Compte-rendu

Implémentation d'un client POP3

et d'un client TFTP

Médéric HURIER - Armand PY-PATINEC - Apoté AMBOUNDA ZEKPA

2012-2013

1. Introduction

1.1 Protocoles

POP3

Le protocole POP (Post Office Protocol) a d'abord été défini en octobre 1984 (RFC 918), puis remplacé par la version 2 du même protocole en février 1985 (RFC 937).
La version actuelle du protocole, la version 3, a d'abord été définie en novembre 1988 (RFC 1081). Ce standard a été mis à jour en mai 1991 (RFC 1225), juin 1993 (RFC 1460), en novembre 1994 (RFC 1725) et finalement en mai 1996 (RFC 1939). Deux autres documents complètent la RFC 1939: la RFC 1957 donne quelques informations supplémentaires, tandis que la RFC 2449 définit des extensions au protocole.

TFTP

Défini en 1980, le protocole TFTP (Trivial File Transfer Protocol) est un protocole simplifié de transfert de fichiers.
La dernière version de ce protocole est la version 2, définie dans RFC 1350 Pour ce projet, seul le mode de transfert binary sera utilisé.

1.2 Répartition des tâches

Nos compétences en réseau sont très hétérogènes. Pour que le travail soit intéressant pour tout le monde, nous avons choisi cette organisation:

1. **Apote** sur le protocole POP3
2. **Armand** sur le protocole TFTP
3. **Médéric** sur la refactorisation et les tests unitaires

Nous avons effectué la conception ensemble, mais de nombreuses modifications ont été apportées depuis cette étape. Aussi, les éléments que nous vous proposons sont tirés de rétro-ingénierie.

1.3 Interface utilisateur

L'interface utilisateur est entièrement **en mode texte.**

Elle s'inspire du client TFTP par défaut de Linux, fourni avec une invite de commande. Les actions sont accessibles en tapant le mot clé "help" après le lancement.

Les messages de sortie sont redirigés vers la sortie standard, et les erreurs vers la sortie d'erreur du système.

2. Usage

Pour tester le fonctionnement des protocoles, il est nécessaire de lancer les serveurs correspondants aux clients. **Les clients ne prennent aucun d'arguments.**

2.1 Serveurs

Lancement du serveur POP3

1. aller dans le dossier server/pop3/
2. lancer l'exécutable PopServer.class

Après la fermeture de la connexion client, il est nécessaire de relancer le serveur.

Lancement du serveur TFTP

1. aller dans le dossier server/tftp
2. lancer l'exécutable TFTPServer avec:
    1. en premier paramètre la gestion de la perte de paquet (0 pour non, 1 pour oui)
    2. en second paramètre le numéro de port (par exemple: 1069)

Le répertoire d'exécution fournit au serveur la liste des fichiers disponibles. Il est possible d'en changer pour tester la commande PUT ou GET.

2.2 Client

Lancement du client POP3

1. aller dans le dossier dist/
2. lancer l'exécutable pop3.jar (java -jar pop3.jar)

Un message de bienvenue doit s'afficher, avec en début d'invite de commande le texte
pop3 >

Liste des commandes :

1. help: affiche l'aide
2. connect host <port>: connexion à un hôte/port **(host doit être une adresse IP)**
3. auth user pass: authentification d'un utilisateur
4. stat: statistique sur le stockage des mails de votre compte
5. collect: récupère les emails du compte (suppression en cas de succès)
6. list: liste les messages stockés localement
7. read: lit le contenu d'un mail
8. quit: quitte le programme

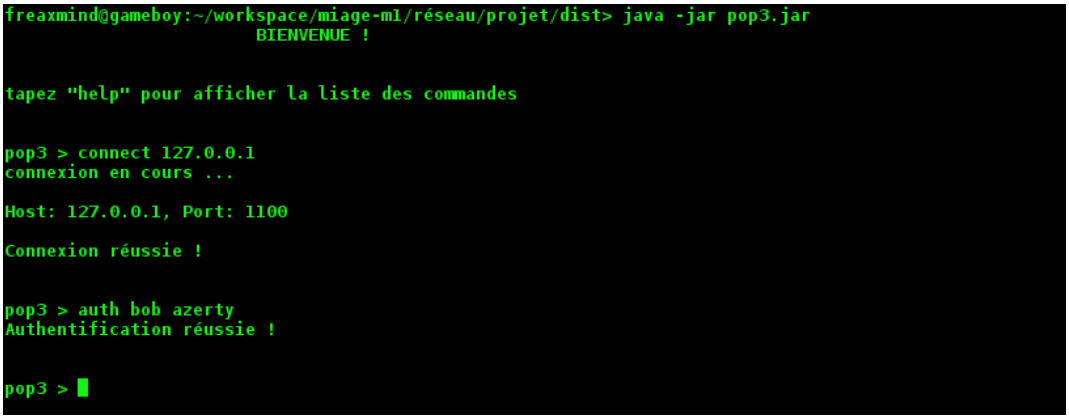

Figure 1 : Invite de commande pour le client POP3

Lancement du client TFTP

Le client TFTP nécessite 2 répertoires pour fonctionner:

