Ľubomír Dlhý, ID: 102897

KOMUNIKÁCIA S VYUŽITÍM UDP PROTOKOLU

Ľubomír Dlhý, ID: 102897

ZADANIE ÚLOHY

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 10-60s pokiaľ používateľ neukončí spojenie. Odporúčame riešiť cez vlastne definované signalizačné správy.

RIEŠENIE

Nad UDP protokolom vytvoríme vlastný pseudo-protokol, ktorý bude zabezpečovať komunikáciu s overovaním chýb a straty paketov a následnou retransmisiou chybných paketov. Protokol sa bude starať o fragmentáciu na užívateľom zadanú veľkosť v rámci dovoleného rozmedzia. Taktiež bude dovolovať prenášanie súborov a textových správ, v rámci čoho treba informovať o začatí a ukončení prenosu, potvrdzovať prijatie fragmentov a prípadné vyžiadanie chybných fragmentov. Spojenie medzi klientom (odosielateľom) a serverom (prijímačom) bude udržiavané keep alive správami.

FRAGMENTÁCIA

Keďže pakety nesmú byť fragmentované na linkovej vrstve, musíme na tejto vrstve dodržať stanovenú veľkosť 1500B. Výsledná maximálna povolená veľkosť pre dátovú časť nášho pseudo-protokolu bude

1500B – veľkosť IP hlavičky (20B) – veľkosť UDP hlavičky (8B) – veľkosť hlavičky vlastného protokolu (9B),

teda ***1463B***. Táto veľkosť bude zároveň maximálna hodnota, ktorú užívateľ môže na vstupe zadať.

Ľubomír Dlhý, ID: 102897

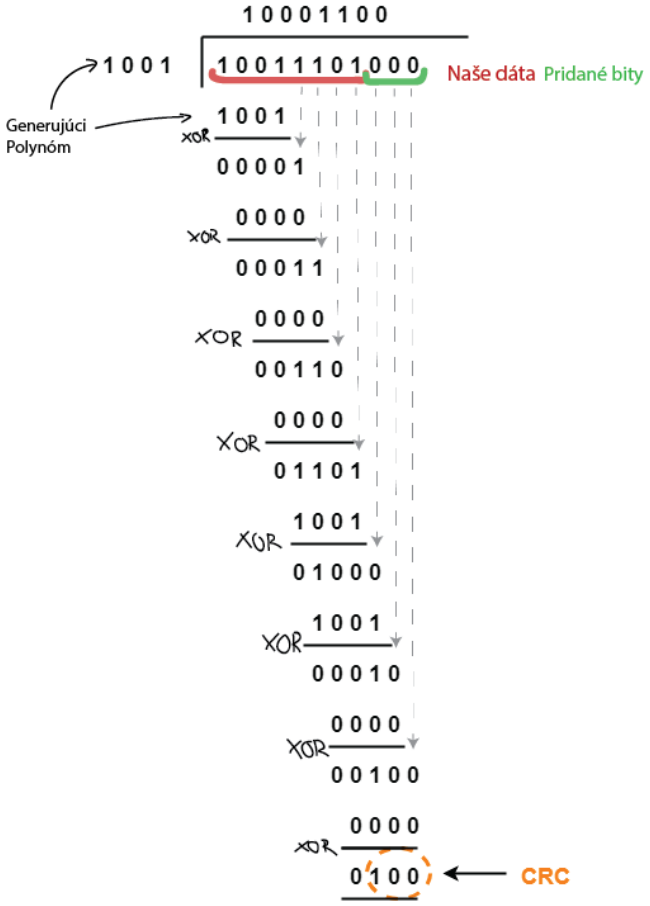
METÓDA KONTROLNEJ SUMY

Na kontrolu správnosti dátovej časti používam metódu CRC – kontrola cyklickým kódom. Používam generujúci polynóm

$$x^{16} + x^{12} + x^5 + 1$$

Tento kód je taktiež známy pod názvom CRC16-CCITT a je všeobecne uznávaný inštitúciou CCITT.
Postup výpočtu CRC:

Na začiatku sa správa, z ktorej chceme vypočítať CRC vyjadrí sekvenciou bitov a k nej sa pridá toľko núl, aký je najväčší exponent v generujúcom polynóme (v našom prípade 16). Potom sa táto sekvencia delí generujúcim polynómom. V binárnej sústave to prebieha pomocou operácie XOR. Výsledok tejto operácie opäť XOR-ujeme s generujúcim polynómom až pokiaľ je výsledok tejto operácie menší ako generujúci polynóm. Výsledkom je sekvencia, ktorá nám zostala, čiže zvyšok.



