Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Počítačové a komunikačné siete

Zadanie č.2

Akademický rok 2022/2023

Meno: Rodion Burmistrov
Cvičiaci: Ing. Miroslav Bahleda, PhD. **Dátum:** 7.12.2022

Obsah

1 Opis problematiky a zadania

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.

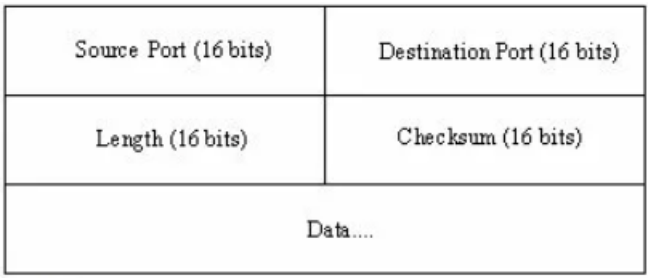
Program musí obsahovať kontrolu chýb pri komunikácii a znovuvyžiadanie chybných fragmentov, vrátane pozitívneho aj negatívneho potvrdenia. Po zapnutí programu, komunikátor automaticky odosiela paket pre udržanie spojenia každých 5s pokiaľ používateľ neukončí spojenie ručne. Odporúčame riešiť cez vlastne definované signalizačné správy a samostatný thread.

2 Navrhnuté riešenie

2.1 Hlavička

Kvôli tomu, že riešenie úlohy spočíva nad UDP protokolom – chcel by som doplniť do klasickej UDP hlavičky niekoľko veci, ktoré mi pomôžu pri riešení problému.

Pri navrhoveni svojej hlavičky potrebné vedieť, že mame hornú hranicu posielaneho fajlu 2MB, čo je 2097152 Bajtov. Tým pádom budeme musieť fragmentovat posielany súbor. Kvôli tomu, že komunikácia priebeha v lokálnej sieti Ethernet – veľkosť každého fragmentu nemôže byt viac ako 1500B. 1500B - 20B (IP hlavička) – 8B (UDP hlavička) = 1474B pre dáta a pre veci, ktoré pridám ako svoju vlastnú hlavičku.



Klasická UDP hlavička ma Source Port (2B), Destination Port (2B) DĺžkaUDP paketu (2B) a Checksum pre poslaní paket (2B).

Vo svojej hlavičke chcel by som okrem vyššie uvedeného použiť flagi (Pre flag je v hlavičke uhradeny 1B), ktoré budú inšpirované TCP protokolom, konkrétne to budú flagi:

SYN – Flag na vytvorenie spojenia.
FIN – Flag na ukončenie spojenia.
ACK – Flag na kvitanciu pre potvrdenie prijatia packet bez chyb.
NACK – Flag na potvrdenie prijatia packeta s chybou.
CHA – Flag na žiadosť o výmene roli (vysielanie a prijímanie).
DAT – Flag pre prenos paketu.
URG – Flag podstatnú správu.

Ďalej čo pladujem vo svojej hlavičke mať je index (alebo celkove číslo) paketu, ktoré bude mať 3B (kvôli fragmentácii na maximálnu veľkosť 2MB). Posledne čo bude v mojej navrhnutej hlavičke je CRC (kontrolná summa) pre prenesene dáta, ta bude mať 4B.

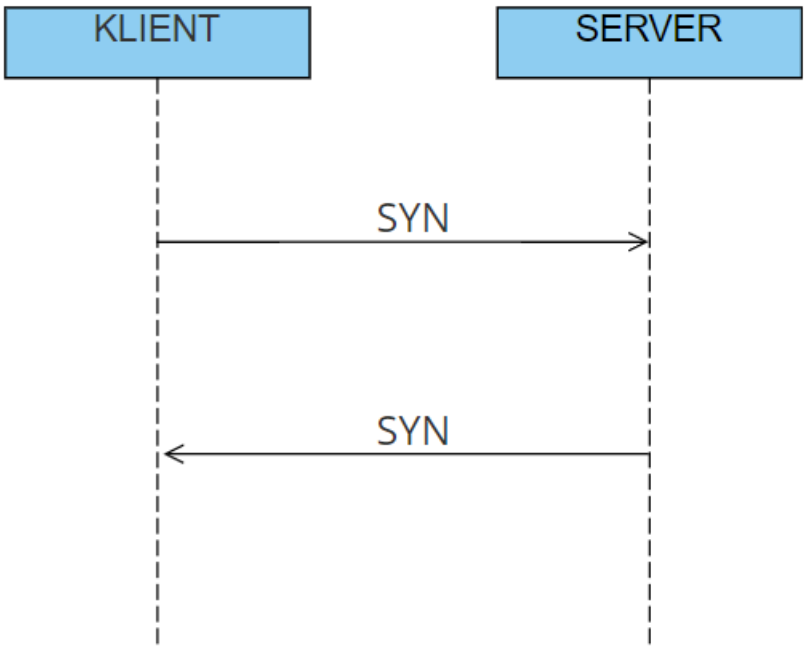
20B	IP Hlavička
8B	UDP Hlavička
1B	Flag
3B	Index paketu
4B	Kontrolna summa
0-1464 B	DATA

