Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Ilkovičova 2, 842 16 Bratislava 4

**Zadanie 1 – SIP Proxy (telefónna ústredňa)**

Mobilné technológie a aplikácie

Patrik Kecera

ID: 110815

ak. rok: 2021/22

Cvičiaci: Ing. Adam Ševčík

**OBSAH**

[1. ZADANIE ÚLOHY 3](#_Toc89893105)

[2. NÁVRH RIEŠENIA 4](#_Toc89893106)

[2.1. Štruktúra hlavičky 4](#_Toc89893107)

[2.2. Metóda kontrolnej sumy (checksum) 5](#_Toc89893108)

[2.3. ARQ metóda 5](#_Toc89893109)

[2.4. Keep – Alive správa 5](#_Toc89893110)

[2.5. Inicializácia spojenia 5](#_Toc89893111)

[2.6. Ukončenie spojenia 5](#_Toc89893112)

[2.7. Príklad komunikácie 5](#_Toc89893113)

[2.8. Diagram spracovávania komunikácie 6](#_Toc89893114)

[3. ZMENY NÁVRHU 7](#_Toc89893115)

[3.1. Štruktúra hlavičky 7](#_Toc89893116)

[3.2. Metóda kontrolnej sumy (checksum) 8](#_Toc89893117)

[3.3. Simulácia chyby 8](#_Toc89893118)

[3.4. Keep-alive metóda 8](#_Toc89893119)

[4. IMPLEMENTÁCIA 9](#_Toc89893120)

[5. POUŽÍVATEĽSKÉ PROSTREDIE 12](#_Toc89893121)

[5.1. Klient 12](#_Toc89893122)

[5.2. Server 14](#_Toc89893123)

[5.3. Výmena servera s klientom 15](#_Toc89893124)

[6. DIAGRAM SPRACOVÁVANIA KOMUNIKÁCIE 16](#_Toc89893125)

[7. ZOBRAZENIE VO WIRESHARK 17](#_Toc89893126)

[8. ZÁVER 17](#_Toc89893127)

**DOKUMENTÁCIA**

## ZADANIE ÚLOHY

## NÁVRH RIEŠENIA

## Štruktúra hlavičky

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok 1 štruktúra vlastnej hlavičky

**Flags** (1B)

Jednotlivé flagy budú jednobitové čísla. 1 označuje že je flag aktívný a 0 že je neaktívny. Flagy označujú typ paketu. Veľkosť políčka flags je 1 B ako je aj ukazané na Obrázku 1.

I - inicializačná správa ( ako SYN u TCP protokolu)

K - koniec spojenia ( ako FIN u TCP protokolu)

S - posielanie správy

F - posielanie súboru

U - udržiavanie spojenia tzv. keep-alive správa

A - správa o úspešnom doručení (acknowledgement)

E - doručene dáta boli chybné

Z - správa o poslaní dát znova

|  |  |
| --- | --- |
| Flag | Označenie v dátach |
| I | 10000000 |
| K | 01000000 |
| S | 00100000 |
| F | 00010000 |
| U | 00001000 |
| A | 00000100 |
| E | 00000010 |
| Z | 00000001 |

*Tabuľka 1 ukážka flagov v bitoch*

**Total Length**  (2B)

Veľkosť paketu v bytoch. Podľa výpočtu môže byť maximálna dĺžka 1463 B. Výpočet:

1500B – 20B( IP ) – 8B ( UDP protokol ) – 9 B(moja hlavička) = 1463 B.

**Fragment number** (2B)

Fragment number označuje poradové číslo fragmentu.

**Checksum** (4B)

Checksum je výsledok funkcie crc32 z knižnice zlib. Výsledok je 4 bytové nezáporne číslo.

## Metóda kontrolnej sumy (checksum)

Na výpočet kontrolnej sumy ( checksum ) použijeme funkciu crc32 z knižnice zlib. Crc je algoritmus, ktorý vypočíta z dát kontrolnú sumu. Kontrolná suma bude slúžiť na overenie správnosti prijatého fragmentu. Bude sa porovnávať checksum v hlavičke a checksum z dát prijatého fragmentu. Funkcia crc32 funguje na delení polynómu, ktorý je nižsie. Zvyšok po delení je checksum teda kontrolná suma, ktorá sa bude posielať v hlavičke. Checksum je vždy 4 bytový unsigned int.