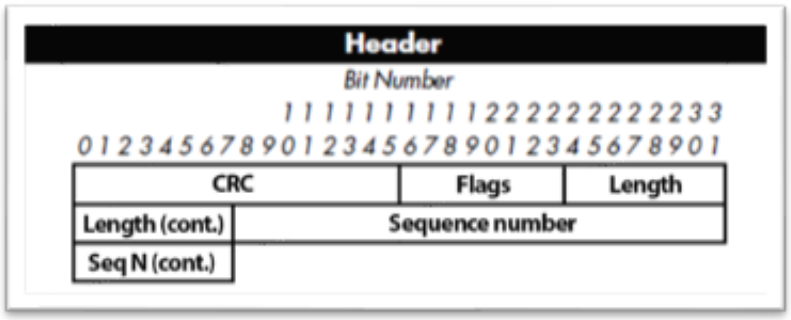
Príklad počítania CRC

Zdroj: https://en.wikipedia.org/wiki/Cyclic_redundancy_check

Ľubomír Dlhý, ID: 102897

HLAVIČKY

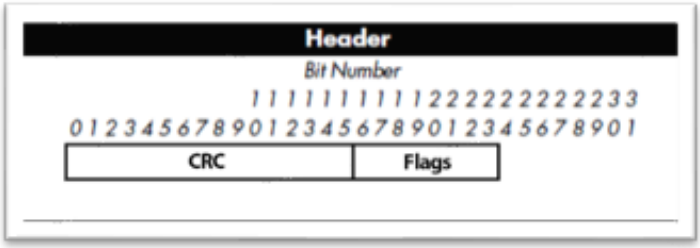
Protokol bude používať 2 druhy hlavičiek – pre prenos súborov alebo správ a pre udržiavanie spojenia.



Hlavička obsahuje polia:

- **CRC (2B)** – Cyclic redundancy check, resp. Kontrola cyklickým kódom slúži na overenie správnosti prenášaných dát. Bude použitý CRC16, ktorý vracia výsledok o veľkosti 2B
- **Flags (1B)** – príznaky, ktoré budú identifikovať, aký účel má paket:
  - ACK – acknowledgement – slúži na potvrdenie prijatia fragmentu, resp. paketu
  - NACK – negative acknowledgement – slúži na vyžiadanie preposlania chybného fragmentu
  - TEXT – identifikuje typ prenosu ako textovú správu
  - FILE – identifikuje typ prenosu ako súbor
  - CONN_INIT – inicializácia spojenia, resp. prenosu súboru alebo textovej správy
  - CONN_FIN – ukončenie prenosu súboru alebo textovej správy
  - KEEP_ALIVE – príznak určujúci, že ide o paket udržujúci spojenie
- **Length (2B)** – dĺžka dátovej časti paketu, maximálna hodnota je 1463
- **Sequence number (4B)** – poradie fragmentu, dôležité pre usporiadanie fragmentov, keďže fragmenty nemusia prísť v poradí alebo je nutné pre vyžiadanie znovuodoslania fragmentu

Hlavička pre *udržiavanie spojenia* (keep alive) neobsahuje polia Length a Sequence number, keďže sú zbytočné.

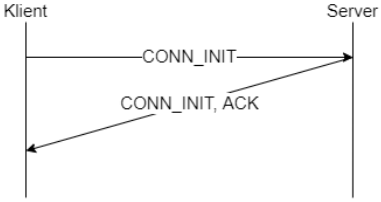


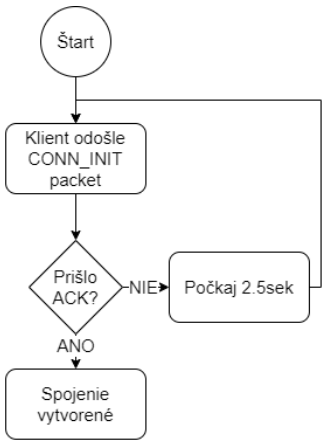
Hlavička pre udržiavanie spojenia (keep alive)

FUNGOVANIE PROGRAMU

- ***Inicializácia spojenia***

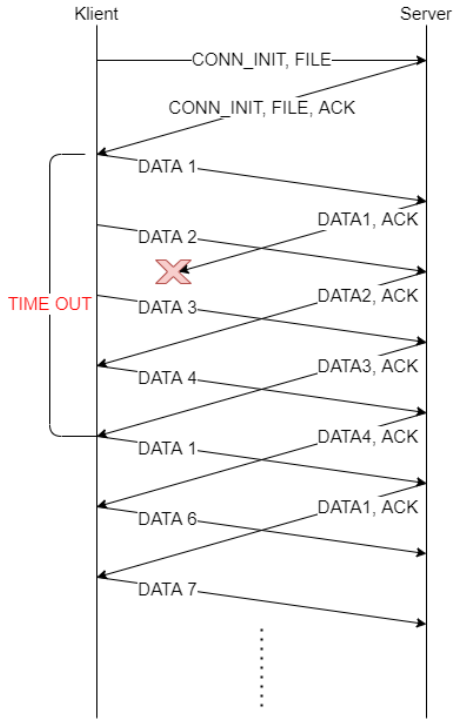
Klient sa po spustení snaží o vytvorenie spojenia so serverom pomocou posielania inicializačného paketu (Packet s CONN_INIT flagom). Ak server do 2.5sek neodpovie pozitívnym potvrdením, tak klient odošle opäť packet na adresu servera. Toto sa vykonáva až pokiaľ server neodpovie potvrdením.



