Slovenská technická univerzita

Fakulta informatiky a informačných technológii
Ilkovičova 3, 812 19 Bratislava

Princípy komunikačných systémov

Zadanie č. 1

Adam Žúrek

Cvičiaci: Ing. Richard Roštecký
Študijný odbor: Informatika
Ročník: 2. Bc
Akademický rok: 2016/2017

OBSAH

1. Znenie zadanie ... 3
1.1. Zadanie úlohy ... 3
1.2. Program musí mať nasledovné vlastnosti (minimálne): ... 3
1.3. Zadanie sa odovzdáva: ... 3
2. Analýza: ... 4
2.1. Enkapsulácia a hlavičky ... 4
2.2. Hlavné vlastnosti protokolu UDP z pohľadu zadania ... 4
2.3. Chyby v prenose a znovuvyžiadanie rámca ... 5
2.4. Veľkosť fragmentu ... 5
3. Špecifikácia požiadaviek ... 5
4. Návrh riešenia ... 6
4.1. Návrh vlastného protokolu ... 6
4.2. Vývojový diagram ... 6
4.3 Používateľské rozhranie ... 7
5. Implementácia ... 8
5.1 Zmeny oproti návrhu ... 8
5.2 Použité knižnice a funkcie ... 9
5.3 Organizácia návrhu ... 10
5.4 Používateľská príručka - GUI ... 11
6. Zhodnotenie ... 11
7. Bibliografia ... 11

1. ZNENIE ZADANIE

1.1. ZADANIE ÚLOHY

Nad protokolom UDP (User Datagram Protocol) transportnej vrstvy sieťového modelu TCP/IP navrhnite a implementujte program, ktorý umožní komunikáciu dvoch účastníkov v sieti Ethernet, teda prenos správ ľubovoľnej dĺžky medzi počítačmi (uzlami). Program bude pozostávať z dvoch častí – vysielacej a prijímacej. Vysielací uzol pošle správu inému uzlu v sieti. Predpokladá sa, že v sieti dochádza k stratám dát. Vysielajúca strana rozloží správu na menšie časti - fragmenty, ktoré samostatne pošle. Správa sa fragmentuje iba v prípade, ak je dlhšia ako max. veľkosť fragmentu. Veľkosť fragmentu musí mať používateľ možnosť nastaviť aj menší ako je max. príspustný pre trasportnú vrstvu. Po prijatí správy na cieľovom uzle tento správu zobrazí. Ak je správa poslaná ako postupnosť fragmentov, najprv tieto fragmenty spojí a zobrazí pôvodnú správu. Komunikátor musí vedieť usporiadať správy do správneho poradia, musí obsahovať kontrolu proti chybám pri komunikácii a znovuvyžiadanie rámca, vrátane pozitívneho/negatívneho potvrdenia. Pri nečinnosti komunikátor automaticky odošle paket pre udržanie spojenia. Odporúčame riešiť cez vlastne definované signalizačné správy.

1.2. PROGRAM MUSÍ MAŤ NASLEDOVNÉ VLASTNOSTI (MINIMÁLNE):

- Pri posielaní správy musí používateľovi umožniť určiť cieľovú stanicu.
- Používateľ musí mať možnosť zvoliť si max. veľkosť fragmentu.
- Obe komunikujúce strany musia byť schopné zobrazovať: - poslanú resp. prijatú správu, - veľkosť fragmentov správy.
- Možnosť odoslať chybný rámec
- Možnosť odoslať dáta zo súboru a v tom prípade ich uložiť na prijímacej strane do súboru

1.3. ZADANIE SA ODOVZDÁVA:

- Návrh riešenia
- Predvedenie riešenia v súlade s prezentovaným návrhom

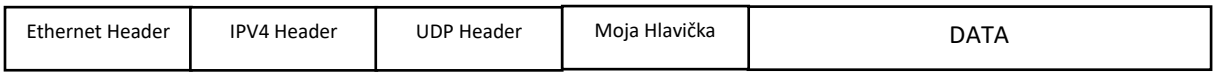
Program musí byť organizovaný tak, aby oba komunikujúce uzly mohli byť (nie súčasne) vysielačom a prijímačom správ.

