Fedor Viest
ID: 103177

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológii
Ilkovičova 2, 842 16, Bratislava

PKS Zadanie 2 – UDP komunikátor

Fedor Viest

Termín odovzdania: 8.12.2021

Prednášajúci: prof. Ing. Ivan Kotuliak, PhD.

Cvičiaci: Ing. Lukáš Mastiľak

Fedor Viest
ID: 103177

Obsah

Zadanie 3
Návrh hlavičky 4
Typ správy (1B) 4
Číslo packetu (3B) 4
Checksum (4B) 5
ARQ metóda 6
Keep Alive metóda 6
Dôležité knižnice a funckie 6
Flowchart 7
Zmeny oproti návrhu 10
Fungovanie programu 11
Začatie komunikácie 11
Klient 11
Funkcia client() 11
Send_message() 12
Ako používať klient stranu 13
Server: 14
Receive_data() 14
Ako používať server stranu 15
Použité knižnice 16
Flowchart 16
Wireshark 19

Fedor Viest
ID: 103177

Zadanie

Navrhnite a implementujte program s použitím vlastného protokolu nad protokolom UDP (User Datagram Protocol) transportnej vrstvy sieťového modelu TCP/IP. Program umožní komunikáciu dvoch účastníkov v lokálnej sieti Ethernet, teda prenos textových správ a ľubovoľného súboru medzi počítačmi (uzlami).

Program bude pozostávať z dvoch častí – vysielacej a prijímacej. Vysielací uzol pošle súbor inému uzlu v sieti. Predpokladá sa, že v sieti dochádza k stratám dát. Ak je posielaný súbor väčší, ako používateľom definovaná max. veľkosť fragmentu, vysielajúca strana rozloží súbor na menšie časti - fragmenty, ktoré pošle samostatne. Maximálnu veľkosť fragmentu musí mať používateľ možnosť nastaviť takú, aby neboli znova fragmentované na linkovej vrstve.

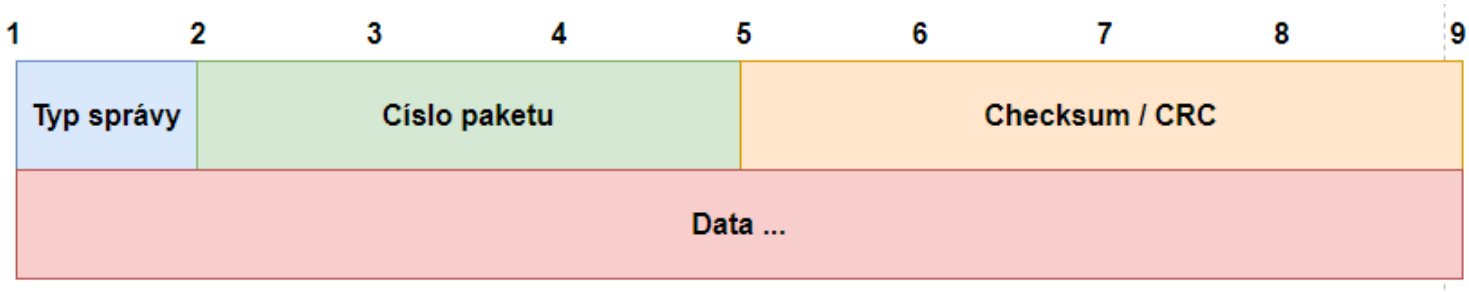
Ak je súbor poslaný ako postupnosť fragmentov, cieľový uzol vypíše správu o prijatí fragmentu s jeho poradím a či bol prenesený bez chýb. Po prijatí celého súboru na cieľovom uzle tento zobrazí správu o jeho prijatí a absolútnu cestu, kam bol prijatý súbor uložený.

Program musí obsahovať kontrolu chýb pri komunikácii a znovuvyžiadanie chybných fragmentov, vrátane pozitívneho aj negatívneho potvrdenia. Po prenesení prvého súboru pri nečinnosti komunikátor automaticky odošle paket pre udržanie spojenia každých 5-20s pokiaľ používateľ neukončí spojenie. Odporúčame riešiť cez vlastne definované signalizačné správy.

Fedor Viest
ID: 103177

Program bude implementovaný v jazyku Python, kvôli ľahkej prístupnosti k pamäti a množstvu preddefinovaných funkcii, ktoré mi v konečnom dôsledku ušetria čas.

Návrh hlavičky



Typ správy (1B)

V tejto časti hlavičky sa nachádza informácia o aký typ správy ide. Správy sú reprezentované ako čísla v decimálnom formáte

- **0 – INIT -** Začiatok komunikácie
- **1 – MSG -** Inicializácia posielania správy, posiela klient serveru
- **2 – FILE -** Inicializácia posielania súboru, posiela klient serveru
- **3 – ACK** – potvrdenie o doručení packetu, posiela server klientovi
- **4 – NACK** – potvrdenie o doručení packetu s chybou, posiela server klientovi
- **5 – KPA -** Keep alive
- **6 – SWAP –** Výmena rolí
- **7 – END -** Ukončenie komunikácie

Číslo packetu (3B)

V tomto poli sa nachádza poradové číslo packetu. Zvolil som veľkosť 3B, lebo je potrebné preniesť 2MB súbor

Fedor Viest
ID: 103177

Checksum (4B)

Na výpočet checksum som sa rozhodol použiť metódu **crc32** z knižnice **zlib.** Táto funkcia využíva 32 bitový polynóm vo formáte:

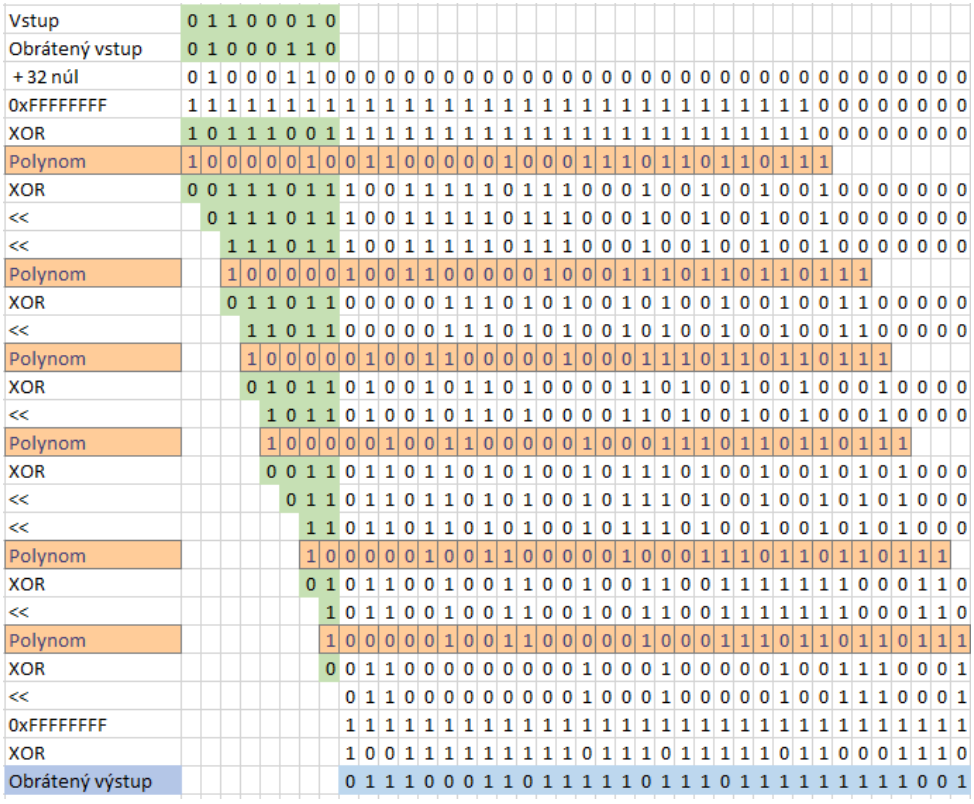
$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1$, čo v binárnom formáte vyzerá takto:

1 0000 0100 1100 0001 0001 1101 1011 0111

Tento polynóm sa využíva na XOR so vstupom. V metóde crc sa využíva binárny posun vľavo a XOR.

Algoritmus:

1. Obráť vstup
2. Pridaj nakoniec 32 núl
3. XOR s 0xFFFFFFFF
4. XOR s polynómom, ak je prvý bit 1
5. Bitshift vľavo a posun celého čísla vpravo
6. Opakuj 4-5 pokým prvých 8 bitov nie je 0
7. XOR s 0xFFFFFFFF
8. Obráť vstup



Fedor Viest
ID: 103177

Najväčšia veľkosť fragmentu môže byť 1463B, lebo:

1518 - **18***(eth2)* – **20***(ip)* – **8***(udp)* – **8**(*moja hlavička)* = 1464 bajtov

ARQ metóda

Ako ARQ metódu som si vybral **Stop-and-Wait (Bloková metóda).** Klient vždy čaká na ACK od servera aby poslal ďalší packet. Pokiaľ server pošle NACK, klient odošle packet znova, až pokým odpoveď servera nebude ACK, čiže packet dostal v poriadku.

