

Dokumentácia
Zadanie č. 2
Počítačové a komunikačné siete
2023 – 24

Table of Contents

Zadanie	2
Programovací jazyk a prostredie	4
Návrh projektu	4
Návrh hlavičky – riadiace pakety	4
Návrh hlavičky – prenosné pakety	5
Metóda ARQ.....	5
Metóda Checksum	5
„Keep alive“ metóda	5
Diagram spracovávaní komunikácie	6
Zmeny oproti návrhu	7
Implementácia	7
Inicializácia a údržba spojenia.....	7
Posielanie správ a súborov	9
Simulácia chyby prenosu	10
Testovanie - Wireshark screeny	11
Prepínanie úloh	17
Ukončenie spojenia.....	17
Záver.....	17

Zadanie

Navrhните 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.

Program musí spĺňať nasledujúce požiadavky (minimálne):

1. Program musí byť implementovaný v jazykoch C/C++ alebo Python s využitím knižníc na prácu s UDP socket, skompilovateľný a spustiteľný v učebniach. Odporúčame použiť python modul socket, C/C++ knižnice sys/socket.h pre linux/BSD a winsock2.h pre Windows. Iné knižnice a funkcie na prácu so socketmi musia byť schválené cvičiacim. V programe môžu byť použité aj knižnice na prácu s IP adresami a portami: *arpa/inet.h* a *netinet/in.h*.
2. Program musí pracovať s dátami optimálne (napr. neukladať IP adresy do 4x int).
3. Pri posielaní súboru musí používateľovi umožniť určiť cieľovú IP a port.
4. Používateľ (stačí na strane vysielateľa) musí mať možnosť zvoliť si max. veľkosť fragmentu a meniť ju dynamicky počas behu programu pred poslaním správy/súboru (neplatí pre pakety na udržanie spojenia).
5. Obe komunikujúce strany musia byť schopné zobrazovať:
 - a) názov a absolútnu cestu k súboru na danom uzle,
 - b) veľkosť a počet fragmentov vrátane celkovej veľkosti správy/súboru.
6. Možnosť simulovať chybu prenosu odoslaním minimálne 1 chybného fragmentu pri prenose správy a súboru (do dátovej časti fragmentu alebo do checksum je cielene vnesená chyba, to znamená, že prijímajúca strana deteguje chybu pri prenose).
7. Prijímajúca strana musí byť schopná oznámiť odosielateľovi správne aj nesprávne doručenie fragmentov. Pri nesprávnom doručení fragmentu vyžiada znovu poslať poškodené dáta.
8. Možnosť odoslať 2MB súbor a v tom prípade ho uložiť na prijímacej strane ako rovnaký súbor, pričom používateľ zadáva iba cestu k adresáru kde má byť uložený.
9. Program musí byť organizovaný tak, aby oba komunikujúce uzly mohli prepínať medzi funkciou vysielateľa a prijímateľa bez reštartu programu automatizovane (jedna strana pošle správu na prepnutie, dostane ACK z druhej strany a uzly sa automaticky prepnú) - program nemusí (ale môže) byť vysielateľ a prijímateľ súčasne.

Programovací jazyk a prostredie

Svoj protokol budem programovať v programovacom jazyku Python 3.11, v programe JetBrains PyCharm 2023.2.3 s použitím knižníc socket, threading, time, crcmod, os a struct.

Návrh projektu

V mojom protokole budú dva typy paketov: riadiace a prenosné – každý typ bude mať vlastnú hlavičku podľa potrieb komunikácie. Úlohou riadiacich paketov bude sprostredkovať komunikáciu medzi uzlami na začiatku spojenia aj v priebehu, čiže medzi prenosmi dát. Prenosné pakety budú slúžiť výlučne na prenos dát. Úlohy odosielateľa a prijímateľa budú rozdelené do dvoch samostatných zdrojových kódov sender.py a receiver.py a komunikácia medzi dvoma uzlami bude bežať v main.py, kde budú vytvorené dva objekty, „uzly“, ktoré budú implementovať funkcie z oboch vedľajších súborov.

Na začiatku si používateľ zvolí či chce byť odosielateľ alebo prijímateľ. Ak si používateľ zvolí úlohu odosielateľa, program si od neho vypýta číslo portu a IP adresu prijímateľa, s ktorým sa chce spojiť a prebehne nadviazanie spojenia.

Po nadviazaní spojenia sa automaticky spustí vlákno pre udržiavanie spojenia – „keep alive“ a spustí sa aj kontrolné vlákno, cez ktoré si môže zvoliť čo chce robiť:

1. Odoslať správu
2. Odoslať súbor
3. Zmeniť úlohu na prijímateľa
4. Nerobiť nič

Tieto rozhodnutia sa budú vykonávať pomocou riadiacich paketov. Ak sa rozhodne poslať súbor alebo správu, vlákno udržiavania spojenia sa skončí a po skončení odosielania sa znovu automaticky zapne. Odosielanie dát budú realizovať prenosné pakety.

Ak sa používateľ rozhodne, že chce byť prijímateľ, program si vypýta číslo portu, na ktorom bude počúvať prichádzajúce správy. Po obdržaní riadiaceho paketu, so žiadosťou o nadviazanie spojenia automaticky odošle riadiaci paket s potvrdením a spojenie je nadviazané. Následne čaká na správu od odosielateľa.

V oboch prípadoch môže používateľ odoslať žiadosť o zmenu úlohy (odosielateľ / prijímateľ). Túto žiadosť môže samozrejme poslať iba vtedy, keď neprebieha prenos správ alebo súborov – čiže keď beží len udržiavanie spojenia. Po odoslaní žiadosti čaká na potvrdenie od druhej strany a po obdržaní potvrdenia si role prehodia.

Návrh hlavičky – riadiace pakety

Riadiace pakety budú vyzeráť nasledovne:
Žiadne ďalšie informácie pri nich nebudú potrebné.

TYP (1B)

Typy paketov budú podľa čísel označené nasledovne:

- 0) Paket pre udržiavanie spojenia – „keep alive“
- 1) Paket pre nadviazanie spojenia na začiatku komunikácie
- 2) Paket s potvrdením – „ACK“
- 3) Paket s negatívnym potvrdením – „NACK“
- 4) Paket so žiadosťou o odoslanie správy
- 5) Paket so žiadosťou o odoslanie súboru
- 6) Paket s informáciou o prijatí všetkých dát
- 7) Paket so žiadosťou o zmenu úloh
- 8) Paket so žiadosťou o koniec komunikácie

Návrh hlavičky – prenosné pakety

Prenosné pakety budú podľa môjho protokolu vyzeráť nasledovne:

Veľkosť fragmentu (2B)	Počet fragmentov (2B)	Poradie fragmentu (2B)	Dáta	CRC (2B)
------------------------	-----------------------	------------------------	------	----------

Veľkosť fragmentu – veľkosť odosielaného fragmentu

Počet fragmentov – celkový počet fragmentov správy

Poradie fragmentu – poradie odosielaného paketu

CRC – kontrolný údaj pre stanovenie správnosti prenášaných dát

Veľkosť mojej hlavičky je 8B, veľkosť UDP hlavičky je 8B a veľkosť IP hlavičky je 20B, preto maximálna veľkosť fragmentu, ktorý môžem odoslať je $65\,000 - 36 = 64\,964$ Bajtov.

Metóda ARQ

V mojom programe budem používať ARQ metódu „Go Back-N“. Pointa tejto metódy spočíva v tom, že odosielateľ rozdeľuje dáta na pevne definované bloky, nazývané okná o veľkosti N. Každý blok je odoslaný bez čakania na potvrdenie, až kým nie je celé okno naplnené. Prijímateľ potvrdzuje prijaté dáta pomocou riadiacich paketov typu 2 – „ACK“ alebo v prípade korupcie prijatých dát pomocou riadiacich paketov typu 3 – „NACK“.

