SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

UDP komunikátor

Martin Nemec
FIIT STU
Počítačové a komunikačné siete
Cvičenie: Piatok 8:00

Obsah

1 Zadanie

Navrhnite a implementujte program s použitím vlastného protokolu nad protokolom UDP (User Datagram Protocol) transportnej vrstvy sieťového modelu TCP/IP. Program umožní komunikáciu dvoch účastníkov v lokálnej sieti Ethernet, teda prenos textových správ a ľubovoľného súboru medzi počítačmi (uzlami).

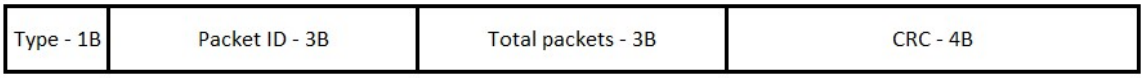
Program bude pozostávať z dvoch častí – vysielacej a prijímacej. Vysielací uzol pošle súbor inému uzlu v sieti. Predpokladá sa, že v sieti dochádza k stratám dát. Ak je posielaný súbor väčší, ako používateľom definovaná max. veľkosť fragmentu, vysielajúca strana rozloží súbor na menšie časti - fragmenty, ktoré pošle samostatne. Maximálnu veľkosť fragmentu musí mať používateľ možnosť nastaviť takú, aby neboli znova fragmentované na linkovej vrstve.

Ak je súbor poslaný ako postupnosť fragmentov, cieľový uzol vypíše správu o prijatí fragmentu s jeho poradím a či bol prenesený bez chýb. Po prijatí celého súboru na cieľovom uzle tento zobrazí správu o jeho prijatí a absolútnu cestu, kam bol prijatý súbor uložený. Program musí obsahovať kontrolu chýb pri komunikácii a znovuvyžiadanie chybných fragmentov, vrátane pozitívneho aj negatívneho potvrdenia. Po prenesení prvého súboru pri nečinnosti komunikátor automaticky odošle paket pre udržanie spojenia každých 5-20s pokiaľ používateľ neukončí spojenie. Odporúčame riešiť cez vlastne definované signalizačné správy.

2 Návrh

2.1 Návrh hlavičky vlastného protokolu

Veľkosť mojej hlavičky je 11B. Hlavička protokolu sa skladá z následujúcich častí:



Type - 1B: Type označuje o aký typ packetu sa jedná. Môžu byť následovné:
- 0 - nadzviazanie komunikácie zo strany klienta
- 1 - potvrdenie komunikácia zo strany servera
- 2 - dátový packet od klienta
- 3 - prijmutie packetu prebehlo v poriadku
- 4 - prijmutie packetu neprebehlo v poriadku
- 5 - keep alive packet zo strany klienta

Packet ID - 3B:

Packet ID je poradové číslo daného packetu. 3B z dôvodu že v zadaní je napísané, aby sme vedeli poslať aj súbor o veľkosti 2MB, a keďže veľkosť fragmentácie sa môže voliť - môže byť aj 1B, tak môže byť ID až do 2 000 000. Číslo 2M v hexadecimalnom tvare je 1E8480 - to znamená že potrebujeme 3B.

Total packets - 3B:

Total packets je číslo, ktoré znázorňuje celkové množstvo packetov. Rovnaký prípad ako v Packet ID, taktiež môže byť až 2 000 000 packetov.

CRC - 4B:

Pre crc som vyčlenil v hlavičke 4B, pretože funkcia crc32 používa 32 bitov čo sú 4 bajty.

Data - 1461B:

Dáta môžu byť o veľkosti 1461 bytov.
K tomuto výpočtu som došiel na základe: 1500 - IP_header - UDP_header - MY_header

2.2 CRC metóda

V programe mám v pláne využívať knižnicu zlib, z ktorej by som chcel použiť funkciu crc32. Na to, aby som ju mohol použiť v krátkosti opíšem funkčnosť crc.

Najprv si zoberieme dáta, na ktoré chceme uplatniť crc. Tieto dáta si premeníme na bity. K vstupu pridáme 32 núl, keďže využívame crc32. Následne ideme urobiť operáciu XOR s polynomom. Pre crc32 je to 1 0000 0100 1100 0001 0001 1101 1011 0111. Po vykonaní operácie XOR znovu použijeme ten istý polynóm, posunieme ho o jeden bit do prava a znovu urobíme XOR. Takto pokračujeme až na koniec, kde výsledkom je crc.

2.3 ARQ metóda

Stop-and-wait ARQ Protocol

Vybral som si túto metódu, pretože mi prišla jednoduchá na pochopenie a implementovanie.

