

**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

Navrhните 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

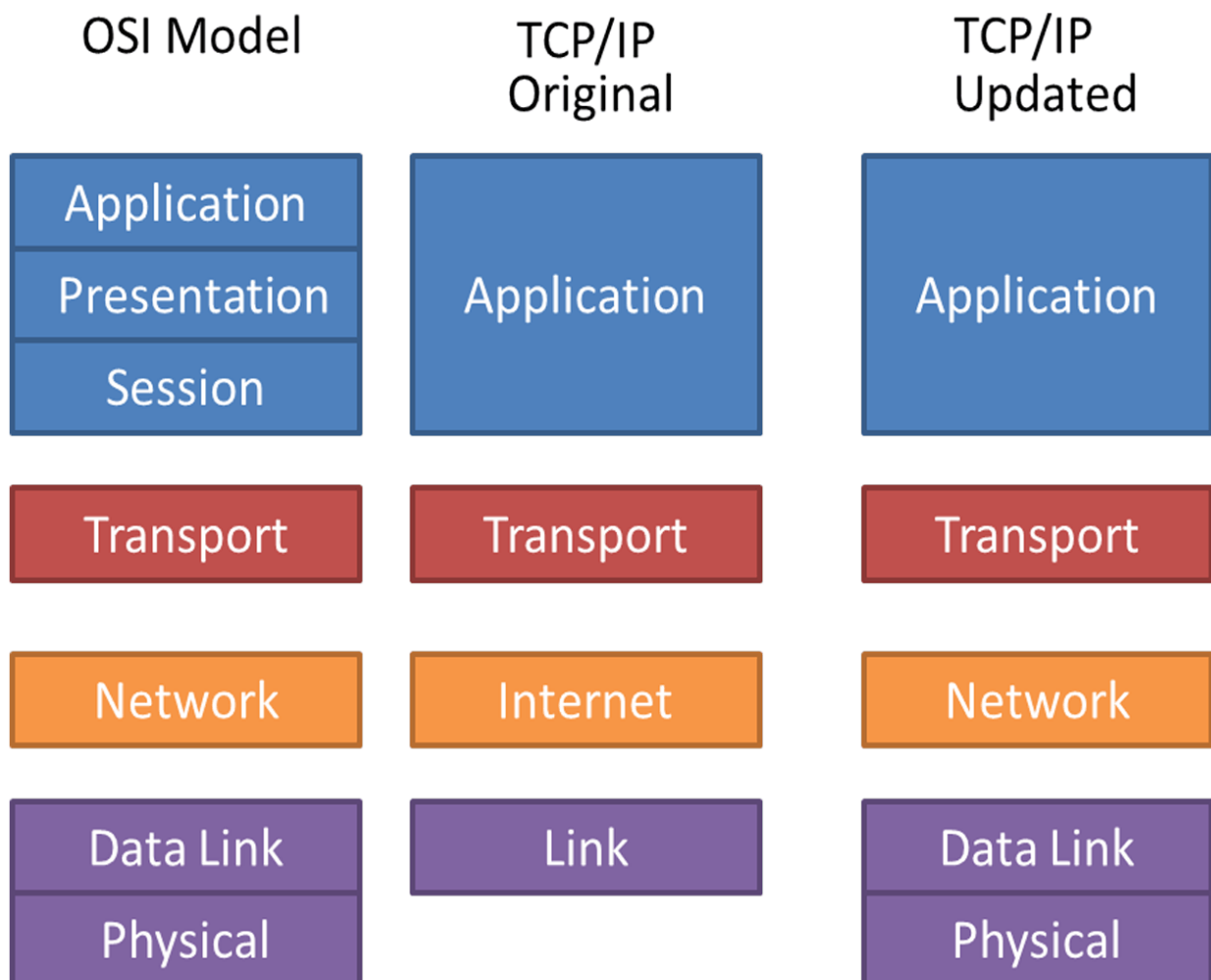
6. Možnosť odoslať minimálne 1 chybný fragment (do fragmentu je cielene vnesená chyba, to znamená, že prijímajúca strana deteguje chybu pri prenose).

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 uistuje či 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)**

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). Na rozdiel od TCP nie je potrebné nadviazať 3-way-handshake. Taktiež podporuje broadcast a multicast. Jedna sprava v UDP sa nazýva datagram a jeden datagram môže mať maximálnu veľkosť 65 535 B – hlavička IP (20B) - veľkosť hlavičky UDP (8B) - vlastnú hlavičku (8). Nato aby sa sprava nedelila na transportnej vrstve môže mať maximálnu veľkosť 65 499. Na linkovej vrstve nechcem fragmentovať takže môj packet musí mať 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 funkcií vďaka ktorým nebudem musieť robiť niektoré veci manuálne a taktiež ma veľké množstvo knižníc. Ovládanie bude realizované pomocou console GUI. Testovanie prebehne pri nadviazaní spojenia medzi počítačom s operačným systémom Windows a počítačom s operačným systémom Windows.

## Odosielanie a prijímanie dát

Na začiatku programu si užívateľ vyberie či chce byť server (prijímať dáta) alebo klient (vysielat dáta). Po úspešnom poslaní dát bude možné odhlásiť sa zo svojej role a prehodit sa na druhú. Samozrejme ak si prvý užívateľ vyberie funkciu odosielať a 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 či 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 správa ktorá hovorí z akej adresy sa klient pripojil. Po nadviazaní spojenia môže klient začať posilať správy a súbory. Užívateľ môže zadať maximálnu veľkosť fragmentu ak ju nezadá bude určená sama programom. V prípade dlhšej pasivity bude musieť byť odoslaný keep alive správa 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 správa ak mi táto potvrdzujúca správa 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 správa žiadajúca o inicializáciu spojenia. Server čaká kým nepríde správa 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 správa prišla úspešne pošle správu o úspechu. Po obdržaní všetkých fragmentov ktoré prešli kontrolou CRC budú fragmenty spojené a uložené ako súbor respektíve ako správa. Po skončení prenosu súbor znova počúva a čaká na správu od klienta.

