TD M2102 - Architecture des réseaux
Programmation des Sockets TCP en C#
Projet Client POP3

Avant de commencer (rappel) :

L'archive **SocketsTCP-CSharp.7z** récupérée sur Moodle contient :

- les sources C# d'une application **Client/Serveur BigBen** (exemple du cours traité en langage C),
- les sources C# d'un squelette d'un **Client POP3** dont vous allez poursuivre le développement en Projet.

Le but du projet est de développer un Client POP3 en C# en mode console. Pour vous aider, un squelette de départ est déjà fourni.

Cependant, avant de commencer toute programmation, il est indispensable de bien connaitre le protocole POP3 : la syntaxe des requêtes que le client peut envoyer au serveur, et celle des réponses associées qui reviennent du serveur. C'est l'objectif de la première partie ci-dessous. C'est un rappel de la feuille de TD 1.2.

I) (Re)Découverte du protocole POP3 grâce à telnet

POP3 est un protocole pour la consultation d'une boîte à lettre distante : il est exploité par les lecteurs de courriers (Thunderbird, Outlook, Eudora, ...) et utilise le port **110**.
Nous allons dans cet exercice lire le courrier sur le serveur de **laposte.net** de l'utilisateur « **iutinfo** » :

Utilisateur : **iutinfo**
Mot de passe : **Iutinfo2020**

- 1. Lisez le courrier de l'utilisateur « **iutinfo** » en utilisant un client lecteur de courriers de votre choix configuré avec POP3 (le service POP3 de **laposte.net** est disponible sur le serveur **pop.laposte.net**) ou depuis le webmail de **laposte.net.**

- 2. Connectez-vous au service POP3 de **laposte.net** en essayant ceci **sous un interpréteur de commandes** :

```
telnet  pop.laposte.net  110
```

Il faut ensuite donner une ligne de commande, terminée par CR/LF (en pratique, un retour chariot suffit).

Voici quelques commandes du protocole POP3 :

Commande	Fonction
USER *utilisateur*	Nom du compte de l'utilisateur
PASS *mot de passe*	Mot de passe
STAT	Donne le nombre de messages non lus et leur taille totale
RETR *n*	Récupère le message numéro *n*
DELE *n*	Supprime le message numéro
LAST	Donne le numéro du dernier message auquel on a accédé
LIST *[n]*	Taille de *n*-ème message ou de tous les messages
RSET	Annule la suppression de tous les messages
TOP *k n*	Affiche les entêtes et *k* lignes du message numéro *n*
NOOP	Ne fait rien
QUIT	Termine la session POP3

- 3. Lisez le courrier de l'utilisateur « **iutinfo** ». Vous pouvez vous inspirer du scénario suivant qui, après une phase d'authentification, demande le transfert d'un message :

```
S>+OK QPOP (version 2.53) at boite.info.iuta.u-bordeaux.fr starting.
C>USER felix
S>+OK Password required for felix.
C>PASS xxxxxxxx
S>+OK felix has 2 messages (2828 octets).
C>STAT
S>+OK 2 2828
C>LIST
S>+OK 2 messages (2828 octets)
S>1 1823
S>2 1005
S>.
C>RETR 2
S>+OK 1005 octets
S>Return-Path: <felix@labri.fr>
S>Received: from smtprt16.wanadoo.fr (smtprt16.wanadoo.fr [193.252.19.183])
S>Status: RO
S>
S>coucou
S>pf
S>.
```

Remarques importantes sur le protocole POP3 :
Le client envoie des lignes de commande (ou requêtes) et le serveur POP3 répond par une ligne commençant par +OK ou -ERR. Tout dialogue commence par une phase d'authentification (nom et mot de passe). Après une phase d'authentification réussie, le client peut consulter sa boîte à lettres en émettant des requêtes permettant les opérations suivantes :

- Demande du nombre de messages,
- Demande de la taille de chaque message,
- Lecture d'un message

Toute réponse d'un serveur POP3 qui contient plusieurs lignes est terminée par une ligne contenant un caractère '.' tout seul en début de ligne.
Et « par malchance », si une ligne de données commence par un caractère '.', celui-ci est doublé. Ainsi, on ne pourra pas confondre une ligne de données qui ne contiendrait qu'un point (elle sera alors codée "..CR/LF") avec la ligne de fin (".CR/LF"). Pour plus d'informations, voir la RFC !

