Komunikácia s využitím UDP protokolu

Andrej Krivošík

Fakulta informatiky a informačných technológií STU v Bratislave

Ilkovičova 2, 842 16 Karlova Ves

ID:120955

Obsah

Štruktúra hlavičky:

1.Source port (16 bit)

Číslo portu odosielateľa (zdrojový port). Identifikuje, ktorý port na odosielateľskom zariadení odoslal segment.

2.Destination port (16 bit)

Číslo portu prijímateľa (cieľový port). Určuje, na ktorý port na cieľovom zariadení je segment určený.

3. Fragment size(32 bit)

Číslo, ktoré definuje veľkosť fragmentov.

4.Sequence number (32 bitov)

Poradové číslo fragmentu. Slúži na správne zoradenie dátových fragmentov pri prenosoch, najmä keď dorazia v inom poradí alebo ak je nutné zopakovať ich odoslanie.

5.Flags (4 bit)

ACK (1 bit) – Acknowledgment, potvrdzuje správne prijatie správy, aktivitu alebo odpoveď.

NAK (1 bit) – Negatívne potvrdenie, použijeme ak prijímateľ zistí, že správa je poškodená

SYN (1 bit) – Synchronization, používa sa pri nadväzovaní spojenia.

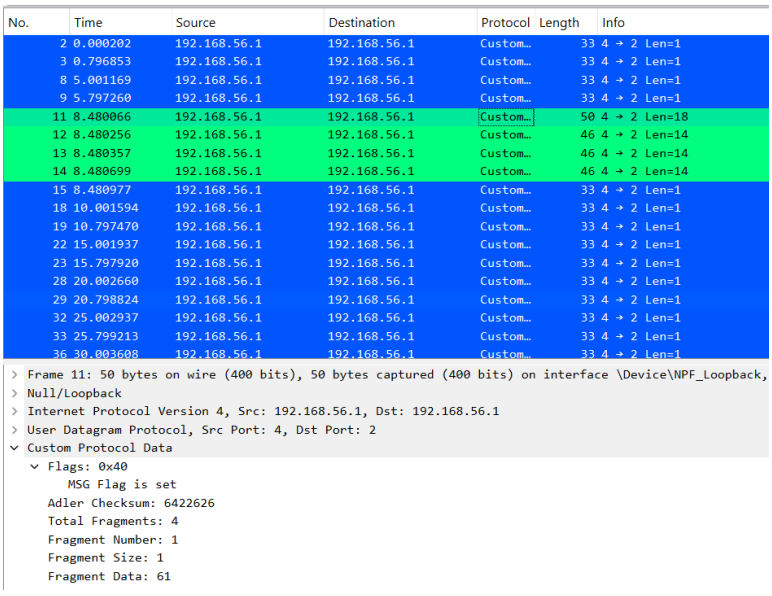
FIN (1 bit) – Finish, označuje ukončenie prenosu dát.

6.Checksum (32 bit)

Kontrolný súčet na kontrolu integrity dát. Vypočítava sa pre hlavičku aj dáta, aby sa zistili chyby počas prenosu. Použijem Adler-32.

7.Data (variabilná veľkosť)

Samotné dáta, ktoré sú odosielané v rámci segmentu. Môže byť prázdne, ak sa jedná len o kontrolné alebo riadiace správy, napríklad pri otváraní alebo zatváraní spojenia.



Zachytenie protokolu vo WS: bola poslaná správa o veľkosti 4 bajtov, rozdelená na 4 fragmenty po 1 bajte.

Kontrola integrity prenesenej správy:

Použijeme algoritmus Adler-32, ktorý sa správa nasledovne:

1.) Inicializujeme hodnoty

Adler-32 používa dve hodnoty, **A** a **B**.

Inicializujeme ich takto: **A** = 1, **B** = 0.

2.) Pre každý znak správy vypočítame hodnoty A a B.

A = (A + ASCII(hodnota znaku)) mod 65521

B = (B + A) mod 65521

Použijeme hodnotu 65521 lebo je to najväčšie prvočíslo, ktoré je menšie ako 2^16. Použitie prvočíselného modulu pomáha pri distribúcii hodnôt kontrolného súčtu, čím znižuje pravdepodobnosť kolízií a minimalizuje pravdepodobnosť pretečenia pri aritmetických operáciách, čo môže viesť k nežiaducim výsledkom.

