Vysokého učení technického v Brně
Fakulta informačních technologií

ISA - Síťové aplikace a správa sítí

TFTP Klient + Server

14. listopadu 2023

Ivan Golikov(xgolik00)

Obsah

1 Úvod

Cílem tohoto projektu bylo napsat klientskou a serverovou část, která bude mezi sebou komunicovat pomocí protokolu TFTP[8]. Všechny informace budou zobrazeny jako logy, která budou zobrazena na standardním výstupním toku 'stderr' a budou obsahovat základní příchozí informace z druhé strany.

Kromě základních funkcí, které TFTP server zahrnuje, je rozšířen podle RFC 2347[6], který umožňuje spolu s prvním balíčkem odeslat určité optiony, které změní typ přenosu souboru.

Mezi takové možnosti patří velikost přenášených dat v jednom balíčku[5], interval čekání a velikost souboru[7], který umožňuje určit maximální velikost souboru ještě před jeho odesláním.

2 Implementace

Program je umístěn ve dvou složkách. Složka 'src' obsahuje hlavní soubory s kódem a složka 'include' záhlaví k nim. V každé složce jsou čtyři soubory:

- tftp-client - soubor klienta
- tftp-server - soubor serveru
- packet - soubor pro skládání a odesílání různých typů paketů
- common - společné funkce pro klienta a server kromě odesílání paketů

Všechny zkompilované kódy budou umístěny do složky 'bin'.

2.1 Client

Klient je určen pro komunikaci v módě 'octet' se serverem a již má přednastavené parametry voleb na {`'blksize':1024, 'timeout':10, 'tsize': XXX`}. Hodnota 'tsize' závisí na typu komunikace. Pokud se jedná o RRQ, pak hodnota proměnné bude 0, pokud WRQ, že velikost souboru. Parametry klienta jsou nastaveny společně s jeho spuštěním a předány `struct parametrs`, což slouží k pohodlnějšímu uložení všech vstupních parametrů.

```
/**
 * Structure for input parametrs
*/
struct parametrs {
    char* hostname;
    int port = STANDART_PORT;
    char* filepath;
    char* dest_filepath;
};
```

Obrázek 1: Client struct parameters

2.1.1 Funkce get_parametrs()

Funkce kontroluje, přepočítává a přiřazuje všechny vstupní parametry z příkazového řádku do proměnné pro vstupní parametry. Kromě toho funkce určuje další typ komunikace.

Hlavní součástí funkce je cyklus while, ve kterém probíhá kontrola krátkých parametrů a jejich hodnoty pomocí funkce getopt()[2]. Pokud parametr-f chybí, funkce změní typ operace z WRQ na RRQ

2.1.2 Funkce request()

Hlavní funkcí je komunikace. Funkce vytvoří cestu (pokud je to nutné) k souboru a vytvoří samotný soubor, poté odešle první balíček pro zahájení přenosu nebo přijetí souboru. Po odeslání balíčku, funkce čeká na odpověď od serveru a zkontroluje, zda je správně, pokud nejsou nalezeny žádné chyby, přejde do funkce RRQ_loop() nebo WRQ_loop() v závislosti na operaci.

2.1.3 Funkce RRQ_loop()

Funkce určená k nahrávání přicházejícího souboru v módě 'octet'. Přijetí souboru vypadá jako cyklus, kde se nejprve očekává datový paket a cyklus skončí po zaznamenání přijatých dat, když bude přijat balíček o velikosti menší než maximální.

Po přijetí paketu je definován 'opcode' paketu. Pokud se jedná o DATA, dojde k výše uvedenému záznamu dat, pokud se jedná o ERROR, pak je výměna balíčků dokončena a program končí.

2.1.4 Funkce WRQ_loop()

Funkce odesílání souboru na server je nastavena v režimu 'octet'. Odesílání paketu začíná záznamem z cílového souboru na proměnnou 'data'. Pokud byl zaznamenán maximální počet znaků nebo když byl znak načten na hodnotu 0, odešle se datový paket a očekává se potvrzení od serveru, že byl balíček přijat

2.1.5 Funkce control_OACK()

Funkce, která kontroluje příchozí balíček OACK. Konkrétně kontroluje všechny options a hodnotu těchto možností. Pokud nějaká option chybí nebo se objeví nová, funkce vrátí chybu. Chyba bude také pokud se jakékoli hodnoty optiona neshodují. Výjimkou je 'tsize' v režimu RRQ. V tomto případě program počítá velikost budoucího souboru a zkontroluje pro něj volnou paměť.

