Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Počítačové a komunikačné siete
Komunikácia s využitím UDP protokolu

Meno: Peter Plevko

Cvičiaci: doc. Ing. Dominik Macko, PhD.

Sk. Rok: 2020/2021

Čas cvičenia štvrtok 10:00-11:50

Zadanie

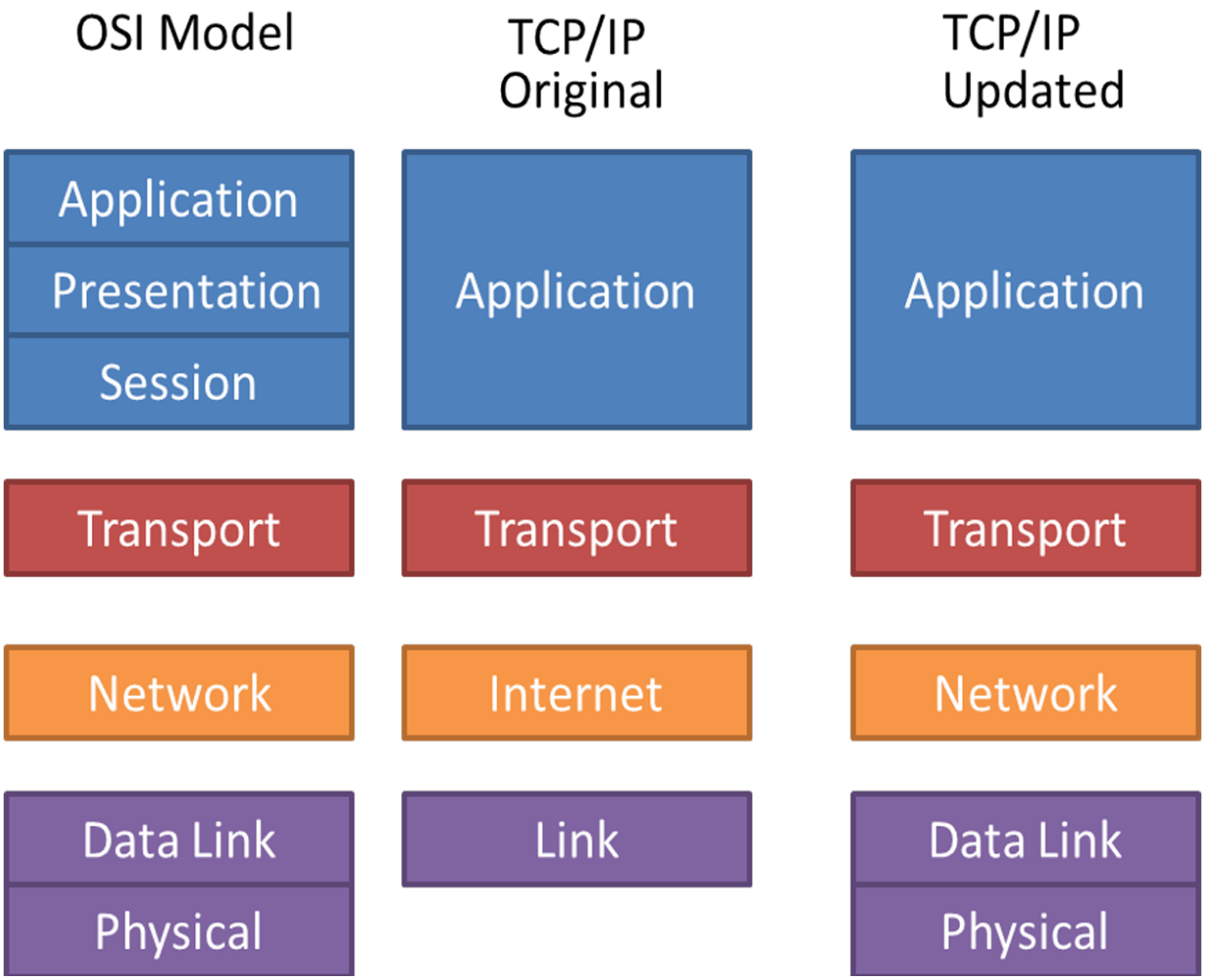
Navrhnite a implementujte program s použitím vlastného protokolu nad protokolom UDP (User Datagram Protocol) transportnej vrstvy sieťového modelu TCP/IP. Program umožní komunikáciu dvoch účastníkov v lokálnej sieti Ethernet, teda prenos textových správ a ľubovoľného binárneho súboru medzi počítačmi (uzlami).
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.
Komunikátor 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 20-60s 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. Použité knižnice a funkcie 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 odoslať minimálne 1 chybný fragment (do fragmentu je cielene vnesená chyba, to znamená, že prijímajúca strana deteguje chybu pri prenose).
7. Prijímajúca strana musí byť schopná oznámiť odosielateľovi správne aj nesprávne doručenie fragmentov.
8. Možnosť odoslať súbor a v tom prípade ich uložiť na prijímacej strane ako rovnaký súbor. Akceptuje sa iba ak program prenesie 2MB súbor do 60s bez chýb.