V prípade, že odosielateľ dostane negatívnu odpoveď, alebo v stanovenom čase nedostane kladnú odpoveď, odosielateľ opakuje odoslanie všetkých dát od posledného potvrdeného bodu. Prijímateľ zahadzuje všetky prichádzajúce bloky okrem požadovaného bloku.

Túto metódu som si zvolil, lebo zistenie a oprava chyby prenosu paketu je pri nej pomerne jednoduchá. Jej nevýhodou však je, že v prípade chyby prenosu môže dôjsť k neefektívnemu opakovanému odosielaniu dát.

Metóda Checksum

Na kontrolu integrity prijatého paketu bude môj protokol používať CRC algoritmus. Vzhľadom na povolenie cvičiaceho budem používať 16-bitový kontrolný súčet pomocou knižnice `crcmod` s polynómom „0x8005“. Týmto polynómom sa vydedia dáta zakódované do reťazca Bajtov a zvyšok po tomto delení sa zakóduje do reťazca o veľkosti dvoch Bajtov a pridá sa na koniec reťazca dát. Takto zakódované dáta potom prijímateľ znovu vydolí stanoveným polynómom a ak je výsledok tohto delenia 0, dáta boli prenesené správne. Ak je výsledok hocikáky iný, nastala korupcia dát pri prenose.

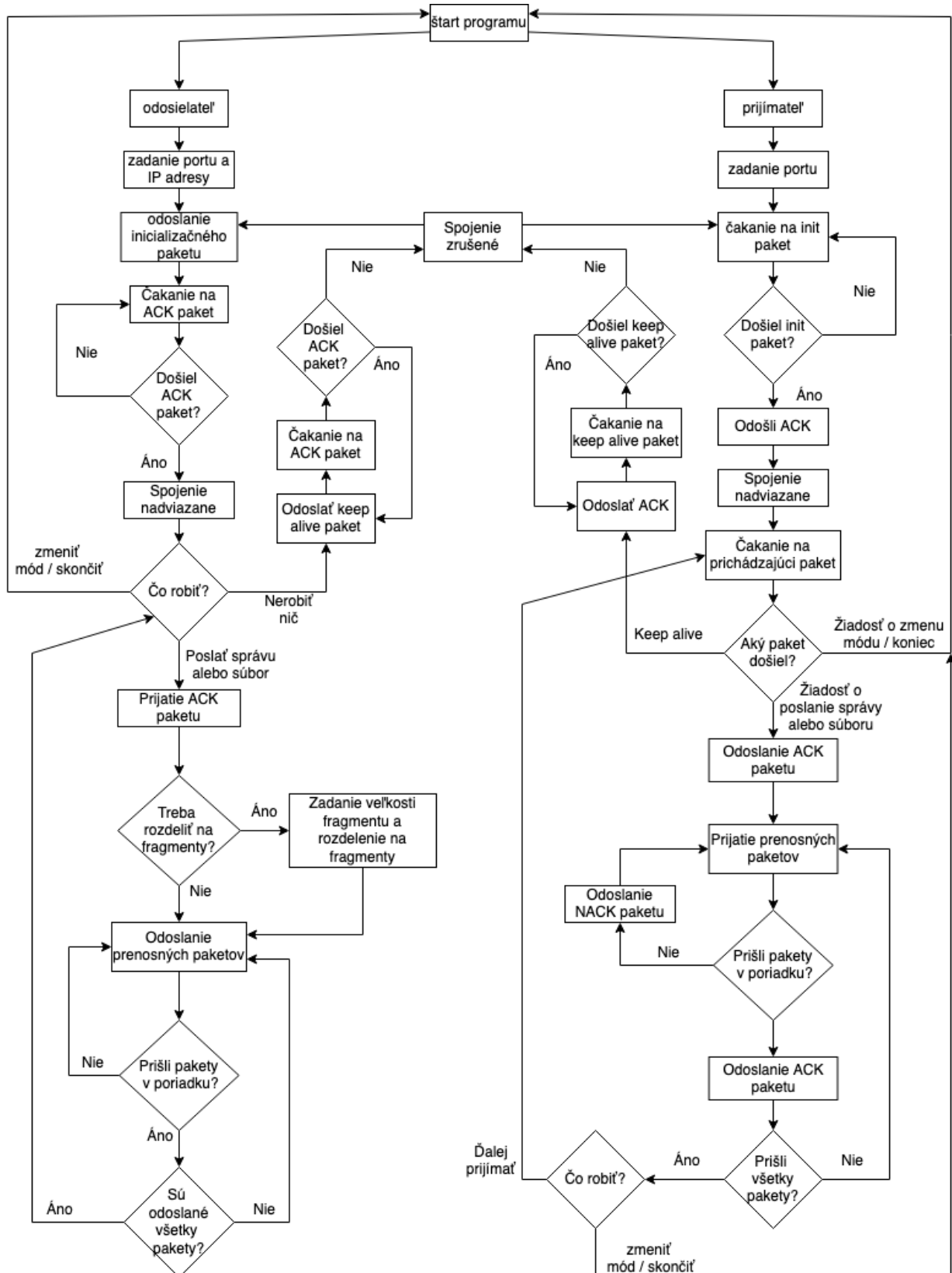
Túto metódu som si vybral pretože vie veľmi spoľahlivo zachytávať chyby v pakete a pridáva iba malé množstvo informácií k paketu.

„Keep alive“ metóda

Metóda udržania spojenia medzi komunikujúcimi uzlami vyplýva zo zadania. Odosielateľ musí každých 5 sekúnd odoslať riadiaci paket typu 0. Po obdržaní takéhoto paketu prijímateľ automaticky posiela riadiaci paket typu 2 – „ACK“. Obidva uzly majú v svojom kóde implementovaný časovač vypršania spojenia, ktorý sa úspešným prijatím

takýchto paketov zakaždým resetuje. Časovač je nastavený na 16 sekúnd. Ak časovač vyprší, spojenie sa terminuje a pre ďalšiu komunikáciu je potrebné ho znovu nadviazať.

Diagram spracovávanie komunikácie



Zmeny oproti návrhu

Oproti návrhu som zmenil metódu ARQ, z GoBack-N som prešiel na Stop and Wait. Pointa tejto metódy je, že odosielateľ pošle jednu správu a potom čaká na potvrdenie (ACK) od prijímateľa, ktorý informuje o úspešnom doručení alebo potrebe opakovania. Ak odosielateľ v stanovenom čase nedostane potvrdenie alebo dostane negatívne potvrdenie (NACK), opätovne posíla posledný paket.

Výhodou tejto metódy je jednoduchšia implementácia na oboch uzloch, nevýhodou je, že pri vyššej chybovosti sa znižuje efektivita využitia siete.

Ešte som predĺžil aj časovač na oboch stranách, pôvodných 16 sekúnd mi nestačilo a pri niektorých I/O udalostiach sa mi spojenie zavrelo aj keď som nechcel.

Implementácia

Po spustení sa program používateľa opýta, či chce byť odosielateľ alebo prijímateľ.

```
Choose role (sender/receiver): sender  
Enter port: 5050  
Enter IP address: 127.0.0.1
```

Ak si používateľ zvolí rolu odosielateľa, program si ďalej vypýta cieľovú IP adresu a port druhej strany, na ktorú sa budú dáta posílať.

```
Choose role (sender/receiver): receiver  
Enter port: 5050
```

Ak si vyberie rolu prijímateľa, program si vypýta číslo portu, na ktorom bude počúvať prichádzajúce správy.

