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

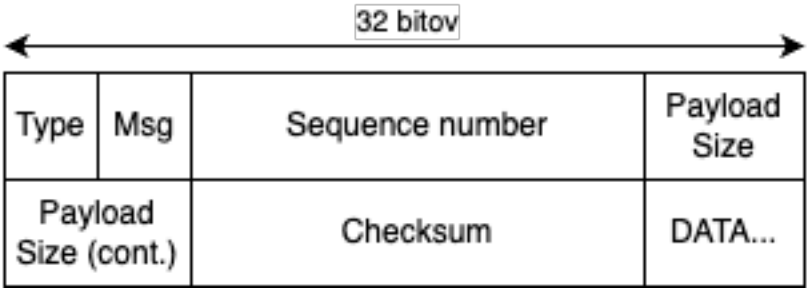
## Cieľ zadania:

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




- **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) |
| 0001 | Autentifikácia |
| 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) |
| 1111 | Posledný fragment |

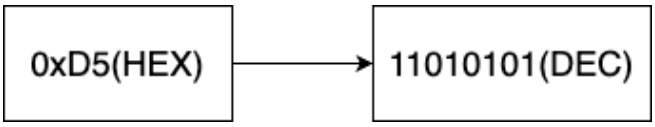
- **Sequence number (16B):**
  - Tiež možno nazvať aj číslo poradia packetu
  - Pri odosielaní packetov dochádza k fragmentácií, teda pri prijatí je potrebné poskladať fragmenty do pôvodného poradia na základe ich poradového čísla.
  - Ak niektoré číslo chýba, je potrebné vyslať správu a vyžiadať si packet znovu
- **Payload Size (16B):**
  - Obsahuje informáciu o množstve prenášaných dát vrámci jedného packetu
  - Nevzťahuje sa na hlavičku, tá sa do celkovej veľkosti packetu neráta
- **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
  - 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) – 7(Moja hlavička) = 1465B
  - Maximálna veľkosť dát je 1464B. 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

## 2. Opis metódy na overenie integrity prenesenej správy

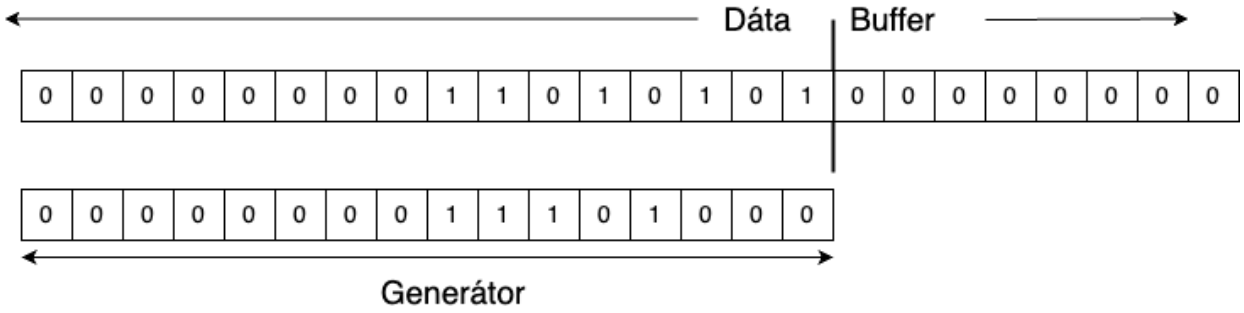
V našom projekte implementuje metódu CRC16 (Cyclic redundancy check) na overenie integrity prenesenej správy. Túto metódu využijú obidve strany spojenia

a výsledok musí byť na oboch stranách rovnaký. Ak výsledok rovnaký nebude, vyšle sa signál odosielateľovi správy že daný paket treba poslať znovu.

