

**Slovenská Technická Univerzita  
Fakulta Informatiky a Informačných  
Technológií**

Viktor Barczy  
PKS  
Zadanie 2  
2020/2021

# 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.

## Moje riešenie

Program som urobil v programovacom jazyku C/C++, pracoval som vo Visual-Studio a používal som knižnice ako napríklad: *stdlib.h*, *winsock2.h*, *WS2tcpip.h* a *iostream*.

Na začiatku používateľ môže vybrať že či potrebuje **servera** alebo **klienta**.

## Začiatok programu

### Klient

Keď používateľ vybral servera tak potom sa začne **WSA startup** ktorý keď nezbehne tak server sa ukončí a do *cmd* sa vyíše že „Can't start Winsock!“, ale zbehne tak používateľ môže zadať jeden „command“.

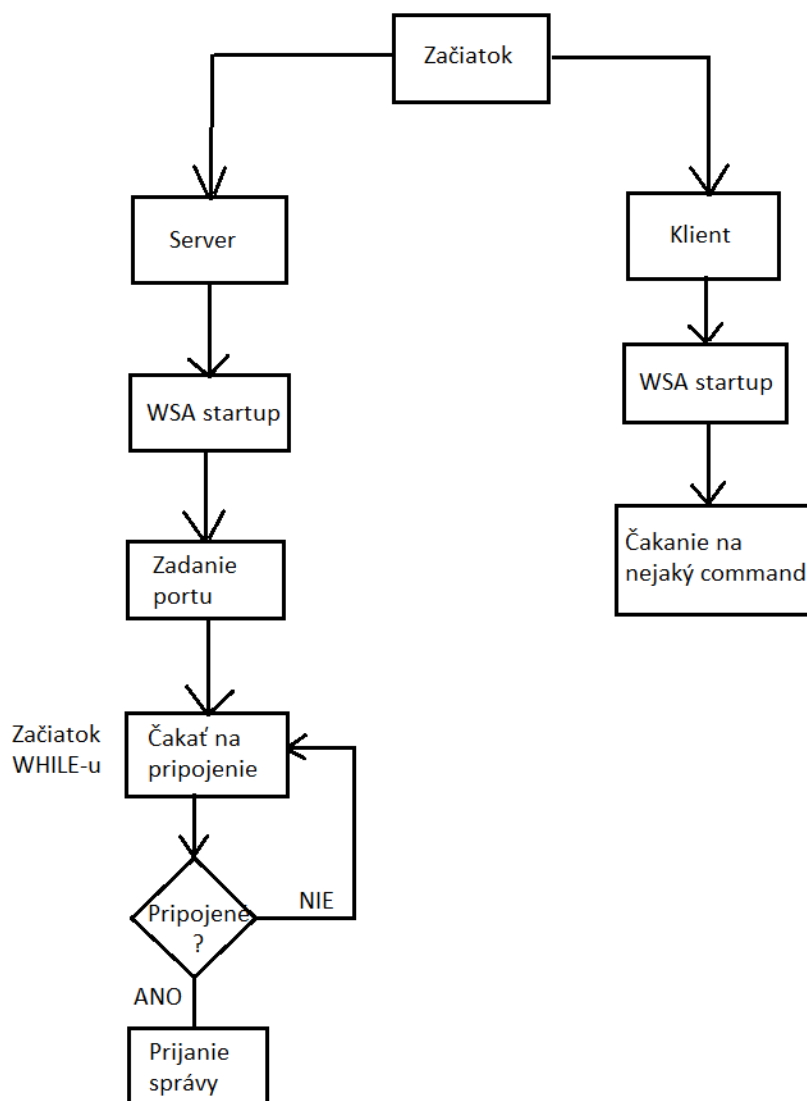
Príkazy pre klienta:

- PORT – pomocou tohto príkazu používateľ môže nastaviť port pre klienta, keď používateľ to nenastaví tak port bude *DEAFULT\_PORT* (1234).
- SERVER - pomocou tohto príkazu používateľ môže nastaviť IP adresu klienta, používateľ vždy musí nastaviť IP adresu pred poslaným správou inak nebude vedieť nič poslať serverovi.
- CHAT – tento príkaz je na poslanie textového správ na server.
- FILE – tento príkaz je na poslanie súboru na server.
- FRAG – pomocou tohto príkazu používateľ môže nastaviť veľkosť fragmentov pri odoslaní správ, keď to používateľ nenastaví tak default je 1400 a používateľ nevie zvoliť menšiu alebo rovnakú veľkosť ako header.

