VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ISA - Síťové aplikace a správa sítí

Projekt - TFTP Klient

15. novembra 2021

Timotej Ponek, xponek00

1 Úvod do problematiky

Cieľom projektu bolo vytvoriť TFTP klienta, ktorý bude vyhovovať 2. špecifikácií TFTP[1] s podporou rozšírení[2], menovite timeout[3], tsize[3], blocksize[4] a multicast[5].

TFTP (Trivial transfer file protocol) je protokol na prenos súborov medzi klientom a serverom. Podporuje zápis súboru na server a získanie súboru zo serveru. Okrem týchto dvoch jednoduchých operácií neposkytuje v základe nič viac, neposkytuje možnosť vypísať súbory dostupné na severi, veľkosť voľného dátového priestoru na disku a iné veci, ktoré by boli užitočné. Prenos začína Read alebo Write requestom na server. Server odpovedá ACK paketom, prípadne Error paketom ak nastala nejaká chyba. V prípade, že klientovi prišiel error paket, prenos sa ukončuje. Inak si klient uloží získané číslo portu serveru a pokračuje následovne:

- Pri read requeste klient pošle ack na 0 data paket a čaká na odpoveď serveru - prvý data paket. Klient pokračuje v posielaní ack na data paket s daným číslom až pokiaľ mu nepríde paket s dátami menšími ako 512B, ptm pošle ack na tento posledný paket a prenos sa ukončuje úspechom -¿ klient obdržal celý súbor.
- Pri write requeste klient pošle write request na server a čaká na ack. Klient ďalej posiela data pakety číslované od 1 na server a vždy čaká na ack od serveru. Ak ack nedostane, paket na ktorý nedostal ack by sa mal pokúsiť poslať znovu. Takto klient pokračuje v posielaní data paketov až kým neodošle celý súbor a nedostane na posledný data paket ack. V tomto prípade prenos končí úspechom.

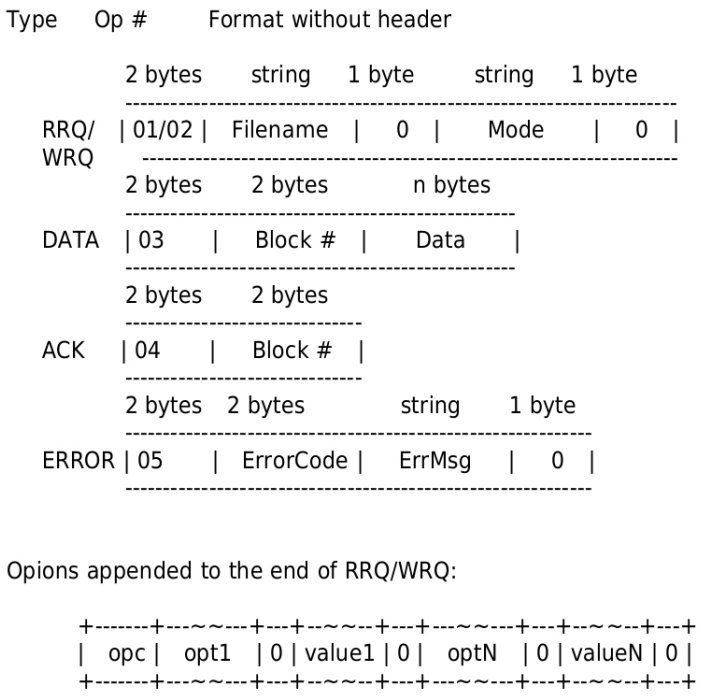
V oboch operáciach read a write, ak je obdržaný error paket v hociktorom momente, prenos je ukončený neúspechom. Naviac ak je od serveru obdržaný paket s nesprávnym (nedohodnutým) číslom portu, klient by mal na tento port poslať Error paket.

