SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Komunikácia s využitím UDP protokolu

Adrián Vančo
ID: 103171
Predmet: Počítačové a komunikačné siete

Zadanie úlohy

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 znovu vyžiadanie 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.

Program musí mať nasledovné vlastnosti (minimálne):

1. Program musí byť implementovaný v jazykoch C/C++ alebo Python s využitím knižníc na prácu s UDP socket, skompilovateľný a spustiteľný v učebniach. Odporúčame použiť python modul *socket,* C/C++ knižnice *sys/socket.h* pre linux/BSD a *winsock2.h* pre Windows. Iné knižnice a funkcie na prácu so socketmi musia byť schválené cvičiacim. V programe môžu byť použité aj knižnice na prácu s IP adresami a portami:
   arpa/inet.h
   netinet/in.h
2. Program musí pracovať s dátami optimálne (napr. neukladať IP adresy do 4x int).
3. Pri posielaní súboru musí používateľovi umožniť určiť cieľovú IP a port.
4. Používateľ musí mať možnosť zvoliť si max. veľkosť fragmentu.
5. Obe komunikujúce strany musia byť schopné zobrazovať:
   a. názov a absolútnu cestu k súboru na danom uzle,
   b. veľkosť a počet fragmentov.
6. Možnosť simulovať chybu prenosu odoslaním minimálne 1 chybného fragmentu pri prenose súboru (do dátovej časti fragmentu je cielene vnesená chyba, to znamená, že prijímajúca strana deteguje chybu pri prenose).

Adrián Vančo
ID: 103171
Predmet: Počítačové a komunikačné siete

7. Prijímajúca strana musí byť schopná oznámiť odosielateľovi správne aj nesprávne doručenie fragmentov. Pri nesprávnom doručení fragmentu vyžiada znovu poslať poškodené dáta.
8. Možnosť odoslať 2MB súbor a v tom prípade ich uložiť na prijímacej strane ako rovnaký súbor, pričom používateľ zadáva iba cestu k adresáru kde má byť uložený.

Odovzdáva sa:

1. Návrh riešenia
2. Predvedenie riešenia v súlade s prezentovaným návrhom

Program musí byť organizovaný tak, aby oba komunikujúce uzly mohli prepínať medzi funkciou vysielača a prijímača bez reštartu programu - program nemusí (ale môže) byť vysielač a prijímač súčasne. Pri predvedení riešenia je podmienkou hodnotenia schopnosť doimplementovať jednoduchú funkcionalitu na cvičení.

Adrián Vančo
ID: 103171
Predmet: Počítačové a komunikačné siete

Analýza

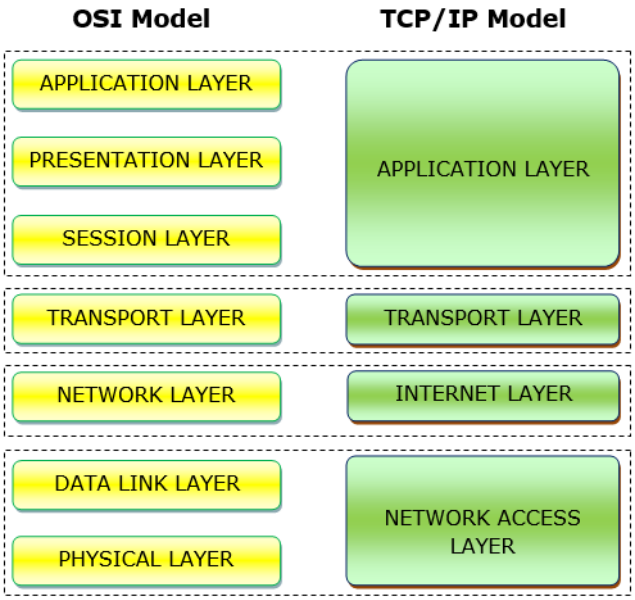
Program bude vyhotovený v jazyku **Python** vo verzii 3.8.6 s python modulmi **socket**, **struct, threading** a prípadne ďalšími užitočnými modulmi.
Ovládanie bude realizované cez konzolu.

Sieťové modely RM OSI a TCP/IP

Komunikácia medzi jednotlivými uzlami siete vyžaduje množstvo hardvérových a softvérových prostriedkov. Riadenie komunikácie predstavuje rozsiahly problém (napr. od el. signálov, káblov a konektorov až po samotný prenos súborov). Ako najlepšie riešenie sa ukázalo **rozloženie tohto problému na menšie časti**, do hierarchicky usporiadaných **vrstiev**.