- 4. Vérifiez la remarque précédente en lisant avec telnet un message de l'utilisateur « **iutinfo** » qui contient des lignes avec un ou des '.' en début de ligne.

Remarques importantes sur la programmation d'un client POP3 :
Quel que soit le type de client, qu'il soit en mode console comme le vôtre ci-après, ou en mode graphique comme un webmail ou Thunderbird ou autre, le client ne doit pas montrer à l'utilisateur final derrière son écran les ajouts du protocole pour fonctionner.
Donc en particulier, lors de l'affichage de la liste des mails par exemple (LIST), ou du contenu d'un message (RETR), votre client ne devra pas afficher le caractère '.' final ou les caractères '.' doublés en début de ligne s'il y en a, vous devrez algorithmiquement faire en sorte de ne pas les afficher.

II) Programmation en C# d'un client POP3

Projet à rendre : l'archive du programme produit pendant les séances 5.1 et 5.2 sera à remettre **sur Moodle à la fin de la séance 5.2 le lundi 25 mai 2020**. Il fera l'objet d'une note de Projet.

Les sources du squelette de départ sont fournies au sein d'une solution Visual Studio en C# dans le répertoire SocketsTCP-CSharp/**ConsoleClientPOP3**.
L'application fournie est une ébauche d'un client pour POP3. Elle réalise la connexion au serveur, l'envoi du nom de l'utilisateur et de son mot de passe, et récupère les réponses du serveur.
Elle propose aussi un menu, au sein duquel, par la suite, vous rajouterez les fonctionnalités demandées. Pour exemple, les fonctionnalités « STAT » et « QUIT » sont déjà fournies.

1) Ouvrez la solution avec Visual Studio en cliquant sur **ConsoleClientPOP3.sln**.
Consultez les 3 fichiers sources :

- Preferences.cs : regroupe les valeurs paramètres du programme (nom du serveur POP3, port, nom et mot de passe utilisateur) – ***Pas de modification à effectuer ici***.
- Program.cs : c'est le point d'entrée du programme, c'est le même code que le client de BigBen (ou Echo) (sauf le cœur de la fonction Travail bien sûr) – ***Pas de modification à effectuer ici***.
- Communication.cs : regroupe toutes les fonctions nécessaires aux échanges avec le serveur et la gestion du menu et de l'ensemble des fonctionnalités. C'est bien sûr ce fichier que vous modifierez par la suite.

2) Exécutez cette application, testez-la, et familiarisez-vous avec le code source fourni.
En particulier, il est crucial de comprendre :

- l'option de « Debug » (variable booléenne VERBOSE), cela permet d'activer ou désactiver l'affichage « brut » des envois et des retours du serveur (c'est utilisé dans les fonctions fournies LireLigne et EcrireLigne),
- qu'il est donc essentiel d'utiliser, tout le temps et uniquement, ces deux fonctions LireLigne et EcrireLigne pour les lectures de ce qui revient du serveur et les envois vers le serveur,
- le découpage fonctionnel pour la gestion du menu et le traitement des fonctionnalités, pour pouvoir par la suite, rajouter simplement les nouvelles fonctionnalités,
- la manière dont sont gérés les premiers échanges avec le serveur : USER, PASS et la fonctionnalité STAT, sans ça, vous serez incapable d'écrire les autres fonctionnalités !

3) Rajoutez au programme les fonctionnalités suivantes :

(a) Afficher le *n*-ème message (demandez la valeur de *n* à l'utilisateur).
(b) Afficher uniquement le nom de l'expéditeur et le sujet du *n*-ème message.
(c) Afficher le nom de l'expéditeur et le sujet de tous les messages.

*Indices : pour manipuler les **string**, C# propose de nombreuses propriétés et méthodes comme Length, Substring(), Contains(), Split(), etc…, pensez à les utiliser !*

4) Imaginez des extensions des fonctionnalités précédentes, et de nouvelles fonctionnalités de votre choix.