Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Ilkovičova 6276/2

842 16 Bratislava 4

Semestrálne zadanie

**Komunikácia s využitím UDP protokolu**

**Adam** **Strelec**

Akademický rok: 2024/2025

Predmet: PKS

AIS ID: 127275

Cvičiaci: Bc. Marián Rohun

**Obsah**

**Úvod ........................................................................................... 3**

**Cieľ zadania .......................................................................3**

**Teoretická časť ............................................................................ 3**

**Štruktúra hlavičky protokolu .............................................. 3**

**Opis metódy na overenie integrity prenesenej správy ………. 4**

**Opis metódy na udržanie spojenia ...................................... 7**

**Opis metódy na zabezpečenie spoľahlivého prenosu dát ….. 7**

**Diagram opisujúci predpokladané správanie uzlov .............. 9**

**Opis metódy na nadviazanie spojenia medzi dvoma**

**stranami a dohodnutie si parametrov spojenia .................... 10**

**Vytvorenie chyby ............................................................... 11**

**Fragmentácia .................................................................... 12**

**Ukončenie spojenia ........................................................... 12**

**Wireshark skript ................................................................ 13**

**Zdroje ............................................................................... 13**

**Úvod**

**Cieľ zadania:**

Navrhnite a implementujte P2P aplikáciu, ktorá bude používať vlastný komunikačný protokol založený na UDP (User Datagram Protocol) v transportnej vrstve sieťového modelu TCP/IP. Aplikácia umožní dvom účastníkom komunikovať v lokálnej Ethernet sieti, vrátane možnosti posielať textové správy a prenášať rôzne súbory medzi počítačmi (uzlami). Oba uzly budú súčasne plniť funkciu odosielateľa aj prijímača.

**Teoretická časť:**

**Štruktúra hlavičky protokolu**

Pre náš TCP protokol využijeme len prvky ktoré sú naozaj potrebné na správne fungovanie

A screenshot of a computer

Description automatically generated

* + **Type(4b):**

|  |  |
| --- | --- |
| **Bitový stav** | **Funkcia** |
| 0000 | Žiadne dáta (napr. Keep-alive) |
| 0001 | Text (obyčajná tečxtová správa) |
| 0010 | Súbor |

* + **Msg(4b)**

|  |  |
| --- | --- |
| **Bitový stav** | **Funkcia** |
| 0000 | Inicializácia spojenia(SYN - začiatok 3WH) |
| 0010 | Potvrdenie spojenia(SYN-ACK, odpoveď na inicializáciu v 3WH) |
| 0011 | Potvrdenie prijatia (ACK - posledný krok 3WH) |
| 0100 | Odoslané dáta (klient -> server) |
| 0101 | Ukončenie spojenia (FIN - žiadosť o ukončenie spojenia) |
| 0110 | Potvrdenie ukončenia spojenia (FIN-ACK) |
| 1000 | Keep-alive informácia (na udržanie spojenia) |
| 1001 | Negatívne potvrdenie prijatia (NACK) |
| 1011 | Názov súboru (FILE\_NAME) |
| 1111 | Posledný fragment |

* **Sequence number (24b):**
  + Tiež možno nazvať aj číslo poradia packetu
  + Pri odosielaní packetov dochádza k fragmentácií, teda pri prijatí je potrebné poskladať fragmenty do pôvodného poradia na základe ich poradového čísla.
  + Ak niektoré číslo chýba, je potrebné vyslať správu a vyžiadať si packet znovu
* **Checksum (16b):**
  + Checksum (kontrolný súčet) slúži na detekciu chýb pri prenose dát
  + Pre náš protokol využijeme CRC16 (Vysvetlenie nižšie)
  + Odosielateľ aj prijímateľ vyrátajú svoj checksum
  + Výsledok si vypíšeme na konzolu v hexadecimálnom tvare
  + Ak sa hodnoty checksumu zhodujú, dáta prišli v poriadku, ak sú odlišné, niekde došlo k chybe a paket treba poslať znovu
* **Data():**
  + V tejto časti nasledujú už len dáta a hlavička končí
  + **Max veľkosť dát:** 1500B – 20B(IP) – 8(UDP) – 6(Moja hlavička) = 1466B
  + Maximálna veľkosť dát je 1466B. Hoci dáta v pakete môžu obsahovať až 1500B, musíme ešte odrátať miesto pre:
    - 20B pre IP hlavičku
    - 8B pre UDP hlavičku
    - 8B pre vlastnú hlavičku
  + Túto veľkosť obmedzujeme preto, aby sme predišli fragmentácií na linkovej vrstve

1. **Opis metódy na overenie integrity prenesenej správy**

V našom projekte implementuje metódu CRC16 (Cyclic redundancy check) na overenie integrity prenesenej správy. Túto metódu využijú obidve strany spojenia a výsledok musí byť na oboch stranách rovnaký. Ak výsledok rovnaký nebude, vyšle sa signál odosielateľovi správy že daný paket treba poslať znovu.