2. ANALÝZA:

2.1. ENKAPSULÁCIA A HLAVIČKY

Komunikácia prostredníctvom siete bude využívať Ethernet ako protokol vrstvy sieťového rozhrania. Nad Ethernet protokolom bude používaný protokol IP (IPv4), ktorý je súčasťou sieťovej vrstvy. Z transportnej vrstvy bude (nad protokolom IP) pracovať protokol UDP, tak ako je to definované v zadaní.

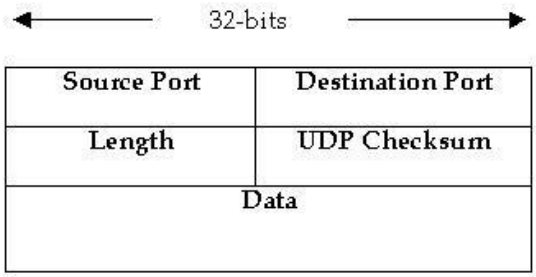
Zadanie máme vypracovať nad protokolom UDP, pričom máme implementovať do rozhrania výber veľkosti fragmentu používateľom. K samotnej fragmentácií dôjde v prípade, že veľkosť správy je väčšia ako maximálna veľkosť fragmentu. Preto som sa rozhodol vytvoriť vlastnú hlavičku protokolu, ktorá bude zahrnutá ako súčasť dát UDP protokolu.



Obr. 1 Grafické zobrazenie vnárania protokolov

2.2. HLAVNÉ VLASTNOSTI PROTOKOLU UDP Z POHĽADU ZADANIA

Používateľský datagramový protokol, skr. UDP, je tzv. "nespoľahlivý" protokol z balíka internetových protokolov. UDP protokol prenáša datagramy medzi počítačmi v sieti, ale na rozdiel od TCP nezaručuje, že prenášaný paket sa nestratí, že sa nezmení poradie paketov, ani že sa niektorý paket nedoručí viackrát. Vďaka tomu je UDP pre ľahké a časovo citlivé účely rýchlejšie a efektívnejšie.



Obr. 2 Grafické zobrazenie hlavičky UDP protokolu

2.3. CHYBY V PRENOSE A ZNOVUVYŽIADANIE RÁMCA

Ako som už spomenul, UDP protokol nezaručuje správne doručenie, resp. doručenie, fragmentov adresátovi. Rovnako nezaručuje výmenu paketov, resp. ich dvojnásobné doručenie.

Adresát jednoduchým spôsobom (napr. podľa checksumu) zistí, že prijal poškodenú správu. Následne prebehne znovuvyžiadanie rámca od odosielateľa. Tento proces znovuvyžiadania rámca prebehne po odoslaní správy klienta. Server pošle klientovi charakteristické dáta, ktoré značia klientovi, aby poslal správu znova.

2.4. VEĽKOSŤ FRAGMENTU

Štandardná veľkosť Ethernet rámca odosielaného po sieti je v rozsahu od 46-1518B (bajtov). Veľkosť 1518B v sebe zahŕňa aj veľkosť hlavičky Ethernet rámca (18B), t.j. veľkosť dát v rámci je 1500B. V rámci dát Ethernet protokolu bude zahrnutá hlavička (dáta) IP protokolu, ktorej štandardná veľkosť je 20B. To znamená 1480B pre dáta IP protokolu. Nad IP protokolom pracuje UDP protokol, ktorého hlavička má veľkosť 8B. Maximálnu veľkosť fragmentu teda bude používateľ môcť zadať do 65508B.

3. ŠPECIFIKÁCIA POŽIADAVIEK

- Program by mal spĺňať minimálne požiadavky zadania a teda: používateľ by mal mať umožnené zadať cieľovú stanicu (server, s ktorým bude komunikovať) a taktiež zvoliť si maximálnu veľkosť fragmentu.

- Architektúra programu bude vo forme server-klient, pričom pre testovanie programu bude potrebné mať dve zariadenia obsahujúce sieťovú kartu pripojené medzi sebou v sieti napr. pomocou ethernet kábla.

- Program bude naprogramovaný v jazyku Java a bude mať používateľské rozhranie pre aplikáciu postavené v knižnici Java.Swing.

- Posielanie správ za pomoci UDP protokolu je zrealizované za pomoci knižnice Java.NET.