3.) Vypočítame hodnotu pre každý znak správy

Berieme ASCII hodnotu každého znaku a vložíme do výpočtových funkcií hodnôt **A** a **B**. Napríklad správa „OK“ by sa spracovávala nasledovne:

O = ASCII hodnota: 79

A = (1 + 79) mod 65521 = 80

B = (0 + 80) mod 65521 = 80

K = ASCII hodnota: 75

A = (80 + 75) mod 65521 = 155

B = (80 + 155) mod 65521 = 235

4.) Vypočítame kontrolný súčet

Po spracovaní znakov máme: **A** = 155, **B** = 235.

Kontrolný súčet Adler-32 vypočítame následovne: Adler-32 = (B << 16) | A.

V našom prípade:

$B \ll 16 = 235 \times 65536 = 15435080$

Adler-32=15435080|155=15401115

5.) Prevedieme hodnotu na hexadecimálny tvar

15401115(dec) -> EB009B(hex)

Aby sme mali checksum o veľkosti 32 bitov, doplníme na začiatok nuly:

EB009B -> 00EB009B.

Zabezpečenie spoľahlivého prenosu dát:

Použijeme protokol Stop-And-Wait ARQ (Odosielateľ odosiela jednu správu a čaká na potvrdenie od prijímateľa pred odoslaním ďalšej správy).

Detekcia chýb

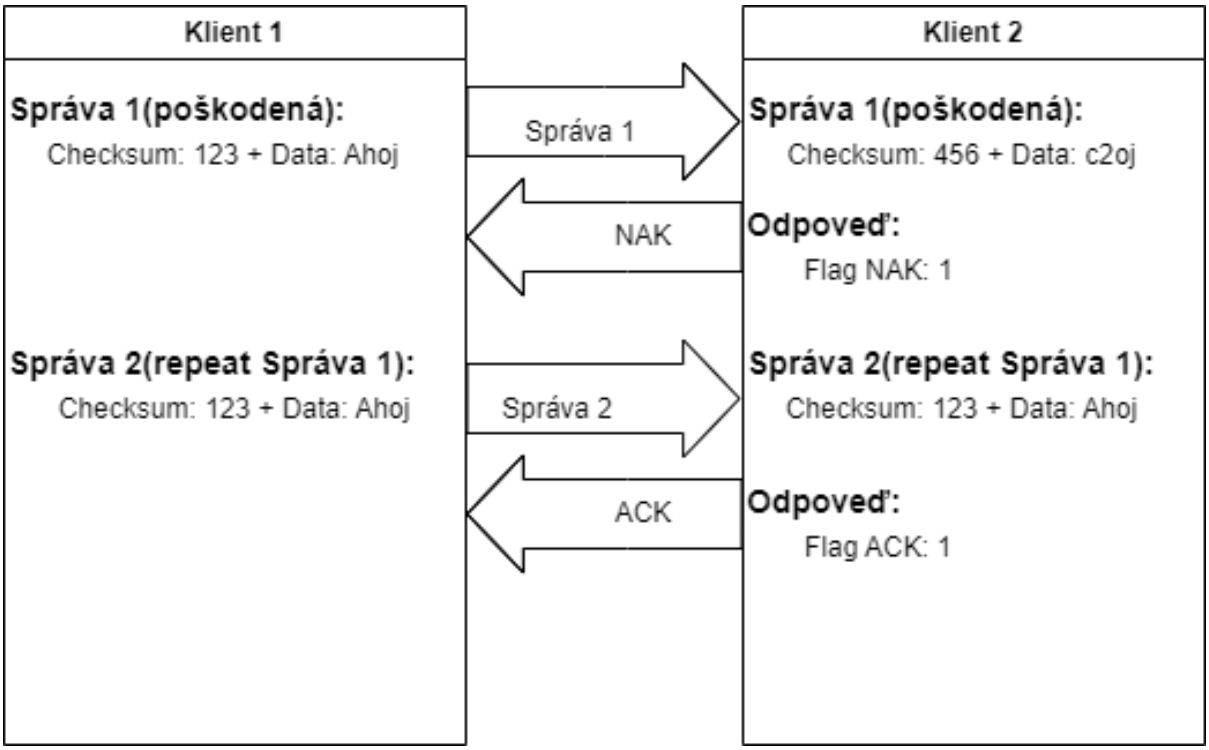
ARQ využíva kontrolné súčty (checksum). Ak prijímateľ zistí, že správa je poškodená, pošle späť negatívne potvrdenie (NAK).

Potvrdenie prijatia

Po prijatí správy prijímateľ posiela potvrdenie (ACK), že správu úspešne prijal. Ak odosielateľ nedostane potvrdenie v určitom časovom limite, predpokladá, že správa bola stratená alebo poškodená, a opätovne ju odosiela.

