Serveur Proxy Cache
Rapport

Guillaume Huysmans, Julien Amalaberque

May 4, 2015

1 Introduction

Ce projet a été réalisé dans le cadre du cours de Réseaux 1 de monsieur Bruno QUOITIN (Université de Mons).

2 Fonctionnalités

2.1 Lancement

Le proxy HTTP peut être démarré indépendamment d'un IDE. Nous avons décidé de ne pas implémenter un client HTTP. Il était selon nous plus efficace de rendre notre travail compatible avec les clients existants. Notre proxy a été testé avec Firefox 37.

2.2 Redémarrage

Les informations en cache sont conservées lors d'un redémarrage (elles sont stockées sous forme de fichiers).

2.3 Protocole

Le proxy supporte HTTP 1.1 et l'IPv6.

2.4 Connexions multiples

2.4.1 proxyx.c

Une boucle dans main() crée un fork pour chaque client.

2.5 Persistance

La connexion TCP reste bien ouverte après avoir envoyé la réponse au client.

2.5.1 proxyx.c

Dans make_request(), le socket n'est pas fermé après l'appel de transmit_response(), une méthode envoyant la réponse au client.

2.6 Pipelining

Un élément clé de notre implémentation du pipelining est la gestion des buffers.

2.6.1 buffer.c

read_buffered() lit les données d'un buffer créé par read_until et puis d'un descripteur de fichiers UNIX. read_until recherche une chaîne de caractère dans un buffer, et désalloue de la mémoire le buffer quand il n'est plus utile.

2.6.2 proxyx.c

Dans handle_client(), une boucle ne ferme la connexion que lorsque l'on reçoit un message "Connection: close" ou si la connexion est fermée de l'autre côté.

3 Outils

3.1 Dépendances

- Les bibliothèques OpenSSL et Lua;
- Un compilateur C++ car CMake en a besoin.

3.2 Documentation

Nous avons utilisé Doxygen pour gérer la documentation de nos fonctions. Elle est accessible par `doc/html/index.html` une fois la commande "`make doc`" effectuée.

3.3 Tests unitaires

Nous utilisons CuTest pour tester nos fonctions. Ils sont regroupés et exécutés automatiquement lorsque la commande "`make test`" est utilisée.

3.4 CMake

CMake génère un makefile et vérifie la configuration du système.

3.5 Mozilla Firefox 37.2

Nous avons créé des profils différents et un script qui ouvre le profil de test.

3.5.1 Création d'un profil

Lancez Firefox via la commande "`firefox -new-instance -P`". Cela ouvrira le gestionnaire de profils. Cliquez sur "`Create profile`", nommez-le et lancez Firefox avec.

3.5.2 Configuration de Firefox

Dans "Menu","Préférences","Avancé", naviguez dans l'onglet "Réseau" puis cliquez sur "Paramètres". Sélectionnez ensuite "Configuration manuelle du proxy". Dans "Proxy HTTP", entrez localhost et dans "Port" 8080.

3.6 Testé sous Ubuntu et Debian

Devrait fonctionner sous tout système UNIX avec sockets.