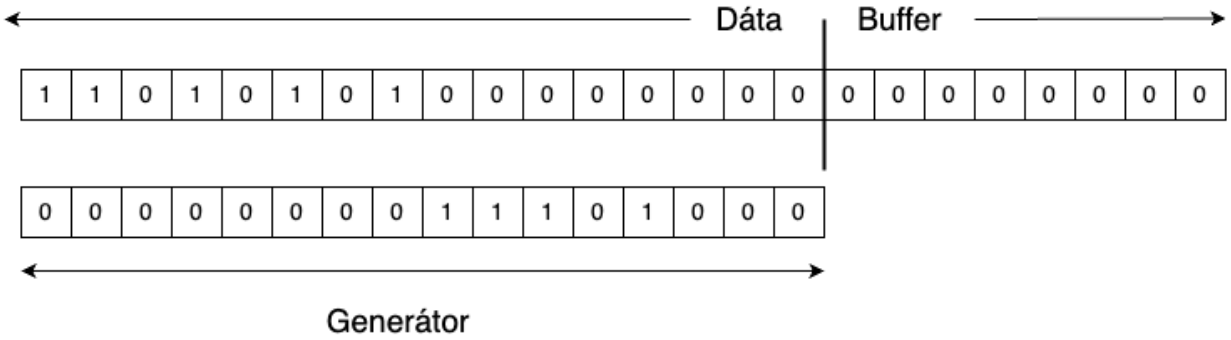
- **CRC výpočet vyzerá nasledovne:**
  - Fragment dát preveď na typ byte



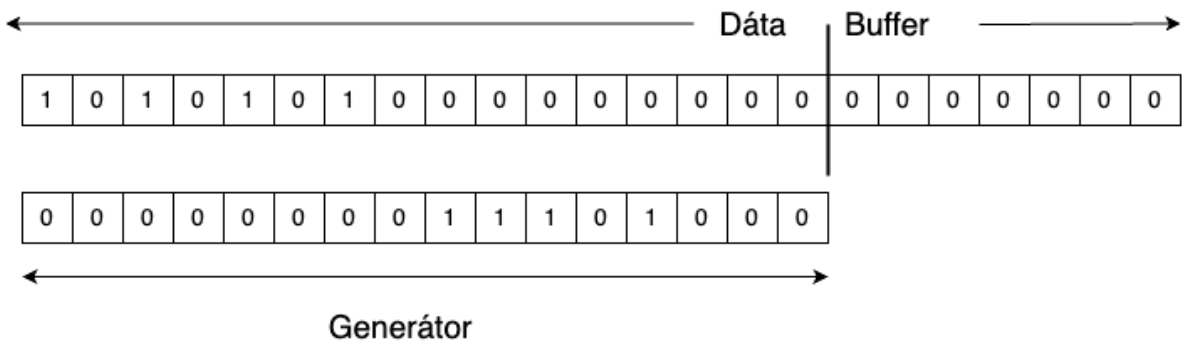

  - Určíme si generátor, ktorým budeme vykonávať operáciu XOR
    - 0xE8 (prevedieme do dvojkovej sústavy)
  - Za naše dáta si ešte dopíšeme 16 bitových núl, do buffera



