Komunikácia s využitím UDP protokolu - Dokumentácia

Fakulta Informatiky a Informčaých technológií Slovenskej Technickej Univerzity

Semestrálny projekt z PKS

Ondrej Krajčovič

xkrajcovico@stuba.sk

17.Októbra 2024

Zadanie

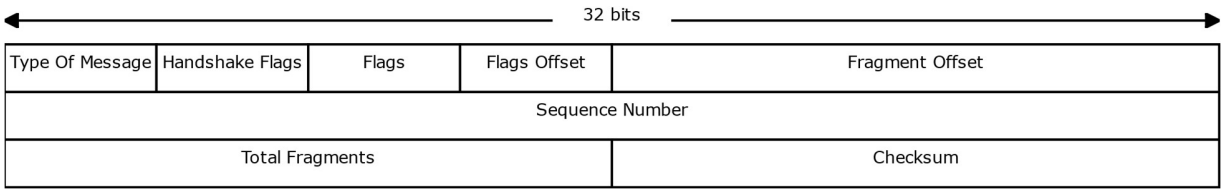
[1]Cieľom zadania je navrhnúť a implementovať P2P aplikáciu využívajúcu vlastný protokol postavený nad UDP (User Datagram Protocol) v transportnej vrstve sieťového modelu TCP/IP. Aplikácia bude umožňovať komunikáciu dvoch účastníkov v lokálnej Ethernet sieti, vrátane výmeny textu a prenosu ľubovoľných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

Výstupom tejto časti je dokumentácia, v ktorej predstavíte návrh svojho protokolu a plán implementácie jednotlivých mechanizmov. Následne túto dokumentáciu prediskutujete so svojím cvičiacim počas kontrolného bodu, kde získate pripomienky na zapracovanie do praktickej časti. Dokumentácia musí obsahovať nasledujúce časti:

- štruktúru hlavičiek vášho protokolu,
- opis metódy na kontrolu integrity prenesenej správy (napr. ak si zvolíte CRC16, tak ho opíšete ako sa vypočíta, rovnako postupovať pri iných metódach),
- opis metódy na zabezpečenie spoľahlivého prenosu dát (ARQ),
- opis metódy na udržanie spojenia (Keep-Alive),
- diagramy opisujúce predpokladané správanie vášho uzla/uzlov, použite UML diagramy ako napr. sekvenčný, aktivity a stavový (používajte vhodné nástroje, ako napr. miro alebo draw.io, nie skicár).

[1] - github

Hlavička



Obr1.: schéma vlastnej hlavičky

*Hlavička protokolu má dĺžku **12 bajtov**, resp 96 bitov. Hlavička sa skladá z nasledovných údajov:*

4b: Type Of Message(TOM): Typ posielanej správy:

- 0000(0): reprezentuje textovú správu
- 0001(1): reprezentuje súbor

4b: Handshake Flags: Obsahuje aktuál

- SYN = 0001
- ACK = 0010
- ACK-SYN = SYN|ACK = 0011
- FIN = 0100

4b: Flags(F):

- Skladá sa zo štyroch bitov:
  - Alive: určuje či je správa prázdna (podáva informáciu o aktuálnosti spojenia), alebo obsahuje dáta
  - ACK: Acknowledgement: určuje či bola správa úspešne prijatá
  - Fragmented: určuje či je správa fragmentovaná
  - More fragments: určuje či je aktuálny packet posledným fragmentom

4b: Flags Offset(FFO): 0000: reserved

- Má rezervovanú hodnotu: 0000(0): neskôr ho bude možnosť použiť pre rozšírenie dĺžky F

2B: Fragment Offset(FO):

- Identifikačné číslo aktuálneho fragmentu, ak je správa nefragmentovaná, je rezervovaný na 0

4B: Sequence number(SQN):

- Postupným posielaním jednotlivých správ sa inkrementuje

2B: Total Fragments(TF):

- Celkový počet Fragmentov

2B: Checksum(CKS):

- Dva bajty obsahujúce „checksum“ headeru a dát kalkulovaných pomocou CRC16, určený na verifikáciu obsahu packetu. Tento segment bude v budúcnosti pravdepodobne menený, kvôli rôznosti dĺžky checksum-u a jeho oddelovaniu od hlavičky.

Nadviazanie spojenia - Handshake

Na nadviazanie spojenia plánujem použiť TCP inšpirovaný three way hadnshake.

1. Proces začína tým, že Point#1 vyšle SYN segment so žiadosťou o spojenie zároveň s inicializačným sequence number(SQN).
2. Point#2 odpovie segmentom SYN-ACK na potvrdenie prijatia
3. Point#1 následne potvrdí spojenie zaslaním ACK segmentu.

Inicializácia bude prebiehať dvakrát, a to z dôvodu že každá osoba bude mať vlastný sending port a listening port.