Inicializácia a údržba spojenia

Po inicializovaní oboch strán sa spustí funkcia pre nadviazanie spojenia. Inicializácia spojenia prebieha na strane odosielateľa tak, že odosielateľ pošle druhej strane paket typu 1 – žiadosť o nadviazanie spojenia a čaká, kým mu prijímateľ pošle ACK – paket typu 2. Ak odpoveď nepríde, paket so žiadosťou sa znovu posíla a znovu sa čaká na odpoveď. Tento proces beží, kým odpoveď nepríde.

```
def establish_connection(sender_sock, DESTINATION):  
    established = False  
    while not established:  
        sender_sock.send("1".encode(FORMAT))  
        print(f"Sent message to {DESTINATION}: message type 1 - connect")  
        data, address = sender_sock.recvfrom(RECEIVE_SIZE)  
        message = data.decode()  
        if message[0] == '2':  
            print(f"Received message from {address}: message type 2 - acknowledgement")  
            print("Connection established.")  
            established = True  
        time.sleep(1)
```

Na strane prijímateľa to funguje podobne. Program čaká na inicializačnú správu od druhej strany, a keď tá správa príde posíla paket typu 2 – ACK. Ak inicializačná správa neprišla, proces sa opakuje až kým nedôjde.

```
def establish_connection(receiver_sock):
    established = False
    while not established:
        data, address = receiver_sock.recvfrom(RECEIVE_SIZE)
        message = data.decode()
        if message[0] == '1':
            print(f"Received message from {address}: message type 1 - connect")
            receiver_sock.sendto("2".encode(), address)
            print(f"Sent message to {address}: message type 2 - acknowledgement")
            established = True
        time.sleep(1)
```

Po správnom inicializovaní spojenia sa na prijímajúcom uzle spúšťa hlavná funkcia pre prijímanie správ, ktorá riadi celú komunikáciu na tejto strane. V rámci tejto funkcie prijímateľ prijíma všetky riadiace pakety vrátane paketov typu 0 – udržiavanie spojenia.

```
def receive(receiver_sock, switch_roles_event, connection_closed_event):
    receiver_sock.settimeout(TIMEOUT)
    while not switch_roles_event.is_set():
        try:
            data, address = receiver_sock.recvfrom(RECEIVE_SIZE)
            message = data.decode()
            if message[0] == '0':
                receiver_sock.settimeout(TIMEOUT)
                print(f"Received message from {address}: message type 0 - keep alive")
                receiver_sock.sendto("2".encode(), address)
                print(f"Sent message to {address}: message type 2 - acknowledgement")
            elif message[0] == '4':
                receiver_sock.settimeout(None)
                print(f"Received message from {address}: message type 4 - message send request")
                receiver_sock.sendto("2".encode(), address)
                print(f"Sent message to {address}: message type 2 - acknowledgement ")
                receive_message(receiver_sock)
                user_input(receiver_sock, switch_roles_event, connection_closed_event, address)
            elif message[0] == '5':...
            elif message[0] == '7':...
            elif message[0] == '8':...
        except socket.timeout:
            print("Connection timed out.")
            receiver_sock.close()
            connection_closed_event.set()
            break
```

Na odosielajúcom uzle sa spustí riadiace vlákno, v rámci ktorého si používateľ môže zvoliť čo sa bude diať. Používateľ môže:

1. Odoslať správu
2. Odoslať súbor
3. Vymeniť si role s druhou stranou
4. Skončiť spojenie

Kým si používateľ vyberá čo chce robiť, začne bežať aj vlákno pre udržiavanie spojenia. Toto vlákno beží až kým si používateľ nevyberie nejakú akciu.

Vlákno pre údržbu spojenia funguje tak, že sa každých 5 sekúnd posiela paket typu 0 – „keep alive“ a čaká sa na odpoveď od prijímateľa. Prijímateľ zasa čaká na pakety typu 0 a za každý jeden prijatý posiela paket typu 2 – ACK. Dobu čakania som na oboch koncoch stanovil na 16 sekúnd, čiže tri cykly vymieňania daných správ. Preto ak z ktorejkoľvek strany za sebou neprídu 3 správy, spojenie je ukončené a program o tom vypíše správu do konzoly a končí.

Na odosielajúcom uzle sa vlákno pre údržbu automaticky spúšťa vždy, keď sa neodosielajú žiadne dáta – čiže jedna zo štyroch možností uvedených vyššie, a ukončí sa automaticky keď si používateľ zvolí nejakú akciu.

Na prijímajúcom uzle je za údržbu zodpovedná hlavná funkcia pre prijímanie správ, ktorá má nastavený časovač na spomínaných 16 sekúnd a vždy keď prijme správu typu 0, časovač sa resetuje. Keď dostane hocikakú inú správu – čiže jednu zo štyroch možností používateľa na druhej strane, časovač sa vypne a spustí sa potrebná funkcia.

Posielanie správ a súborov

Keď si používateľ na strane odosielať zvoli, že chce poslať správu, vlákno pre údržbu dokončí aktuálnu iteráciu cyklu – pár sekúnd treba počkať. Potom odošle prijímateľovi paket typu 4 – žiadosť o odoslanie správy – týmto sa skončí údržba aj na prijímajúcej strane.

Následne si program vypýta od používateľa aby zadal správu, ktorú chce poslať a veľkosť fragmentov na ktoré sa má správa rozdeliť a začne po jednom fragmenty posilať. Po každom čaká na kladný ACK od druhej strany, až potom posiela ďalší fragment. Fragmenty majú hlavičku „prenosných“ paketov, čiže na začiatku je informácia o veľkosti daného fragmentu, celkovom počte všetkých fragmentov a poradovom čísle daného fragmentu. Po týchto informáciách nasledujú samotné dáta a na konci je 2 bajtová kontrolná suma vypočítaná CRC algoritmom.

```
Connection established.
If you want to send a message, type 'message'.
If you want to send a file, type 'file'.
If you want to switch roles, type 'switch'.
If you want to close the connection, type 'close'.
Sent message to ('127.0.0.1', 5050): message type 0 - keep alive
Received message from ('127.0.0.1', 5050): message type 2 - acknowledgement
message
Sent message to ('127.0.0.1', 5050): message type 4 - message send request
Received message from ('127.0.0.1', 5050): message type 2 - acknowledgement
Enter your message: Good morning Vietnam!
Enter fragment size: 2
```

Na strane prijímateľa sa následne správa naspäť poskladá a vypíše do terminálu aj s informáciou o jej veľkosti. Po úspešnom prenose dát prijímateľ posiela druhej strane paket typu 6 – dáta boli úspešne prijaté a tá mu zas odošle paket s potvrdením. Následne sa znovu spúšťa proces údržby spojenia.

```
Sent message to ('127.0.0.1', 59432): message type 2 - acknowledgement
Received fragment 8 of 11 , size: 2 Bytes
Sent message to ('127.0.0.1', 59432): message type 2 - acknowledgement
Received fragment 9 of 11 , size: 2 Bytes
Sent message to ('127.0.0.1', 59432): message type 2 - acknowledgement
Received fragment 10 of 11 , size: 2 Bytes
Sent message to ('127.0.0.1', 59432): message type 2 - acknowledgement
Received fragment 11 of 11 , size: 1 Bytes
Sent message to ('127.0.0.1', 59432): message type 2 - acknowledgement
Received message: Good morning Vietnam! , size: 21 Bytes
Sent message to ('127.0.0.1', 59432): message type 6 - data received successfully
Received message from ('127.0.0.1', 59432): message type 2 - acknowledgement
```

Posielanie súborov funguje podobne. Namiesto zadania správy si program od používateľa vypýta cestu k súboru, ktorý chce poslať a takisto sa ho opýta aj na veľkosť fragmentov.

