POLYTECHNIQUE MONTRÉAL

# École Polytechnique de Montréal

## Département Génie Informatique et Génie Logiciel

## INF3405 – Réseaux Informatiques

**TP4 : Projet en réseaux informatiques**

# Système de clavardage interactif

## 1. Informations générales

| | |
|---|---|
| Session | Automne 2017 |
| Public cible | Étudiants de 1er cycle |
| Date et Lieu de réalisation | À partir du 24 octobre 2017 au Laboratoire L-4708 |
| Taille de l'équipe | 2 étudiants |
| Pondération | 10 % |
| Date de remise du projet | **Remise pour tous les groupes le 24 novembre 2017 (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.<br>✓ Soumission par ***moodle*** uniquement (http://moodle.polymtl.ca).<br>✓ **Soumission d'une archive compressée (.zip / .rar) contenant le rapport (en format PDF) et les solutions Visual Studio contenant les projets client et serveur avec leurs codes sources**<br>✓ Toute soumission de l'archive en retard est pénalisée à raison de 3 points par jour de retard. |
| Chargé de laboratoire | Groupe 01 - Bilal Itani (bilal.itani@polymtl.ca)<br>Groupe 02 - Mehdi Kadi (mehdi.kadi@polymtl.ca)<br>Groupe 03 - Aymen Djellal (aymen.djellal@polymtl.ca)<br>Groupe 04 - Jackie Phung(jackie.phung@polymtl.ca) |
| Auteurs : | Aymen Djellal, Bilal Itani, Mehdi Kadi, Jackie Phung |

## 2. Connaissances requises

- Sockets
- Threads
- Programmation **C++ uniquement**

## 3. Objectifs du laboratoire

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

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

## 4. Description

**Introduction :**

Avec les récents événements et révélations concernant notre sécurité digitale, et le manque de vie privée et d'intimité, que ce soit l'espionnage relatif aux différentes allégations d'Edward Snowdon, ou aux récentes fuites de la NSA et le FBI partagées sur WikiLeak. Ces problèmes devenant de plus en plus présents et inquiétants, vous décidez de créer une application de clavardage *OpenSource*, sécuritaire et qui protège votre vie privée, que vous voulez utiliser avec vos amis et collègues.

Un de vos amis se porte volontaire pour faire toutes les couches de sécurité, et un autre s'occupera de l'esthétique ainsi que de l'interface utilisateur.

Toutefois, aucun de vos amis n'a jamais utilisé de *socket* ou de *threads*, il est donc de votre tâche de vous occuper de ces couches-là.

De plus, un riche investisseur s'intéresse à votre projet, et il vous demande une PoC (preuve de concept) de votre système de clavardage. Pour l'instant, il demande qu'une **interface console simple** sans sécurité (ça se fera pour plus tard), mais il veut s'assurer que vous êtes capable de lui livrer un produit fonctionnel.

**Requis :**

Vous aurez à faire 2 produits, un serveur, et un client pour communiquer avec le serveur.

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 10000 et 10050 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 777.-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 10000 et 10050**), un nom d'utilisateur, et un mot de passe. Le client doit vérifier la validité du format de l'adresse IP et du port entrés 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. Le

serveur vérifie alors la correspondance nom d'utilisateur / mot de passe, et accepte la connexion si le nom d'utilisateur et le mot de passe est valide. Sinon, il rejette la connexion du client, et le client affiche « Erreur dans la saisie du mot de passe ». Si le nom d'utilisateur n'existe pas dans la base de données locale du serveur, le serveur va créer automatiquement un compte pour l'usager et y assigne le mot de passe. Si la connexion a été acceptée, le serveur connecte l'utilisateur à la salle de clavardage puis lui envoie les 15 derniers messages **(donc les 15 plus récents), ou moins s'il y en a moins que 15**. Le client peut ensuite envoyer et recevoir des messages (**taille ne dépassant pas les 200 caractères**) en temps réel avec les autres utilisateurs connectés au serveur.

