Dokumentácia

Zadanie č. 2

Počítačové a komunikačné siete

2023 – 24

# **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 samostatné vlákno.

Program musí spĺňať nasledujúce požiadavky (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 a 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 (stačí na strane vysielača) musí mať možnosť zvoliť si max. veľkosť fragmentu a meniť ju dynamicky počas behu programu pred poslaním správy/súboru (neplatí pre pakety na udržanie spojenia).
5. Obe komunikujúce strany musia byť schopné zobrazovať:

a) názov a absolútnu cestu k súboru na danom uzle,

b) veľkosť a počet fragmentov vrátane celkovej veľkosti správy/súboru.

1. Možnosť simulovať chybu prenosu odoslaním minimálne 1 chybného fragmentu pri prenose správy a súboru (do dátovej časti fragmentu alebo do checksum je cielene vnesená chyba, to znamená, že prijímajúca strana deteguje chybu pri prenose).
2. Prijímajúca strana musí byť schopná oznámiť odosielateľovi správne aj nesprávne doručenie fragmentov. Pri nesprávnom doručení fragmentu vyžiada znovu poslať poškodené dáta.
3. Možnosť odoslať 2MB súbor a v tom prípade ho uložiť na prijímacej strane ako rovnaký súbor, pričom používateľ zadáva iba cestu k adresáru kde má byť uložený.
4. Program musí byť organizovaný tak, aby oba komunikujúce uzly mohli prepínať medzi funkciou vysielača a prijímača bez reštartu programu automatizovane (jedna strana pošle správu na prepnutie, dostane ACK z druhej strany a uzly sa automaticky prepnú) - program nemusí (ale môže) byť vysielač a prijímač súčasne.

Obsah

[Zadanie 2](#_Toc152059300)

[Programovací jazyk a prostredie 4](#_Toc152059301)

[Návrh projektu 4](#_Toc152059302)

[Návrh hlavičky – riadiace pakety 4](#_Toc152059303)

[Návrh hlavičky – prenosné pakety 5](#_Toc152059304)

[Metóda ARQ 5](#_Toc152059305)

[Metóda Checksum 5](#_Toc152059306)

[„Keep alive“ metóda 5](#_Toc152059307)

[Diagram spracovávania komunikácie 6](#_Toc152059308)

[Zmeny oproti návrhu 7](#_Toc152059309)

[Implementácia 7](#_Toc152059310)

[Inicializácia a údržba spojenia 7](#_Toc152059311)

[Posielanie správ a súborov 9](#_Toc152059312)

# Programovací jazyk a prostredie

Svoj protokol budem programovať v programovacom jazyku Python 3.11, v programe JetBrains PyCharm 2023.2.3 s použitím knižníc socket, threading , time, crcmod, os a struct.

# Návrh projektu

V mojom protokole budú dva typy paketov: riadiace a prenosné – každý typ bude mať vlastnú hlavičku podľa potrieb komunikácie. Úlohou riadiacich paketov bude sprostredkovať komunikáciu medzi uzlami na začiatku spojenia aj v priebehu, čiže medzi prenosmi dát. Prenosné pakety budú slúžiť výlučne na prenos dát. Úlohy odosielateľa a prijímateľa budú rozdelené do dvoch samostatných zdrojových kódov sender.py a receiver.py a komunikácia medzi dvoma uzlami bude bežať v main.py, kde budú vytvorené dva objekty, „uzly“, ktoré budú implementovať funkcie z oboch vedľajších súborov.

Na začiatku si používateľ zvolí či chce byť odosielateľ alebo prijímateľ. Ak si používateľ zvolí úlohu odosielateľa, program si od neho vypýta číslo portu a IP adresu prijímateľa, s ktorým sa chce spojiť a prebehne nadviazanie spojenia.

Po nadviazaní spojenia sa automaticky spustí vlákno pre udržiavanie spojenia – „keep alive“ a spustí sa aj kontrolné vlákno, cez ktoré si môže zvoliť čo chce robiť:

1. Odoslať správu
2. Odoslať súbor
3. Zmeniť úlohu na prijímateľa
4. Nerobiť nič

Tieto rozhodnutia sa budú vykonávať pomocou riadiacich paketov. Ak sa rozhodne poslať súbor alebo správu, vlákno udržiavania spojenia sa skončí a po skončení odosielania sa znovu automaticky zapne. Odosielanie dát budú realizovať prenosné pakety.