* **CRC výpočet vyzerá nasledovne:**
  + Fragment dát preveď na typ byte

A black square with a white square

Description automatically generated

* + Určíme si generátor, ktorým budeme vykonávať operáciu XOR
    - 0xE8 (prevedieme do dvojkovej sústavy)
  + Za naše dáta si ešte dopíšeme 16 bitových núl, do buffera

A black background with white squares

Description automatically generated

* + Dáta posúvame doľava až kým na prvom mieste nedostaneme bitovú jednotku, dáta z buffera postupne preúvame do ľavej časti.

A number in a row

Description automatically generated with medium confidence

* + Jednotku opäť posunieme doľava(Nesmieme zabunúť aj na 0 z buffera)

A number in a row

Description automatically generated with medium confidence

* + Prých 16 bitov, ktoré sú náše dáta dáme do XOR-u s našim polnyómom

A number grid with numbers

Description automatically generated with medium confidence

* + Výsledok sa stane našimi novými dátami

A number and numbers in a row

Description automatically generated with medium confidence

* + Toto opakujeme pokiaľ buffer nebude prázdy
  + Výsledné bitové číslo je náš checksum
* **Prečo CRC16:**
  + **Spoľahlivosť**: CRC8 používa menší kontrolný súčet ako CRC16 a tak môže nastať problém s detekciou chýb
  + **Jednoduchosť**: Oproti CRC32, je CRC16 podstatne jednoduchšie a ľahšie na implementáciu

1. **Opis metódy na udržanie spojenia**

* Náš program implementuje metódu Keep-Alive na udržanie spojenia medzi zariadeniami, aj keď momentálne neprebieha žiadna komunikácia.
* Táto metóda je dočasne deaktivovaná počas prenosu dát.
* Ak však dôjde počas prenosu dát k prerušeniu komunikácie trvajúcemu 5 sekúnd, mechanizmus *Keep-Alive* sa automaticky reaktivuje.
* Po obdržaní potvrdenia ACK pre *Keep-Alive* správu sa funkcia opäť pozastaví, aby umožnila pokračovanie prenosu dát.
* Ak však počas aktívneho režimu *Keep-Alive* nedôjde k odpovedi na tri po sebe idúce *Keep-Alive* pakety, program považuje spojenie za prerušené a ukončí jeho činnosť.
* Po odoslaní celej správy sa mechanizmus *Keep-Alive* znovu spustí a pokračuje až do ukončenia programu alebo začiatku prenosu dát.

1. **Opis metódy na zabezpečenie spoľahlivého prenosu dát**

Na zabezpečenie spoľahlivého prenosu dát máme na výber niekoľko možností. V našom programe sme si zvolili Stop & Wait metódu.

* **Stop & Wait:**
  1. **Odoslanie jedného rámca:** Odosielajúce zariadenie vyšle jeden packet k prijímajúcemu zariadeniu.
  2. **Čakanie na potvrdenie (ACK):** Po odoslaní zariadenie prestane posielať ďalšie rámce a čaká na potvrdenie (ACK) od prijímajúceho zariadenia, že rámec bol prijatý.
  3. **Prijatie potvrdenia (ACK):** Ak prijímajúce zariadenie správne prijalo rámec, pošle ACK späť odosielateľovi.
  4. **Odoslanie ďalšieho rámca:** Po prijatí ACK odosielateľ pošle ďalší rámec.
  5. **Opätovné odoslanie:** Ak program príjme správu NACK odošle posledý paket znovu. To isté platí v prípade že prijme potvrdenie na poslaný Keep-Alive paket. Značí to, že komunikácia sa obnovila no posledný odoslaný paket nemusel byť korektne prijatý a preto sa odošle znovu.

