Fakulta informatiky a informačných technológií

Slovenská technická univerzite v Bratislave

Zadanie 2: Komunikácia s využitím UDP protokolu

Počítačové a komunikačné siete

Tomáš Brček

Cvičenie: Utorok 16:00

2023/2024

Obsah

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 zapnutí programu, komunikátor automaticky odosiela paket pre udržanie spojenia každých 5s pokiaľ používateľ neukončí spojenie ručne. Odporúčame riešiť cez vlastne definované signalizačné správy a samostatné vlákno.

Návrh

Implementácia

Na implementáciu programu s použitím vlastného protokolu nad UDP, som si zvolil ako programovací jazyk Python. Tento jazyk som si zvolil z dôvodu, že mi je najbližší, najlepšie sa mi v ňom pracuje, ale taktiež má množstvo vstavaných knižníc, ktoré poskytujú veľa rôznych funkcií, ktoré uľahčujú prácu, a teda nemusím všetko implementovať ručne.

Na interakciu používateľa s programom bude použitá konzola s jednoduchým a intuitívnym menu tak, aby to bolo pre používateľa jednoduché používať.

Fungovanie programu

Po spustení programu si používateľ vyberie, či chce byť **klient**, a teda bude odosielať dáta, alebo či chce byť **server,** a teda bude počúvať a prijímať dáta. Neskôr v programe bude možné medzi týmito funkciami prepínať.

```
1 - Server
2 - Client
3 - exit
--------------------------------------------------------------------------------
Choice:
```

Klient – odosielanie dát

V prípade, že si používateľ zvolí, že chce byť klientom, program si od neho vypýta IP adresu servera a port, na ktorý sa budú odosielať dáta. Na túto adresu sa následne odošle správa na nadviazanie spojenia.

```
--------------------------------------------------------------------------------
Choice: 2
--------------------------------------------------------------------------------
Server address: 192.168.0.12
Port: 45678
```

Po nadviazaní spojenia so serverom sa zobrazí menu, v ktorom si používateľ môže vybrať, čo sa bude vykonávať.

```
--------------------------------------------------------------------------------
1 - Send a message
2 - Send a file
3 - Show Keep-Alive messages
4 - Don't show Keep-Alive messages
5 - Switch users
6 - End
--------------------------------------------------------------------------------
Zadajte volbu:
```

Na výber má z možností:

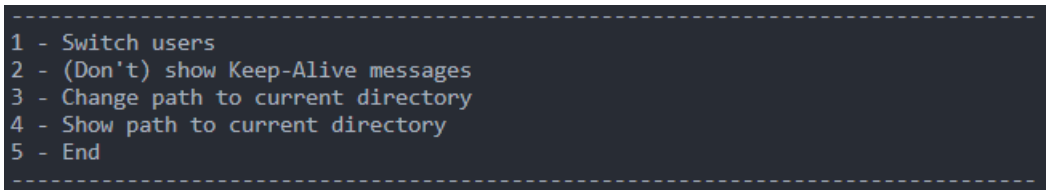
1. **Odoslanie správy**
   - po zvolení tejto možnosti používateľ zadá textovú správu, ktorú chce odoslať na server

- po zadaní správy, zvolí tiež veľkosť fragmentu, ak je zadaná veľkosť fragmentu väčšia ako je veľkosť správy, správa sa odošle ako jeden fragment, ak je však zadaná veľkosť fragmentu menšia ako veľkosť správy, táto správa sa rozdelí na menšie časti, ktoré sa po jednom odošlú
- ešte pred odosielaním sa program používateľa spýta, či chce simulovať chyby, ktoré nastávajú s pravdepodobnosťou 5% pre každý fragment
- najprv sa odošle inicializačná správa, ktorá hovorí serveru, že klient ide odosielať dáta a koľko fragmentov odošle
- po prijatí ACK na inicializačnú správu, môže klient začať odosielať dáta
- na každý odoslaný fragment musí prísť správa o prijatí tohto fragmentu (ACK)
- ak príde NACK, teda doručená správa je chybná, fragment, na ktorý prišiel NACK sa odošle znova
- po prijatí ACK na všetky fragmenty, sa ešte odošle správa o ukončení odosielania dát
- doručená bola celá správa a táto sa vypíše na strane servera

2. **Odoslanie súboru**
   - po zvolení tejto možnosti používateľ zadá názov súboru v aktuálnom adresári, ktorý chce odoslať na server, v prípade, že súbor s takýmto názvom neexistuje, vypíše sa chybová hláška a bude sa čakať na nový názov súboru
   - po korektnom zadaní názvu súboru, zvolí tiež veľkosť fragmentu, ak je zadaná veľkosť fragmentu väčšia ako je veľkosť správy, správa sa odošle ako jeden fragment, ak je však zadaná veľkosť fragmentu menšia ako veľkosť správy, táto správa sa rozdelí na menšie časti, ktoré sa po jednom odošlú
   - ešte pred odosielaním sa program používateľa spýta, či chce simulovať chyby, ktoré nastávajú s pravdepodobnosťou 5% pre každý fragment
   - najprv sa odošle inicializačná správa, ktorá hovorí serveru, že klient ide odosielať dáta a koľko fragmentov odošle
   - po prijatí ACK na inicializačnú správu, môže klient začať odosielať dáta
   - na každý odoslaný fragment musí prísť správa o prijatí tohto fragmentu (ACK)
   - ak príde NACK, teda doručená správa je chybná, fragment, na ktorý prišiel NACK sa odošle znova
   - po prijatí ACK na všetky fragmenty, sa ešte odošle správa o ukončení odosielania dát
   - po prijatí ACK na všetky fragmenty, bol odoslaný celý súbor
   - na strane servera sa vypíše jeho veľkosť a absolútna cesta k doručenému súboru
3. **Zobrazovanie Keep-Alive správ**
   - po zvolení tejto možnosti sa začnú do terminálu vypisovať správy na udržiavanie spojenia (Keep-Alive správy)
   - tieto správy sa budú odosielať počas celého behu programu, no nebudú vypisované, aby nerušili v termináli
   - po zvolení tejto možnosti sa začnú vypisovať až do ukončenia používateľom
4. **Nezobrazovanie Keep-Alive správ**
   - po zvolení tejto možnosti sa prestanú vypisovať Keep-Alive správy v prípade, že boli vypisované, ak neboli, nestane sa nič
5. **Zmena role**
   - po zvolení tejto možnosti si klient aj server vymenia svoje role
   - klient sa prepne na server a bude čakať na správy od klienta a server sa zmení na klienta, ktorý bude odosielať správy
6. **Ukončenie**
   - ukončenie spojenia a taktiež aj programu