## 1,0b vlastná hlavička

Takto budú vyzerat' dáta ktoré budem posielat'.

TYP	VELKOST	POCET FRAGMENTOV	PORADIE FRAGMENTU	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

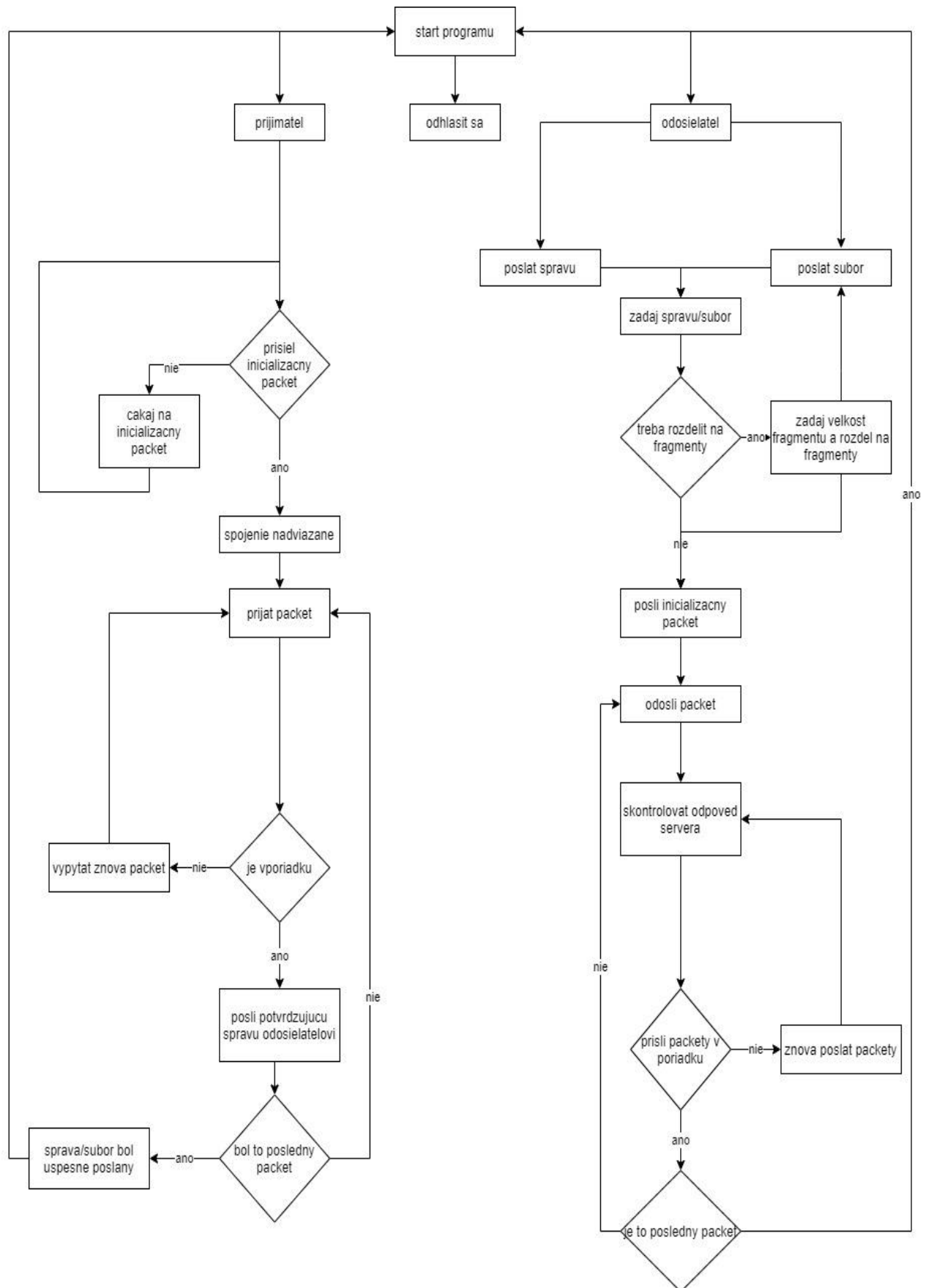
Veľkosť fragmentu (2 bajty): veľkosť odosielaného fragmentu

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



## 1,0b ARQ metóda

Hlavná funkcia tohto protokolu je kontrola či bol packet správne poslaný 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 ďalší rámec. Plus odosielateľ si drží kópiu poslaného packetu.

Výhody arp:

Zistenie a oprava erroru je veľmi jednoduchá oproti ostatným technikám.

Nevýhody arp:

Vysoký počet chýb odosielateľa môže viesť k strate informácii a zmenši efektívnosť a produktívnosť systému.

## 0,5b checksum algoritmus

Ako checksum algoritmus použijem CRC metódu. Tá funguje tak že na začiatku nastavím divisora pre server a pre odosielateľa. V tomto prípade je môj divisor polynóm. CRC vypočítam tak že mám 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 appendnem 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 správnosť, ale zvyšuje jej spoľahlivosť

## 0.5b keepalive(ka) metóda

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