Každá vrstva má presne definovaný spôsob komunikácie so susednou nižšou a vyššou vrstvou. Vyššia vrstva je vždy **žiadateľom** o vykonanie nejakej služby, nižšia vrstva je **poskytovateľom** služby pre vyššiu vrstvu. Komunikácia v sieti prebieha medzi dvoma uzlami, pritom medzi sebou komunikujú rovnocenné vrstvy (P2P komunikácia). Rovnocenné vrstvy komunikujú medzi sebou na základe presne dohodnutých pravidiel, **protokolov**.

Súbor pravidiel, procedúr a formátov pre komunikáciu tvorí **komunikačný protokol**. Každá vrstva obsahuje aspoň jeden protokol. **Výhodou vrstvového modelu** je, že sa dá nevyhovujúci protokol nahradiť iným protokolom, čo však neovplyvní funkciu ostatných vrstiev (napr. kvôli nedostatku verejných IP adries sa postupne protokol IPv4 – 32 bitová IP adresa, nahrádza protokolom IPv6 – 128 bitová)**.**[1]



1. obrázok zdroj: https://www.tutorialspoint.com/assets/questions/images/45855-1531481367.png

[1] Zdroj: https://www.pocitacovyexpert.eu/sietove-modely-rm-osi-a-tcpip/

Transportná vrstva

Je štvrtou zo siedmich vrstiev definovaných referenčným modelom OSI, ktorá zodpovedá tretej vrstve štvorvrstvového modelu TCP/IP.

- **Transmission Control Protocol (TCP) [segment]**
  Najznámejším transportným protokolom zo sady internetových protokolov. Používa sa na prenosy orientované na spojenie (zaručuje, že dáta odoslané z jedného konca spojenia budú prijaté na druhej strane spojenia v rovnakom poradí a bez chýbajúcich častí.). TCP je komplexnejší protokol vďaka svojmu stavovému dizajnu, ktorý zahŕňa spoľahlivé služby prenosu a toku údajov. Na začiatku spojenia sa využíva 3-way handshake na nadviazanie spojenia potom nasleduje prenos dát a po prenose 4-way handshake na ukočenie spojenia. Je možné aj 3-way handshake, kde po prijatí FIN segmentu odošle druhá strana ACK+FIN v jednom pakete. Následne mu príde odpoveď ACK. Počas vytvárania spojenia sa inicializujú parametre ako poradové čísla paketov, aby sa zabezpečila robustnosť a poradie doručenia. Kvôli už spomenutým vlastnostiam je tento protokol pomalší na rozdiel do UDP.

  Nadviazanie spojenia - Hoci je možné, aby dva stroje nadviazali spojenie zároveň, zvyčajne na jednom stroji beží serverová služba počúvajúca na určitom porte a pasívne počúva, t. j. čaká na prichádzajúce spojenia. Bežne sa to nazýva pasívne otvorenie a určuje stranu spojenia, ktorá funguje ako server. Klientská strana spojenia začne aktívne otvorenie tým, že pošle úvodný SYN segment serveru. Server by mal odpovedať platnou požiadavkou SYN so SYN/ACK. Nakoniec by mal klient odpovedať ACK, čím sa 3-way handshake, a teda fáza nadviazania TCP spojenia ukončí.
  Počas fázy nadviazania TCP spojenia sa medzi dvoma strojmi vymenia tzv. initial sequence numbers (ISN). Tieto slúžia na identifikáciu dát v dátovom toku a počítanie dátových bytov.

