**Komunikácia s využitím UDP protokolu**

*Počítačové a komunikačné siete*

[xfridrichj@stuba.sk](mailto:xfridrichj@stuba.sk)

Obsah

[Zadanie 2](#_Toc180444057)

[Návrh 2](#_Toc180444058)

[Druhy správ a Flagy 2](#_Toc180444059)

[Hlavička 2](#_Toc180444060)

[Kontrola integrity poslanej správy (CRC) 3](#_Toc180444061)

[Prenos textu/súboru 3](#_Toc180444062)

[Zabezpečenie spoľahlivého prenosu dát (ARQ) 3](#_Toc180444063)

[Proces odosielania / doručovania dát 3](#_Toc180444064)

[Vývojový diagram 5](#_Toc180444065)

[Použité metódy 5](#_Toc180444066)

[Záver 6](#_Toc180444067)

[Zdroje 6](#_Toc180444068)

## Zadanie

Navrhnúť a implementovať P2P aplikáciu využívajúcu vlastný protokol postavený nad UDP (User Datagram Protocol) v transportnej vrstve sieťového modelu TCP/IP. Aplikácia bude umožňovať komunikáciu dvoch účastníkov v lokálnej Ethernet sieti, vrátane výmeny textu a prenosu ľubovoľných súborov medzi počítačmi (uzlami). Oba uzly budú fungovať súčasne ako prijímač aj odosielateľ.

Zadanie pozostáva z dvoch častí: teoretickej a praktickej. V teoretickej časti navrhnete vlastný protokol nad UDP. V praktickej časti následne implementujete a predvediete jeho funkčnosť pri komunikácii medzi dvoma uzlami, pričom zdôrazníte úpravy vykonané počas implementácie v porovnaní s pôvodným návrhom.

## Návrh

### **Druhy správ a Flagy**

|  |  |
| --- | --- |
| **Pozícia bitu** | **Význam Flagu** |
| 0000 0001 | ACK signál |
| 0000 0010 | SYN signál |
| 0000 0100 | FIN signál |
| 0000 1000 | Signál opakovaného poslania |
| 0001 0000 | Bežná správa |
| 0010 0000 | Súborová správa |

### **Hlavička**

Navrhnutá hlavička má 8B:

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

Automaticky generovaný popis

**Message type + Flags –** výber z uvedených vyššie - kombinovaná veľkosť 1B

**Payload size –** veľkosť odoslaného fragmentu v bitoch (0-65 536) - 2B

**Fragment offset –** indikácia štartovného bitu fragmentovaných dát (0-65 536) - 2B

**Total Fragments –** celkový počet odosielaných fragmentov (0-255) – 1B

**Checksum –** zvyšok po delení 65 536 – 2B

**Data –** samotná správa

## Kontrola integrity poslanej správy (CRC)

Na kontrolu integrity bude použitý Cyklický Redundantný Súčet, konkrétne CRC16. Tento mechanizmus využíva 16-bitový kontrolný súčet na detekciu chýb počas prenosu dát. Proces prebieha nasledovne: Dáta sa spracujú pomocou algoritmu, ktorý na základe definovaného polynómu vypočíta kontrolný súčet. Tento súčet sa následne pridá k prenášaným dátam. Príjemca opäť vypočíta CRC z prijatých dát a porovná ho s pripojeným kontrolným súčtom, aby overil, či pri prenose nedošlo k chybe. Pre tento proces bude použitý známy CRC16 polynóm (0x8005)

## Prenos textu/súboru

### **Zabezpečenie spoľahlivého prenosu dát (ARQ)**

Na zabezpečenie spoľahlivého prenosu dát bude v projekte implementovaná metóda **Selective Repeat (SR)**. Táto metóda umožňuje odosielanie viacerých paketov bez čakania na potvrdenie (ACK) pre každý paket zvlášť, na rozdiel od metódy **Go-Back-N**. Prijímateľ môže prijať pakety v ľubovoľnom poradí a posiela späť ACK len pre tie pakety, ktoré prijal správne. Metóda Selective Repeat bola vybraná pre jej výhody, ako je efektívnejší prenos dát a skutočnosť, že oproti metóde Go-Back-N opakuje len tie pakety, ktoré boli stratené alebo poškodené, čím zvyšuje rýchlosť a efektivitu prenosu.

### **Proces odosielania / doručovania dát**

Po spustení programu si používateľ môže zvoliť svoju úlohu v P2P komunikácii, či bude **odosielať dáta (klient)** alebo **prijímať dáta (server)**.