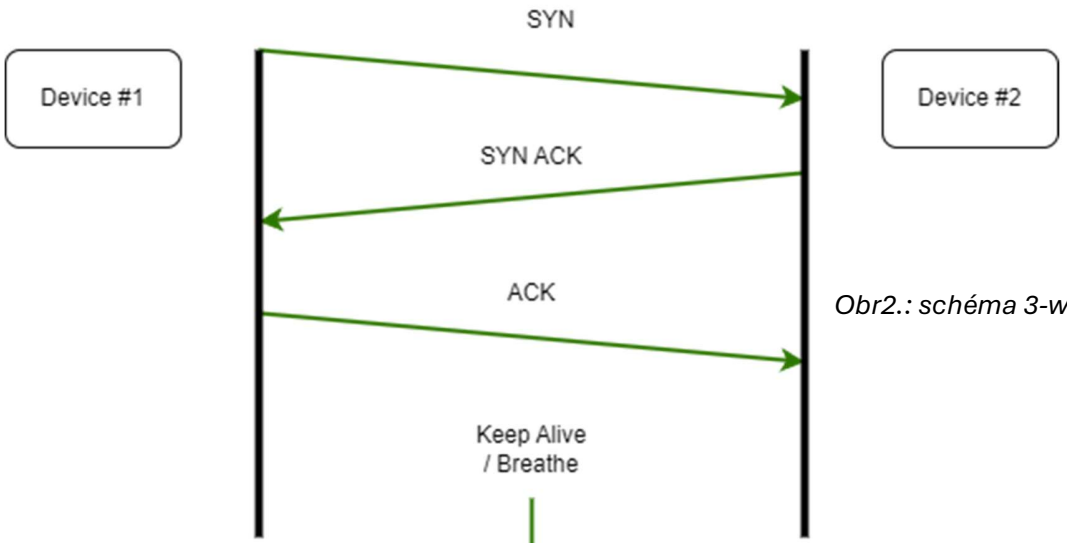
Binárne hodnoty syn, ack, syn ack, a fin

SYN = 0b0001

ACK = 0b0010

FIN = 0b0100 *využíva sa v pri zatváraní spojenia

SYN_ACK = SYN | ACK = 0b011



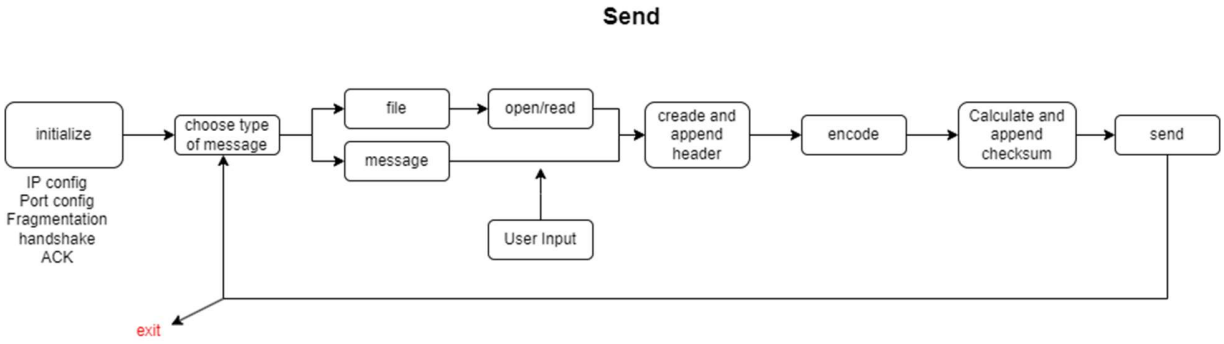
Obr2.: schéma 3-way handshake-u

Obr3.: schéma komunikácie medzi dvoma bodmi

Udržiavanie spojenia a posielanie packetov

Na udržanie spojenia, inicializovaného handshake-om bude každým kanálom v pravidelných intervaloch poslianý packet bez dát, s keep alive flag, v prípade že iný časový interval jedno zariadenie neobdrží daný packet, bude komunikácia následne zrušená.

Pre posielanie packetov bude každým odoslaním otvorený vopred dohodnutý port odosielatľom, následne bude správa odoslaná a po prijatí spätného Acknowledgement packetu bude komunikácia zo strany odosielateľa dočastne zastavená



Obr3.: schéma procesu odosielania správy

Fragmentácia

Pre zníženie dĺžky jednotivých odoslaných packetov budú správy posielané v častiach (ďalej fragmentoch). Správy teda budú postupne odosielané zo send portu odosielateľa v packetoch o vopred určenej veľkosti. Postup je teda nasledovný. Správa sa postupne načíta do packetu, pridá sa header a checksum. Následne sa packet odošle. Po prijatí packetu sa na strane prijateľa správa rozdelí porovná sa checksum a prípadne sa pripojí ku zvyšku doteraz prijatej správy.

