# **Semestrálne zadanie: Komunikácia s využitím UDP protokolu**

**Autor: Samuel Roháč**

**Cvičiaci: Ing. Ladislav Zemko**

**Predmet: Počítačové a komunikačné siete**

**Dátum: 21.10.2024**

# **Obsah**

[**Semestrálne zadanie: Komunikácia s využitím UDP protokolu** 1](#_Toc180440270)

[**Obsah** 2](#_Toc180440271)

[**Úvod** 3](#_Toc180440272)

[*Opis projektu* 3](#_Toc180440273)

[*Používanie UDP* 3](#_Toc180440274)

[*Hlavné ciele* 3](#_Toc180440275)

[**Návrh Protokolu** 4](#_Toc180440276)

[*Štruktúra hlavičky protokolu* 4](#_Toc180440277)

[*Nadviazanie spojenia* 4](#_Toc180440278)

[*Kontrola integrity* 5](#_Toc180440279)

[*Zabezpečenie prenosu (ARQ)* 6](#_Toc180440280)

[*Udržiavanie spojenia (Keep-Alive)* 7](#_Toc180440281)

[*Simulácia chýb v procese* 7](#_Toc180440282)

[**Popis implementácie** 8](#_Toc180440283)

[*Implementačné prostredie* 8](#_Toc180440284)

[*Opis kódu a jeho štruktúry* 8](#_Toc180440285)

[*Diagram komponentov* 12](#_Toc180440286)

[**Plán testovania** 13](#_Toc180440287)

[*Typy testovania* 13](#_Toc180440288)

[*Nástroje* 15](#_Toc180440289)

[**Záver** 16](#_Toc180440290)

[*Zhrnutie výsledkov* 16](#_Toc180440291)

[*Možné vylepšenia* 16](#_Toc180440292)

[**Prílohy / Zdroje** 17](#_Toc180440293)

# **Úvod**

## [Obrázok, na ktorom je čierny, temnota Automaticky generovaný popis](#_Obsah)*Opis projektu*

Cieľom projektu je navrhnúť a implementovať P2P (Peer-to-Peer) aplikáciu, ktorá umožňuje priamu komunikáciu medzi dvoma uzlami v lokálnej Ethernet sieti. Aplikácia je postavená nad UDP (User Datagram Protocol) a využíva vlastný protokol na úrovni aplikačnej vrstvy. Hlavnými funkciami aplikácie sú výmena textových správ a prenos ľubovoľných súborov medzi dvoma uzlami (počítačmi). Oba uzly fungujú ako prijímač aj odosielateľ súčasne, čo umožňuje plnohodnotnú obojsmernú komunikáciu

## [Obrázok, na ktorom je čierny, temnota Automaticky generovaný popis](#_Obsah)*Používanie UDP*

UDP (User Datagram Protocol) bol zvolený ako základ pre vytvorenie vlastného protokolu z niekoľkých dôvodov:

* **Rýchlosť:** UDP je oproti TCP menej náročný na režijné spracovanie, čo umožňuje rýchlejší prenos dát bez potreby nadväzovania a udržiavania spojenia.
* **Jednoduchosť:** UDP neposkytuje záruku doručenia, čo umožňuje jednoduchší návrh protokolu a väčšiu kontrolu nad mechanizmami ako je kontrola integrity a opakovanie prenosu.
* **Flexibilita pre P2P aplikácie:** Keďže UDP nevyžaduje riadiace správy pre nadviazanie spojenia, je vhodný pre P2P aplikácie, kde obe strany môžu byť prijímateľom aj odosielateľom správ. To zjednodušuje komunikáciu medzi dvoma rovnocennými uzlami v sieti.