Analýza

V rámci sieťovej komunikácie existuje veľké množstvo protokolov ktoré musia spolu spolupracovať. Na vyjadrenie vzťahov medzi jednotlivými protokolmi sa spopularizovali dva modeli TCP/IP a referenčný model OSI.



Aplikačná vrstva

Je najvyššou vrstvou a slúži na komunikáciu medzi koncovými aplikáciami Typy protokolov aplikačnej vrstvy sú: DNS, http, HTTPS, BGP a pod.

Transportná vrstva

Slúži na komunikáciu medzi hostami ktorí sú adresovaní jednotlivými portmi. Typy protokolov transportnej vrstvy sú: TCP, UDP, DCCP a pod.

TCP (transmission control protocol)

TCP je spojovo orientovaný protokol ktorý operuje v transportnej vrstve a ktorý sa v porovnaní s UDP vyznačuje spoľahlivosťou. To znamená že sa používa pri prenose dát kde je potrebná spoľahlivosť keďže sa uisťuje ci transfer dát prebehol úspešne ak neprebehol úspešne pošle sa packet znovu. Na začiatku spojenia sa využíva three-way-handshake na zabezpečenie spoľahlivej komunikácie. Kvôli hore uvedeným vlastnostiam sa ale tento protokol stáva pomalším.

UDP (user datagram protocol)

Je nespojovo orientovaný protokol. Vyznačuje sa svojou rýchlosťou ktorú nadobúda pretože nekontroluje straty a nežiada opätovne poslanie packetu. Používa sa keď si nemôžeme dovoliť oneskorenie pri opakovanom prenose chybných packetov (online hry streamovanie). Narozdiel od TCP nie je potrebne nadviazať 3-way-handshake. Taktiež podporuje broadcast a multicast. Jedna sprava v UDP sa nazýva datagram a jeden datagram môže mat maximálnu veľkosť 65 535 B – hlavička IP (20B) - veľkosť hlavičky UDP (8B) - vlastnú hlavičku (8). Nato aby sa sprava nedelila na transportnej vrstve môže mat maximálnu veľkosť 65 499. Na linkovej vrstve nechcem fragmentovat takže môj packet musí mat veľkosť 1500 – UDP – IP – moja hlavička 1500 -8 -20 -8 =1464 Takže maximálna veľkosť ktorú môžem poslať je teda 1464B.

Návrh

Programovací jazyk a používateľské rozhranie

Implementácia protokolu prebehne v jazyku python dôvodom je že ma veľké množstvo funkcii vďaka ktorým nebudem musieť robiť niektoré veci manuálne a taktiež ma veľké množstvo knižníc. Ovládanie bude realizovane pomocou console GUI. Testovanie prebehne pri nadviazaní spojenia medzi počítacom s operačným systémom Windows a počítacom s operačným systémom Windows.

Odosielanie a prijímanie dát

Na začiatku programu si užívateľ vyberie ci chce byt server (prijímať dáta) alebo klient (vysielať dáta). Po úspešnom poslaní dát bude možne odhlásiť sa zo svojej role a prehodiť sa na druhu. Samozrejme ak si prvý užívateľ vyberie funkciu odosielateľa tak druhému ostáva už len funkcia príjemcu.

