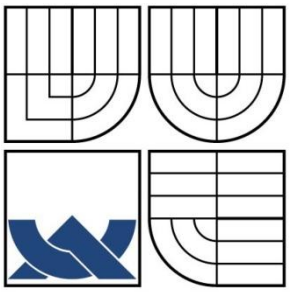
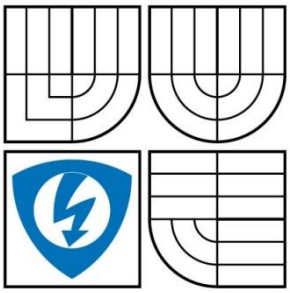


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKACNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

NÁSTROJ PRO STATISTICKÉ VYHODNOCENÍ PRŮBĚHU SIMULACE KNIHOVNY JSIMLIB4

TOOL FOR STATISTICAL EVALUATION OF JSIMLIB4 SIMULATION LIBRARY

DIPLOMOVÁ PRÁCE
THESIS WORK

AUTOR PRÁCE — BC. ŠTEFAN PIVODA
AUTHOR

VEDOUCÍ PRÁCE — ING. RADIM BURGET
SUPERVISOR

BRNO 2008

ABSTRAKT

Táto diplomová práca sa zaoberá analýzou simulácií počítačových sietí. Na simuláciu počítačových sietí sa využíva javovská knižnica JSimlib4, ktorá je jednoduchá, ale robustná knižnica, určená pre tvorbu jednoduchých, ale aj celkom komplexných simulačných modelov.

Na analýzu sietí bol navrhnutý štatistický nástroj, ktorý bol realizovaný v programovacom jazyku Java. Aplikácia bola vyvíjaná vo vývojom prostredí EasyEclipse a bola navrhnutá ako aplikácia platformy Eclipse Rich Client Platform. Jedná sa o relatívne novú technológiu, ktorú však kvôli jej priaznivým vlastnostiam využíva čoraz viac vývojárov.

Štatistický nástroj je rozdelený na 3 časti:

- vytváranie logovacieho súboru,
- vytvorenie filtra,
- vytvorenie a zobrazenie výstupu.

Štatistický nástroj vytvorí záznamy do logovacieho súboru pri každom prenose medzi stanicami. Na prvom riadku tohto súboru je komentár, ktorý popisuje výzor záznamov. Každý záznam obsahuje zdrojovú a cieľovú IP adresu s portom, čas a množstvo prenesených bytov, použitý protokol a nakoniec smer prenosu. Každá položka v riadku je oddelená dvojbodkou.

Nástroj dokáže vyhodnotiť prenos medzi stanicami. Na analýzu potrebuje filtračné hodnoty, podľa ktorých má analýzu vykonať. Toto filtrovanie môže byť vykonané buď pomocou IP adries alebo názvov staníc. Ďalším spôsobom vytvorenia filtrovania je načítanie týchto údajov z konfiguračného súboru.

Po definovaní filtra a spustenia analýzy, štatistický nástroj vytvorí buď diagram alebo súbor na ďalšiu prácu so získanými hodnotami. Najprv program prečíta celý logovací súbor a porovná každú zadanú IP adresu alebo názov porovná s IP adresami alebo názvami v záznamoch logovacieho súboru. V prípade, že nájde vyhovujúci záznam, prečíta celý riadok a pridá počet prenesených bytov a čas do konečného výsledku simulácie. Tento výsledok je potom zobrazený v jednom z podporovaných formátov. Ak aplikácia nájde viac prenosov v rovnakom časovom okamihu, tak spočíta súčet prenesených bytov, ktorý pridá do výsledku. Po prečítaní celého logovacieho súboru a porovnaní každej IP adresy zobrazí výstupný formát na dolnej časti aplikácie.

Aplikácia podporuje 2 typy výstupných formátov.

Prvým z nich je diagram, ktorý je vytvorený pomocou javovskej knižnice JFreechart a znázorňuje počet prenesených bytov v čase. Na x-ovú os je vynesený čas a na y-ovú počet prenesených bytov.

Druhým výstupným formátom je súbor, ktorý umožňuje prácu so získanými hodnotami aj v iných programoch. Tento súbor sa dá načítať v programoch Matlab alebo Octave. V prvom riadku tohto súboru sú uložené časové hodnoty a druhý riadok obsahuje počet prenesených bytov. Na poslednom riadku je príkaz, ktorý vykreslí graf z týchto hodnôt vo vyššie spomenutých programoch.

ABSTRACT

This thesis work deals with the analysis of simulation of computer networks. A statistical tool was designed for evaluating a communication between stations in the computer network.

JSimlib4 Java library was used for the simulation of computer networks. It is a lightweight but robust simulation library, designed for creating simple or even quite complex simulation models of distributed systems.

The Statistical tool was written in Java programming language and developed in an EasyEclipse integrated development environment. It was designed as an Eclipse Rich Client Platform application. Eclipse Rich Client Platform is a relatively new technology, which has a lot of favorable properties.

The Statistical tool can be divided into 3 parts:

- creating a log file,
- creating a filter,
- creating/showing an output.

The Statistical tool creates records into a log file during every communication between stations. This log file contains a comment on the first line. This comment describes the definition of records. Every record contains 2 IP addresses with the used ports at the beginning of the record, then the time, the amount of bytes, the protocol and the direction of communication. Every item in the record is divided by a colon.

The Statistical tool can evaluate a communication between stations. At first, it has to define a filter according to which it will assess the suitable stations. It can be created either by adding IP addresses or names of the stations onto a list on a workbench. Filter can be loaded from a file that already includes these IP addresses and other information too.

After defining the filter, the Statistical tool can create either a diagram or a file. At first, the Statistical tool reads the log file line by line and compares the IP addresses, which were added to the filter with the IP addresses in a log file. When the Statistical tool finds a complying IP address, it reads the whole line and adds an amount of bytes defined on that line to the final diagram that is shown after reading the whole log file. In the case of finding 2 or more complying IP addresses that sent the data at the same time, an amount of these transfers is calculated by adding them and this amount is then shown in a diagram. After reading the whole log file a diagram or a file is shown at the bottom of the window.

The diagram is created by a Java library JFreechart and it shows the amount of transferred bytes. The x-axis represents the time and the y-axis represents the amount of transferred bytes.

The created file is for next work with the calculated data and it can be loaded for example by Matlab or Octave. The first line in the file represents the time for axis x and the second one represents the amount of transferred bytes for axis y. These lines are followed by a command "plot(x,y)" for drawing a diagram.

KĽÚČOVÉ SLOVÁ

Štatistický nástroj; Jsimlib4 simulačná knižnica; Platforma Eclipse RCP; Java; EasyEclipse; Window Builder Pro; SVN;

KEYWORDS

Statistical tool; Jsimlib4 simulation library; Eclipse Rich Client Platform; Java; EasyEclipse; Window Builder Pro; SVN;

BIBLIOGRAFICKÁ CITACE PRÁCE

PIVODA, Š. *Nástroj pro statistické vyhodnocení průběhu simulace knihovny JSimlib4.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 64 s. Vedoucí diplomové práce Ing. Radim Burget.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Nástroj pro statistické vyhodnocení průběhu simulace knihovny JSimlib4“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.“

V Brně dne …… ……..

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu diplomové práce Ing. Radimovi Burgetovi, za velmi užitečnou metodickou pomoc a cenné rady při zpracování diplomové práce.

V Brně dne ……. ……...……..

(podpis autora)

OBSAH

ZOZNAM OBRÁZKOV

ÚVOD

Počítačové siete majú významný vplyv na náš každodenný život. Umožňuje ľuďom komunikovať, spolupracovať a reagovať spôsobom, ktorý si pred pár rokmi nedokázali ani predstaviť. Siete a najmä Internet využívame rôznymi spôsobmi, napríklad ako rôzne webové aplikácie, IP telefonovanie, video konferencie, interaktívne hry, elektronické reklamy alebo vzdelávanie.

Pri rýchlom rozvoji nových technológií je neoddeliteľnou súčasťou ich návrhu aj softwarová simulácia a analýza, ktoré pomáhajú pri predvídaní možných problémov ešte pred nasadením týchto technológií do reálneho užívania. Tieto analýzy dokážu návrhárom ušetriť nielen zbytočné výdavky v prípade neúspešného návrhu, ale aj čas, ktorý by museli investovať do realizácie. Práve z tohto dôvodu je u firiem vyvíjajúcich sieťové technológie, protokoly a súčiastky samozrejmosťou ich otestovanie pred nasadením nových technológií do prevádzky.

Táto diplomová práca popisuje návrh a realizáciu štatistického nástroja, ktorý má pomáhať pri analýze počtu prenesených dát medzi stanicami v počítačových sieťach. Práca rozširuje knižnicu JSimlib4, ktorá slúži na simuláciu sietí. Pretože je napísaná v programovacom jazyku Java, ktorý je okrem iného známy aj svojou nezávislosťou na operačnom systéme, knižnicu spolu s modulom je možné využívať pod každým operačným systémom, nezávisle od platformy.

Praktická časť diplomovej práce sa skladá zo štyroch hlavných častí. Prvú časť tvorí logovanie základných údajov prenosu a staníc, medzi ktorými bol prenos uskutočnený. Ďalšími dvoma časťami je filtrovanie vhodných záznamov a výpočet hodnôt, ktoré sú zapísané do výstupného súboru. Výstupom z modulu je obrázok PNG znázorňujúci prenesené byty v čase alebo súbor vo formáte *.m. Tento súbor obsahuje maticu a príkaz na vykreslenie prenesených bytov pre programy Matlab a Octave. Poslednou časťou tejto práce bol návrh grafického rozhrania aplikácie.

V prvej kapitole sú popísané prostriedky, ktoré boli využité počas vývoja štatistického nástroja. Nájdeme v nej popis simulačnej knižnice JSimlib4 a všeobecnú charakteristiku programovacích jazykov C++ a Java. Tieto dva programovacie jazyky majú popísané výhody a nevýhody a sú vzájomne porovnané. Popisuje aj spôsoby tímovej práce a čitateľ sa zoznámi s potrebnými nastaveniami pre tento druh práce. Vznik, prostriedky a jednotlivé prvky platformy Eclipse RCP sú obsiahnuté v podkapitole s totožným názvom. Druhá kapitola

popisuje realizáciu štatistického nástroja – požiadavky kladené na tento nástroj, spôsob zaznamenávania logovacích záznamov zo simulácie, filtrovanie vhodných záznamov a zobrazenie výstupných súborov. Ďalej zoznamuje čitateľa s procesom analýzy, od vytvorenia logovacích záznamov až po zobrazenie výstupných súborov. V tejto kapitole je popísaný aj zvolený spôsob návrhu grafického užívateľského rozhrania štatistického nástroja. Kapitola s názvom Zloženie Eclipse RCP aplikácie je všeobecným popisom jej konštrukčných prvkov. Predposledná kapitola slúži v istej miere ako návod k aplikácii. Popisuje jednotlivé časti na grafickom rozhraní, ich funkciu a prácu s nástrojom. V závere sú zhrnuté vedomosti získané počas riešenia diplomovej práce a jej následná rekapitulácia.

1 POPIS VYUŽITÝCH PROSTRIEDKOV

1.1 Simulačná knižnica JSimlib4

Simulačná knižnica JSimlib4 je jednoduchá, ale robustná knižnica, určená pre tvorbu jednoduchých, ale aj celkom komplexných simulačných modelov distribuovaných systémov. Knižnica sa líši od ostatných dostupných simulačných knižníc svojou jednoduchou použiteľnosťou, voľne šíriteľným zdrojovým kódom a hlavne tým, že bola navrhnutá tak, aby mohla byť jednoducho premyslená do podoby reálnej distribuovanej knižnice.

Je to knižnica diskrétnych udalostí, ktorá sa zaoberá modelovaním systémov, ktoré sú vyvíjané priebežne. Je napísaná v Jave a dokáže ju využívať jednoducho hocikto, kto pozná tento programovací jazyk. Ak má vývojár základné vedomosti z Javy, bude môcť ihneď, bez akéhokoľvek ďalšieho vzdelávania napísať simulačný model. Užívateľ tejto knižnice má tú výhodu, že môže využívať databázu voľne dostupných zdrojových kódov. Kód každého simulačného modelu je veľmi podobný reálnej aplikácii a môže byť jednoducho pochopený z reálnej situácie. Simulácia je založená na procese, ktorý je zastupovaný inštanciou triedy. Táto trieda by mala byť dedičom triedy Proc a musí mať definovanú metódu action(). Z dôvodu vyhovieť podobnosti reálnej aplikácie do simulačnej knižnice boli implementované metódy `sleep()` a `wait()`.

Jadro simulačnej knižnice môžeme rozdeliť na 4 časti: simulácia, procesy, kalendár a udalosti. Simulácia funguje ako mozog knižnice. Rozhoduje o tom, kedy proces začne pracovať, kedy skončí, kedy zaspí, kedy bude čakať na ďalší proces alebo kedy je prerušený z čakania. Kým kalendár nie je prázdny, simulácia ho žiada o sadu udalostí E z ďalšieho časového okamihu. Tieto udalosti sú postupne odstránené z kalendára. Kalendár je vlastne skladom, kde sú uložené všetky udalosti a požiadavky podľa času, kedy majú byť vykonané. Kalendár môže na požiadavku procesu vrátiť naraz aj viac udalostí, keď pre žiadaný časový okamih obsahuje súčasne viac udalostí. Na zaistenie rýchlej simulačnej realizácie je kalendár implementovaný s použitím abstraktnej dátovej štruktúry Red-Black stromu [1], ktorý zaistí správne zaradenie udalostí a rýchlosť prístupu k záznamom. Udalosť je relatívne jednoduchá dátová štruktúra, ktorá obsahuje proces s plánovaným časom, kedy má byť udalosť vykonaná. Posledná časť – proces, definuje chovanie sa simulácie. Všetky procesy implementujú metódu `action()` a rozširujú triedu `Proc` alebo triedu `NetProc`, keď je potrebné sieťové

rozširovanie. Táto metóda definuje správanie sa procesu. Dokonca aj viacvláknové aplikácie sú podporované, ktoré sú realizované niekoľkými procesmi a spojené kompozičnou triedou `ProcComposite`. V metóde `action()` sú definované všetky simulačné udalosti, ako napríklad wait (`waitProc()`), sleep (`sleepProc(milisekundy)`), signal (`signalProc(iný proces)`). V prípade sieťového procesu je možné poslať paket rovnako ako v reálnej sieti.

1.2 Voľba programového prostredia

Prvým krokom pri návrhu tohto modulu bol výber programovacieho jazyka. V súčasnosti máme možnosť vybrať si z mnohých programovacích jazykov, z ktorých každý vyhovuje iným požiadavkám. Práve tieto fakty boli prvoradé pri výbere.

Okrem iných programovacích jazykov sa uvažovalo aj nad jazykom C++ [2, 3], ktorý je objektovo orientovaný programovací jazyk a je jedným z najrozšírenejších programovacích jazykov. Vyvinul ho Bjarne Stroustrup s kolegami v Bellových laboratóriách AT&T rozšírením jazyka C, ktorý mal byť rovnako efektívny ako C, umožniť podporu pre vytváranie knižníc, kombinovanie s inými jazykmi (C, Fortran, ...), ale zároveň mal obsahovať podporu pre objektovo orientované programovanie. Tento jazyk sa nazýval C with Classes (C s triedami) a prekladal sa pomocou špeciálneho preprocesoru Cpre do čistého C. Dôvodom bola hlavne dostupnosť a efektivita jazyka C a navyše nebolo nutné písať pre každú platformu zvlášť generátor kódu. V ďalších verziách s prekladačom sa podstatným spôsobom vyvíjal aj samotný jazyk, úpravami prešiel návrh objektovo orientovaného programovania, pridali sa výnimky, šablóny a nový jazyk sa premenoval na C++ a prekladač na Cfront.

C++ podporuje procedurálne, objektovo orientované a generické programovanie, čiže nie je čisto objektovým jazykom. Medzi jeho najhlavnejšie oblasti využitia patria matematické výpočty, telekomunikácia, systémové programovanie a programy na hry.

Napriek priaznivým vlastnostiam jazyka C++, bol nakoniec vybraný programovací jazyk Java [4, 5, 6], ktorý patrí tiež medzi najpoužívanejšie programovacie jazyky. Beží na virtuálnom stroji, ktorý zabezpečuje bezpečnosť a vysokú prenositeľnosť. Vďaka svojej prenositeľnosti je používaný pre programy, ktoré majú pracovať na rôznych systémoch, počínajúc čipovými kartami (platforma JavaCard), cez mobilné telefóny, rôzne zabudované zariadenia (platforma Java ME), aplikácie pre stolné počítače (platforma Java SE) až po rozsiahle distribuované systémy, ktoré pracujú na rade spolupracujúcich počítačov a sú

rozprestreté po celom svete (platforma Java EE). Tieto technológie sa ako celok nazývajú platforma Java a sú vyvíjané s otvoreným zdrojovým kódom, ktorého základnými vlastnosťami sú:

jednoduchý – Syntax jazyka je zjednodušenou (a trochu upravenou) verziou syntaxe jazyka C a C++.

objektovo orientovaný – Okrem ôsmych primitívnych dátových typov sú všetky ostatné dátové typy objektové.

distribuovaný – je navrhnutý pre podporu aplikácie v sieti (podporuje rôzne úrovne sieťového spojenia, práce so vzdialenými súbormi, umožňuje vytvárať distribuované klientské aplikácie a servery).

distribuovaný – miesto skutočného strojového kódu sa vytvára iba tzv. medzikód (bytecode). Tento formát je nezávislý na architektúre počítača alebo zariadenia. Program potom môže pracovať na ľubovoľnom počítači alebo zariadení, ktorý má k dispozícii prekladač Javy, tzv. virtuálny stroj Javy - Java Virtual Machine (JVM).

robustný – je určený pre písanie vysoko spoľahlivého softwaru – z tohto dôvodu neumožňuje niektoré programátorské konštrukcie (napríklad správa pamäti, príkaz `goto`, používanie ukazovateľov), ktoré sú častou príčinou chýb. Správa pamäti je realizovaná pomocou automatického zberateľa odpadkov (Garbage collectoru), ktorý automaticky vyhľadáva už nepoužívané časti pamäti a uvoľňuje ich pre ďalšie použitie.

bezpečný – má vlastnosti, ktoré chránia počítač v sieťovom prostredí, na ktorom je program spracovávaný, pred nebezpečnými operáciami alebo napadnutím vlastného operačného systému nepriateľským kódom.

nezávislý na architektúre – vytvorená aplikácia beží na ľubovoľnom operačnom systéme alebo ľubovoľnej architektúre. Na spustenie programu je potrebné iba to, aby bol na danej platforme inštalovaný správny virtuálny stroj. Podľa konkrétnej platformy sa môže prispôsobiť vzhľad a chovanie aplikácie.

viac úlohový – podporuje spracovávanie viacvláknových aplikácií.

dynamický – Java bola navrhnutá pre nasadenie vo vyvíjajúcom sa prostredí. Knižnica môže byť za chodu dynamicky rozširovaná o nové triedy a funkcie, a to ako z externých zdrojov, tak vlastným programom.