## [Obrázok, na ktorom je čierny, temnota Automaticky generovaný popis](#_Obsah)*Hlavné ciele*

* **Nadviazanie spojenia a výmena riadiacich správ:** Implementovať mechanizmus na nadviazanie spojenia medzi dvoma uzlami, ktorý bude podobný TCP 3-Way Handshake. Tento mechanizmus umožní uzlom dohodnúť si parametre spojenia, ako je veľkosť fragmentov alebo časové intervaly pre kontrolné správy.
* **Prenos textových správ:** Umožniť odosielanie a prijímanie textových správ medzi uzlami. Každá správa bude obsahovať vlastnú hlavičku, ktorá umožní identifikáciu typu správy, veľkosti a kontrolnej sumy.
* **Prenos súborov a kontrola integrity prenášaných dát:** Implementovať fragmentáciu veľkých súborov na menšie časti, ktoré budú odosielané postupne. Na overenie správnosti prijatých dát sa bude používať kontrolná suma (CRC16), čo zabezpečí detekciu a opätovné odoslanie poškodených dátových fragmentov.

# **Návrh Protokolu**

## [Obrázok, na ktorom je čierny, temnota Automaticky generovaný popis](#_Obsah)*Štruktúra hlavičky protokolu*

Protokol nad UDP definuje vlastnú hlavičku, ktorá obsahuje potrebné informácie na správu komunikácie medzi uzlami. Štruktúra hlavičky umožňuje správne spracovanie prijatých správ, identifikáciu typu dát a kontrolu integrity.  
**grafické znázornenie:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Verzia | Typ správy | ID správy | Dĺžka správy | Kontrolná suma | Dáta |

* Verzia (1 byte)
  + Určuje verziu protokolu, čo umožňuje budúce rozšírenia a spätnú kompatibilitu. V prípade zmien v protokole môže príjemca správy rozlíšiť, s akou verziou pracuje.
* Typ správy (1 byte)
  + 0 - Riadiaca správa (INIT, ACK, READY)
  + 1 - Textová správa (prenáša textový obsah)
  + 2 - Dátový fragment (časť súboru)
* ID správy (4 byty)
  + unikátne identifikačné číslo správy. Používa sa na identifikáciu správy pri retransmisiách a pri opakovaniach prenosu.
* Dĺžka správy (2 byty)
  + Určuje veľkosť dátovej časti v bajtoch, čo umožňuje príjemcovi vedieť, koľko dát očakáva
* Kontrolná suma (2 byty)
  + Vypočítaná pomocou CRC16, slúži na overenie integrity správy. Príjemca overí, či sa kontrolná suma správy zhoduje s vypočítanou hodnotou.
* Dáta (ľubovoľná veľkosť v závislosti od veľkosti buffera)
  + Obsah samotnej správy. Môže ísť o text, časť súboru alebo inú informáciu v závislosti od typu správy

## [Obrázok, na ktorom je čierny, temnota Automaticky generovaný popis](#_Obsah)*Nadviazanie spojenia*

Proces nadviazania spojenia sa podobá na TCP 3-Way Handshake, čo zaručuje obom uzlom dohodu na parametre spojenia a pripravenosť komunikácie.

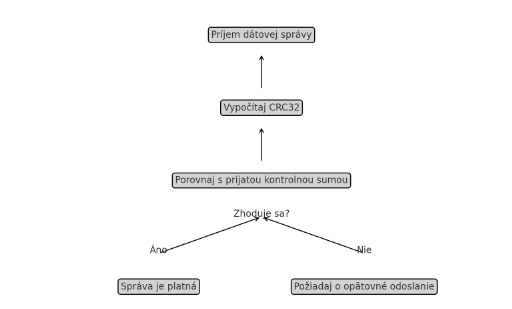
Proces:

1. **INIT správa:** Uzol A (iniciátor spojenia) pošle uzlu B riadiacu INIT správu, v ktorej informuje o svojom zámere nadviazať spojenie.
2. **ACK správa:** Uzol B odpovie ACK správou, čím potvrdí prijatie INIT správy a ochotu pokračovať v nadväzovaní spojenia.
3. **READY správa:** Uzol A odpovie správou READY, čím potvrdí, že je pripravený prijímať správy od uzla B. Spojenie je týmto nadviazané a uzly môžu začať s výmenou dát.