- Taktiež program bude schopný odoslať chybný rámec a poradiť si s jeho opätovným poslaním.

4. NÁVRH RIEŠENIA

4.1. NÁVRH VLASTNÉHO PROTOKOLU

Hlavička bude obsahovať päť typov informácií o pakete. Pre každú informáciu bude vyhradená pamať štyri bajty a teda celkovo bude hlavička mať 20 bajtov. Typy informácií sú:

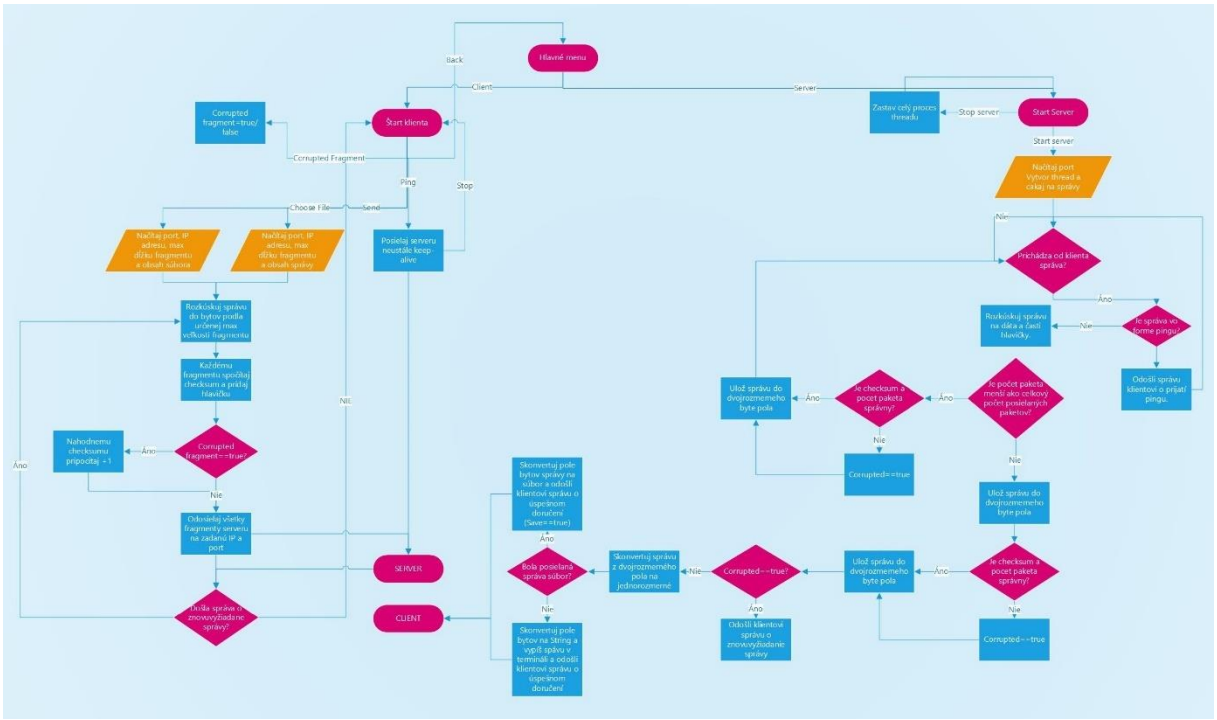
- CheckSum – premenná typu int informujúca o možnom poškodení dát v celom pakete.
- Veľkosť dát – premenná typu int informujúca o veľkosti posielaných dát v bytoch.
- Poradie paketu – premenná typu int informujúca o poradí paketu správy.
- Počet paketov – premenná typu int informujúca o celkovom počte paketov správy.
- Typ súboru – objekt typu String uchovávajúci príponu súboru.

Adresát si vie z veľkosti dát prirátať veľkosť hlavičky. Tým dostane celkovú veľkosť prijatého paketu.