Vyberiem si klienta: Pre nadviazanie spojenia medzi klientom a serverom musí klient zadať ip adresu servera a číslo portu na ktorý sa chce pripojiť následne si vyberie ci chce poslať správu alebo súbor ak si vyberie súbor musí ešte zadať cestu k súboru ktorý chce odoslať. Následne pošle inicializačnú správu serveru, potom server odošle klientovi správu v ktorej hovorí že pripojenie bolo úspešne a prenos sa môže začať. Ak server nič nepošle do 10 sekúnd užívateľ pravdepodobne zadal zlu ip adresu alebo číslo portu a musí ho zadať znova. Týmto sa zaisti že obe strany sú aktívne klient nebude odosielať dáta do prázdna a server nebude donekonečna čakať na začatie prenosu. Serveru sa taktiež zobrazí sprava ktorá hovorí z akej adresy sa klient pripojil. Po nadviazaní spojenia môže klient začať posielať správy a súbory. Užívateľ môže zadať maximálnu veľkosť fragment ak ju nezadá bude určená sama programom. V prípade dlhšej pasivity bude musieť byt odoslaný keep alive sprava každých 60 sekúnd aby sa udržalo spojenie. V prípade neobdržania keep alive sa spojenie po 60 sekundách preruší. Keďže používam stop and wait tak pošlem jeden fragment a čakám kým mi príde potvrdzujúca sprava ak mi tato potvrdzujúca sprava nepríde do 10 sekúnd pošlem packet znova. Po poslaní správy alebo súboru môže užívateľ začať ďalší prenos.

Vyberiem si server: Používateľ musí serveru nastaviť port. Následne začne server počúvať no neprichádza sprava žiadajúca o inicializáciu spojenia. Server čaká kým nepríde sprava o inicializácii spojenia, keď príde pošle správu potvrdzujúcu spojenie. Fragmenty kontrolujem po jednom. Ak sa nájde chybný fragment odošle správu informujúcu klienta o chybných dátach a dáta sa znova pošlú. Ak sprava prišla úspešne pošle správu o úspechu. Po obdržaní všetkých fragmentov ktoré prešli kontrolou CRC budú fragmenty spojene a uložene ako súbor respektíve ako sprava Po skončení prenosu súbor znova počúva a čaká na správu od klienta.

1,0b vlastná hlavička

Takto budú vyzerať dáta ktoré budem posielať.

TYP	VELKOST	POCET	PORADIE	DATA	CRC

Typ (1 bajt): jeden z dole uvedených typov sprav

Sprava (dek)	Sprava (bin)	Význam správy
1	0001	Inicializácia spojenia
2	0010	Prenos dát
3	0011	Doručene dáta boli chybne
4	0100	Keep alive
5	0101	Doručene dáta boli správne
6	0110	Sprava bola cela odoslaná

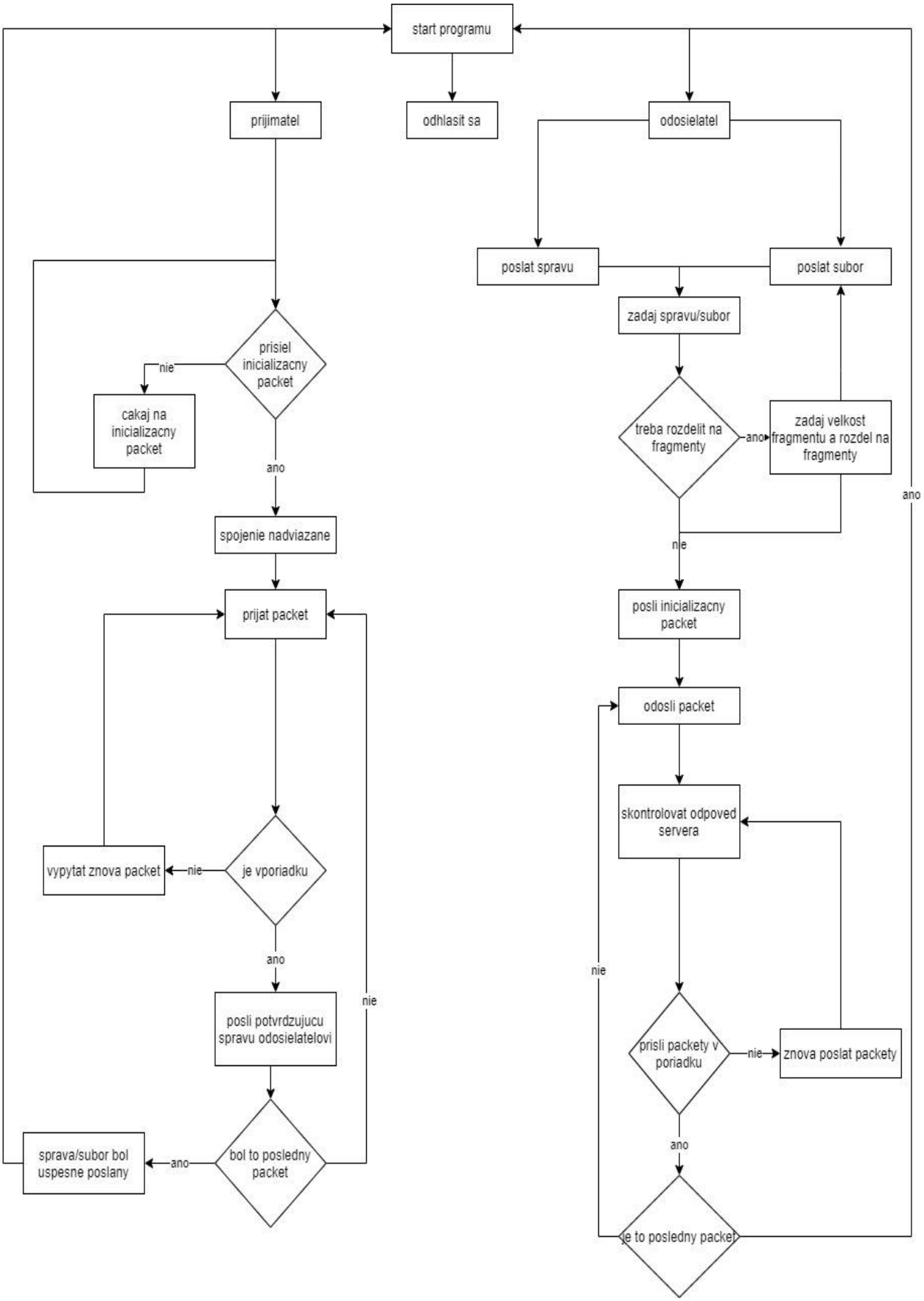
Veľkosť fragmentu (2 bajty): veľkosť odosielaného fragmentu