A computer screen with white text

Description automatically generated

1. A black background with white squares

   Description automatically generated**Diagram opisujúci predpokladané správanie uzlov**
2. **Opis metódy na nadviazanie spojenia medzi dvoma stranami a dohodnutie si parametrov spojenia (Three-Way Handshake)**

V našom programe na nadviazane spojenie využívame proces zvaný Three-Way Handshake

1. **SYN (Synchronize):** Iniciátor pošle druhému zariadeniu požiadavku na začatie spojenia. Odošle servisnú správu SYN. Ak do 5 sekúnd nedostane odpoveď tak odošle SYN paket znovu
2. **SYN\_ACK (Synchronize-Acknowledgment):** Zariadenie prijme SYN správu a odpovie klientovi so správou SYN-ACK. Následné čaká na ACK aby mohol považovať spojenie za korektne nadviazané
3. **ACK (Acknowledgment):** Klient prijme SYN-ACK správu a odpovie ACK správou, čím potvrdí prijatie SYN-ACK. Spojenie je teraz nadviazané a môže sa začať výmena dát.
4. Po tomto kroku už zariadenia nadviazali spojenie a môže prebiehať komunikácia

**A screenshot of a computer screen

Description automatically generated**

**A computer screen with white text

Description automatically generated with medium confidence**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screen shot of a computer

Description automatically generated**

1. **Vytvorenie chyby**
   * Pri zadávaní údajov, ako je veľkosť paketu, správa alebo súbor, má používateľ možnosť zvoliť, či chce aktivovať simuláciu umelej chyby.
   * V prípade kladnej voľby sa upraví kontrolný súčet (*checksum*) prvého fragmentu, ktorý sa následne odošle s úmyselnou chybou.
   * Ak príjemca deteguje chybu a odošle späť negatívne potvrdenie (*NACK*), kontrolný súčet paketu sa nanovo prepočíta, čím sa chyba odstráni, a správny fragment sa opätovne odošle.
   * Na nižšie priloženom obrázku môžme pozorovať simulovanie chyby.
   * Prvý odoslaný paket ma zmenený checksum a následne na to je odoslaný NACK.

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. **Fragmentácia**
   * **Voľba veľkosti fragmentu**: Pred začiatkom prenosu dát je používateľ vyzvaný, aby zvolil veľkosť fragmentu. Maximálna povolená veľkosť je obmedzená na 1466 bajtov, aby sa predišlo fragmentácii na linkovej vrstve.
   * **Podmienky fragmentácie**:
     + Ak je veľkosť prenášaného obsahu menšia ako zadaná veľkosť fragmentu, k fragmentácii nedochádza.
     + Ak je obsah väčší, z celkového objemu dát sa postupne oddeľujú bloky dát zodpovedajúce zvolenej veľkosti fragmentu, až kým nie je celý obsah odoslaný.
   * **Spracovanie servisných správ**: Servisné správy sa vždy prenášajú ako celok bez fragmentácie, aby sa zachovala ich integrita a jednoduchšia správa na strane prijímateľa.
2. **Ukončenie spojenia**
   * Na korektné ukončenie spojenia v našom programe je implementovaný mechanizmus **Four-Way Handshake**, využívajúci flagy *FIN* a *FIN\_ACK.*
   * Proces ukončenia spojenia môže iniciovať ktorýkoľvek používateľ.
   * V prípade, že aktuálne neprebieha prenos dát, dôjde k okamžitej výmene paketov obsahujúcich flagy *FIN, FIN\_ACK* a následne *ACK,* čím sa spojenie korektne uzavrie.
   * Ak však počas prebiehajúceho prenosu dát niektorý používateľ požaduje ukončenie spojenia, paket s flagom *FIN* je odoslaný, avšak príjemca ho dočasne odloží.
   * Po dokončení prenosu dát príjemca spracuje odložený *FIN* paket a pokračuje v procese ukončovania spojenia podľa štandardného mechanizmu Four-Way Handshake.

A computer screen with black background

Description automatically generated

1. **Wireshark skript**
   * Skript vo wiresharku vie rozoznať naše pakety pokiaľ smerujú na určité, nami nastavené porty
   * Tieto pakety farebne rozlíši podľa toho či je to servisný paket (sivá), textový paket (modrá), alebo paket so súborom (zelená)

A screenshot of a computer

Description automatically generated

1. **Zdroje**
   * [**Lua skript**](https://www.youtube.com/watch?v=T8yRoh2Wp5A) **– 18.11.2024**
   * [**Stop & Wait protokol**](https://www.youtube.com/watch?v=LnbvhoxHn8M) **– 5.11.2024**
   * [**UDP protokol**](https://learn.microsoft.com/en-us/dotnet/api/system.net.sockets.udpclient?view=net-9.0) **– 30.10.2024**
   * [**Threading**](https://learn.microsoft.com/en-us/dotnet/api/system.threading.thread?view=net-8.0) **– 2.11.2024**