Pri posielaní súborov sa najprv pošle meno súboru, aby druhá strana vedela pod akým názvom ho má uložiť a následne sa pomocou rovnakej Stop and Wait ARQ metódy pošlú fragmenty súboru. Prijímajúca strana si súbor ukladá do momentálneho pracovného adresára. Na konci si opäť vymenia potvrdenia o úspešnom prenose dát.

```
file
Sent message to ('127.0.0.1', 5050): message type 5 - file send request
Received message from ('127.0.0.1', 5050): message type 2 - acknowledgement
Enter the path to the file: /Users/peterbrenkus/Desktop/cat.jpg
Enter fragment size: 1400
```

Simulácia chyby prenosu

Pri každom posielaní správy alebo súboru má používateľ na strane odosielateľa navyše možnosť simulovať chybu prenosu. Program sa ho opýta, či chce simulovať chybu, a ak áno, ďalej si od neho vypýta číslo paketu, ktorý chce skorumpovať. Korupcia funguje tak, že do položky kontrolnej sumy je vnesená chyba.

```
message
Sent message to ('127.0.0.1', 5050): message type 4 - message send request
Received message from ('127.0.0.1', 5050): message type 2 - acknowledgement
Enter your message: Good morning Vientam!
Enter fragment size: 3
Do you want to introduce errors in the message? (y/n): y
Enter the number of the packet you want to corrupt: 2
Sent packet 1 of 7
Received message from ('127.0.0.1', 5050): message type 2 - acknowledgement
Sent packet 2 of 7
Received message from ('127.0.0.1', 5050): message type 3 - negative acknowledgement
Sent packet 2 of 7
Received message from ('127.0.0.1', 5050): message type 2 - acknowledgement
```

Program na strane prijímateľa túto chybu deteguje a odosiela paket typu 3 – NACK. Do konzoly tiež vypíše informáciu o prijatí chybného paketu. Keďže chyby prenosu riešim pomocou Stop and Wait ARQ, odosielateľ po každom odoslanom pakete čaká na odpoveď od servera. Po prijatí NACK paketu odosielateľ znovu posíla posledný paket.

```
Received message from ('127.0.0.1', 60226): message type 4 - message send request
Sent message to ('127.0.0.1', 60226): message type 2 - acknowledgement
Received fragment 1 of 7 , size: 3 Bytes
Sent message to ('127.0.0.1', 60226): message type 2 - acknowledgement
Received fragment 2 of 7 , size: 3 Bytes, CRC error
Sent message to ('127.0.0.1', 60226): message type 3 - negative acknowledgement
Received fragment 2 of 7 , size: 3 Bytes
Sent message to ('127.0.0.1', 60226): message type 2 - acknowledgement
Received fragment 3 of 7 , size: 3 Bytes
```

Testovanie - Wireshark screeny

11	17.867576	127.0.0.1	127.0.0.1	UDP	33	58281 → 5050 Len=1
12	17.867614	127.0.0.1	127.0.0.1	ICMP	60	Destination unreachable (Port unreachable)
13	22.888966	127.0.0.1	127.0.0.1	UDP	33	58281 → 5050 Len=1
14	22.889045	127.0.0.1	127.0.0.1	ICMP	60	Destination unreachable (Port unreachable)
15	27.900220	127.0.0.1	127.0.0.1	UDP	33	58281 → 5050 Len=1
16	27.905631	127.0.0.1	127.0.0.1	UDP	33	5050 → 58281 Len=1
17	27.940791	127.0.0.1	127.0.0.1	UDP	33	58281 → 5050 Len=1
18	28.941793	127.0.0.1	127.0.0.1	UDP	33	5050 → 58281 Len=1
19	33.961843	127.0.0.1	127.0.0.1	UDP	33	58281 → 5050 Len=1
20	33.973302	127.0.0.1	127.0.0.1	UDP	33	5050 → 58281 Len=1
21	36.988381	127.0.0.1	127.0.0.1	UDP	33	58281 → 5050 Len=1
22	36.999024	127.0.0.1	127.0.0.1	UDP	33	5050 → 58281 Len=1
23	101.601175	127.0.0.1	224.0.0.251	MDNS	259	Standard query response 0x0000 TXT, cache f
24	101.601225	fe80::1	ff02::fb	MDNS	279	Standard query response 0x0000 TXT, cache f
25	101.601264	192.168.0.24	224.0.0.251	MDNS	259	Standard query response 0x0000 TXT, cache f
26	101.601338	fe80::1c82:8a1a:15...	ff02::fb	MDNS	279	Standard query response 0x0000 TXT, cache f
27	101.601388	fe80::347d:daff:fe...	ff02::fb	MDNS	279	Standard query response 0x0000 TXT, cache f

> Frame 11: 33 bytes on wire (264 bits), 33 bytes captured (264) on interface 0				0000	02 00 00 00 45 00 00 1d ba 75 00 00 40 11 00 00E...
> Null/Loopback				0010	7f 00 00 01 7f 00 00 01 e3 a9 13 ba 00 09 fe 1c
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1				0020	31	1
> User Datagram Protocol, Src Port: 58281, Dst Port: 5050						
Data (1 byte)						
Data: 31						
[Length: 1]						

Paket č. 11 – odosielateľ sa snaží nadviazať spojenie, ale prijímateľ ešte nie je zapnutý (typ 1)

13	22.888966	127.0.0.1	127.0.0.1	UDP	33	58281 → 5050 Len=1
14	22.889045	127.0.0.1	127.0.0.1	ICMP	60	Destination unreachable (Port unreachable)
15	27.900220	127.0.0.1	127.0.0.1	UDP	33	58281 → 5050 Len=1
16	27.905631	127.0.0.1	127.0.0.1	UDP	33	5050 → 58281 Len=1
17	27.940791	127.0.0.1	127.0.0.1	UDP	33	58281 → 5050 Len=1
18	28.941793	127.0.0.1	127.0.0.1	UDP	33	5050 → 58281 Len=1
19	33.961843	127.0.0.1	127.0.0.1	UDP	33	58281 → 5050 Len=1
20	33.973302	127.0.0.1	127.0.0.1	UDP	33	5050 → 58281 Len=1
21	36.988381	127.0.0.1	127.0.0.1	UDP	33	58281 → 5050 Len=1
22	36.999024	127.0.0.1	127.0.0.1	UDP	33	5050 → 58281 Len=1
23	101.601175	127.0.0.1	224.0.0.251	MDNS	259	Standard query response 0x0000 TXT, cache f
24	101.601225	fe80::1	ff02::fb	MDNS	279	Standard query response 0x0000 TXT, cache f
25	101.601264	192.168.0.24	224.0.0.251	MDNS	259	Standard query response 0x0000 TXT, cache f
26	101.601338	fe80::1c82:8a1a:15...	ff02::fb	MDNS	279	Standard query response 0x0000 TXT, cache f
27	101.601388	fe80::347d:daff:fe...	ff02::fb	MDNS	279	Standard query response 0x0000 TXT, cache f

> Frame 13: 33 bytes on wire (264 bits), 33 bytes captured (264) on interface 0				0000	02 00 00 00 45 00 00 1d 8f 4c 00 00 40 11 00 00E...
> Null/Loopback				0010	7f 00 00 01 7f 00 00 01 e3 a9 13 ba 00 09 fe 1c
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1				0020	31	1
> User Datagram Protocol, Src Port: 58281, Dst Port: 5050						
Data (1 byte)						
Data: 31						
[Length: 1]						