Server – prijímanie dát

V prípade, že si používateľ zvolí, že chce byť serverom, program si od neho vypýta port, na ktorý chce prijímať dáta. Následne čaká na správu na nadviazanie spojenia zo strany klienta. Po prijatí tejto správy, pošle klientovi ACK a čaká na správy zo strany klienta, ktoré následne spracováva. Server má tiež svoje vlastné menu na základe ktorého si môže vybrať, čo sa bude diať. Toto menu sa zobratí vždy po stlačení tlačidla ENTER.



Na výber má z možností:

1. **Prepnutie používateľov**
    - po zadaní tejto možnosti server čaká na najbližší keep-alive paket
    - po prijatí tohto paketu, odošle naspäť klientovi správu o prepnutí používateľov
    - keď klient stlačí ENTER odošle sa ACK správa serverovi a role sa vymenia
2. **(Ne)zobrazovanie keep-alive správ**
    - po zadaní tejto možnosti sa budú vypisovať keep-alive správy na strane servera
    - ak boli správy vypisované, tak sa vypisovanie vypne
3. **Zmena adresára**
    - táto možnosť slúži na zmenu adresára, kam sa budú ukladať súbory posielané klientom
    - ak to používateľ nezmení, tak sa budú súbory ukladať do aktuálneho adresára
4. **Zobrazenie aktuálneho adresára**
    - zobrazí, ktorý adresár je uložený na ukladanie súborov
5. **Ukončenie spojenia**
    - server čaká na nasledujúci keep-alive paket, v ktorom odošle správu klientovi o ukončení spojenia, keď klient zadá ENTER odošle sa ACK správa a spojenie sa ukončí

Okrem menu, server taktiež počúva správy od klienta. Klient mu môže posielať textové správy alebo súbory.

Pri prijímaní správ a súborov, odosiela ACK, respektíve NACK, na každý doručený fragment. Po prijatí správy s dátami, skontroluje, či neboli dáta poškodené. Ak neboli, odošle sa klientovi ACK, ak neboli doručené korektne odošle sa NACK, na základe čoho bude klient vedieť, že má daný fragment odoslať znovu.

Každých 5 sekúnd dostáva od klienta Keep-Alive správy, na ktoré odpovedá odoslaním ACK správy klientovi, aby vedel, že server je stále dostupný.

Vlastná hlavička

Typ správy	Veľkosť správy	Poradie fragmentu	Checksum	Dáta

Typ správy – 1B

- určuje, o čom daná správa hovorí
- typy správ:

Typ správy [bin]	Typ správy [hex]	Význam správy
0001	1	ACK – úspešne prijatá správa
0010	2	NACK – neúspešne prijatá správa
0011	3	Inicializácia odosielania dát
0100	4	Odosielanie dát
0101	5	Ukončenie odosielania dát
0110	6	Keep-Alive správa
0111	7	Prepnutie používateľov
1000	8	Ukončenie spojenia

Veľkosť správy – 2B

- určuje veľkosť aktuálne odosielanej správy

Poradie fragmentu – 4B

- určuje poradie aktuálne odosielaného fragmentu
- pre iné ako dátové fragmenty je 0

Checksum – 4B

- kontrolná suma
- slúži na kontrolu správnosti prijatej správy

Dáta

- správa pre druhú stranu, môže obsahovať nejaké dodatočné informácie, napríklad či ide byť odosielaný súbor alebo správa, počet fragmentov, názov súboru a podobne

Veľkosť hlavičky

- 1B + 2B + 4B + 4B = **11B**

Maximálna veľkosť fragmentu

- 1500B – IP hlavička – UDP hlavička – moja hlavička
- 1500B – 8B – 20B – 11B = 1461B
- Maximálna veľkosť fragmentu, ktorú môže používateľ zadať, aby dáta neboli fragmentované na linkovej vrstve je **1461B**.

Metóda kontrolnej sumy

Na výpočet kontrolnej sumy som si zvolil metódu CRC32, ktorú implementujem z knižnice zlib.

Funkcia CRC32 je metóda na vytvorenie kontrolného súčtu pre dáta, ktorá sa často používa na overenie správnosti údajov v digitálnych komunikáciách. CRC32 vytvára 32-bitový kontrolný súčet zo vstupných dát, preto som aj v hlavičke vyhradil 4B pre checksum.

Princíp fungovania spočíva v použití polynómu s pevným počtom bitov, ktorý sa aplikuje na všetky bity vstupných dát. Dáta sú rozdelené na bloky, a pre každý blok sa vypočíta zvyšok delenia bitov polynómu. Tento zvyšok sa potom pridá k pôvodným dátam a proces sa opakuje, kým nie sú spracované všetky bloky.