Keep Alive metóda

V prípade nečinnosti na strane klienta, program prejde do fázy pre udržanie spojenia. Najprv klient pošle serveru správu **KPA**, následne v danom intervale (každých 5/10 sekúnd) server pošle **ACK** správu klientovi.

Dôležité knižnice a funckie

Nižšie je zoznam knižníc a niektorých funckii, ktoré určite použijem. Tento zoznam sa ešte určite rozrastie

Knižnice:

- Socket
- Zlib
- Threading

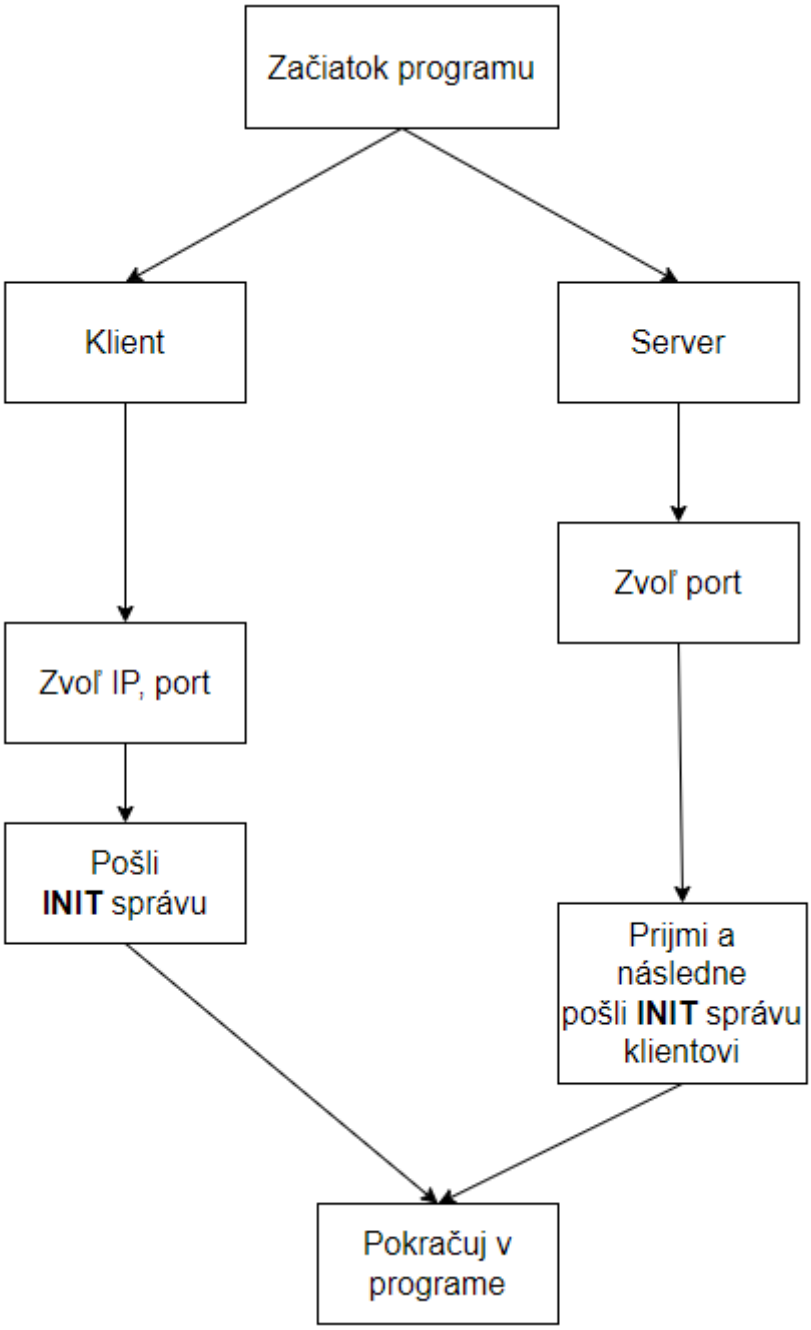
Funckie:

- sendto() – funckia z knižnice socket, pošle dáta
- recvfrom() – funckia z knižnice socket, prijme dáta
- crc32() – funckia z knižnice zlib, vypočíta checksum 32 bitov dlhý

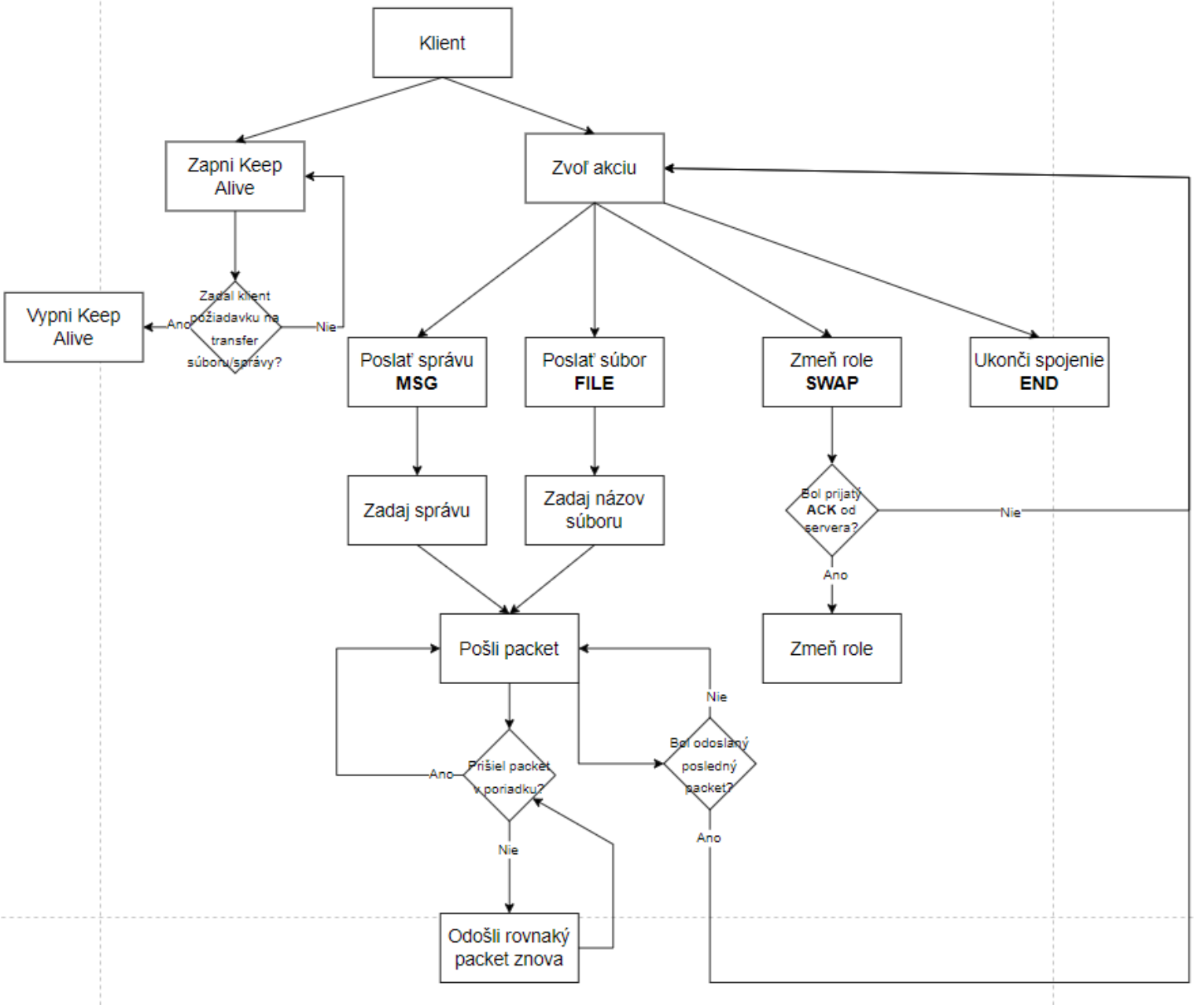
V programe použijem vlastné funkcie na poslanie správy/súboru a dalšie pomocné, v prípade potreby, keep alive budem riešiť v novom threade.