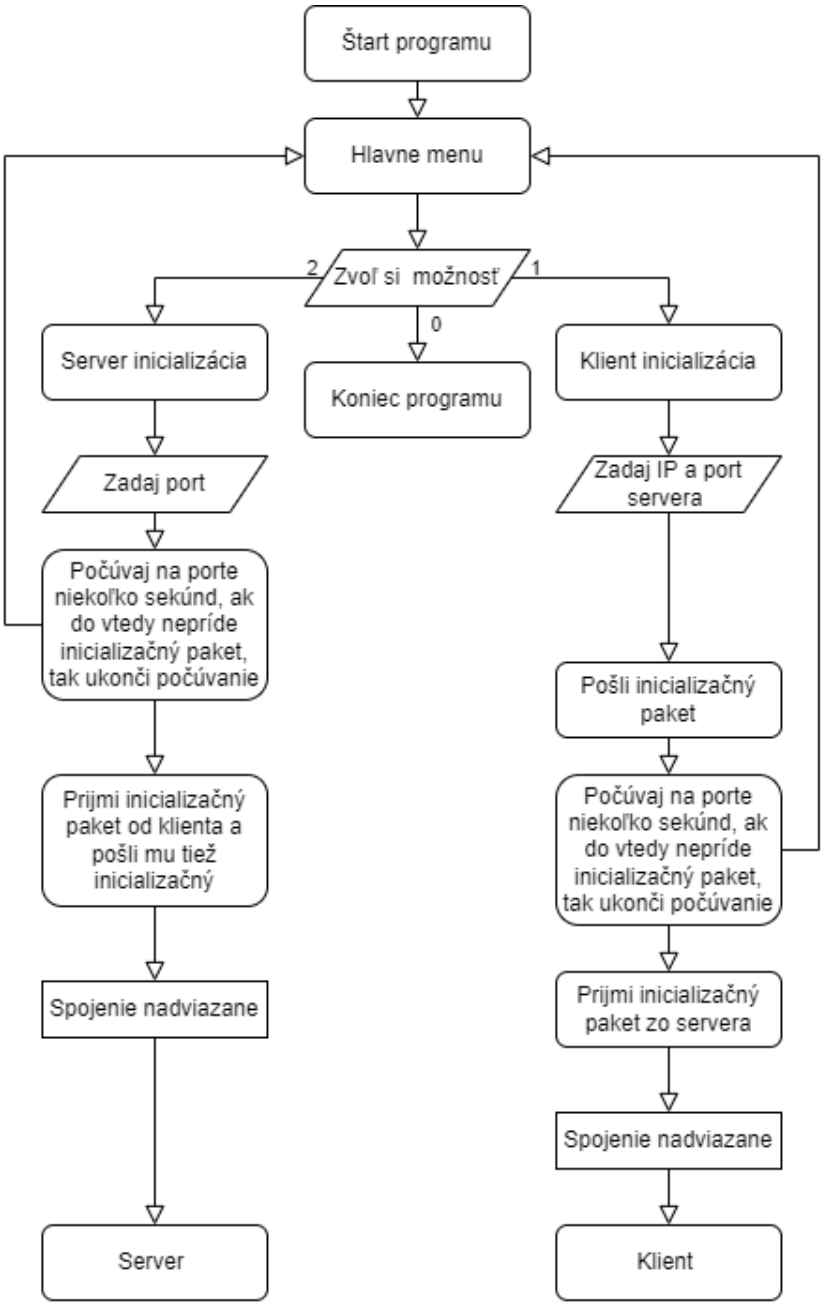
  Prenos dát - Počas fázy prenosu dát určuje niekoľko kľúčových mechanizmov spoľahlivosť a robustnosť TCP. Patria medzi ne poradové čísla pre určenie poradia TCP segmentov a detekciu duplicitných dát, kontrolné súčty pre detekciu chýb v segmentoch a potvrdzovanie a časovače pre detekciu a prispôsobenie sa strate alebo oneskoreniu dát. [2]

- **User Datagram Protocol (UDP) [datagram]**
  Je tzv. "nespoľahlivý" protokol z balíka internetových protokolov. UDP protokol prenáša datagramy medzi počítačmi v sieti, ale na rozdiel od TCP nezaručuje, že prenášaný paket sa nestratí, že sa nezmení poradie paketov, ani že sa niektorý paket nedoručí viackrát a nieje ani potrebné nadviazať spojenie 3-way handshakeom a ešte aj podporuje broadcast a multicast. Vďaka tomu je UDP pre ľahké a časovo citlivé účely rýchlejší a efektívnejší. Jeho bezstavová povaha je užitočná aj pre servery, ktoré odpovedajú na malé požiadavky mnohých klientov. UDP sa používa napríklad na DNS, streamované médiá, prenos hlasu alebo videa (VoIP) a online hry.
  UDP pridáva iba kontrolné súčty a schopnosť roztrieďovať UDP pakety medzi viaceré aplikacie bežiace na jednom počítači.
  Maximálna veľkosť datagramu pre Ipv4 je 65535 B.
  Maximálna veľkosť nášho payloadu je 65535B – IP Hlavička (20 B až 60 B) – UDP Hlavička (8 B) – Vlastná hlavička
  A keďže nechceme aby nenastala fragmentácia na linkovej vrstve tak veľkosť fragmentu musí byť 1500 B – IP Hlavička – UDP Hlavička – Vlastná hlavička.