Tým pádom, packet bude schopný poslať 1464B dat.

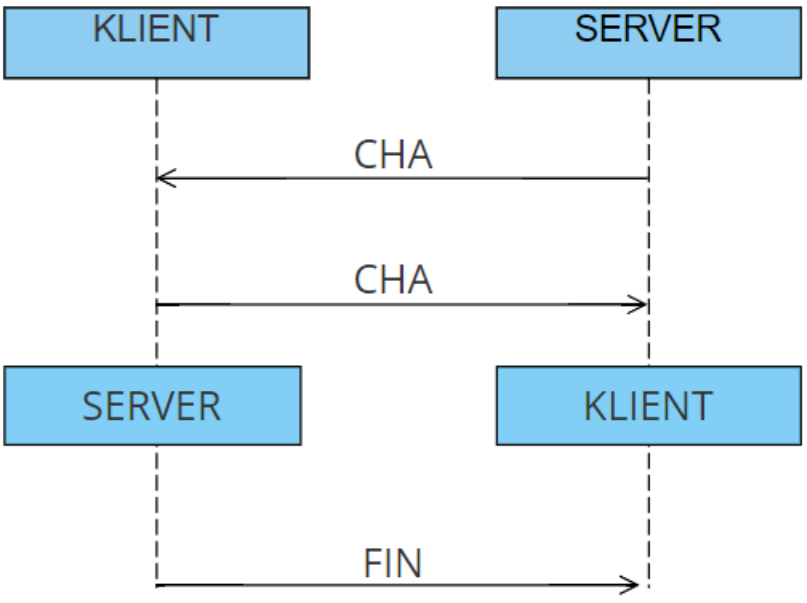
2.2 Princip fungovania

Pri štarte programu používateľ najprv zvoli, že či ide byt vysielač (klient), alebo prijímač (server). Pri nastavovanii portov a IP komunikáciu sa bude dať realizovať.

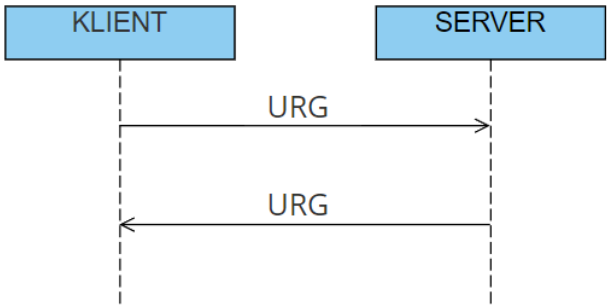
Najprv bude treba spojiť klienta a prijímača pomocou zaslania správy s SYN flagom zo strany klienta. Pri úspešnom prijatí Server si uloží údaje o Klientovi a odošle spätnú správu SYN. Ak všetko priebehne bez chýb – spojenie bude úspešne a môžeme posielať súbor.



