**Slovenská technická univerzita v Bratislave**

Fakulta informatiky a informačných technológií

**Počítačové a komunikačné siete**

**Zadanie č. 2**

**UDP komunikátor**

Autor: Marek Smutný

Sk. Rok: ZS 2022/2023

Cvičenie: štvrtok 16:00, Ing. Kristián Košťál, PhD.

**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 zapnutí programu, komunikátor automaticky odosiela paket pre udržanie spojenia každých 5s pokiaľ používateľ neukončí spojenie ručne. Odporúčame riešiť cez vlastne definované signalizačné správy a samostatný thread.

**Návrh Protokolu**

Typy správ využívané vysielačom:

|  |
| --- |
| Typ správy |

**Štruktúra hlavičky pre 1, 2, 3:**

Typ správy (1B)

1. Inicializácia spojenia (0000 0001 bin)

2. Keep Alive (0000 0010 bin)

3. Zmena vysielania (0000 0011 bin)

|  |  |  |
| --- | --- | --- |
| Typ správy | Počet fragmentov | Prípona súboru |

**Štruktúra hlavičky pre 4, 5:**

Typ správy (1B), Počet fragmentov (2B)

4. Informačný paket text – vysielač odošle predtým ako začne posielať dátové pakety s textom

(0000 0100 bin)

5. Informačný paket súbor – vysielač odošle predtým ako začne posielať dátové pakety so súborom

(0000 0101 bin)

|  |  |  |  |
| --- | --- | --- | --- |
| Typ správy | Číslo fragmentu | CRC | Dáta |

**Štruktúra hlavičky pre 6, 7:**

Typ správy (1B), Číslo fragmentu (2B), CRC(4B), Dáta = 1500B – 8B – 20B – 7B = 1465B (1500B je maximálna veľkosť dát, ktorú je možné poslať bez fragmentovania na linkovej vrstve)

6. Dátový paket text (0000 0110 bin)

7. Dátový paket súbor (0000 0111 bin)

Typy správ využívané prijímačom:

|  |  |
| --- | --- |
| Typ správy | Číslo fragmentu |

**Štruktúra hlavičky pre ACK a ERR:**

Typ správy (1B), Číslo fragmentu (2B)

1. ACK - potvrdenie spojenia, doručenia fragmentu, zmeny vysielania (0000 0001 bin)

2. ERR – dáta prišli poškodené (0000 0010 bin)

**ARQ metóda**

Pre moje riešenie som si vybral Stop and wait ARQ metódu. Táto metóda funguje tak, že vysielač pošle packet s dátami a potom čaká na potvrdenie od prijímača. V prípade, že príde kladná odozva, posiela ďalší fragment v poradí. Ak prijímačovi príde poškodený paket, pošle zápornú odozvu (error) a vysielač následne znovu odošle posledný odoslaný fragment. Nevýhoda tejto metódy je jej rýchlosť, vysielač musí čakať na odozvu a až potom môže odoslať ďalší fragment.

**Metóda na udržanie spojenia**

Po nadviazaní spojenia bude vysielač každých 5 sekúnd posielať Keep Alive paket. Taktiež po nadviazaní spojenia sa na prijímačovi spustí timer, ktorý ak každých 30 sekúnd nedostane Keep Alive paket, spojenie preruší. Taktiež ak vysielač nedostane potvrdenie po odoslaní Keep Alive paketu, prijímač pravdepodobne ukončil spojenie a taktiež ukončí spojenie.