Paket č. 13 – o 5 sekúnd to skúša znovu, rovnaký výsledok (typ 1)

15	27.900220	127.0.0.1	127.0.0.1	UDP	33	58281 → 5050	Len=1	
16	27.905631	127.0.0.1	127.0.0.1	UDP	33	5050 → 58281	Len=1	
17	27.940791	127.0.0.1	127.0.0.1	UDP	33	58281 → 5050	Len=1	
18	28.941793	127.0.0.1	127.0.0.1	UDP	33	5050 → 58281	Len=1	
19	33.961843	127.0.0.1	127.0.0.1	UDP	33	58281 → 5050	Len=1	
20	33.973302	127.0.0.1	127.0.0.1	UDP	33	5050 → 58281	Len=1	
21	36.988381	127.0.0.1	127.0.0.1	UDP	33	58281 → 5050	Len=1	
22	36.999024	127.0.0.1	127.0.0.1	UDP	33	5050 → 58281	Len=1	
23	101.601175	127.0.0.1	224.0.0.251	MDNS	259	Standard query response	0x0000 TXT, cache	
24	101.601225	fe80::1	ff02::fb	MDNS	279	Standard query response	0x0000 TXT, cache	
25	101.601264	192.168.0.24	224.0.0.251	MDNS	259	Standard query response	0x0000 TXT, cache	
26	101.601338	fe80::1c82:8a1a:15...	ff02::fb	MDNS	279	Standard query response	0x0000 TXT, cache	
27	101.601388	fe80::347d:daff:fe...	ff02::fb	MDNS	279	Standard query response	0x0000 TXT, cache	

> Frame 15: 33 bytes on wire (264 bits), 33 bytes captured (264	0000	02 00 00 00 45 00 00 1d	e2 d5 00 00 40 11 00 00E...
> Null/Loopback	0010	7f 00 00 01 7f 00 00 01	e3 a9 13 ba 00 09 fe 1c
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1	0020	31		1
> User Datagram Protocol, Src Port: 58281, Dst Port: 5050				
> Data (1 byte)				
Data: 31				
[Length: 1]				

Paket č. 15, tretí pokus o nadviazanie spojenia (typ 1)

16	27.905631	127.0.0.1	127.0.0.1	UDP	33	5050 → 58281	Len=1	
17	27.940791	127.0.0.1	127.0.0.1	UDP	33	58281 → 5050	Len=1	
18	28.941793	127.0.0.1	127.0.0.1	UDP	33	5050 → 58281	Len=1	
19	33.961843	127.0.0.1	127.0.0.1	UDP	33	58281 → 5050	Len=1	
20	33.973302	127.0.0.1	127.0.0.1	UDP	33	5050 → 58281	Len=1	
21	36.988381	127.0.0.1	127.0.0.1	UDP	33	58281 → 5050	Len=1	
22	36.999024	127.0.0.1	127.0.0.1	UDP	33	5050 → 58281	Len=1	
23	101.601175	127.0.0.1	224.0.0.251	MDNS	259	Standard query response	0x0000 TXT, cache	
24	101.601225	fe80::1	ff02::fb	MDNS	279	Standard query response	0x0000 TXT, cache	
25	101.601264	192.168.0.24	224.0.0.251	MDNS	259	Standard query response	0x0000 TXT, cache	
26	101.601338	fe80::1c82:8a1a:15...	ff02::fb	MDNS	279	Standard query response	0x0000 TXT, cache	
27	101.601388	fe80::347d:daff:fe...	ff02::fb	MDNS	279	Standard query response	0x0000 TXT, cache	

> Frame 16: 33 bytes on wire (264 bits), 33 bytes captured (264	0000	02 00 00 00 45 00 00 1d	ac 54 00 00 40 11 00 00E...
> Null/Loopback	0010	7f 00 00 01 7f 00 00 01	13 ba e3 a9 00 09 fe 1c
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1	0020	32		2
> User Datagram Protocol, Src Port: 5050, Dst Port: 58281				
> Data (1 byte)				
Data: 32				
[Length: 1]				

Paket č. 16 – ACK od serveru, spojenie nadviazané (typ 2)

17	27.940791	127.0.0.1	127.0.0.1	UDP	33	58281 → 5050	Len=1	
18	28.941793	127.0.0.1	127.0.0.1	UDP	33	5050 → 58281	Len=1	
19	33.961843	127.0.0.1	127.0.0.1	UDP	33	58281 → 5050	Len=1	
20	33.973302	127.0.0.1	127.0.0.1	UDP	33	5050 → 58281	Len=1	
21	36.988381	127.0.0.1	127.0.0.1	UDP	33	58281 → 5050	Len=1	
22	36.999024	127.0.0.1	127.0.0.1	UDP	33	5050 → 58281	Len=1	
23	101.601175	127.0.0.1	224.0.0.251	MDNS	259	Standard query response	0x0000 TXT, cache	
24	101.601225	fe80::1	ff02::fb	MDNS	279	Standard query response	0x0000 TXT, cache	
25	101.601264	192.168.0.24	224.0.0.251	MDNS	259	Standard query response	0x0000 TXT, cache	
26	101.601338	fe80::1c82:8a1a:15...	ff02::fb	MDNS	279	Standard query response	0x0000 TXT, cache	
27	101.601388	fe80::347d:daff:fe...	ff02::fb	MDNS	279	Standard query response	0x0000 TXT, cache	
28	102.601496	127.0.0.1	224.0.0.251	MDNS	259	Standard query response	0x0000 TXT, cache	
29	102.601718	fe80::1	ff02::fb	MDNS	279	Standard query response	0x0000 TXT, cache	
30	102.601850	192.168.0.24	224.0.0.251	MDNS	259	Standard query response	0x0000 TXT, cache	
31	102.601961	fe80::1c82:8a1a:15...	ff02::fb	MDNS	279	Standard query response	0x0000 TXT, cache	

> Frame 17: 33 bytes on wire (264 bits), 33 bytes captured (264	0000	02 00 00 00 45 00 00 1d	d4 da 00 00 40 11 00 00E...
> Null/Loopback	0010	7f 00 00 01 7f 00 00 01	e3 a9 13 ba 00 09 fe 1c
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1	0020	30		0
> User Datagram Protocol, Src Port: 58281, Dst Port: 5050				
> Data (1 byte)				
Data: 30				
[Length: 1]				

Paket č. 17 – keep alive paket od odosielateľa (typ 0)

18	28.941793	127.0.0.1	127.0.0.1	UDP	33	5050 → 58281	Len=1
19	33.961843	127.0.0.1	127.0.0.1	UDP	33	58281 → 5050	Len=1
20	33.973302	127.0.0.1	127.0.0.1	UDP	33	5050 → 58281	Len=1
21	36.988381	127.0.0.1	127.0.0.1	UDP	33	58281 → 5050	Len=1
22	36.999024	127.0.0.1	127.0.0.1	UDP	33	5050 → 58281	Len=1
23	101.601175	127.0.0.1	224.0.0.251	MDNS	259	Standard query response 0x0000 TXT, cache	
24	101.601225	fe80::1	ff02::fb	MDNS	279	Standard query response 0x0000 TXT, cache	
25	101.601264	192.168.0.24	224.0.0.251	MDNS	259	Standard query response 0x0000 TXT, cache	
26	101.601338	fe80::1c82:8a1a:15...	ff02::fb	MDNS	279	Standard query response 0x0000 TXT, cache	
27	101.601388	fe80::347d:daff:fe...	ff02::fb	MDNS	279	Standard query response 0x0000 TXT, cache	
28	102.601496	127.0.0.1	224.0.0.251	MDNS	259	Standard query response 0x0000 TXT, cache	
29	102.601718	fe80::1	ff02::fb	MDNS	279	Standard query response 0x0000 TXT, cache	
30	102.601850	192.168.0.24	224.0.0.251	MDNS	259	Standard query response 0x0000 TXT, cache	
31	102.601961	fe80::1c82:8a1a:15...	ff02::fb	MDNS	279	Standard query response 0x0000 TXT, cache	