Nevýhodou oproti programovacím jazykom, ktoré vykonávajú tzv. statickú kompiláciu (napr. C++), je pomalší štart programov písaných v Jave, pretože prostredie musí najprv program preložiť a až potom ho môže spustiť.

Pri porovnávaní dvoch jazykov o C++ môžeme povedať, že je mocný jazyk s tendenciou na aplikácie a knižnice orientovaných na výkonnosť. Java bola navrhnutá pre väčšiu jednoduchosť, ale nie vždy poskytuje plný prístup k funkciám a výkonu platformy, na ktorej software beží. C++ má ale jednoduchšie štandardné knižnice, kým Java knižnice sú podstatne rozsiahlejšie pre štandardnú knižnicu. Okrem nezávislosti na platforme pre tento modul bol priaznivejší jazyk Java aj kvôli tomu, že už pôvodne bol navrhnutý na podporu sieťových operácií.

1.3 Voľba vývojového prostredia

Aplikácia bola vyvíjaná vo vývojovom prostredí (IDE) EasyEclipse [7].

EasyEclipse je odľahčená verzia vývojového prostredia Eclipse [10]. Využíva modulárnosť Eclipse, ktorú je možné pomocou zásuvných modulov (plug-in) používať na vývoj programov v niekoľkých programovacích jazykoch. To znamená, že EasyEclipse nepodporuje iba vývoj programov v Jave, ktorá je primárnym programovacím jazykom pre toto vývojové prostredie, ale aj programy napríklad v jazykoch C/C++ alebo PHP. EasyEclipse je šírený s voľným zdrojovým kódom, čo je jeho ďalšou výhodou. V základnej verzii obsahuje iba prostriedky pre vývoj štandardnej Javy ako napríklad prekladač, ladiaci program atď., ale neobsahuje nástroj pre vizuálny návrh grafického užívateľského rozhrania.

Projekt EasyEclipse bol vytvorený kvôli zbytočne rozsiahlemu obsahu Eclipse s mnohými komponentmi a zásuvnými modulmi, ktoré nie sú potrebné pri každom vývoji. Ďalšími nevýhodami Eclipse, ktoré sa EasyEclipse snaží odstrániť sú:

- príliš veľa podobných zásuvných modulov,
- nekompatibilita verzií medzi zásuvnými modulmi a platformou Eclipse alebo inými zásuvnými modulmi,
- komplikované menu a preplnené užívateľské rozhranie.

Cieľom EasyEclipse je vytvorenie vývojového prostredia založeného na Eclipse z hľadiska individuálneho vývojára alebo malého tímu. Individuálny vývojár potrebuje

špecifickú a robustnú sadu nástrojov obsiahnutú v najmenšom a najjednoduchšom balíčku. Práve túto vlatnosť EasyEclipse poskytuje.

Každá EasyEclipse distribúcia je prispôsobená na špecifické požiadavky a funkcionality pre dané vývojové prostredie – neobsahuje žiadne ďalšie zložitosti, ktoré nie sú potrebné. Všetky rozširujúce moduly sú dostupné na stránke EasyEclipse – http://www.easyeclipse.org/site/plugins/index.html.

Súčasne sú dostupné tieto distribúcie:

- EasyEclipse Expert Java,
- EasyEclipse Desktop Java,
- EasyEclipse Server Java,
- EasyEclipse Mobile Java,
- EasyEclipse for Plugins and RCP Apps,
- EasyEclipse for LAMP,
- EasyEclipse for PHP,
- EasyEclipse for Ruby and Rails,
- EasyEclipse for Python,
- EasyEclipse for C and C++.

Každá distribúcia je dostupná pomocou inštalačných súborov pre operačné systémy Microsoft Windows, Linux a Mac Os.

Počas vývoja štatistického nástroja bola využívaná distribúcia EasyEclipse for Plugins and RCP Apps [8]. Táto distribúcia stopercentne vyhovovala požiadavkám, pretože bez ďalších zásuvných modulov podporuje vývoj Rich Client Platform aplikácie.

1.4 SVN a spôsob pre tímovú prácu

Pri každom väčšom projekte je samozrejmosťou súčasná práca viacerých programátorov na jednom projekte. Vývoj tohto modulu nemôžeme považovať za väčší projekt, ale z dôvodu bezpečného zálohovania práce a možnosti ľubovoľného prístupu k projektu bol využívaný zásuvný modul Subclipse [9].

Subversion [23] je systém pre spravovanie a vytváranie verzií zdrojových kódov a je založený na princípe centrálneho repozitára. Subversion patrí do kategórie nástrojov pre prácu s verziami projektu (version control) a uspokojuje základné potreby pri spravovaní verzií. Je vyvíjaný firmou CollabNet, Inc. [24] a šírený pod licenciou, ktorá umožňuje jeho bezplatné komerčné použitie. Užívateľom sú k dispozícii aj zdrojové kódy. Skladá sa z dvoch hlavných častí – klientská a serverová časť.

Klientská časť poskytuje nástroje pre prácu s verziami priamo v pracovnom adresári a komunikáciu so serverovou časťou, ktorá sa stará o centrálny repozitár.

K centrálnemu repozitáru sa dá pristupovať rôznymi spôsobmi (lokálne, cez natívny protokol svn://, DAV). Existuje niekoľko klientských nástrojov, medzi ktoré patrí aj Subclipse.

Subclipse je zásuvný modul pre Eclipse a funguje ako tímový správca poskytujúci dostupnosť k podverziám projektu. Software je vyvíjaný pod licenciou voľného zdrojového kódu Eclipse Public License (EPL) 1.0 [12].

Jeho inštalácia [11] prebieha veľmi jednoduchým spôsobom pomocou manažéra aktualizácií (Update manager). Z menu treba vybrať položku *Pomoc (Help) / Aktualizácia softwaru (Software updates) / Nájdi a nainštaluj (Find and Install)*. Na nasledujúcom okne je potrebné zvoliť *Hľadať nové časti programu na inštalovanie (Search for new features to install)* a potom kliknúť na tlačítko *Ďalej (Next)*. Zobrazí sa okno, v ktorom treba zvoliť tlačítko *Nová vzdialená stránka (New Remote Site)* a zadať adresu http://subclipse.tigris.org/update_x, kde x zastupuje aktuálnu verziu modulu. Po zadaní adresy a stlačení tlačítka *Ok* je dôležité, aby bol vybraný modul k inštalácii. Po kliknutí na tlačítko *Dokončiť (Finish)* manažér aktualizácií prehľadá stránku a hľadá nové prvky. Neskôr sa objaví okno, kde je potrebné zaškrtnúť možnosť *Subclipse* a nasledovať príkazy, ktoré sa zobrazia v okne. Inštalácia sa spustí pomocou príkazu *Inštaluj všetko (Install All)*. Po dokončení inštalácie je treba EasyEclipse reštartovať, aby bolo možné pracovať s novým zásuvným modulom,.

Po nainštalovaní Subclipse a reštartovaní EasyEclipse je možné určiť cestu k centrálnemu repozitáru. To je možné dvoma spôsobmi.

Prvým z nich je využitie spravovania centrálneho repozitára ku ktorému sa dostaneme pomocou menu *Okná (Windows) / Ukáž pohľad (Show view) / Iné (Other) / SVN centrálny repozitár (SVN Repository)*. V spodnej časti Eclipsu sa otvorí okno. Cez lokálne menu po

zvolení *Nové (New) / Umiestnenie centrálneho repozitára (Repository location)* sme vyzvaní k zadaniu URL adresy centrálneho repozitára.

Po úspešnom načítaní centrálneho repozitára je možné spravovať zadaný projekt. V okne SVN centrálneho úložiska (SVN Repository) vyberieme projekt, ktorý chceme stiahnuť – zvyčajne hlavné vedenie (trunk) a v lokálnom menu zvolíme *Odbaviť* (*CheckOut)*. Keď projekt z centrálneho úložiska v pracovnom priestore Eclipse ešte nie je stiahnutý, ponúkne sa nám voľba *Odbaviť projekt pomocou sprievodcu nového projektu (Check out as a project configured using New Project Wizard)*. V prípade, že v pracovnom prostredí Eclipse už je projekt napojený na centrálny repozitár, ponúkne sa voľba *Odbaviť ako projekt v pracovnom prostredí (Check out as a project in the workspace).*

Druhý spôsob stiahnutia projektu je cez položku menu *Súbor (File)* / *Import* / *Odbaviť projekt z SVN (Checkout Projects from SVN).*

Po napojení projektu na repozitár, sú základné príkazy pre prácu so zálohami dostupné cez lokálne menu pomocou položky *Tím (Team)*. Môžeme vykonávať *Aktualizáciu (Update), Zapisovanie (Commit)* alebo vytvárať záložky a vetvy – *Tím (Team) / Vetva (Branch) / Označenie (Tag)*.

Pre prácu so zálohami existuje aj ďalší zásuvný modul s názvom SubVersive [13]. Je to novší zásuvný modul ako Subclipse, ale kvôli priaznivým predchádzajúcim skúsenostiam so Subclipse nebol dôvod použiť tento novší modul.

1.5 Platforma Eclipse RCP

Kto ešte nepočul výraz Eclipse, určite je zvedavý, čo tento výraz znamená. Eclipse je v prvom rade komunita ľudí, ktorá podporuje aplikácie s otvoreným zdrojovým kódom, vytvárajúca nástroje založené na Jave a infraštruktúru na pomoc prekonaniu možných problémov. Ich najhlavnejším produktom je vývojové prostredie (IDE) Eclipse Java [10]. Toto vývojové prostredie patrí medzi najuznávanejšie vývojové prostredia na svete. Je voľne dostupné na webovej stránke http://eclipse.org.

Pod vývojovým prostredím Eclipse je všeobecná nástrojová platforma, ktorá podporuje široký rozsah nástrojov pre programovacie jazyky a systémy od Javy, C, Python, webové technológie až po manipulácie s dátami a ich reportovanie. Konštrukčný model Eclipse znamená, že tieto nástroje môžu byť kombinované a integrované podľa potreby.

Pod touto nástrojovou platformou je Eclipse Rich Client Platform, ktorá je všeobecnou platformou pre spúšťanie aplikácií. Aj vývojové prostredie (IDE) Eclipse je práve takouto aplikáciou.

Výraz „rich client“ bol založený začiatkom 90-tych rokov so snahou vytvoriť klientské aplikácie využívajúce podobnosti Visual Basic a Delphi. Veľký rast v počte a popularite týchto klientských aplikácií vďačil čiastočne túžbe po „bohatých zážitkoch“.

„Bohatí klienti“ podporujú vysokokvalitné užívateľské skúsenosti poskytovaním bohatých, prirodzených užívateľských rozhraní ako aj vysokorýchlostné lokálne spracovanie.

Projekt Eclipse nebol založený so zámerom vytvoriť RCP. Jeho zámerom bolo vytvoriť platformu na integrovanie vývojových nástrojov. Eclipse ako RCP začal s verziou Eclipse 2.1. Výraz vznikol tak, že vývojové prostredie založené na Eclipse bolo profesionálne, dobre vyzerajúce a výkonné. Niekoľko smelých vývojárov si potom všimlo, že rovnaká konštrukcia (framework), ktorá robila nástroj atraktívnejším a jednoduchším na písanie, mohla byť využívaná aj na vytváranie všeobecných aplikácií.

Eclipse 3.0 bol hlavným krokom pre Eclipse k RCP. Prakticky všetky vzájomné závislosti súvisiace s vývojovým prostredím boli odstránené a mnoho odlišných častí grafického rozhrania bolo upravených. Základy pre dynamickú inštaláciu, odstránenie a aktualizáciu zásuvných modulov boli založené s predstavením behu programu (Runtime), ktorý je založený na OSGi [25].

S týmito vylepšeniami záujem o RCP rapídne rástol a začali sa objavovať komerčné aplikácie založené na RCP. IBM predstavil jeho Workplace™ produkty, NASA začala používať RCP na manažment, modelovanie a analýzu vesmírnych misií a RCP sa ukázal aj v mnohých ďalších aplikáciách.

Napriek relatívne krátkej existencii Eclipse RCP, bolo vytvorené množstvo aplikácií pracujúcich na tejto platforme. Niekoľko z nich je uvedených na webovej stránke Eclipse RCP http://eclipse.org/rcp.

Prostredie Eclipse je veľmi bohaté, ale je iba zopár konceptov a mechanizmov, ktoré sú preň základné. V nasledujúcej časti práce budú popísané niektoré najdôležitejšie koncepty a mechanizmy prostredia Eclipse. Môžeme medzi ne uvádzať:

- zásuvný modul,
- konštrukciu OSGi,

- beh programu Eclipse,
- aplikáciu,
- produkt,
- nízkoúrovňovú grafickú knižnicu SWT,
- súbor nástrojov JFace,
- pracovný panel.

Základnou funkčnou jednotkou v konštrukcii je zásuvný modul, jednotka modularity. V Eclipse môžeme všetko chápať ako zásuvný modul. Samotná RCP aplikácia je zbierkou rôznych zásuvných modulov a behu programu, na ktorom bežia. RCP vývojár zhromaždí kolekciu týchto zásuvných modulov zo základne Eclipse a z iných miest, ku ktorým dodá ním napísané zásuvné moduly. Tieto nové zásuvné moduly zahŕňajú samotnú aplikáciu a definíciu produktu.

Zásuvný modul je zbierkou súborov a zoznamu, ktorý popisuje modul a jeho vzťah k ostatným modulom. Zásuvné moduly môžu obsahovať zdrojový kód a iný obsah, ako napríklad obrázky, webové stránky alebo dokumentáciu. Obsahuje tiež súbor plugin.xml. Je to z historických dôvodov, pretože tento súbor bol domovským súborom pre informácie súvisiace so spúšťaním, ktoré sú teraz uložené v súbore MANIFEST.MF. Súbor plugin.xml zostal aj naďalej domovským súborom pre rozšírenia a deklarácie rozširujúcich bodov zásuvného modulu.

Výraz zásuvný modul slúži z historických dôvodov na odkazovanie na konštrukčné časti v Eclipse a je používaný všade v dokumentácii. V Eclipse medzi zásuvným modulom a balíkom nie je žiadny základný alebo funkcionálny rozdiel. Obidva sú mechanizmy na zoskupovanie, dodávanie a riadenie obsahu.

„Eclipse komponentový model zásuvných modulov“ (Eclipse plug-in component model) je založený na implementácii konštrukcie OSGi [25]. V skratke, špecifikácia OSGi tvorí konštrukciu pre definovanie, skladanie a spúšťanie konštrukčných častí alebo celého balíku. Na balíky sa môžeme pozerať aj ako na implementáciu zásuvných modulov.

Hlavnou úlohou konštrukcie OSGi je spojiť nainštalované zásuvné moduly a tým im umožniť spoluprácu a vzájomnú interakciu.

Špecifikácia OSGi poskytuje mechanizmus pre definovanie a spúšťanie oddelených konštrukčných častí. Beh programu Eclipse pridáva k nemu mechanizmus pre deklarovanie

vzťahov medzi zásuvnými modulmi, tzv. rozširovacie registre. Zásuvné moduly môžu byť sprístupnené pre rozšírenia alebo konfigurácie deklarovaním rozširovacieho bodu. Zásuvný modul v podstate hovorí „Ak mi dáte nasledujúcu informáciu, ja spravím ...“. Iné zásuvné moduly potom poskytnú požadovanú informáciu rozširovaciemu bodu vo forme rozšírenia.

V minulosti beh programu Eclipse (Eclipse Runtime) zahŕňal tiež model zásuvných modulov, ktorý bol neskôr posunutý dole na OSGi vrstvu. Zvyšok behu programu bol ponechaný na vrchu. Beh programu obsahuje niekoľko kľúčových mechanizmov, hlavne pre aplikačný model a rozširovacie registre. Rovnako ako konštrukcii OSGi a JVM aj behu programu Eclipse je nutné povedať čo má vykonať. Na spúšťanie Eclipse treba mať zadefinovanú aplikáciu.

Aplikácia je veľmi podobná metóde `main()`, ktorá sa využíva v bežnom Java programe. Beh programu po spustení nájde a spustí špecifikovanú aplikáciu. Aplikácie sú definované použitím rozšírenia. Rozšírenia aplikácie identifikujú triedu na použitie, ktorá má slúžiť ako hlavný vstupný bod. Pri spúšťaní Eclipse je možné určiť, ktorá aplikácia sa má spustiť. Po spustení aplikácie je Eclipse pod jej kontrolou. Pri ukončení aplikácie sa Eclipse zatvorí.

Pojem produkt je o úroveň nad aplikáciou. Užívateľ môže spustiť Eclipse definovaním aplikácie, ktorá však nezahŕňa značkovanie produktu (úvodná obrazovka a ikony) a rôzne ladenia (výber a konfiguračné súbory). Pojem produkt zhromažďuje toto rozšírenie informácií do jedného prevedenia, ktoré užívatelia môžu spúšťať.

SWT [26] je nízkoúrovňová grafická knižnica, ktorá poskytuje štandardné grafické rozhranie, ako napríklad lišty, menu, písma a farby. SWT poskytuje efektívny prístup k vlastnostiam grafického rozhrania operačného systému, na ktorom je implementovaný. Takto je dosiahnuté, že SWT je tenkou vrstvou na vrchu existujúceho operačného systému. Ozajstná výhoda SWT spočíva v tom, že všade kde sa to dá, používa prirodzený tvar prvkov. Toto umožňuje aplikáciám založeným na SWT mať prirodzený tvar na každom operačnom systéme.