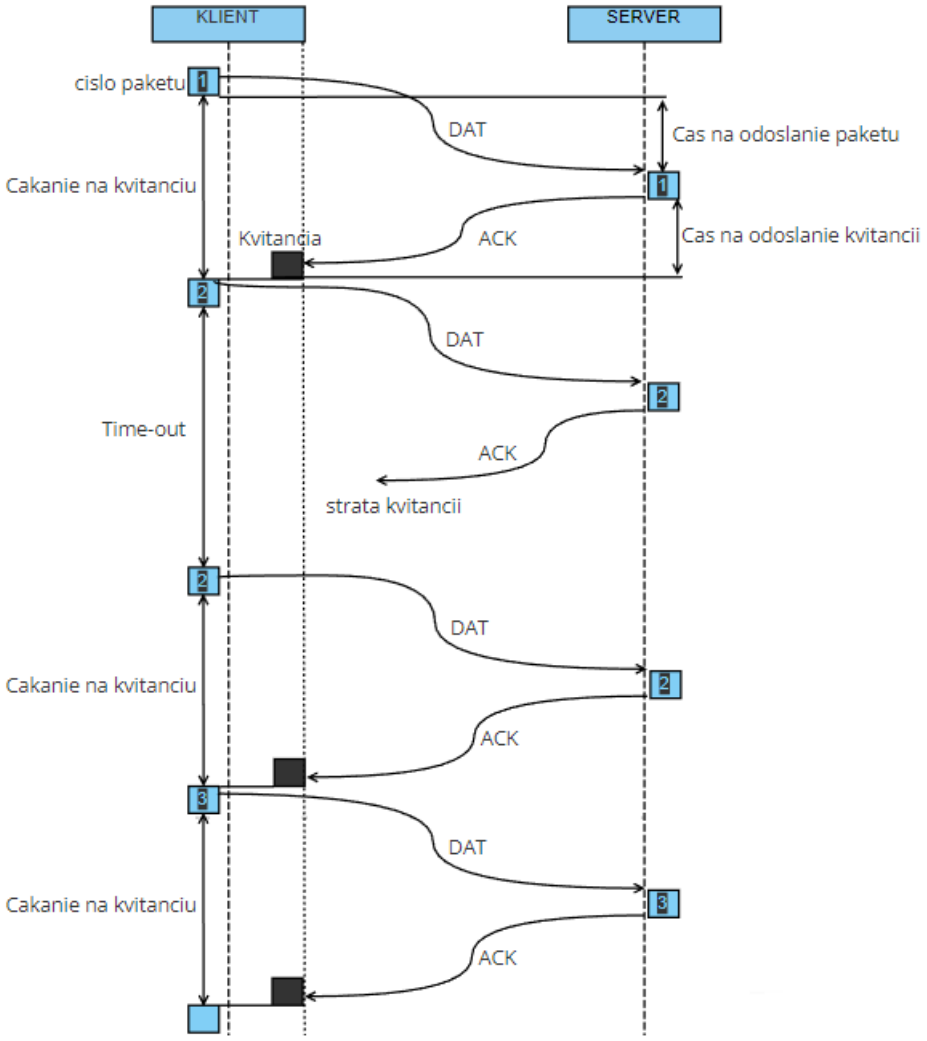
Po úspešnom spojení budeme si môcť zvoliť niekoľko možnosti. Server bude mať len možnosti, ktorý bude mať aj Klient – či budem chcieť vykonat výmenu roli, pomocou odoslaniu flagu CHA, alebo či ukončiť spojenie, pomocou flagu FIN



Klient si ale bude môcť zvoliť či chce preniesť súbor, alebo správu. Ak to bude podstatný výber v mojom programe – použijem flag URG, v ktorom budú zasifrovane informácie, ktoré Server bude chcieť vedieť, aby prijať zvolený súbor. Spätne poslanie flagu URG bude znamenať že používateľ môže pokračovať v poslaní súboru Serveroví.



Prenos paketov plánujem robiť pomocou ARQ metódy a konkrétne Stop-and-Wait. V tejto metóde klient posiela poradie paketov a metóda potrebuje, aby odosielac sa dočkal na kvitanciu od servera a len potom posielal ďalší paket. Pri posielali paketu sa štartuje timer, ak po uplynutí času timeru kvitancia nedôjde – paket sa berie ako stratený, alebo pokazený a jeho posielanie sa opakuje. Pri tejto metóde je jasne, že v prípade straty kvitancii, server môže dostať 2 rovnaké pakety. Preto musí vedieť si ich rozlisit a nepotrebný vyhodiť. Ak, ale prvá kvitancia nebola stratená a len dlho šla – nastane kolízia a klientovi prídu 2 rovnaké kvitancie, klient môže tuto kópiu rozlíšiť ako kvitanciu na ďalší paket. Preto klient tiež musí vedieť si rozlíšiť dublikaty kvitancij. Pri zvyčajnom UDP tento problém by vedelo vyriešiť pridanie špeciálneho bitu v hlavičke, ale vo svojom riešení preto použijem index paketu pri posielani paketu serveru a pri získaní kvitancii.