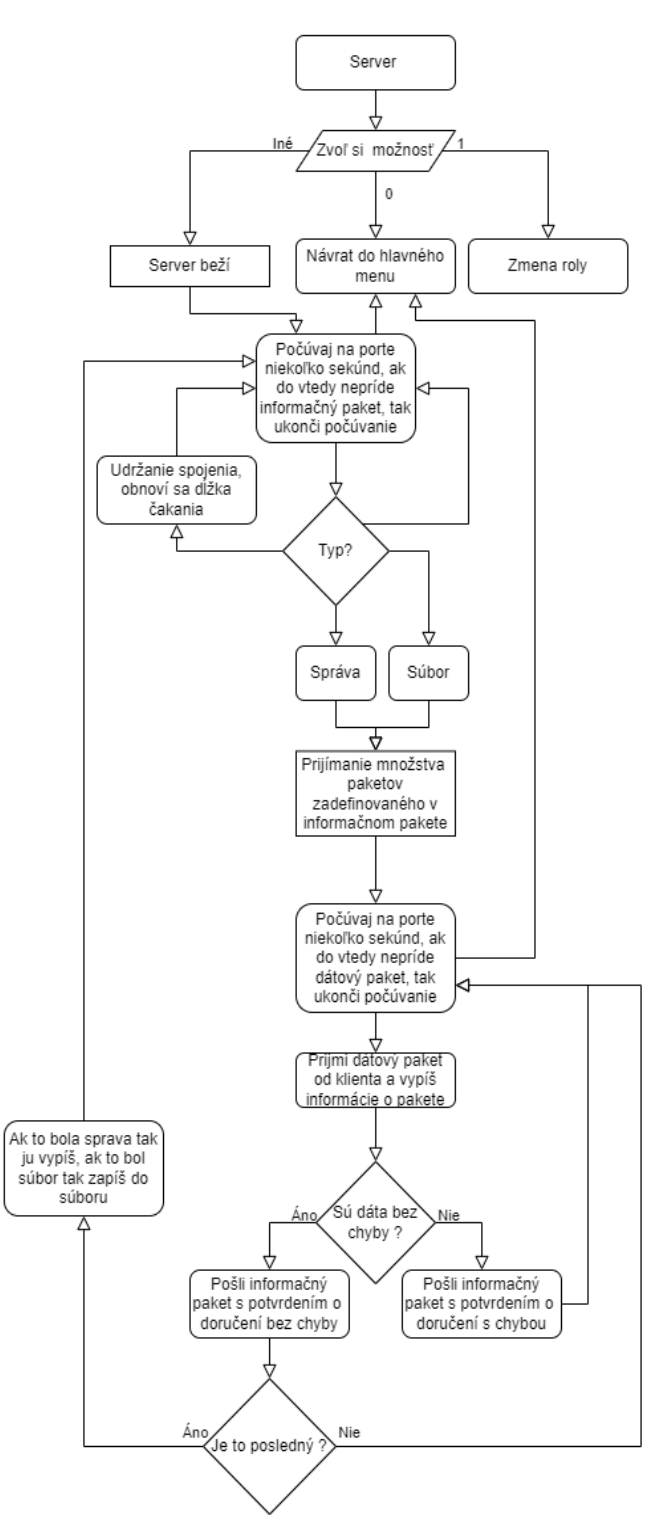
[2] Zdroj: https://en.wikipedia.org/wiki/Transport_layer