Fedor Viest
ID: 103177

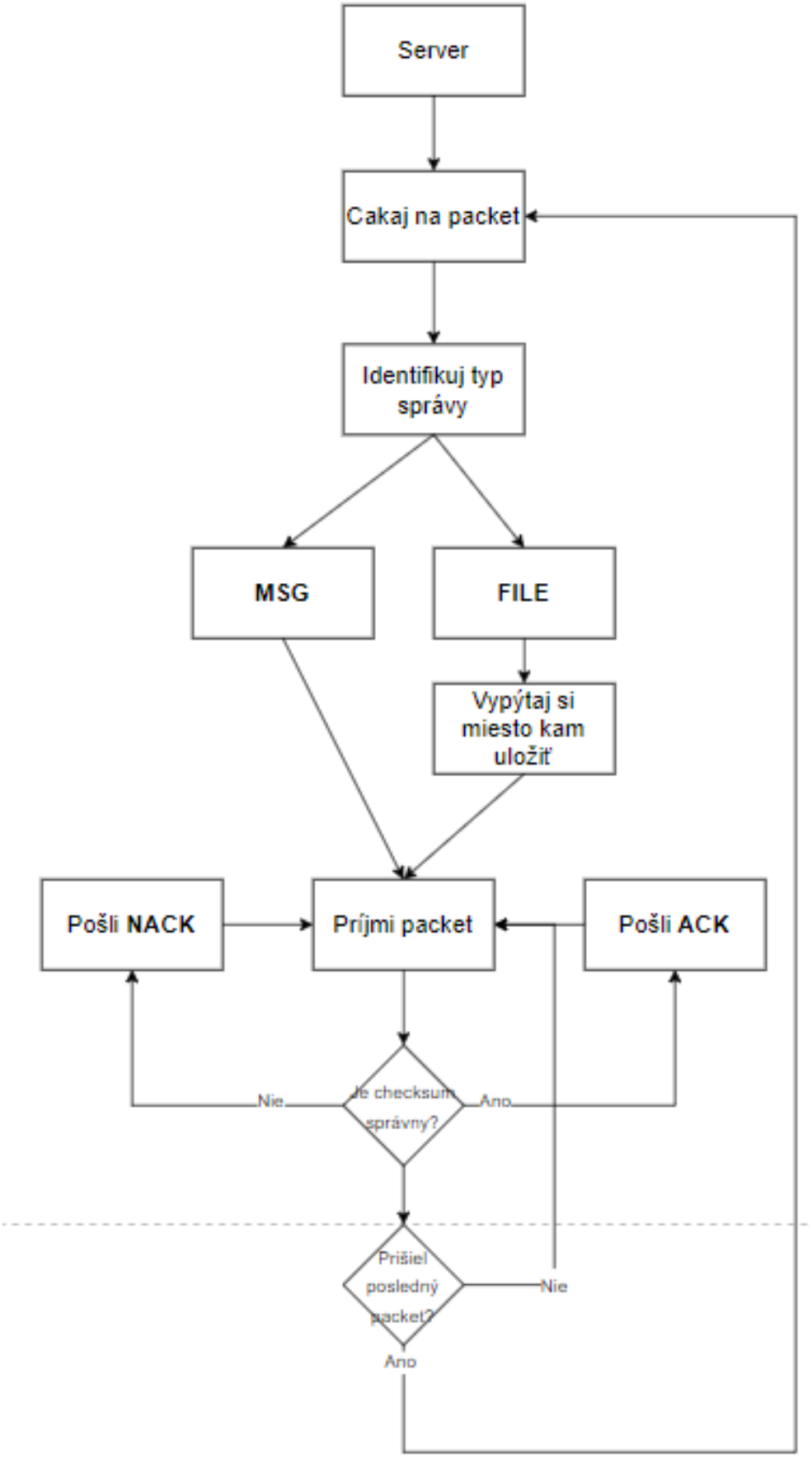
Flowchart



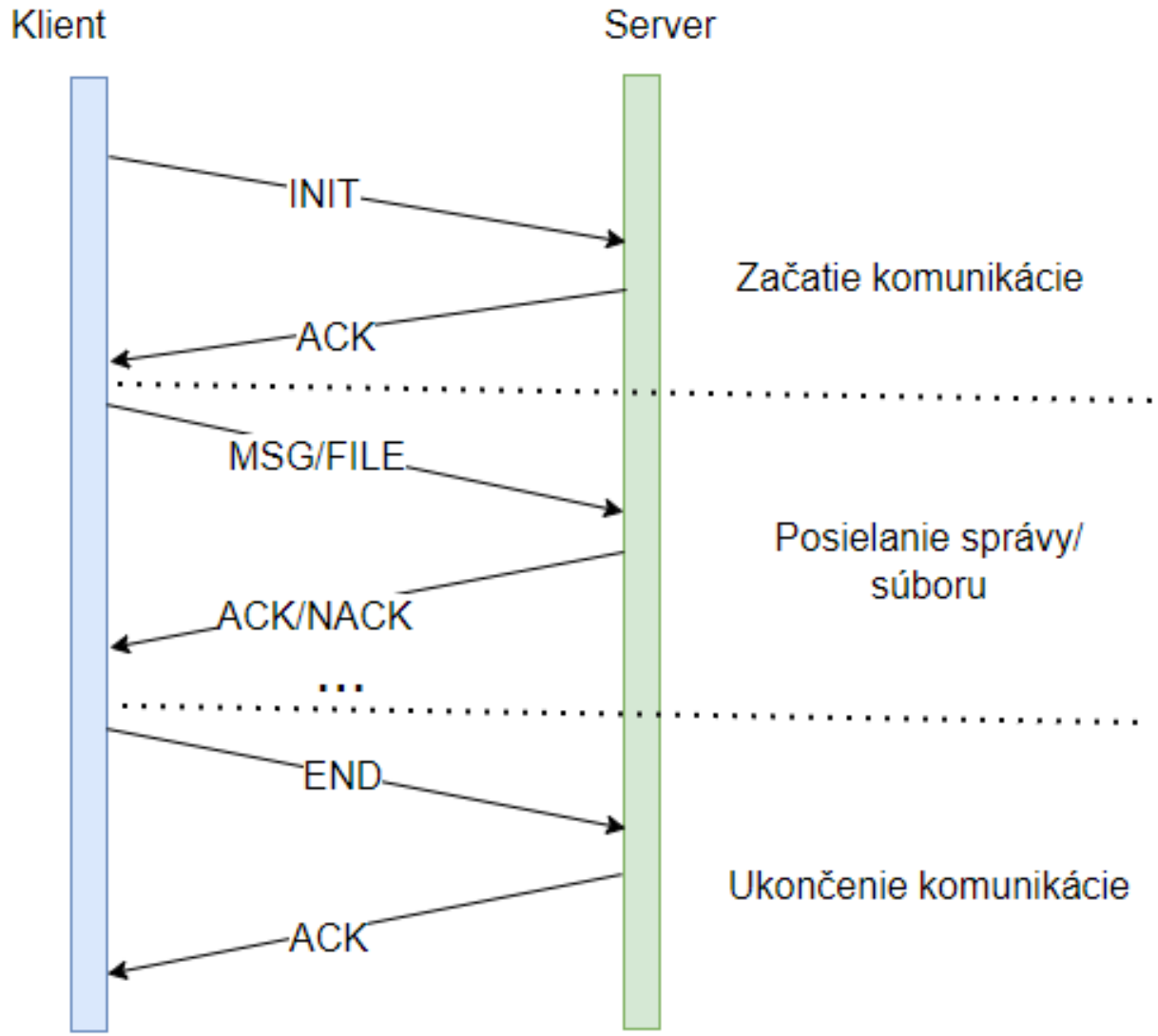
Fedor Viest
ID: 103177



Fedor Viest
ID: 103177



Fedor Viest
ID: 103177



Zmeny oproti návrhu

- Keep Alive správy sa posielajú každých 5 sekúnd. Pričom klient pošle KPA správu serveru a server odpovedá správou ACK.
- Pridanie nových použitých knižníc.
- Zmeny vo flowchartoch a pridanie ďalších flowchartov

Fedor Viest
ID: 103177

Fungovanie programu

Začatie komunikácie

Najprv si program od používateľa vypýta ip a port v prípade klienta a port v prípade servera. Komunikácia medzi serverom a klientom sa začína 2-way handshake, pričom klient pošle serveru inicializačnú správu a čaká pokým server pošle ACK.

Klient

Pre klient stranu program používa funkcie **client(), send_message(), swap(), keep_alive(), end_connection()**

Funkcia client()

V tejto funkcii si používateľ vyberá, čo chce spraviť. Má možnosti poslať správu, poslať súbor, vymeniť role, alebo ukončiť spojenie. Klient podľa tohto pošle serveru typ správy a čaká na ACK od servera.