**Metóda kontrolnej sumy CRC**

Na výpočet checksum budem používať metódu crc32() z knižnice zlib. Táto funkcia využíva 32 bitový polynóm: + + + + + + + + + + + + + 1

Tento polynóm vyzerá v binárnom formáte takto: 1 0000 0100 1100 0001 0001 1101 1011 0111

Algoritmus na vypočítanie crc:

1. Obráť vstup

2. Na koniec pridaj 32 núl

3. Urob XOR s 0xFFFFFFFF = 1111 1111 1111 1111 1111 1111 1111 1111

4. Urob XOR s polynómom ak je na začiatku 1, ak nie posuň medzivýsledok doľava

5. Opakuj krok 4. kým prvých x bitov nie je 0, x = počet bitov vstupu na začiatku

6. Urob XOR s 0xFFFFFFFF

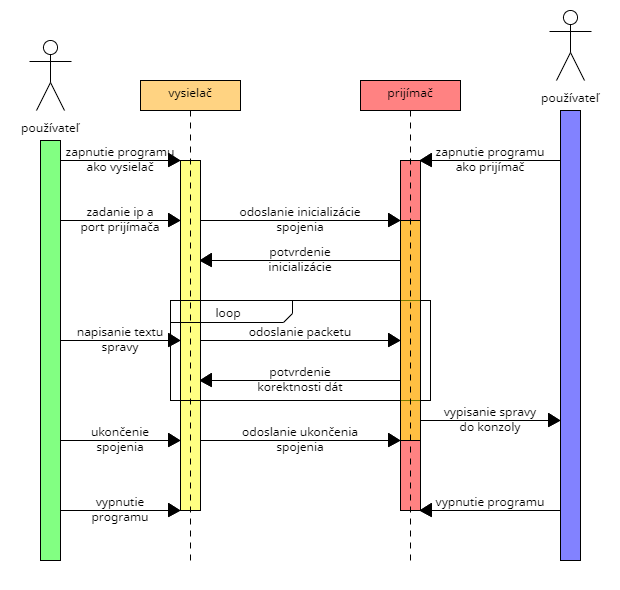
8. Obráť výsledok

Výpočet crc písmena „a“.