JFace [27] je súbor nástrojov s triedami pre prácu s bežnými úlohami pre programovanie grafického rozhrania. JFace je nezávislý na systéme a je navrhnutý pre prácu so SWT. Zahŕňa mnoho komponentov grafického rozhrania ako napríklad registre písma a obrázkov, podporu textu, dialógy, konštrukcie pre výber a sprievodcov na hlásenie behu pri dlhých operáciách. Tieto a iné JFace štruktúry tvoria základ grafického rozhrania Eclipse.

Ako JFace dáva štruktúru pre SWT, pracovný panel (Workbench) poskytuje predstavenie a koordináciu pre JFace. Pracovný panel sa skladá zo záložiek (view) a editorov umiestnených na určitej konštrukcii. Kým JFace predstavuje akcie, výbery, sprievodcov a okná, pracovný panel poskytuje rozširovacie body, ktoré dovoľujú zásuvným modulom definovať tieto grafické prvky deklaratívne. Užívateľ môže chápať pracovný panel aj ako kolekciu okien, ktorá umožňuje užívateľom organizovať ich prácu v každom okne podobne ako to robia napríklad v každodennom živote, keď rovnaké dokumenty dávajú do jednej záložky a zhromažďujú ich na jednej kope.

Perspektíva je vizuálny kontajner pre kolekciu editorov a záložiek, v ktorom sú užívateľovi zobrazené informácie z aplikácie. Editory boli navrhnuté na prácu s aplikáciou.

Táto minimálna kolekcia prvkov potrebných k vytvoreniu Rich Client Platform aplikácií je všeobecne známa ako platforma Eclipse RCP.

2 REALIZÁCIA ŠTATISTICKÉHO NÁSTROJA

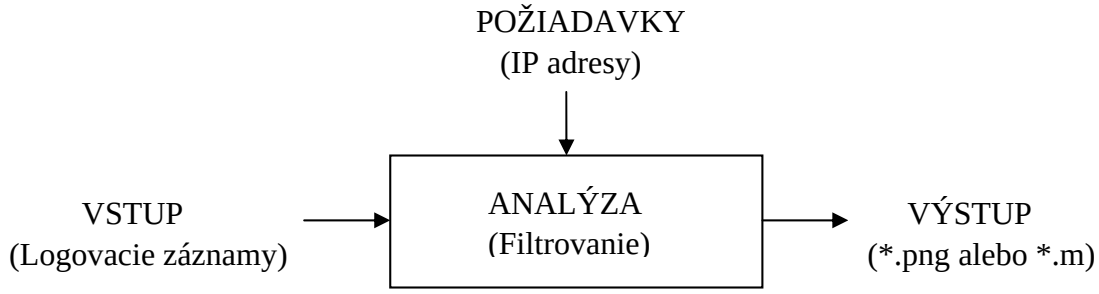
2.1 Požiadavky na aplikáciu

Štatistický nástroj by mal užívateľovi pomáhať pri analýze sieťových simulácií, vytvorených pomocou knižnice JSimlib4. Analyzovať sa má počet prenesených bytov medzi dvoma alebo viacerými uzlami v sieti. Modul sa dá rozdeliť na niekoľko častí, do ktorých patria logovanie, filtrovanie vyhovujúcich záznamov a zobrazenie vyfiltrovaných hodnôt.

Po premyslení vhodného postupu realizácie štruktúry štatistického nástroja bolo zistené, že každú analýzu vo všeobecnosti a tým pádom aj v tomto prípade je možné rozložiť na nasledujúce časti:

- definovanie vstupných hodnôt,
- požiadavky analýzy,
- samotný proces analýzy,
- získanie a zobrazenie výstupných hodnôt.

Toto rozdelenie znázorňuje obrázok 2.1 Proces analýzy.



Obr. 2.1 Proces analýzy

Vstupné hodnoty pre túto aplikáciu sú umiestnené v logovacom súbore, ktorý obsahuje zdrojovú a cieľovú IP adresu s využívaným portom, prenesené byty, časový údaj, použitý protokol a smer prenosu. Každý záznam v logovacom súbore je oddelený novým riadkom. Zdrojová a cieľová IP adresa určí stanice v sieti, medzi ktorými boli prenesené byty v určitom časovom okamihu. Port presnejšie definuje použitú aplikáciu. Protokol a smer prenosu slúžia na špecifickejšie definovanie prenosu.

Požiadavkou, podľa ktorej má byť analýza vykonaná, môže byť špecifická IP adresa v sieti, skupina IP adries alebo názov stanice. Možnosť zadávania tzv. divokých kariet, čiže zadanie univerzálnej a spoločnej IP adresy pre skupinu uzlov, pomáha pri definovaní viacerých uzlov jedným zápisom. Tieto divoké karty je možné zadať dvoma spôsobmi. Buď pomocou pomlčky (-) alebo pomocou hviezdičky (*). Každý znak má špecifický význam a aplikácia ho vyhodnocuje inak. V istých prípadoch je potrebné určiť len zdrojové alebo cieľové stanice, inokedy je požadovaná analýza medzi špecifickou zdrojovou a cieľovou stanicou, poprípade stanicami.

Po zadaní IP adries, podľa ktorých má aplikácia vyfiltrovať vyhovujúce záznamy nasleduje proces filtrovania. Aplikácia postupne číta jednotlivé záznamy v logovacom súbore a porovnáva IP adresy v týchto záznamoch s IP adresami, ktoré boli zadané do aplikácie. Na konci tohto procesu sú do polí uložené užívateľom požadované záznamy – čas a počet prenesených bytov.

V prípade, že chceme vykonať analýzu bez reštartovania aplikácie je výhodou možnosť upravovať, poprípade vymazať už definované IP adresy, ktoré nie sú potrebné k nasledujúcej analýze.

Výhodou je, samozrejme, možnosť vytvorenia zálohy už raz definovaných staníc. V prípade, že by užívateľ chcel zistiť neskôr prenos medzi rovnakými stanicami, má možnosť uložiť si aktuálne zadané IP adresy do textového súboru a neskôr ich načítať znova.

Posledným krokom je získanie výstupných súborov. Aplikácia podporuje 2 výstupné formáty, ktoré umožňujú prácu so získanými údajmi rôznymi spôsobmi.

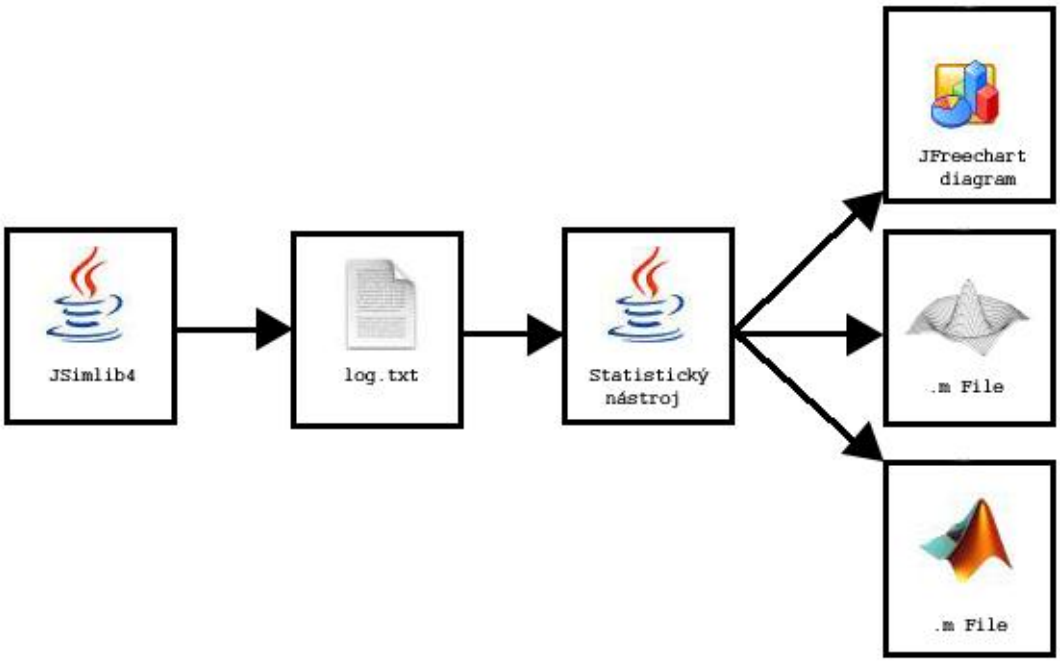
Výstup z aplikácie v obrázkovom formáte PNG umožňuje užívateľovi vyhodnotiť analýzu najjednoduchším spôsobom, pretože takto má možnosť vidieť hodnoty vynesené do grafu. Už na prvý pohľad tak získa informácie o výstupných hodnotách analýzy. Keby mal tieto údaje podané v inom formáte, tak toto vyhodnotenie by mu mohlo trvať rádovo niekoľkokrát dlhšiu dobu. Hoci pre človeka je jedným z najzrozumiteľnejších formátov práve grafický, neznamená to, že tento spôsob podania informácie je aj najefektívnejší. Tento formát je pre ďalšiu prácu s výsledkom analýzy v istej miere nevhodný, pretože z diagramu neskôr nie je možné načítať získané údaje a pracovať s nimi v iných programoch tak jednoducho ako napríklad s maticami.

Práve preto bola vytvorená možnosť ukladania získaných hodnôt aj v textovom formáte s príponou *.m, ktorý obsahuje maticu s rovnakými hodnotami ako sú vynesené do grafu

a príkaz na jej vykreslenie. Súbor *.m je spoločným vstupným formátom pre programy Matlab a Octave. Tieto programy sú účinné nástroje pre matematické a rôzne výpočty a sú vhodné na simuláciu dynamických systémov.

Užívateľ aplikácie si tieto výstupné formáty môže zobraziť v dolnej časti aplikácie. Týmto je dosiahnutá efektívnosť aplikácie, pretože výstupné súbory nie je nutné otvárať v iných programoch po dokončení analýzy. Na tento účel je navrhnuté plátno, kde sa po vykonaní analýzy zobrazí buď obrázok ukazujúci prenesené byty medzi zadanými stanicami alebo výpis obsahu výstupného súboru s maticou.

Obrázok 2.2: Vstup a výstup z aplikácie znázorňuje vstupné a výstupné súbory štatistického nástroja aj so spolupracujúcou knižnicou JSimlib4.



Obr. 2.2 Vstup a výstup z aplikácie

2.2 Zaznamenávanie priebehu simulácie

Prvú časť analýzy tvorí zaznamenávanie prenesených bytov medzi stanicami v sieti. Tieto záznamy nám neskôr môžu pomôcť napríklad pri určení zaťaženia linky medzi uzlami v sieti alebo vyhľadávaní stanice, ktorá posiela, poprípade prijíma najväčšie množstvo bytov.

Samozrejme, využitie záznamov z logovacieho súboru je neobmedzené a záleží iba na fantázii a potrebách konkrétneho užívateľa.

Z tohto dôvodu bola táto časť navrhnutá tak, aby bola čo najuniverzálnejšia a aby sa jej metódy dali využiť aj v iných častiach aplikácie. Pred jej návrhom boli naštudované triedy `java.io.BufferedWriter`, `java.io.File`, `java.io.FileWriter` programovacieho jazyka JAVA, ktoré boli neskôr použité pri návrhu triedy `Logwriter.java`. Trieda `java.io.File` je hlavnou triedou na vytváranie a spravovanie súborov a adresárov. Súbory a adresáre sú reprezentované objektmi tejto triedy. Java poskytuje dve sady tried na zápis a čítanie súborov. Triedy `OutputStream/InputStream` slúžia pre byty, primitívne dáta a objekty, kým triedy `Reader/Writer` sú pre zápis a čítanie charakterov a reťazcov. Objekt triedy `java.io.BufferedWriter` poskytuje efektívny spôsob zapisovania niekoľkých charakterov naraz. Objekty tejto triedy používajú vyrovnávaciu pamäť na ukladanie výstupu pre spolupracujúcu triedu `Writer`. Inými slovami, veľké množstvo charakterov je uložené vo vnútornej vyrovnávacej pamäti a vypíše sa vtedy, keď sa pamäť zaplní alebo je explicitne naplnená. Trieda `BufferedWriter` je efektívnejšia ako regulárna trieda `Writer`, pretože dáta sú zapísané do pamäti a nie na pevný disk alebo sieť. Práve minimalizácia počtu zápisov na pevný disk alebo sieť minimalizuje narastajúcu réžiu pre tieto operácie. Trieda `java.io.FileWriter` slúži na výstup charakteru do diskového súboru.

Zápis do logovacieho súboru je vykonaný simulačnou knižnicou JSimlib4, ktorá najprv vytvorí logovací súbor pomocou konštruktoru triedy `LogWriter.java` a neskôr pri každom prenose zavolá metódu na zápis záznamov, ktorej predá údaje potrebné k zaznamenávaniu prenosu.

Konštruktor triedy `LogWriter.java` bol navrhnutý takým spôsobom, aby sa staral o vytváranie logovacieho súboru automaticky. Otestuje, či už logovací súbor s daným názvom existuje. V prípade, že neexistuje, vytvorí súbor log.log v koreňovom adresári simulačnej knižnice. V opačnom prípade, napríklad pri nasledujúcom zápise záznamov do logovacieho súboru, konštruktor zistí existenciu súboru s rovnakým názvom, a preto ho znova nevytvára, iba zapíše logovacie záznamy, ktoré popisujú daný prenos.

Prvý riadok logovacieho súboru obsahuje komentár v tvare:

```
#comment Source_IP_address:Port:Destination_IP_address:Port:Time:Bytes:Protocol:Direction
```

Uvedený komentár udáva poradie prvkov v zázname, ktoré sú oddelené dvojbodkou (:). Každý záznam je tvorený zdrojovou adresou a portom, cieľovou adresou a portom, časom prenosu, počtom prenesených bytov medzi zdrojovou a cieľovou adresou. Ďalej obsahuje používaný protokol a nakoniec smer prenosu. Týmto spôsobom je každý riadok jednoznačný a logovací súbor je prehľadný už na prvý pohľad.

Zdrojové a cieľové IP adresy v logovacom súbore reprezentujú stanice v sieti medzi ktorými je prenos zaznamenaný. Ich hodnoty sa zapisujú v tvare X.X.X.X, kde X zastupuje celé číslo v rozmedzí 0 až 255. Tento tvar zápisu je štandardný tvar tzv. IP adresy.

IP adresa sa skladá zo 4 oktetov, kde každý oktet môže zastupovať hodnotu medzi 0 až 255. 4 oktety vytvárajú spoločne 1 IP adresu, ktorá prezentuje danú stanicu.

Porty jednotlivých staníc presnejšie špecifikujú použitú aplikáciu. Sú definované číslom od 0 až 65535 a slúžia v počítačových sieťach pri komunikácii pomocou transportných protokolov TCP a UDP k rozlíšeniu aplikácie v rámci počítača.

Ďalší prvok v zázname udáva časový údaj v milisekundách, tzv. časový odtlačok, ktorý určí čas prenosu medzi zdrojovou a cieľovou stanicou. Tento údaj sa zapisuje v UNIX-ovom formáte, čo znamená, že hodnota udáva milisekundy ubehnuté od polnoci 1.1.1970. V tomto formáte bol ponechaný kvôli tomu, že v takejto podobe je získaný zo simulácie. Neskôr, v prípade potreby, je možné ho ľubovoľne prekonvertovať do iného požadovaného tvaru.

Za časovým údajom nasleduje hodnota prenesených bytov medzi zdrojovou a cieľovou stanicou.

Ďalším prvkom v zázname je typ protokolu. Simulačná knižnica JSimlib4 dokáže simulovať transportné protokoly UDP (User Datagram Protocol) aj TCP (Transmission Control Protocol). Z tohto dôvodu je dôležité rozpoznávať typ protokolu, ktorý bol použitý počas daného prenosu. Do logovacieho súboru na miesto protokolu sa vždy zapíše skrátený názov protokolu, čiže v prípade využívania protokolu User Datagram Protocol sa zapíše skratka „UDP“ a pri využívaní TCP protokolu „TCP“.

Posledný prvok v zázname je smer prenosu. Môže byť buď prichádzajúci alebo odchádzajúci. Tieto možnosti smeru sa zapisujú do každého záznamu anglicky. Pri prichádzajúcom smere sa zapíše skratka RECV z anglického slova recieved, čo znamená „prijatý“. Pri odchádzajúcom smere sa zapisuje anglické slovo SENT, čiže „poslaný“.

Logovacie záznamy sa zapisujú pomocou metódy `writeLog()` triedy `LogWriter.java`, ktorá využíva spomínané triedy Javy na zápis do súborov. Po zapísaní

záznamu, čiže jedného riadku do logovacieho súboru, skočí kurzor na nasledujúci riadok a čaká na zápis ďalšieho záznamu. Po zápise logovacieho záznamu je práca s vnútornými triedami ukončená pomocou metódy `close()`. Je veľmi dôležité, aby objekty tried používané na zápis a čítanie záznamov boli po ukončení práce správne zatvorené, inak by mohlo dôjsť k neočakávanej udalosti.

2.3 Filtrovanie vhodných záznamov

Filtrovanie spočíva v získaní vhodných logovacích záznamov podľa zadaných kritérií, ktoré sú potom predané do polí konečných výsledkov .

2.3.1 Filtračné typy podľa typu stanice

Aplikácia umožňuje zadať podľa pôvodu staníc nasledujúce filtračné typy:

- zdrojová stanica,
- cieľová stanica,
- zdrojová aj cieľová stanica.

Prvý filtračný typ, ktorý sa definuje pomocou zdrojovej stanice, obsahuje iba IP adresu zdrojových staníc. Tento typ je vhodný pre užívateľa, ktorý chce vyfiltrovať záznamy, ktoré vytvorila istá zdrojová stanica, poprípade zdrojové stanice.

Ďalší typ – zadaný pomocou cieľovej stanice – je podobný prvému. Rozdiel je v pôvode stanice, ktorá je v tomto prípade vždy cieľová.

Poslednou možnosťou zadania stanice je najšpecifickejší typ filtrovania. Je ním typ obsahujúci naraz obidva druhy stanice, čiže ako zdrojovú, tak aj cieľovú. Je možné ho využívať v prípade, keď chceme vyfiltrovať záznamy medzi špecifickými zdrojovými a cieľovými stanicami a tým sa dostať k presnej hodnote prenosu medzi týmito stanicami.