## [Obrázok, na ktorom je čierny, temnota Automaticky generovaný popis](#_Obsah)Obrázok, na ktorom je snímka obrazovky, rad, text, rovnobežný Automaticky generovaný popis*Kontrola integrity*

Na kontrolu integrity prenášaných dát sa používa CRC16

* **Použitie CRC16**:
  + CRC16 generuje kontrolnú sumu s veľkosťou 2 byty, ktorá sa vypočíta na základe obsahu správy (dáta a hlavička).
  + Príjemca správy po prijatí správy vypočíta CRC16 z prijatých dát a porovná ju s hodnotou, ktorá bola prijatá v hlavičke správy.
  + Ak sa hodnoty nezhodujú, správa je považovaná za poškodenú a je potrebné požiadať o jej opätovné odoslanie.
* Výpočet
  + Výpočet CRC16 je založený na polynóme, ktorý sa používa na vytvorenie kontrolnej sumy. Základným princípom je rozdelenie správy (sekvencie bitov) polynómom, čo vytvorí zvyšok delenia, ktorý predstavuje kontrolnú sumu.
  + **Inicializácia**: Začnite s 16-bitovým registrovaním CRC inicializovaným na určitú hodnotu (často je to 0xFFFF alebo 0x0000, v závislosti od použitého protokolu).
  + **Spracovanie bajtov**: Pre každý bajt správy vykonajte nasledovné:
    - XOR (logický súčet modulo 2) s najvyšším bajtom aktuálneho CRC registra.
    - Pre každý bit bajtu (spracovávaného zľava doprava):
      * Ak je najvyšší bit v CRC registre 1, posuňte CRC o jeden bit doľava a XOR ho s polynómom.
      * Ak je najvyšší bit v CRC registre 0, iba posuňte CRC o jeden bit doľava.
  + **Získanie výsledku**: Po spracovaní všetkých bajtov správy bude výsledný zvyšok v registri CRC predstavovať 16-bitovú kontrolnú sumu.



vypočítaj CRC16

## [Obrázok, na ktorom je čierny, temnota Automaticky generovaný popis](#_Obsah)*Zabezpečenie prenosu (ARQ)*

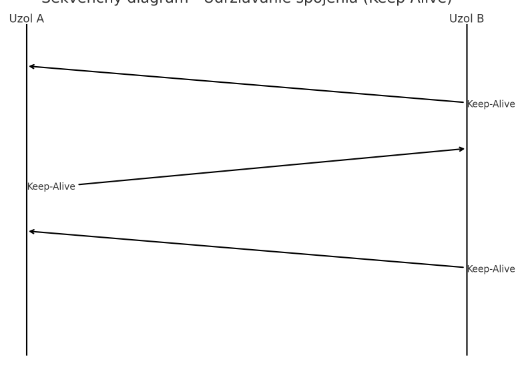
ARQ mechanizmus sa používa na zabezpečenie spoľahlivého prenosu dát a opätovného odoslania stratených alebo poškodených správ.

* **Mechanizmus opakovaného odosielania:**
  + Každá správa má svoj unikátny ID správy, ktoré príjemca použije na potvrdenie prijatia pomocou ACK správy.
  + Po odoslaní správy uzol čaká na ACK správu. Ak ju nedostane do určitého časového limitu (timeout), správu odošle znova.
  + Obrázok, na ktorom je rad, text, rovnobežný, diagram

    Automaticky generovaný popisPríjemca po prijatí fragmentu pošle ACK správu späť odosielateľovi s potvrdením prijatia konkrétneho fragmentu.
  + Ak odosielateľ dostane ACK správu, prejde na odosielanie ďalšieho fragmentu.

## [Obrázok, na ktorom je čierny, temnota Automaticky generovaný popis](#_Obsah)*Udržiavanie spojenia (Keep-Alive)*

Mechanizmus na overenie, či je druhý uzol stále pripojený a aktívny.

* **Popis mechanizmu:**
  + Oba uzly periodicky odosielajú špeciálne Keep-Alive správy s cieľom informovať druhú stranu o tom, že sú stále pripojené.
  + Ak uzol nedostane Keep-Alive správu od druhého uzla v definovanom časovom intervale, predpokladá, že spojenie je stratené, a pokúsi sa znovu nadviazať spojenie alebo ukončí spojenie.