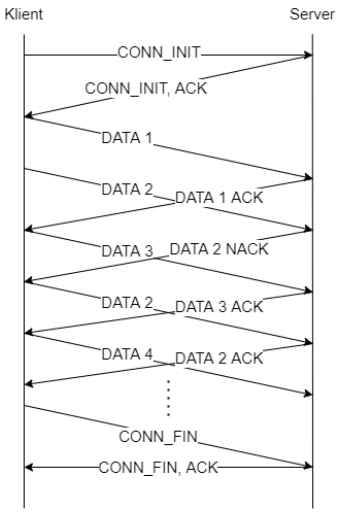


- ***Prenos súboru***

Na začiatku prenosu sa odošle prvý informačný paket (budú zapnuté príznaky File a CONN_INIT), ktorý bude v hlavičke niesť informáciu o počte fragmentov v poli Sequence Number a maximálnej veľkosti fragmentu v poli Length. V dátovej časti sa bude nachádzať názov súboru. Server na informačný paket odpovie taktiež paketom rovnakého typu ale so zapnutým príznakom ACK, čím potvrdí prijatie inicializačného paketu.
Keďže je použitá metóda Selective repeat ARQ, budú odosielané dátové pakety z odosielateľa a prijímač bude na ne odpovedať potvrdzovacou správou (ACK). Ak sa nájde chybný fragment, prijímač požiada o preposlanie fragmentu pomocou negatívneho potvrdenia (**NACK**) a vysielač daný fragment prepošle, zatiaľ čo ďalšie fragmenty už nie je nutné preposielať, keďže sa nachádzajú v bufferi prijímajúceho uzlu.
Klient má taktiež naimplementované časovače na každý posielaný packet kvôli prípadnej strate fragmentu alebo ACK. **Časovač** je nastavený na **200ms**. Ak nebolo prijaté ACK do tohto časového limitu, klient daný fragment prepošle. Takto sa odošlú všetky fragmenty a smie sa ukončiť prenos súboru. Dátové pakety obsahujú informácie o dĺžke dát a poradia fragmentu.
Ukončenie prenosu prebieha pomocou odoslania informačného paketu so zapnutými príznakmi CONN_FIN a FILE, ktorý značí, že prenos bol úspešný a smie sa ukončiť. Prijímací uzol naň odpovie rovnakým paketom, iba so zapnutým ACK príznakom, čiže ide o potvrdenie. Súbor bude uložený v ceste, akú užívateľ zadal a súbor bude mať rovnaký názov, aký vysielač odoslal v prvom (informačnom) pakete.



Stratené ACK, time out prepošle packet



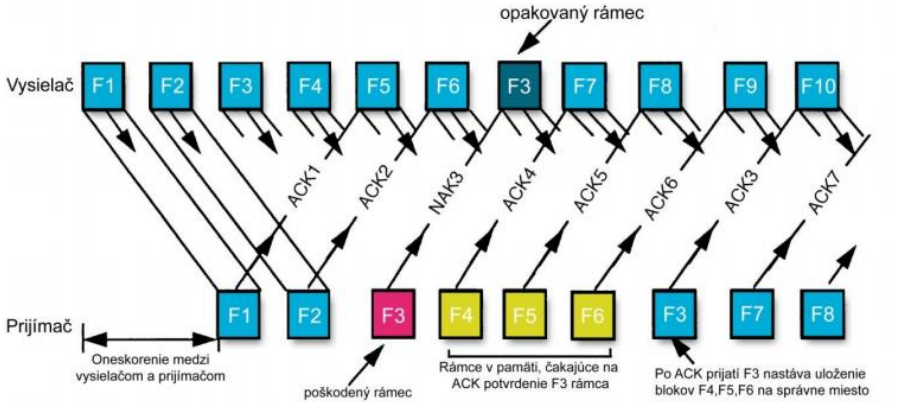
Prenos súboru s jedným chybným fragmentom (DATA 2)

- ***Posielanie textovej správy***

Odosielanie textovej správy bude prebiehať analogicky ako pri prenose súboru, iba nebude potrebné odosielať názov súboru v prvom informačnom pakete a taktiež budú zapnuté príznaky TXT a nie FILE.

- ***ARQ***

V programe máme použitú metódu ARQ Selective reapeat, čiže ARQ so Selektívnym opakovaním, v ktorom príjemca (server) potvrdzuje (ACK) každý bezchybný fragment a odosielateľ preposiela iba tie fragmenty, na ktoré bolo prijaté negatívne potvrdenie (NACK) alebo tie, na ktoré vypršal časovač 200ms.



Prijímač aj vysielač majú okno o konštantnej veľkosti. Vysielač dokáže vysielať fragmenty iba v rámci svojho okna. Ak bol fragment potvrdený prijímacím uzlom, okno sa posunie o 1 fragment ďalej. Prijímajúci uzol prijíma fragmenty a korektné si ukladá do bufferu, resp. keď sú v poradí, smie ich ihneď zapísať. Ak fragment je chybný, vyšle sa negatívne potvrdenie (NACK) a vysielač odošle tento fragment ihneď, ako bude NACK prijaté. Prijímač si udržuje ostatné fragmenty v bufferi a po znovuprijatí chybného fragmentu, jednotlivé rámce preusporiada do správneho poradia a zapíše.

Ľubomír Dlhý, ID: 102897

IMPLEMENTAČNÉ PROSTREDIE

Program je implementovaný v jazyku Python, verzia 3.8.5 a sú použité nasledovné knižnice:

- socket - posielanie dát medzi uzlami v sieti
- crcmod - vypočítanie CRC16 kódu (http://crcmod.sourceforge.net/)
- time - keep alive časovač, ARQ časovače, trvanie prenosu
- threading - keep alive thread, client thread, server thread, ARQ thready
- struct - práca s jednotlivými bytemi na vytváranie a rozbaľovanie hlavičky
- tkinter - grafické rozhranie