2.3.2 Filtračné typy podľa IP adresy a názvu stanice

Užívateľ má možnosť určiť špecifickú stanicu alebo skupinu staníc rôznymi spôsobmi. Tieto spôsoby sa dajú rozdeliť podľa hodnoty IP adresy alebo názvu stanice do nasledujúcich skupín:

- konkrétna IP adresa,
- skupina IP adries,
- všetky IP adresy v oktete,
- názov stanice.

Stanica zadaná pomocou konkrétnej IP adresy je najjednoduchším spôsobom určenia zdrojovej alebo cieľovej stanice. Obsahuje iba 1 IP adresu, ktorá určuje 1 stanicu v sieti. Príkladom môže byť IP adresa 192.168.1.0 alebo 172.17.10.0.

Pomlčka (-) zastupuje skupinu IP adries. Táto možnosť je vhodná pri definovaní viacerých staníc pomocou jedného zápisu. Je tak možné naraz zadať niekoľko IP adries, ktoré sa nachádzajú priamo za sebou. Tento typ IP adresy je definovaný v istom oktete začiatočným a koncovým číslom, ktoré sú oddelené pomlčkou. Všetky čísla medzi začiatočným a koncovým číslom sú započítané do IP adries, podľa ktorých má aplikácia vykonať filtrovanie. Podmienkou správnosti zápisu tejto možnosti je, že začiatočné číslo musí byť vždy menšie ako koncové. Ako príklad môžeme uviesť adresu 192.168.1.1-10, kde sa pomocou jedného zápisu započítajú adresy 192.168.1.1, 192.168.1.2, 192.168.1.3, ... , 192.168.1.10 do filtračných adries.

Ďalším špeciálnym znakom je hviezdička (*). Tento charakter zastupuje všetky čísla od 0 až po 255 v oktete, v ktorom sa nachádza. Preto slúži na zadávanie všetkých staníc, ktoré vyhovujú nasledujúcej podmienke:

Ak hľadaná IP adresa má na tých troch oktetoch, ktoré sú zadané vo filtračnej IP adrese s konkrétnym číslom, rovnaké čísla požadovanej IP adresy a na mieste štvrtého oktetu má ľubovoľné číslo medzi 0 až 255, ktorý vo filtračnej IP adrese je definovaný pomocou hviezdičky, tak vyhovuje podmienke a započíta sa do vyfiltrovaných adries.

To znamená, že keď užívateľ zadá IP adresu napríklad v tvare 192.168.1.*, tak tento zápis vyfiltruje každú adresu, ktorá obsahuje na prvých troch oktetoch hodnotu 192.168.1 bez ohľadu na číslo vyskytujúce sa v poslednom oktete. Tomuto zápisu vyhovujú všetky stanice s IP adresou 192.168.1.0, 192.168.1.1, 192.168.1.2 až 192.168.1.255.

Tento typ zápisu môže užívateľovi aplikácie pomáhať v rôznych situáciách, napríklad keď užívateľ dopredu nevie presnú IP adresu alebo potrebuje vyfiltrovať všetky IP adresy, ktoré na danom oktete môžu obsahovať ľubovoľné číslo.

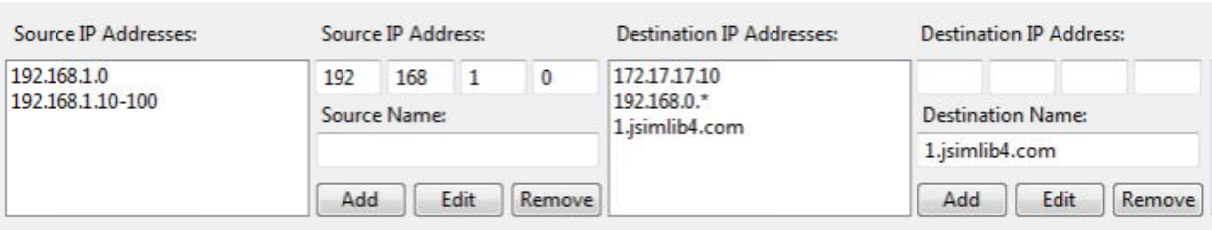
So zápisom *.*.*.* je možné vyhovieť požiadavke, aby každá IP adresa, ktorá sa objavila v logovacom súbore, bola započítaná do výsledku analýzy. To pomáha napríklad v situáciách,

keď potrebujeme dostať hodnoty všetkých prenosov vyskytujúcich sa v logovacom súbore na výstup a nepoznáme IP adresy zapísané do logovacieho súboru.

Simulačná knižnica JSimlib4 podporuje zápis názvu koncových staníc, medzi ktorými bol prenos uskutočnený. Knižnica môže vytvárať napríklad názvy ako 0.jsimlib4.com alebo 1.jsimlib4.com, ale je možné generovanie názvov upraviť podľa požiadaviek. Posledná možnosť filtrovania staníc zaznamenaných do logovacieho súboru je práve preto umožnená pomocou definovania názvu stanice.

Je možné kombinovať jednotlivé typy filtrov. To znamená, že je dovolené definovať napríklad zdrojovú stanicu s konkrétnou IP adresou a cieľové stanice so skupinou IP adries alebo všetkými IP adresami v oktete. Je povolené zadať súčasne aj viac IP adries pre jeden druh stanice (čiže pre zdrojovú alebo cieľovú), podľa ktorej má byť filtrovanie vykonané alebo kombinovať typy týchto IP adries.

Ako príklad filtrovania s rôznymi filtračnými typmi podľa IP adresy a názvu stanice je možné uviesť zápis pre zdrojové stanice IP adresy 192.168.1.0, 192.168.1.10-100 a pre cieľové stanice IP adresy 172.17.17.10, 192.168.0.* a 1.jsimlib4.com. Je to kombinácia definovania stanice, zadaním IP adresy a názvu stanice. Tento zápis je znázornený na obrázku 2.3 Príklad zadania rôznych filtračných typov.



Obr. 2.3Príklad zadania rôznych filtračných typov

2.3.3 Proces filtrovania

Po každom prečítaní záznamu, ktorý vyhovuje zadaným filtrom sa zapíše tento údaj do pomocného pola, ktoré sa neskôr využíva pri analýze.

Úlohou filtrovania je porovnať užívateľom zadanú IP adresu (IP adresy) s adresami v logovacom súbore a vybrať z neho iba užívateľom vyhovujúce záznamy.

Proces filtrovania má niekoľko krokov.

Prvým z nich je načítanie zadaných IP adries, podľa ktorých má byť filtrovanie vykonané. Podľa potreby užívateľa je možné zadať IP adresu vo vyššie uvedených tvaroch.

Nasledujúci krok je porovnanie IP adresy s IP adresami v logovacom súbore. Na túto úlohu využíva aplikácia Java triedy `java.io.BufferedReader`, `java.io.DataInputStream` a `java.io.FileInputStream`. Aplikácia najprv otestuje ktoré druhy stanice (zdrojová, cieľová, poprípade obidve) boli zadané a podľa toho porovnáva jednotlivé záznamy. V prípade, že bol zadaný iba jeden druh stanice (buď zdrojová alebo cieľová) je situácia jednoduchšia ako v prípade zadania obidvoch typov súčasne.

Ak užívateľ zadá iba zdrojovú alebo iba cieľovú stanicu, aplikácia postupne prechádza logovací súbor. Najprv vytvorí pomocnú premennú a deklaruje ju na hodnotu 0, ktorú postupne inkrementuje, až kým nedosiahne hodnotu, ktorá sa rovná počtu užívateľom zadaných IP adries. Keď je hodnota tejto pomocnej premennej 0, čo je po jej deklarácii, je proces filtrovania iný ako v nasledujúcich prípadoch, čiže po prvej inkrementácii.

Rozlišuje sa to z toho dôvodu, aby nebola tá istá IP adresa započítaná dvakrát do konečného výsledku. V prípade, že sa jedná o prvú zadanú IP adresu, ktorú aplikácia porovnáva s údajmi v logovacom súbore, táto IP adresa nemôže byť ešte započítaná do konečných výsledkov, pretože ešte žiadna IP adresa nebola porovnávaná s logovacími údajmi. To však už neplatí pri druhom a ďalších porovnaniach, pretože napríklad druhá, tretia alebo ľubovoľná ďalšia IP adresa už mohla byť porovnávaná a započítaná do výsledku v prvom kroku.

Predstavme si prípad, keď užívateľ zadá nasledujúce IP adresy pre zdrojové stanice: 192.168.1.2, 147.229.82.213, 192.168.1.0-10. V prvom kroku by si aplikácia rozdelila skupinový zápis IP adresy 192.168.1.0-10 na samostatné IP adresy. Potom by nasledoval zápis každej IP adresy do prvku pola, čiže 192.168.1.2, 147.229.82.213, 192.168.1.0, 192.168.1.1, 192.168.1.2, ..., 192.168.1.10 a tak by vzniklo pole, ktoré má 13 prvkov. Máme možnosť vidieť, že IP adresu 192.168.1.2 obsahuje pole dvakrát, a preto by bol proces filtrovania nasledujúci. Aplikácia by najprv zobrala na filtrovanie prvý prvok pola, s IP adresou 192.168.1.2, ktorá bola zadaná ako prvá a začala by ju porovnávať postupne s IP adresami zdrojových staníc v každom logovacom zázname. V prípade nájdenia vyhovujúceho záznamu by zaznamenala počet prenesených bytov udávaných v tomto zázname. Aplikácia by pokračovala v prechádzaní cez zvyšok logovacieho súboru a zapísala by počet prenesených bytov v každom vyhovujúcom zázname. Nasledoval by druhý a tretí prvok pola rovnakým procesom. Avšak štvrtý prvok pola, ktorý by obsahoval IP adresu 192.168.1.2 by pri takomto

postupe zapríčinil nesprávny výpočet konečného výsledku. Došlo by k nesprávnemu výpočtu, pretože počet prenesených bytov zo stanice definovanej touto IP adresou by bol znova započítaný do konečného výsledku, čo samozrejme nie je správne, pretože sa to už raz stalo.

Preto bola vytvorená podmienka, ktorá rozlišuje prvý prvok pola od ostatných.

Z toho dôvodu je proces filtrovania nasledujúci. V prípade, že sa jedná o prvú IP adresu, aplikácia prejde celý logovací súbor a postupne porovnáva IP adresy jednotlivých záznamov s touto IP adresou. Keď nájde rovnakú IP adresu v zázname, zapíše počet prenesených bytov z tohto záznamu do konečného výsledku a pokračuje ďalej, až kým nenarazí na charakter koniec súboru. Potom vezme ďalší prvok (ďalšiu IP adresu) z pola a opäť začne postupne porovnávať IP adresy v záznamoch s IP adresou v tomto prvku pola. Zmena nastane oproti prvému prvku v poli v bode, keď aplikácia nájde vyhovujúci záznam v logovacom súbore s touto IP adresou. Aplikácia si vtedy vytvorí pomocnú premennú a deklaruje ju na 0. Vytvorí si cyklus, v ktorom postupne inkrementuje túto pomocnú premennú. Hodnota tejto pomocnej premennej odkazuje na prvky pola, ktoré obsahuje zadané IP adresy. Cyklus postupne inkrementuje pomocnú premennú a porovnáva aktuálnu IP adresu s IP adresou, ktorá je umiestnená v tomto inkrementovanom prvku a na ktorý odkazuje pomocná premenná. Ak sa nájde prvok, ktorý obsahuje rovnakú IP adresu ako aktuálny prvok, cyklus sa hneď ukončí a nezapočíta sa prenos do výsledku, pretože aplikácia vie, že s danou IP adresou už raz porovnávala logovacie záznamy. Ak aplikácia nenašla IP adresu v predchádzajúcich prvkoch, tak pripočíta počet prenesených bytov z aktuálneho záznamu do konečného výsledku.

V prípade súčasného zadania obidvoch typov stanice je filtrovanie o niečo komplikovanejšie. Začiatok postupu filtrovania je podobný postupu, pri ktorom je zadaný iba jeden druh stanice. Zmena však nastane, keď aplikácia nájde vyhovujúci záznam. Postup tohto typu filtrovania môžeme rozdeliť na 2 kroky:

Aplikácia najprv zistí, či logovací súbor obsahuje záznam, ktorý má IP adresu zdrojovej stanice rovnakú ako zadaná zdrojová IP adresa. Keď nájde takýto záznam, tak vykoná druhý krok procesu.

V druhom kroku vytvorí aplikácia pomocnú premennú a deklaruje ju na 0. Vytvorí cyklus, v ktorom pomocná premenná odkazuje na prvky pola, v ktorom sú umiestnené IP adresy cieľových staníc. Postupne inkrementuje túto pomocnú premennú a tak sa dostane ku každému prvku „pola cieľových staníc“. Tento cyklus slúži na porovnanie cieľových IP adries v záznamoch logovacieho súboru s filtračnými cieľovými IP adresami. V prípade, že počas porovnania cieľových IP adries aplikácia nájde rovnakú filtračnú IP adresu ako je uvedená v

niektorom zázname logovacieho súboru, tak počet prenesených bytov a časový údaj zapíše do konečného výsledku. V prípade, že aplikácia nájde vyhovujúcu zdrojovú IP adresu v záznamoch, a preto prejde do druhého kroku, v ktorom však nenájde cieľovú IP adresu z logovacieho záznamu v „poli cieľových staníc“, tak sa prenos nezapočíta do konečného výsledku.

Aplikácia používa na ukladanie hodnôt prenesených bytov a časových údajov pole `java.util.ArrayList`. Táto trieda bola zvolená pre tieto údaje preto, lebo dopredu nie je známa veľkosť týchto polí. Takto sa môžu polia dynamicky rozširovať a nepotrebujú zložité algoritmy na vytvárania a odstraňovanie prvkov, pretože o tieto procesy sa stará `java.util.ArrayList`.

Pri zapisovaní prenesených bytov do konečného výsledku môžu nastať 2 situácie.

Prvou z nich je, že konečný výsledok ešte neobsahuje rovnaký časový údaj, ktorý je v aktuálnom zázname. V tom prípade sa jednoducho zapíše počet prenesených bytov do jedného pola a časový údaj do ďalšieho. Nakoniec sa poradie prvkov v týchto poliach zoradí od najmenšej po najväčšiu hodnotu .

Môže nastať situácia, keď 2 rozdielne stanice pošlú dáta iným staniciam v rovnakom časovom okamihu. Pri výskyte tejto situácie sa do konečného výsledku započíta súčet prenesených bytov z obidvoch prenosov. V tomto prípade, keď už časové pole obsahuje rovnaký časový údaj ako je v aktuálnom vyhovujúcom logovacom zázname, sa počet prenesených bytov pripočíta k preneseným bytom pre tento časový okamih.

Všeobecne sa to dá popísať nasledujúcou vetou: Ak aplikácia nájde vyhovujúci záznam s časovým údajom, ktorý už aktuálne obsahuje, tak počet prenesených bytov z nového záznamu pripočíta k hodnote prvku „pola prenesených bytov“ pre tento časový okamih. Táto operácia sa dá obecne vyjadriť nasledujúcou rovnicou (1):

$$SUMA_{byty} = hodnota_{existujúca} + hodnota_{nová} \quad (1)$$

kde $SUMA_{byty}$ je nová hodnota prenesených bytov v danom časovom okamihu,

$hodnota_{existujúca}$ je zapísaná hodnota vyhovujúcich prenesených bytov v danom časovom okamihu pred prečítaním aktuálneho riadku,

$hodnota_{nová}$ je prečítaná hodnota prenesených bytov z aktuálneho riadku v danom časovom okamihu.

Táto situácia by mohla nastať v prípade, keby užívateľ zadal viac IP adries, podľa ktorých by mala aplikácia vyfiltrovať zdrojové stanice. Aplikácia by začala postupne prechádzať logovací súbor a porovnávať jednotlivé záznamy s aktuálnou IP adresou, ktorú by po prečítaní celého logovacieho súboru postupne menila. Toto porovnávanie by vykonávala až kým by neporovnala každú IP adresu, ktorú užívateľ zadal na filtrovanie s každým záznamom v logovacom súbore.

Predstavme si, že by aplikácia našla hneď s prvou zadanou IP adresou vyhovujúci záznam s časovým údajom 1 milisekunda a tak by zapísala časový údaj do jedného pola a prenesené byty do druhého. Pokračovala by ďalej a zapísali by sa do pola napríklad aj ďalšie vyhovujúce záznamy, ale s inými časovými údajmi. Po prejdení celého súboru by aplikácia zobrala na porovnávanie ďalšiu filtračnú IP adresu a začala by porovnávať záznamy s touto IP adresou. Predstavme si, že by narazila na vyhovujúci záznam, ktorý sa zaznamenal v prvej milisekunde, podobne ako sa to stalo aj v prípade s prvou filtračnou IP adresou. V tomto prípade by nebolo vhodné prepísať počet bytov prenesených v tomto časovom okamihu, pretože by tak aplikácia nepodávala korektné konečné výsledky. Z toho dôvodu, aplikácia najprv zistí, či „časové pole“ obsahuje prvok s daným časovým okamihom a keď nájde rovnaký časový okamih v poli, tak vie, že má pripočítať počet prenesených bytov z aktuálneho záznamu k počtu bytov v „bytovom poli“. Keď nenájde časový okamih rovnaký okamihu z aktuálneho záznamu, tak umiestni časový údaj do novovytvoreného prvku v „časovom poli“, preskupí poradie prvkov, tak aby nový prvok bol zaradený podľa hodnoty a rovnakým postupom vytvorí aj prvok v poli pre počet prenesených bytov.

Posledným krokom procesu filtrovania je zápis vyhovujúcich údajov do obyčajného pola. Pre „časové pole“ je používané pole dátového typu `Long`, pretože časové hodnoty sú zapísané v milisekundách, a preto im nestačí dátový typ `Integer`. Na počet prenesených bytov sa používa pole s dátovým typom `Integer`, pretože tento typ vyhovuje najviac požiadavkám ukladania celých čísel. Po správnom uložení prvkov do pola môže nasledovať zobrazenie výstupu simulácie.

2.4 Zobrazenie výsledkov simulácie

Jednou z daných požiadaviek na aplikáciu bolo navrhnúť taký výstupný formát, ktorý je kompatibilný s inými programami. Preto bol navrhnutý výstupný súbor vo formáte *.m a tým pádom umožnená práca so získanými hodnotami aj v iných programoch. Pretože tento formát

je kompatibilný s programami Matlab a Octave, hodnoty sa dajú načítať súčasne v obidvoch programoch. Poskytuje to väčšiu univerzálnosť aplikácie, pretože užívateľ takto nie je obmedzený možnosťami aplikácie, ale dokáže pracovať s vyfiltrovanými hodnotami aj v iných programoch.