## ARQ metóda

ARQ metóda, je metóda na kontrolu chýb pri prenose dát. Využívajú sa pritom správy o potvrdení a timeouty. V našom protokole budeme používať najľahšiu stop and wait ARQ metódu. Stop and wait ARQ metóda funguje nasledovne. Klient odošle fragment na server a čaká správu ack teda správu, ktorú odošle server, že úspešne prijal bezchybný fragment. Ak klient nedostane ack správu do určitého časového limitu alebo dostane inú správu od servera ( správu o chybnom fragmente alebo o poslaní fragmentu znova), tak klient odošle ten istý fragment znovu.

## Keep – Alive správa

Keep -Alive správu sa bude posielať od klienta ak server neposlal správu o obdržaní dát po dobu 5 sekund. Ak klient pošle 3 krát za sebou správu keep-alive a server neodpovie, tak klient odošle správu fin o ukončení spojenia. V prípade, že server obdrží správu keep-alive, tak odošle správu ack a tak sa udrží spojenie. Potom komunikácia pokračuje

## Inicializácia spojenia

Inicializácia alebo nadviazanie spojenia funguje veľmi jednoducho. Najprv klient zadá ip adresu, port a maximálnu veľkosť fragmentu a server zadá port na ktorom počúva. Následne klient pošle serveru inicializačnú správu ( flag I je aktivovaný ). Server pri prijatí inicializačnej správy odošle správu ACK teda správu o úspešnom doručení (flag A je aktivovaný).

## Ukončenie spojenia

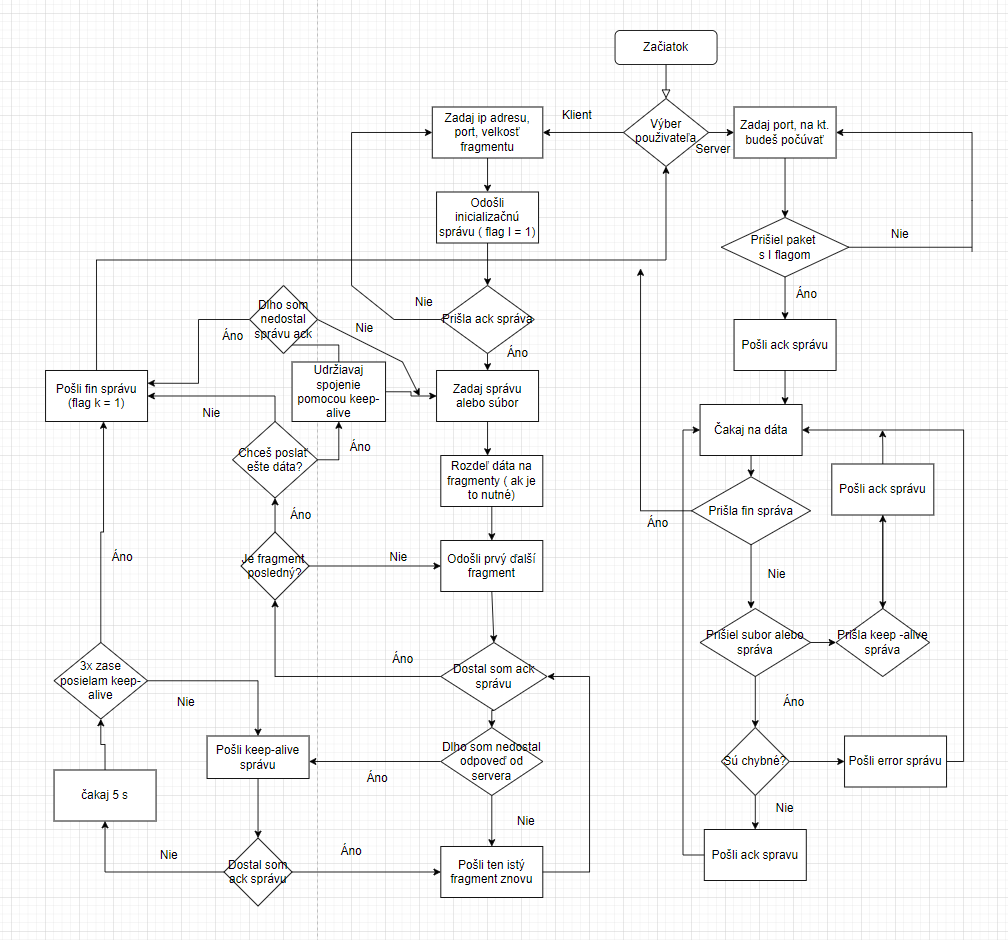
Ukončenie spojenia prebieha ešte jednoduchšie ako nadviazanie. Klient odošle serveru správu o ukončení a obaja ukončia spojenie. Ukončiť spojenie sa môže buď ak server neodpovedá na keep-alive správy alebo ak klient už nechce ďalej odosielať dáta.