Ako metódu pre kontrolu či dáta prišli nepoškodené som si zvolil metódu crc32, ktorá bude reprezentovana knižnicou zlib. Kvôli svojej komplexnosti dokáže zistiť viac chýb. CRC pracuje s dátami po jednotlivých bitoch. Pomocou svojho polynomu a algoritmu, založenom na XOR operáciách s hodnotou 0xFFFFFFFF vypočíta kontrolný súčet dať, ktorý dostal server a sa porovná s tými, ktoré boli v hlavičke. Polynom v metóde crc32 je daný:

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

Alebo binárnym číslom:

1 0000 0100 1100 0001 0001 1101 1011 0111

Vybraný mnou algoritmus funguje nasledovne:

1. Vstup prevediem do dvojkovej sústavy a zrkadlovo obrátim.
2. Pridám ku koncu (vpravo) 32 nuly
3. XOR s hodnotou 0xFFFFFFFF (1111 1111 1111 1111 1111 1111 1111 1111 0000 0000)
4. Ďalej XORujem polynom s medzivysledkom v prípade ak prvý bit je 1
   Ak nie je jedna, ale nula – tak sa posune m vpravo
5. Opakujem, kým prvých 8 bitov nebude nulových
6. XOR s hodnotou 0xFFFFFFFF (1111 1111 1111 1111 1111 1111 1111 1111 0000 0000)
7. Obrátim výsledok zrkadlovo
8. Výsledok je kontrolný súčet

Pri riešení plánujem použiť knižnice:

Socket – pre prácu s paketami a ich prenosom
Zlib – pre kontrolu dať
Time – timer pre ARQ metódu
Struct – pre vytvorenie bajtového objektu, jeho balenie a rozbalenie

