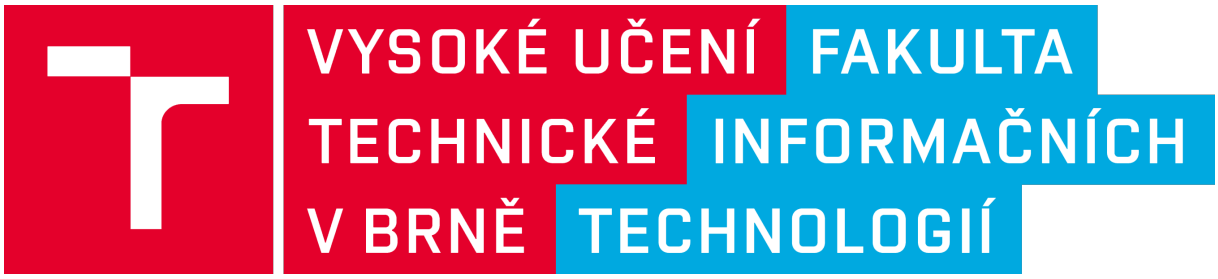

Dokumentace k projektu do předmětu ISA

TFTP Klient + Server

Martin Soukup
xsouku15

Obsah

1 Úvod

Cílem projektu je vytvořit klientskou a serverovou aplikaci pro přenos souborů prostřednictvím TFTP (Trivial File Transfer Protocol) podle specifikace RFC.

1.1 TFTP

TFTP je jednoduchý protokol pro přenos souborů přes počítačovou síť. Je vhodný pro situace, kdy je potřeba rychle a jednoduše přenášet soubory, například při konfiguraci síťových zařízení.

1.2 Options

TFTP umožňuje rozšíření pomocí volitelných options. Jsou to informace, které mohou být použity k modifikaci chování protokolu.

1.3 Rozšíření blocksize

Umožňuje specifikovat velikost bloku, který je přenášen po síti. Toto rozšíření umožňuje zvýšit efektivitu přenosu tím, že dovoluje přenos většího množství dat najednou. Standardní velikost přenášeného bloku je 512 bajtů, ale pomocí tohoto rozšíření může být upřesněno na hodnotu z intervalu <8, 65464> bajtů.

1.4 Rozšíření transfersize

Rozšíření transfersize umožňuje získat informaci o velikosti souboru před jeho stažením, nebo informovat server o velikosti souboru, který bude nahráván.

1.5 Rozšíření timeout

Timeout rozšíření umožňuje vyjednat časový úsek, po kterém bude odeslaný blok nebo potvrzení žádosti považováno za neúspěšné a dojde k opětovnému zaslání.

1.6 Módy přenosu

Existují dva módy přenosu v TFTP:

1. Netascii mode: Tento mód přenosu je určen pro přenos textových souborů mezi zařízeními s různými znakovými sadami.
2. Octet (Binary) mode: Data jsou přenášena v binárním formátu bez provedení žádných konverzí.

2 Návrh programu a implementace

Návrh spočívá v rozdělení na tři důležité části: Klient, Server a společná část utilit.

Klient je celý zpracován v jednom souboru, zatímco Server se dělí na hlavní část předávající argumenty třídě server. Utility jsou společná část pro klientskou a serverovou část, jedná se o výpisu a struktury.

2.1 Zpracování parametrů

Části klienta a serveru zpracovávají argumenty pomocí funkce **getopt()**, která je rozdělí do jednotlivých struktur specifikovaných v utilitách.

2.2 Klient

Klient po zpracování argumentů založí schránku a na IP adresu a port daný parametry programu odesílá žádost o sdílení souboru. Výchozí port pro TFTP je 69. Klient má povinně parametr cílového souboru, do kterého buď nahrává, nebo pokud je předán i nepovinný argument souboru pro stažení, tak do cílového souboru stahuje obsah vzdáleného.

2.3 Server

Server po spuštění očekává, ve výchozím stavu, na portu 69, pokud mu nebyl přidělen jiný port v rámci zpracování argumentů. Server čeká na příchozí žádost od klienta, po přijetí této žádosti se podle operačního kódu rozhodne ve funkci **handleTFTPRequest()** jestli je o stažení souboru nebo o nahrání souboru do kořenové složky serveru dané povinným argumentem.

2.3.1 Komunikace podle specifikace

Server načte a zpracuje požadavek, pomocí přetypování do struktury **TFTPRequest**. Server poté kontroluje zda-li je soubor předaný v požadavku skutečně v kořenové složce, pokud ne, posílá nazpět klientu paket s chybovou hláškou a kódem. Server se nadále chová podle rozšíření přijatých v požadavku, pokud žádná rozšíření nepřišla, začne stahování dat souboru, nebo se odešle paket přijetí a očekává se přijetí datových bloků od klienta. Pokud v průběhu komunikace přijde paket s chybou, komunikace se ruší a server očekává nový požadavek.

2.3.2 Zápis souboru

Pokud se začínají zasílat datové bloky, server je zpracuje převedením na strukturu **TFTPDataBlock**, tyto datové bloky hned po příchodu zapisují na do serveru přidaná data. Jelikož TFTP nemůže přepisovat již vytvořené soubory, kontroluje se jak jejich existence, tak jestli byly správně otevřeny.

3 Utility

Společná část pro klient i server obsahující důležité struktury a i výpis přijatých paketů.

3.1 Struktury

Server pro své argumenty má strukturu **s_parameters**, klient **c_parameters**. Každý typ paketu má vlastní strukturu (ERROR, DATA, ACK, …), každá z těchto struktur obsahuje operační kód, poté už jsou různé. Za zmínku stojí struktura **TFTPRequest**. Tato struktura obsahuje v sobě další strukturu **options**, tím ukládá rozšíření při odesílání požadavku na server. Poté struktura **OptionInfo**, která ukládá informace o rozšířeních a v rámci implementace serveru i klienta pomáhá kontrole jednotlivých rozšířeních, jejich výpisu a jejich zaslání v OACK.

3.2 Výpis

Pro výpis je speciální soubor **infowriter.cpp/hpp**, který sjednocuje výpis komunikace mezi klientem a serverem.

4 Spuštění

Server se spustí pomocí příkazu:

sudo ./tftp-server [-p port] root_dirpath

- **port** je nepovinný argument a ve výchozím stavu je nastaven na 69
- **root_dirpath** je povinný argument, který označuje kořenovou složku serveru

Klient se spustí pomocí příkazu:

./tftp-client -h hostname [-p port] [-f filepath] -t dest_filepath

- **hostname** je povinný argument, jméno nebo IPv4 adresa serveru
- **port** je nepovinný argument a ve výchozím stavu je nastaven na 69
- **filepath** je cesta k souboru, který bude stahován ze serveru, pokud není zadán, očekává se na stdin soubor pro nahrání.
- **dest_path** je povnný argument a označuje jméno, pod kterým se má soubor uložit

5 Zdroje

Sollins, K., "The TFTP Protocol (Revision 2)", STD 33, RFC 1350 (Online at https://datatracker.ietf.org/doc/html/rfc1350), MIT, July 1992.

Malkin, G., and A. Harkin, "TFTP Option Extension", RFC 2347 (Online at https://datatracker.ietf.org/doc/html/rfc2347), May 1998.

Malkin, G., and A. Harkin, "TFTP Blocksize Option", RFC 2348 (Online at https://datatracker.ietf.org/doc/html/rfc2348), May 1998.

Malkin, G., and A. Harkin, "TFTP Timeout Interval and Transfer Size Options", RFC 2349 (Online at https://datatracker.ietf.org/doc/html/rfc2349), May 1998.