Počet fragmentov (2 bajty): celkový počet odosielaných fragmentov

Poradie fragment (2 bajty): poradie odosielaného fragmentu

CRC (1 bajty): zvyšok po delení stanovený polynómom

1,0b message sequence diagram


start programu
prijimatel
odhlasit sa
odosielatel
prisiel inicializacny packet
nie
cakaj na inicializacny packet
ano
spojenie nadviazane
prijat packet
vypytat znova packet
nie
je vporiadku
ano
posli potvrdzujucu spravu odosielatelovi
nie
sprava/subor bol uspesne poslany
ano
bol to posledny packet
poslat spravu
poslat subor
zadaj spravu/subor
treba rozdelit na fragmenty
ano
zadaj velkost fragmentu a rozdel na fragmenty
ano
nie
posli inicializacny packet
odosli packet
skontrolovat odpoved servera
nie
prisli packety v poriadku
nie
znova poslat packety
ano
je to posledny packet

1,0b ARQ metóda

Hlavná funkcia tohto protokolu je kontrola či bol packet správne poslaní sprava o úspešnom poslaní musí prísť od príjemcu za určitý časový limit. keď packet nepríde, nepríde za časový interval alebo je zlý tak je packet znova poslaný.

Stop And Wait ARQ:

Je metóda používaná v obojsmernej komunikácii na posielanie informácii medzi dvomi pripojenými zariadeniami odosielateľ a prijímateľ. Tato metóda sa nazýva stop and wait pretože posiela jeden rámec a neposiela ďalšie čaká kým nepríde potvrdenie od príjemcu a až potom pošle dali rámec. Plus odosielateľ si drží kópiu poslaného paketu.

Výhody arp:

Zistenie a oprava erroru je veľmi jednoduchá oproti ostatným technikám.

Nevýhody arp:

Vysoký počet chyb odosielateľa môže viest k strate informácii a zmenši efektivitu a produktivitu systému.

0,5b checksum algoritmus

Ako checksum algoritmus použijem CRC metódu. Ta funguje tak že na začiatku nastavím divisora pre server a pre odosielateľa. V tomto prípade je môj divisor polynóm. CRC vypočítam tak že mam hlavičku k nej pridám k-1 núl na koniec hlavičky pričom k-1 je dĺžka polynómu pričom dĺžka polynómu napríklad $x^3 + x + 1$. je 1011 v bytoch nasledovne použijem modulo 2 binárne delenie (xor) a zvyšok po delení čo dostanem je moje CRC a to apendnem na koniec hlavičky. Následne keď prijmem packet vykonám delenie divisorom a ak dostanem k-1 počet núl na konci sprava bola poslaná úspešne. Kontrola cyklickým kódom, ako každý kontrolný súčet mierne zväčšuje správy, ale zvyšuje jej spoľahlivosť

0.5b keepalive(ka) metóda

Keep alive sa používa na udržanie spojenia. Odosielateľ bude posielať serveru keep alive správu každých 10 sekúnd. Ak tato sprava nebude poslaná alebo nedostane sa k serveru do 60 sekúnd ukonči sa spojenie. Ak sa sprava dostane resetuje sa timer a znova ma odosielateľ 60 sekúnd na poslanie keep alive správy.