2.2 Server

Server je schopen pracovat ve dvou režimech: 'octet' a 'netascii'. V případě odeslání režim neovlivní průběh programu, při zápisu souboru bude v závislosti na režimu vybrán odpovídající cyklus pro čtení dat ze souboru.

Server navíc podporuje jak klasickou komunikaci, tak rozšířenou o OPTS. V případě, že při obdržení první žádosti chybí některý z parametrů, je následně ignorován. Pokud se zobrazí parametr, který není nastaven, server odešle chybu a ukončí komunikaci.

První balíček přijímá hlavní proces, ale veškerá následná komunikace probíhá podřízeným procesem, který končí, když je celý soubor přijat nebo odeslán.

```
/**
 * Structure for input parametrs
*/
struct parametrs {
    int port = STANDART_PORT;
    char* root_dirpath;
};
```

Obrázek 2: Server struct parameters

2.2.1 Funkce get_parametrs()

Funkce rozděluje vstupní parametry do struktury určené pro parametry. Pokud chybí -p, server bude fungovat na standardním portu 69. Předpokladem je, že složka, kam budou soubory uloženy, bude posledním parametrem.

2.2.2 Funkce main_loop()

V této funkci se vytvoří a naváže Socket, načež se proces dostane do nekonečného cyklu, který bude čekat na příchozí pakety. Jakmile je takový balíček získán, hlavní proces vytvoří podřízený proces, který pokračuje v komunikaci se zákazníkem.

2.2.3 Funkce processing_request()

Funkce, kam se dceřiný proces po jeho vytvoření okamžitě dostane. Účelem funkce je příprava souboru, kontrola všech přijatých možností nebo zjištění, že chybí.

2.2.4 Funkce RRQ_handling()

Po přípravě a ověření všeho, co potřebujete, se do této funkce dostane podřízený proces, pokud potřebujete odeslat soubor. Hlavním účelem této funkce je zkontrolovat příchozí soubor ACK, pokud komunikace probíhá bez dalších možností a určení výběru cyklu pro čtení dat ze souboru. Více o těchto cyklech se můžete podívat níže.

2.2.5 Funkce RRQ_octet_loop()

V této funkci dochází ke čtení dat ze souboru v binárním režimu, který se skládá z 8 bitů. Při úspěšném čtení pomocí funkce fread()[1] vrátí 1, pokud soubor skončí, vrátí 0. Cyklus pokračuje, dokud tato funkce vrátí 1, tedy dokud není soubor dokončen.

2.2.6 Funkce RRQ_netascii_loop()

Cyklus je určen pro čtení dat ze souboru v režimu netascii. Cyklus pokračuje, dokud nebude přečten EOF. V případě, že je přečteno maximální množství dat, pak jsou tato data odeslána zákazníkovi a data samotná jsou stažena a začnou data číst dále. Po každém odeslání se očekává a kontroluje balíček ACK.

2.2.7 Funkce WRQ_handling()

Funkce pro záznam přijatých dat v režimu WRQ. Vzhledem k tomu, že rozdíl kvůli režimům je pouze v tom, v jakém módu otevřít soubor pro záznam, neexistují žádné další funkce závislé na režimu. Samotná funkce čeká na příchod dat a poté je přepíše do cílového souboru. Jakmile je v přicházejícím balíčku délka všech dat menší než maximální, pak se předpokládá, že jde o poslední balíček. A po zapsání těchto dat do souboru, komunikace končí a podřízený proces je ukončen.

2.2.8 Funkce open_file()

Funkce určená k otevření souboru ve správném módu. V případě, že je plánována komunikace RRQ a soubor chybí, bude odeslán balíček s chybovým kódem 1. Pokud bude komunikace WRQ a soubor již existuje, bude také odeslán balíček s chybovým kódem 6. Kromě toho, pokud neexistují žádné existující složky, kde by měl být soubor umístěn, budou v této funkci vytvořeny.