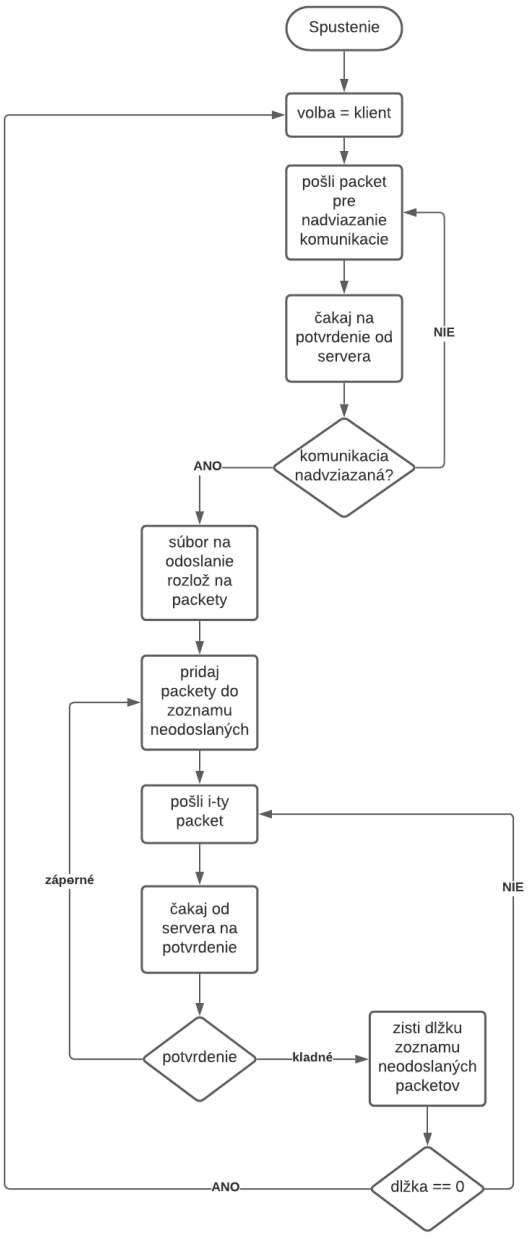
Klient pošle packet a čaká od servera potvrdenie. Ak packet dorazí v poriadku server pošle kladné potvrdenie. Ak packet dorazí a nie je v poriadku pošle klientovi zaporné potvrdenie a klient daný packet potom pošle znova. Môže nastať, že na server nepríde packet, v takom prípade by server čakal donekonečna, kým by neprišiel packet. Tým pádom, ak do niekoľkých sekúnd napr. 10s, nepríde na server packet, tak server pošle záporné potvrdenie a klient potom pošle daný packet opätovne.

2.4 Keep alive metóda

Ak aktuálne klient neposiela packety, tak každych 5s pošle keep alive packet, aby sa spojenie neprerušilo. Ak server nedostane do 5s žiaden packet spojenie sa preruší.