Matlab [14] je aplikácia určená hlavne pre numerické a matematické výpočty. Je to vysokoúrovňový programovací jazyk a vývojové prostredie, ktoré umožňuje vykonávať výpočty rýchlejšie ako programovacie jazyky, napríklad Java alebo C++. Je možné k nemu pripojiť rozširujúce nástrojové moduly, ktoré umožňujú jednoduchú simuláciu rôznych dynamických systémov. Ďalej poskytuje nástroje pre výpočty z oblasti lineárnej algebry, harmonickej analýzy, optimalizácie, numerických integrácií a mnoho iného.

Octave [15, 16] je voľne dostupným variantom komerčného softwaru Matlab. Je to vyšší programovací jazyk, primárne určený pre numerické výpočty. Poskytuje pohodlné prostredie príkazového riadku pre číselné riešenie lineárnych aj nelineárnych problémov a pre vykonávanie iných numerických experimentov. Tiež môže byť použitý ako dávkovo orientovaný jazyk. Väčšina jeho príkazov je ekvivalentných príkazom Matlabu.

Ďalšou požiadavkou aplikácie bol grafický výstup znázorňujúci vyhovujúce záznamy po analýze. Táto úloha bola splnená pomocou diagramu, ktorý má na x-ovú os vynesené časové údaje a na y-ovú počet prenesených bytov. Ako najuniverzálnejší spôsob realizácie tejto požiadavky sa zdalo využitie javovskej knižnice JFreeChart. Pomocou tejto knižnice je možné vygenerovať graf a neskôr ho uložiť ako obrázok v rôznych grafických formátoch.

2.4.1 Vytváranie diagramu pomocou JFreeChart

JFreeChart [17] je javovská knižnica pre vytváranie diagramov. Umožňuje vytvárať širokú paletu grafov, spomedzi ktorých by sa dali uviesť koláčové, stĺpcové alebo XY diagramy. Pri riešení úlohy bola využitá aj z dôvodu voľnej šíriteľnosti a pohodlnej práce. Tieto kritéria sa javili pri výbere ako najdôležitejšie.

Po stiahnutí poslednej verzii JFreeChart z domovskej stránky http://www.jfree.org/jfreechart/ je nasledujúcim krokom rozbalenie zabalených zdrojových súborov. Po tejto operácii môže prebehnúť zoznámenie sa s funkciami a triedami knižnice. Medzi zdrojovými kódmi je uložená aj ukážková aplikácia jednotlivých diagramov. Tento demonštračný program znázorňuje širokú paletu diagramov, ktoré sa dajú vygenerovať pomocou JFreeChart. Spustenie aplikácie prebieha príkazom:

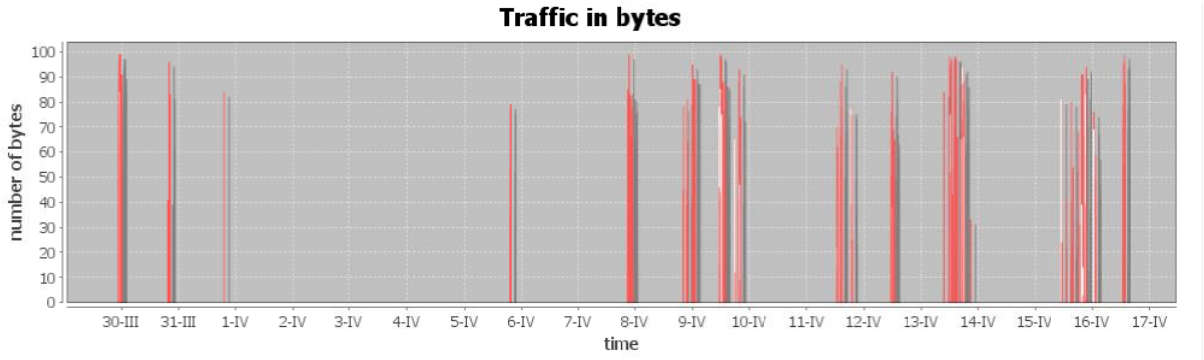
java -jar jfreechart-1.0.0-rc1-demo.jar

Zdrojový kód z demonštračného programu nie je obsahom JfreeChart distribúcie. Je dostupný iba oddelene, pri kúpe sprievodcu vývojára JfreeChart (JfreeChart Developer Guide) [18]. Sprievodca vývojára JfreeChart poskytuje rozšírenú dokumentáciu ku knižnici JFreeChart.

JFreeChart využíva knižnicu JCommon, ktorá je obsahom aj stiahnutého, zabaleného súboru. K práci s diagramami musia byť preto obidve knižnice pridané do vývojového prostredia.

JFreeChart dokáže generovať a uložiť diagramy do výstupných formátov JPEG [19] a PNG [20]. Vhodnejšou voľbou sa javil grafický formát PNG, ktorý je určený pre bezstratovú kompresiu rastrovej grafiky a do istej miery nahradzuje grafický formát GIF. PNG je vhodnejší formát než JPEG pre obrázky obsahujúce text, čiarovú grafiku, čisté farebné plochy a ostré rozhranie farieb. Pretože PNG má voľnú licenciu, umožňuje 24b farbu a 8b priesvitnosť (alfa-kanál), stopercentne vyhovuje kritériám.

Z dôvodu, že diagram generovaný pomocou knižnice JFreechart zapadne presne do popisu formátu PNG (čiarová grafika a čisté farebné plochy) bol zvolený tento grafický formát.



Obr. 2.4 Diagram generovaný pomocou JFreeChart

Obrázok je generovaný volaním konštruktoru triedy `PNG.java`, do ktorého sú ako vstup predané polia získané z výstupu filtrovania. Postupne sú načítané a predané hodnoty do objektu `series` triedy `org.jfree.data.xy.XYSeries` a nasledovne sa pomocou metódy `createXYBarChart` vygeneruje požadovaný diagram. V prvej fáze vývoja aplikácie bol zvolený úsekový typ diagramu, ale neskôr bol zmenený na typ `XYBarChart.` Tento typ diagramu je výhodnejší v prípadoch, keď sa majú hodnoty na y-ovej osi vykresliť

iba vtedy, keď obsahujú nenulovú hodnotu. Práve tento prípad sa vyskytuje aj vo vytvorených diagramoch, čiže nie v každom časovom okamihu sú prenášané byty medzi stanicami. Úsekový typ by generoval spojový graf, čo by znamenal, že dve susedné hodnoty by boli spojené priamkou a to by bolo v tomto prípade nevhodné.

Predstavme si, že stanice, ktoré by užívateľ chcel vyfiltrovať a zobraziť ich výsledok v diagrame, by komunikovali medzi sebou len od prvej milisekundy do piatej, teda by vytvárali prenos v týchto časových okamihoch. Potom by na 10 milisekúnd neposielali žiadne údaje, čiže by neprebiehal žiadny prenos a nakoniec od pätnástej do dvadsiatej milisekundy by znova posielali dáta medzi sebou. V prípade, že by aplikácia využívala spojový graf, hodnota prenesených bytov v piatej milisekunde by bola spojená priamkou s hodnotou prenesených bytov v pätnástej milisekunde. Užívateľ by si teda myslel, že stanice komunikovali aj v týchto časových okamihoch, čo samozrejme nie je pravda.

Z tohto dôvodu bol navrhnutý diagram, ktorý využíva grafový typ `XYBarChart` a tak zobrazuje na y-ovej osi iba tie hodnoty prenesených bytov, ktoré skutočne vytvárali stanice v sieti. V časových okamihoch, keď stanice nekomunikujú, nezakreslí do diagramu žiadne hodnoty, a preto užívateľ dostane správny výstupný diagram.

JFreeChart umožňuje rôzne nastavenia pri generovaní diagramov. Pre každý graf sa dá nastaviť jeho názov, popis x-ovej a y-ovej osi, nastavenie vynesených údajov, orientácia grafu (vertikálna alebo horizontálna), výpis legendy a ďalšie.

Na generovanie PNG obrázku z diagramu sa používa metóda `saveChartAsPNG` triedy `org.jfree.chart.ChartUtilities.` Ďalšou možnosťou na uloženie grafu je metóda `saveChartAsJPEG`, ktorá ukladá grafy vo formáte JPEG. Vstupom do týchto metód je veľkosť a názov vygenerovaného obrázku.

Vygenerovaný obrázok pomocou JFreeChart je znázornený na obrázku Obr. 4.3: Diagram generovaný pomocou JFreeChart.

2.4.2 Vytváranie výstupných súborov pre programy Matlab a Octave

Na generovanie výstupných súborov pre programy Matlab a Octave bola využívaná trieda `LogWriter.java.` Konštruktor tejto triedy vytvorí súbor vo formáte *.m a uloží ho do užívateľom definovaného adresára.

Do prvého riadku tohto súboru sa zapíše "`x=[`". Potom aplikácia postupne načíta a zapíše za sebou prvky „časového pola“ a oddelí ich čiarkou (,). Po načítaní a zapísaní všetkých prvkov „časového pola“ sa uzatvorí riadok znakom "`]`" a kurzor skočí na nový riadok. Tento riadok reprezentuje x-ovú os.

V ďalšom riadku sa zopakuje vyššie popísaný postup, s tým rozdielom, že na začiatok riadku sa zapíše "`y=[`" namiesto "`x=[`" a aplikácia nenačítava prvky z „časového pola“, ale z „pola prenesených bytov“ . Tento riadok definuje y-ovú os.

Posledný riadok, `plot(x,y)`, reprezentuje jednoduchý príkaz, ktorý slúži na vykreslenie grafu pre vyššie spomínané programy.

2.5 Návrh grafického užívateľského rozhrania

Aplikácia bola vyvíjaná v pracovnom prostredí EasyEclipse for Plugins and RCP Apps [8]. V základnej verzii EasyEclipse obsahuje iba integrované prostriedky pre vývoj štandardnej Javy ako prekladač, ladiaci program atď., ale neobsahuje napríklad nástroj pre vizuálny návrh grafických užívateľských rozhraní stolných aplikácií.

Pretože Eclipse RCP aplikácie využívajú nízkoúrovňovú grafickú knižnicu SWT pre realizovanie grafického rozhrania, prvou úlohou pri návrhu grafického užívateľského rozhrania bolo nájdenie vhodného programu alebo modulu, ktorý poskytuje možnosť navrhovať rozhranie pomocou SWT.

Po uvážení rôznych programov pre návrh užívateľského rozhrania bol nakoniec zvolený program WindowBuilder Pro [21]. Je to interaktívny program typu WYSIWYG (What You See Is What You Get), čiže ovládanie väčšinou prebieha pomocou myši a iba niektoré príkazy sa zapisujú ručne. Jednotlivé položky grafického rozhrania sa jednoducho chytia myšou a pretiahnu na požadované miesto. WindowBuilder Pro umožňuje jednoduchý návrh rôznych editorov okien, dialógov, atď. a Java kód pre tieto prvky sa generuje automaticky.

WindowBuilder Pro sa skladá z niekoľkých častí – SWT Designer, Swing Designer, GWT Designer a umožňuje automatickú inštaláciu do vývojového prostredia Eclipse.

Na stránke http://www.instantiations.com/windowbuilder/pro/download.html?id=1 je umiestnených niekoľko distribúcií programu. Užívateľ nájde na tejto adrese verziu pre operačné systémy Microsoft Windows a Linux. Po vybratí a stiahnutí vyhovujúcej distribúcie, je treba spustiť inštalačný súbor, samozrejme, súhlasiť s licenčnými podmienkami a potom

vybrať Eclipse produkt, ktorý chce užívateľ nainštalovať spolu s WindowBuilder Pro. Keď na hostiteľskom počítači už je nejaká Eclipse verzia nainštalovaná, tak sa táto verzia automaticky vyberie. Ak inštalačný súbor nenájde automaticky nainštalovaný Eclipse produkt, je ho možné aj manuálne vybrať pomocou tlačítka Vybrať Eclipse (*Choose Eclipse*). V prípade potreby je možné WindowBuilder Pro nainštalovať aj do viacerých Eclipse produktov. Po stlačení tlačítka Ďalej (*Next*) je v prípade behu nejakého Eclipse produktu užívateľ vyzvaný na jeho ukončenie. Potom sa objaví dialógové okno s názvom Konfiguračná možnosť vyčistenia vyrovnávacej pamäti Eclipse (*Clean Eclipse Cached Configuration Option*), kde by sa mala vybrať voľba *Yes* (Áno), pretože vyčistenie konfiguračných informácií je odporúčané z dôvodu prevencie konfiguračných konfliktov. Ďalšie kroky sú už bezproblémové a nakoniec sa vypíše úspešné dokončenie inštalácie.

Ďalším krokom po nainštalovaní programu je jeho registrácia. Dá sa tak učiniť cez menu Eclipse Okno (*Window*) / Výber (*Preferences*), vybrať Designer alebo WindowBuilder (*Designer or WindowBuilder*) podľa nainštalovaného typu programu / Licencia (*License*) a kliknúť na tlačítko Registrácia a aktivácia (*Registration and Activation*).

WindowBuilder Pro je k dispozícií zadarmo v skúšobnej verzii iba na 14 dní a potom ho treba zakúpiť. Existuje však aj bezplatná verzia, ktorá má obmedzené funkcie. Na návrh aplikácie s jednoduchším grafickým rozhraním však stopercentne vyhovuje aj táto verzia. Okrem vyplnenia dotazníka, ktorú verziu programu si praje užívateľ aktivovať, je potrebné cez stránku zaregistrovať produkt alebo zadať do ďalšieho okna osobné údaje a e-mailovú adresu, na ktorú sa potom pošle e-mail s aktivačným kódom. Pomocou tohto kódu treba program aktivovať.

Na stránke http://download.instantiations.com/D2WBDoc/continuous/latest/docs/html/quick_start.html sú návody k programu, odpovedané často kladené otázky, podrobné popisy funkcií programu a ďalšie užitočné informácie.

Po úspešnej aktivácii je možné začať pracovať s WindowBuilder Pro. Návrh grafického prostredia pomocou tohto programu je podobný návrhu vo vývojovom prostredí Netbeans [22].

Na návrh grafického užívateľského rozhrania štatistického nástroja bola použitá obmedzená verzia, ktorá je k dispozícii zdarma a zahŕňa nasledujúce prvky: SWT kompozície, SWT rozloženia, Obsluha SWT, Záložky JFace.

Všetky grafické prvky sú umiestnené v palete *Designer Palette*. Táto paleta je ďalej rozdelená do kategórií SWT kompozície (*SWT Composites)*, SWT rozloženia *(SWT layouts)*, Obsluha SWT *(SWT Controls)* a Záložky JFace *(JFace Viewers)*.

V kategórii SWT kompozície nájdeme základné združovacie prvky – Kompozičný rám (*Composite)*, Skupinový rám (*Group*), Zložka pre záložky (*TabFolder*) atď. Kategória SWT rozloženia obsahuje rôzne základné rozloženia – Absolútne rozloženie (*Absolute Layout*), Mriežkové rozloženie (*Grid Layout*), Vyplnené rozloženie (*Fill Layout*) a ďalšie. Najpoužívanejšie prvky ako napríklad Tlačítko (*Button*), Popis (*Label*), Textové pole (*Text*), Tabuľka (*Table*) a ďalšie sú umiestnené v Obsluhe SWT. V Záložkách JFace nájdeme napríklad Tlačítko pre dialógy (*Dialog Button*). Z týchto prvkov je poskladané celé grafické rozhranie.

Nové grafické rozhranie sa vytvára pomocou techniky „ťahaj a pusť“ (drag and drop). Najprv treba vytvoriť hlavnú konštrukciu aplikácie a potom na nej umiestniť prvky na požadované miesto. Každý prvok sa dá aj dodatočne upravovať a premiestniť. Po kliknutí ľavým tlačítkom na prvok, ktorý je už umiestnený na rozhraní sa zobrazí Editor vlastností (*Property Editor*). Pomocou tohto editoru môže užívateľ konfigurovať udalosti a nastaviť špecifické vlastnosti prvku. V hornej časti pracovnej položky Eclipse sú umiestnené najvyužívanejšie tlačítka ako Rýchly test (*Quickly test*), ktorý zobrazí návrh v novom okne, Obnoviť (*Refresh*), Vystrihnúť (*Cut*), Kopírovať (*Copy*), Vložiť (*Paste*), Vymazať (*Delete*) a ďalšie.

Základný panel, na ktorom sú umiestnené všetky prvky záložky, tvorí inštancia triedy *Composite.* Rozloženie je vytvorené pomocou triedy *FormLayout*. Pod menu a nástrojovou lištou, ktoré neboli vytvárané pomocou WindowBuilder Pro je umiestnená táto záložka. Na nej sú umiestnené prvky, ktoré tvoria hlavnú časť grafického užívateľského rozhrania. Záložka obsahuje zoznam IP adries, ktoré je definované pomocou zoznamu (*List*), okná na zadávanie IP adries (inštancie triedy Text), tlačítka (*Button*), popisy (*Label*) a plátna (Composite) na zobrazenie diagramu.

3 ZLOŽENIE ECLIPSE RCP APLIKÁCIE

3.1 Aplikácia

RCP SDK (Software Development Kit) obsahuje iba konštrukčné knižnice, ktoré sa nedajú spúšťať. Je to ako mať mnoho zdrojových kódov, ktoré ešte stále potrebujú, aby im niekto povedal čo majú robiť. V bežných Java systémoch je to dosiahnuté pomocou hlavnej triedy, ktorá obsahuje metódu `main()`. V Eclipse RCP je k tomu potrebné napísať aplikáciu (`Application`), ktorá slúži ako vstupný bod k programu alebo produktu. Aplikácia sa spustí, keď sa beh programu naštartuje a kontroluje Eclipse. Keď sa aplikácia ukončí, Eclipse sa uzavrie.