Ak sa používateľ rozhodne, že chce byť prijímateľ, program si vypýta číslo portu, na ktorom bude počúvať prichádzajúce správy. Po obdržaní riadiaceho paketu, so žiadosťou o nadviazanie spojenia automaticky odošle riadiaci paket s potvrdením a spojenie je nadviazané. Následne čaká na správy od odosielateľa.

V oboch prípadoch môže používateľ odoslať žiadosť o zmenu úlohy (odosielateľ / prijímateľ). Túto žiadosť môže samozrejme poslať iba vtedy, keď neprebieha prenos správ alebo súborov – čiže keď beží len udržiavanie spojenia. Po odoslaní žiadosti čaká na potvrdenie od druhej strany a po obdržaní potvrdenia si role prehodia.

## Návrh hlavičky – riadiace pakety

|  |
| --- |
| TYP  (1B) |

Riadiace pakety budú vyzerať nasledovne:

Žiadne ďalšie informácie pri nich nebudú potrebné.

Typy paketov budú podľa čísel označené nasledovne:

1. Paket pre udržiavanie spojenia – „keep alive“
2. Paket pre nadviazanie spojenia na začiatku komunikácie
3. Paket s potvrdením – „ACK“
4. Paket s negatívnym potvrdením – „NACK“
5. Paket so žiadosťou o odoslanie správy
6. Paket so žiadosťou o odoslanie súboru
7. Paket s informáciou o prijatí všetkých dát
8. Paket so žiadosťou o zmenu úloh
9. Paket so žiadosťou o koniec komunikácie

## Návrh hlavičky – prenosné pakety

Prenosné pakety budú podľa môjho protokolu vyzerať nasledovne:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Veľkosť fragmentu (2B) | Počet fragmentov (2B) | Poradie fragmentu (2B) | Dáta | CRC  (2B) |

Veľkosť fragmentu – veľkosť odosielaného fragmentu

Počet fragmentov – celkový počet fragmentov správy

Poradie fragmentu – poradie odosielaného paketu

CRC – kontrolný údaj pre stanovenie správnosti prenášaných dát

Veľkosť mojej hlavičky je 8B, veľkosť UDP hlavičky je 8B a veľkosť IP hlavičky je 20B, preto maximálna veľkosť fragmentu, ktorý môžem odoslať je 65 000 – 36 = 64 964 Bajtov.

## Metóda ARQ

V mojom programe budem používať ARQ metódu „Go Back-N“. Pointa tejto metódy spočíva v tom, že odosielateľ rozdeľuje dáta na pevne definované bloky, nazývané okná o veľkosti N. Každý blok je odoslaný bez čakania na potvrdenie, až kým nie je celé okno naplnené. Prijímateľ potvrdzuje prijaté dáta pomocou riadiacich paketov typu 2 – „ACK“ alebo v prípade korupcie prijatých dát pomocou riadiacich paketov typu 3 – „NACK“.

V prípade, že odosielateľ dostane negatívnu odpoveď, alebo v stanovenom čase nedostane kladnú odpoveď, odosielateľ opakuje odoslanie všetkých dát od posledného potvrdeného bodu. Prijímateľ zahadzuje všetky prichádzajúce bloky okrem požadovaného bloku.

Túto metódu som si zvolil, lebo zistenie a oprava chyby prenosu paketu je pri nej pomerne jednoduchá. Jej nevýhodou však je, že v prípade chyby prenosu môže dôjsť k neefektívnemu opakovanému odosielaniu dát.

## Metóda Checksum

Na kontrolu integrity prijatého paketu bude môj protokol používať CRC algoritmus.

