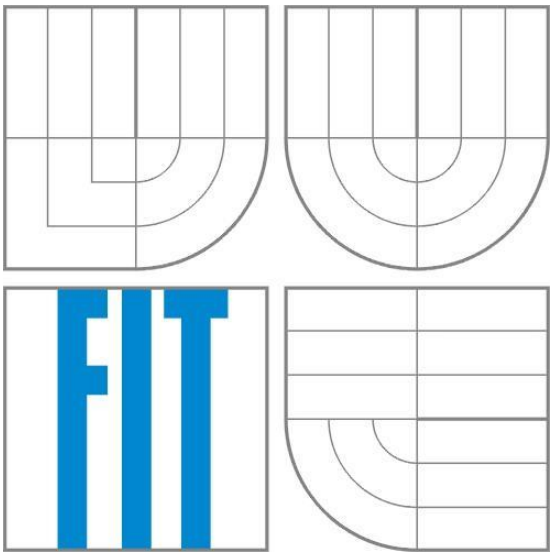
Vysoké učení technické v Brně

Fakulta informačních technologií



Dokumentace k projektu pro předmět ISA

Programování síťové služby

TFTP Klient

20. listopadu 2014

Autor:

Daniel Žůrek, xzurek12

Obsah

1. Úvod

Tento dokument popisuje návrh a implementaci protokolu síťové služby TFTP (Trivial File Transfer Protocol). Protokol využívá přenosových funkcí protokolu UDP a slouží pro jednoduchý přenos souborů mezi klientem a serverem. Nepodporuje žádný mechanismus ověřování ani šifrování a jako takový se nedoporučuje u služeb s přístupem na internet. V implementaci je využito knihoven pro práci s BSD sockety. Klient je implementován jako blokující, je tedy nutné počkat na dokončení operace. Program funguje jako konzolová aplikace, která ze standartního vstupu zpracuje příkaz a na standartní výstup v určeném formátu vypíše výstup operace.

2. Zadání projektu

V jazyce C implementovat TFTP klienta. Program po spuštění nabídne vlastní příkazovou řádku, která zpracovává zadané příkazy. Aplikace informuje o své činnosti výpisem na standartní výstup ve zvoleném formátu (časové razítko, rozbor zprávy, IP adresa serveru). Při chybě program vypíše hlášení na standartní chybový výstup. Program podporuje rozšíření blocksize option, timeout interval a transfer size option. Volitelné rozšíření multicast není implementováno. Podpora mail-módu není vyžadována.

3. Popis řešení

3. 1 Zpracování parametrů příkazové řádky

Ke zpracování parametrů příkazové řádky dochází na začátku programu zavoláním funkce `rozdelParametry()`, která provede zpracování zadaných parametrů. Jedná se o jednoduchý konečný automat, který se na základě načteného znaku rozhoduje o své další činnosti. Kontrola správné kombinace je vytvořena pomocí proměnných `param_x`, kde x označuje název parametru. Pokud je hodnota této proměnné větší než jedna, jedná se o nepovolenou kombinaci a program vypíše chybové hlášení. Parametry se mezi sebou můžou libovolně kombinovat, přičemž parametry `-d` a `-W/R` jsou povinné. Výpis nápovědy se provede po spuštění programu.

Po zpracování následuje kontrola správnosti každého parametru. Parametr musí splňovat formát, který se od něj očekává (např. `-a` [řetězec], `-t` [číslo] rozsah [1 - 255]). Pokud není formát dodržen, program vypíše chybové hlášení.

3. 2 Vytvoření socketu na základě IP adresy

Dle zadání má program podporovat formát adresy IPv4 i IPv6. Tuto skutečnost je nutné zohlednit při vytváření socketu. Pomocí funkce `getaddrinfo()` se naplní položka `ai_family` ve struktuře typu `addrinfo` a na základě této hodnoty (AF_INET – IPv4, AF_INET6 – IPv6) se vytvoří příslušná schránka typu SOCK_DGRAM.

Je-li socket úspěšně vytvořen, je nutné nastavit hodnotu timeout pro následný přenos (funkce `setsockopt()`). Vyprší-li tento interval při přijímaní/odesílání zprávy ze serveru, jedná se o chybu a spojení se serverem je ukončeno.

3. 3 Nalezení nejmenší hodnoty MTU

Pro nalezení nejmenší hodnoty MTU ze všech dostupných rozhraní je použita funkce `findMinMTU()`. Pomocí funkce `ioctl()` s parametrem SIOCGIFCONF se nejdříve zjistí celkový počet dostupných rozhraní. Následně se provede iterace přes tyto rozhraní a pro každé z nich se provede druhé volání funkce `ioctl()` tentokrát s parametrem SIOCGIFMTU, pro nalezení konkrétní hodnoty MTU. Tato hodnota se porovná s dosavadním nalezeným minimem a na základě výsledku porovnávání se rozhodne, zdali se jedná o novou minimální hodnotu.

Pokud klient zadá hodnotu parametru blocksize větší než je minimální hodnota MTU pro dané rozhraní, je zadaná hodnota parametru ignorována a serveru je odeslán blocksize option o velikosti minimální MTU.