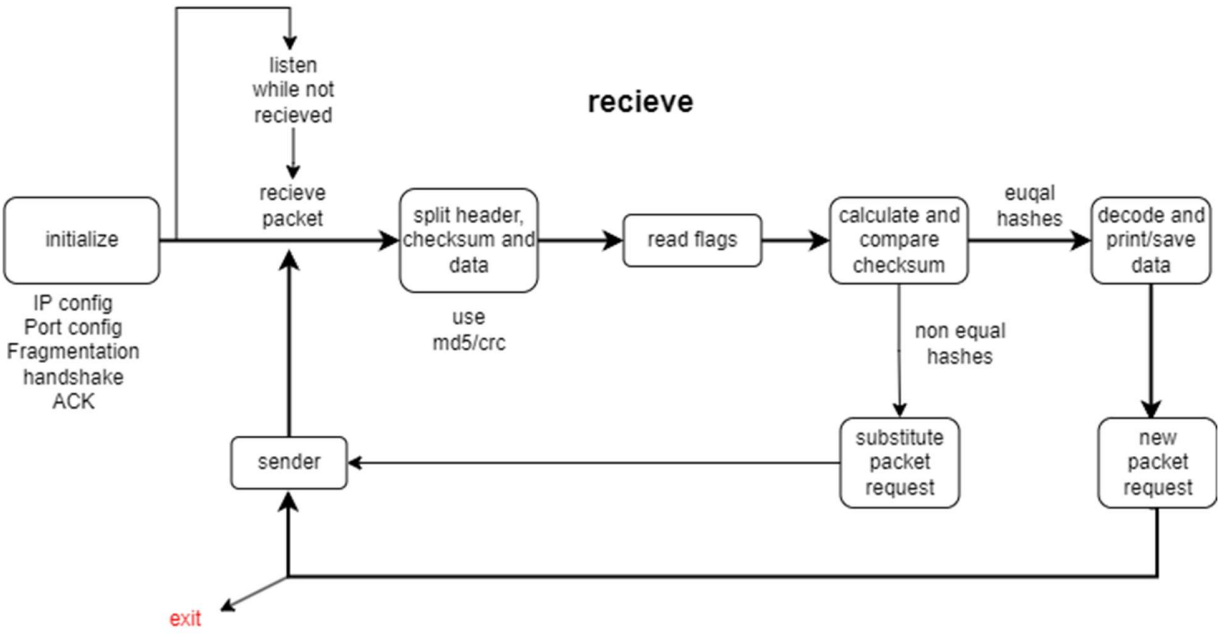
Overovanie korektnosti a prijímanie packetov

pred odoslaním správy sa z hlavičky a dát vytvorí checksum pomocou CRC16, ktorý bude následne vložený medzi hlavičku a dáta. Po prijatí packet bude správa rozdelené naspäť na hlavičku checksum a dáta. Následne sa opäť vyráta checksum z prijatých údajov a porovná sa s prijatým checksumom. Ak sa rovnajú správa bola neporušená a prijmateľ odošle Acknowledgement o prijatí správy resp, žiadosť o ďalší packet. Ak sa však prijatý checksum a vykalkulovaný checksum nerovnajú prijatý packet sa zadodí, a žiada sa o daľší, nahrádný packet.

CRC16, ktorý bude implementovaný funguje postupným delením(binárny XOR) binárneho vstupu(ku ktorému sme pripojili 16-nulových bitov) vopred určeným polynómiálom. zvyškom delenia je checksum

ARQ - zabezpečenie spoľahlivého prenosu dát

Vo finálnej verzií bude implementovaný selective repeat ARQ. Sliding window, mechanizmus ktorý po odoslaní k+n frames prijíma k-ty spätný acknowledgement od reciever-a.v prípade že sa m-tý ($m \in \langle k-k+n \rangle$) frame nedostane k odosielateľovi, všetky ďalšie frame-y sa rušia a odosiela sa opäť m-tý frame.



Obr4.: schéma procesu prijímania správ

Keep Alive/ Haertbeat

Program bude fo finálnej podobe počas trvania spojenia posielať header správy bez dát, ktorý bude obsahovať keep alive/haertbeat flag. Tento packet sa bude vysielať v pravidelnom intervale, v prípade že ho príjmač nezachytí počas istého časového intervalu (predpokladane dvojnásobok intervalu posielania) spojenie sa uzavrie a program sa ukončí.

Chyba

Nakoľko sa pri bežnom prenose dát chybovosť limitne blíži nulovej, pri testovaní budeme musieť chybu simulovať. Bude tak mechanickým pozmenením checksum-u.

Záver

V prvej implementácií som realizoval nasledovné: Štruktúru hlavičky, inicializáciu spojenia pomocou three-way handshake, štruktúru spojenia cez listening port a sending port. Vo finálnej verzií je dôležité dostatočne ošetriť CRC 16 checksum, keep alive spojenia a fragmentáciu dát. Po daných úpravách by mala byť implementácia daného protokolu porovnateľne spoľahlivá ako TCP. Informácie uvedené v tejto dokumentácií nie sú definitívne a je veľmi pravdepodobné, že sa budú meniť.

Aktuálne použité knižnice:

- *socket:* pre potreby sieťového spojenia p2p komunikácie
- *threading:* pre vytváranie stále-aktívnych vlákien na listening/sending port
- *struct:* na kódovanie správ na byty

Obr.5: všeobecná schéma riešenia

Obsah

Zdroje:

[1.] - zadanie

[2.] - CRC16

[3.] - prednášky

[4.] - design protokolu

[5.] – selective repeat ARQ

Cvičenia: Adrián Ondov; utorok 16:00