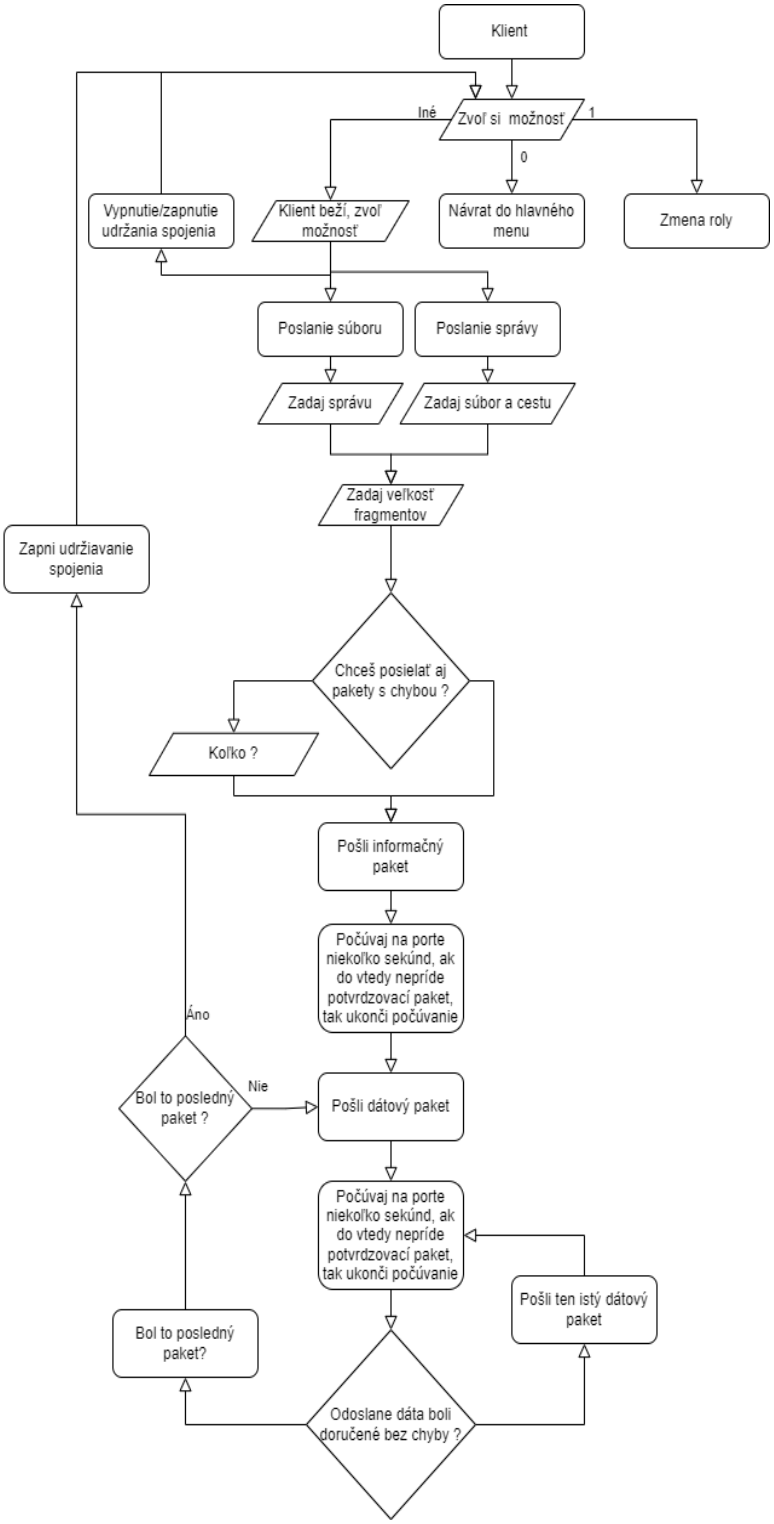
Návrh

Program diagramy:



Adrián Vančo
ID: 103171
Predmet: Počítačové a komunikačné siete





Adrián Vančo
ID: 103171
Predmet: Počítačové a komunikačné siete

Hlavička:

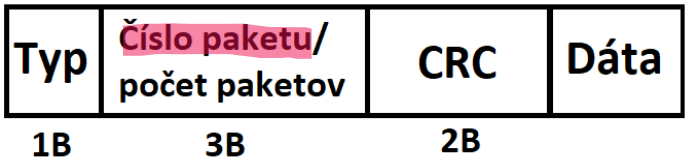
Pre informačné správy bude vyzerať hlavička nasledovne a pre typ na prenášanie dát inak:

Názov súboru maximálne 256B preto lebo Windows nedovoľuje mať väčšiu absolútnu cestu.

3B pre číslo paketu/počet paketov pretože súbor, ktorý chceme preniesť má 2MB a keby užívateľ zvolí fragmentovanie na 1B tak by číslo v tomto poli bolo 2 000 000.