> Frame 18: 33 bytes on wire (264 bits), 33 bytes captured (264) on interface 0				0000	02 00 00 00 45 00 00 1d	a6 2f 00 00 40 11 00 00E....
> Null/Loopback				0010	7f 00 00 01 7f 00 00 01	13 ba e3 a9 00 09 fe 1cB.....
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1				0020	32		2
> User Datagram Protocol, Src Port: 5050, Dst Port: 58281							
▼ Data (1 byte)							
Data: 32							
[Length: 1]							

Paket č. 18 – ACK od servera na keep alive (typ 2)

59	144.354833	127.0.0.1	127.0.0.1	UDP	33	60226 → 5050	Len=1
60	144.371339	127.0.0.1	127.0.0.1	UDP	33	5050 → 60226	Len=1
61	169.472855	127.0.0.1	127.0.0.1	UDP	43	60226 → 5050	Len=11
62	169.485977	127.0.0.1	127.0.0.1	UDP	33	5050 → 60226	Len=1
63	169.502798	127.0.0.1	127.0.0.1	UDP	43	60226 → 5050	Len=11
64	169.527910	127.0.0.1	127.0.0.1	UDP	33	5050 → 60226	Len=1
65	169.534112	127.0.0.1	127.0.0.1	UDP	43	60226 → 5050	Len=11

> Frame 59: 33 bytes on wire (264 bits), 33 bytes captured (264) on interface 0				0000	02 00 00 00 45 00 00 1d	21 a5 00 00 40 11 00 00E....
> Null/Loopback				0010	7f 00 00 01 7f 00 00 01	eb 42 13 ba 00 09 fe 1cB.....
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1				0020	34		4
> User Datagram Protocol, Src Port: 60226, Dst Port: 5050							
▼ Data (1 byte)							
Data: 34							
[Length: 1]							

Paket č. 59 – žiadosť o odoslanie správy (typ 4)

60	144.371339	127.0.0.1	127.0.0.1	UDP	33	5050 → 60226	Len=1
61	169.472855	127.0.0.1	127.0.0.1	UDP	43	60226 → 5050	Len=11
62	169.485977	127.0.0.1	127.0.0.1	UDP	33	5050 → 60226	Len=1
63	169.502798	127.0.0.1	127.0.0.1	UDP	43	60226 → 5050	Len=11
64	169.527910	127.0.0.1	127.0.0.1	UDP	33	5050 → 60226	Len=1
65	169.534112	127.0.0.1	127.0.0.1	UDP	43	60226 → 5050	Len=11

> Frame 60: 33 bytes on wire (264 bits), 33 bytes captured (264) on interface 0				0000	02 00 00 00 45 00 00 1d	22 11 00 00 40 11 00 00E....
> Null/Loopback				0010	7f 00 00 01 7f 00 00 01	13 ba eb 42 00 09 fe 1cB.....
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1				0020	32		2
> User Datagram Protocol, Src Port: 5050, Dst Port: 60226							
▼ Data (1 byte)							
Data: 32							
[Length: 1]							

Paket č. 60 – ACK od servera, môže sa začať posilať správa (typ 2)

61	169.472855	127.0.0.1	127.0.0.1	UDP	43	60226 → 5050	Len=11
62	169.485977	127.0.0.1	127.0.0.1	UDP	33	5050 → 60226	Len=1
63	169.502798	127.0.0.1	127.0.0.1	UDP	43	60226 → 5050	Len=11
64	169.527910	127.0.0.1	127.0.0.1	UDP	33	5050 → 60226	Len=1
65	169.534112	127.0.0.1	127.0.0.1	UDP	43	60226 → 5050	Len=11
66	169.550615	127.0.0.1	127.0.0.1	UDP	33	5050 → 60226	Len=1

> Frame 61: 43 bytes on wire (344 bits), 43 bytes captured (344) on interface 0				0000	02 00 00 00 45 00 00 27	32 e1 00 00 40 11 00 00E....
> Null/Loopback				0010	7f 00 00 01 7f 00 00 01	eb 42 13 ba 00 13 fe 26B.....
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1				0020	00 03 00 07 00 00 47 6f	6f 6b 9cGo ok..
> User Datagram Protocol, Src Port: 60226, Dst Port: 5050							
▼ Data (11 bytes)							
Data: 000300070000476f6b9c							
[Length: 11]							

Paket č. 61 – prvý fragment správy – „Goo“

62	169.485977	127.0.0.1	127.0.0.1	UDP	33	5050 → 60226	Len=1
63	169.502798	127.0.0.1	127.0.0.1	UDP	43	60226 → 5050	Len=11
64	169.527910	127.0.0.1	127.0.0.1	UDP	33	5050 → 60226	Len=1
65	169.534112	127.0.0.1	127.0.0.1	UDP	43	60226 → 5050	Len=11
66	169.550615	127.0.0.1	127.0.0.1	UDP	33	5050 → 60226	Len=1

> Frame 62: 33 bytes on wire (264 bits), 33 bytes captured (264	0000	02 00 00 00 45 00 00 1d	ec 42 00 00 40 11 00 00E...B...@...
> Null/Loopback	0010	7f 00 00 01 7f 00 00 01	13 ba eb 42 00 09 fe 1cB...&
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1	0020	32		2
> User Datagram Protocol, Src Port: 5050, Dst Port: 60226				
> Data (1 byte)				
Data: 32				
[Length: 1]				

Paket č. 62 – ACK od servera na fragment (typ 2)

63	169.502798	127.0.0.1	127.0.0.1	UDP	43	60226 → 5050	Len=11
64	169.527910	127.0.0.1	127.0.0.1	UDP	33	5050 → 60226	Len=1
65	169.534112	127.0.0.1	127.0.0.1	UDP	43	60226 → 5050	Len=11
66	169.550615	127.0.0.1	127.0.0.1	UDP	33	5050 → 60226	Len=1

> Frame 63: 43 bytes on wire (344 bits), 43 bytes captured (344	0000	02 00 00 00 45 00 00 27	dc ed 00 00 40 11 00 00E...@...
> Null/Loopback	0010	7f 00 00 01 7f 00 00 01	eb 42 13 ba 00 13 fe 26B...&
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1	0020	00 03 00 07 00 01 64 20	6d 30 30d m00
> User Datagram Protocol, Src Port: 60226, Dst Port: 5050				
> Data (11 bytes)				
Data: 00030007000164206d3030				
[Length: 11]				

Paket č. 63 – druhý fragment správy - „d m“

64	169.527910	127.0.0.1	127.0.0.1	UDP	33	5050 → 60226	Len=1
65	169.534112	127.0.0.1	127.0.0.1	UDP	43	60226 → 5050	Len=11
66	169.550615	127.0.0.1	127.0.0.1	UDP	33	5050 → 60226	Len=1