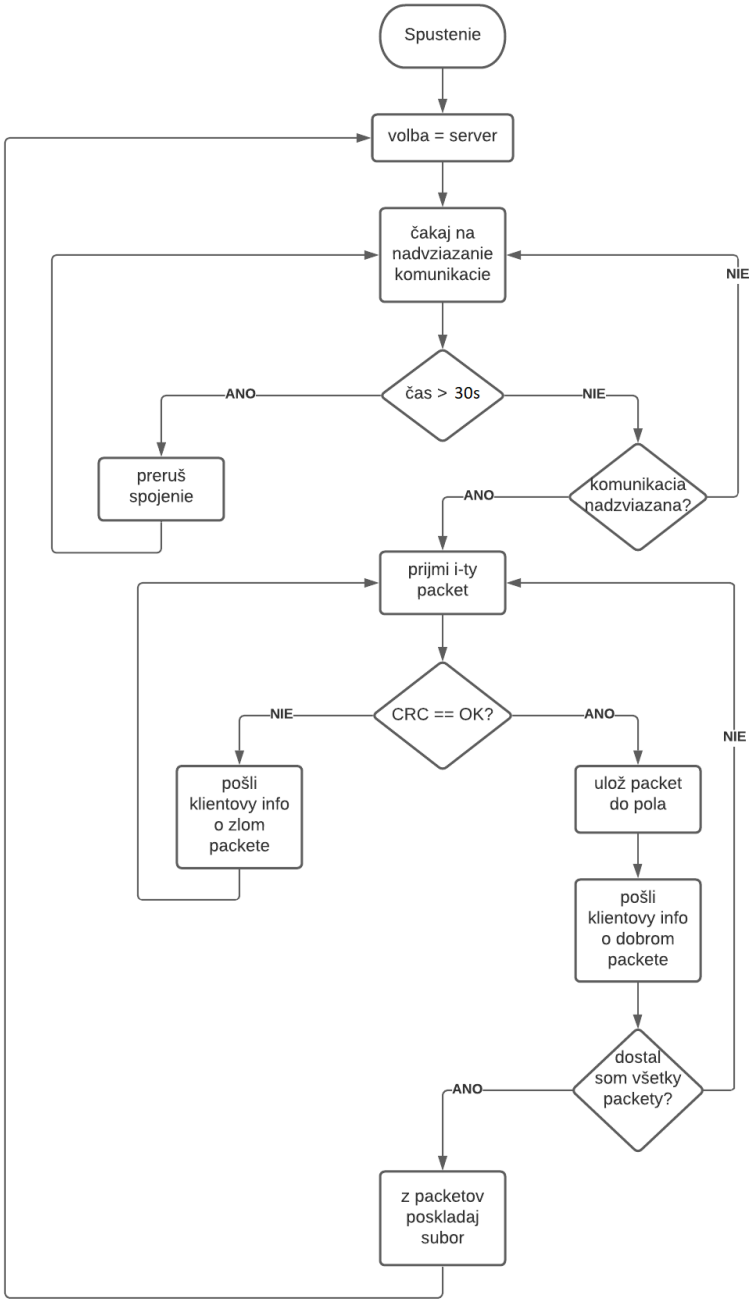
2.5 Diagramy

2.5.1 Klient diagram



Klient diagram

2.5.2 Server diagram



Server diagram

3 Zmeny oproti návrhu

Jediné zmeny oproti návrhu nastali v hlavičke môjho protokolu v časti Type.
Nové rozloženie typu hlavičky:

0 - nadviazanie komunikácie zo strany klienty
1 - potvrdenie komunikácie od servera
2 - správa od klienta
3 - súbor od klienta
4 - packet došiel v poriadku
5 - došiel poškodený packet
6 - packet bol stratený
7 - packet pre výmenu rolí medzi serverom a clientom
8 - keep alive packet

4 Prostredie

Implementačné prostredie

Za implementačné prostredie som si vybral Python, pretože neviem si predstaviť robiť toto zadanie napríklad v C. Python poskytuje množštvo knižníc a funkcií, ktoré zjednodušujú prácu na danom zadaní.
Knižnice, ktoré som využíval:

- math - na zaokruhľovanie fragmentov na hor
- os - pre nájdenie absolútnej cesty k súborom
- socket - pre prácu so socketmi
- threading - pre využívanie threadu pre keep alive
- time - pre stopovanie času, pri keep alive
- zlib - z tejto knižnice som využil funkciu crc32()

Používateľské prostredie
Pri spustení programu je potrebné zvoliť či dané zariadenie chceme používať ako Client alebo ako Server

```
Program starting...
This PC IP address: 192.168.0.103
Client mode - 1
Server mode - 2
Exit - 0
```

Po zvolení klienta je potrebné zadať IP adresu a port servera, s ktorým chceme nadviazať komunikáciu. Po nadviazaní komunikácie máme ďalšie možnosti, či chceme posielať súboru alebo správu.

```
Toto zariadenie je používané ako CLIENT

Zadaj IP kam chceš posielať súbory: 192.168.0.104
Zadaj port: 5005
Nadviazane spojenie s 192.168.0.104
Poslať správu - 1
Poslať subor - 2
Menu - 0
```

Po zvolení servera sa čaká na nadviazanie komunikácie zo strany klienta. Ak sa komunikácia nadviaže server už len čaká na príjmanie packetov.

```
Toto zariadenie je používané ako SERVER

Nadviazane spojenie s 192.168.0.103
```

5 Implementácia

5.1 Opis hlavných častí programu

Program je rozdelený na 2 veľké časti. Jedna časť je pre klienta a druhá časť je pre server. Niektoré funkcie na vytváranie packetov alebo na dekódovanie packetov majú spoločné.

5.2 Keep alive

Keep alive sa vždy automaticky zapína na klientovi ak so serverom nadviaže komunikáciu. Každých 5 sekúnd posiela na server keep alive packety. Keep alive sa vypne vtedy ak klient chce posielať súbor alebo správy. Po odoslaní všetkých packetov sa keep alive znovu zapne. Na server prichádzajú keep alive packety. Ak sa klient vypne a keep alive packety sa prestanú posielať, server sa po 30 sekundách odpojí od komunikácie a vypne sa.

5.3 Stop and wait ARQ

Pri tomto arq program funguje tak ako som písal v návrhu. Klient posiela packety a server ich kontroluje. Ak je crc v poriadku, tak server odošle pozitívne arq klientovi a ten následne pošle další packet. Ak server detekuje chybu v dátovej časti pošle na server packet s tým že došiel zlý packet a vyžiada si znovu poslať daný packet. V prípade že sa packet stratí, respektíve ho server očakáva ale klient ho neodošle tak po 5 sekundách si ho vyžiada od klienta aby ho poslal.

5.4 Zmena rolí

Ak klient sa rozhodne vymeniť rolu so serverom, tak najprv klient pošle inicializačný packet na server aby mu dal info o tom, že sa má zmeniť na klienta. Po tomto odoslanom packet sa klient prepne na server a server sa prepne na klienta. Vypne sa keep alive a tak isto sa preruší komunikácia. Takže je potrebné znovu nadviazať komunikáciu a môže sa pokračovať ďalej už ako si zvolí používateľ.

6 Záver

Na záver by som zhodnotil toto zadanie. Tento udp komunikátor sa mi páčil oveľa viac ako predošlé zadanie. Naopak od predošlého ma toto zadanie bavilo. Oceňujem, že v zadaní nie je všetko striktne napísané ako to bolo v predošlom zadaní a veľmi veľa veci je na nás ako si to urobíme.