1. get/ : dossier où seront stockés les fichiers téléchargés du serveur
2. put/ : dossier dont le contenu sert à téléverser des fichiers au serveur

Un message de bienvenue doit d'afficher, avec en début d'invite de commande le texte tftp >

Liste des commandes :

1. help: affiche l'aide
2. connect host <port>: connexion à un hôte/port **(host doit être une adresse IP)**
3. get fichier: récupère un fichier sur le serveur
4. put fichier: envoie un fichier sur le serveur
5. quit: quitte le programme

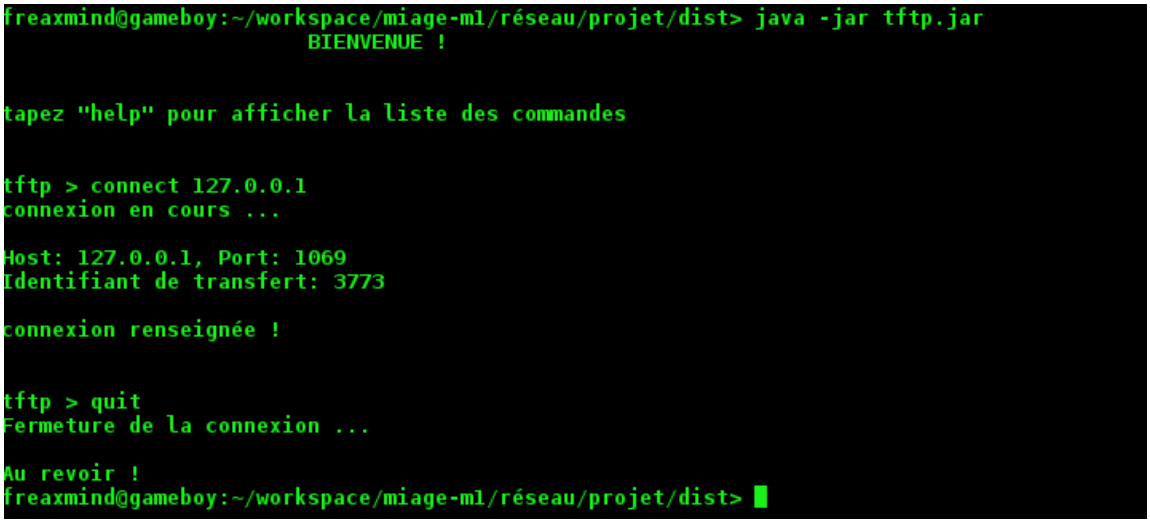

Figure 2 : Invite de commande pour le client TFTP

3. Structure logique

3.1 POP3

Le diagramme de classe suivant présente la structure logique du client POP3. Notez que le paquet “common” est commun au client POP3 et TFTP.

Figure 3 : Diagramme de classe du client POP3

Le point d’entrée est la classe **POP3**. Il crée la classe client et appelle la méthode *run*.

La classe **Client** dispose de toutes les commandes POP3 supportées (*connect*, *auth* …). Ces méthodes spéciales prennent un tableau de paramètre en entrée et sont appelées grâce à un patron de conception décorateur (méthode *runCommand*) et dispatcher (méthode *run*).

Les méthodes *sendCommand* et *getResponse* servent à gérer la réception et l’envoi de messages au serveur. Ils sont codés dans par les classes **Message** et **Response**.

En cas d’erreur, on lève une instance de la classe **POP3Exception**.

3.2 TFTP

Le diagramme de classe suivant présente la structure logique du client TFTP. Notez que le paquet “common” est commun au client POP3 et TFTP.

