mickael.hoerdt@hesge.ch 2016/2017

Mini-projet de conception de protocoles réseaux informatiques

Un client/serveur TFTP encore plus simple

Trivial File Transfert Protocol (TFTP) est un protocole client/serveur extrêmement simple conçu pour être implémenté sur des équipements ne disposant que de très peu de ressources matérielles. Il est spécifié dans la RFC 1350[1]

On le trouve très souvent dans les mémoires flash d'équipements divers nécessitant d'effectuer des mises à jour du système ou le téléchargement d'un fichier de démarrage ou de configuration d'un équipement depuis un serveur contenant ces fichiers.

Le but de ce mini-projet sera de concevoir un client/serveur TFTP simplifié, en utilisant les méthodes vues en cours. Il sera divisé en trois étapes, décrites ci-après.

1. Modélisation et analyse du protocole

Le service, l'environnement, les types de messages et leur encodage ainsi que leur règles d'échange devront être décrit informellement à partir du RFC 1350 pour ensuite être modélisés en utilisant le formalisme vu en cours : machine à états et diagrammes temps séquence.

La modélisation devra spécifier le fonctionnement et le déroulement du service offert par le protocole du coté client et du coté serveur en se limitant au mode de transfert binaire du protocole (c'est à dire le mode « octet »). Une machine a état devra spécifier le fonctionnement du service et du protocole coté client, une autre machine à état devra spécifier le coté serveur.

Les différents cas d'erreurs pouvant se produire devront être déroulés sur des diagramme temps séquence et démontrer que le protocole récupère bien l'erreur dans les cas présentés, sur la base des machines à état proposées. Un diagramme temps séquence ne devra dérouler qu'un seul cas d'erreur.

La modélisation du protocole devra aussi contenir un premier diagramme de classes représentant les différents objets qui serviront à implémenter le protocole.

2. Implémentation du client TFTP en lecture/écriture

A partir de la spécification réalisée de l'étape 1 sera implémenté le protocole TFTP du coté client ainsi qu'une interface en ligne de commande permettant de l'utiliser/le tester. L'implémentation devra utiliser le langage python 2 ou 3 et les sockets BSD[2] UDP.

Le client TFTP devra permettre de lire ou écrire un fichier à distance pour le stocker dans le répertoire d'exécution. Votre implémentation devra aussi permettre de simuler des pertes de paquets

1 https://tools.ietf.org/html/rfc1350
2 https://en.wikipedia.org/wiki/Berkeley_sockets

selon un pourcentage de perte passé en paramètre au client précisant la probabilité qu'un paquet soit perdu en réception ou en émission sur le client.

L'interface de votre client devra se présenter sour la forme d'un outil en ligne de commande dont le synopsis sera le suivant :

```
mytftpc <put/get> <nom_DNS_du_serveur> <numero_de_port> <nom_fichier> <taux_de_perte>
```

Le premier paramètre précisera si il s'agit d'une lecture ou d'une écriture distante. Si il s'agit d'une lecture (`get`), le nom de fichier indique le nom de fichier distant que l'on souhaite recevoir. Si il s'agit d'une écriture (`put`), le nom de fichier indique le nom de fichier que l'on souhaite envoyer vers le serveur.

Le deuxième ainsi que le troisième paramètre indique sur quel serveur le client devra aller lire ou écrire le fichier.

Le dernier paramètre indique le taux de perte simulé par le client.

Pour aider le test du bon fonctionnement de votre implémentation, un serveur tftp sur le port par défaut du protocole est disponible en lecture à l'adresse suivante, et sert les fichiers suivants dont la taille est indiquée en deuxième colonne. Le checksum md5 du fichier est indiqué en 3ème colonne pour vous permettre de vérifier que le transfert de fichier s'est bien effectué.

```
hepia.infolibre.ch

nom de fichier      taille                  hash md5

test100             100 octets              a743a69fa909a4b05137afa71195635f
test512             512 octets              b764dcd5f509de6b80c1f9187647ee86
test1024            1024 octets             57ffd4a26a6af245882d9f4206600ec5
test1025            1025 octets             60c34afe07d94be2788eddc2eec75802
test5000            5000 octets             56daa9999763fa21d747af2dbc6b9a57
test100k            100 kilo-octets         8b8db96116c6bc2f721284485fe97c18
test1100k           1100 kilo-octets        5e4791181222b26fcdab8da6f14deb15
test7777k           7777 kilo-octets        a60814a887e267e2412c268c549be5c0
```

Pour l'écriture, un serveur tftp est disponible en écriture à la même adresse, avec les mêmes noms de fichiers, mais préfixés par la chaine de caractère `w_`

3. Implémentation d'un serveur TFTP en lecture uniquement

Un serveur permettant de répondre aux requêtes en lecture d'un client TFTP devra être implémenté à partir de la spécification de l'étape 1. De même que pour l'étape 2, l'implémentation du serveur devra utiliser le langage python et les sockets BSD UDP et se présentera sous la forme d'un programme séparé, a exécuter en ligne de commande.

Le serveur TFTP servira en lecture uniquement les fichiers contenus dans un répertoire donné en paramètre. Il ne devra **pas** implémenter les requêtes en écriture. Il ne pourra servir **qu'un seul client** à la fois.

L'interface du serveur devra avoir le synopsis suivant :

```
mytftps <numero_de_port> <nom_de_répertoire>
```

Le premier paramètre précise le numéro de port sur lequel le serveur se mettra en attente de requête de la part d'un client, le deuxième paramètre le répertoire dans lequel se trouvent les fichiers à servir.

Pour vérifier la bonne conformité de votre implémentation, votre serveur devra être testé avec un autre client TFTP que le votre. Ces tests devront être documentés dans le rendu de l'étape.

Modalité de rendu

L'ensemble du projet devra être réalisé en priorité en binôme. L'évaluation de l'étape 1 compte pour 40 % de la note, l'étape 2 pour 45%, l'étape 3 pour 15%. Dans le cas où une personne se retrouverait seule, elle doit effectuer le projet en trinôme et dans ce cas l'évaluation de chaque étape comptera pour 1/3 chacune.

- L'étape 1 devra être rendu sous la forme d'un document au format pdf envoyé par mail au professeur. Ce document devra contenir la modélisation et l'analyse telle que décrite au point 1. Elle devra être rendue avant le **25/11/2016 23h59.**

- L'étape 2 devra être rendu sous la forme d'un fichier zip envoyée par mail au professeur. Elle devra contenir l'ensemble du code de l'implémentation. Un fichier README.txt présent dans l'archive contiendra des captures de la sortie de votre programme ainsi que des exemples d'exécution et de test de votre implémentation. Ce fichier devra être au format ASCII. Le fichier d'archive devra être envoyé avant le **23/12/2016 23h59** et avec un nom respectant obligatoirement le format suivant :

```
<nom_binome1_nom_binome2_etape_2_tftp>.zip
```

- L'étape 3 devra être rendu sous la même forme que pour l'étape 2, à part que le nom du fichier contiendra `etape_3` plutôt que `etape_2` dans son nom. Elle devra être rendu avant le **20/1/2017 23h59**