Quand le serveur affiche un message de la part d'un client, il doit l'afficher dans le format suivant: [Nom d'utilisateur - Adresse IP : Port client - Date et Heure (min, sec)]: Message. Voir l'exemple d'affichage attendu plus bas.

Notez que le programme client et le programme serveur ne sont (en théorie) pas exécutés 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. Durant la session de clavardage, le serveur affiche et stocke dans un fichier les messages reçus, ainsi que les informations mentionnées précédemment: adresse IP, port utilisé (celui de l'application client) et réponse. Le mécanisme de stockage des données de la conversation est laissé à votre discrétion, tant que le stockage est fait adéquatement et sans qu'il n'y a d'incohérences. Lors de l'affichage des messages, on s'attend à une sortie comme suit:

[Utilisateur 1 - 132.207.29.107:46202 - 2017-10-13@13:02:01]: Salut Utilisateur 2 !
[Utilisateur 2 - 132.207.29.117:37608 - 2017-10-13@13:03:24]: Yo Utilisateur 1 !

Le serveur doit aussi pouvoir stocker la base de données (dans un fichier dans le format de votre choix (e.g .csv/.txt/.json/.xml ou autre...)) des correspondances nom d'utilisateur et mot de passe.

## 5. Requis fonctionnels

Les fonctionnalités attendues sont indiquées comme suit :

✓ **Serveur**

- Saisie des paramètres du serveur (adresse IP, port d'écoute entre 10000 et 10050)
- Pouvoir connecter des utilisateurs avec leurs mots de passe
- Pouvoir envoyer un historique des sessions de clavardage
- Pouvoir fermer le serveur puis le rouvrir, et avoir tous les profils usagers et messages disponibles (**écriture sur disque et pas uniquement sur RAM**)
- Recevoir les messages des clients
- Tenir un historique de toutes les messages.
- Tenir une base de données des usagers et leurs mots de passe
- Afficher en temps réel les messages
- Effectuer correctement la vérification nom d'utilisateur/ mot de passe
- Créer les comptes automatiquement s'ils n'existent pas

✓ **Client**

- Saisir au clavier l'adresse IP du serveur et le port sur lequel le clavardage se déroule.
- Vérifier la validité de l'adresse IP saisie (uniquement le format) et le numéro de port (entre 10000 et 10050)
- Se connecter au serveur.
- Réceptionner les messages ou l'erreur de mot de passe.
- Saisir une réponse (200 caractères maximum).
- Transmettre la réponse au serveur.
- Se déconnecter.

# 6. Livrable

## 1. Langages et bibliothèques autorisés

- Le client et le serveur doivent être développés uniquement en C/C++.
- Dans le cadre de ce travail pratique, la librairie de communication réseau obligatoire est WinSock 2.
- Usage permis de librairies externes comme Boost ou STL.

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_TP4_matriculeX_matriculeY où matriculeX > matriculeY

Votre archive contiendra les fichiers suivants :

- Les solutions Visual Studio contenant le projet client et le projet serveur incluant les fichiers sources (.hpp, .cpp, .h, …), autrement dit le dossier en entier contenant votre projet
- le rapport au format PDF
- **les fichiers exécutables (.exe)**

****Assurez-vous que les livrables compilent et s'exécutent adéquatement sur les ordinateurs de l'école!****

## 3. Rapport

Le rapport **d'une longueur maximale de 3 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.
- **Présentation** de vos travaux. Une explication de votre solution mettant en lumière la prise en compte des principaux requis du système. Si vous utilisez des configurations particulières des bibliothèques ou des projets, précisez-les également.
- **Difficultés rencontrées** lors de l'élaboration du TP et les éventuelles solutions apportées.
- **Critiques et Améliorations** : 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.

## 7. Évaluation

| | |
|---|---|
| Évaluation de l'exécutable | **6** |
| 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. | **8** |
| Rapport | **6** |
| **Total des points** | **20** |