S tým že o tom či bude obsahovať pole číslo paketu/počet paketov a pole názov súboru bude rozhodovať **typ**:

- „0“ - Udržiavanie spojenia (keep alive) [hlavička bude obsahovať iba typ]
- „1“ - Nadviazanie komunikácie (inicializácia spojenia) [hlavička bude obsahovať iba typ]
- „2“ - Bude poslaná správa [hlavička bude obsahovať typ, veľkosť frag. a počet paketov]
- „3“ - Bude poslaný súbor [hlavička bude obsahovať typ, počet paketov, veľkosť frag. a názov súboru]
- „4“ - Dáta



Pole pre dáta na to aby sa nenastala fragmentácia na linkovej vrstve, bude mať veľkosť maximálne 1467B.

Pole CRC sa bude nachádzať zvyšok po CRC metóde o veľkosti 2B.

- „5“ - Obdržané dáta boli v poriadku [hlavička bude obsahovať iba typ]
- „6“ - Obdržané dáta boli chybné [hlavička bude obsahovať iba typ]
- „7“ - Ukončujem spojenie [hlavička bude obsahovať iba typ]

Adrián Vančo
ID: 103171
Predmet: Počítačové a komunikačné siete

Kontrola chýb

Bude zabezpečená pomocou CRC (Cyclic Redundancy Check Code) s pevne daným polynómom. CRC-16-CCITT x^16 + x^12 + x^5 + 1. Odosielateľ pred odoslaním fragmentu dát pripojí zvyšok po delení polynómom (2B). Prijímateľ si potom overí fragment tým že namiesto pridania 0x0000 za dáta Pridá obdržané CRC od odosielateľa a ak zvyšok nieje nulový tak nastala chyba a príjemca pošle informačnú správu že dáta boli poškodené počas prenosu a odosielateľ mu pošle ten istý fragment znova.

ARQ metóda

Vybral som si Stop and Wait ARQ. Teda sa posiela vždy jeden dátový paket. A ďalší sa pošle až po prijatí potvrdenia od predchádzajúceho. Ak však prišlo potvrdenie že paket prišiel poškodený tak pošle znova ten istý, až keď príde potvrdenie že dáta boli v poriadku tak pošle ďalší.

Udržanie spojenia

Odosielateľ po dokončení prvého prenosu každých 10s pošle informačný paket s typom „0“, ktorý resetuje časovač na ukončenie spojenia. Ak by to nestihol pokiaľ by časovač uplynul, tak server pošle informačnú správu s typom „7“ že ukončuje spojenie.

Adrián Vančo
ID: 103171
Predmet: Počítačové a komunikačné siete

Základné informácie

Program vyhotovený v jazyku Python vo verzii 3.8.6
Program je spustiteľný ako .exe
Použité knižnice:

- socket
- struct
- os
- math
- time
- threading

prevzaté crc16 zo https://gist.github.com/oysstu/68072c44c02879a2abf94ef350d1c7c6
so zmeneným polynómom na 0x11021 ako v návrhu

Program je rozdelený do nasledujúcich častí:

- keep alive – funkcia , ktorá udržiava spojenie

- client – obsahuje možnosť poslať správu, súbor na server alebo sa zmeniť rolu na server prípadne skončíť

- server – obsahuje možnosť pokračovať v počúvaní, prípadne prepnúť rolu na client alebo skončiť

Prostredie je CLI, na začiatku ponúkne možnosť výberu následne aj pre odosielateľa alebo príjemcu je výber podobne zobrazený a je intuitívny

```
Dostupné možnosti:
     1 - Odosielatel
     2 - Prijemca
     0 - Pre ukončenie
Zadaj možnosť:
```

Používateľ má možnosť

- prepínať medzi Príjemcom a Odosielateľom
- zadať IP a port príjemcu na strane odosielateľa
- nastaviť si port na strane príjemcu
- zvoliť veľkosť fragmentácie
- zvoliť paket, ktorý sa pošle ako chybný
- ukončiť komunikáciu
- odosielať súbory, správy
- na strane príjemcu zvoliť kam sa súbor má uložiť

Zmeny oproti návrhu nenastali hoci veľkosť fragmentácie som nemusel vôbec implementovať do hlavičky a tým by sa ušetrili 2B tak isto typ pre dátové pakety.

Adrián Vančo
ID: 103171
Predmet: Počítačové a komunikačné siete