Vzhľadom na povolenie cvičiaceho budem používať 16-bitový kontrolný súčet pomocou knižnice crcmod s polynómom „0x8005“. Týmto polynómom sa vydelia dáta zakódované do reťazca Bajtov a zvyšok po tomto delení sa zakóduje do reťazca o veľkosti dvoch Bajtov a pridá sa na koniec reťazca dát. Takto zakódované dáta potom prijímateľ znovu vydelí stanoveným polynómom a ak je výsledok tohto delenia 0, dáta boli prenesené správne. Ak je výsledok hocijaký iný, nastala korupcia dát pri prenose.

Túto metódu som si vybral pretože vie veľmi spoľahlivo zachytávať chyby v pakete a pridáva iba malé množstvo informácii k paketu.

## „Keep alive“ metóda

Metóda udržania spojenia medzi komunikujúcimi uzlami vyplýva zo zadania. Odosielateľ musí každých 5 sekúnd odoslať riadiaci paket typu 0. Po obdržaní takéhoto paketu prijímateľ automaticky posiela riadiaci paket typu 2 – „ACK“. Obidva uzly majú v svojom kóde implementovaný časovač vypršania spojenia, ktorý sa úspešným prijatím takýchto paketov zakaždým resetuje. Časovač je nastavený na 16 sekúnd. Ak časovač vyprší, spojenie sa terminuje a pre ďalšiu komunikáciu je potrebné ho znovu nadviazať.

## Diagram spracovávania komunikácie

A black background with white rectangular shapes

Description automatically generated

# Zmeny oproti návrhu

Oproti návrhu som zmenil iba metódu ARQ, z GoBack-N som prešiel na Stop and Wait. Pointa tejto metódy je, že odosielateľ pošle jednu správu a potom čaká na potvrdenie (ACK) od prijímateľa, ktorý informuje o úspešnom doručení alebo potrebe opakovania. Ak odosielateľ v stanovenom čase nedostane potvrdenie alebo dostane negatívne potvrdenie (NACK), opätovne posiela posledný paket.

Výhodou tejto metódy je jednoduchšia implementácia na oboch uzloch, nevýhodou je, že pri vyššej chybovosti sa znižuje efektivita využitia siete.

# Implementácia

Po spustení sa program používateľa opýta, či chce byť odosielateľ alebo prijímateľ.

A screen shot of a black background

Description automatically generated

A black background with white text

Description automatically generatedAk si používateľ zvolí rolu odosielateľa, program si ďalej vypýta cieľovú IP adresu a port druhej strany, na ktorú sa budú dáta posielať.

Ak si vyberie rolu prijímateľa, program si vypýta číslo portu, na ktorom bude počúvať prichádzajúce správy.

## Inicializácia a údržba spojenia

A computer screen with text

Description automatically generatedPo inicializovaní oboch strán sa spustí funkcia pre nadviazanie spojenia. Inicializácia spojenia prebieha na strane odosielateľa tak, že odosielateľ pošle druhej strane paket typu 1 – žiadosť o nadviazanie spojenia a čaká, kým mu prijímateľ pošle ACK – paket typu 2. Ak odpoveď nepríde, paket so žiadosťou sa znovu posiela a znovu sa čaká na odpoveď. Tento proces beží, kým odpoveď nepríde.

Na strane prijímateľa to funguje podobne. Program čaká na inicializačnú správu od druhej strany, a keď tá správa príde posiela paket typu 2 – ACK. Ak inicializačná správa neprišla, proces sa opakuje až kým nedôjde.

A screen shot of a computer program

Description automatically generatedA computer screen shot of a program

Description automatically generatedPo správnom inicializovaní spojenia sa na prijímajúcom uzle spúšťa hlavná funkcia pre prijímanie správ, ktorá riadi celú komunikáciu na tejto strane. V rámci tejto funkcie prijímateľ prijíma všetky riadiace pakety vrátane paketov typu 0 – udržiavanie spojenia.

Na odosielajúcom uzle sa spustí riadiace vlákno, v rámci ktorého si používateľ môže zvoliť čo sa bude diať. Používateľ môže:

1. Odoslať správu
2. Odoslať súbor
3. Vymeniť si role s druhou stranou
4. Skončiť spojenie

Kým si používateľ vyberá čo chce robiť, začne bežať aj vlákno pre udržiavanie spojenia. Toto vlákno beží až kým si používateľ nevyberie nejakú akciu.