Posielanie správy: Program si vypýta veľkosť fragmentu a či chce správu poslať s chybami a následne správu, ktorú chce používateľ poslať. Program si sám vypočíta počet fragmentov, ktoré treba poslať. Pošle typ správy a celkový počet fragmentov serveru v hlavičke a čaká na ACK. V prípade, že prišiel prechádza do funkcie **send_message().**

Posielanie súboru: V tomto prípade je jediný rozdiel oproti posielaniu správy, že si program vypýta názov súboru, ak je potrebné aj cestu k nemu a pri odosielaní packetu serveru do dátovej časti pridá názov súboru.

Výmena rolí: Klient pošle správu o výmene serveru a čaká na ACK. Ak príde prechádza do funkcie **swap().**

Keep Alive: Keep alive v programe riešim v threade, pričom nastavujem eventy. V prípade, že KeepAlive event je nastavený(is_set), program vie, že má posielať keep

alive správy a očakávať ACK, v prípade, že nie je nastavený(clear) neposiela keep alive správy. V prípade, že sa spojenie preruší, nastaví sa end_event a vtedy program vie, že má ukončiť thread.

Ukončenie spojenia: Klient pošle správu na ukončenie spojenia a čaká na ACK, v prípade, že ACK nepríde, ukončí spojenie s výpisom, že musel ukončiť spojenie aj tak. V opačnom prípade, ukončí spojenie korektne.

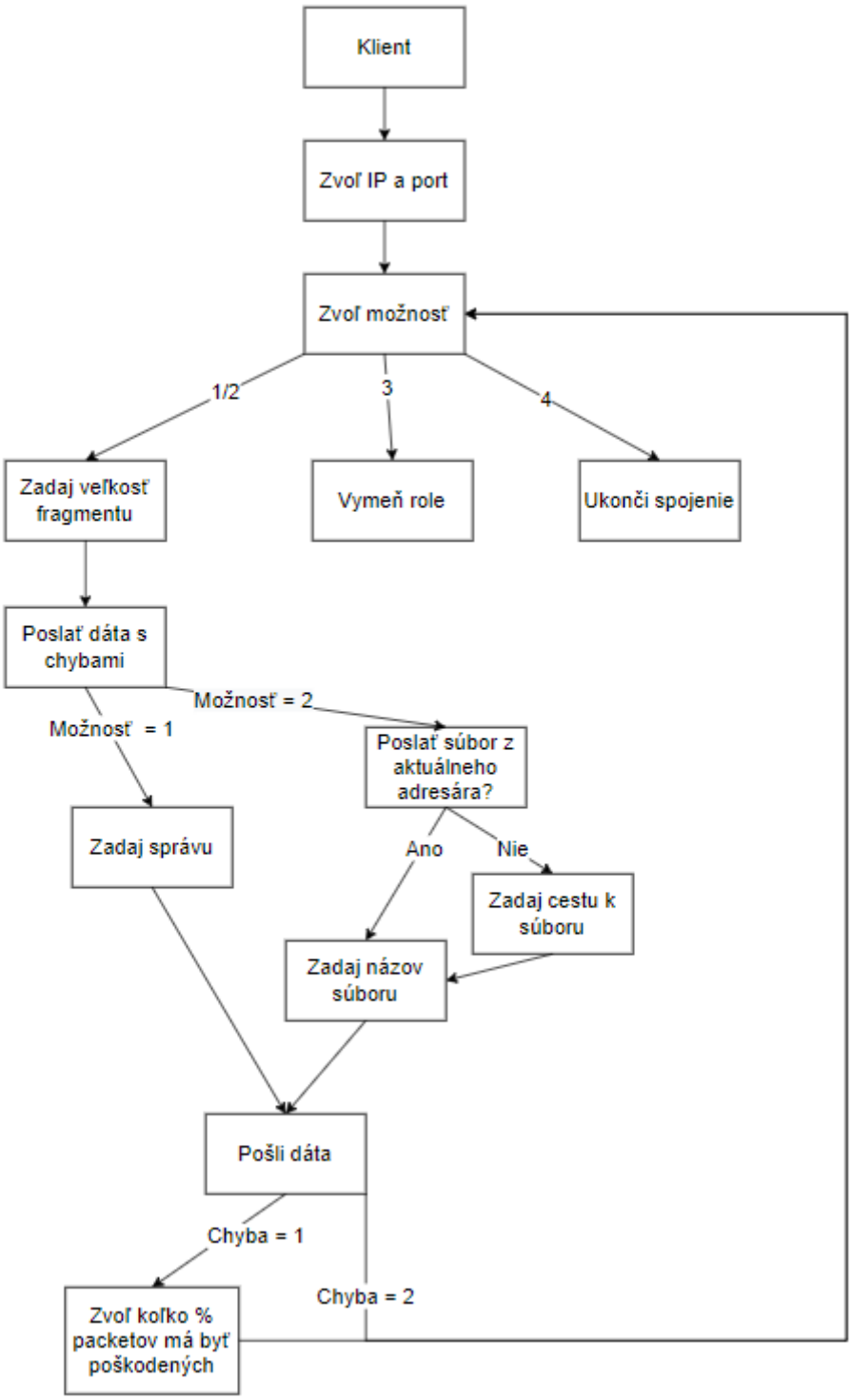
Send_message()

Táto funkcia slúži na posielanie dát serveru. Najprv vygeneruje čísla chybných packetov, pokiaľ je zvolená možnosť, že klient chce poslať dáta s chybami. Následne prejde do nekonečného cyklu, kde posiela dáta o zvolenej veľkosti, pokým počet odoslaných packetov sa nerovná počtu všetkých packetov, ktoré mal odoslať. V prípade, že od servera prišiel na dáta ACK, klient posiela ďalší packet, v opačnom prípade klient posiela znova rovnaký packet, pokým nepríde ACK.

Simulácia chyby: Simulácia chyby sa deje v dátovej časti, najprv sa vypočíta crc z korektných dát, následne sa dáta zmenia a pošlú sa. Na strane servera sa porovná prijaté crc a vypočítané crc z prijatých dát, ak sa zhodujú server pošle ACK, ak nie, pošle NACK.

Fedor Viest
ID: 103177

Ako používať klient stranu



Fedor Viest
ID: 103177

Server:

Server neustále počúva požiadavky klienta a na to reaguje, v prípade, že príde typ správy:

- 1 -> server prechádza do funkcie **receive_data()**
- 2 -> server si vypýta umiestnenie súboru a prejde do funkcie **receive_data()**
- 5 -> server odošle ACK na keep_alive správu
- 6 -> server prejde do funkcie **swap()**
- 7 -> server ukončí spojenie

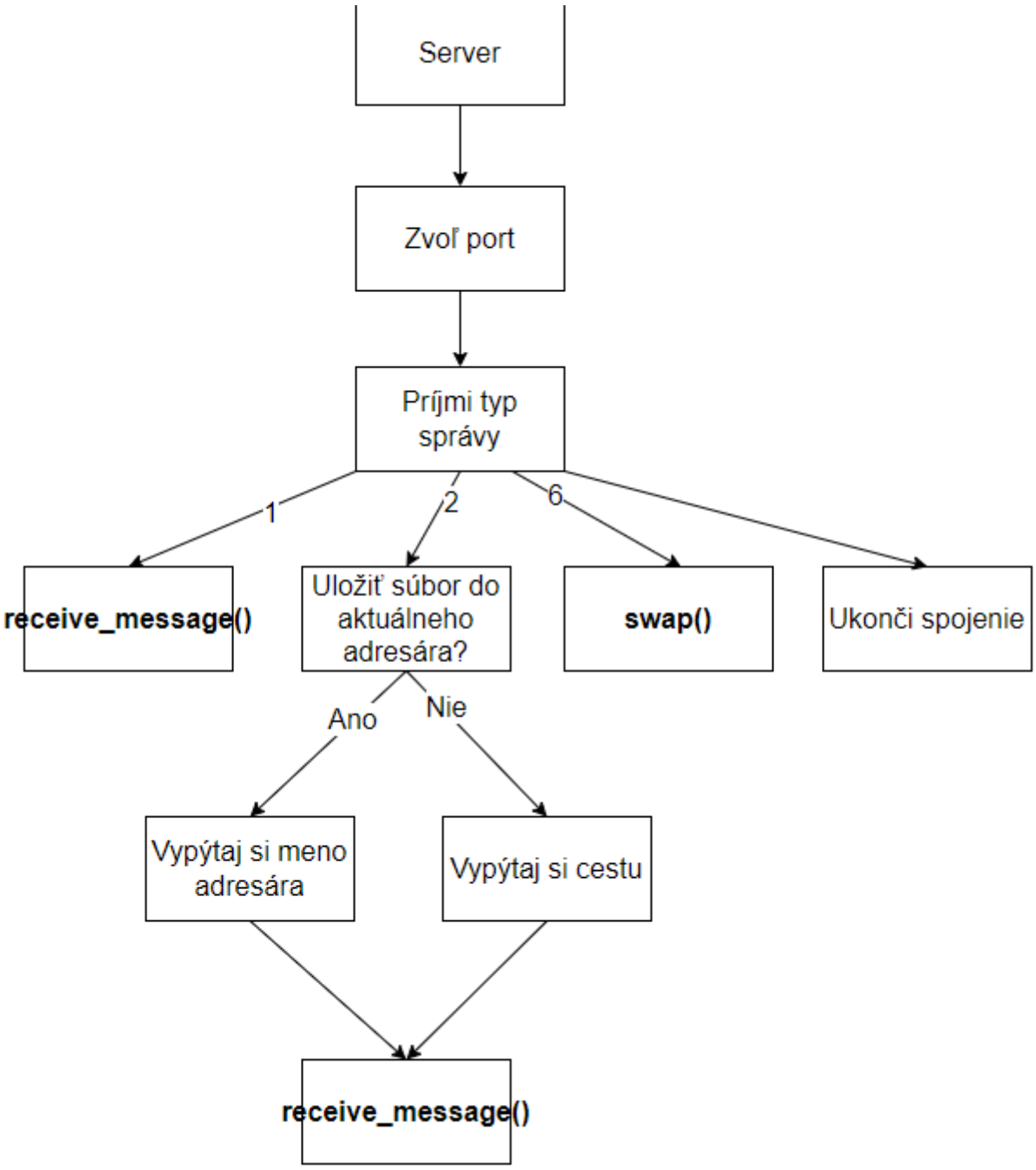
Server na každú správu odpovedá ACK, odpovedá NACK iba v prípade, že prišli chybné dáta.