2.2.9 Funkce handling_opts() and funkce control_opts()

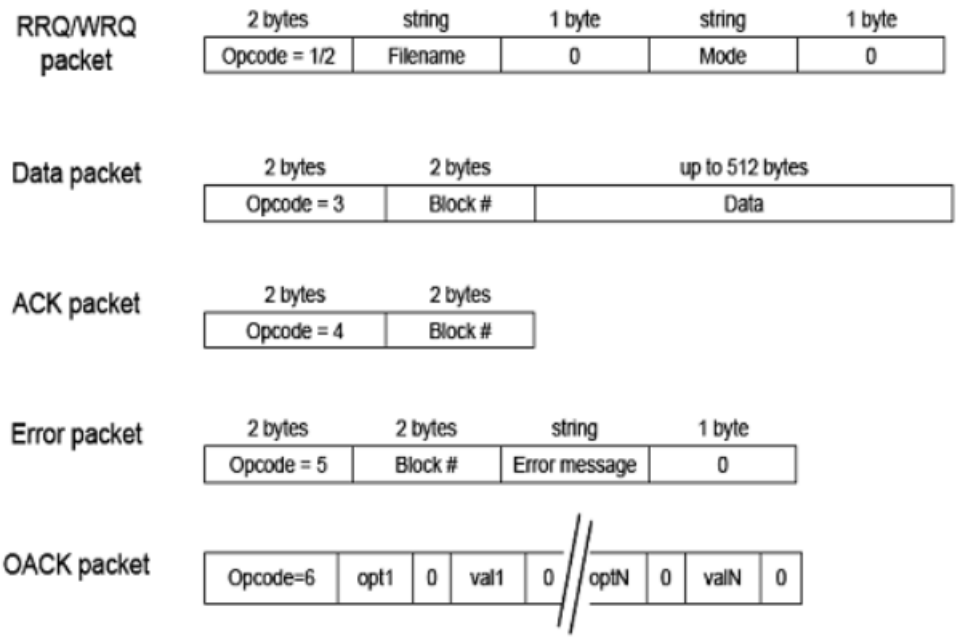
Funkce zajištění pracují s přicházejícími parametry spolu s prvním balíčkem o vyžádání komunikace. Pokud neexistují žádné parametry, funkce control_opts () vrátí 1. Pokud jsou nějaké parametry, vrátí se 2. Pokud byly odeslány neznámé možnosti, funkce vrátí 0. Dále bude v závislosti na výsledku této funkce spuštěna funkce handling_opts(), která bude konfigurovat komunikaci. Může tak spočítat Velikost souboru pomocí funkce fseek()[4], zkontrolovat dostatečnou volnou paměť pomocí sysconf()[3] nebo nakonfigurovat zásuvku čekací doby.

2.2.10 Funkce control_ACK()

Funkce určená k ověření přicházejícího balíčku ACK a jeho ověření. Pokud dojde k chybě, program bude ukončen, jinak bude program pokračovat ve své práci.

2.3 Packet

Soubor obsahuje funkce, které odesílají různé balíčky. Každá z funkcí odešle určitý typ balíčku, kterých je celkem šest. Strukturálně jsou všechny tyto funkce podobné, nejprve jde o vyplnění vyrovnávací paměti různými daty, která závisí na typu Pokémona, a poté jde o jeho odeslání.



Obrázek 3: Typy packetu

2.4 Common

Funkce v tomto souboru jsou určeny pro veřejné použití. Většina funkcí pomáhá převést běžné znaky na kód ASCII a zpět. Kromě toho má soubor funkce, které porovnávají běžný řádek s řádkem, který se skládá z ASCII a funkce, která převádí celý řádek ASCII na normální a Ambasador si zachovává jeho hodnotu. Kromě všech funkcí, které fungují s řádky, existuje jedna, která vytváří složky. Používá se na obou stranách k vytvoření cesty kam soubor uložit.

Literatura

[1] AmanSrivastava1. C fread() function. https://www.geeksforgeeks.org/fread-function-in-c//.

[2] Michael Kerrisk. getopt(3) — linux manual page. https://man7.org/linux/man-pages/man3/getopt.3.html/.

[3] Michael Kerrisk. Library functions manual. https://man7.org/linux/man-pages/man3/sysconf.3.html/.

[4] Michael Kerrisk. Posix programmer's manual. https://man7.org/linux/man-pages/man3/fseek.3p.html#:~:text=The%20fseek()%20function%20shall,set%20and%20fseek()%20fails./.

[5] G. Malkin and A. Harkin. Tftp blocksize option. https://datatracker.ietf.org/doc/html/rfc2348/, May 1998.

[6] G. Malkin and A. Harkin. Tftp option extension. https://datatracker.ietf.org/doc/html/rfc2347/, May 1998.

[7] G. Malkin and A. Harkin. Tftp timeout interval and transfer size options. https://datatracker.ietf.org/doc/html/rfc2349/, May 1998.

[8] K. Sollins. The tftp protocol (revision 2). https://datatracker.ietf.org/doc/html/rfc1350/, July 1992.