Takto vytvorený checksum sa pridá do mojej hlavičky a odošle sa s dátami. Pri prijímaní sa rovnakým spôsobom vypočíta kontrolný súčet a porovná sa s prijatým súčtom. Ak sa zhodujú, dáta sa považujú za správne, inak sa predpokladá, že dáta boli poškodené v priebehu prenosu.

Kontrolnú sumu počítam zo všetkých dát, ale aj z mojej hlavičky (okrem checksumu).

ARQ metóda

ARQ sa zameriava na detekciu a opravu chýb v prenášaných dátach. V rámci tejto metódy sa odosielateľ po odoslaní každého rámca (bloku dát) dožaduje potvrdenia (ACK - Acknowledgment) od prijímateľa. Ak odosielateľ neobdrží potvrdenie v stanovenom čase alebo ak dostane potvrdenie o chybe, pošle daný rámec znovu. Tento proces sa opakuje, kým odosielateľ nedostane potvrdenie bez chýb.

V mojom programe som si zvolil ako ARQ metódu **Selective Repeat**.

Selective Repeat

Selective Repeat je špecifická varianta ARQ metódy, ktorá umožňuje odosielateľovi posielať viacero rámcov bez potvrdenia v prípade, že niektoré z nich boli stratené alebo obsahovali chyby. Pri tejto metóde má odosielateľ vyrovnávaciu pamäť (buffer), do ktorej ukladá odoslané rámce a ich kópie. Prijímateľ potvrdí prijatie každého rámca a v prípade chyby pošle žiadosť o opätovné odoslanie konkrétneho rámca.

V prípade, že prijímateľ prijme rámec bez chýb, potvrdí ho a odstráni ho z vyrovnávacej pamäte. Selective Repeat umožňuje nezávislé opakovanie len tých rámcov, ktoré boli chybné alebo sa stratili. Týmto spôsobom sa zvyšuje efektivita využitia šírky pásma.

Metóda na udržanie spojenia

Túto metódu budem uskutočňovať pomocou samostatného vlákna, ktoré sa bude starať o posielanie správ na udržanie spojenia.

Metóda na udržanie spojenia (Keep-Alive) sa uskutočňuje tak, že klient posiela keep-alive správu serveru každých 5 sekúnd a následne čaká na odpoveď od servera. Ak odpoveď od servera nedostane, znamená to, že spojenie bolo prerušené, a preto vypíše chybovú hlášku.

V prípade ak server nedostane žiadnu keep-alive správu do 60 sekúnd, došlo k prerušeniu spojenia s klientom a vypíše hlášku.

Počas posielania správ alebo súborov sú keep-alive správy vypnuté, pretože nie sú potrebné, keďže aj na základe posielaných dát vieme povedať, či je server stále dostupný. Po skončení odosielania dát je posielanie keep-alive správ obnovené.

Časti zdrojového kódu

Knižnice

V mojom programe budem využívať nasledujúce knižnice:

- **socket** – na prácu s IP adresami a portami
- **threading** – na samostatné vlákno pre keep-alive správy
- **time** – na posielanie keep-alive správ každých 5 sekúnd
- **math** – na zaokrúhľovanie
- **zlib** – implementácia crc32 metódy
- **random** – na simulovanie chyby (pravdepodobnosť chyby – 5%)
- **os** – na načítanie súboru a nájdenie cesty k nemu
- **sys** – ukončenie programu

Metódy

- metóda na resetovanie globálnych premenných (*reset_global_variables*)
- metódy na pripojenie servera a klienta (*connect_server*, *connect_client*)
- metódy na spracovanie servera a klienta (*handle_server*, *handle_client*)
- metódy na posielanie správy alebo súboru (*send_message*, *send_file*)
- metódy na prijatie správy alebo súboru (*receive_message*, *receive_file*)
- metóda zabezpečujúca zabalenie mojej hlavičky a odoslanie správy (*send*)
- metóda zabezpečujúca prijatie správy, rozbalenie hlavičky a načítanie údajov (*receive*)
- metóda na výpočet kontrolnej sumy (*checksum_calculator*)
- metóda zabezpečujúca výmenu klienta a servera (*switch_users*)
- metóda zabezpečujúca udržiavanie spojenia (*maintain_connection*)
- metóda zabezpečujúca, že server počúva na vstup od klienta (*listen_input*)

Prepínanie používateľov

Zo strany klienta

- po zvolení tejto možnosti, pošle klient správu serverovi o tom, že sa chce vymeniť
- na strane servera je potrebné stlačiť tlačidlo ENTER, aby sa resetoval input
- po stlačení tlačidla sa odošle ACK správa klientovi, že sa môžu vymeniť
- následne sa čaká 5 sekúnd, aby bolo zaručené, že sa odoslala aj posledná keep-alive správa, a role používateľov sa vymenia