## [Obrázok, na ktorom je čierny, temnota Automaticky generovaný popis](#_Obsah)*Simulácia chýb v procese*

Pre testovanie ARQ mechanizmu je možné úmyselne simulovať chyby v prenose dát.

* Možnosť úmyselne poškodiť správu:
  + Pri odosielaní správ je možné upraviť kontrolnú sumu alebo zmeniť obsah správy, čím sa simuluje poškodenie dát.
  + Príjemca deteguje nesúlad medzi kontrolnou sumou v správe a vypočítanou hodnotou, čo vyvolá požiadavku na opätovné odoslanie správy.
  + Takáto simulácia umožňuje otestovať správnosť implementácie ARQ a spoľahlivosť prenosového mechanizmu.
* Popis testovania:
  + Zmena jedného bitu v správe pred odoslaním.
  + Príjemca zistí chybu a požaduje opätovné odoslanie.
  + Testovanie sa vykonáva s cieľom overiť, či ARQ mechanizmus správne reaguje na detekciu chýb.

# **Popis implementácie**

## [Obrázok, na ktorom je čierny, temnota Automaticky generovaný popis](#_Obsah)*Implementačné prostredie*

* **Programovací jazyk:**  
  Implementácia P2P aplikácie je realizovaná v programovacom jazyku **Python**, ktorý je vhodný pre svoju jednoduchosť, rozsiahlu komunitu a dostupnosť knižníc, ktoré uľahčujú prácu so sieťovými protokolmi.
* **Knižnice:**
  + **socket**: Základná knižnica pre prácu so sieťovými protokolmi v Pythone. Umožňuje vytvárať UDP sokety, odosielať a prijímať dátové pakety.
  + **threading**: Knižnica pre prácu s vláknami. Umožňuje simultánne vykonávanie viacerých úloh, čo je potrebné na to, aby aplikácia mohla súčasne počúvať a odosielať správy. Táto knižnica zabezpečuje, že používateľ môže v reálnom čase odosielať správy a prijímať správy z druhého uzla.

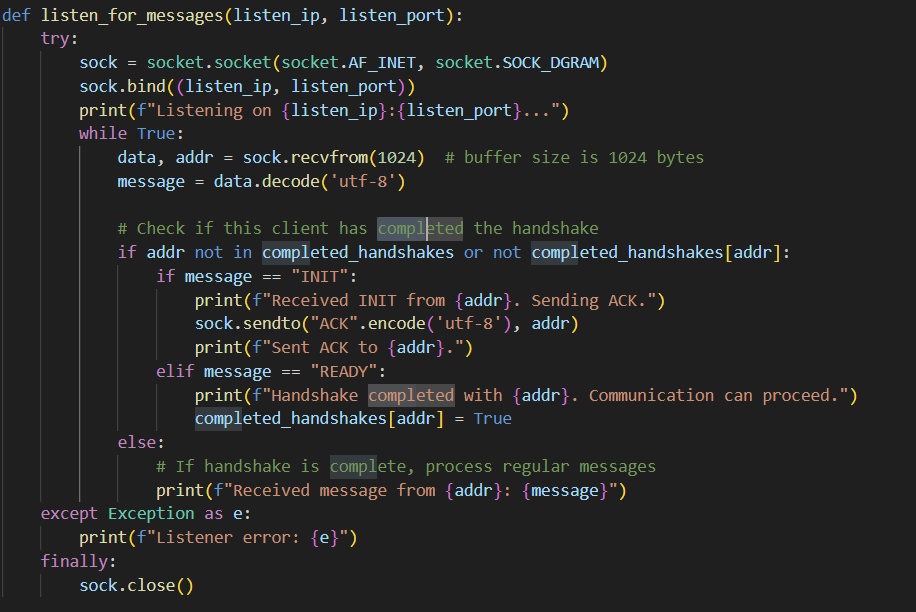
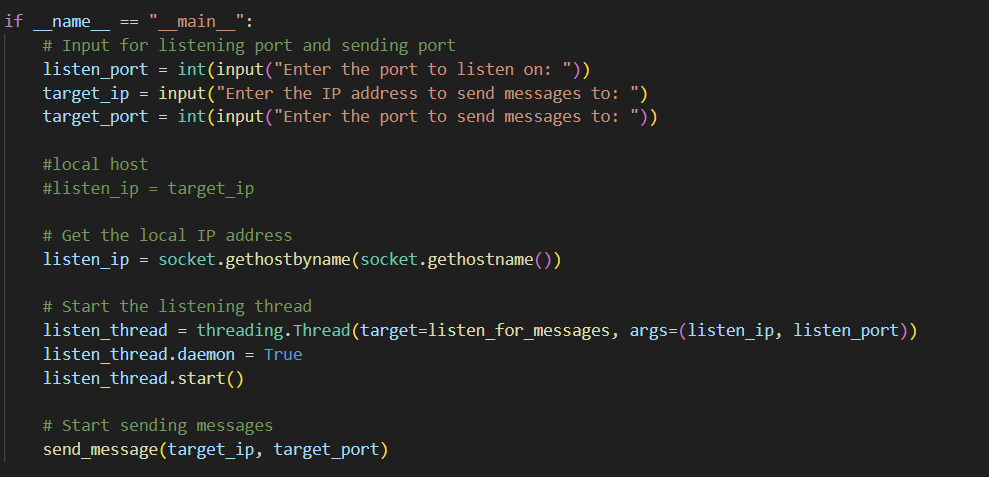
## [Obrázok, na ktorom je čierny, temnota Automaticky generovaný popis](#_Obsah)*Opis kódu a jeho štruktúry*

