

Implémentation du protocole TFTP

Kevin Estalella, Federico Lerda, Federico Pfeiffer

5 décembre 2016

Résumé

Ce rapport décrit l'implémentation du protocole TFTP dans le langage python.

Table des matières

1 Présentation

Le projet consiste en l'implémentation du protocole TFTP. Ce protocole est une version light du protocole FTP. Le but de ce projet est d'implémenter une version simplifiée de TFTP en utilisant les méthodes vues en cours. Le protocole est implémenté en se basant sur la RFC 1350.

2 Protocole TFTP

2.1 Vue d'ensemble

Le protocole TFTP est un protocole servant à envoyer / recevoir des fichiers entre un client et un serveur. Les deux parties peuvent se comporter comme émetteur et récepteur. La transmission d'un fichier débute par une requête de la part du client, dans le but de demander à envoyer un fichier (requête représentée par un paquet WRQ), ou dans le but de demander à recevoir un fichier (requête représentée par un paquet RRQ). Si le serveur accepte, le transfert de fichier commence.

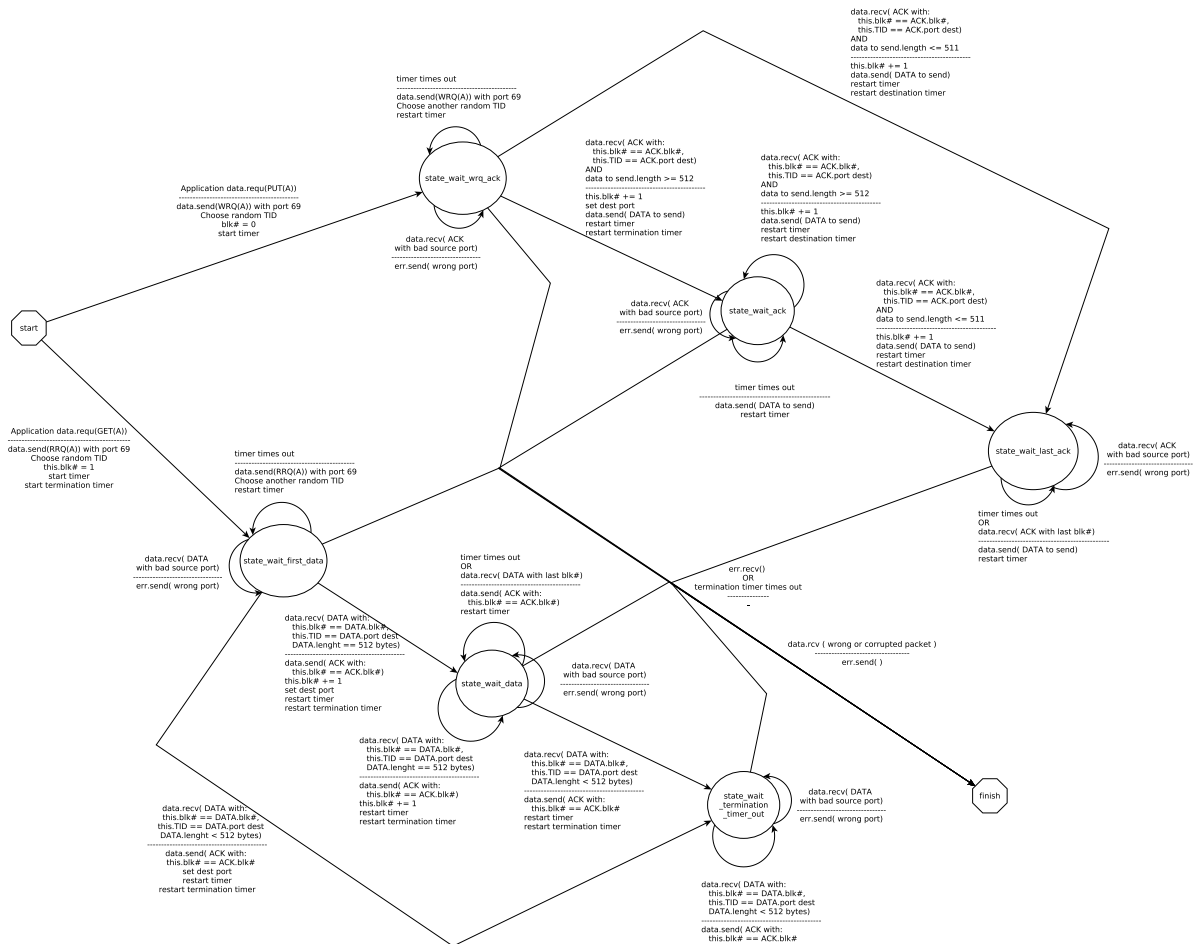
Les fichiers sont transmis par paquets de données d'une taille de 512 Octets. Un paquet de données contenant moins de 512 Octets signifie qu'il s'agit le dernier paquet.