Aplikácie musia implementovať `IPlatformRunnable` a s ním metódu `run()`. Je to možné chápať ako metódu `main()`.

```
public class Application implements IPlatformRunnable {
  public Object run(Object args) throws Exception {
    Display display = PlatformUI.createDisplay();
    try {
      int returnCode = PlatformUI.createAndRunWorkbench(
          display, new ApplicationWorkbenchAdvisor());
      if (returnCode == PlatformUI.RETURN_RESTART) {
        return IPlatformRunnable.EXIT_RESTART;
      return IPlatformRunnable.EXIT_OK;
    } finally {
      display.dispose();
    }
  }
}
```

Aplikácia vytvorí `Display` a potom spustí `Eclipse Workbench` volaním `PlatformUI.createAndRunWorkbench(Display,WorkbenchWindowAdvisor)`. Ten otvorí okno a jednoducho ho cyklí. Týmto spôsobom obsluhuje užívateľom generované udalosti, ako napríklad kliknutie a pohyb myšou alebo stlačenie klávesnice. Cyklus udalosti sa ukončí, keď posledné okno sa uzatvorí alebo keď je mu to explicitne povedané. Vytvorený

`Display` musí byť odstránený pred ukončením aplikácie, aby uvoľnil alokované systémové zdroje.

3.2 Poradca pracovného panelu

Už ako aj názov naznačuje, poradca pracovného panelu (`WorkbenchAdvisor`) slúži na výpomoc pracovného panelu. Priradí mu hodnoty, určí ako sa má chovať, čo a ako má vykresliť, atď. Predovšetkým definuje dve hlavné veci:

- počiatočnú perspektívu,
- poradcu okna pracovného panelu (`WorkbenchWindowAdvisor`), ktorý má byť použitý.

3.3 Perspektíva

Počiatočná perspektíva (`Perspective`) je definovaná jej identifikátorom rozšírenia. Rozšírenie dáva perspektíve názov čitateľný aj človekovi a špecifikuje triedu, ktorá definuje rozloženie perspektívy. Daná trieda musí implementovať rozhranie `IPerspectiveFactory` a metódu `createInitial Layout(IPageLayout)`.

V systéme môže byť mnoho perspektív.

Každá RCP aplikácia musí definovať najmenej jednu perspektívu, inak by nebolo kde zobraziť záložky. Na perspektívu je možné sa pozerať ako na súpravu príkazov, ktoré slúžia na definovanie rozloženia okna. Každé pracovné okno (`IWorkbenchWindow`) má definovanú jednu stranu. Strana má svoj editor, inštancie záložiek a používa aktívnu perspektívu k rozhodnutiu jej rozloženia. Perspektíva určí, či má alebo nemá zobraziť isté prvky, ako napríklad záložky, editory, akcie a keď áno, kde ich má zobraziť.

3.4 Poradca okna pracovného panelu

Každé okno v aplikácii má svojho poradcu okna pracovného panelu (`WorkbenchWindowAdvisor`), ktorý riadi grafické rozhranie pri vykreslení okna.

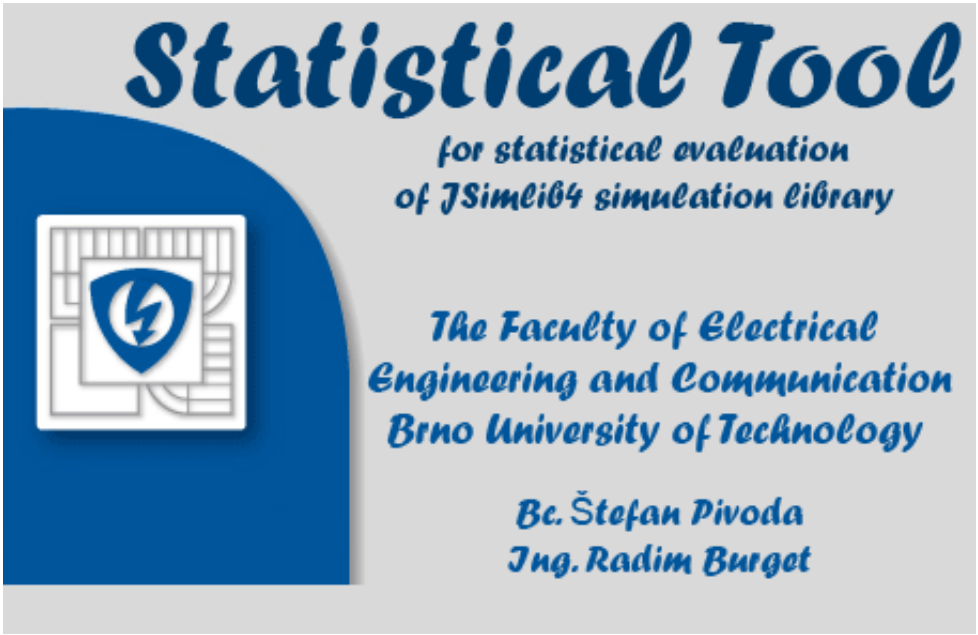
Poradca okna je informovaný počas cyklu okna vo viacerých bodoch, napríklad pred otvorením okna (`preWindowOpen()`) a po vytvorení okna (`postWindowCreate()`). Má možnosť tiež kontrolovať vytváranie obsahu okna.

3.5 Poradca lišty

Poradca lišty (`ActionBarAdvisors`) tvorí akcie pre okno a umiestňuje ich naň. Akcie sú inštancované využívaním metódy `createActionBarAdvisor()` v poradcovi okna pracovnej plochy.

4 POUŽITIE ŠTATISTICKÉHO NÁSTROJA

Po spustení aplikácie statical_tool.exe sa zobrazí úvodná obrazovka (Obr. 4.1), ktorá je viditeľná počas načítania základnej konfigurácie programu. Obsahuje názov programu, logo a názov fakulty a mená vývojárov. Týmto spôsobom je fakulta reprezentovaná užívateľovi pri každom spustení aplikácie.



Obr. 4.1 Úvodná obrazovka

Po načítaní zásuvných modulov sa zobrazí grafické rozhranie, ktoré má všetky prvky uvádzané v anglickom jazyku. Je to z toho dôvodu, aby program mohol využívať väčší počet ľudí, pretože práve angličtinu môžeme považovať v dnešnej dobe za základný jazyk informatiky. Ak by malo grafické rozhranie programu popisy v slovenskom alebo českom jazyku, zahraniční užívatelia by boli značne znevýhodnení a bolo by im znemožnené jednoducho sa orientovať v programe.

Jednotlivé prvky boli rozmiestnené takým spôsobom, aby boli čo najzrozumiteľnejšie každému užívateľovi už na prvý pohľad. Grafické rozhranie je možné rozdeliť na 3 hlavné časti:

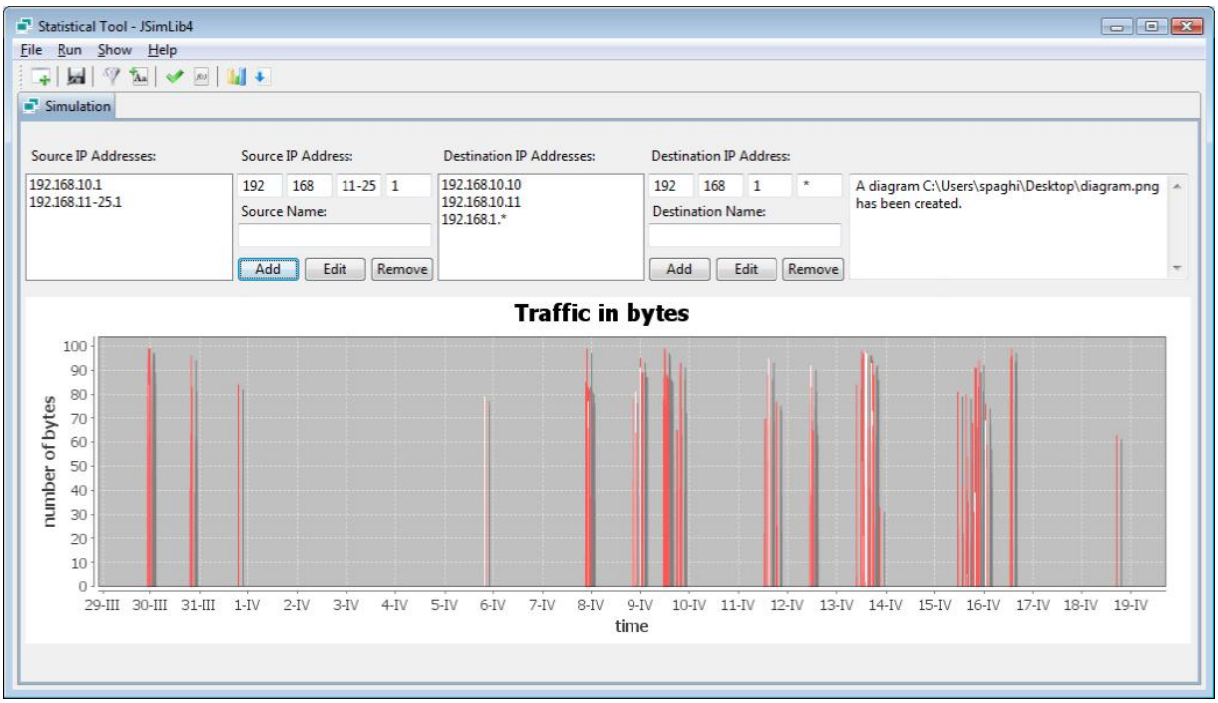
- menu,
- nástrojová lišta,
- záložky.

Menu obsahuje rôzne príkazy na prácu s aplikáciou. Obsahuje položky, ako napríklad vytvorenie novej simulačnej záložky, vytvorenie a zobrazenie diagramu alebo súboru vo formáte *.m alebo načítanie vstupných súborov.

Nástrojová lišta uľahčuje prácu s programom. Umožňuje užívateľovi rýchlejší prístup k jednotlivým príkazom.

Záložky sú jednotlivé analýzy, nezávislé od seba. Záložka tvorí hlavnú časť grafického užívateľského rozhrania.

Grafické rozhranie aplikácie znázorňuje obrázok 4.2 Grafické rozhranie štatistického nástroja.



Obr. 4.2 Grafické rozhranie štatistického nástroja

4.1 Menu

Menu obsahuje 4 hlavné položky:

- Súbor (*File*),
- Spusti (*Run*),
- Zobraz (*Show*),
- Pomocník (*Help*).

V hlavnej položke *Súbor* sú príkazy pre prácu so súbormi. Obsahuje nasledujúce podpoložky:

- Nová simulácia (*New simulation*),
- Načítaj logovací súbor (*Load a logfile*),
- Načítaj súbor s IP adresami (*Load an IP address file*),
- Ulož filter (*Save a filter*),
- Koniec (*Exit*).

Položka *Nová simulácia* (*New simulation*) slúži na vytvorenie novej simulačnej záložky. Po stlačení tejto položky sa zobrazí nová záložka, ktorá je popísaná v ďalšej časti tejto kapitoly.

V prípade, že simulácia bola spustená viackrát a užívateľ si jednotlivé logovacie súbory uložil vždy pod iným názvom, pomocou položky *Načítaj logovací súbor* (*Load a logfile*) si vie vybrať a načítať ten logovací súbor, ktorý má byť využitý v aktuálnej analýze. Po kliknutí na položku sa zobrazí vyberač súboru (File Chooser) a užívateľ má možnosť vybrať si jeden z logovacích súborov, ktoré si pred tým vytvoril. Logovací súbor má príponu *.log a vyberač súboru zobrazuje pri výbere iba súbory v tomto formáte. V prípade, že voľba načítania logovacieho súboru nie je využitá, analýza sa vykoná s implicitne nastaveným logovacím súborom. Tento logovací súbor je umiestnený v domovskom adresári aplikácie a má názov log.log.

Nasledujúca ukážka znázorňuje tvar logovacieho súboru:

```
#comment      Source_IP_address:Port:Destination_IP_address:
Port:Time:Bytes:Protocol:Direction
5.1.1.1:80:192.123.123.21:80:1239549928283:83:UDP:RECT
5.1.1.1:80:192.123.123.21:80:1239549995809:87:UDP:RECT
```

```
5.1.1.1:80:192.123.123.21:80:1239607965863:12:UDP:SENT
...
```

Prvým riadkom je komentár a popisuje význam zaznamenaných prvkov v ďalších riadkoch, ktoré sú umiestnené za sebou. Tieto prvky v poradí sú: zdrojová IP adresa a port, cieľová IP adresa a port, čas v UNIX-ovom formáte, počet prenesených bytov, použitý transportný protokol a smer prenosu.

V prípade, že užívateľ plánuje neskôr pracovať s IP adresami, zadanými v aktuálnej záložke, napríklad filtrovať s nimi iné logovacie súbory alebo ten istý logovací súbor, ale pri najbližšom spustení aplikácie, má možnosť si tieto IP adresy uložiť. Najjednoduchšie to vie vykonať pomocou položky v menu s názvom *Ulož filter (Save a filter)*, ktorá slúži na ukladanie IP adries z aktuálnej položky do externého súboru. Po zvolení tejto možnosti sa zobrazí vyberač súboru a užívateľ je vyzvaný k zadaniu názvu súboru. Je možné vybrať si aj iné umiestnenie ako implicitne navrhnuté. Jedinou podmienkou uloženia súboru je, že musí byť v textovom formáte s príponou *.txt. To je uskutočnené pomocou automatického nastavenia prípony a tak užívateľ vie zadať iba názov súboru bez definovania jeho formátu. Aplikácia uloží súbor s IP adresami v nasledujúcom tvare:

```
/* comment: 1st line: Source_IP_address, 2nd line:number_of_
source_IP_address ,3rd line: Destination_IP_address, 4th line:
number_of_destination_IP_address */
[192.168.0.10, 192.168.0.11]
2
[192.168.0.1]
1
```

Tento súbor sa potom dá načítať príkazom *Načítaj súbor s IP adresami (Load an IP address file)*.

Položka *Načítaj súbor s IP adresami (Load an IP address file)* načíta súbor, ktorý bol vytvorený niekedy počas predchádzajúcich prác pomocou tlačítka *Ulož filter (Save a filter)*. Jeho obsahom sú IP adresy. Táto položka je užitočná napríklad, keď užívateľ často potrebuje filtrovať logovací súbor podľa rovnakých IP adries. V tomto prípade nemusí pri každom spustení aplikácie znova zadať tie isté IP adresy, ale jednoducho ich vie načítať z takéhoto súboru, ktorý už niekedy pred tým vytvoril.

Načítaný súbor musí byť uložený v textovom formáte s príponou „sim“.

Ukážkou súboru s IP adresami môžu byť nasledujúce riadky:

```
/* comment: 1st line: Source_IP_address, 2nd line:number_of_
source_IP_address ,3rd line: Destination_IP_address, 4th line:
number_of_destination_IP_address */

[192.168.0.10, 192.168.0.11]

2

[192.168.0.1]

1
```

Ako vidíme, prvý riadok je označený ako komentár a jeho účelom je popísať užívateľovi výzor tohto súboru. Druhý a tretí riadok popisuje zdrojovú stanicu, kým štvrtý a piaty riadok popisuje cieľovú stanicu v sieti. Druhý riadok definuje IP adresy, podľa ktorých majú byť vyfiltrované zdrojové stanice a tretí riadok udáva počet týchto IP adries. Štvrtý a piaty riadok má podobný význam s tým rozdielom, že je definovaný pre cieľovú stanicu.

Ďalšou hlavnou položkou v menu je *Spusti (Run)*. Táto položka zahŕňa položky

- Vytvor diagram (*Create a diagram*),
- Vytvor *.m súbor (*Create an *.m file*),

ktoré slúžia na vytvorenie výstupných formátov aplikácie.

Príkaz *Vytvor diagram (Create a diagram)* vytvorí grafikon z výsledku analýzy. Ako to už bolo popísané v kapitole Realizácia štatistického nástroja, tento grafikon znázorňuje počet prenesených bytov medzi stanicami, ktoré boli vyfiltrované podľa zadaných IP adries z logovacieho súboru. Na x-ovú os sú vynesené časové údaje, kedy bol prenos uskutočnený a y-ová os prezentuje počet prenesených bytov. Grafikon sa uloží vo formáte PNG a názov obrázku určí užívateľ.

Podobne, príkaz *Vytvor *.m súbor (Create an *.m file)* vytvorí výstupný súbor, ale tentoraz vo formáte *.m. Príkladom vytvoreného súboru sú nasledujúce riadky:

```
x=[1239549928283,1239549995809,1239607965863,1239726899292,
1239728347003,1239729312097,1239731285464]

y=[83, 87, 12, 66, 24, 65, 75]

plot(x,y).
```

Prvý riadok popisuje hodnoty pre x-ovú os, ktorá udáva časové údaje v UNIX-ovom formáte. Druhý riadok, ktorý obsahuje hodnoty pre y-ovú os, zastupuje počet prenesených bytov medzi vyfiltrovanými stanicami.

Pred vytvorením týchto výstupných formátov je užívateľ vždy vyzvaný k zadaniu názvu výstupného formátu. O ukladanie výstupného súboru do vybraného formátu sa stará aplikácia, ktorá tieto formáty určí podľa zvolenej položky v menu.

Treťou hlavnou položkou v menu je *Zobraz (Show).* Zahŕňa položky, ktoré načítajú a zobrazia výstupné súbory, vytvorené niekedy počas predchádzajúcich analýz. Jej podpoložkami sú:

- Zobraz diagram (*Show a diagram*),
- Zobraz *.m súbor (*Show an *.m file*).

