

**TCSMP - Time-Cost Stamped Mail Protocol**



Par

Mohamad **CHALHOUB**

Abdallah **JAFFAL**

Sous-direction de

Dr. Yasser FADLALLAH

Contents

[1. Introduction 3](#_Toc48625006)

[2. Conception 3](#_Toc48625007)

[3. Structure 3](#_Toc48625008)

[3.1. Les Clients 3](#_Toc48625009)

[3.2. Les Serveurs 5](#_Toc48625010)

[3.2.1. TCSMP Client Handler 5](#_Toc48625011)

[3.2.2. TPOP Client Handler 5](#_Toc48625012)

# Introduction

L'objectif de ce projet est de définir un protocole, et sa RFC associée, qui permettrait de limiter l'envoie de mails indésirables. Pour cela on fait payer un coût à l'envoie du mail, le prix à payer est du temps CPU. L'expéditeur du mail doit résoudre un "puzzles" envoyés par le serveur de destination et lui renvoyer la bonne réponse pour avoir le droit de transmettre son email. La résolution d'un puzzle doit prendre quelques secondes par courrier électronique. Ainsi pour l'utilisateur courant, cela est transparent. Mais par contre cela pénalise très fortement un envoi massif de mails.

L'implantation de ce protocole est codée en java.

# Conception

Nous avons choisi de faire une extension similaire de la [RFC 5321](https://tools.ietf.org/html/rfc5321) pour le TCSMP et une extension similaire de la [RFC 1081](https://tools.ietf.org/html/rfc1081) plutôt qu'un protocole indépendant car notre protocole est très proche de SMTP et POP3.

Il s'agit alors de payer des frais à l'expéditeur de l'e-mail. Pour ce faire, il est logique de disposer d 'une solution qui ne coûte que du temps au client et non au serveur. C'est pourquoi les puzzles sont intéressants.

Le puzzle utilisé dans notre code est un puzzle simple qui consiste des lettres d’alphabet arranger aléatoirement et vous devez les trier alphabétiquement. (Le puzzle devrait être un puzzle d’éternité, mais j’ai plus le temps pour les compléter).

# Structure

Notre Structure consiste de 2 partie principale les serveurs et les clients.

## Les Clients

Notre protocole TCSMP est une extension du protocole SMTP.

Chacun de ces 2 protocole a une Session TCSMPSession.java (respectivement TPOPSession.java) qui contient tout le Command et Méthode nécessaire a effectué une connexion TCSMP (respectivement TPOP) au serveur.

Voici les états que nous avons mis en place pour respecter notre protocole pour faire la connexion :

* Registration : Tout d’abord le client doit s’enregistrer dans la liste des utilisateurs dans le serveur alors il envoie le command <REG> et selon la réponse du serveur le client prend action.
* Envoie d’un courrier : Chaque command envoyé au serveur et rencontrer par une réponse (réponse code + message du serveur) et selon la réponse le client détermine ce qu’il doit faire (En d’une réponse positive le client continue l’échange de command avec le serveur, en cas d’une réponse négative le client throw une Exception). L’envoie du message est divisé en plusieurs command :
  + Initialisions : Tout d’abord le client envoie <TELO> avec le domaine du serveur du client (ex : TELO BINIOU.com) et attend la réponse du serveur pour la vérifier.
  + Détermination d’expéditeur : Cet étape s’occupe d’envoie l’adresse de l’expéditeur. Le command utiliser à cette étape est <FROM>.
  + Détermination du destinataire : Cet étape s’occupe d’envoie l’adresse du destinataire. Le command utiliser à cette étape est <RCPT>.
  + Demande du puzzle : Cet étape s’occupe d’envoie le demande d’un puzzle. Le command utiliser à cette étape est <APZL>.
  + Envoie la solution du puzzle : Cet étape s’occupe d’envoie la solution du puzzle reçu. Le command utiliser à cette étape est <PKEY>.
  + Envoie du message : Cet étape s’occupe d’envoie le message (Headers + Sujet + Message). Le command utiliser à cette étape est <MAIL>.
  + Fin de la session : Cet étape s’occupe de la fermeture de la session. Le command utiliser à cette étape est <QUIT>.
* Réception d’un courrier : De la même manière précédente, chaque requête au serveur a une réponse et a chaque fois le client vérifie la réponse, mais cette fois le client initialise une Session TPOP, le TPOP est identique à POP3 (même command). La récupération du message est divisée en plusieurs command :
  + Authentification : Tout d’abord le client envoie <USER> avec le nom de l’utilisateur. Puis il envoie le command <PASS> avec le mot de passe.
  + Demande la liste des messages : Cet étape s’occupe de la demande de liste des messages trouver sur le serveur (dans le Inbox). Le command utiliser à cette étape est <LIST>.
  + Demande d’un message : Selon la liste obtenue par l’étape précédente le client récupérer les messages un par un. Le command utiliser à cette étape est <RETR> avec l’identificateur de chaque message.
  + Fin de la session : Cet étape s’occupe de la fermeture de la session. Le command utiliser à cette étape est <QUIT>.

## Les Serveurs

Chaque serveur implémente les 2 protocoles TCSMP et TPOP. Au début de lancement le serveur crée 2 threads une pour écouter si une connexion TCSMP (respectivement TPOP) est fait au serveur, en cas une connexion est effectuée le thread du TCSMP (respectivement TPOP) lance un nouveau thread pour traiter cette connexion TCSMPClientHandler (respectivement TPOPClientHandler) qui par son tour effectue l’envoie et la réception des command pour envoyer (respectivement retriever) les messages du serveur.

### TCSMP Client Handler

Ce Thread s’occupe des clients qui sert à s’enregistrer ou à envoyer un message. Le Handler répond à chaque command par une réponse positive (commence par 200 ou 300) ou par une réponse négative (commence par 500).

### TPOP Client Handler

Ce Thread s’occupe des clients qui sert à récupérer un message. Le Handler répond à chaque command par une réponse positive (commence par +) ou par une réponse négative (commence par -).