> Frame 64: 33 bytes on wire (264 bits), 33 bytes captured (264	0000	02 00 00 00 45 00 00 1d	1f bf 00 00 40 11 00 00E...@...
> Null/Loopback	0010	7f 00 00 01 7f 00 00 01	13 ba eb 42 00 09 fe 1cB...&
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1	0020	33		3
> User Datagram Protocol, Src Port: 5050, Dst Port: 60226				
> Data (1 byte)				
Data: 33				
[Length: 1]				

Paket č. 64 – NACK od servera na fragment (typ 3)

65	169.534112	127.0.0.1	127.0.0.1	UDP	43	60226 → 5050	Len=11
66	169.550615	127.0.0.1	127.0.0.1	UDP	33	5050 → 60226	Len=1
67	169.573796	127.0.0.1	127.0.0.1	UDP	43	60226 → 5050	Len=11

> Frame 65: 43 bytes on wire (344 bits), 43 bytes captured (344	0000	02 00 00 00 45 00 00 27	a7 39 00 00 40 11 00 00E...9...@...
> Null/Loopback	0010	7f 00 00 01 7f 00 00 01	eb 42 13 ba 00 13 fe 26B...&
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1	0020	00 03 00 07 00 01 64 20	6d ac d9d m...
> User Datagram Protocol, Src Port: 60226, Dst Port: 5050				
> Data (11 bytes)				
Data: 00030007000164206dacd9				
[Length: 11]				

Paket č. 65 – znovu odoslanie druhého fragmentu správy „d m“

66	169.550615	127.0.0.1	127.0.0.1	UDP	33	5050 → 60226	Len=1
67	169.573796	127.0.0.1	127.0.0.1	UDP	43	60226 → 5050	Len=11
68	169.586616	127.0.0.1	127.0.0.1	UDP	33	5050 → 60226	Len=1

> Frame 66: 33 bytes on wire (264 bits), 33 bytes captured (264	0000	02 00 00 00 45 00 00 1d	26 ef 00 00 40 11 00 00E...&...@...
> Null/Loopback	0010	7f 00 00 01 7f 00 00 01	13 ba eb 42 00 09 fe 1cB...&
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1	0020	32		2
> User Datagram Protocol, Src Port: 5050, Dst Port: 60226				
> Data (1 byte)				
Data: 32				
[Length: 1]				

Paket č. 66 - ACK od servera na fragment (typ 2)

67	169.573796	127.0.0.1	127.0.0.1	UDP	43	60226 → 5050	Len=11
68	169.586616	127.0.0.1	127.0.0.1	UDP	33	5050 → 60226	Len=1

> Frame 67: 43 bytes on wire (344 bits), 43 bytes captured (344	0000	02 00 00 00 45 00 00 27	93 a2 00 00 40 11 00 00E...@...
> Null/Loopback	0010	7f 00 00 01 7f 00 00 01	eb 42 13 ba 00 13 fe 26B...&
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1	0020	00 03 00 07 00 02 6f 72	6e 4b d5or nK.
> User Datagram Protocol, Src Port: 60226, Dst Port: 5050				
> Data (11 bytes)				
Data: 0003000700026f726e4bd5				
[Length: 11]				

Paket č. 67 – tretí fragment správy „orn“

68	169.586616	127.0.0.1	127.0.0.1	UDP	33	5050 → 60226	Len=1
69	169.609060	127.0.0.1	127.0.0.1	UDP	43	60226 → 5050	Len=11
> Frame 68: 33 bytes on wire (264 bits), 33 bytes captured (264) on interface 0: eth0							
> Null/Loopback							
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1							
> User Datagram Protocol, Src Port: 5050, Dst Port: 60226							
Data (1 byte)							
Data: 32							
[Length: 1]							

Paket č. 68 – ACK od servera na fragment (typ 2)

69	169.609060	127.0.0.1	127.0.0.1	UDP	43	60226 → 5050	Len=11
70	169.614749	127.0.0.1	127.0.0.1	UDP	33	5050 → 60226	Len=1
> Frame 69: 43 bytes on wire (344 bits), 43 bytes captured (344) on interface 0: eth0							
> Null/Loopback							
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1							
> User Datagram Protocol, Src Port: 60226, Dst Port: 5050							
Data (11 bytes)							
Data: 000300070003696e6770fc							
[Length: 11]							

Paket č. 69 – ďalší fragment správy „ing“

70	169.614749	127.0.0.1	127.0.0.1	UDP	33	5050 → 60226	Len=1
71	169.658376	127.0.0.1	127.0.0.1	UDP	43	60226 → 5050	Len=11
> Frame 70: 33 bytes on wire (264 bits), 33 bytes captured (264) on interface 0: eth0							
> Null/Loopback							
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1							
> User Datagram Protocol, Src Port: 5050, Dst Port: 60226							
Data (1 byte)							
Data: 32							
[Length: 1]							

Paket č. 70 - ACK od servera na fragment (typ 2)

71	169.658376	127.0.0.1	127.0.0.1	UDP	43	60226 → 5050	Len=11
72	169.671410	127.0.0.1	127.0.0.1	UDP	33	5050 → 60226	Len=1
> Frame 71: 43 bytes on wire (344 bits), 43 bytes captured (344) on interface 0: eth0							
> Null/Loopback							
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1							
> User Datagram Protocol, Src Port: 60226, Dst Port: 5050							
Data (11 bytes)							
Data: 000300070004205669d6be							
[Length: 11]							

Paket č. 71 – ďalší fragment správy „Vi“

72	169.671410	127.0.0.1	127.0.0.1	UDP	33	5050 → 60226	Len=1
73	169.683681	127.0.0.1	127.0.0.1	UDP	43	60226 → 5050	Len=11
> Frame 72: 33 bytes on wire (264 bits), 33 bytes captured (264) on interface 0: eth0							
> Null/Loopback							
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1							
> User Datagram Protocol, Src Port: 5050, Dst Port: 60226							
Data (1 byte)							
Data: 32							
[Length: 1]							

Paket č. 72 - ACK od servera na fragment (typ 2)

73	169.683681	127.0.0.1	127.0.0.1	UDP	43	60226 → 5050	Len=11
74	169.705334	127.0.0.1	127.0.0.1	UDP	33	5050 → 60226	Len=1
> Frame 73: 43 bytes on wire (344 bits), 43 bytes captured (344) on interface 0: eth0							
> Null/Loopback							
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1							
> User Datagram Protocol, Src Port: 60226, Dst Port: 5050							
Data (11 bytes)							
Data: 000300070005656e74367d							
[Length: 11]							

Paket č. 73 - ďalší fragment správy „ent“

74	169.705334	127.0.0.1	127.0.0.1	UDP	33	5050 → 60226	Len=1
75	169.721583	127.0.0.1	127.0.0.1	UDP	43	60226 → 5050	Len=11
> Frame 74: 33 bytes on wire (264 bits), 33 bytes captured (264) on interface 0: eth0							
> Null/Loopback							
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1							
> User Datagram Protocol, Src Port: 5050, Dst Port: 60226							
Data (1 byte)							
Data: 32							
[Length: 1]							

Paket č. 74 - ACK od servera na fragment (typ 2)

75	169.721583	127.0.0.1	127.0.0.1	UDP	43	60226 → 5050	Len=11
76	169.749985	127.0.0.1	127.0.0.1	UDP	33	5050 → 60226	Len=1
> Frame 75: 43 bytes on wire (344 bits), 43 bytes captured (344				0000	02 00 00 00 45 00 00 27	7b e2 00 00 40 11 00 00E...@...
> Null/Loopback				0010	7f 00 00 01 7f 00 00 01	eb 42 13 ba 00 13 fe 26B.....&
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1				0020	00 03 00 07 00 06 61 6d	21 7c fcam! .
> User Datagram Protocol, Src Port: 60226, Dst Port: 5050							
Data (11 bytes)							
Data: 000300070006616d217cfc							
[Length: 11]							