Tento kód implementuje jednoduchý systém komunikácie medzi dvoma aplikáciami pomocou UDP protokolu. Kód sa skladá z dvoch hlavných funkcií: **listen\_for\_messages** a **send\_message**, ktoré pracujú na základe handshake mechanizmu pre zaistenie, že komunikácia medzi klientom a serverom je správne inicializovaná.

**Hlavné časti kódu:**

1. **Import knižníc:**
   * socket: Pre vytvorenie UDP spojenia.
   * threading: Na spustenie vlákna pre prijímanie a odosielanie správ súčasne.



1. **Funkcia listen\_for\_messages:**
   * Prijíma správy na zadanom IP a porte.
   * Odpovedá na správu "INIT" s "ACK", čím signalizuje, že prijalo žiadosť o inicializáciu.
   * Ak dostane "READY", označí stav handshake pre daný adres ako dokončený.
   * Po úspešnom handshaku prijíma ďalšie správy od klienta a vypisuje ich na obrazovku.
2. **Funkcia send\_message:**
   * Iniciuje handshake tým, že pošle "INIT" správy na cieľový IP a port.
   * Očakáva odpoveď "ACK", potom odošle "READY" a nastaví handshake na dokončený.
   * Po dokončení handshake umožňuje používateľovi zadávať správy, ktoré sú odosielané na server.
3. **Hlavná časť programu (\_\_main\_\_):**
   * Získa IP adresu zariadenia a porty pre príjem a odosielanie správ od používateľa.
   * Spustí vlákno pre príjem správ (listen\_for\_messages) a zároveň spustí odosielanie správ cez send\_message.
4. **Pracovný tok:**
   * Program začína tým, že sa spustí vlákno na príjem správ.
   * Paralelne beží funkcia na odosielanie správ, kde používateľ môže inicializovať handshake.
   * Po úspešnom handshaku medzi klientom a serverom prebieha komunikácia prostredníctvom textových správ.

## [Obrázok, na ktorom je čierny, temnota Automaticky generovaný popis](#_Obsah)*Diagram komponentov*

Obrázok, na ktorom je text, snímka obrazovky, diagram, rad

Automaticky generovaný popis

# **Plán testovania**

Plán testovania zahŕňa overenie správnej funkčnosti implementovaného protokolu nad UDP, jeho schopnosť nadviazať spojenie medzi dvoma uzlami, spoľahlivý prenos textových správ a súborov, a reakcie na rôzne chybové scenáre. Testovanie bude vykonávané pomocou simulácií v lokálnej sieti, monitorovaním prenosu pomocou nástroja Wireshark a pozorovaním správneho správania aplikácie v rôznych podmienkach.