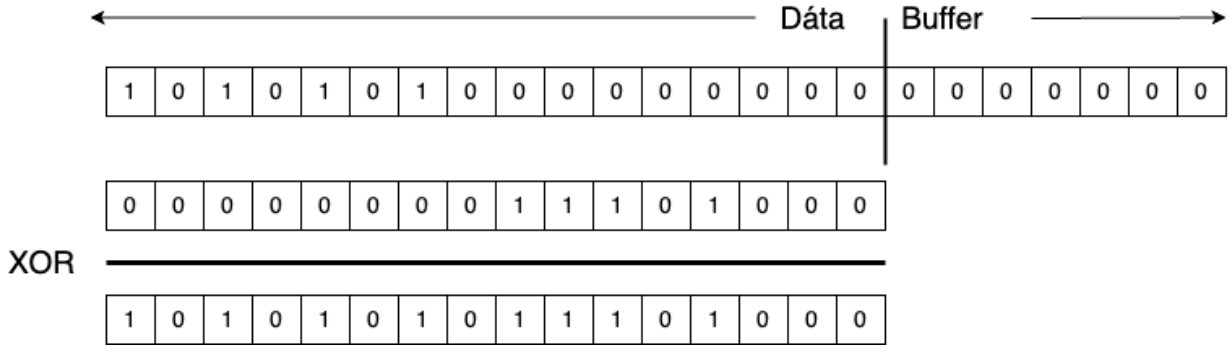

  - Dáta posúvame doľava až kým na prvom mieste nedostaneme bitovú jednotku, dáta z buffera postupne preúvame do ľavej časti.



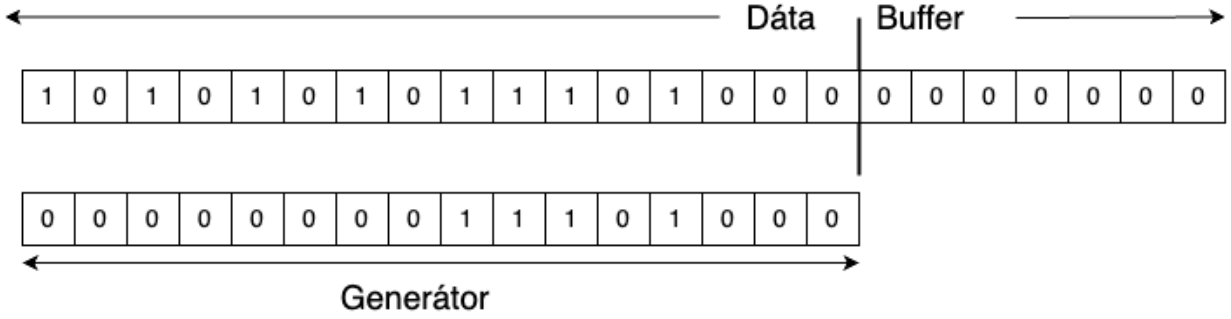

  - Jednotku opäť posunieme doľava(Nesmieme zabunúť aj na 0 z buffera)




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




- Výsledok sa stane našimi novými dátami




- 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

### 3. Opis metódy na udržanie spojenia

Náš program implementuje metódu Keep-Alive na udržanie spojenia medzi zariadeniami, aj keď momentálne neprebieha žiadna komunikácia.