Rozšírenia (ktoré fungujú iba ak ich server podporuje):

- *multicast:* umožňuje klientovi získať od servera multicast adresu na ktorú bude počúvať, v prípade že je klient master tak je navyše zodpovvedný za odosielanie ACK paketov
- *tsize:* umožňuje klientovi získať od servera veľkosť súboru alebo poslať serveru veľkosť súboru, ešte pred začiatkom prenosu
- *timeout:* určuje dobu, ktorú bude server čakať na ACK alebo DATA paket od klienta
- *blocksize:* určuje veľkosť datovej časti DATA paketu, ktorú si pýta klient od serveru

TFTP pakety a ich popis:

opcode	**operation**
1	Read request (RRQ)
2	Write request (WRQ)
3	Data (DATA)
4	Acknowledgment (ACK)
5	Error (ERROR)
6	Option Acknowledgment (OACK)
8	Option Acknowledgment ERROR (OACK ERROR)



Obr. 1: Formát jednotlivých paketov

2 Implementácia

Logika programu bola inšpirovaná jednoduchým TFTP klientom fungujúcim iba pre read request od Sumit Jha[6]. Spracovanie argumentov prebieha cez unix funkciu getopt_long(), kde sa musí vstup najskôr rozdeliť na jednotlivé reťazce, ako delimiter sa používa medzera. Vstup je získavaný cez std::getline() ktorá získa vstup ako jeden reťazec.

Následne je vytvorený Write alebo Read request, podľa získaných argumentov, k nemu sú pridané špecifikované options z argumentov a request je poslaný serveru.

Pri kladnej odpovedi, teda ak nám nepríde ako odpoveď error packet, sa uloží port získaný od serveru (pretože requesty posielame na dobre známy tftp port 69) a ak ide o OACK packet, spracujú sa ešte získané optiony zo servera. Následne pri ACK a OACK pakete sa pokračuje potvrdením - poslaním ack paketu s číslom dátového bloku 0. Pri read requeste sa opakuje následujúca sekvencia: získaj paket od serveru, ak je to error paket, ukonči prenos, ak nevyhovuje/nezhoduje sa port, pošli error paket na daný port, inak zapíš dáta a pošli ack na získaný DATA paket s aktuálnym číslom dátového bloku. Toto všetko sa vykonáva až kým nepríde paket, ktorého dĺžka je menšia ako maximálna dĺžka paketu, potom je prenos súboru (sťahovanie súboru) ukončený.

Ak sme mali pritomnú option multicast, tak ak sme master, prenos pokračuje podobne ako by táto option nebola špecifikovaná. Inak klient zbiera pakety z multicast adresy a ukladá si číslo dátového paketu a dáta do internej štruktúry DataBlock, ktorá je zoskupená v c++ sete allDataBlocks zoradenom podľa čísla paketu, až pokiaľ sa celková veľkosť dát v allDataBlocks nerovná veľkosti súboru (získanej od serveru pomocou option tSize). V prípade že tSize nebola získaná, teda veľkosť súboru nie je nastavená, multicast môže správne prebehnúť iba ak sme od začiatku master, inak nemôže prebehnúť a prenos skončí s errorom.

Write request - znovu sa od serveru spracuje ACK alebo OACK paket, prípadne sa prenos ukončí ak ide o Error paket. Následne sa pokračuje v smyčke pokiaľ je celková veľkosť dát prenesených v data paketoch menšia ako veľkosť súboru. Dáta sa načítaju do pomocného bufferu (ten sa využije v prípade že by sa odosielaný data paket stratil), následuje smyčka ktorá prebieha až dokiaľ sa nezíska odpoveď od serveru a vytvorí sa data paket,

skopírujú sa dáta z pomocného bufferu do bufferu paketu a odošle sa paket. Následne sa čaká na odpoveď od serveru, ak príde nejaký error paket, data paket sa pošle znova.

Optiony, ktoré si pýta klient od servera sú vždy akceptované. Teda v prípade že nám server zamietne 'timeout' alebo 'blksize', použije sa defaultná hodnota a s ňou sa pokračuje ďalej v prenose, klient neskončí s errorom keď mu server zamietne hociktorú option.

3 Testovanie

3.1 Testovanie skenovania

Funkčnosť programu bola testovaná voči tftp klientovi: `https://github.com/reinerh/rtftp`

Bola otestovaná funkčnost blocksize, tsize a timeout pri read aj write requeste. Vyššie uvedený klient nepodporuje multicast, ten nebol otestovaný.