- CHYBA - tento príkaz funguje tak ako CHAT ale tak pošle serverovi packet že dá mu zlú CRC a keď packet nemá dobrý CRC tak sa nastane chyba.
- QUIT – je na ukončenie klienta.

## Server

Keď používateľ vybral servera tak potom sa začne **WSA startup** ktorý keď nezbehne tak server sa ukončí a do *cmd* sa vyíše že „Can't start Winsock!“, ale zbehne tak používateľ môže zadať port. Keď používateľ nezadal žiadny port, tak port bude *DEFAULT\_PORT* (1234). Potom sa začne jeden WHILE cyklus kde najprv server bude čakať na prvý packet od klienta keď nič neprišiel čo znamená že klient a server nie sú pripojený tak server bude ďalej čakať. Keď príde nejak packet (message) tak server do CMD vypíše IP adresu klienta od koho prišiel a potom pozrie v pakete že či prišla správa (text) alebo súbor (file). Keď jeden z nich, tak server pošle klientovi jednu správu že **ÁNO dostal som tvoju správu** (acknowledgement).



## CRC

Tak robím CRC pre jednu srpávu že v data, čo chcem odoslať serverovi, prejdem všetky bity s pomocou **xor** a výsledok si napíšem do header v packet. Potom server tak isto urobí crc pre data a keď dostane to čo je v header tak všetko je v pohode (takto vie server že dostal to čo mu klient odoslal).

KÓD NA VYPOČÍTANIE CRC:

```
char crc(char* buf, int buflen) {  
    char mycrc = 0;  
    for (int i = 0; i < buflen; ++i)  
        mycrc ^= buf[i];  
    return mycrc;  
}
```

## Informačný packet

Typ srpávy	Počet paketov	Celková dĺžka dát	Ostatné dáta
------------	---------------	-------------------	--------------

Tento packet pošle klient serverovi predtým ako pošle dáta. Server keď posiela klientovi „acknowledgement“ tak používa takýto packet.

Typ správy môžu byť:

TYP\_NONE 0  
TYP\_ACK 10  
TYP\_NAK 11  
TYP\_PART 99

## Packet na poslanie dát

Typ srpávy	Poradové číslo	Veľkosť	CRC	DATA
------------	----------------	---------	-----	------

Takto vyzerá jeden packet na poslanie dát.

Typ správy môžu byť:

TYP\_NONE 0  
TYP\_CHAT 1  
TYP\_FILE 2  
TYP\_PART 99

# Poslanie a prijatie textu

## Klient/Odosielateľ

Keď používateľ chce poslať textovú správu tak najprv musí nastaviť IP adresu a port s príkazmi SERVER a PORT. Potom s príkazom CHAT môže poslať textovú správu (napr.: CHAT ahoj). Preto ako klient pošle text packet klient najprv pošle jeden packet na to aby prekontroloval že či je pripojený so serverom ak nie tak nebude posilať ďalšie packety (klien tak vie že server dostal správu že server odošle klientovi jeden „*acknowledgement*“), potom si vytvorím jeden packet a prvé 4 bity si nastavím na 1 (takto vie server že prišla textová správa). Potom do ďalších 4 bitov si napíšem poradové číslo fragmenta, do ďalších 4 dám veľkosť fragmenta a ešte 4 na CRC (). Nakoniec zostávajúce bity budú na text čo chcem odoslať, potom tento packet odošlem k serverovi a server má zase poslať naspäť jeden „*acknowledgement*“.

## Server/Prijímateľ

Najprv server od klienta dostane jeden informačný packet, keď to nedostane alebo dostane niečo zle tak server vypíše do CMD že: „*Error receiving from client...*“. Keď dostane informačný packet správne tak server bude čakať na ďalší packet. Tento packet môže byť textová správa alebo nejaký súbor. Keď je to text tak server najprv si pozrie že či packet má dobrý CRC keď áno tak iba do CMD vypíše správu.