## [Obrázok, na ktorom je čierny, temnota Automaticky generovaný popis](#_Obsah)*Typy testovania*

1. **Test nadviazania spojenia medzi uzlami:**
   * **Cieľ:** Overiť správne nadviazanie spojenia medzi dvoma uzlami pomocou riadiacich správ (INIT, ACK, READY).
   * **Postup:**
     + Spustiť aplikáciu na oboch uzloch.
     + Uzol A odošle INIT správu uzlu B.
     + Overiť, že uzol B prijal INIT správu a poslal späť ACK správu.
     + Overiť, že uzol A prijal ACK správu a odoslal READY správu.
     + Overiť vo Wiresharku, že riadiace správy boli správne odoslané a prijaté.
   * **Očakávaný výsledok:** Spojenie sa úspešne nadviaže a oba uzly sú pripravené na výmenu dát.
2. **Test odosielania a prijímania textových správ:**
   * **Cieľ:** Otestovať schopnosť uzlov odosielať a prijímať textové správy.
   * **Postup:**
     + Po nadviazaní spojenia odoslať textovú správu z uzla A na uzol B.
     + Overiť, že uzol B správu prijal a zobrazil obsah.
     + Odpovedať textovou správou z uzla B na uzol A a overiť, že uzol A správu správne prijal.
     + Sledujte prenos správ vo Wiresharku.
   * **Očakávaný výsledok:** Textové správy sú správne prijaté a zobrazia sa na oboch uzloch bez straty alebo poškodenia.
3. **Test prenosu súborov:**
   * **Cieľ:** Overiť správny prenos súborov medzi uzlami pomocou fragmentácie dát.
   * **Postup:**
     + Vybrať malý súbor (napr. 10 KB) a odoslať ho z uzla A na uzol B.
     + Overiť, že uzol B prijal všetky fragmenty a správne ich zostavil do pôvodného súboru.
     + Zopakovať test s väčším súborom (napr. 1 MB) a sledovať prenos fragmentov vo Wiresharku.
   * **Očakávaný výsledok:** Súbor je úspešne odoslaný, prijatý a rekonštruovaný bez straty dát.
4. **Test kontroly integrity (CRC16):**
   * **Cieľ:** Overiť, že CRC16 mechanizmus deteguje poškodené dáta a žiada o opätovné odoslanie.
   * **Postup:**
     + Zaslať správu z uzla A a umelo poškodiť kontrolnú sumu pred jej odoslaním.
     + Overiť, že uzol B deteguje nesúlad kontrolnej sumy a pošle požiadavku na opätovné odoslanie.
     + Uzol A následne správu znova odošle.
     + Sledovať tento proces vo Wiresharku.
   * **Očakávaný výsledok:** Poškodená správa je správne detegovaná a úspešne retransmitovaná.
5. **Test timeout mechanizmu :**
   * **Cieľ:** Otestovať správnu funkciu timeout mechanizmu pre prípad straty ACK správy.
   * **Postup:**
     + Zaslať fragment z uzla A na uzol B, ale na uzle B umelo zablokovať zaslanie ACK správy.
     + Overiť, že uzol A po uplynutí timeoutu znova odošle fragment.
     + Sledovať retransmisiu vo Wiresharku.
   * **Očakávaný výsledok:** Uzol A správne deteguje neprijatú ACK správu a fragment odošle znova.
6. **Test udržiavania spojenia (Keep-Alive):**
   * **Cieľ:** Overiť, že uzly správne udržujú spojenie pomocou Keep-Alive správ.
   * **Postup:**
     + Po nadviazaní spojenia monitorovať periodické zasielanie Keep-Alive správ medzi uzlami.
     + Simulovať výpadok jedného uzla (napr. vypnutie aplikácie) a overiť, že druhý uzol deteguje stratu spojenia.
   * **Očakávaný výsledok:** Uzly si správne vymieňajú Keep-Alive správy a v prípade výpadku detegujú stratu spojenia.
7. **Test na reálnych uzloch v lokálnej sieti:**
   * **Cieľ:** Overiť funkčnosť aplikácie v reálnom prostredí mimo localhostu.
   * **Postup:**
     + Spustiť aplikáciu na dvoch fyzických počítačoch v rovnakej lokálnej sieti.
     + Vykonať test nadviazania spojenia, odoslania textovej správy a prenosu súboru.
     + Monitorovať prenos pomocou Wiresharku na oboch uzloch.
   * **Očakávaný výsledok:** Aplikácia správne funguje aj v reálnej lokálnej sieti a komunikuje medzi dvoma fyzickými zariadeniami.