Pripomienky od cvičiaceho

v hlavicke je zbytocne v kazdej sprave posielat pocet fragmentov, na druhu stranu 1B CRC je velmi slabe pre velkost spravy 64kB (limitovanu UDP protokolom). Ak keepalive posiela len odosielatel a server nie, tak ako bude odosielatel vediet ci je server stale dostupny?

Napráva:

Vytvoril som inicializačnú hlavičku a už zbytočne neposielam počet fragmentov v každej správe.

CRC som zväčšil z 1 B na 2 B.

Keepalive odosiela iba odosielateľ ale server po obdŕžaní keep alivu pošle potvrdzovaciu správu ak tato sprava neprišla spojenie sa konci.

Zmeny oproti návrhu

Pridal som jednu hlavičku ktorá vyzerá nasledovne

Informačná hlavička

TYP	Počet packetov

Typ 1 pre textovú správu

Typ 2 pre súbor

Počet packetov – počet koľko mi príde packetov resp. (fragmentov)

Informačný packet pre začatie posielania správy

Moja normálna hlavička tiež prešla par zmenami, zmenil som poradie CRC som dal pred dáta aby som vždy mal fixnú pozíciu dát. Kvôli tomuto dôvodu môžem odstrániť veľkosť správy avšak musel by som meniť kód tak som ju tam nechal. Poradie tiež nemusí byt v hlavičke ale z rovnakého dôvodu ako predtým som ho tam nechal.

Takto budú vyzerať dáta ktoré budem posielať.

TYP	VELKOST	PORADIE	CRC	DATA

Typ (1 bajt): jeden z dole uvedených typov sprav

Sprava (dek)	Význam správy
1	Inicializácia spojenia
2	Prenos dát
3	Doručene dáta boli chybne
4	Keep alive
5	Doručene dáta boli správne

Veľkosť fragmentu (2 bajty): veľkosť odosielaného fragmentu

Poradie fragment (2 bajty): poradie odosielaného fragmentu

CRC (2 bajty): zvyšok po delení stanovený polynómom

Veľkosť mojej hlavičky je 7 B.

Takže maximálna veľkosť fragmentu ktorí môžem odoslať je

65 000 – UDP – IP – moja hlavička

65 000 – 8 – 20 – 7 = 64965

Timeout keď klient neprijme správu o nadviazaní spojenia som z mojich návrhových 10 sekúnd dal na 60 pretože som nestíhal poslať správu.

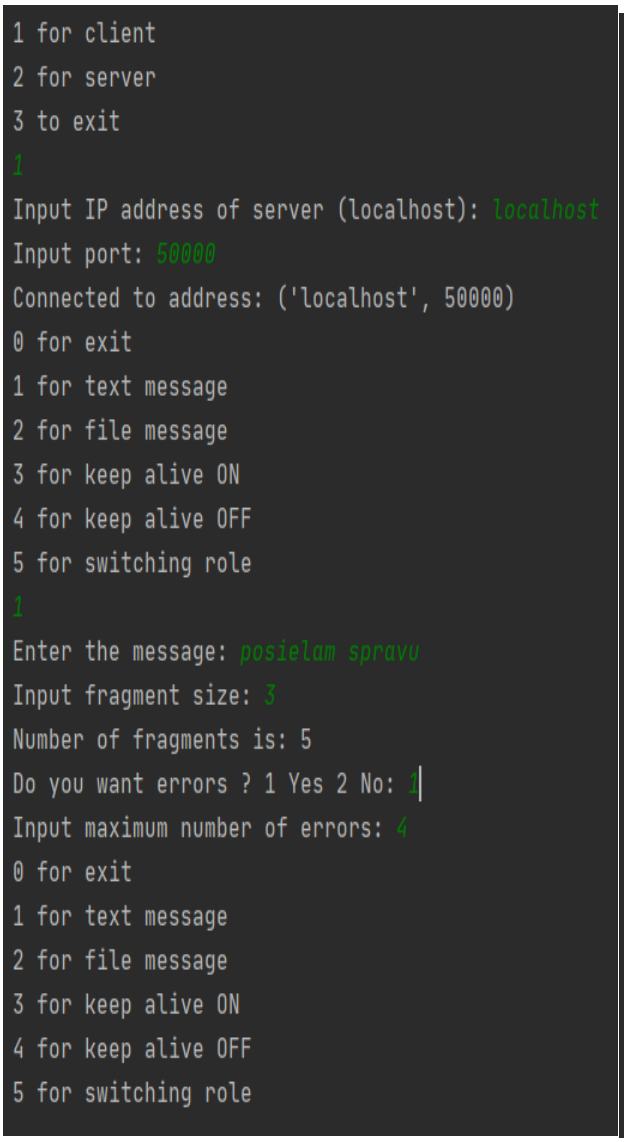
Ďalej som aj zmenil oproti návrhu že užívateľ nezadá cestu k súboru ktorý chce poslať ale zadá jeho meno. Ďalej tento súbor prijmem do toho istého adresára akurát z menom received.

