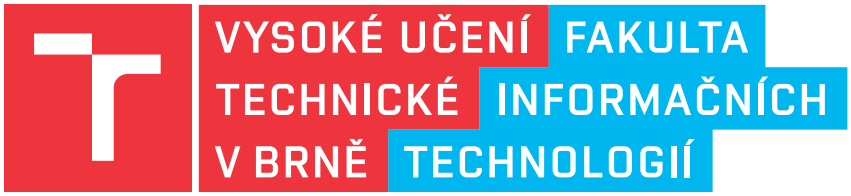

Projektová dokumentácia

TFTP klient

Sieťové aplikácie a správa sietí

15. novembra 2021 Johann Adam Gawron (xgawro00)

Obsah

1 Úvod do zadania a TFTP

Zadanie je nasledujúce, implementujte program DNS, ktorý bude zasielať dotazy na DNS servery a včitatelnej podobe vypisovať prijaté odpovede od daného DNS servera na štandartný výstup. Zostavenie a analýza DNS paketov musí byť implementovaná priamo v programe. Stačí považovať iba UDP komunikáciu.

1.1 Čo je to TFTP

Trivial File Transfer Protocol je veľmi jednoduchý protokol, slúžiaci na prenos súborov. Čiastočne obsahuje základnú funkcionalitu zložitejšieho FTP. Funguje nad protokolom UDP, ktorý sám o sebe nie je spoľahlivý, a preto je nutné zabezpečenie spojenia so serverom a kontrolovanie prijímania a odosielania rôznych typov paketov. Jeho využitie je vhodné v prípade, že existujú dôvody, kedy je použitie FTP protokolu príliš zložité.

1.2 TFTP klient

V implementácií TFTP klienta mám:

- Čítanie súboru zo serveru (READ)
- Zapisovanie súboru na server (WRITE)
- blocksize
- tsize
- IPv4 a IPv6 funkcionalita
- timeout

1.2.1 Základná funkcionalita

Keď chceme čítať dáta zo serveru(z nejakého súboru), potrebujeme dať serveru vedieť

1.2.2 Typy paketov

- 1 RRQ
  Read Request, pozostáva z opcode, filename, mode + options
  Jeho pomocou otvárame komunikáciu so serverom na sťahovanie/čítanie dát

- 2 WRQ
  Write Request, pozostáva z opcode, filename, mode + options
  Otvára komunikáciu na nahrávanie/zapisovanie dát

- 3 DAT
  Data packet, pozostáva z opcode, block number, a samotných dát
  Forma na prenos dát

- 4 ACK
  Acknowledge packet pozostáva z opcode a block number, kde číslo blocku je rovnaké, ako posledný prijatý blok

- 5 ERR
  Error, pozostáva z opcode, errorcode a errormsg

- 6 OPT
  Option extension, slúži na rozšírenie možností vyjednávania, ako tsize a blocksize

1.2.3 Komunikácia so serverom

So serverom sa komunikuje pomocou protokolu UDP. Inicializácia sa defaultne začína na porte 69, čo je port pre TFTP. Následne server zašle odpoveď, ktorá príde z novo inicializovaného portu pre daný socket, kde sa potom odohráva zvyšok komunikácie. Komunikácia začne ak klient zašle buď Read alebo Write Request a server mu odošle pozitívnu odpoveď buď vo forme ack alebo data packetu. Ukončuje sa ak príde/odíde packet s veľkosťou menšiou ako blocksize+header.

2 Implementácia

Keďže počet riadkov zdrojového kódu presahuje niekoľko stoviek, bolo vhodné tento kód rozdeliť do viacerých súborov a hlavičkových súborov aby bol kód a celkovo program prehľadnejší. Štruktúra súborov pre môj TFTP klient:

- `clifunctionality.cpp` — sparsovanie vstupných parametrov a ich validácia
- `clifunctionality.h` — pomocné štruktúry a konštanty
- `mytftpclient.cpp` — hlavný kód implementujúci najmä prenos paketov
- `mytftpclient.h` — hlavičkový súbor k hlavnej časti programu
- `pcapDataTypes.h` –¿ štruktúry k správnemu pracovaniu IPv4 a IPv6 + ďalšie pomocné konštanty
- `tftpSocket.cpp` –¿ Komunikácia zo serverom a otváranie socketu
- `tftpSocket.h` –¿ protypy funkcii využívaných v tftpSocket.cpp

V mojom TFTP klientovi nie je implementovaná podpora pre multicast(-m) a nie je možné meniť mód(-c) na netascii(je k dispozícii iba binárny - tj. oktet).

2.1 Spracovanie vstupu

Parametre nutné k použitiu môjho TFTP klienta nie je možné spracovať pomocou tzv. command line argumentov ale cez štandardný vstup(stdin) v mnou vytvorenom command line interfaci(CLI), ktorého implementácia sa nachádza v súboroch *clifuncionality.c* a *clifuncionality.h*.

Vstup je spracovaný vo funkcií `parseArgs()`, v ktorom pomocou cyklu for rozdelím vstupný string na základe oddeľovačov(biele znaky a čiarky) a rozdelím si vstupy do rôznych kategórií podľa parametrov, kde následne kontrolujem ich validitu a či súv súlade s očakávaním vstupom.

V prípade chybného vstupu je užívateľ požiadaný o opätovné zadanie vstupných parametrov(tj. program nie je okamžite ukončený).

2.2 tftpSocket

V súbore tftpSocket, sa nachádzajú špagety, ktoré ukrývajú implementáciu ipv4 a ipv6 spracovania TFTP packetov a ich zasielanie na, v súbore, vytvorené sockety.

Rozdelené sú na Read IPv4, Read IPv6, Write IPv4 a Write IPv6.

2.3 clifunctionality

Obsahuje kompletnú funkcionalitu pre custom CLI rozhranie. Je v nej štrukturovane zapisané spracovanie argumentov.

2.4 Použitie TFTP serveru

Keďže bolo potrebné implementovať iba TFTP klienta, využil som voľné dostupný TFTP server dosstupný na githube. Odkaz: `https://github.com/reinerh/rtftp`

3 Spúšťanie programu

Prekladanie programu je pomocou súboru **Makefile**. Spustenie je nasledujúce:

```
./mytftpclient
>-R/W [-d] adresar/soubor [-s] velikost [-a] adresa,port

-R/-W - čítanie zo serveru alebo nahrávanie na server
-d    - absolútna cesta čítaného alebo nahrávaného súboru
-s    - maximálna veľkosť bloku v násobkoch oktetu
-a    - adresa serveru(IPv4 aj IPv6 fomrmát), port na ktorom server počúva
```

3.1 Príklad spustenia

```
./mytftpclient
>-R -d myfolder/index.php -s 1024 -a 192.168.0.25,69
>-W -d test -s 4096 -t 10
>-R -d folder/movie.mp4 -s 4096 -a ae70:8bb4:9eb9:e5ea:d062:24fe:0816:29b0,65535
```

Použitá Literatúra

- https://datatracker.ietf.org/doc/html/rfc1350 — RFC 1350
- https://datatracker.ietf.org/doc/html/rfc1785 — RFC 1785
- https://datatracker.ietf.org/doc/html/rfc2347 — RFC 2347
- https://datatracker.ietf.org/doc/html/rfc2348 — RFC 2348
- https://datatracker.ietf.org/doc/html/rfc2349 — RFC 2349
- https://www.linkedin.com/pulse/tftp-client-implementation-c-sumit-jha/ — example TFTP implementation