Receive_data()

V tejto funkcii server prijíma dáta a postupne si ich zapisuje pokiaľ prišli v poriadku. Server porovnáva prijaté crc s vypočítaným crc z prijatých dát. V prípade, že sa crc nezhodujú, server posiela NACK a nezapisuje si nič. V opačnom prípade server pošle ACK a zapíše si prijaté dáta.

Fedor Viest
ID: 103177

Ako používať server stranu



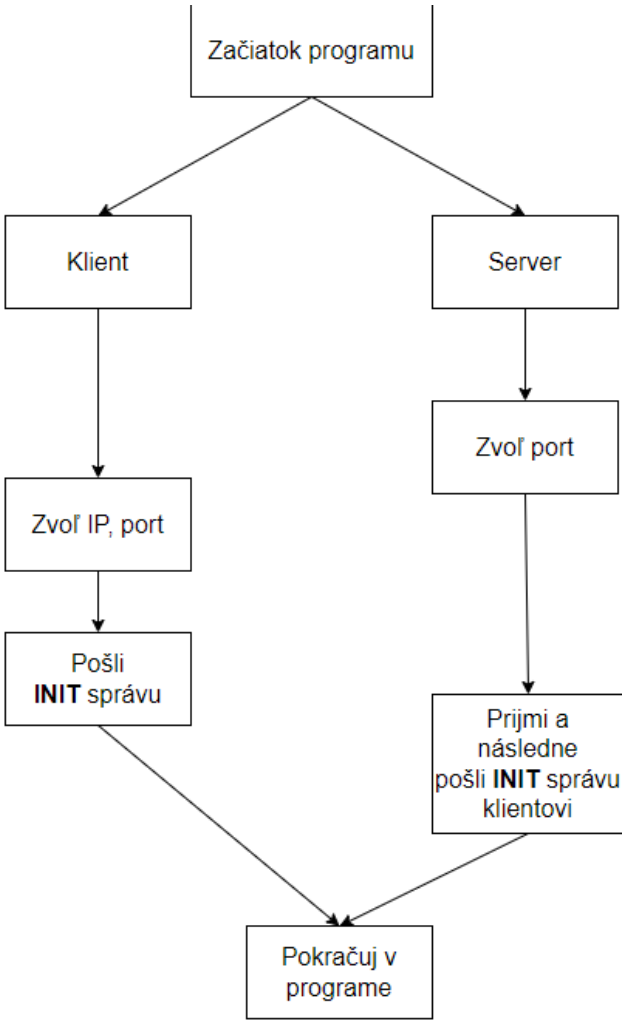
Fedor Viest
ID: 103177

Použité knižnice

- Socket
- Threading – na keep alive
- Zlib – výpočet crc
- Struct – na zakódovanie hlavičiek podľa požadovanej veľkosti a formátu
- Random – na vygenerovanie náhodných chybných packetov
- Math – na math.ceil()
- Os – na zadávanie absolútnych ciest
- Sys – na výpis počtu keep alive do jedného riadku
- Time – posielanie každých 5 sekúnd