ZMENY OPROTI NÁVRHU

- ***Používateľské rozhranie***

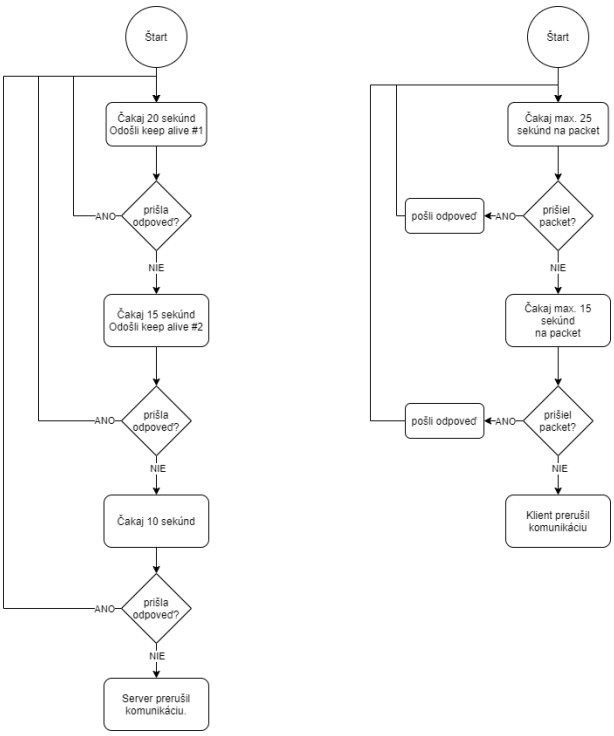
V okne servera nie je potrebné zadávať IP. V klientovom okne boli rozšírené možnosti napr. s typom vloženej chyby. Celkovo má GUI viac možností ako v návrhu.

- ***Udržiavanie spojenia***

Na udržiavanie spojenia sú urobené špeciálne signalizačné Keep Alive správy. Tieto správy sú vysielané klientom (vysielacím uzlom) a potvrdzované serverom.

Klient pošle keep alive packet najprv po 20 sekundách, ak však nedostane do 15 sekúnd odpoveď, tak odošle druhý keep alive, ale tentokrát s časovačom 10 sekúnd. Ak nepríde odpoveď ani na túto správu, server sa odpojil a ukončí sa spojenie.

Server má naimplementovaný časovač na 25 sekúnd pre prvý keep alive. Ak nedostane do 25 sekúnd žiaden packet, dá časovač ešte na 15 sekúnd. Ak ani do tohto času nepríde žiaden packet od klienta, klient sa odpojil a ukončíme spojenie.



Klient *Server*

- ***Fungovanie programu***
  - ***Inicializácia spojenia***

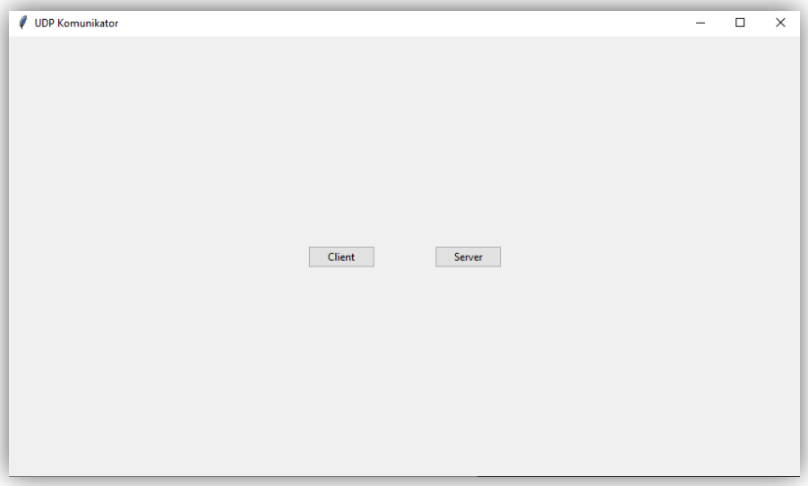
Spojenie sa neinicializuje pri prenose súboru, ale pri stlačení tlačidla spustiť v grafickom rozhraní.

  - ***ARQ***

ARQ metóda obsahuje časovače na odosielané fragmenty práve na také prípady, kedy by sa stratil fragment alebo ACK a bolo by nutné odoslať tento fragment opäť.