## Príklad komunikácie

Na obrázku 2 je ukážka odoslania 1 správy. Najprv sa medzi klientom a serverom nadviaže spojenie pomocou inicializačnej správy a správy o úspešnom prijatí. Následne klient pošle správu a server odpovie správou o úspešnom prijatí. Ak mi nastala chyba (checksum z dát a checksum z hlavičky by sa nerovnal), tak by sa správa vyhodila a server by odoslal error správu a vyžiadal si správu znova. Nakoniec po odoslaní správ alebo po dlhom neodpovedaní servera klient odošle správu o ukončení spojenia.

## Diagram spracovávania komunikácie

Obrázok 2 ukážka odoslania 1 správy



Obrázok 3 Diagram spracovávania komunikácie

## ZMENY NÁVRHU

## Štruktúra hlavičky

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok 4 upravená štruktúra hlavičky

V hlavičke sa zmenila veľkosť Checksumu z 4 B na 2 B. Potom sa nahradila dĺžka fragmentu počtom prenášaných fragmentov, ktoré má veľkosť 4 B.

**Flags** (1B)

Flagy sa zmenili minimálne. Jediná zmenená flag je Z, ktorá reprezentuje správu o vymenení rolí(server, klient)

I - inicializačná správa ( ako SYN u TCP protokolu)

K - koniec spojenia ( ako FIN u TCP protokolu)

S - posielanie správy

F - posielanie súboru

U - udržiavanie spojenia tzv. keep-alive správa

A - správa o úspešnom doručení (acknowledgement)

E - doručene dáta boli chybné

Z - správa o vymenení rolí

|  |  |  |
| --- | --- | --- |
| Flag | Označenie v dátach | Dekadicky |
| I | 10000000 | 128 |
| K | 01000000 | 64 |
| S | 00100000 | 32 |
| F | 00010000 | 16 |
| U | 00001000 | 8 |
| A | 00000100 | 4 |
| E | 00000010 | 2 |
| Z | 00000001 | 1 |

*Tabuľka 2 ukážka upravených flagov*

**Number of fragments** (4B)

Number of fragments označuje počet fragmentov prenášaného súboru alebo správy.

4B preto aby mohlo byť prenášaných až 232 fragmentov.

## Metóda kontrolnej sumy (checksum)

Checsum bol upravený z 4B na 2B. Na výpočet checksumu bola použitá funkcia crc\_hqx z knižnice binascii. Funkcia crc\_hqx vypočíta 16 bitový crc výsledok. Na vypočítanie výsledku používa polynóm zobrazený na nasledujúcom obrázku. Táto funkcia bola vybraná lebo obsahuje minimálne zhody crc výpočtov a v porovnaní s 4 B metódou šetrí pamäť v hlavičke.

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok 5 polynóm CRC-CCITT

## Simulácia chyby

Simulácia chyby prebieha veľmi jednoducho. S 20 % pravdepodobnosťou je poslaný chybný fragment. Chybný fragment je taký, ktorému najprv vypočítame checksum z dát a potom dáta zmeníme.