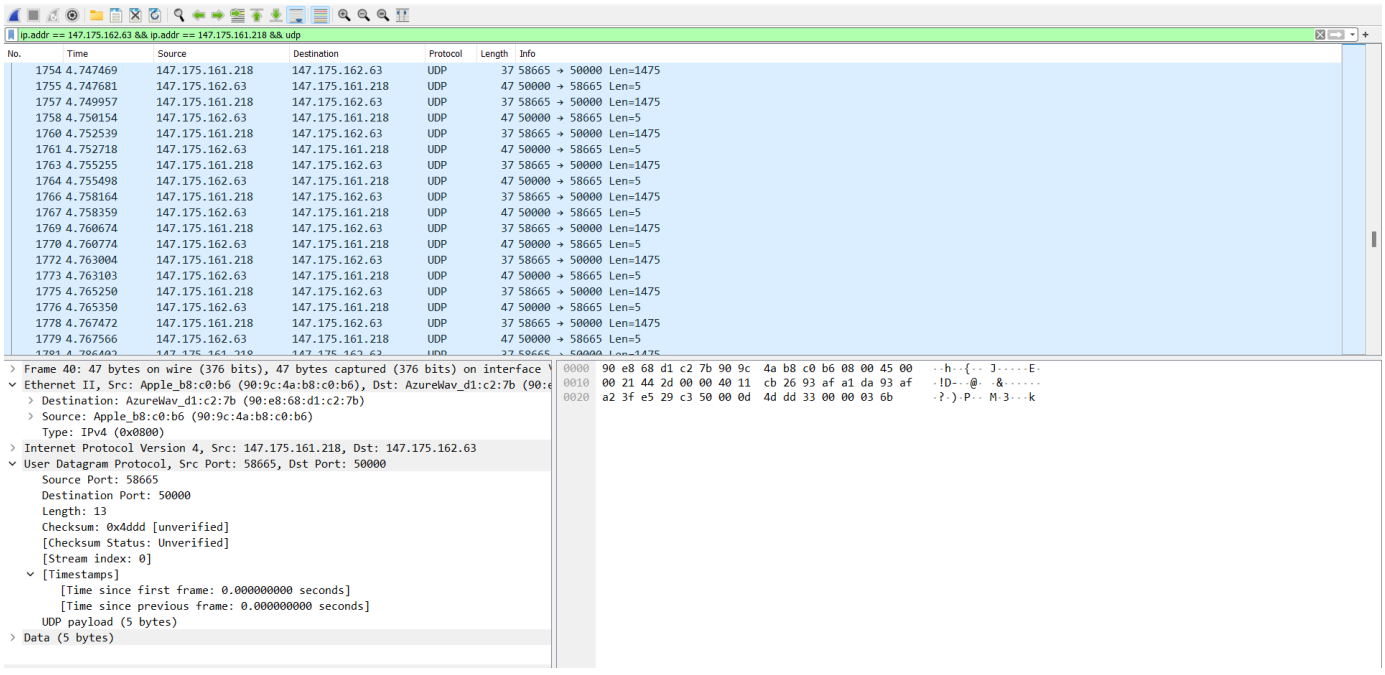
3 Vysledne riešenie

3.1 Podstatne zmeny

Vo vyslednom rieseni som sa drzal vsetkeho, co som mal napisane v navrhu riesenia. Flagom som pridelil svoje specificke simvoly, ktore sa jednoducho zvladli poslat medzi dvoma zariadeniami. Bola mnou zmenena ARQ metoda podla mojej potreby. Vynechal som casovac, po uplynuti ktoreho packet sa mal poslat znovu, ak je pokazeny. Nakoniec ale bol mnou nahradeny pridanim ineho flagu, ktory ukazoval fakt toho, ze prijaty paket sa pokazil po ceste. Okem vyssie uvedeneho zobral som Serverovi chocijake moznosti, ako napriklad ziadanie o zmenu roli. Server len si moze volti kam ulozi subor, ktory dostane od klienta.

3.2 Wireshark

Pri skusani svojho programmu som pouzil 2 notebooka. Jeden funguje na MacOS, druhy na Windows. Na screenshote je uvedena komunikacia medzi tymito pocitacmi. Konkretne na screenshote priebeha posielanie fragmentov fajlu. Klient (Windows) posiela fragmenty na server (MacOS), a ten spatne posiela flag ze dostal spravny fragment.



3.2 Screenshoty

Na nizsie uvedenych screenshotoch je strucny popis programu. A podstatnych jeho casti.

```python
def main():  # Len menu, kde pouzivatel si zvoli rolu
    print("1 for client")
    print("2 for server")
    print("3 to exit")
    choice = input()
    if choice == '1':
        client_login()
    elif choice == '2':
        server_login()
    else:
        print("Try to input something different")


main()
```

```python
def server_login():  # Login pre server
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)  # Inet - ETHERNET    DGRAM - UDP
    ip = socket.gethostbyname(socket.gethostname())  # Dostanem IP servera do premennej
    print("Zadajte PORT servera")
    port = int(input())  # Zadam svoj port
    server_socket.bind((ip, port))  # Bind

    print('Adresa servera je ', ip)

    while True:
        data, client_address = server_socket.recvfrom(1500)  # Dostane flag a prejde do menu
        if data == b'1':
            server_socket.sendto(b'1', client_address)  # Odosle flag, ze flag je prijaty
            server(server_socket, client_address)  # Prejde do menu ak spojenie je uspesne
```

```python
def client_login():  # Login pre klienta
    print("Zadajte IP servera")
    ip = input()  # Zada IP servera
    print("Zadajte PORT servera")
    port = input()  # Zada port servera

    client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)  # Inet - ETHERNET    DGRAM - UDP
    client_socket.bind(("0.0.0.0", 0))  # bind
    print('klientsky port:', client_socket.getsockname()[1])  # Vypise port klienta

    server_address = (ip, int(port))  # Odosle flag serveru
    client_socket.sendto(b'1', server_address)

    while True:
        data, server_address = client_socket.recvfrom(1500)  # Dostane flag a prejde do menu
        if data == b'1':
            client(client_socket, server_address)  # Prechod do menu pri uspesnom spojeni
```

Vyberove menu a login pre klienta a server. Tu priebeha spojenie.