Ľubomír Dlhý, ID: 102897

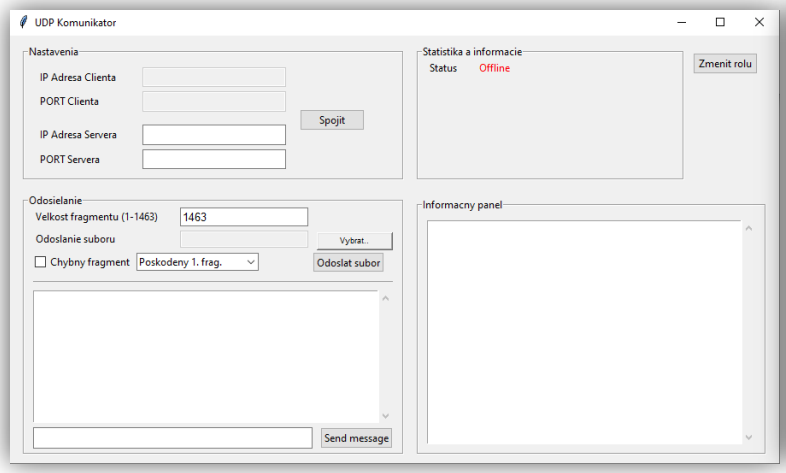
POUŽÍVATEĽSKÉ ROZHRANIE



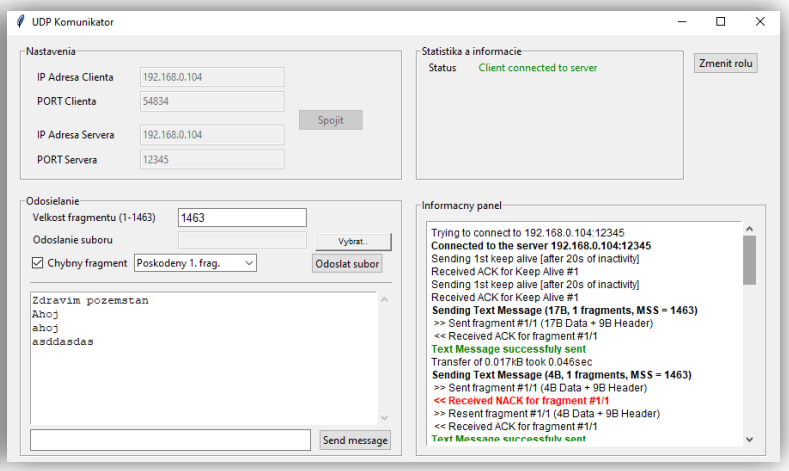
Používateľské rozhranie je robené formou GUI – grafického prostredia. Po spustení programu sa zobrazí hlavné menu, kde používateľ vyberie svoju rolu. Má na výber **Client (Vysielač)** alebo **Server (Prijímač).**

Klientovo okno pozostáva zo 4 častí – *Nastavenia, Štatistika a informácie, Odosielanie, Informačný panel.*

- ➔ V paneli *Nastavenia* si užívateľ zvolí IP a Port servera a po stlačení tlačidla **Spojiť** sa bude pokúšať pripojiť na server. Do okien IP Adresa Clienta a PORT Clienta sa vložia reálne informácie o klientovom sockete – t.j. IP jeho stroja a pridelený port.
- ➔ V paneli *Štatistika a informácie* je iba informácie o statuse daného uzla v rámci spojenia. To znamená, či je daný uzol offline, počúva alebo je spojený s druhým uzlom.
- ➔ V časti *Odosielanie* má užívateľ možnosť nastaviť veľkosť fragmentu, zvoliť a odoslať súbor, posielať textové správy a vybrať vloženie chyby do fragmentu – buď poškodenie 1. fragmentu, alebo stratenie ACK 1. fragmentu.
- ➔ *Informačný panel* informuje užívateľa o všetkých udalostiach – zmeny v spojení, odosielanie a potvrdzovanie fragmentov, atď..



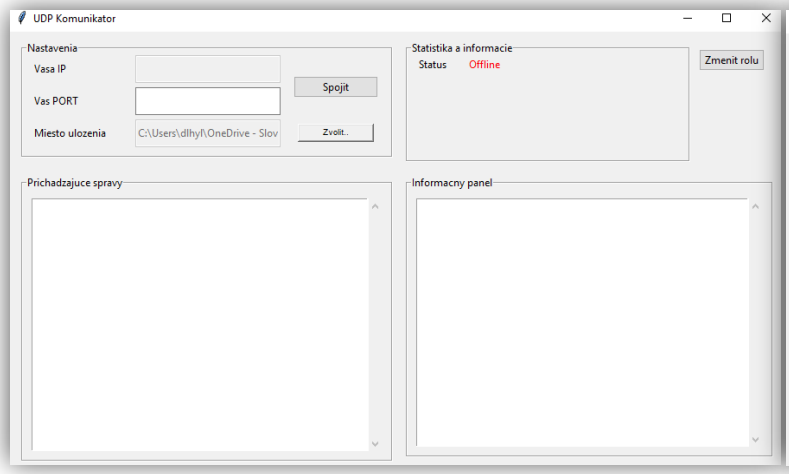
Okno klienta po spustení



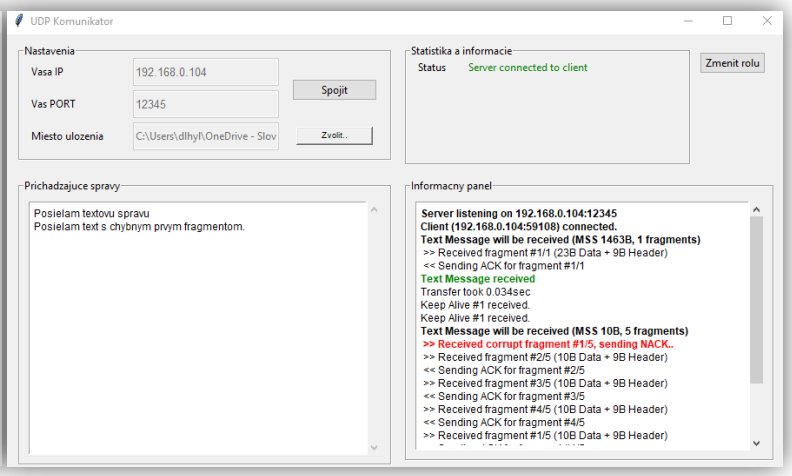
Okno klienta v prebiehajúcej komunikácii

Okno servera pozostáva taktiež zo 4 častí – *Nastavenia, Štatistika a informácie, Prichádzajúce správy* a *Informačný panel.*

- ➔ V paneli *Nastavenia* si užívateľ zvolí Port servera a po stlačení tlačidla **Spojiť** bude server počúvať na danom porte a IP adrese daného stroja. Okno Vaša IP sa vyplní po stlačení tlačidla reálnou IP adresou stroja.
- ➔ V paneli *Štatistika a informácie* je iba informácie o statuse daného uzla v rámci spojenia. To znamená, či je daný uzol offline, počúva alebo je spojený s druhým uzlom.
- ➔ V časti *Prichádzajúce správy* sa prijímacému uzlu zobrazia všetky prichádzajúce textové správy, ktoré vyslal klient.
- ➔ *Informačný panel* informuje užívateľa o všetkých udalostiach – zmeny v spojení, odosielanie a potvrdzovanie fragmentov, atď..