Manuálne som si vždy skontroloval či je súbor správne stiahnutý alebo odoslaný

```
root@Swift-SF314-52G:~/Documents/Skola/ISA/xponek00# ./tftp-client
<
-R -d luna -c netascii -t 40
[2021-16-11 22:07:25.200]       Transfered 331 B of 331 B
<
-R -d aven -c netascii
[2021-16-11 22:08:07.143]       Transfered 512 B of 3839 B
[2021-16-11 22:08:07.143]       Transfered 1024 B of 3839 B
[2021-16-11 22:08:07.143]       Transfered 1536 B of 3839 B
[2021-16-11 22:08:07.143]       Transfered 2048 B of 3839 B
[2021-16-11 22:08:07.143]       Transfered 2560 B of 3839 B
[2021-16-11 22:08:07.143]       Transfered 3072 B of 3839 B
[2021-16-11 22:08:07.143]       Transfered 3584 B of 3839 B
[2021-16-11 22:08:07.143]       Transfered 3839 B of 3839 B
<
-W -d FUSK -c netascii
[2021-16-11 22:09:15.557]       Transfered 512 B of 5579 B
[2021-16-11 22:09:15.557]       Transfered 1024 B of 5579 B
[2021-16-11 22:09:15.558]       Transfered 1536 B of 5579 B
[2021-16-11 22:09:15.558]       Transfered 2048 B of 5579 B
[2021-16-11 22:09:15.558]       Transfered 2560 B of 5579 B
[2021-16-11 22:09:15.558]       Transfered 3072 B of 5579 B
[2021-16-11 22:09:15.558]       Transfered 3584 B of 5579 B
[2021-16-11 22:09:15.558]       Transfered 4096 B of 5579 B
[2021-16-11 22:09:15.558]       Transfered 4608 B of 5579 B
[2021-16-11 22:09:15.558]       Transfered 5120 B of 5579 B
[2021-16-11 22:09:15.558]       Transfered 5632 B of 5579 B
```

```
root@Swift-SF314-52G:~/Documents/Skola/ISA/xponek00# ./tftp-client
<
-R -d aven -c octet -a 127.0.0.1,69
[2021-16-11 22:40:37.205]       Transfered 512 B of 3722 B
[2021-16-11 22:40:37.206]       Transfered 1024 B of 3722 B
[2021-16-11 22:40:37.206]       Transfered 1536 B of 3722 B
[2021-16-11 22:40:37.206]       Transfered 2048 B of 3722 B
[2021-16-11 22:40:37.206]       Transfered 2560 B of 3722 B
[2021-16-11 22:40:37.206]       Transfered 3072 B of 3722 B
[2021-16-11 22:40:37.206]       Transfered 3584 B of 3722 B
[2021-16-11 22:40:37.206]       Transfered 3722 B of 3722 B
<
-R -d luna -c octet -t 40 -m
[2021-16-11 22:41:52.512]       Transfered 327 B of 327 B
<
-R -d aven -c octet -t 40 -m
[2021-16-11 22:42:02.319]       Transfered 512 B of 3722 B
[2021-16-11 22:42:02.319]       Transfered 1024 B of 3722 B
[2021-16-11 22:42:02.319]       Transfered 1536 B of 3722 B
[2021-16-11 22:42:02.319]       Transfered 2048 B of 3722 B
[2021-16-11 22:42:02.319]       Transfered 2560 B of 3722 B
[2021-16-11 22:42:02.320]       Transfered 3072 B of 3722 B
[2021-16-11 22:42:02.320]       Transfered 3584 B of 3722 B
[2021-16-11 22:42:02.320]       Transfered 3722 B of 3722 B
<
```

Obr. 2: Výstupy z testovania

4 Zhodnotenie, nedostatky a obmedzenia

Program bol nedostatočne otestovaný čo sa týka IPv6 adries a multicastu, tieto veci sú implementované bez otestovania funkčnosti. Multicast bol otestovaný v prípade že klient je od začiatku master, vtedy prenos pokračuje ako pri normálnom read requeste. Netascii mód funguje pre všetky druhy súborov len vo write módu, v read módu spracováva správne len textové súbory.

Literatura

[1] [online], [vid. 2021-11-15]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc1350

[2] [online], [vid. 2021-11-15]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc2347

[3] [online], [vid. 2021-11-15]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc2349

[4] [online], [vid. 2021-11-15]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc2348

[5] [online], [vid. 2021-11-15]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc2090

[6] [online], [vid. 2021-11-25]. Dostupné z: https://www.linkedin.com/pulse/tftp-client-implementa