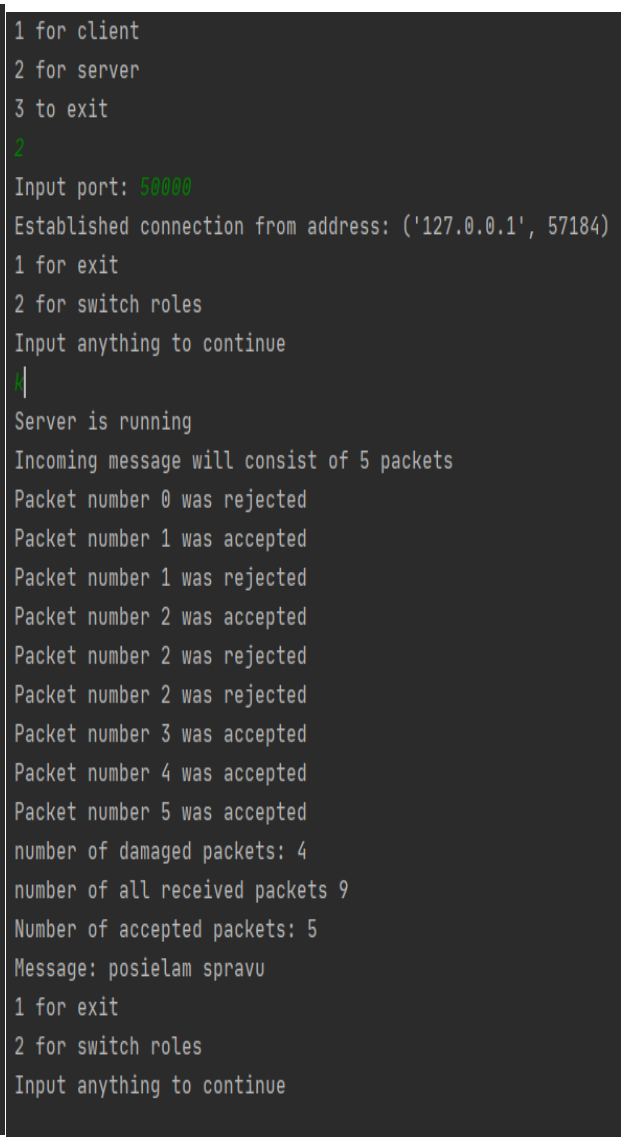
Chyby simulujem nasledovne vypýtam si od užívateľa ci chce chyby ak zadá áno následne si od neho vypýtam koľko maximálne chyb chce to znamená že užívateľ zadá napríklad číslo 5. následne každý packet ma 50% šancu že sa pokazí ale celkový počet pokazených fragmentov nesmie presiahnuť číslo 5 je šanca že sa pokazí 5 krát ten istý alebo že sa nepokazí žiaden.

keď mi príde dobrý packet pošlem ack 5 čo znamená packet som prijal správne ináč pošlem 3 to znamená packet som prijal zle.

Užívateľské rozhranie

Moje rozhranie je realizovane pomocou console gui. Je intuitívne a vždy je napísane čo sa požaduje od užívateľa.