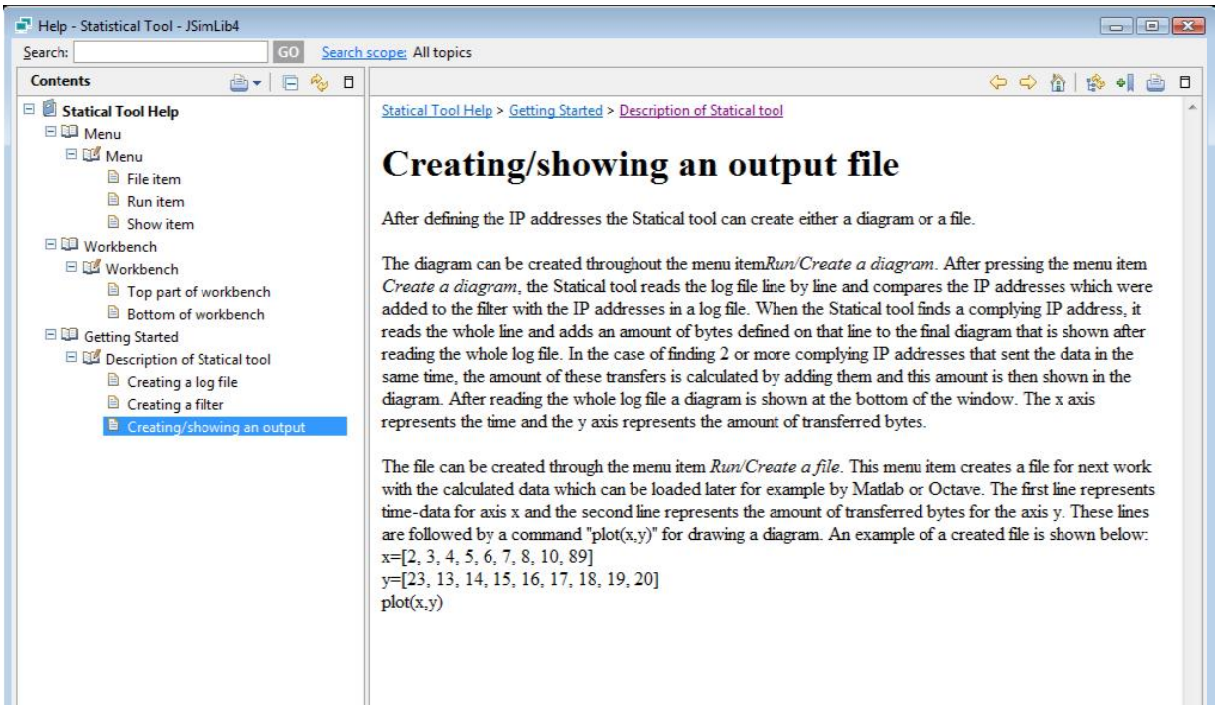
Ako už aj názov naznačuje *Zobraz diagram (Show a diagram)* zobrazí diagram, ktorý užívateľ zvolí pomocou vyberača súboru. Diagram sa potom zobrazí v dolnej časti položky.

Podobne, príkaz *Zobraz *.m súbor (Show an *.m file)* zobrazí v dolnej časti položky užívateľom zvolený súbor vo formáte *.m.

Poslednou položkou v menu je *Pomocník (Help)* a zastupuje podpoložky:

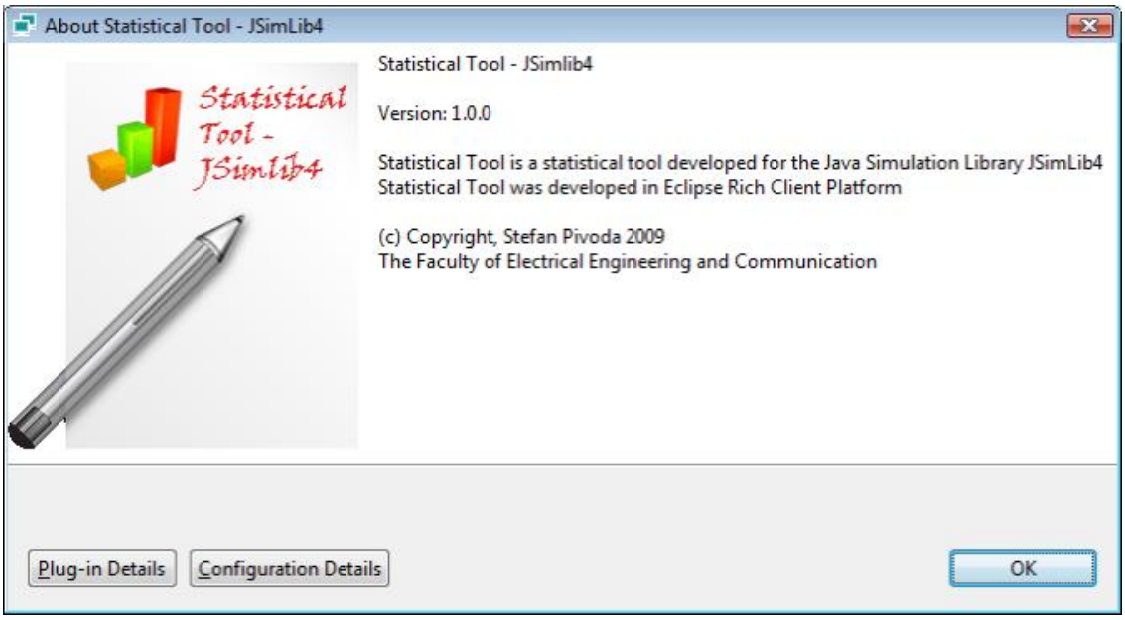
- Obsah pomocníka (*Help Contents*),
- O štatistickom nástroji (*About Statistical Tool*).

Obsah pomocníka (*Help Contents*) je rozdelený na niekoľko častí (Popis funkcií nástroja, Menu, Pracovná plocha). Na ľavej strane je menu Pomocníka, pomocou ktorého sa dá vybrať požadovaný popisný súbor. Po vybratí niektorého súboru sa na pravej strane zobrazí jeho stručný popis. Okno Pomocníka je znázornené na obrázku 4.3 Pomocník.



Obr. 4.3 Pomocník

Podpoložka *O štatistickom nástroji* (*About Statistical Tool*) vypíše základnú charakteristiku aplikácie (obr. 4.4). Obsahuje názov, verziu, účel a autora programu. Po stlačení tlačítka *Podrobnosti o zásuvných moduloch (Plug-in Details)* sa zobrazia názvy všetkých zásuvných modulov, ktoré program používa.



Obr. 4.4 Podrobnosti o konfigurácii

Tlačítko *Podrobnosti o konfigurácii (Configuration Details)* zobrazí konfiguračné detaily.

Položky v menu majú priradené klávesové skratky. Aj takýmto spôsobom je užívateľovi uľahčená práca, pretože na spustenie požadovaného príkazu počas práce sa užívateľ nemusí zložito preklikávať cez jednotlivé položky v menu, ale vie sa k nim dostať aj pomocou klávesových skratiek.

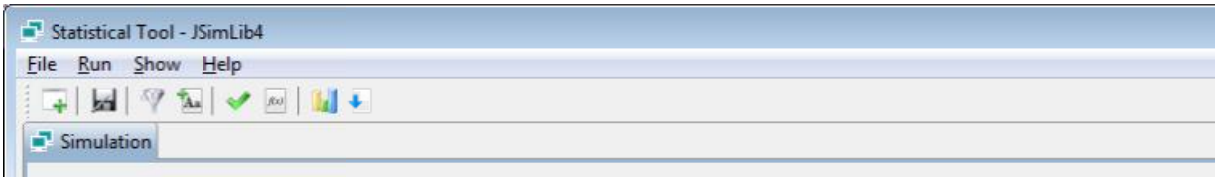
Takmer každý príkaz je sprístupnený pomocou kombinácie kláves Ctrl + jedného písmena na klávesnici. Napríklad príkaz *Vytvor *.m súbor (Create an *.m file)* má klávesovú skratku Ctrl + L.

Na položkách menu, naľavo od názvu sú umiestnené ikony. Tieto ikony sú zobrazené aj na tlačítkach v lište nástrojov.

4.2 Lišta nástrojov

Lišta nástrojov (obr. 4.5) má zjednodušiť prístup k príkazom, ktoré sú dostupné cez menu. Každé tlačítko obsahuje obrázok a po položení myši na tlačítko sa zobrazí stručný popis tohto tlačítka. Tento popis udáva funkciu tlačítka.

Obrázky použité pre tlačítka v lište nástrojov boli stiahnuté z adresy http://www.iconspedia.com/, kde sú dostupné obrázky a ikony v rôznych formátoch k stiahnutiu zdarma. Licenčné podmienky pre použitie týchto obrázkov a ikon sú definované zvlášť pre každú súpravu. Súprava (http://www.iconspedia.com/pack/crystal-clear-actions-1303/), ktorá bola použitá pre štatistický nástroj má licenčné podmienky definované ako GNU Lesser General Public License.

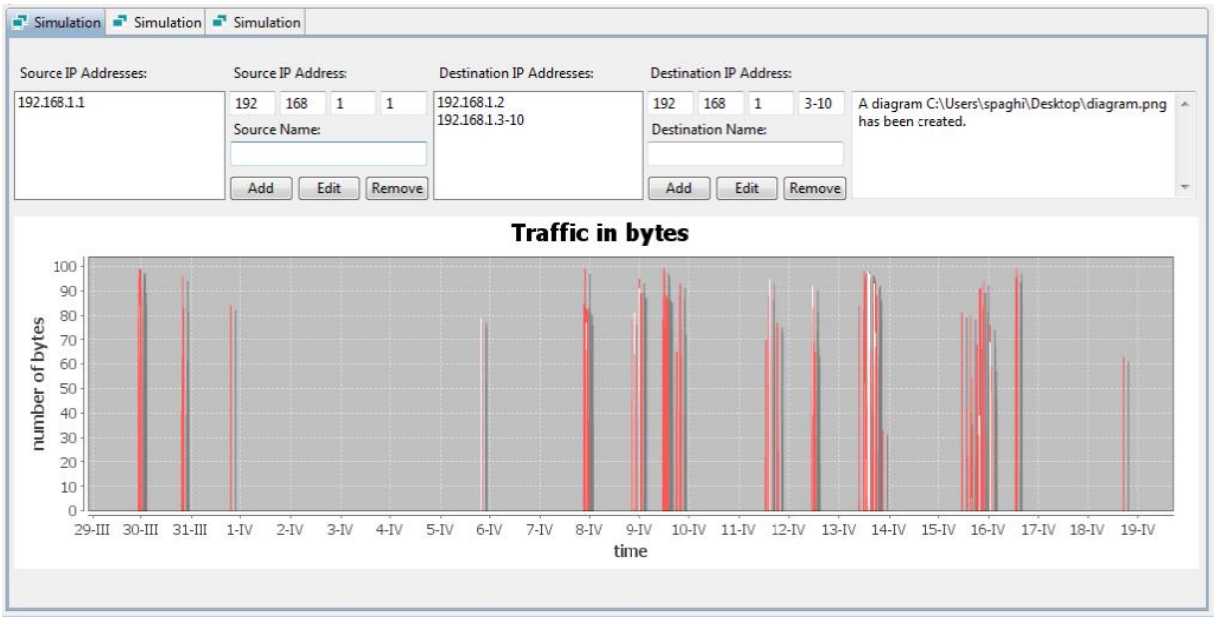


Obr. 4.5Lišta nástrojov

4.3 Záložky

Každá záložka (obr. 4.6) slúži na individuálnu analýzu pre zadané IP adresy v jeho oknách a nie je závislá od IP adries ostatných záložiek. Záložku môžeme chápať ako nezávislý program so špecifickými parametrami (IP adresy), ktorý sa spúšťa pod jedným grafickým rozhraním. Každá záložka je rozdelená na 2 časti:

- Horná
- Dolná.



Obr. 4.6 Záložka

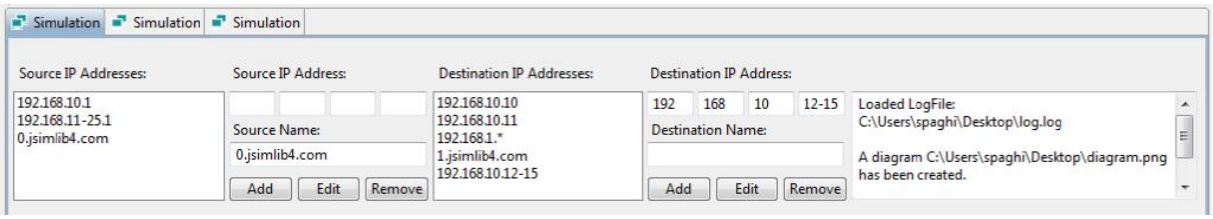
Horná časť záložky slúži na prácu s IP adresami a výpis informácií o vykonaných udalostiach. S IP adresami sa pracuje pomocou tlačítok, okien a zoznamov. V tejto časti sa zadávajú požadované IP adresy, podľa ktorých má aplikácia vyfiltrovať logovacie záznamy, ktoré sa potom zobrazia v dolnej časti záložky. Ďalej je možné podľa potreby upravovať a vymazávať IP adresy.

Po analýze sa v dolnej časti zobrazujú výstupné súbory v požadovanom formáte. Táto časť slúži aj na zobrazenie výstupných súborov z predchádzajúcich simulácií.

4.4 Horná časť záložky

Horná časť záložky (obr. 4.7) je rozdelená horizontálne na 3 časti.

Ľavá a stredná časť v hornej polovici záložky obsahujú rovnaké prvky s tým rozdielom, že slúžia pre iný typ stanice. Kým v ľavej časti má užívateľ možnosť pracovať s IP adresami pre zdrojové stanice, v strednej časti sú definované IP adresy udávajúce cieľové stanice. Na pravej časti sa vypisujú informácie o vykonaných úkonoch.



Obr. 4.7 Horná časť záložky

Každý odborník pracujúci so sieťami a IP adresami, ktorá je definovaná verziou 4 vie, že táto verzia IP adresy má 32 bitov. Na zjednodušenie práce s IP adresami sa používa zápis v dekadickom tvare. Umožňuje to jednoduchší spôsob zapamätania si týchto adries, pretože inak by bolo nutné zapamätať si 32 za sebou nasledujúcich binárnych čísiel, čiže 0 alebo 1. 8 bitov tvorí 1 byte, ktorý sa pri IP adrese volá oktet. Takýmto spôsobom získame z tridsiatich dvoch bitov štyri byty, čiže 4 oktety, ktoré spoločne tvoria jednu IP adresu. Jednotlivé oktety môžu mať hodnotu od 0 až 255 a oddeľujú sa bodkou (.). V štatistickom nástroji je pre každý oktet umiestnené samostatné pole a aplikácia sa automaticky stará o doplnenie bodiek medzi jednotlivými oktetmi. Do pola sa teda zadáva iba hodnota oktetu bez bodky. Polia sú okrem toho navrhnuté tak, aby umožnili zadávať iba povolené tvary IP adries. Užívateľ vie zadať korektný tvar troma spôsobmi:

- číslo medzi 0-255,
- dve dekadické čísla oddelené pomlčkou (-),
- hviezdičkou (*).

Prvá možnosť, čiže zadanie jedného čísla medzi 0 – 255 (čo je vlastne dekadický zápis binárneho tvaru čísiel 00000000 – 11111111) je najjednoduchším spôsobom definovania hodnoty oktetu. V tomto prípade oktet zastupuje toto číslo.

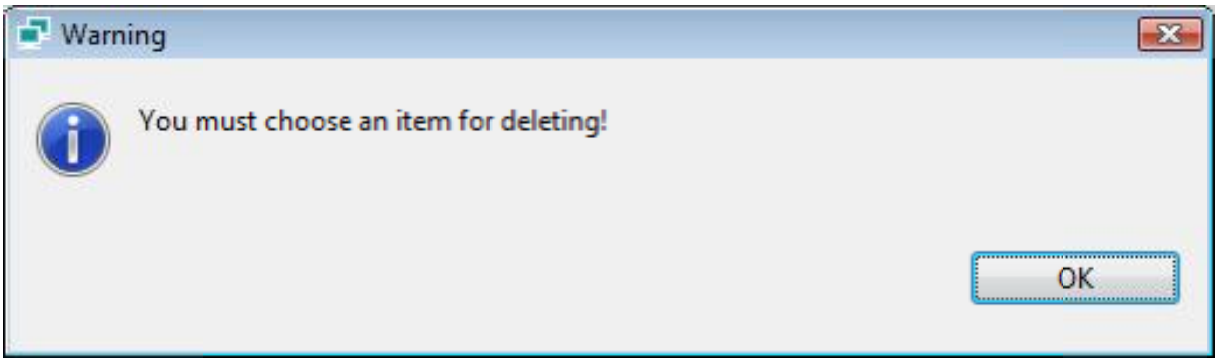
V prípade, že užívateľ potrebuje zadať IP adresy, ktoré sa odlišujú iba v jednom oktete a čísla v tomto oktete sa nachádzajú za sebou, najvýhodnejším typom zápisu je zadanie oktetu pomocou dvoch dekadických čísiel oddelenými pomlčkou (-). Tento zápis znamená, že do IP adresy na mieste tohto oktetu sa majú započítať všetky čísla medzi týmito číslami, vrátane čísla pred a po pomlčke. Napríklad pri pridaní IP adresy 192.168.0.1-10 do zoznamu, kde štvrtý oktet je zapísaný dvoma číslami oddelenými pomlčkou sa do filtrovaných adries započítajú všetky adresy medzi 192.168.0.1 až192.168.0.10, čiže 192.168.0.1, 192.168.0.2, 192.168.0.3, ..., 192.168.0.10. V tomto prípade sa do zoznamu zapíše iba zápis 192.168.0.1-10, ale aplikácia vie, že pri filtrovaní má počítať so všetkými vyššie uvedenými IP adresami.

Nakoniec, zápis oktetu pomocou hviezdičky (*), ktorá reprezentuje všetky čísla od 0 až po 255 má užívateľom uľahčiť zápis všetkých čísiel v oktete a tým umožniť zápis viacerých riadkov do jedného. Tento znak (*) sa veľakrát spomína aj ako zástupný znak (wildcard) alebo žolík, pretože môže zastupovať ľubovoľné číslo medzi 0 až 255. V prípade zápisu IP adresy 192.168.*.1 sa do filtrovaných adries započítajú adresy 192.168.0.1, 192.168.1.1, 192.168.2.1, ..., 192.168.255.1. Do zoznamu sa opäť zapíše iba zápis 192.168.*.1, tak ako to bolo aj v prípade zápisu prevedeného pomocou pomlčky (-). Filtrovaniu však vyhovie každá IP adresa, ktorá na oktete zadaného s hviezdičkou má ľubovoľné číslo medzi 0 až 255 a na ostatných oktetoch má rovnaké čísla ako zapísaná IP adresa.

V prípade, že sa užívateľ pokúsi zadať iné znaky, ako napríklad písma alebo charaktery, ktoré nevyhovujú korektnému zápisu, IP adresa sa nezapíše a zobrazí sa upozorňujúce hlásenie na túto udalosť.

Horná časť záložky sa skladá z niekoľkých grafických prvkov, ktoré slúžia na zobrazenie a prácu s IP adresami. Na ľavej strane je zoznam, ktorý ukazuje zadané IP adresy. Na pravej strane sú umiestnené tlačítka Pridaj (*Add*) , Uprav (*Edit*) a Odstráň (*Remove*), pomocou ktorých sa dajú pridávať, editovať a odstraňovať IP adresy. Polia nad tlačítkami slúžia na zadávanie jednotlivých oktetov IP adresy a názvu stanice.