3.2.1 Klient

```
if choice_client == '1':
    flag = b'2'
    print("Zadajte spravu: ")
    client_massage = input()
    print("Zadajte velkost paketu: ")
    packetsize = int(input())
    while packetsize >= 1464 or packetsize <= 0:  # Osetri velkost paketu
        print("Neplatny vstupny udaj \n")
    fragmentsnumber = math.ceil(len(client_massage) / packetsize)  # Spocita kolko bude paketov
    print("Dlzka spravy je ", len(client_massage), "--- Packetov bude ", fragmentsnumber)
    sprava = struct.pack("!cI", flag, fragmentsnumber)  # Zabali udaje do packetu ze co
    client_socket.sendto(sprava, server_address)  # ide robit a posle serverovi
    send_message(client_socket, server_address, client_massage, packetsize, fragmentsnumber)  # Funkcia
```

```
elif choice_client == '2':
    flag = b'3'
    print('Zadajte cestu k fajlu')
    filename = input()
    with open(filename, mode='rb') as file:  # Hned zakoduje file
        filestuct = file.read()
    print("Zadajte velkost paketu: ")
    packetsize = int(input())
    print("Chcete chybu v datach? 1 ano 0 nie")  # Ze ci
    chyba = int(input())  # pouzivatel
    if chyba == 1:  # chce chyby
        print("V akom pakete?")
        chyba = int(input())
    while packetsize >= 1464 or packetsize <= 1:  # Osetri velkost paketu
        print("Neplatny vstupny udaj \n")
    fragmentsnumber = math.ceil(len(filestuct) / packetsize)  # Spocita kolko bude paketov
    print("Dlzka spravy je ", len(filestuct), "--- Packetov bude ", fragmentsnumber)
    sprava = struct.pack("!cI", flag, fragmentsnumber)  # Zabali udaje do packetu ze co
    client_socket.sendto(sprava, server_address)  # ide robit a posle serverovi
    send_file(client_socket, server_address, filename, packetsize, fragmentsnumber, chyba)  # Funkcia
```

```
if choice_client == '5':
    print("Changing roles")
    flag = b'9'
    integger = 1
    sprava = struct.pack("!cI", flag, integger)  # Pocle ziadost na server
    client_socket.sendto(sprava, server_address)
    changingroles(client_socket, server_address)  # Ide do funkcii
```

```
if choice_client == '6':
    print("Bye-Bye")
    flag = b'6'
    integger = 1
    sprava = struct.pack("!cI", flag, integger)  # Pocle ziadost na server
    client_socket.sendto(sprava, server_address)
    bye_bye(client_socket, server_address)  # Ide do funkcii
```

```
udrz = Lock()
stop_thread = False                                          # Definujem Thread
def infinit_worker():
    while True:
        #    print("--> thread work")
        flagg = b'5'
        integger = 1
        sprava = struct.pack("!cI", flagg, integger)         # Posle ziadost na server
        client_socket.sendto(sprava, server_address)

        sprava, sender_address = client_socket.recvfrom(1500)   # Prijme ziadost zo servera

        if sprava == b'5':
        #    print('its ok')
            premenna = ''
        elif sprava == None:
            print('crash')

        udrz.acquire()
        if stop_thread is True:
            break
        udrz.release()
        sleep(3)
# print("Stop infinit_worker()")
th = Thread(target=infinit_worker)
th.start()              # Vytvorim a startnem Thread
sleep(2)
```

Pouzivatel si zvoli jednu zo 4 moznosti. Ze ci chce poslat spravu, alebo file, ze ci chce byt prijimac a ze ci chce ukoncit spojenie. Okrem toho vyssie bezi Thread, ktory paralelne kontroluje ze ci server je aktivny a ze ci spojenie ide.

```
def send_file(client_socket, server_address, client_file, packetsize, fragmentsnumber, chyba):
    with open(client_file, mode='rb') as file:
        filestuct = file.read()  # Znovu zakodujem file do potrebnej podoby,
    FILE_SIZE = len(filestuct)  # lebo do funkcie sa ho neda poslat
    packetpomocny = [0] * FILE_SIZE  # Dummy packet
    checksum = checksum_calculator(filestuct)  # Checksum filu
    number_of_packet = fragmentsnumber

    i = 0
    while i <= int(len(packetpomocny)):
        # print(i, i+packetsize)
        # print(filestuct[i:i+packetsize])
        packetDATApart = filestuct[i:i + packetsize]  # Vystrih udajov do paketu
        i = i + packetsize

        source_port = client_socket.getsockname()[1]  # Source port clienta
        destination_port = server_address[1]  # Dest port servera
        data_length = len(packetDATApart)  # Dlzka paketu
        checksumpart = checksum_calculator(packetDATApart)  # Checksum fragmentu
        udp_header = struct.pack("!IIII", source_port, destination_port, data_length, checksumpart)  # Balime header

        flag = b'3'
        myhead = struct.pack("!cII", flag, number_of_packet, checksum)  # Balim moj header
```

V ramci dokumentacii rozoberem funkciu send_file(), kvoli tomu, ze je velmi podobna funkcii send_meaasge(), ale je viac zlozita vdaka implimentacii tvorby chybnych fragmentov. Zadany file sa prevedie do bytovej podoby a sa rozlozi na fragmenty urcitej velkosti a poslu sa serverovi, ktory ich prijme a odosle naspat flag o ich spravnom, alebo nespravnom prijati.

```
packet_with_header = udp_header + myhead + packetDATApart  # Spojenie vsetkeho

client_socket.sendto(packet_with_header, server_address)  # Odosielanie vsetkeho

sprava, sender_address = client_socket.recvfrom(1500)  # Dostavame odpoved
sprava = struct.unpack("!cI", sprava)  # od servera
fakture = sprava[0]
fragmentsnumber = sprava[1]

if fakture == b'5':  # Flag 5 znamena
    if fragmentsnumber == number_of_packet:  # ze server
        print('Fakturu mam od packeta ', number_of_packet)  # dostal spravny
        print(number_of_packet)  # paket
        number_of_packet = number_of_packet - 1

if fakture == b'6':  # Flag 6 znamena
    print('Nieco sa pokazilo pri pakete ', number_of_packet, ', skusim znovu')  # ze vznikla chyba
    number_of_packet = number_of_packet  # pri odoslani
    #    print(fragmentsnumber)                    # paketu
    #    print(number_of_packet)                   # paket sa odosle
    #    print(i)                                  # znovu
    i = i - packetsize
#    print(i)

if number_of_packet == 0:
    print("Sprava uspesne odoslava\n")
```