## Keep-alive metóda

Keep-alive metóda bola skoro úplne zachovaná z návrhu. Rozdiel je v tom, že keep-alive metóda nefunguje počas posielania fragmentov. Keď sa posielajú fragmenty, vtedy sú nastavené timeouty a ak príde timeout, tak je správa poslaná znova. Ak je tá istá správa poslaná 3x ( timeout 3x na jeden fragment), tak je klient pošle serveru správu o ukončení a vypne sa.

## IMPLEMENTÁCIA

Zadanie bolo implementované v jazyku python 3.9. V programe boli použité knižnice zobrazené na obr. 6. Zadanie som implementoval ako jeden program, v ktorom sú 2 hlavné funkcie (server a client), preto pre komunikovanie klienta so serverom musí byť pustený program 2x na jednom PC ( pri loopbacku) alebo na 2 PC, kde jeden spustí sa v programe ako server a druhý ako klient.

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok 6 knižnice programu

V programe som implementoval aj 2 hlavné triedy. Jedna trieda sa volá fragment, ktorá reprezentuje fragment ktorý sa ide odosielať a druhá fragment\_rec, ktorá reprezentuje fragment, ktorý je prijímaný.. Fragment berieme ako hlavička (9B) + prenášané dáta.

**Trieda fragment**

Ako bolo spomenuté, trieda fragment označuje fragment, ktorý sa ide odosielať. Táto trieda je používaná v funkcií client, pri posielaní fragmentov. Pred posielaním fragmentu sa najprv vytvorí objekt triedy fragment, ktorému sa inicializuje hlavička a pridajú sa dáta. Potom sa fragment odošle.

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok 7 trieda fragment

**Trieda fragment\_rec**

Trieda fragment\_rec označuje fragment, ktorý prichádza. Trieda fragment\_rec je preto využívaná v funkcii server, keď prijíma fragmenty. Pri prijatí sa vytvorí objekt triedy fragment\_rec a následne sa získajú z tohto fragmentu informácie z hlavičky.

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok 8 trieda fragment\_rec

**Funkcia client**

Funkcia client je volaná ak si používateľ zvolí, že chce byť klientom (chce odosielať dáta serveru). Klient si zo začiatku vypýta ip adresu, port servera a maximálnu veľkosť fragmentu (1-1465B). Potom spustí funkciu na inicializovanie spojenia. Keď sa podarí inicializovať spojenie so serverom, tak následne sa opýta používateľa akú správu má poslať serveru. Používateľ si môže zvoliť 3 typy správy M – poslanie správy, F – poslanie súboru, K – ukončenie spojenia. Potom funkcia následne podľa zvoleného príkazu zvolí čo ide odosielať sa. Pri správe používateľ zadá správu a následne ju funkcia rozdelí na fragmenty zadanej veľkosti a odosiela fragment s tým že ak je simulácia chyby úspešná, tak ešte predtým zmení dáta, aby boli chybné. Potom čaká na ACK správu od servera. Ak ju nedostane do 1 sekundy alebo ak dostane error správu, tak posiela daný fragment znova. Po odoslaní všetkých fragmentov funkcia vypíše počet správne odoslaných fragmentov a čaká 4 sekundy či sa chce server zmeniť na klient, ak ju nedostane, tak zase funkcia zobrazí akú správu má poslať.

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok 9 časť funkcie client

**Funkcia server**

Funkcia server je volaná, ak si používateľ zvolí stranu servera (prijímača). Server na začiatku si vypýta od používateľa, na ktorom porte má počúvať. V jednoduchosti potom server čaká na pakety na danom porte a ak príde nejaký paket, potom rozkóduje hlavičku a podľa flagu v nej sa rozhodne, čo robiť ďalej. Teda ak príde flag I ( inicializácia spojenia), tak odošle správu ACK a inicializácia spojenia je úspešna. Pri prijímaní správy čaká na flag S a postupne si ukladá do stringu správu. Na konci keď príde posledný fragment, správu vypíše. Samozrejme pri prijímaní všetkých fragmentov je najprv porovnávaný checksum v hlavičke s vypočítaným checksumom z dát. Ak sa tieto 2 hodnoty nerovnajú, tak je fragment zahodený a je poslaná error správa. Ak sa hodnoty rovnajú, tak je poslaná ACK správa. Pri prijímaní súboru je to komplikovanejšie. Najprv musí byť prijatý paket s flagmi I a F, ktorý inicializuje prijímanie súboru. Pri tomto si server zvolí, kde sa má prijatí súbor uložiť a pošle ACK. Potom prichádzajú fragmenty súboru. Keď príde posledný fragment súboru, tak sa súbor zapíše do vybraného miesta.