CheckSum	Veľkosť dát	Poradie paketu	Počet paketov	Typ súboru	DATA

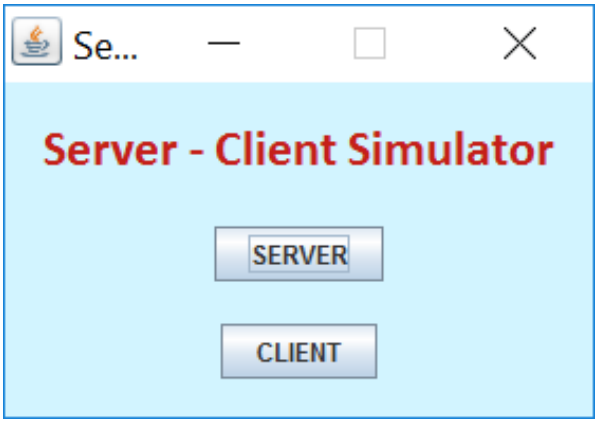
Obr. 3 Grafické zobrazenie návrhu mojej hlavičky

4.2. VÝVOJOVÝ DIAGRAM



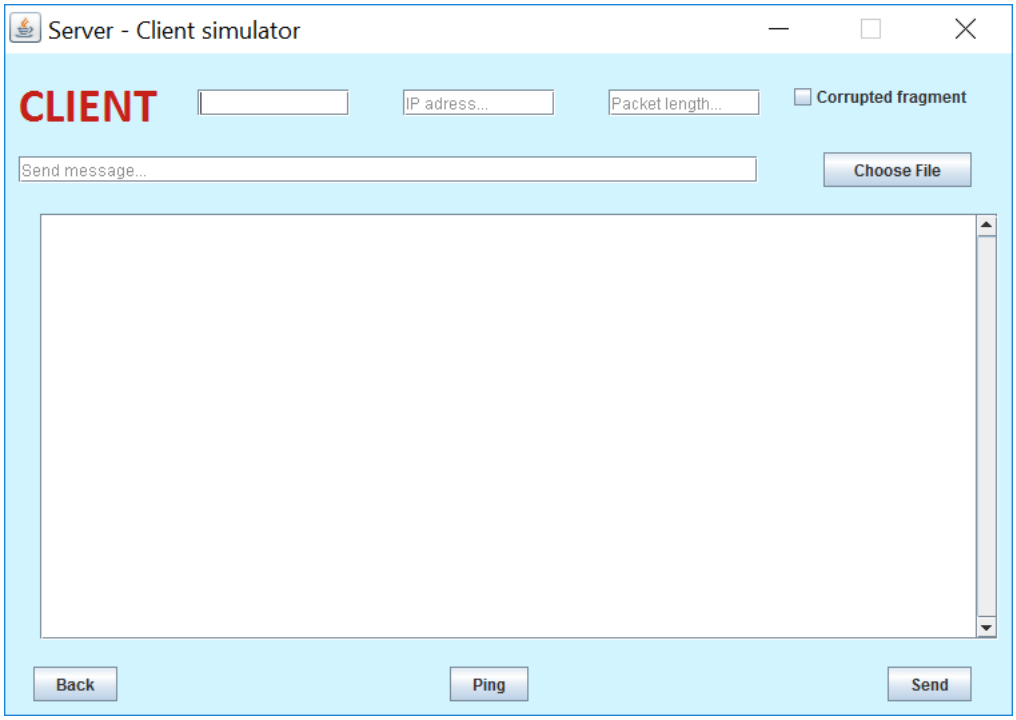
4.3 POUŽÍVATEĽSKÉ ROZHRANIE

Celé grafické rozhranie bude naprogramované v knižnici Java.Swing, kde hneď po spustení si bude môcť v hlavnom okne aplikácie používateľ vybrať, či chce spustiť klienta alebo server.