Pri prijati flagu od serveru sa urci flag a podla toho sa rozhodne, ze ci klient odosle ten isty paket znovu, alebo ci pokaruje dalej.

```
import random  # Ak chceme chybny packet tak ho
r = random.uniform(0, 1)  # pokazime s moznostou 90% aby
if number_of_packet == chyba:  # pri preposielani v buducnosti
    if r < 0.9:  # sme odoslali spravny paket
        packetDATApart = packetDATApart + b'x1'
```

Ako nastavenie chybneho fragmentu pre overenie ARQ metody mnou bola vybrana kniznica random, ktora s 90% moznostou pokazi vybrany fragment. 90% kvoli tomu, aby cyklus odoslania paketu nebol nekonecny, a aby raz sa tan paket poslal bez problemov.

```
def bye_bye(client_socket, server_address):  # Koniec spojenia
    sprava, server_address = client_socket.recvfrom(1500)
    sprava = struct.unpack("!c", sprava)
    flag = sprava[0]  # Dostane packet od servera
    if flag == b'6':  # ze ziadost prijata
        client_socket.close()  # zavrie socket a prejde


def changingroles(client_socket, server_address):  # Zmena rol
    sprava, server_address = client_socket.recvfrom(1500)
    sprava = struct.unpack("!c", sprava)
    flag = sprava[0]  # Dostane packet od servera
    if flag == b'9':  # ze ziadost prijata
        client_socket.close()  # zavrie socket a prejde
        server_login()  # do loginu
```

Vymena roli funguje tak isto, ako aj opustanie spojenia, okrem toho, ze pri opusteni spojenia pocitace neprejdu do loginu.

3.2.2 Server

```
def server(server_socket, client_address):  # Menu pre server
    print("Hi this is menu")
    while True:
        message = ''
        fileend = []
        sprava, sender_address = server_socket.recvfrom(1500)  # Prijima packet od Klienta
        sprava = struct.unpack("!cI", sprava)  # dostaneme spravu s flagom a poctom fragmentov
        flag = sprava[0]
        fragmentsnumber = sprava[1]

        if flag == b'2':  # Flag 2 je na pripravu k spravam
            print('Prijimam message, fragmentov bude ', fragmentsnumber)
            while fragmentsnumber != 0:
                message += receive_message(server_socket, fragmentsnumber, client_address)
                fragmentsnumber = fragmentsnumber - 1
            print("Server prijal mesage \n\n", message)  # Zlozi spravu z paketov a vypise

        if flag == b'3':  # Flag 3 je na pripravu k filu
            print('Prijimam file, fragmentov bude ', fragmentsnumber)
            print("\nZadajte cestu kam chcete ulozit file")  # a sa spyta kam file
            way = input()  # treba ulozit
            while fragmentsnumber != 0:
                fileendhelp, fragmentsnumber = receive_file(server_socket, fragmentsnumber, client_address)
                if fileendhelp == None:
                    continue
                else:
                    fileend += fileendhelp
            fileend = bytearray(fileend)  # Po nacitani spoji pakety
            with open(way, "wb") as file:
                file.write(fileend)
```

```
    if flag == b'9':  # Flag 9 je na vymenu roli
        print("Changing roles")
        flag = b'9'  # Posle flag ze je pripraveny
        sprava = struct.pack("!c", flag)
        server_socket.sendto(sprava, client_address)
        server_socket.close()  # Zatvori socket
        client_login()

    if flag == b'6':  # Flag 6 je na opustanie systemu
        print("Bye-Bye")
        flag = b'6'  # Posle flag ze je pripraveny
        sprava = struct.pack("!c", flag)
        server_socket.sendto(sprava, client_address)
        server_socket.close()  # Zatvori socket
        break

    if flag == b'5':  # Flag 5 je na udrziavanie spojenia
        #    print('its ok')
        flag = b'5'
        sprava = struct.pack("!c", flag)
        server_socket.sendto(sprava, client_address)
```