**Funkcia ka\_func**

Funkcia ka\_func je funkcia na vykonávanie keep-alive správ. Táto funkcia je vykonávana v samostatnom threade (vlákne). Funkcia je spustená z funkcie klient, keď sa neodosielajú dáta. A funkcia odosiela každých 5 s keep alive správu a čaká na odpoveď. Ak nedostane 3x za sebou odpoveď, tak pošle signál do funkcie client a tá vypíše, že server 3x neodpovedal na KA správu a ukončí spojenie.

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok KA funkcia

## POUŽÍVATEĽSKÉ PROSTREDIE

Používateľské prostredie začína v menu ( obr. 11), v ktorom si používateľ zvolí server alebo klienta alebo ukončenie programu.

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok 11 menu

## Klient

Pri zvolení klienta ( stlačením 0) sa zobrazí informácia, že je používateľ v klientovi a následne zadá používateľ zadá ip adresu servera port a maximálnu veľkosť fragmentu. Po zadaní veľkosti začne klient odosielať každé 3 sekundy inicializačné správy, ak na žiadnu z nich server neodpovie, tak sa soket zatvorí a program vráti do menu.(obr. 12)

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok 12 neúspešné inicializovanie spojenia

Pri úspešnom inicializovaní spojenia so serverom sa ukáže menu, kde si môže používateľ zvoliť akú

správu chce poslať M - poslanie správy, F – poslanie súboru, K - ukončenie spojenia. Pri vybraní poslania súboru musí potom používateľ zadať absolútnu cestu ku súboru, ktorý chce poslať. Po zadaní klient odošle názov súboru a čaká na ACK správu od servera. Ak do 60 s žiadna nepríde, tak spojenie sa ukončí. Ak príde ACK, tak sa začnú odosielať fragmenty. Na koniec sa vypíše počet správne odoslaných fragmentov a absolútna cesta. (obr. 13)

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok 13 posielanie súboru

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok 14 posielanie správy

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok 15 pri neodpovedaní servera

## Server

Pri zvolení servera používateľ musí zadať na ktorom porte bude server počúvať. Potom server čaká na dáta. Keď serveru príde správa s flagom I a F ( správa o inicializicií posielania súboru), server si musí zvoliť, kde uložiť súbor ( absolútnu cestu adresára, kde sa má súbor uložiť). Následne sa prijíma súbor.

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok 16 ukážka prijímania na serveri

## Výmena servera s klientom

Výmena servera s klientom prebieha po odosielaní správy alebo súboru. Vtedy si server môže vymeniť úlohy s klientom. Má na to 3 sekundy po výmene správy alebo súboru. Počas tých troch sekúnd musí server stlačiť ctrl + c. Ak stlačí, tak sa úlohy vymenia server bude klientom a klient serverom. V prípade, že nestlačí, tak sa vypíše nestihol si pokračujeme.