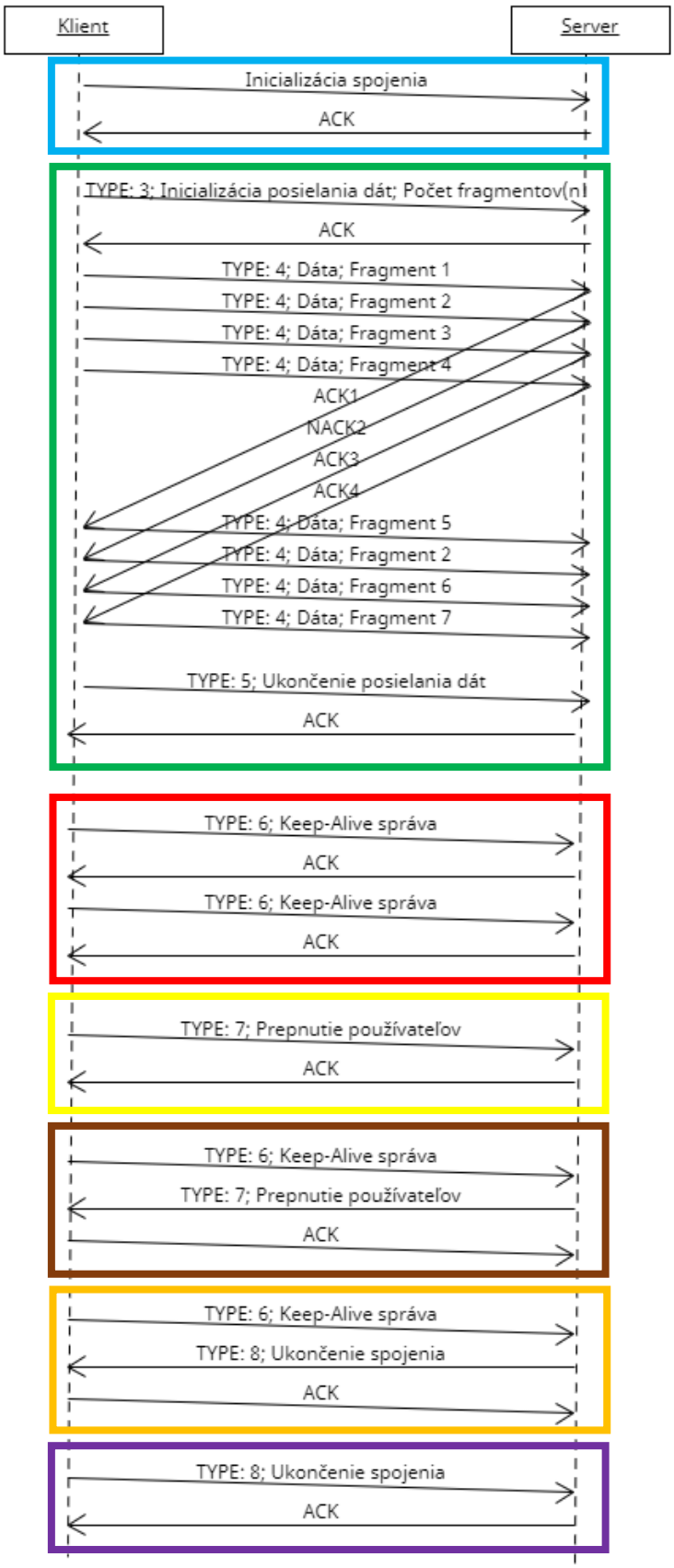
Zo strany servera

- po zvolení tejto možnosti, čaká server kým príde najbližšia keep-alive správa zo strany klienta
- na túto správu následne neposiela ACK správu, ale posiela správu o výmene používateľov
- na strane klienta je potrebné stlačiť tlačidlo ENTER, aby sa resetoval input
- po stlačení tlačidla sa odošle ACK správa serverovi, že sa môžu vymeniť
- následne sa čaká 5 sekúnd, aby bolo zaručené, že sa odoslala aj posledná keep-alive správa, a role používateľov sa vymenia

Zmeny oproti návrhu

1. **Odstránenie typu správy '7'**
   - odstránil som typ správy na informovanie servera o vypisovaní keep-alive správ
   - vo finálnej implementácií si aj klient aj server môžu sami zvoliť kedy sa budú vypisovať keep-alive správy na danom uzle
2. **Pridanie menu pre server**
   - pridal som menu pre server, ako je opísané v dokumentácií vyššie
   - server si môže zvoliť, kedy sa chce prepnúť, vypisovať keep-alive správy alebo zmeniť cestu k adresár, kam sa ukladajú súbory posielané zo strany klienta
3. **Zmena metódy kontrolnej sumy**
   - na výpočet kontrolnej sumy používam klasickú CRC32 metódu definovanú v knižnici *zlib*
4. **Zmena veľkosti hlavičky**
   - v návrhu som používal na zabalenie hlavičky knižnicu *struct*, ktorá mi z hlavičky, ktorá mala 11B, spravila hlavičku s veľkosťou 12B, čím bol 1B nevyužitý
   - vo finálnej implementácií som odstránil knižnicu *struct* z môjho projektu a hlavička už má len **11B**
   - tým sa zmenila aj maximálna veľkosť fragmentu, ktorú môže používateľ zadať na **1461B**
5. **Zmena v diagrame spracovávania**
   - odstránil som odosielanie správy na vypisovanie keep-alive správ
   - pridal som prepnutie používateľov zo strany servera
   - pridal som ukončenie spojenia zo strany servera