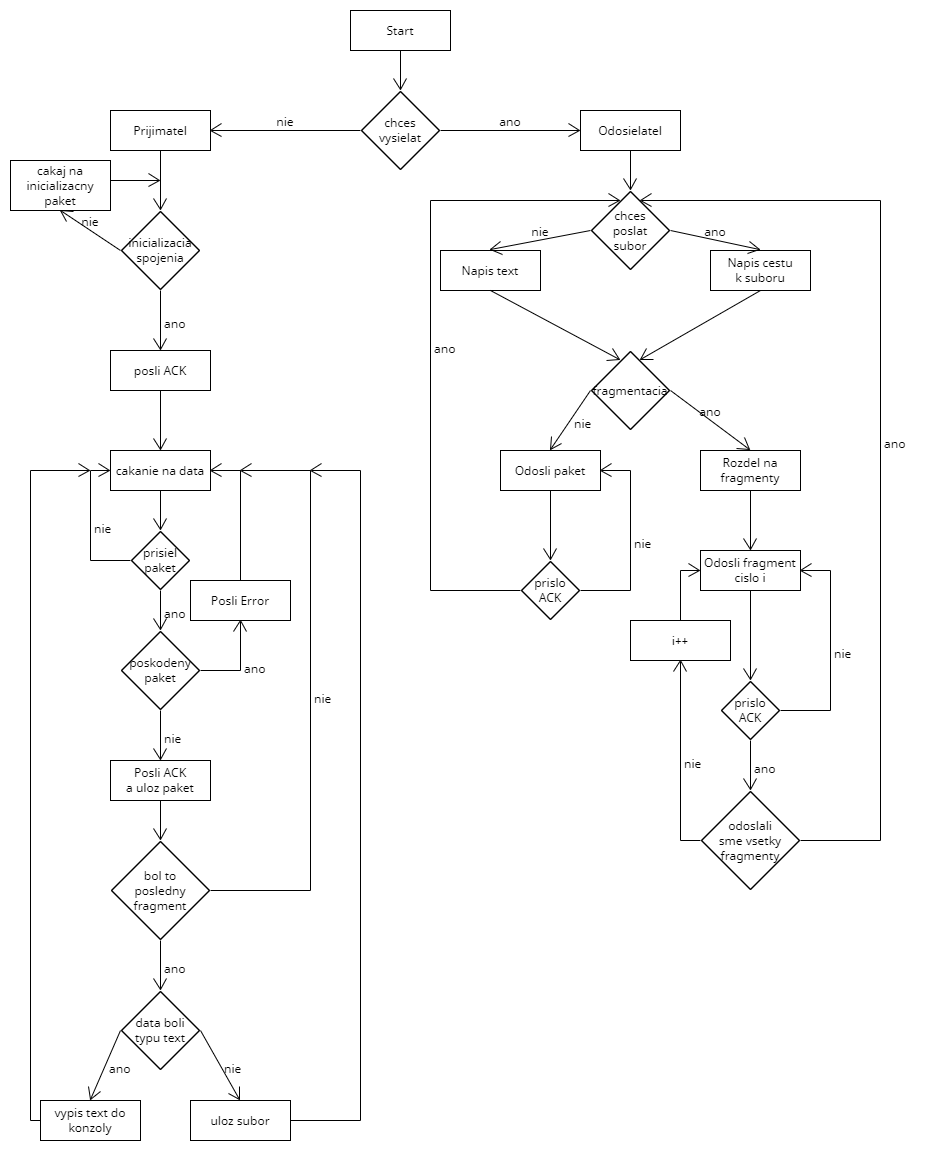


**Diagram spracovávania komunikácie**

Sekvenčný diagram:



Flowchart:



**Zmeny v porovnaní s návrhom**

Po konzultácii s cvičiacim, som vyskúšal viaceré funkcie crc a počas toho, som zistil, že crc32 z knižnice zlib nemusí byť najlepšia voľba. Nakoniec som vo svojom riešení použil funkciu crc\_hqx z knižnice binascii. Táto funkcia využíva CRC-CCITT polynóm (10001000000100001), čo znamená, že v porovnaní so zlib.crc32 je výsledný zvyšok menší, konkrétne 2 bajty. Tým pádom sa uvoľnia 2 bajty, ktoré môžeme využiť na prenos dát. Toto bol jeden z dôvodov, že som si zvolil práve crc\_hqx z knižnice binascii.

Taktiež v porovnaní s návrhom pribudla funkcionalita zmeny vysielania zo strany prijímača. V prípade, že používateľ na strane prijímača sa rozhodne pre zmenu roly (swap) obidve strany si ju vymenia. Podrobnejšie v časti Dôležité časti kódu.

**Užívateľské rozhranie**

Používateľ a program komunikujú výlučne pomocou konzoly. Program vypisuje inštrukcie na správne ovládanie programu a čaká na vstup od používateľa.

Obrázok

Obrázok, na ktorom je text

Automaticky generovaný popis

Na obrázku vyššie vidíme, že používateľ si vybral rolu prijímača, nastavil port, nadviazalo sa spojenie a prišiel mu súbor bash.txt, ktorý uložil do priečinka k main.py.

Obrázok

Obrázok, na ktorom je text

Automaticky generovaný popis

Na druhom obrázku vidíme, že používateľ si zvolil rolu vysielača, zadal ip prijímača a port. Následne si zvolil, že chce poslať súbor, napísal jeho názov a veľkosť fragmentu.

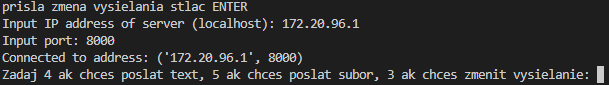
Obrázok

Obrázok, na ktorom je text

Automaticky generovaný popis

Na treťom obrázku vidíme, že vysielač zmenil vysielanie, spojenie sa uzatvorilo a nakoniec zadal port, na ktorom bude počúvať ako prijímač a vytvorilo sa nové spojenie.