Paket č. 75 - ďalší fragment správy „am!“

```
Received message: Good morning Vientam! , size: 21 Bytes
Sent message to ('127.0.0.1', 60226): message type 6 - data received successfully
Received message from ('127.0.0.1', 60226): message type 2 - acknowledgement
```

Výpis rekonštruovanej správy spolu s jej veľkosťou v konzole servera

76	169.749985	127.0.0.1	127.0.0.1	UDP	33	5050 → 60226	Len=1
77	171.811827	127.0.0.1	127.0.0.1	UDP	33	5050 → 60226	Len=1
> Frame 76: 33 bytes on wire (264 bits), 33 bytes captured (264				0000	02 00 00 00 45 00 00 1d	a0 a1 00 00 40 11 00 00E...@...
> Null/Loopback				0010	7f 00 00 01 7f 00 00 01	13 ba eb 42 00 09 fe 1cB.....
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1				0020	32		2
> User Datagram Protocol, Src Port: 5050, Dst Port: 60226							
Data (1 byte)							
Data: 32							
[Length: 1]							

Paket č. 76 - ACK od servera na fragment (typ 2)

77	171.811827	127.0.0.1	127.0.0.1	UDP	33	5050 → 60226	Len=1
78	171.828016	127.0.0.1	127.0.0.1	UDP	33	60226 → 5050	Len=1
> Frame 77: 33 bytes on wire (264 bits), 33 bytes captured (264				0000	02 00 00 00 45 00 00 1d	bf ac 00 00 40 11 00 00E...@...
> Null/Loopback				0010	7f 00 00 01 7f 00 00 01	13 ba eb 42 00 09 fe 1cB.....
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1				0020	36		6
> User Datagram Protocol, Src Port: 5050, Dst Port: 60226							
Data (1 byte)							
Data: 36							
[Length: 1]							

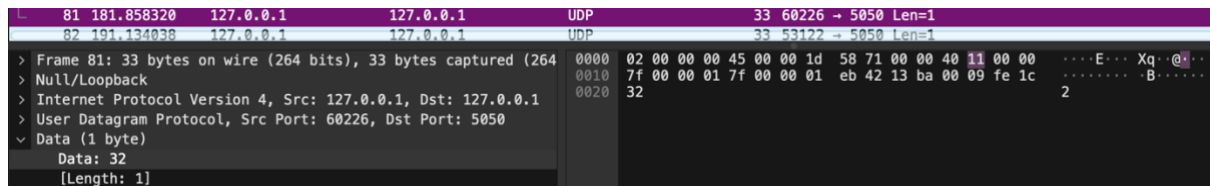
Paket č. 77 – Správa bola doručená úspešne (typ 6)

78	171.828016	127.0.0.1	127.0.0.1	UDP	33	60226 → 5050	Len=1
79	174.812562	127.0.0.1	127.0.0.1	UDP	33	5050 → 60226	Len=1
> Frame 78: 33 bytes on wire (264 bits), 33 bytes captured (264				0000	02 00 00 00 45 00 00 1d	70 f9 00 00 40 11 00 00E...p...@...
> Null/Loopback				0010	7f 00 00 01 7f 00 00 01	eb 42 13 ba 00 09 fe 1cB.....
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1				0020	32		2
> User Datagram Protocol, Src Port: 60226, Dst Port: 5050							
Data (1 byte)							
Data: 32							
[Length: 1]							

Paket č. 78 - ACK od odosielateľa na predošlú správu (typ 2)

79	174.812562	127.0.0.1	127.0.0.1	UDP	33	5050 → 60226	Len=1
80	181.845945	127.0.0.1	127.0.0.1	UDP	33	60226 → 5050	Len=1
> Frame 79: 33 bytes on wire (264 bits), 33 bytes captured (264				0000	02 00 00 00 45 00 00 1d	8d 8c 00 00 40 11 00 00E...@...
> Null/Loopback				0010	7f 00 00 01 7f 00 00 01	13 ba eb 42 00 09 fe 1cB.....
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1				0020	37		7
> User Datagram Protocol, Src Port: 5050, Dst Port: 60226							
Data (1 byte)							
Data: 37							
[Length: 1]							

Paket č. 79 – žiadosť od servera o výmenu rolí (typ 7)



No.	Time	Source	Destination	Protocol	Length	Info
81	181.858320	127.0.0.1	127.0.0.1	UDP	33	60226 → 5050 Len=1
82	191.134838	127.0.0.1	127.0.0.1	UDP	33	53122 → 5050 Len=1

Frame 81: 33 bytes on wire (264 bits), 33 bytes captured (264) on interface 0	0000	02 00 00 00 45 00 00 1d 58 71 00 00 40 11 00 00	...	E... Xq...@...
> Null/Loopback	0010	7f 00 00 01 7f 00 00 01 eb 42 13 ba 00 09 fe 1c	B.....
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1	0020	32		2
> User Datagram Protocol, Src Port: 60226, Dst Port: 5050				
> Data (1 byte)				
Data: 32				
[Length: 1]				

Paket č. 81 – ACK od odosielateľa na predošlú správu (typ 2)

Prepínanie úloh

Vymieňanie úloh („switch“) je v mojom programe možné uskutočniť z oboch strán bez reštartovania programu. Jedna strana odošle druhej paket typu 7 – žiadosť o výmenu úloh a druhá odošle naspäť ACK. Oba uzly následne zavrú svoje sockety, prepnú svoj mód a čakajú na zinicializovanie nového spojenia.

Odosielateľ má možnosť vybrať si zmenu úloh vždy keď zrovna neposiela žiadnu správu alebo súbor. Prijímateľ má možnosť vybrať si zmenu úloh po prijatí každej správy alebo súboru.

Ukončenie spojenia

Odpojenie a korektné vypnutie programu („disconnect“) je v mojom programe možné uskutočniť z oboch strán. Jedna strana odošle druhej paket typu 8 – žiadosť o ukončenie spojenia a druhá odošle naspäť ACK. Odosielateľ zavrie svoj socket a ukončí svoj program. Prijímateľ má možnosť zvoliť si to isté alebo zostať počúvať.

Odosielateľ má možnosť vybrať si ukončenie spojenia vždy keď zrovna neposiela žiadnu správu alebo súbor. Prijímateľ má možnosť vybrať si ukončenie spojenia po prijatí každej správy alebo súboru.

Záver

Môj program prenáša dáta po sieti pomocou UDP protokolu a zvyšuje jeho spoľahlivosť. Program sa skladá z dvoch častí, prijímacej a odosielacej. Po úspešnom pripojení si môžu užívatelia medzi sebou vymeniť role bez toho aby sa spojenie prerušilo.

Program vie posielať textové správy aj súbory. Pri ich posielaní si od užívateľa vypýta veľkosť fragmentov, na ktoré sa dáta majú rozdeliť, následne dáta rozdelí, pridá k nim hlavičky a posiela po sieti.

Program vie na druhej strane fragmenty prijať, overiť ich integritu, ak sú poškodené, znovu ich vypýtať a celú správu naspäť zložiť. Testovanie v dokumentácii prebehlo na jednom počítači, testoval som aj so spolužiakom na dvoch počítačoch cez Ethernet kábel a LAN sieť. Posielal som textové správy aj obrázky – najväčší mal 4,6 MB.

Možným vylepšením môjho programu by bola implementácia komplexnejšej ARQ metódy. Stop and Wait je síce jednoduchá na implementáciu, ale nie veľmi efektívna pri väčšom počte chýb prenosu.