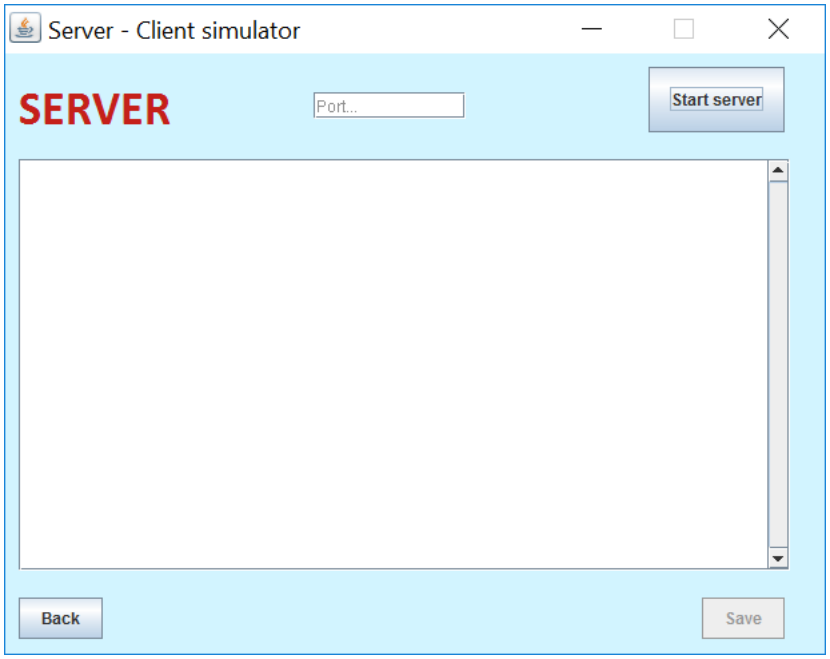
Obr. 4 Grafické zobrazenie hlavného okna aplikácie

Akonáhle si používateľ zvolí, či chce spustiť server alebo klienta, podľa voľby sa mu otvorí okno.



Obr. 5 Grafické zobrazenie okna klienta.

Okno klienta v sebe bude obsahovať kolónku na zmenu portu , IP adresy servera, veľkosť fragmentu v bytoch a správy. Používateľ dostane taktiež možnosť po zaškrtnutí „Corrupted fragment“ poslať chybný rámec. Ďalej si bude môcť vybrať typ súboru kliknutím na tlačidlo „Choose File“, aký serveru pošle, alebo bude môcť napísať klasickú správu. V terminále bude výpis správ, ktoré server pošle klientovi naspäť. Tlačidlo „back“ sa vráti do hlavného okna, „ping“ na spustenie keep-alive a „send“ na poslanie správy serveru.



Obr. 6 Grafické zobrazenie okna servera.

V okne servera sa bude taktiež dať prenastaviť port a terminál bude vypisovať správy, ktoré klient poslal serveru. Tlačidlo napravo „Start server“ bude spúšťať a vypínať čakanie servera na určitom porte. Tlačidlo „save“ sa sprístupní ihneď po úspešnom odoslaní súboru z klienta na server a po stlačení si používateľ bude môcť súbor uložiť. Tlačidlo „back“ ako u klienta bude vracať používateľa do hlavného okna.

Aplikácia bude fungovať aj na localhoste. Bude si stačiť otvoriť okno servera a klienta na jednom PC a pripojiť sa.

5. IMPLEMENTÁCIA

5.1 ZMENY OPROTI NÁVRHU

Vo finálnej verzii programu som zmenil veľa prvkov. Prerobil som hlavičku, kde som zmenil ID správy na veľkosť správy a pridal typ súboru, čím sa zväčšila jej veľkosť o 4bajty. Taktiež po implementácii grafického rozhrania, ktoré malo za následok lepšiu štruktúru programu som zmenil jeho logiku a tým prerobil vývojový diagram na finálnu verziu presne podľa zmenenej logiky môjho finálneho programu. Do sekcie používateľské rozhranie som použil reálne GUI z môjho programu namiesto náčrtu. No myslím si, že hlavná štruktúra a logika programu ostala naďalej nezmenená a hlavné idei z predošlej verzie ostali zachované.

5.2 POUŽITÉ KNIŽNICE A FUNKCIE

Medzi najhlavnejšie knižnice v mojom programe patria Javax.Swing, ktorá má na starosti celé grafické rozhranie a hlavnú štruktúru proramu. Java.net, ktorá umožnila môjmu programu predimplementovaný UDP protokol. Samozrejme na vstup a výstup programu a debug Java.io a na špeciálne typy objektov Java.utils a zopár ďalších.