# Poslanie súboru

## Klient/Odosielateľ

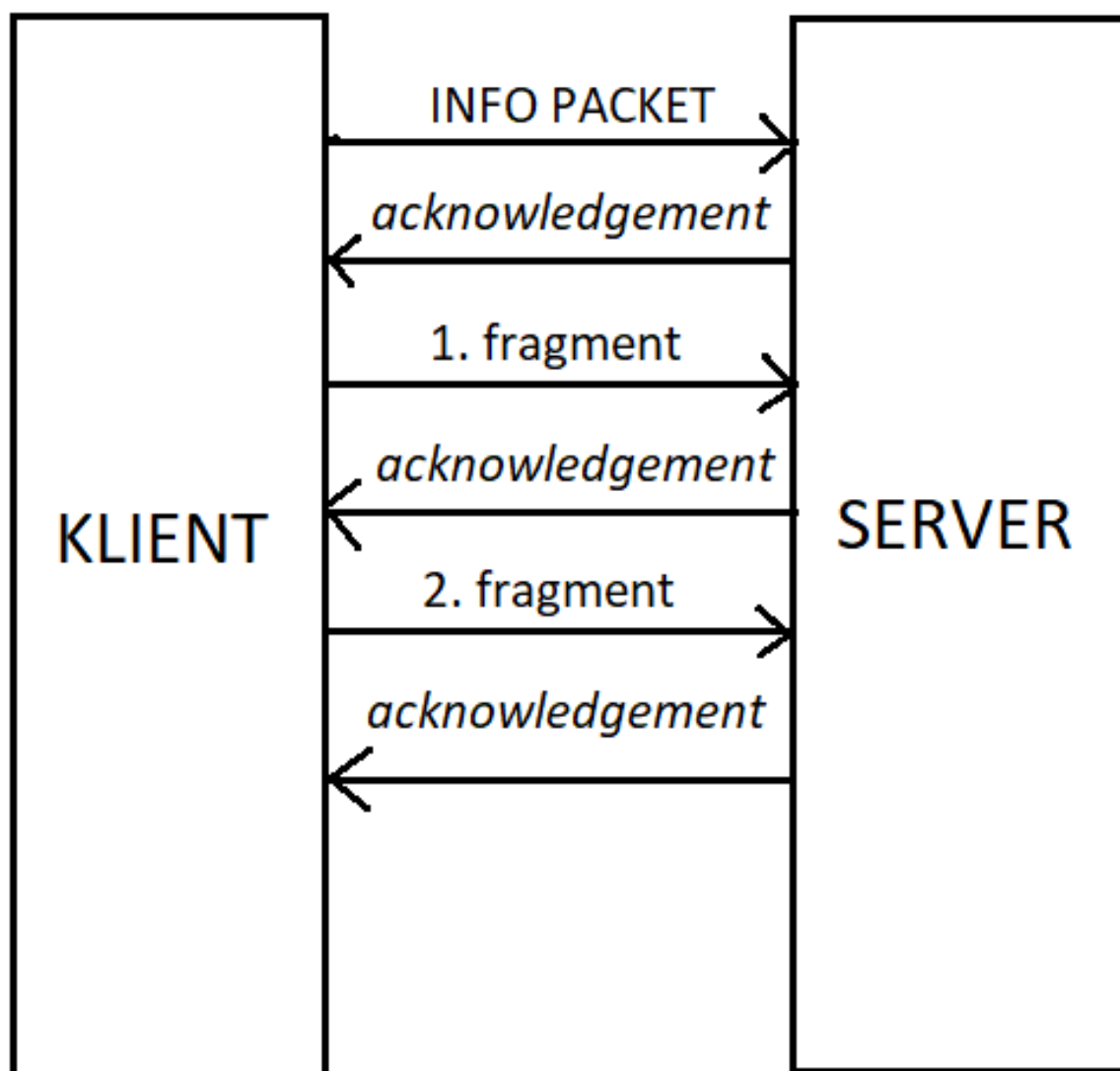
Odoslanie súboru funguje takmer tak isto ako odoslanie textu. Keď klient dostane príkaz FILE tak klient odošle jeden info packet a bude čakať na odpoveď, potom používateľ musí zadať meno súboru, keď to program našiel tak to natiahne do priečinka **tmp**. Funguje to tak, že keď program našiel súbor tak pozrie že aký veľký je ten súbor, predelí veľosť súboru s veľkosťou najväčšieho fragmenta a spolu s menom súboru najprv pošle informačný packet k serverovi a potom mu pošle

## Server/Prijímateľ

Prijímanie súboru funguje tak že keď server dostane správu že príde súbor (informačný packet) tak najprv otvorí priečinok **tmp** (kde bude súbor uložený) potom podľa to koľko packetov príde (čo nájde v informačnom packety) postupne bude pridávať dáta, tak ako prídu od klienta, na to isté miesto a vždy keď dostane nejaký packet tak pošle naspäť klientovi jeden ACK. Aby klient vedel že server dostal packet. Na konci keď už server dostal všetky dáta zatvorí súbor a do CMD vypíše počet fragmentov, počet bitov a to že kde to uložil (file path).