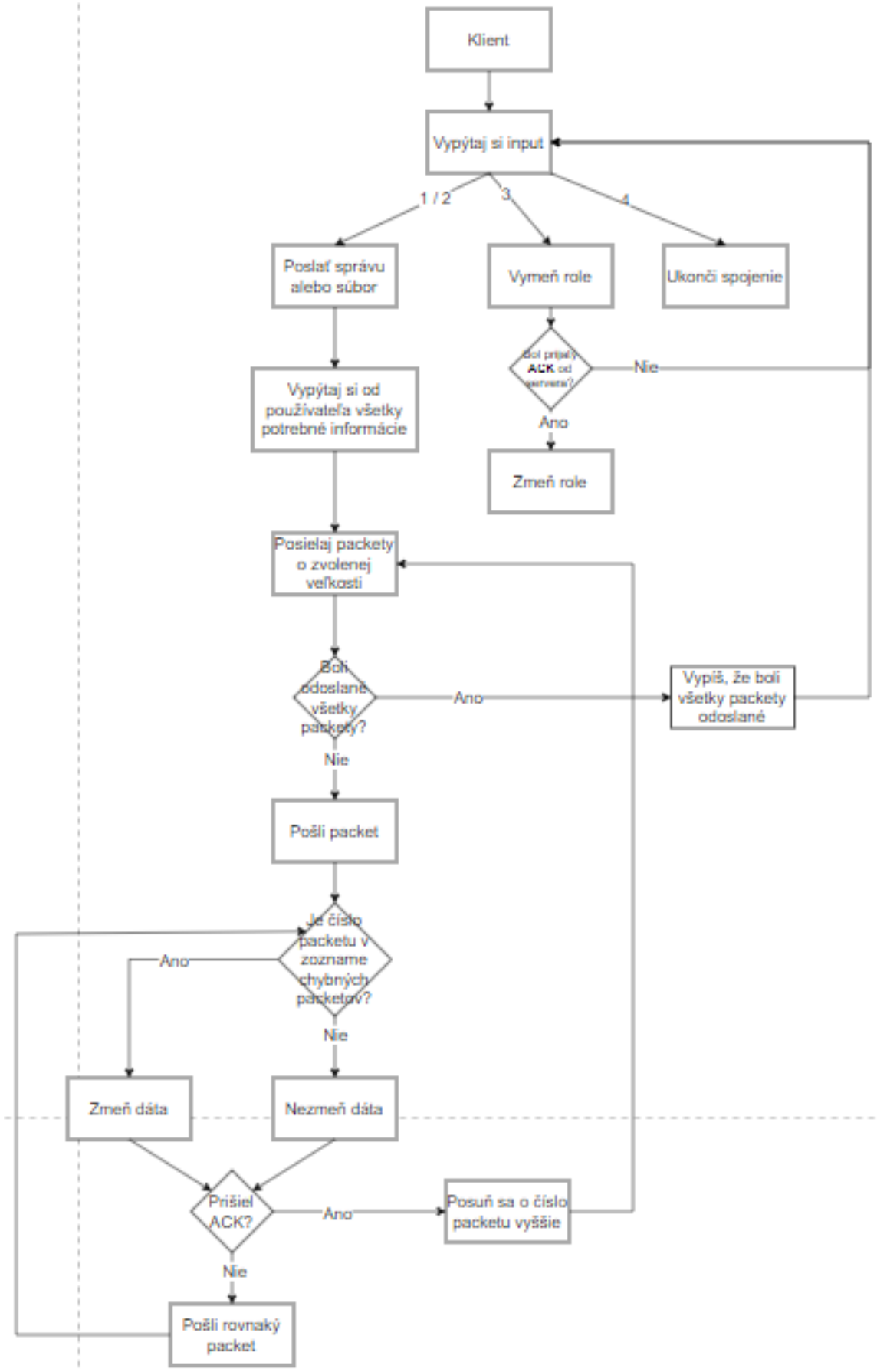
Flowchart

Inicializácia spojenia



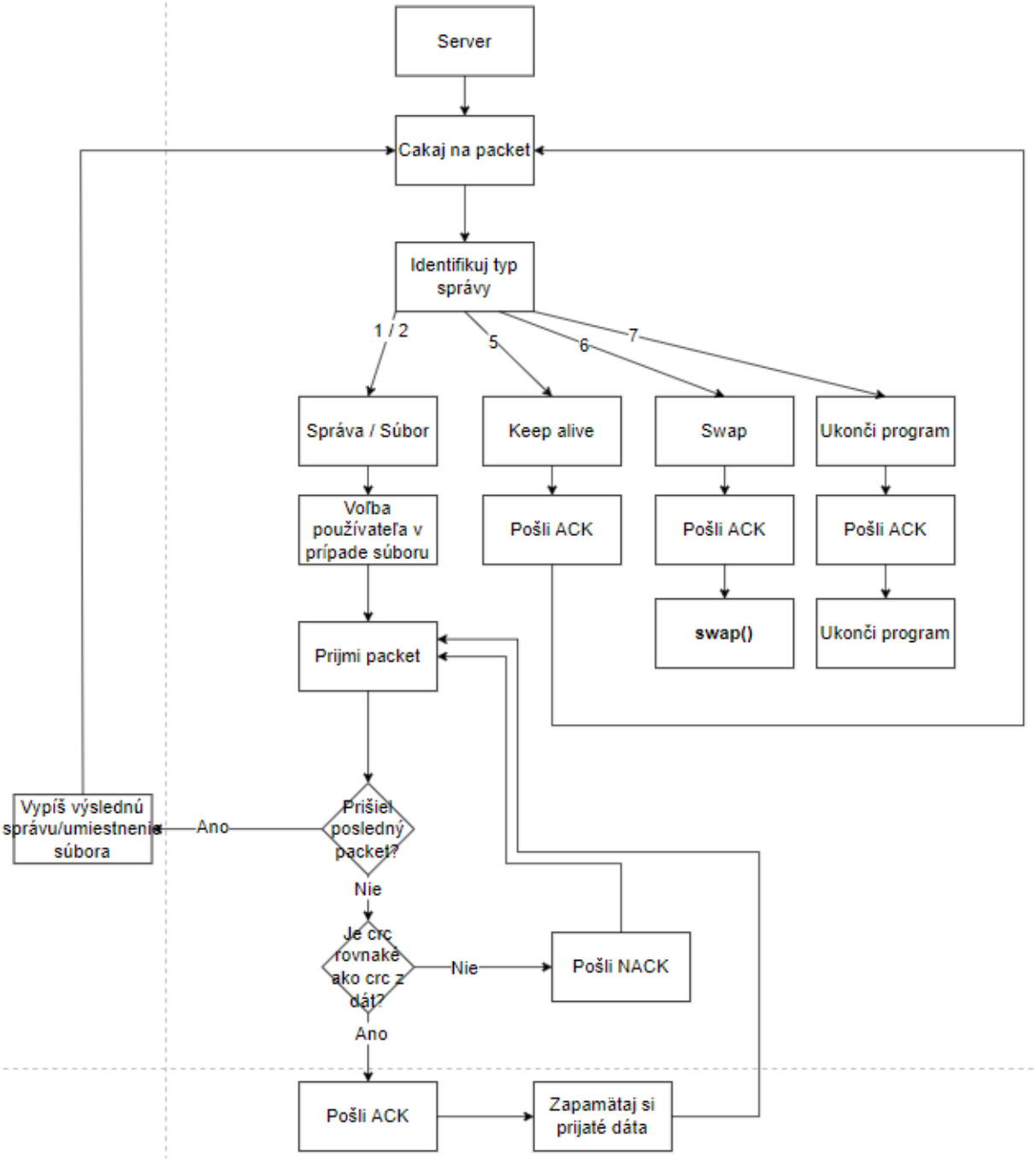
Fedor Viest
ID: 103177

Klient – do flowchartu nebudem opäť dávať voľby používateľa, ktoré sú spomenuté vyššie



Fedor Viest
ID: 103177

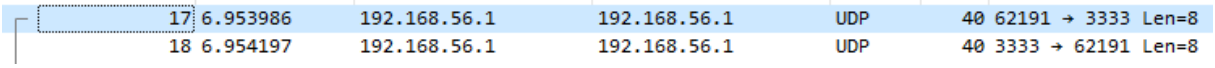
Server – rovnako ako pri klientovi nebudem vo flowcharte znázorňovať voľby používateľa

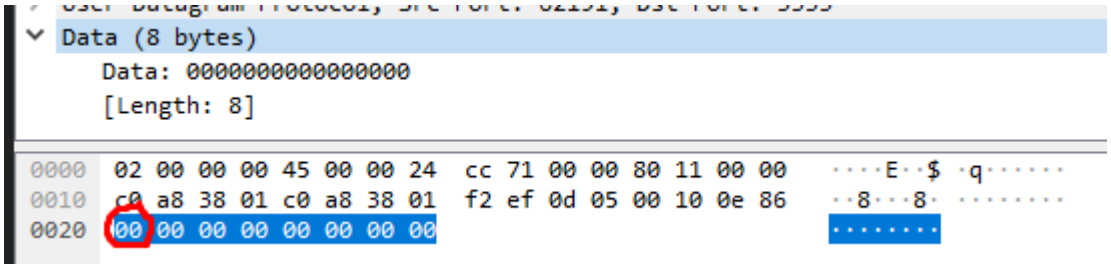


Fedor Viest
ID: 103177

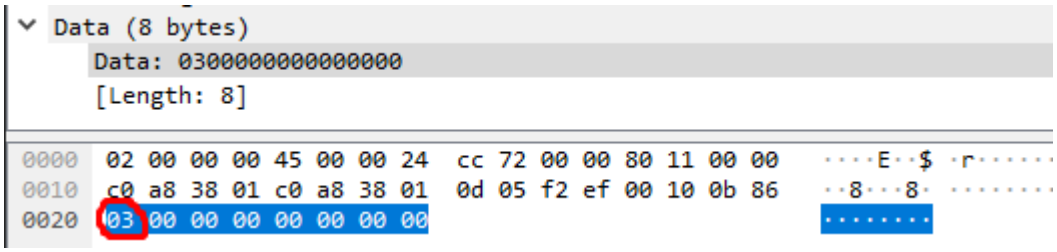
Wireshark

Inicializácia spojenia:



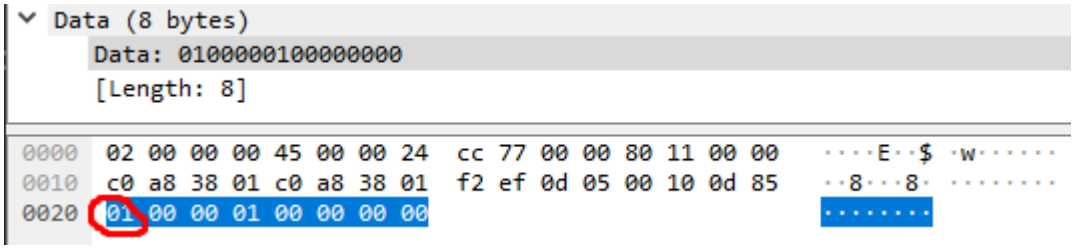


V červenom krúžku je typ správy INIT

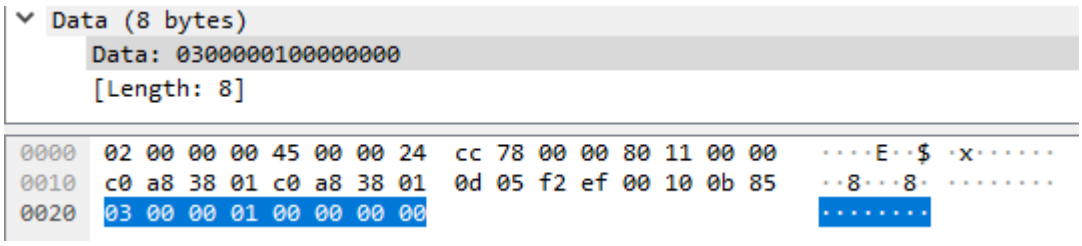


Server odpovedal správou typu 3 -> ACK

Poslanie správy:

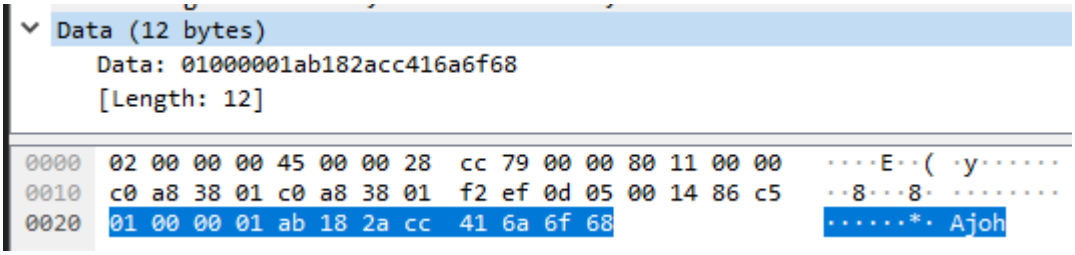


Typ správy 1 -> MSG a zároveň na 4. bajte môžeme vidieť 1, to znamená, že má prísť dokopy 1 packet.