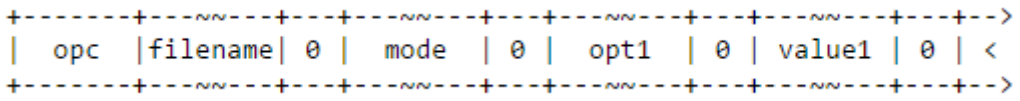
3. 4 Operace čtení (RRQ)

Protokol TFTP rozlišuje několik typů packetů, které lze odeslat nebo přijmout. Jednotlivé packety se odlišují na základě hodnoty v atributu opcode:

- 1 RRQ operace čtení
- 2 WRQ operace zápis

- 3 DATA datový packet
- 4 ACK acknowledgment packet
- 5 ERROR chybový packet
- 6 OACK option acknowledgment packet

Dle dostupných RFC dokumentů, začíná komunikace mezi klientem a serverem odesláním žádosti o čtení nebo zápis, přičemž operace čtení je identifikována hodnotou RRQ. Dále je nutné vytvořit packet následujícího formátu:



Obrázek 1 - Formát request packetu

Odeslání požadavku na server se provede funkcí `sendo()`. Server odpoví packetem typu OACK, ve kterém potvrdí nebo zamítne jednotlivé rozšiřujíc možnosti přenosu. Po zpracování tohoto packetu může klient začít přijímat data od serveru, přičemž odeslání ACK packetu se řídí přijmutím packetu s daty. Je vytvořen packet typu ACK, nastaví se příslušná hodnota block a packet je odeslán na server. Rozlišuje se přenos typu ‘octet‘ a ‘ascii‘. Při přenosu typu ‘ascii‘ je nutné převést konce řádků z CRLF na LF. Konec přenosu je identifikován podle velikosti zaslaných dat, která je menší než zadaný blocksize.

3. 4. 1 Zpracování OACK packetu

Zpracování zprávy v OACK packetu probíhá ve funkci `checkOptions()`. Jednotlivé atributy zprávy jsou odděleny znakem ‘\0‘. Funkce nahradí nuly znakem ‘#‘ a pomocí funkce `strtok()` rozdělí atributy zprávy do samostatných řetězců. V případě, že server nepotvrdí zadanou hodnotu blocksize, která však splňuje rozsah MTU na daném rozhraní, vypíše se upozornění a jako nový blocksize se použije hodnota navrhovaná serverem. Hodnota timeout se kontroluje analogicky k blocksize. Pokud server zamítne navrhovanou hodnotu, použije se hodnota navrhovaná serverem. Funkce dále vyhodnocuje hodnotu parametru file transfer size.

3. 5 Operace zápis (WRQ)

Formát packetu pro zápis je stejný jak pro operaci čtení (Obrázek 1), liší se však v hodnotě opcode (zápis - WRQ). Po přijmutí a zpracování OACK packetu (3. 4. 1), může klient začít posílat data na server. Je vytvořen packet typu DATA, naplní se daty, inkrementuje se hodnota atributu block a packet je odeslán. Konec přenosu je identifikován podle velikosti odesílaných dat, která je menší než zadaný blocksize.

3. 5. 1 Zjištění délky souboru před odesláním požadavku

Před samotným odesláním požadavku na server je nutné zjistit velikost zapisovaného souboru pro rozšíření transfer size option. Problém tohoto rozšíření nastává při módu přenosu typu ‘ascii‘. V tomto módu se převedou znaky konce řádků z LF na CRLF, přičemž server potřebuje znát velikost již převedeného souboru (aniž byla velikost souboru změněna). Funkce `findRealFileLenth()` vypočte požadovanou velikost souboru se změněnými konci řádků. Při každém nalezení znaku ‘\n‘ se připočte 1 k celkové velikosti souboru.

3. 6 Timeout a retransmit

Podle informací z RFC dokumentů je možné se po vypršení intervalu timeout, chybě odeslaní ACK nebo DATA packetu pokusit o znovu odeslání požadovaných dat. Tato možnost je řešena prostřednictvím proměnné `retry`, která je na začátku programu nastavena na hodnotu 10. Nastane-li jedna z výše uvedených chyb, dekrementuje se hodnota v této proměnné a pomocí příkazu `continue` se program pokusí o znovu provedení příslušného příkazu. Pokud je hodnota proměnné `retry` rovna 0, další retransmit se již neprovádí a program vypíše chybové hlášení.

4. Testování

Program byl testován na referenčním virtuálním stroji. Jako protistrana byla použita aplikace TFTPD32, která byla spuštěna v prostředí Windows 7 64bit. Hlavní důraz testování

byl kladen na správnost operací čtení a zápis. Jako vstupy jednotlivých operací byly použity soubory různých typů (textové dokumenty, obrázky, dokumenty typu pdf a krátké videa). Při testování byla ověřena funkčnost komunikace mezi klientem a serverem pomocí programu Wireshark.

5. Závěr

Na základě zadání byl vytvořen program TFTP klienta, který umožňuje zasílání a čtení souborů na požadovaný server. Klient podporuje přenosové módy typu ‘ascii‘ (netascii) a ‘octet‘ (binary). Jsou podporovány adresy typu IPv4 i IPv6. Jako rozšíření přenosu byly implementovány možnosti blocksize option, timeout interval a transfer size option. Během vypracování projektu došlo ke změně v zadání, když rozšíření multicast bylo označeno jako volitelné. Toto rozšíření nebylo implementováno z důvodů časové náročnosti.

6. Reference

1. IEN 133
2. RFC 1350
3. RFC 1785
4. RFC 2090
5. RFC 2347
6. RFC 2348
7. RFC 2349