Užívateľ zadá ci chce byt klient alebo server

Klient: klient musí zadať ip adresu na ktorú sa chce pripojiť ďalej jej port vyberie si čo chce spraviť napríklad poslať správu vyberie si veľkosť fragmentu vyberie si ci chce simulovať chyby a koľko ich chce.

Server: server musí zadať svoj port ďalej musí zadať znak aby počúval a následne akceptuje packety a robí výpis.

Použité knižnice:

- binascii
- math
- os
- socket
- struct
- threading
- time
- random

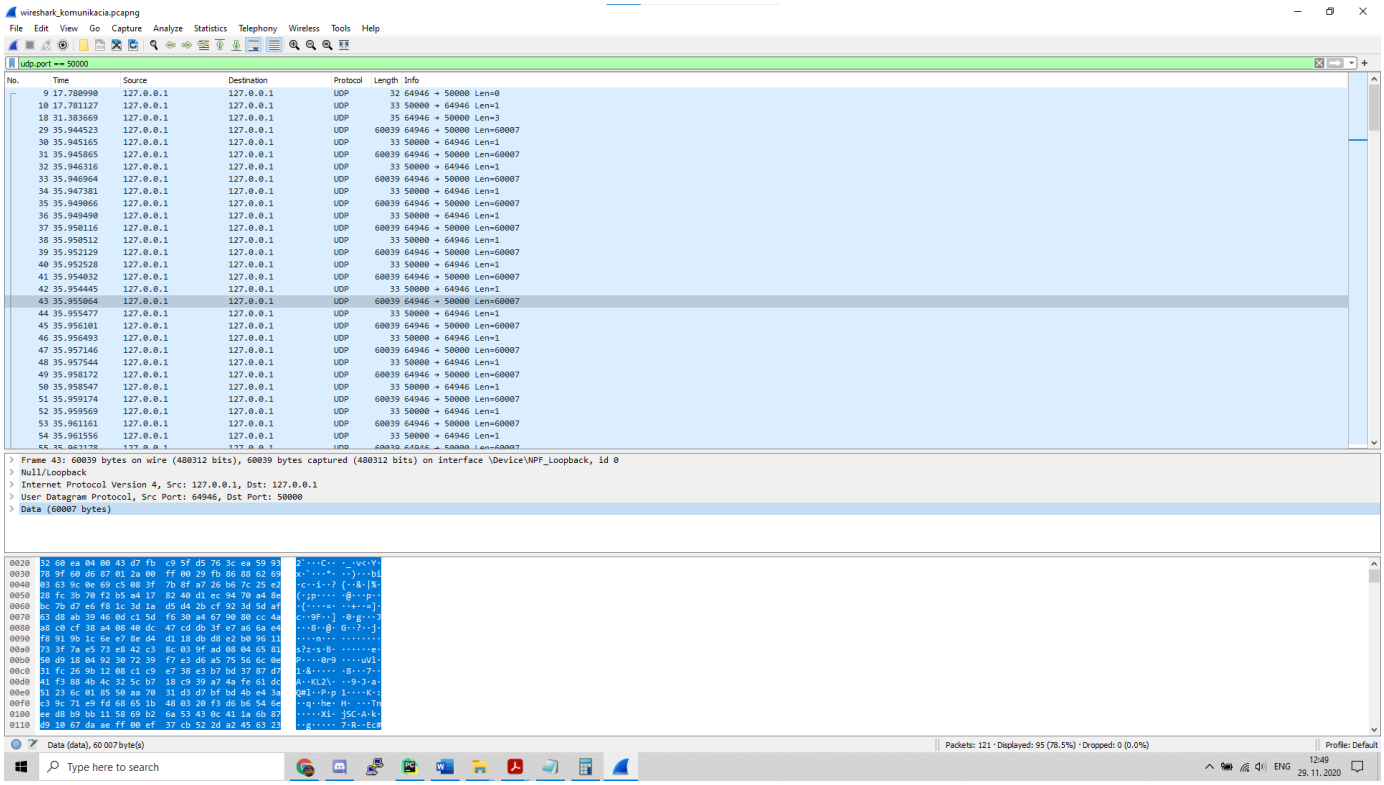
Splnene požiadavky a scenáre

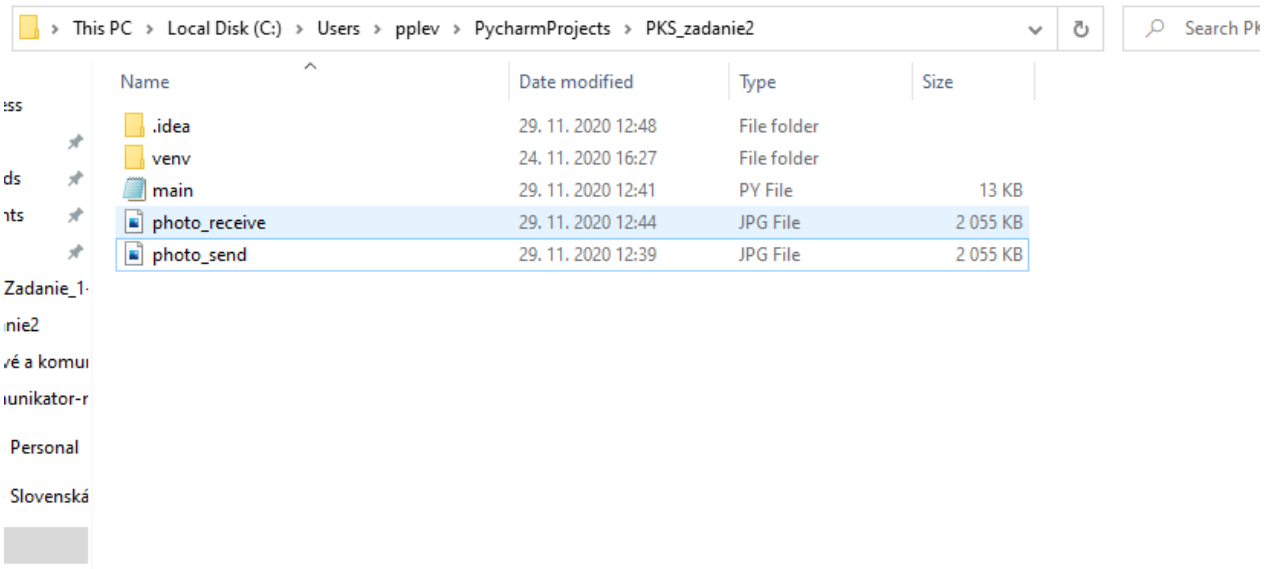
Program splna všetky minimálne vlastnosti, pracuje s dátami optimálne, používateľ je schopný zadať IP adresu a port serveru taktiež aj veľkosť fragmentu, Ta môže byt pri posielaní sprav najviac 64965 lebo nemá dôjsť k fragmentácii na linkovej vrstve. Taktiež môže užívateľ zadať počet chybných packetov ktoré sú detegovane na strane príjemcu. Ten potom vie odosielateľovi oznámiť , správne aj chybne doručenie packetov. Taktiež sa da úspešne preniesť 2 MB súbor.

1. Program je implementovaný v jazyku Python
2. Pri odosielaní je umožnené zadať IP adresu a port
3. Pri odosielaní si užívateľ môže zvoliť max. veľkosť fragmentu
4. Odosielateľ aj prijímateľ zobrazujú absolútnu cestu k súboru, veľkosť a počet fragmentov