Opätovné odoslanie

V prípade negatívneho potvrdenia alebo nedostania potvrdenia od prijímateľa, ARQ zabezpečuje opätovné odoslanie správy.



Umožnenie úmyselného vytvorenia chyby:

Chyba by sa vytvárala po vypočítaní checksum a potom zmeníme dáta správy. Keď správa dorazí druhému klientovi, bude prepočítaný checksum, kde zistíme že správa nesedí stým čo sme čakali, klient potom odošle požiadavku o opätovné poslanie správy.

Nadväzovanie spojenia:

Krok 1: Klient1 pošle SYN klientovi2

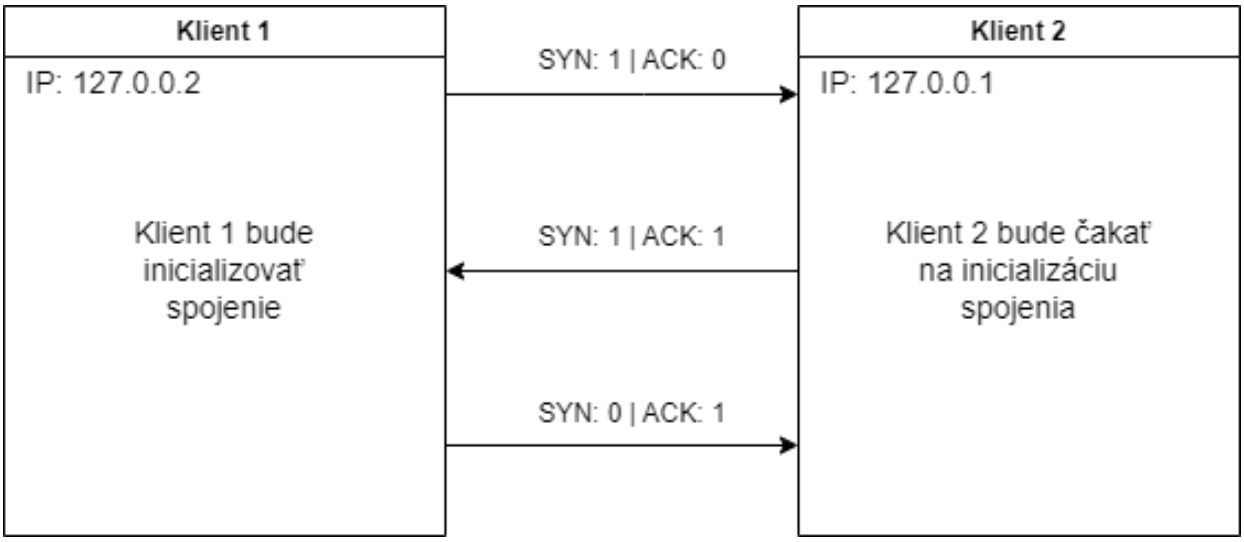
- Na začiatku sa rozhodne kto bude inicializovať spojenie podľa IP adries oboch klientov. Klient, ktorý bude nadväzovať spojenie, pošle druhému klientovi správu s nastaveným flagom **SYN**. Tento flag signalizuje požiadavku na nadviazanie spojenia.

Krok 2: Klient 2 odpovedá SYN + ACK

- Po prijatí segmentu s flagom SYN klient odpovie správou s dvomi flagmi: **SYN** a **ACK**. SYN flag indikuje, že súhlasí s nadviazaním spojenia, a ACK flag potvrdzuje prijatie klientovho SYN.

Krok 3: Klient1 odpovedá ACK

- Klient potvrdí prijatie správy poslaním segmentu s flagom **ACK**. Tento segment potvrdzuje prijatie SYN. V tomto bode je spojenie úspešne nadviazané a obe strany môžu začať komunikovať.



Udržanie spojenia:

Vysielanie Keep-Alive paketov:

Po určitom čase nečinnosti začne odosielateľ posielať Keep-Alive paket. Tento paket je malý segment, ktorý neobsahuje žiadne dáta, ale má v sebe aktuálne poradové číslo (sequence number) a potvrdzujúce číslo (acknowledgement number).

Keep-Alive paket v podstate testuje, či je spojenie stále funkčné, aj keď sa momentálne neprenášajú žiadne dáta.