Diagram spracovávania



Inicializácia spojenia

Posielanie správ a súborov

Posielanie Keep-Alive správ

Prepnutie používateľov zo strany klienta

Prepnutie používateľov zo strany servera

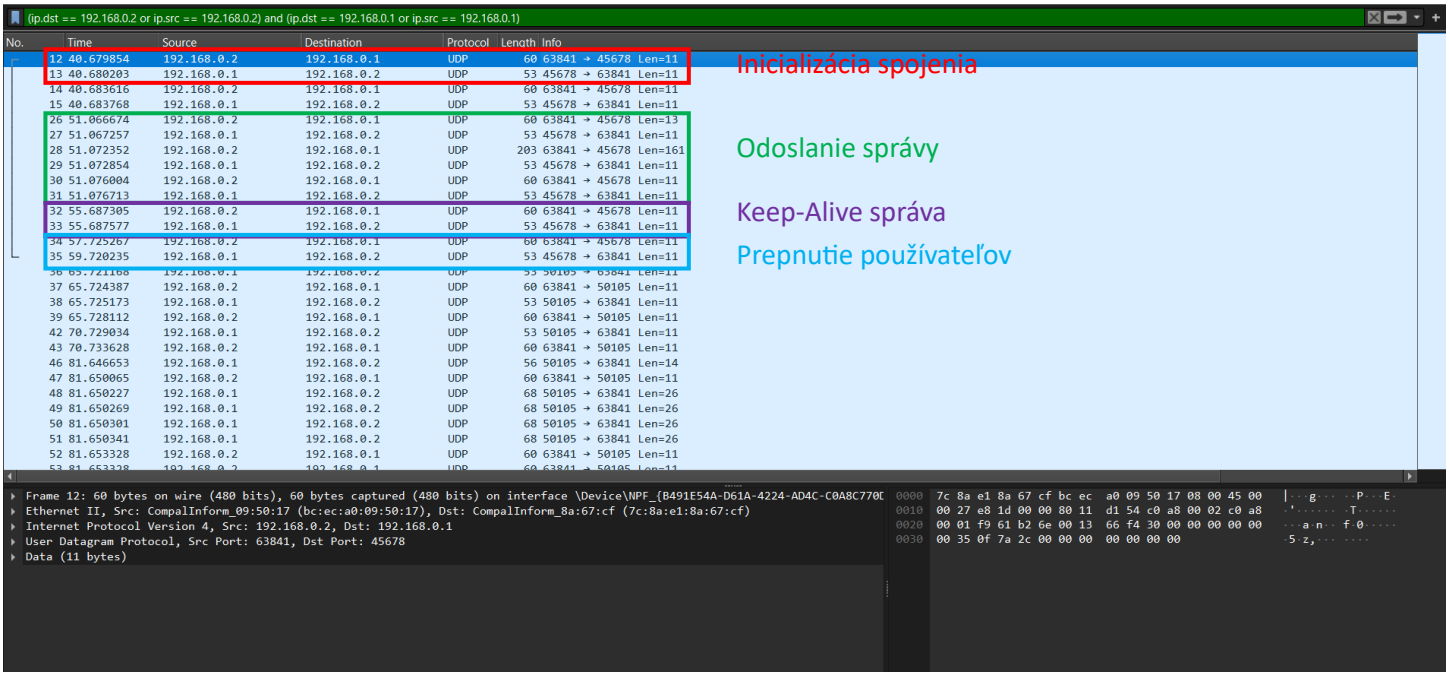
Ukončenie spojenia zo strany servera

Ukončenie spojenia zo strany klienta

Testovací scenár

Správnosť môjho riešenia ukážem na nasledujúcom testovacom scenáre:

1. spojenie používateľov
2. odoslanie správy menšej ako maximálna veľkosť fragmentu
3. prepnutie používateľov zo strany klienta
4. odoslanie správy väčšej ako maximálna veľkosť fragmentu
5. prepnutie používateľov zo strany servera
6. odoslanie súboru s veľkosťou 2MB
7. ukončenie spojenia zo strany klienta



Inicializácia spojenia

Server:

```
--------------------------------------------------------------------------------
1 - Server
2 - Client
3 - exit
--------------------------------------------------------------------------------
Choice: 1
--------------------------------------------------------------------------------
Port: 45678
192.168.0.1
[CONNECTION] Connection from address: ('192.168.0.2', 63841)
--------------------------------------------------------------------------------
```

Klient:

```
--------------------------------------------------------------------------------
1 - Server
2 - Client
3 - exit
--------------------------------------------------------------------------------
Choice: 2
--------------------------------------------------------------------------------
Server address: 192.168.0.1
Port: 45678
[CONNECTION] Connected to address: ('192.168.0.1', 45678)
--------------------------------------------------------------------------------
```

Odoslanie správy menšej ako maximálna veľkosť fragmentu

Server:

```
--------------------------------------------------------------------------------
[RECEIVING] Receiving 1 packets...
[RECEIVED] Packet number 1 was received. Size: 150B
[RECEIVE END] Receiving message ended.
[MESSAGE RECEIVED] Message: Lorem ipsum dolor sita met, consectetur adipiscing alit. Donec quis neque porttitor, maximus
 massa id, faucibus tortor. Lorem ipsum dolor sit integer.
--------------------------------------------------------------------------------
```

Klient:

```
--------------------------------------------------------------------------------
Message: Lorem ipsum dolor sita met, consectetur adipiscing alit. Donec quis neque porttitor, maximus massa id, faucibus
 tortor. Lorem ipsum dolor sit integer.
Fragment size (1B - 1461B): 200
[SENDING] Sending 1 packets...
[RESPONSE] Fragment 1 was received. Size: 150B
[END] Sending fragments ended.
--------------------------------------------------------------------------------
```