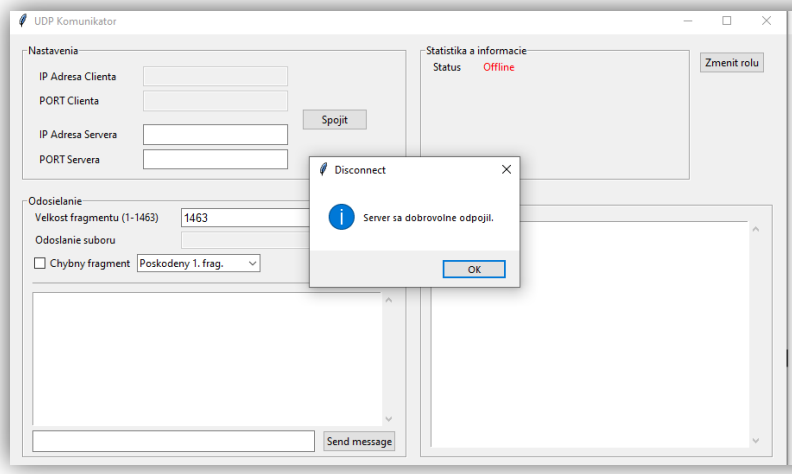


Okno servera ihneď po spustení

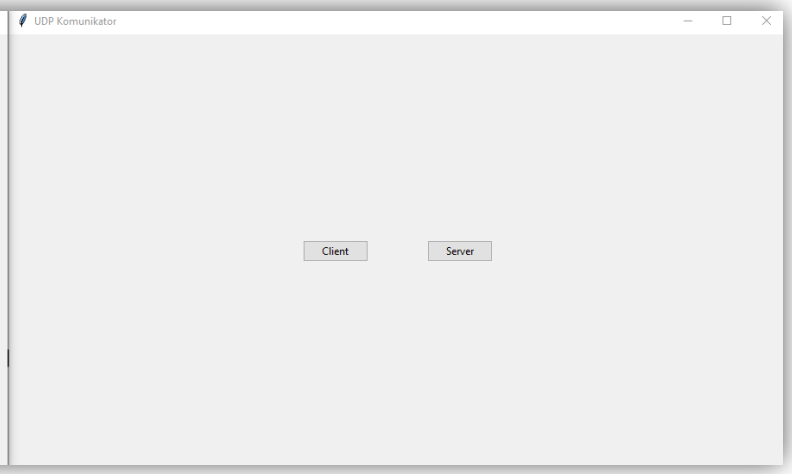


Okno servera v komunikácii s klientom

Oba uzly majú v pravom hornom rohu obrazovky tlačidlo **Zmeniť rolu.** Toto tlačidlo spôsobuje dobrovoľné ukončenie komunikáce. Po stlačení tlačidla daný uzol vyšle informačný packet s príznakom CONN_FIN, čím informuje opačný uzol o dobrovoľnom odpojení. Uzol, ktorý inicioval odpojenie sa vráti do hlavného menu a opačný uzol vypíše informáciu o odpojení a resetne komunikáciu.