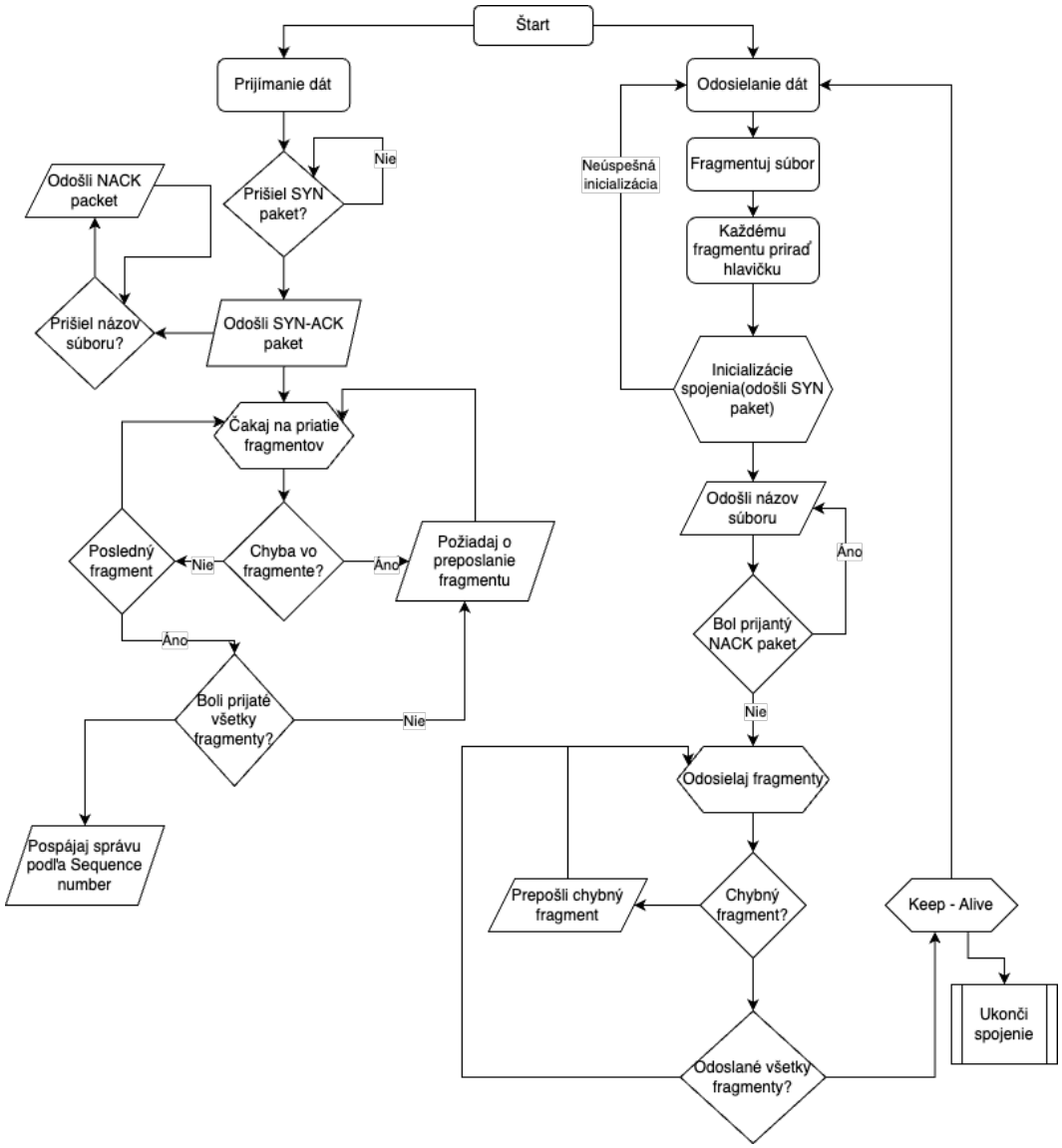
- **Keep -Alive:**
    - Po odoslaní celej správy sa spojenie neukončí automaticky, ale odošle sa správa KEEP ALIVE (v hlavičke sa *type* mastaví na 0000 a *msg* sa nastaví na 1000)
    - Táto správa sa bude posielať každých 5 sekúnd.
    - Ak jedno zo zariadní bude chcieť ukončiť komunikáciu, bude mať na to špeciálne vyhradenú funkciu

### 4. Opis metódy na zabezpečenie spoľahlivého prenosu dát

Na zabezpečenie spoľahlivého prenosu dát máme ny 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 pri strate:** Ak ACK nepríde v určenej lehote (timeout), odosielajúce zariadenie predpokladá stratu rámca a odošle ho znova.

## 5. Diagram opisujúci predpokladané správanie uzlov




## 6. 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):** Klient pošle serveru požiadavku na začatie spojenia. Táto správa obsahuje SYN bit nastavený na 1 a počiatočné poradové číslo (Sequence Number), ktoré klient plánuje použiť.
2. **SYN_ACK (Synchronize-Acknowledgment):** Server prijme SYN správu a odpovie klientovi so správou, ktorá obsahuje jeho vlastné SYN bit (na synchronizáciu) a zároveň ACK bit na potvrdenie prijatia klientovho SYN. V tejto správe server posiela svoje vlastné poradové číslo a potvrdzuje poradové číslo klienta.

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