Obrázok



Na štvrtom obrázku vidíme zmenu vysielania z pohľadu prijímača. Program mu oznámil, že prišla zmena vysielania, následne zadal ip prijímača a port a vytvorilo sa spojenie.

**Kontrola vo Wiresharku**

Keep alive správy:

Obrázok

Obrázok, na ktorom je text

Automaticky generovaný popis

Posielanie súboru:

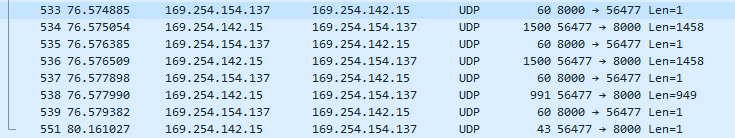
Obrázok

Obrázok, na ktorom je stôl

Automaticky generovaný popis

Na obrázku 6 môžeme vidieť začiatok procesu posielania fragmentov súboru. Prvý packet je informácia pre prijímač, že mu príde súbor. Následne sa posielajú všetky fragmenty súboru.

Obrázok

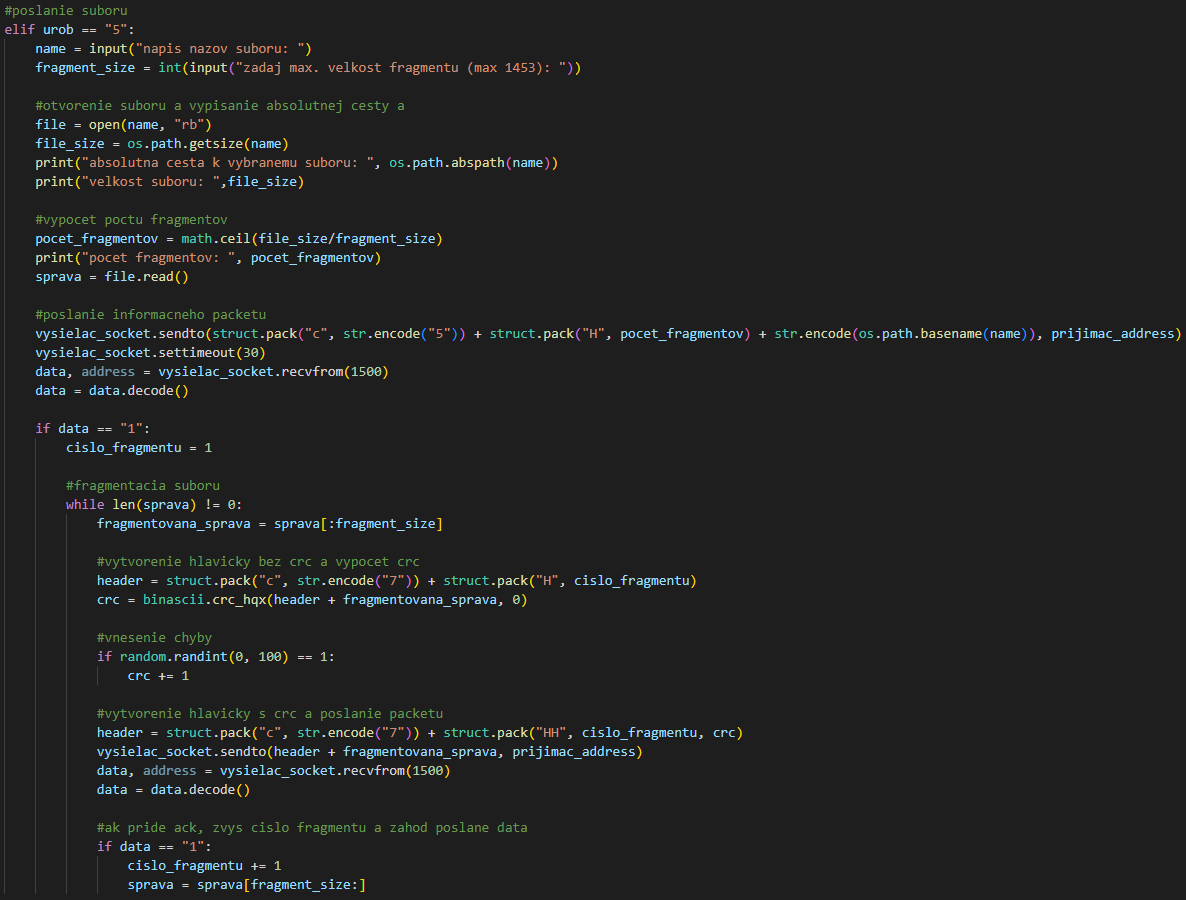


Na obrázku 7 vidíme posledné fragmenty súboru (posledný má 991 bajtov) a jednu správu keep alive.

**Dôležité časti kódu**

**Fragmentácia súboru a odosielanie fragmentov:**

Obrázok



Táto časť kódu (obrázok 8) sa nachádza vo funkcii vysielac. V prípade, že používateľ na strane vysielača zvolí možnosť odoslať súbor, program si vypýta názov súboru a maximálnu veľkosť fragmentu. Do názvu súboru je možné zadať aj absolútnu cestu k súboru alebo v prípade, že používateľ zadá iba názov, súbor sa musí nachádzať v rovnakom priečinku ako kód programu. Následne program vypočíta počet fragmentov a odošle informačný packet s počtom fragmentov a názvom súboru. Ak od prijímača príde potvrdenie, fragmentuje prečítaný súbor, počíta crc, vnáša chyby a odosiela fragmenty vždy, keď dostane potvrdenie od prijímača o predošlom fragmente. Šanca, že program vnesie chybu je 1:101 a v podstate je to zmena crc hodnoty.

**Keep alive:**

Obrázok



Obrázok

Obrázok, na ktorom je text