## Komunikácia medzi klientom a serverom

Takto komunikuje server a klient keď klient posiela taký súbor ktorý rozdelí na 2 fragmenty.



## Splnené požiadavky

- 1 Nastavenie IP a port
- 2 Prenos súboru menšieho ako nastavená veľkosť fragmentu
- 3 Prenos 2MB súboru
- 4 Používateľ má možnosť zvoliť si max. veľkosť fragmentu.
- 5 Server je schopný zobrazovať:
  - a názov a absolútnu cestu k súboru na danom uzle,
  - b veľkosť a počet fragmentov.
- 6 Program je implementovaný v jazyku C/C++ a používa knižnicu winsock2.h
- 7 Simulovanie minimálne jednej chyby (chyba pri CRC).

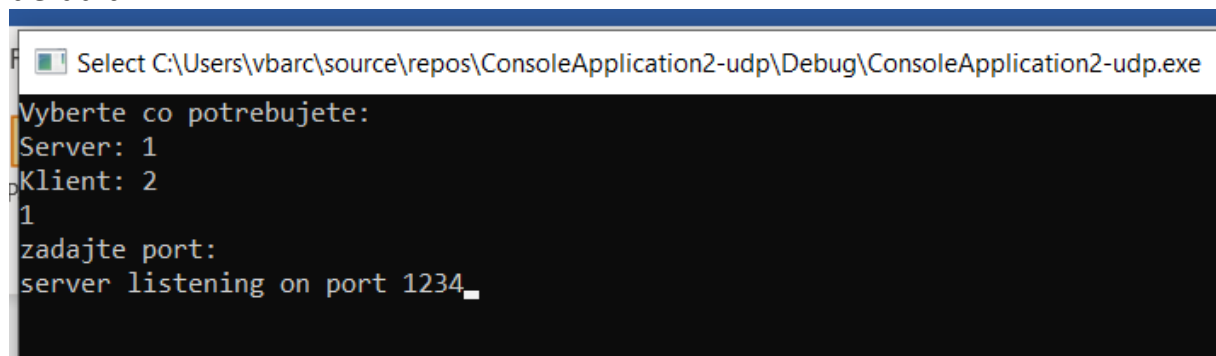
## Zmeny ktoré nastali v implementácii oproti návrhu.

Klient a server mám v jednom kóde a nie sú oddelené na dvoch ako v návrhu som to chcel. V návrhu som mal dve rôzne packety na odoslanie textových a súborových správ ale v implementácii mám iba jeden typ packetu na odoslanie dát.

## Testy

### 1. Test na poslanie textu

Otvoril som jeden program, nastavil som ho ako **server** a port som nechal ako default.



```
Select C:\Users\vbarc\source\repos\ConsoleApplication2-udp\Debug\ConsoleApplication2-udp.exe
Vyberte co potrebujete:
Server: 1
Klient: 2
1
1234
server listening on port 1234
```

Potom som otvoril ešte jeden program ale ten som už nastavil ako **klient** a IP adresu som nastavil na 127.0.0.1 s príkazom SERVER.

```
C:\Users\vbarc\source\repos\ConsoleApplication2-udp\Debug\ConsoleApplication2-udp.exe
Vyberte co potrebujete:
Server: 1
Klient: 2
2
>SERVER 127.0.0.1

ip nastavene na 127.0.0.1
>
```

Potom do **klienta** som napísal tento príkaz „CHAT ahoj“ stlačil som enter a ešte raz som odoslal jednu správu „CHAT co robis?“

```
C:\Users\vbarc\source\repos\ConsoleApplication2-udp\Debug\ConsoleApplication2-udp.exe
Vyberte co potrebujete:
Server: 1
Klient: 2
2
>SERVER 127.0.0.1

ip nastavene na 127.0.0.1
>CHAT ahoj
>CHAT co robis?
>
```