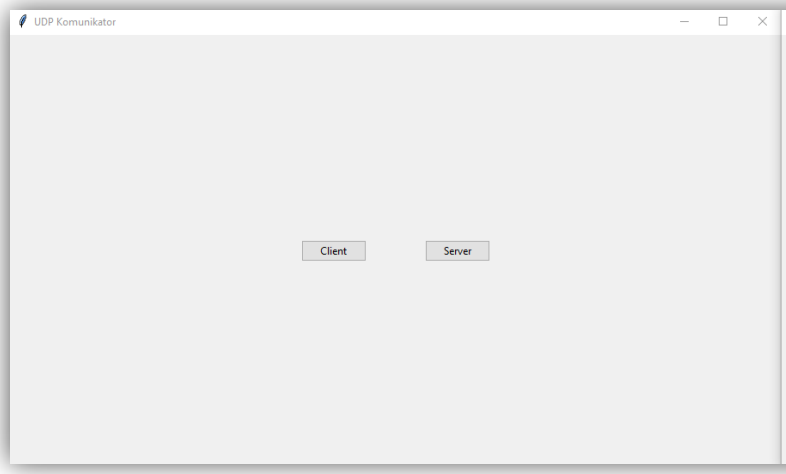


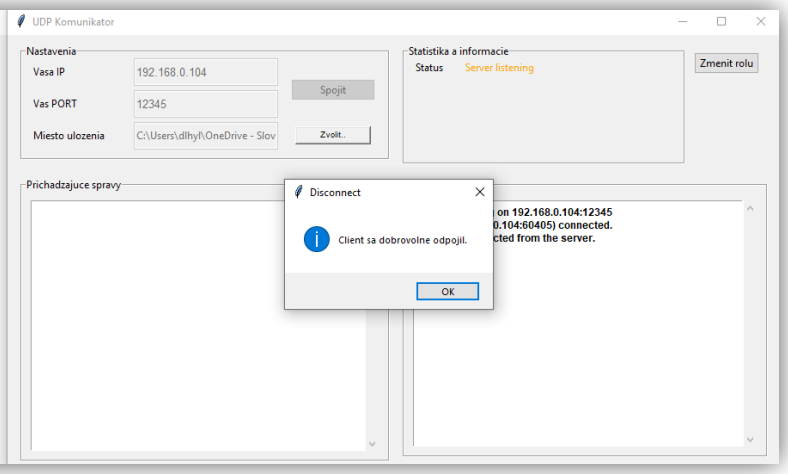
KLIENT



SERVER – Stlačil ZMEŇ ROLU

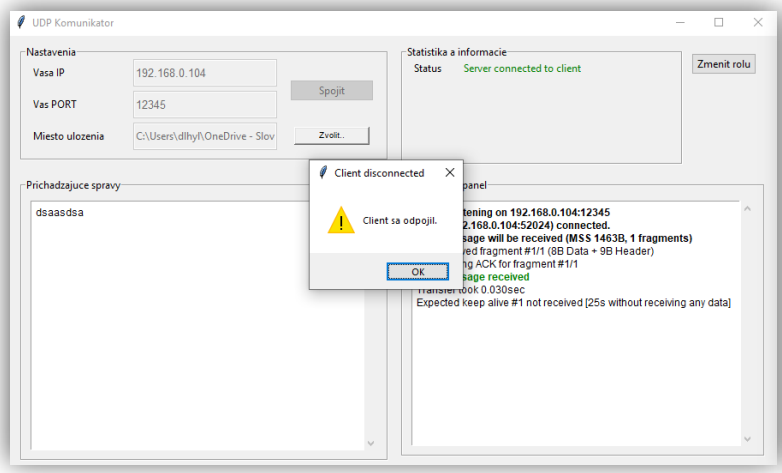
Ľubomír Dlhý, ID: 102897

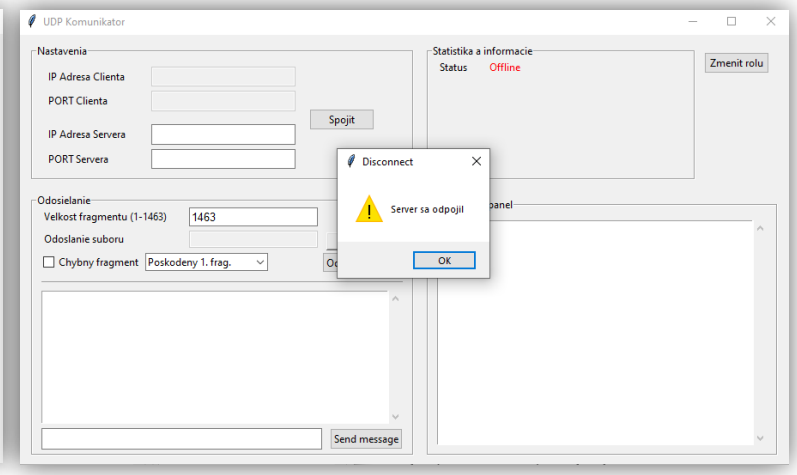




KLIENT– Stlačil ZMEŇ ROLU *SERVER*

Pri nedobrovoľnom odpojení – t.j. užívateľ nestlačil tlačidlo Zmeň rolu – sa spojenie **ukončí** pomocou **keep alive** signalizačných správ. Ak po 2 keep alive správach nepríde odpoveď klientovi, server sa odpojil. Ak server neprijme žiadnu z dvoch keep alive správ do určených časových okien zväčšených o 5 sekundovú rezervu, tak sa odpojil klient. Po zistení neočakávaného ukončenia spojenia sa vypíše správa.