Prepnutie používateľov

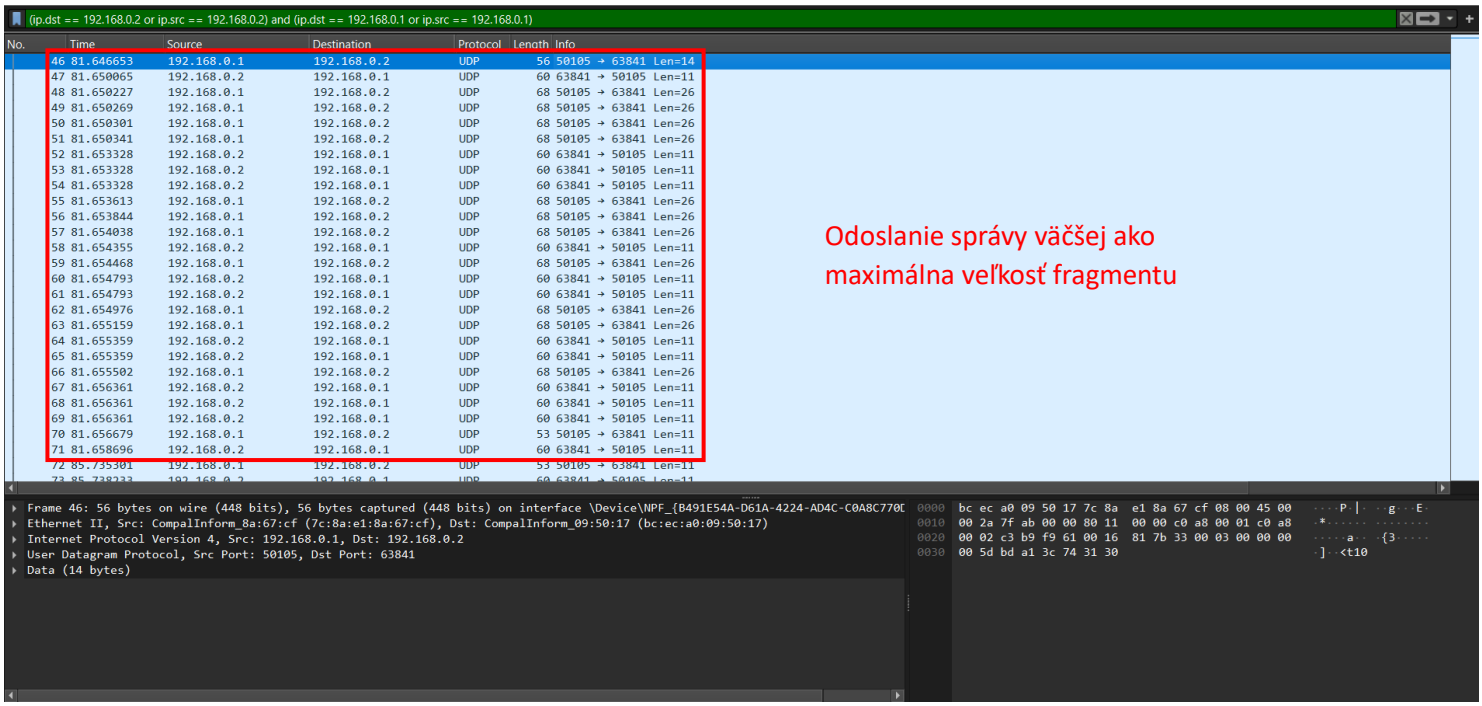
Server:

```
--------------------------------------------------------------------------------
[SWITCH] Press ENTER...

[SWITCH] Switching users.
--------------------------------------------------------------------------------
[CONNECTION] Connected to address: ('192.168.0.2', 63841)
--------------------------------------------------------------------------------
```

Klient:

```
--------------------------------------------------------------------------------
1 - Send a message
2 - Send a file
3 - Show Keep-Alive messages
4 - Don't show Keep-Alive messages
5 - Switch users
6 - End
--------------------------------------------------------------------------------
Your choice: 5
--------------------------------------------------------------------------------
192.168.0.2
[CONNECTION] Connection from address: ('192.168.0.1', 50105)
--------------------------------------------------------------------------------
```



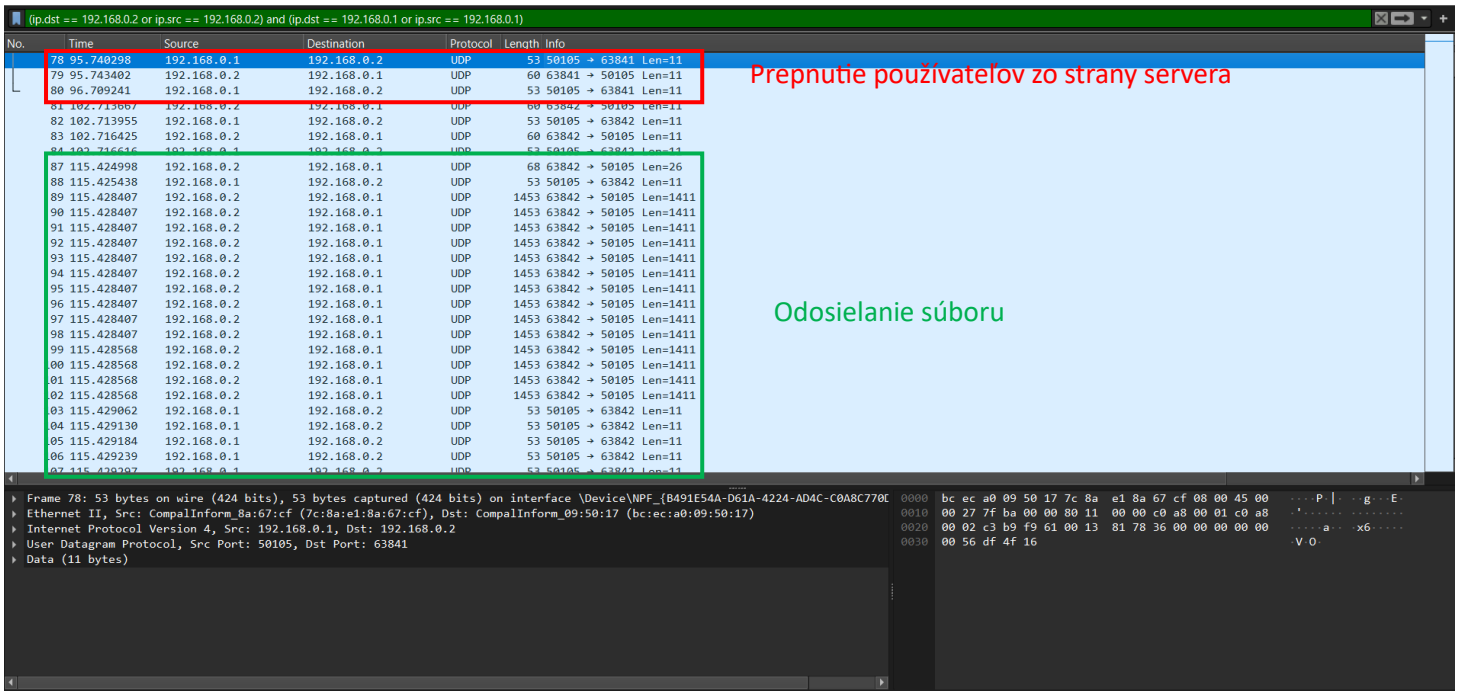
Odoslanie správy väčšej ako maximálna veľkosť fragmentu