Automaticky generovaný popis

Vytvorenie threadu pre keep alive (obrázok 9) sa nachádza vo funkcii vysielac\_login. Tento thread sa vytvorí po úspešnom nadviazaní spojenia. Funkcia keep\_alive (obrázok 10) každých 5 sekúnd posiela keep alive packet a kontroluje, či prijímač náhodou nevyžiadal zmenu role (swap).

**Prijatie fragmentov a uloženie súboru:**

Obrázok

Obrázok, na ktorom je text, monitor, snímka obrazovky

Automaticky generovaný popis

Táto časť kódu (obrázok 11) sa nachádza vo funkcii prijimac. V prípade, že prijímač dostane packet typu 5 (informačný packet súbor) vypíše názov súboru a počet fragmentov a odošle potvrdenie. Následne v loope prijíma fragmenty súboru a keep alive správy. Po prijatí fragmentu skontroluje crc a v prípade, že je správne, vypíše číslo fragmentu a jeho veľkosť nakoniec uloží dáta do listu cela\_sprava. Potom ako prijme posledný fragment, používateľ môže zadať absolútnu cestu adresára určeného na uloženie súboru alebo nemusí zadať nič a súbor sa uloží do adresára, v ktorom sa nachádza kód programu. Nakoniec vypíše absolútnu cestu k uloženému súboru, jeho veľkosť a počet prijatých fragmentov.

**Výmena role na strane prijímača:**

Obrázok

Obrázok, na ktorom je text

Automaticky generovaný popis

Funkcia swap\_thread (obrázok 12) zapína a vypína thread, na ktorom beží funkcia listen\_swap. Táto funkcia obsahuje input. Keď používateľ na strane prijímača napíše „y“, prijímač nebude na keep alive správy odpovedať potvrdením „1“ ale „3“, čo spôsobí, že vysielač taktiež zmení rolu. Zmena roly je možná aj na strane vysielača ale toto riešenie je jednoduché a nezaslúži si dostať sa do dokumentácie.😊