V mojom programe používam veľa rôznych funkcií ako napríklad:

```
private static ArrayList<byte[]> SplittedByte(byte[] byteToSplit, int
length)
```

Slúži na nasekanie pola bytov do byte arrayu podľa určenej dĺžky fragmentu.

```
private String getFileExtension(File file)
```

Slúži na zistenie prípony súboru z file path a uloženie jej do Stringu.

```
private static int BytetoInt(byte[] bytes)
```

Slúži na prekonvertovanie poľa bytov na int.

```
private static byte[] copySmallArraysToBigArray(byte[][] smallArrays)
```

Slúži na prekonvertovanie dvojrozmerného poľa bytov do jedného poľa bytov.

```
private static byte[] trim(byte[] bytes) {
```

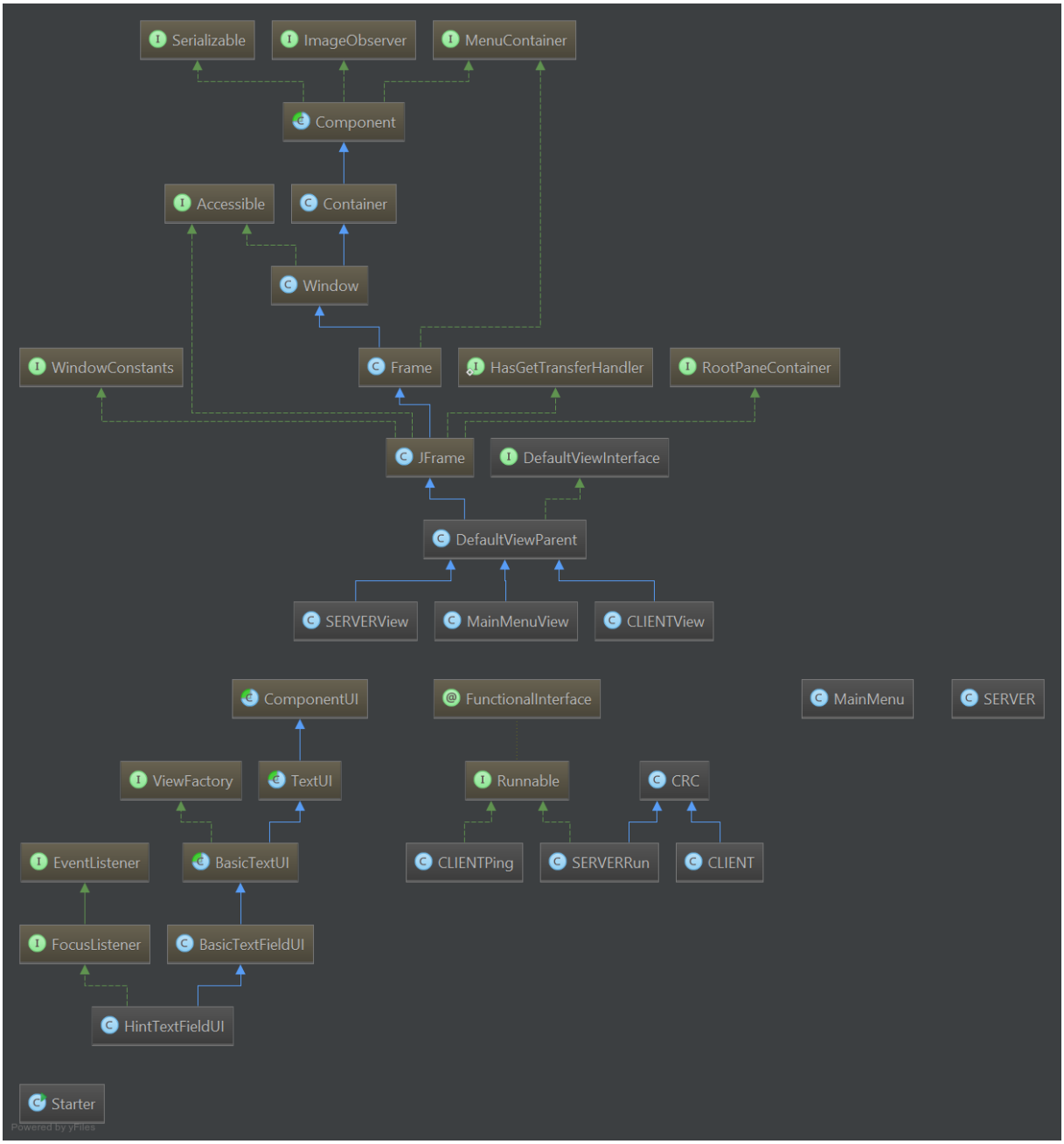
Slúži na odstránenie koncových núl z poľa bytov.

```
protected static int CRC(byte[] buf, int len)
```