Ak používateľ zvolí rolu **klient**, bude mať možnosť zadať adresu (IP servera) a port (port servera), na ktorý plánuje odoslať správu. Ďalej musí zadať typ správy (text alebo súbor). V prípade textu zadá samotnú správu, v prípade súboru musí uviesť cestu k súboru. Používateľ si môže tiež zvoliť veľkosť fragmentu. Ak nie je zadaná žiadna hodnota, nastaví sa predvolená hodnota 0. Po zadaní všetkých údajov klient odošle **SYN správu** na vybraný server. Ak nepríde odpoveď od servera, používateľ bude vyzvaný zadať adresu a port znova. Ak server odpovie správou **SYN-ACK**, klient pošle **ACK** a tým sa inicializuje komunikácia. Správa sa následne rozdelí na fragmenty podľa zvolených parametrov a začne sa odosielanie na server.

Ak používateľ zvolí rolu **server**, musí zadať dostupný port na prijímanie dát. Následne server čaká na **SYN správu** od klienta. Po jej prijatí server odošle klientovi **SYN-ACK** a čaká na **ACK** správu od klienta. Po prijatí ACK je spojenie nadviazané a server môže začať prijímať fragmenty. V prípade, že niektorý fragment je poškodený, server odošle klientovi správu o poškodení a požaduje opätovné odoslanie tohto fragmentu (**Selective Repeat**). Poškodenie fragmentu sa zisťuje pomocou CRC. Ak fragmenty dorazia bez chyby, server pokračuje v čakaní na ďalšie dáta alebo fragmenty.

Obrázok, na ktorom je text, rad, diagram, rovnobežný

Automaticky generovaný popis

*Obrázok 1 Opis procesu 3 way handshake a prenosu chybného fragmentu*

V oboch rolách je implementovaný mechanizmus udržiavania spojenia pomocou tzv. „heartbeat“ správ. Každých 5 sekúnd sa odosiela heartbeat správa, ak medzi zariadeniami neprebieha žiadna aktivita. Ak jedna zo strán neobdrží odpoveď na tri po sebe odoslané heartbeat správy, spojenie sa ukončí.

## Vývojový diagram

Obrázok, na ktorom je diagram, plán, technický výkres, schematický

Automaticky generovaný popis

## Použité metódy

*Trieda Header:*

**build\_packet(self)** – poskladá jednotlivé zložky a samotnú správu do jedného packetu

**parse\_packet(packet)**– rozoberie paket na jednotlivé zložky a vráti ich oddelené

*Trieda Peer:*

Táto trieda slúži ako základ pre objekty Klient a Server.

**\_\_init\_\_(self, ip, port)**- inicializácia hodnôt

**send\_message(self, message, receiver=None)** – slúži na odoslanie SYN, SYN-ACK, ACK správ

**receive\_packet(self)** – pomocou threadovania príjme správu, rozoberie ju na zložky a printne ju užívateľovi

**send\_packet(self)** – odošle poskladaný paket používateľom zvolenému Peer-ovi

**chatting(self)** – prebieha tu threadovanie a používa sa na posielanie správ P2P

**quit(self)** – metóda na ukončenie threadovania

*Trieda Client (Peer):*

**three\_way\_hs\_c(self, server\_ip, server\_port)** – klientova verzia *3 way handshake*-u, odošle SYN signál, očakáva SYN-ACK signál a odošle ACK signál

*Trieda Server (Peer):*

**three\_way\_hs\_s(self)** – serverová verzia *3 way handshake*-u, hneď po spustení čaká na SYN signál, odošle SYN-ACK klientovi a zapamätá si klientove IP a port, čaká na ACK signál

## Záver

Predstavený program slúži na komunikáciu medzi 2 uzlami, čo znamená prenos súborov, ich prípadné delenie na časti / fragmenty, pridanie vlastne navrhnutej hlavičky a odosielanie fragmentu/fragmentov, kontrolovanie integrity zaslaných fragmentov a spájanie do celku.

## Zdroje

<https://github.com/fiit-ba/PKS-course-2425/tree/main/assignments/communication_over_udp#literat%C3%BAra>

<https://www.infineon.com/dgdl/Infineon-Component_CRC_V2.20-Software%20Module%20Datasheets-v02_05-EN.pdf?fileId=8ac78c8c7d0d8da4017d0e854974162c>

<https://www.geeksforgeeks.org/difference-between-go-back-n-and-selective-repeat-protocol/>

prednášky a cviká