Zoznam môže zobrazovať naraz aj viac IP adries. Zadané IP adresy sa oddeľujú riadkami a zobrazujú sa podľa časového poradia, v ktorom boli zadané. To znamená, že prvú zadanú adresu zobrazí v prvom riadku a naposledy zadanú v poslednom riadku. Samozrejme, pri pokuse zadania nesprávnej IP adresy alebo názvu stanice je na túto udalosť užívateľ upozornený. Zobrazí sa chybové hlásenie upozorňujúce užívateľa na nesprávnu akciu (obr. 4.8). Toto hlásenie sa zobrazí aj v prípade, že do polí IP adresy a názvu stanice je zapísaná súčasne nejaká hodnota.



Obr. 4.8 Chybové hlásenie

Pomocou tlačítok vieme pracovať s IP adresami. Umožňujú pridávanie, upravovanie a odstraňovanie už zadaných IP adries.

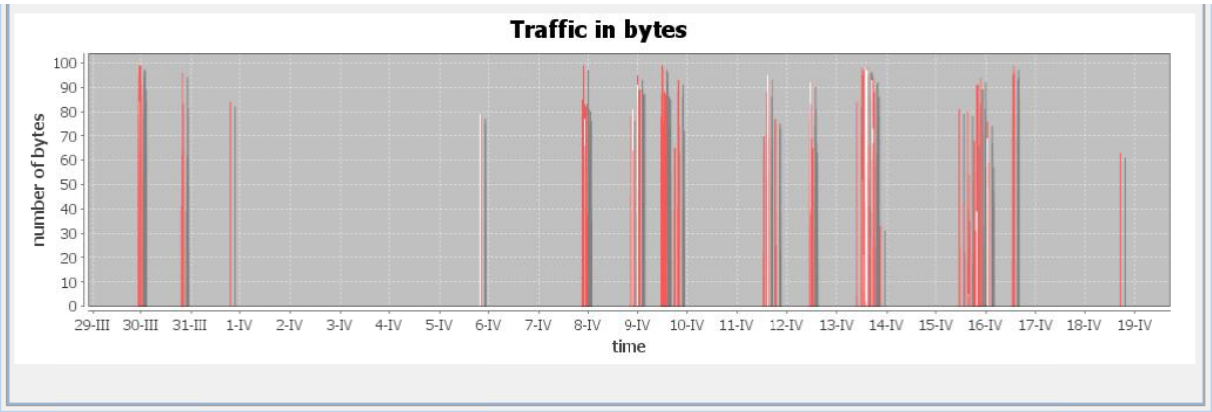
Po správnom vyplnení štyroch oktetov a stlačení tlačítka *Pridaj (Add)* sa z nich vytvorí IP adresa a následne sa pridá do zoznamu, podľa ktorého sa neskôr budú vstupné záznamy filtrovať. Názov stanice do tohto zoznamu sa pridáva tiež pomocou tohto tlačítka.

Na odstránenie IP adresy alebo názvu stanice zo zoznamu slúži tlačítko *Odstráň (Remove)*. Najprv treba vybrať kliknutím požadovanú IP adresu alebo názov, ktoré chce užívateľ vymazať a potom kliknúť na tlačítko *Odstráň*. IP adresa sa vymaže zo zoznamu a nasledujúce IP adresy sa posunú o riadok vyššie, aby nebol voľný riadok medzi predchádzajúcou a nasledujúcou adresou.

Často sa stáva, že užívateľ omylom zadá nesprávnu IP adresu alebo názov stanice a zistí to až po analýze alebo v lepšom prípade ešte pred ňou, napríklad hneď po zadaní. Pre takýto prípad slúži tlačítko *Uprav (Edit)*, ktoré umožňuje upravovať už zadané IP adresy. Podobne ako pri vymazávaní treba kliknúť na požadovanú IP adresu alebo názov stanice a potom stlačiť tlačítko *Uprav*. IP adresa sa zo zoznamu vymaže podobným spôsobom ako pri vymazávaní a načíta sa do polí určených pre oktetov. V prípade upravovania názvu sa vymaže zo zoznamu a načíta sa do okna určeného preň. Ak užívateľ neklikne na žiadnu adresu a stlačí tlačítko *Uprav* alebo *Odstráň* vypíše sa chybové hlásenie upozorňujúce užívateľa na túto skutočnosť.

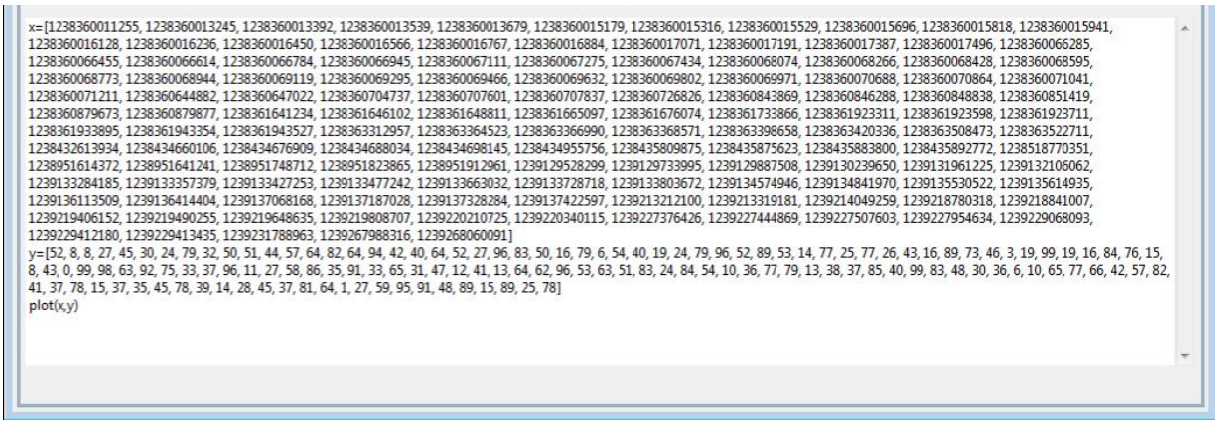
4.5 Dolná časť záložky

Dolná časť záložky slúži na zobrazenie výstupných súborov. Aplikácia podporuje 2 výstupné formáty, *.png a *.m. Na zobrazenie týchto výstupných súborov je v tejto časti záložky umiestnené plátno, ktoré sa implicitne nezobrazuje. Zobrazí sa až pri zjavení sa jedného z výstupných formátov, ale aj vtedy je hneď zakryté s týmto výstupom, čiže je v podstate celý čas neviditeľné.



Obr. 4.9 Diagram zobrazený v dolnej časti záložky

Ako to bolo popísané už v predchádzajúcej časti, formát PNG slúži na vizuálne zobrazenie výstupu aplikácie. Týmto vizuálnym výstupom je graf, ktorý znázorňuje prenos medzi zvolenými stanicami. Po úspešnom zadaní IP adries, podľa ktorých má program vyfiltrovať vyhovujúce stanice z logovacieho súboru treba zvoliť z menu položku *Spusti (Run)/ Vytvor diagram (Create a diagram)*. Druhým spôsobom zobrazenia grafu je použitie lišty nástrojov, ktorá obsahuje tlačítko so zelenou fajkou a má rovnakú funkciu ako položka v menu *Spusti (Run)/ Vytvor diagram (Create a diagram)*. Užívateľovi sa potom zobrazí vyberač súboru a je vyzvaný k zadaniu názvu a umiestnenia obrázku, ktorý má v každom prípade príponu „png“. Po tomto procese štatistický nástroj začne prechádzať logovací súbor a porovnávať zadané IP adresy s IP adresami v logovacom súbore. Pri nájdení vyhovujúceho záznamu, tento záznam započíta do výstupných hodnôt a pokračuje ďalej, kým neporovná každú zadanú IP adresu s každým záznamom v logovacom súbore. Po dokončení tejto analýzy program vytvorí obrázok a potom ho zobrazí v dolnej časti záložky (obr. 4.9).



*Obr. 4.10 Súbor *.m zobrazený v dolnej časti záložky*

Podobný postup je pri vytvorení a zobrazení výstupného formátu *.m. Prvým krokom je zadanie IP adries, podľa ktorých má program vyfiltrovať vhodné stanice. Po dokončení tohto kroku treba zvoliť položku v menu *Spusti (Run)/ Vytvor *.m súbor (Create an *.m file)* alebo tlačítko z lišty nástrojov, na ktorom je nakreslený papier s nápisom f(x). Potom sa zobrazí vyberač súboru, do ktorého musí užívateľ zadať umiestnenie a názov súboru. Po splnení týchto požiadaviek a kliknutí na tlačítko *Ulož filter (Save a filter)* sa spustí proces analýzy, ktorý má rovnaký postup ako v prípade uloženia výstupného súboru v grafickom formáte. Znamená to, že program postupne porovnáva IP adresy s IP adresami v logovacom súbore a pri zhode sa zapíše záznam do výstupných hodnôt. Po dokončení analýzy sa uloží výstupný súbor a potom sa zobrazí v dolnej časti záložky (obr. 4.10).

Obr. 4.11 Ikona oznamovacej oblasti

Po spustení aplikácie sa na hlavnom paneli operačného systému zobrazí tzv. ikona oznamovacej oblasti vedľa ikon iných, často využívaných aplikáciách ako napríklad čas, zvuk, atď. Pomocou tejto ikony sú dostupné tiež niektoré príkazy programu. Túto ikonu znázorňuje obrázok 4.11 Ikona oznamovacej oblasti. Pri minimalizácii programu, sa okno programu minimalizuje do tejto ikony a zmizne aj z hlavnej lišty. Po dvojitom kliknutí na túto ikonu sa okno programu opäť obnoví do pôvodnej veľkosti.

ZÁVER

V tomto popise diplomovej práce sa čitateľ zoznámil s návrhom a riešením Eclipse Rich Client Platform aplikácie, ktorá má slúžiť ako rozširujúci modul pre javovskú knižnicu JSimlib4. Táto knižnica je navrhnutá na simulovanie diskrétnych udalostí a umožňuje aj simuláciu vlastností reálnej siete. Rozširujúci modul slúži na štatistické vyhodnotenie priebehu simulácií, ktoré sú vykonané pomocou tejto knižnice.

Vstupom do modulu je logovací súbor, ktorý obsahuje záznamy o prenesených bytoch medzi jednotlivými stanicami v sieti. Do modulu bol nasadený filter, ktorého úlohou je vyfiltrovať záznamy z logovacieho súboru a poskytnúť výstupu iba vyhovujúce záznamy. Výstupom aplikácie je graf generovaný pomocou javovskej knižnice JFreechart alebo textový súbor obsahujúci maticu prenesených bytov a príkaz vykreslenia tejto matici pre programy Matlab a Octave.

Aplikácia spĺňa zadané požiadavky diplomovej práce. Medzi tieto požiadavky patrí napríklad voľba vhodného filtračného postupu, návrh a realizácia grafického užívateľského rozhrania a zobrazenie výstupných súborov.

Počas riešenia diplomovej práce som sa dozvedel mnoho nových vecí. Zoznámil som sa s platformou Eclipse RCP a s jej konštrukciou. Jedná sa o relatívne novú technológiu, ktorej znalosť môže byť prínosom pre každého programátora. Myslím si, že vedomosť tejto platformy môže byť pre mňa veľkou výhodou pri uplatnení sa v zamestnaní po ukončení štúdia.

Ďalej som sa zdokonalil v riešení problémov, ktoré sa vyskytovali počas vývoja aplikácie. Spoznal som viacero typov riešenia jednej problematiky, z ktorých som sa snažil vždy vybrať ten najvýhodnejší.

Myslím si, že táto práca bola pre mňa veľmi užitočná aj preto, že som si pomocou nej rozšíril vedomosti nielen v programovaní v jazyku Java, ale zoznámil som sa aj so spôsobom tímovej práce.

Pri vyhľadávaní popisov na danú problematiku, najmä pri hľadaní informácií o Eclipse RCP aplikáciách, ktoré sa vyskytovali hlavne v angličtine, som bol prinútený využívať zdroje v anglickom jazyku, čo ma viedlo k zdokonaleniu sa v porozumení odborných textov v tomto jazyku. Je to základnou požiadavkou na každého programátora a odborníka informačnej technológie, a preto si myslím, že aj týmto spôsobom ma táto práca obohatila.

POUŽITÁ LITERATÚRA

[1] Introduction to Algorithms, Second Edition. MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7 . Chapter 13: Red-Black Trees, pp.273–301.

[2] NĚMEC, Jan. *Linuxsoft*. [online]. 9.1.2006, [cit. 2009-05-14]. URL: <http://www.linuxsoft.cz/article.php?id_article=1058>.

[3] The C++ Resources Network. *cplusplus.com*. [online]. © 2000-2008 [cit. 2009-05-14]. URL: <http://www.cplusplus.com/>.

[4] Sun Microsystems, Amaio Technologies, a.s., MathAn Praha, s.r.o. . *Java portál*. [online]. [cit. 2009-05-14] . URL: <http://java.cz/>.

[5] ETKI. *The Java Tutorials*. [online] . poslední revize 1.8.2007 [cit. 2009-05-14] . URL: <http://www.gymspmkr.edu.sk/informatika/JavaTutorial/>.

[6] Sun Microsystems, Inc. *Java*. [online]. Copyright 1994-2008 [cit. 2009-05-14]. URL: <http://www.java.com/en/about/>.

[7] nexB Inc. Easy*Eclipse*. [online]. Copyright 2005-2007 [cit. 2009-05-14]. URL: <http://www.easyeclipse.org/>.

[8] Inc. Easy*Eclipse*. [online]. Copyright 2005-2007 [cit. 2009-05-14]. URL: http://www.easyeclipse.org/site/distributions/plugin-warrior.html/

[9] CollabNet, Inc. *Tigris.org Open Source Software Engineering Tools*. [online]. © 2001 - 2008, [cit. 2009-05-14]. URL: <http://subclipse.tigris.org/>.

[10] The Eclipse Foundation. *Eclipse*. [online]. © 2008, [cit. 2009-05-14]. URL: <http://www.eclipse.org/>.

[11] Vysoká škola ekonomická v Praze. *Java.VŠE*. [online]. poslední revize 22.01.2008 [cit. 2009-05-14]. URL: <http://java.vse.cz/Main/EclipseInstalace>.

[12] The Eclipse Foundation. *Eclipse Public License - v 1.0*. [online]. © 2008, [cit. 2009-05-14]. URL: <http://www.eclipse.org/legal/epl-v10.html>.

[13] The Eclipse Foundation. *Subversive – SVN Team Provider*. [online]. © 2008, [cit. 2009-05-14]. URL: <http://www.eclipse.org/subversive/>.

[14] The MathWorks, Inc. *MATLAB - The Language of Technical Computing*. [online]. © 1994-2008, [cit. 2009-05-14]. URL: <http://www.mathworks.com/products/matlab/>.

[15] W. EATON, John. *Octave*. [online]. © 1998-2006, [cit. 2009-05-14]. URL: <http://www.gnu.org/software/octave/>.

[16] JUST, Michal. *Octave*. [online]. © 2006, poslední revize 6.2006 [cit. 2009-05-14]. URL: <http://www.octave.cz/>.

[17] Object Refinery Limited. *JFreeChart*. [online]. © 2007 [cit. 2009-05-14]. URL: <http://www.object-refinery.com/jfreechart/index.html>.

[18] Object Refinery Limited. *JFreeChart Developer Guide*. [online]. © 2006 [cit. 2009-05-14]. URL: <http://www.object-refinery.com/jfreechart/guide.html>.

[19] Elysium Ltd. *The JPEG committee home page*. [online]. © 2007 [cit. 2009-05-14]. URL: <http://www.jpeg.org/ >.

[20] ROELOFS, Greg. *PNG (Portable Network Graphics)*. [online]. poslední revize 19.9.2008 [cit. 2009-05-14]. URL: < http://www.libpng.org/pub/png/ >.

[21] Instantiations, Inc. *WindowBuilder* [online]. © 2009 [cit. 2009-05-14]. URL: http://www.instantiations.com/windowbuilder/index.html >.

[22] Sun Microsystems, Inc. and CollabNet, Inc. *Netbeans*. [online]. [cit. 2009-05-14]. URL: <http://www.netbeans.org/>.

[23] CollabNet, Inc. *Tigris.org Open Source Software Engineering Tools*. [online]. © 2001 - 2008, [cit. 2009-05-14]. URL: <http://subversion.tigris.org/>.

[24] CollabNet, Inc. *Collabnet*. [online]. ©2008. [cit. 2009-05-14]. URL: <http://www.collab.net/>.

[25] OSGi™ Alliance. *OSGi™ The Dynamic Module System for Java™*. [online]. ©2009. [cit. 2009-05-14]. URL: <http://www.osgi.org/Main/HomePage>.

[26] The Eclipse Foundation. *SWT: The Standard Widget Toolkit*. [online]. ©2009. [cit. 2009-05-14]. URL: <http://www.eclipse.org/swt/>.

[27] The Eclipse Foundation. *JFace*. [online]. ©2009. [cit. 2009-05-14]. URL: <http://wiki.eclipse.org/index.php/JFace>.

ZOZNAM POUŽITÝCH SKRATIEK

EPL – Eclipse Public License

GIF – Graphics Interchange Format

IBM – International Business Machines Corporation

IDE – Integrated Development Environment

IP – Internet Protocol

JAR – Java™ Archive

Java SE – Java Platform, Standard Edition

Java EE – Java Platform, Enterprise Edition

Java ME – Java Platform, Micro Edition

JPEG – Joint Photographic Experts Group

JVM – Java Virtual Machine

LAMP – Linux, Apache, MySQL and Perl/PHP/Python

NASA – National Aeronautics and Space Administration

OSGi – Open Services Gateway initiative

PHP – Hypertext Preprocessor

PNG – Portable Network Graphics

RCP – Rich Client Platform

SDK – Software Development Kit

SVN – Subversion

SWT – The Standard Widget Toolkit

TCP – Transmission Control Protocol

UDP – User Datagram Protocol

URL – Uniform Resource Locator

WYSIWYG – What You See Is What You Get

ZOZNAM PRÍLOH

CD s elektronickou verziou práce, zdrojovým kódom aplikácie a spustiteľným programom