Odozva na Keep-Alive paket:

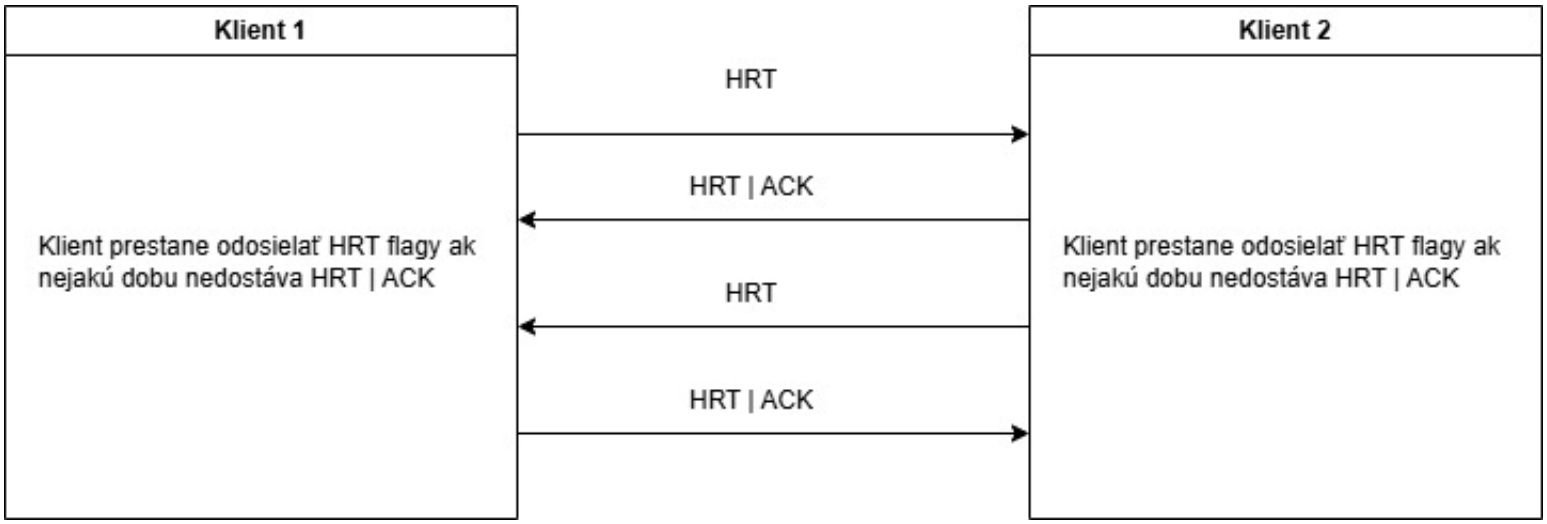
Ak je spojenie aktívne, prijímateľ odpovie ACK paketom, čím potvrdí, že spojenie stále existuje.

Ak je spojenie prerušené alebo je prijímateľ neaktívny, odosielateľ nedostane odpoveď.

Opakované pokusy:

Ak odosielateľ nedostane odpoveď na Keep-Alive paket, po určitej dobe pošle ďalší Keep-Alive paket.

Ak po niekoľkých neúspešných pokusoch stále nedostane žiadnu odpoveď, spojenie sa považuje za ukončené.



Zmeny vykonané počas implementácie oproti návrhu riešenia:

1. Štruktúra hlavičky:

1. Flags (7 bit)

ACK (1 bit) – Acknowledgment, potvrdzuje správne prijatie správy, aktivitu alebo odpoveď.

NAK (1 bit) – Negatívne potvrdenie, použijeme ak prijímateľ zistí, že správa je poškodená

SYN (1 bit) – Synchronization, používa sa pri nadväzovaní spojenia.

FIN (1 bit) – Finish, označuje ukončenie prenosu dát.

HRT (1 bit) – Flag, ktorý pracuje len v Keep-Alive, aby sme vedeli že sa jedná o Heartbeat správy

MSG (1 bit) – Označuje že sa jedná o textovú správu

FIL (1 bit) – Označuje že sa jedná o súborovú správu

2. Checksum (32 bit)

Kontrolný súčet na kontrolu integrity dát. Vypočítava sa pre hlavičku aj dáta, aby sa zistili chyby počas prenosu. Použijem Adler-32.

3. Number of fragments (32 bit)

Označuje celkový počet fragmentov, ktoré sa pošlú a koľko má druhý klient očakávať.

4. Fragment number (32 bit)

Poradové číslo fragmentu. Slúži na správne zoradenie dátových fragmentov pri prenosoch, najmä keď dorazia v inom poradí alebo ak je nutné zopakovať ich odoslanie.

5. Fragment size (32 bit)

Veľkosť fragmentu v bajtoch.