Na server som dostal takúto správu

```
C:\Users\vbarc\source\repos\ConsoleApplication2-udp\Debug\ConsoleApplication2-udp.exe
Vyberte co potrebujete:
Server: 1
Klient: 2
1
zadajte port:
server listening on port 1234
Message recv from 127.0.0.1

CHAT>ahoj

Message recv from 127.0.0.1

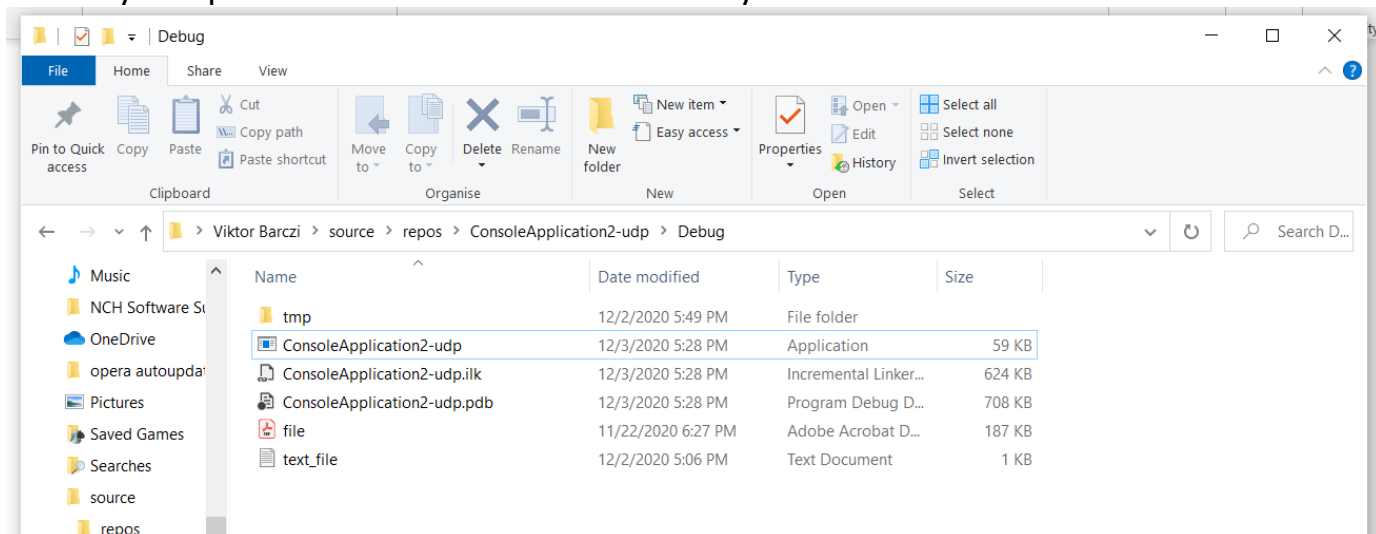
CHAT>co robis?
```



## 2. Testovanie poslanie súboru

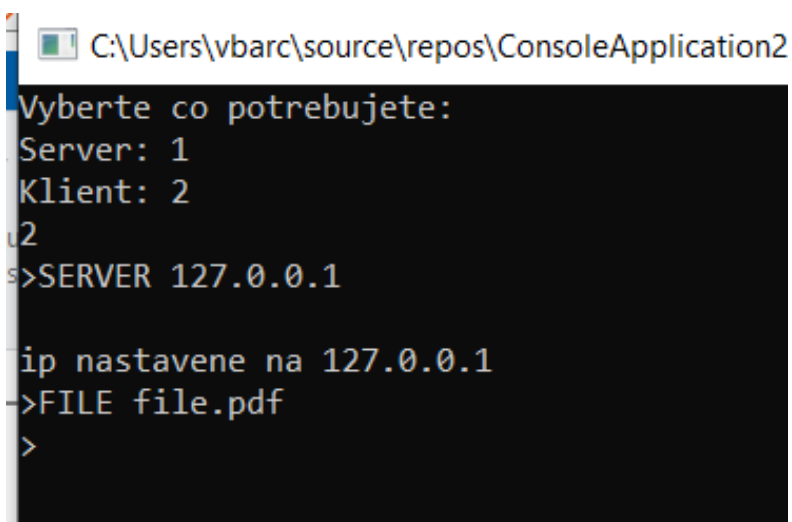
Ďalej som testoval poslanie súboru tak, že do klienta som najprv napísal príkaz FRAG 1500 tento príkaz nastavil najväčší fragment size na 1500 bitov (default je 1400). Potom som poslal jeden súbor do priečinka **tmp** jeden pdf súbor s menom file (tento tmp priečinok je umiestnený tam kde je aj .exe). Používal som FILE príkaz a takto:  
„FILE file.pdf“

Takto vyzerá priečinok kde mám .exe umiestnený:



tmp je momentálne prázdny.