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok 17 stlačenie ctrl+c

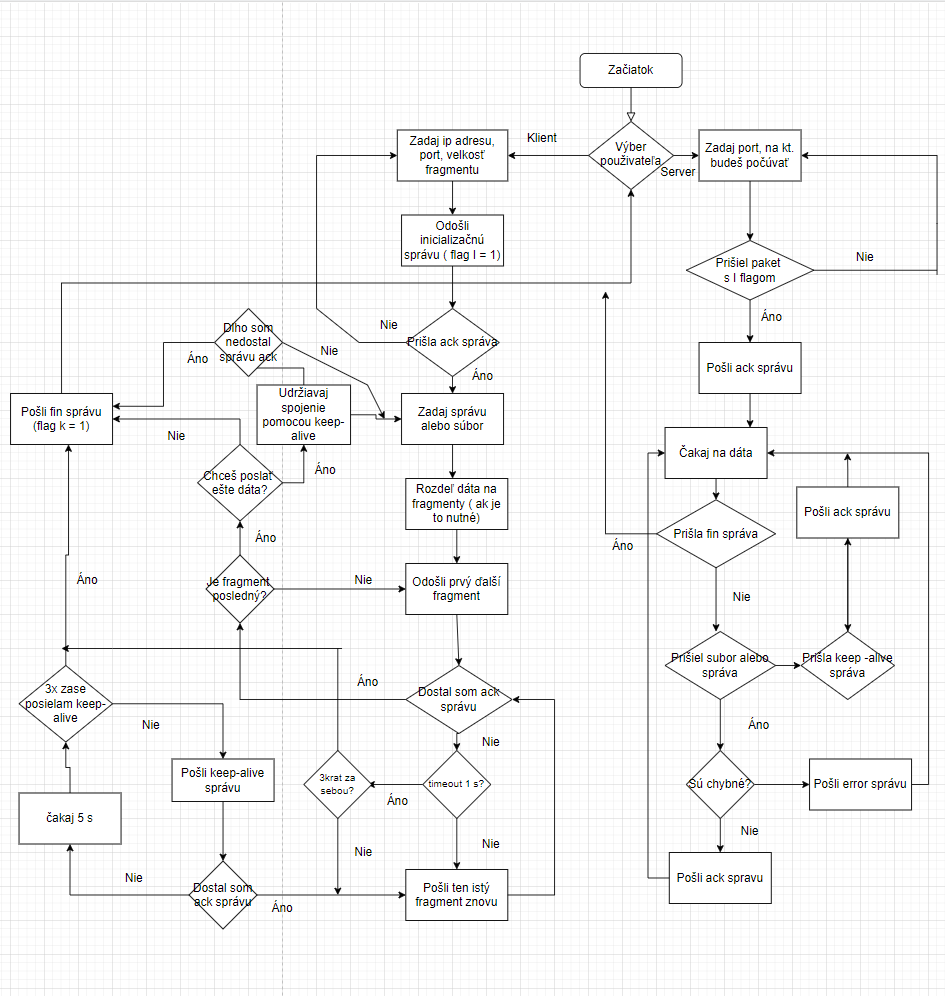
Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok 18 klient sa zmeni na server

Výmena môže byť vykonaná aj tak, že klient ukončí spojenie a potom si v menu prehodia úlohy. Táto druhý typ metódy je na inicializovaný na strane klienta.

## DIAGRAM SPRACOVÁVANIA KOMUNIKÁCIE



Obrázok 19 finálny diagram

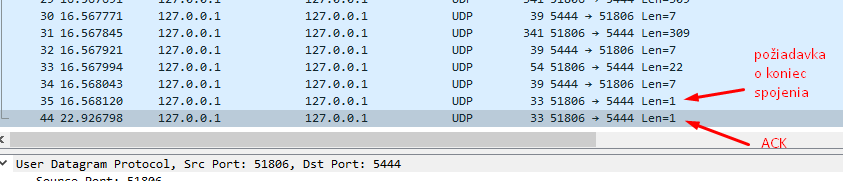
## ZOBRAZENIE VO WIRESHARK

Komunikácia je otestovaná na loopbacku. Pre zachytávanie paketov bol použitý program wireshark. Pre filtráciu tejto komunikácie je treba do filtra zadať názov protokolu „sip“.

Obrázok, na ktorom je stôl

Automaticky generovaný popis

Obrázok 20 inicializácia a prenos vo Wiresharku



Obrázok 21 koniec spojenia

## ZÁVER

Program bol úspešne implementovaný v pythone. Program bol testovaný na jednom počítači, kde jedna inštancia programu bola spustená v pyCharm a druhá v príkazovom riadku. Otestované boli súbory rôznej veľkosti až do desiatok MB a správy rozličného druhu. Simulácia chyby bola nastavená na 20 percentnú pravdepodobnosť. Server môže inicializovať výmenu úloh ako aj klient. Bola taktiež implementovaná stop & wait ARQ metóda, ktorá čaká vždy na ACK.