Server:

```
--------------------------------------------------------------------------------
[RECEIVING] Receiving 10 packets...
[RECEIVED] Packet number 1 was received. Size: 15B
[RECEIVED] Packet number 2 was received. Size: 15B
[RECEIVED] Packet number 3 was received. Size: 15B
[ERROR] There was an error receiving packet 4.
[RECEIVED] Packet number 5 was received. Size: 15B
[RECEIVED] Packet number 6 was received. Size: 15B
[RECEIVED] Packet number 7 was received. Size: 15B
[RECEIVED] Packet number 4 was received. Size: 15B
[RECEIVED] Packet number 8 was received. Size: 15B
[RECEIVED] Packet number 9 was received. Size: 15B
[RECEIVED] Packet number 10 was received. Size: 15B
[RECEIVE END] Receiving message ended.
[MESSAGE RECEIVED] Message: Lorem ipsum dolor sita met, consectetur adipiscing alit. Donec quis neque porttitor, maximus
 massa id, faucibus tortor. Lorem ipsum dolor sit integer.
--------------------------------------------------------------------------------
```

Klient:

```
--------------------------------------------------------------------------------
Message: Lorem ipsum dolor sita met, consectetur adipiscing alit. Donec quis neque porttitor, maximus massa id, faucibus
 tortor. Lorem ipsum dolor sit integer.
Fragment size (1B - 1461B): 15
[SENDING] Sending 10 packets...
[RESPONSE] Fragment 1 was received. Size: 15B
[RESPONSE] Fragment 2 was received. Size: 15B
[RESPONSE] Fragment 3 was received. Size: 15B
[ERROR] Sending fragment 4 failed, sending again.
[RESPONSE] Fragment 5 was received. Size: 15B
[RESPONSE] Fragment 6 was received. Size: 15B
[RESPONSE] Fragment 7 was received. Size: 15B
[RESPONSE] Fragment 4 was received. Size: 15B
[RESPONSE] Fragment 8 was received. Size: 15B
[RESPONSE] Fragment 9 was received. Size: 15B
[RESPONSE] Fragment 10 was received. Size: 15B
[END] Sending fragments ended.
--------------------------------------------------------------------------------
```



Prepnutie používateľov zo strany servera

Server:

```
------------------------------------------------------------------------------
1 - Switch users
2 - (Don't) show Keep-Alive messages
3 - Change path to current directory
4 - Show path to current directory
5 - End
------------------------------------------------------------------------------
1
[WAITING] Waiting for next Keep-Alive packet.
[SWITCH] Switching users.
------------------------------------------------------------------------------
[CONNECTION] Connected to address: ('192.168.0.1', 50105)
------------------------------------------------------------------------------
```

Klient:

```
------------------------------------------------------------------------------
Your choice:
[SWITCH] Press ENTER...
------------------------------------------------------------------------------
192.168.0.1
[CONNECTION] Connection from address: ('192.168.0.2', 63842)
------------------------------------------------------------------------------
```

Odoslanie súboru s veľkosťou 2MB

Server:

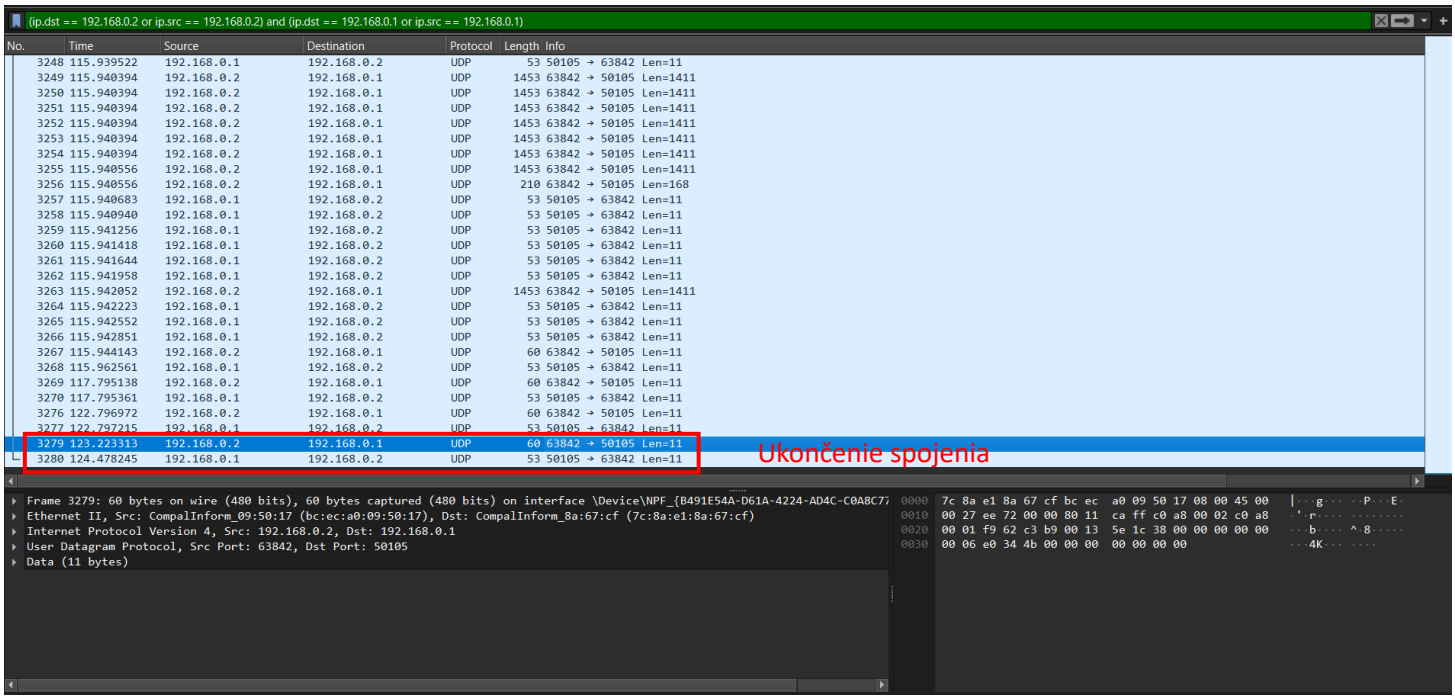
```
--------------------------------------------------------------------------------
[RECEIVING] Receiving 1504 packets...
[RECEIVED] Packet number 1 was received. Size: 1400B
[RECEIVED] Packet number 2 was received. Size: 1400B
[RECEIVED] Packet number 3 was received. Size: 1400B
[RECEIVED] Packet number 4 was received. Size: 1400B
[RECEIVED] Packet number 5 was received. Size: 1400B
[RECEIVED] Packet number 6 was received. Size: 1400B
[RECEIVED] Packet number 7 was received. Size: 1400B
[RECEIVED] Packet number 8 was received. Size: 1400B
[RECEIVED] Packet number 9 was received. Size: 1400B
```

```
[ERROR] There was an error sending packet 1497.
[RECEIVED] Packet number 1498 was received. Size: 1400B
[RECEIVED] Packet number 1499 was received. Size: 1400B
[RECEIVED] Packet number 1500 was received. Size: 1400B
[RECEIVED] Packet number 1501 was received. Size: 1400B
[RECEIVED] Packet number 1502 was received. Size: 1400B
[RECEIVED] Packet number 1503 was received. Size: 1400B
[RECEIVED] Packet number 1504 was received. Size: 157B
[RECEIVED] Packet number 1497 was received. Size: 1400B
[RECEIVE END] Receiving file ended.
[FILE PATH] Absolute path: D:\vysoka_skola\2_rocnik\1_semester\PKS\riesenia\zadanie2/photo.jpg
[FILE SIZE] File size: 2104357B
--------------------------------------------------------------------------------
```

Klient:

```
--------------------------------------------------------------------------------
Filename: photo.jpg
Fragment size (1B - 1461): 1400
[FILE PATH] Path to file: C:\Users\admin\Desktop\Tomas\photo.jpg
[FILE SIZE] File size: 2104357B
[SENDING] Sending 1504 packets...
[RESPONSE] Fragment 1 was received. Size: 1400B
[RESPONSE] Fragment 2 was received. Size: 1400B
[RESPONSE] Fragment 3 was received. Size: 1400B
[RESPONSE] Fragment 4 was received. Size: 1400B
[RESPONSE] Fragment 5 was received. Size: 1400B
[RESPONSE] Fragment 6 was received. Size: 1400B
```

```
[ERROR] Sending fragment 1497 failed, sending again.
[RESPONSE] Fragment 1498 was received. Size: 1400B
[RESPONSE] Fragment 1499 was received. Size: 1400B
[RESPONSE] Fragment 1500 was received. Size: 1400B
[RESPONSE] Fragment 1501 was received. Size: 1400B
[RESPONSE] Fragment 1502 was received. Size: 1400B
[RESPONSE] Fragment 1503 was received. Size: 1400B
[RESPONSE] Fragment 1504 was received. Size: 157B
[RESPONSE] Fragment 1497 was received. Size: 1400B
[END] Sending fragments ended.
--------------------------------------------------------------------------------
```



Ukončenie spojenia zo strany klienta

Server:

```
--------------------------------------------------------------------------------
[SWITCH] Press ENTER...

[END] Ending connection with ('192.168.0.2', 63842).
--------------------------------------------------------------------------------
```

Klient:

```
--------------------------------------------------------------------------------
1 - Send a message
2 - Send a file
3 - Show Keep-Alive messages
4 - Don't show Keep-Alive messages
5 - Switch users
6 - End
--------------------------------------------------------------------------------
Your choice: 6
--------------------------------------------------------------------------------
```

Záver

Program zabezpečuje spoľahlivý prenos textových správ a súborov medzi vysielajúcim a prijímajúcim uzlom, využívajúc pre to protokol UDP na transportnej vrstve.

Jedným z hlavných prínosov zadania je možnosť prenosu súborov aj v prípade väčších veľkostí, kde sa súbor rozkladá na menšie fragmenty. Táto funkcionalita zabezpečuje efektívne využitie siete a minimalizuje potenciálne straty dát pri prenose.

Kontrola chýb a mechanizmy znovuvyžiadania chybných fragmentov sú taktiež implementované, čo zabezpečuje spoľahlivú komunikáciu v prostredí s možnými výpadkami a stratami dát. Okrem toho, automatické odosielanie paketov pre udržanie spojenia prispieva k stabilite komunikácie, keďže vieme overiť, či je spojenie stále aktuálne.

Úspešne som dosiahol cieľ vytvoriť spoľahlivý a efektívny nástroj na komunikáciu medzi uzlami v lokálnej sieti Ethernet, využívajúci vlastný protokol nad UDP. Implementovaný program predstavuje riešenie pre prenos textových správ a súborov, ktoré môže byť ďalej vylepšované a rozširované.