Samozrejme funkcia CRC16, ktorej funkcia je z posielaných dát spraviť checksum a kontrolovať správnosť dát. Funguje na princípe riadiaceho polynóma stupňa x^{16} napríklad 0x1081 je hexadecimálne číslo s binárnou hodnotou „0001 0000 1000 0001“, ktoré sa používa ako maska pri algorytme, pomocou ktorej program s jednotlivými bytmi robí bytové operácie ako napríklad moja funkcia XOR s maskou 0xFFFF.

Tieto funkcie majú veľmi dôležitú úlohu v mojom programe. Ostatné funkcie ako gettery, settery nebudem spomínať.

5.3 ORGANIZÁCIA NÁVRHU



Trieda Starter obsahuje main, ktorý spúšťa triedu MainMenu, ktorej view je trieda MainMenuView. MainMenu zavolá podla vstupu používateľa triedu CLIENT alebo SERVER, ktoré majú takisto svoje view triedy CLIENTView a SERVERView. Všetky view triedy dedia od parenta DefaultViewParent, ktorú som si predpripravil.

Z class diagramu je zrejmé, že triedy CLIENTPing a SERVERRun sú Thready a že ServerRun aj CLIENT potrebujú kontrolovať CRC16, ktoré zdedia z triedy CRC. Trieda HintTextFieldUI slúži na priesvitný hint v textFieldoch.

5.4 POUŽÍVATEĽSKÁ PRÍRUČKA - GUI

Ihneď po spustení sa otvorí hlavné okno (viz.4.3 – používateľské rozhranie) s možnosťou vybrania si servera alebo klienta.

Klient má 3 kolónky, kde treba podľa sivého textu zadať port,IP adresu alebo veľkosť fragmentu. Program bude sám navádzať používateľa, či zadal všetko. Veľká kolónka slúži na napísanie správy serveru. Ak po napísaní správy stlačíte tlačidlo „send“ práva sa odošle serveru. Ak chcete poslať súbor, treba stlačiť na „Choose File“, vybrať si súbor, aký chcete odoslať a dať save. Na keep-alive slúži button ping.

Server má kolónku na zadanie portu a tlačidlo na jeho zapnutie/vypnutie. Tlačidlo „save“ sa sprístupní po úspešnom prijatí súbora od klienta.

Tlačidlá „back“ v oboch prípadoch vrátia používateľa do hlavného menu.

6. ZHODNOTENIE

Program je funkčný, no bohužial kvôli nedostatku času som nestihol implementovať a optimalizovať všetko ako by som si prial. Hlavička sa dá zmenšiť a typ súboru zakódovať do CRC. Klient by mal bežať na samostatnom threade, čím by sa zefektívnilo posielanie správ medzi klientom a serverom a zefektívnila sa rýchlosť odosielania a opravovania chýb. Zlepšil by som vzhľad grafického rozhrania a tým aj orientáciu v programe. Program by kontroloval viac chýb, ako napríklad premiešanie jednotlivých správ medzi sebou, nielen fragmentov jedenej správy.

Napriek nedostatkom si myslím, že program splnil minimálne požiadavky, dokonca má funkcionality naviac. Program je funkčný v praxi a testovaný na viac operačných systémoch ako Windows, MAC, Linux za pomoci pripojenia cez ethernet alebo wifi.

7. BIBLIOGRAFIA

- TANENBAUM, A S. Computer networks. Upper Saddle River: Pearson Education Limited, 2003. 891 s. ISBN 0-13-038488-7.

- JAMES F. KUROSE, KEITH W. ROSS Computer Networking: A Top-Down Approach, 4th edition. Pearson Education, Inc., ISBN 0-321-51325-8, 878 s., 2008

- ODOM, WENDELL. Cisco CCENT/ CCNA ICND1 100-101 Official Cer Guide. Pearson Education. Pp. Ch. 1. ISBN 978-1-58714-385-4, 2013