Vlákno pre údržbu spojenia funguje tak, že sa každých 5 sekúnd posiela paket typu 0 – „keep alive“ a čaká sa na odpoveď od prijímateľa. Prijímateľ zasa čaká na pakety typu 0 a za každý jeden prijatý posiela paket typu 2 – ACK. Dobu čakania som na oboch koncoch stanovil na 16 sekúnd, čiže tri cykly vymieňania daných správ. Preto ak z ktorejkoľvek strany za sebou neprídu 3 správy, spojenie je ukončené a program o tom vypíše správu do konzoly a končí.

Na odosielajúcom uzle sa vlákno pre údržbu automaticky spúšťa vždy, keď sa neodosielajú žiadne dáta – čiže jedna zo štyroch možností uvedených vyššie, a ukončí sa automaticky keď si používateľ zvolí nejakú akciu.

Na prijímajúcom uzle je za údržbu zodpovedná hlavná funkcia pre prijímanie správ, ktorá má nastavený časovač na spomínaných 16 sekúnd a vždy keď prijme správu typu 0, časovač sa resetuje. Keď dostane hocijakú inú správu – čiže jednu zo štyroch možností používateľa na druhej strane, časovač sa vypne a spustí sa potrebná funkcia.

## Posielanie správ a súborov

Keď si používateľ na strane odosielateľa zvolí, že chce poslať správu, vlákno pre údržbu dokončí aktuálnu iteráciu cyklu – pár sekúnd treba počkať. Potom odošle prijímateľovi paket typu 4 – žiadosť o odoslanie správy – týmto sa skončí údržba aj na prijímajúcej strane.

Následne si program vypýta od používateľa aby zadal správu, ktorú chce poslať a veľkosť fragmentov na ktoré sa má správa rozdeliť a začne po jednom fragmenty posielať.

A screenshot of a computer program

Description automatically generatedPo každom čaká na kladný ACK od druhej strany, až potom posiela ďalší fragment. Fragmenty majú hlavičku „prenosných“ paketov, čiže na začiatku je informácia o veľkosti daného fragmentu, celkovom počte všetkých fragmentov a poradovom čísle daného fragmentu. Po týchto informáciách nasledujú samotné dáta a na konci je 2 bajtová kontrolná suma vypočítaná CRC algoritmom.

Na strane prijímateľa sa následne správa naspäť poskladá a vypíše do terminálu aj s informáciou o jej veľkosti. Po úspešnom prenose dát prijímateľ posiela druhej strane paket typu 6 – dáta boli úspešne prijaté a tá mu zas odošle paket s potvrdením. Následne sa znovu spúšťa proces údržby spojenia.

A screenshot of a computer program

Description automatically generated

Posielanie súborov funguje podobne. Namiesto zadania správy si program od používateľa vypýta cestu k súboru, ktorý chce poslať a takisto sa ho opýta aj na veľkosť fragmentov.

A screen shot of a computer

Description automatically generatedPri posielaní súborov sa najprv pošle meno súboru, aby druhá strana vedela pod akým názvom ho má uložiť a následne sa pomocou rovnakej Stop and Wait ARQ metódy pošlú fragmenty súboru. Prijímajúca strana si súbor ukladá do momentálneho pracovného adresára. Na konci si opäť vymenia potvrdenia o úspešnom prenose dát.

## Simulácia chyby prenosu

Pri každom posielaní správy alebo súboru má používateľ na strane odosielateľa navyše možnosť simulovať chybu prenosu. Program sa ho opýta, či chce simulovať chybu, a ak áno, ďalej si od neho vypýta číslo paketu, ktorý chce skorumpovať. Korupcia funguje tak, že do položky kontrolnej sumy je vnesená chyba. Program na strane prijímateľa túto chybu deteguje a odosiela paket typu 3 – NACK. Do konzoly tiež vypíše informáciu o prijatí chybného paketu.

A computer screen with white text

Description automatically generated

A screenshot of a computer program

Description automatically generatedDjcbhfbv