5. Odosielateľ môže odoslať hocikoľko chybných fragmentov avšak mam ich založene na 50% náhode tak sa môžu vykonať všetky alebo žiaden
6. Odosielateľovi je oznamované správne/nesprávne doručenie fragmentov, pričom sa vypíšu indexy nesprávne doručených fragmentov
7. Odosielanie súborov a ich následné ukladanie
8. Keep alive

Wireshark

Pri nastavení veľkosti fragmentu na 60000 vidím že moje dáta majú 60 007 tých 7 B je moja hlavička.





Súbor sa preniesol úspešne.

Záver

Môj program ktorý prenáša dáta po sieti pomocou UDP protokolu zvyšuje jeho spoľahlivosť. Program sa delí na dve časti, prijímaciu a odosielaciu po úspešnom pripojení sa môžu užívatelia medzi sebou vymeniť, bez toho aby sa spojenie prerušilo. Tento program slúži na spracovanie vstupnej správy alebo vstupného súboru následne rozdelenie na fragmenty a odoslanie fragmentov spolu s hlavičkami cez sieť. Následne prijatie týchto fragmentov kontrola ich správnosti v prípade nesprávnosti vyžiadanie nových na konci spojenie týchto fragmentov do jednej správy, jedného súboru.

Testovanie prebehlo na jednom počítací. Počas testovania som testoval súbor s veľkosťou 2 MB a správy o dĺžke 10 znakov. Skúsil som aj rôzne kombinácie chybných packetov.