KLIENT:



Klient nevypísal že *došlo k chybe* takže úspešne to odoslal.

## SERVER:

```
C:\Users\vbarc\source\repos\ConsoleApplica
Vyberte co potrebujete:
Server: 1
Klient: 2
1
zadajte port:
server listening on port 1234
Message recv from 127.0.0.1

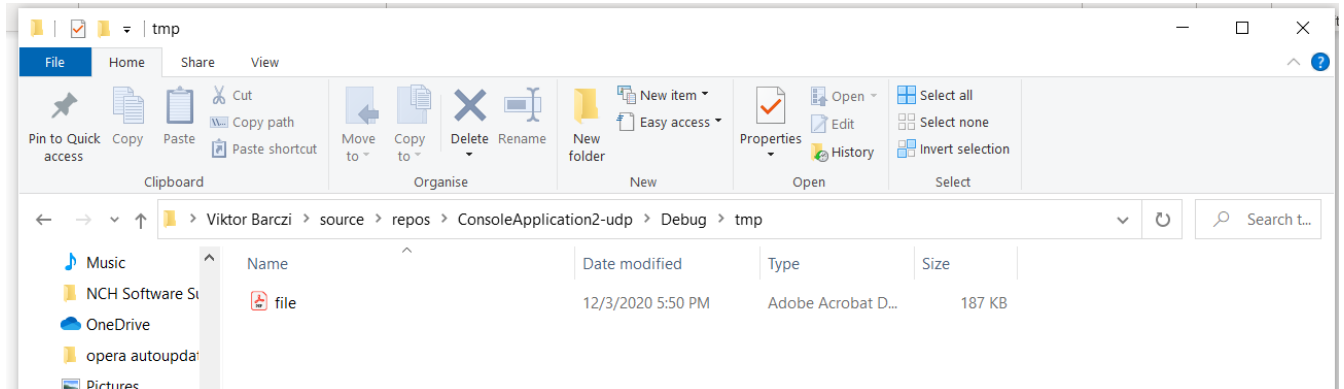
doslo 1384bytov
doslo 1384bytov
doslo 1384bytov
doslo 1384bytov
doslo 1384bytov
doslo 1384bytov
doslo 1384bytov
doslo 1384bytov
doslo 1384bytov
doslo 1384bytov
doslo 1384bytov
doslo 1384bytov
```

.  
.  
.  
.  
.  
.  
.

```
C:\Users\vbarc\source\repos\ConsoleApplication2-udp\Debug\ConsoleApplication2-udp.exe
doslo 1384bytov
doslo 1384bytov
doslo 1384bytov
doslo 1384bytov
doslo 1384bytov
doslo 1384bytov
doslo 1384bytov
doslo 1384bytov
doslo 1384bytov
doslo 1384bytov
doslo 1249bytov

koniec prenosu
zapisane 190857bytov, pocet fragmentov:138, subor:C:\Users\vbarc\source\repos\ConsoleApplication2-udp\Debug\tmp\file.pdf
```

Server vypísal všetky fragmenty a to že koľko bytov prišiel v jednom fragmente. Na koniec server vypísal že koľko bytov prišiel, koľko fragmentov a že kde to uložil. Všetko fungovalo tak ako má. A už v priečinku **tmp** nájdeme ten súbor file.pdf.



Potom som otvoril ten súbor aby som bol presvedčený že či to dobre prekopíroval a všetko bolo v poriadku.

## Funkcie

**Client()** – táto funkcia je hlavná funkcia pre klienta

**Server()** – táto funkcia je hlavná funkcia pre servera

**Chat()** – klient pomocou tohto funkcií pošle textovú správu serverovi

**Sendfile()** – klient pomocou tohto funkcií pošle súborovú správu serverovi

**Recv\_chat()** – dostane server textovú správu

**Recv\_file()** – dostane server súborovú správu

**Crc()** - pomocou tohto funkcií vypočítam CRC.

**Send\_ACK()** – toto používam vtedy keď server pošle klientovi že *áno dostal som správu (acknowledgement)*.

**waitforACK()** – klient čaká na *acknowledgement*.

**Clinet\_send()** – pošle jeden packet serverovi.