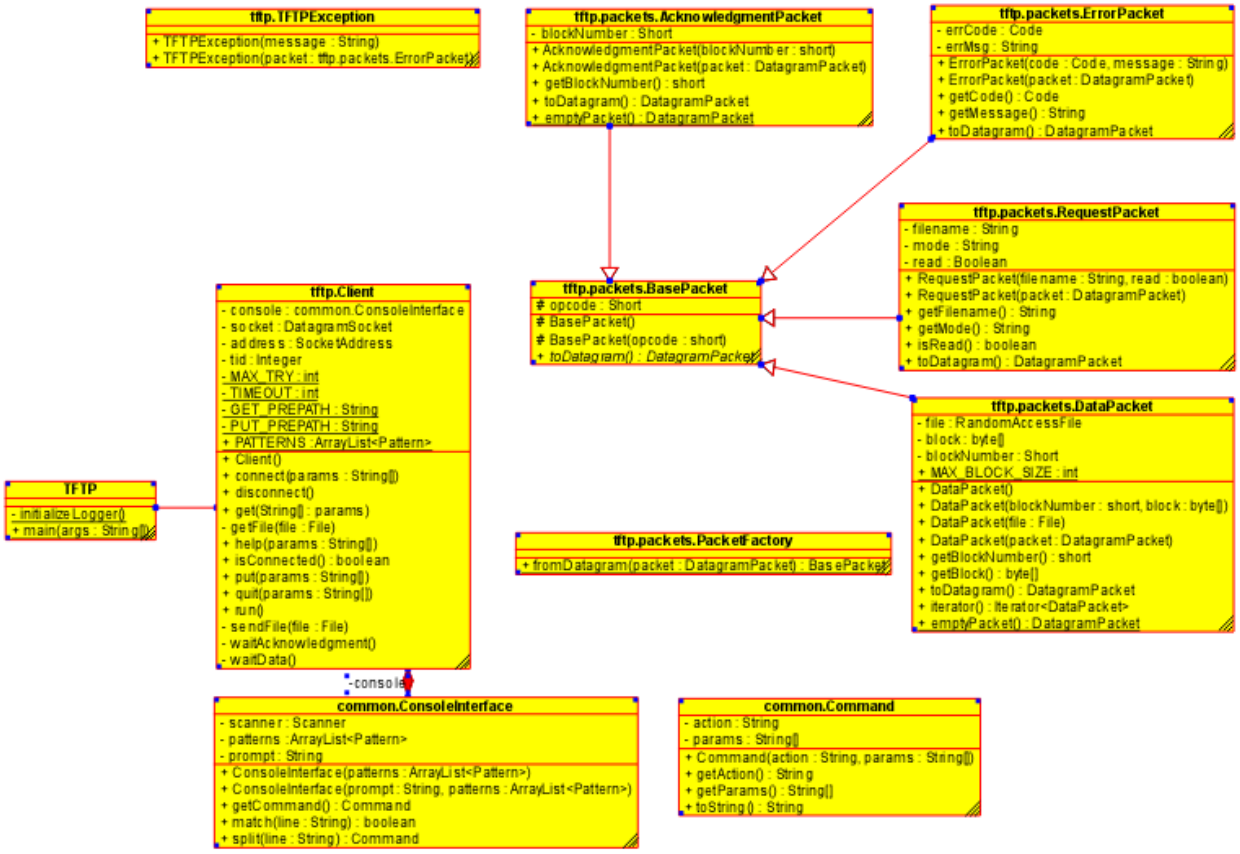

Figure 4 : Diagramme de classe du client POP3

Le point d’entrée du programme est la classe **TFTP** qui créée et lance le **Client**.

Le client dispose de 4 méthodes principales: *connect*, *get*, *put* et *quit*. Elles correspondent aux commandes utilisateurs de l’invite de commande. Des méthodes privées *getFile* et *sendFile* servent de conteneur (handler) pour recevoir et envoyer des fichiers.
Les méthodes *waitAcknoledgment* et *waitData* servent à recevoir les réponses du serveur.

Chaque type de paquet de TFTP est codé dans une classe paquet qui hérite de la classe de base **BasePacket**. Celle-ci doit fournir un *constructeur* prenant un **DatagramPacket** pour décoder un paquet UPD et une méthode *toDatagram* pour coder le paquet UDP.

Une classe implémentant le patron de conception fabrique **PacketFactory** permet de renvoyer le paquet en fonction d’un datagramme. En cas de problème, on lève une **TFTPException**.

4. Structure physique

Figure 5 : Dossier du projet

1. dist : exécutable Java des protocoles (format .jar)
2. docs : documentation du projet (diagrammes, rapports, javadoc)
3. server : exécutable Java des serveurs (format .class)
4. src : source du projet
    1. POP3.java : point d'entrée du client POP3
    2. TFTP.java : point d'entrée du client TFTP
    3. common : classes communes aux deux protocoles
5. test : tests unitaires (outil JUnit)

5. Difficultés

Un projet réseau est plus difficile à mettre en œuvre et à tester qu'un simple projet Java. L'exécution dépend de la réception et de l'envoie de paquets, ainsi que du système d'exploitation. Il n'y a pas de flot d'exécution, ce qui rend le débogage plus ardu.

Cependant, nous n'avons pas rencontré de difficultés majeures grâce aux tests unitaires JUnit que nous avons réalisés au préalable et l'utilisation de la commande tcpdump sous Linux.
Le plus dur pour nous a été de trouver les classes Java adaptées à la programmation réseau. Après plusieurs recherches, nous avons trouvé les classes ByteBuffer et ByteBufferOutputStream très pratique.

En conclusion, le projet était très simple, mais nous avons essayé de le faire soigneusement et de lire attentivement les RFC pour comprendre les subtilités des protocoles.