975 - 2018-09-15@13:02:05]: ls
[132.207.29.107:42975 - 2018-09-15@13:02:01]: upload img.jpg
[132.207.29.122:35928 - 2018-09-15@13:02:08]: download fich.txt
```

Le client doit afficher dans sa console une différenciation entre les fichiers et les répertoires dans le cas de la commande *ls*. Aussi, le client doit afficher une confirmation pour chacune des autres commandes.

Exemples d'affichage au niveau du **client** pour les différentes commandes envoyées :

```
>> ls
[Folder] INF3405
[Folder] Autres
[File] img.jpg
[File] bye.pdf

>> upload img.jpg
Le fichier img.jpg a bien été téléversé.

>> download fich.txt
Le fichier fich.txt a bien été téléchargé.

>> delete bye.pdf
Le fichier bye.pdf a bien été supprimé.

>> exit
Vous avez été déconnecté avec succès.
```

Correction

Afin de pouvoir valider l'intégrité de l'envoi et de la réception de vos fichiers, servez-vous de l'outil certUtil disponible **à partir de la console**. Lors de la correction du laboratoire, nous effectuerons aussi cette vérification à l'aide de cet outil. Pour vérifier le hash d'un fichier, il suffit de lancer la commande comme suit :

Pour *Windows* : certUtil -hashfile <chemin/vers/le/fichier> <algorithme de hashage>

Pour *MacOS* : <algorithme de hashage> <chemin/vers/le/fichier>

Nous utiliserons l'algorithme de hashage MD5 pour effectuer nos tests sur l'intégrité d'envoi d'un fichier. L'idée est de s'assurer que le hash du fichier au niveau du client correspond au hash du fichier au niveau du serveur de stockage. Le cas échéant, l'envoi du fichier vers le serveur a été effectué avec succès.

Votre client et votre serveur doivent être bien développés, comme si vous alliez le livrer à un de vos clients. Une erreur d'exécution entraînera une pénalité.

6. Résumé des requis fonctionnels

Qualité évaluée

4.3 Procéder à la conception

Critère d'évaluation : Intégrer les concepts de programmation en réseautique retenus au premier laboratoire en répondant aux besoins et en respectant les requis fonctionnels du projet courant.

1. Saisie et vérification des paramètres du serveur (uniquement format de l'adresse IP et intervalle du port).
2. Saisie et vérification des paramètres du client (uniquement format de l'adresse IP et intervalle du port).
3. Saisie et vérification des informations d'identification du client.

   Pour un nouvel utilisateur : Création de l'espace de stockage individuel au niveau du serveur.
4. Énumération de tous les répertoires et les fichiers au niveau de l'espace de stockage.
5. Téléversement et téléchargement un fichier.
6. Retrait d'un fichier.
7. Déconnexion adéquate client-serveur.
8. Affichage en temps réel des demandes à traiter (logs au niveau de la console serveur).

7. Livrable

1. Langages et bibliothèques autorisés

- Le client et le serveur doivent être développés en Java ou en C/C++.
- Usage permis de librairies externes comme JSON.simple ou GSON. Si vous utilisez ces librairies, veuillez justifier dans la section « Présentation » de votre rapport la raison de leur utilisation.
- Usage non-permis de librairies permettant de simuler une console (ex. *Runtime* en Java).

Tout non-respect de ces consignes entraînera la note de 0.

2. Soumission

Le livrable est une archive (ZIP ou RAR) ayant le format suivant :

INF3405_TP1_matriculeX_matriculeY où matriculeX > matriculeY

Votre archive contiendra les fichiers suivants :

- Les projets du client et du serveur incluant les fichiers sources (.java ou .cpp), autrement dit le dossier en entier contenant votre projet.
- **Les 2 fichiers exécutables, un pour le client et un pour le serveur (.jar pour Java et .exe ou .out pour C++).**
- Le rapport au format PDF.

****Assurez-vous que les livrables compilent et s'exécutent adéquatement sur les ordinateurs du laboratoire ! ****

3. Rapport

Qualité évaluée : 7.1 Lire et rédiger de la documentation

Critère d'évaluation : Rédiger un rapport technique documentant efficacement le travail d'ingénierie réalisé dans ce projet en utilisant différentes formes de langage (naturel, informatique, etc.)

Le rapport, d'une longueur maximale de 4 pages (excluant la page de présentation), doit comporter les éléments suivants :

- **Page présentation** qui doit contenir le nom ou le logo de l'école, le libellé et l'identifiant du cours, la session, le numéro et l'identification du projet, la date de remise, les matricules et noms des membres de l'équipe, la mention « Soumis à : **nom et prénom du chargé de laboratoire** ».
- **Introduction** en vos propres mots pour mettre en évidence le contexte et les objectifs du TP. Aussi, vous devez faire le lien entre votre application et la théorie du cours (i.e. le protocole de transport utilisé par l'application et justifications).
- **Présentation** de vos travaux. Une explication de votre solution mettant en lumière la prise en compte des principaux requis du système. Il ne s'agit pas d'une reformulation des requis, mais des moyens entrepris pour les réaliser (i.e. expliquer brièvement le comportement des méthodes principales et l'usage des librairies importantes à votre application). Si vous utilisez des configurations particulières des bibliothèques ou des projets, précisez-les également.
- **Difficultés rencontrées, critiques et améliorations** pour de l'élaboration du TP et les éventuelles solutions apportées. Il serait intéressant d'inclure vos suggestions pour améliorer le laboratoire.
- **Conclusion :** Expliquez en quoi ce laboratoire vous a été utile, ce que vous avez appris, si vos attentes ont été comblées, etc.

Faites toujours attention à la qualité du français dans vos rapports, cela peut entraîner des pénalités.

7. Évaluation

Évaluation de l'exécutable	**8**
Évaluation de l'implémentation : gestion adéquate des variables et de toute ressource (création, utilisation, **libération**), gestion des erreurs, logique de développement, **documentation du code**, etc.	**6**
Rapport	**6**
Total des points	**20**