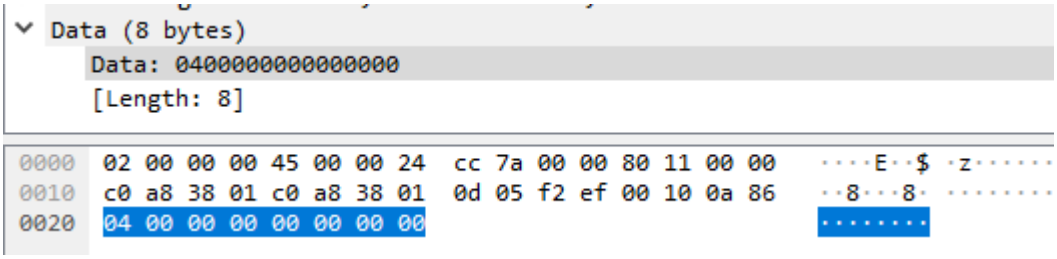


Server odpovedal ACK

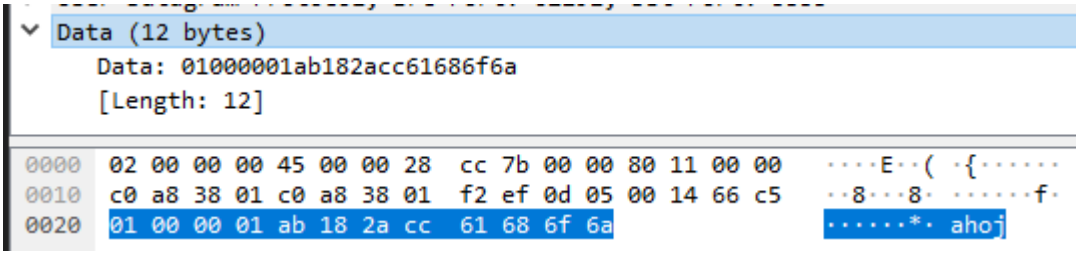
Fedor Viest
ID: 103177



Poslanie 1. packetu v správe, tu už je možné vidieť číslo packetu, za tým 4 bajty crc a za tým data

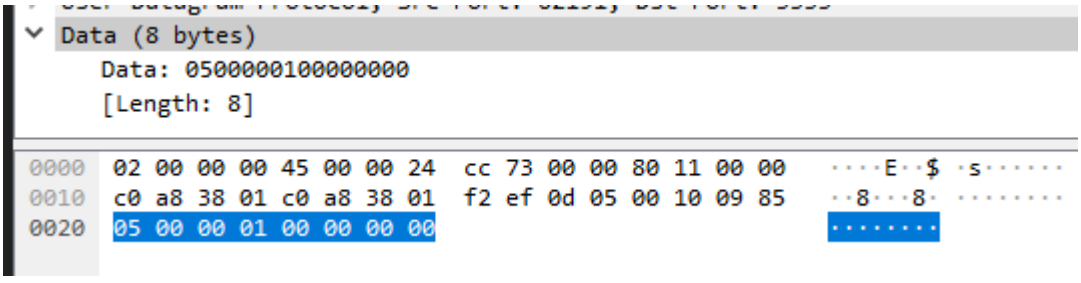


V tomto prípade bola simulovaná chyba, takže server odpovedal typom správy 4 -> NACK



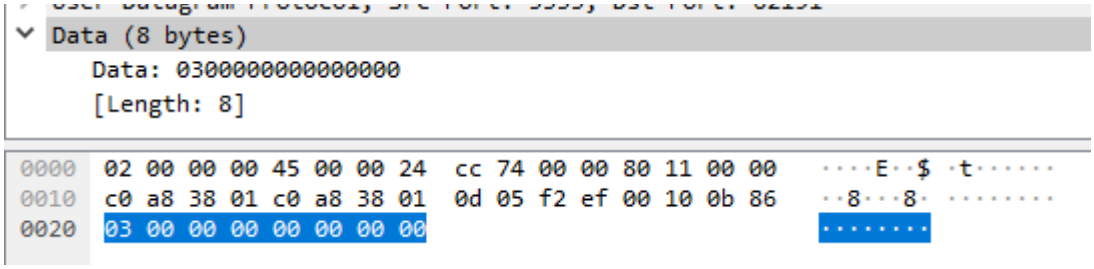
Tu môžeme pozorovať opätovné poslanie packetu s číslom 1 a teraz už so správnymi dátami

Keep alive



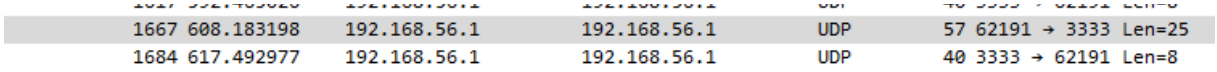
Klient poslal typ správy 5 -> KPA

Fedor Viest
ID: 103177

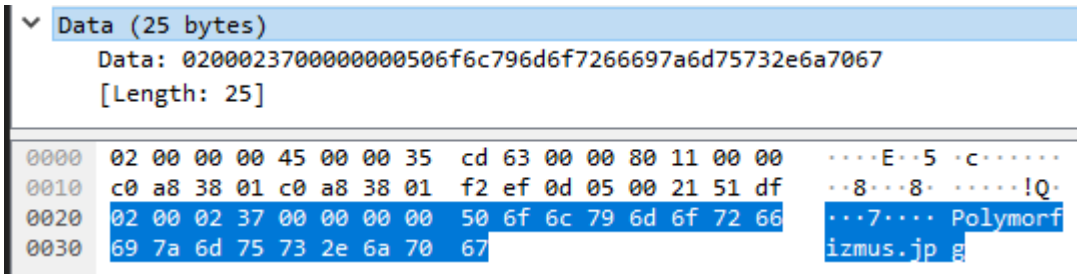


Server odpovedal 3 -> ACK

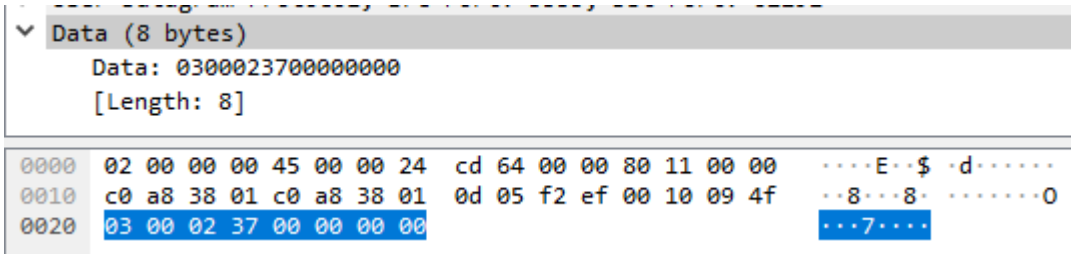
Posielanie súboru:



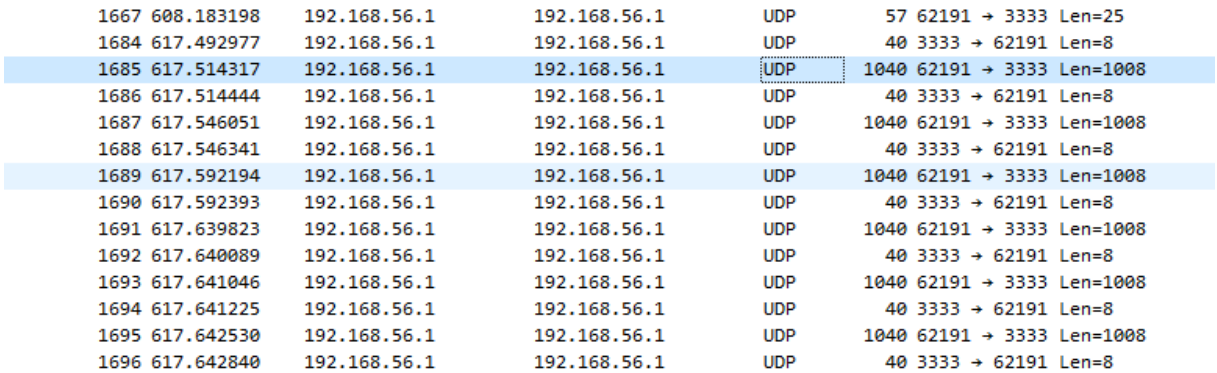
Inicializácia poslania súboru



Typ správy -> 2, počet packetov -> 237(hex) = 567(dec), crc -> 4 bajty, zatiaľ 0, keďže ide o inicializačnú správu, dáta -> názov súboru