Server prijima flagy od klienta a podla toho sa rozhoduje ze co robi a ako odpoveda. Pre vymenu roli, ukoncenie a pre udrzanie spojenia netreba ist do ziadnych inych funkcij.

```python
def receive_file(server_socket, fragmentnumber, client_adderss):
    full_packet, sender_address = server_socket.recvfrom(1500)
    # Prijima file od klienta
    udp_header = full_packet[:16]  # Vyberie UDP header
    myhead = full_packet[16:25]  # Vyberie moj header
    data = full_packet[25:]  # Vyberie data

    udp_header = struct.unpack("!IIII", udp_header)  # Rozbali UDP header
    correct_checksum = udp_header[3]
    checksum = checksum_calculator(data)  # Vypocita checksum pre data

    myhead = struct.unpack("!cII", myhead)  # Rozbali moj header
    numberofpacket = myhead[1]
   # print('prisiel packet ', numberofpacket)

    if correct_checksum == checksum:  # Ak sa to zhoduje s headerom
        print('Spravny paket, ', fragmentnumber, ' -posielam fakturu')  # poslem fakturu
        fakture = b'5'  # ze prijaty packet bol spravny
        sprava = struct.pack("!cI", fakture, fragmentnumber)
        server_socket.sendto(sprava, client_adderss)
        fragmentnumber = fragmentnumber - 1

        return data, fragmentnumber
```

```python
elif correct_checksum != checksum:  # Ak sa to nezhoduje s hlavickou
    print('Nepravny paket, ', fragmentnumber, ' - posielam ziadost')  # poslem fakturu
    fakture = b'6'  # ze prijaty packet nebol spravny
    sprava = struct.pack("!cI", fakture, fragmentnumber)
    server_socket.sendto(sprava, client_adderss)
    fragmentnumber = fragmentnumber

    return None, fragmentnumber
```

Pri prijati filu sa rozbalia data a na zaklade porovnavania checksumu dat a checksumu, stanoveneho v headere zistim ze ci fragment je chybny, alebo nie. Podla vysledku sa rozhodnem ze aky flag odoslem klientovi a ci si ulozim data, alebo nie.

3.2.3 Vystup

```
1 for client
2 for server
3 to exit
1
Zadajte IP servera
192.168.88.225
Zadajte PORT servera
5000
klientsky port: 63533
0 for exit
1 for text message
2 for file message
5 for switching role
6 end communikation
```

```
1 for client
2 for server
3 to exit
2
Zadajte PORT servera
5000
Adresa servera je  192.168.88.146
Hi this is menu
```

Pri starte programu si zvolim rolu a postupujem podla pokynov, aby vytvorit spojenie.

```
2 for file message
5 for switching role
6 end communikation
2
Zadajte cestu k fajlu
C:\Users\user\Desktop\macbook.png
Zadajte velkost paketu:
1400
Chcete chybu v datach? 1 ano 0 nie
1
V akom pakete?
12
Dlzka spravy je  1267811 --- Packetov bude  906
```

Pri odosielani filu postupujem tiez podla pokynov. Dolezite je vediet, ze minimalna dlzka fragment moze byt len 2B, a maximalne 1464B. Pri posielani spravy, ale minimalna dlzka je 1B.

```
Fakturu mam od packeta  14
14
Fakturu mam od packeta  13
13
Nieco sa pokazilo pri pakete  12 , skusim znovu
Nieco sa pokazilo pri pakete  12 , skusim znovu
Nieco sa pokazilo pri pakete  12 , skusim znovu
```

```
Fakturu mam od packeta  2
2
Fakturu mam od packeta  1
1
Sprava uspesne odoslava
```

Dalej sa posiela file a sa zobrazuje stav paketov nasledovne. Na konci bude ukazane uspesne odoslanie spravy.

```
Prijimam file, fragmentov bude  906

Zadajte cestu kam chcete ulozit file
C:\Users\user\Desktop\macbookTEST.png
```

```
Spravny paket,  12  -posielam fakturu
Spravny paket,  11  -posielam fakturu
Nepravny paket,  10  - posielam ziadost
Nepravny paket,  10  - posielam ziadost
Nepravny paket,  10  - posielam ziadost
```

Zo strany servera treba len zadat cestu, kam sa ulozi file a pozerat na vysledok.