## [Obrázok, na ktorom je čierny, temnota Automaticky generovaný popis](#_Obsah)*Nástroje*

* **Wireshark:**  
  Wireshark bude použitý na sledovanie sieťovej komunikácie medzi uzlami. Pomocou neho bude možné analyzovať odoslané a prijaté UDP pakety, sledovať riadiace správy (INIT, ACK, READY), textové správy, dátové fragmenty a Keep-Alive správy. Wireshark umožní overiť správnosť nadväzovania spojenia a sledovať retransmisiu stratených alebo poškodených správ.
* **Simulácia chýb:**  
  V rámci implementácie sa pridá možnosť úmyselne meniť obsah správy (napr. zmena kontrolnej sumy), čo umožní otestovať, ako systém reaguje na poškodené dáta a či správne funguje mechanizmus opakovaného odoslania.

# **Záver**

V rámci tohto projektu bola navrhnutá a implementovaná P2P aplikácia, ktorá využíva vlastný protokol postavený nad UDP pre spoľahlivú výmenu textových správ a prenos súborov medzi dvoma uzlami v lokálnej sieti. Aplikácia je schopná nadviazať spojenie, udržiavať ho a zabezpečiť spoľahlivý prenos dát aj v prípade, že sa niektoré správy stratia alebo sú poškodené. Pri implementácii boli využité mechanizmy, ktoré umožňujú efektívnu a bezpečnú komunikáciu medzi uzlami, čo z aplikácie robí robustný nástroj pre prenos dát v lokálnom prostredí.

## [Obrázok, na ktorom je čierny, temnota Automaticky generovaný popis](#_Obsah)*Zhrnutie výsledkov*

## [Obrázok, na ktorom je čierny, temnota Automaticky generovaný popis](#_Obsah)*Možné vylepšenia*

# [Obrázok, na ktorom je čierny, temnota Automaticky generovaný popis](#_Obsah)**Prílohy / Zdroje**

**https://github.com/SamuelRohac/PKS\_UDP**

[**https://pythontic.com/modules/socket/udp-client-server-example**](https://pythontic.com/modules/socket/udp-client-server-example)

[**https://www.geeksforgeeks.org/user-datagram-protocol-udp/?ref=gcse\_outind**](https://www.geeksforgeeks.org/user-datagram-protocol-udp/?ref=gcse_outind)

[**https://www.geeksforgeeks.org/tcp-3-way-handshake-process/?ref=header\_outind**](https://www.geeksforgeeks.org/tcp-3-way-handshake-process/?ref=header_outind)

[**https://sk.wikipedia.org/wiki/Pou%C5%BE%C3%ADvate%C4%BEsk%C3%BD\_datagramov%C3%BD\_protokol**](https://sk.wikipedia.org/wiki/Pou%C5%BE%C3%ADvate%C4%BEsk%C3%BD_datagramov%C3%BD_protokol)

[**https://en.wikipedia.org/wiki/Cyclic\_redundancy\_check**](https://en.wikipedia.org/wiki/Cyclic_redundancy_check)

[**https://en.wikipedia.org/wiki/Keepalive**](https://en.wikipedia.org/wiki/Keepalive)

[**https://www.umletino.com/umletino.html**](https://www.umletino.com/umletino.html)

[**https://chatgpt.com/?model=gpt-4o**](https://chatgpt.com/?model=gpt-4o)