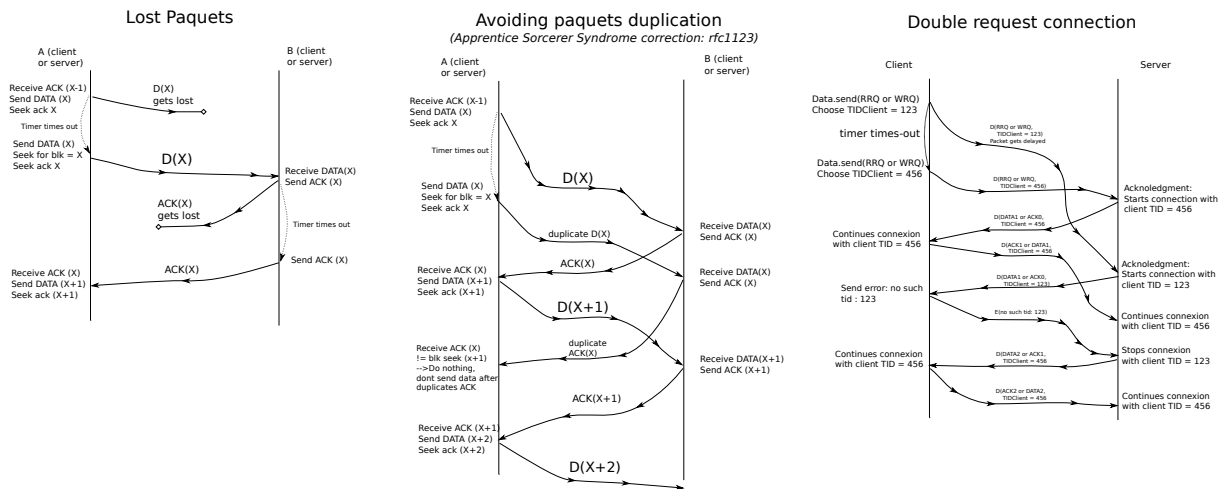
Chaque paquet contient un numéro de paquet et est acquitté par le destinataire par un paquet ACK ; l'émetteur d'un paquet de données doit attendre la réception d'un ACK avant d'envoyer le paquet de données suivant. En cas d'erreur, un paquet ERR est envoyé et le transfert est annulé et terminé.

2.2 Machines d'états

Les machines d'états suivantes ont été créés en se basant sur le descriptif de la RFC 1350.

2.2.1 Client





3 Architecture du programme

L'architecture de notre implémentation a été définie comme suit. Deux fichiers `client.py` et `server.py` qui contiendront l'algorithme défini dans les machines d'états.

Partant du principe que la manière de transférer les fichiers est la même pour le serveur et le client, nous avons défini un fichier `utils.py` qui contiendra les méthodes pour envoyer et recevoir des fichiers. Ces méthodes sont utilisées par le client et le serveur.

Enfin, un fichier nommé `packet.py`, également utilisé par le serveur et le client, contient toutes les fonctions nécessaires pour encoder et décoder les paquets TFTP.

utils.py:

```
Boolean(success), String(err_msg) : send_file(socket, file_descriptor)

Boolean(success), String(err_msg) : receive_file(socket, file_descriptor, first_data_blk)

(option) : parse_user_input(args)
```

client.py: (pseudo code)

```
parse input
send request
open corrects ports, file descriptors
send_file or receive_file
handle error if any
close everything
```

server.py: (pseudo code)

```
start listening
loop:
    open correct ports, file descriptors
    send_file or receive_file
    handle error if any
    close files / correct ports
```

packet.py:

```
Enum : OP_CODE
Enum : ERR_CODE

BinaryString : build_packet_rrq(Filename, mode)
BinaryString : build_packet_wrq(Filename, mode)
BinaryString : build_packet_data(Block_num, data)
BinaryString : build_packet_ack(Block_num)
BinaryString : build_packet_err(Err_code)

For RRQ and WRQ
Op_code, File_name, Mode : decode_packet(BinaryString)

For ACK
Op_code, Block_num : decode_packet(BinaryString)

For DATA and ERR
Op_code, Block_num, Msg : decode_packet(BinaryString)
```