6.Data (variabilná veľkosť)

Samotné dáta, ktoré sú odosielané v rámci segmentu. Môže byť prázdne, ak sa jedná len o kontrolné alebo riadiace správy, napríklad pri otváraní alebo zatváraní spojenia.

2. Zabezpečenie spoľahlivého prenosu dát:

Použil som Selective Repeat (SR) bez okna:

1. Odosielanie všetkých fragmentov naraz:

- Odosielateľ naraz odošle všetky fragmenty správy bez čakania na žiadne priebežné potvrdenia.

2. Prijímanie a kontrola fragmentov:

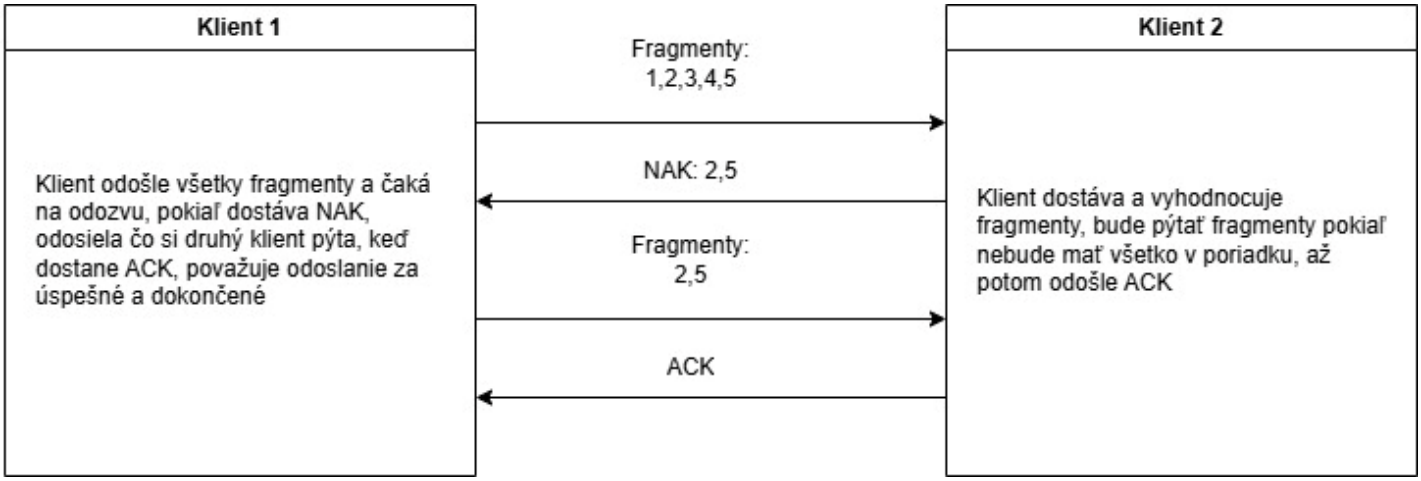
- Prijímač overuje každý fragment pri prijatí (napr. pomocou kontrolného súčtu).
- Ak **všetky fragmenty** prídu správne, prijímač odošle jedno **ACK**, ktoré signalizuje, že celá správa bola prijatá.

3. Spracovanie chýb alebo strát:

- Ak je ktorýkoľvek fragment poškodený alebo chýba, prijímač odošle jedno **NAK** s informáciou o konkrétnych chýbajúcich alebo chybných fragmentoch.
- Tento cyklus pokračuje, až kým všetky fragmenty nebudú prijaté správne.

4. Retransmisia iba chybných fragmentov:

- Po prijatí NAK odosielateľ znovu odošle iba tie fragmenty, ktoré boli označené ako chybné alebo chýbajúce.
- Keď prijímač skompletizuje správu, odošle konečné **ACK**.



Zhodnotenie a použité zdroje:

Moje spracovanie zadania skončilo celkom inak ako som plánoval pri návrhu. Najväčšiu zmenou bola štruktúra hlavičky a spôsob zabezpečenie spoľahlivého prenosu dát som dokázal implementovať komplexnejší než som si pri návrhu myslel. Myslím že štruktúra hlavičky je celkom efektívna, keďže dokážeme flagy vmestiť do jedného bajtu, každá má svoje využitie a žiadna nie je zbytočná. Takisto sa v hlavičke posielajú len informácie o správe/fragmente ktoré sú potrebné. Celkovo si myslím, že som zadanie spracoval efektívne.

Zdroje:

https://www.geeksforgeeks.org/types-of-network-protocols-and-their-uses/

https://www.javatpoint.com/tcp

https://developer.arm.com/documentation/101964/0300/Chromium-optimization--Adler-32