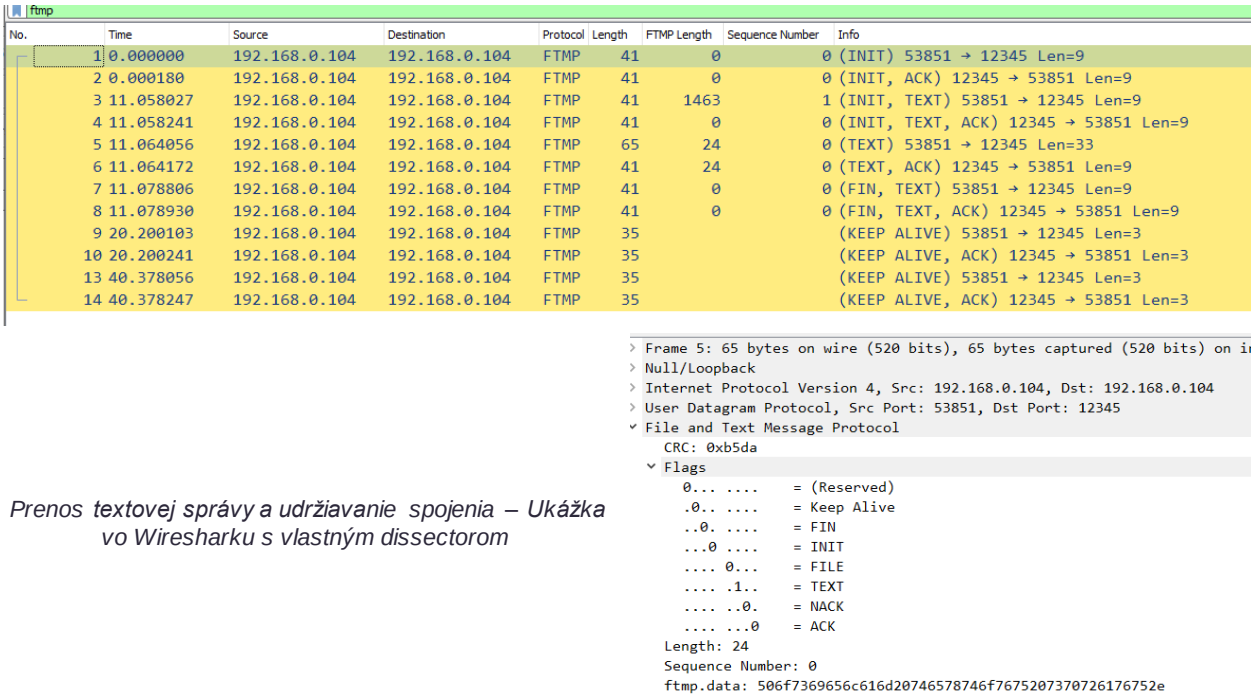




Ľubomír Dlhý, ID: 102897

WIRESHARK

Pre správne otestovanie a jednoduchšie rozpoznávanie packetov môjho protokolu som vytvoril pre program **Wireshark dissector** – kód, ktorý identifikuje rámce podľa protokolu a vypíše jednotlivé polia a dôležité informácie v jednoduchom a prívetivom spôsobe. Na to musím protokol viazať na port, ja svojmu protokolu **FTMP** prideľujem **port 12345**.



ftmp

No.	Time	Source	Destination	Protocol	Length	FTMP Length	Sequence Number	Info
1	0.000000	192.168.0.104	192.168.0.104	FTMP	41	0	0	(INIT) 53851 → 12345 Len=9
2	0.000180	192.168.0.104	192.168.0.104	FTMP	41	0	0	(INIT, ACK) 12345 → 53851 Len=9
3	11.058027	192.168.0.104	192.168.0.104	FTMP	41	1463	1	(INIT, TEXT) 53851 → 12345 Len=9
4	11.058241	192.168.0.104	192.168.0.104	FTMP	41	0	0	(INIT, TEXT, ACK) 12345 → 53851 Len=9
5	11.064056	192.168.0.104	192.168.0.104	FTMP	65	24	0	(TEXT) 53851 → 12345 Len=33
6	11.064172	192.168.0.104	192.168.0.104	FTMP	41	24	0	(TEXT, ACK) 12345 → 53851 Len=9
7	11.078806	192.168.0.104	192.168.0.104	FTMP	41	0	0	(FIN, TEXT) 53851 → 12345 Len=9
8	11.078930	192.168.0.104	192.168.0.104	FTMP	41	0	0	(FIN, TEXT, ACK) 12345 → 53851 Len=9
9	20.200103	192.168.0.104	192.168.0.104	FTMP	35			(KEEP ALIVE) 53851 → 12345 Len=3
10	20.200241	192.168.0.104	192.168.0.104	FTMP	35			(KEEP ALIVE, ACK) 12345 → 53851 Len=3
13	40.378056	192.168.0.104	192.168.0.104	FTMP	35			(KEEP ALIVE) 53851 → 12345 Len=3
14	40.378247	192.168.0.104	192.168.0.104	FTMP	35			(KEEP ALIVE, ACK) 12345 → 53851 Len=3


```
> Frame 5: 65 bytes on wire (520 bits), 65 bytes captured (520 bits) on i
> Null/Loopback
> Internet Protocol Version 4, Src: 192.168.0.104, Dst: 192.168.0.104
> User Datagram Protocol, Src Port: 53851, Dst Port: 12345
v File and Text Message Protocol
    CRC: 0xb5da
  v Flags
      0... ....    = (Reserved)
      .0.. ....    = Keep Alive
      ..0. ....    = FIN
      ...0 ....    = INIT
      .... 0...    = FILE
      .... .1..    = TEXT
      .... ..0.    = NACK
      .... ...0    = ACK
    Length: 24
    Sequence Number: 0
    ftmp.data: 506f7369656c616d2074657874746f767520737072617752e
```

Prenos textovej správy a udržiavanie spojenia – Ukážka vo Wiresharku s vlastným dissectorom

ZÁVER

Program prenáša dáta a textové správy pomocou špeciálneho protokolu nad UDP s prvkami TCP protokolu, ktoré zabezpečujú výrazne väčšiu ochranu a zabezpečenie. Dáta sú fragmentované tak, aby sa nefragmentovali na linkovej vrstve. Posielané informácie sú overované pomocou CRC-16, čím sa zaručí správne prijatie. Spojenie je otvárané a ukončované pomocou špeciálnych paketov a udržiavané pomocou Keep Alive signálov. Správnosť doručenia fragmentov a retransmisia poškodených alebo stratených paketov je riešené pomocou selektívneho ARQ. Riešenie ukázalo výhody UDP v časovej efektivite a bezpečnostné prvky TCP.