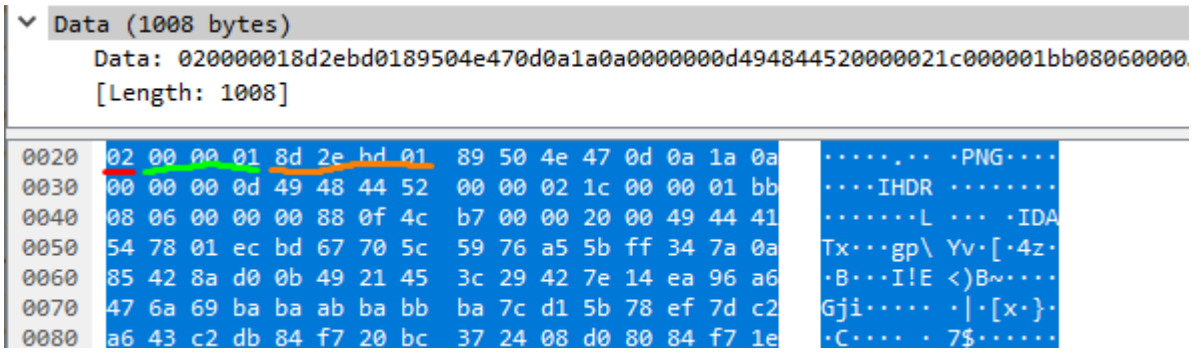


Server odpovedal ACK

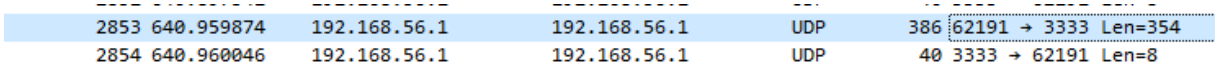


Od packetu číslo 1685 sa začali posielať dáta o veľkosti 1000B + 8B hlavička

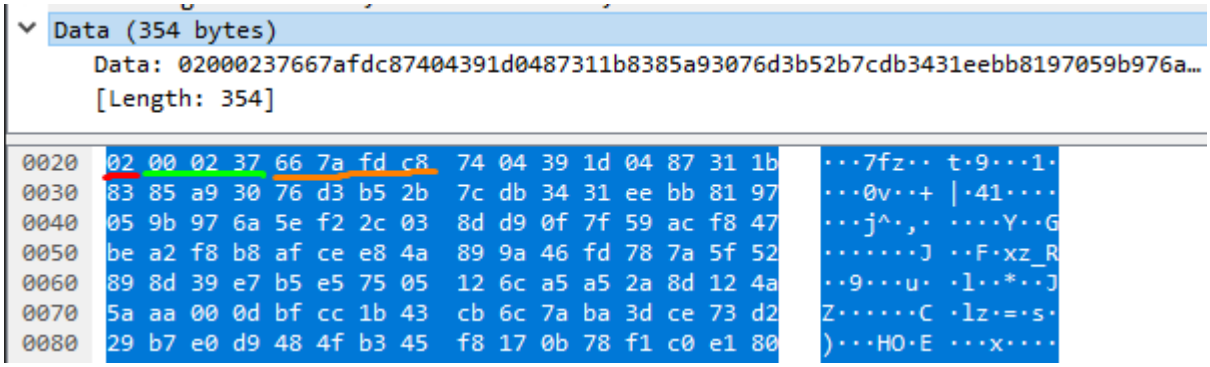
Fedor Viest
ID: 103177



Červenou je typ správy, zelenou číslo packetu, keďže je to prvý, tak 1, oranžovým crc a ostatné sú prenášané dáta

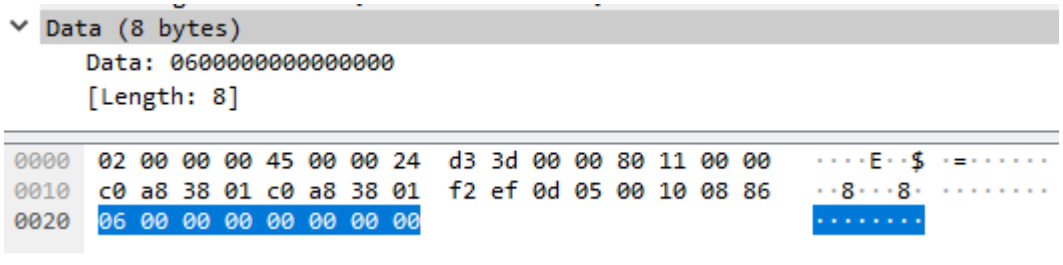


Tu vidno posledný packet prenášaný spolu s potvrdením o prijatí. Ako je možné vidieť, jeho veľkosť je 354B a nie 1008B

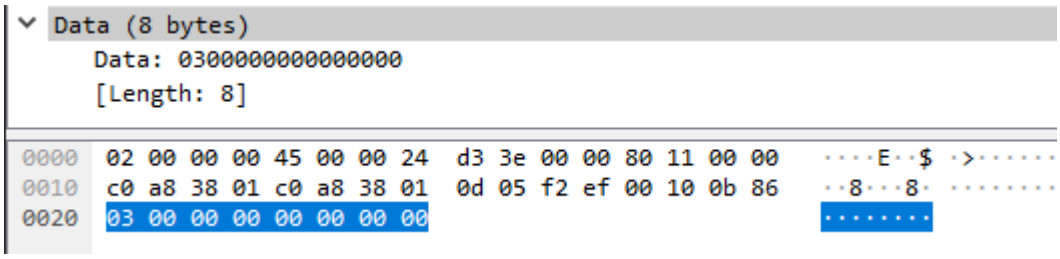


Číslo packetu je teraz 237(hex) = 567(dec) -> posledný packet

Výmena rolí



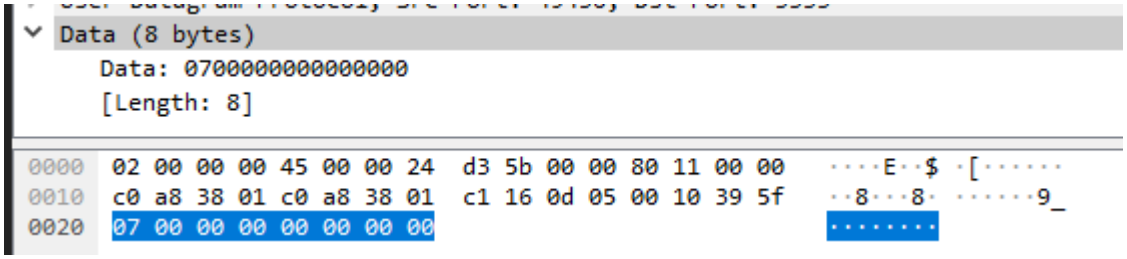
Typ správy od klienta -> 6 SWAP



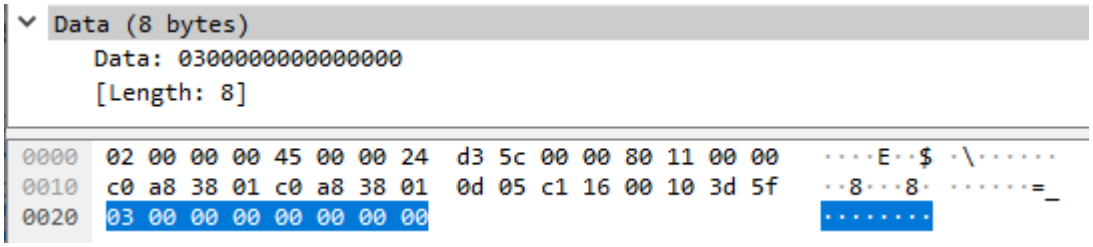
Server potvrdil -> ACK

Fedor Viest
ID: 103177

Koniec spojenia



Klient poslal typ správy 7 -> END



Server poslal ACK